Глава 16

Поиск выбросов в данных

В ЭТОЙ ГЛАВЕ...

- Что является выбросом
- Различие между экстремальными значениями и новыми
- Использование простой статистики для поиска выбросов
- Поиск наиболее сложных выбросов с помощью передовых методов

Ошибки происходят, когда их меньше всего ждешь, и это также верно в отношении ваших данных. Кроме того, ошибки в данных трудно обнаружить, особенно если набор данных содержит много переменных разных типов и масштабов. Ошибки в данных могут принимать различные формы. Например, значения могут систематически отсутствовать для определенных переменных, ошибочные числа могут появляться здесь и там, а данные могут включать выбросы. Красный флаг должен быть поднят, когда:

- отсутствие значений в определенных группах случаев или переменных означают, что ошибку вызывает какая-то конкретная причина;
- ошибочные значения зависят от того, как приложение создало или обработало данные. Например, вам нужно знать, получило ли приложение данные от измерительного прибора. На надежность приборов могут повлиять внешние условия или ошибка человека;
- случай, по-видимому, действителен, но весьма отличается от обычных значений, которые характеризуют эту переменную. Когда вы не можете объяснить причину разницы, вы можете наблюдать выброс.

Среди описанных ошибок самая сложная проблема, которую нужно решить, — это когда в наборе данных есть выбросы, поскольку у вас не всегда есть уникальное определение выбросов или нет явной причины их наличия в данных. В результате многое остается только на ваше усмотрение.


СОВЕТ

Вам не нужно вводить исходный код этой главы вручную — используйте загружаемый исходный код (подробности о загрузке исходного кода см. во введении). Исходный код этой главы представлен в файле `P4DS4D2_16_Detecting_Outliers.ipynb`.

Обнаружение выбросов

Как общее определение, *выбросы* (outlier) — это данные, которые значительно отличаются (далеки) от других данных в выборке. Причина, по которой они находятся далеко, в том, что одно или несколько значений являются слишком высокими или слишком низкими по сравнению с большинством других. Они также могут отображать почти уникальную комбинацию значений. Например, если вы анализируете записи студентов, зачисленных в университет, ваше внимание могут привлечь студенты, которые слишком молоды или слишком стары. Студенты, изучающие несколько необычных предметов, также потребуют тщательного изучения.

Выбросы искажают распределения данных и влияют на все ваши основные статистические данные о тенденциях. Средние значения выдвигаются вверх или вниз, влияя на все другие описательные меры. Выброс всегда будет увеличивать дисперсию и изменять корреляции, поэтому вы можете получить неверные предположения о данных и отношениях между переменными.

Этот простой пример может отображать влияние (в небольшом масштабе) одного выброса на более чем тысячу регулярных наблюдений:

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
%matplotlib inline

import numpy as np
from scipy.stats.stats import pearsonr
np.random.seed(101)
normal = np.random.normal(loc=0.0, scale= 1.0, size=1000)
print('Mean: %0.3f Median: %0.3f Variance: %0.3f' %
                                  (np.mean(normal),
                                   np.median(normal),
                                   np.var(normal)))
```

Генератор случайных чисел NumPy создает в примере переменную `normal`, которая содержит 1000 наблюдений, полученных из стандартного нормального распределения. Базовая описательная статистика (среднее, медиана, дисперсия) не показывает ничего неожиданного. Вот итоговое среднее, медиана и дисперсия:

```
Mean: 0.026 Median: 0.032 Variance: 1.109
```

Теперь мы изменим одно значение, вставив выброс:

```
outlying = normal.copy()
outlying[0] = 50.0
print('Mean: %0.3f Median: %0.3f Variance: %0.3f' %
                                  (np.mean(outlying),
                                   np.median(outlying),
                                   np.var(outlying)))

print('Pearson''s correlation: %0.3f p-value: %0.3f' %
                              pearsonr(normal,outlying))
```

Вы можете взять эту новую переменную, `outlying`, и поместить в нее выброс (при индексе 0 у вас есть положительное значение 50,0). Теперь будет получена немного другая описательная статистика:

```
Mean: 0.074 Median: 0.032 Variance: 3.597
Pearsons correlation coefficient: 0.619 p-value: 0.000
```

Теперь статистика показывает, что среднее значение стало в три раза выше, чем раньше, как и дисперсия. Изменение влияет не только на медиану, которая зависит от положения (она сообщает значение, занимающее среднее положение, когда все наблюдения расположены по порядку).

Что еще более важно, корреляция исходной переменной и переменной с выбросом довольно далека от +1,0 (значение корреляции переменной по отношению к самой себе), что указывает на то, что мера линейных отношений между двумя переменными была серьезно повреждена.

Что еще может пойти не так

Выбросы не просто смещают ключевые показатели исследовательской статистики, они также меняют структуру отношений между переменными в данных. Выбросы могут повлиять на алгоритмы машинного обучения двумя способами.

» Алгоритмы, основанные на коэффициентах, могут получать неправильные коэффициенты, чтобы минимизировать их неспособность понять отдаленные случаи. Ярким примером являются линейные модели (суммы коэффициентов), но они не единственные. Выбросы

могут также влиять на алгоритмы на основе *обучения дерева решений* (tree-based learner), таких как алгоритмы AdaBoost или Grastent Boosting Machines.

» Поскольку алгоритмы обучаются на выборках данных, выбросы могут побудить алгоритм к искажению веса вероятности чрезвычайно низких или высоких значений при определенной конфигурации переменных.

Обе ситуации ограничивают способность алгоритма обучения хорошо обобщать новые данные. Другими словами, они приводят к искажению процесса обучения на этом наборе данных.


ЗАПОМНИ!

Существует несколько способов устранения выбросов — некоторые из них требуют изменения существующих данных, а другие — выбора подходящей функции ошибок для алгоритма машинного обучения. (Некоторые алгоритмы позволяют выбрать другую функцию ошибок в качестве параметра при настройке процедуры обучения.) Большинство алгоритмов машинного обучения могут применять различные функции ошибок. Функция ошибок важна, поскольку она помогает алгоритму учиться, понимая ошибки и применяя корректировки в процессе обучения, но некоторые функции ошибок чрезвычайно чувствительны к выбросам, в то время как другие довольно устойчивы к ним. Например, мера квадратичной ошибки имеет тенденцию подчеркивать выбросы, поскольку ошибки, получаемые из примеров с большими значениями, возводятся в квадрат, становясь тем самым еще более заметными.

Понятие аномалий и новых данных

Поскольку выбросы возникают как ошибки или крайне редкие случаи, их обнаружение никогда не бывает легкой задачей; но это важно для получения эффективных результатов от вашего проекта по науке о данных. В определенных областях обнаружение аномалий само по себе является целью науки о данных: обнаружение мошенничества в страховании и банковском деле, обнаружение неисправностей на производстве, мониторинговые системы в здравоохранении и других критически важных областях, а также обнаружение событий в системах безопасности и раннего предупреждения.

Важное различие заключается в поиске выбросов в существующих данных или в проверке любых новых данных, содержащих аномалии относительно существующих случаев. Возможно, вы потратили много времени на очистку данных или разработали приложение для машинного обучения на основе

доступных данных, поэтому было бы важно выяснить, похожи ли новые данные на старые, и будут ли продолжать работать алгоритмы классификации или прогнозирования.

В таких случаях аналитики данных говорят об обнаружении *новизны* (novelty), поскольку они должны знать, насколько хорошо новые данные напоминают старые. Быть исключительно новым считается аномалией: новизна может скрывать значительное событие или помешать правильной работе алгоритма, поскольку машинное обучение в значительной степени основано на обучении, исходя из прошлых примеров, и оно может не обобщаться до совершенно новых случаев. При работе с новыми данными вам следует переучить алгоритм.

Опыт учит, что мир редко бывает стабилен. Иногда новинки появляются вполне естественно, поскольку мир так изменчив. Следовательно, данные изменяются с течением времени неожиданным образом, как в целевых, так и в прогнозирующих переменных. Это явление называется *дрейфом понятий* (concept drift). Термин *понятие* (concept) относится к вашей цели и *дрейфует* к исходным данным, используемым для выполнения прогноза, который движется медленно и не поддается контролю, как лодка, дрейфующая из-за сильных приливов. Рассматривая модель науки о данных, вы будете различать разные дрейфы концепций и ситуации новизны.

- **Физическая** (physical). Системы распознавания лиц или голоса или даже климатические модели никогда не меняются. Не ожидайте новинок, но проверьте те выбросы, которые возникают в результате проблем с данными, таких как ошибочные измерения.
- **Политическая и экономическая** (political and economic). Эти модели иногда меняются, особенно в долгосрочной перспективе. Вы должны следить за долгосрочными эффектами, которые начинаются медленно, а затем распространяются и консолидируются, делая модели неэффективными.
- **Социальное поведение** (social behavior). Социальные сети и язык, которым вы пользуетесь каждый день, со временем меняются. Ожидайте появления новинок и предпринимайте меры предосторожности; в противном случае ваша модель внезапно испортится и станет непригодной для использования.
- **Данные поисковых систем, банковские операции и схемы мошенничества в электронной торговле** (search engine data, banking, and e-commerce fraud schemes). Эти модели меняются довольно часто. Вам необходимо проявлять особую осторожность при проверке появившихся новинок, предлагая обучить новую модель для поддержания точности.

- **Угрозы кибербезопасности и рекламные тренды** (cyber security threats and advertising trends). Эти модели постоянно меняются. Обнаружение новинок является нормой, и повторное использование одних и тех же моделей в течение длительного времени является опасным.

Изучение простого одномерного метода

Хорошим началом при поиске выбросов, независимо от количества имеющихся в данных переменных, является рассмотрение каждой переменной по отдельности, используя как графическую, так и статистическую проверку. Это однофакторный подход, позволяющий определить выбросы по неподходящему значению переменной. Пакет `pandas` позволяет легко обнаружить выбросы благодаря

- простому методу `describe`, информирующему о среднем значении, дисперсии, квартилях и экстремумах числовых значений для каждой переменной;
- системе автоматической визуализации диаграмм размаха.

Использование обоих методов позволяет легко узнать, есть ли выбросы и где их искать. Набор данных diabetes из модуля наборов данных Scikit-learn является хорошим примером для начала.

```
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
X,y = diabetes.data, diabetes.target
```

После этих команд все данные содержатся в переменной `X` типа `ndarray` NumPy. Затем код преобразует его в объект `pandas DataFrame` и запрашивает некоторую описательную статистику (вывод приведен на рис. 16.1):

```
import pandas as pd
pd.options.display.float_format = '{:.2f}'.format
df = pd.DataFrame(X)
df.describe()
```

Вы можете определить проблемные переменные, посмотрев на экстремумы распределения (максимальные значения переменной). Например, вы должны рассмотреть, находятся ли минимальное и максимальное значения соответственно далеко от 25-го и 75-го процентиля. Как показано в выводе, многие переменные имеют подозрительно большие максимальные значения. Анализ диаграммы размаха прояснит ситуацию. Следующая команда создает диаграмму размаха всех переменных, показанных на рис. 16.2:

```
In [5]: import pandas as pd
        pd.options.display.float_format = '{:.2f}'.format
        df = pd.DataFrame(X)
        df.describe()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9
count	442.00	442.00	442.00	442.00	442.00	442.00	442.00	442.00	442.00	442.00
mean	-0.00	0.00	-0.00	0.00	-0.00	0.00	-0.00	0.00	-0.00	-0.00
std	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
min	-0.11	-0.04	-0.09	-0.11	-0.13	-0.12	-0.10	-0.08	-0.13	-0.14
25%	-0.04	-0.04	-0.03	-0.04	-0.03	-0.03	-0.04	-0.04	-0.03	-0.03
50%	0.01	-0.04	-0.01	-0.01	-0.00	-0.00	-0.01	-0.00	-0.00	-0.00
75%	0.04	0.05	0.03	0.04	0.03	0.03	0.03	0.03	0.03	0.03
max	0.11	0.05	0.17	0.13	0.15	0.20	0.18	0.19	0.13	0.14

Рис. 16.1. Описательная статистика для `DataFrame`

```
fig, axes = plt.subplots(nrows=1, ncols=1,
                         figsize=(10, 5))
df.boxplot(ax=axes);
```

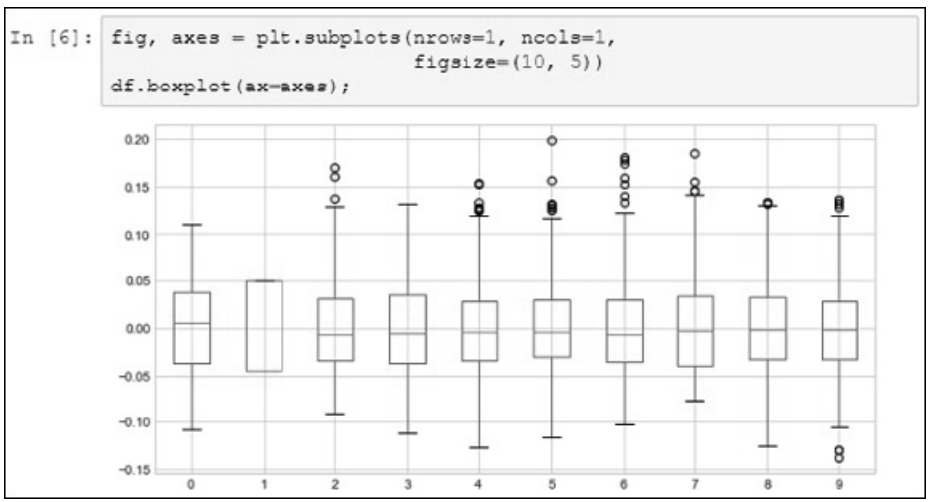

Рис. 16.2. Диаграмма размаха

Диаграммы размаха, созданные `pandas DataFrame`, будут иметь усы, установленные на “плюс” или “минус” 1,5 IQR (межквартильный диапазон или расстояние между нижним и верхним квартилем) по отношению к верхней и нижней стороне ящика (верхний и нижний квартили). Этот стиль диаграммы размаха Тьюки (по имени статистика Джона Тьюки (John Tukey), который

создал и продвигал его среди статистиков вместе с другими методами описания данных), позволяет визуализировать случаи за пределами усов. (Все точки за пределами этих усов считаются выбросами.)

Опора на гауссово распределение

Еще одной эффективной проверкой на выбросы в данных является использование нормального распределения. Даже если данные не распределены нормально, их стандартизация позволит вам предположить определенные вероятности обнаружения аномальных значений. Например, 99,7% значений, находящихся в стандартизированном нормальном распределении, должны находиться в диапазоне +3 и –3 стандартных отклонений от среднего значения, как показано в следующем коде:

```
from sklearn.preprocessing import StandardScaler
Xs = StandardScaler().fit_transform(X)
# метод .any(1) позволит избежать дублирования
df[(np.abs(Xs)>3).any(1)
```

На рис. 16.3 приведены результаты, описывающие строки в наборе данных, и демонстрирующие некоторые возможные выбросы значений.

```
In [7]: from sklearn.preprocessing import StandardScaler
        Xs = StandardScaler().fit_transform(X)
        # .any(1) method will avoid duplicating
        df[(np.abs(Xs)>3).any(1)]
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9
58	0.04	-0.04	-0.06	0.04	0.01	-0.06	0.18	-0.08	-0.00	-0.05
123	0.01	0.05	0.03	-0.00	0.15	0.20	-0.06	0.19	0.02	0.07
216	0.01	0.05	0.04	0.05	0.05	0.07	-0.07	0.15	0.05	0.05
230	-0.04	0.05	0.07	-0.06	0.15	0.16	0.00	0.07	0.05	0.07
256	-0.05	-0.04	0.16	-0.05	-0.03	-0.02	-0.05	0.03	0.03	0.01
260	0.04	-0.04	-0.01	-0.06	0.01	-0.03	0.15	-0.08	-0.08	-0.02
261	0.05	-0.04	-0.04	0.10	0.04	-0.03	0.18	-0.08	-0.01	0.02
269	0.01	-0.04	-0.03	-0.03	0.04	-0.01	0.16	-0.08	-0.01	-0.04
322	0.02	0.05	0.06	0.06	0.02	-0.04	-0.09	0.16	0.13	0.08
336	-0.02	-0.04	0.09	-0.04	0.09	0.09	-0.06	0.15	0.08	0.05
367	-0.01	0.05	0.17	0.01	0.03	0.03	-0.02	0.03	0.03	0.03
441	-0.05	-0.04	-0.07	-0.08	0.08	0.03	0.17	-0.04	-0.00	0.00

Рис. 16.3. Сообщение о возможных выбросах

Модуль Scikit-learn предоставляет простой способ стандартизации данных и записи всех преобразований для последующего использования в различных наборах данных. Это означает, что все данные, независимо от того, используются ли они для машинного обучения или для проверки производительности, стандартизируются одинаково.

СОВЕТ

Правило 68-95-99.7 гласит, что в стандартизированном нормальном распределении 68% значений находятся в пределах одного стандартного отклонения, 95% — в пределах двух стандартных отклонений, а 99,7% — в пределах трех. При работе с искаженными данными правило 68-95-99.7 может не выполняться, и в таком случае может потребоваться более консервативная оценка, такая как неравенство Чебышева. *Неравенство Чебышева* опирается на формулу, гласящую, что для `k` стандартных отклонений вокруг среднего значения, количество случаев в процентах не должно превышать `1/k^2` вокруг среднего значения[1]. Следовательно, при семи стандартных отклонениях от среднего значения вероятность нахождения корректного значения составляет не более 2%, независимо от того, каково распределение (2% — это низкая вероятность, поэтому ваш случай можно считать почти наверняка выбросом).

СОВЕТ

Неравенство Чебышева является консервативным. Высокая вероятность быть выбросом соответствует семи или более стандартным отклонениям от среднего значения. Используйте это, когда велика цена посчитать ценность выбросом, когда это не так. Для всех остальных применений достаточно правила 68-95-99.7.

Предположения и проверка

Найдя некоторые возможные одномерные выбросы, вы должны решить, как с ними бороться. Если вы полностью не доверяете отдаленным случаям, полагая, что это ошибки или заблуждения, удалите их. (В Python можно просто отменить их выбор, используя *прихотливую индексацию* (fancy indexing).)

СОВЕТ

Изменение значений в данных или решение об исключении определенных значений — это решение, которое необходимо принять после того, как вы поймете, почему в данных есть некоторые выбросы. Вы можете исключить необычные значения или случаи, для которых

[1] Проще говоря, случайная величина в основном принимает значения, близкие к своему среднему. См. лучше `https://ru.wikipedia.org/wiki/Неравенство_Чебышёва`. — *Примеч. ред.*

предполагаете, что произошла некоторая ошибка в измерении, в записи или предшествующей обработке данных. Если же вы поймете, что отдаленный случай является вполне законным, хотя и редким, лучшим выходом будет снизить его вес (если алгоритмы обучения используют вес для наблюдений) или увеличить размер выборки данных.

В нашем случае, решив сохранить и стандартизировать данные, мы могли бы просто ограничить значения выбросов, используя простой множитель стандартного отклонения:

```
Xs_capped = Xs.copy()
o_idx = np.where(np.abs(Xs)>3)
Xs_capped[o_idx] = np.sign(Xs[o_idx]) * 3
```

В предлагаемом коде функция `sign` из NumPy восстанавливает знак отдаленного наблюдения (+1 или –1), который затем умножается на значение 3 и присваивается соответствующей точке данных, восстановленной с помощью логической индексации стандартизированного массива.

Этот подход имеет ограничение. Будучи стандартным отклонением, используемым как для высоких, так и для низких значений, оно подразумевает симметрию в распределении данных, чего в реальных данных зачастую нет. В качестве альтернативы можете использовать более сложный подход — *винзоризацию* (winsorizing). При его использовании, считающиеся выбросами значения обрезаются до значения определенных процентилей, которые действуют как пределы значений (обычно 5-й процентиль для нижней границы, 95-й для верхней):

```
from scipy.stats.mstats import winsorize
Xs_winsorized = winsorize(Xs, limits=(0.05, 0.95))
```

Таким образом, вы создаете другое значение барьера для б*о*льших и меньших значений с учетом любой асимметрии в распределении данных. Независимо от того, что вы решите использовать для ограничения (стандартное отклонение или винзоризацию), ваши данные теперь готовы для дальнейшей обработки и анализа.

Наконец, альтернативное решение — позволить Scikit-learn автоматически преобразовать данные и обрезать выбросы с помощью преобразователя `RobustScaler`, основанного на IQR (как в диаграмме размаха, обсуждавшейся ранее в этой главе).

Выработка многомерного подхода

Работа с отдельными переменными позволяет обнаружить большое количество выбросов в наблюдениях. Но выбросы не обязательно являются значениями, слишком далекими от нормы. Иногда выбросы состоят из необычных комбинаций значений в нескольких переменных. Это редкие, но влиятельные комбинации, способные обмануть алгоритмы машинного обучения.

В таких случаях точная проверка каждой переменной не сможет исключить аномальные случаи из набора данных. Только несколько выбранных методов, учитывающих больше переменных за раз, смогут выявить проблемы в данных.

Следующие методики позволяют подойти к проблеме с разных точек зрения.

- Уменьшение размерности.
- Кластеризация плотности.
- Нелинейное моделирование распределения.

Использование этих методов и сравнение их результатов позволяет обратить внимание на повторяющиеся сигналы в конкретных случаях — иногда уже найденных в одномерном исследовании, а иногда еще нет.

Использование анализа основных компонентов

Анализ основных компонентов может полностью реструктурировать данные, устраняя избыточность и упорядочивая вновь полученные компоненты в соответствии с количеством исходных отклонений, которые они выражают. Этот тип анализа предлагает синтетическое и полное представление о распределении данных, делая многовариантные выбросы особенно очевидными.

Первые два компонента, будучи наиболее информативными с точки зрения дисперсии, могут отображать общее распределение данных, если они визуализируются. Вывод обеспечивает хороший намек на возможные очевидные выбросы.

Последние два компонента, являющиеся наиболее остаточными, они отображают всю информацию, которая иначе не могла бы быть подобрана методом `PCA`. Они также могут дать представление о возможных, но менее очевидных выбросах.

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from pandas.plotting import scatter_matrix
pca = PCA()
Xc = pca.fit_transform(scale(X))
```

```
first_2 = sum(pca.explained_variance_ratio_[:2]*100)
last_2 = sum(pca.explained_variance_ratio_[-2:]*100)

print('variance by the components 1&2: %0.1f%%' % first_2)
print('variance by the last components: %0.1f%%' % last_2)

df = pd.DataFrame(Xc, columns=['comp_' + str(j)
                  for j in range(10)])
fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(15, 5))
first_two = df.plot.scatter(x='comp_0', y='comp_1',
                            s=50, grid=True, c='Azure',
                            edgecolors='DarkBlue',
                            ax=axes[0])
last_two = df.plot.scatter(x='comp_8', y='comp_9',
                           s=50, grid=True, c='Azure',
                           edgecolors='DarkBlue',
                           ax=axes[1])
plt.show()
```

На рис. 16.4 показаны две диаграммы рассеяния первого и последнего компонентов. В выводе также сообщается об отклонении, объясненном первыми двумя компонентами (половина информативного содержимого набора данных) PCA и последними двумя:

```
variance by the components 1&2: 55.2%
variance by the last components: 0.9%
```

Обратите особое внимание на точки данных вдоль осей (где ось x определяет независимую переменную, а ось y — зависимую). Вы можете увидеть возможный порог, используемый для отделения обычных данных от подозрительных.

Используя два последних компонента, вы можете найти несколько точек для исследования, используя порог –0,3 для десятого компонента и –1,0 для девятого. Все случаи ниже этих значений являются возможными выбросами (рис. 16.5).

```
outlying = (Xc[:,-1] > 0.3) | (Xc[:,-2] > 1.0)
df[outlying]
```

Использование кластерного анализа для определения выбросов

Выбросы являются изолированными точками в пространстве переменных, а DBScan является алгоритмом кластеризации, который связывает плотные части данных и отмечает разреженные. Таким образом, DBScan — это идеальный инструмент для автоматического исследования данных и проверки возможных выбросов.

```
first_2 = sum(pca.explained_variance_ratio_[:2]*100)
last_2 = sum(pca.explained_variance_ratio_[-2:]*100)

print('variance by the components 1&2: %0.1f%%' % first_2)
print('variance by the last components: %0.1f%%' % last_2)

df = pd.DataFrame(Xc, columns=['comp_' + str(j)
                               for j in range(10)])

fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(15, 5))
first_two = df.plot.scatter(x='comp_0', y='comp_1',
                            s=50, grid=True, c='Azure',
                            edgecolors='DarkBlue',
                            ax=axes[0])
last_two  = df.plot.scatter(x='comp_8', y='comp_9',
                            s=50, grid=True, c='Azure',
                            edgecolors='DarkBlue',
                            ax=axes[1])

plt.show()
```

```
variance by the components 1&2: 55.2%
variance by the last components: 0.9%
```

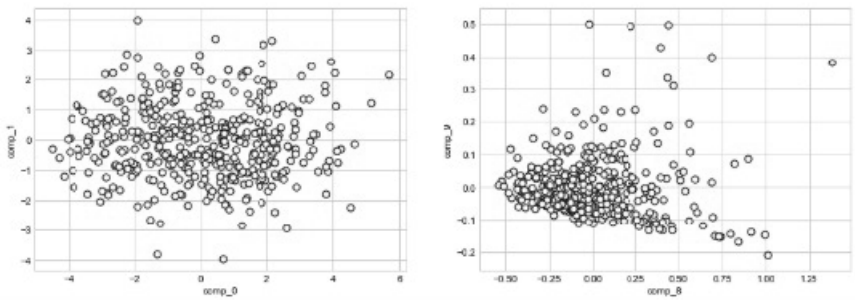

Рис. 16.4. Два первых и два последних компонента из PCA

```
In [12]: outlying = (Xc[:,-1] > 0.3) | (Xc[:,-2] > 1.0)
         df[outlying]
```

Out[12]:

	comp_0	comp_1	comp_2	comp_3	comp_4	comp_5	comp_6	comp_7	comp_8	comp_9
23	3.77	-1.76	1.09	0.72	-0.64	1.90	0.56	1.09	0.44	0.50
58	-2.65	2.23	2.79	-0.63	0.26	-0.13	1.44	0.67	1.01	-0.21
110	-2.04	-0.76	0.74	-1.93	-0.07	0.24	-1.75	-0.41	0.47	0.31
169	2.35	0.15	-0.13	1.19	-0.64	0.64	2.65	-0.31	0.22	0.50
254	3.82	-1.03	1.06	0.44	0.27	0.86	0.97	0.66	0.43	0.33
322	4.52	-2.24	-0.14	0.85	-0.47	0.73	1.28	0.34	1.39	0.38
323	3.87	0.69	0.26	0.58	0.97	0.76	1.79	0.36	0.69	0.40
353	0.98	1.61	-1.16	1.14	-0.36	1.46	2.53	0.90	-0.02	0.50
371	2.11	-0.28	0.64	-0.65	-0.36	-0.26	2.22	1.09	0.07	0.35
394	2.24	-1.13	0.51	1.54	-1.30	-0.12	2.28	-0.10	0.40	0.43

Рис. 16.5. Возможные случаи выбросов, обнаруженные PCA

Ниже приведен пример использования DBScan для обнаружения выбросов.

```
from sklearn.cluster import DBSCAN
DB = DBSCAN(eps=2.5, min_samples=25)
DB.fit(Xc)

from collections import Counter
print(Counter(DB.labels_))
df[DB.labels_==-1]
```

Однако для `DBSCAN` требуются два параметра, `eps` и `min_samples`, которые требуют многократных попыток найти правильные значения, что делает их использование немного сложным.

Как упоминалось в предыдущей главе, начинайте с низкого значения `min_samples` и пробуйте увеличивать значения `eps` с 0,1 и выше. После каждого испытания с измененными параметрами проверяйте ситуацию, подсчитав количество наблюдений в классе –1 внутри атрибута `labels`, и остановитесь, когда количество выбросов окажется разумным при визуальном осмотре.

СОВЕТ

На границе распределения плотных частей всегда будут точки, поэтому трудно предоставить пороговое значение для количества случаев, которые могут быть классифицированы в классе –1. Обычно выбросы не должны превышать 5% случаев, поэтому используйте это указание как общее эмпирическое правило.

Вывод предыдущего примера сообщат, сколько примеров попадает в группу –1, которую алгоритм считает не частью основного кластера, и предоставляет список случаев, которые являются его частью.

СОВЕТ

Алгоритм кластеризации *k*-средних менее автоматизирован, но вы также можете использовать его для обнаружения выбросов. Сначала запустите кластерный анализ с достаточным количеством кластеров. (Вы можете опробовать разные решения, если не уверены.) Затем отыщите кластеры с несколькими примерами (или, может быть, с одним); они, вероятно, являются выбросами, поскольку выглядят как небольшие отдельные кластеры, которые отделены от больших кластеров, содержащих большинство примеров.

Автоматическое обнаружение с помощью изоляционного леса

Случайные леса (random forest) и *сверхслучайные деревья* (extremely randomized tree) — это мощные методы машинного обучения, которые будут описаны в главе 20. Они работают, разделяя набор данных на меньшие наборы

на основании определенных значений переменных, чтобы упростить прогнозирование классификации или регрессии для каждого меньшего подмножества (принцип “разделяй и властвуй”).

`IsolationForest` — это алгоритм, который использует тот факт, что выброс легче выделить из большинства случаев на основе различий между его значениями или комбинацией значений. Алгоритм отслеживает, сколько времени требуется, чтобы отделить случай от других и поместить его в свое собственное подмножество. Чем меньше усилий требуется на его отделение, тем больше вероятность того, что случай является выбросом. В качестве меры `IsolationForest` производит измерение расстояния (чем короче расстояние, тем больше вероятность того, что это выброс).


СОВЕТ

Пока ваши алгоритмы машинного обучения находятся в разработке, обученный `IsolationForest` может действовать как проверка работоспособности, поскольку многие алгоритмы машинного обучения не могут справиться с выбросами и новыми примерами.

Чтобы настроить `IsolationForest` на обнаружение выбросов, достаточно выбрать уровень, который определит процент случаев, считающихся выбросами, на основе измерения расстояния. Выберите такой процент, исходя из вашего опыта и ожиданий качества данных. Для создания работающего `IsolationForest` выполните следующий сценарий:

```
from sklearn.ensemble import IsolationForest
auto_detection = IsolationForest(max_samples=50,
                                 contamination=0.05,
                                 random_state=0)
auto_detection.fit(Xc)
evaluation = auto_detection.predict(Xc)
df[evaluation==-1]
```

Вывод содержит список случаев, подозреваемых как выбросы. Кроме того, алгоритм обучен распознавать нормальные примеры наборов данных. Когда вы предоставляете в набор данных новые наборы данных и оцениваете их с помощью обученного `IsolationForest`, вы можете сразу определить, что с вашими новыми данными что-то не так.


ЗАПОМНИ!

`IsolationForest` — вычислительно требовательный алгоритм. Выполнение анализа большого набора данных занимает много времени и памяти.