

O'REILLY®

2-е издание

Практическая статистика для специалистов Data Science

50+ важнейших понятий с использованием R и Python



Питер Брюс
Эндрю Брюс
Питер Гедек

SECOND EDITION

Practical Statistics for Data Scientists

50+ Essential Concepts Using R and Python

Peter Bruce, Andrew Bruce, and Peter Gedeck

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Питер Брюс, Эндрю Брюс, Питер Гедек

Практическая статистика для специалистов Data Science

50+ важнейших понятий с использованием R и Python

2-е издание

Санкт-Петербург

«БХВ-Петербург»

2021

УДК 004.6+519.2
ББК 32.81+22.172
Б89

Брюс, П.

Б89 Практическая статистика для специалистов Data Science: Пер. с англ. / П. Брюс, Э. Брюс, П. Гедек. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2021. — 352 с.: ил.

ISBN 978-5-9775-6705-3

Книга рассчитана на специалистов в области Data Science, обладающих некоторым опытом работы с языком программирования R и имеющих предварительное понятие о математической статистике. В ней в удобной и легкодоступной форме представлены ключевые понятия из статистики, которые относятся к науке о данных, а также объяснено, какие понятия важны и полезны с точки зрения науки о данных, какие менее важны и почему. Подробно раскрыты темы: разведочный анализ данных, распределения данных и выборок, статистические эксперименты и проверка значимости, регрессия и предсказание, классификация, статистическое машинное обучение и обучение без учителя. Во второе издание включены примеры на языке Python, что расширяет практическое применение книги.

Для аналитиков данных

УДК 004.6+519.2
ББК 32.81+22.172

Группа подготовки издания:

Руководитель проекта	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Людмила Гауль</i>
Перевод с английского	<i>Андрея Логунова</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Оформление обложки	<i>Карины Соловьевой</i>

© 2021 BHV

Authorized Russian translation of the English edition of *Practical Statistics for Data Scientists 2nd edition*

ISBN 9781492072942 © 2020 Peter Bruce, Andrew Bruce, and Peter Gedeck.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

Авторизованный перевод с английского языка на русский издания *Practical Statistics for Data Scientists 2nd edition*

ISBN 9781492072942 © 2020 Peter Bruce, Andrew Bruce, Peter Gedeck.

Перевод опубликован и продается с разрешения компании-правообладателя O'Reilly Media, Inc.

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-1-492-07294-2 (англ.)
ISBN 978-5-9775-6705-3 (рус.)

© Peter Bruce, Andrew Bruce, Peter Gedeck, 2020
© Перевод на русский язык, оформление. ООО "БХВ-Петербург",
ООО "БХВ", 2021

Оглавление

Об авторах.....	13
Предисловие	15
Условные обозначения, принятые в книге	15
Использование примеров кода	16
Благодарности.....	16
Комментарии переводчика	17
Глава 1. Разведывательный анализ данных.....	19
Элементы структурированных данных	20
Дополнительные материалы для чтения	22
Прямоугольные данные	23
Кадры данных и индексы.....	24
Непрямоугольные структуры данных.....	25
Дополнительные материалы для чтения.....	26
Оценки центрального положения	26
Среднее	27
Медиана и робастные оценки.....	29
Выбросы	29
Пример: средние оценки численности населения и уровня убийств	30
Дополнительные материалы для чтения.....	32
Оценки вариабельности	32
Стандартное отклонение и связанные с ним оценки.....	33
Оценки на основе процентилей	35
Пример: оценки вариабельности населения штатов.....	36
Дополнительные материалы для чтения.....	37
Разведывание распределения данных.....	38
Процентили и коробчатые диаграммы	38
Частотные таблицы и гистограммы	40
Графики и оценки плотности.....	42
Дополнительные материалы для чтения.....	44
Разведывание двоичных и категориальных данных.....	44
Мода.....	46
Ожидаемое значение	47
Вероятность.....	47
Дополнительные материалы для чтения.....	48
Корреляция.....	48
Диаграммы рассеяния	52
Дополнительные материалы для чтения.....	53

Разведывание двух или более переменных	53
Сетка из шестиугольных корзин и контуры (сопоставление числовых данных с числовыми данными на графике)	54
Две категориальные переменные	57
Категориальные и числовые данные	58
Визуализация многочисленных переменных	60
Дополнительные материалы для чтения	62
Резюме	63
Глава 2. Распределение данных и распределение выборок	65
Случайный отбор и смещенная выборка	66
Смещение	68
Случайный отбор	69
Размер против качества: когда размер имеет значение?	70
Выборочное среднее против популяционного среднего	71
Дополнительные материалы для чтения	71
Систематическая ошибка отбора	72
Регрессия к среднему	73
Дополнительные материалы для чтения	75
Выборочное распределение статистической величины	75
Центральная предельная теорема	78
Стандартная ошибка	79
Дополнительные материалы для чтения	80
Бутстрап	80
Повторный отбор против бутстрапирования	84
Дополнительные материалы для чтения	84
Доверительные интервалы	84
Дополнительные материалы для чтения	87
Нормальное распределение	87
Стандартное нормальное распределение и квантиль-квантильные графики	89
Длиннохвостые распределения	91
Дополнительные материалы для чтения	93
<i>t</i> -Распределение Стьюдента	93
Дополнительные материалы для чтения	95
Биномиальное распределение	95
Дополнительные материалы для чтения	98
Распределение хи-квадрат	98
Дополнительные материалы для чтения	99
<i>F</i> -распределение	99
Дополнительные материалы для чтения	100
Распределение Пуассона и другие связанные с ним распределения	100
Пуассоновские распределения	101
Экспоненциальное распределение	101
Оценивание интенсивности отказов	102
Распределение Вейбулла	102
Дополнительные материалы для чтения	103
Резюме	104
Глава 3. Статистические эксперименты и проверка значимости	105
<i>A/B</i> -тестирование	105
Зачем нужна контрольная группа?	108

Почему только A/B ? Почему не $C, D...$?	109
Дополнительные материалы для чтения	110
Проверки гипотез	110
Нулевая гипотеза	112
Альтернативная гипотеза	112
Односторонняя проверка гипотезы против двухсторонней	113
Дополнительные материалы для чтения	114
Повторный отбор	114
Перестановочный тест	115
Пример: прилипчивость веб-страниц	115
Исчерпывающий и бутстраповский перестановочные тесты	119
Перестановочные тесты: сухой остаток для науки о данных	119
Дополнительные материалы для чтения	120
Статистическая значимость и p -значения	120
p -Значение	123
Альфа	124
Разногласия по поводу p -значения	124
Практическая значимость	125
Ошибки 1-го и 2-го рода	125
Наука о данных и p -значения	126
Дополнительные материалы для чтения	126
Проверки на основе t -статистики	127
Дополнительные материалы для чтения	129
Множественное тестирование	129
Дополнительные материалы для чтения	132
Степени свободы	133
Дополнительные материалы для чтения	134
Дисперсионный анализ	134
F -статистика	138
Двухсторонний дисперсионный анализ	139
Дополнительные материалы для чтения	140
Проверка на основе статистики хи-квадрат	140
Проверка хи-квадрат: подход на основе повторного отбора	141
Проверка хи-квадрат: статистическая теория	143
Точный тест Фишера	144
Релевантность для науки о данных	146
Дополнительные материалы для чтения	147
Алгоритм многорукого бандита	147
Дополнительные материалы для чтения	150
Мощность и размер выборки	151
Размер выборки	152
Дополнительные материалы для чтения	155
Резюме	155
Глава 4. Регрессия и предсказание	157
Простая линейная регрессия	157
Уравнение регрессии	158
Подогнанные значения и остатки	161
Наименьшие квадраты	162
Предсказание против объяснения (профилирование)	163
Дополнительные материалы для чтения	164

Множественная линейная регрессия	164
Пример: данные жилого фонда округа Кинг.....	165
Оценивание результативности модели	167
Перекрестный контроль	169
Отбор модели и пошаговая регрессия	170
Взвешенная регрессия	173
Дополнительные материалы для чтения.....	175
Предсказание с использованием регрессии	175
Опасности экстраполяции.....	175
Доверительный и предсказательный интервалы	176
Факторные переменные в регрессии.....	178
Представление фиктивных переменных.....	178
Факторные переменные с многочисленными уровнями.....	181
Упорядоченные факторные переменные	183
Интерпретирование уравнения регрессии.....	184
Коррелированные предсказатели	185
Мультиколлинеарность.....	186
Искажающие переменные	187
Взаимодействия и главные эффекты	188
Диагностика регрессии	190
Выбросы	191
Влиятельные значения	193
Гетероскедастичность, ненормальность и коррелированные ошибки	196
Графики частных остатков и нелинейность	199
Многочленная и сплайновая регрессия	201
Многочлены	202
Сплайны.....	203
Обобщенные аддитивные модели	206
Дополнительные материалы для чтения	207
Резюме	208
Глава 5. Классификация	209
Наивный Байес.....	210
Почему точная байесова классификация непрактична?.....	211
Наивное решение	211
Числовые предсказательные переменные	214
Дополнительные материалы для чтения.....	215
Дискриминантный анализ.....	215
Матрица ковариаций	216
Линейный дискриминант Фишера	217
Простой пример	217
Дополнительные материалы для чтения.....	221
Логистическая регрессия	221
Функция логистического отклика и логит	222
Логистическая регрессия и ОЛМ	223
Обобщенные линейные модели.....	225
Предсказанные значения из логистической регрессии	225
Интерпретирование коэффициентов и отношений перевесов.....	226
Линейная и логистическая регрессия: сходства и различия	228
Подгонка модели	228

Оценивание результативности модели	229
Анализ остатков	231
Дополнительные материалы для чтения	232
Оценивание классификационных моделей	233
Матрица путаницы	234
Проблема редкого класса	236
Прецизионность, полнота и специфичность	236
ROC-кривая	238
Площадь под ROC-кривой	240
Лифт	241
Дополнительные материалы для чтения	243
Стратегии для несбалансированных данных	243
Понижающий отбор	244
Повышающий отбор и повышающая/понижающая перевесовка	245
Генерация данных	246
Стоимостная классификация	247
Разведывание предсказаний	247
Дополнительные материалы для чтения	249
Резюме	249
Глава 6. Статистическое машинное обучение	251
k ближайших соседей	252
Небольшой пример: предсказание невыплаты ссуды	253
Метрики расстояния	255
Кодировщик с одним активным состоянием	256
Стандартизация (нормализация, z -оценки)	257
Выбор числа k	260
k ближайших соседей как механизм порождения признаков	261
Древесные модели	263
Простой пример	264
Алгоритм рекурсивного подразделения	267
Измерение однородности или загрязненности	268
Остановка выращивания дерева	270
Контроль за сложностью дерева в R	270
Контроль за сложностью дерева в Python	271
Предсказывание непрерывного значения	271
Каким образом используются деревья	272
Дополнительные материалы для чтения	273
Бэггинг и случайный лес	273
Бэггинг	274
Случайный лес	275
Важность переменных	279
Гиперпараметры	282
Бустинг	283
Алгоритм бустирования	285
XGBoost	286
Регуляризация: предотвращение перепогонки	288
Гиперпараметры и перекрестный контроль	292
Резюме	296

Глава 7. Неконтролируемое самообучение.....	297
Анализ главных компонент	298
Простой пример	299
Вычисление главных компонент	301
Интерпретирование главных компонент	302
Анализ соответствия	305
Дополнительные материалы для чтения.....	307
Кластеризация на основе K средних	307
Простой пример	308
Алгоритм K средних	310
Интерпретирование кластеров	311
Выбор числа кластеров	313
Иерархическая кластеризация	315
Простой пример	316
Дендограмма	317
Агломеративный алгоритм	318
Меры несхожести	319
Модельно-ориентированная кластеризация.....	321
Многомерное нормальное распределение	321
Смеси нормальных распределений	322
Выбор числа кластеров	325
Дополнительные материалы для чтения.....	327
Шкалирование и категориальные переменные	328
Шкалирование переменных	328
Доминантные переменные.....	330
Категориальные данные и расстояние Говера	332
Проблемы кластеризации смешанных данных	334
Резюме	336
Библиография	337
Предметный указатель.....	339

*Мы хотим посвятить эту книгу памяти наших родителей,
Виктора Г. Брюса и Нэнси С. Брюс, которые воспитали в нас
страсть к математике и точным наукам,
а также нашим первым учителям, Джону У. Тьюки и Джулиану Саймону,
и нашему верному другу, Джеффу Уотсону, который вдохновил нас на то,
чтобы мы посвятили свою жизнь статистике.*

*Питер Гедек хотел бы посвятить эту книгу
Тиму Кларку и Кристиану Крамеру с глубокой благодарностью
за их научное сотрудничество и дружбу.*

Об авторах

Питер Брюс (Peter Bruce) основал и расширил Институт статистического образования Statistics.com, который теперь предлагает порядка 100 курсов в области статистики, из которых примерно половина предназначена для аналитиков данных. Нанимая в качестве преподавателей ведущих авторов и шлифуя маркетинговую стратегию для привлечения внимания профессиональных аналитиков данных, Питер развил широкое представление о целевом рынке и свои собственные экспертные знания для его завоевания.

Эндрю Брюс (Andrew Bruce) имеет более чем 30-летний стаж работы в области статистики и науки о данных в академической сфере, правительстве и бизнесе. Он обладает степенью кандидата наук в области статистики Вашингтонского университета и опубликовал несколько работ в рецензируемых журналах. Он разработал статистико-ориентированные решения широкого спектра задач, с которыми сталкиваются разнообразные отрасли, начиная с солидных финансовых фирм до интернет-стартапов, и располагает глубоким пониманием практики науки о данных.

Питер Гедек (Peter Gedeck) имеет более чем 30-летний опыт работы в области научных вычислений и науки о данных. После 20 лет работы в качестве вычислительного химика в компании Novartis он занимает должность старшего исследователя данных в компании Collaborative Drug Discovery. Питер специализируется на разработке алгоритмов машинного обучения для предсказания биологических и физико-химических свойств препаратов-кандидатов. Соавтор книги "Добыча закономерностей из данных для бизнес-аналитики" (Data Mining for Business Analytics). Имеет докторскую степень по химии, которую он получил в Университете Эрланген-Нюрнберг в Германии, а в Университете Фернуниверситет-Хаген (Германия) изучал математику.

Предисловие

Книга рассчитана на исследователя данных, имеющего некоторый опыт работы с языком программирования R и/или Python и имеющего предшествующий (возможно, обрывочный или сиюминутный) контакт с математической статистикой. Двое из трех авторов пришли в мир науки о данных из мира статистики и поэтому обладают некоторым пониманием того вклада, который статистика может привнести в науку о данных как прикладную дисциплину. В то же время мы хорошо осведомлены об ограничениях традиционного статистического образования: статистика как дисциплина насчитывает полтора столетия, и большинство учебников и курсов по статистике отягощены кинетикой и инерцией океанского лайнера.

В основе настоящей книги лежат две цели:

- ◆ представить в удобоваримой, пригодной для навигации и легкодоступной форме ключевые понятия из статистики, которые относятся к науке о данных;
- ◆ объяснить, какие понятия важны и полезны с точки зрения науки о данных, какие менее важны и почему.

Условные обозначения, принятые в книге

В книге используются следующие типографические условные обозначения:

- ◆ *курсив* указывает на новые термины;
- ◆ **полужирный шрифт** — URL-адреса, адреса электронной почты;
- ◆ моноширинный шрифт используется для листингов программ, а также внутри абзацев для ссылки на элементы программ, такие как переменные или имена функций, базы данных, типы данных, переменные среды, инструкции и ключевые слова.

Ключевые термины

Наука о данных — это сплав многочисленных дисциплин, включая статистику, информатику, информационные технологии и конкретные предметные области. В результате при упоминании конкретной идеи могут использоваться несколько разных терминов. Ключевые термины и их синонимы в данной книге будут выделяться в специальных врезках, таких как показанные ниже.



Данный элемент обозначает подсказку или совет.



Данный элемент обозначает общее замечание.



Данный элемент обозначает предупреждение или предостережение.

Использование примеров кода

Во всех случаях в этой книге даются примеры исходного кода сначала на языке R и потом на Python. Во избежание ненужных повторов мы обычно показываем только результат и графики, создаваемые кодом на R. Мы также пропускаем код, необходимый для загрузки требуемых пакетов и наборов данных. Вы можете найти полный код, а также наборы данных для скачивания по адресу

<https://github.com/gedeck/practical-statistics-for-data-scientists>.

Эта книга предназначена для того, чтобы помочь вам в выполнении вашей работы. В целом, если код примеров предлагается вместе с книгой, вы можете использовать его в своих программах и документации. Вам не нужно связываться с нами с просьбой о разрешении, если вы не воспроизводите значительную часть кода. Например, написание программы, которая использует несколько фрагментов кода из данной книги, официального разрешения не требует. Ответ на вопрос путем цитирования этой книги и цитирования кода примера разрешения не требует. Включение значительного количества примеров кода из этой книги в документацию вашего продукта действительно требует отдельной миссии.

Адаптированный вариант примеров в виде электронного архива вы можете скачать по ссылке **<ftp://ftp.bhv.ru/9785977567053.zip>**. Эта ссылка доступна также со страницы книги на сайте www.bhv.ru.

Благодарности

Авторы признают усилия многих людей, которые помогли сделать эту книгу реальностью.

Герхард Пилчер (Gerhard Pilcher), генеральный директор компании по добыче регулярностей из данных Elder Research, ознакомился с ранними черновиками книги и дал нам подробные и полезные исправления и комментарии. Помимо этого, Аня

МакГирк (Anya McGuirk) и Вэй Сяо (Wei Xiao), статистики из компании SAS, а также Джей Хилфигер (Jay Hilfiger), автор издательства O'Reilly, предоставили полезные отзывы о первоначальных набросках книги. Тосиаки Курокава (Toshiaki Kurokawa), который перевел первое издание на японский язык, проделал всестороннюю работу по пересмотру и исправлению этого процесса. Аарон Шумахер (Aaron Schumacher) и Вальтер Пачковский (Walter Paczkowski) тщательно рассмотрели второе издание книги и предоставили целый ряд полезных и ценных предложений, за которые мы им чрезвычайно признательны. Излишне говорить, что все оставшиеся ошибки принадлежат только нам.

В издательстве O'Reilly Шеннон Катт (Shannon Cutt) в дружеской атмосфере сопровождал нас в течение всего процесса публикации и в меру подстегивал нашу работу, в то время как Кристен Браун (Kristen Brown) плавно провела нашу книгу через стадию производства. Рэйчел Монаган (Rachel Monaghan) и Элиаху Сассман (Eliahu Sussman) старательно и терпеливо исправляли и улучшали наш почерк, а Эллен Траутман-Зайг (Ellen Troutman-Zaig) готовила предметный указатель. Николь Таш (Nicole Tache) взяла на себя бразды правления вторым изданием и одновременно эффективно руководила этим процессом, а также предоставила ряд хороших редакторских предложений для улучшения читаемости книги для широкой аудитории. Мы также благодарим Мари Бо-Гуро (Marie Beaugureau), которая инициировала наш проект в O'Reilly, а также Бена Бенгфорта (Ben Bengfort), автора O'Reilly и инструктора в Statistics.com, который познакомил нас с O'Reilly.

Мы, как и эта книга, также извлекли пользу из многочисленных бесед Питера с Галитом Шмуэли (Galit Shmueli), соавтором других книжных проектов.

Наконец, мы хотели бы особо поблагодарить Элизабет Брюс (Elizabeth Bruce) и Дебору Доннелл (Deborah Donnell), чьи терпение и поддержка сделали это начинание возможным.

Комментарии переводчика

В центре внимания машинного обучения и его подобласти, глубокого обучения, находится автоматически обучающаяся система, т. е. система, способная с течением времени приобретать новые знания и улучшать свою работу, используя поступающие данные. В зарубежной специализированной литературе *передача* знаний и *получение* знаний выражаются отдельными терминами: *teach/train* (*обучать/тренировать*), с одной стороны, и *learn* (*заучивать, усваивать, учиться*) — с другой.

В контексте строительства автоматически обучающихся моделей *тренировка* — это подача исследователем на вход обучающегося алгоритма нормализованных данных с целью получения обучившейся модели, которую можно применять к ранее не встречавшимся данным с целью предсказания или классифицирования. *Самообучение* же (*learning*) — это работа, которую выполняет обучающийся алгоритм по усвоению связей и закономерностей в данных или изменению и закреплению своего поведения.

В зарубежной литературе появление термина learning подразумевает исключительно вышесказанное — *самообучение*, т. е. самостоятельное *заучивание, усвоение* алгоритмом регулярностей или параметров. Иными словами, учитель как таковой отсутствует. Усвоение происходит под контролем (контролируемое самообучение, когда данные, на которых модель тренируется, были заранее помечены) и бесконтрольно (неконтролируемое самообучение, когда данные поступают непомеченными). Использование терминов "обучение с учителем" и "обучение без учителя" уводит в сторону от сути дела и затрудняет вразумительный перевод, скажем, таких разновидностей самообучения, как semi-supervised learning или self-supervised learning.

Опираясь на такое размежевание терминов, зонтичный термин machine learning (машинное обучение) следует всегда понимать как *усвоение знаний алгоритмической машиной*, а следовательно, более соответствовать оригиналу будет термин *машинное самообучение* или *автоматическое усвоение [регулярностей]*.

Весомым аргументом в пользу этих вариантов термина является и то, что с начала 1960-х и до середины 1980-х годов в ходу был похожий термин — "обучающиеся машины" (см. работы А. Тьюринга, К. Шеннона, Н. Винера, Н. Нильсона, Я. З. Цыпкина и др.).

Следуя принципам здравого смысла и бритвы Оккама, в настоящем переводе за основу принят зарубежный подход.

Разведывательный анализ данных

Эта глава посвящена первому шагу в любом проекте науки о данных — разведыванию данных.

Классическая статистика фокусировалась почти исключительно на *статистическом выводе* — иногда сложном наборе процедур для получения заключения о крупных популяциях, основываясь на малых выборках. В 1962 году Джон У. Тьюки (рис. 1.1) призвал к реформированию статистики в своей концептуальной работе "Будущее анализа данных" [Tukey-1962]¹. Он предложил новую научную дисциплину под названием "*анализ данных*", которая включала статистический вывод как всего лишь один из компонентов. Тьюки наладил связи с инженерным и вычислительным сообществами (он придумал термины *bit* от англ. *binary digit*, т. е. *бит*, и *software*, т. е. *программно-информационное обеспечение, вычислительная система*), и его первоначальные принципы оказались удивительно прочными и составляют часть фундамента науки о данных. Область разведывательного анализа данных была сформулирована во многом благодаря ставшей уже классической книге Тьюки 1977 года "Разведывательный анализ данных" [Tukey-1977]. Тьюки представил простые диаграммы (например, коробчатые диаграммы, диаграммы рассеяния), которые, наряду со сводной статистикой (среднее, медиана, квантили и др.), помогали рисовать картину набора данных.



Рис. 1.1. Джон Тьюки, выдающийся статистик, чьи идеи, разработанные более чем 50 лет назад, формируют фундамент науки о данных

¹ См. <https://oreil.ly/LQw6q>.

Благодаря доступности вычислительных мощностей и выразительному программному обеспечению для анализа данных разведывательный анализ данных эволюционировал далеко за пределы своей исходной области. Ключевыми факторами развития этой дисциплины стали быстрое развитие новых технологий, доступ ко все большему объему данных и более широкое использование количественного анализа в различных дисциплинах. Дэвид Донохо, профессор статистики Стэнфордского университета и бывший студент-старшекурсник Тьюки, написал превосходную статью "50 лет науки о данных", основанную на его выступлении на семинаре в честь 100-летия Тьюки, проходившем в Принстоне, шт. Нью-Джерси [Donoho-2015]. В ней Донохо прослеживает генезис науки о данных вплоть до новаторской работы Тьюки в области анализа данных.

Элементы структурированных данных

Данные поступают из многочисленных источников: показаний датчиков, событий, текста, фотоснимков и видео. *Интернет вещей* (Internet of Things, IoT) извергает потоки информации. Значительная часть этих данных не структурирована: фотоснимки представляют собой набор пикселей, при этом каждый пиксел содержит информацию о цвете в формате RGB (красный, зеленый, синий). Тексты состоят из последовательностей словарных и несловарных символов, часто разбитых на разделы, подразделы и т. д. Потоки нажатий клавиш представляют собой последовательности действий пользователя, взаимодействующего с приложением или веб-страницей. По сути дела, главенствующая задача науки о данных состоит в том, чтобы перерабатывать этот поток сырых данных в информацию, имеющую практическое значение. Для того чтобы применить охваченные в этой книге статистические понятия, неструктурированные сырые данные нужно переработать в структурированную форму. Одной из наиболее часто встречающихся форм структурированных данных является таблица со строками и столбцами, т. к. данные могут поступать из реляционной базы данных либо собираться для исследования.

Существуют два базовых типа структурированных данных: числовой и категориальный. Числовые данные имеют две формы: *непрерывную*, как, например, скорость ветра или продолжительность времени, и *дискретную*, как, например, количество возникновений события. *Категориальные* данные принимают только фиксированный набор значений, как, например, тип экрана телевизора (плазма, LCD, LED и т. д.) или название штата (Алабама, Аляска и т. д.). *Двоичные* данные являются важным особым случаем категориальных данных. Эти данные принимают только одно из двух значений, таких как 0/1, да/нет или истина/ложь. Еще один полезный тип категориальных данных — *порядковые* данные, в которых категории упорядочены; их примером является числовой рейтинг (1, 2, 3, 4 или 5).

Зачем нам вообще вникать в таксономию типов данных? Оказывается, что для целей анализа данных и предсказательного моделирования тип данных играет важную роль, помогая определять тип визуального изображения, анализа данных либо статистической модели. По сути дела, в вычислительных системах науки о данных, таких как *R* и *Python*, эти типы данных используются для улучшения вычислитель-

ный производительности. Еще важнее, что тип переменной определяет то, каким образом вычислительная система будет трактовать вычисления для этой переменной.

Ключевые идеи для типов данных

Числовые (numeric)

Данные, которые выражаются на числовой шкале.

Непрерывные (continuous)

Данные, которые могут принимать любое значение в интервале.

Синонимы: интервал, число с плавающей точкой, числовая величина.

Дискретные (discrete)

Данные, которые могут принимать только целочисленные значения, такие как количества.

Синонимы: целое число, количество, счетная величина.

Категориальные (categorical)

Данные, которые могут принимать только конкретное множество значений, представляющих множество возможных категорий.

Синонимы: перечисления, пронумерованные и номинальные данные, факторы.

Двоичные (binary)

Частный случай категориальных данных с двумя категориями значений, например, 0/1, истина/ложь.

Синонимы: дихотомическая, логическая, индикаторная, булева величина.

Порядковые (ordinal)

Категориальные данные, которые имеют явно заданное упорядочение.

Синоним: упорядоченный фактор.

Разработчики вычислительных систем и программисты баз данных могут задаться вопросом: а зачем нам в аналитике вообще нужно разделение на *категориальные* и *порядковые* данные? В конце концов, категории являются просто коллекцией текстовых (либо числовых) значений, и опорная база данных автоматически работает с их внутренним представлением. Однако четкая идентификация данных в качестве категориальных, в отличие от текстовых, действительно предлагает ряд преимуществ.

- ◆ Знание о том, что данные являются категориальными, может служить сигналом для вычислительной системы о том, каким образом должны вести себя статистические процедуры, такие как продуцирование графика или подгонка модели. В частности, в R порядковые данные могут быть представлены как упорядоченный фактор `ordered.factor`, сохраняя заданную пользователем упорядоченность

в графиках, таблицах и моделях. В Python пакет `scikit-learn` поддерживает порядковые данные с помощью класса `sklearn.preprocessing.OrdinalEncoder`.

- ◆ Хранение и индексация данных могут быть оптимизированы (как в реляционной базе данных).
- ◆ Возможные значения, принимаемые конкретной категориальной переменной, поддерживаются в вычислительной системе (как, например, структура данных `enum`).

Третья "выгода" может привести к непреднамеренному или неожиданному поведению: дефолтное² поведение функций импорта данных в R (например, `read.csv`) состоит в автоматическом конвертировании текстового столбца в `factor`. Последующие операции на этом столбце будут исходить из предположения о том, что единственно допустимыми значениями для этого столбца являются те, которые были первоначально импортированы, и присвоение нового текстового значения выдаст предупреждение и сообщение `NA` (not available) об отсутствии значения. Пакет `pandas` в Python не будет выполнять такую конвертацию автоматически. Однако вы можете явно указать столбец как категориальный в функции `read_csv`.

Ключевые идеи для структурированных данных

- В вычислительной системе данные, как правило, классифицируются по типу.
- Типы данных включают числовые (непрерывные, дискретные) и категориальные (двоичные, порядковые).
- Ввод данных в вычислительной системе действует как сигнал для вычислительной системы о том, как обрабатывать данные.

Дополнительные материалы для чтения

- ◆ Документация пакета `pandas`³ описывает разные типы данных и то, как ими можно манипулировать на языке Python.
- ◆ Типы данных могут вызывать путаницу, поскольку типы могут накладываться друг на друга, а таксономия в одной вычислительной системе может отличаться от таксономии в другой. Веб-сайт `R-tutorial`⁴ с практическими занятиями по языку R охватывает таксономию для R. Документация `pandas`⁵ описывает разные типы данных и то, как ими можно манипулировать в языке Python.

² То есть такое поведение, которое срабатывает автоматически, если не указано иное. Термин `default` в атрибутивной позиции встречается в зарубежной информатике слишком часто, чтобы его игнорировать. — *Прим. перев.*

³ См. <https://oreil.ly/UGX-4>.

⁴ См. <https://oreil.ly/2YUoA>.

⁵ См. <https://oreil.ly/UGX-4>.

- ◆ Базы данных более детализированы в том, как они классифицируют типы данных, инкорпорируя уровни прецизионности, поля фиксированной или переменной длины и многое другое; см. руководство W3Schools по SQL⁶.

Прямоугольные данные

В науке о данных типичным опорным каркасом для анализа является объект с *прямоугольными данными* наподобие электронной таблицы или таблицы базы данных.

Прямоугольные данные — это общий термин для двумерной матрицы, в которой строки обозначают записи (случай), а столбцы — признаки (переменные). *Кадр данных* — это специфичный формат, присущий языкам R и Python. Исходно данные не всегда поступают в такой форме: неструктурированные данные (например, текст) необходимо обработать и привести к такому виду, чтобы их можно было представить как множество признаков в прямоугольных данных (см. раздел "*Элементы структурированных данных*" ранее в этой главе). Данные в реляционных базах данных должны быть извлечены и помещены в одну-единственную таблицу для большинства задач по анализу данных и моделированию.

Ключевые термины для прямоугольных данных

Кадр данных (data frame)

Прямоугольные данные (подобно электронной таблице) — это базовая структура данных для статистических и автоматически обучающихся моделей.

Признак (feature)

Столбец в таблице обычно называется признаком.

Синонимы: атрибут, вход, предсказатель, предиктор, переменная.

Исход (outcome)

Многие проекты науки о данных предусматривают с предсказание исхода — нередко в формате да/нет (например, в табл. 1.1 это ответ на вопрос "Были ли торги состязательными или нет?"). Признаки иногда используются для предсказания исхода в эксперименте или статистическом исследовании.

Синонимы: результат, зависимая переменная, отклик, цель, выход.

Записи (records)

Строка в таблице обычно называется записью.

Синонимы: случай, пример, прецедент, экземпляр, наблюдение, шаблон, паттерн, образец.

⁶ См. <https://oreil.ly/cThTM>.

Таблица 1.1. Типичный формат данных

Категория	Валюта	Рейтинг продавца	Длительность	День закрытия	Цена закрытия	Цена открытия	Состязательность?
Music/Movie/Game	US	3249	5	Mon	0,01	0,01	0
Music/Movie/Game	US	3249	5	Mon	0,01	0,01	0
Automotive	US	3115	7	Tue	0,01	0,01	0
Automotive	US	3115	7	Tue	0,01	0,01	0
Automotive	US	3115	7	Tue	0,01	0,01	0
Automotive	US	3115	7	Tue	0,01	0,01	0
Automotive	US	3115	7	Tue	0,01	0,01	1
Automotive	US	3115	7	Tue	0,01	0,01	1

В табл. 1.1 показана смесь измеряемых или количественных данных (например, длительность и цена) и категориальных данных (например, категория и валюта). Как уже упоминалось ранее, специальной формой категориальной переменной является двоичная переменная (да/нет либо 0/1), которую можно увидеть в самом правом столбце табл. 1.1, — индикаторная переменная, показывающая, были ли торги состязательными (было несколько претендентов или нет). Эта индикаторная переменная также является переменной *результата*, когда сценарий заключается в том, чтобы предсказать состязательность или несостязательность аукциона.

Кадры данных и индексы

Традиционные таблицы базы данных имеют один или несколько столбцов, именуемых *индексом*. Он значительно повышает эффективность некоторых запросов к базе данных. В Python при использовании библиотеки `pandas` основной прямоугольной структурой данных является объект `DataFrame`, содержащий таблицу данных. По умолчанию для объекта `DataFrame` создается автоматический целочисленный индекс, который основывается на порядке следования строк таблицы. В `pandas` также существует возможность задавать многоуровневые/иерархические индексы с целью повышения эффективности некоторых операций.

В R базовой прямоугольной структурой данных является объект `data.frame`, который тоже имеет неявный целочисленный индекс, основанный на порядке следования строк. Нативный для R объект `data.frame` не поддерживает определяемые пользователем либо многоуровневые индексы, хотя собственный ключ может быть создан посредством атрибута `row.names`⁷. Для преодоления этого недостатка широкое

⁷ Атрибут `row.names` — это символьный вектор длиной, которая соответствует числу строк в кадре данных, без дубликатов и пропущенных значений. — *Прим. перев.*

распространение получили два новых пакета: `data.table` и `dplyr`. Оба поддерживают многоуровневые индексы и обеспечивают значительное ускорение в работе с объектом `data.frame`.



Различия в терминологии

Терминология для прямоугольных данных может вызывать путаницу. В статистике и науке о данных используются разные термины, которые говорят об одном и том же. Для статистиков в модели существуют *предсказательные переменные*, которые применяются для предсказания *отклика* либо *зависимой переменной*. В отличие от них для исследователя данных существуют признаки, которые используются для предсказания цели. В особенности сбивает с толку один синоним: специалисты по информатике употребляют термин *sample* для обозначения одной-единственной строки, имея под ним в виду образец, тогда как для статистика *sample* означает коллекцию строк, т. е. выборку.

Непрямоугольные структуры данных

Помимо прямоугольных данных существуют и другие структуры данных.

Данные временного ряда записывают поочередные замеры одной и той же переменной. Эти данные представляют собой сырой материал для методов статистического прогнозирования, и они также являются ключевым компонентом данных, производимых устройствами — Интернет вещей.

Пространственные структуры данных, которые используются в картографической и геопространственной аналитике, более сложны и вариабельны, чем прямоугольные структуры данных. В их *объектном* представлении центральной частью данных являются объект (например, дом) и его пространственные координаты. В *полевой* проекции, в отличие от него, основное внимание уделяется малым единицам пространства и значению соответствующей метрики (яркости пиксела, например).

Графовые (или сетевые) структуры данных используются для представления физических, социальных и абстрактных связей. Например, граф социальной сети, такой как Facebook или LinkedIn, может представлять связи между людьми в сети. Соединенные дорогами центры распределения являются примером физической сети. Графовые структуры широко применяются в некоторых типах задач, таких как оптимизация сети и рекомендательные системы.

Каждый из этих типов данных имеет в науке о данных свою специализированную методологию. В центре внимания настоящей книги находятся прямоугольные данные — основополагающий структурный элемент в предсказательном моделировании.



Графы и графики в статистике

В англоязычной информатике и информационных технологиях термин *graph* (граф) обычно обозначает описание связей среди сущностей и опорную структуру данных. В статистике термин *graph* (график) чаще используется для обозначения самых разных диаграмм, графиков и визуализаций, а не только связей среди сущностей.

Ключевые идеи для прямоугольных данных

- В науке о данных базовой структурой данных является прямоугольная матрица, в которой строки являются записями, а столбцы — переменными (признаками).
- Терминология может вызывать путаницу, поскольку существуют разнообразные синонимы, вытекающие из разных дисциплин, которые вносят свой вклад в науку о данных (статистика, информатика и информационные технологии).

Дополнительные материалы для чтения

- ♦ Документация по кадрам данных в R⁸.
- ♦ Документация по кадрам (таблицам) данных в Python⁹.

Оценки центрального положения

Переменные с измеряемыми или количественными данными могут иметь тысячи четко различимых значений. Базовый шаг в разведывании данных состоит в получении "типичного значения" для каждого признака (переменной): оценки того, где расположено большинство данных (т. е. их центральной тенденции).

Ключевые термины оценок центрального положения

Среднее (mean)

Сумма всех значений, деленная на число значений.

Синонимы: среднее арифметическое.

Среднее взвешенное (weighted mean)

Сумма произведений всех значений на их веса, деленная на сумму весов.

Синонимы: среднее арифметическое взвешенное.

Медиана (median)

Такое значение, что половина сортированных данных находится выше и ниже данного значения.

Синоним: 50-й процентиль.

Медиана взвешенная (weighted median)

Такое значение, что половина суммы весов находится выше и ниже сортированных данных.

⁸ См. <https://oreil.ly/NsONR>.

⁹ См. <https://oreil.ly/oxDKQ>.

Среднее усеченное (trimmed mean)

Среднее число всех значений после отбрасывания фиксированного числа предельных значений.

Синонимы: обрезанное среднее.

Робастный (robust)

Не чувствительный к предельным значениям.

Синоним: устойчивый.

Выброс (outlier)

Значение данных, которое сильно отличается от большинства данных.

Синонимы: предельное, экстремальное или аномальное значение.

На первый взгляд задача обобщения данных выглядит довольно тривиальной: надо просто взять *среднее арифметическое* данных (см. раздел "*Среднее*" далее в этой главе). На самом деле несмотря на то, что среднее вычисляется довольно просто и его выгодно использовать, оно не всегда бывает лучшей мерой центрального значения. По этой причине в статистике были разработаны и популяризированы несколько альтернативных оценок среднего значения.



Метрические и оценочные показатели

В статистике термин "*оценка*" часто используется для значений, вычисляемых из данных, которые находятся под рукой, для того чтобы провести различие между тем, что мы видим из этих данных, и теоретически истинным или точным положением дел. Исследователи данных и бизнес-аналитики с большей вероятностью будут называть такие значения метрическими показателями, или *метриками*. Эта разница отражает подходы, принятые в статистике, в отличие от науки о данных: учет неопределенности лежит в основе статистики, тогда как центром внимания науки о данных являются конкретные деловые или организационные целевые критерии. Следовательно, статистики оценивают, а исследователи данных измеряют.

Среднее

Самой базовой оценкой центрального положения является *среднее значение*, или *среднее арифметическое*. Среднее — это сумма всех значений, деленная на число значений. Рассмотрим следующий ряд чисел: $\{3, 5, 1, 2\}$. Среднее составит $(3 + 5 + 1 + 2) / 4 = 11 / 4 = 2,75$. Вы часто будете встречать символ \bar{x} (произносится "икс с чертой"), который обозначает среднее значение выборки из популяции, или генеральной совокупности. Формула среднего значения для ряда из n значений x_1, x_2, \dots, x_n такова:

$$\text{среднее} = \bar{x} = \frac{\sum_{i=1}^n x_i}{n}.$$



N (или n) обозначает общее число записей или наблюдений. В статистике это обозначение используется с заглавной буквы, если оно обозначает популяцию, и строчной, если оно обозначает выборку из популяции. В науке о данных это различие не является принципиальным, и поэтому можно увидеть и то и другое.

Разновидностью среднего является *среднее усеченное*, которое вычисляется путем отбрасывания фиксированного числа сортированных значений с каждого конца последовательности и затем взятия среднего арифметического оставшихся значений. Если представить сортированные значения как $x_{(1)}, x_{(2)}, \dots, x_{(n)}$, где $x_{(1)}$ — это наименьшее значение и $x_{(n)}$ — наибольшее, то формула для вычисления усеченного среднего с пропуском p самых малых и самых крупных значений будет следующей:

$$\text{среднее усеченное} = \bar{x} = \frac{\sum_{i=p+1}^{n-p} x_{(i)}}{n - 2p}.$$

Усеченное среднее устраняет влияние предельных значений. Например, в международных состязаниях по прыжкам в воду верхние и нижние баллы пяти судей отбрасываются, и итоговым баллом считается среднеарифметический балл трех оставшихся судей¹⁰. Такой подход не дает одному судье манипулировать баллом, возможно, чтобы оказать содействие спортсмену из своей страны. Усеченные средние получили широкое распространение и во многих случаях являются предпочтительными вместо обычного среднего (продолжения обсуждения этой темы *см. в разделе "Медиана и робастные оценки" далее в этой главе*).

Еще один вид среднего значения — это *средневзвешенное значение*, которое вычисляется путем умножения каждого значения данных x_i на свой вес w_i и деления их суммы на сумму весов. Формула средневзвешенного выглядит так:

$$\text{среднее взвешенное} = \bar{x}_w = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}.$$

Существуют два главных побудительных мотива для использования средневзвешенного значения.

- ♦ Некоторые значения внутренне более переменчивы, чем другие, и сильно переменчивым наблюдениям придается более низкий вес. Например, если мы берем среднее арифметическое данных, поступающих от многочисленных датчиков, и один из датчиков менее точен, тогда вес данных от этого датчика можно понизить.
- ♦ Собранные данные не одинаково представляют разные группы, которые мы заинтересованы измерить. Например, в зависимости от того, каким образом про-

¹⁰ См. <https://oreil.ly/uV4P0>.

водится онлайн-эксперимент, у нас может не быть набора данных, который точно отражает все группы в пользовательской базе. Для того чтобы это исправить, можно придать более высокий вес значениям из тех групп, которые были представлены недостаточно.

Медиана и робастные оценки

Медиана — это число, расположенное в отсортированном списке данных ровно посередине. Если имеется четное число данных, то срединным значением является то, которое не находится в наборе данных фактически, а является средним арифметическим двух значений, которые делят отсортированные данные на верхнюю и нижнюю половины. По сравнению со средним, в котором используются абсолютно все наблюдения, медиана зависит только от значений в центре отсортированных данных. Хотя это может выглядеть как недостаток, поскольку среднее значение намного чувствительнее к данным, существует много примеров, в которых медиана является более подходящей метрикой центрального положения. Скажем, мы хотим взглянуть на типичные доходы домохозяйств в округах, расположенных на побережье озера Вашингтон в Сиэтле. При сравнении округа Медина с округом Уиндермир использование среднего значения дало бы совершенно разные результаты, потому что в Медине живет Билл Гейтс. Если же мы будем использовать медиану, то уже не будет иметь значения, насколько богатым является Билл Гейтс, — позиция срединного наблюдения останется той же.

По тем же самым причинам, по которым используется среднее взвешенное, можно вычислить и *медиану взвешенную*. Как и с медианой, мы сначала выполняем сортировку данных, несмотря на то что с каждым значением данных связан вес. В отличие от срединного числа медиана взвешенная — это такое значение, в котором сумма весов равна для нижней и верхней половин отсортированного списка. Как и медиана, взвешенная медиана робастна к выбросам.

Выбросы

Медиана называется *робастной* оценкой центрального положения, поскольку она не находится под влиянием *выбросов* (предельных случаев), которые могут исказить результаты. Выброс — это любое значение, которое сильно дистанцировано от других значений в наборе данных. Точное определение выброса является несколько субъективным, несмотря на то что в различных сводных данных и графиках используются некоторые правила (см. раздел "*Процентили и коробчатые диаграммы*" далее в этой главе). Выброс как таковой не делает значение данных недопустимым или ошибочным (как в предыдущем примере с Биллом Гейтсом). Вместе с тем выбросы часто являются результатом ошибок данных, таких как смешивание данных с разными единицами измерения (километры с метрами) или плохие показания от датчика. Когда выбросы являются результатом неправильных данных, среднее значение будет показывать плохую оценку центрального положения, тогда как медиана будет по-прежнему допустимой. В любом случае выбросы должны быть выявлены и обычно заслуживают дальнейшего расследования.



Обнаружение аномалий

В отличие от типичного анализа данных, где выбросы иногда информативны, а иногда — досадная помеха, в *обнаружении аномалий* целевыми объектами являются именно выбросы, и значительный массив данных преимущественно служит для определения "нормы", с которой соизмеряются аномалии.

Медиана не единственная робастная оценка центрального положения. На самом деле в целях предотвращения влияния выбросов широко используется и среднее усеченное. Например, усечение нижних и верхних 10% данных (общепринятый выбор) обеспечит защиту от выбросов во всех, кроме самых малых, наборах данных. Среднее усеченное может считаться компромиссом между медианой и средним: оно робастно к предельным значениям в данных, но использует больше данных для расчета оценки центрального положения.



Другие робастные метрики центрального положения

В статистике была разработана масса других оценщиков, так называемых эстиматоров, центрального положения преимущественно с целью разработки более робастных инструментов оценки, чем среднее, и более эффективных (т. е. способных лучше обнаруживать небольшие разницы в центральном положении между наборами данных). Эти методы потенциально полезны для небольших наборов данных. Вместе с тем они едва дают дополнительные выгоды в условиях крупных или даже умеренно размерных наборов данных.

Пример: средние оценки численности населения и уровня убийств

В табл. 1.2 показаны первые несколько строк из набора данных, содержащего сведения о численности населения и уровне убийств (в единицах убийств на 100 тыс. человек в год) по каждому штату.

Таблица 1.2. Несколько строк кадра данных `data.frame` о численности населения и уровне убийств по штатам

№	Штат	Население	Уровень убийств	Аббревиатура
1	Alabama	4 779 736	5,7	AL
2	Alaska	710 231	5,6	AK
3	Arizona	6 392 017	4,7	AZ
4	Arkansas	2 915 918	5,6	AR
5	California	37 253 956	4,4	CA
6	Colorado	5 029 196	2,8	CO
7	Connecticut	3 574 097	2,4	CT
8	Delaware	897 934	5,8	DE

Вычислим среднее, среднее усеченное и медиану численности населения, используя R:

```
> state <- read.csv(file="/Users/andrewbruce1/book/state.csv")
> mean(state[["Population"]])
[1] 6162876
> mean(state[["Population"]], trim=0.1)
[1] 4783697
> median(state[["Population"]])
[1] 4436370
```

Для вычисления среднего и медианы на Python мы можем использовать методы кадра данных пакета `pandas`. Для усеченного среднего значения требуется функция `trim_mean` из библиотеки `scipy.stats`:

```
state = pd.read_csv('state.csv')
state['Population'].mean()
trim_mean(state['Population'], 0.1)
state['Population'].median()
```

Среднее больше среднего усеченного, которое больше медианы.

Это вызвано тем, что среднее усеченное исключает самые крупные и самые малые пять штатов (`trim=0.1` отбрасывает по 10% с каждого конца). Если мы захотим вычислить среднестатистический уровень убийств в стране, то должны использовать среднее взвешенное или медиану, чтобы учесть разную численность населения в штатах. Поскольку базовый R не имеет функции для взвешенной медианы, мы должны установить пакет, такой как `matrixStats`:

```
> weighted.mean(state[["Murder.Rate"]], w=state[["Population"]])
[1] 4.445834
> library("matrixStats")
> weightedMedian(state[["Murder.Rate"]], w=state[["Population"]])
[1] 4.4
```

Взвешенное среднее значение доступно с помощью пакета `NumPy`. Для взвешенной медианы мы можем использовать специализированный пакет `wquantiles`:

```
np.average(state['Murder.Rate'], weights=state['Population'])
wquantiles.median(state['Murder.Rate'], weights=state['Population'])
```

В этом случае среднее взвешенное и медиана почти одинаковы.

Ключевые идеи для оценок центрального положения

- Базовой метрикой центрального положения является среднее, но оно может быть чувствительным к предельным значениям (выбросам).
- Другие метрики (медиана, среднее усеченное) являются более робастными.

Дополнительные материалы для чтения

- ♦ Статья Википедии¹¹ о центральной тенденции содержит обширное обсуждение различных мер оценивания центрального положения.
- ♦ Классическая книга Джона Тьюки "Разведывательный анализ данных" [Tukey-1977] по-прежнему пользуется широким спросом.

Оценки вариабельности

Центральное положение — это всего одна из размерностей в обобщении признака. Вторая размерность, *вариабельность*, именуемая также *дисперсностью*, показывает, сгруппированы ли значения данных плотно, или же они разбросаны. В основе статистики лежит вариабельность: ее измерение, уменьшение, различение случайной вариабельности от реальной, идентификация различных источников реальной вариабельности и принятие решений в условиях ее присутствия.

Ключевые термины для метрик вариабельности

Отклонения (deviations)

Разница между наблюдаемыми значениями и оценкой центрального положения.

Синонимы: ошибки, погрешности, остатки.

Дисперсия (variance)

Сумма квадратичных отклонений от среднего, деленная на $n-1$, где n — это число значений данных.

Синонимы: среднеквадратическое отклонение, среднеквадратическая ошибка.

Стандартное отклонение (standard deviation)

Квадратный корень из дисперсии.

Синонимы: норма l_2 , евклидова норма.

Среднее абсолютное отклонение (mean absolute deviation)

Среднее абсолютных значений отклонений от среднего¹².

Синонимы: норма l_1 , манхэттенская норма.

Медианное абсолютное отклонение от медианы (median absolute deviation from the median)

Медиана абсолютных значений отклонений от медианы.

¹¹ См. <https://oreil.ly/qUW2i>.

¹² Абсолютным оно является потому, что суммируются отклонения по модулю, т. к. в противном случае сумма всех разбросов будет равна нулю. — *Прим. перев.*

Размах (range)

Разница между самым крупным и самым малым значениями в наборе данных.

Порядковые статистики (order statistics)

Метрики на основе значений данных, отсортированных от самых малых до самых крупных.

Синоним: ранги.

Процентиль (percentile)

Такое значение, что P процент значений принимает данное значение или меньшее и $(100 - P)$ процент значений принимает данное значение или большее.

Синоним: квантиль.

Межквартильный размах (interquartile range)

Разница между 75-м и 25-м процентилями.

Синонимы: МКР, IQR.

Так же как и в случае центрального положения, которое можно измерить разными способами (среднее, медиана и т. д.), существуют разные способы измерить вариативность.

Стандартное отклонение и связанные с ним оценки

Наиболее широко используемые оценки вариативности основаны на разностях, или *отклонениях*, между оценкой центрального положения и наблюдаемыми данными. Для набора данных $\{1, 4, 4\}$ среднее равняется 3, а медиана — 4. Отклонения от среднего представляют собой разницы: $1 - 3 = -2$, $4 - 3 = 1$, $4 - 3 = 1$. Эти отклонения говорят о том, насколько данные разбросаны вокруг центрального значения.

Один из способов измерить вариативность состоит в том, чтобы оценить типичное значение этих отклонений. Усреднение самих отклонений мало, поэтому отрицательные отклонения нейтрализуют положительные. Фактически сумма отклонений от среднего как раз равна нулю. Вместо этого простой подход заключается в том, чтобы взять среднее абсолютных значений отклонений от среднего значения. В предыдущем примере абсолютное значение отклонений равно $\{2, 1, 1\}$, а их среднее — $(2 + 1 + 1) / 3 = 1,33$. Это и есть *среднее абсолютное отклонение*, которое вычисляется по следующей ниже формуле:

$$\text{среднее абсолютное отклонение} = \frac{\sum_{i=1}^n |x_i - \bar{x}|}{n},$$

где \bar{x} — среднее значение в выборке, или выборочное среднее.

Самыми известными оценками вариабельности являются *дисперсия* и *стандартное отклонение*, которые основаны на квадратических отклонениях. Дисперсия — это среднее квадратических отклонений, стандартное отклонение — квадратный корень из дисперсии.

$$\text{Дисперсия} = s^2 = \frac{\sum (x - \bar{x})^2}{n - 1};$$

$$\text{стандартное отклонение} = s = \sqrt{\text{Дисперсия}}.$$

Стандартное отклонение интерпретируется намного проще, чем дисперсия, поскольку оно находится на той же шкале измерения, что и исходные данные. Однако, учитывая его более сложную и интуитивно менее понятную формулу, может показаться странным, что в статистике стандартному отклонению отдается предпочтение по сравнению со средним абсолютным отклонением. Такое преобладание обязано статистической теории: математически работа с квадратическими значениями намного удобнее, чем с абсолютными, в особенности в случае статистических моделей.

Степени свободы и n или $n - 1$?

В книгах по статистике всегда так или иначе обсуждается вопрос: почему в формуле дисперсии у нас в знаменателе $n - 1$, вместо n , который приводит к понятию *степеней свободы*? Это различие не важно, поскольку n обычно является крупным настолько, что уже не имеет большого значения, будет ли деление выполняться на n или $n - 1$. Однако если вам интересно, то вот объяснение. Оно базируется на предпосылке, что вы хотите получить оценки популяции, основываясь на выборке из нее.

Если в формуле дисперсии применить интуитивно понятный знаменатель n , то истинные значения дисперсии и стандартного отклонения в популяции будут недооценены. Такая оценка называется *смещенной*. Однако если разделить на $n - 1$ вместо n , то стандартное отклонение становится *несмещенной* оценкой.

Полное объяснение причины, почему использование n приводит к смещенной оценке, сопряжено с понятием степеней свободы, которое принимает во внимание число ограничений при вычислении оценки. В данном случае существуют $n - 1$ степеней свободы, поскольку имеется одно ограничение: стандартное отклонение зависит от вычисления среднего в выборке. В большинстве задач исследователям данных не нужно беспокоиться по поводу степеней свободы.

Ни дисперсия и стандартное отклонение, ни среднее абсолютное отклонение не устойчивы к выбросам и предельным значениям (обсуждение темы робастных оценок центрального положения см. в разделе "*Медиана и робастные оценки*" ранее в этой главе). Дисперсия и стандартное отклонение чувствительны к выбросам больше всего, поскольку они основаны на квадратических отклонениях.

Робастной оценкой вариабельности является *медианное абсолютное отклонение от медианы* (median absolute deviation, MAD):

$$\begin{aligned} & \text{медианное абсолютное отклонение} = \\ & = \text{медиана}(|x_1 - m|, |x_2 - m|, \dots, |x_N - m|), \end{aligned}$$

где m — это медиана. Как и в случае с медианой, медианное абсолютное отклонение от медианы не находится под влиянием предельных значений. Можно также вычислить усеченное стандартное отклонение по аналогии с средним усеченным (см. раздел "Среднее" ранее в этой главе).



Дисперсия, стандартное отклонение, среднее абсолютное отклонение и медианное абсолютное отклонение от медианы не являются эквивалентными оценками, даже в случае, когда данные поступают из нормального распределения. На деле стандартное отклонение всегда больше среднего абсолютного отклонения, которое, в свою очередь, больше медианного абсолютного отклонения. Иногда медианное абсолютное отклонение (MAD) умножают на постоянный поправочный коэффициент, чтобы поставить эту метрику в ту же шкалу, что и стандартное отклонение в случае нормального распределения. Часто используемый фактор 1,4826 означает, что 50% нормального распределения попадает в диапазон $\pm \text{MAD}$ (см., например, <https://oreil.ly/SfDk2>).

Оценки на основе процентилей

Другой подход к оцениванию дисперсии основан на рассмотрении разброса сортированных данных, или их спреда. Статистические показатели на основе сортированных (ранжированных) данных называются *порядковыми статистиками*. Элементарная мера — это *размах*: разница между самым крупным и самым малым числом. Минимальные и максимальные значения как таковые полезно знать, поскольку они помогают выявлять выбросы, но размах чрезвычайно чувствителен к выбросам и не очень полезен в качестве общей меры дисперсности в данных.

Во избежание чувствительности к выбросам вы можете обратиться к размаху данных после отбрасывания значений с каждого конца. Эти типы оценок формально основываются на разнице между *процентилями*. В наборе данных P -й процентиль является таким значением, что, по крайней мере, P процентов значений принимает это значение или меньшее, и по крайней мере $(100 - P)$ процентов значений принимает это значение или большее. Например, для нахождения 80-го процентиля надо отсортировать данные. Затем, начиная с самого малого значения, продолжить 80% вверх к самому крупному значению. Отметим, что медиана — это то же самое, что и 50-й процентиль. Процентиль, по существу, аналогичен *квантилю*, при этом квантили индексируются долями (так, квантиль 0,8 — это то же самое, что и 80-й процентиль).

Общепринятой мерой вариабельности является разница между 25-м и 75-м процентилями, которая называется *межквартильным размахом* (interquartile range, IQR). Вот простой пример: 3, 1, 5, 3, 6, 7, 2, 9. Эти числа мы сортируем, получив 1, 2, 3, 3, 5, 6, 7, 9. 25-й процентиль находится в 2,5, и 75-й процентиль — в 6,5, поэтому

межквартильный размах будет $6,5 - 2,5 = 4$. Вычислительная система может иметь немного другие подходы, которые дают отличающиеся ответы (см. приведенное ниже примечание); в типичной ситуации эти отличия малы.

Для очень крупных наборов данных расчет точных процентилей бывает вычислительно очень затратным, поскольку он требует сортировки всех значений данных. В автоматически обучающихся и статистических вычислительных системах используются специальные алгоритмы, такие как [Zhang-Wang-2007], которые получают приближенный процентиль, вычисляя его очень быстро и гарантированно обеспечивая некоторую точность.



Процентиль: точное определение

Если имеется четное число данных (n — четное), то процентиль является неоднозначным понятием в рамках предыдущего определения. На деле можно взять любое значение между порядковыми статистиками $x_{(j)}$ и $x_{(j+1)}$, где j удовлетворяет:

$$100 \cdot \frac{j}{n} \leq P \leq 100 \cdot \frac{j+1}{n}.$$

В формальном плане *процентиль* (P) — это средневзвешенное значение

$$P = (1 - w)x_{(j)} + wx_{(j+1)}$$

для некоторого веса w между 0 и 1. В статистических вычислительных системах содержатся слегка различающиеся подходы к выбору значения w . На самом деле функция `quantile` языка R предлагает девять разных способов вычисления квантиля. За исключением малых наборов данных, вам, как правило, не придется беспокоиться о точном методе расчета процентиля. На момент написания этих строк в Python функция `numpy.quantile` поддерживает только один подход — линейную интерполяцию.

Пример: оценки вариабельности населения штатов

В табл. 1.3 (для удобства взятой повторно из ранее приведенной табл. 1.2) показаны первые несколько строк в наборе данных, содержащем численность населения и уровни убийств для каждого штата.

Таблица 1.3. Несколько строк из кадра `data.frame` с данными о численности населения и уровне убийств по каждому штату

№	Штат	Население	Уровень убийств	Аббревиатура
1	Alabama	4 779 736	5,7	AL
2	Alaska	710 231	5,6	AK
3	Arizona	6 392 017	4,7	AZ
4	Arkansas	2 915 918	5,6	AR
5	California	37 253 956	4,4	CA
6	Colorado	5 029 196	2,8	CO

Таблица 1.3 (окончание)

№	Штат	Население	Уровень убийств	Аббревиатура
7	Connecticut	3 574 097	2,4	CT
8	Delaware	897 934	5,8	DE

Используя встроенные в R функции для стандартного отклонения, межквартильного размаха (IQR) и медианного абсолютного отклонения от медианы (MAD), мы можем вычислить оценки вариабельности данных о населении штатов:

```
> sd(state[["Population"]])
[1] 6848235
> IQR(state[["Population"]])
[1] 4847308
> mad(state[["Population"]])
[1] 3849870
```

Кадр данных пакета `pandas` предоставляет методы для вычисления стандартного отклонения и квантилей. Используя квантили, мы можем легко определить межквартильный размах (IQR). Для робастной оценки медианного абсолютного отклонения (MAD) мы используем функцию `robust.scale.mad` из пакета `statsmodels`:

```
state['Population'].std()
state['Population'].quantile(0.75) - state['Population'].quantile(0.25)
robust.scale.mad(state['Population'])
```

Стандартное отклонение почти вдвое больше медианного абсолютного отклонения MAD (в R по умолчанию показатель MAD корректируется, чтобы быть на той же шкале, что и среднее). И это не удивительно, поскольку стандартное отклонение является чувствительным к выбросам.

Ключевые идеи для оценок вариабельности

- Дисперсия и стандартное отклонение — наиболее широко распространенные и в рутинном порядке регистрируемые статистики вариабельности.
- Обе статистики чувствительны к выбросам.
- Более робастные метрики включают среднее абсолютное отклонение, медианное абсолютное отклонение от медианы и процентиля (квантили).

Дополнительные материалы для чтения

- ♦ Онлайн-статистический ресурс Дэвида Лэйна (David Lane) содержит раздел по процентиям¹³.

¹³ См. <http://onlinestatbook.com/2/introduction/percentiles.html/>.

- ◆ Кевин Дэйвенпорт (Kevin Davenport) на R-bloggers, агрегаторе статей о языке R, предлагает полезный пост¹⁴, посвященный отклонениям от медианы и их робастным свойствам.

Разведывание распределения данных

Все охваченные нами оценки обобщают данные в одном-единственном числе с целью описания центрального положения либо вариабельности данных. Также полезно разведать то, каким образом данные распределены в совокупности.

Ключевые термины для разведывания распределения данных

Коробчатая диаграмма (boxplot)

График, введенный в употребление Тьюки в качестве быстрого способа визуализации распределения данных.

Синоним: диаграмма типа "ящик с усами".

Частотная таблица (frequency table)

Сводка количеств числовых значений, которые разбиты на серию частотных интервалов (корзин, бинов).

Гистограмма (histogram)

График частотной таблицы, в котором корзины откладываются на оси x , а количество (или доля) — на оси y . Несмотря на внешнее сходство, столбиковые графики не следует путать с гистограммами. Обсуждение этой разницы см. в разделе *"Разведывание двоичных и категориальных данных"* далее в этой главе.

График плотности (density plot)

Сглаженная версия гистограммы, часто на основе ядерной оценки плотности.

Процентили и коробчатые диаграммы

В разделе *"Оценки на основе процентилей"* ранее в этой главе мы развели то, каким образом процентили могут использоваться для измерения разброса данных. Процентили также важны для обобщения всего распределения в целом. Общепринято сообщать о квартилях (25-й, 50-й и 75-й перцентили) и децилях (10-й, 20-й, ..., 90-й процентили). Процентили особенно важны для обобщения *хвостов* (внешнего размаха) распределения. Массовая культура ввела в обиход термин *"однопроцентовики"*, который относится к людям в верхнем 99-м процентиле богатства.

В табл. 1.4 показаны некоторые процентили уровня убийств по штатам. В языке R их можно получить при помощи функции `quantile`:

¹⁴ См. <http://www.r-bloggers.com/absolute-deviation-around-the-median/>.


```
quantile(state[["Murder.Rate"]], p=c(.05, .25, .5, .75, .95))
5%    25%    50%    75%    95%
1.600 2.425 4.000 5.550 6.510
```

Таблица 14. Процентили уровня убийств по штатам

5%	25%	50%	75%	95%
1,60	2,42	4,00	5,55	6,51

Медиана равна 4 убийствам на 100 тыс. человек, несмотря на то что присутствует довольно большая вариабельность: 5-й процентиль составляет всего 1,6, тогда как 95-й процентиль — 6,51.

Коробчатые диаграммы, введенные в употребление Тьюки [Tukey-1977], основаны на процентилях и обеспечивают быстрый способ визуализации распределения данных. На рис. 1.2 показана коробчатая диаграмма населения по штатам, произведенная на языке R:

```
boxplot(state[["Population"]]/1000000, ylab="Население, млн человек")
```

Пакет `pandas` предоставляет ряд базовых разведывательных графиков для кадра данных; одним из них являются коробчатые диаграммы:

```
ax = (state['Population']/1_000_000).plot.box()
ax.set_ylabel('Население, млн человек')
```

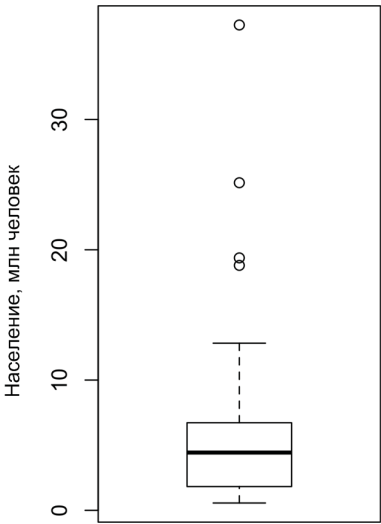


Рис. 1.2. Коробчатая диаграмма населения штатов

Из этой коробчатой диаграммы мы сразу видим, что медианная численность населения штатов составляет около 5 млн человек, половина штатов приходится на численность от 2 до 7 млн, и существуют некоторые выбросы с высоким населением. Верхняя и нижняя части коробки — это соответственно 75-й и 25-й процентили. Медиана показана горизонтальной линией в прямоугольнике. Пунктирные линии,

именуемые *усами*, простираются от верхней и нижней части коробки, и говорят о размахе для основной части данных. Существует много вариантов коробчатой диаграммы; например, обратитесь к документации по R-функции `boxplot` в [R-base-2015]. По умолчанию функция R расширяет усы до самой дальней точки за пределами коробки, за исключением того, что она не будет выходить за пределы 1,5-кратного межквартильного размаха (IQR). Библиотека `matplotlib` использует ту же самую имплементацию; другие вычислительные системы могут использовать другое правило.

Любые данные за пределами усов изображаются в виде отдельных точек или кругов (часто рассматриваемых как выбросы).

Частотные таблицы и гистограммы

Частотная таблица переменной делит диапазон переменной на равноотстоящие сегменты и сообщает о том, сколько значений попадает в каждый сегмент. В табл. 1.5 показана частотная таблица численности населения по штатам, вычисленная в R:

```
breaks <- seq(from=min(state[["Population"]]),
               to=max(state[["Population"]]), length=11)
pop_freq <- cut(state[["Population"]], breaks=breaks,
                right=TRUE, include.lowest = TRUE)
table(pop_freq)
```

Таблица 1.5. Частотная таблица численности населения по штатам

№ корзины	Размах корзины	Количество	Штаты
1	563 626– 4 232 658	24	WY,VT,ND,AK,SD,DE,MT,RI,NH,ME,HI,ID,NE, WV,NM,NV,UT,KS,AR,MS,IA,CT,OK,OR
2	4 232 659– 7 901 691	14	KY,LA,SC,AL,CO,MN,WI,MD,MO,TN,AZ,IN, MA,WA
3	7 901 692– 11 570 724	6	VA,NJ,NC,GA,MI,OH
4	11 570 725– 15 239 757	2	PA,IL
5	15 239 758– 18 908 790	1	FL
6	18 908 791– 22 577 823	1	NY
7	22 577 824– 26 246 856	1	TX
8	26 246 857– 29 915 889	0	
9	29 915 890– 33 584 922	0	
10	33 584 923– 37 253 956	1	CA

Функция `pandas.cut` создает серию, которая увязывает значения с сегментами. Используя метод `value_counts`, мы получаем частотную таблицу:

```
binnedPopulation = pd.cut(state['Population'], 10)
binnedPopulation.value_counts()
```

Наименее густонаселенным штатом является Вайоминг с населением 563 626 человек (согласно переписи 2010 г.), а самым густонаселенным — Калифорния с населением 37 253 956 человек. Это дает нам размах $37\,253\,956 - 563\,626 = 36\,690\,330$, который мы должны разделить на равные корзины — скажем, 10 корзин. При наличии 10 равноразмерных корзин каждая из них будет иметь ширину 3 669 033; таким образом, первая корзина будет простираться от 563 626 до 4 232 658. Верхняя же корзина, 33 584 923–37 253 956, имеет всего один штат — Калифорнию. Две корзины, которые идут непосредственно ниже штата Калифорния, остаются пустыми до тех пор, пока мы не достигнем штата Техас. Важно учитывать пустые корзины; тот факт, что в этих корзинах значения отсутствуют, является полезной информацией. Возможно также будет полезно поэкспериментировать с разными размерами корзин. Если они слишком крупные, то важные признаки распределения могут быть затусшеваны. Если же они слишком малые, то результат будет слишком гранулярным, и возможность наблюдать общую картину будет потеряна.



И таблицы частот, и проценти́ли обобщают данные за счет создания корзин, т. е. частотных интервалов. В общем случае кварти́ли и деци́ли будут иметь одинаковое количество в каждой корзине (равноколичественные корзины), но размеры корзин будут различаться. Частотная таблица, в отличие от них, будет иметь разные количества в корзинах (равноразмерные корзины), и размеры корзин будут одинаковыми.

Гистограмма — это способ визуализации частотной таблицы, где корзины (частотные интервалы) откладываются на оси *x*, а количество данных — на оси *y*. Для того чтобы создать гистограмму, соответствующую табл. 1.5 на языке R, используйте функцию `hist` с аргументом `breaks`:

```
hist(state[["Population"]], breaks=breaks)
```

Пакет `pandas` поддерживает гистограммы для кадров данных благодаря методу `DataFrame.plot.hist`. Используйте именованный аргумент `bins` для определения числа корзин. Различные методы построения графиков возвращают координатный объект `ax`, который позволяет точнее отрегулировать визуализацию с помощью пакета `matplotlib`:

```
ax = (state['Population'] / 1_000_000).plot.hist(figsize=(4, 4))
ax.set_xlabel('Численность населения')
```

В результате получится гистограмма, которая показана на рис. 1.3. В общем случае гистограммы строятся таким образом, что:

- ◆ пустые корзины включаются в график;
- ◆ корзины имеют равную ширину;

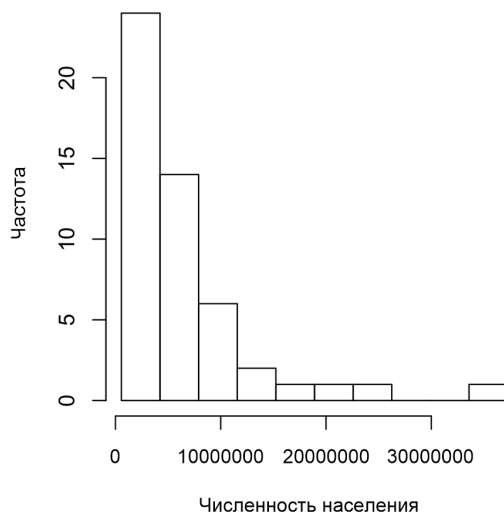


Рис. 1.3. Гистограмма численности населения штатов

- ♦ число корзин (или, что то же самое, размер корзины) задается пользователем;
- ♦ размещение столбцов гистограммы непрерывно — пробелы между ними отсутствуют, если нет пустой корзины.



Статистические моменты

В статистической теории центральное положение и вариабельность упоминаются как моменты *распределения* первого и второго порядка. Моменты третьего и четвертого порядка — это *асимметрия* и *эксцесс*. Под асимметрией понимается смещение данных к большим или меньшим значениям, а под эксцессом — склонность данных к предельным значениям. Для измерения асимметрии и эксцесса, как правило, метрики не используются; вместо этого они выявляются при визуальном осмотре изображений, таких как на рис. 1.2 и 1.3.

Графики и оценки плотности

С гистограммой связан график плотности, который показывает распределение значений данных в форме сплошной линии. График плотности можно рассматривать как сглаженную гистограмму, несмотря на то что он обычно вычисляется непосредственно из данных посредством *ядерной оценки плотности* (краткое руководство см. в [Duong-2001]). На рис. 1.4 представлена оценка плотности, наложенная на гистограмму. В языке R оценку плотности вычисляют при помощи функции `density`:

```
hist(state[["Murder.Rate"]], freq=FALSE)
lines(density(state[["Murder.Rate"]]), lwd=3, col="blue")
```

Пакет `pandas` предоставляет метод `density` для создания плотностного графика. Используйте аргумент `bw_method` для управления плавностью кривой плотности:

```
ax = state['Murder.Rate'].plot.hist(density=True, xlim=[0,12], bins=range(1,12))
state['Murder.Rate'].plot.density(ax=ax) ❶
ax.set_xlabel('Уровень убийств на 100 тыс. населения')
```

❶ Функции построения графика `plot` часто принимают необязательный координатный аргумент (`ax`), что приводит к добавлению нового графика в тот же базовый график.

Ключевое отличие от гистограммы, показанной на рис. 1.3, состоит в шкале оси y : график плотности соответствует отображению гистограммы как доли, а не количеств (в языке R она задается при помощи аргумента `freq=FALSE`). Обратите внимание, что общая площадь под кривой плотности равна 1, и вместо количеств в корзинах вы вычисляете площади под кривой между любыми двумя точками на оси x , которые соответствуют доли распределения, лежащей между этими двумя точками.

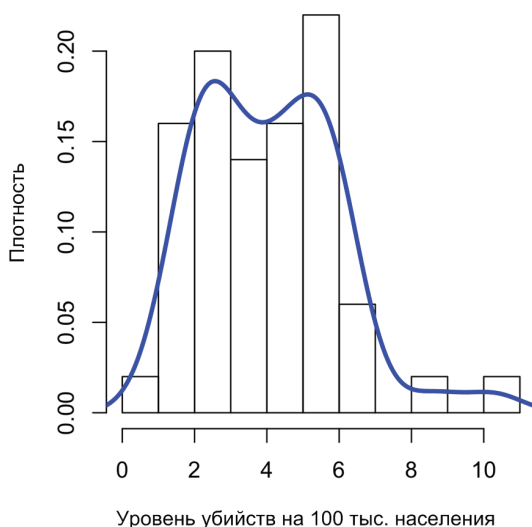


Рис. 1.4. Плотность уровня убийств в штатах



Оценивание плотности

Оценивание плотности является обширной темой с долгой историей в статистической литературе. Фактически было опубликовано свыше 20 программных пакетов R, предлагающих функции оценки плотности. [Deng-Wickham-2011] дает всесторонний анализ пакетов R, особо выделяя пакеты `ASH` и `KernSmooth`. Методы оценки плотности в пакетах `pandas` и `scikit-learn` также предлагают хорошие имплементации. В большинстве задач науки о данных нет необходимости беспокоиться по поводу различных типов оценок; достаточно использовать базовые функции.

Ключевые идеи для разведывания распределения данных

- Частотная гистограмма показывает частотные количества на оси y и значения переменных — на оси x ; она дает визуальное представление о распределении данных.
- Частотная таблица является табличной версией частотных количеств, которые можно найти на гистограмме.
- Коробчатая диаграмма — это диаграмма, в которой верх и низ "коробки" находятся соответственно в 75-м и 25-м процентилях. Она также дает быстрое представление о распределении данных; ее часто используют на парных изображениях с целью сравнения распределений.
- График плотности — это сглаженная версия гистограммы; для оценивания графика плотности на основе данных требуется специальная функция (разумеется, возможны многочисленные оценки).

Дополнительные материалы для чтения

- ◆ Преподаватель университета Освего, штат Нью Йорк, предлагает пошаговое руководство по созданию коробчатой диаграммы¹⁵.
- ◆ Оценка плотности в R рассматривается в работе Генри Денга (Henry Deng) и Хэдли Уикхэма (Hadley Wickham) под тем же названием¹⁶.
- ◆ На R-bloggers есть полезный пост по поводу гистограмм в R¹⁷, включая такие элементы настройки, как разложение на корзины (разбиение на интервалы).
- ◆ На R-bloggers также имеются аналогичные посты в отношении коробчатых диаграмм в R¹⁸.
- ◆ Мэтью Конлен (Matthew Conlen) опубликовал интерактивную презентацию¹⁹, демонстрирующую влияние выбора различных ядер и пропускной способности при ядерном оценивании плотности.

Разведывание двоичных и категориальных данных

В случае категориальных данных представление о них дают простые доли или процентные соотношения.

¹⁵ См. <https://oreil.ly/wTpnE>.

¹⁶ См. <https://oreil.ly/TbWYS>.

¹⁷ См. <https://oreil.ly/Ynp-n>.

¹⁸ См. <https://oreil.ly/0DSb2>.

¹⁹ См. <https://oreil.ly/bC9nu>.

Ключевые термины для разведывания двоичных и категориальных данных

Мода (mode)

Наиболее часто встречающаяся категория или значение в наборе данных.

Ожидаемое значение (expected value)

Когда категории могут быть ассоциированы с числовыми значениями, этот показатель дает среднее значение, основываясь на вероятности появления категории.

Синоним: математическое ожидание.

Столбиковые диаграммы (bar charts)

Частота или доля каждой категории, изображаемая в виде столбиков.

Круговые диаграммы (pie charts)

Частота или доля каждой категории, изображаемая в виде сектора круга.

Получение сводной информации о двоичной либо категориальной переменной с несколькими категориями представляет собой довольно простую задачу: мы просто выясняем долю единиц либо доли важных категорий. Например, в табл. 1.6 показан процент задержанных рейсов из-за проблем в аэропорту Даллас/Форт-Уэрт, начиная с 2010 года. Задержки распределены по категориям в силу факторов, связанных с перевозчиком (Carrier), системными задержками при управлении воздушным движением (ATC), погодных условий (Weather), соображений безопасности (Security) или опоздания прибывающего воздушного судна (Inbound).

Таблица 1.6. Процент задержек в аэропорту Даллас/Форт-Уэрт

Carrier	ATC	Weather	Security	Inbound
23,02	30,40	4,03	0,12	42,43

Столбиковые диаграммы — это общепринятый визуальный инструмент для отображения одной-единственной категориальной переменной, который можно часто встретить в популярной прессе. Категории перечислены на оси *x*, а частоты или доли — на оси *y*. На рис. 1.5 показаны годовые задержки авиарейсов из-за проблем в аэропорту Даллас/Форт-Уэрт; график создан при помощи функции `barplot` языка R:

```
barplot(as.matrix(dfw) / 6, cex.axis=0.8, cex.names=0.7,  
        xlab='Причина задержки', ylab='Количество')
```

Пакет `pandas` тоже поддерживает столбиковые диаграммы для кадров данных:

```
ax = dfw.transpose().plot.bar(figsize=(4, 4), legend=False)  
ax.set_xlabel('Причина задержки')  
ax.set_ylabel('Количество')
```

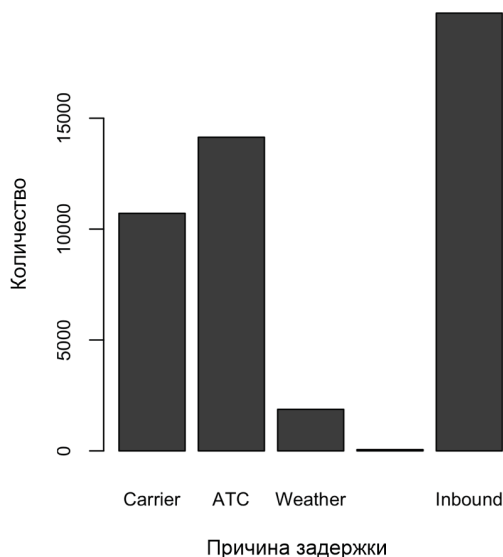


Рис. 1.5. Столбиковая диаграмма задержек авиарейсов из-за проблем в аэропорту Даллас/Форт-Уэрт

Отметим, что столбиковая диаграмма напоминает гистограмму; на столбиковой диаграмме ось x представляет разные категории факторной переменной, тогда как на гистограмме ось x представляет значения одной-единственной переменной на числовой шкале. На гистограмме столбики обычно изображаются вплотную друг к другу, а разрывы указывают на то, что значения в данных отсутствовали. На столбиковой диаграмме столбики отображаются отдельно друг от друга.

Круговые диаграммы являются альтернативой столбиковым диаграммам, хотя статистики и эксперты по визуализации данных обычно сторонятся круговых диаграмм, как менее визуально информативных (см. [Few-2007]).



Числовые данные как категориальные данные

В разделе "Частотные таблицы и гистограммы" ранее в этой главе мы рассмотрели частотные таблицы на основе разбивки данных на корзины, в результате чего числовые данные неявно конвертировались в упорядоченный фактор. В этом смысле гистограммы и столбиковые диаграммы подобны, за одним исключением — категории на оси x в столбиковой диаграмме не упорядочены. Конвертирование числовых данных в категориальные является важным и широко используемым этапом в анализе данных, поскольку эта процедура уменьшает сложность (и размер) данных. Она помогает обнаруживать связи между признаками, в особенности на начальных стадиях анализа.

Мода

Мода — это значение (или значения в случае, если они являются одинаковыми), которое появляется в данных чаще других. Например, модой задержек из-за проблем в аэропорту Даллас/Форт-Уэрт является фактор "опоздание прибывающего

воздушного судна" (Inbound). Вот еще один пример: в большинстве уголков США модой религиозных предпочтений будет христианство. Мода представляет собой простую сводную статистику для категориальных данных и обычно для числовых данных не используется.

Ожидаемое значение

Особым типом категориальных данных являются данные, в которых категории представлены или могут быть увязаны с дискретными значениями на одинаковой шкале измерения. Маркетолог новой "облачной" технологии, например, продает два уровня веб-служб, один по цене 300 \$ в месяц, а другой по цене 50 \$ в месяц. При этом он предлагает бесплатные вебинары для формирования списка потенциальных клиентов, и фирма полагает, что 5% посетителей подпишутся на веб-службы за 300 \$, 15% на веб-службы за 50 \$, и 80% не подпишутся совсем. Эти данные можно обобщить для проведения финансовых расчетов в одном "ожидаемом значении", являющемся формой среднего взвешенного, в котором весами выступают вероятности.

Ожидаемое значение, или математическое ожидание, вычисляется следующим образом:

1. Умножить каждый исход на вероятность его наступления.
2. Просуммировать эти значения.

В примере "облачной" службы ожидаемое значение посетителя вебинара, таким образом, составит 22,50 \$ в месяц, которое рассчитывается следующим образом:

$$EV = 0,05 \cdot 300 + 0,15 \cdot 50 + 0,80 \cdot 0 = 22,5.$$

Ожидаемое значение, по сути дела, является формой среднего взвешенного: оно привносит понятие будущих ожиданий и вероятностных весов, зачастую основываясь на субъективном суждении. Ожидаемое значение (математическое ожидание) является фундаментальным понятием в оценивании рыночной стоимости бизнеса и составлении бюджета долгосрочных расходов — например, ожидаемое значение для пятилетних прибылей от нового приобретения или ожидаемое сокращение затрат от новой вычислительной системы учета пациентов в клинике.

Вероятность

Мы уже упоминали выше о вероятности того, что некое значение произойдет. Большинство людей имеют интуитивное понимание вероятности, часто сталкиваясь с этим понятием в прогнозах погоды (вероятность дождя) или спортивном анализе (вероятность победы). Спорт и игры чаще всего выражаются как шансы, или перевесы, которые легко конвертируются в вероятности (если шансы на победу команды перевешивают и равны 2 к 1, то ее вероятность выигрыша равна $2 / (2 + 1) = 2 / 3$). Удивительно, однако, что понятие вероятности может быть источником глубоких философских дискуссий, когда речь заходит о его определении. К счастью, мы не нуждаемся здесь в формальном математическом или философ-

ском определении. Для наших целей вероятность того, что какое-то событие произойдет, есть доля того, сколько раз оно произойдет, если ситуация будет повторяться снова и снова, бесчисленное число раз. Чаще всего такое понимание вероятности представляет собой мнимую конструкцию, но зато оно является адекватным операционным пониманием.

**Ключевые идеи
для разведывания двоичных и категориальных данных**

- Категориальные данные, как правило, обобщаются в долях, и их можно визуализировать на столбиковой диаграмме.
- Категории могут представлять четко различимые объекты (яблоки и апельсины, мужчин и женщин), уровни факторной переменной (низкий, средний и высокий) либо числовые данные, которые были разбиты на корзины, или частотные интервалы.
- Ожидаемое значение, или математическое ожидание — это сумма значений, умноженных на вероятность их возникновения, которая часто используется для обобщения уровней факторных переменных.

Дополнительные материалы для чтения

Ни один курс статистики не будет полным без урока по дезориентирующим графикам, под которыми часто подразумеваются столбиковые и круговые диаграммы²⁰.

Корреляция

Разведывательный анализ данных во многих проектах моделирования (будь то в науке о данных либо в статистическом исследовании) предусматривает обследование корреляции среди предсказателей и между предсказателями и целевой переменной. Принято говорить, что переменные X и Y (каждая с измеряемыми данными) коррелируют положительно, если высокие значения X сопровождаются высокими значениями Y и низкие значения X сопровождаются низкими значениями Y . Если высокие значения X сопровождаются низкими значениями Y , и наоборот, то переменные коррелируют отрицательно.

Ключевые термины для корреляции

Коэффициент корреляции (correlation coefficient)

Метрика, которая измеряет степень, с которой числовые переменные связаны друг с другом (в диапазоне от -1 до $+1$).

²⁰ См. <https://oreil.ly/rDMuT>.

Корреляционная матрица (correlation matrix)

Таблица, в которой строки и столбцы — это переменные, а значения ячеек — корреляции между этими переменными.

Диаграмма рассеяния (scatterplot)

График, в котором по оси x откладываются значения одной переменной, а по оси y — значения другой.

Рассмотрим эти две переменные, идеально коррелированные в том смысле, что каждая движется параллельно от низкого значения до высокого:

v1: {1, 2, 3}

v2: {4, 5, 6}

Векторная сумма произведений равна $1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 32$. Теперь попробуем перетасовать одну из них и вычислить повторно — векторная сумма произведений никогда не будет больше 32. Поэтому данная сумма произведений может использоваться в качестве метрики; т. е. наблюдаемую сумму, равную 32, можно сравнивать со многочисленными случайными перетасовками (по сути дела, эта идея касается оценки на основе повторного отбора: см. раздел "Перестановочный тест" главы 3). Производимые этой метрикой значения тем не менее не особо содержательны, кроме как со ссылкой на распределение повторных выборок.

Более полезный стандартизированный вариант — это *коэффициент корреляции*, дающий оценку корреляции между двумя переменными, которые всегда находятся на одинаковой шкале измерения. Для того чтобы вычислить *коэффициент корреляции Пирсона*, мы умножаем отклонения от среднего для переменной 1 на отклонения для переменной 2, а затем делим результат на произведение стандартных отклонений:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}.$$

Обратите внимание, что мы делим на $n-1$, а не на n (см. врезку "Степени свободы и n или $n-1$?" ранее в этой главе). Коэффициент корреляции всегда находится между $+1$ (идеальная положительная корреляция) и -1 (идеальная отрицательная корреляция); 0 свидетельствует об отсутствии корреляции.

Переменные могут иметь нелинейную связь, и в этом случае коэффициент корреляции может не быть полезной метрикой. Связь между налоговыми ставками и поступлениями, полученными за счет налогов, служит примером: когда налоговые ставки увеличиваются от 0 , налоговые поступления тоже увеличиваются. Однако, как только налоговые ставки достигают высокого уровня и приближаются к 100% , уклонение от уплаты налогов увеличивается, и налоговые поступления в итоге уменьшаются.

В табл. 1.7, которая называется *корреляционной матрицей*, показана корреляция между ежедневными доходностями акций телекоммуникационных компаний

с июля 2012 по июнь 2015 года. Из таблицы видно, что Verizon (VZ) и ATT (T) имеют самую высокую корреляцию. Инфраструктурная компания Level Three (LVLТ) имеет самую низкую корреляцию. Обратите внимание на диагональ из единиц (корреляция акции с самой собой равна 1) и избыточность информации выше и ниже диагонали.

Таблица 1.7. Корреляция между ежедневными финансовыми возвратами акций телекоммуникационных компаний

	T	CTL	FTR	VZ	LVLТ
T	1,000	0,475	0,328	0,678	0,279
CTL	0,475	1,000	0,420	0,417	0,287
FTR	0,328	0,420	1,000	0,287	0,260
VZ	0,678	0,417	0,287	1,000	0,242
LVLТ	0,279	0,287	0,260	0,242	1,000

Таблица корреляций, аналогичная табл. 1.7, широко используется для визуального отображения связи между многочисленными переменными. На рис. 1.6 показана корреляция между ежедневными возвратностями крупнейших биржевых инвестиционных фондов (ETF-фондов). В R мы можем легко создать ее с помощью пакета `corrplot`:

```
etfs <- sp500_px[row.names(sp500_px)>"2012-07-01",
               sp500_sym[sp500_sym$sector=="etf", 'symbol']]
library(corrplot)
corrplot(cor(etfs), method = "ellipse")
```

Тот же самый график можно создать в Python, но в часто используемых пакетах его имплементация отсутствует. Однако большинство из них поддерживают визуализацию матриц корреляций с помощью тепловых карт. Следующий ниже код демонстрирует это с помощью функции `heatmap` пакета `seaborn`. В прилагаемый к книге репозиторий исходного кода мы включили код Python для генерирования более всесторонней визуализации:

```
etfs = sp500_px.loc[sp500_px.index > '2012-07-01',
                  sp500_sym[sp500_sym['sector'] == 'etf']['symbol']]
sns.heatmap(etfs.corr(), vmin=-1, vmax=1,
           cmap=sns.diverging_palette(20, 220, as_cmap=True))
```

Биржевые инвестиционные фонды для индексов S&P 500 (SPY) и Доу Джонса (Dow Jones, DIA) имеют высокую корреляцию. Схожим образом, фонды QQQ и XLK, состоящие главным образом из технологических компаний, коррелированы положительно. Защитные биржевые инвестиционные фонды, которые отслеживают цены на золото (GLD), цены на нефть (USO) или волатильность рынка (VXX), имеют тенденцию отрицательно коррелироваться с другими ETF. Ориентация эллипса говорит о том, коррелируют ли две переменные положительно (эллипс по-

вернут вправо) либо отрицательно (эллипс повернут влево). Заливка и ширина эллипса свидетельствуют о силе связи: более тонкие и темные эллипсы соответствуют более сильным связям.

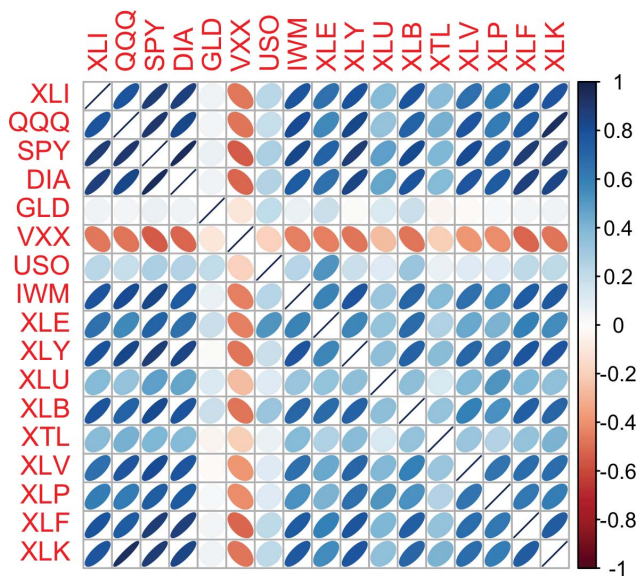


Рис. 1.6. Корреляция между финансовыми возвратами биржевых инвестиционных фондов (ETF)

Подобно среднему значению и стандартному отклонению, коэффициент корреляции чувствителен к выбросам в данных. В программных пакетах предлагаются робастные альтернативы классическому коэффициенту корреляции. Например, пакет `robust`²¹ языка R использует функцию `covRob` для вычисления робастной оценки корреляции. Методы в модуле `sklearn.covariance`²² пакета `scikit-learn` имплементируют целый ряд подходов.



Другие оценки корреляции

В статистике давно были предложены другие типы коэффициентов корреляции, такие как коэффициент ранговой корреляции *Спирмена* ρ (ро) или коэффициент ранговой корреляции *Кенделла* τ (тау). Эти коэффициенты корреляции основаны на ранге данных, т. е. номерах наблюдений в наборе. Поскольку они работают с рангами, а не значениями, эти оценки робастны к выбросам и могут справляться с некоторыми типами нелинейности. Однако исследователи данных в целях разведывательного анализа обычно могут придерживаться коэффициента корреляции Пирсона и его робастных альтернатив. Ранговые оценки привлекательны главным образом в случае малых наборов данных и специфических проверок гипотез.

²¹ См. <https://oreil.ly/isORz>.

²² См. <https://oreil.ly/su7wi>.

Диаграммы рассеяния

Стандартным средством визуализации связи между двумя переменными с измеряемыми данными является *диаграмма рассеяния*, чья ось x представляет одну переменную, ось y — другую, а каждая точка на графике является записью. Посмотрим на рис. 1.7 с графиком, содержащим ежедневные финансовые возвраты акций АТТ и Verizon. Указанный график создан в R следующей ниже командой:

```
plot(telecom$T, telecom$VZ, xlab="T", ylab="VZ")
```

Тот же самый график можно сгенерировать в Python с помощью метода `scatter` пакета `pandas`:

```
ax = telecom.plot.scatter(x='T', y='VZ', figsize=(4, 4), marker='$\u25EF$')
ax.set_xlabel('ATT (T)')
ax.set_ylabel('Verizon (VZ)')
ax.axhline(0, color='grey', lw=1)
ax.axvline(0, color='grey', lw=1)
```

Финансовые возвраты имеют сильную положительную связь: в большинстве торговых дней обе акции движутся вверх или вниз в tandem (правый верхний и левый нижний квадранты). Существует меньшее число дней, когда цена на одну акцию значительно падает, в то время как цена на другую акцию растет, и наоборот (правый нижний и левый верхний квадранты).

В то время как на рис. 1.7 показаны только 754 точки данных, уже очевидно, насколько сложно идентифицировать детали в центре графика. Позже мы увидим, как добавление прозрачности к точкам или использование сетки из шестиугольных корзин и графиков плотности помогает отыскивать дополнительную структуру в данных.

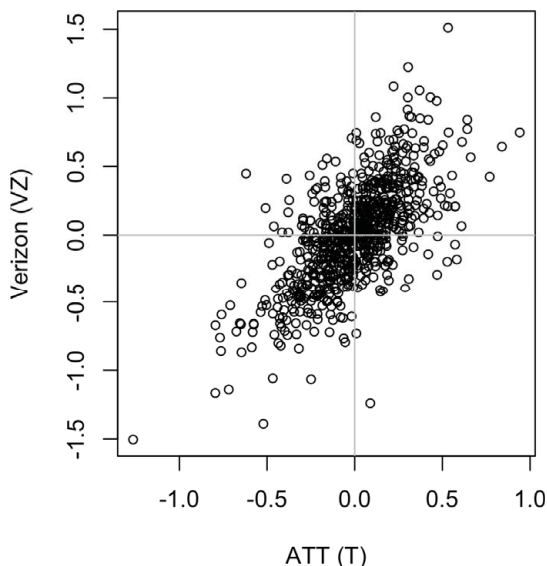


Рис. 1.7. Диаграмма рассеяния между финансовыми возвратами акций АТТ и Verizon

Ключевые идеи для корреляции

- Коэффициент корреляции измеряет степень, с которой две переменные связаны между собой.
- Когда высокие значения v_1 сопровождаются высокими значениями v_2 , v_1 и v_2 связаны положительно.
- Когда высокие значения v_1 связаны с низкими значениями v_2 , v_1 и v_2 связаны отрицательно.
- Коэффициент корреляции — это стандартизированная метрика, которая всегда варьирует в интервале от -1 (идеальная отрицательная корреляция) до $+1$ (идеальная положительная корреляция).
- Коэффициент корреляции 0 говорит об отсутствии корреляции, но следует учитывать, что случайные перестановки данных будут порождать как положительные, так и отрицательные значения коэффициента корреляции по чистой случайности.

Дополнительные материалы для чтения

В книге "Статистика" Дэвида Фридмана, Роберта Пизани и Роджера Парвеса (Freedman D., Pisani R., Purves R. Statistics. — 4th edition. — W. W. Norton, 2007) приведено превосходное обсуждение корреляции.

Разведывание двух или более переменных

Уже знакомые нам оценщики, такие как среднее значение и дисперсия, рассматривают по одной переменной за раз (*одномерный анализ*). Корреляционный анализ (см. раздел "Корреляция" ранее в этой главе) — это важная методика, которая сравнивает две переменные (*двумерный анализ*). В этом разделе мы обратимся к дополнительным оценкам, графикам и более чем к двум переменным (*многомерному анализу*).

Ключевые термины для разведывания двух или более переменных

Таблица сопряженности (contingency table)

Сводка количеств двух или более категориальных переменных.

Сетка из шестиугольных корзин (hexagonal binning)

График двух числовых переменных, в котором записи сгруппированы в шестиугольниках.

Контурные графики (contour plots)

График, показывающий плотность двух числовых переменных в форме топографической карты.

Скрипичные графики (violin plots)

График, аналогичный коробчатой диаграмме, но показывающий оценку плотности.

Двумерный анализ подобно одномерному анализу связан с вычислением сводных статистик и их визуализацией. Соответствующий тип двумерного или многомерного анализа зависит от природы данных: числовых относительно категориальных.

Сетка из шестиугольных корзин и контуры (сопоставление числовых данных с числовыми данными на графике)

Диаграммы рассеяния прекрасны, когда имеется относительно малое число значений данных. График возвратности акций на рис. 1.7 содержит всего около 750 точек. Для наборов данных с сотнями тысяч и миллионов записей диаграмма рассеяния будет слишком плотной, и поэтому требуется другой способ визуализации связи. В качестве иллюстрации рассмотрим набор данных `kc_tax`, который содержит налогооблагаемую стоимость жилой недвижимости в округе Кинг, штат Вашингтон. Для того чтобы сосредоточиться на главной части данных, мы исключим очень дорогое жилье, а также жилье очень малых или очень больших размеров при помощи функции `subset`:

```
kc_tax0 <- subset(kc_tax, TaxAssessedValue < 750000 & SqFtTotLiving > 100 &
                  SqFtTotLiving < 3500)
nrow(kc_tax0)
[1] 432733
```

В пакете `pandas` мы фильтруем набор данных следующим образом:

```
kc_tax0 = kc_tax.loc[(kc_tax.TaxAssessedValue < 750000) &
                    (kc_tax.SqFtTotLiving > 100) &
                    (kc_tax.SqFtTotLiving < 3500), :]
kc_tax0.shape
(432693, 3)
```

На рис. 1.8 показан *график с сеткой из шестиугольных корзин* для отображения связи между общей площадью в квадратных футах²³ и налогооблагаемой стоимостью жилья в округе Кинг. Вместо того чтобы выводить на график точки, которые будут появляться как монолитное темное облако, мы сгруппировали записи в шестиугольные корзины и вывели шестиугольники в цвете, указывающем на число записей в конкретной корзине. На этом графике положительная связь между квадрат-

²³ Общая площадь жилой недвижимости (finished square footage) — в американском налогообложении и среди торговцев недвижимостью за исключением межпотолочных и межстенных пространств и сводчатых перекрытий это практически вся площадь жилья, находящаяся выше и ниже уровня земли. — *Прим. перев.*

ными футами и налогооблагаемой стоимостью жилой недвижимости очевидна. Интересной особенностью указанного графика является тень второго облака над главным облаком, указывающая на дома, которые имеют одинаковую площадь с теми, что расположены в главном облаке, но с более высокой налогооблагаемой стоимостью.

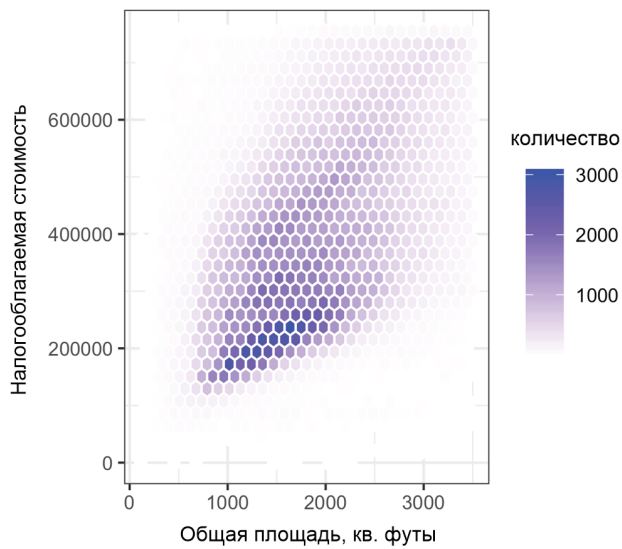


Рис. 1.8. График с сеткой из шестиугольных корзин для налогооблагаемой стоимости против общей площади в квадратных футах

Рис. 1.8 был сгенерирован мощным пакетом `ggplot2` языка R, разработанным Хэдли Уикхэмом (Hadley Wickham) [`ggplot2`]. Пакет `ggplot2` — один из нескольких, предназначенных для продвинутого разведывательного визуального анализа данных (см. раздел *"Визуализация многочисленных переменных"* далее в этой главе).

```
ggplot(kc_tax0, (aes(x=SqFtTotLiving, y=TaxAssessedValue))) +
  stat_binhex(colour="white") +
  theme_bw() +
  scale_fill_gradient(low="white", high="black") +
  labs(x="Общая площадь, кв. футы", y="Налогооблагаемая стоимость")
```

Графики с сеткой из шестиугольных корзин всегда доступны в Python с помощью метода `hexbin` пакета `pandas`:

```
ax = kc_tax0.plot.hexbin(x='SqFtTotLiving', y='TaxAssessedValue',
                        gridsize=30, sharex=False, figsize=(5, 4))
ax.set_xlabel('Общая площадь, кв. футы')
ax.set_ylabel('Налогооблагаемая стоимость')
```

На рис. 1.9 используются контуры, которые наложены на диаграмму рассеяния, с целью визуализации связи между двумя числовыми переменными. Контуры представляют собой, по существу, топографическую карту, соответствующую двум

переменным; каждая полоса контура представляет соответствующую плотность точек, увеличиваясь по мере того, как она приближается к "пику". Этот график говорит о том же, что и график на рис. 1.8: имеется вторичный пик "к северу" от главного пика. Этот график тоже был построен при помощи пакета `ggplot2` и встроенной функции `geom_density2d`.

```
ggplot(kc_tax0, aes(SqFtTotLiving, TaxAssessedValue)) +  
  theme_bw() +  
  geom_point(alpha=0.1) +  
  geom_density2d(colour="white") +  
  labs(x="Общая площадь, кв. футы", y="Налогооблагаемая стоимость")
```

Функция `kdeplot` пакета `seaborn` в Python создает контурный график:

```
ax = sns.kdeplot(kc_tax0.SqFtTotLiving, kc_tax0.TaxAssessedValue, ax=ax)  
ax.set_xlabel('Общая площадь, кв. футы')  
ax.set_ylabel('Налогооблагаемая стоимость')
```

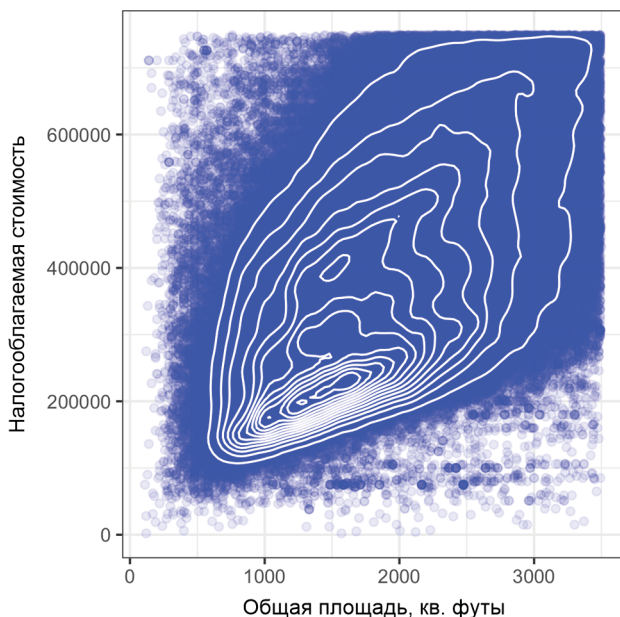


Рис. 1.9. Контурный график для налогооблагаемой стоимости против общей площади в квадратных футах

Другие типы диаграмм используются для изображения связи между двумя числовыми переменными, в том числе *тепловые карты*. Тепловые карты, графики с сеткой из шестиугольных корзин и контурные графики — все они дают визуальное представление о двумерной плотности. В этом смысле они являются естественными аналогами гистограмм и графиков плотности.

Две категориальные переменные

Полезным инструментом обобщения двух категориальных переменных является таблица сопряженности — таблица количеств по категориям. Табл. 1.8 является таблицей сопряженности между уровнями персональной ссуды и ее исходом. Таблица построена на основе данных, предоставленных кредитным клубом Lending Club, лидером в инвестиционно-кредитном бизнесе равноправного кредитования. Уровень может быть от А (верхний) до G (низкий). Исходом может быть один из следующих: погашена, активная, просрочена или списана (остаток ссуды не будет взыскан). Эта таблица показывает количества и строчные проценты. Ссуды высокого уровня имеют очень низкий процент просрочки/списания по сравнению с ссудами более низкого уровня.

Таблица 1.8. Таблица сопряженности уровня ссуд и их состояния

Уровень	Полностью погашена	Активна	Просрочена	Списана	Всего
A	20715	52058	494	1588	74855
	0,277	0,695	0,007	0,021	0,161
B	31782	97601	2149	5384	136916
	0,232	0,713	0,016	0,039	0,294
C	23773	92444	2895	6163	125275
	0,190	0,738	0,023	0,049	0,269
D	14036	55287	2421	5131	76875
	0,183	0,719	0,031	0,067	0,165
E	6089	25344	1421	2898	35752
	0,170	0,709	0,040	0,081	0,077
F	2376	8675	621	1556	13228
	0,180	0,656	0,047	0,118	0,028
G	655	2042	206	419	3322
	0,197	0,615	0,062	0,126	0,007
Итого	99426	333451	10207	23139	466223

Таблицы сопряженности могут содержать лишь количества либо также включать столбцовые и итоговые проценты. Сводные таблицы в Excel являются, возможно, наиболее часто встречающимся инструментом, используемым для создания таблиц сопряженности. В R функция `CrossTable` из пакета `descr` создает таблицы сопряженности, и следующий ниже фрагмент кода использовался для создания табл. 1.8:

```
library(descr)
x_tab <- CrossTable(lc_loans$grade, lc_loans$status,
                    prop.c=FALSE, prop.chisq=FALSE, prop.t=FALSE)
```

Метод `pivot_table` создает сводную таблицу в Python. Аргумент `aggfunc` позволяет нам получить количества. Вычисление процентов немного запутаннее:

```
crosstab = lc_loans.pivot_table(index='grade', columns='status',
                                aggfunc=lambda x: len(x), margins=True) ❶
df = crosstab.loc['A':'G',:].copy() ❷
df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged Off':'Late'].div(df['All'],
                                                                    axis=0) ❸
df['All'] = df['All'] / sum(df['All']) ❹
perc_crosstab = df
```

- ❶ Именованный аргумент `margins` добавит столбцовые и строчные суммы.
- ❷ Мы создаем копию сводной таблицы, игнорируя столбцовые суммы.
- ❸ Мы делим строки на строчную сумму.
- ❹ Мы делим столбец 'All' на его сумму.

Категориальные и числовые данные

Коробчатые диаграммы (см. раздел *"Процентили и коробчатые диаграммы"* ранее в этой главе) являются простым способом визуального сравнения распределений числовой переменной, сгруппированной согласно категориальной переменной. Например, нам нужно сравнить, каким образом процент задержек авиарейсов варьируется среди авиакомпаний. На рис. 1.10 отображен процент задержанных авиарейсов за месяц, где задержка была вызвана перевозчиком.

```
boxplot(pct_delay ~ airline, data=airline_stats, ylim=c(0, 50))
```

Метод `boxplot` пакета `pandas` берет аргумент, который разбивает набор данных на группы, и создает индивидуальные коробчатые диаграммы:

```
ax = airline_stats.boxplot(by='Авиакомпания', column='pct_carrier_delay')
ax.set_xlabel('')
ax.set_ylabel('Доля ежедневных задержанных рейсов, %')
plt.suptitle('')
```

Авиакомпания Alaska Airlines выделяется тем, что имеет наименьшее число задержек авиарейсов, тогда как American Airlines имеет подавляющее число задержек: нижний квартиль для American Airlines выше верхнего квартиля для Alaska Airlines.

Скрипичный график, введенный в употребление [Hintze-Nelson-1998], представляет собой дополнение к коробчатой диаграмме и графически изображает оценки плотности, где плотности расположены на оси *y*. Плотность зеркально отражена и перевернута, и получившаяся фигура заполнена, создавая изображение, напоминающее скрипку. Преимущество скрипичного графика состоит в том, что он может показывать нюансы в распределении, которые не заметны на коробчатой диаграмме. С другой стороны, коробчатая диаграмма яснее показывает выбросы в данных. В пакете `ggplot2` функция `geom_violin` используется для создания скрипичного графика следующим образом:

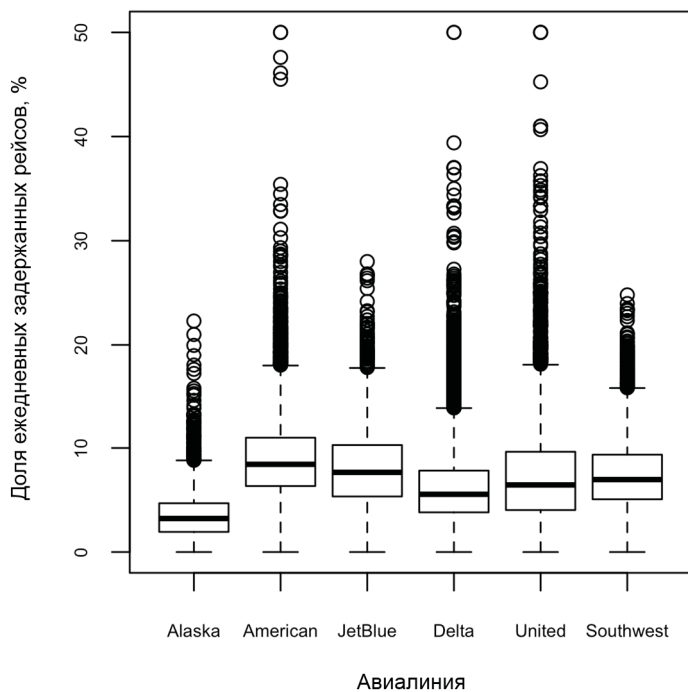


Рис. 1.10. Коробчатая диаграмма процента задержек авиарейсов по перевозчикам

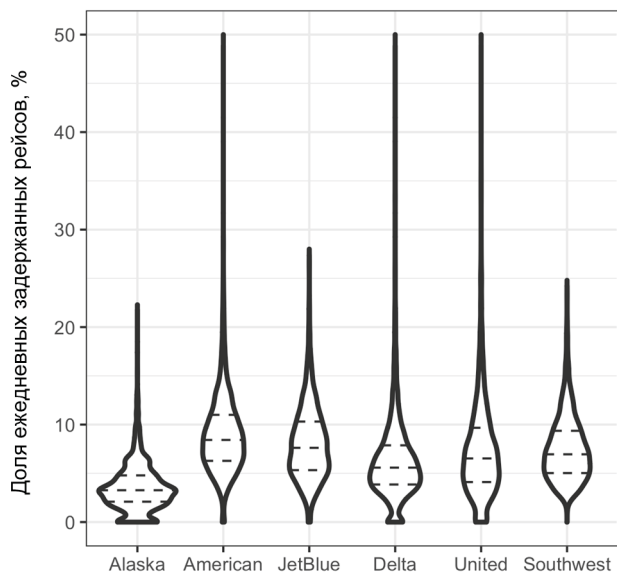


Рис. 1.11. Сочетание коробчатой диаграммы и скрипичного графика, показывающее процент задержек авиарейсов по перевозчикам

```
ggplot(data=airline_stats, aes(airline, pct_carrier_delay)) +
  ylim(0, 50) +
  geom_violin() +
  labs(x="", y="Доля ежедневных задержанных рейсов, %")
```

Скрипичные графики можно построить с помощью метода `violinplot` пакета `seaborn`:

```
ax = sns.violinplot(airline_stats.airline, airline_stats.pct_carrier_delay,
                    inner='quartile', color='white')
ax.set_xlabel('')
ax.set_ylabel('Доля ежедневных задержанных рейсов, %')
```

Соответствующий график показан на рис. 1.11. Скрипичный график показывает концентрацию в распределении около нуля для Alaska Airlines и в меньшей степени для Delta Airlines. Этот феномен не так очевиден на коробчатой диаграмме. Скрипичный график можно объединить с коробчатой диаграммой путем добавления `geom_boxplot` в график (правда, в этом случае лучше всего использовать цвета).

Визуализация многочисленных переменных

Все виды графиков, которые использовались ранее для сравнения двух переменных — диаграммы рассеяния, графики с сеткой из шестиугольных корзин и коробчатые диаграммы, — легко расширяемы на большее число переменных через понятие *кондиционности*. В качестве примера вернемся назад к рис. 1.8, в котором была показана связь между общей площадью домов и налогооблагаемой стоимостью. Мы отметили, что, по всей видимости, существует кластер домов, которые имеют более высокую налогооблагаемую стоимость в расчете на квадратный фут. Давайте копнем поглубже и рассмотрим рис. 1.12, который объясняет эффект местоположения домов путем отображения данных набора почтовых индексов. Теперь картина становится намного яснее: в некоторых почтовых индексах (98112, 98105) налогооблагаемая стоимость намного выше, чем в других (98108, 98057). Именно эта несоразмерность дает начало кластерам, наблюдаемым на рис. 1.8.

Мы создали рис. 1.12, используя пакет `ggplot2` и идею *фасетов (граней)*, или *кондиционных переменных*²⁴ (в данном случае почтовых индексов):

```
ggplot(subset(kc_tax0, ZipCode %in% c(98188, 98105, 98108, 98126)),
       aes(x=SqFtTotLiving, y=TaxAssessedValue)) +
  stat_binhex(colour="white") +
  theme_bw() +
  scale_fill_gradient(low="white", high="blue") +
  labs(x="Общая площадь, кв. футы", y="Налогооблагаемая стоимость") +
  facet_wrap("ZipCode") ❶
```

²⁴ Кондиционные переменные (conditioning variables) — это фоновые переменные, характеризующие объект исследования, которые используются для построения правдоподобных значений путем приведения недостающих данных. — *Прим. перев.*

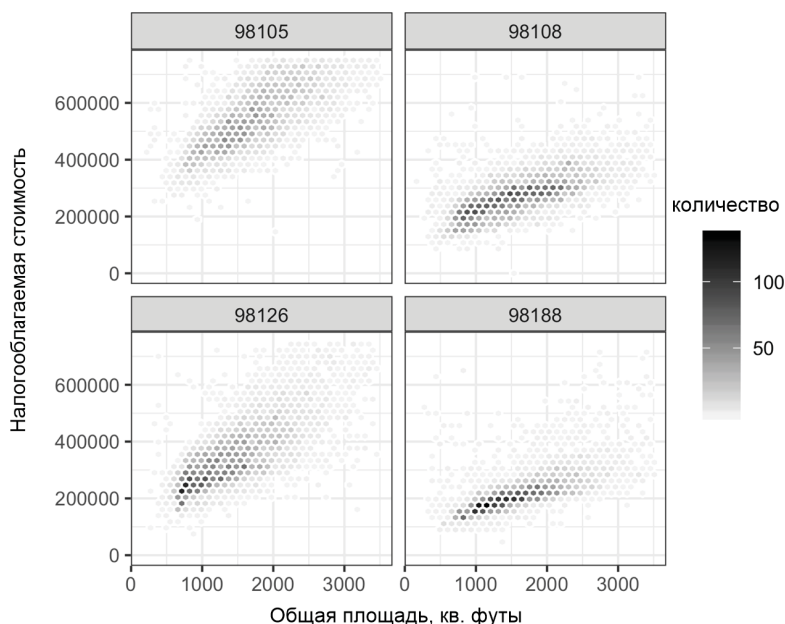


Рис. 1.12. Налогооблагаемая стоимость против общей площади в квадратных футах по почтовому индексу

❶ Используйте функции `facet_wrap` и `facet_grid` пакета `ggplot` для указания конди-
ционной переменной.

Большинство пакетов Python основывают свои визуализации на библиотеке `matplotlib`. Хотя в принципе можно создавать фасетные графики с помощью `matplotlib`, код может стать сложным для написания и восприятия. К счастью, в па-
кете `seaborn` есть относительно простой способ создания таких графиков:

```
zip_codes = [98188, 98105, 98108, 98126]
kc_tax_zip = kc_tax0.loc[kc_tax0.ZipCode.isin(zip_codes),:]
kc_tax_zip
```

```
def hexbin(x, y, color, **kwargs):
    cmap = sns.light_palette(color, as_cmap=True)
    plt.hexbin(x, y, gridsize=25, cmap=cmap, **kwargs)
```

```
g = sns.FacetGrid(kc_tax_zip, col='ZipCode', col_wrap=2) ❶
g.map(hexbin, 'SqFtTotLiving', 'TaxAssessedValue',
      extent=[0, 3500, 0, 700000]) ❷
g.set_axis_labels('Общая площадь, кв. футы', 'Налогооблагаемая стоимость')
g.set_titles('Zip code {col_name:0f}')
```

❶ Используйте аргументы `col` и `row` для указания кондиционных переменных. Для
одной кондиционной переменной используйте аргумент `col` вместе с `col_wrap`, что-
бы обернуть фасетные графики в несколько строк.

② Метод `map` вызывает функцию `hexbin` с подмножествами исходного набора данных для разных почтовых индексов. Аргумент `extent` задает пределы осей `x` и `y`.

Понятие кондиционных переменных в графической системе было впервые применено в Trellis graphics, методологии для отображения наборов сложных многомерных данных, которая получила развитие благодаря Рикку Беккеру (Rick Becker), Биллу Кливленду (Bill Cleveland) и другим в Bell Labs [Trellis-Graphics]. Эта идея распространилась на различные современные графические системы, такие как пакеты `lattice` [lattice] и `ggplot2` в R и модули `seaborn` [seaborn] и `Bokeh` [bokeh] в Python. Кондиционные переменные также являются неотъемлемой частью бизнес-аналитических платформ, таких как Tableau и Spotfire. С появлением обширных вычислительных возможностей современные платформы визуализации переместились далеко за пределы скромных начинаний разведывательного анализа данных. Однако ключевые понятия и инструменты, разработанные за эти годы, по-прежнему формируют фундамент для этих систем.

Ключевые идеи для разведывания двух или более переменных

- Графики с сеткой из шестиугольных корзин и контурные графики — это полезные инструменты, которые позволяют выполнять графический анализ сразу двух числовых переменных, не перегружаясь огромными объемами данных.
- Таблицы сопряженности — это стандартный инструмент для просмотра количеств двух категориальных переменных.
- Коробчатые диаграммы и скрипичные графики позволяют изображать числовую переменную в сопоставлении с категориальной.

Дополнительные материалы для чтения

- ◆ В книге Baumer B., Kaplan D., Horton. Modern Data Science with R. — CRC Press, 2017 ("Современная наука о данных вместе с R") представлено превосходное введение в "грамматику для графиков" ("`gg`" в названии пакета `ggplot`).
- ◆ Ggplot2: Elegant Graphics for Data Analysis. — Hadley Wickham, Springer, 2009 (Ggplot2: изящная графика для анализа данных) — это превосходный ресурс от создателя пакета `ggplot2`.
- ◆ Джозеф Фрювальд (Josef Fruehwald) предлагает онлайн-учебное руководство по пакету `ggplot2`²⁵.

²⁵ См. <https://oreil.ly/zB2Dz>.

Резюме

Разведывательный анализ данных (exploratory data analysis, EDA), впервые внедренный Джоном Тьюки, заложил фундамент науки о данных. Ключевая идея разведывательного анализа данных состоит в том, что первый и самый важный шаг в любом проекте, основанном на данных, состоит в том, чтобы *посмотреть на данные*. Благодаря обобщению и визуализации данных можно получить ценное интуитивное понимание проекта.

Данная глава была посвящена обзору понятий, начиная с простых метрик, таких как оценки центрального положения и вариабельности до насыщенного визуального изображения, с целью разведывания связей между многочисленными переменными (см. рис. 1.12). Разнообразный набор инструментов и технических приемов, создаваемых сообществом разработчиков вычислительных систем с открытым исходным кодом, в сочетании с выразительностью языков R и Python открыли массу способов разведывания и анализа данных. Разведывательный анализ должен быть краеугольным камнем любого проекта науки о данных.

Распределение данных и распределение выборок

Популярное мнение ошибочно утверждает, что эра больших данных сводит на нет потребность в выборке. На самом деле быстрое распространение данных переменного качества и релевантности укрепляет потребность в выборке как инструменте эффективной работы с разнообразными данными и минимизации смещения. Даже в проекте больших данных предсказательные модели, как правило, разрабатываются и тестируются при помощи выборок. Выборки также используются в самых разнообразных тестах (например, в сравнении эффекта дизайна веб-страниц на основе нажатий пользователя на тех или иных элементах).

На рис. 2.1 представлена схема, которая подкрепляет понятия из данной главы. С левой стороны представлена популяция, которая в статистике предположительно подчиняется базовому, но неизвестному распределению. Единственное, что имеется, — это выборка данных и ее эмпирическое распределение, показанное с правой стороны. Для того чтобы можно было "попасть" из левой стороны в правую, используется процедура *отбора* (представленная стрелкой). Традиционная статистика сосредоточена главным образом на левой стороне, используя теорию, основанную на серьезных допущениях о популяции. Современная статистика переместилась в правую сторону, где такие допущения не нужны.

В общем случае исследователям данных не нужно беспокоиться о теоретической природе левой стороны; вместо этого им следует уделять основное внимание про-

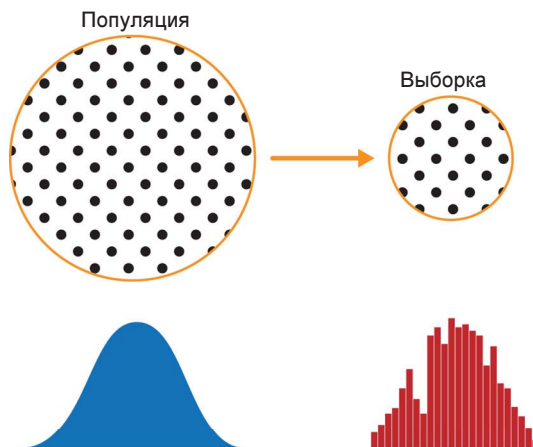


Рис. 2.1. Сравнение популяции с выборкой

цедурам отбора и располагаемым данным. Правда, существуют некоторые существенные исключения. Иногда данные генерируются из физического процесса, который может быть смоделирован. Самый простой пример — бросание монеты: оно подчиняется биномиальному распределению. Любая реальная биномиальная ситуация (купить или не купить, мошенничество или не мошенничество, нажать или не нажать) может быть эффективно смоделирована монетой (разумеется, с подстроенной вероятностью приземления орла). В этих случаях можно получить дополнительную информацию посредством использования нашего понимания популяции.

Случайный отбор и смещенная выборка

Выборка — это подмножество данных из более крупного набора данных; в статистике этот более крупный набор данных называется *популяцией*, или *генеральной совокупностью*. Популяция в статистике — это не то же самое, что популяция в биологии, это большой, определенный, но нередко теоретический или воображаемый, набор данных.

Случайный отбор — это процесс, в котором каждый доступный член популяции, подвергаемой отбору, имеет равную возможность попасть в выборку при каждом извлечении. Результирующая выборка называется *простой случайной выборкой*. Отбор может быть выполнен *с возвратом*, когда после каждого извлечения наблюдения кладутся назад в популяцию для возможно повторного отбора в будущем. Как альтернатива отбор может быть выполнен *без возврата*, и в этом случае однажды выбранные наблюдения недоступны для будущих извлечений.

Качество данных часто имеет большее значение, чем их количество, когда выполняется оценка либо создается модель на основе выборки. Качество данных в науке о данных предусматривает полноту, согласованность формата, чистоту и точность отдельных точек данных. Статистика добавляет сюда понятие *репрезентативности*.

Ключевые термины для случайного отбора

Выборка (sample)

Подмножество большего по размеру набора данных.

Популяция (population)

Более крупный набор данных или идея набора данных.

Синоним: генеральная совокупность.

N (n)

Размер популяции (выборки).

Случайный отбор (random sampling)

Извлечение элементов в выборку в случайном порядке.

Стратифицированный отбор (stratified sampling)

Разделение популяции на страты (подгруппы) и случайный отбор элементов из каждой страты.

Простая случайная выборка (simple random sample)

Выборка, получаемая в результате случайного отбора без разбиения популяции на страты.

Смещенная выборка (sample bias)

Выборка, которая представляет популяцию в искаженном виде.

Классический пример — опрос, выполненный в 1936 году журналом "Литературный дайджест" (Literary Digest), который предсказал победу Альфреда Лэндона (Al Landon) над Франклином Рузвельтом (Franklin Roosevelt). Периодическое ежедневное издание "Литературный дайджест" опросило подписчиков из своей базы данных плюс людей из дополнительных списков (в общей сложности более 10 млн человек) и предсказало сокрушительную победу Лэндона. Джордж Гэллуп (George Gallup), основатель института опроса общественного мнения, проводил опросы каждые две недели всего по 2 тыс. респондентов и точно предсказал победу Рузвельта. Разница заключалась в том, как выбирались респонденты.

Журнал "Литературный дайджест" сделал ставку на количество, мало обращая внимания на метод отбора. В итоге оказалось, что сотрудники журнала опросили людей с относительно высоким социально-экономическим статусом (их собственные подписчики плюс те, кто входили в списки маркетологов на основании владения предметами роскоши, такими как телефоны и автомобили). Результатом стала *смещенная выборка*, т. е. она отличалась некоторым содержательным неслучайным характером от остальной более многочисленной популяции, которую эта выборка должна была представлять. Термин "*неслучайный*" очень важен — едва ли любая выборка, включая случайные выборки, будет для популяции строго репрезентативной. Смещенная выборка возникает, когда разница становится содержательной, и ожидается, что она продолжится в отношении других выборок, извлекаемых таким же образом, что и первая.



Систематическая ошибка самоотбора

Отзывы о ресторанах, гостиницах, кафе и подобных заведениях, которые вы читаете в социальных сетях, таких как Yelp, подвержены смещению, потому что предлагающие их люди не отбираются в случайном порядке, а наоборот, они сами взяли на себя инициативу высказаться. Это приводит к так называемой систематической ошибке самоотбора — люди, мотивированные написать отзыв, могут иметь неудачный опыт, быть связаны с предприятием или же просто отличаться по складу от тех людей, которые отзывы не пишут. Отметим, что, хотя выборки на основе самоотбора могут быть ненадежными индикаторами истинного положения дел, они могут быть надежнее в простом сравнении одного предприятия с аналогичным ему; та же самая систематическая ошибка самоотбора может касаться каждого из них.

Смещение

Статистическое смещение относится к ошибкам измерения или отбора, которые систематичны и порождаются процессом измерения или отбора данных. Важно проводить различие между ошибками вследствие случайности и ошибками вследствие смещения. Рассмотрим физический процесс стрельбы из ружья по мишени. Поражение абсолютного центра мишени не будет происходить каждый раз или даже не будет происходить вообще. Несмещенный процесс произведет ошибку, но она будет случайной и не проявит тенденцию к любому из направлений (рис. 2.2). Приведенные на рис. 2.3 результаты показывают смещенный процесс — по-прежнему имеется случайная ошибка как в направлении x , так и в направлении y , но помимо этого имеется смещение. Выстрелы демонстрируют тенденцию попадать в правый верхний квадрант.

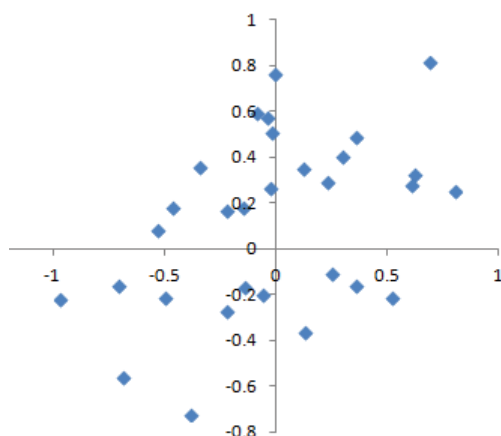


Рис. 2.2. Диаграмма рассеяния выстрелов из ружья с отрегулированным прицелом

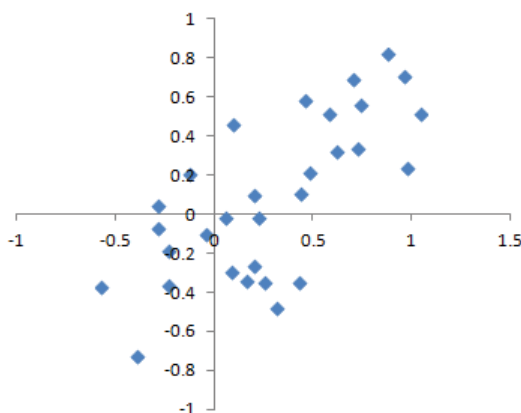


Рис. 2.3. Диаграмма рассеяния выстрелов из ружья со смещенным прицелом

Смещение возникает в разных формах и может быть заметным или невидимым. Когда результат действительно наводит на мысль о смещении (например, опираясь на эталон или фактические значения), это сигнализирует о том, что статистическая или автоматически обучающаяся модель была задана неправильно либо важная переменная была упущена из виду.

Случайный отбор

Во избежание проблемы смещенной выборки, которая привела журнал "Литературный дайджест" к предсказанию победы Лэндона над Рузвельтом, Джордж Гэллап (чья фотография приведена на рис. 2.4) сделал ставку на более отборные в научном плане методы достижения выборки, которая была репрезентативной для избирателей США. Сегодня существуют разнообразные методы, которые позволяют достигать репрезентативности, но в основе их всех лежит *случайный отбор*.



Рис. 2.4. Джордж Гэллап, который стал знаменит благодаря провалу журнала "Литературный дайджест" при работе с "большими данными"

Организовать случайный отбор не всегда просто, и надлежащее определение достижимой популяции является ключом. Предположим, что мы хотим сгенерировать репрезентативный профиль покупателей и нам нужно провести их пилотный статистический обзор. Обзор должен быть репрезентативным, но он трудоемок.

Сначала мы должны определить, кто является покупателем. Мы могли бы отобрать все записи покупателей с суммой покупки, превышающей 0. Стоит ли включать всех прошлых покупателей? Стоит ли включать компенсации? Внутренние тестовые покупки? Перекупщиков? Как биллингового агента, так и покупателя?

Далее мы должны определить процедуру отбора. Она может заключаться в том, чтобы "отобрать 100 покупателей наугад". Там, где задействован отбор из потока (например, реально-временные транзакции покупателей или посетители веб-сайта), особую важность принимают соображения, касающиеся времени (например, посе-

титель веб-сайта в 10:00 буднего дня может отличаться от посетителя веб-сайта в 22:00 на выходных).

В *стратифицированном отборе* популяция разделяется на страты, и случайные выборки берутся из *каждой страты*. Политические социологи могли бы попытаться выяснить электоральные предпочтения белых, чернокожих и латиноамериканцев. Простая случайная выборка, взятая из населения США, привела бы к очень малому числу афроамериканцев и латиноамериканцев, и поэтому в стратифицированном отборе этим стратам мог бы быть придан больший вес для получения эквивалентных размеров выборок.

Размер против качества: когда размер имеет значение?

В эру больших данных иногда вызывает удивление утверждение "Чем меньше, тем лучше". Время и усилие, затрачиваемое на случайный отбор, не только уменьшают смещение, но и позволяют уделять больше внимания на разведывание данных и качество данных. Например, пропущенные данные и выбросы могут содержать полезную информацию. Розыск отсутствующих значений или вычисление выбросов в миллионах записей могут иметь непозволительно дорогую стоимость, но эта же работа в выборке, состоящей из нескольких тысяч записей, вполне выполнима. Вывод данных на графики и ручное обследование захлебываются, если данных слишком много.

Когда же *нужны* массивные объемы данных?

Классический сценарий важности больших данных предполагает, что данные не только большие, но и разреженные. Возьмем, к примеру, поисковые запросы, получаемые компанией Google, где столбцы — это условия, строки — отдельные поисковые запросы, а значения ячеек равны 0 или 1 в зависимости от того, содержит ли запрос термин или нет. Задача состоит в том, чтобы определить наилучшим образом предсказанное назначение поиска для заданного запроса. В английском языке свыше 150 тыс. слов, и Google обрабатывает более 1 трлн запросов в год. В результате получится огромная матрица, подавляющее большинство записей в которой будут равны 0.

По своему размаху эта задача поистине относится к задачам обработки больших данных — на большинство запросов могут быть возвращены эффективные поисковые результаты, только когда накоплены такие огромные объемы данных. И чем больше данных накапливается, тем лучше результаты. Для популярных критериев поиска это не такая проблема — эффективные данные можно найти довольно быстро для небольшого числа чрезвычайно популярных тем, находящихся в тренде в то или иное время. Действительная важность современной поисковой технологии заключается в способности возвращать подробные и полезные результаты для огромного разнообразия поисковых запросов, включая те, которые возникают с частотой, скажем, всего один на миллион.

Возьмем поисковую фразу "Рики Рикардо и Красная Шапочка" (Ricky Ricardo and Little Red Riding Hood). В первые годы Интернета этот запрос, вероятно, возвратил

бы результаты по лидеру группы Рики Рикардо, телешоу "Я люблю Люси" (I Love Lucy), в котором он снимался в главной роли, и детскую сказку "Красная Шапочка". Теперь же, когда накоплены триллионы поисковых запросов, этот поисковый запрос возвращает в точности эпизод из телешоу "Я люблю Люси", в котором Рики театрально рассказывает сказку о Красной Шапочке своему новорожденному сыну, пользуясь комичным смешением английского и испанского языков.

Следует иметь в виду: для того чтобы поисковый запрос был эффективным, число фактических *искомых* записей, в которых появляется этот точный поисковый запрос или что-то очень похожее (вместе с информацией о том, на какой ссылке люди в конечном итоге нажали), возможно, должно измеряться тысячами. Однако для того чтобы получить эти релевантные записи, необходимо иметь триллионы точек данных (и случайный отбор тут, конечно, не поможет). См. также раздел "Длиннохвостые распределения" далее в этой главе.

Выборочное среднее против популяционного среднего

Символ \bar{x} (произносится "икс с чертой") используется для представления среднего значения *выборки* из популяции, тогда как μ (мю) служит для представления среднего значения *по всей популяции*. В чем важность этого различия? Информация о выборках наблюдаема, а информация о крупных популяциях часто статистически выводится из меньших по размеру выборок. В статистике предпочитают разделять эти две вещи в плане символического обозначения.

Ключевые идеи для случайного отбора и смещенной выборки

- Даже в эру больших данных случайный отбор остается важным инструментом в арсенале исследователя данных.
- Смещение возникает, когда данные измерений, или наблюдения, являются систематически ошибочными, потому что они не представляют всю популяцию в целом.
- Качество данных часто важнее их количества, а случайный отбор может уменьшить смещение и оказать содействие повышению качества, достижение которого было бы непозволительно дорогим.

Дополнительные материалы для чтения

- ♦ Полезный обзор процедур отбора можно найти в главе Рональда Фрикера (Ronald Fricker) "Методы отбора для статистических опросов в Веб и по электронной почте" (Sampling Methods for Web and E-mail Surveys) в книге "Руководство Sage по онлайн-методам статистического исследования" (Sage Handbook of Online Research Methods). Данная глава включает обзор модифика-

ций метода случайного отбора, которые часто используются по практическим причинам стоимости или выполнимости.

- ♦ Историю о провале статистического опроса, проведенного журналом "Литературный дайджест", можно найти на веб-сайте Capital Century¹.

Систематическая ошибка отбора

Перефразируем бейсболиста Йоги Берра (Yogi Berra) "Если вы не знаете, что ищите, присмотритесь повнимательнее, и вы это найдете".

Систематическая ошибка отбора (или смещение, предвзятость при отборе — selection bias) относится к практике избирательного подбора данных — осознанно или неосознанно. Таким образом, что она приводит к обманчивому или эфемерному выводу.

Ключевые термины для систематической ошибки отбора

Смещение (bias)

Систематическая ошибка.

Прочесывание данных (data snooping)

Подробная ревизия данных с целью найти что-то интересное.

Эффект бескрайнего поиска (vast search effect)

Смещение или невоспроизводимость, вытекающие из многократного моделирования данных либо моделирования данных с большим числом предсказательных переменных.

Если вы определяете гипотезу и проводите хорошо проработанный эксперимент с целью ее проверки, то можете быть твердо уверенным в выводе. Однако зачастую это не так. Вместо этого часто смотрят на имеющиеся данные в попытке разглядеть регулярности. Но является ли регулярность реальной или же она всего лишь продукт *прочесывания данных*, т. е. подробной ревизии данных, пока не появится нечто интересное? Среди статистиков популярна поговорка: "Если мучить данные слишком долго, то рано или поздно они дадут признательные показания".

Разницу между явлением, в котором вы удостоверяетесь, когда проверяете гипотезу при помощи эксперимента, и явлением, которое вы обнаруживаете, преследуя имеющиеся данные, можно разъяснить следующим мысленным экспериментом.

Предположим, что кто-то говорит вам, что он может заставить приземлиться подбрасываемую им монету орлом в течение следующих 10 бросков. Вы принимаете вызов (эквивалент эксперимента), и он приступает к 10-кратному подбрасыванию монеты, и всякий раз монета приземляется орлом. Совершенно очевидно, что вы

¹ См. <https://oreil.ly/iSoQT>.

припишите этому человеку какой-то особый талант — вероятность, что в результате 10 бросков монеты она просто по чистой случайности повернется орлом, составляет 1 из 1000.

Теперь предположим, что диктор на стадионе просит, чтобы все присутствующие 20 тыс. человек подбросили монету 10 раз и сообщили работнику стадиона, в случае если они получают 10 орлов подряд. Шанс, что *кто-то* на стадионе доберется до 10 орлов, чрезвычайно высокий (более 99% — это 1 минус вероятность того, что никто не получит 10 орлов). Безусловно, отбор постфактум человека (или людей), который получил 10 орлов на стадионе, не говорит о том, что он имеет какой-то особый талант, — скорее всего, это просто удача.

Поскольку неоднократная ревизия больших наборов данных является в науке о данных ключевым ценностным предложением, от которого бывает трудно отказаться, систематической ошибке отбора следует уделять особое внимание. Форму систематической ошибки отбора, имеющую особое значение для исследователей данных, Джон Элдер (John Elder), основатель компании Elder Research, уважаемой консалтинговой компании в области добычи регулярностей из данных, называет *эффектом бескрайнего поиска*. Если вы неоднократно строите разные модели и задаете разные вопросы в условиях крупных наборов данных, то вы непременно найдете нечто интересное. Является ли найденный результат по-настоящему чем-то заслуживающим внимания или же это случайный выброс?

Против этого можно принять защитные меры, задействовав контрольный набор с отложенными данными, а иногда более одного контрольного набора, на основе которых можно подтвердить результативность. Помимо этого, Элдер также выступает за использование того, что он называет *перетасовкой целей* (target shuffling — по сути дела, это перестановочный тест) для проверки достоверности предсказательных ассоциаций, которые предлагает модель добычи регулярностей из данных.

Типичные формы систематической ошибки отбора в статистике, в дополнение к эффекту бескрайнего поиска, включают неслучайный отбор (*см. раздел "Случайный отбор и смещенная выборка" ранее в этой главе*), данные, получаемые в результате отбора по принципу снятия сливок, отбор временных интервалов, которые подчеркивают тот или иной статистический эффект, и остановку эксперимента, когда результаты выглядят "интересными".

Регрессия к среднему

Регрессия к среднему значению относится к явлению, связанному с поочередными измерениями заданной переменной: предельные наблюдения имеют тенденцию сопровождаться более центральными значениями. Придание особого внимания и смысла предельному значению может привести к одной из форм систематической ошибки отбора.

Любители спорта знакомы с явлением "новичка года и кризиса прежнего лидера". Среди спортсменов, которые начинают свою карьеру в конкретный сезон (класс новичков), всегда присутствует тот, кто оказывается результативнее, чем все

остальные. Обычно этот "новичок года" не достигает таких же результатов в следующем году. Почему?

Почти во всех главных видах спорта, по крайней мере в командных состязаниях с мячом или шайбой, существуют два элемента, которые играют важную роль в общей результативности:

- ◆ навык;
- ◆ удача.

Регрессия к среднему значению является следствием отдельной формы систематической ошибки отбора. Когда мы отбираем новобранца с лучшей результативностью, навык и удача, вероятно, этому содействуют. В свой следующий сезон навык по-прежнему будет на месте, но в большинстве случаев удача будет отсутствовать, и поэтому его результативность упадет — она будет регрессировать. Это явление было впервые идентифицировано Фрэнсисом Гальтоном в 1886 году [Galton-1886], который описал его в связи с генетическими тенденциями; например, дети чрезвычайно высоких мужчин склонны не быть столь же высокими, что и их отцы (рис. 2.5).

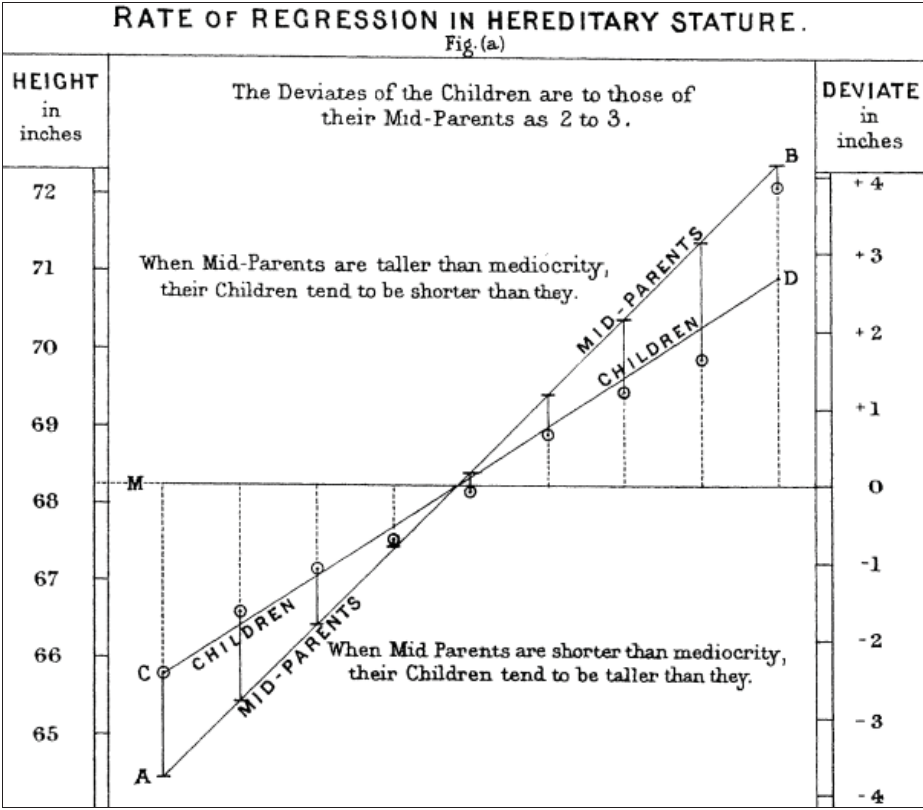


Рис. 2.5. Исследовательская работа Гальтона, в которой он идентифицировал феномен регрессии к среднему



Регрессия, т. е. "возвращение назад", к среднему отличается от метода статистического моделирования, такого как линейная регрессия, в котором линейная связь оценивается между предсказательными переменными и переменной исхода.

Ключевые идеи для систематической ошибки отбора

- Определение гипотезы и далее сбор данных, следуя принципам рандомизации и случайного отбора, обеспечивают защиту от смещения.
- Все другие формы анализа данных подвергаются риску появления смещения, вытекающего из процесса сбора/анализа данных (многократное выполнение моделей в добыче регулярностей из данных, прочесывание данных во время статистического исследования и отбор интересующих событий постфактум).

Дополнительные материалы для чтения

- ♦ Статья "Идентификация и предотвращение смещения в статистических исследованиях" (Pannucci C. J., Wilkins E. G. Identifying and Avoiding Bias in Research) в сборнике (что удивительно) "Пластмассовая и восстановительная хирургия" (Plastic and Reconstructive Surgery // 2010. — August) содержит превосходный анализ различных типов смещения, которые могут попасть в статистическое исследование, включая систематическую ошибку отбора.
- ♦ Статья "Одурачен randomness из-за систематической ошибки отбора"² (Fooled by Randomness Through Selection Bias) Майкла Хэрриса (Michael Harris) предоставляет интересный обзор соображений в отношении систематической ошибки отбора в стратегиях торговли на фондовом рынке с точки зрения трейдеров.

Выборочное распределение статистической величины

Термин "*выборочное распределение*" статистической величины обозначает распределение некоторой выборочной статистики над многочисленными выборками, извлекаемыми из одной и той же популяции. Значительная часть классической статистики занимается получением статистических выводов из (малых) выборок до (очень крупных) популяций.

² См. https://oreil.ly/v_Q0u.

Ключевые термины выборочного распределения статистической величины

Выборочная статистика (sample statistic)

Метрика, которая вычисляется для выборки данных, извлекаемой из более крупной популяции.

Синонимы: выборочная статистическая величина, статистика выборки³.

Распределение данных (data distribution)

Частотное распределение индивидуальных значений в наборе данных.

Выборочное распределение (sampling distribution)

Частотное распределение *выборочной статистики* над многочисленными выборками или повторными выборками⁴.

Синоним: распределение выборок.

Центральная предельная теорема (central limit theorem)

Тенденция выборочного распределения принимать нормальную форму по мере увеличения размера выборок.

Синоним: ЦПТ.

Стандартная ошибка (standard error)

Вариабельность (стандартное отклонение) *выборочной статистики* над многочисленными выборками (не путать со *стандартным отклонением*, которое как таковое относится к вариабельности индивидуальных значений *данных*).

В типичной ситуации выборка извлекается с целью измерения чего-нибудь (при помощи *выборочной статистики*) либо моделирования чего-нибудь (при помощи статистической или автоматически обучающейся модели). Поскольку наша оценка или модель основывается на выборке, она может быть ошибочной; она может иной, если мы решим извлечь другую выборку. Мы, следовательно, заинтересованы знать, насколько она может отличаться, — ключевой проблемой является *выборочная вариабельность*. Если бы у нас было много данных, мы могли бы извлекать дополнительные выборки и наблюдать распределение выборочной статистики непосредственно. Как правило, мы будем вычислять нашу оценку или модель, используя столько данных, сколько их имеется в наличии, так что возможность извлечения дополнительных выборок из популяции есть далеко не всегда.



Важно проводить различие между распределением индивидуальных точек данных, именуемым *распределением данных*, и распределением выборочной статистики, именуемым *выборочным распределением*.

³ Статистика — числовая величина, описывающая выборку из популяции. — *Прим. перев.*

⁴ Выборочное распределение статистики — это распределение этой статистики для всех возможных выборок фиксированного размера, скажем n , взятых из популяции. — *Прим. перев.*

Распределение выборочной статистики, такой как среднее, вероятно, будет более регулярным и колоколообразным, чем распределение самих данных. Чем больше выборка, на которой основывается статистика, тем больше она является истинной. Кроме того, чем больше выборка, тем уже распределение выборочной статистики.

Это утверждение иллюстрируется примером с использованием годового дохода для ссудозаявителей кредитного клуба Lending Club (описание данных см. в разделе "Небольшой пример: предсказание невыплаты ссуды" главы 6). Возьмем из этих данных три выборки: выборку 1000 значений, выборку 1000 средних по 5 значений и выборку 1000 средних по 20 значений. Затем построим гистограмму каждой выборки (рис. 2.6).

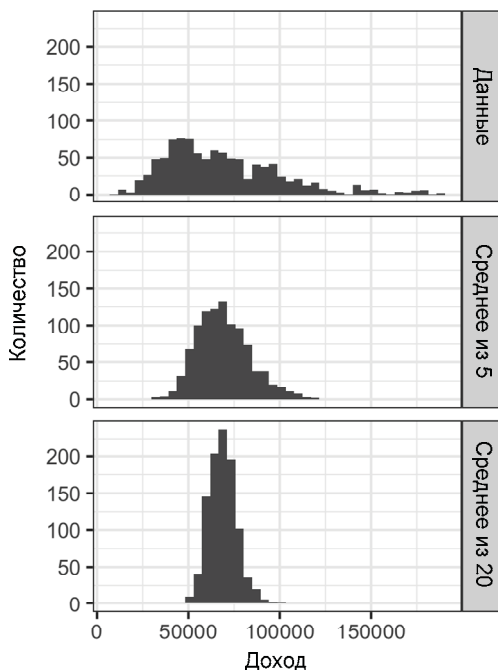


Рис. 2.6. Гистограмма годовых доходов 1000 ссудозаявителей (вверху), затем 1000 средних с числом заявителей $n = 5$ (в середине) и $n = 20$ (внизу)

Гистограмма индивидуальных значений данных широко разбросана и скошена к более высоким значениям, как и должно ожидатьсся с данными о доходах. Обе гистограммы средних из 5 и 20 значений имеют более компактный и более колоколообразный вид. Ниже приведен фрагмент кода на R, который генерирует эти гистограммы с использованием пакета визуализации `ggplot2`.

```
library(ggplot2)
# взять простую случайную выборку
samp_data <- data.frame(income=sample(loans_income, 1000),
                        type='data_dist')
```

```
# взять выборку средних по 5 значений
samp_mean_05 <- data.frame(
  income = tapply(sample(loans_income, 1000*5),
    rep(1:1000, rep(5, 1000)), FUN=mean),
  type = 'mean_of_5')
# взять выборку средних по 20 значений
samp_mean_20 <- data.frame(
  income = tapply(sample(loans_income, 1000*20),
    rep(1:1000, rep(20, 1000)), FUN=mean),
  type = 'mean_of_20')
# связать data.frames и конвертировать тип в фактор
income <- rbind(samp_data, samp_mean_05, samp_mean_20)
income$type = factor(income$type,
  levels=c('data_dist', 'mean_of_5', 'mean_of_20'),
  labels=c('Data', 'Mean of 5', 'Mean of 20'))
# построить гистограммы
ggplot(income, aes(x=income)) + geom_histogram(bins=40) + facet_grid(type ~ .)
```

Фрагмент кода на Python использует класс FacetGrid пакета seaborn для изображения трех гистограмм:

```
import pandas as pd
import seaborn as sns

sample_data = pd.DataFrame({
  'income': loans_income.sample(1000),
  'type': 'Данные',
})
sample_mean_05 = pd.DataFrame({
  'income': [loans_income.sample(5).mean() for _ in range(1000)],
  'type': 'Среднее из 5',
})
sample_mean_20 = pd.DataFrame({
  'income': [loans_income.sample(20).mean() for _ in range(1000)],
  'type': 'Среднее из 20',
})
results = pd.concat([sample_data, sample_mean_05, sample_mean_20])

g = sns.FacetGrid(results, col='type', col_wrap=1, height=2, aspect=2)
g.map(plt.hist, 'income', range=[0, 200000], bins=40)
g.set_axis_labels('Доход', 'Количество')
g.set_titles('{col_name}')
```

Центральная предельная теорема

Явление, которое мы только что описали, называется *центральной предельной теоремой*. Он говорит, что средние значения, извлеченные из многочисленных выборок, будут напоминать знакомую колоколообразную нормальную кривую (см. раздел

"Нормальное распределение" далее в этой главе), даже если исходная популяция не является нормально распределенной, при условии, что размер выборок достаточно крупный и отступление данных от нормальности не слишком большое. Центральная предельная теорема позволяет использовать нормально-аппроксимационные формулы, такие как t -распределение, применяемые в вычислении распределений выборок для статистического вывода, а именно доверительные интервалы и проверки гипотез.

В традиционных статистических проверках центральной предельной теореме уделяется большое внимание, потому что она лежит в основе механизма доверительных интервалов и проверок гипотез, которые сами занимают половину содержимого таких текстов. Исследователи данных должны знать об этой ее роли, но поскольку формальные проверки гипотез и доверительные интервалы играют в науке о данных незначительную роль, и так или иначе всегда есть *бутстреп* (см. раздел "Бутстреп" далее в этой главе), центральная предельная теорема не занимает какое-то особое место в практике науки о данных.

Стандартная ошибка

Стандартная ошибка — это одиночная метрика, которая обобщает вариабельность статистической величины в выборочном распределении. Стандартную ошибку можно оценить с использованием статистической величины, опираясь на стандартное отклонение s значений выборки и размер выборки n :

$$\text{стандартная ошибка} = \frac{s}{\sqrt{n}}.$$

По мере увеличения размера выборки стандартная ошибка уменьшается, соответствующая тому, что наблюдалось на рис. 2.6. Связь между стандартной ошибкой и размером выборки иногда носит название *правила квадратного корня из n* : для сокращения стандартной ошибки в 2 раза размер выборки должен быть увеличен в 4 раза.

Достоверность формулы стандартной ошибки вытекает из центральной предельной теоремы. На деле вам не нужно опираться на центральную предельную теорему, чтобы понять стандартную ошибку. Рассмотрим следующий подход к измерению стандартной ошибки:

1. Собрать ряд совершенно новых выборок из популяции.
2. По каждой новой выборке вычислить статистику (например, среднее).
3. Рассчитать стандартное отклонение статистики, вычисленной на шаге 2; использовать ее в качестве оценки стандартной ошибки.

На практике этот подход получения новых выборок для оценивания стандартной ошибки в типичной ситуации не выполним (и статистически очень расточителен). К счастью, как оказалось, нет необходимости извлекать совершенно новые выборки; вместо этого можно использовать *бутстраповские* повторные выборки. В современной статистике бутстреп стал типичным способом оценивания стандартной ошибки. Этот метод может использоваться фактически для любой статистики и не

опирается на центральную предельную теорему либо другие допущения о природе распределения.



Стандартное отклонение против стандартной ошибки

Не путайте стандартное отклонение (которое показывает вариабельность отдельных точек данных) со стандартной ошибкой (которая показывает вариабельность выборочной метрики).

Ключевые идеи для выборочного распределения статистической величины

- Частотное распределение выборочной статистики говорит о том, насколько эта метрика будет отличаться от выборки к выборке.
- Это выборочное распределение можно оценить посредством бутстрапа либо формул, которые опираются на центральную предельную теорему.
- Ключевой метрикой, которая обобщает вариабельность выборочной статистики, является ее стандартная ошибка.

Дополнительные материалы для чтения

Онлайновый мультимедийный ресурс по статистике⁵ Дэвида Лэйна располагает полезной симуляцией, которая позволяет вам отбирать выборочную статистику, размер выборки и число итераций, а также визуализировать гистограмму получившегося частотного распределения.

Бутстреп

Один из простых и эффективных способов оценивания выборочного распределения статистической величины или модельных параметров состоит в том, чтобы извлекать дополнительные выборки с возвратом из самой выборки и перевычислять статистику или модель для каждой повторной выборки. Данная процедура называется *бутстрапом* (от англ. *bootstrap* — раскрутка, самонастройка, исходно, "поднять себя за шнурки"), и она не сопряжена с какими-либо допущениями о том, что данные или выборочная статистика нормально распределены.

Ключевые термины для бутстрапа

Бутстраповская выборка (bootstrap sample)

Выборка, взятая с возвратом из набора наблюдаемых данных.

Синоним: бутстрап-выборка.

⁵ См. <https://oreil.ly/pe7ra>.

Повторный отбор (resampling)

Процесс многократного взятия выборок из наблюдаемых данных; включает процедуры бутстрапа и перестановки (пермутации)⁶.

Синонимы: многократный отбор, перевыборка, ресемплинг.

Концептуально вы можете представить бутстрап как репликацию исходной выборки тысячи или миллионы раз с тем, чтобы получить гипотетическую популяцию, которая воплощает все знание, исходя из оригинальной выборки (она просто крупнее). Затем из этой гипотетической популяции можно извлекать выборки в целях оценивания выборочного распределения (рис. 2.7).

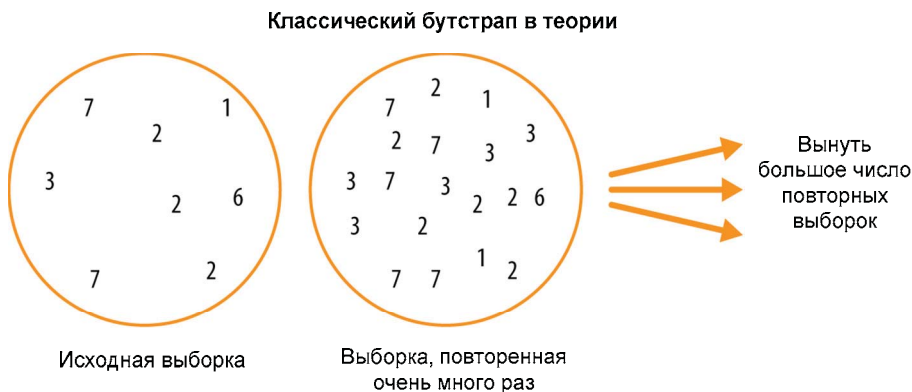


Рис. 2.7. Идея бутстрапа

На практике нет необходимости фактически реплицировать выборку огромное число раз. Мы просто возвращаем каждое наблюдение после каждого извлечения, т. е. выполняем *отбор с возвратом*. Благодаря этому мы эффективно создаем бесконечную популяцию, в которой вероятность извлекаемого элемента остается неизменной от извлечения к извлечению. Алгоритм бутстраповского повторного отбора среднего значения для выборки размера n будет следующим:

1. Извлечь выборочное значение, записать его и вернуть назад.
2. Повторить n раз.
3. Записать среднее для n повторно отобранных значений.
4. Повторить R раз шаги 1–3.
5. Использовать R результатов, чтобы:
 - вычислить их стандартное отклонение (оно оценивает стандартную ошибку выборочного среднего);

⁶ Повторный отбор — это любой метод генерирования новой выборки из существующего набора данных. — *Прим. перев.*

- построить гистограмму или коробчатую диаграмму;
- найти доверительный интервал.

В языке R число итераций бутстрапа устанавливается несколько произвольно. Чем больше вы делаете итераций, тем точнее оценка стандартной ошибки или доверительного интервала. Результатом данной процедуры является бутстраповский набор выборочных статистик или оценочных модельных параметров, которые далее можно проэкзаменовать, чтобы увидеть, насколько они переменчивы.

Пакет `boot` языка R совмещает эти шаги в одной функции. Например, в следующем примере бутстрап применяется к доходам людей, которые берут ссуды:

```
library(boot)
stat_fun <- function(x, idx) median(x[idx])
boot_obj <- boot(loans_income, R = 1000, statistic=stat_fun)
```

Функция `stat_fun` вычисляет медиану для заданной выборки, определенной индексом `idx`. Результат будет следующим:

```
Bootstrap Statistics :
      original    bias      std. error
t1*      62000 -70.5595    209.1515
```

Первоначальная оценка медианы составляет 62 000 \$. Бутстраповское распределение говорит о том, что оценка имеет *смещение* примерно -70 \$ и стандартную ошибку 209 \$. Результаты будут слегка варьироваться между поочередными прогонами алгоритма.

Главенствующие пакеты Python не предоставляют имплементаций подхода к бутстрапу. Он может быть имплементирован с помощью метода `resample` пакета `scikit-learn`:

```
results = []
for nrepeat in range(1000):
    sample = resample(loans_income)
    results.append(sample.median())
results = pd.Series(results)
print('Статистика бутстрапа:')
print(f'исходно: {loans_income.median()}')
print(f'смещение: {results.mean() - loans_income.median()}')
print(f'станд. ошибка: {results.std()}')
```

Бутстрап можно использовать с многомерными данными, где строки отбираются как единое целое (рис. 2.8). Затем на бутстрапированных данных можно выполнить модель, например, чтобы оценить стабильность (вариабельность) модельных параметров или улучшить предсказательную силу. Что касается классификационных и регрессионных деревьев (так называемых *деревьев решений*), то выполнение многочисленных деревьев на бутстраповских выборках и далее усреднение их предсказаний (или, в случае классификации, принятие решения большинством голосов) обычно оказывается результативнее, чем использование одного-единственного

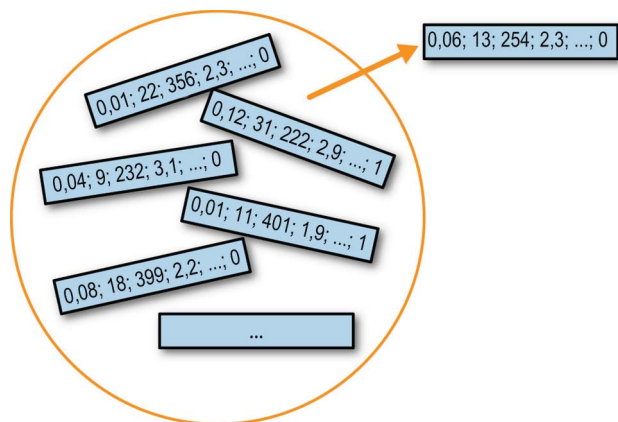


Рис. 2.8. Отбор многомерных бутстраповских выборок

дерева. Этот процесс называется *бэггингом* (от англ. *bootstrap aggregating* — агрегирование бутстраповских выборок, см. раздел "Бэггинг и случайный лес" главы 6).

Концептуально повторный отбор бутстрапа является простым, и экономист и демограф Джулиан Саймон (Julian Simon) в своей работе 1969 года "Методы фундаментального исследования в социологии" (Basic Research Methods in Social Science, Random House) опубликовал резюме примеров повторного отбора, включая бутстрап. Однако этот метод также является вычислительно емким, и до начала широкого распространения вычислительных мощностей он оставался физически неосуществимой возможностью. Метод получил свое название и приобрел популярность с опубликованием книги стэнфордского статистика Брэдли Эфрона (Bradley Efron) и благодаря нескольким статьям в журналах в конце 1970-х и в начале 1980-х годов. Данный прием в особенности был популярен среди исследователей, которые применяли статистику, но не являлись специалистами-статистиками, и предназначался для использования с метриками или моделями, где математические аппроксимации не были легкодоступны. Выборочное распределение среднего было хорошо проработано, начиная с 1908 года, что нельзя было сказать в отношении выборочного распределения многих других метрик. Бутстрап можно использовать для определения размера выборок, экспериментов с разными значениями n , чтобы понять, как они влияют на выборочное распределение.

Когда бутстрап был представлен впервые, его встретили со значительным скептицизмом; для многих он был связан с трюком превращения соломы в золото. Этот скептицизм проистекал из непонимания цели бутстрапа.



Бутстрап не компенсирует малый размер выборки; он не создает новые данные и при этом не заполняет дыры в существующем наборе данных. Он просто сообщает о том, как поведут себя многочисленные дополнительные выборки, когда они будут извлекаться из популяции, такой как наша исходная выборка.

Повторный отбор против бутстрапирования

Иногда термин "*повторный отбор*" используется в качестве синонима для термина "*бутстрапирование*", который был только что представлен в общем виде. Чаще всего термин "*повторный отбор*" также включает процедуры перестановки/пермутации (см. раздел "*Перестановочный тест*" главы 3), где многочисленные выборки объединяются и отбор может осуществляться без возврата. В любом случае термин "*бутстреп*" всегда подразумевает выборку из наблюдаемого набора данных с возвратом.

Ключевые идеи для бутстрапа

- Бутстреп (отбор образцов из набора данных с возвратом) является мощным инструментом для определения вариабельности выборочной статистики.
- Бутстреп может применяться одинаковым образом в самых различных обстоятельствах без обширного анализа математических аппроксимаций выборочных распределений.
- Этот метод также позволяет выполнять оценку выборочных распределений для статистик, где математическая аппроксимация не разработана.
- Когда этот метод применяется к предсказательным моделям, агрегирование многочисленных бутстраповских предсказаний (бэггинг) превосходит по результативности одиночную модель.

Дополнительные материалы для чтения

- ♦ "Введение в бутстреп" (Efron B., Tibshirani R. An Introduction to the Bootstrap. — Chapman Hall, 1993) — первая исследовательская работа книжного формата, где в центре внимания был метод бутстрапирования. Данная книга по-прежнему пользуется популярностью.
- ♦ Ретроспектива, посвященная бутстрапу, в журнале "Статистическая наука" (Hall P. Prehistory // Statistical Science. — 2003. — Vol. 18. — № 2). В ней обсуждается (среди других предыдущих работ) в разделе "Предыстория" Питера Халла первая публикация о бутстрапе Джулиана Саймона 1969 года.
- ♦ В книге "Введение в статистическое обучение" обратитесь к разделам по бутстрапу и, в частности, бэггингу (Gareth James et al. Introduction to Statistical Learning. — Springer, 2013).

Доверительные интервалы

Частотные таблицы, гистограммы, коробчатые диаграммы и стандартные ошибки — все они являются способами понять потенциальную ошибку в оценке выборки. Доверительные интервалы — это еще один такой способ.

Ключевые термины для доверительных интервалов

Уровень доверия (confidence level)

Процент доверительных интервалов, сконструированных одинаковым образом из одной и той же популяции, которые ожидаемо будут содержать целевую статистику.

Конечные точки интервала (interval endpoints)

Верх и низ доверительного интервала.

Человеку естественным образом свойственно избегать неопределенности; люди (в особенности эксперты) говорят "Я не знаю" крайне редко. Аналитики и менеджеры, признавая наличие неопределенности, тем не менее неоправданно доверяются оценке, когда она представлена единственным числом (*точечной оценкой*). Представляя оценку не единственным числом, а диапазоном, вы противодействуете этой тенденции. Доверительные интервалы делают это способом, который берет свое начало в статистических принципах отбора.

Доверительные интервалы всегда сопровождаются уровнем покрытия, выражаемым (высоким) процентом, скажем, 90 или 95%. 90%-ный доверительный интервал можно представить следующим образом: это интервал, который окружает центральные 90% бутстраповского выборочного распределения выборочной статистики (см. раздел "Бутстрап" ранее в этой главе). В более общем случае, x %-ный доверительный интервал вокруг выборочной оценки должен в среднем содержать похожие выборочные оценки в x % случаев (когда выполнена похожая процедура отбора).

С учетом выборки размера n и целевой выборочной статистики алгоритм для бутстраповского доверительного интервала будет следующим:

1. Извлечь из данных случайную выборку размера n с возвратом (повторная выборка).
2. Записать целевую статистику для повторной выборки.
3. Повторить шаги 1–2 много (R) раз.
4. Для x %-ного доверительного интервала отсечь $[(1 - [x/100])/2]\%$ от R результатов повторного отбора с обоих концов распределения.
5. В качестве точек отсечения принять конечные точки x %-ного бутстраповского доверительного интервала.

На рис. 2.9 показан 90%-ный доверительный интервал для среднегодового дохода ссудозаявителей на основе выборки по 20 значений, для которой среднее значение составило 57 573 \$.

Бутстрап является инструментом общего характера, который используется с целью генерирования доверительных интервалов для большинства статистик или модельных параметров. Статистические учебники и вычислительные системы с корнями

в более полувековом бескомпьютерном статистическом анализе будут также ссылаться на доверительные интервалы, генерируемые формулами, в особенности на *t*-распределение (см. раздел "*t*-Распределение Стьюдента" далее в этой главе).

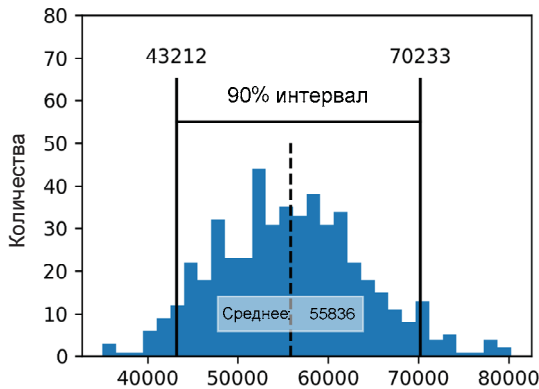


Рис. 2.9. Бутстрапский доверительный интервал для годового дохода ссудозаявителей на основе выборки из 20 значений



Разумеется, когда у нас есть результат в виде выборки, нас больше всего интересует, какова вероятность, что истинное значение лежит внутри некоторого интервала. На самом деле это не тот вопрос, на который отвечает доверительный интервал, но в итоге он сводится к тому, как большинство людей интерпретирует ответ.

Вопрос о вероятности, связанной с доверительным интервалом, начинается с фразы "какова вероятность, что с учетом процедуры отбора и популяции...". Ответ на противоположный вопрос — "какова вероятность, что (что-то является истинным в отношении популяции) с учетом результата выборки", сопряжен с более сложными расчетами и более глубокими не поддающимися точному определению факторами.

Процент, связанный с доверительным интервалом, называется *уровнем доверия*. Чем выше уровень доверия, тем шире интервал. Кроме того, чем меньше выборка, тем шире интервал (т. е. тем больше неопределенность). Оба свойства достаточно логичны: чем больше вы хотите быть уверенным и чем меньше данных у вас есть, тем шире следует сделать доверительный интервал, чтобы быть достаточно уверенным в получении истинного значения.



Для исследователя данных доверительный интервал является инструментом для получения представления о том, насколько переменным может быть результат выборки. Исследователи данных используют эту информацию не для публикации академической работы или представления результата контролирующему органу (что обычно и делает научный исследователь), а скорее всего, чтобы сообщить о потенциальной ошибке в оценке и, возможно, узнать, необходима ли более крупная выборка.

Ключевые идеи для доверительных интервалов

- Доверительные интервалы — это типичный способ представления оценок в форме интервального диапазона.
- Чем больше данных, тем менее вариабельной будет оценка выборки.
- Чем ниже уровень доверия, который вы можете допустить, тем уже будет доверительный интервал.
- Бутстрап — это эффективный способ конструирования доверительных интервалов.

Дополнительные материалы для чтения

- ♦ В изданиях "Вводная статистика и аналитика: под углом повторного отбора" (Bruce P. Introductory statistics and analytics: a resampling perspective. — John Wiley & Sons, 2014) и "Статистика" (Lock R. et al. Statistics. — Wiley, 2012) описывается подход к доверительным интервалам на основе бутстрапа.
- ♦ Инженеры, которым необходимо знать прецизионность своих измерительных данных, используют доверительные интервалы, возможно, больше, чем в других областях, и как раз книга "Современная инженерная статистика" (Ryan T. Modern Engineering Statistics. — Wiley, 2007) содержит материал о доверительных интервалах. В этой книге дается обзор инструмента, который также полезен, но привлекает меньше внимания: предсказательные интервалы (интервалы вокруг одиночного значения, в противоположность среднему значению или другой сводной статистике).

Нормальное распределение

В традиционной статистике колоколообразное нормальное распределение является каноническим⁷. Тот факт, что распределения выборочных статистик часто имеют нормальную форму, сделал его мощным инструментом в разработке математических формул, которые аппроксимируют эти распределения.

Ключевые термины для нормального распределения

Ошибка (error)

Разница между точкой данных и предсказанным либо средним значением.

Синоним: погрешность.

⁷ Кривая нормального распределения является канонической, но, возможно, она переоценена. Джордж У. Кобб (George W. Cobb), статистик из Mount Holyoke, известный за свой вклад в концепцию преподавания вводного курса статистики, в редакционной статье в журнале "Американский статистик" (American Statistician) за ноябрь 2015 года утверждает, что "стандартный вводный курс, который отводит нормальному распределению центральное место, исчерпал полезность своей центральности".

Стандартизировать (standardize)

Вычесть среднее значение и разделить на стандартное отклонение.

Z-оценка (z-score)

Результат стандартизации отдельной точки данных.

Синонимы: стандартная оценка, z-балл.

Стандартное нормальное распределение (standard normal)

Нормальное распределение со средним, равным 0, и стандартным отклонением, равным 1.

Квантиль-квантильный график (QQ-plot)

График, позволяющий визуализировать степень близости выборочного распределения к нормальному распределению.

Синонимы: QQ-график, график квантиль-квантиль.

В нормальном распределении (рис. 2.10) 68% данных находятся в пределах одного стандартного отклонения от среднего и 95% — в пределах двух стандартных отклонений.

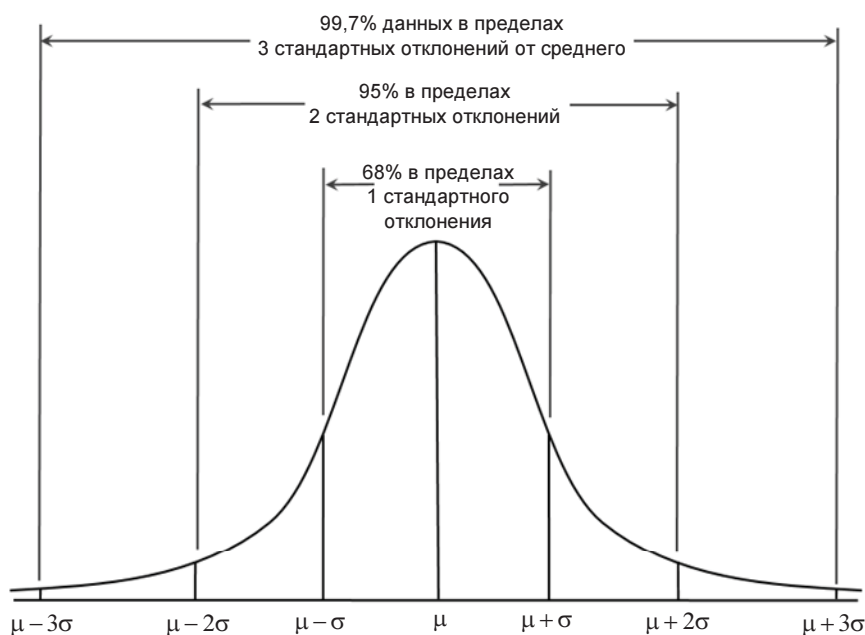


Рис. 2.10. Нормальная кривая



Существует расхожее заблуждение, что нормальное распределение так вызывается, потому что большая часть данных подчиняется нормальному распределению, т. е. это нормальная штука. Большинство переменных, используемых в типичном проекте науки о данных — по сути дела, большинство сырых данных в целом, — не являются нормально распределенными (см. раздел "Длинно-

хвостые распределения" далее в этой главе). Полезность нормального распределения вытекает из того факта, что многие статистики действительно нормально распределены в своих выборочных распределениях. Несмотря на это, допущения о нормальности являются крайней мерой, обычно используемой, когда эмпирические вероятностные распределения, или бутстраповские распределения, отсутствуют.



Нормальное распределение также называется *гауссовым распределением* в честь Карла Фридриха Гаусса (Carl Friedrich Gauss), потрясающего немецкого математика конца XVIII — начала XIX века. Для нормального распределения есть еще одно название, которое использовалось ранее, — распределение "ошибок". С точки зрения статистики *ошибка* — это разница между фактическим значением и статистической оценкой, к примеру, средним по выборке. Так, стандартное отклонение (см. раздел "Оценки вариабельности" главы 1) основывается на ошибках от среднего значения данных. Разработка Гауссом нормального распределения стала результатом его исследования ошибок астрономических измерений, которые, как было обнаружено, нормально распределены.

Стандартное нормальное распределение и квантиль-квантильные графики

Стандартное нормальное распределение — это такое распределение, в котором единицы на оси x выражены в стандартных отклонениях от среднего. Для того чтобы сравнить данные со стандартным нормальным распределением, нужно вычесть среднее и затем разделить на стандартное отклонение; эта процедура также называется *нормализацией* или *стандартизацией* (см. раздел "Стандартизация (нормализация, z -оценки)" главы 6). Отметим, что "стандартизация" в данном смысле не связана со стандартизацией записей базы данных (т. е. приведением к общему формату). Преобразованное значение называется *z -оценкой*, или стандартной оценкой, а нормальное распределение иногда называют *z -распределением*.

Квантиль-квантильный график используют, чтобы визуально определить, насколько выборка близка к нормальному распределению. Квантиль-квантильный график упорядочивает z -оценки снизу вверх и чертит z -оценки каждого значения на оси y ; ось x — это соответствующий квантиль нормального распределения для ранга этого значения. Поскольку данные нормализованы, единицы соответствуют числу стандартных отклонений данных от среднего. Если точки примерно ложатся на диагональную линию, то распределение выборки можно считать близким к нормальному. На рис. 2.11 показан квантиль-квантильный график для выборки, состоящей из 100 значений, сгенерированных в случайном порядке из нормального распределения; как и ожидалось, точки расположены вблизи диагонали. В R это изображение можно получить при помощи функции `qqnorm`:

```
norm_samp <- rnorm(100)
qqnorm(norm_samp)
abline(a=0, b=1, col='grey')
```

В Python для создания квантиль-квантильного графика используйте метод `scipy.stats.probplot`:

```
fig, ax = plt.subplots(figsize=(4, 4))
norm_sample = stats.norm.rvs(size=100)
stats.probplot(norm_sample, plot=ax)
```

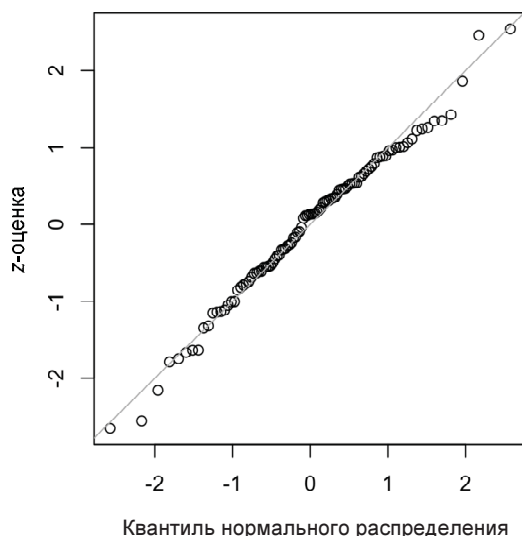


Рис. 2.11. Квантиль-квантильный график выборки, состоящей из 100 значений, извлеченных из нормального распределения



Конвертирование данных в z -оценки (т. е. стандартизация или нормализация данных) *не делает* данные нормально распределенными. Эта процедура просто помещает данные на ту же шкалу, что стандартное нормальное распределение, часто в целях сравнения.

Ключевые идеи для стандартного нормального распределения и квантиль-квантильных графиков

- Нормальное распределение имело решающее значение в историческом развитии статистики, поскольку оно позволило развить математическую аппроксимацию неопределенности и вариабельности.
- Хотя сырые данные в типичной ситуации не являются нормально распределенными, ошибки часто таковыми являются, так же как и средние и суммарные величины в крупных выборках.
- Для того чтобы конвертировать данные в z -оценки, нужно вычесть из данных среднее значение и разделить на стандартное отклонение; затем данные можно сравнить с нормальным распределением.

Длиннохвостые распределения

Несмотря на важность нормального распределения в статистике в историческом плане и в отличие от того, о чем говорит его название, данные обычно не являются нормально распределенными.

Ключевые термины для длиннохвостых распределений

Хвост (tail)

Длинная узкая часть частотного распределения, где относительно предельные значения встречаются с низкой частотой.

Асимметрия (skew)

Состояние, когда один хвост распределения длиннее другого.

Синоним: скошенность.

В то время как нормальное распределение нередко является обоснованным и полезным в отношении распределения ошибок и выборочных статистик, оно обычно не характеризует распределение сырых данных. Иногда распределение сильно *скошено* (асимметрично), как в случае данных о доходах, или же распределение может быть дискретным, как в примере с биномиальными данными. И симметричные, и асимметричные распределения могут иметь *длинные хвосты*. Хвосты распределения соответствуют (малым и большим) предельным значениям. Длинные хвосты и принятие заградительных мер против них широко признаны в практике. Нассим Талеб (Nassim Taleb) предложил теорию *черного лебедя*, которая предсказывает, что аномальные события, такие как обвал фондового рынка, могут возникать с намного большей вероятностью, чем их предсказание нормальным распределением.

Хорошим примером, который иллюстрирует длиннохвостую природу данных, является финансовая возвратность акций. На рис. 2.12 показан квантиль-квантильный график ежедневной возвратности акций компании Netflix (NFLX). В R он генерируется следующим образом:

```
nflx <- sp500_px[, 'NFLX']  
nflx <- diff(log(nflx[nflx>0]))  
qqnorm(nflx)  
abline(a=0, b=1, col='grey')
```

Соответствующий код на Python таков:

```
nflx = sp500_px.NFLX  
nflx = np.diff(np.log(nflx[nflx>0]))  
fig, ax = plt.subplots(figsize=(4, 4))  
stats.probplot(nflx, plot=ax)
```

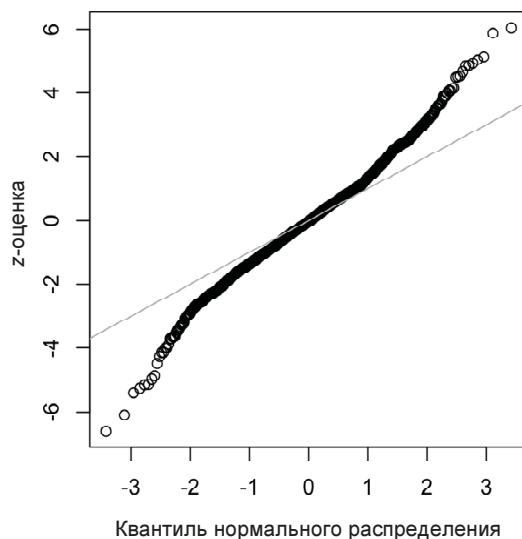


Рис. 2.12. Кванталь-квантильный график возвратности для акций компании Netflix (NFLX)

В отличие от рис. 2.11, точки расположены намного ниже линии для низких значений и намного выше прямой для высоких значений. Это означает, что мы наблюдаем предельные значения с намного большей вероятностью, чем можно ожидать, если бы данные имели нормальное распределение. На рис. 2.12 продемонстрирован еще один распространенный феномен: точки лежат близко к линии для данных в пределах одного стандартного отклонения от среднего. Тьюки именует это явление как данные, которые "нормальны в середине", но имеют более длинные хвосты (см. [Tukey-1987]).



Задаче подгонки статистических распределений к наблюдаемым данным посвящено много статистической литературы. Остерегайтесь подхода, чрезмерно центрированного на данных, к этой работе, которая является искусством в такой же мере, как и наукой. Данные вариабельны и нередко непротиворечивы на первый взгляд, и при этом могут иметь более одной формы и более одного типа распределения. Как правило, должно быть пушено в ход предметное и статистическое знание с целью определить, какое распределение является обоснованным для моделирования данной ситуации. Например, мы можем располагать данными об уровне интернет-трафика на сервере за большое число поочередных 5-секундных периодов. Полезно знать, что самое подходящее распределение для моделирования "событий в расчете на период времени" будет пуассоновским (см. раздел "Пуассоновские распределения" далее в этой главе).

Ключевые идеи для длиннохвостых распределений

- Большая часть данных не является нормально распределенной.
- Принятие нормального распределения в качестве исходного допущения может привести к недооценке предельных событий ("черных лебедей").

Дополнительные материалы для чтения

- ◆ Талеб Н. Черный лебедь. Под знаком непредсказуемости (Taleb N. The Black Swan. — 2nd ed. — Random House, 2010).
- ◆ "Справочник по статистическим распределениям с применениями" (Krishna K. Handbook of Statistical Distributions with Applications. — 2nd ed. — CRC Press, 2016).

t-Распределение Стьюдента

Распределение Стьюдента, или *t-распределение* — это распределение нормальной формы, но немного толще и длиннее в хвостах. Оно широко используется для изображения распределений выборочных статистик. Распределения выборочных средних, как правило, имеют форму как у *t*-распределения, при этом существует семейство *t*-распределений, которые различаются в зависимости от того, насколько большой является выборка. Чем больше выборка, тем более нормальную форму принимает *t*-распределение.

Ключевые термины для t-распределения Стьюдента

n

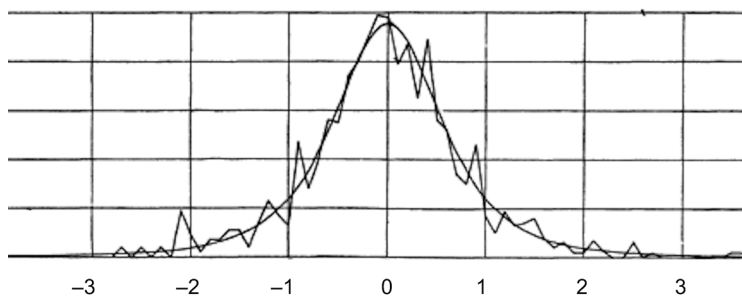
Размер выборки.

Степени свободы (degrees of freedom)

Параметр, который позволяет *t*-распределению адаптироваться к разным размерам выборок, статистикам и числам групп.

t-Распределение часто называют *распределением Стьюдента*, потому что оно было опубликовано в 1908 году в журнале "Биометрика" (Biometrika) У. С. Госсетом (W. S. Gossett) под псевдонимом "Студент". Работодатель Госсетта, руководство пивоваренного завода Guinness, не хотело, чтобы конкуренты знали, что он использует статистические методы, поэтому настояло на том, чтобы Госсет не упоминал свое имя в своей статье.

Госсет хотел ответить на вопрос "Каково выборочное распределение среднего по выборке, извлеченной из более крупной популяции?". Он начал с эксперимента на основе повторного отбора — извлекая случайные выборки по 4 элемента из набора данных, состоящего из 3 тыс. замеров роста и длины левого среднего пальца преступников. (То время часто называют эрой евгеники, когда в центре внимания находились данные о преступниках и обнаружение корреляций между склонностями к преступлениям и физическими или психологическими особенностями.) Госсет нанес стандартизированные результаты (*z*-оценки) на ось *x* и частоты — на ось *y*. Параллельно с этим он получил функцию, ныне известную как функция *t* Стьюдента, и выполнил подгонку этой функции над результатами выборок, графически отобразив сравнение (рис. 2.13).



Шкала стандартного отклонения выборки

Рис. 2.13. Результаты эксперимента Госсетта на основе повторного отбора и подогнанная кривая функции t (из его работы в журнале "Биометрика", 1908 г.)

Целый ряд разных статистик, после стандартизации, можно сопоставить с t -распределением с целью оценить доверительные интервалы в свете выборочной вариации. Рассмотрим выборку размера n , для которой было вычислено выборочное среднее \bar{x} . Если s — это выборочное стандартное отклонение, то 90%-ный доверительный интервал вокруг выборочного среднего задается следующей формулой:

$$\bar{x} \pm t_{n-1}(0,05) \cdot \frac{s}{t},$$

где $t_{n-1}(0,05)$ — это значение t -статистики с $(n-1)$ степенями свободы (см. раздел "Степени свободы" главы 3), которое "отсекает" 5% t -распределения с обоих концов. t -Распределение используется в качестве эталона для распределения выборочного среднего, разницы между двумя выборочными средними, параметрами регрессии и другими статистиками.

Если бы в 1908 году вычислительные мощности были широко доступны, то, вне всякого сомнения, наука статистика с самого начала намного в большей степени опиралась бы на вычислительно емкие методы повторного отбора. Лишенные компьютеров, ученые-статистики обратились к математике и функциям, таким как t -распределение, с целью аппроксимации выборочного распределения. В 1980-х годах вычислительные мощности компьютеров позволили активизировать практические эксперименты с повторным отбором, но к тому времени использование t -распределения и подобных распределений уже глубоко укоренилось в учебниках и вычислительных системах.

Точность t -распределения в описании поведения выборочной статистики требует, чтобы распределение этой статистики для этой выборки имело форму как у нормального распределения. Как оказалось, выборочные статистики нередко действительно являются нормально распределенными, даже когда данные опорной популяции таковыми не являются (факт, который привел к широкому применению t -распределения). Это возвращает нас назад к явлению, известному как *центральная предельная теорема* (см. раздел "Центральная предельная теорема" ранее в этой главе).



Что исследователи данных должны знать о t -распределении и центральной предельной теореме? Не очень-то и много. t -Распределение используется в классическом статистическом выводе, но оно не имеет столь важного значения для целей науки о данных. Для исследователей данных важны понимание и количественная оценка неопределенности и вариации, но эмпирическое взятие бутстраповских выборок способно ответить на большинство вопросов об ошибке в выборке. Однако исследователи данных будут рутинно сталкиваться с t -статистикой в результатах работы статистических вычислительных систем и статистических процедур в R, например при A/B -тестировании и регрессиях, поэтому знакомство с его назначением будет очень полезным.

Ключевые идеи для t -распределения Стьюдента

- t -Распределение — это на самом деле семейство распределений, напоминающих нормальное распределение, но с более толстыми хвостами.
- t -Распределение широко используется в качестве эталонного базиса для распределения выборочных средних, разниц между двумя выборочными средними, параметров регрессии и т. д.

Дополнительные материалы для чтения

- ♦ Исходная работа Госсетта в журнале "Биометрика" за 1908 год доступна в формате PDF⁸.
- ♦ Стандартную трактовку t -распределения можно найти в онлайн-ресурсе Дэвида Лэйна⁹.

Биномиальное распределение

Исходы в форме да/нет, т. е. биномиальные, лежат в основе аналитики, поскольку они часто являются кульминацией решения либо другого процесса: купить/не купить, нажать/не нажать, выжить/погибнуть и т. д. Центральное место для понимания биномиального распределения занимает идея множества *испытаний*, где каждое испытание имеет два возможных исхода с определенными вероятностями.

Например, 10-кратное подбрасывание монеты является биномиальным экспериментом с 10 испытаниями, где каждое испытание имеет два возможных исхода (орел или решка). Взгляните на рис. 2.14. Такие исходы в форме да/нет или 0/1 называются *двоичными*, и они не обязательно имеют вероятности 50/50. Любые вероятности, которые в сумме составляют 1,0, возможны. В статистике принято называть исход "1" *успехом*; на практике также общепринято назначать "1" более редкому исходу. Использование термина "*успех*" не говорит о том, что исход является

⁸ См. <https://oreil.ly/J6gDg>.

⁹ См. <https://oreil.ly/QxUkA>.

желательным или выгодным; на самом деле он скорее говорит об интересующем исходе. Например, невыплаты ссуд или мошеннические транзакции являются относительно редкими событиями, в предсказании которых мы можем быть заинтересованы, поэтому им дают название "1" или "успех".



Рис. 2.14. Сторона с решкой бизоньего пятицентовика

Ключевые термины для биномиального распределения

Испытание (trial)

Событие с дискретным исходом (например, подбрасывание монеты).

Успех (success)

Интересующий результат испытания.

Синоним "1" (в противоположность "0").

Биномиальный (binomial)

Имеющий два исхода.

Синонимы: да/нет, 0/1, двоичный, бинарный, двучленный.

Биномиальное испытание (binomial trial)

Испытание с двумя исходами.

Синоним: бернуллиево испытание.

Биномиальное распределение (binomial distribution)

Распределение числа успехов в x испытаниях.

Синоним: бернуллиево распределение.

Биномиальное распределение — это частотное распределение числа успехов (x) в заданном числе испытаний (n) с указанной вероятностью (p) успеха в каждом испытании. Существует семейство биномиальных распределений, которые подразделяются в зависимости от значений x , n и p . Биномиальное распределение отвечает на такой вопрос:

Если вероятность нажатия, которое конвертируется в продажу, составляет 0,02, то какова вероятность наблюдать 0 продаж при 200 нажатиях?

Функция `dbinom` языка R вычисляет биномиальные вероятности. Например:

```
dbinom(x=2, n=5, p=0.1)
```

вернет 0,0729, вероятность наблюдать ровно $x = 2$ успехов при $n = 5$ испытаниях, где вероятность успеха для каждого испытания равна $p = 0,1$. В приведенном выше примере мы используем $x = 0$, размер, равный 200, и $p = 0,02$. С этими аргументами `dbinom` возвращает вероятность 0,0176.

Часто мы заинтересованы в определении вероятности x или меньшего количества успехов при n испытаниях. В этом случае мы используем функцию `pbinom`:

```
pbinom(2, 5, 0.1)
```

Она вернет 0,9914, вероятность наблюдать два или меньшее число успехов в пяти испытаниях, где вероятность успеха для каждого испытания равна 0,1.

Модуль `scipy.stats` имплементирует большое разнообразие статистических распределений. Для биномиального распределения используйте функции `stats.binom.pmf` и `stats.binom.cdf`:

```
stats.binom.pmf(2, n=5, p=0.1)
```

```
stats.binom.cdf(2, n=5, p=0.1)
```

Среднее биномиального распределения равно $n \times p$; вы также можете думать о нем как об ожидаемом числе успехов при n испытаниях для вероятности успеха, равной p .

Дисперсия равна $n \times p(1 - p)$. При достаточно большом числе испытаний (в особенности когда p близко к 0,50) биномиальное распределение фактически неотлично от нормального распределения. На самом деле вычисление биномиальных вероятностей с выборками крупных размеров является вычислительно затратным, и в большинстве статистических процедур используется нормальное распределение со средним и дисперсией в качестве аппроксимации.

Ключевые идеи для биномиального распределения

- Биномиальные исходы важны для моделирования, поскольку они представляют, среди всего прочего, фундаментальные решения (купить или не купить, нажать или не нажать, выжить или погибнуть и т. д.).
- Биномиальное испытание — это эксперимент с двумя возможными исходами: один с вероятностью p и другой с вероятностью $1 - p$.
- При крупном n и при условии, что p не слишком близка к 0 или 1, биномиальное распределение может быть аппроксимировано нормальным распределением.

Дополнительные материалы для чтения

- ♦ Почитайте о "кинконксе" (`quincunx`)¹⁰ — симуляционном устройстве, похожем на игру в пинбол и предназначенном для демонстрации биномиального распределения.
- ♦ Биномиальное распределение составляет важнейшую часть вводного курса статистики, и все введения в статистику обязательно будут содержать одну-две главы по этой теме.

Распределение хи-квадрат

Важной идеей в статистике является *отступление от ожидания*, особенно в отношении количеств по категориям. Ожидание в широком понимании определяется как "ничего необычного или примечательного в данных" (например, отсутствие корреляции между переменными или предсказуемыми регулярностями). Оно также называется *нулевой гипотезой* или *нулевой моделью* (см. раздел "*Нулевая гипотеза*" главы 3). Например, вы, возможно, захотите протестировать независимость одной переменной (скажем, строчной переменной, представляющей пол) от еще одной (скажем, столбцовой переменной, представляющей "был повышен в должности"), и у вас есть количества в каждой ячейке таблицы данных. Статистическая величина, которая измеряет степень отступления результатов от нулевого ожидания независимости, называется *статистикой хи-квадрат*. Это указанная разница между наблюдаемыми и ожидаемыми значениями, деленная на квадратный корень из ожидаемого значения, возведенная в квадрат, а затем суммированная по всем категориям. Этот процесс стандартизирует статистику, в результате чего ее можно сравнивать с эталонным распределением. Более общий способ ее выразить — отметить, что статистика хи-квадрат является мерой степени, с которой набор наблюдаемых значений "укладывается" в указанное распределение (тест "оптимальности подгонки"). Она полезна для определения того, отличаются ли многочисленные варианты эксперимента ("тест A/B/C...") друг от друга по своим эффектам.

Распределение хи-квадрат является распределением этой статистической величины при неоднократных повторных извлечениях из нулевой модели (см. раздел "*Проверка на основе статистики хи-квадрат*" главы 3), где приводится подробный алгоритм и формула хи-квадрат для таблицы данных. Низкое значение хи-квадрата для набора количественных данных указывает на то, что они точно следуют ожидаемому распределению. Высокий хи-квадрат указывает на то, что они заметно отличаются от того, что ожидается. Существует разнообразие распределений хи-квадрат, которые связаны с разными степенями свободы (например, число наблюдений — см. раздел "*Степени свободы*" главы 3).

¹⁰ См. <https://oreil.ly/nmkcs>.

Ключевая идея для распределения хи-квадрат

- Распределение хи-квадрат в типичной ситуации имеет отношение к количеству (появлений) предметов или элементов по категориям.
- Статистика хи-квадрат измеряет степень отступления от того, что вы ожидали бы в нулевой модели.

Дополнительные материалы для чтения

- ♦ Распределение хи-квадрат обязано своим местом в современной статистике великому статистику Карлу Пирсону и рождению проверки гипотез — об этом и многом другом читайте в книге Дэвида Салсбурга "Леди дегустирует чай: как статистика революционизировала науку в двадцатом веке" (Salsburg D. The lady tasting tea: how statistics revolutionized science in the twentieth century. — Henry Holt and Company, 2001. — 352 p.).
- ♦ Более подробное изложение см. в разделе *"Проверка на основе статистики хи-квадрат"* главы 3, посвященном проверке хи-квадрат.

F-распределение

Обычная процедура в научном эксперименте заключается в тестировании многочисленных вариантов эксперимента по всем группам — скажем, различных удобрений на разных участках поля. Это похоже на тест *A/B/C*, упомянутый в разделе *"Распределение хи-квадрат"* ранее в этой главе, за исключением того, что мы имеем дело с измеряемыми непрерывными значениями, а не с количествами. В данном случае нас интересует степень, в которой различия между средними значениями групп больше, чем мы могли бы ожидать при нормальной случайной вариации. *F*-статистика измеряет это и представляет собой отношение вариальности между средними группами к вариальности внутри каждой группы (так называемую остаточную вариальность). Это сравнение называется *дисперсионным анализом* (см. раздел *"Дисперсионный анализ"* главы 3). Распределение *F*-статистики — это частотное распределение всех значений, которые были бы получены путем случайной перестановки данных, в которых все групповые средние равны (т. е. нулевая модель). Существует разнообразие *F*-распределений, которые связаны с различными степенями свободы (например, число групп, см. раздел *"Степени свободы"* главы 3). Расчет величины *F* иллюстрируется в разделе, посвященном процедуре дисперсионного анализа (ANOVA). *F*-статистика также используется в линейной регрессии для сравнения вариации, объясняемой регрессионной моделью, с совокупной вариацией в данных. *F*-статистика автоматически производится в языках R и Python в рамках процедур регрессионного и дисперсионного анализа.

Ключевая идея для F -распределения

- F -распределение используется в экспериментах и линейных моделях, включающих измеряемые данные.
- F -статистика сравнивает вариации, обусловленные интересующими факторами, с совокупной вариацией.

Дополнительные материалы для чтения

Книга Джорджа Кобба "Введение в дизайн и анализ экспериментов" (Cobb G. Introduction to design and analysis of experiments. — Wiley, 2008) содержит прекрасное изложение декомпозиции дисперсионных компонентов, которое помогает в понимании процедуры дисперсионного анализа и F -статистики.

Распределение Пуассона и другие связанные с ним распределения

Многие процессы порождают события в случайном порядке при заданной совокупной интенсивности — посетители, прибывающие на веб-сайт, автомобили, прибывающие в пункт сбора дорожной пошлины (события, распространяющиеся во времени), изъяны в квадратном метре ткани или опечатки в расчете на 100 строк исходного кода (события, распространяющиеся в пространстве).

Ключевые термины для распределения Пуассона и других с ним связанных распределений

Лямбда (λ)

Интенсивность (в расчете на единицу времени или пространства), с которой события происходят.

Распределение Пуассона (Poisson distribution)

Частотное распределение числа событий в отобранных единицах времени или пространства.

Экспоненциальное распределение (exponential distribution)

Частотное распределение времени или расстояния от одного события до следующего события.

Распределение Вейбулла (Weibull distribution)

Обобщенная версия экспоненциального распределения, в котором допускается смещение интенсивности события во времени.

Пуассоновские распределения

Исходя из предшествующих данных, мы можем оценить среднестатистическое число событий в расчете на единицу времени или пространства, но мы также, возможно, захотим узнать, насколько оно может отличаться от одной единицы времени/пространства к другой. Распределение Пуассона говорит нам о распределении событий в расчете на единицу времени или пространства, когда мы выбираем много таких единиц. Оно полезно, когда отвечают на вопросы о массовом обслуживании, к примеру, такой: "Какие мощности нам потребуются, чтобы на 95% быть уверенными в полной обработке интернет-трафика, который прибывает на сервер в любой 5-секундный период?"

Ключевым параметром в распределении Пуассона является λ (лямбда). Это среднее число событий, которое происходит в указанный интервал времени или пространства. Дисперсия пуассоновского распределения тоже равна λ .

Общепринятый прием состоит в генерировании случайных чисел из распределения Пуассона в качестве составной части симуляции массового обслуживания. Функция `rpois` в R делает это, беря всего два аргумента — количество искоемых случайных чисел и лямбда:

```
rpois(100, lambda=2)
```

Этот фрагмент кода генерирует 100 случайных чисел из распределения Пуассона, где $\lambda = 2$. Например, если среднее число входящих звонков в службу поддержки клиентов равно 2 в минуту, то этот фрагмент кода симулирует 100 минут, возвращая число вызовов в каждую из этих 100 минут.

Экспоненциальное распределение

Взяв тот же параметр λ , который мы использовали в распределении Пуассона, мы также можем смоделировать распределение времени между событиями: время между посещениями веб-сайта или между прибытиями автомобилей в пункт сбора дорожной пошлины. Оно также используется в инженерном деле для моделирования времени безотказной работы и в управлении процессами для моделирования, например, времени, требуемого в расчете на сервисный вызов. Фрагмент кода на R для генерирования случайных чисел из экспоненциального распределения берет два аргумента: `n` (количество чисел, которые будут сгенерированы) и `rate` (число событий в расчете на период времени). Например:

```
rexp(n = 100, rate = .2)
```

В функции `stats.expon.rvs` порядок этих аргументов обратный:

```
stats.expon.rvs(0.2, size=100)
```

Этот фрагмент кода генерирует 100 случайных чисел из экспоненциального распределения, где среднее число событий в расчете на период времени равно 2. Таким образом, его можно использовать для моделирования 100 интервалов в минутах между сервисными вызовами, где средняя интенсивность входящих вызовов равна 0,2 в минуту.

Ключевое допущение в любом симуляционном исследовании как для пуассоновского, так и для экспоненциального распределений состоит в том, что интенсивность λ остается постоянной в течение рассматриваемого периода. Это редко вызывает проблемы в глобальном смысле; например, движение на дорогах или трафик в сетях передачи данных варьируется по времени суток и по дню недели. Однако периоды времени либо области пространства обычно могут быть поделены на сегменты, которые являются достаточно гомогенными, в результате чего допустимы анализ или симуляция в течение этих периодов.

Оценивание интенсивности отказов

Во многих приложениях интенсивность события λ известна, или ее можно оценить из предшествующих данных. Однако в редких случаях это не обязательно так. Отказ авиационного двигателя, например, случается довольно редко (к счастью), так что для данного типа двигателя может иметься мало данных, на которых можно было бы базировать оценку времени между отказами. При полном отсутствии данных почти нет никакой базы, на основе которой можно оценивать интенсивность события. Однако можно высказать какие-то догадки: если никакие события не были замечены по прошествии 20 часов, то можно быть вполне уверенным в том, что интенсивность не равна 1 в расчете на час. Посредством симуляции либо прямого вычисления вероятностей вы можете определять разные гипотетические интенсивности событий и оценивать пороговые значения, ниже которых интенсивность вряд ли когда-либо упадет. Если данные имеются, но их недостаточно для того, чтобы обеспечить точную, надежную оценку интенсивности, то к различным интенсивностям может быть применена проверка оптимальности подгонки (*см. раздел "Проверка на основе статистики хи-квадрат" главы 3*), чтобы можно было определить, насколько хорошо они соответствуют наблюдаемым данным.

Распределение Вейбулла

Во многих случаях интенсивность события не остается постоянной во времени. Если период, за который она изменяется, намного длиннее, чем типичный интервал между событиями, то никаких проблем; вы просто подразделяете анализ на сегменты, где интенсивности являются относительно постоянными, как упомянуто ранее. Если же интенсивность события изменяется внутри временного интервала, то экспоненциальное либо пуассоновское распределение больше не приносит пользы. Это, скорее всего, будет касаться случаев механического отказа — риск отказа увеличивается с течением времени. *Распределение Вейбулла* является расширением экспоненциального распределения, в котором допускается изменение интенсивности события в соответствии с *параметром формы* β . Если $\beta > 1$, то вероятность события увеличивается во времени, если $\beta < 1$, то она уменьшается. Поскольку распределение Вейбулла используется вместе с анализом времени безотказной работы (наработки на отказ) вместо интенсивности события, второй параметр выражен с точки зрения характерного времени жизни (ресурсной характеристики), а не с точки зрения интенсивности событий в расчете на интервал. Здесь используется

символ η , греческая буква "эта", который также называется *параметром масштаба*, или *шкалы*¹¹.

С распределением Вейбулла задача оценивания теперь предусматривает оценивание двух параметров: β и η . Вычислительная система используется с целью моделирования данных и оценивания оптимально подогнанного распределения Вейбулла.

Фрагмент кода на R для генерирования случайных чисел из распределения Вейбулла принимает три аргумента: *n* (количество чисел, которые будут сгенерированы), форму *shape* и масштаб *scale*. Например, следующий фрагмент кода сгенерирует 100 случайных чисел (времена жизни) из распределения Вейбулла с формой 1,5 и характерным временем жизни 5000:

```
rweibull(100, 1.5, 5000)
```

Для того чтобы добиться того же в Python, используйте функцию `stats.weibull_min.rvs`:

```
stats.weibull_min.rvs(1.5, scale=5000, size=100)
```

Ключевые идеи для распределения Пуассона и других с ним связанных распределений

- Для событий, которые происходят с постоянной интенсивностью, число событий в расчете на единицу времени или пространства может быть смоделировано как распределение Пуассона.
- Вы также можете смоделировать время или расстояние между одним событием и следующим событием как экспоненциальное распределение.
- Изменяющаяся во времени интенсивность события (например, увеличивающаяся вероятность отказа устройства) может быть смоделирована распределением Вейбулла.

Дополнительные материалы для чтения

- ♦ Книга "Современная инженерная статистика" (Ryan T. Modern Engineering Statistics. — Wiley, 2007) содержит главу, посвященную распределениям вероятностей, используемым в инженерных приложениях.
- ♦ О распределении Вейбулла с точки зрения инженерного дела можно почитать в соответствующих источниках¹².

¹¹ Параметр масштаба (*scale*) — это параметр вероятностного распределения, чье физическое конкретное значение связано с выбором шкалы измерения. — *Прим. перев.*

¹² См. <https://oreil.ly/1x-ga>; <https://oreil.ly/9bn-U>.

Резюме

В эру больших данных принципы отбора случайных выборок по-прежнему имеют большое значение, когда необходимы точные оценки. Случайный отбор данных уменьшает смещение и приводит к набору данных более высокого качества, чем тот, который можно получить в результате простого использования легкодоступных данных. Знание различных выборочных распределений и распределений, порождающих данные, позволяет нам квантифицировать потенциальные ошибки в оценке, которые могут быть обусловлены случайной вариацией. В то же время бутстрап (отбор из наблюдаемого набора данных с возвратом) является привлекательным шаблонным ("единым для всех") методом для определения возможной ошибки в оценках выборок.

Статистические эксперименты и проверка значимости

Планирование экспериментов является краеугольным камнем практической статистики с приложениями фактически во всех областях исследования. Цель состоит в том, чтобы спланировать эксперимент, который подтвердит или отклонит гипотезу. Исследователи данных сталкиваются с потребностью проводить непрерывные эксперименты, в особенности относительно пользовательского интерфейса и товарного маркетинга. В этой главе представлен обзор традиционного планирования экспериментов и обсуждается несколько распространенных задач в науке о данных. В ней также будут рассмотрено несколько часто цитируемых в статистическом выводе понятий и дано объяснение их смысла и актуальности (или отсутствия таковой) для науки о данных.

Всякий раз упоминание статистической значимости, p -значений или проверки на основе t -статистики происходит, как правило, в контексте классического "конвейера" статистического вывода (рис. 3.1). Этот процесс начинается с гипотезы ("препарат A лучше существующего стандартного препарата", "цена A прибыльнее существующей цены B "). Эксперимент (это может быть A/B -тест) предназначен для проверки гипотезы, построенной таким образом, чтобы обеспечивать неоспоримые результаты. Данные собираются и анализируются, и далее делается заключение. Термин "*вывод*" отражает намерение применить экспериментальные результаты, которые предусматривают лимитированный набор данных, к более крупному процессу или популяции.

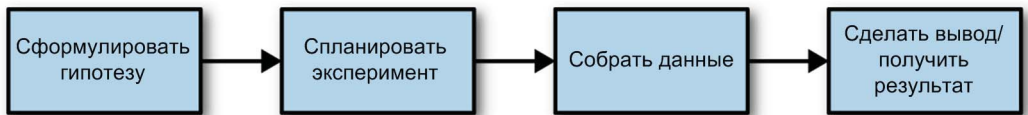


Рис. 3.1. Классический конвейер статистического вывода

A/B -тестирование

A/B -тест — это эксперимент с двумя группами для определения лучшего из двух вариантов, двух продуктов, двух процедур, двух лекарственных средств и т. п. Нередко один вариант из двух является стандартным существующим вариантом или отсутствует вообще. Если используется стандартный вариант (или же он отсутствует), то он называется *контрольным*. Типичная гипотеза состоит в том, что предлагаемый вариант лучше контрольного.

Ключевые термины для A/B-тестирования

Вариант эксперимента (treatment)

Нечто (лекарственное средство, цена, заголовок веб-страницы), что испытуемому предлагается в целях тестирования.

Тестовая группа (treatment group)

Группа испытуемых, которым предлагается конкретный вариант.

Контрольная группа (control group)

Группа испытуемых, которым предлагается стандартный вариант или не предлагается никакой.

Рандомизация (randomization)

Процесс случайного отнесения испытуемых к вариантам.

Испытуемые (subjects)

Субъекты (посетители веб-сайта, пациенты и т. д.), которым предлагаются варианты.

Проверочная статистика (test statistic)

Метрика, которая используется для измерения эффекта варианта.

A/B-тесты часто встречаются в веб-дизайне и маркетинге, поскольку их результаты очень легко измеряются. Некоторые примеры A/B-тестирования включают:

- ♦ тестирование двух методов обработки почвы, чтобы определить, какая из них приводит к наилучшему прорастанию саженцев;
- ♦ тестирование двух методов лечения, чтобы определить, какое из них эффективнее подавляет рак;
- ♦ тестирование двух цен, чтобы определить, какая из них приносит больше чистой прибыли;
- ♦ тестирование двух заголовков веб-страницы, чтобы определить, какой из них порождает больше нажатий (рис. 3.2);
- ♦ тестирование двух веб-объявлений, чтобы определить, какое из них генерирует больше конверсий.

Надлежащий A/B-тест имеет испытуемых, которые могут быть отнесены к тому или иному варианту эксперимента. Испытуемым может быть человек, саженец, посетитель веб-сайта; главное, что испытуемому предлагается вариант эксперимента. В идеальном случае испытуемые *рандомизируются* (назначаются в случайном порядке) по двум предлагаемым вариантам. Благодаря этому вы знаете, что любая разница между тестовыми группами происходит вследствие одного из двух:

- ♦ эффекта разных вариантов эксперимента;
- ♦ чистой случайности, с которой испытуемые назначаются вариантам (т. е. случайное назначение, возможно, привело к тому, что более результативные испытуемые были естественным образом сконцентрированы в A или B).

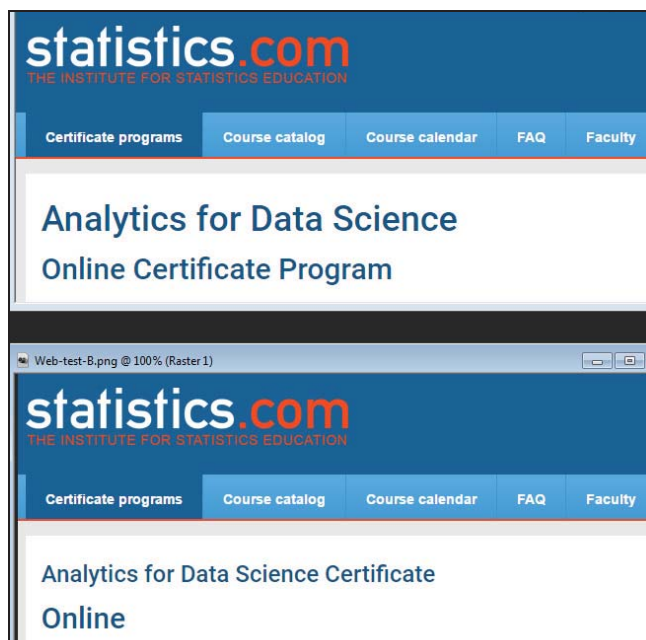


Рис. 3.2. Маркетологи все время тестируют веб-презентации, сравнивая одну с другой

Также необходимо обратить внимание на *проверочную статистику*, или метрику, которую вы используете для сравнения группы *A* с группой *B*. Возможно, наиболее часто встречающейся метрикой в науке о данных является двоичная переменная: наличие или отсутствие нажатия, наличие или отсутствие покупки, наличие или отсутствие мошенничества и т. д. Эти результаты обобщаются в таблице размера 2×2 . В табл. 3.1 приведена таблица 2×2 с результатами тестирования фактической цены (дальнейшее обсуждение этих результатов см. в разделе "*Статистическая значимость и р-значения*" далее в этой главе).

Таблица 3.1. Таблица 2×2 с результатами эксперимента с электронной коммерцией

Исход	Цена A	Цена B
Конверсия	200	182
Нет конверсии	23 539	22 406

Если метрика представлена непрерывной переменной (суммой покупки, прибылью и т. д.), или количеством (например, днями в стационаре, посещенными страницами), то результат может быть показан по-разному. Если интересует не конверсия, а выручка в расчете на один просмотр страницы, то результаты ценового теста в табл. 3.1 могут выглядеть, как в типичном результате вычислительной системы, генерируемом по умолчанию:

Доход/просмотр с ценой A: среднее = 3.87, CO = 51.10

Дохода/просмотра с ценой B: среднее = 4.11, CO = 62.98

"CO" обозначает стандартное отклонение значений внутри каждой группы.



Тот факт, что статистические вычислительные системы, включая R и Python, генерируют результат по умолчанию, не означает, что вся эта информация является полезной или релевантной. Вы можете видеть, что приведенные выше стандартные отклонения не особо полезны; судя по всему, они говорят о том, что многочисленные значения могли бы быть отрицательными, когда отрицательная выручка не является возможной. Эти данные состоят из малого набора относительно высоких значений (просмотры с конверсиями) и огромного числа нулевых значений (просмотры без конверсий). Очень трудно обобщить вариабельность таких данных в одном-единственном числе, хотя среднее абсолютное отклонение от среднего (7,68 для A и 8,15 для B) представляется более разумным, чем стандартное отклонение.

Зачем нужна контрольная группа?

Почему нельзя проигнорировать контрольную группу и просто выполнить эксперимент, применив интересующий вариант эксперимента только к одной группе и сравнив результат с предшествующим опытом?

Без контрольной группы нет никакой гарантии, что "прочие условия будут равными", а любая разница действительно обусловлена вариантом эксперимента (либо случайным образом). Когда есть контрольная группа, она подчиняется тем же условиям (за исключением интересующего варианта), что и тестовая группа. Если просто сравнивать с "базовой линией" или предшествующим опытом, то помимо варианта могут разниться и другие факторы.



Ослепление в статистическом исследовании

Слепое исследование — это исследование, в котором испытуемые не осведомлены о том, что им предлагается вариант A либо вариант B . Осведомленность о том или ином варианте может повлиять на отклик. В *двойном слепом исследовании* исследователи и помощники (например, врачи и медсестры в медицинском исследовании) не осведомлены о том, какие испытуемые участвуют и какой вариант им предлагается. Слепое исследование невозможно, когда природа варианта прозрачна, например когнитивная психотерапия при помощи компьютера в отличие от психолога.

Использование A/B -тестирования в науке о данных, как правило, находится в веб-контексте. В качестве вариантов эксперимента могут выступать дизайн веб-страницы, цена товара, формулировка заголовка объявления или какой-либо другой элемент. При этом необходимо серьезно задуматься о том, как обеспечить сохранность принципов рандомизации. В типичной ситуации испытуемым в эксперименте является посетитель веб-сайта, а измеряемыми нами исходами, в которых мы заинтересованы, — нажатия, покупки, продолжительность посещения, число посещаемых веб-страниц, просмотрена ли определенная страница и т. п. В стандартном A/B -эксперименте нужно выбрать одну метрику заранее. Могут быть собраны и представлять интерес многочисленные поведенческие метрики, но если эксперимент ожидаемо ведет к выбору между вариантом A и вариантом B , то одна метрика, или проверочная статистика, должна быть установлена заранее. Выбор провероч-

ной статистики после того, как эксперимент проведен, открывает дверь для смещения вследствие предвзятости исследователя.

Почему только *A/B*? Почему не *C*, *D*...?

A/B-тесты популярны в мире маркетинга и электронной коммерции, но это далеко не единственный тип статистического эксперимента. Дополнительные варианты вполне возможны. Испытуемые могут быть подвергнуты повторным измерительным исследованиям. Фармацевтические испытания, где субъекты дефицитны, дорогостоящи и участвуют в течение долгого времени, иногда планируются с многочисленными возможностями остановить эксперимент и достичь окончательного заключения.

В традиционном планировании статистического эксперимента центральное внимание уделяется ответу на статический вопрос об эффективности указанных вариантов. Исследователи данных меньше интересуются вопросом:

"Является ли разница между ценой *A* и ценой *B* статистически значимой?",

чем вопросом:

"Какая из многочисленных возможных цен является лучшей?"

Для этого используется относительно новый тип планирования эксперимента: *многорукий бандит* (см. раздел "*Алгоритм многорукого бандита*" далее в этой главе).



Получение разрешения

В научном и медицинском исследованиях, в которых принимают участие люди, обычно требуется получать их согласие на проведение эксперимента, а также одобрение институционального ревизионного совета по вопросам этики. Эксперименты в бизнесе, которые выполняются в рамках непрерывных операций, почти никогда не подвергаются сбору предварительных согласий. В большинстве случаев (например, ценовые эксперименты или эксперименты, связанные с тем, какой заголовок показать либо какое предложение следует сделать) такая практика является общепринятой. Компания Facebook, однако, столкнулась с этими общепринятыми правилами в 2014 году, когда проводила эксперименты с эмоциональным тоном в пользовательских лентах новостей. Компания использовала сентиментный анализ для классификации постов ленты новостей на положительные или отрицательные, затем поменяла положительно-отрицательный баланс в материале, который она показывала пользователям. Несколько случайно отобранных пользователей испытывали на себе более положительные посты, тогда как другие — более отрицательные. Было обнаружено, что пользователи, которые читали более положительную ленту новостей, с большей вероятностью сами отправляли положительные посты, и наоборот. Однако величина эффекта была малой, при этом компания Facebook столкнулась с большой критикой в том плане, что эксперимент проводился без ведома пользователей. Некоторые пользователи полагали, что компания Facebook вполне могла подтолкнуть чрезвычайно подавленных пользователей к краю, когда те получали отрицательную версию своего канала.

Ключевые идеи для A/B-тестирования

- Испытуемые распределяются на две (или более) групп, с которыми обращаются строго одинаково, за исключением того, что варианты отличаются один от другого.
- В идеальном случае испытуемые распределяются по группам в случайном порядке.

Дополнительные материалы для чтения

- ◆ Двухгрупповые сравнения (A/B-тесты) являются элементом традиционной статистики, и почти любой вводный курс статистики будет содержать разносторонний материал по принципам планирования экспериментов и процедурам статистического вывода. Обсуждение, которое помещает A/B-тесты в контекст, более соответствующий науке о данных, и использует повторный отбор, можно найти в книге "Введение в статистику и аналитику: под углом повторного отбора" (Bruce P. Introductory statistics and analytics: a resampling perspective. — John Wiley & Sons, 2014).
- ◆ По поводу веб-тестирования можно сказать: логистические аспекты тестирования могут быть столь же сложными, что и статистические. Хорошей отправной точкой является раздел справки, посвященный экспериментам, в Google Analytics¹.
- ◆ Будьте осторожны с советами из широко распространенных руководств по A/B-тестированию, которые можно увидеть в сети, такими как следующие слова из одного такого руководства: "Дождитесь примерно 1000 посетителей и постарайтесь, чтобы тест проводился в течение одной недели". Такие эмпирические правила не являются статистически содержательными; дальнейшие подробности см. в разделе "Мощность и размер выборки" далее в этой главе.

Проверки гипотез

Проверки гипотез, так называемые *проверки значимости*, получили широкое распространение в традиционном статистическом анализе, который встречается в публикуемых исследованиях. Такие проверки предназначены для того, чтобы помочь узнать, может ли случайность быть ответственной за наблюдаемый эффект.

¹ См. <https://oreil.ly/mAbqF>.

Ключевые термины для проверок гипотез

Нулевая гипотеза (null hypothesis)

Гипотеза о том, что виной всему является случайность.

Альтернативная гипотеза (alternative hypothesis)

Гипотеза, компенсирующая нулевую (то, что вы надеетесь доказать).

Односторонняя проверка (one-way test)

Проверка гипотезы, при которой количество случайных результатов подсчитывается только в одном направлении.

Двусторонняя проверка (two-way test)

Проверка гипотезы, при которой количество случайных результатов подсчитывается в двух направлениях.

В типичной ситуации *A/B-тест* (см. раздел "*A/B-тестирование*" ранее в этой главе) конструируется с учетом гипотезы. Например, гипотеза может заключаться в том, что цена *B* приносит более высокую прибыль. Зачем нам нужна гипотеза? Почему нельзя просто взглянуть на результат эксперимента и остановиться на любом варианте, который работает лучше?

Ответ кроется в склонности человеческого разума недооценивать размах естественного случайного поведения. Одно из проявлений этой склонности состоит в неумении предвидеть предельные события, или так называемых "черных лебедей" (см. раздел "*Длиннохвостые распределения*" главы 2). Еще одним ее проявлением является тенденция неправильно истолковывать случайные события как имеющие признаки некой значимости. Статистическая проверка гипотез была изобретена как способ защитить исследователей от того, чтобы оказаться обманутым случайностью.

Неправильное истолкование случайности

Склонность людей недооценивать случайность (или randomness) можно наблюдать в следующем эксперименте. Попросите нескольких своих друзей вообразить 50-кратное подбрасывание монеты и записать серию случайных исходов: О (орел) и Р (решка). Затем попросите их на самом деле подбросить монету 50 раз и записать результаты. Пусть они поместят реальные результаты подбрасывания монеты в одну колонку, а выдуманные результаты — в другую. Легко различить, какие результаты являются настоящими: настоящие результаты будут иметь более длинную вереницу, состоящую из О или Р. Увидеть в наборе из 50 настоящих подбрасываний пять или шесть О или Р подряд не является чем-то необычным. Однако, когда большинство из нас воображает случайные подбрасывания монеты и у нас получается три или четыре О подряд, мы говорим себе: чтобы серия выглядела случайной, нам лучше всего переключиться на Р.

Другая сторона этой монеты, если можно так выразиться, состоит в том, что, когда мы на деле видим реальный эквивалент, состоящий из шести О подряд (например, когда один заголовок объявления превосходит другой на 10%), мы склонны приписывать это чему-то реальному, а не просто случайности.

В надлежаще спланированном A/B -тесте вы собираете данные о вариантах A и B таким образом, что любая наблюдаемая разница между A и B должна произойти вследствие одного из двух:

- ◆ случайности в отнесении испытуемых в группы;
- ◆ истинной разницы между A и B .

Статистическая проверка гипотез является дальнейшим анализом A/B -теста или любого рандомизированного эксперимента с целью определить, является ли случайность разумным объяснением наблюдаемой разницы между группами A и B .

Нулевая гипотеза

В проверках гипотез используется следующая логика: "С учетом склонности человека реагировать на необычное, но случайное поведение и истолковывать его как нечто содержательное и реальное, в наших экспериментах нам потребуются доказательства того, что разница между группами является более предельной, чем та, которая обоснованно могла бы быть порождена случайностью". Эта логика сопряжена с базисным допущением о том, что варианты эксперимента эквивалентны и любая разница между группами обусловлена случайностью. Это базисное допущение называется нулевой гипотезой. И наша надежда тогда состоит в том, что мы сможем на деле доказать неправильность нулевой гипотезы и показать, что исходы для групп A и B различаются больше, чем то, что может породить случайность.

Один из путей сделать это лежит через процедуру повторного отбора с перестановкой, в которой мы перетасовываем результаты групп A и B и далее неоднократно раздаем данные в группы аналогичных размеров, а затем наблюдаем за тем, как часто мы получаем такую же предельную разницу, что и наблюдаемая разница. Объединенные перетасованные результаты из групп A и B и процедура повторного отбора из них воплощают нулевую гипотезу о том, что группы A и B являются эквивалентными и взаимозаменяемыми, и называется *нулевой моделью*. Для получения дополнительных подробностей см. раздел "*Повторный отбор*" далее в этой главе.

Альтернативная гипотеза

Проверки гипотез по их природе предусматривают не только нулевую гипотезу, но и компенсирующую ее альтернативную гипотезу. Вот несколько примеров:

- ◆ нулевая гипотеза — "разницы между средними в группе A и группе B нет", альтернативная гипотеза — " A отличается от B " (может быть больше или меньше);
- ◆ нулевая гипотеза — " $A \leq B$ ", альтернативная — " $B > A$ ";
- ◆ нулевая гипотеза — " B не больше A на $X\%$ ", альтернативная — " B больше A на $X\%$ ".

Взятые вместе нулевая и альтернативная гипотезы охватывают абсолютно все имеющиеся возможности. Природа нулевой гипотезы определяет структуру проверки гипотезы.

Односторонняя проверка гипотезы против двухсторонней

Нередко в A/B -тесте вы проверяете новую возможность (скажем, B) относительно взятой по умолчанию возможности (A) и изначально исходите из того, что будете придерживаться исходной возможности, если только новая возможность не окажется определенно лучше. В таком случае при проверке гипотез вам потребуется защититься от того, чтобы не быть обманутым случайностью по направлению в пользу B . Вас не волнует, что вы можете быть обманутыми случайностью в другом направлении, потому что будете оставаться на стороне A , если только B не окажется определенно лучше. Поэтому вы хотите иметь направленную альтернативную гипотезу (B лучше A). В таком случае вы используете проверку односторонней гипотезы (или гипотезу с одним хвостом). Это означает, что предельный шанс приводит только к однонаправленному подсчету в сторону p -значения.

Если вы хотите, чтобы проверка гипотезы защитила вас от того, чтобы быть обманутым случайностью в любом направлении, то альтернативная гипотеза должна быть *двунаправленной* (A отличается от B и может быть больше или меньше). В таком случае вы используете *двустороннюю гипотезу* (или гипотезу с двумя хвостами). Это означает, что предельный шанс приводит к двунаправленному подсчету в сторону p -значения.

Проверка гипотезы с одним хвостом часто соответствует природе принятия решения в A/B -тестировании, в котором принятие решения является обязательным, и одной возможности обычно назначается "дефолтный" статус, если только другая не оказывается лучше. Однако вычислительные системы, включая R, как правило, по умолчанию предоставляют на выходе двухстороннюю проверку, и многие специалисты-статистики отдают предпочтение более консервативной двухсторонней проверке, только чтобы предотвратить споры. Тема различий между проверками с одним хвостом или с двумя хвостами является довольно запутанной и не имеет прямого отношения к науке о данных, где прецизионность расчетов p -значения не особо важна.

Ключевые идеи для проверок статистических гипотез

- Нулевая гипотеза — это логический конструкт, воплощающий понятие о том, что ничего особенного не произошло, и любой эффект, который вы наблюдаете, обусловлен случайностью.
- Проверка гипотезы исходит из того, что нулевая гипотеза является истинной, создает "нулевую модель" (вероятностную модель) и проверяет, является ли эффект, который вы наблюдаете, разумным результатом этой модели.

Дополнительные материалы для чтения

- ♦ Книга "Походка алкаша" (The Drunkard's Walk, Leonard Mlodinow, Vintage Books, 2008) — это удобочитаемый обзор ситуаций, когда "рандомность управляет нашими жизнями".
- ♦ Книга "Статистика, 4-е издание" (Statistics, 4th ed., David Freedman, Robert Pisani, Roger Purves W. W. Norton, 2007) — классическое статистическое издание, в котором содержатся превосходные нематематические выкладки большинства статистических тем, включая проверку гипотез.
- ♦ Книга "Введение в статистику и аналитику: под углом повторного отбора" (Introductory Statistics and Analytics: A Resampling Perspective, Peter Bruce, Wiley, 2014) развивает тему проверки гипотез с использованием повторного отбора.

Повторный отбор

Повторный отбор в статистике означает многократное извлечение значений из наблюдаемых данных с общей целью определения случайной вариабельности в статистической величине. Он также используется для определения и улучшения точности некоторых автоматически обучающихся моделей (например, предсказания моделей на основе деревьев решений, построенных на многократно бутстрапированных наборах данных, которые усредняются в результате процесса, именуемого *бэггингом* (см. раздел "Бэггинг и случайный лес" главы 6).

Существуют два главных типа процедур повторного отбора: *бутстрап* и *перестановка* (пермутация). Бутстрап используется для определения надежности оценки; этот метод обсуждался в предыдущей главе (см. раздел "Бутстрап" главы 2). Перестановочные тесты применяются для проверки гипотез и в типичной ситуации предусматривают две или более групп, и мы обсудим их в данном разделе.

Ключевые термины для повторного отбора

Перестановочный тест (permutation test)

Процедура объединения двух или более выборок и случайное (либо исчерпывающее) перераспределение наблюдений в повторные выборки.

Синонимы: рандомизационный тест, случайный пермутационный тест, точный тест.

Повторный отбор (resampling)

Извлечение дополнительных выборок ("повторных выборок") из наблюдаемого набора данных.

С возвратом либо без возврата (with or without replacement)

При отборе элемент либо возвращается, либо не возвращается в выборку перед следующим извлечением.

Перестановочный тест

В процедуре *перестановки* задействуются две или более выборки, как правило, группы в *A/B*-тесте или другой проверке гипотез. Перестановка означает изменение порядка следования значений, или их пермутацию. Первый шаг в *перестановочной проверке* гипотезы состоит в объединении результатов из групп *A* и *B* (и, групп *C*, *D*, ..., если они используются). В этом заключается логическое воплощение нулевой гипотезы — варианты эксперимента, которые были предложены группам, не различаются. Затем мы проверяем эту гипотезу путем случайного извлечения групп из этого объединенного множества и смотрим, насколько они отличаются от друг друга. Процедура перестановки такова:

1. Объединить результаты из разных групп в единый набор данных.
2. Перетасовать объединенные данные, затем в случайном порядке извлечь (без возврата) повторную выборку того же размера, что и группа *A* (очевидно, что она будет содержать немного данных из других групп).
3. Из оставшихся данных в случайном порядке извлечь (без возврата) повторную выборку того же размера, что и группа *B*.
4. Сделать то же для групп *C*, *D* и т. д. Теперь вы собрали один набор повторных выборок, которые отражают размеры исходных выборок.
5. В зависимости от статистики или оценки, которая была вычислена для исходных выборок (например, разница в групповых долях), теперь рассчитать ее для повторных выборок и записать; это будет одной итерацией перестановки.
6. Повторить предыдущие шаги *R* раз для получения перестановочного распределения проверочной статистики.

Теперь вернемся к наблюдаемой разнице между группами и сравним ее с набором перестановленных разниц. Если наблюдаемая разница убедительно лежит в пределах набора перестановленных разниц, то мы ничего не доказали — наблюдаемая разница находится внутри диапазона того, что может породить случайность. Однако если наблюдаемая разница лежит вне большей части перестановочного распределения, то мы приходим к заключению, что случайность не несет ответственности. Говоря техническим языком, разница является *статистически значимой* (см. раздел "*Статистическая значимость и p-значения*" далее в этой главе).

Пример: прилипчивость веб-страниц

Компания, продающая относительно дорогостоящую услугу, хочет протестировать, какая из двух веб-презентаций справляется с продажей лучше. Вследствие дороговизны продаваемой услуги продажи случаются нечасто, и цикл продаж продолжительный; аккумулятивное достаточного количества продаж с целью узнать, какая презентация превосходит, заняло бы слишком много времени. Поэтому компания решает измерить результаты при помощи эрзац-переменной, используя подробную внутреннюю страницу веб-сайта, описывающую услугу.



Эрзац-переменная, или *прокси* (косвенная переменная), — это переменная, которая подменяет истинную интересующую переменную, возможно отсутствующую, либо слишком дорогостоящую, либо измерение которой требует слишком продолжительного времени. В климатических исследованиях, например, содержание кислорода в кернах древнего льда используется в качестве эрзаца для температуры. При этом полезно иметь по крайней мере немного данных об истинной переменной, являющейся предметом интереса, чтобы определить силу ее связи с эрзацем.

Одна из потенциальных эрзац-переменных для нашей компании — это число нажатий на подробной посадочной странице. Еще лучше выяснить, сколько времени люди проводят на указанной странице. Разумно полагать, что веб-презентация (веб-страница), которая задерживает внимание людей дольше, приведет к большему числу продаж. Следовательно, при сравнении страницы *A* со страницей *B* нашей метрикой будет среднее время сеанса.

Вследствие того факта, что мы имеем дело с внутренней страницей специального назначения, она не получает огромного числа посетителей. Также обратите внимание на то, что служба Google Analytics (GA), а именно так мы измеряем время сеанса, не может измерить время для последнего сеанса посещения страницы клиентом. Вместо удаления этого сеанса из данных GA записывает его как 0, поэтому данные требуют доработки, чтобы удалить такие сеансы. В результате получается 36 сеансов для двух разных презентаций: 21 для страницы *A* и 15 для страницы *B*. Используя пакет `ggplot`, мы можем визуально сравнить времена сеансов при помощи парных коробчатых диаграмм:

```
ggplot(session_times, aes(x=Page, y=Time)) +  
  geom_boxplot()
```

Функция `boxplot` пакета `pandas` использует именованный аргумент `by`, чтобы создать рисунок:

```
ax = session_times.boxplot(by='Page', column='Time')  
ax.set_xlabel('')  
ax.set_ylabel('Время, c')  
plt.suptitle('')
```

Приведенная на рис. 3.3 коробчатая диаграмма говорит о том, что страница *B* приводит к более продолжительным сеансам, чем страница *A*. Средние для каждой группы могут быть вычислены следующим образом:

```
mean_a <- mean(session_times[session_times['Page']=='Page A', 'Time'])  
mean_b <- mean(session_times[session_times['Page']=='Page B', 'Time'])  
mean_b - mean_a  
[1] 21.4
```

В Python мы фильтруем кадр данных `pandas` сначала по странице, а затем определяем среднее значение столбца `Time`:

```
mean_a = session_times[session_times.Page == 'Page A'].Time.mean()  
mean_b = session_times[session_times.Page == 'Page B'].Time.mean()  
mean_b - mean_a
```

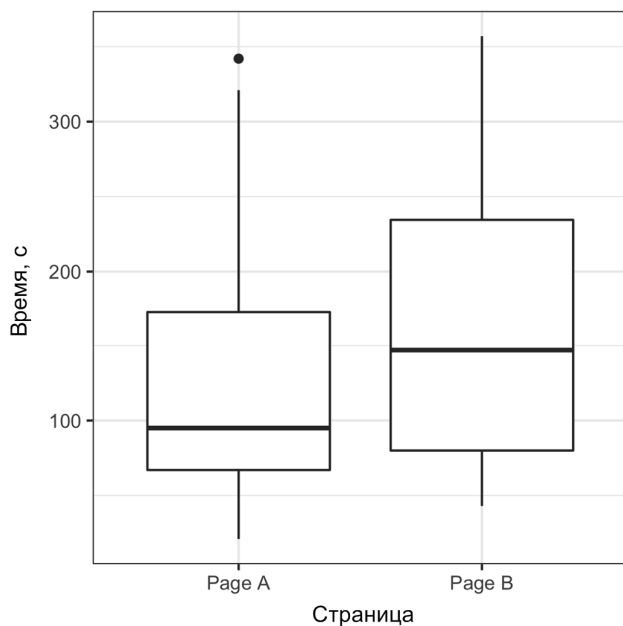


Рис. 3.3. Времена сеансов для веб-страниц A и B

Времена сеансов страницы B продолжительнее в среднем на 21,4 секунды в отличие от страницы A. Вопрос состоит в том, находится ли эта разница внутри диапазона того, что может быть порождено случайностью, или же, напротив, является статистически значимой? Один из путей ответить на этот вопрос состоит в применении перестановочного теста — объединить все времена сеансов, затем многократно перетасовать и поделить их на группы, состоящие из 21 элемента (вспомним, что $n = 21$ для страницы A) и из 15 элементов ($n = 15$ для B).

Для применения перестановочного теста нам нужна функция, которая будет случайно назначать 36 времен сеансов группе из 21 элемента (страница A) и группе из 15 элементов (страница B):

```
perm_fun <- function(x, n1, n2)
{
  n <- n1 + n2
  idx_b <- sample(1:n, n1)
  idx_a <- setdiff(1:n, idx_b)
  mean_diff <- mean(x[idx_b]) - mean(x[idx_a])
  return(mean_diff)
}
```

Версия этого перестановочного теста на языке Python выглядит следующим образом:

```
def perm_fun(x, nA, nB):
    n = nA + nB
    idx_B = set(random.sample(range(n), nB))
```

```

idx_A = set(range(n)) - idx_B
return x.loc[idx_B].mean() - x.loc[idx_A].mean()

```

Указанная функция работает путем отбора n_B индексов без возврата и отнесения их к группе B ; оставшиеся n_A индексы назначаются группе A . Данная функция возвращает разницу между двумя средними. Вызов этой функции в количестве $R=1000$ раз и установка $n_A=21$ и $n_B=15$ приводят к распределению разниц во временах сеансов, которые могут быть изображены с помощью функции `hist`.

```

perm_diffs <- rep(0, 1000)
for(i in 1:1000)
  perm_diffs[i] = perm_fun(session_times[, 'Time'], 21, 15)
hist(perm_diffs, xlab='Session time differences (in seconds)')
abline(v = mean_b - mean_a)

```

В Python мы можем создать подобный график с помощью пакета `matplotlib`:

```

perm_diffs = [perm_fun(session_times.Time, nA, nB) for _ in range(1000)]

fig, ax = plt.subplots(figsize=(5, 5))
ax.hist(perm_diffs, bins=11, rwidth=0.9)
ax.axvline(x = mean_b - mean_a, color='black', lw=2)
ax.text(50, 190, 'Observed\ndifference', bbox={'facecolor':'white'})
ax.set_xlabel('Разница во времени сессий, с')
ax.set_ylabel('Частота')

```

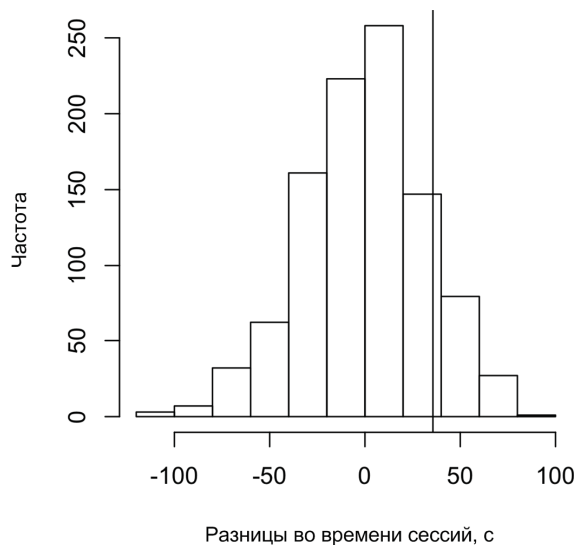


Рис. 3.4. Частотное распределение разниц во временах сеансов между страницами A и B

Гистограмма на рис. 3.4, показывает, что средняя разница случайных перестановок часто превышает наблюдаемую разницу во временах сеансов (вертикальная линия). Для наших результатов это происходит в 12,6% случаев:

```
mean(perm_diffs > (mean_b - mean_a))
---
0.126
```

Поскольку при моделировании используются случайные числа, указанный процент будет варьировать. Например, в версии Python мы получили 12,1%:

```
np.mean(perm_diffs > mean_b - mean_a)
---
0.121
```

Это свидетельствует о том, что наблюдаемая разница во временах сеансов между страницей *A* и страницей *B* находится далеко внутри диапазона случайной вариации и, следовательно, не является статистически значимой.

Исчерпывающий и бутстраповский перестановочные тесты

В дополнение к предыдущей процедуре случайной перетасовки, также именуемой *случайным перестановочным тестом*, или *рандомизационным тестом*, существуют два варианта перестановочного теста:

- ♦ исчерпывающий перестановочный тест;
- ♦ бутстраповский перестановочный тест.

В исчерпывающем перестановочном тесте вместо простой случайной перетасовки и разделения данных мы фактически выясняем все возможные способы, которыми они могут быть разделены. Это практически осуществимо только для относительно малых размеров выборок. При большом числе повторных перетасовок результаты случайного перестановочного теста аппроксимируют результаты исчерпывающего перестановочного теста и приближаются к ним в пределе. Исчерпывающие перестановочные тесты также иногда именуются точными тестами из-за их статистического свойства, которое гарантирует, что нулевая модель не протестируется как "значимая" больше альфа-уровня теста (*см. раздел "Статистическая значимость и p-значения" далее в этой главе*).

В бутстраповском перестановочном тесте извлечения, кратко описанные в шагах 2 и 3 случайного перестановочного теста, делаются *с возвратом*, а не без возврата. Благодаря этому процедура повторного отбора моделирует не только случайный элемент в назначении варианта испытуемому, но и случайный элемент в отборе испытуемых из популяции. Обе процедуры встречаются в статистике, и разница между ними носит довольно замысловатый характер и не имеет особой важности в практике науки о данных.

Перестановочные тесты: сухой остаток для науки о данных

Перестановочные тесты являются полезными эвристическими процедурами для разведывания роли случайной вариации. Они относительно легко программируются, интерпретируются и объясняются. Они также предлагают нетривиальный

окольный путь в обход формализма и "ложного детерминизма" формализованных статистик.

Одно из достоинств повторного отбора, в отличие от формульных подходов, состоит в том, что он почти вплотную сближается с шаблонным ("единым для всех") подходом к статистическому выводу. Данные могут быть числовыми или двоичными. Размеры выборок могут быть одинаковыми или разными. Допущения о нормально распределенных данных не требуются.

Ключевые идеи для перестановочного теста

- В перестановочном тесте многочисленные выборки объединяются и далее перетасовываются.
- Перетасованные значения затем делятся на повторные выборки, и далее вычисляется интересующая статистика.
- Этот процесс повторяется далее, и повторно отобранная статистика сводится в таблицу.
- Сравнение наблюдаемого значения статистики с повторно отобранным распределением позволяет судить о том, могла ли наблюдаемая разница между выборками произойти случайно.

Дополнительные материалы для чтения

- ♦ Прочитайте книгу "Перестановочные тесты" (Edgington E., Onghena P. Randomization Tests. — 4th ed. — Chapman Hall, 2007), но не слишком лезьте в дебри неслучайного отбора.
- ♦ "Вводная статистика и аналитика: под углом повторного отбора" (Bruce P. Introductory statistics and analytics: a resampling perspective. — John Wiley & Sons, 2014).

Статистическая значимость и p -значения

Статистическая значимость описывает, как специалисты-статистики проводят измерения, дает ли эксперимент (или вообще любое исследование существующих данных) более предельный результат, чем тот, который может быть получен случайностью. Если результат лежит за пределами случайной вариации, то принято говорить, что он является *статистически значимым*.

Ключевые термины для статистической значимости и p -значения

p -Значение (p -value)

С учетом случайной модели, которая воплощает нулевую гипотезу, p -значение является вероятностью получения результатов столь же необычных или предельных, что и наблюдаемые результаты.

Альфа (alpha)

Вероятностный порог "необычности", который случайные результаты должны превзойти, чтобы фактические исходы считались статистически значимыми.

Синоним: уровень значимости.

Ошибка 1-го рода (type 1 error)

Ошибочный вывод о том, что эффект является реальным (тогда, как он обусловлен случайностью).

Ошибка 2-го рода (type 2 error)

Ошибочный вывод о том, что эффект обусловлен случайностью (тогда, как он является реальным).

Рассмотрим в табл. 3.2 результаты веб-теста, показанного ранее.

Таблица 3.2. Таблица 2×2 для результатов эксперимента с электронной коммерцией

Исход	Цена А	Цена В
Конверсия	200	182
Нет конверсии	23 539	22 406

Цена *A* конвертирует (т. е. превращает посетителей в покупателей) почти на 5% лучше, чем цена *B* (0,8425% против 0,8057% — разность 0,0368 процентных пунктов); это достаточно много, чтобы компания имела содержательное значение в крупномасштабном бизнесе. Здесь имеется более 45 тыс. точек данных, и возникает соблазн рассматривать их как "большие данные", не требуя проверок статистической значимости (необходимой главным образом для того, чтобы учитывать выборочную вариабельность в небольших выборках). Однако интенсивность конверсии является настолько низкой (менее 1%), что фактические содержательные значения — конверсии — находятся лишь в сотых долях, и нужный размер выборки по настоящему определяется этими конверсиями. Мы можем протестировать разность в конверсиях между ценами *A* и *B* на предмет ее расположения внутри диапазона случайной вариации, воспользовавшись процедурой повторного отбора. Под "случайной вариацией" мы имеем в виду случайную вариацию, произведенную вероятностной моделью, которая воплощает нулевую гипотезу о том, что нет никакой разницы между степенями (см. раздел "Нулевая гипотеза" ранее в этой главе). Следующая ниже перестановочная процедура ставит вопрос: если этим двум ценам присуща одна и та же интенсивность конверсии, то сможет ли случайная вариация произвести разницу размером 5%?

1. Поместить карточки, помеченные 1 и 0, в коробку: она будет представлять предполагаемую совместную интенсивность конверсии 382 единиц и 45 945 нулей: $0,008246 = 0,8246\%$.
2. Перетасовать и извлечь повторную выборку размера 23 739 (число *n* такое же, что и у цены *A*), записать число единиц.
3. Записать число единиц в оставшихся 22 588 (число *n* такое же, что и у цены *B*).

4. Записать разницу в доле единиц.
5. Повторить шаги 2–4.
6. Ответить на вопрос: как часто наблюдалась разница $\geq 0,0368$?

Снова воспользовавшись функцией `perm_fun`, определенной в разделе *"Пример: прилипчивость веб-страниц"* ранее в этой главе, мы можем создать гистограмму случайно переставленных разниц в интенсивностях конверсии на языке R:

```
obs_pct_diff <- 100 * (200 / 23739 - 182 / 22588)
conversion <- c(rep(0, 45945), rep(1, 382))
perm_diffs <- rep(0, 1000)
for (i in 1:1000) {
  perm_diffs[i] = 100 * perm_fun(conversion, 23739, 22588)
}
hist(perm_diffs, xlab='Интенсивность конверсии, %', main='')
abline(v=obs_pct_diff)
```

Соответствующий код Python таков:

```
obs_pct_diff = 100 * (200 / 23739 - 182 / 22588)
print(f'Наблюдаемая разница: {obs_pct_diff:.4f}%')
conversion = [0] * 45945
conversion.extend([1] * 382)
conversion = pd.Series(conversion)

perm_diffs = [100 * perm_fun(conversion, 23739, 22588)
               for _ in range(1000)]

fig, ax = plt.subplots(figsize=(5, 5))
ax.hist(perm_diffs, bins=11, rwidth=0.9)
ax.axvline(x=obs_pct_diff, color='black', lw=2)
ax.text(0.06, 200, 'Наблюдаемая разница', bbox={'facecolor':'white'})
ax.set_xlabel('Интенсивность конверсии, %')
ax.set_ylabel('Частота')
```

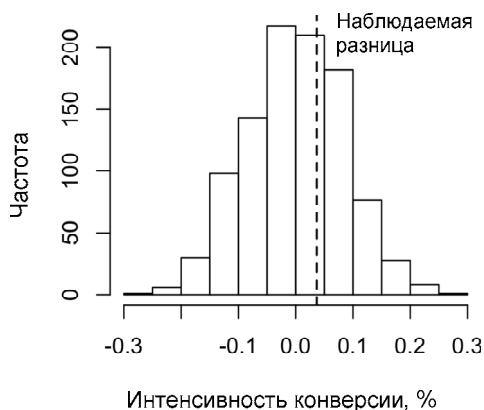


Рис. 3.5. Частотное распределение для разницы в интенсивностях конверсии между страницами A и B

Взгляните на гистограмму 1000 повторно отобранных результатов на рис. 3.5: как оказалось, в данном случае наблюдаемая разница 0,0368% лежит далеко внутри диапазона случайной вариации.

***p*-Значение**

Простой осмотр графика — не совсем точный способ измерить статистическую значимость, больший интерес все же вызывает *p*-значение. Это частота, с которой случайная модель производит результат, более предельный, чем наблюдаемый. Мы можем оценить *p*-значение из нашего перестановочного теста путем взятия доли числа раз, когда перестановочный тест порождает разницу, равную или большую, чем наблюдаемая разница:

```
mean(perm_diffs > obs_pct_diff)
[1] 0.308
```

```
np.mean([diff > obs_pct_diff for diff in perm_diffs])
```

Здесь и R, и Python используют тот факт, что истина интерпретируется как 1, а ложь — как 0.

p-Значение составляет 0,308, а значит, мы ожидаемо будем достигать такого же предельного результата, как и этот, или более предельного в силу случайности, превышающей 30% времени.

В данном случае нам не нужно было использовать перестановочный тест, чтобы получить *p*-значение. Поскольку у нас биномиальное распределение, мы можем аппроксимировать *p*-значение при помощи нормального распределения. Во фрагменте кода на R мы это делаем при помощи функции `prop.test`:

```
> prop.test(x=c(200,182), n=c(23739,22588), alternative="greater")

      2-sample test for equality of proportions with continuity correction

data:  c(200, 182) out of c(23739, 22588)
X-squared = 0.14893, df = 1, p-value = 0.3498
alternative hypothesis: greater
95 percent confidence interval:
 -0.001057439 1.000000000
sample estimates:
      prop 1      prop 2 
0.008424955 0.008057376
```

Аргумент *x* — это число успехов для каждой группы, аргумент *n* — число испытаний.

Метод `scipy.stats.chi2_contingency` принимает значения, показанные в табл. 3.2:

```
survivors = np.array([[200, 23739 - 200], [182, 22588 - 182]])
chi2, p_value, df, _ = stats.chi2_contingency(survivors)

print(f'p-значение для одностороннего теста: {p_value / 2:.4f}')
```

Нормальная аппроксимация производит p -значение 0,3498, которое близко расположено к p -значению, полученному в результате перестановочного теста.

Альфа

Специалисты в области статистики осуждают практику, когда на усмотрение исследователя оставляется решение о том, является или нет результат "слишком необычным", чтобы произойти случайно. Вместо этого заранее устанавливается порог, как, например, "более предельный, чем 5% случайных результатов (нулевая гипотеза)"; этот порог называется *альфой*. Типичными альфа-уровнями являются 5 и 1%. При этом выбор любого из двух является произвольным решением — в данной процедуре нет ничего, что будет гарантировать правильные решения в $x\%$ случаев. Это вызвано тем, что получаемый ответ на вопрос о вероятности состоит *не* в том, "какова вероятность, что он произошел случайно", а в том, "какова вероятность настолько предельного результата с учетом случайной модели". Затем мы делаем обратный логический вывод о правомерности случайной модели, но такое суждение не несет в себе вероятность. Этот момент всегда был источником часто возникающей путаницы.

Разногласия по поводу p -значения

В последние годы использование p -значения было окружено значительными дискуссиями. Один из журналов по психологии пошел насколько далеко, что "запретил" использование p -значений в предоставляемых ему статьях на том основании, что публикационные решения, основанные исключительно на p -значении, приводили к публикациям исследовательских работ плохого качества. Слишком много исследователей, лишь смутно представляющих, что в действительности представляет собой p -значение, копаются в данных и среди разных возможных гипотез, подлежащих проверке, пока не находят комбинацию, которая приводит к значимому p -значению и, следовательно, к работе, подходящей для публикации.

Реальная проблема состоит в том, что люди хотят получать от p -значения больше смысла, чем оно имеет. Вот что мы *хотели бы*, чтобы p -значение содержало:

Вероятность, что результат обусловлен случайностью.

Мы рассчитываем на низкое значение и поэтому можем заключить, что мы что-то доказали. Именно так многие редакторы журналов интерпретируют p -значение. Но вот что p -значение представляет *на самом деле*:

Вероятность, что с учетом случайной модели могут произойти настолько же предельные результаты, что и наблюдаемые.

Разница является тонкой, но реальной. Значимое p -значение не проводит вас далеко по дороге к "доказательству", как оно, похоже, обещает. Логический фундамент для заключения о "статистической значимости" несколько ослабевает, когда понятный смысл p -значения.

В марте 2016 года Американская статистическая ассоциация (American Statistical Association, ASA) после долгих внутренних дискуссий показала степень недоразу-

нения по поводу p -значений, когда выпустила предостережение относительно их использования. В заявлении ASA² были подчеркнуты шесть принципов для исследователей и редакторов журналов.

1. p -Значения могут указывать на то, насколько несовместимы данные с заданной статистической моделью.
2. p -Значения не измеряют вероятность, что изучаемая гипотеза является истинной, либо вероятность, что данные были произведены исключительно в силу случайности.
3. Научные выводы и деловые или стратегические решения не должны базироваться только на том, переходит ли p -значение определенный порог.
4. Надлежащий статистический вывод требует полной отчетности и прозрачности.
5. p -Значение, или статистическая значимость, не измеряет размер эффекта или важность результата.
6. Как таковое p -значение не обеспечивает хорошую меру фактических данных относительно модели или гипотезы.

Практическая значимость

Даже если результат является статистически значимым, это вовсе не означает, что он имеет практическую значимость. Малая разница, которая не имеет практического смысла, может быть статистически значимой, если она возникла из достаточно крупных выборок. Крупные выборки усиливают малые, несущественные эффекты настолько, чтобы исключить случайность в объяснении. Исключение случайности волшебным образом не делает результат важным, который по своей сути не является важным.

Ошибки 1-го и 2-го рода

В определении статистической значимости возможны два рода ошибок:

- ♦ ошибка 1-го рода, когда вы ошибочно заключаете, что эффект является реальным, тогда как в действительности он обусловлен чистой случайностью;
- ♦ ошибка 2-го рода, когда вы ошибочно заключаете, что эффект не является реальным (т. е. обусловлен случайностью), в то время как в действительности он является реальным.

Ошибка 2-го рода, в сущности, является не столько ошибкой, сколько суждением, что размер выборки слишком мал, чтобы можно было обнаружить эффект. Когда p -значение недотягивает до статистической значимости (например, оно превышает 5%), мы фактически хотим сказать, что "эффект не доказан". Может оказаться, что более крупная выборка произведет меньшее p -значение.

² См. <https://oreil.ly/WVfYU>.

Основная функция проверок значимости (также именуемых *проверками гипотез*) — защитить от того, чтобы быть обманутым случайностью; следовательно, они, как правило, структурируются таким образом, чтобы минимизировать ошибки 1-го рода.

Наука о данных и p -значения

Работа, которая выполняется исследователями данных, как правило, не предназначена для публикации в научных журналах, поэтому дебаты о смысле p -значения являются несколько академичными. Для исследователя данных p -значение — полезная метрика в ситуациях, где вы хотите знать, лежит ли модельный результат, который кажется интересным и полезным, в диапазоне нормальной случайной вариабельности. Как инструмент для принятия решения в эксперименте, p -значение следует считать не решающим показателем, а просто еще одной точкой информации, которая оказывает влияние на решение. Например, p -значения иногда используются в качестве промежуточных входов в некие статистические или автоматически обучающиеся модели — признак может быть включен в состав модели или исключен из нее в зависимости от его p -значения.

Ключевые идеи для статистической значимости и p -значения

- Проверки значимости используются для определения того, лежит ли наблюдаемый эффект внутри диапазона случайной вариации для модели нулевой гипотезы.
- p -Значение — это вероятность, которая приводит к результатам, таким же предельным, какими могут оказаться наблюдаемые результаты с учетом модели нулевой гипотезы.
- Значение альфы — это порог "необычности" в случайной модели нулевой гипотезы.
- Проверка значимости остается намного более релевантной для формализованной отчетности об исследовании, чем для науки о данных (но чья важность в последнее время начала угасать даже для первой).

Дополнительные материалы для чтения

- ♦ Статья "Фишер и 5%-ный уровень" (Stigler S. Fisher and the 5% Level // Chance. — 2008. — Vol. 21. — № 4. — P. 12) представляет собой короткий комментарий относительно книги "Статистические методы для научных работников" (Fisher R. Statistical Methods for Research Workers. — 1925), где делается акцент на 5%-ном уровне значимости.
- ♦ См. также *раздел "Проверки гипотез"* ранее в этой главе и упомянутые там дополнительные материалы для чтения.

Проверки на основе t -статистики

Существуют многочисленные типы проверок значимости, которые зависят от того, являются ли данные количественными или измеряемыми, сколько имеется выборок и что измеряется. Самой общепринятой является t -тест, или проверка на основе t -статистики, отсылающая к t -распределению Стьюдента, первоначально разработанному У. С. Госсетом для аппроксимации распределения одновыборочного среднего (см. раздел " t -Распределение Стьюдента" главы 2).

Ключевые термины для проверок на основе t -статистики

Проверочная статистика (test statistic)

Метрика для интересующей разницы или эффекта.

t -Статистика (t -statistic)

Стандартизированная версия часто встречающихся проверочных статистик, таких как среднее значение.

t -Распределение (t -distribution)

Эталонное распределение (в данном случае полученное из нулевой гипотезы), с которым может быть сопоставлена наблюдаемая t -статистика.

Все проверки значимости требуют, чтобы вы определили *проверочную статистику*, которая измеряет интересующий вас эффект и помогает установить, находится ли этот наблюдаемый эффект внутри диапазона нормальной случайной вариации. В тесте на основе повторного отбора (обсуждение процедуры перестановки см. в разделе "*Перестановочный тест*" ранее в этой главе) шкала данных не играет роли. Вы создаете эталонное распределение (нулевая гипотеза) непосредственно из самих данных и используете проверочную статистику как есть.

В 1920–1930-х годах, когда статистическая проверка гипотез была в стадии разработки, процедура случайной перетасовки данных несколько тысяч раз для выполнения теста на основе повторного отбора была невыполнимой. Специалисты-статистики обнаружили, что хорошей аппроксимацией перестановочного (перетасованного) распределения является t -тест, опирающийся на t -распределение Госсета. Он используется для очень распространенного двухвыборочного сравнения — A/B -теста, в котором данные являются числовыми. Но для того чтобы t -распределение можно было применять без привязки к какой-либо конкретной шкале данных, должна использоваться стандартизированная форма проверочной статистики.

Классический статистический учебник на данном этапе покажет различные формулы, которые инкорпорируют распределение Госсета и демонстрируют, каким образом выполняется стандартизация ваших данных с целью их сравнения со стандартным t -распределением. Эти формулы здесь не показаны, потому что все статистические вычислительные системы, а также R и Python содержат команды, которые воплощают эти формулы. В R это функция `t.test`:


```
> t.test(Time ~ Page, data=session_times, alternative='less' )
```

Welch Two Sample t-test

data: Time by Page

t = -1.0983, df = 27.693, p-value = 0.1408

alternative hypothesis: true difference in means is less than 0

95 percent confidence interval:

-Inf 19.59674

sample estimates:

mean in group Page A mean in group Page B

126.3333

162.0000

Функция `scipy.stats.ttest_ind` **может использоваться в Python:**

```
res = stats.ttest_ind(session_times[session_times.Page == 'Page A'].Time,  
                      session_times[session_times.Page == 'Page B'].Time,  
                      equal_var=False)
```

```
print(f'p-значение для одностороннего теста: {res.pvalue / 2:.4f}')
```

Альтернативная гипотеза состоит в том, что среднее время сеанса для страницы *A* меньше, чем для страницы *B*. Это достаточно близко к *p*-значению 0,124 перестановочного теста (см. раздел *"Пример: прилипчивость веб-страниц"* ранее в этой главе).

В режиме повторного отбора мы структурируем решение так, чтобы отразить наблюдаемые данные и подлежащую проверке гипотезу, не беспокоясь о том, являются ли данные числовыми или двоичными, сбалансированы ли размеры выборок или нет, дисперсии выборок либо множество других факторов. В мире формул многие вариации представляют самих себя, и они могут быть изумительными. Специалистам-статистикам приходится путешествовать по этому миру и изучать его карту, но аналитикам данных в этом нет необходимости — они, как правило, не занимаются выжимкой подробностей из проверок гипотез и доверительных интервалов, как это делает исследователь, который готовит научную работу к публикации.

Ключевые идеи для проверки на основе *t*-статистики

- До появления компьютеров проверки на основе повторного отбора не были практичными, и специалисты-статистики использовали стандартные эталонные распределения.
- Проверочная статистика затем может быть стандартизирована и сопоставлена с эталонным распределением.
- Одной из таких широко используемых стандартизированных статистик является *t*-статистика.

Дополнительные материалы для чтения

- ♦ Любой вводный курс статистики содержит иллюстрации t -статистики и ее использования; вот два хороших из них: "Статистика" (Freedman D., Pisani R., Purves R. Statistics. — 4th ed. — W.W. Norton & Company, 2007) и "Основополагающие принципы статистики" (Moore D. S. The Basic Practice of Statistics. — Palgrave Macmillan, 2010).
- ♦ По поводу параллельной трактовки процедур проверки на основе t -статистики и повторного отбора образцов обратитесь к книгам "Вводная статистика и аналитика: под углом повторного отбора" (Bruce P. Introductory statistics and analytics: a resampling perspective. — John Wiley & Sons, 2014) или "Статистика" (Lock R. et al. Statistics. — Wiley, 2012).

Множественное тестирование

Как мы упомянули ранее, в статистике существует высказывание: "Если мучить данные слишком долго, то рано или поздно они дадут признательные показания". Оно означает, что если смотреть на данные с очень большого числа разных точек зрения и задавать слишком много вопросов, то почти неизменно можно найти статистически значимый эффект.

Например, если имеются 20 предсказательных переменных и одна переменная исхода, и все они сгенерированы *случайным образом*, то имеются достаточно хорошие шансы на то, что по крайней мере один предсказатель (ложным образом) окажется статистически значимым, если выполнить серию из 20 проверок значимости с альфа на уровне 0,05. Как уже обсуждалось ранее, эта ситуация называется *ошибкой 1-го рода*. Вы можете рассчитать эту вероятность, сперва отыскав вероятность того, что все они пройдут проверку, *правильно* показав незначимость на уровне 0,05. Вероятность, что одна из них пройдет проверку, правильно показав незначимость, равна 0,95, поэтому вероятность того, что все 20 предсказателей пройдут проверку, правильно показав незначимость, будет равна $0,95 \times 0,95 \times 0,95 \dots$ или $0,95^{20} = 0,36^3$. Вероятность того, что по крайней мере один предсказатель (ложным образом) покажет значимость, является обратной этой вероятности, или 1 (*вероятность того, что все будут незначимыми*), равна 0,64. Указанное явление известно как *инфляция альфы*.

Этот вопрос связан с проблемой перепогонки в добыче регулярностей из данных, или "подгонкой модели к шуму". Чем больше переменных вы добавляете или больше моделей выполняете, тем больше вероятность того, что нечто проявится как "значимое" просто по чистой случайности.

³ Правило умножения вероятностей утверждает, что вероятность одновременного наступления n независимых событий является произведением индивидуальных вероятностей. Например, если вы и я каждый подбросим монету один раз, то вероятность, что Ваша и моя монеты обе повернутся орлом, равна $0,5 \times 0,5 = 0,25$.

Ключевые термины для множественного тестирования

Ошибка 1-го рода (type 1 error)

Ошибочный вывод, что эффект является статистически значимым.

Коэффициент ложных открытий (false discovery rate)

Доля совершения ошибки 1-го рода в результате множественного тестирования.

Инфляция альфы (alpha inflation)

Явление множественного тестирования, в котором альфа, вероятность ошибки 1-го рода, увеличивается по мере проведения большего числа тестов.

Корректировка p -значений (adjustment of p -values)

Уточнение значения при выполнении множественного тестирования на одинаковых данных.

Перепогонка (overfitting)

Подгонка к шуму.

В задачах контролируемого самообучения контрольный набор с отложенными данными, где модели определяются на данных, которые модель не видела раньше, снижает этот риск. В задачах статистического и автоматического обучения, не сопряженных с помеченным отложенным набором, сохраняется риск прихода к заключениям, основанным на статистическом шуме.

В статистике есть несколько процедур, предназначенных для преодоления этой проблемы при очень специфических обстоятельствах. Например, при сравнении результатов по многочисленным контрольным группам можно задавать многочисленные вопросы. Так, для вариантов $A-C$ вы могли бы спросить:

- ◆ Отличается ли A от B ?
- ◆ Отличается ли B от C ?
- ◆ Отличается ли A от C ?

Или же в клиническом испытании вы, возможно, захотите посмотреть на результаты терапии на нескольких этапах. В каждом случае вы задаете многочисленные вопросы и с каждым вопросом увеличиваете шанс, что будете обмануты случайностью. Корректировочные процедуры в статистике могут это компенсировать путем установки более строгой планки для статистической значимости, чем планка для одной проверки гипотезы. Такие корректировочные процедуры, как правило, предусматривают "деление альфы" по числу проверок. Данный метод приводит к меньшему альфа-уровню (т. е. более строгой планке для статистической значимости) для каждой проверки. Одна такая процедура, корректировка Бонферрони, просто делит альфу на число сравнений. Еще одна, используемая при сравнении нескольких групповых средних, называется "честной значимой разницей" Тьюки, или *HSD Тьюки* (honest significant difference). Этот тест применяют к максимальной разнице между групповыми средними, сравнивая ее с эталоном, основанным на

t-распределении (примерно эквивалентном перетасовке всех значений, выделению повторно отобранных групп тех же размеров, что и исходные группы, и отысканию максимальной разницы между средними значениями повторно отобранных групп).

Однако проблема множественных сравнений выходит за пределы этих высокоструктурированных случаев и связана с феноменом многократного "прочесывания" данных, которое дает начало высказыванию об издевательстве над данными. Говоря иначе, если вы располагаете достаточно сложными данными и не нашли в них ничего интересного, значит, вы просто не всматривались в них долго и внимательно. Сегодня, как никогда ранее, доступно все больше данных. Так, число опубликованных между 2002 и 2010 годами журнальных статей почти удвоилось. А это дает начало массе возможностей найти что-то интересное в данных, включая вопросы множественности, такие как:

- ◆ сравнение многочисленных попарных разниц по всем группам;
- ◆ рассмотрение результатов многочисленных подгрупп ("мы не нашли значимого эффекта варианта по совокупности, но зато мы нашли эффект для незамужних женщин моложе 30");
- ◆ испытание большого числа статистических моделей;
- ◆ включение большого числа переменных в модели;
- ◆ постановка большого числа разных вопросов (т. е. разных возможных исходов).



Коэффициент ложных открытий

Термин "коэффициент ложных открытий" первоначально использовался для описания интенсивности, с которой данный набор проверок гипотез ложно выявляет значимый эффект. Он стал в особенности полезным с появлением геномных исследований, в которых могут проводиться массовые статистические проверки в рамках проекта генетического секвенирования. В этих случаях данный термин применяется к протоколу тестирования, и отдельное ложное "открытие" связано с исходом проверки гипотезы (например, между двумя выборками). Исследователи стремились установить параметры процесса тестирования так, чтобы удерживать коэффициент ложных открытий на заданном уровне. Этот термин также использовался в добыче регулярностей из данных в контексте частоты ошибок неправильной классификации внутри предсказаний класса 1. Или, другими словами, это вероятность, что "открытие" (закрепление за записью метки "1") является ложным. Здесь мы, как правило, имеем дело со случаем, когда нули в избытке, а нас интересуют единицы, но их мало (см. главу 5 и раздел "Проблема редкого класса" главы 5).

По ряду причин, в особенности включая общую проблему "множественности", дополнительные исследования не обязательно означают более качественные исследования. Например, сотрудники фармацевтической компании Bayer в 2011 году обнаружили, что при попытке репликации 67 научных исследований они смогли полностью реплицировать только 14 из них. Почти 2/3 невозможно было реплицировать вообще.

В любом случае корректировочные процедуры для подробно определенных и структурированных статистических проверок являются слишком специфичными

и негибкими, чтобы найти широкое применение у исследователей данных. Сухой остаток для исследователей данных в отношении множественности таков:

- ◆ для предсказательного моделирования риск получения иллюзорной модели, очевидная эффективность которой во многом является продуктом случайности, уменьшается за счет перекрестного контроля (см. раздел "Перекрестный контроль" главы 4) и использования отложенной выборки;
- ◆ для других процедур без помеченного отложенного набора, предназначенного для проверки работы модели, вы должны полагаться на:
 - осознание того, что чем больше вы опрашиваете данные и ими манипулируете, тем больше шансов на то, что в игру вступит случай;
 - эвристики на основе повторного отбора и симуляции с целью обеспечения случайных эталонов, с которыми могут сопоставляться наблюдаемые результаты.

Ключевые идеи для множественного тестирования

- Множественность в исследовательской работе или проекте добычи регулярностей из данных (многократные сравнения, много переменных, много моделей и т. д.) увеличивает риск сделать вывод о том, что нечто является значимым в силу чистой случайности.
- Для ситуаций, сопряженных с множественными статистическими сравнениями (т. е. множественными проверками значимости), существуют статистические корректировочные процедуры.
- В ситуации добычи регулярностей из данных использование отложенной выборки с помеченными переменными исхода помогает предотвращать неверные результаты.

Дополнительные материалы для чтения

- ◆ Краткое изложение одной из процедур (тест Даннета) корректировки под многочисленные сравнения см. в онлайн-овом статистическом учебнике⁴ Дэвида Лэйна.
- ◆ Меган Голдмэн (Megan Goldman) предлагает немного более длинную трактовку процедуры корректировки Бонферрони⁵.
- ◆ Книга "Множественная проверка на основе повторного отбора" (Westfall P. H., Young S. S. Resampling-based multiple testing: examples and methods for p-value adjustment. — Wiley, 1993) содержит всестороннюю трактовку более гибких статистических процедур корректировки p -значений.

⁴ См. https://oreil.ly/hd_62.

⁵ См. <https://oreil.ly/Dt4Vi>.

- ◆ Обсуждение темы разделения данных и использования отложенных выборок в предсказательном моделировании см. в главе 2 книги "Добыча регулярностей из данных для бизнес-аналитики" (Shmueli G., Bruce P., Patel N. Data mining for business analytics. — 3rd ed. — John Wiley & Sons, 2016) с изданиями для R, Python, Excel и JMP (Wiley, 2007–2020).

Степени свободы

В документации и условиях по многим статистическим проверкам и распределениям вероятностей можно увидеть отсылки на "степени свободы". Указанное понятие применяется к статистикам, вычисляемым из выборочных данных, и относится к числу значений, которые могут варьироваться свободно. Например, если вы знаете среднее для выборки из 10 значений, то у вас имеется 9 степеней свободы (как только вы знаете 9 значений выборки, 10-е может быть рассчитано и не может меняться свободно). Параметр степеней свободы, применяемый ко многим распределениям вероятностей, влияет на форму распределения.

Число степеней свободы является входом во многие статистические проверки. Например, степени свободы — это название, которое дано знаменателю $n - 1$, встречавшемуся при расчетах дисперсии и стандартного отклонения. В чем важность этого параметра? Выполняя обработку выборки с целью оценить дисперсию для популяции, вы в итоге придете к оценке, которая, если использовать в знаменателе n , будет немного смещена вниз. Если же знаменателе вы используете в $n - 1$, то оценка будет свободна от этого смещения.

Ключевые термины для степени свободы

n или размер выборки (n or sample size)

Число наблюдений (строк или записей) в данных.

d.f.

Степени свободы (degrees of freedom).

Значительная доля традиционного курса или учебника статистики отводится на разнообразные стандартные проверки гипотез (проверки на основе t -статистики, F -статистики и т. д.). Когда выборочные статистики стандартизированы для использования в традиционных статистических формулах, степени свободы являются частью расчетов по стандартизации с целью обеспечения того, чтобы ваши стандартизированные данные соответствовали надлежащему эталонному распределению (t -распределению, F -распределению и т. д.).

Важно ли это для науки о данных? Не совсем, по крайней мере в контексте проверки значимости. С одной стороны, формальные статистические проверки используются в науке о данных довольно экономно. С другой — размер данных обычно является крупным настолько, что для исследователя данных редко важна разница, использу-

ется ли в знаменателе n или $n-1$. (По мере того, как n становится все крупнее, смещение, которое возникло бы от использования n в знаменателе, исчезает.)

Тем не менее существует один контекст, в котором степени свободы остаются релевантными: применение факторизованных переменных в регрессии (включая логистическую регрессию). Регрессионные алгоритмы спотыкаются, если присутствуют строго избыточные предсказательные переменные. Это чаще всего происходит при факторизации категориальных переменных в двоичные индикаторы (фиктивные переменные). Возьмем переменную "день недели". Несмотря на то, что в неделе 7 дней, будет всего шесть степеней свободы при указании дня недели. Например, как только вы знаете, что день недели не понедельник по субботу, то вы также знаете, что этот день должен быть воскресеньем. Таким образом, включение индикаторов Пн–Сб означает, что включение еще и Вс станет причиной сбоя регрессии вследствие *ошибки мультиколлинеарности*.

Ключевые идеи для степеней свободы

- Число степеней свободы (d.f.) формирует часть расчетов по стандартизации проверочных статистик, в результате которой они могут сопоставляться с эталонными распределениями (t -распределением, F -распределением и т. д.).
- Понятие степеней свободы лежит в основе факторизации категориальных переменных в $n-1$ индикаторных или фиктивных переменных при выполнении регрессии (во избежание мультиколлинеарности).

Дополнительные материалы для чтения

По теме степеней свободы есть несколько онлайн-учебных пособий⁶.

Дисперсионный анализ

Предположим, что вместо A/B -теста мы сравнивали многочисленные группы, скажем A , B , C и D , каждая из которых содержит числовые данные. Статистическая процедура, которая выполняет проверку на статистически значимую разницу среди групп, называется *дисперсионным анализом* (ANalysis Of VAriance, ANOVA).

Ключевые термины для дисперсионного анализа

Попарное сравнение (pairwise comparison)

Проверка гипотезы (к примеру, о средних значениях) между двумя группами из множества групп.

⁶ См. <https://oreil.ly/VJyts>.

Универсальный тест (omnibus test)

Одна-единственная проверка гипотезы о совокупной дисперсии среди средних значений множества групп.

Синоним: омнибус-тест.

Разложение дисперсии (decomposition of variance)

Выделение компонентов, которые вносят вклад в отдельное значение (к примеру, из совокупного среднего, из среднего значения варианта эксперимента и из остаточной ошибки).

F-статистика (F-statistic)

Стандартизированная статистика, измеряющая степень, с которой разницы в групповых средних превышают то, что можно ожидать в случайной модели.

Сумма квадратов (SS, sum of squares)

Обозначает отклонения от какого-либо среднего значения.

В табл. 3.3 показана прилипчивость (т. е. степень удержания внимания посетителями) четырех веб-страниц, определенная как число секунд, проведенных посетителем на веб-странице. Четыре веб-страницы подменяются, чтобы каждый посетитель веб-сайта получал одну из них в случайном порядке. В общей сложности имеется пять посетителей на каждую страницу, и в табл. 3.3 каждый столбец представляет независимый набор данных. Первый посетитель страницы 1 никак не связан с первым посетителем страницы 2. Обратите внимание, что в такого рода веб-тесте мы не можем полностью имплементировать классический план рандомизированного отбора, в котором каждый посетитель отбирается в случайном порядке из некоторой огромной популяции. Мы должны принимать посетителей по мере их прибытия. Посетители могут систематически различаться в зависимости от времени суток, дня недели, времени года, условий интернет-соединения, характера используемого ими устройства и т. д. Эти факторы следует рассматривать как потенциальное смещение, когда результаты эксперимента пересматриваются.

Таблица 3.3. Прилипчивость (в секундах) для четырех веб-страниц

	Страница 1	Страница 2	Страница 3	Страница 4
	164	178	175	155
	172	191	193	166
	177	182	171	164
	156	185	163	170
	195	177	176	168
Среднее	172	185	176	162
Общее среднее				173,75

Теперь возникает головоломка (рис. 3.6). Когда мы сравнивали только две группы, все было просто; мы смотрели всего лишь на разницу между средними каждой группы. В условиях четырех средних существует шесть возможных сравнений между группами:

- ◆ страница 1 по сравнению со страницей 2;
- ◆ страница 1 по сравнению со страницей 3;
- ◆ страница 1 по сравнению со страницей 4;
- ◆ страница 2 по сравнению со страницей 3;
- ◆ страница 2 по сравнению со страницей 4;
- ◆ страница 3 по сравнению со страницей 4.

Чем больше мы делаем таких *парных* сравнений, тем больше потенциал для того, чтобы оказаться обманутым случайностью (см. раздел "Множественное тестирование" ранее в этой главе). Вместо того чтобы беспокоиться обо всех возможных сравнениях между отдельными страницами, которые мы могли бы провести, можно выполнить всего один совокупный тест, который дает ответ на вопрос: могут ли все страницы иметь в своей основе одинаковую прилипчивость, и различия между этими страницами быть обусловлены случайным образом, которым общий набор времен сеансов распределялся между четырьмя страницами?

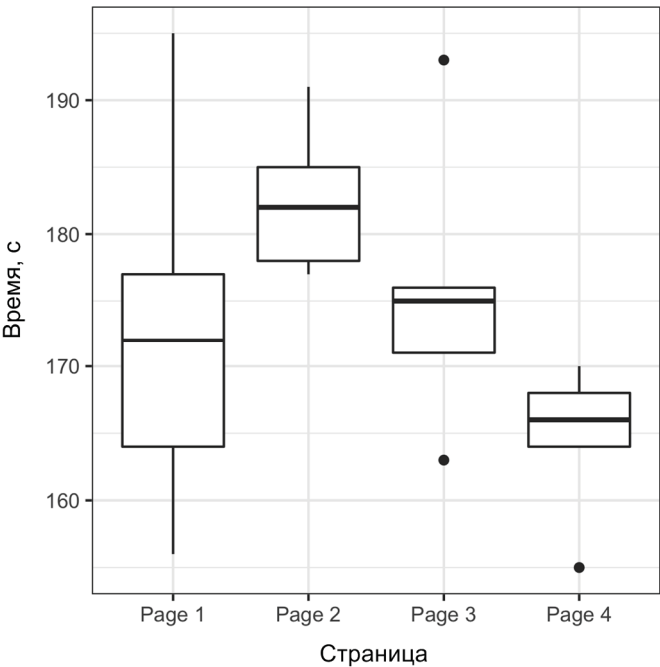


Рис. 3.6. Коробчатые диаграммы для четырех групп демонстрируют существенные различия среди страниц

Для проверки этого вопроса используется процедура дисперсионного анализа. Основополагающие принципы ее применения можно увидеть в следующей ниже процедуре повторного отбора (заданной здесь для *A–B–C–D*-теста прилипчивости веб-страниц):

1. Объединить все данные в одной коробке.
2. Перетасовать и извлечь четыре повторные выборки с пятью значениями в каждой.
3. Записать среднее значение каждой из четырех групп.
4. Записать дисперсию среди средних значений четырех групп.
5. Повторить шаги 2–4 много раз (скажем, 1000).

Какова доля случаев, когда повторно отобранная дисперсия превышала наблюдаемую дисперсию? Этот вопрос подразумевает *p*-значение.

Указанный тип перестановочного теста чуть более замысловат, чем тот, который использовался в разделе *"Перестановочный тест"* ранее в этой главе. К счастью, в пакете `lmPerm` имеется функция `aovp`, которая вычисляет перестановочный тест для этого случая:

```
> library(lmPerm)
> summary(aovp(Time ~ Page, data=four_sessions))
[1] "Settings: unique SS "
Component 1 :
      Df R Sum Sq R Mean Sq Iter Pr(Prob)
Page      3    831.4    277.13 3104  0.09278 .
Residuals 16   1618.4    101.15
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

p-Значение, показанное в `Pr(Prob)`, равно 0,09278. Другими словами, имея ту же самую опорную прилипчивость, в 9,3% случаев интенсивность откликов среди четырех страниц могла отличаться в той же мере, в какой фактически наблюдалась, по чистой случайности. Эта степень невероятности недотягивает до традиционного статистического порога в 5%, поэтому мы заключаем, что разница между четырьмя страницами могла возникнуть случайно.

Столбец `Iter` выводит число итераций, потребовавшихся в перестановочном тесте. Другие столбцы соответствуют традиционной таблице дисперсионного анализа и описаны далее.

В Python мы можем вычислить тест перестановки, используя следующий код:

```
observed_variance = four_sessions.groupby('Page').mean().var()[0]
print('Наблюдаемые средние:', four_sessions.groupby('Page').mean().values.ravel())
print('Дисперсия:', observed_variance)

def perm_test(df):
    df = df.copy()
```

```
df['Time'] = np.random.permutation(df['Time'].values)
return df.groupby('Page').mean().var()[0]

perm_variance = [perm_test(four_sessions) for _ in range(3000)]
print('Pr(Prob)', np.mean([var > observed_variance for var in perm_variance]))
```

F-статистика

Аналогично проверке на основе t -статистики, которая может использоваться вместо перестановочного теста для сравнения средних значений двух групп, для дисперсионного анализа существует статистическая проверка на основе *F-статистики*. Указанная F -статистика опирается на отношение дисперсии по всем групповым средним (т. е. вариантного эффекта) к дисперсии вследствие остаточной ошибки. Чем выше это отношение, тем более статистически значимым является результат. Если данные подчиняются нормальному распределению, то статистическая теория обязывает статистику иметь некое распределение. На этом основании есть возможность вычислить p -значение.

В R мы можем вычислить таблицу дисперсионного анализа, используя функцию `aov`:

```
> summary(aov(Time ~ Page, data=four_sessions))
          Df Sum Sq Mean Sq F value Pr(>F)
Page         3   831.4    277.1    2.74 0.0776 .
Residuals   16 1618.4    101.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Пакет `statsmodels` предоставляет имплементацию процедуры дисперсионного анализа на Python:

```
model = smf.ols('Time ~ Page', data=four_sessions).fit()

aov_table = sm.stats.anova_lm(model)
aov_table
```

Результат работы кода Python практически идентичен результату на языке R.

Df — это степени свободы, Sum Sq — сумма квадратов, Mean Sq — средние квадраты (аббревиатура для среднеквадратических отклонений), F value — F -статистика. Для общего среднего сумма квадратов — это отступление общего среднего от 0, возведенное в квадрат и умноженное на 20 (по числу наблюдений). Параметр степеней свободы для общего среднего по определению равно 1.

Для вариантных средних значений степени свободы равны 3 (после того как определены три значения, а затем определено общее среднее, другое вариантное среднее варьироваться не может). Сумма квадратов для вариантных средних — это сумма квадратичных отступлений между вариантными средними и общим средним.

Для остатков степени свободы равны 20 (все наблюдения могут варьироваться), и SS — это сумма квадратичных разниц между отдельными наблюдениями и вари-

антными средними. Средние квадраты (MS) — это сумма квадратов, деленная на степени свободы.

F -статистика равна $MS(\text{вариант})/MS(\text{ошибка})$. Таким образом, значение F зависит только от данного отношения и может быть сопоставлено со стандартным F -распределением с целью установления, являются ли разницы в вариантных средних больше, чем ожидается в случайной вариации.



Разложение дисперсии

Наблюдаемые значения в наборе данных можно рассматривать как суммы, состоящие из разных компонентов. Любое наблюдаемое внутри набора данных значение можно разбить на общее среднее, вариантный эффект и остаточную ошибку. Данная процедура называется "разложением дисперсии".

1. Начать с общего среднего (173,75 для данных прилипчивости веб-страниц).
2. Добавить вариантный эффект, который может быть отрицательным (независимая переменная = веб-страница).
3. Добавить остаточную ошибку, которая может отрицательной.

Таким образом, разложение дисперсии для левого верхнего значения в таблице $A-B-C-D$ -теста таково:

1. Начать с общего среднего: 173,75.
2. Добавить вариантный (групповой) эффект: $-1,75$ ($172 - 173,75$).
3. Добавить остаток: -8 ($164 - 172$).
4. Получаем: 164.

Двухсторонний дисперсионный анализ

Только что описанный $A-B-C-D$ -тест является односторонним дисперсионным анализом, в котором мы имеем один варьирующий фактор (группу). Но мы могли бы привлечь второй фактор, скажем "выходной день по сравнению с будним днем", где данные собираются по каждой комбинации (выходной день группы A , будний день группы A , выходной день группы B и т. д.). Это был бы двухсторонний дисперсионный анализ, и мы будем работать с ним точно так же, как и с односторонним дисперсионным анализом путем выявления "эффекта взаимодействия". После выявления общесреднего эффекта и вариантного эффекта мы выделяем наблюдения, относимые к выходным и будним дням, для каждой группы и отыскиваем разницу между средними для этих подмножеств и вариантным средним.

Можно видеть, что регулярный дисперсионный анализ и затем двухсторонний дисперсионный анализ являются первыми шагами по дороге к полной статистической модели, такой как регрессия и логистическая регрессия, в которой могут быть смоделированы многочисленные факторы и их эффекты (см. главу 4).

Ключевые идеи для дисперсионного анализа

- Дисперсионный анализ (ANOVA) — это статистическая процедура для анализа результатов эксперимента с многочисленными группами.
- Указанная процедура является расширением аналогичных процедур для *A/B*-теста, используемая для определения, находится ли совокупная вариация среди групп внутри диапазона случайной вариации.
- Полезным результатом дисперсионного анализа служит выявление дисперсионных компонентов, ассоциированных с групповыми вариантами, эффектами взаимодействия и ошибками.

Дополнительные материалы для чтения

- ♦ "Вводная статистика и аналитика: под углом повторного отбора" (Bruce P. Introductory statistics and analytics: a resampling perspective. — John Wiley & Sons, 2014) содержит главу по ANOVA.
- ♦ "Введение в планирование и анализ экспериментов" (Cobb G. Introduction to design and analysis of experiments. — John Wiley & Sons, 2008) предлагает всестороннюю и легко читаемую трактовку заявленной в заголовке темы.

Проверка на основе статистики хи-квадрат

Веб-тестирование часто выходит за рамки *A/B*-тестирования и проверяет сразу несколько вариантов эксперимента. Проверку на основе статистики *хи-квадрат* используют с количественными данными, чтобы проверить, насколько хорошо они укладываются в ожидаемое распределение. В статистической практике широко принято использовать статистику *хи-квадрат* вместе с таблицами сопряженности $r \times c$, чтобы определить, является ли разумной нулевая гипотеза о независимости среди переменных (см. также раздел "*Распределение хи-квадрат*" главы 2).

Проверка на основе статистики *хи-квадрат* первоначально была разработана Карлом Пирсоном в 1900 году⁷. Термин "*хи*" происходит от греческой буквы χ , которую Пирсон использовал в своей статье.

Ключевые термины для проверки на основе статистики хи-квадрат

Статистика хи-квадрат (chi-square statistic)

Метрика, измеряющая степень, с которой наблюдаемые данные отступают от ожидания.

Ожидание или ожидаемое (expectation or expected)

Поведение данных, по нашим ожиданиям, в условиях некоего допущения, как правило, нулевой гипотезы.

⁷ См. <https://oreil.ly/cEubO>.



$r \times c$ означает "число строк на число столбцов" — таблица размера 2×3 имеет две строки и три столбца.

Проверка хи-квадрат: подход на основе повторного отбора

Предположим, что вы тестируете три разных заголовка объявления — A , B и C , причем каждый из них — на 1000 посетителях; результаты сведены в табл. 3.4.

Таблица 3.4. Результаты веб-тестирования для трех разных заголовков

	Заголовок А	Заголовок В	Заголовок С
Нажатия	14	8	12
Нет нажатий	986	992	988

Заголовки, без всякого сомнения, выглядят разными. Заголовок A дает почти в два раза больше нажатий, чем заголовок B . Правда, фактические числа небольшие. Процедура повторного отбора может проверить, отличается ли процент нажатий в еще большей степени, чем может быть вызвано случайностью. Для такой проверки мы должны располагать "ожидаемым" распределением нажатий, и в этом случае проверка будет осуществляться на основе допущения нулевой гипотезы о том, что всем трем заголовкам присущ один и тот же процент нажатий при совокупном проценте нажатий, равном $34/3000$. В условиях этого допущения наша таблица сопряженности будет выглядеть как табл. 3.5.

Таблица 3.5. Ожидается, если все три заголовка имеют одинаковый процент нажатий (нулевая гипотеза)

	Заголовок А	Заголовок В	Заголовок С
Нажатия	11,33	11,33	11,33
Нет нажатий	988,67	988,67	988,67

Остаток Пирсона задается формулой

$$R = \frac{\text{наблюдаемое} - \text{ожидаемое}}{\sqrt{\text{ожидаемое}}}$$

и показывает, насколько фактические количества отличаются от их ожидаемых количеств (табл. 3.6).

Таблица 3.6. Пирсоновские остатки

	Заголовок А	Заголовок В	Заголовок С
Нажатия	0,792	-0,990	0,198
Нет нажатий	-0,085	0,106	-0,021

Статистика хи-квадрат задается как сумма квадратичных пирсоновских остатков:

$$\chi^2 = \sum_i^r \sum_j^c R^2,$$

где r и c — это соответственно число строк и столбцов. Статистика хи-квадрат для данного примера составит 1,666. Действительно ли это больше, чем может разумно произойти в случайной модели?

Мы можем проверить это при помощи следующего ниже алгоритма повторного отбора:

1. Положить в коробку 34 единиц (нажатия) и 2966 нулей (нажатия отсутствуют).
2. Перетасовать, извлечь три отдельные выборки по 1000 элементов и подсчитать нажатия в каждом.
3. Найти квадратичные разницы между перетасованными количествами и ожидаемыми количествами и их просуммировать.
4. Повторить шаги 2 и 3, скажем, 1000 раз.
5. Как часто повторно отобранная сумма квадратичных отклонений превышает наблюдаемые? Этот показатель будет p -значением.

Функция `chisq.test` может использоваться для вычисления повторно отобранной статистики хи-квадрат. Для данных с нажатиями проверка хи-квадрат будет такой:

```
> chisq.test(clicks, simulate.p.value=TRUE)
```

```
Pearson's Chi-squared test with simulated p-value (based on 2000 replicates)
```

```
data: clicks
```

```
X-squared = 1.6659, df = NA, p-value = 0.4853
```

Указанная проверка показывает, что этот результат мог легко быть получен в силу случайности.

Для того чтобы выполнить перестановочный тест на языке Python, используйте следующую ниже имплементацию:

```
box = [1] * 34
box.extend([0] * 2966)
random.shuffle(box)

def chi2(observed, expected):
    pearson_residuals = []
    for row, expect in zip(observed, expected):
        pearson_residuals.append([(observe - expect) ** 2 / expect
                                   for observe in row])
    # вернуть сумму квадратов
    return np.sum(pearson_residuals)
```

```

expected_clicks = 34 / 3
expected_noclicks = 1000 - expected_clicks
expected = [34 / 3, 1000 - 34 / 3]
chi2observed = chi2(clicks.values, expected)

def perm_fun(box):
    sample_clicks = [sum(random.sample(box, 1000)),
                     sum(random.sample(box, 1000)),
                     sum(random.sample(box, 1000))]
    sample_noclicks = [1000 - n for n in sample_clicks]
    return chi2([sample_clicks, sample_noclicks], expected)

perm_chi2 = [perm_fun(box) for _ in range(2000)]
resampled_p_value = sum(perm_chi2 > chi2observed) / len(perm_chi2)

print(f'Наблюдаемое х-квадрат: {chi2observed:.4f}')
print(f'Повторно отобранное р-значение: {resampled_p_value:.4f}')
```

Проверка хи-квадрат: статистическая теория

Асимптотическая статистическая теория показывает, что распределение статистики хи-квадрат может быть аппроксимировано *распределением хи-квадрат* (см. "Распределение хи-квадрат" главы 2). Надлежащее стандартное распределение хи-квадрат определяется *степенями свободы* (см. раздел "Степени свободы" ранее в этой главе). Для таблицы сопряженности степени свободы связаны с числом строк (r) и столбцов (c) следующим образом:

$$\text{степени свободы} = (r - 1)(c - 1).$$

Распределение хи-квадрат в типичной ситуации имеет асимметричный вид, длинный хвост которого скошен вправо (см. рис. 3.7, где показано распределение с 1, 2, 5 и 10 степенями свободы). Чем дальше на распределении хи-квадрат находится наблюдаемая статистика, тем ниже p -значение.

Функция `chisq.test` может применяться для вычисления p -значения с использованием распределения хи-квадрат в качестве эталона:

```
> chisq.test(cticks, simulate.p.value=FALSE)
```

```
Pearson's Chi-squared test test
```

```
data: clicks
X-squared = 1.6659, df = 2, p-value = 0.4348
```

В Python используйте функцию `scipy.stats.chi2_contingency`:

```
chisq, pvalue, df, expected = stats.chi2_contingency(clicks)
print(f'Наблюдаемое хи-квадрат: {chi2observed:.4f}')
print(f'p-значение: {pvalue:.4f}')
```

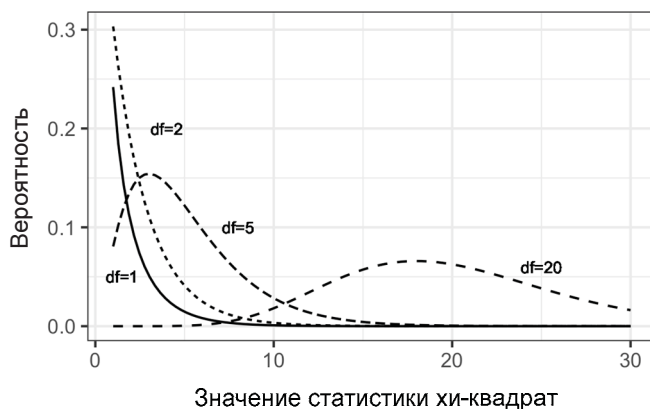



Рис. 3.7. Распределение хи-квадрат с различными степенями свободы

Полученное p -значение немного меньше, чем p -значение повторного отбора: это вызвано тем, что распределение хи-квадрат является всего лишь аппроксимацией фактического распределения статистики.

Точный тест Фишера

Распределение хи-квадрат является хорошей аппроксимацией только что описанной проверки на основе перетасованного повторного отбора, кроме тех случаев, когда количества являются чрезвычайно низкими (однозначными, в особенности пять или меньше). В таких случаях процедура на основе повторного отбора будет давать более точные p -значения. На деле в большинстве статистических вычислительных систем имеется процедура для фактического перечисления всех возможных перестановок, которые могут произойти, сводит в таблицу их частоты и точно определяет, насколько предельным является наблюдаемый результат. Эта процедура называется *точным тестом Фишера* в честь великого статистика Р. А. Фишера. Исходный код на R для точной проверки Фишера в своей канонической форме очень прост:

```
> fisher.test(cticks)
```

```
Fisher's Exact Test for Count Data
```

```
data: clicks
p-value = 0.4824
alternative hypothesis: two.sided
```

Это p -значение очень близко к p -значению 0,4853, полученному с использованием метода повторного отбора.

Там, где одни количества являются очень низкими, а другие — довольно высокими (например, знаменатель в степени конверсии), может возникнуть необходимость выполнить перетасованный перестановочный тест вместо полного точного теста из-за трудности вычисления всех возможных перестановок. Ранее упомянутая

функция R имеет несколько аргументов, которые управляют тем, использовать или нет эту аппроксимацию (`simulate.p.value=TRUE` или `FALSE`), сколько следует использовать итераций (`B=...`), и задают вычислительное ограничение (`workspace=...`) на то, насколько далеко должны зайти расчеты для получения точного результата.

Обнаружение мошенничества в науке

Интересный пример представлен делом об исследователе Университета Тафтса Терезой Иманиси-Кари (Thereza Imanishi-Kari), которая в 1991 году обвинялась в фабрикации данных своей научной работы. В этом оказался замешанным конгрессмен Джон Динджелл (John Dingell), и данный случай в конечном счете привел к отставке ее коллеги, Дэвида Балтимора (David Baltimore), от президентства в Рокфеллеровском университете.

Один из элементов в данном деле базировался на статистических уликах относительно ожидаемого распределения цифр в ее лабораторных данных, где каждое наблюдение имело много цифр. Следователи сосредоточились на внутренних цифрах, которые ожидаемо должны были подчиняться равномерному распределению случайной величины. Иными словами, они должны появляться в случайном порядке, где каждая цифра имеет равную вероятность появления (первая цифра может иметь преимущественно одно значение, и заключительные цифры могут быть затронуты округлением). В табл. 3.7 перечислены частоты внутренних цифр из фактических данных в этом деле.

Таблица 3.7. Центральная цифра в лабораторных данных

Цифра	Частота	Цифра	Частота
0	14	5	19
1	71	6	12
2	7	7	45
3	65	8	53
4	23	9	6

Распределение 315 цифр, показанное на рис. 3.8, конечно же выглядит неслучайным.

Следователи рассчитали отклонение от ожидания (31,5, т. е. как часто каждая цифра появлялась в строго равномерном распределении), и использовали проверку χ^2 (в равной степени могла быть использована процедура повторного отбора), чтобы показать, что фактическое распределение было далеко за пределами диапазона нормальной случайной вариации.

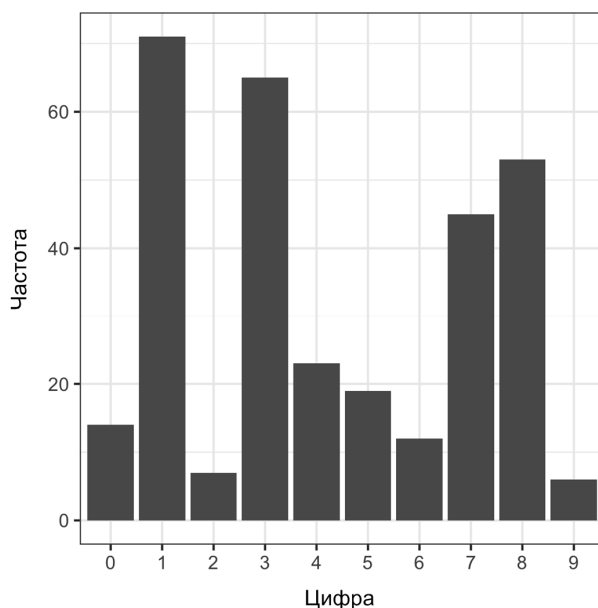


Рис. 3.8. Частотная гистограмма для лабораторных данных Иманиси-Кари

Релевантность для науки о данных

Проверка хи-квадрат или точный тест Фишера используют, когда вы хотите знать о том, является ли эффект реальным, или же он может быть продуктом случайности. В большинстве стандартных применений проверки хи-квадрат ее роль заключается в установлении статистической значимости, что в типичной ситуации необходимо до того, как исследование или эксперимент могут быть опубликованы. Это не особо важно для науки о данных. В большинстве экспериментов в науке о данных, будь то A/B или $A/B/C...$, цель состоит не в том, чтобы просто установить статистическую значимость, а в том, чтобы прийти к лучшему варианту эксперимента. С этой целью многорукие бандиты (см. раздел "Алгоритм многорукого бандита" далее в этой главе) предлагают более полное решение.

Одно из применений проверки хи-квадрат в науке о данных, и в особенности ее точной версии — теста Фишера, находится в определении надлежащих размеров выборок для веб-экспериментов. Такие эксперименты часто имеют очень низкие проценты нажатий, и, несмотря на тысячи показов, уровни количеств могут быть слишком малыми, чтобы делать категорические заключения в эксперименте. В таких случаях точный тест Фишера, проверка хи-квадрат и другие проверки могут быть полезными как составной компонент расчета мощности и размера выборки (см. раздел "Мощность и размер выборки" далее в этой главе).

Проверки хи-квадрат широко используются экспертами в исследованиях в поисках неуволимого статистически значимого p -значения, которое позволит опубликоваться. Проверки хи-квадрат или подобные симуляции на основе повторного отбора используются в приложениях науки о данных больше как фильтр для определе-

ния того, заслуживает ли эффект или признак дальнейшего рассмотрения, чем в качестве формальной проверки значимости. Например, они применяются в геопространственной статистике и картографии для определения, соответствуют ли пространственные данные указанному нулевому распределению (скажем, действительно ли преступления сконцентрированы в некой области больше, чем это позволяет случайность?). Их также можно использовать в автоматизированном отборе признаков в машинном обучении, чтобы определить преобладание класса во всех признаках и выявить признаки, в которых преобладание некоторого класса является необычно высоким либо низким, в степени, которая не совместима со случайной вариацией.

Ключевые идеи для проверки хи-квадрат

- Общепринятая процедура в статистике состоит в проверке, согласуются ли наблюдаемые количества данных с допущением о независимости (например, склонность покупать ту или иную вещь не зависит от пола).
- Распределение хи-квадрат — это эталонное распределение (которое воплощает допущение о независимости), с которым должна быть сопоставлена наблюдаемая расчетная статистика хи-квадрат.

Дополнительные материалы для чтения

- ◆ Знаменитый пример Р. А. Фишера "Lady Tasting Tea" (процедура *"леди дегустирует чай"* обозначает рандомизированный эксперимент) с начала XX века остается простой и эффективной иллюстрацией его точной статистической проверки. Погуглите "Lady Tasting Tea", и вы найдете много хороших рецензий.
- ◆ Сайт Stat Trek предлагает хорошее учебное пособие по проверке хи-квадрат⁸.

Алгоритм многорукого бандита

Многорукие бандиты предлагают подход к тестированию, в особенности веб-тестированию, который позволяет выполнять явную оптимизацию и принимать более скорые решения, чем традиционный статистический подход к планированию экспериментов.

Ключевые термины для алгоритма многорукого бандита

Многорукый бандит (multi-arm bandit)

Воображаемый игровой автомат с несколькими пусковыми рычагами, или руками, на которые игрок может нажимать по выбору, при этом каждая рука имеет разный выигрыш; здесь взят как аналогия многовариантного эксперимента.

⁸ См. <https://oreil.ly/77DUf>.

Рычаг (arm)

Вариант в эксперименте (например, "заголовок *A* в веб-тесте").

Выигрыш (win)

Экспериментальный аналог выигрыша в автомате (например, "клиент щелкает на ссылке").

Традиционный *A/B*-тест предусматривает, что данные, собранные в эксперименте в соответствии с детальным планом, отвечают на конкретно поставленный вопрос: что лучше — вариант *A* или вариант *B*? Изначально предполагается, что, как только мы получаем ответ на этот вопрос, экспериментирование заканчивается, и мы переходим к действиям по его результатам.

Вы, вероятно, усмотрите в таком подходе несколько трудностей. Во-первых, наш ответ может быть неокончательным или неопределенным: "эффект не доказан". Другими словами, результаты эксперимента могут свидетельствовать об эффекте, но если и будет эффект, то у нас не окажется достаточно большой выборки, чтобы его доказать (к полному удовлетворению традиционных статистических стандартов). Какие шаги нам предпринять? Во-вторых, мы, возможно, захотим начать пользоваться поступающими результатами до выводов эксперимента. В-третьих, мы, возможно, захотим иметь право передумать либо попробовать нечто другое, основываясь на дополнительных данных, которые поступают после того, как эксперимент закончен. Традиционный подход к экспериментам и проверке гипотез восходит к 1920-м годам и довольно негибок. Приход вычислительной мощности компьютеров и программно-информационного обеспечения сделал возможными более мощные и гибкие подходы. Кроме того, наука о данных (и бизнес в целом) не особо беспокоятся по поводу статистической значимости, и более заинтересованы в оптимизации совокупных усилий и результатов.

Бандитские алгоритмы, очень популярные в веб-тестировании, позволяют тестировать несколько вариантов одним махом и делать выводы быстрее, чем традиционные статистические планы экспериментов. Они заимствуют свое имя у автоматов, используемых в азартных играх, так называемых одноруких бандитов (поскольку они сконфигурированы таким образом, чтобы выкачивать из игрока деньги непрерывным потоком). Если вообразить автомат, у которого больше одного пускового рычага, или руки, при этом каждая выплачивает выигрыши с разной интенсивностью, то у вас получится многорукий бандит, который дает полное название этому алгоритму.

Ваша цель состоит в том, чтобы выиграть как можно больше денег, а именно выявить выигрышный рычаг и остановиться на нем как можно скорее. Трудность состоит в том, что вы не знаете, с какой совокупной интенсивностью они выплачивают выигрыши, — вы узнаете результаты, только если потяните за рычаг. Предположим, что каждый "выигрыш" приносит одинаковую сумму, независимо от того, за какой рычаг вы потянули. Отличается только вероятность выигрыша. Далее, допустим, что сначала вы пробуете каждый рычаг по 50 раз и получаете следующие результаты:

- ◆ рычаг A : 10 выигрышей из 50;
- ◆ рычаг B : 2 выигрыша из 50;
- ◆ рычаг C : 4 выигрыша из 50.

Один из предельных подходов состоит в том, чтобы сказать: "Похоже, рычаг A приносит выигрыш — надо прекратить попытки и остановиться на A ". Этот шаг в полной мере пользуется информацией начального испытания. Если A действительно превосходит, то мы извлекаем из этого выгоду с самого начала. С другой стороны, если B или C реально лучше, то мы теряем любую возможность это обнаружить. Другой предельный подход состоит в том, чтобы сказать: "Сдается, что все это во власти случая, — попробую понажимать их одинаково". Это дает максимальную возможность проявиться другим альтернативам помимо A . Однако по ходу мы задействуем варианты, которые выглядят менее успешные. Сколько времени мы позволим этому продолжаться? Бандитские алгоритмы действуют в соответствии с гибридным подходом: мы чаще начинаем нажимать на рычаг A , пользуясь его очевидным превосходством, но мы не отказываемся от B и C . Мы просто обращаемся к ним не так часто. Если A продолжает превосходить, то мы продолжаем смещать ресурсы (нажимать рычаги), отдаляясь от B и C и чаще нажимая на A . Если с другой стороны C начинает работать лучше, а A — хуже, то мы можем сместиться от A назад к C . Если окажется, что один из них превосходит A и это было скрыто в начальном испытании в силу случайности, то теперь он имеет возможность проявить себя при дальнейшем тестировании.

Теперь подумаем, как это применить к веб-тестированию. Вместо нескольких пустых рычагов автомата у вас могут быть многочисленные предложения, заголовки объявлений, цвета и т. д., которые тестируются на веб-сайте. Клиенты либо нажимают ("выигрыш" для коммерсанта), либо не нажимают. Первоначально предложения предлагаются случайно и одинаково. Если, однако, одно предложение начинает превосходить другие, то оно может предлагаться ("нажиматься") чаще. Но какими должны быть параметры алгоритма, который изменяет интенсивность нажатий рычага? На какую "интенсивность нажатий рычага" мы должны перейти и когда мы должны это сделать?

Вот один из простых алгоритмов, эпсилон-жадный алгоритм для A/B -теста:

1. Сгенерировать равномерно распределенное случайное число в интервале между 0 и 1.
2. Если число находится между 0 и эпсилон (где эпсилон — это число между 0 и 1 в типичной ситуации довольно малое), подбросить справедливую монету (т. е. с вероятностью 50/50), и:
 - если монета повернется орлом, то предложить вариант A ;
 - если монета повернется решкой, то предложить вариант B .
3. Если число больше или равно ϵ , то показать любое предложение, которое до настоящего времени имело самую высокую интенсивность откликов.

Эпсилон — единственный параметр, который управляет указанным алгоритмом. Если эпсилон равен 1, то мы заканчиваем стандартным простым A/B -экспери-

ментом (случайное размещение между A и B для каждого испытуемого). Если эпсилон равен 0, то мы заканчиваем чисто жадным алгоритмом — он не стремится продолжать экспериментировать, просто относя испытуемых (посетителей веб-сайта) к наиболее превосходящему варианту.

Более сложный алгоритм использует "отбор по методу Томпсона". Данная процедура "извлекает выборки" (нажимает пусковой рычаг бандита) на каждом этапе, максимизируя вероятность выбора наилучшего рычага. Разумеется, вы не знаете, какой рычаг является лучшим, — в это-то и проблема! Но по мере того, как вы наблюдаете за выплатой при каждом последующем извлечении, вы получаете все больше информации. В отборе по методу Томпсона используется байесов подход: первоначально принимают какое-то априорное распределение вознаграждений, используя так называемое бета-распределение (это общепринятый механизм для указания априорной информации в байесовой задаче). По мере накопления информация после каждого извлечения она может обновляться, давая возможность лучше оптимизировать следующее извлечение с точки зрения выбора верного рычага.

Бандитские алгоритмы способны эффективно обрабатывать три варианта и более и двигаться к оптимальному отбору "наилучшего". Что касается традиционных процедур статистического тестирования, то сложность принятия решений для трех и более вариантов далеко превосходит традиционный A/B -тест, и преимущество бандитских алгоритмов гораздо больше.

Ключевые идеи для алгоритма многорукого бандита

- Традиционные A/B -тесты предусматривают процесс случайного отбора, который может привести к чрезмерному показу варианта худшего по качеству.
- Многорукие бандиты, напротив, изменяют процесс отбора, инкорпорируя информацию, усваиваемую во время эксперимента, и уменьшают частоту показа менее качественного варианта.
- Они также упрощают эффективную обработку более двух вариантов.
- Существуют разные алгоритмы, которые смещают вероятность отбора от менее качественного варианта(ов) к (предполагаемому) более качественному.

Дополнительные материалы для чтения

- ♦ Книга "Бандитские алгоритмы для веб-оптимизации" (White J. M. Bandit algorithms for website optimization: developing, deploying, and debugging. — O'Reilly, 2012) предлагает превосходное краткое изложение алгоритмов многоруких бандитов. Автор приводит примеры на Python, а также результаты имитационного моделирования для анализа результативности бандитов.
- ♦ Дополнительную (несколько техническую) информацию об отборе по методу Томпсона см. в работе "Анализ отбора по методу Томпсона для задачи многору-

кого бандита" (Agrawal S., Goyal N. Analysis of Thompson Sampling for the Multi-armed Bandit Problem)⁹.

Мощность и размер выборки

Если вы выполняете веб-тест, то каким образом вы решаете, сколько времени он должен выполняться (т. е. сколько необходимо показов в расчете на вариант)? Не смотря на все то, что вы можете прочесть в многочисленных руководствах по веб-тестированию, хорошей общей рекомендации нет — все зависит главным образом от частоты, с которой достигается желаемая цель.

Ключевые термины для мощности и размера выборки

Размер эффекта (effect size)

Минимальный размер эффекта, который вы надеетесь обнаружить в результате статистической проверки, такой как "20%-ный рост интенсивностей нажатий".

Синоним: величина эффекта.

Мощность (power)

Вероятность обнаружить данный размер эффекта при данном размере выборки.

Уровень значимости (significance level)

Уровень статистической значимости, при котором будет проводиться проверка.

Синоним: альфа.

Один из шагов в статистических расчетах размера выборки состоит в том, чтобы задаться вопросом: покажет ли проверка гипотезы на самом деле разницу между вариантами *A* и *B*? Исход проверки гипотезы — *p*-значение — зависит от того, какова реальная разница между вариантами *A* и *B*. Он также зависит от жребия — тех, кого отбирают в группы для эксперимента. Но вполне логично, что чем больше фактическая разница между вариантами *A* и *B*, тем больше вероятность, что наш эксперимент ее покажет; и чем меньше разница, тем больше данных будет необходимо для ее обнаружения. Для того чтобы в бейсболе отличить 0,350-го бьющего игрока от 0,200-го бьющего¹⁰, потребуется не так уж много подходов к бите. А вот чтобы различить 0,300-го бьющего и 0,280-го бьющего, уже потребуется гораздо больше подходов.

⁹ См. <https://oreil.ly/OgWrG>.

¹⁰ В бейсболе бьющий игрок с отношением количества подходов или попыток подходов к бите к количеству хитов, т. е. к любым точным ударам по мячу, которые позволяют бьющему добежать до базы. Например, если бьющий имеет четыре подхода или попытки ударить и записывает два хита, то его коэффициент попаданий составит 2/4, или 0,500. — *Прим. перев.*

Мощность — это вероятность обнаружения указанного размера эффекта с указанными характеристиками выборки (размером и вариабельностью). Например, (гипотетически) мы можем сказать, что вероятность провести различие между 0,330-м бьющим игроком и 0,200-м бьющим игроком при 25 подходах к бите составляет 0,75. Размер эффекта здесь — это разница 0,130. И "обнаружение" означает, что проверка гипотезы отклонит нулевую гипотезу "нет разницы" и заключит, что существует реальный эффект. Так, эксперимент с 25 подходами к бите ($n = 25$) для двух бьющих игроков с размером эффекта 0,130 имеет (гипотетическую) мощность 0,75, или 75%.

Вы можете видеть, что здесь есть несколько подвижных элементов, и легко запутаться в многочисленных статистических допущениях и формулах, которые будут необходимы (чтобы указать выборочную вариабельность, размер эффекта, размер выборки, альфа-уровень для проверки гипотезы и т. д. и рассчитать мощность). На самом деле существуют статистические вычислительные системы специального назначения для расчета мощности. Большинству исследователей данных не придется проходить все формальные шаги, необходимые, чтобы сообщить мощность, например, в опубликованной работе. Однако они могут столкнуться со случаями, когда для A/B -теста им понадобится собрать какие-то данные, и сбор или обработка данных влечет за собой какие-то затраты. В этом случае приблизительные сведения о том, сколько данных необходимо собрать, помогают предотвратить ситуацию, когда вы собираете данные, затрачивая на это какие-то усилия, и в итоге дело заканчивается тем, что результат оказывается неокончательным или неопределенным. Вот весьма интуитивно понятный альтернативный подход:

1. Начать с каких-либо гипотетических данных, представляющих вашу наилучшую догадку о данных, которые получатся в результате (возможно, на основе априорных данных), например: коробка с 20 единицами и 80 нулями, которая представляет 0,200-х бьющих игрока, или коробка с несколькими наблюдениями о "времени, проведенном на веб-сайте".
2. Создать вторую выборку, добавив к первой выборке желаемый размер эффекта, например: вторая коробка с 33 единицами и 67 нулями или вторая коробка с 25 секундами, добавленными к каждому исходному "времени, проведенному на веб-сайте".
3. Извлечь бутстраповскую выборку размера n из каждой коробки.
4. Выполнить перестановочную (либо формульную) проверку гипотезы на двух бутстраповских выборках и записать, является ли разница между ними статистически значимой.
5. Повторить предыдущие два шага много раз и определить, как часто разница была значимой. Этот показатель будет расчетной мощностью.

Размер выборки

Наиболее часто встречающееся применение расчетов мощности состоит в оценивании необходимой вам величины выборки.

Например, предположим, что вы смотрите на кликабельности¹¹ (нажатия как процент показов) и тестируете новое объявление против существующего объявления. Сколько нажатий необходимо накопить в исследовании? Если вы заинтересованы только в тех результатах, которые показывают огромную разницу (скажем, 50%-ную разницу), то может сработать относительно малая выборка. Если же, с другой стороны, интерес представляет даже незначительная разница, то потребуется намного более крупная выборка. Стандартный подход состоит в принятии политики, что новое объявление должно работать лучше, чем существующее объявление, на некоторый процент, скажем на 10%; в противном случае существующее объявление останется на месте. Эта цель, "размер эффекта", в дальнейшем управляет размером выборки.

Например, предположим, что текущие кликабельности составляют примерно 1,1%, и вы стремитесь к 10%-ному росту до 1,21%. Поэтому у нас будет две коробки: коробка *A* с 1,1% единицами (скажем, со 110 единицами и 9890 нулями) и коробка *B* с 1,21% единицами (скажем, со 121 единицей и 9879 нулями). Для начала давайте попробуем сделать 300 извлечений из каждой коробки (это будет соответствовать 300 "показам" от каждого объявления). Допустим, что наше первое извлечение производит следующий результат:

- ◆ коробка *A*: 3 единицы;
- ◆ коробка *B*: 5 единиц.

Мы сразу же видим, что любая проверка гипотезы показала бы, что эта разница (5 против 3) находится далеко внутри диапазона случайной вариации. Данное сочетание размера выборки ($n = 300$ в каждой группе) и размера эффекта (10%-ная разница) является слишком малым для того, чтобы любая проверка гипотезы могла надежно показать разницу.

Поэтому мы можем попытаться увеличить размер выборки (давайте попробуем 2000 показов) и предусмотрим более крупный рост (30% вместо 10%).

Например, предположим, что текущие кликабельности по-прежнему составляют 1,1%, но теперь мы претендуем на 50%-ный рост до 1,65%. Таким образом, у нас есть две коробки: коробка *A* по-прежнему с 1,1% единицами (скажем, со 110 единицами и 9890 нулями) и коробка *B* с 1,65% единицами (скажем, со 165 единицами и 9868 нулями). Теперь мы попробуем сделать 2000 извлечений из каждой коробки. Допустим, что наше первое извлечение производит следующий результат:

- ◆ коробка *A*: 19 единиц;
- ◆ коробка *B*: 34 единицы.

Проверка значимости на этой разнице (34–19) показывает, что она по-прежнему регистрируется как "незначимая" (хотя намного ближе к значимой, чем более ранняя разница 5–3). Для того чтобы рассчитать мощность, нам потребуется повторить

¹¹ Кликабельность (click-through rate, CTR) — это метрика в интернет-маркетинге, которая определяется как отношение числа кликов на рекламное объявление к числу показов и измеряется в процентах. — *Прим. перев.*

предыдущую процедуру много раз либо использовать статистическую вычислительную систему, которая может вычислять мощность, но наше первоначальное извлечение говорит о том, что детектирование даже 50%-ного улучшения потребует нескольких тысяч показов.

Резюмируя, для вычисления мощности или требуемого размера выборки, существует четыре подвижных элемента:

- ◆ размер выборки;
- ◆ размер эффекта, который вы хотите обнаружить;
- ◆ уровень значимости (альфа), при котором будет проводиться тест;
- ◆ мощность.

Укажите любые три из них, и четвертый может быть вычислен. Чаще всего у вас возникнет желание вычислить размер выборки, и поэтому вы должны указать три других. В R и Python вам также придется указать альтернативную гипотезу как "более высокую" или "более крупную", чтобы получить одностороннюю проверку (подробное обсуждение темы односторонних и двухсторонних проверок *см. в разделе "Односторонняя проверка гипотезы против двухсторонней" ранее в этой главе*). Ниже приведен фрагмент кода на R для проверки с привлечением двух пропорций, где обе выборки имеют одинаковый размер (в примере используется пакет `pwr`):

```
effect_size = ES.h(p1=0.0121, p2=0.011)
pwr.2p.test(h=effect_size, sig.level=0.05, power=0.8, alternative='greater')
--
      Difference of proportion power calculation for binomial distribution
                                (arcsine transformation)

      h = 0.01029785
      n = 116601.7
sig.level = 0.05
power = 0.8
alternative = greater
```

NOTE: same sample sizes

Функция `ES.h` вычисляет размер эффекта. Мы видим, что если мы хотим получить мощность 80%, то нам потребуется выборка размером почти 120 тыс. показов. Если мы претендуем на 50%-ный рост ($p_1=0.0165$), то размер выборки сокращается до 5500 показов.

Пакет `statsmodels` содержит несколько методов расчета мощности. Здесь мы используем метод `proportion_effectsize` для вычисления размера эффекта и класс `TTestIndPower` для того, чтобы найти решение для размера выборки:

```
effect_size = sm.stats.proportion_effectsize(0.0121, 0.011)
analysis = sm.stats.TTestIndPower()
result = analysis.solve_power(effect_size=effect_size,
                              alpha=0.05, power=0.8, alternative='larger')
```

```
print('Размер выборки: %.3f' % result)
```

```
--
```

```
Размер выборки: 116602.393
```

Ключевые идеи для мощности и размера выборки

- Выяснение того, насколько большой размер выборки вам нужен, требует задуматься о перспективе статистической проверки, которую вы планируете провести.
- Вы должны определить минимальный размер эффекта, который хотите обнаружить.
- Вы также должны определить требуемую вероятность обнаружить этот размер эффекта (мощность).
- Наконец, вы должны указать уровень значимости (альфа), при котором будет проводиться проверка.

Дополнительные материалы для чтения

- ♦ Книга "Определение размера выборки и мощность" (Ryan T. P. Sample size determination and power. — John Wiley & Sons, 2013) предлагает всесторонний и удобочитаемый анализ данной темы.
- ♦ Стив Саймон (Steve Simon), статистический консультант, написал очень привлекательный пост по данной теме в стиле рассказа¹².

Резюме

Принципы планирования эксперимента — рандомизация испытуемых на две или более групп, которым предлагаются разные варианты — позволяет делать допустимые выводы о том, насколько хорошо эти варианты работают. Лучше всего при этом задействовать контрольный вариант "не вносящий никакого изменения". Предмет формального статистического вывода — проверка гипотез, p -значения, проверки на основе t -статистики и многое другое далее по списку — занимает много времени и пространства в традиционном курсе или учебнике статистики, и формализованность изложения является главным образом ненужной с точки зрения науки о данных. Однако все-таки остается важным признать ту роль, которую случайная вариация может сыграть в обмане человеческого мозга. Интуитивно понятные процедуры повторного отбора (перестановки и бутстрапа) позволяют исследователям данных измерять степень, с которой случайная вариация может сыграть свою роль в их анализе данных.

¹² См. <https://oreil.ly/18mtp>.

Регрессия и предсказание

Возможно, наиболее часто встречающейся целью в статистике являются ответы на вопросы: связана ли переменная X (или, что более вероятно, X_1, \dots, X_p) с переменной Y , и если да, то в чем эта связь заключается, и можем ли мы ее использовать для того, чтобы предсказать Y ?

Нигде единение между статистикой и наукой о данных не является более тесным, чем в контексте предсказания — в частности, предсказания (целевой) переменной исхода на основе значений других "предсказательных" переменных. Этот процесс тренировки модели на данных, где исход известен, для последующего применения к данным, где исход неизвестен, называется *контролируемым самообучением*. Еще одна важная связь между наукой о данных и статистикой лежит в области *обнаружения аномалий*, где диагностика регрессии, первоначально предназначенная для анализа данных и совершенствования регрессионной модели, может использоваться для обнаружения необычных записей.

Простая линейная регрессия

Простая линейная регрессия, или парная линейная регрессия, моделирует связь между величиной одной переменной и величиной второй, например по мере увеличения X увеличивается и Y . Или же по мере увеличения X уменьшается и Y ¹. Корреляция — еще один способ измерить то, каким образом две переменные связаны между собой (см. раздел "Корреляция" главы 1). Разница между ними состоит в том, что корреляция измеряет силу связи между двумя переменными, тогда как регрессия оценивает *природу* этой связи количественно.

Ключевые термины для простой линейной регрессии

Отклик (response)

Переменная, которую мы пытаемся предсказать.

Синонимы: зависимая переменная, Y -переменная, цель, исход.

¹ Этот и последующие разделы в настоящей главе закреплены за © 2020 Datastats, LLC, Питер Брюс, Эндрю Брюс и Питер Гедек; использовано с разрешения.

Независимая переменная (independent variable)

Переменная, которая используется для предсказания отклика.

Синонимы: X -переменная, предсказатель, предиктор, предсказательная переменная, признак, атрибут.

Запись (record)

Вектор, который состоит из значений предсказателей и значения исхода для отдельного элемента данных или случая.

Синонимы: строка, случай, прецедент, образец, экземпляр, пример.

Пересечение (intercept)

Пересечение регрессионной прямой с осью y , т. е. предсказанное значение, когда $X = 0$.

Синонимы: b_0 , β_0 , точка пересечения, коэффициент сдвига на оси y .

Коэффициент регрессии (regression coefficient)

Наклон регрессионной прямой по отношению к оси x .

Синонимы: наклон, b_1 , β_1 , оценки параметров, вес, угловой коэффициент.

Подогнанные значения (fitted values)

Оценки \hat{Y}_i , полученные из регрессионной прямой.

Синоним: предсказанные значения.

Остатки (residuals)

Разница между наблюдаемыми значениями и подогнанными значениями.

Синоним: ошибки.

Наименьшие квадраты (least squares)

Метод подгонки регрессии путем минимизации суммы квадратов остатков.

Синонимы: обычный метод наименьших квадратов, обычный МНК.

Уравнение регрессии

Простая линейная регрессия оценивает, насколько именно изменится Y , когда X изменяется на некоторую величину. Для коэффициента корреляции переменные X и Y взаимозаменяемы. В случае регрессии мы пытаемся предсказать переменную Y из переменной X , используя линейную связь (т. е. прямую):

$$Y = b_0 + b_1 X.$$

Эта формула читается, как " Y равняется b_1 , умноженное на X , плюс константа b_0 ". Компонент уравнения b_0 называется *пересечением* (или константой), а b_1 — *наклоном* по отношению к оси x . Оба члена отображаются на выходе в R как *коэффициенты*, хотя в повсеместном использовании термин "коэффициент" часто зарезерви-

рован за b_1 . Переменная Y называется *откликом*, или *зависимой переменной*, поскольку она зависит от X . Переменная X называется *предсказателем* (предиктором, от англ. *predictor*), или *независимой переменной*. Сообщество машинного обучения тяготеет к использованию других терминов, называя Y *целью* и X — *вектором признаков*. На протяжении всей этой книги мы будем использовать термины "*предсказатель*" и "*признак*" взаимозаменяемо.

Рассмотрим диаграмму рассеяния на рис. 4.1, показывающую число лет, в течение которых рабочий был подвержен воздействию хлопчатобумажной пыли (*Exposure*) против показателя объема легких (PEFR — "пиковая объемная скорость выдоха"). Каким образом переменная PEFR связана с *Exposure*? Трудно сказать что-то конкретное, просто глядя на изображение.

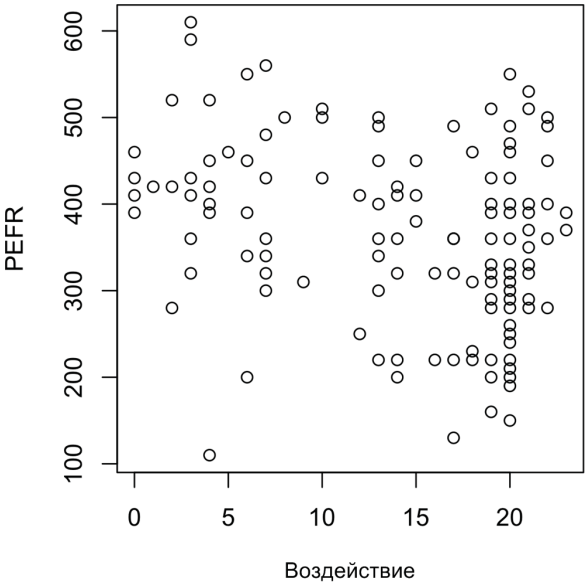


Рис. 4.1. Воздействие хлопка против объема легких

Простая линейная регрессия пытается отыскать "оптимальную" прямую, чтобы предсказать отклик PEFR, как функцию от предсказательной переменной *Exposure*.

$$PEFR = b_0 + b_1 \text{Exposure}.$$

Функция `lm` в R используется для подгонки линейной регрессии.

```
model <- lm(PEFR ~ Exposure, data=lung)
```

`lm` читается как *линейная модель*, а символ `~` обозначает, что переменная PEFR предсказывается переменной *Exposure*. С этим определением модели пересечение автоматически включается и подгоняется. Если вы хотите исключить пересечение из модели, то вам нужно написать определение модели следующим образом:

```
PEFR ~ Exposure - 1
```

Распечатка объекта `model` производит следующий ниже результат:

Call:

```
lm(formula = PEFr ~ Exposure, data = lung)
```

Coefficients:

(Intercept)	Exposure
424.583	-4.185

Пересечение, или b_0 , равно 424,583 и может быть истолковано как предсказанное PEFr для рабочего с нулевым числом лет воздействия. Коэффициент регрессии, или b_1 , может быть истолкован следующим образом: для каждого дополнительного года, в течение которого рабочий подвергается воздействию хлопчатобумажной пыли, результат измерения PEFr у рабочего уменьшается на $-4,185$.

В Python мы можем использовать класс `LinearRegression` из пакета `scikit-learn`. (пакет `statsmodels` содержит имплементацию линейной регрессии, которая больше похожа на имплементацию в языке R (`sm.OLS`); мы будем использовать ее позже в этой главе):

```
predictors = ['Воздействие']
```

```
outcome = 'PEFr'
```

```
model = LinearRegression()
```

```
model.fit(lung[predictors], lung[outcome])
```

```
print(f'Пересечение: {model.intercept_:.3f}')
```

```
print(f'Коэффициент воздействия: {model.coef_[0]:.3f}')
```

Регрессионная прямая указанной модели изображена на рис. 4.2.

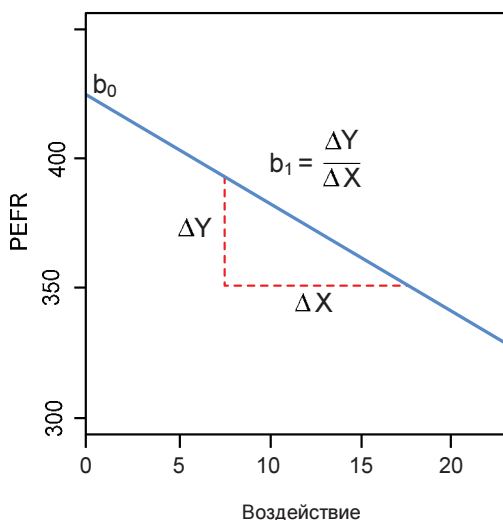


Рис. 4.2. Наклон и пересечение для регрессии, подогнанной к легочным данным

Подогнанные значения и остатки

В регрессионном анализе важными понятиями являются *подогнанные значения* и *остатки*. В общем случае данные не ложатся ровно на прямую, поэтому уравнение регрессии должно включать явный остаточный член e_i :

$$Y = b_0 + b_1X + e_i.$$

Подогнанные значения, также именуемые *предсказанными значениями*, в типичной ситуации обозначаются как \hat{Y}_i (Y -hat, Y с шляпой). Они задаются следующей формулой:

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_1X_i.$$

Обозначение \hat{b}_0 и \hat{b}_1 говорит о том, что эти коэффициенты оцениваются по отношению к известным, т. е. являются оценочными.



Обозначение со шляпой: оценки против известных значений

Обозначение со "шляпой" используется для дифференциации между оценками и известными значениями. Так, символ \hat{b} (" b с шляпой") является оценкой неизвестного параметра b . Почему в статистике проводится различие между оценкой и истинным значением? Оценка имеет неопределенность, тогда как истинное значение фиксировано².

Мы вычисляем остатки \hat{e}_i путем вычитания *предсказанных значений* из исходных данных:

$$\hat{e}_i = Y_i - \hat{Y}_i.$$

В R мы можем получить подогнанные значения и остатки при помощи функций `predict` и `residuals`:

```
fitted <- predict(model)
resid <- residuals(model)
```

С помощью линейной регрессионной модели на основе класса `LinearRegression` пакета `scikit-learn` мы используем метод `predict` на тренировочных данных, чтобы получить подогнанные значения `fitted`, и в последующем получить остатки `residuals`. Как мы увидим, это общий шаблон, которому следуют все модели в `scikit-learn`:

```
fitted = model.predict(lung[predictors])
residuals = lung[outcome] - fitted
```

На рис. 4.3 иллюстрируются остатки от регрессионной прямой, подогнанной к легочным данным. Остатки — это длина вертикальных пунктирных линий от данных до прямой.

² В байесовой статистике истинное значение считается случайной величиной с заданным распределением. В байесовом контексте вместо оценок неизвестных параметров существуют априорные и апостериорные распределения.

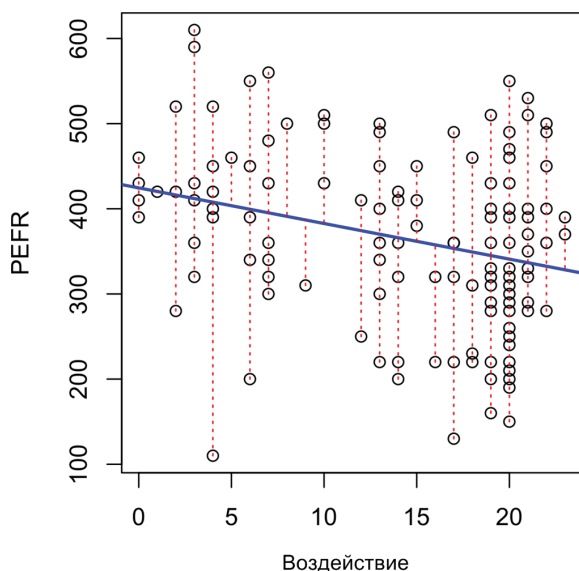


Рис. 4.3. Остатки от регрессионной прямой (для того чтобы разместить все данные, шкала оси у отличается от рис. 4.2, отсюда и очевидным образом другой угол наклона)

Наименьшие квадраты

Каким образом выполняется подгонка модели к данным? Когда существует четкая связь, вы можете мысленно представить подгонку прямой вручную. На практике прямая регрессии является оценкой, которая минимизирует значение суммы квадратичных остатков, также именуемых *суммой квадратов остатков* или *остаточной суммой квадратов* (residual sum of squares, RSS):

$$RSS = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 = \sum_{i=1}^n (Y_i - \hat{b}_0 - \hat{b}_1 X_i)^2.$$

Оценки \hat{b}_0 и \hat{b}_1 — это значения, которые минимизирует сумму квадратов остатков (RSS).

Метод минимизации суммы квадратов остатков называется *регрессией на основе наименьших квадратов*, или *регрессией на основе обычных наименьших квадратов* (ОНК). Данный метод часто приписывается Карлу Фридриху Гауссу, немецкому математику, но он был впервые опубликован в 1805 г. французским математиком Андре-Мари Лежандром (Adrien-Marie Legendre). Регрессию на основе наименьших квадратов можно легко и быстро вычислить с помощью стандартной статистической вычислительной системы.

Исторически, вычислительное удобство наименьших квадратов является одной из причин широкого применения данного метода в регрессии. С появлением больших данных вычислительная скорость по-прежнему остается важным фактором. Наименьшие квадраты, как и среднее значение (см. раздел "Медиана и робастные

оценки" главы 1), как метод чувствителен к выбросам, хотя этот факт имеет тенденцию быть значительной проблемой только в малых или умеренных по размеру наборам данных. Обсуждение темы выбросов в регрессии см. в разделе "Выбросы" главы 4.



Терминология регрессии

Когда аналитики и исследователи используют термин "регрессия" сам по себе, они обычно имеют в виду линейную регрессию; в центре внимания обычно находится разработка линейной модели для объяснения связи между предсказательными переменными и числовой переменной исхода. В своем формальном статистическом смысле регрессия также охватывает нелинейные модели, которые производят функциональную связь между предсказательными переменными и переменной исхода. В сообществе машинного обучения данный термин также временами используется в широком смысле для ссылки на использование любой предсказательной модели, которая производит предсказываемый числовой исход (в отличие от классификационных методов, которые предсказывают двоичный или категориальный исход).

Предсказание против объяснения (профилирование)

Исторически, первоочередное использование регрессии состояло в вычленении предполагаемой линейной связи между предсказательными переменными и переменной исхода. Цель заключалась в том, чтобы понять связь и объяснить ее при помощи данных, к которым выполнялась подгонка регрессии. В этом случае первоочередное внимание находится на оценочном наклоне уравнения регрессии \hat{b} . Экономисты хотят знать связь между потребительскими расходами и ростом ВВП. Чиновники Министерства здравоохранения могут захотеть понять, является ли кампания по информированию общественности эффективной при продвижении практик безопасного секса. В таких случаях в центре внимания находится не предсказание отдельных случаев, а наоборот, понимание общей связи.

С появлением больших данных регрессия широко используется для формирования модели с целью предсказания индивидуальных исходов для новых данных вместо статистического объяснения данных, имеющихся под рукой (т. е. предсказательной модели). В этом случае главными интересующими элементами являются подогнанные значения \hat{Y} . В маркетинге регрессия может использоваться для предсказания изменения в доходе в ответ на размер рекламной кампании. Университеты используют регрессию для предсказания среднего академического балла GPA студентов на основе их баллов за экзамен на определение академических способностей SAT³.

Регрессионная модель, которая подогнана к данным хорошо, настроена так, что изменения в X приводят к изменениям в Y . Однако само по себе уравнение регрессии не доказывает направление причинно-следственной обусловленности. Выводы

³ GPA (grade point average) — средний академический балл студентов; SAT (scholastic assessment test) — отборочный экзамен для выпускников школ, поступающих в вузы. — *Прим. перев.*

о причинно-следственной обусловленности надлежит делать, исходя из более широкого контекста понимания связи. Например, уравнение регрессии могло бы показать определенную связь между числом нажатий на веб-рекламе и числом конверсий. Как раз наше знание маркетингового процесса, а не уравнение регрессии приводит нас к заключению о том, что нажатия на рекламе генерируют продажи, и не наоборот.

Ключевые идеи для простой линейной регрессии

- Уравнение регрессии моделирует связь между переменной отклика Y и предсказательной переменной X в форме прямой.
- Регрессионная модель производит подогнанные значения и остатки — предсказания отклика и ошибки предсказаний.
- Подгонка регрессионных моделей в типичной ситуации выполняется методом наименьших квадратов.
- Регрессия используется как для предсказания, так и для объяснения.

Дополнительные материалы для чтения

Подробности о всесторонней трактовке темы предсказания относительно объяснения см. в статье Галита Шмуели (Galit Shmueli) "Объяснить или предсказать" (To Explain or to Predict?)⁴.

Множественная линейная регрессия

Когда предсказателей несколько, данное уравнение просто расширяется для их размещения:

$$Y = b_0 + b_1X_1 + b_2X_2 + \dots + b_pX_p + e.$$

Вместо прямой теперь у нас линейная модель — связь между каждым коэффициентом и его переменной (признаком) линейная.

Ключевые термины для множественной линейной регрессии

Корень из среднеквадратической ошибки (root mean squared error)

Квадратный корень из среднеквадратической ошибки регрессии (это наиболее широко используемая метрика для сравнения регрессионных моделей).

Синоним: RMSE.

⁴ См. <https://oreil.ly/4fVUY>.

Стандартная ошибка остатков (residual standard error)

То же самое, что и среднеквадратическая ошибка, но скорректированная для степеней свободы.

Синоним: RSE.

R-квадрат (R-squared)

Доля дисперсии, объясняемая моделью, со значениями в интервале от 0 до 1.

Синонимы: коэффициент детерминации, R^2 .

t-Статистика (t-statistic)

Коэффициент для предсказателя, деленный на стандартную ошибку коэффициента, дающий метрику для сравнения важности переменных в модели (см. раздел "Проверки на основе t-статистики" главы 3).

Взвешенная регрессия (weighted regression)

Регрессия, в которой записям поставлены в соответствие разные веса.

Все другие понятия из простой линейной регрессии, такие как подгонка наименьшими квадратами, подогнанные значения и остатки, относятся и к условиям множественной линейной регрессии. Например, подогнанные значения задаются следующей формулой:

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_{1,i} + \hat{b}_2 X_{2,i} + \dots + \hat{b}_p X_{p,i} + e.$$

Пример: данные жилого фонда округа Кинг

Примером использования регрессии является оценивание стоимости домов. Налоговые агенты округа должны оценивать стоимость домов в целях обложения налогами. Риэлтеры и покупатели недвижимости консультируются на популярных веб-сайтах, таких как Zillow, чтобы удостовериться в справедливости цены. Ниже приведено несколько строк данных жилого фонда округа Кинг (Сиэтл, шт. Вашингтон) из кадра данных house:

```
head(house[, c("AdjSalePrice", "SqFtTotLiving", "SqFtLot", "Bathrooms",  
              "Bedrooms", "BldgGrade")])
```

Source: local data frame [6 x 6]

	AdjSalePrice (dbl)	SqFtTotLiving (int)	SqFtLot (int)	Bathrooms (dbl)	Bedrooms (int)	BldgGrade (int)
1	300805	2400	9373	3.00	6	7
2	1076162	3764	20156	3.75	4	10
3	761805	2060	26036	1.75	4	8
4	442065	3200	8618	3.75	5	7
5	297065	1720	8620	1.75	4	7
6	411781	930	1012	1.50	2	8

Метод `head` кадра данных пакета `pandas` содержит список верхних строк:

```
subset = ['AdjSalePrice', 'SqFtTotLiving', 'SqFtLot', 'Bathrooms',  
         'Bedrooms', 'BldgGrade']  
house[subset].head()
```

Цель состоит в том, чтобы предсказать продажную цену на основе остальных переменных. Переменная `lm` обрабатывает случай множественной регрессии просто путем включения большего числа членов на правой части уравнения; аргумент `na.action=na.omit` заставляет модель отбрасывать записи с отсутствующими значениями:

```
house_lm <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +  
              Bedrooms + BldgGrade,  
              data=house, na.action=na.omit)
```

Класс `LinearRegression` пакета `scikit-learn` также можно использовать для множественной линейной регрессии:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']  
outcome = 'AdjSalePrice'
```

```
house_lm = LinearRegression()  
house_lm.fit(house[predictors], house[outcome])
```

Распечатав объект `house_lm`, мы получим следующий результат:

```
house_lm
```

Call:

```
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +  
    Bedrooms + BldgGrade, data = house, na.action = na.omit)
```

Coefficients:

(Intercept)	SqFtTotLiving	SqFtLot	Bathrooms
-5.219e+05	2.288e+02	-6.051e-02	-1.944e+04
Bedrooms	BldgGrade		
-4.778e+04	1.061e+05		

Для регрессионной модели на основе класса `LinearRegression` пересечение и коэффициенты являются полями `intercept_` и `coef_` подогнанной модели:

```
print(f'Пересечение: {house_lm.intercept_:.3f}')
```

```
print('Коэффициенты:')  
for name, coef in zip(predictors, house_lm.coef_):  
    print(f' {name}: {coef}')
```

Коэффициенты интерпретируются точно так же, как и в простой линейной регрессии: предсказанное значение \hat{Y} изменяется на коэффициент b_j для каждого единичного изменения в X_j с учетом того, что все остальные переменные X_k для $k \neq j$ остаются прежними. Например, добавление лишнего квадратного фута об-

щей площади в доме увеличивает его оценочную стоимость примерно на 229 \$; из добавления 1000 кв. футов следует, что это значение увеличится на 228 800 \$.

Оценивание результативности модели

Самой важной метрикой результативности с точки зрения науки о данных является *корень из среднеквадратической ошибки* (root mean squared error, RMSE). RMSE — это квадратный корень из среднеквадратической ошибки в предсказанных \hat{y}_i значениях:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}}.$$

Она измеряет совокупную точность модели и является основанием для ее сравнения с другими моделями (включая модели, подогнанные с использованием специальных технических приемов машинного обучения). Похожей на RMSE является *стандартная ошибка остатков* (residual standard error, RSE). В этом случае мы имеем p предсказателей, и RSE задается следующей формулой:

$$RSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n - p - 1}}.$$

Единственная разница состоит в том, что знаменатель является степенями свободы, в противоположность числу записей (см. раздел "Степени свободы" главы 3). На практике разница между RMSE и RSE для линейной регрессии является очень малой, в особенности для приложений больших данных.

Функция `summary` в R вычисляет RSE, а также другие метрики регрессионной модели:

```
summary(house_lm)
```

Call:

```
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +  
    Bedrooms + BldgGrade, data = house, na.action = na.omit)
```

Residuals:

Min	1Q	Median	3Q	Max
-1199508	-118879	-20982	87414	9472982

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-5.219e+05	1.565e+04	-33.349	< 2e-16 ***
SqFtTotLiving	2.288e+02	3.898e+00	58.699	< 2e-16 ***
SqFtLot	-6.051e-02	6.118e-02	-0.989	0.323
Bathrooms	-1.944e+04	3.625e+03	-5.362	8.32e-08 ***
Bedrooms	-4.778e+04	2.489e+03	-19.194	< 2e-16 ***

```
BldgGrade      1.061e+05  2.396e+03  44.287  < 2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 261200 on 22683 degrees of freedom
Multiple R-squared:  0.5407,    Adjusted R-squared:  0.5406
F-statistic: 5340 on 5 and 22683 DF, p-value: < 2.2e-16
```

Пакет `scikit-learn` предоставляет ряд метрик для регрессии и классификации. Здесь мы используем `mean_squared_error`, чтобы получить RMSE, и `r2_score` для коэффициента детерминации:

```
fitted = house_lm.predict(house[predictors])
RMSE = np.sqrt(mean_squared_error(house[outcome], fitted))
r2 = r2_score(house[outcome], fitted)
print(f'RMSE: {RMSE:.0f}')
print(f'r2: {r2:.4f}')
```

Используйте пакет `statsmodels` для получения более детального анализа регрессионной модели в Python:

```
model = sm.OLS(house[outcome], house[predictors].assign(const=1))
results = model.fit()
results.summary()
```

Используемый здесь метод `assign` пакета `pandas` добавляет к предсказателям константный столбец со значением 1. Это необходимо для моделирования пересечения.

Еще одна полезная метрика, которую вы увидите на выходе из вычислительных систем, называется *коэффициентом детерминации*, или *статистикой R-квадрат*, или R^2 . R-квадрат варьирует в интервале от 0 до 1 и измеряет долю вариации в данных, объясняемую в модели. Он полезен главным образом в объяснительных применениях регрессии, где вы хотите определить, насколько хорошо модель подогнана к данным. Формула для R^2 такова:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}.$$

Знаменатель пропорционален дисперсии отклика Y . Результат в R также сообщает о скорректированном R-квадрате, который корректируется с учетом степеней свободы. В множественной регрессии с крупными наборами данных он очень редко значительно отличается.

Вместе с оценочными коэффициентами язык R и пакет `statsmodels` сообщает о *стандартной ошибке коэффициентов* (standard error, SE) и *t*-статистике:

$$t_b = \frac{\hat{b}}{SE(\hat{b})}.$$

t -Статистика — и ее зеркальное отображение, p -значение — измеряет степень, с которой коэффициент является "статистически значимым", т. е. за пределами диапазона того, что может породить случайная компоновка предсказательной и целевой переменных. Чем выше t -статистика (и ниже p -значение), тем значимее предсказатель. Поскольку экономность является ценным модельным свойством, полезно иметь такого рода инструмент, который позволяет ориентироваться в выборе переменных для включения в качестве предсказателей (см. раздел "Отбор модели и пошаговая регрессия" далее в этой главе).



В дополнение к t -статистике язык R и другие пакеты часто будут сообщать о p -значении (в R это $\Pr(>|t|)$) и F -статистике. Исследователи данных обычно не слишком увлекаются интерпретацией этих статистик, равно как и вопросом статистической значимости. Они преимущественно сосредотачиваются на t -статистике как полезной рекомендации о том, следует ли включать предсказатель в модель или нет. Высокие t -статистики (которые сопровождаются p -значениями, близкими к 0) указывают на то, что предсказатель должен быть оставлен в модели, в то время как очень низкие t -статистики указывают на то, что предсказатель может быть отброшен. Более подробную информацию см. в разделе " p -Значение" главы 3.

Перекрестный контроль

Все классические статистические регрессионные метрики (R^2 , F -статистики и p -значения) являются "внутривыборочными" метриками — они применяются к тем же данным, которые использовались для подгонки модели. Интуитивно вы понимаете, что будет вполне логичным откладывать немного исходных данных, не используя их для подгонки модели, а затем применять модель к отложенным данным, чтобы увидеть, как хорошо она справляется со своей работой. Обычно вы будете использовать большую часть данных для подгонки модели, а оставшуюся часть — для ее тестирования.

Идея "вневыборочной" проверки не является новой, но она не утвердилась до тех пор, пока крупные наборы данных не стали более преобладающими; имея в распоряжении малый набор данных, аналитики, как правило, хотят использовать все имеющиеся данные и на их основе выполнять подгонку лучшей модели.

Использование отложенной выборки тем не менее ставит вас в зависимость от некоторой неопределенности, возникающей просто из-за вариабельности в малой отложенной выборке. Насколько будут отличаться результаты анализа модели, если бы вы извлекали другую отложенную выборку?

Перекрестный контроль расширяет идею отложенной выборки до многочисленных последовательных отложенных выборок. Алгоритм базового k -блочного перекрестного контроля выглядит следующим образом:

1. Отложить $1/k$ данных в качестве отложенной выборки.
2. Натренировать модель на оставшихся данных.
3. Применить модель к отложенной выборке $1/k$ (выставить ей балл) и записать необходимые метрики оценивания результативности модели.

4. Восстановить первые $1/k$ данных и отложить следующее $1/k$ (исключая любые записи, которые были выбраны в первый раз).
5. Повторить шаги 2 и 3.
6. Повторять до тех пор, пока каждая запись не будет использована в отложенной доле.
7. Усреднить либо иным образом скомбинировать метрики анализа модели.

Деление данных на тренировочную и отложенную выборки также называется *делением на блоки*.

Отбор модели и пошаговая регрессия

В некоторых задачах многие переменные могут использоваться в качестве предсказателей в регрессии. Например, для предсказания стоимости дома могут использоваться дополнительные переменные, такие как размер подвала или год постройки. В R они легко добавляются в уравнение регрессии:

```
house_full <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
  Bedrooms + BldgGrade + PropertyType + NbrLivingUnits +
  SqFtFinBasement + YrBuilt + YrRenovated +
  NewConstruction,
  data=house, na.action=na.omit)
```

В Python нам нужно конвертировать категориальные и булевы переменные в числа:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade',
  'PropertyType', 'NbrLivingUnits', 'SqFtFinBasement', 'YrBuilt',
  'YrRenovated', 'NewConstruction']
```

```
X = pd.get_dummies(house[predictors], drop_first=True)
X['NewConstruction'] = [1 if nc else 0 for nc in X['NewConstruction']]
```

```
house_full = sm.OLS(house[outcome], X.assign(const=1))
results = house_full.fit()
results.summary()
```

Добавление большего числа переменных, однако, не обязательно означает, что у нас получится более оптимальная модель. В статистике в качестве ориентира в выборе модели пользуются принципом бритвы Оккама: при прочих равных условиях предпочтение отдается использованию более простой модели над более сложной.

Включение в состав дополнительных переменных всегда уменьшает RMSE и увеличивает R^2 для тренировочных данных. Следовательно, они не подходят в качестве ориентиров, которые помогают выбрать модель. Один из подходов к включению модельной сложности состоит в использовании скорректированного R^2 :

$$R_{adj}^2 = 1 - (1 - R^2) \frac{n-1}{n-P-1}.$$

Здесь n — это число записей; P — число переменных в модели.

В 1970-х годах Хиротугу Акаике (Hirotugu Akaike), выдающийся японский статистик, разработал метрику под названием AIC (Akaike's Information Criteria — информационный критерий Акаике), которая штрафует добавление членов в модель. В случае регрессии AIC имеет следующую форму:

$$AIC = 2P + n \log(RSS/n),$$

где P — это число переменных; n — число записей. Цель состоит в том, чтобы найти модель, которая минимизирует AIC; модели с k лишними переменными штрафуются на $2k$.



AIC, BIC и CP Мэллоуза

Формула метрики AIC может показаться немного таинственной, но фактически она основывается на асимптотических результатах в теории информации. Существует несколько вариантов метрики AIC:

- AICc — версия AIC, скорректированная для выборок малых размеров;
- BIC (Bayesian information criteria — байесов информационный критерий) — похожий на AIC, но с более сильным штрафом за включение в модель дополнительных переменных;
- CP Мэллоуза — вариант AIC, разработанный Колином Мэллоузом (Colin Mallows).

В типичной ситуации все они представляются как внутривыборочные метрики (т. е. на тренировочных данных), и исследователям данных, использующим отложенные данные для определения результативности модели, обычно не нужно беспокоиться по поводу различий среди них или опорной теории, стоящей за ними.

Как отыскать модель, которая минимизирует AIC или максимизирует скорректированный R^2 ? Один из подходов — выполнить поиск среди всех возможных моделей, именуемый *регрессией всех подмножеств*. Данный подход является вычислительно затратным и не выполним для задач с крупными данными и *многочисленными переменными*. Он может начинаться с полной модели и последовательно отбрасывать переменные, которые не вносят вклада в содержательном плане. Эта процедура называется *обратным устранением*. В качестве альтернативы можно начать с константной модели и последовательно добавлять переменные (*прямой отбор*). В качестве третьего варианта мы также можем последовательно добавлять и отбрасывать предсказатели, чтобы отыскать модель, которая понижает AIC или скорректированный R^2 . Пакет MASS, разработанный Венеблз (Venables) и Рипли (Ripley), предлагает функцию пошаговой регрессии, которая называется `stepAIC`:

```
library(MASS)
step <- stepAIC(house_full, direction="both")
step

Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + Bathrooms + Bedrooms +
    BldgGrade + PropertyType + SqFtFinBasement + YrBuilt, data = house0,
    na.action = na.omit)
```

```

Coefficients:
            (Intercept)          SqFtTotLiving
            6227632.22              186.50
            Bathrooms              Bedrooms
            44721.72             -49807.18
            BldgGrade  PropertyTypeSingle Family
            139179.23              23328.69
PropertyTypeTownhouse          SqFtFinBasement
            92216.25              9.04
            YrBuilt
            -3592.47

```

Пакет `scikit-learn` не содержит имплементации для пошаговой регрессии. Мы имплементировали функции `stepwise_selection`, `forward_selection` и `backward_elimination` в нашем пакете `dmbs`:

```
y = house[outcome]
```

```

def train_model(variables): ❶
    if len(variables) == 0:
        return None
    model = LinearRegression()
    model.fit(X[variables], y)
    return model

```

```

def score_model(model, variables): ❷
    if len(variables) == 0:
        return AIC_score(y, [y.mean()] * len(y), model, df=1)
    return AIC_score(y, model.predict(X[variables]), model)

```

```

best_model, best_variables = stepwise_selection(X.columns, train_model,
                                                score_model, verbose=True)

```

```

print(f'Пересечение: {best_model.intercept_:.3f}')
print('Коэффициенты:')
for name, coef in zip(best_variables, best_model.coef_):
    print(f' {name}: {coef}')

```

❶ Определить функцию, которая возвращает подогнанную модель для заданного множества переменных.

❷ Определить функцию, которая возвращает балл для данной модели и множества переменных. В этом случае мы используем метрику `AIC_score`, имплементированную в пакете `dmbs`.

Функция выбрала модель, в которой несколько переменных были отброшены из `house_full`: `SqFtLot` (квадратура в футах), `NbrLivingUnits` (число жилых комнат), `YrRenovated` (год реновации) и `NewConstruction` (недавно построенный).

Еще более простыми являются *прямой отбор* и *обратный отбор*. В прямом отборе вы начинаете без предсказателей и добавляете их по одному, на каждом шаге добавляя предсказатель, который имеет самый большой вклад в R^2 , останавливаясь, когда вклад больше не является статистически значимым. В обратном отборе, или *обратном исключении*, вы начинаете с полной модели и удаляете предсказатели, не являющиеся статистически значимыми до тех пор, пока у вас не останется модель, в которой все предсказатели являются статистически значимыми.

Штрафная, или пенализованная, регрессия по духу похожа на метрику AIC. Вместо того чтобы явным образом выполнять поиск в дискретном множестве моделей, уравнение модельной подгонки инкорпорирует ограничение, которое штрафует модель за слишком большое число переменных (параметров). Взамен полного устранения предсказательных переменных — как в пошаговой регрессии, прямом и обратном отборе — штрафная регрессия применяет штраф путем понижения коэффициентов, в некоторых случаях почти до нуля. Общепринятыми штрафными методами регрессии являются *гребневая* и *лассо-регрессия*.

Пошаговая регрессия и регрессия всех подмножеств — это внутривыборочные методы определения результативности и тонкой регулировки моделей. Это означает, что модель, полученная в результате отбора модели, возможно, будет подвержена переподгонке (подгонке к шуму в данных) и не сможет показывать такую же хорошую результативность применительно к новым данным. Один из общепринятых подходов для ее предотвращения состоит в использовании перекрестного контроля для валидации моделей. В линейной регрессии переподгонка, как правило, не является главной проблемой, вследствие простой (линейной) глобальной структуры, навязываемой данным. Для более сложных типов моделей, в особенности итеративных процедур, которые откликаются на локальную структуру данных, перекрестный контроль является очень важным инструментом; подробности *см. в разделе "Перекрестный контроль" ранее в этой главе*.

Взвешенная регрессия

Взвешенная регрессия в статистике используется для разнообразных целей; в частности, она имеет большое значение для анализа сложных опросов. Исследователи данных могут посчитать взвешенную регрессию полезной в двух случаях:

- ♦ инверсно-дисперсионное взвешивание, когда разные наблюдения были измерены с разной прецизионностью; более высоко дисперсионные получают более низкие веса;
- ♦ анализ данных, где строки представляют многочисленные случаи; весовая переменная кодирует число исходных наблюдений, которое содержится в каждой строке.

Например, в данных жилой недвижимости более старые продажи являются менее надежными, чем более свежие. Используя дату документа `DocumentDate` для определения года продажи, мы можем вычислить вес `Weight` как число лет, начиная с 2005 года (начала данных).

R:

```
library(lubridate)
house$Year = year(house$DocumentDate)
house$Weight = house$Year - 2005
```

Python:

```
house['Year'] = [int(date.split('-')[0]) for date in house.DocumentDate]
house['Weight'] = house.Year - 2005
```

Мы можем вычислить взвешенную регрессию с помощью функции `lm`, используя аргумент `weight`.

```
house_wt <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
               Bedrooms + BldgGrade,
               data=house, weight=Weight)
round(cbind(house_lm=house_lm$coefficients,
            house_wt=house_wt$coefficients), digits=3)
```

	house_lm	house_wt
(Intercept)	-521924.722	-584265.244
SqFtTotLiving	228.832	245.017
SqFtLot	-0.061	-0.292
Bathrooms	-19438.099	-26079.171
Bedrooms	-47781.153	-53625.404
BldgGrade	106117.210	115259.026

Коэффициенты во взвешенной регрессии немного отличаются от исходной регрессии.

Большинство моделей в пакете `scikit-learn` принимают веса в качестве именованного аргумента `sample_weight` при вызове метода `fit`:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']
outcome = 'AdjSalePrice'
```

```
house_wt = LinearRegression()
house_wt.fit(house[predictors], house[outcome], sample_weight=house.Weight)
```

Ключевые идеи для множественной линейной регрессии

- Множественная линейная регрессия моделирует связь между переменной отклика Y и многочисленными предсказательными переменными X_1, \dots, X_p .
- Самыми важными метриками, используемыми для оценивания результативности модели, являются корень из среднеквадратической ошибки (RMSE) и коэффициент детерминации (R^2).
- Стандартная ошибка коэффициентов может использоваться для измерения надежности вклада переменной в модель.

- Пошаговая регрессия — это метод автоматического определения того, какие переменные должны быть включены в состав модели.
- Взвешенная регрессия используется для придания некоторым записям большего или меньшего веса при подгонке уравнения.

Дополнительные материалы для чтения

Превосходную трактовку процедур перекрестного контроля и повторного отбора можно найти в книге Гарета Джеймса, Даниэлы Виттен, Тревора Хаста и Роберта Тибширани "Введение в статистическое автоматическое обучение" (James G., Witten D., Hastie T., Tibshirani R. An introduction to statistical learning. — Springer, 2013).

Предсказание с использованием регрессии

Первоочередное предназначение регрессии в науке о данных заключается в предсказании. Это полезно учитывать потому, что регрессия, будучи проверенным временем и признанным статистическим методом, сопровождается багажом, который более соответствует своей традиционной роли объяснительного моделирования, чем предсказанию.

Ключевые термины для предсказания с использованием регрессии

Предсказательный интервал (prediction interval)

Интервал неопределенности вокруг отдельного предсказанного значения.

Экстраполяция (extrapolation)

Расширение модели за пределы диапазона данных, использованных для ее подгонки.

Опасности экстраполяции

Регрессионные модели не следует использовать для экстраполирования за пределы диапазона данных (оставляя в стороне использование регрессии для прогнозирования временных рядов.). Модель допустима только для тех предсказателей, для которых в данных имеется достаточно значений (и даже в случае достаточности данных могут иметься другие проблемы — см. раздел "Диагностика регрессии" далее в этой главе). В качестве крайнего случая предположим, что `model_lm` используется для предсказания стоимости незанятого земельного участка площадью 5000 кв. футов. В данном случае все предсказатели, связанные со зданием, будут иметь значение 0, и уравнение регрессии выдаст абсурдное предсказание $-521\,900 + 5000 \cdot (-0,0605) = -522\,202$ долларов. Почему это произошло? Данные содержат земельные участ-

ки только со зданиями — записи, соответствующие свободным земельным участкам, отсутствуют. Следовательно, модель не имеет никакой информации, говорящей ей о том, каким образом предсказывать продажную цену на свободный земельный участок.

Доверительный и предсказательный интервалы

Большая часть статистик предусматривает понимание и измерение вариабельности (неопределенности). t -Статистики и p -значения, о которых сообщается на выходе из регрессии, решают это формальным образом, что иногда полезно для отбора переменной (см. раздел "Оценивание результативности модели" ранее в этой главе). Более полезными метриками являются доверительные интервалы, которые суть интервалы неопределенности, помещаемые вокруг регрессионных коэффициентов и предсказаний. Простой путь к пониманию этих метрик лежит через бутстрап (более подробную информацию об общей процедуре бутстрапирования см. в разделе "Бутстрап" главы 2). Доверительные интервалы, наиболее часто встречающиеся на выходе из вычислительных систем, относятся к регрессионным параметрам (коэффициентам). Ниже приведен алгоритм бутстрапа, который генерирует доверительные интервалы для параметров (коэффициентов) регрессии с использованием набора данных с P предсказателями и n записями (строками):

1. Рассматривать каждую строку (включая переменную исхода) как отдельный "пакет" и поместить все n пакетов в коробку.
2. Вынуть пакет наугад, записать его значения и вернуть его в коробку.
3. Повторить n раз шаг 2; теперь у вас есть одна бутстраповская повторно отобранная выборка.
4. Выполнить подгонку регрессии к бутстраповской выборке и записать оценочные коэффициенты.
5. Повторить шаги 2–4, скажем, 1000 раз.
6. Теперь у вас есть 1000 бутстраповских значений для каждого коэффициента; найти соответствующие процентиля для каждого из них (например, 5-й и 95-й для 90%-ного доверительного интервала).

В R вы можете использовать функцию `boot`, чтобы сгенерировать фактические бутстраповские доверительные интервалы для коэффициентов, или же можете просто использовать формульные интервалы, которые в R рутинно показываются на выходе. Концептуальный смысл и интерпретация остаются теми же и не являются первоочередной необходимостью для исследователей данных, потому что они касаются коэффициентов регрессии. Большой интерес для исследователей данных представляют интервалы вокруг предсказанных значений y (\hat{Y}_i). Неопределенность вокруг \hat{Y}_i вытекает из двух источников:

- ◆ неопределенность в том, каковы релевантные предсказательные переменные и их коэффициенты (см. приведенный выше алгоритм бутстрапа);
- ◆ дополнительная ошибка, присущая отдельным точкам данных.

Вы можете думать об ошибке отдельной точки данных следующим образом: даже если бы мы знали определенно, каким было уравнение регрессии (например, если бы мы имели огромное число записей для подгонки), *фактические* значения исхода для заданного множества значений предсказателей будут варьироваться. Например, несколько домов — каждый с 8 комнатами, общей площадью 6500 кв. футов, 3 ванными комнатами и цоколем — могут иметь разные стоимости. Мы можем смоделировать эту отдельную ошибку остатками от подогнанных значений. Алгоритм бутстрапа для моделирования как ошибки регрессионной модели, так и ошибки отдельной точки данных будет выглядеть следующим образом:

1. Взять бутстраповскую выборку из данных (в деталях разъясненную ранее).
2. Выполнить подгонку регрессии и предсказать новое значение.
3. Взять наугад один-единственный остаток из первоначальной подгонки регрессии, прибавить его к предсказанному значению и записать результат.
4. Повторить шаги 1–3, скажем, 1000 раз.
5. Найти 2,5-й и 97,5-й процентиля результатов.

Ключевые идеи для предсказания с использованием регрессии

- Экстраполирование за пределы диапазона данных может приводить к ошибке.
- Доверительные интервалы квантифицируют неопределенность вокруг коэффициентов регрессии.
- Предсказательные интервалы квантифицируют неопределенность в отдельных предсказаниях.
- Большинство вычислительных систем, включая R, на выходе будут производить стандартные предсказательные и доверительные интервалы с использованием формул.
- Бутстрап может также использоваться для производства предсказания и доверительных интервалов; интерпретация и идея остаются прежними.



Предсказательный интервал или доверительный интервал?

Предсказательный интервал касается неопределенности вокруг одного-единственного значения, тогда как доверительный интервал связан со средним значением или другой статистикой, рассчитанными из многочисленных значений. Таким образом, для того же значения предсказательный интервал, как правило, будет намного шире доверительного интервала. Мы моделируем ошибку отдельного значения в бутстраповской модели путем отбора отдельного остатка с тем, чтобы его прикрепить к предсказанному значению. Который из двух интервалов использовать? Все зависит от контекста и предназначения анализа, но в общем случае исследователи данных интересуются специфическими отдельными предсказаниями, поэтому предсказательный интервал будет более подходящим. Выбор доверительного интервала тогда, когда необходимо использовать предсказательный интервал, значительно недооценивает неопределенность в конкретном предсказанном значении.

Факторные переменные в регрессии

Факторные переменные, именуемые также *категориальными* переменными, принимают предельное число дискретных значений. Например, целью ссуды может быть "консолидация задолженности", "свадьба", "автомобиль" и т. д. Двоичная (да/нет) переменная, именуемая также *индикаторной* переменной, является особым случаем факторной переменной. Регрессия требует на входе числовые данные, поэтому факторные переменные нужно перекодировать, чтобы их можно было использовать в модели. Наиболее часто встречающийся подход состоит в конвертировании переменной во множество двоичных *фиктивных* переменных.

Ключевые термины для факторных переменных в регрессии

Фиктивные переменные (dummy variables)

Двоичные переменные, принимающие значения 0 и 1 и выводимые путем перекодирования факторных данных, для использования в регрессии и других моделях.

Опорное кодирование (Reference coding)

Наиболее часто встречающийся тип кодирования, используемый в статистике, в котором один уровень фактора выбирается в качестве опорного, а другие факторы сопоставляются с этим уровнем.

Синонимы: кодирование по опорному варианту, комбинированное кодирование.

Кодировщик с одним активным состоянием (one hot encoder)

Тип кодирования, общепринятый в сообществе машинного обучения, при котором сохраняются все уровни факторов. Широко используется в некоторых автоматически обучающихся алгоритмах; вместе с тем этот прием не подходит для множественной линейной регрессии.

Девияционное кодирование (deviation coding)

Тип кодирования, при котором каждый уровень сравнивается не с опорным уровнем, а с совокупным средним⁵.

Синоним: суммовые контрасты.

Представление фиктивных переменных

В данных жилого фонда округа Кинг имеется факторная переменная, соответствующая типу собственности; ниже показано малое подмножество из шести записей.

⁵ Подробнее о девияционном кодировании и других формах кодирования см. <https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/> и <https://www.statsmodels.org/devel/contrasts.html>. — Прим. перев.

R:

```
head(house[, 'PropertyType'])
Source: local data frame [6 x 1]
```

```
PropertyType
(fctr)
1    Multiplex
2 Single Family
3 Single Family
4 Single Family
5 Single Family
6    Townhouse
```

Python:

```
house.PropertyType.head()
```

Имеются три возможных значения: Multiplex (мультиплекс), Single Family (односемейный) и Townhouse (таунхаус). Для того чтобы воспользоваться указанной факторной переменной, мы должны конвертировать ее во множество двоичных переменных. Это делается путем создания двоичной переменной для каждого возможного значения факторной переменной. Для того чтобы сделать это в R, мы используем функцию `model.matrix`⁶:

```
prop_type_dummies <- model.matrix(~PropertyType -1, data=house)
head(prop_type_dummies)
PropertyTypeMultiplex PropertyTypeSingle Family PropertyTypeTownhouse
1                1                0                0
2                0                1                0
3                0                1                0
4                0                1                0
5                0                1                0
6                0                0                1
```

Функция `model.matrix` конвертирует кадр данных в матрицу, подходящую для линейной модели. Факторная переменная `PropertyType`, которая имеет три отличающихся уровня, представлена матрицей с тремя столбцами. В сообществе машинного обучения такое представление называется кодированием с одним активным состоянием (см. раздел "Кодировщик с одним активным состоянием" главы 6).

В Python мы можем конвертировать категориальные переменные в фиктивные с помощью метода `get_dummies` пакета `pandas`:

```
pd.get_dummies(house['PropertyType']).head() ❶
pd.get_dummies(house['PropertyType'], drop_first=True).head() ❷
```

⁶ Аргумент `-1` в `model.matrix` производит представление с одним активным состоянием (путем удаления константы пересечения, отсюда и "-"). В противном случае в R по умолчанию производится матрица с $P - 1$ столбцами, где первый факторный уровень является опорой.

❶ По умолчанию возвращает одну кодировку категориальной переменной с одним активным состоянием.

❷ Именованный аргумент `drop_first` будет возвращать $P-1$ столбцов. Используйте его во избежание проблемы мультиколлинеарности.

В некоторых автоматически обучающихся алгоритмах, таких как ближайшие соседи и древесные модели, кодирование с одним активным состоянием является стандартным способом представления факторных переменных (например, см. раздел "Древесные модели" главы 6).

В регрессионной формулировке факторная переменная с P четко различимыми уровнями обычно представляется матрицей только с $P-1$ столбцами. Это обусловлено тем, что регрессионная модель в типичной ситуации включает член пересечения. Говоря о пересечении, после того как вы определили значения для $P-1$ двоичных столбцов, значение P -го становится известным и может считаться избыточным. Добавление P -го столбца вызовет ошибку мультиколлинеарности (см. раздел "Мультиколлинеарность" далее в этой главе).

В R дефолтное представление состоит в использовании первого уровня фактора в качестве опоры и интерпретации оставшихся уровней относительно этого фактора.

```
lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +  
+ Bedrooms + BldgGrade + PropertyType, data=house)
```

Call:

```
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +  
Bedrooms + BldgGrade + PropertyType, data = house)
```

Coefficients:

(Intercept)	SqFtTotLiving
-4.469e+05	2.234e+02
SqFtLot	Bathrooms
-7.041e-02	-1.597e+04
Bedrooms	BldgGrade
-5.090e+04	1.094e+05
PropertyTypeSingle Family	PropertyTypeTownhouse
-8.469e+04	-1.151e+05

Метод `get_dummies` принимает необязательный именованный аргумент `drop_first`, чтобы исключить первый фактор в качестве опоры:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms',  
             'BldgGrade', 'PropertyType']
```

```
X = pd.get_dummies(house[predictors], drop_first=True)
```

```
house_lm_factor = LinearRegression()  
house_lm_factor.fit(X, house[outcome])
```

```
print(f'Пересечение: {house_lm_factor.intercept_:.3f}')
print('Коэффициенты:')
for name, coef in zip(X.columns, house_lm_factor.coef_):
    print(f' {name}: {coef}')
```

Результат регрессии в R показывает два коэффициента, соответствующих типу собственности Property Type: PropertyTypeSingle Family и PropertyTypeTownhouse. Коэффициент Multiplex отсутствует, поскольку он неявно определяется, когда PropertyTypeSingle Family == 0 и PropertyTypeTownhouse == 0. Эти коэффициенты интерпретируются как относительные для Multiplex, и поэтому стоимость дома с типом собственности Single Family меньше почти на 85 000 \$, и стоимость дома с типом собственности Townhouse меньше более чем на 150 000 \$⁷.



Другие способы кодирования факторов

Существует несколько других способов кодирования факторных переменных, именуемых системами *контрастного кодирования*. Например, *девиационное кодирование*, именуемое также *суммовыми контрастами*, сравнивает каждый уровень с совокупным средним. Еще одним контрастом является *полиномиальное кодирование*, которое подходит для упорядоченных факторов (см. раздел "Упорядоченные факторные переменные" далее в этой главе). За исключением упорядоченных факторов, исследователи данных обычно не сталкиваются с другими типами кодирования помимо опорного кодирования либо кодировщика с одним активным состоянием.

Факторные переменные с многочисленными уровнями

Некоторые факторные переменные могут производить огромное число двоичных фиктивных переменных — почтовые индексы — это образец факторной переменной, а в США имеется 43 тыс. почтовых индексов. В таких случаях полезно разведать данные, а также связи между предсказательными переменными и исходом, чтобы определить наличие полезной информации в категориях. Если она имеется, то далее вы должны принять решение о том, имеет ли смысл оставить все факторы, или о том, следует ли уровни консолидировать.

В округе Кинг имеется 82 почтовых индекса с продажей домов:

```
table(house$ZipCode)

9800 89118 98001 98002 98003 98004 98005 98006 98007 98008 98010 98011
  1      1   358   180   241   293   133   460   112   291    56   163
98014 98019 98022 98023 98024 98027 98028 98029 98030 98031 98032 98033
  85   242   188   455    31   366   252   475   263   308   121   517
98034 98038 98039 98040 98042 98043 98045 98047 98050 98051 98052 98053
  575   788    47   244   641    1   222    48    7    32   614   499
```

⁷ Это противоречит логике, но может быть объяснено влиянием местоположения как искажающей переменной (см. раздел "Искажающие переменные" главы 4).

98055	98056	98057	98058	98059	98065	98068	98070	98072	98074	98075	98077
332	402	4	420	513	430	1	89	245	502	388	204
98092	98102	98103	98105	98106	98107	98108	98109	98112	98113	98115	98116
289	106	671	313	361	296	155	149	357	1	620	364
98117	98118	98119	98122	98125	98126	98133	98136	98144	98146	98148	98155
619	492	260	380	409	473	465	310	332	287	40	358
98166	98168	98177	98178	98188	98198	98199	98224	98288	98354		
193	332	216	266	101	225	393	3	4	9		

Метод `value_counts` фреймов данных пакета `pandas` возвращает ту же информацию:

```
pd.DataFrame(house['ZipCode'].value_counts()).transpose()
```

Переменная `ZipCode` имеет важно значение, поскольку она является эрзацем для эффекта местоположения, влияющего на стоимость дома. Включение в состав всех уровней требует 81 коэффициента, что соответствует 81 степени свободы. Исходная модель `house_lm` имеет всего 5 степеней свободы (см. раздел *"Оценивание результативности модели"* ранее в этой главе). Более того, несколько почтовых индексов имеют всего одну продажу. В некоторых задачах вы можете консолидировать почтовый индекс при помощи первых двух-трех цифр, что соответствует субметропольному географическому району. Для округа Кинг почти все продажи происходят в индексах 980xx или 981xx, так что это не помогает.

Альтернативный подход состоит в группировании почтовых индексов в соответствии с еще одной переменной, такой как продажная цена. Еще лучше формировать группы с почтовыми индексами, используя остатки из первоначальной модели. В следующем фрагменте кода `dplyr` консолидирует эти 82 почтовых индекса в пять групп, основываясь на медиане остатка из регрессии `house_lm`:

```
zip_groups <- house %>%
  mutate(resid = residuals(house_lm)) %>%
  group_by(ZipCode) %>%
  summarize(med_resid = median(resid),
            cnt = n()) %>%
  arrange(med_resid) %>%
  mutate(cum_cnt = cumsum(cnt),
         ZipGroup = ntile(cum_cnt, 5))
house <- house %>%
  left_join(select(zip_groups, ZipCode, ZipGroup), by='ZipCode')
```

Медианный остаток вычисляется для каждого почтового индекса, и функция `ntile` используется для разложения почтовых индексов, отсортированных по медиане, на пять групп. Пример того, как он используется в качестве члена в регрессии, совершенствуясь на первоначальной подгонке, см. в разделе *"Искажающие переменные"* далее в этой главе.

В Python мы можем вычислить эту информацию следующим образом:

```
zip_groups = pd.DataFrame([
    *pd.DataFrame({
        'ZipCode': house['ZipCode'],
```

```

        'residual' : house[outcome] - house_lm.predict(house[predictors]),
    })
    .groupby(['ZipCode'])
    .apply(lambda x: {
        'ZipCode': x.iloc[0,0],
        'count': len(x),
        'median_residual': x.residual.median()
    })
    ).sort_values('median_residual')
zip_groups['cum_count'] = np.cumsum(zip_groups['count'])
zip_groups['ZipGroup'] = pd.qcut(zip_groups['cum_count'], 5, labels=False,
                                retbins=False)
to_join = zip_groups[['ZipCode', 'ZipGroup']].set_index('ZipCode')
house = house.join(to_join, on='ZipCode')
house['ZipGroup'] = house['ZipGroup'].astype('category')

```

Идея использовать остатки, чтобы помочь сориентировать подгонку регрессии, является фундаментальным шагом в процессе моделирования (см. раздел "*Диагностика регрессии*" далее в этой главе).

Упорядоченные факторные переменные

Некоторые факторные переменные отражают уровни фактора; они называются *упорядоченными факторными переменными* или *упорядоченными категориальными переменными*. Например, категория качества ссуды может быть *A*, *B*, *C* и т. д. — каждая категория несет в себе больший риск, чем предыдущая категория. Нередко упорядоченные факторные переменные могут быть конвертированы в числовые значения и использоваться как есть. Например, переменная *BldgGrade* — это упорядоченная факторная переменная. Несколько типов категория качества приведены в табл. 4.1. Хотя категории качества имеют конкретный смысл, числовое значение упорядочено снизу вверх, соответствуя домам более высокого качества. В регрессионной модели *house_lm*, подгонка которой была выполнена в разделе "*Множественная линейная регрессия*" ранее в этой главе, категория *BldgGrade* трактовалась как числовая переменная.

Таблица 4.1. Построение категорий качества и числовых эквивалентов

Значение	Описание
1	Низкобюджетное
2	Ниже среднего
5	Порядочное
10	Очень хорошее
12	Роскошное
13	Особняк

Рассмотрение упорядоченных факторов в качестве числовой переменной сохраняет информацию, содержащуюся в упорядочении, которая будет потеряна, если его конвертировать в фактор.

Ключевые идеи для факторных переменных в регрессии

- Факторные переменные должны быть конвертированы в числовые переменные для их использования в регрессии.
- Наиболее часто встречающийся метод кодирования факторной переменной с P четко различимыми значениями — представлять их с использованием $P-1$ фиктивных переменных.
- Факторная переменная с многочисленными переменными даже в очень больших наборах данных может потребовать консолидации в переменную с меньшим числом уровней.
- Некоторые факторы имеют упорядоченные уровни и могут быть представлены в качестве единой числовой переменной.

Интерпретирование уравнения регрессии

В науке о данных самое важное применение регрессии состоит в предсказании зависимой переменной (исхода). В некоторых случаях, однако, получение более глубокого представления непосредственно из самого уравнения, чтобы понять природу связи между предсказателями и исходом, является ценностью. В данном разделе приводятся рекомендации по обследованию уравнения регрессии и его интерпретированию.

Ключевые термины для интерпретирования уравнения регрессии

Коррелированные переменные (correlated variables)

Когда предсказательные переменные высоко коррелированы, сложно интерпретировать отдельные коэффициенты.

Мультиколлинеарность (multicollinearity)

Когда предсказательные переменные имеют идеальную или почти идеальную корреляцию, регрессия может быть нестабильной либо ее невозможно вычислить.

Синонимы: коллинеарность, солинейность.

Искажающие переменные (confounding variables)

Важный предсказатель, который, если его опустить, приводит к мнимым связям в уравнении регрессии.

Синоним: спутывающие переменные.

Главные эффекты (main effects)

Связь между предсказательной переменной и переменной исхода, которая не зависит от других переменных.

Взаимодействия (interactions)

Взаимозависимая связь между двумя или несколькими предсказателями и исходом (откликом).

Коррелированные предсказатели

В множественной регрессии предсказательные переменные часто коррелируют друг с другом. В качестве примера давайте проэкстаменуем коэффициенты регрессии для модели `step_lm`, подогнанной в разделе *"Отбор модели и пошаговая регрессия"* ранее в этой главе:

R:

```
step_lm$coefficients
      (Intercept)      SqFtTotLiving
      6.227632e+06      1.865012e+02
      Bathrooms      Bedrooms
      4.472172e+04     -4.980718e+04
      BldgGrade PropertyTypeSingle Family
      1.391792e+05      2.332869e+04
PropertyTypeTownhouse      SqFtFinBasement
      9.221625e+04      9.039911e+00
      YrBuilt
     -3.592468e+03
```

Python:

```
print(f'Пересечение: {best_model.intercept_:.3f}')
print('Коэффициенты:')
for name, coef in zip(best_variables, best_model.coef_):
    print(f' {name}: {coef}')
```

Коэффициент для спален `Bedrooms` является отрицательным! Из этого вытекает, что добавление спальни в дом уменьшит его стоимость. Как это может быть? Это вызвано тем, что предсказательные переменные коррелированы: в более крупных домах наблюдается тенденция к большему количеству спален, и именно размер дома управляет его стоимостью, а не число спален. Рассмотрим два дома одного и того же размера: разумно ожидать, что дом с бóльшим числом спален, но меньших по площади будет считаться менее желательным.

Наличие коррелированных предсказателей усложняет интерпретирование знака и значения регрессионных коэффициентов (и может раздуть стандартную ошибку оценочных значений). Переменные для спален, размера дома и числа ванных комнат — все они являются коррелированными. Это иллюстрируется следующим ниже

примером на R, в котором выполняется подгонка еще одной регрессии после удаления переменных `SqFtTotLiving` (жилая площадь), `SqFtFinBasement` (цокольная площадь) и `Bathrooms` (ванные комнаты) из уравнения:

```
update(step_lm, . ~ . - SqFtTotLiving - SqFtFinBasement - Bathrooms)
```

Call:

```
lm(formula = AdjSalePrice ~ Bedrooms + BldgGrade + PropertyType +  
    YrBuilt, data = house, na.action = na.omit)
```

Coefficients:

(Intercept)	Bedrooms
4913973	27151
BldgGrade	PropertyTypeSingle Family
248998	-19898
PropertyTypeTownhouse	YrBuilt
-47355	-3212

Функция `update` может использоваться для добавления или удаления переменных из модели. Теперь коэффициент для спален является положительным — в соответствии с тем, что мы ожидали бы (хотя теперь, когда эти переменные были исключены, он на самом деле действует как эрзац для размера дома).

В Python нет никакого эквивалента функции `update` языка R. Нам нужно выполнить повторную подгонку модели с использованием модифицированного списка предсказателей:

```
predictors = ['Bedrooms', 'BldgGrade', 'PropertyType', 'YrBuilt']  
outcome = 'AdjSalePrice'
```

```
X = pd.get_dummies(house[predictors], drop_first=True)
```

```
reduced_lm = LinearRegression()  
reduced_lm.fit(X, house[outcome])
```

Коррелированные переменные являются лишь одной трудностью, связанной с интерпретированием регрессионных коэффициентов. В модели `house_lm` нет переменной, которая отвечает за местоположение дома, и модель смешивает очень разные типы районов. Местоположение может быть *искажающей* переменной (последующее обсуждение см. в разделе *"Искажающие переменные"* далее в этой главе).

Мультиколлинеарность

Предельный случай коррелированных переменных производит мультиколлинеарность — условие, в котором существует избыток среди предсказательных переменных. Идеальная мультиколлинеарность случается, когда одна предсказательная переменная может быть выражена как линейная комбинация других. Мультиколлинеарность происходит, когда:

- ♦ переменная включается в состав модели многократно по ошибке;
- ♦ из факторной переменной создаются P фиктивных переменных вместо $P - 1$ (см. раздел "Факторные переменные в регрессии" ранее в этой главе);
- ♦ две переменные почти идеально коррелированы друг с другом.

Вопрос мультиколлинеарности в регрессии должен быть решен — переменные необходимо исключать до тех пор, пока мультиколлинеарность не исчезнет. Регрессия не имеет хорошо определенного решения в присутствии идеальной мультиколлинеарности. Многие пакеты, в том числе на языках R и Python, автоматически обрабатывают некоторые типы мультиколлинеарности. Например, если дважды включить переменную `SqFtTotLiving` в регрессию данных `house`, то результаты будут такими же, что и для модели `house_lm`. В случае неидеальной мультиколлинеарности вычислительная система может получить решение, но результаты могут быть нестабильными.



Мультиколлинеарность не представляет какую-то особую проблему для регрессионных методов, таких как деревья, кластеризация и ближайшие соседи, и в таких методах рекомендуется оставлять P фиктивных переменных (вместо $P - 1$). Тем не менее даже в этих методах избыточность в предсказательных переменных по-прежнему представляет собой достоинство.

Искажающие переменные

С коррелированными переменными проблема состоит во включении переменных в состав: включение разных переменных, которые имеют похожую предсказательную связь с откликом. С *искажающими переменными* проблемой является исключение переменных из состава: важная переменная не включена в уравнение регрессии. Наивная интерпретация коэффициентов уравнения может привести к несостоятельным заключениям.

Возьмем, например, уравнение регрессии `house_lm` для округа Кинг из *раздела "Пример: данные жилого фонда округа Кинг" ранее в этой главе*. Регрессионные коэффициенты `SqFtLot` (площадь земельного участка), `Bathrooms` (ванные комнаты) и `Bedrooms` (спальни) — все являются отрицательными. Исходная регрессионная модель не содержит переменную, которая представляла бы местоположение — очень важный предсказатель цены на недвижимость. Для того чтобы смоделировать местоположение, включите переменную `ZipGroup` (группа почтовых индексов), которая отнесет почтовый индекс в одну из пяти групп от наименее дорогого (1) до самого дорогого (5)⁸.

```
lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot +
  Bathrooms + Bedrooms +
```

⁸ В округе Кинг имеется 82 почтовых индекса, по ряду из которых зарегистрировано всего несколько продаж. Как альтернатива прямому использованию почтового индекса в качестве факторной переменной, переменная `ZipGroup` кластеризует похожие почтовые индексы в одну группу. Подробности см. в разделе "Факторные переменные с многочисленными уровнями" ранее в этой главе.

```
BldgGrade + PropertyType + ZipGroup,
data=house, na.action=na.omit)
```

Coefficients:

(Intercept)	SqFtTotLiving
-6.709e+05	2.112e+02
SqFtLot	Bathrooms
4.692e-01	5.537e+03
Bedrooms	BldgGrade
-4.139e+04	9.893e+04
PropertyTypeSingle Family	PropertyTypeTownhouse
2.113e+04	-7.741e+04
ZipGroup2	ZipGroup3
5.169e+04	1.142e+05
ZipGroup4	ZipGroup5
1.783e+05	3.391e+05

Переменная `ZipGroup`, несомненно, является очень важной: дом в самой дорогой группе почтовых индексов оценивается как имеющий более высокую продажную цену почти на 340 000 \$. Коэффициенты `SqFtLot` и `Bathrooms` теперь являются положительными, и добавление ванной увеличивает продажную цену на 7500 \$.

Коэффициент для `Bedrooms` по-прежнему является отрицательным. Хотя этот феномен противоречит логике, он хорошо известен в торговле недвижимостью. Наличие у домов одинаковой общей жилой площади и большего количества и, следовательно, меньших по размеру спален ассоциируется с менее ценными домами.

Взаимодействия и главные эффекты

Специалисты-статистики предпочитают проводить различие между *главными эффектами*, или независимыми переменными, и взаимодействиями между главными эффектами. Главные эффекты — это то, что подразумевается под *предсказательными переменными* в уравнении регрессии. Неявное допущение, когда в модели используются только главные эффекты, состоит в том, что связь между предсказательной переменной и откликом не зависит от других предсказательных переменных. Зачастую это не так.

Например, модель, подогнанная к данным жилого фонда округа Кинг в разделе *"Искажающие переменные"* ранее в этой главе, включает несколько переменных в качестве главных эффектов, в том числе `ZipCode`. Местоположение в торговле недвижимостью — краеугольный камень, и естественно допустить, что связь между, скажем, размером дома и продажной ценой зависит от местоположения. Большой дом, построенный в районе с низкой арендной платой, не будет сохранять одинаковую стоимость, что и большой дом, построенный в дорогом районе. В R взаимодействия между переменными можно включить в расчеты, используя оператор `*`. Для данных округа Кинг следующий ниже пример выполняет подгонку взаимодействия между переменными `SqFtTotLiving` и `ZipGroup`:

```
lm(AdjSalePrice ~ SqFtTotLiving*ZipGroup + SqFtLot +
    Bathrooms + Bedrooms + BldgGrade + PropertyType,
    data=house, na.action=na.omit)
```

Coefficients:

(Intercept)	SqFtTotLiving
-4.919e+05	1.176e+02
ZipGroup2	ZipGroup3
-1.342e+04	2.254e+04
ZipGroup4	ZipGroup5
1.776e+04	-1.555e+05
SqFtLot	Bathrooms
7.176e-01	-5.130e+03
Bedrooms	BldgGrade
-4.181e+04	1.053e+05
PropertyTypeSingle Family	PropertyTypeTownhouse
1.603e+04	-5.629e+04
SqFtTotLiving:ZipGroup2	SqFtTotLiving:ZipGroup3
3.165e+01	3.893e+01
SqFtTotLiving:ZipGroup4	SqFtTotLiving:ZipGroup5
7.051e+01	2.298e+02

Результирующая модель имеет четыре новых члена: SqFtTotLiving:ZipGroup2, SqFtTotLiving:ZipGroup3 и т. д.

В Python мы должны использовать пакет `statsmodels` для тренировки линейных регрессионных моделей взаимодействиями. Этот пакет был разработан аналогично языку R и позволяет определять модели с использованием формульного интерфейса:

```
model = smf.ols(formula='AdjSalePrice ~ SqFtTotLiving*ZipGroup + SqFtLot + ' +
    'Bathrooms + Bedrooms + BldgGrade + PropertyType', data=house)
results = model.fit()
results.summary()
```

Пакет `statsmodels` берет на себя категориальные переменные (например, `ZipGroup[T.1]`, `PropertyType[T.Single Family]`) и термины взаимодействия (например, `SqFtTotLiving:ZipGroup[T.1]`).

Местоположение и размер дома, похоже, имеют сильное взаимодействие. Для дома в самой низкой группе `ZipGroup` наклон прямой такой же, что и наклон для главного эффекта `SqFtTotLiving`, который составляет 177 \$ за кв. фут (это обусловлено тем, что R использует опорное кодирование для факторных переменных; см. раздел "Факторные переменные в регрессии" ранее в этой главе). Для дома в самой высокой группе `ZipGroup` наклон прямой равен сумме главного эффекта плюс `SqFtTotLiving:ZipGroup5`, или 177 \$ + 230 \$ = 447 \$ за кв. фут. Другими словами, приращение квадратного фута в группе наиболее дорогих почтовых индексов повышает предсказанную продажную цену почти в 2,7 раза по сравнению с повышением в группе наименее дорогих почтовых индексов.



Отбор модели с помощью членов взаимодействия

В задачах с участием многочисленных переменных бывает трудно принять решение о том, какие члены уравнения, характеризующие взаимодействия между переменными, должны быть включены в модель. Чаще всего принимается несколько разных подходов.

- В некоторых задачах априорные знания и интуиция могут направлять выбор того, какие члены взаимодействия следует включать в модель.
- Пошаговый отбор (см. раздел "*Отбор модели и пошаговая регрессия*" ранее в этой главе) может использоваться для отсеивания различных моделей.
- Штрафная регрессия способна автоматически выполнять подгонку к крупному набору возможных членов взаимодействия.
- По-видимому, наиболее часто встречающимся подходом является использование *древесных моделей*, а также их потомков, *случайного леса* и *градиентно-бустированных деревьев*. Указанный класс моделей автоматически выполняет поиск оптимальных членов взаимодействия (см. раздел "*Древесные модели*" главы 6).

Ключевые идеи для интерпретирования уравнения регрессии

- Вследствие корреляции между предсказателями необходимо проявлять осторожность в интерпретации коэффициентов в множественной линейной регрессии.
- Мультиколлинеарность может вызывать числовую нестабильность в подгонке уравнения регрессии.
- Искажающая переменная является важным предсказателем, который не учтен в модели и может привести к уравнению регрессии с мнимыми связями.
- Член уравнения, характеризующий взаимодействие между двумя переменными, необходим, если связь между переменными и откликом является взаимозависимой.

Диагностика регрессии

В объяснительном моделировании (т. е. в контексте исследования) разнообразные шаги в дополнение к упомянутым ранее метрикам (см. раздел "*Оценивание результативности модели*" ранее в этой главе) предпринимаются для того, чтобы определить, насколько хорошо модель подогнана к данным; большинство основывается на анализе остатков. Эти шаги непосредственно не решают вопрос предсказательной точности, но они способны обеспечить полезное углубленное понимание предсказательной обстановки.

Ключевые термины для диагностики регрессии

Стандартизированные остатки (standardized residuals)

Остатки, деленные на стандартную ошибку остатков.

Выбросы (outliers)

Записи (или значения исхода), которые находятся вдали от остальной части данных (или предсказанного исхода).

Влиятельное значение (influential value)

Значение (или запись), присутствие или отсутствие которого играет важную роль в уравнении регрессии.

Плечо (leverage)

Степень влиятельности, которую отдельная запись имеет на уравнение регрессии.

Синонимы: рычаг, hat-значение.

Ненормальные остатки (non-normal residuals)

Ненормально распределенные остатки могут аннулировать некоторые технические условия регрессии; вместе с тем они обычно не являются предметом озабоченности в науке о данных.

Гетероскедастичность (heteroskedasticity)

Ситуация, когда некоторые диапазоны исхода показывают остатки с более высокой дисперсией (что может говорить о предсказателе, который в уравнении отсутствует).

Графики частных остатков (partial residual plots)

Диагностический график для выявления связи между переменной исхода и одиночным предсказателем.

Синонимы: график с добавляемыми переменными, график частичных остатков.

Выбросы

Вообще говоря, предельное значение, так называемый выброс, есть такое значение, которое находится вдали от большинства других наблюдений. Как и в случае с оценками центрального положения и вариабельности (*см. раздел "Оценки центрального положения" и "Оценки вариабельности" главы 1*), где выбросы необходимо улаживать во избежание проблем, выбросы могут вызывать проблемы с регрессионными моделями. В регрессии выброс — это запись, чье фактическое значение у находится вдали от предсказанного значения. Выбросы можно обнаружить путем обследования стандартизированного остатка, т. е. остатка, деленного на стандартную ошибку остатков.

Не существует статистической теории, которая отделяет выбросы от невыбросов. Вместо этого существуют (произвольные) эмпирические правила в отношении то-

го, насколько далеко от основной части данных должно находиться наблюдение, чтобы его можно было назвать выбросом. Например, в коробчатой диаграмме выбросами являются те точки данных, которые находятся слишком высоко или слишком низко от границ коробки (*см. раздел "Процентили и коробчатые диаграммы" главы 1*), где "слишком" означает величину, превышающую "число 1,5, умноженное на межквартильный размах". В регрессии метрика стандартизированного остатка типично используется для определения того, не относится ли запись к классу выбросов. Стандартизированные остатки можно интерпретировать как "число стандартных ошибок от регрессионной прямой".

Давайте в R выполним подгонку регрессии к данным продаж домов в округе Кинг для всех продаж в почтовом индексе 98105:

```
house_98105 <- house[house$ZipCode == 98105,]
lm_98105 <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
               Bedrooms + BldgGrade, data=house_98105)
```

В Python:

```
house_98105 = house.loc[house['ZipCode'] == 98105, ]

predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']
outcome = 'AdjSalePrice'

house_outlier = sm.OLS(house_98105[outcome],
                       house_98105[predictors].assign(const=1))
result_98105 = house_outlier.fit()
```

Мы извлекаем стандартизированные остатки при помощи функции `rstandard` и получаем индекс наименьшего остатка при помощи функции `order`:

```
sresid <- rstandard(lm_98105)
idx <- order(sresid)
sresid[idx[1]]
      20431
    -4.326732
```

В пакете `statsmodels` используйте класс `OLSInfluence`, чтобы проанализировать остатки:

```
influence = OLSInfluence(result_98105)
sresiduals = influence.resid_studentized_internal
sresiduals.idxmin(), sresiduals.min()
```

Самая большая завышенная оценка из модели составляет более четырех стандартных ошибок над регрессионной прямой, что соответствует завышенной оценке в 757 753 \$. Исходная запись данных, которая соответствует этому выбросу, в R такова:

```
house_98105[idx[1], c('AdjSalePrice', 'SqFtTotLiving', 'SqFtLot',
                     'Bathrooms', 'Bedrooms', 'BldgGrade')]
```


AdjSalePrice	SqFtTotLiving	SqFtLot	Bathrooms	Bedrooms	BldgGrade	
(dbl)	(int)	(int)	(dbl)	(int)	(int)	
1	119748	2900	7276	3	6	7

В Python:

```
outlier = house_98105.loc[sresiduals.idxmin(), :]  
print('AdjSalePrice', outlier[outcome])  
print(outlier[predictors])
```

В данном случае, по всей видимости, что-то не так с записью: дом такого размера в типичной ситуации продается за намного бóльшую цену, чем 119 748 \$ в этом почтовом индексе. На рис. 4.4 показана выдержка из установочного акта этой продажи: ясно, что продажа предусматривала всего лишь долю в собственности. В этом случае выброс соответствует продаже, которая является аномальной и не должна быть включена в регрессию. Кроме того, выбросы могут быть результатом других проблем, таких как ошибочный ввод данных из-за "толстого пальца" или несовпадения единиц измерения (например, сообщение о продаже в тысячах долларов вместо просто долларов)⁹.

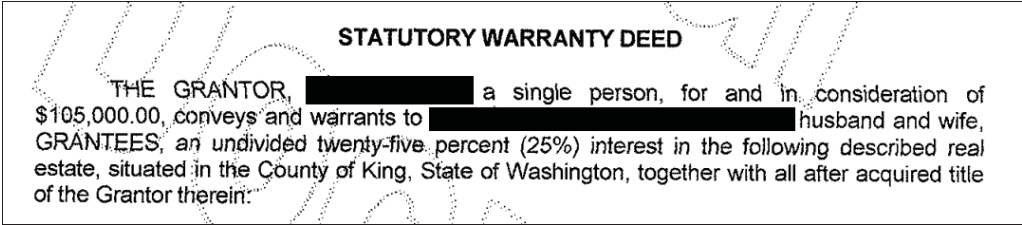


Рис. 4.4. Установочный акт для самого крупного отрицательного остатка

Для задач с привлечением больших данных выбросы обычно не представляют проблему в подгонке регрессии, используемой в предсказании новых данных. Однако выбросы занимают центральное место в обнаружении аномалий, где отыскание выбросов является смыслом всего дела. Кроме того, выброс может соответствовать случаю мошенничества или нечаянного действия. В любом случае обнаружение выбросов бывает критической потребностью бизнеса.

Влиятельные значения

Значение, отсутствие которого значительно изменит уравнение регрессии, называется *влиятельным наблюдением*. В регрессии такое значение не обязательно должно ассоциироваться с крупным остатком. В качестве примера рассмотрим регрессионные прямые на рис. 4.5. Пунктирная прямая соответствует регрессии со всеми дан-

⁹ Перевод установочного акта: правоотчуждатель, отдельное лицо, за вознаграждение в размере 105 000,00 \$ передает и гарантирует мужу и жене, правоприобретателям, неразделенную двадцатипятипроцентную (25%) долю в описанном ниже недвижимом имуществе, расположенном в округе Кинг, штат Вашингтон, вместе со всем титулом правоотчуждателя, приобретаемом в нем после. — Прим. перев.

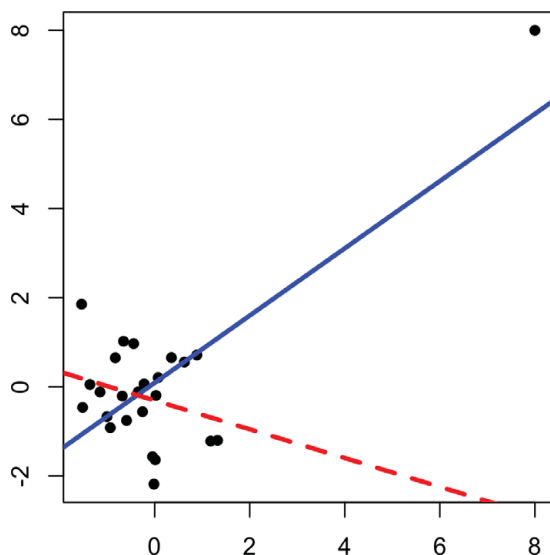


Рис. 4.5. Пример влиятельной точки данных в регрессии

ными, тогда как сплошная прямая — регрессии с точкой далеко в правом верхнем углу. Безусловно, это значение данных имеет огромное влияние на регрессию, хотя она не связана с крупным выбросом (из полной регрессии). Считается, что это значение данных имеет в регрессии сильное плечо (или рычаг).

В дополнение к стандартизированным остаткам (см. раздел "Выбросы" ранее в этой главе) специалисты-статистики разработали несколько метрик для определения влиятельности отдельной записи на регрессию. Часто встречающейся мерой плеча является *hat-значение*; значения выше $2(P+1)/n$ говорят о данных с высоким плечом¹⁰.

Еще одна метрика — *расстояние Кука*, которое определяет влияние как комбинацию плеча и размера остатка. Эмпирическое правило состоит в том, что наблюдение имеет высокое влияние, если расстояние Кука превышает $4/(n - P - 1)$.

График влиятельности, или *пузырьковый график*, объединяет стандартизированные остатки, *hat-значение* и расстояние Кука в одном графике. На рис. 4.6 представлен график влиятельности для данных жилого фонда округа Кинг. Указанный график может быть создан при помощи следующего фрагмента кода R:

```
std_resid <- rstandard(lm_98105)
cooks_D <- cooks.distance(lm_98105)
hat_values <- hatvalues(lm_98105)
plot(subset(hat_values, cooks_D > 0.08), subset(std_resid, cooks_D > 0.08),
     xlab='hat_values', ylab='std_resid',
     cex=10*sqrt(subset(cooks_D, cooks_D > 0.08)), pch=16, col='lightgrey')
```

¹⁰ Термин "hat-значение" происходит от понятия hat-матрицы, т. е. проекционной матрицы с проекцией на пространство регрессоров. Множественная линейная регрессия может быть выражена формулой $\mathbf{Y} = \mathbf{H}\mathbf{Y}$, где \mathbf{H} — это hat-матрица. hat-значения соответствуют диагонали в \mathbf{H} .

```
points(hat_values, std_resid, cex=10*sqrt(cooks_D))
abline(h=c(-2.5, 2.5), lty=2)
```

Ниже приведен фрагмент кода Python для создания похожего рисунка.

```
influence = OLSInfluence(result_98105)
fig, ax = plt.subplots(figsize=(5, 5))
ax.axhline(-2.5, linestyle='--', color='C1')
ax.axhline(2.5, linestyle='--', color='C1')
ax.scatter(influence.hat_matrix_diag, influence.resid_studentized_internal,
           s=1000 * np.sqrt(influence.cooks_distance[0]),
           alpha=0.5)
ax.set_xlabel('hat-значения')
ax.set_ylabel('Стандартные остатки')
```

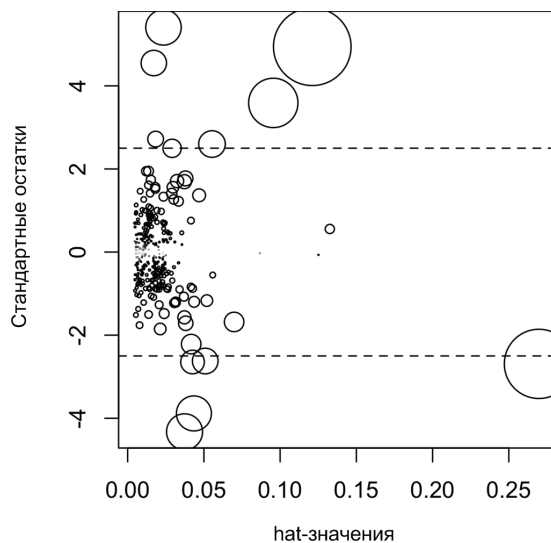


Рис. 4.6. График для определения того, какие наблюдения имеют высокое влияние; точки с расстоянием Кука больше 0.08 выделены серым цветом

Судя по всему, имеется несколько точек данных, которые оказывают крупное влияние в регрессии. Расстояние Кука может быть вычислено при помощи функции `cooks.distance`, и вы можете использовать `hatvalues` для вычисления диагностики. `hat`-Значения откладываются на оси x , остатки — на оси y , и размер точек связан со значением расстояния Кука.

В табл. 4.2 приведено сравнение регрессии с полным набором данных и с дальними очень влиятельными точками данных (расстояние Кука больше 0,08). Коэффициент регрессии для `Bathrooms` изменяется вполне кардинально¹¹.

¹¹ Коэффициент для `Bathrooms` становится отрицательным, что противоречит логике. Местоположение не было принято во внимание, и почтовый индекс 98105 содержит районы несопоставимых типов домов. Обсуждение темы искажающих переменных см. в разделе *"Искажающие переменные"* ранее в этой главе.

Таблица 4.2. Сравнение коэффициентов регрессии с полными данными и с дальними влиятельными данными

	Оригинал	Дальние влиятельные
(пересечение)	−772 550	−647 137
SqFtTotLiving	210	230
SqFtLot	39	33
Bathrooms	2282	−16 132
Bedrooms	−26 320	−22 888
BldgGrade	130 000	114 871

Для целей подгонки регрессии, которая надежно предсказывает будущие данные, выявление влиятельных наблюдений полезно только в меньших по размеру наборах данных. Для регрессий с привлечением большого числа записей существует малая вероятность того, что какое-либо наблюдение будет нести достаточный вес, чтобы вызывать предельное влияние на подогнанное уравнение (хотя регрессия может все же иметь большие выбросы). Для целей обнаружения аномалий тем не менее выявление влиятельных наблюдений бывает очень полезным.

Гетероскедастичность, ненормальность и коррелированные ошибки

В статистике значительное внимание уделяется распределению остатков. Оказывается, что обычные наименьшие квадраты (см. раздел "*Наименьшие квадраты*" ранее в этой главе) являются несмещенными, и в некоторых случаях "оптимальным" оценщиком, в условиях широкого диапазона допущений о природе распределения. Это означает, что в большинстве задач исследователям данных не приходится слишком беспокоиться о распределении остатков.

Распределение остатков является релевантным главным образом для подтверждения достоверности формального статистического вывода (проверок гипотез и p -значений), что имеет минимальное значение для исследователей данных, озабоченных главным образом предсказательной точностью. Нормально распределенные ошибки — это признак того, что модель завершена; ошибки, которые не являются нормально распределенными, указывают на то, что модель, возможно, что-то упускает. Для того чтобы формальный статистический вывод был достоверным, принимается допущение о том, что остатки нормально распределены, имеют одинаковую дисперсию и являются независимыми. Одной из областей, где это может представлять интерес для исследователей данных, является стандартный расчет доверительных интервалов для предсказанных значений, которые основаны на допущениях об остатках (см. раздел "*Доверительные и предсказательные интервалы*" ранее в этой главе).

Гетероскедастичность — это отсутствие постоянной остаточной дисперсии по всему диапазону предсказанных значений. Другими словами, для некоторых частей

диапазона ошибки больше, чем для других. Визуализация данных является удобным способом анализа остатков.

Следующий фрагмент кода на R строит график с абсолютными остатками против предсказанных значений для регрессии `lm_98105`, подогаанной в *разделе "Выбросы" ранее в этой главе*.

```
df <- data.frame(resid = residuals(lm_98105), pred = predict(lm_98105))
ggplot(df, aes(pred, abs(resid))) + geom_point() + geom_smooth()
```

На рис. 4.7 представлен результирующий график. Используя `geom_smooth`, очень легко наложить сглаженную кривую абсолютных остатков. Данная функция вызывает метод `loess` (LOcally Estimated Scatterplot Smoothing — локально оцениваемое сглаживание диаграммы рассеяния), чтобы произвести сглаженную оценку связи между переменными на оси *x* и оси *y* в диаграмме рассеяния (см. *врезку "Сглаживатели диаграмм рассеяния" далее в этой главе*).

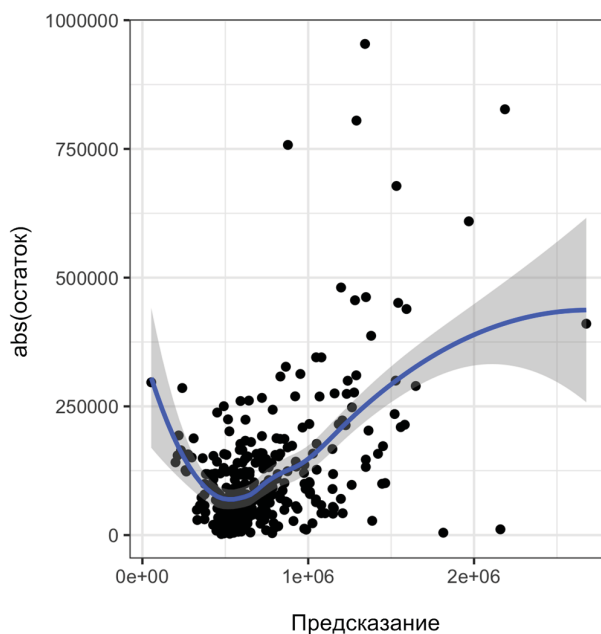


Рис. 4.7. График абсолютного значения остатков против предсказанных значений

Совершенно очевидно, что дисперсия остатков имеет тренд на увеличение для домов с более высокой стоимостью, но является также крупной для домов с более низкой стоимостью. Приведенный выше график говорит о том, что регрессия `lm_98105` имеет гетероскедастичные ошибки.



Почему исследователя данных должна интересовать гетероскедастичность?

Гетероскедастичность говорит о том, что ошибки предсказания отличаются для разных диапазонов предсказанного значения и могут свидетельствовать о незаконченной модели. Например, гетероскедастичность в `lm_98105` может

говорить о том, что регрессия что-то не учла в высокодиапазонных и низкодиапазонных домах.

На рис. 4.8 приведена гистограмма стандартизированных остатков для регрессии `lm_98105`. Ее распределение однозначно имеет более длинные хвосты, чем у нормального распределения, и демонстрирует умеренную асимметричность в сторону более крупных остатков.

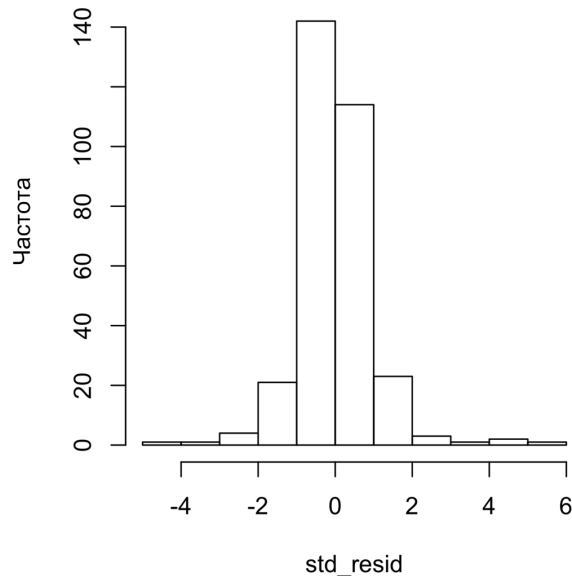


Рис. 4.8. Гистограмма остатков из регрессии данных жилого фонда

Специалисты-статистики могут также проверить допущение о том, что ошибки являются независимыми. Это особенно верно в отношении данных, которые собираются в течение долгого времени. Статистика Дарбина—Уотсона (Durbin—Watson) может использоваться для обнаружения того, существует ли значительная автокорреляция в регрессии с участием данных временного ряда. Если ошибки регрессионной модели коррелируют, то эта информация может быть полезна при составлении краткосрочных прогнозов и должна быть встроена в модель. См. книгу "Практическое прогнозирование временных рядов с помощью R, 2-е изд., Галита Шмуэли и Кеннета Лихтендала (Practical Time Series Forecasting with R, 2nd ed., by Galit Shmueli and Kenneth Lichtendahl, Axelrod Schnall, 2018), чтобы узнать больше о том, как встраивать автокорреляционную информацию в регрессионные модели для данных временных рядов. Если целью являются долгосрочные прогнозы или объяснительные модели, то избыточные автокоррелированные данные на микроуровне могут отвлекать внимание. В этом случае сглаживание, или в первую очередь менее гранулярный сбор данных, может быть в порядке вещей.

Хотя регрессия может нарушить одно из допущений о природе распределения, должно ли это нас заботить? В большинстве случаев в науке о данных первоочередным объектом интереса является предсказательная точность, и поэтому какой-

то анализ гетероскедастичности не помешает. Вы можете обнаружить, что в данных имеется некий сигнал, который ваша модель не уловила. Однако удовлетворение допущений о природе распределения просто ради подтверждения достоверности формального статистического вывода (p -значения, F -статистики и т. д.) не представляет для исследователя данных какую-то особую важность.



Сглаживатели диаграмм рассеяния

Регрессия — это прежде всего моделирование связи между откликом и предсказательными переменными. В оценивании регрессионной модели полезно использовать сглаживатель диаграмм рассеяния, чтобы визуально высвечивать связи между двумя переменными.

Например, на рис. 4.7 сглаженная кривая связи между абсолютными остатками и предсказанным значением показывает, что дисперсия остатков зависит от значения остатка. В данном случае использовалась функция `loess`; указанная функция работает путем неоднократной подгонки серии локальных регрессий к смежным подмножествам, чтобы выработать сглаженную кривую. Хотя сглаживатель `loess`, вероятно, встречается наиболее часто, в R имеются и другие сглаживатели, такие как суперсглаженная кривая (`supsmu`) и ядерное сглаживание (`ksmooth`). Для целей оценивания регрессионной модели, как правило, отсутствует необходимость беспокоиться о деталях сглаживателей диаграмм рассеивания.

Графики частных остатков и нелинейность

Графики частных остатков — это способ визуализации того, насколько хорошо вычисленная подгонка объясняет связь между предсказателем и исходом. Главная идея графика частных остатков состоит в изолировании связи между предсказательной переменной и откликом, *принимая в расчет все другие предсказательные переменные*. Частный остаток можно рассматривать, как "синтетическое результирующее" значение, объединяющее предсказание на основе отдельного предсказателя с фактическим остатком от полного уравнения регрессии. Частный остаток для предсказателя X_i — это обычный остаток плюс член регрессии, связанный с X_i .

$$\text{Частный остаток} = \text{Остаток} + \hat{b}_i X_i,$$

где \hat{b}_i — это оценочный регрессионный коэффициент. Функция `predict` в R имеет возможность возвращать отдельные члены регрессии $\hat{b}_i X_i$:

```
terms <- predict(lm_98105, type='terms')
partial_resid <- resid(lm_98105) + terms
```

График частных остатков показывает X_i предсказателя на оси x и частные остатки — на оси y . Использование пакета `ggplot2` упрощает наложение сглаженной кривой частных остатков.

```
df <- data.frame(SqFtTotLiving = house_98105[, 'SqFtTotLiving'],
                 Terms = terms[, 'SqFtTotLiving'],
                 PartialResid = partial_resid[, 'SqFtTotLiving'])
```

```
ggplot(df, aes(SqFtTotLiving, PartialResid)) +
  geom_point(shape=1) + scale_shape(solid = FALSE) +
  geom_smooth(linetype=2) +
  geom_line(aes(SqFtTotLiving, Terms))
```

Пакет `statsmodels` имеет метод `sm.graphics.plot_ccpr`, который создает похожий график частных остатков:

```
sm.graphics.plot_ccpr(result_98105, 'SqFtTotLiving')
```

Графики R и Python различаются константным сдвигом. В R константа добавляется таким образом, что среднее значение членов равно нулю.

Результирующий график показан в рис. 4.9. Частный остаток — это оценка вклада, который `SqFtTotLiving` вносит в продажную цену. Легко видно, что связь между `SqFtTotLiving` и продажной ценой нелинейна. Регрессионная прямая недооценивает продажную цену для домов площадью менее 1000 кв. футов и переоценивает цену для домов с площадью между 2000 и 3000 кв. футов. Выше 4000 кв. футов имеется слишком мало точек данных, чтобы сделать выводы для этих домов.

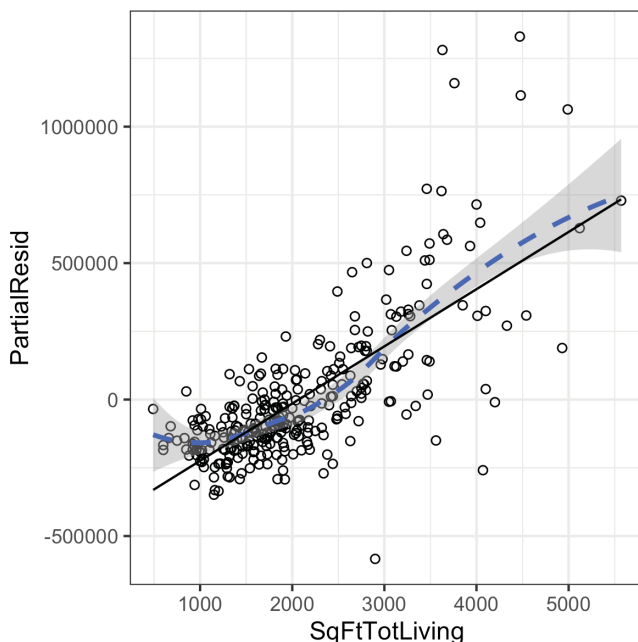


Рис. 4.9. График частных остатков для переменной `SqFtTotLiving`

Эта нелинейность имеет смысл в данном случае: добавление 500 футов в малый дом оказывает гораздо большее влияние, чем добавление 500 футов в крупный дом. Это свидетельствует о том, что вместо простого линейного члена для `SqFtTotLiving` следует рассмотреть нелинейный член (см. раздел "Многочленная и сплайновая регрессия" далее в этой главе).

Ключевые идеи для диагностики регрессии

- Хотя выбросы могут вызывать проблемы для малых наборов данных, первоочередной интерес к выбросам заключается в выявлении проблем с данными или в локализации аномалий.
- Отдельные записи (включая регрессионные выбросы) могут оказывать большое влияние на уравнение регрессии с малыми данными, но этот эффект размывается в больших данных.
- Если регрессионная модель используется для формального статистического вывода (p -значения и т. п.), то должны быть проверены некоторые допущения о природе распределении остатков. В общем случае, однако, распределение остатков не имеет критическую важность в науке о данных.
- График частных остатков может использоваться для качественной диагностики подгонки для каждого члена регрессии, возможно приводя к спецификации альтернативной модели.

Многочленная и сплайновая регрессия

Связь между откликом и предсказательной переменной не обязательно является линейной. Отклик на дозу препарата часто не является линейным: удвоение дозы обычно не приводит к удвоенному отклику. Спрос на продукт не является линейной функцией числа долларов, потраченных на маркетинг, поскольку в какой-то момент спрос, вероятно, будет насыщен. Существует несколько способов, которыми регрессия может быть расширена для получения этих нелинейных эффектов.

Ключевые термины для многочленной и сплайновой регрессии

Многочленная регрессия (polynomial regression)

Добавляет в регрессию полиномиальные члены (квадраты, кубы и т. д.).

Синоним: полиномиальная регрессия, параболическая регрессия.

Сплайновая регрессия (spline regression)

Подгонка гладкой кривой серией полиномиальных сегментов.

Узлы (knots)

Значения, которые отделяют сплайновые сегменты.

Обобщенные аддитивные модели (generalized additive models)

Сплайновые модели с автоматизированным отбором узлов.

Синоним: GAM.



Нелинейная регрессия

Когда специалисты-статистики говорят о нелинейной регрессии, они всегда имеют в виду модели, которые не могут быть подогаданы с помощью наименьших квадратов. Какие модели не являются нелинейными? По существу, это все модели, где отклик не может быть выражен как линейная комбинация предсказателей или некая трансформация предсказателей. Подгонка нелинейных регрессионных моделей дается труднее и вычислительно является более интенсивной, поскольку такие модели требуют численной оптимизации. По этой причине, если это возможно, предпочтение отдается использованию линейной модели.

Многочлены

Многочленная регрессия предусматривает включение в состав уравнения регрессии полиномиальных членов. Использование многочленной регрессии практически берет начало с разработки непосредственно самой регрессии в статье Жергонна (Gergonne) в 1815 году. Например, квадратичная регрессия между откликом Y и предсказателем X примет следующую форму:

$$Y = b_0 + b_1X + b_2X^2 + e.$$

Многочленная регрессия может быть подогадана в R посредством функции `poly`. Например, следующий фрагмент кода выполняет подгонку квадратичного полинома для переменной `SqFtTotLiving` данными жилого фонда округа Кинг:

```
lm(AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot + BldgGrade + Bathrooms + Bedrooms,
    data=house_98105)
```

Call:

```
lm(formula = AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot +
    BldgGrade + Bathrooms + Bedrooms, data = house_98105)
```

Coefficients:

```
(Intercept)  poly(SqFtTotLiving, 2)1
-402530.47      3271519.49
poly(SqFtTotLiving, 2)2      SqFtLot
776934.02      32.56
BldgGrade      Bathrooms
135717.06      -1435.12
Bedrooms
-9191.94
```

В пакете `statsmodels` мы добавляем квадратный термин в определение модели с помощью `I(SqFtTotLiving**2)`:

```
model_poly = smf.ols(formula='AdjSalePrice ~ SqFtTotLiving + ' +
    '+ I(SqFtTotLiving**2) + ' +
    'SqFtLot + Bathrooms + Bedrooms + BldgGrade', data=house_98105)
```

```
result_poly = model_poly.fit()
result_poly.summary() ❶
```

❶ Пересечение и полиномиальные коэффициенты отличаются от аналогичных в R. Это связано с отличающимися имплементациями. Остальные коэффициенты и предсказания являются эквивалентными.

Теперь с переменной `SqFtTotLiving` связаны два коэффициента: один для линейного члена, другой для квадратичного члена.

График частных остатков (см. раздел "Графики частных остатков и нелинейность" ранее в этой главе) говорит о некоторой кривизне в уравнении регрессии, связанном с переменной `SqFtTotLiving`. Подогнанная линия более точно соответствует сглаженной кривой (см. раздел "Сплайны" далее в этой главе) частных остатков по сравнению с линейной подгонкой (рис. 4.10).

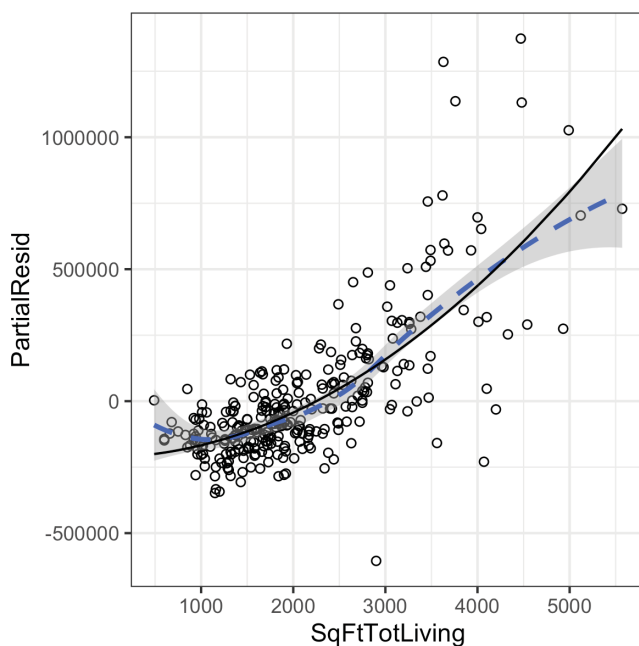


Рис. 4.10. Многочленная регрессия, подогнанная для переменной `SqFtTotLiving` (сплошная линия) против сглаженной кривой (пунктирная линия; см. следующий раздел о сплайнах)

Имплементация в пакете `statsmodels` работает только для линейных членов. Сопроводительный исходный код дает имплементацию, которая будет работать и для многочленной регрессии.

Сплайны

Многочленная регрессия захватывает только некоторый объем кривизны в нелинейной связи. Добавление членов более высоких степеней, таких как кубический квадратный полином, часто приводит к нежелательной "волнистости" в уравнении

регрессии. Альтернативный, и часто превосходящий, подход к моделированию нелинейных связей состоит в использовании *сплайнов*. Сплайны предоставляют способ гладко интерполировать между фиксированными точками. Сплайны первоначально использовались чертежниками для нанесения плавной кривой, в частности в судостроении и самолетостроении.

Сплайны создавались путем деформирования тонкого куска дерева при помощи гирь, которые назывались "утками" (рис. 4.11).



Рис. 4.11. Сплайны первоначально создавались на основе деформируемой древесины и "уток" и использовались в качестве инструмента чертежника для подгонки кривых.
Фото с разрешения Боба Перри (Bob Perry)

В техническом определении сплайн — это серия кусочно-непрерывных многочленов. Впервые они были разработаны во время Второй мировой войны на Абердинском испытательном полигоне в США румынским математиком Дж. Шонебергом (J. Schoenberg). Полиномиальные части гладко соединялись в серии фиксированных точек в предсказательной переменной, которые назывались *узлами*. Формулировка сплайнов намного сложнее, чем многочленная регрессия; подробности подгонки сплайнов обычно берут на себя статистические вычислительные системы. Пакет `splines` в R содержит функцию `bs` для создания *В-сплайнового* члена в регрессионной модели. Например, следующий фрагмент кода добавляет В-сплайновый член в регрессионную модель дома:

```
library(splines)
knots <- quantile(house_98105$SqFtTotLiving, p=c(.25, .5, .75))
lm_spline <- lm(AdjSalePrice ~ bs(SqFtTotLiving, knots=knots, degree=3) +
  SqFtLot + Bathrooms + Bedrooms + BldgGrade, data=house_98105)
```

Здесь необходимо определить два параметра: степень многочлена и местоположение узлов. В данном случае предсказатель `SqFtTotLiving` включается в модель с использованием кубического сплайна (`degree=3`). По умолчанию `bs` помещает узлы на

границах; в дополнение к этому, узлы также были помещены в нижний квартиль, медианный квартиль и верхний квартиль.

Формульный интерфейс пакета `statsmodels` поддерживает использование сплайнов таким же образом, как и в R. Здесь мы указываем *B*-сплайн, используя `df`, степени свободы. Это создаст `df - степень = 6 - 3 = 3` внутренних узлов с позициями, рассчитанными таким же образом, как и в приведенном выше коде R:

```
formula = 'AdjSalePrice ~ bs(SqFtTotLiving, df=6, degree=3) + ' +  
          'SqFtLot + Bathrooms + Bedrooms + BldgGrade'  
model_spline = smf.ols(formula=formula, data=house_98105)  
result_spline = model_spline.fit()
```

В отличие от линейного члена, для которого коэффициент имеет прямой смысл, коэффициенты для сплайнового члена не интерпретируемы. Вместо этого полезнее использовать визуальное изображение для выявления природы сплайновой подгонки. На рис. 4.12 представлен график частных остатков из регрессии. В отличие от многочленной модели, сплайновая модель намного ближе подходит к сглаженной кривой, демонстрируя более высокую гибкость сплайнов. В данном случае линия подогнана к данным намного точнее. Означает ли это, что сплайновая регрессия является более оптимальной моделью? Не обязательно. С экономической точки зрения нет никакого смысла в том, чтобы очень небольшие дома (менее 1000 кв. футов) имели более высокую стоимость, чем дома немного более крупного размера. Это, возможно, артефакт искажающей переменной (см. раздел "Искажающие переменные" ранее в этой главе).

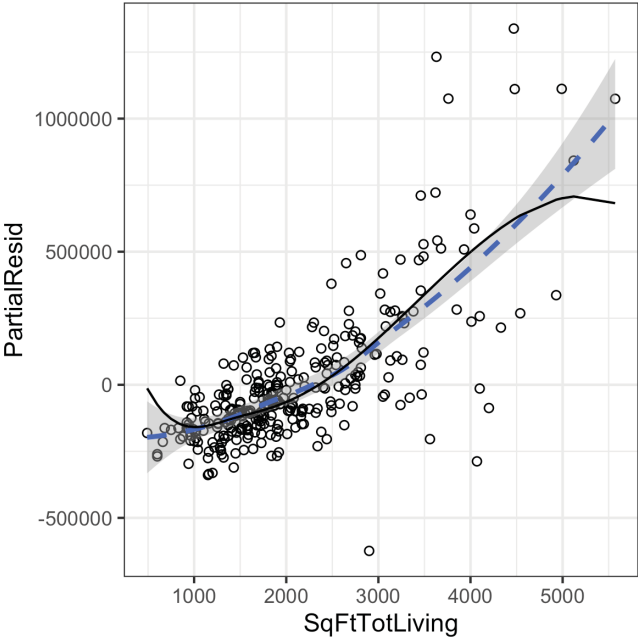


Рис. 4.12. Сплайновая регрессия, подогнанная для переменной `SqFtTotLiving` (сплошная линия) по сравнению со сглаженной кривой (пунктирная линия)

Обобщенные аддитивные модели

Предположим, что вы заподозрили нелинейную связь между откликом и предсказательной переменной в силу априорного знания либо вследствие обследования диагностических показателей регрессии. Полиномиальные члены могут быть недостаточно гибкими для улавливания связи, а сплайновые члены требуют указывать узлы. *Обобщенные аддитивные модели* (generalized additive models, GAM) — это гибкий прием моделирования, предназначенный для автоматической подгонки сплайновой регрессии. Пакет `gam` в R может использоваться для подгонки модели GAM к данным жилого фонда:

```
library(mgcv)
lm_gam <- gam(AdjSalePrice ~ s(SqFtTotLiving) + SqFtLot +
              Bathrooms + Bedrooms + BldgGrade,
              data=house_98105)
```

Член `s(SqFtTotLiving)` говорит функции `gam` найти "наилучшие" узлы для сплайнового члена (рис. 4.13).

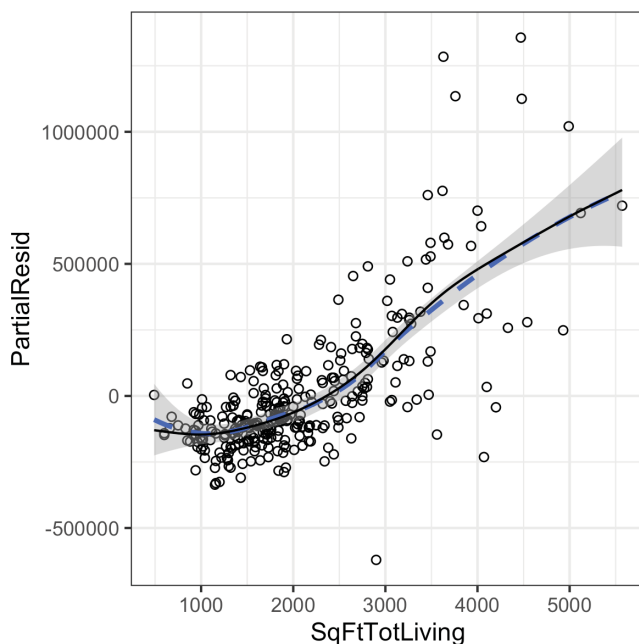


Рис. 4.13. Регрессия GAM, подогнанная для переменной `SqFtTotLiving` (сплошная линия) по сравнению со сглаженной кривой (пунктирная линия)

В Python мы можем использовать пакет `pyGAM`. Он предоставляет регрессионные и классификационные методы. Здесь мы используем класс `LinearGAM` для создания регрессионной модели:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']
outcome = 'AdjSalePrice'
```

```
X = house_98105[predictors].values
y = house_98105[outcome]
```

```
gam = LinearGAM(s(0, n_splines=12) + l(1) + l(2) + l(3) + l(4)) ❶
gam.gridsearch(X, y)
```

❶ Дефолтное значение для `n_splines` равно 20. Оно приводит к перепогонке для более крупных значений переменной `SqFtTotLiving`. Значение 12 приводит к более разумной подгонке.

Ключевые идеи для многочленной и сплайновой регрессии

- Выбросы в регрессии — это записи с крупным остатком.
- Мультиколлинеарность может вызывать числовую нестабильность в подгонке уравнения регрессии.
- Искажающая переменная — это важный предсказатель, который упущен из модели и может приводить к уравнению регрессии с мнимыми связями.
- Член уравнения, характеризующий взаимодействия между двумя переменными, необходим, если эффект одной переменной зависит от уровня или магнитуды другой.
- Многочленная регрессия может подгонять нелинейные связи между предсказателями и переменной исхода.
- Сплайны — это серия нанизанных вместе полиномиальных сегментов, соединяющихся в узлах.
- Мы можем автоматизировать процесс указания узлов в сплайнах, используя обобщенные аддитивные модели (GAM).

Дополнительные материалы для чтения

- ◆ Подробнее о сплайновых моделях и обобщенных аддитивных моделях см. в книге "Элементы статистического автоматического обучения" (Hastie T., Tibshirani R., Friedman J. Elements of Statistical Learning. — 2nd ed. — Springer, 2009) и в ее более кратком варианте на основе R "Введение в статистическое автоматическое обучение" (Gareth J., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning. — Springer, 2013).
- ◆ Дополнительные сведения об использовании регрессионных моделей для прогнозирования временных рядов см. в книге Галита Шмуэли и Кеннета Лихтендаля "Практическое прогнозирование временных рядов с помощью R" (Shmueli G., Lichtendahl K. Practical time series forecasting with R. — Axelrod Schnall, 2018).

Резюме

Возможно, никакой другой статистический метод не видел более широкого применения на протяжении многих лет, чем регрессия — процесс установления связи между многочисленными предсказательными переменными и переменной исхода. Ее фундаментальная форма является линейной: каждая предсказательная переменная имеет коэффициент, который описывает линейную связь между предсказателем и исходом. Более усовершенствованные формы регрессии, такие как многочленная и сплайновая регрессия, допускают нелинейность связи. В классической статистике главный упор делается на отыскании хорошей подгонки к наблюдаемым данным, чтобы объяснить или описать какое-либо явление, и сила этой подгонки зависит от того, как используются традиционные (*"внутривыборочные"*) метрики для определения результативности модели. В науке о данных, в отличие от этого, как правило, цель состоит в том, чтобы предсказывать значения для новых данных, поэтому используются метрики, основанные на предсказательной точности для вневыборочных данных. Кроме того, применяются методы отбора переменных с целью уменьшения размерности и создания более компактных моделей.

Классификация

Исследователи данных часто получают задачу автоматизирования процесса принятия решений для бизнеса. Является ли электронное письмо попыткой выудить чувствительную информацию? Перейдет ли клиент к другому поставщику? Нажмет ли интернет-пользователь на рекламном сообщении? Все эти вопросы представляют собой *классификационные задачи*, форму *контролируемого самообучения*, в котором мы сперва тренируем модель на данных, где исход известен, а затем применяем эту модель к данным, где исход неизвестен. Классификация — это, возможно, самая важная форма предсказания: ее цель состоит в том, чтобы предсказать, является ли запись нулем или единицей (выживание/не выживание, нажмет/не нажмет, перейдет/не перейдет) либо в некоторых случаях одной из нескольких категорий (например, фильтрация ваших входящих сообщений в Gmail на "основные", "соцсети", "промоакции" или "форумы").

Зачастую нам требуется больше, чем простая двоичная классификация: мы хотим знать предсказанную вероятность, что некий конкретный случай принадлежит классу. Вместо того чтобы иметь модель, которая просто относит классифицируемую запись к двоичной категории, большинство алгоритмов могут возвращать балл вероятности (склонности) принадлежать интересующему классу. В случае логистической регрессии результат, выводимый в R по умолчанию, по сути дела, находится в шкале логарифма перевесов, и его необходимо преобразовать в склонность. В пакете `scikit-learn` языка Python логистическая регрессия, как и большинство классификационных методов, предоставляет два метода предсказания: `predict` (который возвращает класс) и `predict_proba` (который возвращает вероятности для каждого класса). Затем используется скользящий порог отсеечения для конвертирования балла склонности в решение. Общий подход таков:

1. Установить пороговую вероятность для интересующего класса, выше которой мы рассматриваем запись как принадлежащую этому классу.
2. Оценить (любой моделью) вероятность того, что запись принадлежит интересующему классу.
3. Если эта вероятность выше пороговой вероятности, то назначить новую запись интересующему классу.

Чем выше порог отсеечения, тем меньше записей, предсказанных как 1, т. е. принадлежащих интересующему классу. Чем ниже порог отсеечения, тем больше записей, предсказанных как 1.

Эта глава охватывает несколько ключевых приемов классифицирования и оценивания склонностей; дополнительные методы, которые могут использоваться

как для классификации, так и для численного предсказания, описаны в следующей главе.

Более двух категорий?

Подавляющее большинство задач предусматривают двоичный отклик. Некоторые классификационные задачи, однако, требуют отклик более чем с двумя возможными исходами. Например, в годовщину действия клиентского договора подписки может быть три исхода: клиент покидает, или "переходит к другому поставщику" ($Y = 2$), переводится на помесечный договор ($Y = 1$) либо подписывает новый долгосрочный договор ($Y = 0$). Цель состоит в том, чтобы предсказать $Y = j$, где $j = 0, 1$ либо 2 . Большинство классификационных методов в данной главе можно применять либо непосредственно, либо со скромной адаптацией к откликам, которые имеют более двух исходов. Даже в случае более двух исходов задачу часто можно переработать в серию двоичных задач с использованием условных вероятностей. Например, чтобы предсказать исход договора, вы можете решить две двоичных задачи предсказания:

- предсказать, что $Y = 0$ либо $Y > 0$;
- если дано $Y > 0$, предсказать, что $Y = 1$ или $Y = 2$.

В этом случае имеет смысл разбить задачу на два подслучая: перейдет ли клиент к другому поставщику, и если он не перейдет, то какой договор он выберет. С точки зрения подгонки модели часто выгодно конвертировать многоклассовую задачу в серию двоичных задач. Это в особенности верно, когда одна категория распространена намного больше, чем другие категории.

Наивный Байес

Наивный байесов алгоритм использует вероятность наблюдать значения предсказательных переменных при наличии некоего исхода, чтобы оценить то, что понастоящему интересует: вероятность наблюдать исход $Y = i$ при наличии множества значений предсказателей¹.

Ключевые термины для наивного байеса

Условная вероятность (conditional probability)

Вероятность наблюдать какое-то событие (скажем, $X = i$) при условии, что имеется какое-то другое событие (скажем, $Y = i$) ; записывается, как $P(X_i | Y_i)$.

Апостериорная вероятность (posterior probability)

Вероятность исхода после того, как предсказательная информация была учтена (в отличие от априорной вероятности исходов, которая ее не учитывает).

¹ Этот и последующие разделы в настоящей главе закреплены за © 2020 Datastats, LLC, Питер Брюс, Эндрю Брюс и Питер Гедек; использовано с разрешения.

Для того чтобы понять байесову классификацию, мы можем начать с того, что вообразим полную или точную байесову классификацию. Для каждой классифицируемой записи:

1. Отыскать все другие записи с одинаковым предсказательным профилем (т. е. в которых значения предсказательных переменных одинаковы).
2. Определить, к каким классам эти записи принадлежат и какой класс является самым преобладающим (т. е. вероятным).
3. Назначить этот класс новой записи.

Приведенный выше подход сводится к отысканию всех записей в выборке, которые выглядят в точности как новая классифицируемая запись в том смысле, что все значения предсказательных переменных являются идентичными.



В стандартном наивном байесовом алгоритме предсказательные переменные должны быть категориальными (факторными) переменными. Два обходных пути для использования непрерывных переменных см. в разделе "Числовые предсказательные переменные" далее в этой главе.

Почему точная байесова классификация непрактична?

Когда число предсказательных переменных становится немалым, многие классифицируемые записи будут без точных совпадений. Это можно понять в контексте модели, которая предсказывает итоги голосования на основе демографических переменных. Даже внушительная выборка не будет содержать ни единого совпадения для новой записи с мужчиной латиноамериканцем с высоким доходом, из Среднего Запада США, который голосовал на последних выборах, не голосовал на предшествующих выборах, имеет трех дочерей и одного сына и разведен. И это всего восемь переменных, совсем небольшое число для большинства классификационных задач. Добавление одной-единственной новой переменной с пятью одинаково частыми категориями уменьшает вероятность совпадения в 5 раз.

Наивное решение

В наивном байесовом решении мы больше не ограничиваем вычисление вероятности теми записями, которые совпадают с классифицируемой записью. Вместо этого мы используем весь набор данных. Наивная байесова модификация такова:

1. Для двоичного отклика $Y = i$ ($i = 0$ либо 1) оценить индивидуальные условные вероятности по каждому предсказателю $P(X_j | Y = i)$; они представляют собой вероятности того, что значение предсказателя находится в записи, когда мы наблюдаем $Y = i$. Указанная вероятность оценивается долей значений X_j среди $Y = i$ записей в тренировочном наборе.
2. Перемножить эти вероятности друг с другом, а затем на долю записей, принадлежащих $Y = i$.

3. Повторить шаги 1 и 2 для всех классов.
4. Оценить вероятность для исхода i , взяв значение, вычисленное на шаге 2 для класса i и поделив его на сумму таких значений для всех классов.
5. Отнести запись к классу с самой высокой вероятностью для этого множества значений предсказателей.

Приведенный выше наивный байесов алгоритм можно также записать в форме уравнения для вероятности наблюдать исход $Y = i$ при наличии множества значений предсказателей X_1, \dots, X_p :

$$P(Y = i | X_1, X_2, \dots, X_p).$$

Вот полная формула для вычисления вероятностей классов с использованием точной байесовой классификации:

$$\begin{aligned} P(Y = i | X_1, X_2, \dots, X_p) &= \\ &= \frac{P(Y = i) P(X_1, \dots, X_p | Y = i)}{P(Y = 0) P(X_1, \dots, X_p | Y = 0) + P(Y = 1) P(X_1, \dots, X_p | Y = 1)}. \end{aligned}$$

В соответствии с наивным байесовым допущением об условной независимости это уравнение превращается в:

$$\begin{aligned} P(Y = i | X_1, X_2, \dots, X_p) &= \\ &= \frac{P(Y = i) P(X_1 | Y = i) \cdots P(X_p | Y = i)}{P(Y = 0) P(X_1 | Y = 0) \cdots P(X_p | Y = 0) + P(Y = 1) P(X_1 | Y = 1) \cdots P(X_p | Y = 1)}. \end{aligned}$$

Почему эта формула называется "наивной"? Мы приняли упрощающее допущение о том, что *точная условная вероятность* вектора значений предсказателей при условии, что наблюдается исход, достаточно хорошо оценивается произведением отдельных условных вероятностей $P(X_j | Y = i)$. Другими словами, оценивая $P(X_j | Y = i)$ вместо $P(X_1, X_2, \dots, X_p | Y = i)$, мы исходим из допущения о том, что X_j не зависит от всех других предсказательных переменных X_k для $k \neq j$.

Значение $P(X_1, X_2, \dots, X_p)$ — это поправочный коэффициент, который гарантирует, что вероятность находится между 0 и 1 и не зависит от Y :

Для оценивания наивной байесовой модели в R можно воспользоваться несколькими пакетами. Следующий ниже фрагмент кода выполняет подгонку модели с помощью пакета `klaR`:

```
library(klaR)
naive_model <- NaiveBayes(outcome ~ purpose_ + home_ + emp_len_,
                          data = na.omit(loan_data))
naive_model$stable
$purpose_
  var
```

```

grouping  credit_card debt_consolidation home_improvement major_purchase
paid off  0.18759649      0.55215915      0.07150104      0.05359270
default   0.15151515      0.57571347      0.05981209      0.03727229

      var
grouping  medical      other small_business
paid off  0.01424728  0.09990737      0.02099599
default   0.01433549  0.11561025      0.04574126

$home_
      var
grouping  MORTGAGE      OWN      RENT
paid off  0.4894800  0.0808963  0.4296237
default   0.4313440  0.0832782  0.4853778

$emp_len_
      var
grouping  < 1 Year  > 1 Year
paid off  0.03105289  0.96894711
default   0.04728508  0.95271492

```

На выходе из модели будут получены условные вероятности $P(X_j | Y = i)$.

В Python мы можем воспользоваться классом `sklearn.naive_bayes.MultinomialNB` из пакета `scikit-learn`. Прежде чем выполнить подгонку модели, нам нужно конвертировать категориальные признаки в фиктивные переменные:

```

predictors = ['purpose_', 'home_', 'emp_len_']
outcome = 'outcome'
X = pd.get_dummies(loan_data[predictors], prefix='', prefix_sep='')
y = loan_data[outcome]

naive_model = MultinomialNB(alpha=0.01, fit_prior=True)
naive_model.fit(X, y)

```

Условные вероятности можно получить из подогнанной модели, используя свойство `feature_log_prob_`.

Эта модель может быть использована для предсказания исхода нового кредита. Мы используем последнее значение набора данных для тестирования:

```

new_loan <- loan_data[147, c('purpose_', 'home_', 'emp_len_')]
row.names(new_loan) <- NULL
new_loan
      purpose_ home_ emp_len_
1 small_business MORTGAGE > 1 Year

```

В Python мы получаем это значение следующим образом:

```
new_loan = X.loc[146:146, :]
```

В данном случае модель предсказывает невыплату ссуды (R):

```
predict(naive_model, new_loan)
$class
[1] default
Levels: paid off default

$posterior
      paid off   default
[1,] 0.3717206 0.6282794
```

Как мы уже обсуждали, классификационные модели пакета `scikit-learn` имеют два метода — `predict`, который возвращает предсказанный класс, и `predict_proba`, который возвращает вероятности классов:

```
print('предсказанный класс: ', naive_model.predict(new_loan)[0])

probabilities = pd.DataFrame(naive_model.predict_proba(new_loan),
                             columns=loan_data[outcome].cat.categories)
print('предсказанные вероятности', probabilities)
--
предсказанный класс: default
предсказанные вероятности
      default   paid off
0  0.653696  0.346304
```

Предсказание также возвращает апостериорную оценку вероятности (`posterior`) невыплаты ссуды. Наивный байесов классификатор известен тем, что он производит *смещенные* оценки. Однако там, где целью является ранжирование записей согласно вероятности, что $Y=1$, несмещенные оценки вероятности не нужны, и наивный байесов классификатор производит хорошие результаты.

Числовые предсказательные переменные

Байесов классификатор работает только с категориальными предсказателями (например, с классификацией спама, где присутствие или отсутствие слов, фраз, символов и т. д. лежат в основе предсказательной задачи). Для применения наивного байесова классификатора к числовым предсказателям должен быть принят один из двух подходов:

- ♦ разложить на корзины и конвертировать числовые предсказатели в категориальные предсказатели и далее применить алгоритм из предыдущего раздела;
- ♦ использовать вероятностную модель — например, нормальное распределение (см. раздел *"Нормальное распределение"* главы 2) — для оценивания условной вероятности $P(X_j | Y = i)$.



Когда предсказательная категория в тренировочных данных отсутствует, алгоритм назначает переменной исхода в новых данных нулевую вероятность вместо того, чтобы просто проигнорировать эту переменную и использовать информацию от других переменных, как могло бы быть в других методах.

Большинство имплементаций наивного Байеса используют сглаживающий параметр (лапласово сглаживание), чтобы это предотвратить.

Ключевые идеи для наивного Байеса

- Наивный байесов алгоритм работает с категориальными (факторными) предсказателями и исходами.
- Он задается вопросом: какие предсказательные категории внутри каждой категории исходов являются наиболее вероятными?
- Эта информация далее инвертируется для оценивания вероятностей категорий исходов при условии, что имеются значения предсказателей.

Дополнительные материалы для чтения

- ♦ "Элементы статистического автоматического обучения" (Hastie T., Tibshirani R., Friedman J. Elements of Statistical Learning. — 2nd ed. — Springer, 2009).
- ♦ "Добыча регулярностей из данных для бизнес-аналитики" (Shmueli G., Bruce P., Patel N. Data mining for business analytics. — 3rd ed. — John Wiley & Sons, 2016) содержит полную главу по наивному Байесу с вариантами для R, Python, Excel и JMP.

Дискриминантный анализ

Дискриминантный анализ — это самый ранний статистический классификатор; он был представлен Р. А. Фишером в 1936 году в статье, опубликованной в журнале "Летопись евгеники" (Annals of Eugenics)².

Ключевые термины для дискриминантного анализа

Ковариация (covariance)

Мера степени, с которой переменная варьируется совместно с другой (т. е. с похожей магнитудой и направлением).

Дискриминантная функция (discriminant function)

Функция, которая при ее применении к предсказательным переменным максимизирует разделение классов.

Дискриминантные веса (discriminant weights)

Баллы, получаемые в результате применения дискриминантной функции, которые используются для исчисления вероятностей принадлежности к тому или иному классу.

² Безусловно, вызывает удивление, что первая статья о статистической классификации была опубликована в журнале, посвященном евгенике. Действительно, существует дезориентирующая связь между ранним развитием статистики и евгеникой. См. <https://oreil.ly/eUJvR>.

Хотя дискриминантный анализ охватывает несколько технических приемов, самое широкое распространение получил *линейный дискриминантный анализ* (linear discriminant analysis, LDA). Исходный метод, предложенный Фишером, на самом деле немного отличался от линейного дискриминантного анализа, но его механизм, по существу, остался прежним. Сегодня линейный дискриминантный анализ используется менее широко с появлением более изощренных технических приемов, таких как древесные модели и логистическая регрессия.

Однако линейный дискриминантный анализ можно все еще встретить в некоторых приложениях, и он имеет связи с другими более широко используемыми методами (таким, как анализ главных компонент; см. раздел "*Анализ главных компонент*" главы 7).



Линейный дискриминантный анализ не следует путать с латентным размещением Дирихле, имеющем такую же аббревиатуру (LDA, latent dirichlet allocation). Латентное размещение Дирихле используется в обработке текста и естественного языка и никоим образом не связано с линейным дискриминантным анализом.

Матрица ковариаций

Для понимания дискриминантного анализа сначала необходимо ввести понятие *ковариации* между двумя или несколькими переменными. Ковариация измеряет связь между двумя переменными — x и z . Обозначим среднее каждой переменной как \bar{x} и \bar{z} (см. раздел "*Среднее*" главы 1). Ковариация $s_{x,z}$ между x и z задается следующей формулой:

$$s_{x,z} = \frac{\sum_{i=1}^n (x_i - \bar{x})(z_i - \bar{z})}{n-1},$$

где n — число записей (обратите внимание, что мы делим не на n , а на $n-1$: см. врезку. "*Степени свободы и n или $n-1$?*" главы 1).

Как и с коэффициентом корреляции (см. раздел "*Корреляция*" главы 1), положительные значения говорят о положительной связи, отрицательные значения — об отрицательной. Корреляция, однако, ограничена значениями между -1 и 1 , тогда как ковариация находится на той же шкале измерения, что и переменные x и z . *Матрица ковариаций* Σ для x и z состоит из дисперсий индивидуальных переменных s_x^2 и s_z^2 на диагонали (где строка и столбец — это одинаковая переменная) и ковариаций между парами переменных вне диагоналей.

$$\hat{\Sigma} = \begin{bmatrix} s_x^2 & s_{x,z} \\ s_{x,z} & s_z^2 \end{bmatrix}.$$



Напомним, что стандартное отклонение применяется для нормализации переменной в z -оценку; матрица ковариаций используется в многомерном расширении этого процесса стандартизации. Оно известно как расстояние Махаланобиса (см. врезку "Другие метрики расстояния" главы 6) и связано с функцией линейного дискриминантного анализа.

Линейный дискриминант Фишера

Для простоты давайте сосредоточимся на классификационной задаче, в которой мы хотим предсказать двоичный исход y , используя всего две непрерывные числовые переменные (x , z). В техническом плане дискриминантный анализ исходит из допущения, что предсказательные переменные являются нормально распределенными непрерывными величинами, но на практике этот метод работает хорошо даже для непредельных отступлений от нормальности и для двоичных предсказателей. Линейный дискриминант Фишера различает вариацию *между* группами, с одной стороны, и вариацию *внутри* групп — с другой. В частности, стремясь разделить записи на две группы, линейный дискриминантный анализ (LDA) фокусируется на максимизации "между" суммой квадратов $SS_{\text{между}}$ (измеряя вариацию между двумя группами) относительно "внутригрупповой" суммы квадратов $SS_{\text{внутри}}$ (измеряющей внутригрупповую вариацию). В этом случае две группы соответствуют записям (x_0, z_0) , для которых $y = 0$, и записям (x_1, z_1) , для которых $y = 1$. Указанный метод отыскивает линейную комбинацию $w_x x + w_z z$, которая максимизирует отношение сумм квадратов.

$$\frac{SS_{\text{между}}}{SS_{\text{внутри}}}.$$

Межгрупповая сумма квадратов — это квадратичное расстояние между двумя групповыми средними, и внутригрупповая сумма квадратов — это разброс вокруг средних внутри каждой группы, взвешенный на матрицу ковариаций. В интуитивном плане путем максимизации межгрупповой суммы квадратов и минимизации внутригрупповой суммы квадратов этот метод дает самое большое разделение между двумя группами.

Простой пример

Пакет MASS, связанный с книгой У. Н. Венабелса и Б. Д. Рипли "Современная прикладная статистика с помощью S" (Venables W. N., Ripley B. D. Modern Applied Statistics with S. — Springer, 1994), предлагает функцию для линейного дискриминантного анализа с помощью R. Следующий фрагмент кода применяет эту функцию к выборке данных о ссудах, используя две предсказательные переменные: балл заемщика `borrower_score` и отношение платежа к доходу `payment_inc_ratio`, — и печатает оценочные веса линейного дискриминатора.

```
library(MASS)
loan_lda <- lda(outcome ~ borrower_score + payment_inc_ratio,
               data=loan3000)
```

```
loan_lda$scaling
LD1
borrower_score    7.17583880
payment_inc_ratio -0.09967559
```

В Python мы можем использовать класс `LinearDiscriminantAnalysis` из модуля `sklearn.discriminant_analysis`. Свойство `scalings_` дает оценочные веса:

```
loan3000.outcome = loan3000.outcome.astype('category')
```

```
predictors = ['borrower_score', 'payment_inc_ratio']
outcome = 'outcome'
```

```
X = loan3000[predictors]
y = loan3000[outcome]
```

```
loan_lda = LinearDiscriminantAnalysis()
loan_lda.fit(X, y)
pd.DataFrame(loan_lda.scalings_, index=X.columns)
```



Использование дискриминантного анализа для отбора признаков

Если предсказательные переменные нормализуются до выполнения линейного дискриминантного анализа, веса дискриминатора являются мерами важности переменной, таким образом обеспечивая вычислительно эффективный метод отбора признаков.

Функция `lda` может предсказать вероятность "невыплаты" (default) относительно "погашено" (paid off):

```
pred <- predict(loan_lda)
head(pred$posterior)
  paid off  default
1 0.4464563 0.5535437
2 0.4410466 0.5589534
3 0.7273038 0.2726962
4 0.4937462 0.5062538
5 0.3900475 0.6099525
6 0.5892594 0.4107406
```

Метод `predict_proba` подогнанной модели возвращает вероятности для исходов "невыплата" и "погашено":

```
pred = pd.DataFrame(loan_lda.predict_proba(loan3000[predictors]),
                    columns=loan_lda.classes_)
pred.head()
```

График предсказаний помогает проиллюстрировать то, как работает линейный дискриминантный анализ. Используя результат функции `predict`, график оценочной вероятности невыплаты ссуды строится следующим образом:

```

center <- 0.5 * (loan_lda$mean[1, ] + loan_lda$mean[2, ])
slope <- -loan_lda$scaling[1] / loan_lda$scaling[2]
intercept <- center[2] - center[1] * slope

ggplot(data=lda_df, aes(x=borrower_score, y=payment_inc_ratio,
                        color=prob_default)) +
  geom_point(alpha=.6) +
  scale_color_gradientn(colors=c('#ca0020', '#f7f7f7', '#0571b0')) +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0), lim=c(0, 20)) +
  geom_abline(slope=slope, intercept=intercept, color='darkgreen')

```

Похожий график создается в Python с помощью вот этого кода:

```

# Использовать scalings и центр средних для определения границы решения
center = np.mean(loan_lda.means_, axis=0)
slope = - loan_lda.scalings_[0] / loan_lda.scalings_[1]
intercept = center[1] - center[0] * slope

# Отношение payment_inc_ratio для балла borrower_score, равного 0 и 20
x_0 = (0 - intercept) / slope
x_20 = (20 - intercept) / slope

lda_df = pd.concat([loan3000, pred['default']], axis=1)
lda_df.head()

fig, ax = plt.subplots(figsize=(4, 4))
g = sns.scatterplot(x='borrower_score', y='payment_inc_ratio',
                   hue='default', data=lda_df,
                   palette=sns.diverging_palette(240, 10, n=9, as_cmap=True),
                   ax=ax, legend=False)

ax.set_ylim(0, 20)
ax.set_xlim(0.15, 0.8)
ax.plot((x_0, x_20), (0, 20), linewidth=3)
ax.plot(*loan_lda.means_.transpose())

```

Результирующий график показан на рис. 5.1. Точки данных слева от диагональной прямой предсказываются как невыплата (вероятность больше 0,5).

Используя веса дискриминантной функции, линейный дискриминантный анализ разбивает предсказательное пространство на два участка, как показано жирной прямой. Предсказания, которые расположены от линии намного дальше в обоих направлениях, имеют более высокий уровень доверия (т. е. вероятность намного дальше от 0,5).



Расширения дискриминантного анализа

Больше предсказательных переменных: хотя в тексте и примере этого раздела использовалось всего две предсказательные переменные, линейный дискриминантный анализ работает точно также с более чем двумя предсказательными переменными. Единственным ограничивающим фактором является число

записей (оценивание матрицы ковариаций требует достаточного числа записей в расчете на переменную, что в типичной ситуации не представляет проблему в приложениях науки о данных).

Существуют другие варианты дискриминантного анализа. Самым известным является квадратичный дискриминантный анализ (quadratic discriminant analysis, QDA). Несмотря на его название, он все же является линейной дискриминантной функцией. Главное, разница состоит в том, что в линейном дискриминантном анализе матрица ковариаций принимается одинаковой для двух групп, соответствующих $Y=0$ и $Y=1$. В квадратичном дискриминантном анализе допускается, что матрица ковариаций может быть разной для двух групп. На практике эта разница в большинстве применений не является критически важной.

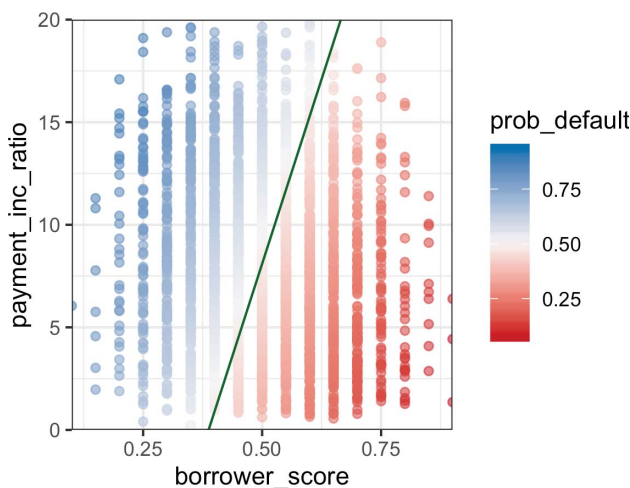


Рис. 5.1. Предсказание линейным дискриминантным анализом невыплаты ссуды с использованием двух переменных: балла кредитоспособности заемщика и отношения платежей к доходу

Ключевые идеи для дискриминантного анализа

- Дискриминантный анализ работает с непрерывными или категориальными предсказателями, а также с категориальными исходами.
- При использовании матрицы ковариаций он вычисляет *линейную дискриминантную функцию*, которая используется для различения записей, принадлежащих одному классу, от тех, которые принадлежат другому.
- Указанная функция применяется к записям для получения весов, или баллов, для каждой записи (один вес для каждого возможного класса), которые определяют его оценочный класс.

Дополнительные материалы для чтения

- ♦ Книга "Элементы статистического автоматического обучения" (Hastie T., Tibshirani R., Friedman J. Elements of Statistical Learning. — 2nd ed. — Springer, 2009) и ее более краткий вариант "Введение в статистическое автоматическое обучение" (Gareth J., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning. — Springer, 2013) содержат раздел по дискриминантному анализу.
- ♦ Книга "Добыча регулярностей из данных для бизнес-аналитики" (Shmueli G., Bruce P., Patel N. Data mining for business analytics. — 3rd ed. — John Wiley & Sons, 2016) с вариантами для R, Python, Excel и JMP содержит полную главу по дискриминантному анализу.
- ♦ В случае интереса в историческом плане оригинальную статью Фишера по указанной теме "Использование многократных замеров в таксономических задачах" (The Use of Multiple Measurements in Taxonomic Problems), опубликованную в 1936 году в журнале "Летопись евгеники" (Annals of Genetics) (теперь носящем название "Летопись генетики"), можно найти онлайн³.

Логистическая регрессия

Логистическая регрессия аналогична множественной линейной регрессии за одним исключением (см. главу 4) — исход является двоичным. При этом используются различные преобразования для того, чтобы конвертировать задачу в ту, в которой может быть подоғнана линейная модель. Как и дискриминантный анализ, и в отличие от k ближайших соседей и наивного Байеса, логистическая регрессия является подходом на основе структурированной модели нежели подходом, центрированным на данных. Указанный метод стал популярным благодаря своему высокому вычислительному быстродействию и результатам модели, которые допускают ускоренное задание баллов новым данным.

Ключевые термины для логистической регрессии

Логит (logit)

Функция, которая отображает вероятность принадлежности классу в интервал $\pm\infty$ (вместо интервала от 0 до 1).

Синонимы: логит-преобразование, логарифм перевесов (см. ниже).

Перевесы (odds)

Отношение "успеха" (1) к "неуспеху" (0).

Синонимы: шансы, фора, гандикап, преимущество.

³ См. https://oreil.ly/_TCR8.

Логарифм перевесов (log odds)

Отклик в преобразованной модели (теперь линейный), который отображается назад в вероятность.

Синоним: логарифмированные перевесы.

Функция логистического отклика и логит

Ключевыми компонентами логистической регрессии являются *функция логистического отклика* и логит, в которых мы отображаем вероятность (находящуюся на шкале 0–1) в более расширенную шкалу, подходящую для линейного моделирования.

Первый шаг состоит в том, чтобы думать о переменной исхода не как о двоичной метке, а как о вероятности p того, что метка равна 1. Может возникнуть соблазн наивно смоделировать p как линейную функцию предсказательных переменных:

$$p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_q x_q.$$

Однако подгонка этой модели не обеспечивает того, чтобы p в итоге оказалась между 0 и 1, где должна быть вероятность.

Вместо этого мы моделируем p путем применения *логистического отклика* или *обратного логита* к предсказателям:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_q x_q)}}.$$

Указанное преобразование обеспечивает, чтобы p оставалась между 0 и 1.

Для того чтобы извлечь экспоненциальное выражение из знаменателя, мы рассматриваем *перевесы* вместо вероятностей. Перевесы, знакомые всем игрокам, делающим ставки, являются отношением "успехов" (1) к "неуспехам" (0). С точки зрения вероятностей перевесы — это вероятность события, деленная на вероятность того, что событие не произойдет. Например, если вероятность, что лошадь выиграет скачки, равна 0,5, то вероятность, что "не выиграет", составит $1 - 0,5 = 0,5$, и шансы равны 1,0.

$$\text{Шансы } (Y = 1) = \frac{p}{1 - p}.$$

Мы можем получить вероятность из шансов, используя обратную функцию перевесов:

$$p = \frac{\text{Шансы}}{1 + \text{Шансы}}.$$

Мы комбинируем эту формулу с функцией логистического отклика, показанной ранее, и получаем:

$$\text{Шансы } (Y = 1) = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_q x_q}.$$

Наконец, взяв логарифм выражений, стоящих справа и слева от знака равенства, мы получаем выражение, которое включает линейную функцию предсказателей:

$$\log(\text{Шансы } (Y = 1)) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_q x_q.$$

Функция *логарифма шансов*, также именуемая логит-функцией, отображает вероятность p из интервала $(0; 1)$ в любое значение из интервала $(-\infty; +\infty)$, рис. 5.2. Цикл преобразования завершен; мы использовали линейную модель для предсказания вероятности, которую, в свою очередь, можем отобразить на метку класса путем применения правила отсечения, — любая запись с вероятностью, превышающей порог отсечения, классифицируется как 1.

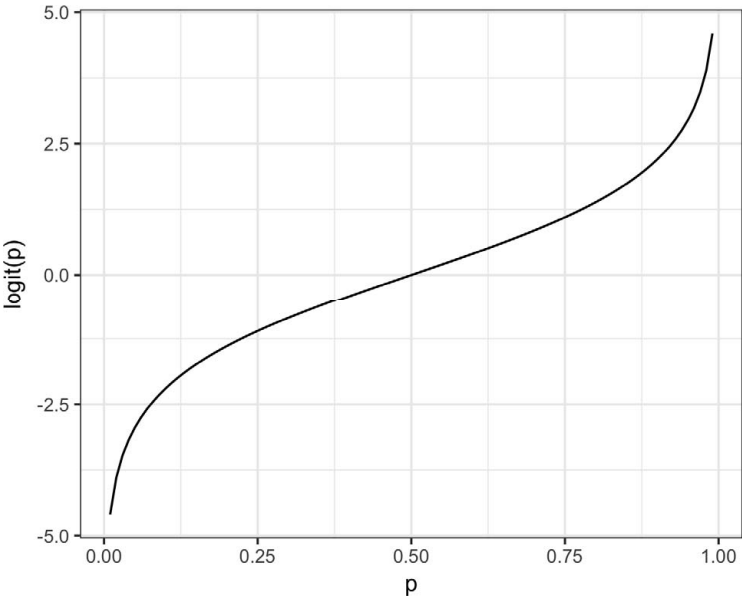


Рис. 5.2. График логит-функции, которая отображает вероятность в шкалу, подходящую для линейной модели

Логистическая регрессия и ОЛМ

Отклик в формуле логистической регрессии — это логарифм перевесов двоичного исхода 1. Мы наблюдаем только двоичный исход, а не логарифм перевесов, и поэтому специальные статистические методы требуются для подгонки уравнения. Логистическая регрессия является особым экземпляром *обобщенной линейной модели* (ОЛМ — GLM, generalized linear model), разработанной для расширения линейной регрессии до других формулировок.

В R для подгонки логистической регрессии используется функция `glm` с аргументом `family`, установленным равным `binomial`. Следующий фрагмент кода выполняет подгонку логистической регрессии к данным о персональных ссудах, представленным в разделе "к ближайших соседей" главы 6.

```
logistic_model <- glm(outcome ~ payment_inc_ratio + purpose_ +
                      home_ + emp_len_ + borrower_score,
                      data=loan_data, family='binomial')

logistic_model

Call: glm(formula = outcome ~ payment_inc_ratio + purpose_ + home_ +
          emp_len_ + borrower_score, family = "binomial", data = loan_data)
```

Coefficients:

(Intercept)	payment_inc_ratio
1.63809	0.07974
purpose_debt_consolidation	purpose_home_improvement
0.24937	0.40774
purpose_major_purchase	purpose_medical
0.22963	0.51048
purpose_other	purpose_small_business
0.62066	1.21526
home_OWN	home_RENT
0.04833	0.15732
emp_len_ > 1 Year	borrower_score
-0.35673	-4.61264

Degrees of Freedom: 45341 Total (i.e. Null); 45330 Residual

Null Deviance: 62860

Residual Deviance: 57510 AIC: 57540

Откликом является `outcome`, который принимает значение 0, если ссуда погашена, и 1, если ссуда не возвращена. Факторные переменные `purpose_` и `home_` представляют предназначение ссуды и статус домовладельца. Как и в линейной регрессии, факторная переменная с P уровнями представлена $P - 1$ столбцами. В R по умолчанию используется *опорное* кодирование и все уровни сравниваются с опорным уровнем (см. раздел "Факторные переменные в регрессии" главы 4). Опорные уровни для этих факторов — это соответственно `credit_card` и `MORTGAGE`. Переменная `borrower_score` — это балл от 0 до 1, который обозначает кредитоспособность заемщика (от плохой до превосходной). Указанная переменная была создана из нескольких других переменных при помощи k ближайших соседей (см. раздел "*k* ближайших соседей как механизм порождения признаков" главы 6).

В Python мы используем класс `LogisticRegression` из модуля `sklearn.linear_model` пакета `scikit-learn`. Аргументы `penalty` и `C` применяются для предотвращения перепогонки регуляризацией L1 или L2. Регуляризация включена по умолчанию. Для того чтобы выполнить перепогонку без регуляризации, мы устанавливаем `C` равным очень большому значению. Аргумент `solver` выбирает используемый минимизатор; по умолчанию используется метод `liblinear`:

```
predictors = ['payment_inc_ratio', 'purpose_', 'home_', 'emp_len_', 'borrower_score']
outcome = 'outcome'
```



```
X = pd.get_dummies(loan_data[predictors], prefix='', prefix_sep='', drop_first=True)
y = loan_data[outcome]

logit_reg = LogisticRegression(penalty='l2', C=1e42, solver='liblinear')
logit_reg.fit(X, y)
```

В отличие от R, пакет `scikit-learn` в Python выводит классы из уникальных значений в `y` (*погашено* и *невыплата*). Внутри классы расположены в алфавитном порядке. Поскольку этот порядок обратен порядку коэффициентов, используемых в R, вы увидите, что коэффициенты инвертированы. Метод `predict` возвращает метку класса, а метод `predict_proba` возвращает вероятности в порядке, заданном в атрибуте `logit_reg.classes_`.

Обобщенные линейные модели

Обобщенные линейные модели (ОЛМ, GLM) характеризуются двумя главными компонентами:

- ♦ вероятностным распределением или семейством (биномиальным в случае логистической регрессии);
- ♦ функцией связи, т. е. функцией преобразования, которая отображает отклик в предсказатели (логит в случае логистической регрессии).

Логистическая регрессия, безусловно, является наиболее часто встречающейся формой ОЛМ. Исследователь данных будет сталкиваться и с другими ОЛМ. Иногда вместо логита используется логарифм функции связи; на практике применение логарифма функции связи вряд ли приведет к сильно отличающимся результатам для большинства приложений. Распределение Пуассона чаще всего выбирается для моделирования количественных данных (например, число раз, когда пользователь посещает веб-страницу на определенное количество времени). Другие семейства включают отрицательное биномиальное и гамма-распределение, часто используемые для моделирования истекшего времени (например, времени безотказной работы). В отличие от логистической регрессии, применение ОЛМ с этими моделями обставлено большим количеством нюансов и сопряжено с большими мерами предосторожности. Их лучше всего избегать, только если вы с ними не знакомы и не понимаете полезность и подводные камни этих методов.

Предсказанные значения из логистической регрессии

Предсказанное значение, полученное из логистической регрессии, рассматривается с точки зрения логарифма перевесов: $\hat{Y} = \log(\text{Шансы } (Y = 1))$. Предсказанная вероятность задается функцией логистического отклика:

$$\hat{p} = \frac{1}{1 + e^{-\hat{Y}}}.$$

Например, посмотрим на предсказания из модели `logistic_model` в R:

```
pred <- predict(logistic_model)
summary(pred)
   Min.    1st Qu.    Median      Mean   3rd Qu.      Max.
-2.704774 -0.518825 -0.008539  0.002564  0.505061  3.509606
```

В Python мы можем конвертировать вероятности в кадр данных и использовать метод `describe`, чтобы получить эти характеристики распределения:

```
pred = pd.DataFrame(logit_reg.predict_log_proba(X),
                    columns=loan_data[outcome].cat.categories)

pred.describe()
```

Конвертирование этих значений в вероятности представляет собой простую операцию преобразования:

```
prob <- 1/(1 + exp(-pred))
> summary(prob)
   Min. 1st Qu.  Median      Mean 3rd Qu.      Max.
0.06269 0.37313 0.49787 0.50000 0.62365 0.97096
```

Эти вероятности непосредственно доступны с помощью методов `predict_proba` в пакете `scikit-learn`:

```
pred = pd.DataFrame(logit_reg.predict_proba(X),
                    columns=loan_data[outcome].cat.categories)

pred.describe()
```

Они находятся на шкале от 0 до 1 и еще не объявляют о том, имеет ли предсказание значение "невыплата" или "погашено". Мы могли бы объявить любое значение свыше 0,5 как невыплата. На практике, если цель состоит в том, чтобы идентифицировать членов редкого класса, часто обоснован более низкий порог отсечения (см. раздел "Проблема редкого класса" далее в этой главе).

Интерпретирование коэффициентов и отношений перевесов

Одно из преимуществ логистической регрессии состоит в том, что она порождает модель, которую ускоренно можно применять для задания баллов новым данным без повторного вычисления. Еще одно преимущество — относительная простота интерпретации модели по сравнению с другими классификационными методами. Ключевая концептуальная идея заключается в понимании *отношения перевесов*, или *шансов*. Отношение перевесов легче всего понять для двоичной факторной переменной X :

$$\text{отношение перевесов} = \frac{\text{Шансы}(Y=1 | X=1)}{\text{Шансы}(Y=1 | X=0)}.$$

Указанная формула интерпретируется как перевесы того, что $Y=1$, когда $X=1$, против перевесов того, что $Y=1$, когда $X=0$. Если отношение перевесов равно 2, то перевесы того, что $Y=1$, будут в два раза выше, когда $X=1$, чем когда $X=0$.

Зачем возиться с отношением перевесов вместо вероятностей? Мы работаем с перевесами, потому что коэффициент β_j в логистической регрессии является логарифмом отношения перевесов для X_j .

Пример поможет всё понять. Для модели, подогнанной в разделе "Логистическая регрессия и ОЛМ" ранее в этой главе, коэффициент регрессии для `purpose_small_business` равен 1,21526. Он означает, что ссуда малому бизнесу в сравнении с ссудой для погашения задолженности по кредитной карте сокращает перевесы невыплаты против перевесов погашения на $\exp(1,21226) \approx 3,4$. Безусловно, кредиты в целях создания или расширения малого бизнеса значительно рискованнее других типов кредитов.

На рис. 5.3 показана связь между отношением перевесов и отношением логарифма перевесов для отношений перевесов больше 1. Поскольку коэффициенты находятся на логарифмической шкале, увеличение на 1 в коэффициентах приводит к увеличению на $\exp(1) \approx 2,72$ в отношении перевесов.

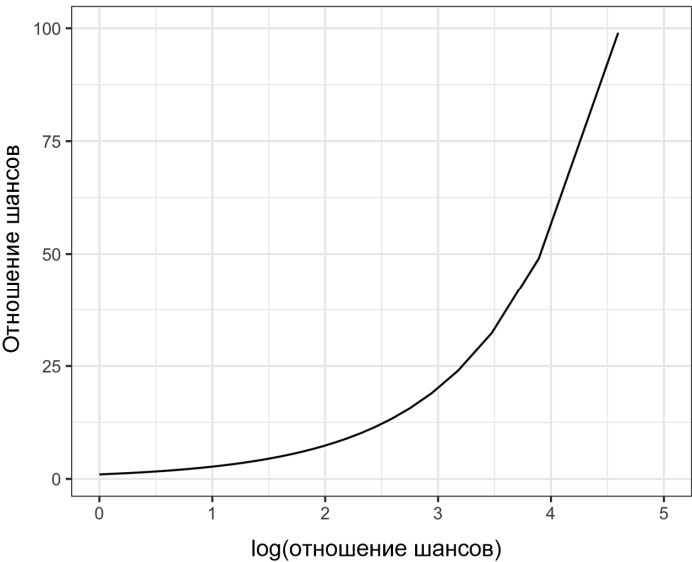


Рис. 5.3. Связь между отношением перевесов и логарифмом отношения перевесов

Отношения перевесов для числовых переменных X можно проинтерпретировать схожим образом: они измеряют изменение в отношении перевесов для единичного изменения в X . Например, эффект увеличения отношения платежей к доходам, скажем, с 5 до 6 увеличивает перевес невыплаты ссуды в $\exp(0,08244) \approx 1,09$ раз. Переменная `borrower_score` — это балл кредитоспособности заемщиков, который колеблется в диапазоне от 0 (низкая) до 1 (высокая). Перевес лучших заемщиков относительно худших, которые не возвращают ссуды, меньше в $\exp(-4,63890) \approx 0,01$ раз. Другими словами, риск невыплаты от заемщиков с самой слабой кредитоспособностью в 100 раз больше, чем риск невыплаты у самых лучших заемщиков!

Линейная и логистическая регрессия: сходства и различия

Линейная регрессия и логистическая регрессия имеют много схожих черт. Обе принимают параметрическую линейную форму, увязывающую предсказателей с откликом. Разведывание и отыскание лучшей модели выполняется очень похоже. Расширения линейной модели, такие как использование операции сплайнового преобразования предсказателя (см. раздел "Сплайны" главы 4), одинаковым образом применимы в логистической регрессионной формулировке. Однако логистическая регрессия отличается двумя фундаментальными составляющими:

- ♦ характером выполнения подгонки модели (наименьшие квадраты не применимы);
- ♦ природой и анализом остатков от модели.

Подгонка модели

Подгонка линейной регрессии выполняется с использованием наименьших квадратов, и качество подгонки оценивается с использованием статистик RMSE и R-квадрат. В логистической регрессии (в отличие от линейной регрессии) решение в замкнутой форме отсутствует, и подгонка модели должна выполняться с использованием *оценивания максимального правдоподобия* (maximum likelihood estimation, MLE). Оценивание максимального правдоподобия — это процедура, пытающаяся отыскать модель, вероятнее всего породившую данные, которые мы видим. В уравнении логистической регрессии откликом является не 0 или 1, а оценка логарифма перевесов о том, что отклик равняется 1. Оценивание максимального правдоподобия отыскивает решение такое, что оценочный логарифм перевесов наилучшим образом описывает наблюдаемый исход. Механизм указанного алгоритма предусматривает квазиньютоновскую оптимизацию, которая итеративно выполняется между шагом количественного оценивания (*оценивания в баллах по Фишеру*), основываясь на текущих параметрах, и обновлением параметров для улучшения подгонки.

Оценивание максимального правдоподобия

Вот еще немного подробностей, если вам нравятся статистические символы: начнем с набора данных (X_1, X_2, \dots, X_n) и вероятностной модели $P_\theta(X_1, X_2, \dots, X_n)$, которая зависит от набора параметров θ . Цель оценивания максимального правдоподобия состоит в том, чтобы отыскать набор параметров $\hat{\theta}$, который максимизирует значение $P_\theta(X_1, X_2, \dots, X_n)$, т. е. максимизирует вероятность наблюдать (X_1, X_2, \dots, X_n) при наличии модели P . В процессе подгонки модель оценивается при помощи метрики, которая называется *девиантностью*:

$$\text{погрешность} = -2 \log(P_{\hat{\theta}}(X_1, X_2, \dots, X_n)).$$

Более низкая девиантность соответствует более хорошей подгонке.

К счастью, большинству практиков не придется заниматься деталями алгоритма подгонки, поскольку они обрабатываются программно-информационным обеспечением. Большинству исследователей данных не придется беспокоиться по поводу метода подгонки, кроме понимания того, что он представляет собой способ отыскать хорошую модель при наличии некоторых допущений.



Работа с факторными переменными

В логистической регрессии факторные переменные должны кодироваться как в линейной регрессии (см. раздел "Факторные переменные в регрессии" главы 4). В R и других вычислительных системах это обычно улаживается автоматически, и при этом в основном используется опорное кодирование. Все другие классификационные методы, охваченные в этой главе, как правило, используют представление в виде кодировщика с одним активным состоянием (см. раздел "Кодировщик с одним активным состоянием" главы 6. В пакете `scikit-learn` языка Python проще всего использовать кодирование с одним активным состоянием, а это значит, что в регрессии можно использовать только $n - 1$ из результирующих фиктивных переменных).

Оценивание результативности модели

Как и другие классификационные методы, результативность логистической регрессии оценивается тем, насколько точно модель классифицирует новые данные (см. раздел "Оценивание классификационных моделей" далее в этой главе). Как и в случае с линейной регрессией, тут имеется несколько дополнительных стандартных статистических инструментов, которые позволяют проэкзаменовать и усовершенствовать модель. Вместе с оценочными коэффициентами R сообщает о стандартной ошибке коэффициентов (SE), z -оценке и p -значении:

```
summary(logistic_model)

Call:
glm(formula = outcome ~ payment_inc_ratio + purpose_ + home_ +
    emp_len_ + borrower_score, family = "binomial", data = loan_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.51951  -1.06908  -0.05853   1.07421   2.15528

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      1.638092   0.073708  22.224  < 2e-16 ***
payment_inc_ratio  0.079737   0.002487  32.058  < 2e-16 ***
purpose_debt_consolidation 0.249373   0.027615   9.030  < 2e-16 ***
purpose_home_improvement  0.407743   0.046615   8.747  < 2e-16 ***
purpose_major_purchase  0.229628   0.053683   4.277 1.89e-05 ***
purpose_medical    0.510479   0.086780   5.882 4.04e-09 ***
```

```

purpose_other          0.620663    0.039436   15.738 < 2e-16 ***
purpose_small_business 1.215261    0.063320   19.192 < 2e-16 ***
home_OWEN              0.048330    0.038036    1.271    0.204
home_RENT              0.157320    0.021203    7.420 1.17e-13 ***
emp_len_ > 1 Year       -0.356731    0.052622   -6.779 1.21e-11 ***
borrower_score         -4.612638    0.083558  -55.203 < 2e-16 ***

```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

```

Null deviance: 62857 on 45341 degrees of freedom
Residual deviance: 57515 on 45330 degrees of freedom
AIC: 57539

```

Number of Fisher Scoring iterations: 4

Пакет `statsmodels` имеет имплементацию для обобщенной линейной модели (GLM), которая дает столь же подробную информацию:

```

y_numbers = [1 if yi == 'default' else 0 for yi in y]
logit_reg_sm = sm.GLM(y_numbers, X.assign(const=1),
                      family=sm.families.Binomial())
logit_result = logit_reg_sm.fit()
logit_result.summary()

```

Интерпретация p -значения сопровождается теми же оговорками, что и в регрессии, и должна рассматриваться больше как относительный индикатор важности переменной (см. раздел "*Оценивание результативности модели*" главы 4), чем как формальная мера статистической значимости. С логистической регрессионной моделью, которая имеет двоичный отклик, не связан ни показатель RMSE, ни R-квадрат. Вместо этого логистическая регрессионная модель в типичной ситуации оценивается при помощи более общих метрик, используемых для классификации (см. раздел "*Оценивание классификационных моделей*" далее в этой главе).

Многие другие понятия относительно линейной регрессии переносятся на логистическую регрессионную формулировку (и других ОЛМ). Например, вы можете применить пошаговую регрессию, выполнить подгонку членов взаимодействия или включить сплайновые члены. То же самое касается применения к логистической регрессии искажающих и коррелированных переменных (см. раздел "*Интерпретирование уравнения регрессии*" главы 4). Вы можете выполнить подгонку обобщенных аддитивных моделей (см. раздел "*Обобщенные аддитивные модели*" главы 4) с использованием пакета `mgcv` в R:

```

logistic_gam <- gam(outcome ~ s(payment_inc_ratio) + purpose_ +
                    home_ + emp_len_ + s(borrower_score),
                    data=loan_data, family='binomial')

```

Формульный интерфейс пакета `statsmodels` в Python также поддерживает эти расширения:

```
import statsmodels.formula.api as smf
formula = ('outcome ~ bs(payment_inc_ratio, df=4) + purpose_ + ' +
          'home_ + emp_len_ + bs(borrower_score, df=4)')
model = smf.glm(formula=formula, data=loan_data, family=sm.families.Binomial())
results = model.fit()
```

Анализ остатков

Одна область, где логистическая регрессия отличается от линейной регрессии, касается анализа остатков. Как и в линейной регрессии (см. рис. 4.9), вычисление частных остатков в R выполняется прямолинейно:

```
terms <- predict(logistic_gam, type='terms')
partial_resid <- resid(logistic_model) + terms
df <- data.frame(payment_inc_ratio = loan_data[, 'payment_inc_ratio'],
                 terms = terms[, 's(payment_inc_ratio)'],
                 partial_resid = partial_resid[, 's(payment_inc_ratio)'])
ggplot(df, aes(x=payment_inc_ratio, y=partial_resid, solid = FALSE)) +
  geom_point(shape=46, alpha=.4) +
  geom_line(aes(x=payment_inc_ratio, y=terms), color='red', alpha=.5, size=1.5)
+ labs(y='Partial Residual')
```

Результирующий график показан на рис. 5.4. Оценочная подгонка, показанная линией, проходит между двумя наборами точечных облаков. Верхнее облако соответствует отклику 1 (невыплаченные ссуды) и нижнее облако — отклику 0 (погашенные ссуды). Такой вид очень типичен для остатков от логистической регрессии, поскольку выходные данные являются двоичными. Предсказание измеряется как

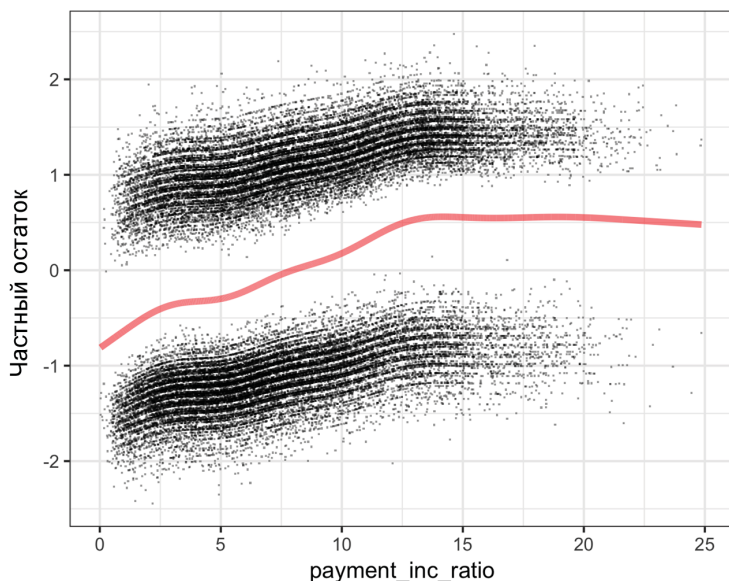


Рис. 5.4. Частные остатки от логистической регрессии

логит (логарифм отношения перевесов), который всегда будет иметь некоторое конечное значение. Фактическое значение, абсолютное 0 или 1, соответствует бесконечному логиту, положительному или отрицательному, поэтому остатки (которые добавляются к подогнанному значению) никогда не будут равны 0. Следовательно, нанесенные точки лежат в облаках выше либо ниже подогнанной прямой на графике частных остатков. Частные остатки в логистической регрессии, хотя и менее ценны, чем в регрессии, все же полезны для подтверждения нелинейного поведения и выявления весьма влиятельных записей.



Часть результата из функции `summary` можно эффективно проигнорировать. Параметр дисперсии не применяется к логистической регрессии и находится там для других типов ОЛМ. Остаточная погрешность и число итераций количественного оценивания связаны с оцениванием максимального правдоподобия (см. врезку "Оценивание максимального правдоподобия" ранее в этой главе).

Ключевые идеи для логистической регрессии

- Логистическая регрессия похожа на линейную регрессию, за исключением того, что откликом является двоичная переменная.
- Для приведения модели к форме, пригодной в качестве линейной модели, требуется несколько преобразований, при этом логарифм отношения перевесов выступает в качестве переменной отклика.
- После подгонки линейной модели (путем итеративного процесса) логарифм перевесов отображается назад в вероятность.
- Логистическая регрессия является популярной по причине своего вычислительного быстродействия, а также потому, что она порождает модель, которую можно применять для задания баллов новым данным с помощью малого числа арифметических операций.

Дополнительные материалы для чтения

- ◆ Стандартный справочник по логистической регрессии "Прикладная логистическая регрессия" (Hosmer D., Lemeshow S., Sturdivan R. Applied Logistic Regression. — 3rd ed. — Wiley, 2013).
- ◆ Две популярные книги Джозефа Хильбе (Joseph Hilbe) "Модели логистической регрессии" (Logistic Regression Models) в углубленном изложении и "Практическое руководство по логистической регрессии" (Practical Guide to Logistic Regression) в компактном изложении, обе опубликованы в издательстве CRC Press.
- ◆ Книга "Элементы статистического автоматического обучения" (Hastie T., Tibshirani R., Friedman J. Elements of Statistical Learning. — 2nd ed. — Springer, 2009) и ее более краткий вариант "Введение в статистическое автоматическое обучение" (Gareth J., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning. — Springer, 2013) содержат раздел по логистической регрессии.

- ◆ Книга "Добыча регулярностей из данных для бизнес-аналитики" ((Shmueli G., Bruce P., Patel N. Data mining for business analytics. — 3rd ed. — John Wiley & Sons, 2016) с вариантами для R, Python, Excel и JMP содержит полную главу по логистической регрессии.

Оценивание классификационных моделей

В предсказательном моделировании общепринято тренировать большое число разных моделей, применять каждую к отложенной выборке и оценивать их работоспособность. Иногда после оценивания и тонкой настройки ряда моделей и если имеется достаточно данных, третья отложенная выборка, не используемая ранее, используется для оценивания того, как выбранная модель будет работать с полностью новыми данными. Разные дисциплины и специалисты-практики также будут употреблять термины "*контроль*" (валидация) и "*тест*" для обозначения отложенной выборки (выборок). По сути, процесс оценивания результативности служит для того, чтобы попытаться усвоить то, какая модель производит наиболее точные и полезные предсказания.

Ключевые термины для классификационных моделей

Точность (accuracy)

Процент (или доля) случаев, классифицированных правильно.

Матрица путаницы (confusion matrix)

Табличное изображение (2×2 в двоичном случае) количеств записей по их предсказанному и фактическому статусу классификации.

Синонимы: матрица неточностей, матрица ошибок.

Чувствительность (sensitivity)

Процент (или доля) всех единиц, правильно классифицированных как единицы.

Синоним: полнота.

Специфичность (specificity)

Процент (или доля) всех нулей, правильно классифицированных как нули.

Прецизионность (precision)

Процент (или доля) предсказанных единиц, которые фактически являются единицами.

ROC-кривая (ROC curve)

График чувствительности против специфичности.

Лифт (lift)

Мера эффективности модели в идентификации (сравнительно редких) единиц при разных порогах вероятности.

Простой способ измерить результативность классификации состоит в том, чтобы подсчитать долю предсказаний, которые являются правильными, т. е. меру *точности*. Точность — это просто мера суммарной ошибки:

$$\text{точность} = \frac{\sum_{\text{истинноположительный}} + \sum_{\text{истинноотрицательный}}}{\text{размер выборки}}.$$

В большинстве классификационных алгоритмов каждому случаю назначается "оценочная вероятность того, чтобы он был равен 1"⁴. Дефолтная точка принятия решения, или отсечение, в типичной ситуации равна 0,50, или 50%. Если вероятность выше 0,5, то результатом классификации является "1", в противном случае — 0. Альтернативным дефолтным отсечением является преобладающая вероятность единиц в данных.

Матрица путаницы

В сердцевине классификационных метрик лежит *матрица путаницы* — это таблица, показывающая число правильных и неправильных предсказаний, сгруппированных по типу отклика. В R и Python имеется несколько пакетов, предназначенных для вычисления матрицы путаницы, но в двоичном случае она легко вычисляется вручную.

Для того чтобы проиллюстрировать матрицу путаницы, рассмотрим модель `logistic_gam`, которая была натренирована на сбалансированном наборе данных с равным числом невыплаченных и погашенных ссуд (см. рис. 5.4). Следуя принятым правилам, $Y=1$ соответствует интересующему событию (например, невыплата), и $Y=0$ соответствует отрицательному (либо обычному) событию (например, погашено). Следующий фрагмент кода вычисляет матрицу путаницы для модели `logistic_gam`, примененной ко всему (несбалансированному) тренировочному набору в R:

```
pred <- predict(logistic_gam, newdata=train_set)
pred_y <- as.numeric(pred > 0)
true_y <- as.numeric(train_set$outcome=='default')
true_pos <- (true_y==1) & (pred_y==1)
true_neg <- (true_y==0) & (pred_y==0)
false_pos <- (true_y==0) & (pred_y==1)
false_neg <- (true_y==1) & (pred_y==0)

conf_mat <- matrix(c(sum(true_pos), sum(false_pos),
                      sum(false_neg), sum(true_neg)), 2, 2)
colnames(conf_mat) <- c('Yhat = 1', 'Yhat = 0')
rownames(conf_mat) <- c('Y = 1', 'Y = 0')
```

⁴ Не все методы обеспечивают несмещенные оценки вероятности. В большинстве случаев достаточно того, что метод обеспечивает ранжирование, эквивалентное ранжированиям, которые были бы результатом несмещенной оценки вероятности; метод отсечения тогда является функционально эквивалентным.

```
conf_mat
      Yhat = 1 Yhat = 0
Y = 1 14295    8376
Y = 0 8052    14619
```

В Python:

```
pred = logit_reg.predict(X)
pred_y = logit_reg.predict(X) == 'default'
true_y = y == 'default'
true_pos = true_y & pred_y
true_neg = ~true_y & ~pred_y
false_pos = ~true_y & pred_y
false_neg = true_y & ~pred_y

conf_mat = pd.DataFrame([[np.sum(true_pos), np.sum(false_neg)],
                        [np.sum(false_pos), np.sum(true_neg)]],
                        index=['Y = default', 'Y = paid off'],
                        columns=['Yhat = default', 'Yhat = paid off'])

conf_mat
```

Предсказанными исходами являются столбцы, а истинными исходами — строки. Диагональные элементы матрицы показывают число правильных предсказаний, внедиагональные элементы — число неправильных предсказаний. Например, 14,295 невыплаченных ссуд были предсказаны правильно, как невыплаченные, но 8,376 невыплаченных ссуд были предсказаны неправильно, как погашенные.

На рис. 5.5 показана связь между матрицей путаницы для двоичного отклика Y и разные метрики (подробности о метриках см. в разделе "Прецизионность, полнота и специфичность" далее в этой главе). Как и в примере с данными о ссудах, фактический отклик расположен вдоль строк и предсказанный отклик — вдоль столбцов. Диагональные поля (левый верхний, правый нижний) показывают, когда предсказания \hat{Y} правильно предсказывают отклик. Одна из важных метрик, которая явно не упоминается, — это интенсивность ложноположительных *исходов* (зеркальное

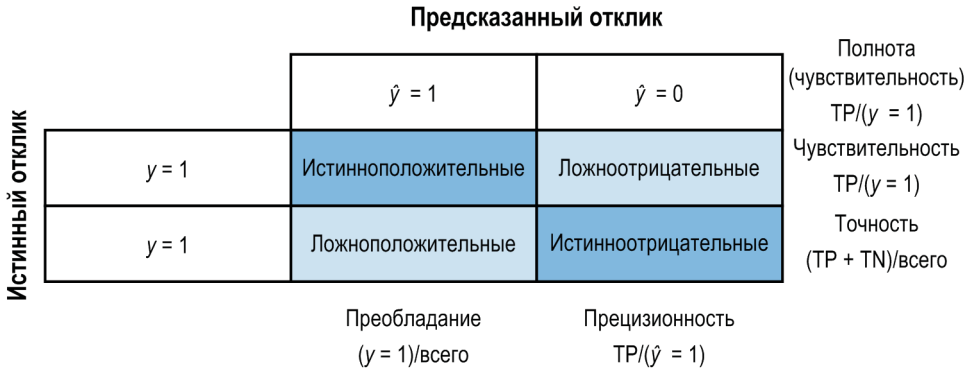


Рис. 5.5. Матрица путаницы для двоичного отклика и различных метрик

отражение прецизионности). Когда единицы встречаются редко, отношение ложноположительных исходов ко всем предсказанным положительным исходам может быть высокой, приводя к нелогичной ситуации, где предсказанная 1, скорее всего, является 0. Эта проблема — бич для широко применяемых диагностических тестов при медицинском обследовании (например, маммограммы): из-за относительной редкости условия положительные тесты, вероятнее всего, не означают рак молочной железы. Это приводит к путанице среди публики.



Здесь мы представляем фактический отклик вдоль строк и предсказанный отклик вдоль столбцов, но нередко это происходит наоборот. Примечательным примером является популярный пакет `caret` в R.

Проблема редкого класса

Во многих случаях существует несбалансированность в предсказываемых классах, когда один класс преобладает намного больше, чем остальные, например законные страховые иски против мошеннических исков либо простые посетители против покупателей на веб-сайте. Редкий класс (например, мошеннические иски) — это обычно класс, который обычно вызывает больший интерес и обозначается единицей, в отличие от более преобладающего, обозначаемого нулями. В типичном сценарии единицы являются более важным случаем в том смысле, что неправильная их классификация в качестве нулей стоит дороже, чем неправильная классификация нулей как единиц. Например, правильная идентификация мошеннического страхового иска может сэкономить тысячи долларов. С другой стороны, правильная идентификация не мошеннического иска просто экономит вам затраты и усилия на перелопачивание иска вручную с более тщательным рассмотрением (что вы и будете делать, если бы иск был помечен как "мошеннический").

В таких случаях, если только классы легко не разделимы, наиболее точной классификационной моделью может быть та, которая просто-напросто классифицирует все как 0. Например, если только 0,1% посетителей в веб-магазине в итоге делают покупку, то модель, которая предсказывает, что каждый посетитель уйдет без покупки, будет точна на 99,9%. И тем не менее она будет бесполезной. Вместо этого мы были бы довольны моделью, которая менее точна в совокупности, но способна вычленять покупателей, даже если она по пути неправильно классифицирует каких-либо непокупателей.

Прецизионность, полнота и специфичность

Метрики, иные, чем чистая точность, — метрики, носящие более нюансированный характер, — широко используются при оценивании классификационных моделей. Несколько из них имеют в статистике долгую историю, в особенности в биостатистике, где они используются для описания ожидаемой результативности диагно-

стических тестов. *Прецизионность* измеряет точность предсказанного положительного исхода (см. рис. 5.5):

$$\text{прецизионность} = \frac{\sum \text{ИП}}{\sum \text{ИП} + \sum \text{ЛП}},$$

где ИП — истинноположительный; ЛП — ложноположительный.

Полнота, также именуемая чувствительностью, измеряет силу модели в предсказании положительного исхода — доля единиц, которые она правильно идентифицирует (см. рис. 5.5). Термин "*чувствительность*" часто используется в биостатистике и медицинской диагностике, тогда как полнота больше употребляется в сообществе машинного обучения. Определение полноты таково:

$$\text{полнота} = \frac{\sum \text{ИП}}{\sum \text{ИП} + \sum \text{ЛО}},$$

где ИП — истинноположительный; ЛО — ложноотрицательный.

Еще одна используемая метрика — это *специфичность*, которая измеряет способность модели предсказывать отрицательный исход:

$$\text{специфичность} = \frac{\sum \text{ИО}}{\sum \text{ИО} + \sum \text{ЛО}},$$

где ИО — истинноотрицательный; ЛО — ложноотрицательный.

Мы можем вычислить эти три метрики из `conf_mat` в R:

```
# прецизионность
conf_mat[1,1]/sum(conf_mat[,1])
# полнота
conf_mat[1,1]/sum(conf_mat[1,])
# специфичность
conf_mat[2,2]/sum(conf_mat[2,])
```

Вот эквивалентный код для расчета этих метрик в Python:

```
conf_mat = confusion_matrix(y, logit_reg.predict(X))
print('Прецизионность', conf_mat[0, 0] / sum(conf_mat[:, 0]))
print('Полнота', conf_mat[0, 0] / sum(conf_mat[0, :]))
print('Специфичность', conf_mat[1, 1] / sum(conf_mat[1, :]))
```

```
precision_recall_fscore_support(y, logit_reg.predict(X),
                                labels=['default', 'paid off'])
```

Пакет `scikit-learn` имеет специализированный метод `precision_recall_fscore_support`, который вычисляет точность и полноту/специфичность одновременно.

ROC-кривая

Вы видите, что между полнотой и специфичностью имеется компромисс. Улавливание большого числа единиц обычно означает неправильную идентификацию большого числа нулей как единиц. Идеальный классификатор будет превосходно справляться с идентификацией единиц без неправильной идентификации большого числа нулей как единиц.

Метрика, которая фиксирует этот компромисс, называется кривой "рабочих характеристик получателя", обычно именуемой *ROC-кривой* (receiver operating characteristics). ROC-кривая наносит на график полноту (чувствительность) на оси y относительно специфичности на оси x ⁵. ROC-кривая показывает компромисс между полнотой и специфичностью по мере изменяемого вами порога отсечения, чтобы определить, каким образом классифицировать запись. На графике чувствительность (полнота) отображается на оси y , и вы можете встретить две формы маркировки оси x :

- ◆ специфичность наносится на оси x , где 1 слева и 0 справа;
- ◆ 1-специфичность наносится на оси x , где 0 слева и 1 справа.

Кривая выглядит идентичной независимо от того, как это делается. Процесс вычисления ROC-кривой таков:

1. Отсортировать записи по предсказанной вероятности быть единицей, начиная с наиболее вероятной и заканчивая наименее вероятной.
2. Вычислить кумулятивную специфичность и полноту, основываясь на сортированных записях.

Вычисление ROC-кривой в R выполняется достаточно прямолинейно. Следующий ниже фрагмент кода вычисляет ROC для данных о ссудах:

```
idx <- order(-pred)
recall <- cumsum(true_y[idx]==1)/sum(true_y==1)
specificity <- (sum(true_y==0) - cumsum(true_y[idx]==0))/sum(true_y==0)
roc_df <- data.frame(recall = recall, specificity = specificity)
ggplot(roc_df, aes(x=specificity, y=recall)) +
  geom_line(color='blue') +
  scale_x_reverse(expand=c(0, 0)) +
  scale_y_continuous(expand=c(0, 0)) +
  geom_line(data=data.frame(x=(0:100)/100), aes(x=x, y=1-x),
    linetype='dotted', color='red')
```

В Python мы можем использовать функцию `sklearn.metrics.roc_curve` пакета `scikit-learn` для вычисления информации, необходимой для ROC-кривой. Вы можете найти похожие пакеты для R, например, `ROCR`:

⁵ Кривая ROC впервые была использована во время Второй мировой войны для описания результатов радарных приемных станций, работа которых состояла в том, чтобы правильно идентифицировать (классифицировать) отраженные радарные сигналы и предупреждать силы обороны о приближающемся самолете.

```
fpr, tpr, thresholds = roc_curve(y, logit_reg.predict_proba(X)[: ,0],
                                pos_label='default')
roc_df = pd.DataFrame({'recall': tpr, 'specificity': 1 - fpr})

ax = roc_df.plot(x='specificity', y='recall', figsize=(4, 4), legend=False)
ax.set_ylim(0, 1)
ax.set_xlim(1, 0)
ax.plot((1, 0), (0, 1))
ax.set_xlabel('Специфичность')
ax.set_ylabel('Полнота')
```

Результат показан на рис. 5.6. Пунктирная диагональная линия соответствует классификатору, который не лучше случайности. Чрезвычайно эффективный классификатор (или в медицинских ситуациях чрезвычайно эффективный диагностический тест) будет иметь ROC-кривую, которая прижимается к левому верхнему углу, — она правильно идентифицирует много единиц без неправильной классификации многих нулей как единицы. Если для этой модели нам нужен классификатор со специфичностью по крайней мере 50%, то полнота составит примерно 75%.

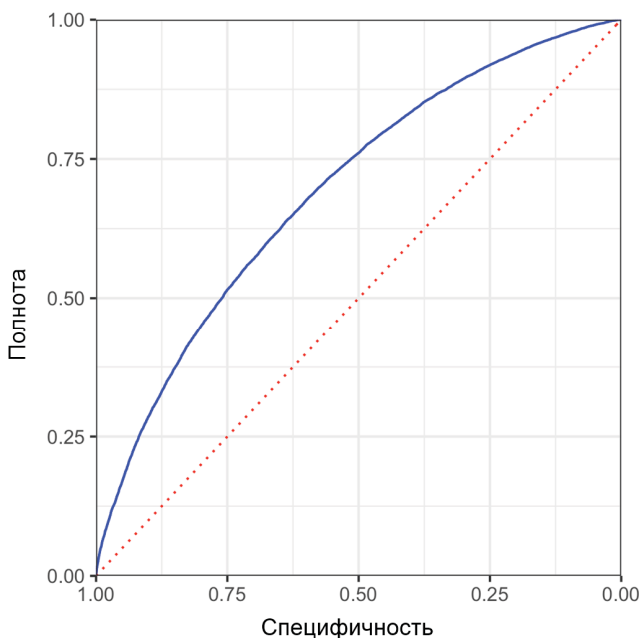


Рис. 5.6. ROC-кривая для данных о ссудах



Кривая прецизионности-полноты

В дополнение к ROC-кривым в информативных целях может быть полезным обследовать кривую прецизионности-полноты, или PR-кривую⁶ (precision-recall). PR-кривые вычисляются схожим образом за исключением того, что данные упорядочены от наименьшей вероятности до наибольшей, и вычисля-

⁶ См. https://oreil.ly/_89Pr.

ются кумулятивные статистики прецизионности и полноты. PR-кривые в особенности полезны в оценивании данных с очень несбалансированными исходами.

Площадь под ROC-кривой

ROC-кривая является ценным графическим инструментом, но сама по себе не представляет единственную меру результативности классификатора. ROC-кривая может использоваться тем не менее для порождения метрики AUC (area under the ROC curve). Метрика AUC — это просто общая площадь под ROC-кривой. Чем больше значение AUC, тем эффективнее классификатор. AUC, равный 1, говорит об идеальном классификаторе: он правильно идентифицирует все единицы и не идентифицирует неправильно любые нули, как единицы.

Совершенно неэффективный классификатор — диагональная прямая — будет иметь AUC, равный 0,5.

На рис. 5.7 показана площадь под ROC-кривой для модели ссуд. Значение AUC может быть вычислено путем численного интегрирования в R:

```
sum(roc_df$recall[-1] * diff(1-roc_df$specificity))  
[1] 0.5924072
```

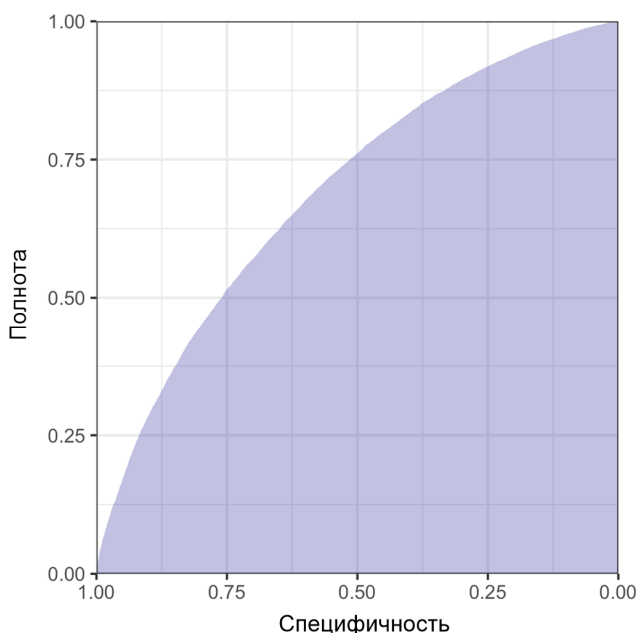


Рис. 5.7. Площадь под ROC-кривой для данных о ссудах

В Python мы можем вычислить точность, как показано для R, либо использовать функцию `sklearn.metrics.roc_auc_score` пакета `scikit-learn`. Вам нужно будет указать ожидаемое значение в виде 0 или 1:


```
print(np.sum(roc_df.recall[:-1] * np.diff(1 - roc_df.specificity)))
print(roc_auc_score([1 if yi == 'default' else 0 for yi in y],
                    logit_reg.predict_proba(X)[:, 0]))
```

Модель имеет AUC примерно 0,69, что соответствует относительно слабому классификатору.



Путаница с интенсивностью ложноположительных исходов

Интенсивности ложноположительных/ложноотрицательных исходов часто путают или объединяют со специфичностью или чувствительностью (даже в публикациях и программно-информационном обеспечении!). Иногда интенсивность ложноположительных исходов определяется как доля истинноотрицательных, которые тестируются положительно. Во многих случаях (таких, как обнаружение вторжения в сеть) указанный термин используется для обозначения доли положительных сигналов, которые суть истинноотрицательные.

Лифт

Использование AUC в качестве метрики для оценивания модели является улучшением по сравнению с простой точностью, поскольку она может оценивать то, насколько хорошо классификатор справляется с компромиссом между общей точностью и потребностью идентифицировать более важные единицы. Однако она не полностью решает проблему редкого случая, где вам необходимо понизить вероятностный порог отсеечения в модели ниже 0,5 во избежание идентификации всех записей, как 0. В таких случаях, чтобы классифицировать запись как 1, возможно, будет достаточным иметь вероятность 0,4; 0,3 или ниже. В результате мы приходим к тому, что сверхидентифицируем единицы, отражая их большую важность.

Изменение этого порога отсеечения повысит ваши шансы улавливать единицы (за счет неправильной классификации большего числа нулей как единиц). Но каким является оптимальный порог отсеечения?

Понятие лифта позволяет отложить ответ на этот вопрос. Вместо этого вы рассматриваете записи в порядке их предсказанной вероятности быть единицами. Скажем, из верхних 10%, классифицированных как единицы, насколько лучше алгоритм работал в сравнении с эталоном простого выбора вслепую? Если вы можете получить 0,3%-ный отклик в этом верхнем дециле вместо 0,1%-ного, который вы получаете по совокупности, выбирая случайным образом, то говорят, что алгоритм имеет *лифт* (или прирост — *gains*), равный 3 в верхнем дециле. График лифта (график прироста) квантифицирует это над диапазоном данных. Такой график можно построить подецильно либо непрерывно над диапазоном данных.

Для того чтобы вычислить график лифта, сначала строят *график кумулятивного прироста*, который показывает полноту на оси *y* и суммарное число записей на оси *x*. *Кривая лифта* — это отношение кумулятивного прироста к диагональной прямой, соответствующей случайному выбору. *Графики децильного прироста* — один из самых старых приемов в предсказательном моделировании, датируемый периодом до появления интернет-коммерции. Они были особенно популярны среди

профессионалов продажи товаров по почте. Продажа товаров по почте является дорогим методом рекламы, если его применять неразборчиво, и рекламодатели использовали предсказательные модели (в первые годы довольно простые) для выявления потенциальных клиентов с наиболее вероятной перспективой оплаты.



Подъем

Иногда термин "*подъем*" (uplift) употребляют для обозначения того же самого, что и лифт. Альтернативный смысл используется в более строгих условиях, когда был проведен *A/B*-тест, и затем вариант (*A* или *B*) используется в предсказательной модели в качестве предсказательной переменной. Подъем — это улучшение в отклике, предсказанном для *отдельного случая*, где вариант *A* противопоставляется варианту *B*. Она определяется путем оценивания результата отдельного случая, сначала когда предсказатель задан как *A* и затем снова, когда предсказатель по-новому задан как *B*. Маркетологи и консультанты политических кампаний используют этот метод для определения того, какой из двух вариантов послания должен использоваться и в отношении каких клиентов или избирателей.

Кривая лифта позволяет посмотреть на последствия установки разных отсечений вероятности для классифицирования записей как единиц. Он может быть промежуточным шагом в окончательном решении остановиться на подходящем уровне отсечения. Например, налоговый орган может иметь лишь ограниченный объем ресурсов, которые он может потратить на налоговые аудиты, и хочет потратить их на наиболее вероятные случаи налогового мошенничества. Учитывая ограниченность ресурсов, руководство пользуется графиком лифта для оценивания того, где прочертить прямую между налоговыми декларациями, отобранными для аудита, и теми, которые были оставлены в покое.

Ключевые идеи для оценивания классификационных моделей

- Точность (процент предсказанных идентификаций, которые являются правильными) является лишь первым шагом в оценивании модели.
- Другие метрики (полнота, специфичность, прецизионность) сосредоточены на более специфических характеристиках результативности (например, полнота измеряет то, насколько хорошо модель проявляет себя в правильной идентификации единиц).
- AUC (площадь под ROC-кривой) — это часто встречающаяся метрика способности модели отличать единицы от нулей.
- Схожим образом, лифт измеряет, насколько эффективной является модель в идентификации единиц, и его часто вычисляют подецильно, начиная с наиболее вероятных единиц.

Дополнительные материалы для чтения

Оценивание качества и работоспособности, как правило, рассматривается в контексте той или иной модели (например, k ближайших соседей или деревьев решений). Далее приведены три книги, в которых излагается эта тема в отдельных главах.

- ♦ "Добыча регуляризаторов из данных" (Whitten I., Frank E., Hall M. Data Mining. — 3rd ed. — Morgan Kaufmann, 2011).
- ♦ "Современная наука о данных с помощью R" (Baumer B., Kaplan D., Horton N. Modern Data Science with R. — CRC Press, 2017).
- ♦ "Добыча регуляризаторов из данных для бизнес-аналитики" (Shmueli G., Bruce P., Patel N. Data mining for business analytics. — 3rd ed. — John Wiley & Sons, 2016) с вариантами для R, Python, Excel и JMP.

Стратегии для несбалансированных данных

В предыдущем разделе рассматривалось оценивание классификационных моделей с помощью метрик, которые выходят за пределы простой точности и годятся для несбалансированных данных — данных, в которых интересующий исход (покупают на веб-сайте, мошенничество со страховкой и т. д.) редок. В этом разделе мы обратимся к дополнительным стратегиям, которые способны улучшить результативность предсказательного моделирования с несбалансированными данными.

Ключевые термины для несбалансированных данных

Понижающий отбор (undersample)

Использовать меньше записей с преобладающим классом в классификационной модели.

Повышающий отбор (oversample)

Использовать больше записей с редкими классами в классификационной модели, задействуя бутстрапирование при необходимости.

Повышающая или понижающая перевесовка (up weight or down weight)

Назначать больший (или меньший) вес редкому (или преобладающему) классу в модели.

Генерация данных (data generation)

Процедура, аналогичная бутстрапированию, за исключением того, что каждая новая бутстрапированная запись немного отличается от своего источника.

z -оценка (z-score)

Значение, которое получается после стандартизации.

K

Число соседей, рассматриваемых при вычислении ближайшего соседа.

Понижающий отбор

Если у вас достаточно данных, как в случае с данными о ссудах, одно из решений состоит в *понижающем отборе* (недоотборе) преобладающего класса, чтобы моделируемые данные были сбалансированнее между нулями и единицами. Базовая идея недоотбора состоит в том, что данные для доминирующего класса имеют много избыточных записей. Работа с меньшим, более сбалансированным набором данных дает выгоды в результативности модели и упрощает подготовку данных, а также разведывание и пилотирование моделей.

Какое количество данных будет достаточным? Это зависит от приложения, но в целом наличие десятков тысяч записей для менее доминирующего класса будет достаточным. Чем легче различимы единицы от нулей, тем меньше необходимо данных.

Данные о ссудах, проанализированные в *разделе "Логистическая регрессия" ранее в этой главе*, были основаны на сбалансированном тренировочном наборе: половина ссуд была погашена, и другая половина не возвращена. Предсказанные значения были схожими: половина вероятностей была меньше 0,5, половина — больше 0,5. В полном наборе данных всего примерно 5% ссуд не были выплачены, как показано на R:

```
mean(full_train_set$outcome=='default')
[1] 0.1889455
```

На Python:

```
print('процент невыплаченных ссуд: ',
      100 * np.mean(full_train_set.outcome == 'default'))
```

Что происходит, если использовать полный набор данных для тренировки модели? Давайте посмотрим, как это выглядит на R:

```
full_model <- glm(outcome ~ payment_inc_ratio + purpose_ + home_ +
                  emp_len_ + dti + revol_bal + revol_util,
                  data=full_train_set, family='binomial')
pred <- predict(full_model)
mean(pred > 0)
[1] 0.003942094
```

И на Python:

```
predictors = ['payment_inc_ratio', 'purpose_', 'home_', 'emp_len_',
              'dti', 'revol_bal', 'revol_util']
outcome = 'outcome'
X = pd.get_dummies(full_train_set[predictors], prefix='', prefix_sep='',
                  drop_first=True)
y = full_train_set[outcome]

full_model = LogisticRegression(penalty='l2', C=1e42, solver='liblinear')
full_model.fit(X, y)
```

```
print('процент ссуд, предсказанных как невыплаченные: ',
      100 * np.mean(full_model.predict(X) == 'default'))
```

Только 0,39% ссуд были предсказаны как невыплаченные, или менее 1/47 от ожидаемого числа⁷. Погашенные ссуды подавляют количеством невыплаченные ссуды, потому что модель натренирована с использованием всех данных одинаково. Если подумать об этом на интуитивном уровне, то присутствие такого количества ссуд, не относящихся к невыплаченным, в сопряжении с неизбежной вариабельностью в предсказательных данных означает, что даже для невыплаченной ссуды модель, скорее всего, отыщет несколько не относящихся к невыплаченным ссуд, с которыми она будет случайно похожа. Когда же использовалась сбалансированная выборка, примерно 50% ссуд были предсказаны как невыплаченные.

Повышающий отбор и повышающая/понижающая перевесовка

Одно из критических замечаний по поводу метода понижающего отбора состоит в том, что он (метод) отбрасывает данные и не использует всю имеющуюся под рукой информацию. Если у вас есть относительно небольшой набор данных и более редкий класс содержит несколько сотен или несколько тысяч записей, тогда понижающий отбор доминирующего класса имеет риск выбросить полезную информацию. В этом случае вместо понижающего отбора доминирующего случая нужно выполнить повышающий отбор более редкого класса путем извлечения дополнительных строк с возвратом (бутстрапирование).

Вы можете добиться похожего эффекта путем взвешивания данных. Многие классификационные алгоритмы принимают весовой аргумент, который позволит вам выполнять повышающую/понижающую перевесовку данных. Например, применим весовой вектор к данным о ссудах при помощи аргумента `weight` к `glm` в R:

```
wt <- ifelse(full_train_set$outcome=='default',
             1 / mean(full_train_set$outcome == 'default'), 1)
full_model <- glm(outcome ~ payment_inc_ratio + purpose_ + home_ +
                  emp_len_ + dti + revol_bal + revol_util,
                  data=full_train_set, weight=wt, family='quasibinomial')
pred <- predict(full_model)
mean(pred > 0)
[1] 0.5767208
```

Большинство методов пакета `scikit-learn` позволяют задавать веса в функции `fit` с помощью именованного аргумента `sample_weight`:

```
default_wt = 1 / np.mean(full_train_set.outcome == 'default')
wt = [default_wt if outcome == 'default' else 1
      for outcome in full_train_set.outcome]
```

⁷ Из-за различий в имплементации результаты на Python незначительно отличаются: 1%, или около 1/18 от ожидаемого числа.

```
full_model = LogisticRegression(penalty="l2", C=1e42, solver='liblinear')
full_model.fit(X, y, sample_weight=wt)
print('процент ссуд, предсказанных как невыплаченные (взвешивание): ',
      100 * np.mean(full_model.predict(X) == 'default'))
```

Веса для невыплаченных ссуд установлены равными $1/p$, где p — это вероятность невыплаты. Ссуды, не относящиеся к невыплаченным, имеют вес 1. Сумма весов для невыплаченных ссуд и ссуд, не относящихся к невыплаченным, примерно эквивалентны. Среднее предсказанных значений теперь составляет примерно 58% вместо 0,39%.

Обратите внимание, что взвешивание обеспечивает альтернативу как для повышающего отбора более редкого класса, так и для понижающего отбора доминирующего класса.



Адаптация функции потери

Многие классификационные и регрессионные алгоритмы оптимизируют некоторый критерий, или *функцию потери*. Например, логистическая регрессия пытается минимизировать девиантность. В специализированной литературе некоторые разработчики предлагают модифицировать функцию потери, чтобы предотвратить проблемы, вызываемые редким классом. На практике это сопряжено с трудностями: классификационные алгоритмы могут быть многосложными и трудномодифицируемыми. Взвешивание является простым способом внесения изменений в функцию потери, дисконтируя ошибки для записей с низкими весами в пользу записей с более высокими весами.

Генерация данных

Вариацией повышающего отбора посредством бутстрапирования (см. раздел "*Повышающий отбор и повышающая/понижающая перевесовка*" далее в этой главе) является *генерация данных* путем перестановки существующих записей для создания новых записей. Логика, лежащая в основе этой идеи, состоит в том, что, поскольку мы наблюдаем только предельный набор случаев, алгоритм не имеет богатого набора информации для построения "правил" классификации. Путем создания новых записей, которые похожи, но не идентичны существующим записям, алгоритм имеет возможность усвоить более робастный набор правил. Эта идея по духу подобна ансамблевым статистическим моделям, в частности бэггингу и бустингу (см. главу 6).

Указанная идея набрала обороты с публикацией алгоритма SMOTE, название которого расшифровывается как "техника повышающего отбора синтетического меньшинства" (synthetic minority oversampling technique). Алгоритм SMOTE находит запись, которая похожа на запись, подвергаемую повышающему отбору (см. раздел "*к ближайших соседей*" главы 6), и создает синтетическую запись, которая является случайно средневзвешенным исходной записи и соседней записи, где вес генерируется отдельно для каждого предсказателя. Число синтетических записей, взятых повышающим отбором, зависит от коэффициента повышающего отбора, который требу-

ется для приведения набора данных в приближенное равновесие относительно классов исхода.

В R имеется несколько имплементаций алгоритма SMOTE. Самым исчерпывающим пакетом для обработки несбалансированных данных является `unbalanced`. Он предлагает разнообразные технические приемы, включая гоночный алгоритм ("Racing") для отбора наилучшего метода. Однако алгоритм SMOTE настолько прост, что может быть имплементирован непосредственно в R с использованием пакета `FNN`.

Пакет `unbalanced-learn` языка Python имплементирует разнообразные методы с API, совместимым с пакетом `scikit-learn`. Он предоставляет различные методы для повышающего и понижающего отбора и поддержку для использования этих технических приемов с бустинговыми и бэггинговыми классификаторами.

Стоимостная классификация

На практике показатели точности и AUC представляют собой упрощенческие приемы выбора правила классификации. Зачастую оценочная стоимость может быть назначена ложноположительным исходам против ложноотрицательных, и более целесообразно включать эти стоимости для определения наилучшего порога отсечения при классификации единиц и нулей. Например, предположим, что оценочная стоимость неуплаты новой ссуды равна C и ожидаемый финансовый возврат от погашенной ссуды равняется R . Тогда ожидаемый финансовый возврат для этой ссуды составит:

$$\text{ожидаемый финансовый возврат} = P(Y = 0) \cdot R + P(Y = 1) \cdot C.$$

Вместо того чтобы просто пометить ссуду как невыплаченную или погашенную либо определить вероятность невыплаты, разумнее определить, имеет ли ссуда положительный ожидаемый финансовый возврат. Предсказанная вероятность невыплаты является промежуточным шагом, и она должна быть объединена с общей величиной ссуды для определения ожидаемой прибыли, которая является окончательной плановой метрикой в бизнесе. Например, ссуду с меньшей величиной можно оставить без внимания в пользу более крупной с немного более высокой вероятностью предсказания невыплаты.

Разведывание предсказаний

Одиночная метрика, такая как AUC, не может оценить все аспекты приемлемости модели для ситуации. На рис. 5.8 показаны правила классификации для четырех разных моделей (подогнанных к данным о ссудах) на основе всего двух предсказательных переменных: `borrower_score` и `payment_inc_ratio`. Моделями являются линейный дискриминантный анализ (LDA), логистическая линейная регрессия, логистическая регрессия, подогнанная с использованием обобщенной аддитивной модели (GAM), и древесная модель (см. раздел "Древесные модели" главы 6). Участок слева вверху от прямой соответствует предсказанной невыплате. Линейный дискрими-

нантный анализ и логистическая линейная регрессия в этом случае дают почти идентичные результаты. Деревянная модель производит наименее регулярное правило с двумя шагами. И наконец, подгонка логистической регрессии на основе GAM представляет собой компромисс между деревянной и линейной моделями.

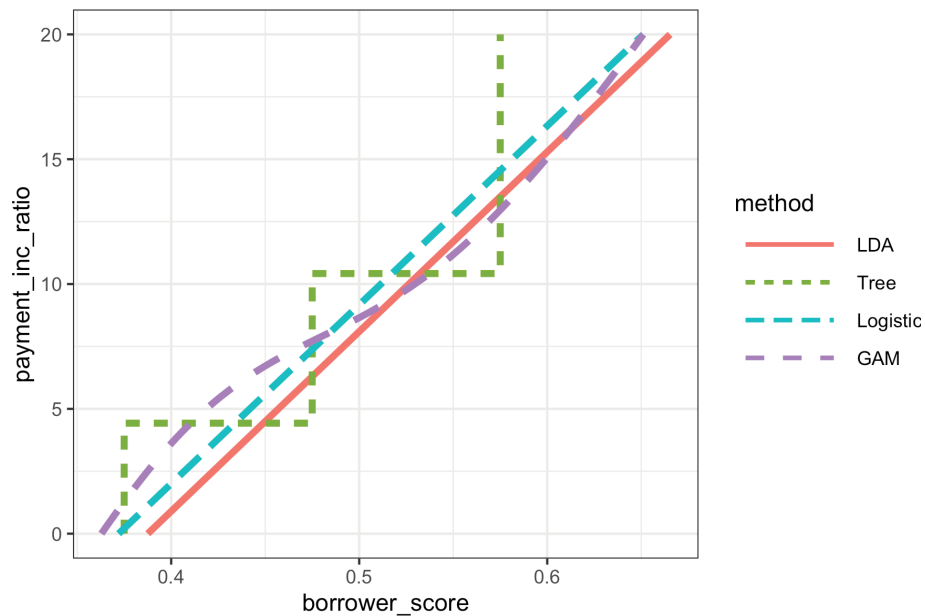


Рис. 5.8. Сравнение правил классификации для четырех разных методов

Не просто визуализировать правила предсказания в более высоких размерностях или, в случае GAM и деревянной модели, даже сгенерировать участки для таких правил.

В любом случае разведывательный анализ предсказанных значений всегда оправдан.

**Ключевые идеи для стратегий
в отношении несбалансированных данных**

- Очень несбалансированные данные (т. е. где интересные исходы, единицы, являются редкими) являются проблематичными для классификационных алгоритмов.
- Одна из стратегий состоит в том, чтобы сбалансировать тренировочные данные посредством понижающего отбора многочисленного случая (либо повышающего отбора редкого случая).
- Если использование всех единиц по-прежнему дает вам слишком мало единиц, то можно применить бутстрапирование редких случаев либо использовать алгоритм SMOTE для создания синтетических данных, похожих на существующие редкие случаи.

- Несбалансированные данные обычно указывают на то, что правильная идентификация класса единиц имеет более высокую величину и что коэффициент величины следует встраивать в оценочную метрику.

Дополнительные материалы для чтения

- ◆ У Тома Фосетта (Tom Fawcett), автора книги "Наука о данных для бизнеса" (Data Science for Business), есть хорошая статья⁸ о несбалансированных классах.
- ◆ Подробности об алгоритме SMOTE см. в статье "Техника повышающего отбора синтетического меньшинства"⁹ (Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P.. SMOTE: Synthetic Minority Over-sampling Technique // Journal of Artificial Intelligence Research. — 2002. — № 16. — P. 321–357).
- ◆ См. также "Практическое руководство по решению проблем несбалансированной классификации в R"¹⁰ (Practical Guide to deal with Imbalanced Classification Problems in R // 2016. — March. — 28) от группы контента портала Analytics Vidya.

Резюме

Классификация, т. е. процесс предсказания, к какой из двух или более категорий запись принадлежит, является фундаментальным инструментом предсказательной аналитики. Останется ли ссуда невыплаченной (да или нет)? Будет ли она погашена досрочно? Нажмет ли посетитель веб-сайта на ссылке? Купит ли он что-нибудь? Является ли страховой иск мошенническим? Нередко в классификационных задачах один класс представляет главный интерес (например, мошеннический страховой иск), и в двоичной классификации этот класс получает значение 1, а другой, более преобладающий класс, — значение 0. Нередко ключевая часть процесса состоит в исчислении *балла склонности*, т. е. вероятности принадлежать интересующему классу. Общепринятый сценарий заключается в том, что интересующий класс является относительно редким. При оценивании классификатора существуют разнообразные метрики результативности модели, которые выходят за рамки простой точности; они являются важными в ситуации редкого класса, когда классификация всех записей как нулей может порождать высокую точность.

⁸ См. <https://oreil.ly/us2rd>.

⁹ См. <https://oreil.ly/bwaIQ>.

¹⁰ См. <https://oreil.ly/gZUDs>.

Статистическое машинное обучение

Недавние достижения в статистике были посвящены развитию более мощных автоматизированных приемов для предсказательного моделирования — как регрессионных, так и классификационных. Эти приемы, как и обсуждавшиеся в предыдущей главе, являются *контролируемыми методами* — они тренируются на данных, где исходы известны, и учатся предсказывать исходы в новых данных. Они относятся к более широкой категории *статистического машинного обучения* и отличаются от классических статистических методов тем, что они управляются данными и не стремятся наложить линейную или иную совокупную структуру на данные. Метод k ближайших соседей, например, является довольно простым: классифицировать запись в соответствии с тем, как классифицируются похожие записи. Самые успешные и широко используемые технические приемы опираются на *ансамблевое обучение*, применяемое к *деревьям решений*. Основная идея ансамблевого обучения состоит в том, чтобы использовать много моделей (в отличие от использования одной-единственной модели), на их основе формировать предсказание. Деревья решений — это гибкий и автоматический технический прием, предназначенный для усвоения правил о связях между предсказательными переменными и переменными исхода. Как оказалось, комбинация ансамблевого обучения с деревьями решений приводит к наиболее результативным готовым к работе приемам предсказательного моделирования.

Развитие многих из этих приемов в статистическом машинном обучении можно проследить до статистиков Лео Бреймана (Leo Breiman) (рис. 6.1) в Калифорнийском университете в Беркли и Джерри Фридмана (Jerry Friedman) в Стэнфордском



Рис. 6.1. Лео Брейман, работавший профессором статистики в Беркли, находился на передовой развития многих приемов, лежащих сегодня в основе инструментария исследователя данных

университете. Их работа вместе с другими исследователями в Беркли и Стэнфорде началась в 1984 году с создания древесных моделей. Последующая в 1990-х годах разработка ансамблевых методов бэггинга и бустинга сформировала фундамент статистического машинного обучения.



Машинное обучение против статистики

В контексте предсказательного моделирования какова разница между машинным обучением и статистикой? Четкой разграничительной линии, которая разделяет эти две дисциплины, нет. Машинное обучение тяготеет к большему вниманию к разработке эффективных алгоритмов, которые масштабируются до больших данных в целях оптимизации предсказательной модели. Статистика обычно больше сосредоточена на теории вероятностей и опорной структуре модели. Бэггинг и случайный лес (см. раздел "Бэггинг и случайный лес" главы 6) выросли, прочно опираясь на статистику. Бустинг (см. раздел "Бустинг" главы 6), с другой стороны, был разработан в обеих дисциплинах, но получает больше внимания на стороне машинного обучения. Независимо от истории, перспективы бустинга гарантируют, что этот метод будет процветать и в статистике, и в машинном обучении.

***k* ближайших соседей**

В основе метода *k* ближайших соседей (KNN, k-nearest neighbors) лежит очень простая идея¹. Для каждой классифицируемой или предсказываемой записи:

- 1. Найти *k* записей, которые имеют похожие признаки (т. е. похожие значения предсказателей).
- 2. Для классификации: выяснить, каким является мажоритарный класс среди этих похожих записей, и назначить этот класс новой записи.
- 3. Для предсказания (также именуемого KNN-регрессией): отыскать среднее среди этих похожих записей и предсказать это среднее для новой записи.

Ключевые термины для *k* ближайших соседей

Сосед (neighbor)

Запись, которая имеет предсказательные значения, похожие на другую запись.

Метрики расстояния (distance metrics)

Метрики, которые суммируют в одном числе, насколько далеко одна запись находится от другой.

Стандартизация (standardization)

Вычесть среднее и разделить на стандартное отклонение.

Синоним: нормализация.

¹ Этот и последующие разделы в настоящей главе закреплены за © 2020 Datastats, LLC, Питер Брюс, Эндрю Брюс и Питер Гедек; использовано с разрешения.

z-оценка (z-score)

Значение, которое получается после стандартизации.

Синоним: стандартная оценка.

k

Число соседей, учитываемых при вычислении алгоритма ближайших соседей.

k ближайших соседей — это один из более простых технических приемов предсказания/классификации: модель, подлежащая подгонке (как в регрессии), отсутствует. Но это не означает, что использование k ближайших соседей выливается в автоматическую процедуру. Результаты предсказания зависят от того, каким образом признаки прошкалированы, каким образом измерено сходство и какая величина k задана. Кроме того, все предсказатели должны быть в числовой форме. Проиллюстрируем работу указанного метода примером классификации.

Небольшой пример: предсказание невыплаты ссуды

В табл. 6.1 приведены несколько записей данных о персональных ссудах в инвестиционно-кредитной компании Lending Club. Компания Lending Club является лидером в равноправном кредитовании, в котором пулы инвесторов выдают персональные ссуды физическим лицам. Цель анализа будет состоять в том, чтобы предсказать исход новой потенциальной ссуды: "погашено" против "невыплата".

Таблица 6.1. Несколько записей и столбцов из данных о ссудах инвестиционно-кредитной компании Lending Club

Исход	Величина ссуды	Доход	Цель	Стаж работы	Домовладение	Штат
Погашено	10 000	79 100	Консолидация долга	11	ИПОТЕКА	NV
Погашено	9600	48 000	Переезд	5	ИПОТЕКА	TN
Погашено	18 800	120 036	Консолидация долга	11	ИПОТЕКА	MD
Невыплата	15 250	232 000	Малый бизнес	9	ИПОТЕКА	CA
Погашено	17 050	35 000	Консолидация долга	4	АРЕНДА	MD
Погашено	5500	43 000	Консолидация долга	4	АРЕНДА	KS

Рассмотрим очень простую модель всего с двумя предсказательными переменными: `dti`, т. е. отношением выплат по задолженности (исключая ипотеку) к доходу, и `payment_inc_ratio`, т. е. отношением выплат по ссуде к доходу. Оба отношения умножены на 100. Используя небольшой набор `loan200` из 200 ссуд с известными двоичными исходами (невыплата или его противоположность, указанных в пред-

сказателе `outcome200`) и k , заданным с величиной 20, можно вычислить в R оценку классификатора на основе k ближайших соседей для новой предсказываемой ссуды `newloan` с `dti=22.5` и `payment_inc_ratio=9` следующим образом²:

```
newloan <- loan200[1, 2:3, drop=FALSE]
knn_pred <- knn(train=loan200[-1, 2:3], test=newloan, cl=loan200[-1, 1], k=20)
knn_pred == 'paid off'
[1] TRUE
```

Предсказание классификатора на основе k ближайших соседей для этой ссуды гласит — "не будет выплачена".

В то время как R имеет собственную функцию `knn`, сторонний пакет `FNN`³ в R (fast nearest neighbor — быстрый ближайший сосед) шкалирует эффективнее до больших данных и предоставляет больше гибкости.

Пакет `scikit-learn` обеспечивает быструю и эффективную имплементацию k ближайших соседей в Python:

```
predictors = ['payment_inc_ratio', 'dti']
outcome = 'outcome'

newloan = loan200.loc[0:0, predictors]
X = loan200.loc[1:, predictors]
y = loan200.loc[1:, outcome]

knn = KNeighborsClassifier(n_neighbors=20)
knn.fit(X, y)
knn.predict(newloan)
```

На рис. 6.2 дано визуальное изображение этого примера. Новая предсказываемая ссуда представлена крестиком в середине. Квадраты (погашено) и круги (невыплата) — это тренировочные данные. Большая черная окружность показывает границу ближайших 20 точек. В данном случае 9 невыплаченных ссуд лежат в пределах круга, по сравнению с 11 погашенными ссудами. Таким образом, предсказываемый исход ссуды является "погашено". Обратите внимание, что если мы рассмотрим только трех ближайших соседей, то будет предсказана невыплата ссуды.



Хотя результатом работы алгоритма k ближайших соседей для классификационной задачи в типичной ситуации является двоичное решение, в частности "невыплата" либо "погашено" для данных о ссудах, подпрограммы k ближайших соседей обычно предлагают возможность давать на выходе вероятность (склонность) между 0 и 1. Вероятность основывается на доле класса единиц в k наближайших соседях, выраженной дробью. В предыдущем примере эта вероятность невыплаты была бы оценена равной 14/20, или 0,7. Использование балла (score) вероятности позволяет применять иные правила классификации,

² В этом примере мы берем первую строку в наборе данных `loan200` как `newloan` и исключаем ее из набора данных для тренировки.

³ См. <https://oreil.ly/RMQFG>.

чем простые мажоритарные голоса (вероятность 0,5). Это имеет особо важное значение в задачах с несбалансированными классами (см. раздел "Стратегии для несбалансированных данных" главы 5). Например, если цель состоит в выявлении членов редкого класса, то порог отсечения, как правило, будет установлен ниже 50%. Один из часто встречающихся подходов состоит в установке порога на уровне вероятности редкого случая.

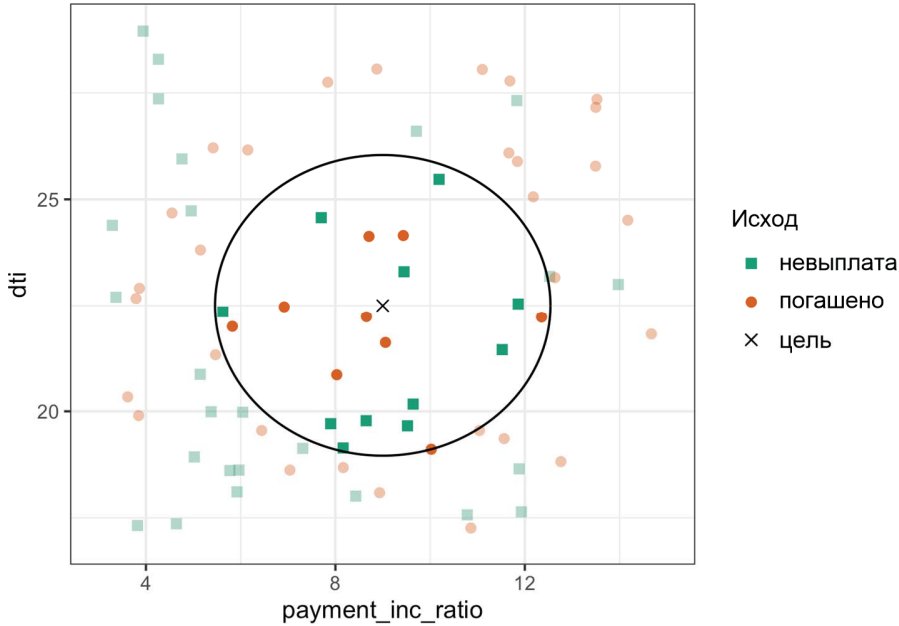


Рис. 6.2. Предсказание моделью на основе k ближайших соседей (KNN) невыплаты ссуды с использованием двух переменных: отношения задолженности к доходу и отношение платежей по ссуде к доходу

Метрики расстояния

Сходство (близость) определяют при помощи *метрики расстояния*, т. е. функции, которая измеряет, насколько далеко две записи (x_1, x_2, \dots, x_p) и (u_1, u_2, \dots, u_p) находятся друг от друга. Самой популярной метрикой расстояния между двумя векторами является *евклидово расстояние*. Для того чтобы измерить евклидово расстояние между двумя векторами, надо вычесть один из другого, возвести в квадрат разности компонент, просуммировать их и взять квадратный корень:

$$\sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + \dots + (x_p - u_p)^2}.$$

Еще одной часто встречающейся метрикой для числовых данных является манхэттенское расстояние (или *расстояние Минковского*):

$$|x_1 - u_1| + |x_2 - u_2| + \dots + |x_p - u_p|.$$

Евклидово расстояние соответствует расстоянию по прямой между двумя точками (как говорится, "так, как летит ворон"). Манхэттенское расстояние — это расстояние между двумя точками, пересекаемыми в одном направлении за один раз (например, перемещаясь вдоль прямоугольных городских кварталов). По этой причине манхэттенское расстояние является полезной аппроксимацией, в случае если сходство определяется как поточечное время в пути.

В измерении расстояния между двумя векторами переменные (признаки), которые измеряются по сравнительно крупной шкале, будут доминировать над этой мерой. Например, для данных о ссудах расстояние было бы почти исключительно функцией от переменных дохода и суммы кредита, которые измеряются в десятках или сотнях тысяч. Переменные на основе соотношений при сравнении будут практически сведены на нет. Указанная проблема решается путем стандартизации данных; см. раздел "*Стандартизация (нормализация, z-оценки)*" далее в этой главе.



Другие метрики расстояния

Для измерения расстояния между векторами существуют другие многочисленные метрики. Для числовых данных привлекательным является *расстояние Махаланобиса*, т. к. оно объясняет корреляцию между двумя переменными. Это полезно, поскольку если две переменные высоко коррелированы, то Махаланобис будет, в сущности, рассматривать их как одну переменную с точки зрения расстояния. Евклидово и манхэттенское расстояния не объясняют корреляцию, практически возлагая больший вес на атрибут, который лежит в основе этих признаков. Обратной стороной применения расстояния Махаланобиса является увеличение вычислительных усилий и сложности; оно вычисляется при помощи *матрицы ковариаций* (см. раздел "*Матрица ковариаций*" главы 5).

Кодировщик с одним активным состоянием

Данные о ссудах в табл. 6.1 включают несколько факторных (строковых) переменных. Большинство моделей статистического и машинного обучения требует, чтобы этот тип переменной был конвертирован в серию двоичных фиктивных переменных, несущих одинаковую информацию, как в табл. 6.2. Вместо одной переменной, обозначающей статус домовладельца, а именно "владеет с ипотекой", "владеет без ипотеки", "арендует" или "другой", мы в итоге приходим к четырем двоичным переменным. Первая будет "владеет с ипотекой — да/нет", вторая будет "владеет без ипотеки — да/нет" и т. д. Этот один предсказатель, статус домовладельца, таким образом, порождает вектор с одной единицей и тремя нулями, который может использоваться в алгоритмах статистического и машинного обучения. Словосочетание "*кодирование с одним активным состоянием*" (one hot encoding) пришло из терминологии цифровых интегральных микросхем, где оно описывает конфигурацию микросхемы, в которой допускается, чтобы только один бит был положительным (активным)⁴.

⁴ В отечественной специализированной литературе для данного типа кодировщика нередко используется менее точный термин "прямой унитарный кодировщик". — Прим. перев.

Таблица 6.2. Представление факторных данных о домовладельце в виде числовой фиктивной переменной

OWNS_WITH_MORTGAGE	OWNS_WITHOUT_MORTGAGE	OTHER	RENT
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
0	0	0	1
0	0	0	1



В линейной и логистической регрессии кодирование с одним активным состоянием вызывает проблемы, связанные с мультиколлинеарностью (см. раздел "Мультиколлинеарность" главы 4). В таких случаях одна фиктивная переменная исключается (ее значение может быть выведено из других значений). Это не представляет проблему с k ближайшими соседями и другими методами, обсуждаемыми в данной книге.

Стандартизация (нормализация, z-оценки)

В мерных данных нас часто в первую очередь интересует не их величина, а насколько они отличаются от среднего. Стандартизация, также часто именуемая *нормализацией*, помещает все переменные на схожие шкалы путем вычитания среднего и деления на стандартное отклонение. Таким образом мы обеспечиваем, чтобы переменная чрезмерно не влияла на модель просто в силу шкалы ее исходных показаний измерения.

$$z = \frac{x - \bar{x}}{s}.$$

Результат этого преобразования принято называть стандартной оценкой, или *z-оценкой*. Показания измерений в дальнейшем используются в "стандартных отклонениях от среднего".



Нормализацию в указанном статистическом контексте не следует путать с *нормализацией баз данных*, т. е. удалением избыточных данных и верификацией зависимостей в данных.

Для k ближайших соседей и нескольких других процедур (например, анализа главных компонент и кластеризации) крайне важно подумать о стандартизировании данных перед тем, как применять процедуру. В качестве иллюстрации этой идеи k ближайших соседей применяется к данным о ссудах с использованием переменных `dti` и `payment_inc_ratio` (см. раздел "Небольшой пример: предсказание невыплаты ссуды" ранее в этой главе) плюс двух других переменных: `revol_bal` — общего

возобновляемого кредита, доступного для заявителя в долларах, и `revol_util` — процента использования кредита. Новая предсказываемая запись показана ниже:

```
newloan
  payment_inc_ratio dti revol_bal revol_util
1          2.3932    1      1687         9.4
```

Величина `revol_bal`, которая исчисляется в долларах, намного больше других переменных. Функция `knn` возвращает индекс ближайших соседей как атрибут `nn.index`, и он может использоваться для показа верхних пяти ближайших строк в кадре данных `loan_df`:

```
loan_df <- model.matrix(~ -1 + payment_inc_ratio + dti + revol_bal +
                        revol_util, data=loan_data)
newloan <- loan_df[1, , drop=FALSE]
loan_df <- loan_df[-1,]
outcome <- loan_data[-1, 1]
knn_pred <- knn(train=loan_df, test=newloan, cl=outcome, k=5)
loan_df[attr(knn_pred, "nn.index"),]
```

```
      payment_inc_ratio dti revol_bal revol_util
35537          1.47212 1.46      1686         10.0
33652          3.38178 6.37      1688          8.4
25864          2.36303 1.39      1691          3.5
42954          1.28160 7.14      1684          3.9
43600          4.12244 8.98      1684          7.2
```

После подгонки модели мы можем использовать метод `kneighbors` для выявления пяти ближайших строк в тренировочном наборе с помощью пакета `scikit-learn`:

```
predictors = ['payment_inc_ratio', 'dti', 'revol_bal', 'revol_util']
outcome = 'outcome'
```

```
newloan = loan_data.loc[0:0, predictors]
X = loan_data.loc[1:, predictors]
y = loan_data.loc[1:, outcome]
```

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, y)
```

```
nbrs = knn.kneighbors(newloan)
X.iloc[nbrs[1][0], :]
```

Значение `revol_bal` в этих соседях находится очень близко к его значению в новой записи, но другие предсказательные переменные разбросаны повсюду и, по существу, не играют роли в определении соседей.

Сравним это с классификатором на основе k ближайший соседей, примененному к стандартизированным данным при помощи функции `scale` в R, которая вычисляет z -оценку для каждой переменной:

```

loan_df <- model.matrix(~ -1 + payment_inc_ratio + dti + revol_bal +
                        revol_util, data=loan_data)
loan_std <- scale(loan_df)
newloan_std <- loan_std[1, , drop=FALSE]
loan_std <- loan_std[-1,]
loan_df <- loan_df[-1,] ❶
outcome <- loan_data[-1, 1]
knn_pred <- knn(train=loan_std, test=newloan_std, cl=outcome, k=5)
loan_df[attr(knn_pred, "nn.index"),]

```

	payment_inc_ratio	dti	revol_bal	revol_util
2081	2.61091	1.03	1218	9.7
1439	2.34343	0.51	278	9.9
30216	2.71200	1.34	1075	8.5
28543	2.39760	0.74	2917	7.4
44738	2.34309	1.37	488	7.2

❶ Нам также нужно удалить первую строку из `loan_df`, чтобы номера строк соответствовали друг другу.

Метод `sklearn.preprocessing.StandardScaler` сначала тренируется предсказателями и в последующем используется для преобразования набора данных перед тем, как приступить к тренировке модели k ближайших соседей:

```

newloan = loan_data.loc[0:0, predictors]
X = loan_data.loc[1:, predictors]
y = loan_data.loc[1:, outcome]

scaler = preprocessing.StandardScaler()
scaler.fit(X * 1.0)

X_std = scaler.transform(X * 1.0)
newloan_std = scaler.transform(newloan * 1.0)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_std, y)

nbrs = knn.kneighbors(newloan_std)
X.iloc[nbrs[1][0], :]

```

Пять ближайших соседей намного больше похожи во всех переменных, обеспечивая более разумный результат. Обратите внимание на то, что результаты изображены в исходной шкале, но классификатор на основе k ближайших соседей был применен к прошкалированным данным и новой предсказываемой ссуде.



Использование z -оценки — это всего лишь один способ перешкалирования переменных. Вместо среднего может использоваться более робастная оценка местоположения, такая как медиана. Похожим образом может использоваться и другая оценка шкалы, такая как межквартильный размах, вместо стандартного отклонения. Иногда переменные "впихивают" в интервал 0–1. Важно по-

нять, что шкалирование каждой переменной так, чтобы она имела единичную дисперсию, носит несколько произвольный характер. Из этого вытекает, что каждая переменная, как считается, имеет одинаковую важность в предсказательной силе. Если вы располагаете субъективным знанием о том, что какие-то переменные важнее других, тогда их можно прошкалировать в сторону повышения. Например, в случае данных о ссудах разумно ожидать, что отношение платежей к доходу имеет большую важность.



Нормализация (стандартизация) не меняет форму распределения данных; она не придает им нормальную форму, если только они ее уже не имеют (см. раздел "Нормальное распределение" главы 2).

Выбор числа k

Выбор числа k имеет очень важное значение для результативности классификатора на основе k ближайших соседей. Самый простой выбор состоит в том, чтобы установить $k = 1$, что соответствует классификатору на основе одного ближайшего соседа. Предсказание интуитивно представляется логичным: оно основывается на отыскании записи в тренировочном наборе, наиболее схожей с новой предсказываемой записью. Установление $k = 1$ редко является лучшим выбором; вы почти всегда будете получать превышающую результативность, используя $k > 1$ ближайших соседей.

Вообще говоря, если значение k является слишком низким, то мы можем получить переподогнанную модель, включая шум в данных. Более высокие значения k обеспечивают сглаживание, которое снижает риск переподгонки в тренировочных данных. С другой стороны, если k является слишком высоким, то мы можем вызвать излишнее сглаживание данных и упустить способность классификатора на основе k ближайших соседей улавливать локальную структуру в данных, одно из его главных преимуществ.

Число k , которое наилучшим образом балансирует между переподгонкой и сверхсглаживанием, в типичной ситуации определяется точностными метриками и, в частности, точностью с отложенными или контрольными данными. Нет никакого общего правила относительно наилучшего числа k — все зависит главным образом от природы данных. Для высокоструктурированных данных с малым шумом меньшие числа k работают лучше всего. Заимствуя термин у сообщества обработки сигналов, этот тип данных иногда называют данными с высоким *отношением "сигнал/шум"* (signal-to-noise ratio, SNR). Примерами данных с высоким отношением "сигнал/шум", как правило, являются данные для распознавания почерка и речи. Для шумных данных с меньшей структурированностью (данных с низким отношением "сигнал/шум"), таких как данные о ссудах, уместными являются более крупные числа k . В типичной ситуации числа k попадают в интервал между 1 и 20. Нередко выбирается нечетное число, чтобы избежать равенства голосов при голосовании.



Компромисс между смещением и дисперсией

Напряжение между сверхсглаживанием и перепогонкой является *экземпляром компромисса между смещением и дисперсией*, повсеместно распространенной проблемы в подгонке статистических моделей. Дисперсия обозначает ошибку моделирования, которая происходит из-за выбора тренировочных данных; т. е. если бы вам пришлось выбрать другой набор тренировочных данных, то результирующая модель была бы другой. Смещение обозначает ошибку моделирования, которая происходит, потому что вы должным образом не идентифицировали опорный реально существующий сценарий; эта ошибка не исчезнет, если просто добавить больше тренировочных данных. Когда гибкая модель перепогоднена, то дисперсия увеличивается. Вы можете уменьшить гибкость при помощи более простой модели, но смещение может увеличиться из-за потери гибкости в моделировании опорной реально существующей ситуации. Общий подход к решению этого компромисса лежит через перекрестный контроль. Дополнительную информацию см. в разделе "Перекрестный контроль" главы 4.

k ближайших соседей как механизм порождения признаков

Классификатор на основе k ближайших соседей стал популярным благодаря простоты и интуитивно понятной природе. С точки зрения результативности классификатор на основе k ближайших соседей сам по себе обычно не конкурентоспособен по сравнению с более изощренными классификационными приемами. При подгонке модели в практических условиях, однако, классификатор на основе k ближайших соседей может использоваться для добавления "локального знания" в многоэтапном процессе с другими классификационными приемами:

1. Классификатор на основе k ближайших соседей выполняется на данных, и для каждой записи выводится классификация (либо квазивероятность класса).
2. Этот результат добавляется в качестве нового признака в запись, и затем еще один классификационный метод выполняется на данных. Исходные предсказательные переменные, следовательно, используются дважды.

Поначалу у вас может возникнуть вопрос, не вызывает ли этот процесс проблему, связанную с мультиколлинеарностью, ввиду того, что некоторые предсказатели используются им дважды (см. раздел "Мультиколлинеарность" главы 4). Это не проблема, поскольку информация, инкорпорируемая в модель второго этапа, высококачественна, получена только из нескольких соседних записей, а следовательно, служит дополнительной информацией и не является избыточной.



Вы можете думать о таком поэтапном классификаторе на основе k ближайших соседей, как о форме ансамблевого обучения, в котором многочисленные предсказывающие методы моделирования используются в сопряжении друг с другом. Его также можно рассматривать как форму выработки признаков, где цель состоит в том, чтобы вывести признаки (предсказательные переменные), которые имеют предсказательную силу. Нередко это сопряжено с некоторым ручным анализом данных; классификатор на основе k ближайших соседей дает вполне автоматический путь достижения этого.

Например, рассмотрим данные о жилом фонде округа Кинг. При установлении продажной цены на дом агент по продаже недвижимости будет основывать цену на похожих домах, которые были недавно проданы, так называемых продажах-аналогах. В сущности, агенты по продаже недвижимости выполняют ручную версию классификатора на основе k ближайших соседей: глядя на продажные цены похожих домов, они могут прикинуть цену, по которой дом будет продан. Мы можем создать новый признак для статистической модели, чтобы симитировать профессионала в области торговли недвижимостью путем применения классификатора на основе k ближайших соседей к недавним продажам. Предсказываемым значением является продажная цена, и существующие предсказательные переменные могут включать местоположение, общую площадь в квадратных футах, тип строения, размер земельного участка и число спален и ванных комнат. Новая предсказательная переменная (признак), которую мы добавляем посредством классификатора на основе k ближайших соседей, — это предсказатель классификатора для каждой записи (аналогичной продажам-аналогам у агентов по продаже недвижимости). Поскольку предсказываемое значение является числовым, вместо мажоритарного голосования используется среднее k ближайших соседей (так называемая регрессия на основе k ближайших соседей, или *KNN-регрессия*).

Схожим образом, для данных о ссудах мы можем создать признаки, которые представляют разные аспекты ссудосберегательного процесса. Например, следующий фрагмент кода на R строит признак, который представляет кредитоспособность заемщика:

```
borrow_df <- model.matrix(~ -1 + dti + revol_bal + revol_util + open_acc +
                           delinq_2yrs_zero + pub_rec_zero, data=loan_data)
borrow_knn <- knn(borrow_df, test=borrow_df, cl=loan_data[, 'outcome'],
                  prob=TRUE, k=20)
prob <- attr(borrow_knn, "prob")
borrow_feature <- ifelse(borrow_knn == 'default', prob, 1 - prob)
summary(borrow_feature)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000  0.400   0.500   0.501  0.600   0.950
```

С помощью пакета `scikit-learn` мы используем метод `predict_proba` натренированной модели для получения вероятностей:

```
predictors = ['dti', 'revol_bal', 'revol_util', 'open_acc',
              'delinq_2yrs_zero', 'pub_rec_zero']
outcome = 'outcome'

X = loan_data[predictors]
y = loan_data[outcome]

knn = KNeighborsClassifier(n_neighbors=20)
knn.fit(X, y)

loan_data['borrower_score'] = knn.predict_proba(X)[:, 1]
loan_data['borrower_score'].describe()
```

Результатом является признак, который предсказывает правдоподобие того, что заемщик не выплатит ссуду, опираясь на его кредитную историю.

Ключевые идеи для k ближайших соседей

- k ближайших соседей (KNN) классифицирует запись путем ее отнесения к классу, к которому принадлежат похожие записи.
- Схожесть (расстояние) определяется евклидовым расстоянием либо другими робственными метриками.
- Число ближайших соседей k , с которыми запись сравнивается, определяется тем, насколько хорошую результативность алгоритм показывает на тренировочных данных с использованием разных значений k .
- В типичной ситуации предсказательные переменные стандартизируются с той целью, чтобы переменные с большой шкалой не доминировали над метриками расстояния.
- В предсказательном моделировании k ближайших соседей часто используется на первом этапе, и предсказанное значение добавляется в данные в качестве предсказателя для моделирования на втором (не-KNN) этапе.

Древесные модели

Древесные модели, так называемые *классификационные и регрессионные деревья* (classification and regression trees, CART)⁵, деревья решений, или просто деревья, — это эффективный и популярный классификационный (и регрессионный) метод, первоначально разработанный в 1984 году. Лео Брейманом (Leo Breiman) и другими. Древесные модели и их более мощные потомки *случайные леса* и *бустинг* (см. раздел "Бэггинг и случайный лес" и "Бустинг" главы 6) формируют фундамент для наиболее широко используемых и мощных инструментов предсказательного моделирования в науке о данных для регрессии и классификации.

Ключевые термины для деревьев

Рекурсивное подразделение (recursive partitioning)

Многократное разделение и подразделение данных с целью получения максимально однородных исходов в каждом финальном подразделе.

Значение в точке ветвления (split value)

Значение предсказателя, которое делит записи на те, где этот предсказатель меньше значения в точке ветвления, и те, и где он больше.

⁵ Термин CART является зарегистрированной торговой маркой компании Salford Systems, которая связана с их конкретной имплементацией древесных моделей.

Узел (node)

В дереве решений или в наборе соответствующих правил ветвления узел — это представление значения в точке ветвления в виде графа либо в виде правила.

Лист (leaf)

Конец набора правил в форме "если, то", или ветвлений дерева, — правил, которые приводят вас к этому листу, обеспечивают одно из правил классификации для любой записи в дереве.

Потеря (loss)

Число неправильных результатов классификации на этапе в процессе ветвления; чем больше потеря, тем больше загрязненность.

Загрязненность (impurity)

Степень, с которой в подразделе данных обнаруживается смесь классов (чем больше смешанность, тем больше загрязненность).

Синонимы: гетерогенность, разнородность, нечистота.

Антонимы: однородность, гомогенность, чистота.

Подрезание (pruning)

Процесс поступательного подрезания ветвей полностью выращенного дерева с целью снижения переподгонки.

Древесная модель — это набор правил формы "если-то-иначе", которые легко понять и имплементировать. В отличие от линейной и логистической регрессии, деревья имеют способность обнаруживать скрытые регулярности (шаблоны, паттерны), соответствующие сложным взаимодействиям в данных. Вместе с тем, в отличие от классификатора на основе k ближайших соседей либо наивного байесова классификатора, простые древесные модели могут быть выражены с точки зрения связей между предсказателями, и эти связи легко поддаются интерпретации.



Деревья решений в исследовании операций

Термин "*деревья решений*" имеет другой (и более старый) смысл в теории принятия решений и исследовании операций, где он обозначает процесс анализа решений человеком. В этом смысле точки принятия решения, возможные исходы и их оценочные вероятности размещаются на диаграмме ветвления, и выбирается путь принятия решений с максимальным ожидаемым значением (т. е. математическим ожиданием).

Простой пример

Двумя главными пакетами для подгонки древесных моделей в R являются `rpart` и `tree`. С помощью пакета `rpart` выполняется подгонка модели к выборке, состоящей из 3 тыс. записей данных о ссудах, используя переменные `payment_inc_ratio` и `borrower_score` (описание данных см. в разделе "*k* ближайших соседей" ранее в этой главе).

```
library(rpart)
loan_tree <- rpart(outcome ~ borrower_score + payment_inc_ratio,
                  data=loan3000, control=rpart.control(cp=0.005))
plot(loan_tree, uniform=TRUE, margin=0.05)
text(loan_tree)
```

Класс `sklearn.tree.DecisionTreeClassifier` обеспечивает имплементацию дерева решений. Пакет `dmdba` предоставляет удобную функцию для создания визуализации внутри блокнота `Jupyter`:

```
predictors = ['borrower_score', 'payment_inc_ratio']
outcome = 'outcome'

X = loan3000[predictors]
y = loan3000[outcome]

loan_tree = DecisionTreeClassifier(random_state=1, criterion='entropy',
                                   min_impurity_decrease=0.003)

loan_tree.fit(X, y)
plotDecisionTree(loan_tree, feature_names=predictors,
                 class_names=loan_tree.classes_)
```

Результирующее дерево показано на рис. 6.3. Вследствие разных имплементаций вы обнаружите, что результаты в R и Python не являются идентичными; это ожидаемо. Указанные правила классификации определяются путем обхода иерархического дерева, начиная в корне, до тех пор, пока не будет достигнут лист.

Обычно дерево изображается перевернутым, чтобы корень находился вверху, а листья — внизу. Например, если мы получаем ссуду с баллом заемщика `borrower_score`, равным 0,6, и отношением платежей к доходу `payment_inc_ratio`, равным 8,0, то в итоге мы приходим к крайнему левому листу и предсказываем, что ссуда будет погашена.

Структурно распечатанная версия дерева также легко строится в R:

```
loan_tree
n= 3000

node), split, n, loss, yval, (yprob)
  * denotes terminal node

1) root 3000 1445 paid off (0.5183333 0.4816667)
  2) borrower_score>=0.575 878 261 paid off (0.7027335 0.2972665) *
  3) borrower_score< 0.575 2122 938 default (0.4420358 0.5579642)
    6) borrower_score>=0.375 1639 802 default (0.4893228 0.5106772)
      12) payment_inc_ratio< 10.42265 1157 547 paid off (0.5272256 0.4727744)
        24) payment_inc_ratio< 4.42601 334 139 paid off (0.5838323 0.4161677) *
        25) payment_inc_ratio>=4.42601 823 408 paid off (0.5042527 0.4957473)
          50) borrower_score>=0.475 418 190 paid off (0.5454545 0.4545455) *
          51) borrower_score< 0.475 405 187 default (0.4617284 0.5382716) *
```



```
13) payment_inc_ratio>=10.42265 482 192 default (0.3983402 0.6016598) *
7) borrower_score< 0.375 483 136 default (0.2815735 0.7184265) *
```

Глубина дерева показана с помощью отступов. Каждый узел соответствует предварительной классификации, определяемой преобладающим исходом в данном разделе. "Потеря" — это число неправильных результатов классификации, производимое предварительной классификацией в разделе. Например, в узле 2 был 261 неправильный результат классификации из общего числа 878 записей. Значения в круглых скобках соответствуют доле записей, которые были погашены и не были выплачены. Например, в узле 13, который предсказывает невыплату, более 60% записей — это невыплаченные ссуды.

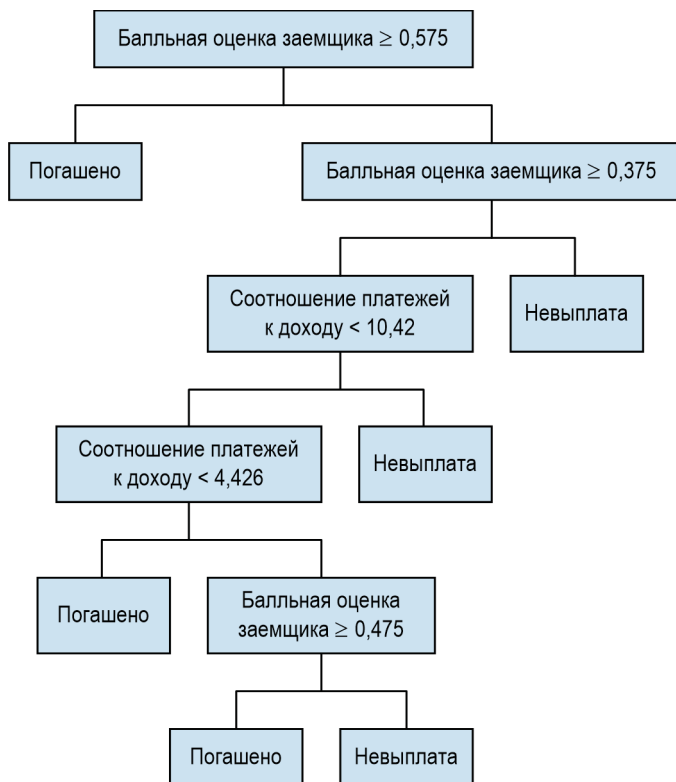


Рис. 6.3. Правила для простой древесной модели, подогнанной к данным о ссудах

Документация пакета `scikit-learn` описывает процесс создания текстового представления модели на основе дерева решений. Мы включили удобную функцию в наш пакет `dmdba`:

```
print(textDecisionTree(loan_tree))
--
node=0 test node: go to node 1 if 0 <= 0.5750000178813934 else to node 6
node=1 test node: go to node 2 if 0 <= 0.32500000298023224 else to node 3
node=2 leaf node: [[0.785, 0.215]]
```

```

node=3 test node: go to node 4 if 1 <= 10.42264986038208 else to node 5
node=4 leaf node: [[0.488, 0.512]]
node=5 leaf node: [[0.613, 0.387]]
node=6 test node: go to node 7 if 1 <= 9.19082498550415 else to node 10
node=7 test node: go to node 8 if 0 <= 0.7249999940395355 else to node 9
node=8 leaf node: [[0.247, 0.753]]
node=9 leaf node: [[0.073, 0.927]]
node=10 leaf node: [[0.457, 0.543]]

```

Алгоритм рекурсивного подразделения

Алгоритм конструирования дерева решений, именуемый *рекурсивным подразделением*, достаточно прямолинеен и интуитивно понятен. Данные многократно подразделяются с использованием предсказательных значений, которые делают все возможное, чтобы разложить данные на относительно однородные сегменты. На рис. 6.4 показаны разделы, созданные для дерева на рис. 6.3. Первое правило, помеченное на графике меткой 1, таково: $\text{borrower_score} \geq 0.525$. Второе правило, $\text{payment_inc_ratio} < 9.732$, сегментирует левую долю.

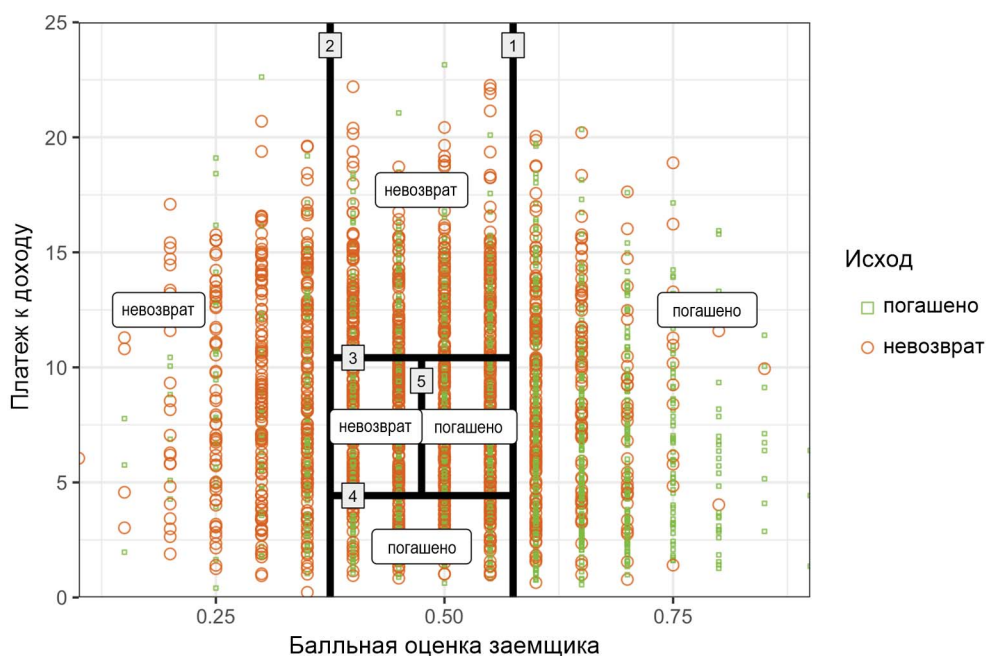


Рис. 6.4. Первые три правила для простой древесной модели, подогнанной к данным о ссудах

Предположим, что у нас есть переменная Y отклика и множество из P предсказательных переменных X_j для $j = 1, \dots, P$. Для раздела A записей рекурсивное подразделение отыщет наилучший способ разделить A на два подраздела:

1. Для каждой предсказательной переменной X_j :
 - для каждого значения s_j из X_j :
 - разветвить записи в A со значениями $X_j < s_j$ как один раздел и оставшиеся записи, где $X_j \geq s_j$ — как еще один раздел;
 - измерить однородность классов внутри каждого подраздела раздела A ;
 - выбрать значение s_j , которое порождает максимальную однородность класса внутри подраздела.
2. Выбрать переменную X_j и значение точки ветвления s_j , которое порождает максимальную однородность класса внутри подраздела.

Теперь наступает очередь рекурсивной части:

1. Инициализировать A всем набором данных.
2. Применить алгоритм сегментирования, чтобы разбить A на два подсегмента, A_1 и A_2 .
3. Повторить шаг 2 на подсегментах A_1 и A_2 .
4. Алгоритм завершается, когда невозможно создать никакой дальнейший сегмент, который в достаточной мере улучшает однородность сегментов.

Конечным результатом является подразделение данных, как на рис. 6.4, исключая данные в P -размерностях, причем каждый сегмент предсказывает результат 0 либо 1 в зависимости от мажоритарного голосования отклика в этом разделе.



В дополнение к двоичному предсказанию в форме 0/1, древесные модели могут производить оценку вероятности, опираясь на число нулей и единиц в разделе. Оценкой является простая сумма нулей или единиц в разделе, деленная на число наблюдений в подразделе.

$$\text{Prob}(Y = 1) = \frac{\text{число единиц в разделе}}{\text{размер раздела}}.$$

Затем оценочная вероятность $\text{Prob}(Y = 1)$ может быть конвертирована в двоичное решение; например, мы установим оценку равной 1, если $\text{Prob}(Y = 1) > 0,5$.

Измерение однородности или загрязненности

Древесные модели рекурсивно создают разделы (наборы записей) A , которые предсказывают исход $Y = 0$ или $Y = 1$. Из приведенного выше алгоритма вы видите, что нам нужен способ измерить однородность, так называемую чистоту класса, внутри раздела. Или, что то же самое, нам нужно измерить загрязненность раздела. Точность предсказаний — это доля p неправильно классифицированных записей внутри этого раздела, которая варьируется от 0 (идеально) до 0,5 (чисто случайное угадывание).

Как оказалось, точность не является хорошей мерой загрязненности. Вместо нее общеприняты две другие меры загрязненности — *загрязненность Джини* и *энтропия информации*. В то время как эти (и другие) меры загрязненности применяются к классификационным задачам с более чем двумя классами, мы сосредоточимся на двоичном случае. Загрязненность Джини для набора записей A такова:

$$I(A) = p(1 - p).$$

Энтропийная мера задается следующей ниже формулой:

$$I(A) = -p \log_2(p) - (1 - p) \log_2(1 - p).$$

На рис. 6.5 показано, что меры (перешкалированной) загрязненности Джини и энтропии похожи, при этом энтропия дает более высокие баллы загрязненности для умеренных и высоких уровней точности.

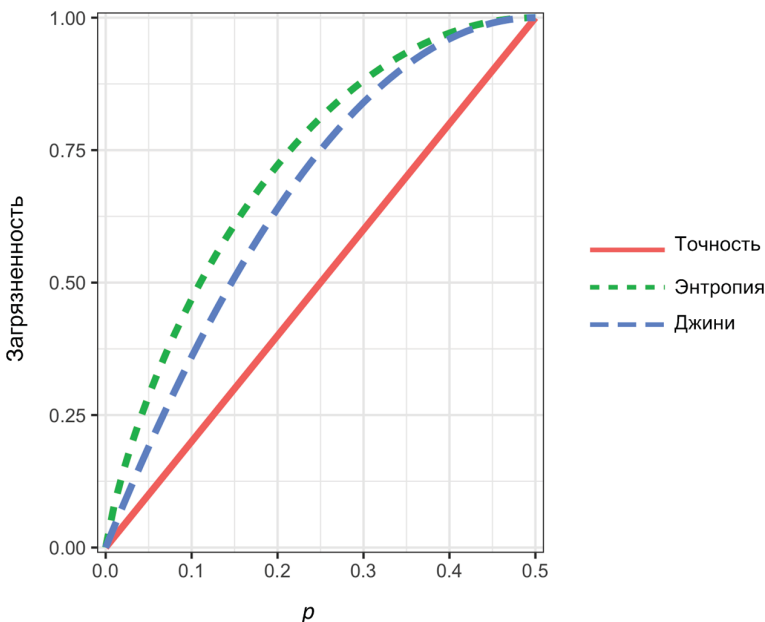


Рис. 6.5. Меры загрязненности Джини и энтропии



Коэффициент Джини

Загрязненность Джини не следует путать с *коэффициентом Джини*. Они представляют схожие понятия, но загрязненность Джини ограничена двоичной классификационной задачей и связана с метрикой AUC (см. раздел "Площадь под ROC-кривой" главы 5).

Метрика загрязненности используется в описанном ранее алгоритме подразделения. По каждому предлагаемому разделу данных загрязненность измеряется для всех разделов, которые получаются в результате ветвления. Затем вычисляется средневзвешенное значение, и (на каждом этапе) выбирается любой из указанных разделов, который выдает самое низкое средневзвешенное значение.

Остановка выращивания дерева

По мере того, как дерево становится все больше, правила ветвления становятся более детализированными, и дерево постепенно смещается от распознавания "больших" правил, идентифицирующих реальные и надежные связи в данных в сторону "крошечных" правил, которые отражают только шум. Полностью выращенное дерево имеет абсолютно чистые листья и, следовательно, 100%-ную точность в классифицировании данных, на которых оно натренировано. Эта точность, разумеется, является иллюзорной — мы выполнили чрезмерно плотную подгонку (см. *врезку "Компромисс между смещением и дисперсией" ранее в этой главе*) к данным, приспособившись к шуму в тренировочных данных, а не тому сигналу, который мы хотим идентифицировать в новых данных.

Нам нужен какой-то способ определить, когда следует прекратить выращивание дерева на этапе, который будет обобщать на новые данные. В языках R и Python имеются самые разные способы прекращения ветвления.

- ◆ Не допускать ветвление раздела, если результирующий подраздел либо терминальный лист являются слишком малыми. В пакете `rpart` эти ограничения отдельно контролируются соответственно параметрами `minsplit` и `minbucket` со значениями 20 и 7, принятыми по умолчанию. В классе `DecisionTreeClassifier` языка Python мы можем контролировать за этим процессом с помощью параметров `min_samples_split` (по умолчанию 2) и `min_samples_leaf` (по умолчанию 1).
- ◆ Не ветвить раздел, если новый раздел "значительно" не уменьшает загрязненность. В пакете `rpart` это контролируется *параметром сложности* `cp`, т. е. мерой того, насколько сложным дерево является, — чем оно сложнее, тем больше значение `cp`. На практике `cp` используется для ограничения роста дерева путем наложения штрафа на дополнительную сложность (дополнительные ветвления) в дереве. Класс `DecisionTreeClassifier` (Python) имеет параметр `min_impurity_decrease`, который ограничивает ветвление, основываясь на взвешенном значении уменьшения загрязненности. Здесь меньшие значения приведут к более сложным деревьям.

Эти методы задействуют произвольные правила и могут быть полезны для разведывательной работы, но мы не можем легко определить оптимальные значения (т. е. значения, которые максимизируют предсказательную точность с новыми данными). Нам нужно сочетать перекрестный контроль с систематическим изменением модельных параметров либо с модифицированием дерева путем его подрезания.

Контроль за сложностью дерева в R

С помощью параметра сложности `cp` мы можем оценить, дерево какого размера будет работать лучше всего с новыми данными. Если `cp` является слишком малым, то дерево будет прилегать к данным слишком плотно (переподогнанно к данным), являясь подогнанным к шуму, а не сигналу. С другой стороны, если `cp` является слишком крупным, то дерево окажется слишком малым и будет иметь малую предсказательную силу. По умолчанию в пакете `rpart` принято значение 0,01, хотя для

более крупных наборов данных это значение, скорее всего, будет слишком большим. В предыдущем примере значение `ср` было установлено равным 0,005, т. к. значение по умолчанию привело к дереву с одним-единственным ветвлением. В разведывательном анализе достаточно просто попробовать несколько значений.

Определение оптимального параметра `ср` является примером компромисса между смещением и дисперсией. Наиболее часто встречающийся способ оценить хорошее значение параметра `ср` предусматривает перекрестный контроль (см. раздел "Перекрестный контроль" главы 4):

1. Подразделить данные на тренировочный и контрольный (отложенный) наборы.
2. Вырастить дерево с помощью тренировочных данных.
3. Выполнить его подрезание поочередно, шаг за шагом, на каждом шаге записывая `ср` (используя *тренировочные данные*).
4. Отметить `ср`, который соответствует минимальной ошибке (потере) на *контрольных данных*.
5. Повторно подразделить данные на тренировочный и контрольный наборы и повторить процесс выращивания дерева, подрезания ветвей и записи параметра `ср`.
6. Выполнять этот процесс снова и снова и усреднять параметры `ср`, которые отражают минимальную ошибку для каждого дерева.
7. Вернуться к исходным данным либо будущим данным и вырастить дерево, остановившись на этом оптимальном значении параметра `ср`.

В пакете `rpart` можно использовать аргумент `cptable` для порождения таблицы значений `ср` и связанной с ними ошибки перекрестного контроля (`xerror` в R), из которой вы можете определить значение `ср`, имеющее самую низкую ошибку перекрестного контроля.

Контроль за сложностью дерева в Python

Ни параметр сложности, ни подрезание не доступны в имплементации дерева решений в пакете `scikit-learn`. Решение состоит в том, чтобы использовать решеточный поиск над комбинациями разных значений параметров. Например, мы можем варьировать `max_depth` в интервале от 5 до 30 и `min_samples_split` от 20 до 100. Класс `GridSearchCV` в пакете `scikit-learn` является удобным способом объединить исчерпывающий поиск по всем комбинациям с перекрестным контролем. Затем отбирают оптимальный набор параметров, используя результативность перекрестно контролируемой модели.

Предсказывание непрерывного значения

Предсказывание непрерывного значения (которое также именуется *регрессией*) с помощью дерева следует той же самой логике и процедуре, за исключением того, что загрязненность измеряется квадратичными отклонениями от среднего (квадратическими ошибками) в каждом подразделе и предсказательная результативность

оценивается квадратным корнем из среднеквадратической ошибки (RMSE) (см. раздел "Оценивание результативности модели" главы 4) в каждом разделе.

Пакет `scikit-learn` имеет класс `sklearn.tree.DecisionTreeRegressor` для тренировки регрессионной модели на основе дерева решений.

Каким образом используются деревья

Одно из самых больших препятствий, с которыми разработчики моделей сталкиваются в организациях, — это "чернозашитная" природа методов, которые они используют, что дает основания для оппозиции со стороны других элементов организации. В этом отношении древесная модель имеет два привлекательных аспекта.

- ◆ Древесные модели обеспечивают визуальный инструмент для обследования данных, для получения представления о том, какие переменные важны и как они связаны друг с другом. Деревья могут улавливать нелинейные связи среди предсказательных переменных.
- ◆ Древесные модели обеспечивают набор правил, которые могут быть эффективным образом переданы неспециалистам для имплементации либо для "продажи" проекта по добыче регулярностей из данных.

Однако, когда речь идет о предсказании, сопряжение результатов из многочисленных деревьев в типичной ситуации оказывается эффективнее, чем использование всего одного дерева. В частности, алгоритмы случайного леса и бустированных деревьев почти всегда обеспечивают превосходящую предсказательную точность и результативность (см. раздел "Бэггинг и случайный лес" и "Бустинг" главы 6), но вышеупомянутые преимущества для одиночного дерева пропадают.

Ключевые идеи для древесных моделей

- Деревья решений порождают набор правил классификации или предсказывают исход.
- Правила соответствуют поочередному подразделению данных на подразделы.
- Каждый раздел, или точка ветвления, соотнесен с конкретным значением предсказательной переменной и делит данные на записи, где значение этого предсказателя находится выше или ниже значения в точке ветвления.
- На каждом этапе древесный алгоритм выбирает точку ветвления, которая минимизирует загрязненность исхода в каждом подразделе.
- Когда никакие дальнейшие ветвления сделать невозможно, дерево считается полностью выращенным, и каждый терминальный узел, или лист, имеет записи с единственным классом; новым случаям, которые следуют этим путем правил (ветвления), назначается этот класс.
- Полностью выращенное дерево слишком плотно прилегает к данным (переподогнано к данным) и должно быть подрезано, чтобы оно получало сигнал вместо шума.

- Мультидревесные алгоритмы, такие как случайные леса и бустированные деревья, производят более высокую предсказательную результативность, но теряют в коммуникативной способности одиночных деревьев, основанной на правилах.

Дополнительные материалы для чтения

- ♦ "Древесные алгоритмы: полное учебное руководство по моделированию на основе деревьев с нуля на Python и R"⁶ (Tree Based Algorithms: A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python) // 2016. — April 12) от группы контента портала Analytics Vidya.
- ♦ "Введение в рекурсивное подразделение с использованием подпрограмм RPART"⁷ (Therneau T. M., Atkinson E. J. An Introduction to Recursive Partitioning Using the RPART Routines // Mayo Foundation. — 2015. — June 29).

Бэггинг и случайный лес

В 1907 году статистик сэр Фрэнсис Гальтон (Sir Francis Galton) посещал окружную ярмарку в Англии, в которой проводился конкурс по угадыванию убойного веса вола, демонстрировавшегося на выставке. Поступило 800 заявок, и, хотя отдельные догадки значительно варьировались, среднее и медиана вышли в пределах 1% от истинного веса вола. Джеймс Суrowецки (James Surowiecki) исследовал этот феномен в своей книге "Мудрость толпы" (The Wisdom of Crowds. — Doubleday, 2004). Указанный принцип также применяется к предсказательным моделям: усреднение (или взятие большинства голосов) многочисленных моделей — *ансамбля моделей* — оказывается точнее, чем выбор всего одной модели.

Ключевые термины для бэггинга и случайного леса

Ансамбль (ensemble)

Формирование предсказания при помощи набора моделей.

Синоним: модельное усреднение.

Бэггинг (bagging)

Общая методика формирования коллекции моделей путем бутстрапирования данных.

Синоним: агрегирование бутстраповских выборок, бутстрап-агрегирование, пакетное усреднение.

⁶ См. <https://oreil.ly/zOr4B>.

⁷ См. <https://oreil.ly/6rLGk>.

Случайный лес (random forest)

Тип бутстрап-агрегированной оценки, опирающейся на модели на основе деревьев решений.

Синоним: бутстрап-агрегированные деревья решений.

Важность переменной (variable importance)

Метрика важности предсказательной переменной в результативности модели.

Ансамблевый подход применяется в многочисленных и разнообразных методах моделирования; наиболее наглядно это выразилось в конкурсе Netflix Contest, в котором компания Netflix предложила приз в 1 000 000 \$ любому конкурсанту, который придумает модель, дающую 10%-ное улучшение в предсказании рейтинга, которым клиент компании Netflix наградит кинофильм. Простая версия ансамблей имеет следующий вид:

1. Разработать предсказательную модель и записать предсказания для заданного набора данных.
2. Повторить для многочисленных моделей на тех же данных.
3. Для каждой предсказываемой записи взять среднее (либо средневзвешенное) предсказаний либо выбрать из них мажоритарным голосованием.

Ансамблевые методы наиболее системно и эффективно применяются к деревьям решений. Ансамблевые древесные модели настолько мощны, что они обеспечивают способ строительства хороших предсказательных моделей с относительно небольшими усилиями.

За рамками простого ансамблевого алгоритма имеются два главных варианта ансамблевых моделей: *бэггинг* и *бустинг*. Если иметь в виду ансамблевые древесные модели, то они называются моделями на основе *случайного леса* и *бустированными древесными моделями*. Настоящий раздел посвящен бэггингу; бустинг рассматривается в одноименном разделе *далее в этой главе*.

Бэггинг

Бэггинг (bagging, как сокращение от bootstrap aggregating — агрегирование бутстраповских выборок, или пакетное усреднение) был представлен Лео Брейманом в 1994 году⁸. Предположим, что мы имеем отклик Y и P предсказательных переменных $\mathbf{X} = X_1, X_2, \dots, X_p$ с N записями.

⁸ Термин bagging — это своего рода технический каламбур, который, с одной стороны, является аббревиатурой термина bootstrap aggregating (агрегирование бутстраповских выборок, а с другой — означает упаковывание в пакеты, поскольку, в сущности, характер работы алгоритма бэггинга в этом и состоит — он отбирает бутстраповские пакеты данных и затем их агрегирует, таким образом строя разные модели параллельно, а затем выбирая лучшую. — *Прим. перев.*

Бэггинг сродни базовому алгоритму для ансамблей с одним исключением, что вместо подгонки различных моделей к одинаковым данным каждая новая модель подгоняется к повторно отбираемой бутстраповской выборке. Вот этот алгоритм, представленный более формально:

1. Инициализировать M , число моделей, подлежащих подгонке, и n , число записей, из которых делается выборка ($n < N$). Установить итерацию равной $m = 1$.
2. Взять из тренировочных данных бутстраповскую повторную выборку (т. е. с возвратом) из n записей, чтобы сформировать подвыборку Y_m и X_m (пакет).
3. Натренировать модель, используя Y_m и X_m , чтобы создать набор правил принятия решений $\hat{f}_m(\mathbf{X})$.
4. Прирастить счетчик моделей $m = m + 1$. Если $m \leq M$, то перейти к шагу 2.

В случае, где \hat{f}_m предсказывает вероятность $Y = 1$, бутстрап-агрегированная оценка задается следующей ниже формулой:

$$\hat{f} = \frac{1}{M} (\hat{f}_1(\mathbf{X}) + \hat{f}_2(\mathbf{X}) + \dots + \hat{f}_M(\mathbf{X})).$$

Случайный лес

Случайный лес основывается на применении бэггинга к деревьям решений с одним важным расширением: в дополнение к отбору записей алгоритм также отбирает переменные⁹. В традиционных деревьях решений, чтобы определить, как создать подраздел раздела A , алгоритм делает выбор переменной и точки ветвления путем минимизации критерия, в частности, загрязненности Джини (см. врезку "Измерение однородности или загрязненности" ранее в этой главе). В ситуации случайных лесов на каждом этапе алгоритма выбор переменной ограничен *случайным подмножеством переменных*. По сравнению с базовым древесным алгоритмом (см. раздел "Алгоритм рекурсивного подразделения" ранее в этой главе) алгоритм случайного леса добавляет еще два шага: обсуждавшийся ранее бэггинг (см. раздел "Бэггинг и случайный лес" ранее в этой главе) и отбор бутстраповских выборок переменных в каждой точке ветвления:

1. Взять из записей бутстраповскую *подвыборку* (с возвратом).
2. Для первого ветвления случайно отобрать $p < P$ переменных без возврата.
3. Для каждой отобранной переменной $X_{j(1)}, X_{j(2)}, \dots, X_{j(p)}$ применить алгоритм ветвления:

⁹ Термин "случайный лес" является товарным знаком Лео Бреймана и Адель Катлер с лицензией, выданной компанией Salford Systems. Стандартное наименование нетоварного знака отсутствует, и термин "случайный лес" является таким же синонимом алгоритма, как и "Клинекс" для косметических салфеток.

- для каждого значения $s_{j(k)}$ из $X_{j(k)}$:
 - разбить записи в разделе A , где $X_{j(k)} < s_{j(k)}$ выступает как один раздел, и остальные записи, где $X_{j(k)} \geq s_{j(k)}$ — как другой раздел;
 - измерить однородность классов внутри каждого подраздела A ;
 - выбрать значение $s_{j(k)}$, которое порождает максимальную однородность класса внутри раздела.
4. Выбрать переменную $X_{j(k)}$ и значение точки ветвления $s_{j(k)}$, которое порождает максимальную однородность класса внутри раздела.
 5. Перейти к следующему ветвлению и повторить предыдущие шаги, начиная с шага 2.
 6. Продолжить дополнительные ветвления, следуя той же процедуре до тех пор, пока дерево не будет выращено.
 7. Вернуться к шагу 1, взять еще одну бутстраповскую подвыборку и начать процесс заново.

Сколько переменных следует отбирать на каждом шаге? Эмпирическое правило состоит в выборе \sqrt{P} , где P — это число предсказательных переменных. Пакет `randomForest` имплементирует случайный лес в R. Следующий фрагмент кода применяет этот пакет к данным о ссудах (описания данных см. в разделе "к ближайших соседей" ранее в этой главе).

```
rf <- randomForest(outcome ~ borrower_score + payment_inc_ratio,
  data=loan3000)

rf
```

Call:

```
randomForest(formula = outcome ~ borrower_score + payment_inc_ratio,
  data = loan3000)
  Type of random forest: classification
    Number of trees: 500
```

No. of variables tried at each split: 1

OOB estimate of error rate: 39.17%

Confusion matrix:

	default	paid off	class.error
default	873	572	0.39584775
paid off	603	952	0.38778135

В Python мы используем класс `sklearn.ensemble.RandomForestClassifier`:

```
predictors = ['borrower_score', 'payment_inc_ratio']
outcome = 'outcome'
```

```
X = loan3000[predictors]
y = loan3000[outcome]
```

```
rf = RandomForestClassifier(n_estimators=500, random_state=1, oob_score=True)
rf.fit(X, y)
```

По умолчанию выполняется тренировка 500 деревьев. Поскольку в наборе предсказателей имеются всего две переменные, алгоритм случайно отбирает переменную, по которой можно выполнить ветвление на каждом этапе (т. е. бутстраповскую подвыборку размера 1).

Внепакетная оценка (out-of-bag, OOB) ошибки — это частота появления ошибки для натренированных моделей, применяемая к данным, отложенным в сторону из тренировочного набора для этого дерева. В языке R, используя результаты, получаемые на выходе из модели, внепакетная ошибка может быть выведена на график в сопоставлении с числом деревьев в случайном лесе:

```
error_df = data.frame(error_rate=rf$err.rate[, 'OOB'], num_trees=1:rf$ntree)
ggplot(error_df, aes(x=num_trees, y=error_rate)) + geom_line()
```

Имплементация класса `RandomForestClassifier` не имеет простого способа получить внепакетные оценки как функции числа деревьев в случайном лесу. Мы можем натренировать последовательность классификаторов с увеличивающимся числом деревьев и отслеживать значения балла `oob_score_`. Однако этот метод не является эффективным:

```
n_estimator = list(range(20, 510, 5))
oobScores = []
for n in n_estimator:
    rf = RandomForestClassifier(n_estimators=n, criterion='entropy',
                               max_depth=5, random_state=1, oob_score=True)
    rf.fit(X, y)
    oobScores.append(rf.oob_score_)
df = pd.DataFrame({ 'n': n_estimator, 'oobScore': oobScores })
df.plot(x='n', y='oobScore')
```

Результат показан на рис. 6.6. Частота появления ошибок быстро уменьшается примерно с 0,44 перед тем, как стабилизироваться на уровне 0,385. Предсказанные значения могут быть получены из функции `predict` и выведены на график в R следующим образом:

```
pred <- predict(rf, prob=TRUE)
rf_df <- cbind(loan3000, pred = pred)
ggplot(data=rf_df, aes(x=borrower_score, y=payment_inc_ratio,
                      shape=pred, color=pred, size=pred)) +
  geom_point(alpha=.8) +
  scale_color_manual(values = c('paid off'='#b8e186', 'default'='#d95f02')) +
  scale_shape_manual(values = c('paid off'=0, 'default'=1)) +
  scale_size_manual(values = c('paid off'=0.5, 'default'=2))
```

В Python мы можем создать похожий график следующим образом:

```
predictions = X.copy()
predictions['prediction'] = rf.predict(X)
predictions.head()
```

```
fig, ax = plt.subplots(figsize=(4, 4))

predictions.loc[predictions.prediction=='paid off'].plot(
    x='borrower_score', y='payment_inc_ratio', style='.',
    markerfacecolor='none', markeredgcolor='C1', ax=ax)
predictions.loc[predictions.prediction=='default'].plot(
    x='borrower_score', y='payment_inc_ratio', style='o',
    markerfacecolor='none', markeredgcolor='C0', ax=ax)
ax.legend(['paid off', 'default']);
ax.set_xlim(0, 1)
ax.set_ylim(0, 25)
ax.set_xlabel('borrower_score')
ax.set_ylabel('payment_inc_ratio')
```

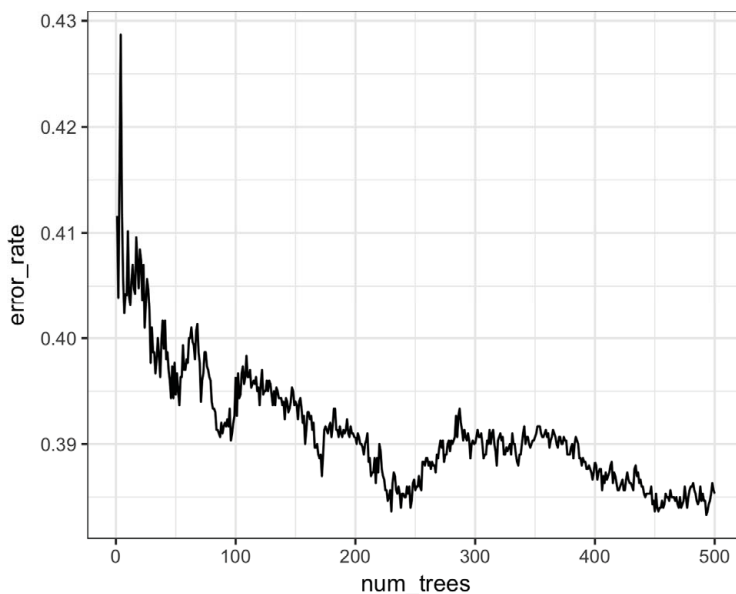


Рис. 6.6. Пример улучшения точности случайного леса с добавлением большего числа деревьев

График, приведенный на рис. 6.7, весьма показателен в отношении природы случайного леса.

Метод случайного леса — это "чернощичный" метод. Он производит более точные предсказания, чем простое дерево, но интуитивно понятные правила решения простого дерева теряются. Предсказания случайного леса также являются несколько шумными: обратите внимание на то, что некоторые заемщики с очень высоким баллом, говорящим о высокой кредитоспособности, по-прежнему в итоге получают предсказание невыплаты ссуды. Это является результатом нескольких необычных записей в данных и демонстрирует опасность переподгонки случайным лесом (см. врезку *"Компромисс между смещением и дисперсией"* ранее в этой главе).

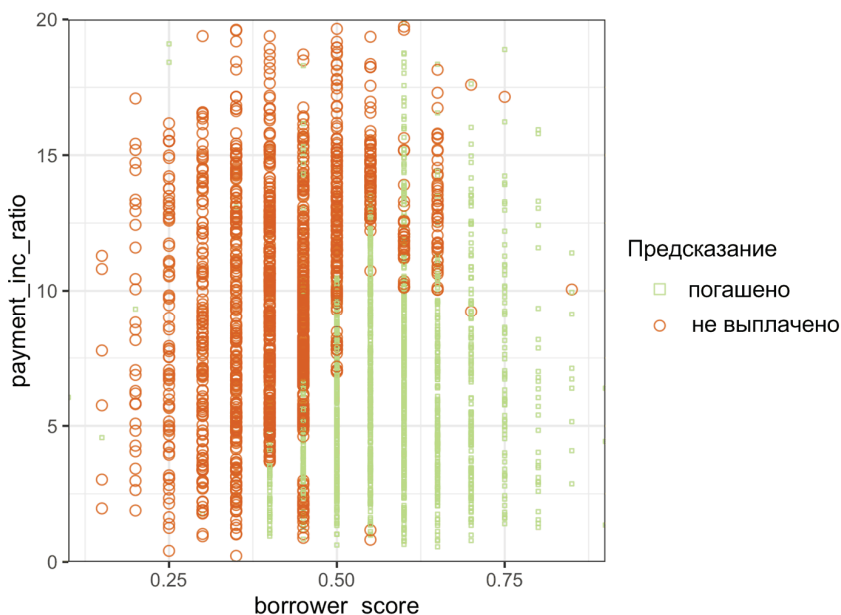


Рис. 6.7. Предсказанные исходы из случайного леса применительно к данным о ссудах

Важность переменных

Мощь алгоритма случайного леса проявляет себя, когда вы строите предсказательные модели для данных с многими признаками и записями. Он обладает способностью автоматически определять то, какие предсказатели являются важными, и обнаруживать сложные связи между предсказателями, соответствующие членам взаимодействия (см. раздел *"Взаимодействия и главные эффекты"* главы 4). Например, выполним подгонку модели к данным о ссудах, включив все столбцы. Следующий фрагмент кода показывает это на R:

```
rf_all <- randomForest(outcome ~ ., data=loan_data, importance=TRUE)
rf_all
Call:
  randomForest(formula = outcome ~ ., data = loan_data, importance = TRUE)
    Type of random forest: classification
    Number of trees: 500
No. of variables tried at each split: 4
```

OOB estimate of error rate: 33.79%

Confusion matrix:

	paid off	default	class.error
paid off	14676	7995	0.3526532
default	7325	15346	0.3231000

И на Python:

```
predictors = ['loan_amnt', 'term', 'annual_inc', 'dti', 'payment_inc_ratio',
              'revol_bal', 'revol_util', 'purpose', 'delinq_2yrs_zero',
              'pub_rec_zero', 'open_acc', 'grade', 'emp_length', 'purpose_',
              'home_', 'emp_len_', 'borrower_score']

outcome = 'outcome'

X = pd.get_dummies(loan_data[predictors], drop_first=True)
y = loan_data[outcome]

rf_all = RandomForestClassifier(n_estimators=500, random_state=1)
rf_all.fit(X, y)
```

Аргумент `importance=TRUE` предписывает, чтобы `randomForest` сохранил дополнительную информацию о важности разных переменных. Функция `varImpPlot` построит график относительной результативности переменных (относительно перестановки этой переменной):

```
varImpPlot(rf_all, type=1) ❶
varImpPlot(rf_all, type=2) ❷
```

- ❶ Среднее снижение точности.
- ❷ Среднее снижение загрязненности узла.

В Python класс `RandomForestClassifier` собирает информацию о важности признаков во время тренировки и делает ее доступной с помощью поля `feature_importances_`:

```
importances = rf_all.feature_importances_
```

"Снижение Джини" доступно как свойство `feature_importance_` подогнанного классификатора. Снижение точности, однако, не доступно для Python из коробки. Мы можем рассчитать его (`scores`), используя следующий фрагмент кода:

```
rf = RandomForestClassifier(n_estimators=500)
scores = defaultdict(list)

# выполнить перекрестный контроль баллов (scores)
# на ряде разных случайных разветвлений данных
for _ in range(3):
    train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.3)
    rf.fit(train_X, train_y)
    acc = metrics.accuracy_score(valid_y, rf.predict(valid_X))
    for column in X.columns:
        X_t = valid_X.copy()
        X_t[column] = np.random.permutation(X_t[column].values)
        shuff_acc = metrics.accuracy_score(valid_y, rf.predict(X_t))
        scores[column].append((acc-shuff_acc)/acc)
```

Результат показан на рис. 6.8. Похожий график можно создать на Python, используя следующий фрагмент кода:

```

df = pd.DataFrame({
    'признак': X.columns,
    'Снижение точности': [np.mean(scores[column]) for column in X.columns],
    'Снижение Джини': rf_all.feature_importances_,
})
df = df.sort_values('Снижение точности')

fig, axes = plt.subplots(ncols=2, figsize=(8, 4.5))
ax = df.plot(kind='barh', x='feature', y='Снижение точности',
              legend=False, ax=axes[0])
ax.set_ylabel('')
ax = df.plot(kind='barh', x='feature', y='Снижение Джини',
              legend=False, ax=axes[1])
ax.set_ylabel('')
ax.get_yaxis().set_visible(False)

```

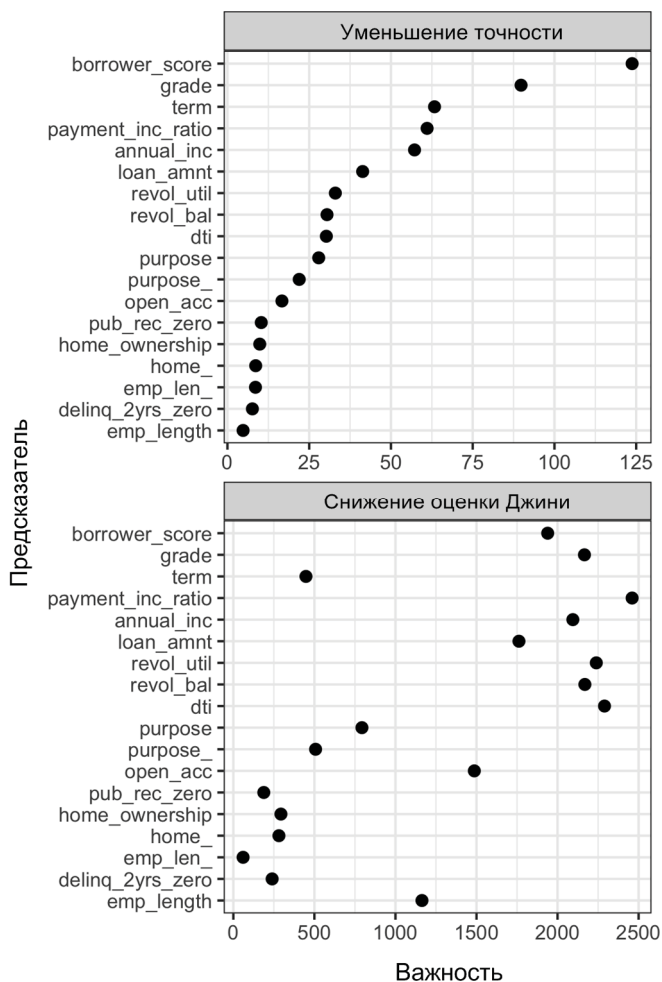


Рис. 6.8. Важность переменных для полной модели, подогнанной к данным о ссудах

Существуют два способа измерить важность переменных.

- ◆ Путем снижения точности модели, если значения переменной случайно перестановлены (`type=1`). Случайная перестановка значений имеет эффект удаления всей предсказательной силы для этой переменной. Точность вычисляется из внепакетных данных (поэтому такая мера является практически перекрестно-контрольной оценкой).
- ◆ Путем среднего снижения балла загрязненности Джини (см. раздел *"Измерение однородности или загрязненности"* ранее в этой главе) для всех узлов, которые были разветвлены на переменной (`type=2`). Этим измеряется то, насколько включение этой переменной в состав улучшает чистоту узлов. Указанная мера основывается на тренировочном наборе и, следовательно, менее надежна, чем мера, вычисленная на внепакетных данных.

Верхние и нижние части рис. 6.8 показывают важность переменных согласно снижению соответственно точности и загрязненности Джини. Переменные в обеих секциях графика ранжированы по снижению точности. Баллы важности переменных, производимые этими двумя мерами, очень различаются.

Поскольку снижение точности является более надежной метрикой, зачем нужно использовать меру снижения загрязненности Джини? По умолчанию `randomForest` вычисляет только загрязненность Джини, которая является побочным продуктом алгоритма, тогда как точность модели в зависимости от переменной требует дополнительных вычислений (случайной перестановки данных и предсказания этих данных). В случаях, где вычислительная сложность является важной, таких как в производственной обстановке, где выполняется подгонка тысяч моделей, возможно, не стоит тратить на дополнительные вычислительные усилия. Вдобавок к этому снижение загрязненности Джини проливает свет на то, какие переменные используются случайным лесом для создания своих правил ветвления (напомним, что указанная информация, легко видимая в простом дереве, практически теряется в случайном лесе).

Гиперпараметры

Случайный лес, как и многие статистические автоматически обучающиеся алгоритмы, можно рассматривать в качестве черного ящика алгоритма с кнопками для настройки работы ящика. Эти кнопки называются *гиперпараметрами*, т. е. параметрами, которые необходимо отрегулировать прежде, чем приступить к подгонке модели; они не оптимизируются как составная часть тренировочного процесса. В то время как традиционные статистические модели требуют выбора (например, выбора предсказателей для использования в регрессионной модели), гиперпараметры случайного леса имеют более важное значение, в особенности в предотвращении переподгонки. В частности, два самых важных гиперпараметра случайного леса таковы.

- ◆ `nodesize/min_samples_leaf` — минимальный размер терминальных узлов (листьев в дереве). По умолчанию в R для классификации используется 1 и для регрес-

сии — 5. Имплементация в пакете `scikit-learn` на Python использует 1 для обоих видов задач.

- ◆ `maxnodes/max_leaf_nodes` — максимальное количество узлов в каждом дереве решений. По умолчанию лимит отсутствует, и самое крупное дерево будет подогнано к ограничениям, установленным в `nodesize`. Обратите внимание на то, что в Python вы указываете максимальное число терминальных узлов. Оба параметра взаимосвязаны:

```
maxnodes = 2max_leaf_nodes - 1
```

Может возникнуть соблазн проигнорировать эти параметры и просто продолжить со значениями, используемыми по умолчанию. Однако такой подход может привести к перепогонке, когда вы применяете случайный лес к шумным данным. Когда вы увеличиваете `nodesize/min_samples_leaf` или устанавливаете `maxnodes/max_leaf_node`, алгоритм будет выполнять подгонку меньших деревьев и будет с меньшей вероятностью создавать мнимые предсказательные правила. Перекрестный контроль (см. раздел "Перекрестный контроль" главы 4) может использоваться для проверки эффектов от установки разных значений гиперпараметров.

Ключевые идеи для бэггинга и случайного леса

- Ансамблевые модели улучшают точность модели путем совмещения результатов многих моделей.
- Бэггинг — это особый тип ансамблевой модели, опирающийся на подгонку большого числа моделей к бутстрапированным выборкам из данных и усреднение моделей.
- Случайный лес — это специальный тип бэггинга, примененный к деревьям решений. В дополнение к повторному отбору данных алгоритм случайного леса отбирает предсказательные переменные во время ветвления деревьев.
- Полезным результатом на выходе из случайного леса является мера важности переменных, которая ранжирует предсказатели с точки зрения их вклада в точность модели.
- Случайный лес имеет набор гиперпараметров, которые следует отрегулировать с помощью перекрестного контроля во избежание перепогонки.

Бустинг

Ансамблевые модели стали стандартным инструментом для предсказательного моделирования. *Бустинг* (ансамблевое усиление) — это общее техническое решение, которое заключается в создании ансамбля моделей. Он был разработан примерно в то же время, что и *бэггинг* (см. раздел "Бэггинг и случайный лес" ранее в этой главе). Подобно бэггингу, бустинг очень широко используется с деревьями решений. Несмотря на их схожесть, бустинг принимает совсем другой подход — такой, кото-

рый сопровождается многими техническими излишествами. В результате, в отличие от бэггинга, который можно применять с относительно небольшой донастройкой, бустинг требует в своем применении намного большей внимательности. Если бы эти два метода были автомобилями, то бэггинг можно было бы рассматривать как автомобиль Honda модели Accord (надежный и устойчивый), тогда как бустинг был бы Porsche (мощный, но требует большего ухода)¹⁰.

В линейных регрессионных моделях часто экзаменуют остатки, чтобы посмотреть, можно ли улучшить подгонку (см. раздел "Графики частных остатков и нелинейность" главы 4). Бустинг продвинул эту идею гораздо дальше и выполняет подгонку серии моделей, в которой каждая последующая модель стремится минимизировать ошибку предыдущей модели. Часто используются несколько вариантов указанного алгоритма: *Adaboost*, *градиентный бустинг* и *стохастический градиентный бустинг*. Последний из перечисленных (стохастический градиентный бустинг) является самым общим и широко используется. Фактически при правильном выборе параметров этот алгоритм способен эмулировать случайный лес.

Ключевые термины для бустинга

Ансамбль (ensemble)

Формирование предсказания путем использования коллекции моделей.

Синонимы: модельное усреднение, модельная композиция.

Бустинг (boosting)

Общее техническое решение по подгонке последовательности моделей путем предоставления большего веса записям с крупными остатками для каждого последующего раунда.

Синонимы: ансамблевое усиление, бустирование.

Adaboost

Ранняя версия бустинга, опирающаяся на перевесовку данных на основе остатков.

Градиентный бустинг (gradient boosting)

Более общая форма бустинга, которая создана с точки зрения минимизации функции стоимости.

Стохастическое градиентный бустинг (stochastic gradient boosting)

Самый общий алгоритм бустинга, который включает повторный отбор записей и столбцов в каждом раунде.

¹⁰ В отличие от бэггинга (пакетного усреднения), где вы строите разные модели параллельно, а затем выбираете лучшую, в бустинге (ансамблевом усилении) вы строите модели, которые следуют друг за другом. При этом веса в алгоритмах корректируются на основе результатов предыдущей модели, что помогает предсказаниям с течением времени становиться лучше. — *Прим. перев.*

Регуляризация (regularization)

Технический прием предотвращения перепогонки путем добавления штрафного члена в функцию стоимости на ряде параметров в модели.

Гиперпараметры (hyperparameters)

Параметры, которые необходимо отрегулировать перед подгонкой алгоритма.

Алгоритм бустирования

Существуют разнообразные алгоритмы бустирования, и базовая идея в основе их всех, в сущности, одна и та же. Самым простым для понимания является алгоритм Adaboost, который работает следующим образом:

1. Инициализировать M , максимальное число моделей, подлежащих подгонке, и установить счетчик итераций в $m=1$. Инициализировать веса наблюдений $w_i = 1/N$ для $i=1, 2, \dots, N$. Инициализировать ансамблевую модель $\hat{F}_0 = 0$.
2. Используя веса наблюдений w_1, w_2, \dots, w_N , натренировать модель \hat{f}_m , которая минимизирует взвешенную ошибку e_m , определяемую путем суммирования весов неправильно классифицированных наблюдений.
3. Добавить модель в ансамбль: $\hat{F}_m = \hat{F}_{m-1} + \alpha_m \hat{f}_m$, где $\alpha_m = \frac{\log 1 - e_m}{e_m}$.
4. Обновить веса w_1, w_2, \dots, w_N так, чтобы были увеличены веса для наблюдений, которые были классифицированы неправильно. Размер увеличения зависит от α_m , при этом более крупные значения α_m приводят к большим весам.
5. Нарастить счетчик моделей $m = m + 1$. Если $m \leq M$, то перейти к шагу 2.

Бустированная оценка задается следующей формулой:

$$\hat{F} = \alpha_1 \hat{f}_1 + \alpha_2 \hat{f}_2 + \dots + \alpha_M \hat{f}_M.$$

Увеличивая веса для наблюдений, которые были классифицированы неправильно, алгоритм побуждает модели интенсивнее тренироваться на данных, для которых он показывал слабую результативность. Фактор α_m обеспечивает, чтобы модели с более низкой ошибкой имели больший вес.

Градиентный бустинг похож на Adaboost, но формирует задачу как оптимизацию функции стоимости. Вместо того чтобы корректировать веса, градиентный бустинг выполняет подгонку моделей к псевдоостатку, что имеет эффект более интенсивной тренировки на более крупных остатках. В духе случайного леса стохастический градиентный бустинг добавляет в алгоритм случайность, отбирая наблюдения и предсказательные переменные на каждом этапе.

XGBoost

Наиболее широко используемым бесплатным программно-информационным обеспечением для бустинга является XGBoost — имплементация стохастического градиентного бустинга, первоначально разработанного Тяньси Ченом (Tianqi Chen) и Карлосом Гестрином (Carlos Guestrin) в Вашингтонском университете. Его вычислительно эффективная имплементация со многими вариантами настройки доступна в качестве пакета для большинства основных языков программирования, используемых в науке о данных. В языке R XGBoost доступен в форме пакета `xgboost`¹¹ и под тем же именем также в Python.

Метод `xgboost` имеет много параметров, которые могут и должны быть скорректированы (см. раздел *"Гиперпараметры и перекрестный контроль"* далее в этой главе). Двумя очень важными параметрами являются `subsample`, управляющий долей наблюдений, которые должны отбираться во время каждой итерации, и `eta` — фактор сжатия, применяемый к α_m в алгоритме бустинга (см. раздел *"Алгоритм бустинга"* ранее в этой главе). Использование параметра `subsample` заставляет бустинг действовать как случайный лес за исключением того, что отбор выполняется без возврата. Параметр сжатия `eta` полезен для предотвращения перепогонки путем сокращения изменений в весах (меньшее изменение в весах означает, что алгоритм с меньшей вероятностью будет перепогодан к тренировочному набору). Следующий ниже фрагмент применяет `xgboost` в R к данным о ссудах всего с двумя предсказательными переменными:

```
predictors <- data.matrix(loan3000[, c('borrower_score', 'payment_inc_ratio')])
label <- as.numeric(loan3000[, 'outcome']) - 1
xgb <- xgboost(data=predictors, label=label, objective="binary:logistic",
               params=list(subsample=0.63, eta=0.1), nrounds=100)

[1]    train-error:0.358333
[2]    train-error:0.346333
[3]    train-error:0.347333
...
[99]    train-error:0.239333
[100]   train-error:0.241000
```

Обратите внимание, что `xgboost` не поддерживает формульный синтаксис, поэтому предсказатели нужно конвертировать в матрицу `data.matrix`, и отклик нужно конвертировать в двоичные переменные 0/1. Аргумент `objective` сообщает `xgboost` тип решаемой задачи; опираясь на эти данные `xgboost` выберет оптимизируемую метрику.

В Python `xgboost` имеет два разных интерфейса: API пакета `scikit-learn` и более функциональный интерфейс, как в R. С целью соответствовать другим методам пакета `scikit-learn` некоторые параметры были переименованы. Например, `eta` пере-

¹¹ См. <https://xgboost.readthedocs.io/>.

именовывается в `learning_rate`; использование `eta` не приведет к сбою, но и не даст желаемого эффекта:

```
predictors = ['borrower_score', 'payment_inc_ratio']
outcome = 'outcome'

X = loan3000[predictors]
y = loan3000[outcome]

xgb = XGBClassifier(objective='binary:logistic', subsample=0.63)
xgb.fit(X, y)
--
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
              max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
              n_estimators=100, n_jobs=1, nthread=None, objective='binary:logistic',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=0.63, verbosity=1)
```

Предсказанные значения можно получить из функции `predict` в R и, поскольку переменных всего две, их можно вывести на графике против предсказателей:

```
pred <- predict(xgb, newdata=predictors)
xgb_df <- cbind(loan3000, pred_default = pred > 0.5, prob_default = pred)
ggplot(data=xgb_df, aes(x=borrower_score, y=payment_inc_ratio,
                        color=pred_default, shape=pred_default, size=pred_default)) +
  geom_point(alpha=.8) +
  scale_color_manual(values = c('FALSE'='#b8e186', 'TRUE'='#d95f02')) +
  scale_shape_manual(values = c('FALSE'=0, 'TRUE'=1)) +
  scale_size_manual(values = c('FALSE'=0.5, 'TRUE'=2))
```

То же самое изображение можно создать на Python с помощью следующего фрагмента кода:

```
fig, ax = plt.subplots(figsize=(6, 4))

xgb_df.loc[xgb_df.prediction=='paid off'].plot(
    x='borrower_score', y='payment_inc_ratio', style='.',
    markerfacecolor='none', markeredgecolor='C1', ax=ax)
xgb_df.loc[xgb_df.prediction=='default'].plot(
    x='borrower_score', y='payment_inc_ratio', style='o',
    markerfacecolor='none', markeredgecolor='C0', ax=ax)
ax.legend(['погашено', 'не выплачено']);
ax.set_xlim(0, 1)
ax.set_ylim(0, 25)
ax.set_xlabel('borrower_score')
ax.set_ylabel('payment_inc_ratio')
```

Результат показан на рис. 6.9. В качественном плане он похож на предсказания из случайного леса (см. рис. 6.7). Предсказания являются несколько шумными в том,

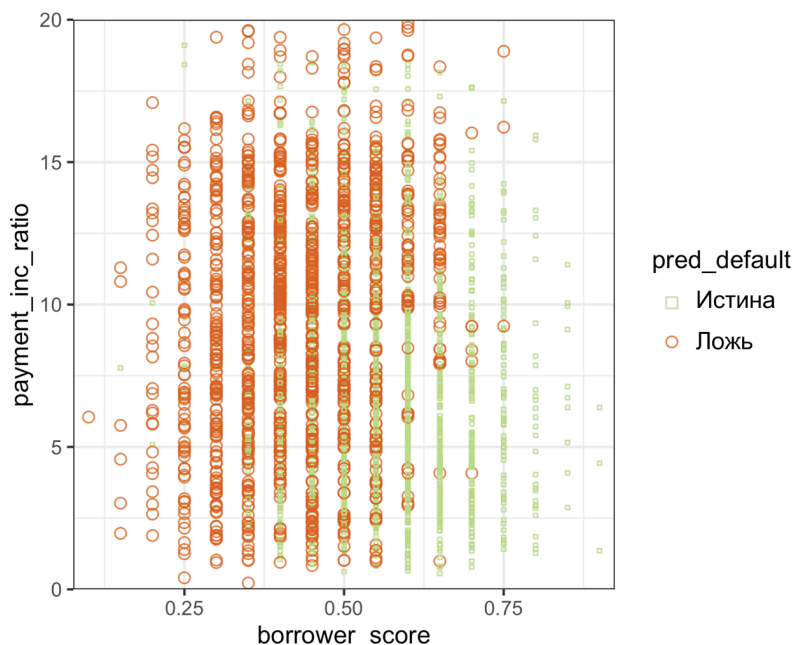


Рис. 6.9. Предсказанные исходы из XGBoost применительно к данным о ссудах

что некоторые заемщики с очень высоким баллом заемщика по-прежнему в итоге получают предсказание невыплаты ссуды.

Регуляризация: предотвращение перепогонки

Слепое применение `xgboost` может привести к нестабильным моделям в результате *перепогонки* к тренировочным данным. Проблема с перепогонкой имеет двоякую природу:

- ◆ точность модели на новых данных не из тренировочного набора будет деградировать;
- ◆ получаемые из модели предсказания являются высоковариабельными, что приводит к нестабильным результатам.

Любой технический прием моделирования потенциально подвержен перепогонке. Например, если слишком много переменных включены в уравнение регрессии, в итоге модель может прийти к мнимым предсказаниями. Однако в случае большинства статистических приемов перепогонку можно предотвратить разумным выбором предсказательных переменных. Даже случайный лес обычно порождает разумную модель без регулировки параметров.

Это, однако, не относится к `xgboost`. Давайте выполним подгонку `xgboost` к данным о ссудах для тренировочного набора, причем все переменные будут включены в модель. На R вы можете сделать это следующим образом:

```

seed <- 400820
predictors <- data.matrix(loan_data[, -which(names(loan_data) %in%
                                         'outcome')])
label <- as.numeric(loan_data$outcome) - 1
test_idx <- sample(nrow(loan_data), 10000)

xgb_default <- xgboost(data=predictors[-test_idx,], label=label[-test_idx],
                      objective='binary:logistic', nrounds=250, verbose=0)
pred_default <- predict(xgb_default, predictors[test_idx,])
error_default <- abs(label[test_idx] - pred_default) > 0.5
xgb_default$evaluation_log[250,]
mean(error_default)
-
iter train_error
1: 250    0.133043

[1] 0.3529

```

Мы используем функцию `train_test_split` в Python для разбиения набора данных на тренировочный и тестовый наборы:

```

predictors = ['loan_amnt', 'term', 'annual_inc', 'dti', 'payment_inc_ratio',
              'revol_bal', 'revol_util', 'purpose', 'delinq_2yrs_zero',
              'pub_rec_zero', 'open_acc', 'grade', 'emp_length', 'purpose_',
              'home_', 'emp_len_', 'borrower_score']
outcome = 'outcome'

X = pd.get_dummies(loan_data[predictors], drop_first=True)
y = pd.Series([1 if o == 'default' else 0 for o in loan_data[outcome]])

train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=10000)

xgb_default = XGBClassifier(objective='binary:logistic', n_estimators=250,
                           max_depth=6, reg_lambda=0, learning_rate=0.3,
                           subsample=1)
xgb_default.fit(train_X, train_y)

pred_default = xgb_default.predict_proba(valid_X)[:, 1]
error_default = abs(valid_y - pred_default) > 0.5
print('невыплата: ', np.mean(error_default))

```

Проверочный набор состоит из 10 тыс. записей, случайно отобранных из полных данных, и тренировочный набор — из оставшихся записей. Бустинг приводит к частоте появления ошибок, равному всего 14,6% для тренировочного набора. Проверочный набор, однако, имеет намного более высокую частоту появления ошибок — 36,2%. Это является результатом перепогонки: в то время как бустинг способен очень хорошо объяснить вариабельность в тренировочном наборе, правила предсказания не применимы к новым данным.

Бустинг обеспечивает несколько параметров во избежание перепогонки, в том числе параметры `eta` и `subsample` (см. раздел *"XGBoost" ранее в этой главе*). Еще один подход состоит в применении *регуляризации* — метода, который модифицирует функцию стоимости, для того чтобы штрафовать сложность модели. Деревья решений подгоняются путем минимизации критериев стоимости, в частности балла загрязненности Джини (см. раздел *"Измерение однородности или загрязненности" ранее в этой главе*). В `xgboost` существует возможность модифицировать функцию стоимости путем добавления члена, который измеряет сложность модели.

В `xgboost` имеются два параметра для регуляризации модели: `alpha` и `lambda`, которые представляют собой соответственно манхэттенское расстояние (L1-регуляризация) и квадратичное евклидово расстояние (L2-регуляризация) (см. раздел *"Метрики расстояния" ранее в этой главе*). Увеличение этих параметров будет штрафовать более сложные модели и уменьшать размер подгоняемых деревьев. Например, посмотрим, что произойдет, если установить `lambda` равным 1000 в R:

```
xgb_penalty <- xgboost(data=predictors[-test_idx,], label=label[-test_idx],
                      params=list(eta=.1, subsample=.63, lambda=1000),
                      objective='binary:logistic', nrounds=250, verbose=0)
pred_penalty <- predict(xgb_penalty, predictors[test_idx,])
error_penalty <- abs(label[test_idx] - pred_penalty) > 0.5
xgb_penalty$evaluation_log[250,]
mean(error_penalty)
-
iter train_error
1: 250      0.30966
[1] 0.3286
```

В API пакета `scikit-learn` эти параметры называются `reg_alpha` и `reg_lambda`:

```
xgb_penalty = XGBClassifier(objective='binary:logistic', n_estimators=250,
                           max_depth=6, reg_lambda=1000, learning_rate=0.1,
                           subsample=0.63)

xgb_penalty.fit(train_X, train_y)
pred_penalty = xgb_penalty.predict_proba(valid_X)[:, 1]
error_penalty = abs(valid_y - pred_penalty) > 0.5
print('штраф: ', np.mean(error_penalty))
```

Теперь тренировочная ошибка только немного ниже ошибки на проверочном наборе.

Метод `predict` в R предлагает удобный аргумент, `ntreelimit`, который заставляет использовать в предсказании только первые *i* деревьев. Это позволяет нам непосредственно сопоставлять внутривыборочную частоту появления ошибок с вневыборочной по мере включения большего числа моделей:

```
error_default <- rep(0, 250)
error_penalty <- rep(0, 250)
for(i in 1:250){
  pred_def <- predict(xgb_default, predictors[test_idx,], ntreelimit=i)
```

```

error_default[i] <- mean(abs(label[test_idx] - pred_def) >= 0.5)
pred_pen <- predict(xgb_penalty, predictors[test_idx,], ntreelimit=i)
error_penalty[i] <- mean(abs(label[test_idx] - pred_pen) >= 0.5)
}

```

В Python мы можем вызвать метод `predict_proba` с аргументом `ntree_limit`:

```

results = []
for i in range(1, 250):
    train_default = xgb_default.predict_proba(train_X, ntree_limit=i)[: , 1]
    train_penalty = xgb_penalty.predict_proba(train_X, ntree_limit=i)[: , 1]
    pred_default = xgb_default.predict_proba(valid_X, ntree_limit=i)[: , 1]
    pred_penalty = xgb_penalty.predict_proba(valid_X, ntree_limit=i)[: , 1]
    results.append({
        'iterations': i,
        'default train': np.mean(abs(train_y - train_default) > 0.5),
        'penalty train': np.mean(abs(train_y - train_penalty) > 0.5),
        'default test': np.mean(abs(valid_y - pred_default) > 0.5),
        'penalty test': np.mean(abs(valid_y - pred_penalty) > 0.5),
    })

```

```

results = pd.DataFrame(results)
results.head()

```

На выходе из модели содержится ошибка для тренировочного набора в компоненте `xgb_default$evaluation_log`. Объединив ее с вневыборочными ошибками, мы можем построить на графике ошибки против числа итераций:

```

errors <- rbind(xgb_default$evaluation_log,
               xgb_penalty$evaluation_log,
               ata.frame(iter=1:250, train_error=error_default),
               data.frame(iter=1:250, train_error=error_penalty))
errors$type <- rep(c('default train', 'penalty train',
                   'default test', 'penalty test'), rep(250, 4))
ggplot(errors, aes(x=iter, y=train_error, group=type)) +
  geom_line(aes(linetype=type, color=type))

```

Мы можем использовать метод построения графика пакета `pandas` для создания линейного графика. Ось, возвращенная из первого графика, позволяет нам накладывать дополнительные линии на один и тот же график. Этот шаблон поддерживается многими графическими пакетами Python:

```

ax = results.plot(x='iterations', y='default test')
results.plot(x='iterations', y='penalty test', ax=ax)
results.plot(x='iterations', y='default train', ax=ax)
results.plot(x='iterations', y='penalty train', ax=ax)

```

Результат, изображенный на рис. 6.10, показывает, как модель, заданная по умолчанию, неуклонно улучшает точность для тренировочного набора, но фактически становится хуже для проверочного набора. Оштрафованная модель такое поведение не проявляет.

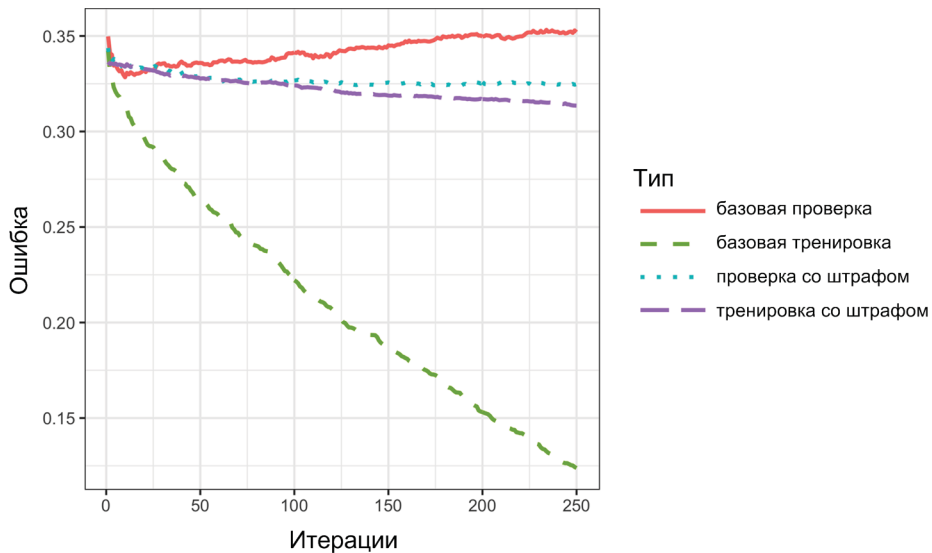


Рис. 6.10. Частота появления ошибок дефолтной модели XGBoost против оштрафованной версии XGBoost

Гребневая регрессия и лассо-регрессия

Наложение штрафа на сложность модели, чтобы помочь предотвратить перепогонку, берет начало с 1970-х годов. Регрессия на основе наименьших квадратов минимизирует остаточную сумму квадратов (RSS) (см. раздел "Наименьшие квадраты" главы 4). Гребневая регрессия минимизирует сумму квадратов остатков плюс штраф на число и размер коэффициентов:

$$\sum_{i=1}^n (Y_i - \hat{b}_0 - \hat{b}_1 x_1 - \dots - \hat{b}_p x_p)^2 + \lambda (\hat{b}_1^2 + \dots + \hat{b}_p^2).$$

Значение λ определяет, насколько много коэффициенты штрафуются; более крупные значения порождают модели, которые с меньшей вероятностью будут перепогонены к данным. Лассо-регрессия является похожей, за исключением того, что в качестве штрафного члена она использует манхэттенское расстояние вместо евклидова:

$$\sum_{i=1}^n (Y_i - \hat{b}_0 - \hat{b}_1 x_1 - \dots - \hat{b}_p x_p)^2 + \alpha (|\hat{b}_1| + \dots + |\hat{b}_p|).$$

Параметры `lambda` (`reg_lambda`) и `alpha` (`reg_alpha`) в `xgboost` действуют похожим образом.

Гиперпараметры и перекрестный контроль

Пакет `xgboost` имеет огромный массив гиперпараметров (см. врезку "Гиперпараметры XGBoost" далее в этой главе, в которой обсуждается этот вопрос). Как явствует из раздела "Регуляризация: предотвращение перепогонки" ранее в этой главе,

конкретный их выбор может существенно изменить подгонку модели. Учитывая огромное число сочетаний гиперпараметров на выбор, чем следует руководствоваться при выборе? Стандартное решение этой задачи — использовать *перекрестный контроль* (см. раздел "*Перекрестный контроль*" главы 4). Он случайно разбивает данные на K разных групп, так называемые *блоки*. Для каждого блока выполняется тренировка модели на данных, которые не находятся в блоке, и затем модель оценивается на данных в блоке. Это порождает меру точности модели на вневыборочных данных. Лучшим набором гиперпараметров является набор, который задан моделью с самой низкой совокупной ошибкой, что вычисляется путем усреднения ошибок из каждого блока.

В качестве иллюстрации указанного технического приема мы применим его к подборке параметров для `xgboost`. В этом примере мы разведем два параметра: параметр сжатия `eta` (см. раздел "*XGBoost*" ранее в этой главе) и `max_depth`. Параметр `max_depth` — это максимальная глубина от листового узла до корня дерева, при этом по умолчанию она равна 6. Он дает нам еще один способ осуществлять контроль над переподгонкой: глубокие деревья тяготеют к большей сложности и могут чрезмерно плотно прилегать к данным. Сначала мы задаем блоки и список параметров. В R это делается следующим образом:

```
N <- nrow(loan_data)
fold_number <- sample(1:5, N, replace=TRUE)
params <- data.frame(eta = rep(c(.1, .5, .9), 3),
                     max_depth = rep(c(3, 6, 12), rep(3,3)))
```

Теперь мы применим приведенный выше алгоритм для вычисления ошибки для каждой модели и каждого блока, используя пять блоков:

```
error <- matrix(0, nrow=9, ncol=5)
for(i in 1:nrow(params)){
  for(k in 1:5){
    fold_idx <- (1:N)[fold_number == k]
    xgb <- xgboost(data=predictors[-fold_idx,], label=label[-fold_idx],
                  params=list(eta=params[i, 'eta'],
                              max_depth=params[i, 'max_depth']),
                  objective='binary:logistic', nrounds=100, verbose=0)
    pred <- predict(xgb, predictors[fold_idx,])
    error[i, k] <- mean(abs(label[fold_idx] - pred) >= 0.5)
  }
}
```

В следующем фрагменте кода на Python мы создаем все возможные сочетания гиперпараметров, а затем выполняем подгонку и оценивание модели с каждым сочетанием:

```
idx = np.random.choice(range(5), size=len(X), replace=True)
error = []
for eta, max_depth in product([0.1, 0.5, 0.9], [3, 6, 9]): ❶
    xgb = XGBClassifier(objective='binary:logistic', n_estimators=250,
                       max_depth=max_depth, learning_rate=eta)
    cv_error = []
```

```

for k in range(5):
    fold_idx = idx == k
    train_X = X.loc[~fold_idx]; train_y = y[~fold_idx]
    valid_X = X.loc[fold_idx]; valid_y = y[fold_idx]

    xgb.fit(train_X, train_y)
    pred = xgb.predict_proba(valid_X)[:, 1]
    cv_error.append(np.mean(abs(valid_y - pred) > 0.5))
error.append({
    'eta': eta,
    'max_depth': max_depth,
    'avg_error': np.mean(cv_error)
})
print(error[-1])
errors = pd.DataFrame(error)

```

❶ Мы используем функцию `itertools.product` из стандартной библиотеки Python для создания всех возможных сочетаний двух гиперпараметров.

Поскольку мы выполняем подгонку в общей сложности 45 моделей, этот процесс займет некоторое время. Ошибки хранятся в качестве матрицы, при этом модели расположены вдоль строк и блоки — вдоль столбцов. Используя функцию `rowMeans`, мы можем сравнить частоту появления ошибок для разных наборов параметров:

```

avg_error <- 100 * round(rowMeans(error), 4)
cbind(params, avg_error)

```

	eta	max_depth	avg_error
1	0.1	3	32.90
2	0.5	3	33.43
3	0.9	3	34.36
4	0.1	6	33.08
5	0.5	6	35.60
6	0.9	6	37.82
7	0.1	12	34.56
8	0.5	12	36.83
9	0.9	12	38.18

Перекрестный контроль говорит о том, что использование более мелких деревьев с меньшим значением `eta` приводит к более точным результатам. Поскольку эти модели также стабильнее, наилучшими параметрами являются `eta=0.1` и `max_depth=3` (либо, возможно, `max_depth=6`).

Гиперпараметры XGBoost

Гиперпараметры в `xgboost` используются преимущественно для того, чтобы сбалансировать перепогонку с точностью и вычислительной сложностью. Полное обсуждение параметров вы можете найти, обратившись к документации по `xgboost`¹².

¹² См. https://oreil.ly/xC_OY.

- `eta/learning_rate` — фактор сжатия (темп усвоения) в диапазоне между 0 и 1, применяемый к α в алгоритме бустинга. По умолчанию используется 0,3, но для шумных данных рекомендуются меньшие значения (например, 0,1). В Python по умолчанию используется 0,1.
- `nrounds/n_estimators` — число раундов бустинга. Если `eta` установлен в малое значение, то важно увеличить число раундов, поскольку алгоритм усваивает медленнее. Раз уж несколько параметров включены с целью предотвращения перепогонки, то наличие большего числа раундов вреда не причинит.
- `max_depth` — максимальная глубина дерева (по умолчанию она равна 6). В отличие от случайного леса, который выполняет подгонку очень глубоких деревьев, бустинг обычно проводит подгонку мелких деревьев. Это имеет преимущество предотвращения мнимых сложных взаимодействий в модели, которые могут возникнуть из-за шумных данных. В Python по умолчанию используется 3.
- `subsample` и `colsample_bytree` — доля записей для отбора без возврата и доля предсказателей, отбираемых для использования в подгонке деревьев. Указанные параметры похожи на те, которые применяются в случайных лесах, и помогают предотвратить перепогонку. По умолчанию используется 1,0.
- `lambda/reg_lambda` и `alpha/reg_alpha` — параметры регуляризации для оказания помощи в контроле над перепогонкой (см. раздел *"Регуляризация: предотвращение перепогонки"* ранее в этой главе). В Python по умолчанию используются `reg_lambda = 1` и `reg_alpha=0`. В R оба значения по умолчанию равны 0.

Ключевые идеи для бустинга

- Бустинг — это класс ансамблевых моделей, основывающихся на подгонке последовательности моделей, где в последующих раундах больший вес придается записям с крупными ошибками.
- Стохастический градиентный бустинг — это самый общий тип бустинга, который обеспечивает наилучшую результативность. Наиболее часто встречающаяся форма стохастического градиентного бустинга использует древесные модели.
- XGBoost — это популярный и вычислительно эффективный пакет программ для стохастического градиентного бустинга; он доступен на всех общепринятых языках программирования, используемых в науке о данных.
- Бустинг подвержен перепогонке к данным, и для ее предотвращения необходимо настроить гиперпараметры.
- Регуляризация — это один из способов предотвратить перепогонку путем применения штрафного члена уравнения на ряде параметров (например, размер дерева) в модели.
- Перекрестный контроль в особенности важен для бустинга из-за большого числа гиперпараметров, которые необходимо задать.

Резюме

В этой главе были описаны два классификационных и предсказательных метода, которые гибко и локально "учатся" на данных вместо того, чтобы начинать со структурной модели (как, например, в линейной регрессии), которая подгоняется ко всему набору данных. Метод k ближайших соседей — это простой процесс, который смотрит вокруг на похожие записи и назначает предсказываемой записи их мажоритарный класс (или усредненное значение). Испытывая разные пороговые значения (точки ветвления) предсказательных переменных, древесные модели многократно делят данные на разделы и подразделы, которые становятся все более гомогенными относительно класса. Самые эффективные значения точки ветвления формируют путь, а также "правило" для классифицирования или предсказания. Древесные модели являются очень мощным и популярным предсказательным инструментом, часто превосходящим другие методы по результативности. Они дали начало различным ансамблевым методам (случайный лес, бэггинг, бустинг), которые оттачивают предсказательную силу деревьев.

Неконтролируемое самообучение

Термин "*неконтролируемое самообучение*" относится к статистическим методам, которые извлекают смысл из данных без тренировки модели на помеченных данных (данных, где интересующий исход известен). В главах 4–6 нашей целью было строительство модели (набора правил) для предсказания переменной отклика из набора предсказательных переменных. Все это относится к контролируемому самообучению. В отличие от него неконтролируемое самообучение тоже конструирует модель данных, но при этом не проводит границы между переменной отклика и предсказательными переменными.

Неконтролируемое обучение можно использовать для достижения самых разных целей. В некоторых случаях оно может применяться для создания предсказательного правила в отсутствие помеченного отклика. Методы *кластеризации* могут использоваться для выявления содержательных групп данных. Например, анализируя нажатия на веб-страницах и демографические данные пользователя на веб-сайте, нам, возможно, удастся сгруппировать разных пользователей по типам. После этого веб-сайт можно персонализировать под эти разные типы.

В других случаях цель может состоять в *уменьшении размерности данных* до более управляемого набора переменных. Этот сокращенный набор затем можно использовать в качестве входа в предсказательную модель, такую как регрессионная или классификационная. Например, у нас могут быть тысячи датчиков контроля за производственным процессом. Сократив данные до меньшего набора признаков, нам, возможно, будет под силу построить более мощную и поддающуюся интерпретации модель для предсказания сбоя в этом процессе, чем включив потоки данных от тысяч датчиков.

Наконец, неконтролируемое самообучение может рассматриваться как расширение разведывательного анализа данных (*см. главу 1*) до ситуаций, где вы сталкиваетесь с большим числом переменных и записей. Задача состоит в том, чтобы проникнуть вглубь набора данных, чтобы узнать, каким образом разные переменные друг с другом соотносятся. Технические приемы неконтролируемого самообучения обеспечивают вас способами просеивания и анализа этих переменных, а также обнаружения связей.



Предсказание и неконтролируемое самообучение

Неконтролируемое обучение может играть важную роль в предсказании, как для регрессионных, так и для классификационных задач. В некоторых случаях мы хотим предсказать категорию в отсутствие каких-либо помеченных данных. Например, мы, возможно, захотим предсказать тип растительности в кон-

кретном географическом регионе, исходя из набора спутниковых сенсорных данных. Поскольку у нас нет переменной отклика, чтобы натренировать модель, кластеризация предоставляет нам способ выявить часто встречающиеся регулярности и классифицировать регионы по категориям.

Кластеризация является особо важным инструментом для "задачи холодного старта". В задачах такого типа, как запуск новой маркетинговой кампании или выявление потенциально новых типов мошенничества или спама, у нас первоначально может не оказаться какого-либо отклика, чтобы натренировать модель. С течением времени, когда данные уже собраны, мы можем узнать о системе больше и построить традиционную предсказательную модель. Но кластеризация помогает нам запустить процесс самообучения быстрее путем выявления сегментов популяции.

Неконтролируемое самообучение также имеет важное значение, как строительный блок для регрессионных и классификационных технических приемов. Если в условиях больших данных малая субпопуляция представлена в совокупной популяции не очень хорошо, то натренированная модель может не показать хорошую результативность для этой субпопуляции. С кластеризацией существует возможность выявлять и помечать субпопуляции. Отдельные модели затем могут быть подогнаны к разным субпопуляциям. В качестве альтернативы субпопуляция может быть представлена своим собственным признаком, заставляя совокупную модель явным образом рассматривать идентичность субпопуляции в качестве предсказателя.

Анализ главных компонент

Нередко переменные будут варьироваться вместе (соварьироваться), и часть вариации в одной переменной фактически дублируется вариацией в другой (счет и чаевые в ресторане). Анализ главных компонент (principal components analysis, PCA) — это технический прием обнаружения пути, которым числовые переменные соварьируются¹.

Ключевые термины для анализа главных компонент

Главная компонента (principal component)

Линейная комбинация предсказательных переменных.

Нагрузки (loadings)

Веса, которые позволяют преобразовывать предсказатели в компоненты.

Синоним: веса.

График каменистой осыпи (screeplot)

График дисперсий компонент, показывающий относительную важность компонент в качестве объясненной дисперсии либо в качестве доли объясненной дисперсии.

¹ Этот и последующие разделы в настоящей главе закреплены за © 2020 Datastats, LLC, Питер Брюс, Эндрю Брюс и Питер Гедек; использовано с разрешения.

Идея анализа главных компонент состоит в том, чтобы совместить многочисленные числовые предсказательные переменные в меньший набор переменных, которые представляют собой взвешенные линейные комбинации исходного набора. Меньший набор переменных, *главные компоненты*, "объясняет" большую часть вариативности полного набора переменных, уменьшая размерность данных. Веса, используемые для формирования главных компонент, раскрывают относительные вклады исходных переменных в новые главные компоненты.

Анализ главных компонент был впервые предложен Карлом Пирсоном². В своей статье, которая, возможно, стала первой работой, посвященной неконтролируемому самообучению, Пирсон признавал, что во многих задачах в предсказательных переменных имеется вариативность, и поэтому он разработал анализ главных компонент как метод моделирования этой вариативности. Анализ главных компонент можно рассматривать как неконтролируемую версию линейного дискриминантного анализа (см. раздел "Дискриминантный анализ" главы 5).

Простой пример

Для двух переменных X_1 и X_2 имеются две главные компоненты Z_i ($i=1$ или $i=2$):

$$Z_i = w_{i,1}X_1 + w_{i,2}X_2.$$

Веса ($w_{i,1}$, $w_{i,2}$) называются *нагрузками* компонент. Они преобразовывают исходные переменные в главные компоненты. Первая главная компонента, Z_1 — это линейная комбинация, которая лучше всего объясняет суммарную вариацию. Вторая главная компонента — Z_2 — ортогональна первой и объясняет большинство оставшейся вариации, как может (если бы существовали дополнительные компоненты, то каждая дополнительная была бы ортогональной другим).



Общепринято также вычислять главные компоненты на отклонениях от среднего предсказательных переменных, а не на самих значениях.

Главные компоненты в R можно вычислить, используя функцию `princomp`. Следующий ниже фрагмент кода выполняет анализ главных компонент (PCA) на возвратности курса акций Chevron (CVX) и ExxonMobil (XOM):

```
oil_px <- sp500_px[, c('CVX', 'XOM')]  
pca <- princomp(oil_px)  
pca$loadings
```

```
Loadings:  
  Comp.1 Comp.2
```

² См. <https://oreil.ly/o4EeC>.

```
CVX -0.747 0.665
XOM -0.665 -0.747
```

	Comp.1	Comp.2
SS loadings	1.0	1.0
Proportion Var	0.5	0.5
Cumulative Var	0.5	1.0

В Python мы можем использовать имплементацию `sklearn.decomposition.PCA` в пакете `scikit-learn`:

```
pcs = PCA(n_components=2)
pcs.fit(oil_px)
loadings = pd.DataFrame(pcs.components_, columns=oil_px.columns)
loadings
```

Для акций CVX и XOM веса для первой главной компоненты составляют $-0,747$ и $-0,665$, для второй главной компоненты они равны $0,665$ и $-0,747$. Как это интерпретировать? Первая главная компонента — это, по существу, среднее от CVX и XOM, отражающее корреляцию между этими двумя энергетическими компаниями. Вторая компонента измеряет, когда курсы акций CVX и XOM дивергируют.

Почетительно вывести на график главные компоненты вместе с данными. Ниже мы создаем визуализацию на R:

```
loadings <- pca$loadings
ggplot(data=oil_px, aes(x=CVX, y=XOM)) +
  geom_point(alpha=.3) +
  stat_ellipse(type='norm', level=.99) +
  geom_abline(intercept = 0, slope = loadings[2,1]/loadings[1,1]) +
  geom_abline(intercept = 0, slope = loadings[2,2]/loadings[1,2])
```

Следующий ниже фрагмент кода создает похожую визуализацию на Python:

```
def abline(slope, intercept, ax):
    """Рассчитать координаты прямой, основываясь на наклоне и пересечении"""
    x_vals = np.array(ax.get_xlim())
    return (x_vals, intercept + slope * x_vals)

ax = oil_px.plot.scatter(x='XOM', y='CVX', alpha=0.3, figsize=(4, 4))
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
ax.plot(*abline(loadings.loc[0, 'CVX'] / loadings.loc[0, 'XOM'], 0, ax),
        '--', color='C1')
ax.plot(*abline(loadings.loc[1, 'CVX'] / loadings.loc[1, 'XOM'], 0, ax),
        '--', color='C1')
```

Результат показан на рис. 7.1.

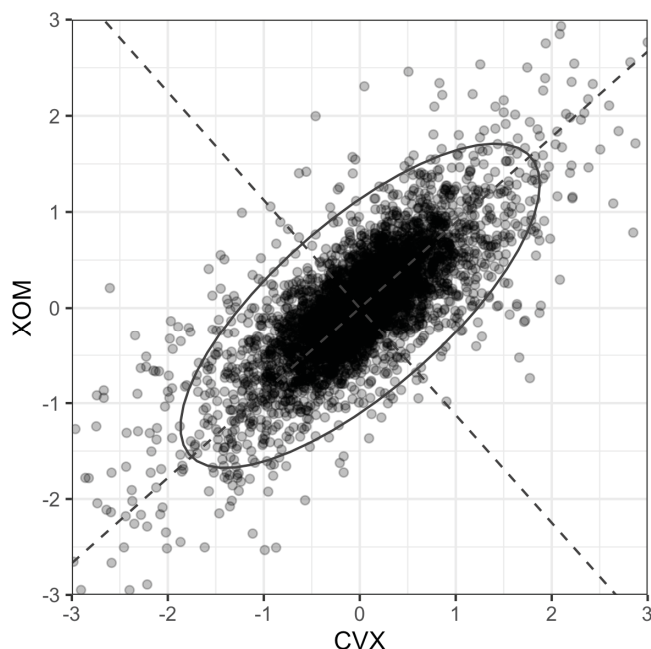


Рис. 7.1. Главные компоненты для возвратности акций Chevron и ExxonMobil

Пунктирные линии показывают направление двух главных компонент: первая проходит вдоль длинной оси эллипса и вторая — вдоль короткой оси. Вы видите, что большинство варибельности в двух возвратностях акций объясняется первой главной компонентой. Это имеет смысл, поскольку курсы энергетических акций имеют тенденцию перемещаться всей группой.



Оба веса первой главной компоненты отрицательны, но инвертирование знака всех весов не изменяет главную компоненту. Например, использование весов 0,747 и 0,665 для первой главной компоненты эквивалентно отрицательным весам, так же как бесконечная прямая, определяемая началом координат и точкой (1, 1) является одинаковой с прямой, определяемой началом координат и точкой (−1, −1).

Вычисление главных компонент

Переход от двух переменных к большому числу переменных является довольно прямолинейным. Для первой компоненты нужно просто внести дополнительные предсказательные переменные в линейную комбинацию, назначая веса, которые оптимизируют коллекцию ковариации из всех предсказательных переменных в эту первую главную компоненту (*ковариация*³ — это статистический термин; см. раз-

³ Термин "ковариация" (covariance) образован из приставки "со" (вместе) и корня "дисперсия" (variance). Такое прочтение облегчает его понимание, как содисперсии двух переменных. — Прим. перев.

дел "Матрица ковариаций" главы 5). Вычисление главных компонент — это классический статистический метод, опирающийся на корреляционную матрицу данных либо на матрицу ковариаций, и выполняется очень быстро, не завися от итерации. Как отмечено ранее, анализ главных компонент работает только с числовыми переменными, не категориальными. Полную процедуру вычисления можно описать следующим образом:

1. При создании первой главной компоненты анализ главных компонент приходит к линейной комбинации предсказательных переменных, которая максимизирует процент объясненной суммарной дисперсии.
2. Эта линейная комбинация далее становится первым "новым" предсказателем, Z_1 .
3. Анализ главных компонент повторяет этот процесс, используя те же переменные, но с разными весами, чтобы создать второй новый предсказатель, Z_2 . Взвешивание выполняется так, чтобы Z_1 и Z_2 не коррелировались.
4. Процедура продолжается до тех пор, пока не будет столько новых переменных, или компонент Z_i , сколько исходных переменных X_i .
5. Оставить столько компонент, сколько нужно для того, чтобы была охвачена большая часть дисперсии.
6. В этом месте получится набор весов для каждой компоненты. Последний шаг состоит в конвертировании исходных данных в новые баллы главных компонент путем применения весов к исходным значениям. Эти новые баллы могут далее использоваться в качестве уменьшенного набора предсказательных переменных.

Интерпретирование главных компонент

Природа главных компонент часто раскрывает информацию о структуре данных. Существует пара стандартных приемов визуализации, которые помогут вам вникнуть в суть главных компонент. Одним таким приемом является *график каменистой осыпи* (screeplot), предназначенный для визуализации относительной важности главных компонент (название происходит от сходства графика с каменистым откосом на боковой поверхности дорожного полотна). Следующий фрагмент кода на R является примером для нескольких ведущих компаний фондового индекса S&P 500:

```
syms <- c( 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM',  
          'SLB', 'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')  
top_sp <- sp500_px[row.names(sp500_px)>='2005-01-01', syms]  
sp_pca <- princomp(top_sp)  
screeplot(sp_pca)
```

Информация для создания графика нагрузок из результата работы пакета `scikit-learn` доступна в `explained_variance_`. Здесь мы конвертируем его в кадр данных пакета `pandas` и используем для построения столбикового графика:

```

syms = sorted(['AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB', 'COP',
              'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST'])
top_sp = sp500_px.loc[sp500_px.index >= '2011-01-01', syms]

sp_pca = PCA()
sp_pca.fit(top_sp)

explained_variance = pd.DataFrame(sp_pca.explained_variance_)
ax = explained_variance.head(10).plot.bar(legend=False, figsize=(4, 4))
ax.set_xlabel('Component')

```

Как видно из рис. 7.2, дисперсия первой главной компоненты является довольно крупной (как это часто и бывает), но другие верхние главные компоненты также важны.

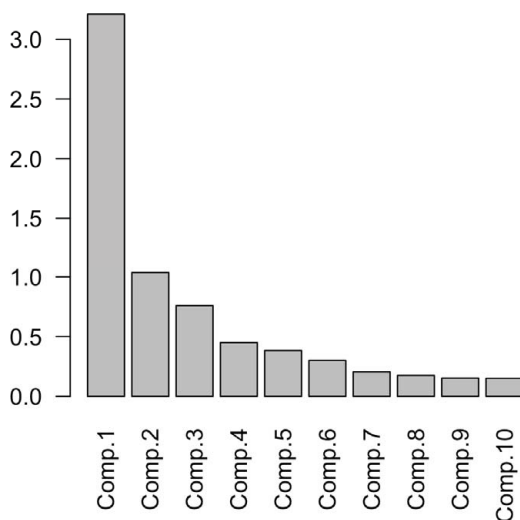


Рис. 7.2. График каменной осыпи для анализа главных компонент лидирующих акций из индекса S&P 500

В особенности показательным может быть изображение весов верхних главных компонент. Один из способов состоит в использовании функции `gather` из пакета `tidyr` в сочетании с пакетом `ggplot`:

```

library(tidyr)
loadings <- sp_pca$loadings[,1:5]
loadings$Symbol <- row.names(loadings)
loadings <- gather(loadings, 'Component', 'Weight', ~Symbol)
ggplot(loadings, aes(x=Symbol, y=Weight)) +
  geom_bar(stat='identity') +
  facet_grid(Component ~ ., scales='free_y')

```

Вот фрагмент кода для создания такой же визуализации на Python:

```

loadings = pd.DataFrame(sp_pca.components_[0:5, :], columns=top_sp.columns)
maxPC = 1.01 * np.max(np.max(np.abs(loadings.loc[0:5, :])))

```

```
f, axes = plt.subplots(5, 1, figsize=(5, 5), sharex=True)
for i, ax in enumerate(axes):
    pc_loadings = loadings.loc[i, :]
    colors = ['C0' if l > 0 else 'C1' for l in pc_loadings]
    ax.axhline(color='#888888')
    pc_loadings.plot.bar(ax=ax, color=colors)
    ax.set_ylabel(f'PC{i+1}')
    ax.set_ylim(-maxPC, maxPC)
```

Нагрузки для верхних пяти компонент показаны на рис. 7.3. Нагрузки для первой главной компоненты имеют одинаковый знак: это типично для данных, в которых все столбцы имеют общий фактор (в данном случае совокупный тренд фондового рынка). Вторая компонента улавливает ценовые изменения энергетических акций по сравнению с другими акциями. Третья компонента преимущественно противопоставляет динамику цен Apple и CostCo. Четвертая компонента противопоставляет динамику цен Schlumberger с другими энергетическими акциями. Наконец, в пятой компоненте доминирующие позиции занимают главным образом финансовые компании.

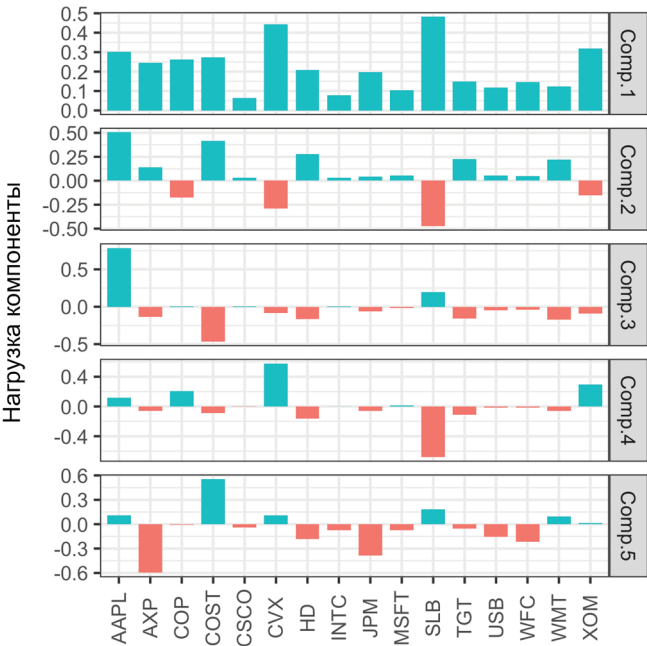


Рис. 7.3. Нагрузки для верхних пяти главных компонент возвратности курса акций



Сколько компонент выбрать?

Если ваша цель состоит в том, чтобы уменьшить размерность данных, то вы должны принять решение о том, сколько главных компонент выбрать. Наиболее часто встречающийся подход состоит в использовании нерегламентированного (ad hoc) правила отбирать компоненты, которые объясняют "большую

часть" дисперсии. Это можно сделать визуально посредством графика каменной осыпи; например, на рис. 7.2 было бы вполне естественно ограничить анализ верхними пятью компонентами. В качестве альтернативы вы можете отобрать верхние компоненты таким образом, чтобы кумулятивная дисперсия превышала порог, к примеру 80%. Кроме того, вы можете проинспектировать нагрузки, чтобы определить, имеет ли компонента интуитивно понятную интерпретацию. Перекрестный контроль предоставляет более формальный метод отбора числа значимых компонент (дополнительную информацию см. в разделе "Перекрестный контроль" главы 4).

Анализ соответствия

Анализ главных компонент (PCA) не может использоваться для категориальных данных; однако несколько родственным методом является *анализ соответствия*. Его цель состоит в том, чтобы распознать ассоциации между категориями или между категориальными признаками. Сходство между анализом соответствия и анализом главных компонент в основном спрятано под капотом — матричная алгебра для шкалирования размерностей. Анализ соответствия применяется главным образом для графического анализа низкоразмерных категориальных данных и не используется таким же образом, как PCA для уменьшения размерностей в качестве подготовительного шага при работе с большими данными.

Вход можно рассматривать как таблицу, в которой строки представляют одну переменную, столбцы — другую, а в ячейках хранится число записей. Выход (после некоторой матричной алгебры) представляет собой *биграфик* — диаграмму рассеяния со шкалированными осями (и с процентами, указывающими, насколько дисперсия объясняется этой размерностью). Смысл единиц измерения на осях интуитивно не связан с исходными данными, и главное значение диаграммы рассеяния состоит в том, чтобы графически проиллюстрировать переменные, которые ассоциированы друг с другом (по близости на графике). См., например, рис. 7.4, на котором работа по дому распределяется в зависимости от того, выполняются ли они совместно или в одиночку (вертикальная ось), а также от того, кто несет основную ответственность — жена или муж (горизонтальная ось). Возврат анализа соответствия насчитывает много десятилетий, как и дух этого примера, судя по назначению задач.

В R имеется целый ряд пакетов для анализа соответствия. Ниже мы используем пакет `ca`:

```
ca_analysis <- ca(housetasks)
plot(ca_analysis)
```

В Python мы можем использовать пакет `prince`, который имплементирует анализ соответствия с помощью API пакета `scikit-learn`:

```
ca = prince.CA(n_components=2)
ca = ca.fit(housetasks)

ca.plot_coordinates(housetasks, figsize=(6, 6))
```

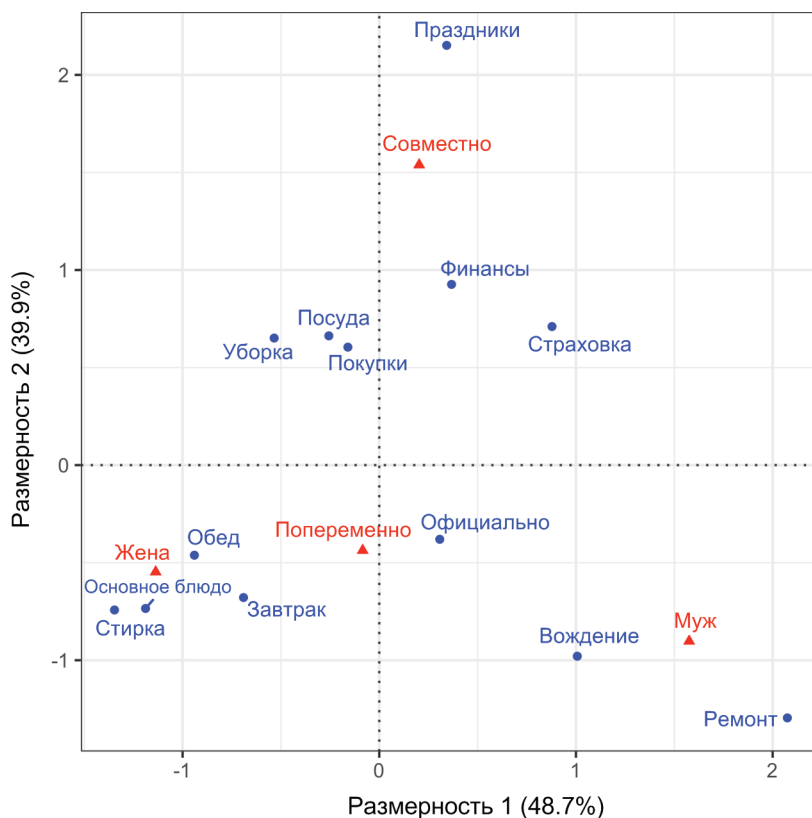



Рис. 7.4. Графическое представление анализа соответствия данных работы по дому

Ключевые идеи для анализа главных компонент

- Главные компоненты — это линейные комбинации предсказательных переменных (только числовые данные).
- Их вычисляют с целью минимизировать корреляцию между компонентами, уменьшая избыточность.
- Лимитированное число компонент в типичной ситуации будет объяснять большую часть дисперсии в переменной исхода.
- Лимитированный набор главных компонент далее можно использовать вместо (более многочисленных) исходных предсказателей, тем уменьшая размерность.
- Внешне похожим технически приемом для категориальных данных является анализ соответствия, но он не полезен в контексте больших данных.

Дополнительные материалы для чтения

Подробный обзор применения перекрестного контроля в анализе главных компонент см. в статье "Перекрестный контроль компонентных моделей: критический взгляд на существующие методы"⁴ (Bro R., Kjeldahl K., Smilde A. K., Kiers H. A. L. Cross-validation of component models: a critical look at current methods // Anal. Bioanal. Chem. — 2008. — № 390. — P. 1241–1251).

Кластеризация на основе K средних

Кластеризация — это технический прием деления данных на разные группы, такие, что записи в каждой группе похожи друг на друга. Цель кластеризации — выявлять значительные и содержательные группы данных. Эти группы могут использоваться непосредственно, анализироваться более углубленно либо передаваться как признак или исход в предсказательную регрессию либо классификацию. K средних как метод кластеризации был разработан самым первым; он по-прежнему широко используется и обязан своей популярностью относительной простоте алгоритма и его способности масштабироваться до крупных наборов данных.

Ключевые термины для кластеризации на основе K средних

Кластер (cluster)

Группа записей, похожих друг на друга.

Центр кластера (cluster mean)

Вектор средних значений переменных для записей в кластере.

Синонимы: центроид, кластерное среднее, центр масс.

K

Число кластеров.

Метод K средних делит данные на K кластеров путем минимизации суммы квадратичных расстояний каждой записи до *среднего значения* назначенного ей кластера. Это среднее, также именуемое *центром кластера*, или *центроидом*, выражается *внутрикластерной суммой квадратов*, или *внутрикластерной SS* (sum of squares). K средних не обеспечивает одинаковый размер кластеров, но он отыскивает кластеры, которые разделены наилучшим образом.



Нормализация

В типичной ситуации непрерывные переменные нормализуются (стандартизируются) путем вычитания среднего значения и деления на стандартное отклонение. В противном случае переменные с крупной шкалой значений будут доминировать над процессом кластеризации (см. раздел "Стандартизация (нормализация, z -оценки)" главы 6).

⁴ См. <https://oreil.ly/yVryf>.

Простой пример

Начнем с того, что рассмотрим набор данных с n записями и всего двумя переменными — x и y . Предположим, что мы хотим разделить данные на $K = 4$ кластеров. Это значит отнесение каждой записи (x_i, y_i) к кластеру k . После отнесения n_k записей к кластеру k центром кластера (\bar{x}_k, \bar{y}_k) является среднее значение точек в кластере:

$$\bar{x}_k = \frac{1}{n_k} \sum_{i \in \text{кластер } k} x_i;$$

$$\bar{y}_k = \frac{1}{n_k} \sum_{i \in \text{кластер } k} y_i.$$



Центр кластера

В кластеризации записей с многочисленными переменными (типичный случай) термин "*центр кластера*" обозначает не одно число, а вектор средних значений переменных.

Сумма квадратов внутри кластера задается следующей формулой:

$$SS_k = \sum_{i \in \text{кластер } k} (x_i - \bar{x}_k)^2 + (y_i - \bar{y}_k)^2.$$

Метод K средних отыскивает такое отнесение записей к кластерам, которое минимизирует внутрикластерную сумму квадратов по всем четырем кластерам $SS_1 + SS_2 + SS_3 + SS_4$.

$$\sum_{k=1}^4 SS_k.$$

Типичное применение кластеризации состоит в локализации естественных, раздельных кластеров в данных. Еще одно применение заключается в делении данных на predetermined число отдельных групп, где кластеры используются с целью обеспечения того, чтобы группы как можно больше отличались друг от друга.

Например, предположим, что мы хотим разделить дневную возвратность акций на четыре группы. Кластеризация K средних может быть использована для разделения данных на наилучшие группы. Обратите внимание, что ежедневные возвратности акций сообщаются в форме, которая, по сути, стандартизирована, поэтому нам не нужно нормализовать данные. В R кластеризация K средних может быть выполнена с помощью функции `kmeans`. Например, ниже приведены четыре кластера, основанные на двух переменных — дневной возвратности акций ExxonMobil (XOM) и Chevron (CVX):

```
df <- sp500_px[row.names(sp500_px) >= '2011-01-01', c('XOM', 'CVX')]  
km <- kmeans(df, centers=4)
```

В Python мы используем метод `sklearn.cluster.KMeans` из пакета `scikit-learn`:

```
df = sp500_px.loc[sp500_px.index >= '2011-01-01', ['XOM', 'CVX']]  
kmeans = KMeans(n_clusters=4).fit(df)
```

Отнесение каждой записи к кластеру возвращается в компоненте `cluster` (R):

```
df$cluster <- factor(km$cluster)
head(df)
```

	XOM	CVX	cluster
2011-01-03	0.73680496	0.2406809	2
2011-01-04	0.16866845	-0.5845157	1
2011-01-05	0.02663055	0.4469854	2
2011-01-06	0.24855834	-0.9197513	1
2011-01-07	0.33732892	0.1805111	2
2011-01-10	0.00000000	-0.4641675	1

В пакете `scikit-learn` метки кластера доступны в поле `labels_`:

```
df['cluster'] = kmeans.labels_
df.head()
```

Первые шесть записей относятся к кластеру 1 либо к кластеру 2. Также возвращаются центры кластеров (R):

```
centers <- data.frame(cluster=factor(1:4), km$centers)
centers
```

	cluster	XOM	CVX
1	1	-0.3284864	-0.5669135
2	2	0.2410159	0.3342130
3	3	-1.1439800	-1.7502975
4	4	0.9568628	1.3708892

В пакете `scikit-learn` центры кластеров доступны в поле `cluster_centers_`:

```
centers = pd.DataFrame(kmeans.cluster_centers_, columns=['XOM', 'CVX'])
centers
```

Кластеры 1 и 3 представляют "падающие" рынки, в то время как кластеры 2 и 4 представляют рынки "растущие".

Поскольку алгоритм *K* средних использует рандомизированные стартовые точки, результаты могут различаться между последующими запусками и разными имплементациями метода. В общем случае вы должны проверить, что флуктуации не являются слишком крупными.

В этом примере всего с двумя переменными визуализация кластеров и их центров довольно прямолинейна:

```
ggplot(data=df, aes(x=XOM, y=CVX, color=cluster, shape=cluster)) +
  geom_point(alpha=.3) +
  geom_point(data=centers, aes(x=XOM, y=CVX), size=3, stroke=2)
```

Функция `scatterplot` пакета `seaborn` позволяет легко расцвечивать (hue) и стилизовать (style) точки по свойству:

```
fig, ax = plt.subplots(figsize=(4, 4))
ax = sns.scatterplot(x='XOM', y='CVX', hue='cluster', style='cluster',
                    ax=ax, data=df)
```

```
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
centers.plot.scatter(x='XOM', y='CVX', ax=ax, s=50, color='black')
```

Результирующий график, приведенный на рис. 7.5, показывает отнесения к кластерам и центры кластеров. Обратите внимание, что алгоритм K средних будет относить записи к кластерам, даже если эти кластеры не очень хорошо отделены (что может быть полезно, если вам нужно оптимально разделить записи на группы).

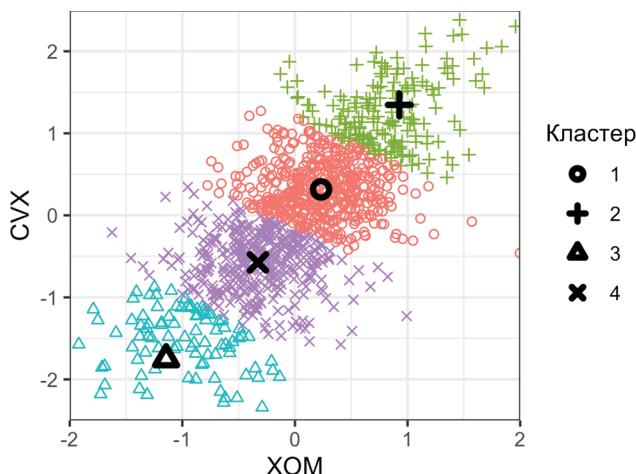


Рис. 7.5. Кластеры K средних применительно к данным курсов акций ExxonMobil и Chevron (центры кластеров выделены черными символами)

Алгоритм K средних

В общем случае алгоритм K средних может применяться к набору данных с p переменными X_1, \dots, X_p . Хотя найти точное решение K средних в вычислительном плане очень трудно, эвристические алгоритмы обеспечивают эффективный способ вычисления локально оптимального решения.

Алгоритм начинается с определенного пользователем числа K и первоначального набора центров кластеров, а затем в цикле прокручивает следующие шаги:

1. Отнести каждую запись к центру ближайшего кластера согласно измеренному квадратичному расстоянию.
2. Пересчитать центры кластеров, основываясь на отнесении записей к кластерам.

Алгоритм сходится, когда отнесение записей к кластерам не изменяется.

Для первой итерации вам нужно указать исходный набор центров кластеров. Обычно вы делаете это путем случайного отнесения каждой записи к одному из K кластеров, затем отыскания средних этих кластеров.

Поскольку не гарантируется, что этот алгоритм найдет лучшее решение, рекомендуется выполнять алгоритм несколько раз с использованием разных случайных выборок для инициализации алгоритма. Когда используется больше одного набора итераций, результат K средних задается итерацией, которая имеет самую низкую внутрикластерную сумму квадратов.

Параметр `nstart` функции `kmeans` в R позволяет указывать число случайных запусков для попыток. Например, следующий фрагмент кода выполняет алгоритм K средних, чтобы отыскать 5 кластеров с использованием 10 разных стартовых центров кластеров:

```
syms <- c('AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB', 'COP',  
          'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')  
df <- sp500_px[row.names(sp500_px) >= '2011-01-01', syms]  
km <- kmeans(df, centers=5, nstart=10)
```

Указанная функция автоматически возвращает наилучшее решение из 10 разных стартовых точек. Вы можете использовать параметр `iter.max` для установления максимального числа итераций, которое дается алгоритму для каждого случайного запуска.

Алгоритм пакета `scikit-learn` по умолчанию повторяется 10 раз (`n_init`). Аргумент `max_iter` (по умолчанию 300) может использоваться для контроля числа итераций:

```
syms = sorted(['AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB', 'COP',  
              'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST'])  
top_sp = sp500_px.loc[sp500_px.index >= '2011-01-01', syms]  
kmeans = KMeans(n_clusters=5).fit(top_sp)
```

Интерпретирование кластеров

Важная часть кластерного анализа может быть сопряжена с интерпретированием кластеров. Двумя самыми важными результатами на выходе из `kmeans` являются размеры кластеров и центры кластеров. Например, в размеры результирующих кластеров из предыдущего раздела задаются такой командой на R:

```
km$size  
[1] 106 186 285 288 266
```

В Python, для того чтобы получить эту информацию, мы можем использовать класс `collections.Counter` из стандартной библиотеки. Из-за различий в имплементации и присущей алгоритму случайности результаты будут различаться:

```
from collections import Counter  
Counter(kmeans.labels_)  
  
Counter({4: 302, 2: 272, 0: 288, 3: 158, 1: 111})
```

Размеры кластеров относительно сбалансированы. Несбалансированные кластеры могут быть результатом, вызванным дальними выбросами либо группами записей,

которые сильно различаются от остальной части данных, — обе причины могут нуждаться в дальнейшем обследовании.

Вы можете построить график центров кластеров с помощью функции `gather` в сопряжении с `ggplot`:

```
centers <- as.data.frame(t(centers))
names(centers) <- paste("Cluster", 1:5)
centers$Symbol <- row.names(centers)
centers <- gather(centers, 'Cluster', 'Mean', -Symbol)
centers$Color = centers$Mean > 0
ggplot(centers, aes(x=Symbol, y=Mean, fill=Color)) +
  geom_bar(stat='identity', position='identity', width=.75) +
  facet_grid(Cluster ~ ., scales='free_y')
```

Код для создания этой визуализации на Python аналогичен тому, что мы использовали для анализа главных компонент:

```
centers = pd.DataFrame(kmeans.cluster_centers_, columns=syms)

f, axes = plt.subplots(5, 1, figsize=(5, 5), sharex=True)
for i, ax in enumerate(axes):
    center = centers.loc[i, :]
    maxPC = 1.01 * np.max(np.abs(center))
    colors = ['C0' if l > 0 else 'C1' for l in center]
    ax.axhline(color='#888888')
    center.plot.bar(ax=ax, color=colors)
    ax.set_ylabel(f'Cluster {i + 1}')
    ax.set_ylim(-maxPC, maxPC)
```

Результирующий график показан на рис. 7.6 и демонстрирует природу каждого кластера. Например, кластеры 1 и 2 соответствуют дням, в которые рынок является соответственно падающим и растущим. Кластеры 3 и 5 характеризуются соответственно днями растущего рынка акций потребительского рынка и днями падающего рынка энергетических акций. Наконец, кластер 4 улавливает дни, в которые энергетические акции росли, а акции потребительского рынка падали.



Кластерный анализ против анализа главных компонент

График центров кластеров, в сущности, аналогичен рассмотрению нагрузок в анализе главных компонент (РСА) (см. раздел *"Интерпретирование главных компонент"* ранее в этой главе). Главенствующая отличительная особенность состоит в том, что в отличие от анализа главных компонент знак кластерных центров имеет содержательный смысл. Анализ главных компонент выявляет главные направления вариации, тогда как кластерный анализ отыскивает группы записей, расположенных близко друг к другу.

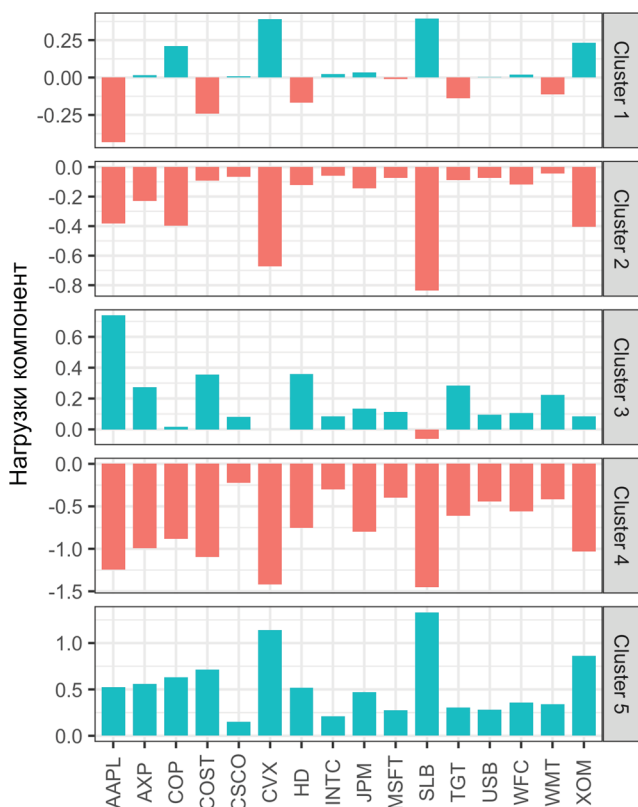


Рис. 7.6. Средние значения переменных в каждом кластере ("центры кластеров")

Выбор числа кластеров

Алгоритм K средних требует, чтобы вы задали число кластеров K . Иногда число кластеров обусловлено приложением. Например, компания, организующая работу отдела продаж, возможно, захочет сгруппировать клиентов в "типажи", на которых следует сосредоточиться и которые будут служить ориентиром во время коммерческих звонков. В таком случае менеджерские соображения будут диктовать число планируемых потребительских сегментов. Например, два сегмента могут не привести к полезной дифференциации клиентов, тогда как восьми может быть слишком много, чтобы с ними справиться.

В отсутствие числа кластеров, продиктованного практическими или организаторскими соображениями, можно воспользоваться статистическим подходом. Не существует единого стандартного метода отыскания "наилучшего" числа кластеров.

Общепринятый подход, который называется *методом локтя* (elbow method), состоит в выявлении точки, когда набор кластеров объясняет "большую часть" дисперсии в данных. Добавление новых кластеров вне этого набора вносит относительно малый вклад в объясненную дисперсию. Локоть — это точка, где кумуля-

тивная объясненная дисперсия сглаживается (выходит на плато) после крутого подъема, отсюда и название указанного метода.

На рис. 7.7 показан кумулятивный процент объясненной дисперсии для данных об акциях для числа кластеров, варьирующегося в интервале от 2 до 15. Где же локоть в этом примере? Очевидный кандидат отсутствует, поскольку поступательное увеличение дисперсии постепенно падает. Это довольно типичная ситуация для данных, которые не имеют четко определенных кластеров. Возможно, в этом заключается недостаток метода локтя, но он действительно показывает природу данных.

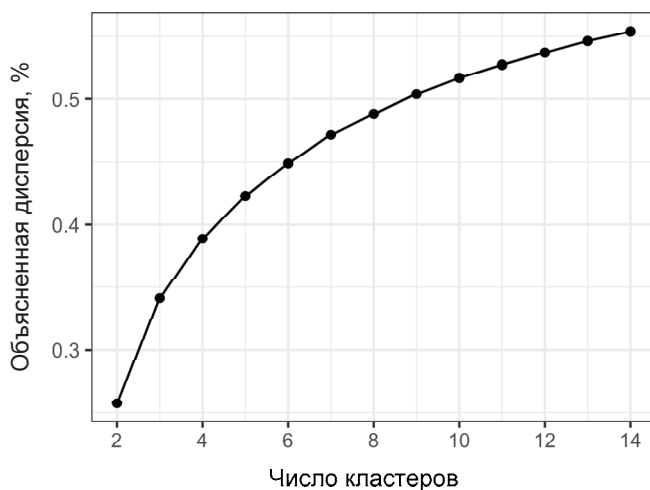


Рис. 7.7. Метод локтя применительно к данным об акциях

В R функция `kmeans` не обеспечивает единую команду для применения метода локтя, но его можно с готовностью применить, используя результат на выходе из `kmeans`, как показано ниже:

```
pct_var <- data.frame(pct_var = 0, num_clusters = 2:14)
totalss <- kmeans(df, centers=14, nstart=50, iter.max=100)$totss
for (i in 2:14) {
  kmCluster <- kmeans(df, centers=i, nstart=50, iter.max=100)
  pct_var[i-1, 'pct_var'] <- kmCluster$betweenss / totalss
}
```

Для результата работы класса `KMeans` мы получаем эту информацию из свойства `inertia_`. После конверсии в кадр данных пакета `pandas` мы можем использовать его метод `plot` для создания графика:

```
inertia = []
for n_clusters in range(2, 14):
    kmeans = KMeans(n_clusters=n_clusters, random_state=0).fit(top_sp)
    inertia.append(kmeans.inertia_ / n_clusters)
```

```
inertias = pd.DataFrame({'n_clusters': range(2, 14), 'inertia': inertia})
ax = inertias.plot(x='n_clusters', y='inertia')
plt.xlabel('Число кластеров (k)')
plt.ylabel('Средние внутрикластерные квадратичные расстояния')
plt.ylim((0, 1.1 * inertias.inertia.max()))
ax.legend().set_visible(False)
```

В оценивании того, сколько кластеров следует оставить, пожалуй, самый важный тест состоит в следующем: какова вероятность, что кластеры будут реплицированы на новых данных? Поддаются ли кластеры интерпретации, и связаны ли они с общими характеристиками данных, или же они просто отражают конкретный экземпляр? Вы можете это частично квалифицировать при помощи перекрестного контроля (см. раздел "Перекрестный контроль" главы 4).

В общем случае нет единого правила, которое будет надежно ориентировать относительно того, сколько кластеров порождать.



Существует несколько более формальных способов определения числа кластеров, которые основываются на статистической теории либо теории информации. Например, Роберт Тибширани, Гюнтер Уолтер и Тревор Хейсти (Robert Tibshirani, Guenther Walther, Trevor Hastie) предлагают гэп-статистику (от англ. *gap* — скачок, зазор)⁵, опираясь на статистическую теорию при выявлении локтя. Для большинства приложений теоретический подход, вероятно, не понадобится или даже не будет уместен.

Ключевые идеи для кластеризации на основе K средних

- Число требуемых кластеров K выбирается пользователем.
- Алгоритм уточняет кластеры путем итеративного отнесения записей к ближайшему кластерному центру до тех пор, пока отнесения к кластерам не перестанут изменяться.
- Над выбором числа K обычно доминируют соображения практического характера; статистически обусловленного оптимального числа кластеров не существует.

Иерархическая кластеризация

Иерархическая кластеризация является альтернативной для K средних, которая может порождать совсем другие кластеры. Иерархическая кластеризация позволяет пользователю визуализировать эффект задания разных чисел кластеров. Она является более чувствительной в обнаружении выбросов или отклоняющихся групп или записей. Иерархическая кластеризация также поддается интуитивно понятному графическому изображению, приводя к более простой интерпретации кластеров.

⁵ См. <https://oreil.ly/d-N3>.

Ключевые термины для иерархической кластеризации

Дендограмма (dendrogram)

Визуальное представление записей и иерархии кластеров, которым они принадлежат.

Расстояние (distance)

Метрика, измеряющая степень близости одной записи к другой.

Несхожесть (dissimilarity)

Метрика, измеряющая степень близости одного кластера к другому.

Синонимы: неподобие, различие.

Гибкость иерархической кластеризации имеет свою стоимость, и иерархическая кластеризация не масштабируется хорошо до крупных наборов данных с миллионами записей. Даже для данных скромного размера всего с десятками тысяч записей иерархическая кластеризация может потребовать интенсивных вычислительных ресурсов. И действительно, большинство приложений иерархической кластеризации сосредоточены на относительно малых наборах данных.

Простой пример

Иерархическая кластеризация работает на наборе данных с n записями и p переменными и базируется на двух главных строительных блоках:

- ♦ метрике расстояния $d_{i,j}$, которая измеряет расстояние между двумя записями — i и j ;
- ♦ метрике несхожести $D_{A,B}$, которая измеряет разницу между двумя кластерами — A и B , основываясь на расстояниях $d_{i,j}$ между членами каждого кластера.

Для приложений, сопряженных с числовыми данными, наиболее важным решением является выбор метрики несхожести. Иерархическая кластеризация начинается с установки каждой записи в качестве самостоятельного кластера и затем работает в цикле, перебирая их с целью объединения наименее несхожих кластеров.

В R функцию `hclust` можно использовать для выполнения иерархической кластеризации. Одно большое отличие `hclust` от `kmeans` состоит в том, что указанная функция оперирует на попарных расстояниях $d_{i,j}$ нежели на самих данных как таковых. Вы можете их вычислить при помощи функции `dist`. Например, следующий фрагмент кода применяет иерархическую кластеризацию к возвратностям акций для ряда компаний:

```
syms1 <- c('GOOGL', 'AMZN', 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB',  
          'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')  
# выполнить транспонирование: для кластеризации компаний нам нужны акции вдоль строк  
df <- t(sp500_px[row.names(sp500_px) >= '2011-01-01', syms1])
```

```
d <- dist(df)
hcl <- hclust(d)
```

Алгоритмы кластеризации будут распределять записи (строки) кадра данных по кластерам. Поскольку мы хотим кластеризовать компании, мы должны *транспонировать* (t) кадр данных и поместить акции вдоль строк, а даты — вдоль столбцов.

Пакет `scipy` предлагает ряд разных методов иерархической кластеризации в модуле `scipy.cluster.hierarchy`. Здесь мы используем функцию связи с "полным" (complete) методом:

```
syms1 = ['AAPL', 'AMZN', 'AXP', 'COP', 'COST', 'CSCO', 'CVX', 'GOOGL', 'HD',
         'INTC', 'JPM', 'MSFT', 'SLB', 'TGT', 'USB', 'WFC', 'WMT', 'XOM']
df = sp500_px.loc[sp500_px.index >= '2011-01-01', syms1].transpose()

Z = linkage(df, method='complete')
```

Дендограмма

Иерархическая кластеризация поддается естественному графическому изображению в виде дерева, которое называется *дендограммой*. Это название происходит от греческих слов *dendro* (дерево) и *gramma* (рисунок). В R вы можете легко произвести дендограмму при помощи команды `plot`:

```
plot(hcl)
```

Мы можем использовать метод `dendrogram` для построения графика результата функции `linkage` в Python:

```
fig, ax = plt.subplots(figsize=(5, 5))
dendrogram(Z, labels=df.index, ax=ax, color_threshold=0)
plt.xticks(rotation=90)
ax.set_ylabel('Расстояние')
```

Результат показан на рис. 7.8 (обратите внимание на то, что теперь мы строим график компаний, которые похожи друг на друга, а не график дней). Листья дерева соответствуют записям. Длина ветви в дереве служит индикатором степени несхожести между соответствующими кластерами. Возвратности акций Google и Amazon характерны довольно сильной несхожестью друг от друга и от возвратностей других акций. Нефтяные акции (SLB, CVX, XOM, COP) попадают в свой кластер, Apple (AAPL) расположена сама по себе, и остальные похожи друг на друга.

В противовес с K средними нет необходимости предварительно задавать число кластеров. Графически вы можете идентифицировать разные числа кластеров с помощью горизонтальной линии, которая скользит вверх или вниз; кластер определяется везде, где горизонтальная линия пересекает вертикальные линии. Для извлечения конкретного числа кластеров вы можете воспользоваться функцией `cutree`:

```
cutree(hcl, k=4)
```

GOOGL	AMZN	AAPL	MSFT	CSCO	INTC	CVX	XOM	SLB	COP	JPM	WFC
1	2	3	3	3	3	4	4	4	4	3	3

USB	AXP	WMT	TGT	HD	COST
3	3	3	3	3	3

В Python вы достигаете того же самого с помощью метода `fcluster`:

```
memb = fcluster(Z, 4, criterion='maxclust')
memb = pd.Series(memb, index=df.index)
for key, item in memb.groupby(memb):
    print(f"{key} : {' '.join(item.index)}")
```

Число извлекаемых кластеров устанавливается равным 4, и вы видите, что Google и Amazon каждые в отдельности принадлежат своему собственному кластеру. Все нефтяные акции принадлежат другому кластеру. Оставшиеся акции находятся в четвертом кластере.

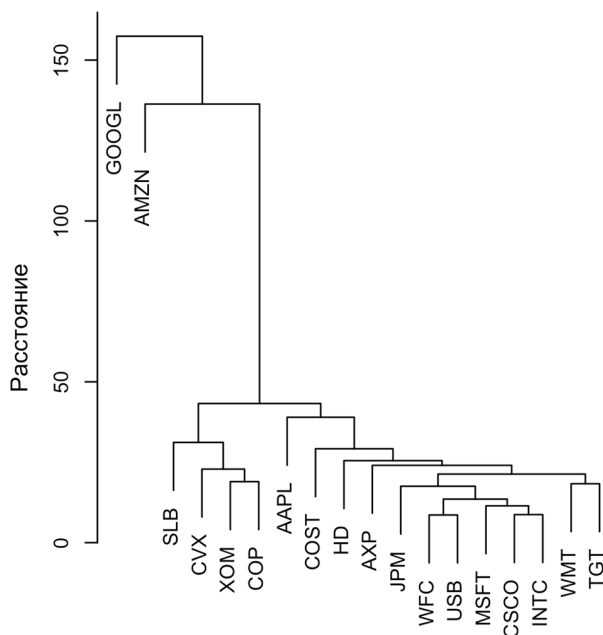


Рис. 7.8. Дендограмма акций

Агломеративный алгоритм

Главным алгоритмом иерархической кластеризации является *агломеративный алгоритм*, который итеративно объединяет похожие кластеры. Этот алгоритм начинает свою работу с того, что делает каждую запись самостоятельным одноэлементным кластером, затем достраивает кластеры все бóльших и бóльших размеров. Первый шаг состоит в вычислении расстояний между всеми парами записей.

Для каждой пары записей (x_1, x_2, \dots, x_p) и (y_1, y_2, \dots, y_p) мы измеряем расстояние между двумя записями $d_{x,y}$, используя метрику расстояния (см. раздел "Метрики расстояния" главы 6).

Например, мы можем использовать евклидово расстояние:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_p - y_p)^2}.$$

Теперь мы переходим к межкластерному расстоянию. Рассмотрим два кластера A и B , каждый с характерным набором записей: $A = (a_1, a_2, \dots, a_m)$ и $B = (b_1, b_2, \dots, b_m)$. Мы можем измерить несхожесть между кластерами $D(A, B)$ путем использования расстояний между членами A и членами B .

Одной из мер несхожести является *метод полной связи*, т. е. максимальное расстояние по всем парам записей между A и B :

$$D(A, B) = \max d(a_i, b_j) \text{ для всех пар } i, j.$$

Эта формула определяет несхожесть как самую большую разницу между всеми парами.

Главные шаги агломеративного алгоритма таковы:

1. Создать первоначальный набор кластеров для всех записей в данных, где каждый кластер состоит из одной-единственной записи.
2. Вычислить несхожесть $D(C_k, C_l)$ между всеми парами кластеров k, l .
3. Объединить два кластера C_k и C_l , которые являются наименее несхожими, что измеряется несхожестью $D(C_k, C_l)$.
4. Если у нас осталось более одного кластера, то вернуться к шагу 2. В противном случае дело сделано.

Меры несхожести

Существует четыре часто встречающиеся меры несхожести: *полная связь*, *одиночная связь*, *средняя связь* и *минимальная дисперсия*. Все перечисленные (плюс другие меры) поддерживаются большинством вычислительных систем иерархической кластеризации, включая пакет `hclust`. Определенный ранее метод полной связи тяготеет к тому, что производит кластеры с членами, которые являются схожими. Метод одиночной связи показывает минимальное расстояние между записями в двух кластерах:

$$D(A, B) = \min d(a_i, b_j) \text{ для всех пар } i, j.$$

Этот метод является "жадным", и он производит кластеры, которые могут содержать довольно разрозненные элементы. Метод средней связи — это среднее всех пар расстояний, он представляет компромисс между методами одиночной и полной связи. Наконец, метод минимальной дисперсии, который также называется *методом Уорда (Ward)*, похож на метод K средних, поскольку он минимизирует внутрикластерную сумму квадратов (см. раздел "Кластеризация на основе K средних" ранее в этой главе).

На рис. 7.8 отражена иерархическая кластеризация с использованием четырех мер для возвратностей акций ExxonMobil и Chevron. Для каждой меры оставлены четыре кластера.

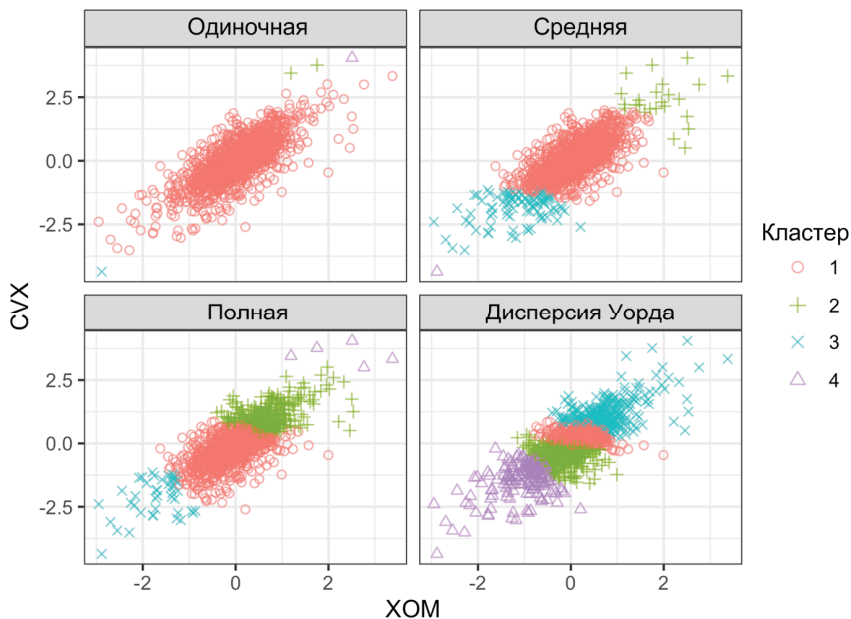


Рис. 7.9. Сравнение мер несхожести применительно к данным об акциях

Результаты являются поразительно разными: мера одиночной связи относит почти все точки в один кластер. За исключением метода минимальной дисперсии (`ward.D` в R; `ward` в Python), все меры заканчивают по крайней мере одним кластером всего с несколькими отдаленными точками (выбросами). Метод минимальной дисперсии наиболее похож на кластер на основе K средних (сравните с рис. 7.4).

Ключевые идеи для иерархической кластеризации

- Иерархическая кластеризация начинается с того, что помещает каждую запись в самостоятельный кластер.
- Кластеры постепенно соединяются с соседними кластерами до тех пор, пока все записи не будут принадлежать одному-единственному кластеру (агломеративный алгоритм).
- История агломерирования сохраняется и наносится на график, и пользователь (без предварительного указания числа кластеров) может наглядно увидеть число и структуру кластеров на разных этапах.
- Межкластерные расстояния вычисляются по-разному, все они опираются на набор всех расстояний между записями.

Модельно-ориентированная кластеризация

Методы кластеризации, такие как иерархическая кластеризация и K средних, базируются на эвристиках и преимущественно опираются на отыскание кластеров, члены которых находятся близко друг к другу, что измеряется непосредственно с помощью данных (вероятностная модель не задействуется). В последние 20 лет значительные усилия были посвящены разработке методов *модельно-ориентированной кластеризации*. Эдриан Рэфтери (Adrian Raftery) и другие исследователи из Вашингтонского университета внесли решающий вклад в методы модельно-ориентированной кластеризации, включая теорию и программно-информационное обеспечение. Указанные технические приемы опираются на статистическую теорию и обеспечивают более строгие способы определения природы и числа кластеров. Они используются, например, в случаях, где может иметься одна группа записей, которые схожи друг с другом, но не обязательно близко расположены друг к другу (например, технологические акции с высокой дисперсией возвратности), и еще одна группа записей, которые схожи и близко расположены (например, акции компаний коммунального хозяйства с низкой дисперсией).

Многомерное нормальное распределение

Наиболее широко используемые методы модельно-ориентированной кластеризации опираются на *многомерное нормальное распределение*. Это обобщение нормально-го распределения на множество из p переменных X_1, X_2, \dots, X_p . Распределение определяется множеством средних $\mu = \mu_1, \mu_2, \dots, \mu_p$ и матрицей ковариаций Σ , которая является мерой того, как переменные коррелируют друг с другом (подробности о ковариации см. в разделе "Матрица ковариаций" главы 5). Матрица ковариаций Σ состоит из p дисперсий $\sigma_1^2, \sigma_2^2, \dots, \sigma_p^2$ и ковариаций $\sigma_{i,j}$ для всех пар переменных $i \neq j$. С переменными, помещенными вдоль строк и продублированными вдоль столбцов, указанная матрица выглядит следующим образом:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,p} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p,1} & \sigma_{p,2} & \cdots & \sigma_p^2 \end{bmatrix}.$$

Обратите внимание, что матрица ковариаций является симметричной относительно своей главной диагонали (от левого верхнего угла до правого нижнего). Поскольку $\sigma_{i,j} = \sigma_{j,i}$, существуют всего $(p \times (p-1))/2$ членов ковариации. В общей сложности матрица ковариаций имеет $(p \times (p-1))/2 + p$ параметров. Распределение обозначается следующей формулой:

$$(X_1, X_2, \dots, X_p) \tilde{N}_p(\mu, \Sigma).$$

Это символический способ сказать, что все переменные нормально распределены, и совокупное распределение полностью описывается вектором средних значений переменных и матрицей ковариаций.

На рис. 7.10 показаны контуры вероятностей для многомерного нормального распределения двух переменных X и Y (контур вероятности 0,5, например, содержит 50% распределения).

Средние составляют $\mu_x = 0,5$ и $\mu_y = -0,5$, и матрица ковариаций имеет вид:

$$\Sigma = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

Поскольку ковариация σ_{xy} является положительной, X и Y коррелируются положительно.

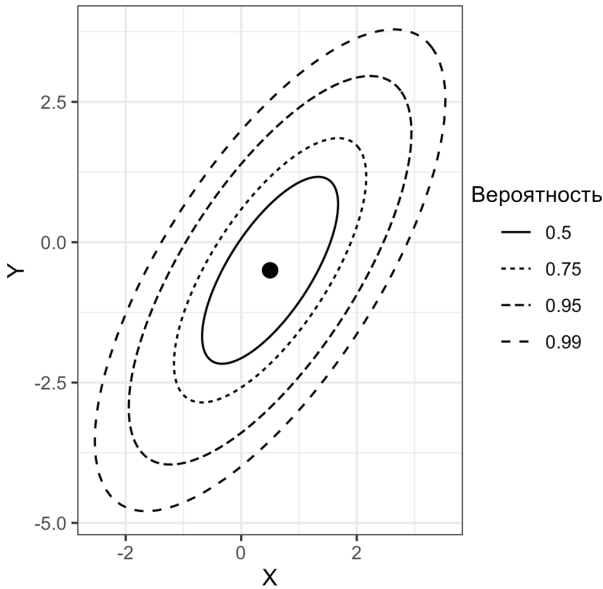


Рис. 7.10. Контуры вероятностей для двухмерного нормального распределения

Смеси нормальных распределений

Ключевая идея в основе модельно-ориентированной кластеризации состоит в том, что каждая запись принимается распределенной, как одно из K многомерных нормальных распределений, где K — это число кластеров. Каждое распределение имеет разное среднее μ и ковариационную матрицу Σ . Например, если имеются две переменные X и Y , тогда каждая строка (X_i, Y_i) моделируется, как отобранная из одного из K многомерных нормальных распределений $(N_1(\mu_1), \Sigma_1)$, $(N_1(\mu_2), \Sigma_2)$, ..., $(N_1(\mu_K), \Sigma_K)$.

Язык R имеет очень насыщенный пакет для модельно-ориентированной кластеризации, который называется `mclust`, первоначально разработанный Крисом Фрейли (Chris Fraley) и Эрианом Рэфтери (Adrian Raftery). С помощью указанного пакета мы можем применить модельно-ориентированную кластеризацию к данным возвратности акций, которые мы ранее проанализировали с использованием K средних и иерархической кластеризации:

```
> library(mclust)
> df <- sp500_px[row.names(sp500_px)>='2011-01-01', c('XOM', 'CVX')]
> mcl <- Mclust(df)
> summary(mcl)
Mclust VEE (ellipsoidal, equal shape and orientation) model with 2 components:
```

```
log.likelihood    n df      BIC      ICL
      -2255.134 1131   9 -4573.546 -5076.856
```

Clustering table:

```
  1  2
963 168
```

Пакет `scikit-learn` имеет класс `sklearn.mixture.GaussianMixture` для модельно-ориентированной кластеризации:

```
df = sp500_px.loc[sp500_px.index >= '2011-01-01', ['XOM', 'CVX']]
mclust = GaussianMixture(n_components=2).fit(df)
mclust.bic(df)
```

Если вы исполните этот фрагмент кода, то заметите, что вычисление занимает значительно больше времени, чем другие процедуры. После извлечения кластерных отнесений посредством функции `predict` мы можем визуализировать кластеры:

```
cluster <- factor(predict(mcl)$classification)
ggplot(data=df, aes(x=XOM, y=CVX, color=cluster, shape=cluster)) +
  geom_point(alpha=.8)
```

Вот код Python для создания похожего рисунка:

```
fig, ax = plt.subplots(figsize=(4, 4))
colors = [f'C{c}' for c in mclust.predict(df)]
df.plot.scatter(x='XOM', y='CVX', c=colors, alpha=0.5, ax=ax)
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
```

Результирующий график показан на рис. 7.11. Имеются два кластера: один кластер — посередине данных, второй кластер — во внешнем крае данных. Это сильно отличается от кластеров, полученных посредством K средних (см. рис. 7.5) и иерархической кластеризации (см. рис. 7.9), которые отыскивают компактные кластеры.

Мы можем извлечь параметры нормальных распределений с помощью функции `summary`:

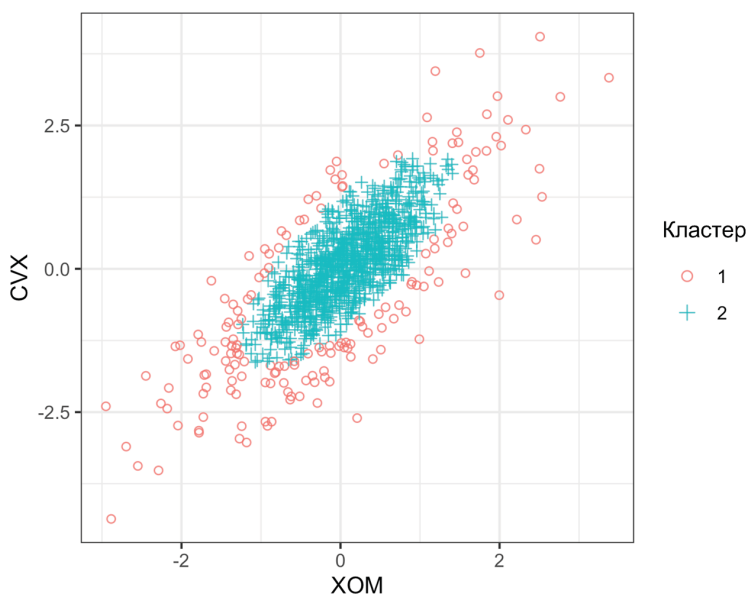


Рис. 7.11. Два кластера, полученные для данных о возвратности акций с помощью `mclust`

```
> summary(mcl, parameters=TRUE)$mean
      [,1]      [,2]
XOM 0.05783847 -0.04374944
CVX 0.07363239 -0.21175715
> summary(mcl, parameters=TRUE)$variance
, , 1
      XOM      CVX
XOM 0.3002049 0.3060989
CVX 0.3060989 0.5496727
, , 2
      XOM      CVX
XOM 1.046318 1.066860
CVX 1.066860 1.915799
```

В Python вы получаете эту информацию из свойств `means_` и `covariances_` результата:

```
print('Среднее')
print(mclust.means_)
print('Ковариации')
print(mclust.covariances_)
```

Распределения имеют схожие средние и корреляции, но у второго распределения намного более крупные дисперсии и ковариации. Из-за случайности алгоритма результаты могут незначительно различаться между разными запусками.

Кластеры из `mclust` могут показаться удивительными, но фактически они иллюстрируют статистическую природу метода. Цель модельно-ориентированной кластеризации состоит в том, чтобы отыскать набор наиболее хорошо вписывающихся

множеств многомерных нормальных распределений. Данные об акциях, по всей видимости, имеют нормальную форму (см. контуры на рис. 7.10). Фактически тем не менее возвратности акций имеют распределение с более длинным хвостом, чем нормальное распределение. Для того чтобы это уладить, `mclust` выполняет подгонку распределения к основной части данных, но затем осуществляет подгонку второго распределения с большей дисперсией.

Выбор числа кластеров

В отличие от K средних и иерархической кластеризации, `mclust` выбирает число кластеров автоматически (в данном случае два). Он делает это путем выбора числа кластеров, для которого *байесов информационный критерий* (bayesian information criteria, BIC) имеет самое крупное значение (байесов информационный критерий похож на информационный критерий Акаике (AIC); см. раздел "*Отбор модели и пошаговая регрессия*" главы 4). Байесов информационный критерий работает путем отбора оптимально подогнанной модели со штрафом за число параметров в модели. В случае модельно-ориентированной кластеризации добавление большего числа кластеров будет всегда улучшать подгонку за счет введения дополнительных параметров в модель.



Обратите в внимание, что в большинстве случаев BIC обычно минимизируется. Авторы пакета `mclust` решили определить BIC с противоположным знаком, чтобы облегчить интерпретацию графиков.

Пакет `mclust` выполняет подгонку 14 разных моделей с увеличивающимся числом компонент и автоматически отбирает оптимальную модель. Вы можете вывести значения BIC на графике для каждого размера кластера, используя функцию `plot` в `mclust`:

```
plot(mcl, what='BIC', ask=FALSE)
```

Число кластеров, или число разных многомерных нормальных моделей (компонент), отмечено по оси x (рис. 7.12).

С другой стороны, имплементация класса `GaussianMixture` в пакете `scikit-learn` не будет опробовать различные комбинации. Как показано далее, в Python легко запусаются несколько комбинаций. Эта имплементация определяет BIC как обычно. Следовательно, рассчитанное значение BIC будет положительным, и нам нужно его минимизировать.

```
results = []
covariance_types = ['full', 'tied', 'diag', 'spherical']
for n_components in range(1, 9):
    for covariance_type in covariance_types:
        mclust = GaussianMixture(n_components=n_components, warm_start=True,
                                covariance_type=covariance_type) ❶
        mclust.fit(df)
```

```

results.append({
    'bic': mclust.bic(df),
    'n_components': n_components,
    'covariance_type': covariance_type,
})

results = pd.DataFrame(results)

colors = ['C0', 'C1', 'C2', 'C3']
styles = ['C0-', 'C1:', 'C0-.', 'C1--']

fig, ax = plt.subplots(figsize=(4, 4))
for i, covariance_type in enumerate(covariance_types):
    subset = results.loc[results.covariance_type == covariance_type, :]
    subset.plot(x='n_components', y='bic', ax=ax, label=covariance_type,
               kind='line', style=styles[i])

```

❶ С помощью аргумента `warm_start` вычисление будет повторно использовать информацию из предыдущей подгонки. Это ускорит схождение последующих расчетов.

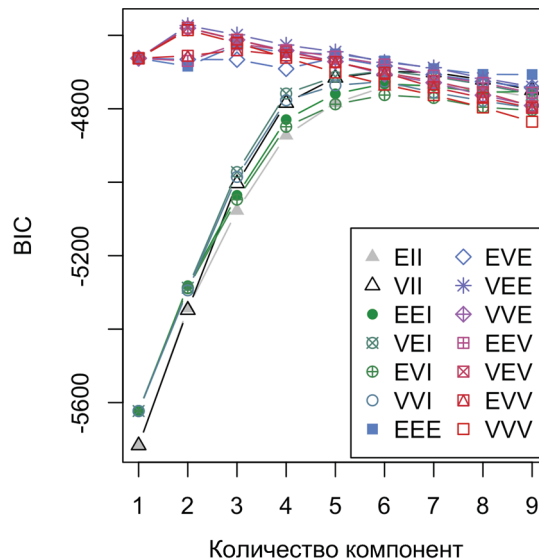


Рис. 7.12. Значения BIC для 14 моделей данных о возвратах акций для разного количества компонент

Указанный график похож на график локтя, используемый для выявления числа кластеров для K средних, за исключением того, что на этом графике выводится значение BIC, а не процент объясненной дисперсии (см. рис. 7.7). Одно большое отличие состоит в том, что вместо одной линии `mclust` показывает 14 разных линий! Это вызвано тем, что `mclust` на самом деле выполняет подгонку 14 разных моделей

для каждого кластерного размера, и в конечном счете он выбирает оптимально подогнанную модель. `GaussianMixture` имплементирует меньше подходов, и поэтому число линий будет только четыре.

Почему пакет `mclust` выполняет подгонку многих моделей для определения наилучшего множества многомерных нормальных распределений? Потому что существуют разные способы параметризации матрицы ковариаций Σ для подгонки модели. В большинстве случаев вам не нужно беспокоиться по поводу деталей моделей, и вы можете спокойно использовать модель, выбранную пакетом `mclust`. В данном примере, согласно BIC, три разные модели (именуемые VEE, VEV и VVE) дают оптимальную подгонку с использованием двух компонент.



Модельно-ориентированная кластеризация является насыщенной и стремительно развивающейся областью исследования, и ее освещение в этом тексте охватывает лишь небольшую ее часть. Действительно, справочный файл пакета `mclust` в настоящее время имеет 154 страниц. Навигация по нюансам модельно-ориентированной кластеризации, пожалуй, требует больше усилий, чем необходимо для большинства задач, которые приходится решать исследователям данных.

Технические приемы модельно-ориентированной кластеризации, правда, имеют несколько ограничений. Указанные методы требуют принятия опорного допущения о модели для данных, и кластерные результаты очень зависят от этого допущения. Ее вычислительные потребности выше, чем даже при иерархической кластеризации, затрудняя масштабирование на крупные данные. Наконец, ее алгоритм сложнее и менее доступен, чем в других методах.

Ключевые идеи для модельно-ориентированной кластеризации

- Считается, что кластеры выводятся из разных генерирующих данных процессов с разными вероятностными распределениями.
- Выполняется подгонка разных моделей с учетом разного числа (в типичной ситуации нормальных) распределений.
- Указанный метод выбирает модель (и связанное с ней число кластеров), которая хорошо вписывается в данные без использования слишком большого числа параметров (т. е. без переподгонки).

Дополнительные материалы для чтения

Подробности о модельно-ориентированной кластеризации см. в документации по пакету `mclust`⁶ и классу `GaussianMixture`⁷.

⁶ См. <https://oreil.ly/bHDvR>.

⁷ См. <https://oreil.ly/GaVVv>.

Шкалирование и категориальные переменные

Технические приемы неконтролируемого самообучения обычно требуют, чтобы данные были соответствующим образом прошкалированы. В этом состоит отличие от многих технических приемов для регрессии и классификации, в которых шкалирование не имеет такой важности (исключением является алгоритм k ближайших соседей; см. раздел " K ближайших соседей" главы 6).

Ключевые термины для шкалирования данных

Шкалирование (scaling)

Сплющивание или расширение данных, обычно для приведения многочисленных переменных к одинаковой шкале измерения.

Нормализация (normalization)

Один из методов шкалирования — вычитание среднего и деление на стандартное отклонение.

Синоним: стандартизация.

Расстояние Говера (Gower's distance)

Алгоритм шкалирования, применяемый к смешанным числовым и категориальным данным для приведения всех переменных к интервалу 0–1.

Например, в случае данных о персональных ссудах переменные имеют самые разные единицы измерения и магнитуды. У некоторых переменных относительно малые значения (например, число используемых лет), в то время как у других — очень большие (например, сумма кредита в долларах). Если данные не прошкалировать, то переменные с большими значениями будут доминировать над анализом главных компонент, K средними и другими методами кластеризации, а переменные с малыми значениями будут ими проигнорированы.

Категориальные данные могут представлять особую проблему для некоторых процедур кластеризации. Как и в случае с k ближайшими соседями, неупорядоченные факторные переменные обычно конвертируются в набор двоичных (0/1) переменных с использованием кодирования с одним активным состоянием (см. раздел "*Кодировщик с одним активным состоянием*" главы 6). Двоичные переменные не только могут быть на шкале измерения, отличающейся от остальных данных, но и тот факт, что двоичные переменные имеют всего два значения, может оказаться проблематичным с такими приемами, как анализ главных компонент и K средних.

Шкалирование переменных

Переменные с совсем другой шкалой и единицами измерения необходимо надлежательно нормализовать перед тем, как вы примените процедуру кластеризации. Например, применим `kmeans` к набору данных о невыплаченных ссудах без нормализации:

```
defaults <- loan_data[loan_data$outcome=='default',]
df <- defaults[, c('loan_amnt', 'annual_inc', 'revol_bal', 'open_acc',
                  'dti', 'revol_util')]
km <- kmeans(df, centers=4, nstart=10)
centers <- data.frame(size=km$size, km$centers)
round(centers, digits=2)
```

	size	loan_amnt	annual_inc	revol_bal	open_acc	dti	revol_util
1	52	22570.19	489783.40	85161.35	13.33	6.91	59.65
2	1192	21856.38	165473.54	38935.88	12.61	13.48	63.67
3	13902	10606.48	42500.30	10280.52	9.59	17.71	58.11
4	7525	18282.25	83458.11	19653.82	11.66	16.77	62.27

Вот соответствующий код на Python:

```
defaults = loan_data.loc[loan_data['outcome'] == 'default',]
columns = ['loan_amnt', 'annual_inc', 'revol_bal', 'open_acc',
          'dti', 'revol_util']
df = defaults[columns]
kmeans = KMeans(n_clusters=4, random_state=1).fit(df)
counts = Counter(kmeans.labels_)

centers = pd.DataFrame(kmeans.cluster_centers_, columns=columns)
centers['size'] = [counts[i] for i in range(4)]
centers
```

Переменные `annual_inc` и `revol_bal` доминируют над кластерами, и кластеры имеют очень разные размеры. Кластер 1 имеет всего 55 членов со сравнительно высоким сальдо годового дохода и возобновляемого кредита.

Часто встречающийся подход к шкалированию переменных состоит в их конвертировании в *z*-оценки путем вычитания среднего значения и деления на стандартное отклонение. Указанная процедура называется *стандартизацией* или *нормализацией* (более подробное обсуждение темы применения *z*-оценок см. в разделе "*Стандартизация (нормализация, z-оценки)*" главы 6):

$$z = \frac{x - \bar{x}}{s}.$$

Посмотрим, что произойдет с кластерами, когда `kmeans` применяется к нормализованным данным:

```
df0 <- scale(df)
km0 <- kmeans(df0, centers=4, nstart=10)
centers0 <- scale(km0$centers, center=FALSE,
                 scale=1 / attr(df0, 'scaled:scale'))
centers0 <- scale(centers0, center=attr(df0, 'scaled:center'), scale=FALSE)
centers0 <- data.frame(size=km0$size, centers0)
round(centers0, digits=2)
  size loan_amnt annual_inc revol_bal open_acc dti revol_util
```


1	7355	10467.65	51134.87	11523.31	7.48	15.78	77.73
2	5309	10363.43	53523.09	6038.26	8.68	11.32	30.70
3	3713	25894.07	116185.91	32797.67	12.41	16.22	66.14
4	6294	13361.61	55596.65	16375.27	14.25	24.23	59.61

В Python мы можем использовать класс `StandardScaler` пакета `scikit-learn`. Метод `inverse_transform` позволяет конвертировать центры кластеров обратно в исходную шкалу:

```
scaler = preprocessing.StandardScaler()
df0 = scaler.fit_transform(df * 1.0)

kmeans = KMeans(n_clusters=4, random_state=1).fit(df0)
counts = Counter(kmeans.labels_)

centers = pd.DataFrame(scaler.inverse_transform(kmeans.cluster_centers_),
                       columns=columns)

centers['size'] = [counts[i] for i in range(4)]
centers
```

Размеры кластеров являются более сбалансированными, обе переменные, `annual_inc` и `revol_bal`, не доминируют над кластерами, которые раскрывают более интересную структуру в данных. Отметим, что в приведенном выше фрагменте кода центры перешкалированы в исходные единицы измерения. Если оставить их не шкалированными, то результирующие значения будут в z-оценках и поэтому менее интерпретируемыми.



Шкалирование также имеет большое значения для анализа главных компонент. Применение z-оценок эквивалентно использованию матрицы корреляций (см. раздел *"Корреляция"* главы 1) вместо матрицы ковариаций в вычислениях главных компонент. Вычислительные системы при имплементировании анализа главных компонент обычно имеют опцию использования матрицы корреляций (в R функция `princomp` имеет аргумент `cor`).

Доминантные переменные

Даже в случаях, где переменные измеряются на одинаковой шкале и точно отражают относительную важность (например, динамику курсов акций), иногда бывает полезным выполнить перешкалирование переменных.

Предположим, что мы добавляем в анализ из раздела *"Интерпретирование главных компонент"* ранее в этой главе акции Google (GOOGL) и Amazon (AMZN).

```
syms <- c('GOOGL', 'AMZN', 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM',
          'SLB', 'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')
top_sp1 <- sp500_px[row.names(sp500_px) >= '2005-01-01', syms]
sp_pca1 <- princomp(top_sp1)
screepplot(sp_pca1)
```

На языке Python мы получаем график каменной осыпи следующим образом:

```
syms = ['GOOGL', 'AMZN', 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM',  
        'SLB', 'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST']  
top_sp1 = sp500_px.loc[sp500_px.index >= '2005-01-01', syms]  
  
sp_pca1 = PCA()  
sp_pca1.fit(top_sp1)  
  
explained_variance = pd.DataFrame(sp_pca1.explained_variance_)  
ax = explained_variance.head(10).plot.bar(legend=False, figsize=(4, 4))  
ax.set_xlabel('Компонента')
```

График каменной осыпи изображает дисперсии для верхних главных компонент. В данном случае указанный график на рис. 7.13 раскрывает, что дисперсии первой и второй компонент намного крупнее других. Такая ситуация часто говорит о том, что одна или две переменные доминируют над нагрузками. В данном примере это как раз тот случай:

```
round(sp_pca1$loadings[,1:2], 3)  
      Comp.1 Comp.2  
GOOGL  0.781  0.609  
AMZN   0.593 -0.792  
AAPL   0.078  0.004  
MSFT   0.029  0.002  
CSCO   0.017 -0.001  
INTC   0.020 -0.001  
CVX    0.068 -0.021  
XOM    0.053 -0.005  
...
```

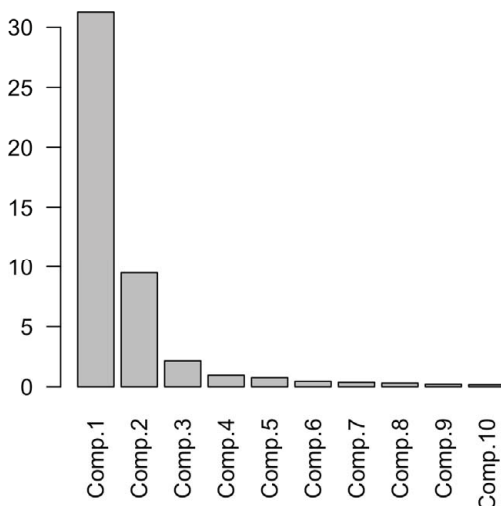


Рис. 7.13. График каменной осыпи для анализа главных компонент возвратности акций из S&P 500, включая GOOGL и AMZN

На языке Python мы используем следующее:

```
loadings = pd.DataFrame(sp_pca1.components_[0:2, :], columns=top_sp1.columns)
loadings.transpose()
```

Первые две главные компоненты почти полностью доминируются акциями GOOGL и AMZN. Это вызвано тем, что динамика курса акций GOOGL и AMZN доминирует над вариабельностью.

Для того чтобы уладить эту ситуацию, вы можете либо включить их в анализ как есть, перешкалировать переменные (см. раздел "*Шкалирование переменных*" ранее в этой главе), либо исключить доминантные переменные из анализа и уладить их отдельно. "Правильного" подхода не существует, и трактовка зависит от применения.

Категориальные данные и расстояние Говера

В случае категориальных данных вы должны конвертировать их в числовые данные путем ранжирования (для упорядоченного фактора) либо путем кодирования как множества двоичных (фиктивных) переменных. Если данные будут состоять из смешанных непрерывных и двоичных переменных, вам потребуется перешкалировать переменные так, чтобы их интервалы были схожими (см. раздел "*Шкалирование переменных*" ранее в этой главе). Один из популярных методов состоит в использовании *расстояния Говера*.

Главная идея, лежащая в основе расстояния Говера, состоит в том, чтобы применять разные метрики расстояния к каждой переменной в зависимости от типа данных:

- ◆ для числовых переменных и упорядоченных факторов расстояние вычисляется как абсолютное значение разницы между двумя записями (*манхэттенское расстояние*);
- ◆ для категориальных переменных расстояние равняется 1, если категории между двумя записями различаются, и 0, если категории одинаковые.

Расстояние Говера вычисляется следующим образом:

1. Вычислить расстояние $d_{i,j}$ для всех пар переменных i и j для каждой записи.
2. Прошкалировать каждую пару $d_{i,j}$ так, чтобы минимум равнялся 0, и максимум равнялся 1.
3. Соединить вместе попарно прошкалированные расстояния между переменными, используя простое либо взвешенное среднее, для создания матрицы расстояний.

В качестве иллюстрации расстояния Говера возьмем несколько строк из данных о судах в R:

```
> x = defaults[1:5, c('dti', 'payment_inc_ratio', 'home', 'purpose')]
> x
# Таблица: 5 × 4
      dti payment_inc_ratio home           purpose
```

	<dbl>	<dbl>	<fctr>	<fctr>
1	1.00	2.39320	RENT	car
2	5.55	4.57170	OWN	small_business
3	18.08	9.71600	RENT	other
4	10.08	12.21520	RENT	debt_consolidation
5	7.06	3.90888	RENT	other

Функция `daisy` в пакете `cluster` языка R может использоваться для вычисления расстояния Говера:

```
> library(cluster)
> daisy(x, metric='gower')
Dissimilarities :
      1      2      3      4
2 0.6220479
3 0.6863877 0.8143398
4 0.6329040 0.7608561 0.4307083
5 0.3772789 0.5389727 0.3091088 0.5056250
```

```
Metric : mixed ; Types = I, I, N, N
Number of objects : 5
```

На момент написания этой главы расстояние Говера недоступно ни в одном из популярных пакетов Python. Тем не менее продолжают мероприятия по включению его в пакет `scikit-learn`. Мы обновим сопровождающий книгу исходный код, как только его имплементация будет выпущена.

Все расстояния находятся между 0 и 1. Пара записей с наибольшим расстоянием равняется 2 и 3: ни одна не имеет одинаковых значений для `home` (дом) или `purpose` (цель), и у них совсем разные уровни `dti` (отношение долга к доходу) и `payment_inc_ratio` (отношение платежей к доходу). Записи 3 и 5 имеют наименьшее расстояние, потому что они делят между собой одинаковые значения для `home` и `purpose`.

Вы можете передать матрицу расстояний Говера, вычисленную из функции `daisy`, к `hclust` для иерархической кластеризации (см. раздел "Иерархическая кластеризация" ранее в этой главе):

```
df <- defaults[sample(nrow(defaults), 250),
                  c('dti', 'payment_inc_ratio', 'home', 'purpose')]
d = daisy(df, metric='gower')
hcl <- hclust(d)
dnd <- as.dendrogram(hcl)
plot(dnd, leaflab='none')
```

Результирующая дендограмма показана на рис. 7.14. Индивидуальные записи неразличимы на оси *x*, но мы можем обрезать дендограмму горизонтально на уровне 0,5 и проэкзаменовать записи в одном из поддеревьев с помощью вот такого фрагмента кода:

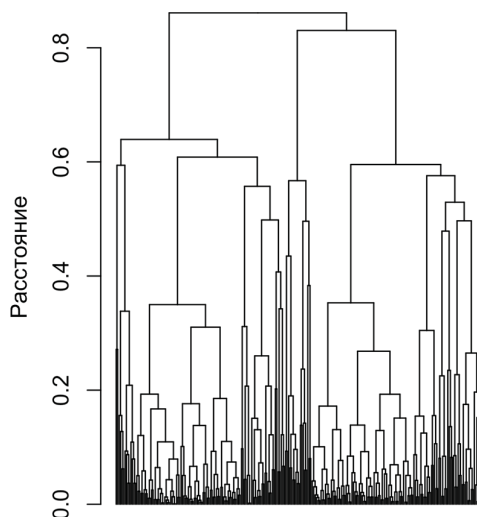


Рис. 7.14. Дендрограмма `hclust` применительно к выборке данных о невыплаченных ссудах со смешанными типами переменных

```
dnd_cut <- cut(dnd, h=0.5)
df[labels(dnd_cut$lower[[1]]),]
      dti payment_inc_ratio home_ purpose_
44532 21.22         8.37694   OWN debt_consolidation
39826 22.59         6.22827   OWN debt_consolidation
13282 31.00         9.64200   OWN debt_consolidation
31510 26.21        11.94380   OWN debt_consolidation
6693  26.96         9.45600   OWN debt_consolidation
7356  25.81         9.39257   OWN debt_consolidation
9278  21.00        14.71850   OWN debt_consolidation
13520 29.00        18.86670   OWN debt_consolidation
14668 25.75        17.53440   OWN debt_consolidation
19975 22.70        17.12170   OWN debt_consolidation
23492 22.68        18.50250   OWN debt_consolidation
```

Это поддерево целиком состоит из владельцев с целевым назначением получения ссуды, помеченной как `debt_consolidation` (консолидация долга). Хотя строгое разделение не относится ко всем поддеревам, это показывает, что категориальные переменные тяготеют к группированию в кластеры.

Проблемы кластеризации смешанных данных

K средних и анализ главных компонент лучше всего подходят для непрерывных переменных. Для меньших наборов данных лучше использовать иерархическую кластеризацию с расстоянием Говера. В принципе нет никакой причины, почему *K* средних не могут применяться к двоичным или категориальным данным. Вы обычно будете использовать представление в виде "кодирования с одним активным состоянием" (см. раздел "Кодировщик с одним активным состоянием" главы 6) для

конвертирования категориальных данных в числовые значения. На практике, однако, бывает трудно использовать алгоритм *K средних* и анализ главных компонент с двоичными данными.

Если используются стандартные *z*-оценки, то двоичные переменные будут доминировать над определением кластеров. Это вызвано тем, что переменные 0/1 принимают всего два значения, и *K средних* могут получить малую внутрикластерную сумму квадратов за счет отнесения всех записей с 0 или 1 к одному-единственному кластеру. Например, давайте применим `kmeans` к данным о невыплаченных ссудах, включив факторные переменные `home` и `pub_rec_zero`, на языке R:

```
df <- model.matrix(~ -1 + dti + payment_inc_ratio + home_ + pub_rec_zero,
                  data=defaults)
df0 <- scale(df)
km0 <- kmeans(df0, centers=4, nstart=10)
centers0 <- scale(km0$centers, center=FALSE,
                 scale=1/attr(df0, 'scaled:scale'))
round(scale(centers0, center=-attr(df0, 'scaled:center'), scale=FALSE), 2)
```

	dti	payment_inc_ratio	home_MORTGAGE	home_OWN	home_RENT	pub_rec_zero
1	17.20	9.27	0.00	1	0.00	0.92
2	16.99	9.11	0.00	0	1.00	1.00
3	16.50	8.06	0.52	0	0.48	0.00
4	17.46	8.42	1.00	0	0.00	1.00

На языке Python:

```
columns = ['dti', 'payment_inc_ratio', 'home_', 'pub_rec_zero']
df = pd.get_dummies(defaults[columns])

scaler = preprocessing.StandardScaler()
df0 = scaler.fit_transform(df * 1.0)
kmeans = KMeans(n_clusters=4, random_state=1).fit(df0)
centers = pd.DataFrame(scaler.inverse_transform(kmeans.cluster_centers_),
                      columns=df.columns)

centers
```

Четыре верхних кластера являются, по существу, эрзацами для разных уровней факторных переменных. Во избежание такого поведения вы можете прошкалировать двоичные переменные и получить дисперсию меньше, чем у других переменных. В качестве альтернативы, в случае очень крупных наборов данных, вы можете применить кластеризацию к разным подмножествам данных, принимающим конкретные категориальные значения. Например, вы можете применить кластеризацию отдельно к тем ссудам, выданным физическим лицам, которые имеют ипотеку, владеют домом напрямую или его арендуют.

Ключевые идеи для шкалирования данных

- Переменные, измеряемые на разных шкалах, должны быть приведены к схожим шкалам с тем, чтобы их влияние на алгоритмы не определялось главным образом их шкалой измерения.
- Общепринятым методом шкалирования является нормализация (стандартизация) — вычитание среднего значения и деление на стандартное отклонение.
- Еще один метод — расстояние Говера — шкалирует все переменные, приводя к интервалу 0–1 (он часто используется со смешанными числовыми и категориальными данными).

Резюме

Для уменьшения размерности числовых данных главными инструментами являются анализ главных компонент либо кластеризация по методу *K* средних. Оба алгоритма требуют уделять внимание надлежащему шкалированию данных с целью обеспечения содержательного уменьшения данных.

В случае кластеризации с очень структурированными данными, в которых кластеры хорошо разделены, все методы, по всей вероятности, будут приводить к схожему результату. Каждый метод предлагает собственное преимущество. *K* средних масштабируется до очень крупных данных и понятен. Иерархическая кластеризация может применяться к смешанным типам данных — числовым и категориальным — и поддается интуитивно понятному изображению (дендограмма). Модельно-ориентированная кластеризация основана на статистической теории и обеспечивает более строгий подход в противоположность эвристическим методам. В случае очень крупных данных, однако, *K* средних является главным используемым методом.

С шумными данными, такими как данные о ссудах и акциях (и подавляющая часть данных, с которыми будет сталкиваться исследователь данных), выбор является еще более спартанским. *K* средних, иерархическая кластеризация, и в особенности модельно-ориентированная кластеризация, — все эти методы будут порождать весьма разные решения. Как должен поступить исследователь данных? К сожалению, нет никакого простого эмпирического правила, которое могло бы помочь сделать выбор. В конечном счете используемый метод будет зависеть от размера данных и цели применения.

Библиография

- [**Baumer-2017**] Baumer B., Kaplan D., Horton N. Modern Data Science with R. — Boca Raton, Fla.: Chapman & Hall/CRC Press, 2017. (Баумер Б., Каплан Д., Хортон Н. Современная наука о данных с помощью R.)
- [**bokeh**] Bokeh: Python library for interactive visualization. URL: <http://www.bokeh.pydata.org>. (Bokeh: библиотека Python для интерактивной визуализации (группа разработчиков пакета Bokeh).)
- [**Deng-Wickham-2011**] Deng H., Wickham H. Density estimation in R. URL: https://oreil.ly/-Ny_6. (Дэнг Х., Уикхэм Х. Оценивание плотности в R.)
- [**Donoho-2015**] Donoho D. 50 Years of Data Science. URL: <https://oreil.ly/kqFb0>. (Донохо Д. 50 лет науки о данных.)
- [**Duong-2001**] Duong T. An introduction to kernel density estimation. URL: <https://oreil.ly/Z5A7W>. (Дуанг Т. Введение в ядерную оценку плотности.)
- [**Few-2007**] Few S. Save the pies for dessert // Visual Intelligence Newsletter. Perceptual Edge. — 2007. URL: https://oreil.ly/_iGAL. (Фью С. Оставьте пирожки на десерт.)
- [**Freedman-2007**] Freedman D., Pisani R., Purves P. Statistics. — 4th ed. — New York: W. W. Norton, 2007. (Фридман Д., Пизани Р., Парвс П. Статистика.)
- [**Hintze-Nelson-1998**] Hintze J., Nelson R. Violin plots: a box plot-density trace synergism // The American Statistician. — 1998. — May. — P. 181–184. (Хинтзе Дж., Нельсон Р. Скрипичные графики: синергия коробчатых диаграмм и трассировки плотности.)
- [**Galton-1886**] Galton F. Regression towards mediocrity in Hereditary stature // J. Anthropol. Instit. Great Britain and Ireland. — № 5. — P. 246–273. URL: <https://oreil.ly/DqoAk>. (Гальтон Ф. Регрессия к посредственности в Наследственном статусе.)
- [**ggplot2**] Wickham H. ggplot2: Elegant graphics for data analysis. — Springer-Verlag New York, 2009. ISBN: 978-0-387-98140-6. URL: <https://oreil.ly/O92vC>. (Уикхэм Х. Изящная графика для анализа данных.)
- [**Hyndman-Fan-1996**] Hyndman R. J., Fan Y. Sample quantiles in statistical packages // American Statistician. — 1996. — № 50. — P. 361–365. (Хайндман Р. Дж., Фан Я. Выборочные квантили в статистических пакетах.)
- [**lattice**] Sarkar D. Lattice: Multivariate Data Visualization with R. — Springer, 2008, ISBN 978-0-387-75968-5. URL: <http://lmdvr.r-forge.r-project.org>. (Саркар Д. Решетка: визуализация многомерных данных вместе с помощью R.)

- [**Legendre**] Legendre A.-M. Nouvelle methodes pour la determination des orbites des cometes. — F. Didot, Paris, 1805. URL: <https://oreil.ly/8FITJ>. (Лежандр А.-М. Новые методы определения орбит комет.)
- [**NIST-Handbook-2012**] Measures of Skewness and Kurtosis / NIST/SEMATECH e-Handbook of Statistical Methods, 2012. URL: <https://oreil.ly/IAdHA>. (Меры асимметрии и эксцесса.)
- [**R-base-2015**] R: A Language and Environment for Statistical Computing R Foundation for Statistical Computing (2015). URL: <http://www.R-project.org/>. (R: язык и среда для статистических расчетов (рабочая группа по языку R).)
- [**Salsburg-2001**] Salsburg D. The lady tasting tea: how statistics revolutionized science in the twentieth century. — New York: W. H. Freeman, 2001. (Салсбург Д. Леди дегустирует чай: как статистика революционизировала науку в двадцатом веке.)
- [**seaborne**] Wasdom M. Seaborn: statistical data visualization. URL: <https://seaborn.pydata.org>. (Васдом М. Seaborn: визуализация статистических данных.)
- [**Trellis-Graphics**] Becker R., Cleveland W., Shyu M., Kaluzny S. A Tour of Trellis Graphics. — 1996. URL: <https://oreil.ly/LVnOV>. (Бэкер Р., Кливленд У., Шу М., Калюжный С. Обзор Trellis Graphics.)
- [**Tukey-1962**] Tukey J. W. The future of data analysis // Ann. Math. Statist. — 1962. — Vol. 33. — N 1. — P. 1–67. URL: <https://oreil.ly/qrYNW>. (Тьюки Дж. У. Будущее анализа данных.)
- [**Tukey-1977**] Tukey J. W. Exploratory data analysis. — Pearson, 1977. — 712 p. ISBN: 978-0-201-07616-5. (Тьюки Дж. У. Разведывательный анализ данных.)
- [**Tukey-1987**] Tukey J. W. The collected works of John W. Tukey: philosophy and principles of data analysis 1965–1986: Vol. IV / ed. by L. V. Jones. — Chapman and Hall/CRC, 1987. (Тьюки Дж. У. Собрание работ Джона У. Тьюки: философия и принципы анализа данных 1965–1986.)
- [**Zhang-Wang-2007**] Zhang Q., Wang W. 19th International Conference on Scientific and Statistical Database Management. — IEEE Computer Society, 2007. URL: <https://oreil.ly/qShjk>. (Быстрый алгоритм для приближенных квантилей в высокоскоростных потоках данных. 19-я международная конференция по вопросам управления научными и статистическими базами данных, IEEE Computer Society.)

Предметный указатель

A

Adaboost, алгоритм бустинга 284
ASA (American Statistical Association) 125
AUC (area under the ROC curve) 240

B

BIC (байесов информационный критерий)
171

C

CP Мэллоуза 171

D

d.f. 133

E

EDA 19

F

F-статистика 134, 138, 169

H

Hat-значение 191, 194

K

k ближайших соседей (KKN) 252, 253
◊ выбор k 260
◊ метрики расстояния 255
◊ регрессия 262
K средних 307, 310
к-блочная перекрестная проверка 169

N

n (размер выборки) 133

P

Principal components analysis (PCA) 298
PR-кривая 239
p-значение 120, 123
◊ и t-статистика 169

R

ROC-кривая 233, 238
RSE 167
R-квадрат 164, 168
◊ скорректированный 168

S

SMOTE, 246
SS (сумма квадратов)
◊ внутрикластерная 307

T

t-распределение 127
t-статистика 127, 164, 168

X

XGBoost 286
◊ гиперпараметры 294

Z

z-оценка 87, 89, 243, 252, 257
◊ перешкалирование переменных 259

А

Алгоритм:

- ◇ агломеративный 318
- ◇ жадный 149, 319
- ◇ эpsilon-жадный 149
- Альфа-уровень 120, 124
- ◇ деление в множественном тестировании 130

Анализ:

- ◇ главных компонент 298, 305
 - против кластерного анализа 312
- ◇ данных разведывательный 19
- ◇ двумерный 53, 54
- ◇ дискриминантный 215
 - квадратичный 219
 - расширения 219
- ◇ дисперсионный (ANOVA) 134
 - двусторонняя процедура 139
 - проверка на основе F-статистики 138
 - разложение дисперсии 139
- ◇ линейный дискриминантный (LDA) 216, 247
- ◇ одномерный 53, 54
- ◇ остатков 231
- ◇ соответствия 305
- Аномалия 30
- Ансамбль 273, 284
- ◇ моделей 273
- Асимметрия 42, 91

Б

- Байес наивный 210
- Балл 172, 220
- ◇ вероятности 209
 - ◇ задание новым данным 221, 226, 232
 - ◇ заемщика 217
- Бандит многорукий 147
- Бета-распределение 150
- Блок 293
- ◇ в перекрестном контроле 169, 170
- Брейман, Лео 251
- Бритва Оккама 170
- Бустинг 252, 263, 283, 284
- ◇ алгоритм бустирования 285
 - ◇ градиентный 284, 285
 - ◇ стохастический градиентный 284
- Бутстрап 80, 83–85, 87, 114
- Бэггинг 83, 114, 252, 273, 274

В

Важность:

- ◇ переменных 273
- ◇ разрежений 109
- Вариабельность 32
- ◇ выборочная 76
- Вариант эксперимента 106
- Веб-тестирование, бандитские алгоритмы 148
- Вектор признаков 159
- Вероятность:
 - ◇ апостериорная 210, 214
 - ◇ условная 210
- Вес 26, 157
- ◇ дискриминантный 215
- ◇ нагрузка компоненты 298
- Взаимодействие 184, 188
- ◇ включение в модель 190
- Выборка 66
- ◇ бутстраповская 80
 - повторная 79
- ◇ проверочная 233
- ◇ простая случайная 66
- ◇ размер 93
- ◇ смещенная 66, 67
- Выброс 26, 29, 191
- ◇ в регрессии 191
- ◇ чувствительность наименьших квадратов 162
- Вывод статистический 105
- ◇ классический конвейер 105
- Выигрыш 147

Г

- Генерация данных 243, 246
- Гетероскедастичность 191, 196
- Гиперпараметр 282, 284
- ◇ в HGBost 294
 - темп усвоения (фактор сжатия) 294
- Гипотеза:
- ◇ альтернативная 111, 112
 - ◇ двусторонняя 113
 - ◇ нулевая 111, 112
- Гистограмма 38, 41, 46
- Грань 60
- Граф 25
- График 25
- ◇ влияния 194
 - ◇ каменистой осыпи 298, 302

- ◇ квантиль-квантильный 87
- ◇ контурный 53
- ◇ плотности 38, 42
- ◇ прироста:
 - кумулятивного 241
 - подецильного 241
- ◇ пузырьковый 194
- ◇ с сеткой из шестиугольных корзин 53, 54
- ◇ скрипичный 53, 58
- ◇ частных остатков 191, 199, 203
 - в логистической регрессии 231
- Группа:
 - ◇ контрольная 106, 108
 - ◇ тестовая 106

Д

- Данные:
 - ◇ большие:
 - выбросы в регрессии 193
 - использование в регрессии 163
 - ◇ двоичные 20
 - ◇ дискретные 20
 - ◇ категориальные 20, 21, 328
 - ◇ количественные, проверочный метрический показатель 107
 - ◇ непрерывные 20
 - непрерывная переменная с проверочной метрикой 107
 - ◇ порядковые 20, 21
 - ◇ прямоугольные 23
 - ◇ шкалированные 328
- Дендограмма 316, 317
- ◇ акций 317
- Дерево:
 - ◇ градиентно-бустированное 190
 - ◇ решений 82, 251
 - значение в исследовании операций 264

Диаграмма:

- ◇ коробчатая 38, 39, 58
 - отображение выбросов 192
- ◇ круговая 45
- ◇ рассеяния 48, 52, 54
- ◇ столбиковая 45
- Дискриминант Фишера линейный 217
- Дисперсия 32, 34, 261
- ◇ анализ (ANOVA) 134
- ◇ стандартная, чувствительность к выбросам 34

Добыча регулярностей из данных 73, 75, 129, 131, 132, 272

З

- Загрязненность 263
- ◇ Джини 269
- Запись 23, 157
- Значение:
 - ◇ в точке ветвления 263
 - ◇ влиятельное 191, 193
 - ◇ ожидаемое 45, 47
 - ◇ подогнанное 161
 - вписанное 157
 - ◇ предсказанное 161, 225
- Значимость статистическая 115, 120

И

- Индекс 24
- Интенсивность ложноположительных исходов 241
- Интервал:
 - ◇ доверительный 84, 87, 176
 - против интервала предсказательного 177
 - ◇ предсказательный 175, 176
 - против интервала доверительного 177
- Инфляция альфы 130
- Исключение обратное 173
- Испытание 95, 96
 - ◇ биномиальное 96
- Испытуемый 106
- Исследование статистическое:
 - ◇ двойное слепое 108
 - ◇ слепое 108
- Исход 23
 - ◇ биномиальный 96

К

- Кадр данных 23
- Квадрат наименьший 157
- Квантиль 35
- Классификатор байесовый 214
- Классификация 209
 - ◇ более двух возможных исходов 210
 - ◇ наивный Байес 210
 - ◇ наивный байесовый классификатор
 - непрактичность точной байесовой классификации 211

Классификация (прод.)

- ◇ неконтролируемое самообучение как структурный элемент 297
- ◇ оценивание моделей 233
 - ROC-кривая 238
- ◇ стоимостная 247
- ◇ стратегии для несбалансированных данных 243
- Кластер 307
- Кластеризация 297, 307
 - ◇ К средних 328
 - ◇ иерархическая 315, 333
 - ◇ кластерный анализ против анализа главных компонент 312
 - ◇ модельно-ориентированная 321
 - ◇ применение в задачах холодного старта 297
 - ◇ стандартизация данных 257
- Кликабельность 153
- Ковариация 215, 216, 301
- Кодирование:
 - ◇ девиационное 178, 181
 - ◇ опорное 178, 179, 189, 224
 - ◇ полиномиальное 181
 - ◇ с одним активным состоянием 256
- Кодировщик с одним активным состоянием 178
- Компонент главный 298
- Компромисс между смещением и дисперсией 261
- Кондиционность 60
- Контраст сумм 181
- Контроль перекрестный 169, 261, 293
 - ◇ для отбора главных компонент 304
- Корень из среднеквадратической ошибки 164, 167
- Корректировка р-значения 130
- Корреляция 157
 - ◇ Кенделла 51
 - ◇ Спирмена 51
- Коэффициент:
 - ◇ в простой линейной регрессии, оценки против известных данных 161
 - ◇ детерминации 164, 168
 - ◇ Джини 269
 - ◇ интерпретация в множественной линейной регрессии 166
 - ◇ корреляции 48, 49, 216
 - Пирсона 49
 - ◇ ложных открытий 130, 131

- ◇ регрессии 157
 - в примере с воздействием хлопка на объем легких 159
- ◇ угловой 157
- Кривая лифта 241
- Критерий
 - ◇ байесовый информационный (BIC) 171, 325
 - ◇ информационный Акаике (AIC) 171

Л

- Лебедь черный 91
- Лес случайный 190, 263, 273, 275
- Лист 263
- Лифт 233, 241
- Логарифм:
 - ◇ перевесов 221, 227
 - ◇ шансов 221, 223
- Логит 221, 222
 - ◇ обратный 222
- ◇ преобразование 221
- Лямбда 100, 101

М

Матрица:

- ◇ ковариаций 216
 - в модельно-ориентированной кластеризации 321
 - применение для вычисления расстояния Махаланобиса 256
- ◇ корреляционная 49
- ◇ неточностей 233
- ◇ ошибок 233
- ◇ путаницы 233, 234
- Медиана 26, 29
 - ◇ взвешенная 26, 29

Метод:

- ◇ диагностики и настройки моделей внутривыборочный 173
- ◇ локтя 313
- ◇ минимальной дисперсии 319
- ◇ наименьших квадратов обычный 157
- ◇ связи:
 - одиночной 319
 - полной 319
 - средней 319
- ◇ Уорда 319

Метрика 27

- ◇ расстояния 252, 255
 - в k ближайших соседей 256

Мода 45, 46

Модель:

- ◇ древесная 190, 247, 263, 264
- ◇ обобщенная аддитивная 201, 206, 247
- ◇ обобщенная линейная 223, 225
- ◇ регрессионная 175
- Мошенничество в науке, обнаружение 145
- Мощность 151, 152
- Мультиколлинеарность 184, 186
 - ◇ проблемы кодирования с одним активным состоянием 257

Н

Нагрузки компонент 298, 299

Наклон 157

Несхожесть 316

Нормализация 89, 252, 257, 307, 328, 329

- ◇ баз данных 257
- ◇ в статистике против контекста баз данных 257
- ◇ и распределение данных 260
- ◇ категориальные переменные перед кластеризацией 328

О

Обнаружение аномалий 157

Обучение:

- ◇ ансамблевое 251
- ◇ машинное 251
 - статистическое 251, 252, 255

Однородность 263

Ожидание 140

- ◇ математическое 45, 47

Остаток 157, 161

◇ ненормальный 191

◇ Пирсона 141

◇ стандартизованный 191

- обследование с целью обнаружения выбросов 191

Отбор 65

- ◇ без возврата 66, 114
- ◇ обратный 173
- ◇ признаков:
 - дискриминантный анализ 218
 - проверка хи-квадрат 146
- ◇ по методу Томпсона 150

- ◇ повторный 80, 84, 114

- ◇ повышающий 243, 245

- ◇ понижающий 243, 244

- ◇ прямой и отбор обратный 173

- ◇ с возвратом 66, 81, 114

- ◇ случайный 66, 69

- ◇ стратифицированный 66, 70

Отказ 102

Отклик 157, 159

- ◇ логистический 222

- ◇ связь между ним и предсказательной переменной 199

Отклонение 32, 33

- ◇ медианное абсолютное от медианы 32, 35

- ◇ среднее абсолютное 32, 33

- ◇ стандартное 32, 34, 80

- чувствительность к выбросам 34

Отношение:

- ◇ "сигнал/шум" 260

- ◇ перевесов 226

- ◇ шансов 226

Оценивание:

- ◇ в баллах по Фишеру 228

- ◇ максимального правдоподобия 228

Оценка 27

- ◇ в форме записи со шляпой (hat) 161

- ◇ несмещенная 34

- ◇ ошибки внепакетная 277

- ◇ плотности, ядерная 42

- ◇ смещенная 34

- из наивного байесова классификатора 214

- ◇ стандартная 87, 257

- ◇ точечная 85

Ошибка 87, 89

- ◇ 1-го рода 120, 125, 130

- ◇ 2-го рода 120, 125

- ◇ гетероскедастичная 197

- ◇ коэффициентов, стандартная 168

- ◇ мультиколлинеарности 134

- ◇ остатков, стандартная 164, 167

- ◇ отбора, систематическая 72

- ◇ среднеквадратическая, корень из нее 164

- ◇ стандартная 76, 79, 80

П

Параметр:

- ◇ масштаба 103

- ◇ сложности 270

- ◇ шкалы 103

- Перевесовка:
 - ◇ повышающая 243
 - ◇ понижающая 243, 245
- Перевесы 221, 222
- Переменная:
 - ◇ доминантная 330
 - ◇ зависимая 25, 159
 - ◇ индикаторная 178
 - ◇ искажающая 184, 187
 - ◇ категориальная 178
 - ◇ коррелированная 184, 185
 - ◇ косвенная 116
 - ◇ независимая 157, 159
 - главные эффекты 188
 - ◇ непрерывная с проверочной метрикой 107
 - ◇ перешкалирование с использованием z-оценок 259
 - ◇ предсказательная 25, 159
 - в наивном байесовом классификаторе 211
 - главные эффекты 188
 - коррелированная 185
 - связь между указанной переменной и откликом 199
 - ◇ упорядоченная:
 - категориальная 183
 - факторная 183
 - ◇ факторная:
 - в наивном байесовом классификаторе 211
 - многоуровневая 181
 - обработка в логистической регрессии 229
 - опорное кодирование 189
 - представление фиктивных переменных 178
 - разные кодировки 181
 - ◇ фиктивная 178
 - представление в регрессии 178
 - ◇ шкалированная 328
- Переменная факторная 178
- Переподгонка 130
- ◇ в линейной регрессии 173
- Пересечение 157
 - ◇ в примере с воздействием хлопка на объем легких 159
- Перестановка 114, 115
- Перетасовка целей 73
- Пермутация 114
- Пирсон, Карл 140, 299
- Плечо 191
 - ◇ влиятельные значения в регрессии 194
- Площадь под ROC-кривой 240
- Подмножество переменных, случайное 275
- Подразделение рекурсивное 263, 267, 275
- Подрезание 263
- Подъем 242
- Поиск бескрайний 73
- Полнота 233, 237
- Популяция 19, 66
 - ◇ размер 66
- Потеря 263
- Правило квадратного корня из n 79
- Предсказание и неконтролируемое самообучение 297
- Прецизионность 233, 237
- Признак 23
- Прирост 241
- Проблема редкого класса 236
- Проверка:
 - ◇ гипотез 126
 - коэффициент ложных открытий 131
 - ◇ двусторонняя 111
 - ◇ значимости 110, 126
 - ◇ односторонняя 111
 - ◇ хи-квадрат:
 - обнаружение мошенничества в науке 145
 - релевантность для науки о данных 146
- Прокси 116
- Процентиль 32, 35, 36
- Прочесывание данных 72
- Псевдоостаток 285

Р

- Разложение дисперсии 134, 139
- Размах 32, 35
 - ◇ межквартильный 32, 35
- Размер эффекта 151, 154
- Разрешение для участия людей в качестве испытуемых 109
- Рандомизация 106
- Распределение:
 - ◇ F-статистики 99
 - ◇ бернуллиево 96
 - ◇ биномиальное 95, 96
 - ◇ Вейбулла 100, 102
 - ◇ второго порядка 42
 - ◇ выборочное 75, 76
 - отбор по методу Томпсона 150
 - популяция против выборки 65

- ◇ гауссово 89
- ◇ данных 76
- ◇ многомерное нормальное 321
- ◇ нормальное 88, 89
- ◇ первого порядка 42
- ◇ Пуассона 100, 101, 225
- ◇ случайной величины, равномерное 145
- ◇ стандартное нормальное 87, 89
- ◇ Стьюдента 93
- ◇ хи-квадрат 98, 143
- ◇ экспоненциальное 100, 101
- Расстояние 316
- ◇ Говера 332, 328
- ◇ евклидово 255, 292
- ◇ Кука 194
- ◇ манхэттенское 255, 292, 332
- ◇ Махаланобиса 256
- Регрессия 157, 271
- ◇ в неконтролируемом самообучении в качестве структурного элемента 297
- ◇ взвешенная 164
- ◇ всех подмножеств 171, 173
- ◇ гребневая 173, 292
- ◇ диагностика 190
 - влиятельные значения 193
 - выбросы 191
 - графики частных остатков и нелинейность 199
 - использование сглаживателей диаграмм рассеяния 199
- ◇ и причинная связь 163
- ◇ интерпретация уравнения 184
- ◇ интерпретация уравнения регрессии
 - взаимодействия и главные эффекты 188
 - искажающие переменные 187
 - коррелированные предсказания 185
- ◇ к среднему значению 73
- ◇ квадратичная 202
- ◇ лассо 173, 292
- ◇ линейная 228
 - множественная 167, 173
- ◇ логистическая 221, 228, 247
 - результативность 229
- ◇ многочленная 201, 202
- ◇ на основе:
 - k ближайших соседей 262
 - наименьших квадратов 162, 292
 - обычных наименьших квадратов 162
- ◇ нелинейная 201, 202

- ◇ парная линейная 157
- ◇ пошаговая 170
- ◇ простая линейная 157, 158
- ◇ разные значения термина 163
- ◇ сплайновая 201, 203
- ◇ факторные переменные 178
 - многоуровневые 181
- ◇ штрафная 173
- Регуляризация 284
- Репрезентативность 66
- Робастность 26
- Рычаг 147
- Ряд временной 25

С

- Самообучение неконтролируемое 297
- Связь причинно-следственная и регрессия 163
- Сглаживатель диаграмм рассеяния 199
- Сеть физическая 25
- Система контрастного кодирования 181
- Скошенность 91
- Смещение 68, 72
- Совокупность генеральная 27, 66
- Сосед 252
- Специфичность 233, 237
- Сплайн 204
- Сравнение попарное 134
- Среднее 26, 27
- ◇ арифметическое 27
- ◇ взвешенное 28
- ◇ по всей популяции 71
- ◇ по выборке 71
- ◇ усеченное 26, 28
- Стандартизация 87, 89, 252, 307, 329
- Статистика:
 - ◇ выборочная 76
 - ◇ Дарбина — Уотсона 198
 - ◇ порядковая 32, 35
 - ◇ проверочная 106, 107, 127
 - выбор перед экспериментом 108
 - ◇ против машинного обучения 252
 - ◇ хи-квадрат 98, 140, 142
- Степень свободы 34, 93, 133, 143
- Структуры данных пространственные 25
- Сумма квадратов 134
- ◇ внутрикластерная 307
- ◇ межгрупповая 217
- ◇ остатков 162

Т

Таблица:

- ◇ сводная в Excel 57
- ◇ сопряженности 53, 57
- ◇ частотная 38, 40

Темп усвоения (фактор сжатия) 294

Теорема центральная предельная 76, 78, 94

Теория черного лебедя 91

Тест:

- ◇ A/B 105
- ◇ исчерпывающий перестановочный 119
- ◇ перестановочный 114, 115
 - для процедуры ANOVA 138
- ◇ рандомизационный 114, 119
- ◇ случайный:
 - перестановочный 119
 - пермутационный 114
- ◇ точный 114, 119
- ◇ универсальный 134
- ◇ Фишера точный 144

Тестирование:

- ◇ множественное 131
- ◇ традиционное 150

Техника повышающего отбора
синтетического меньшинства (SMOTE)
246

Тип данных, ресурсы для дополнительного
чтения 22

Точка интервала, конечная 85

Точность 233, 234

- ◇ улучшение в случайных лесах 277

Тьюки, Джон Уайлдер 19

У

Узел 201, 204, 263

Уровень доверия 85

Уровень значимости 120, 151, 154

Усвоение автоматическое 251

Усиление ансамблевое 284

Успех 96

Усреднение:

- ◇ модельное 273
- ◇ пакетное 273

Устранение обратное 171

Усы 40

Ф

Фактор:

- ◇ конверсия текстовых столбцов 22
- ◇ сжатия (темп усвоения) 294

Фасет 60

Форма записи со шляпой (hat) 161

Фридман, Джером Дж. (Джерри) 251

Функция:

- ◇ дискриминантная 215
- ◇ логистического отклика 222
- ◇ потери 246

Х

Характеристики получателя рабочие (ROC)
238

Хвост 38, 91

- ◇ длинный 91

Хиротугу, Акаике 171

Ц

Центр кластера 307, 308, 312

Ч

Чистота класса 268

Чувствительность 233, 237

Ш

Шкалирование 328

Э

Эксперимент статистический и проверка
значимости 105

Экстраполяция 175

Экссесс 42

Энтропия информации 269

Эрзац-переменная 116

Эффект:

- ◇ бескрайнего поиска 72, 73
- ◇ главный 184, 188