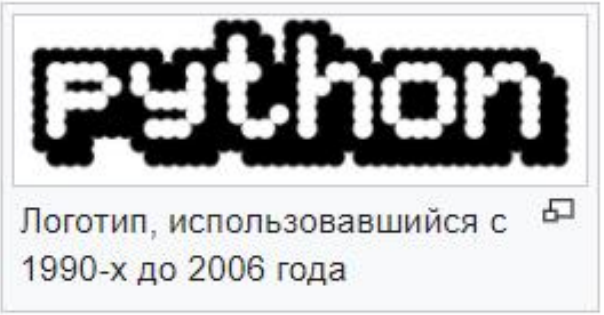
предмет:

Методы машинного обучения

Для начала рассмотрим:

- ✓ Почему Python?
- ✓ Зачем статистика?
- ✓ Как подготовить данные

Питон или Пайтон?
Язык Python назван в честь телешоу комик-группы Монти **Пайтон**.



Логотип, использовавшийся с 1990-х до 2006 года

Какие преимущества есть у Python

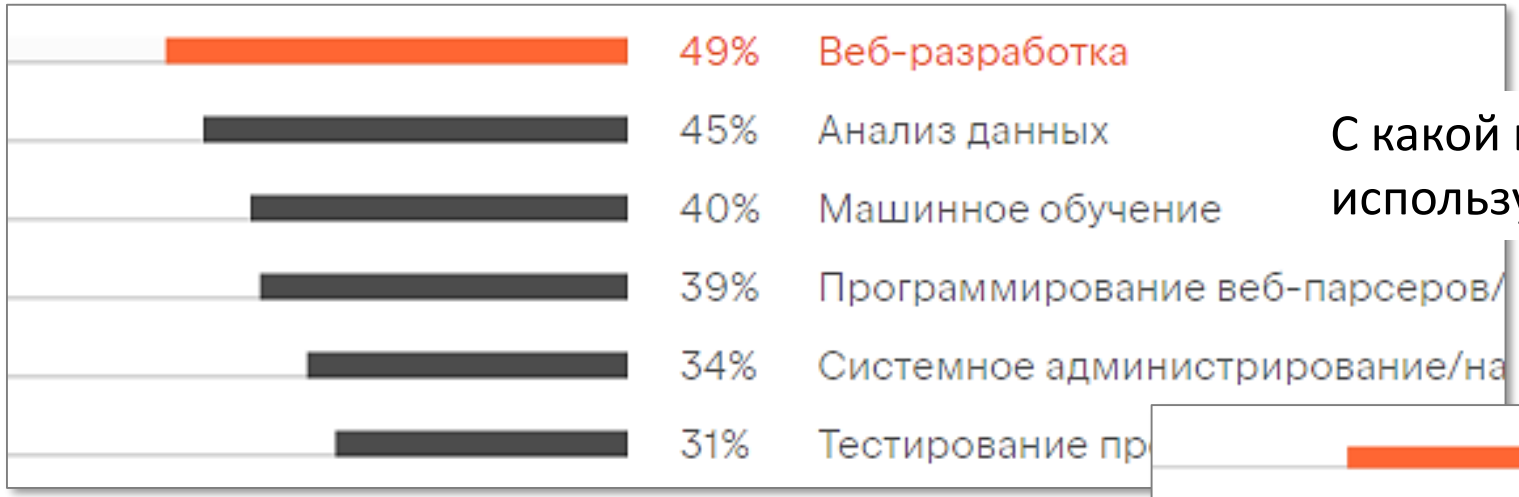
Python невероятно эффективен: программы, написанные на нем, делают больше, чем многие на других языках и в меньшем объеме кода.

Его просто учить
Главной целью основателя *Python*, Гвидо ван Россума, было создать простой и понятный широкому кругу людей язык программирования. Изучение любого языка требует усидчивости и дисциплины. Но *Python* в этом смысле считается одним из самых комфортных, особенно для новичков. Простой синтаксис позволяет легко учиться и читать код.

Он очень распространенный
Python универсален благодаря большой встроенной библиотеке и богатой коллекции библиотек, поэтому его применяют в самых разных областях.

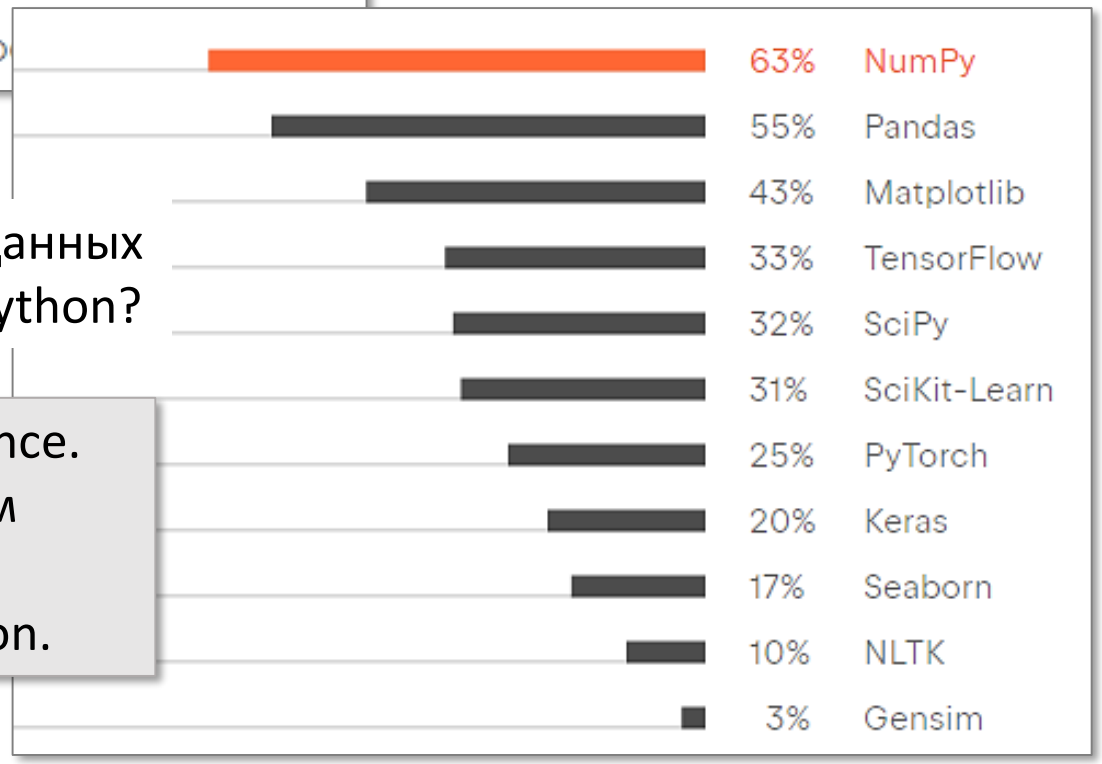

Гвидо Ван Россум

Из отчета об опросе разработчиков на Python, подготовленном компанией JetBrains совместно с Python Software Foundation, 2020г.



С какой целью вы используете Python?

За последние 5 лет проведения опроса JetBrains «Экосистема разработки» основные сферы применения языка Python не изменились.



Какие фреймворки для анализа данных вы используете в дополнение к Python?

Сегодня Python — основной язык для Data Science. Большинство разработчиков, занятых анализом данных (54%), инжинирингом данных (54%) и машинным обучением (71%), используют Python.

В **Amazon** и **Spotify** используют Python для анализа пользовательских данных, информации о продажах и разработки персонализированных рекомендаций.

YouTube и **Instagram**… Эти проекты полностью написаны на Python. Кроме того, холдинг Alphabet использует Python для скрейпинга в Google — извлечения данных со страниц веб-ресурсов.

Netflix создала свой рекомендательный сервис с нуля на Python.

Autodesk в своём редакторе 3D-анимации Maya с помощью Python создаёт мультипликацию. Так же язык использует студия **Pixar**.

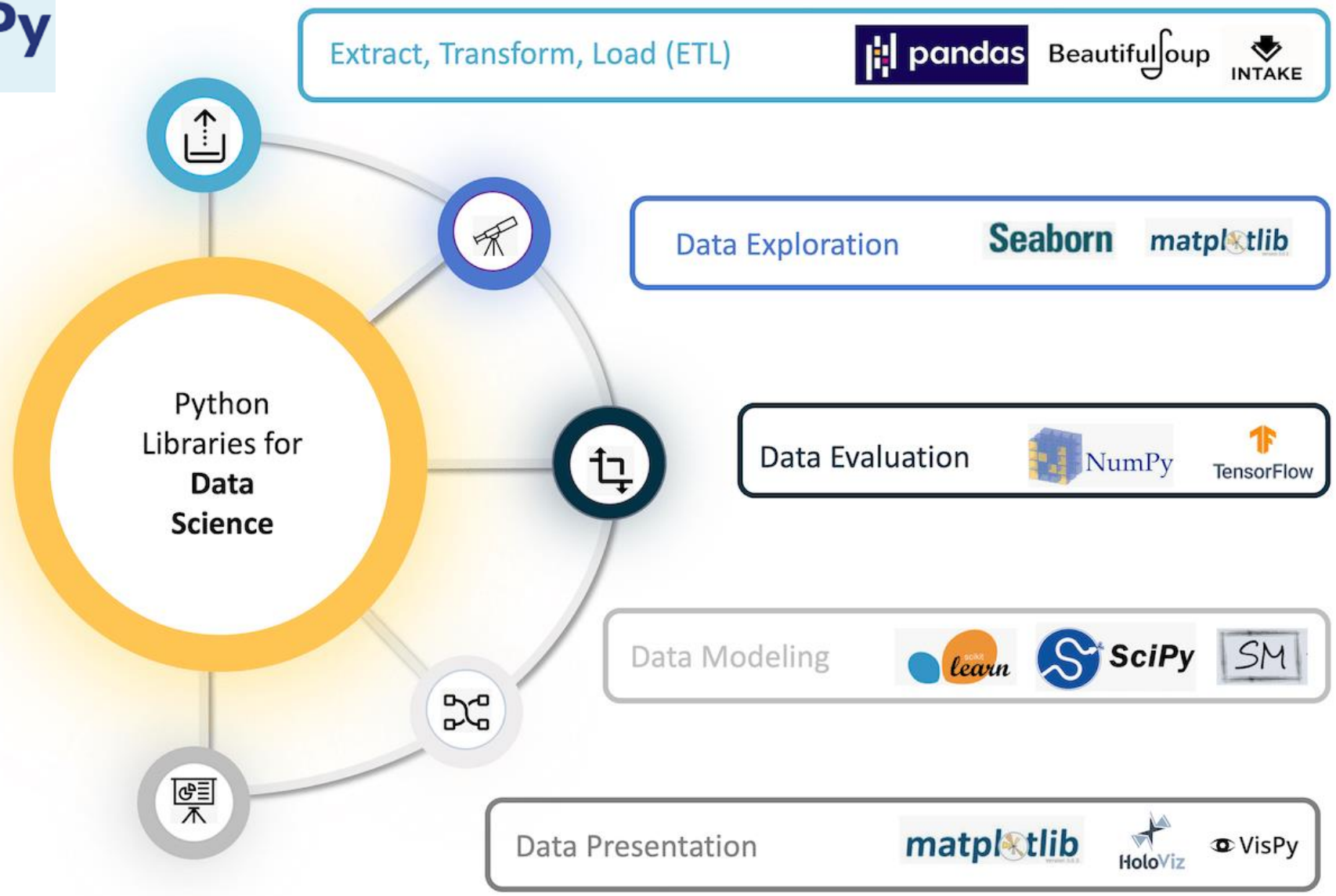
JPMorgan Chase, крупный американский финансовый холдинг, применяет Python для прогнозирования рынка.

NASA работает с проектами на этом языке программирования, чтобы проводить научные вычисления.

В этом списке собрана лишь незначительная часть компаний и сервисов, которые работают с Python. В их числе также **Mail.ru, Яндекс, Yahoo, Facebook, Dropbox, Quora** и многие другие.

Экосистема для анализа больших данных на основе **NumPy**

https://blog.skillfactory.ru/glossary/numpy/



Что такое статистика?

Статистика — это набор математических методов и инструментов, позволяющих ответить на важные вопросы о данных. Она делится на две категории:

- ✓ **Описательная статистика.** Предлагает методы резюмирования данных путем преобразования необработанных наблюдений в значимую информацию, которую легко интерпретировать и распространять.
- ✓ **Логическая статистика**. Предлагает методы изучения экспериментов, выполненных на маленьких образцах данных, и умозаключения для всей популяции (всего домена).

Чтобы трансформировать результаты наблюдений в имеющие смысл идеи, применяется **описательная** статистика. Затем можно применить **логическую** статистику, чтобы изучить небольшие выборки данных и сделать выводы для экстраполяции результатов на всю совокупность данных.

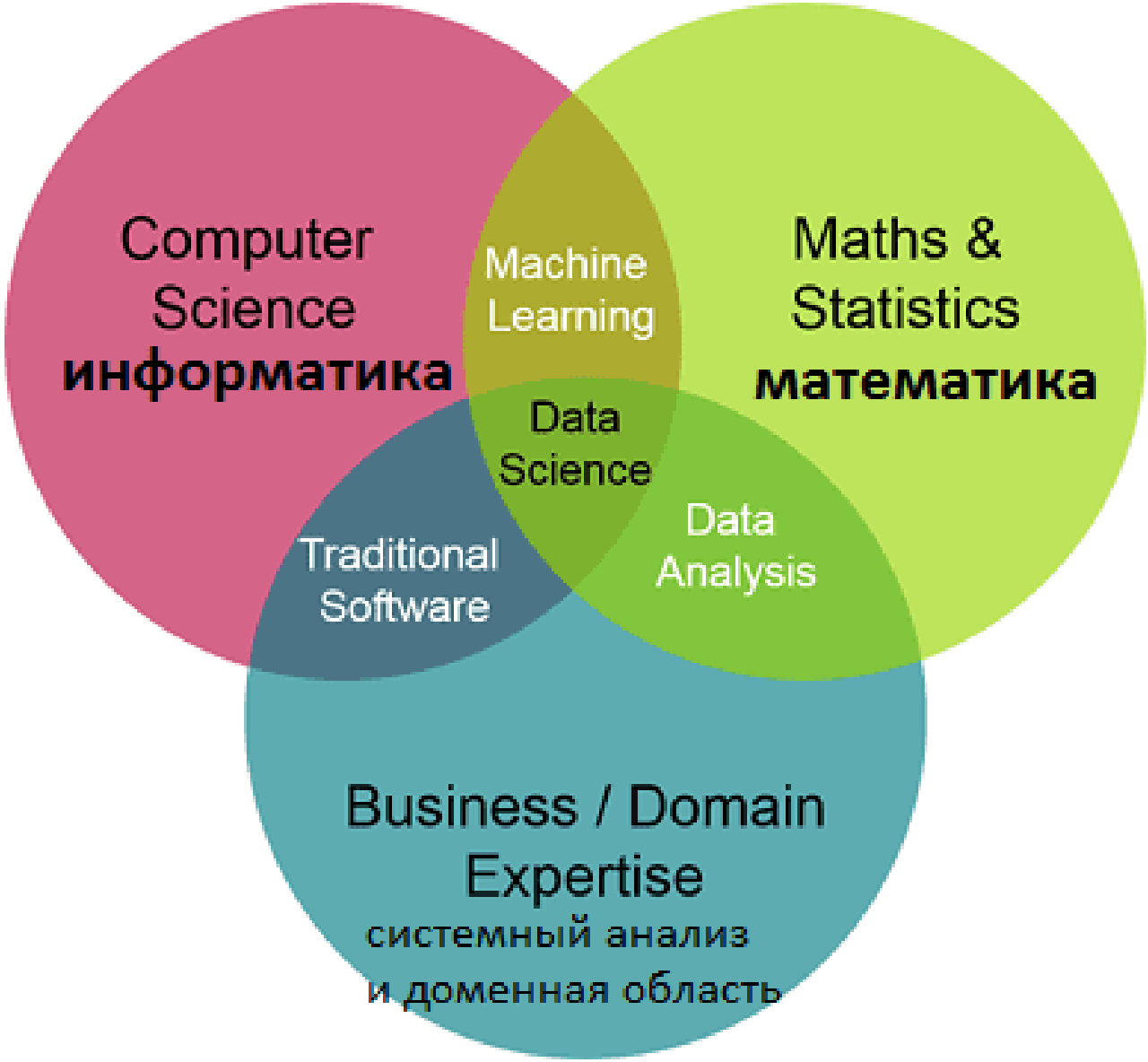
Статистика помогает ответить на вопросы, подобные этим:

- ✓ Какие из признаков наиболее важны?
- ✓ Какие показатели производительности мы должны измерять?
- ✓ Какой самый распространенный и ожидаемый результат?
- ✓ Как отличить шум от достоверных данных?

Статистика и машинное обучение — две тесно связанные между собой области.

Математика и статистика являются основными строительными блоками алгоритмов машинного обучения.

Статистика дает важные предпосылки для прикладного машинного обучения: она помогает выбирать, оценивать и интерпретировать модели прогнозирования.



ph_piter 30 января 2017 в 10:37

Разница между статистикой и наукой о данных

Блог компании Издательский дом «Питер» , Data Mining *, Алгоритмы *, Big Data *, R *

Мнение: Статистикам весь тренд, связанный с наукой о данных, кажется слегка высокомерным. . . эта сфера деятельности весьма пересекается с той работой, которой статистики занимаются уже не одно десятилетие.

“Думаю, data-scientist – распиаренный синоним для «специалист по статистике»” – заявил *Нейт Сильвер** в 2013 году на лекции в Joint Statistical Meeting.

Брэд Шлумич специалист по data science в Twitch: “**Статистика – важнейшая составляющая науки о данных**. У нас в Twitch команда data science обладает тремя компетенциями: статистика, программирование и понимание продукта. **Мы никогда не взяли бы на работу человека, слабо ориентирующегося в статистике**.”

** Нейт Сильвер (Nate Silver) - тот самый человек, который верно спрогнозировал итоги голосования на президентских выборах 2008 года в 49 из 50 штатов США. В 2012 году у него получилось уже 50 из 50.*

Описательная статистика — это описание наборов данных.

Описательная статистика использует два основных подхода:

- ✓ **Количественный подход**, который описывает общие численные характеристики данных.
- ✓ **Визуальный подход**, который иллюстрирует данные с помощью диаграмм, графиков, гистограмм и прочих графических образов.

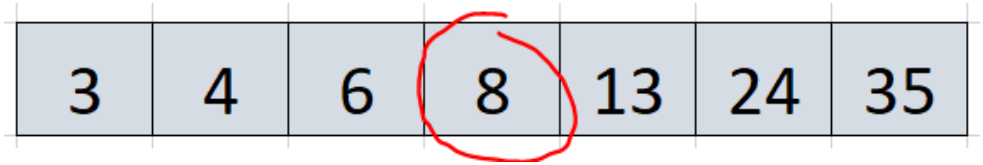
Метрики описательной статистики:

- ✓ **Метрики центрального положения**, которые говорят вам о центрах концентрации данных, таких как *среднее*, *медиана* и *мода*.
- ✓ **Метрики оценки вариативности** данных, которые говорят о разбросе значений, такие как *дисперсия* и *стандартное* ***отклонение***.

Метрики центрального положения

Среднее (mean)
Сумма всех значений, деленная на количество значений или среднее арифметическое.

Медиана (median)
Середина в отсортированных данных.



Мода (mode)
Значение, которое встречается наиболее часто.

мода=3

3	4	3	8	4	5	3

Мода часто употребляется для текстовых данных. Например: цвета автомобилей — белый, чёрный, синий металлик, белый, синий металлик, белый. Какая мода?

Выброс (outlier)
Значение данных, которое сильно отличается от большинства данных.

Метрики оценки вариативности

Размах (range) Разница между самым большим и самым малым значениями в наборе данных.

Математи́ческое ожида́ние — среднее (взвешенное по вероятностям возможных значений) значение случайной величины. На практике математическое ожидание обычно оценивается как среднее арифметическое наблюдаемых значений случайной величины (выборочное среднее, среднее по выборке).

Дисперсия (variance) средний квадрат отклонений индивидуальных значений признака от их средней величины. Еще называют: *среднеквадратическое отклонение, среднеквадратическая ошибка*.

Стандартное отклонение (standard deviation) Квадратный корень из дисперсии. И в отличие от дисперсии оно показывает реальное среднее значение наших отклонений от среднего значения по выборке.

Процентиль – например, **75**-й процентиль – это число, ниже которого находится **75**% всех наблюдений.

Квартили - такие точки, которые делят наше распределение на 4 равные части (25%, 50% и 75%)

Квантили *- такие значения признака, которые делят упорядоченные данные на некоторое число равных частей.*

Данные для машинного обучения. Основные понятия

Генеральная совокупность

Суммарная численность объектов наблюдения, обладающих определенным набором признаков, ограниченная в пространстве и времени.

Выборка (Выборочная совокупность)

Часть объектов из генеральной совокупности, отобранных для изучения, с тем чтобы сделать заключение о всей генеральной совокупности. Для того чтобы заключение, полученное путем изучения выборки, можно было распространить на всю генеральную совокупность, выборка должна обладать свойством репрезентативности.

Репрезентативность выборки

Свойство выборки корректно отражать генеральную совокупность.

Примеры:

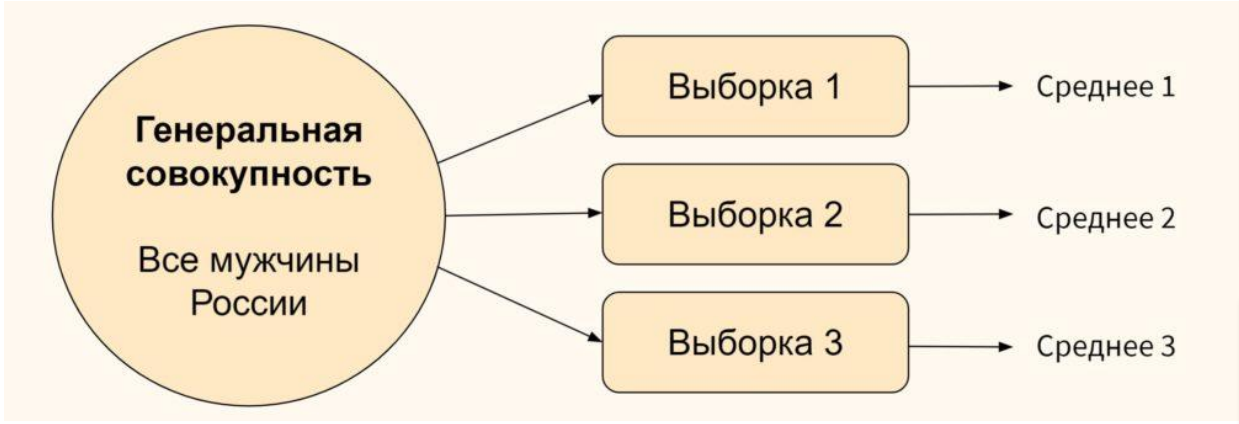
- ✓ Выборка, целиком состоящая из горожан, владеющих автомобилем, не репрезентирует все население города.
- ✓ Выборка только из учеников одной школы не репрезентирует всех школьников страны.

Как знания статистики помогают в формировании выборки?

Задача: получить **количественные данные** (quantitative data) о росте всех мужчин в стране.
Но, опросив тысячу мужчин, можем ли мы сказать, что полученные результаты полностью отражают фактические показатели роста всех мужчин в стране? Нет, не можем, ведь мы же не измеряли рост каждого человека. То есть на первый взгляд **задача кажется нереализуемой**.

По сути, те мужчины, которых удалось измерить, попали в **выборку** (sample). А все мужчины страны — это **генеральная совокупность** (population).

Но можно ли вообще хоть как-то определить генеральную совокупность, имея ограниченный набор данных? На самом деле, да. Существует теоретическое обоснование этой возможности и называется оно **Центральной предельной теоремой**.



$$\mu = \frac{\text{среднее выборки 1} + \text{среднее выборки 2} + \text{среднее выборки 3}}{\text{количество выборок (в данном случае, 3)}}$$

Центральная предельная теорема

Central Limit Theorem говорит, что при множественном выборочном сборе данных среднее средних всех полученных выборок станет стремиться к среднему генеральной совокупности.

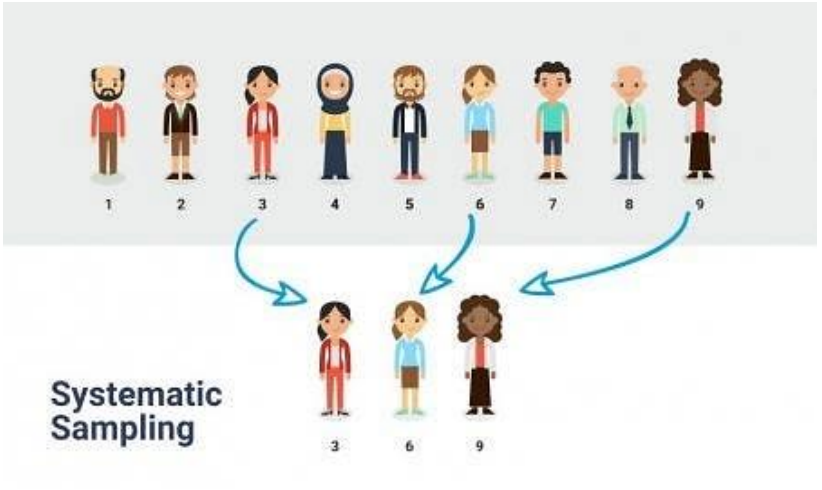
Из центральной предельной теоремы следует, что мы получили возможность сказать что-либо определенное про ту величину, которую в принципе невозможно охватить измерением. Т.е., исследуя данные выборки, мы можем делать выводы о всей популяции.

Итак выборку отбирают из генеральной совокупности. Выборку следует отбирать так, чтобы она обладала всеми свойствами, которыми обладает совокупность. Измеримые свойства генеральной совокупности: среднее значение, стандартное отклонение и т. д.

Рассмотрим вероятностные методы создания выборки (есть еще и невероятностные).

Вероятностная выборка: это метод выборки, при котором выборки из большой совокупности собираются с использованием теории вероятности. Существует три типа вероятностной выборки:

- ✓ **Случайная выборка.** В этом методе каждый член совокупности имеет равные шансы попасть в выборку.
- ✓ **Систематическая выборка.** При систематической выборке каждая *n* -я запись выбирается из генеральной совокупности и становится частью выборки.
- ✓ **Стратифицированная выборка.** При стратифицированной выборке страта используется для формирования выборок из большой совокупности. Страта – это подгруппа населения, имеющая хотя бы одну общую характеристику. После этого методом случайной выборки отбирают достаточное количество испытуемых из каждой страты.



Набор данных и их атрибутов

По горизонтали таблицы располагаются *атрибуты* объекта или его признаки. *По* вертикали таблицы - *объекты*.

Объект описывается как набор атрибутов.

Объект также известен как *запись*, случай, пример, строка таблицы и т.д.

Атрибут - свойство, характеризующее *объект*.

Например: цвет глаз, возраст.

Атрибут также называют переменной, полем таблицы, характеристикой.

Переменная (variable) - свойство или характеристика, общая для всех изучаемых *объектов*, проявление которой может изменяться от *объекта* к *объекту*.

Значение (value) переменной является проявлением признака.

	Атрибуты			
	Код клиента	Возраст	Семейное положение	Доход
	1	18	Single	125
	2	22	Married	100
	3	30	Single	70
	4	32	Married	120
Объекты	5	24	Divorced	95
	6	25	Married	60
	7	32	Divorced	220
	8	19	Single	85

Количественные переменные

- Дискретные данные являются значениями признака, общее число которых конечно либо бесконечно, но может быть подсчитано при помощи натуральных чисел от одного до бесконечности. (*продолжительность тренировки 10, 15, 25 мин.*)
- Непрерывные данные - данные, значения которых могут принимать какое угодно значение в некотором интервале. Измерение непрерывных данных предполагает большую точность. (*температура, высота, вес, длина и т.д.*)

Качественные (номинативные) переменные

Такие переменные используются для разделения наших испытуемых или наблюдений на группы.

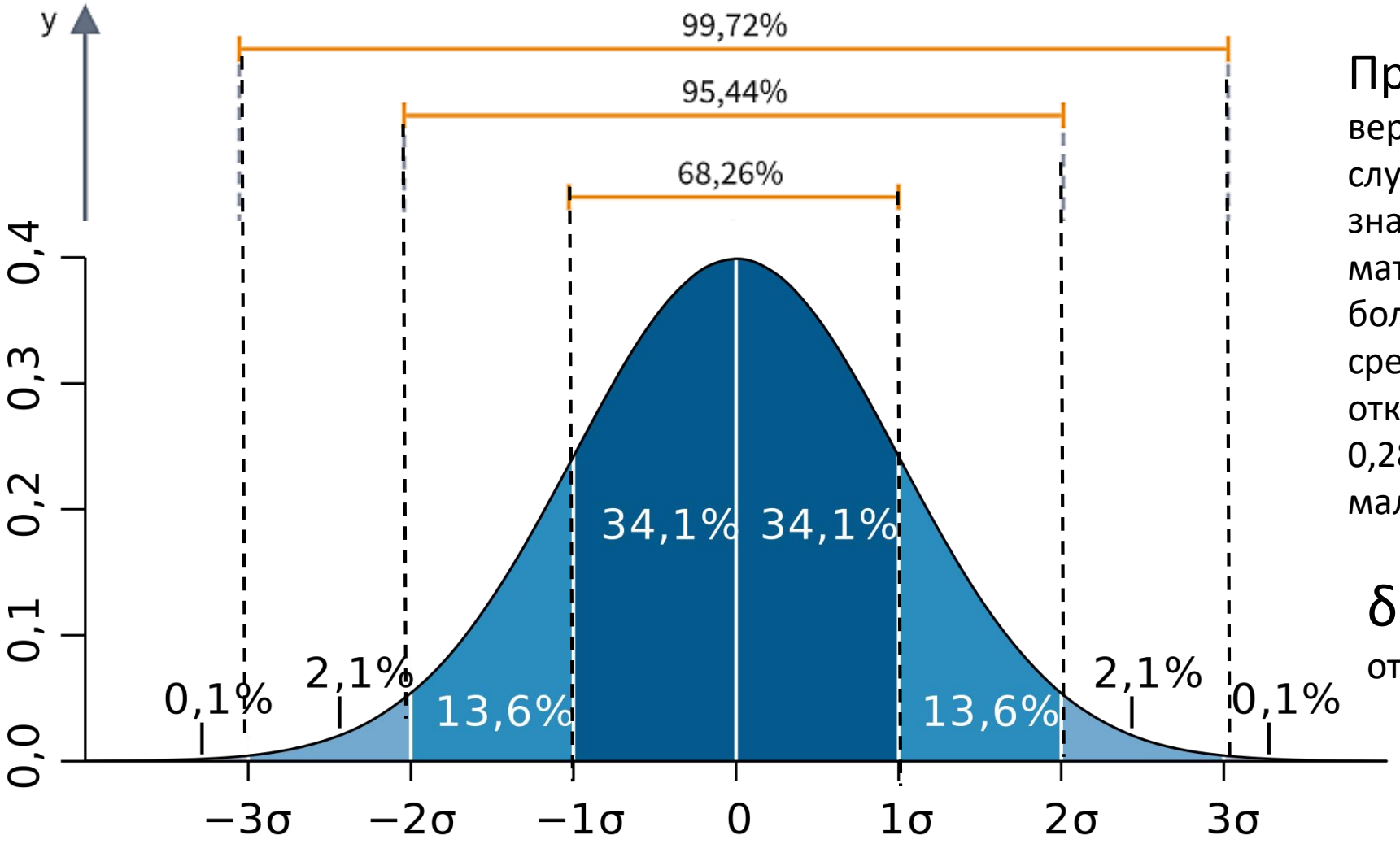
> *Например, мы можем сказать, что все участники эксперимента женского пола будут обозначены цифрой 1, а все участники мужского пола - цифрой 2 соответственно. Таким образом, в случае номинативных переменных за цифрами не стоит никакого математического смысла. В данном случае цифры используются как маркеры различных смысловых групп, в отличие от количественных переменных.*

Ранговые переменные

> *Представьте, что у нас есть информация о марафонском забеге: кто прибежал в каком порядке. Мы можем сказать, что испытуемый с рангом 1 быстрее, выше, сильнее испытуемого с рангом 5. Но вот насколько или во сколько он опережает этого испытуемого мы сказать не можем. Единственной возможной математической операцией является сравнение - кто быстрее, а кто медленнее.*

Нормальное распределение

1. Среднее, медиана и мода равны.
2. Кривая симметрична
3. Площадь под кривой равна 1.

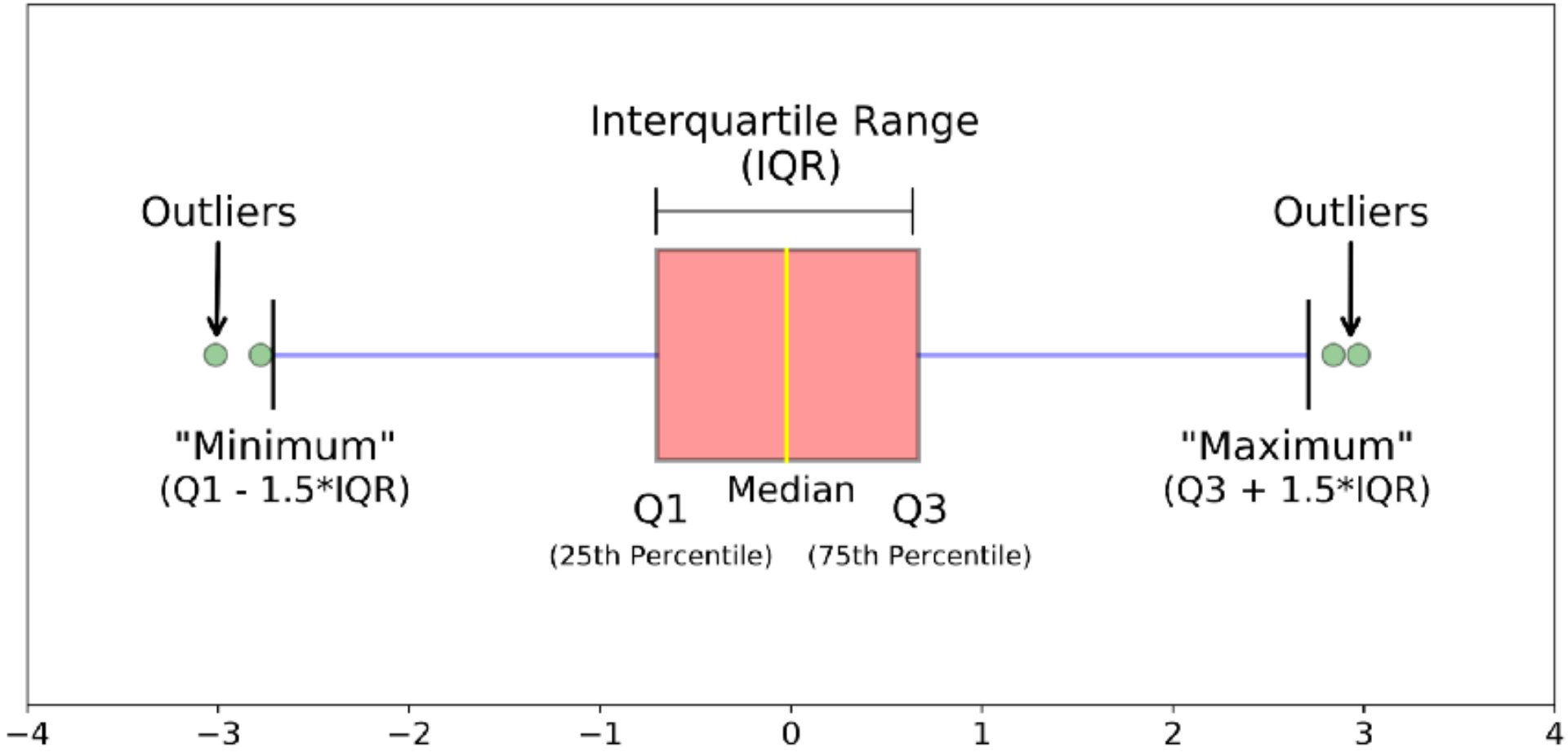


Правило трёх сигм:

вероятность того, что случайная величина примет значение, отклоняющееся от математического ожидания больше чем на три среднеквадратических отклонения, не превышает 0,28%, т. е. пренебрежимо мала.

δ — среднеквадратичное отклонение

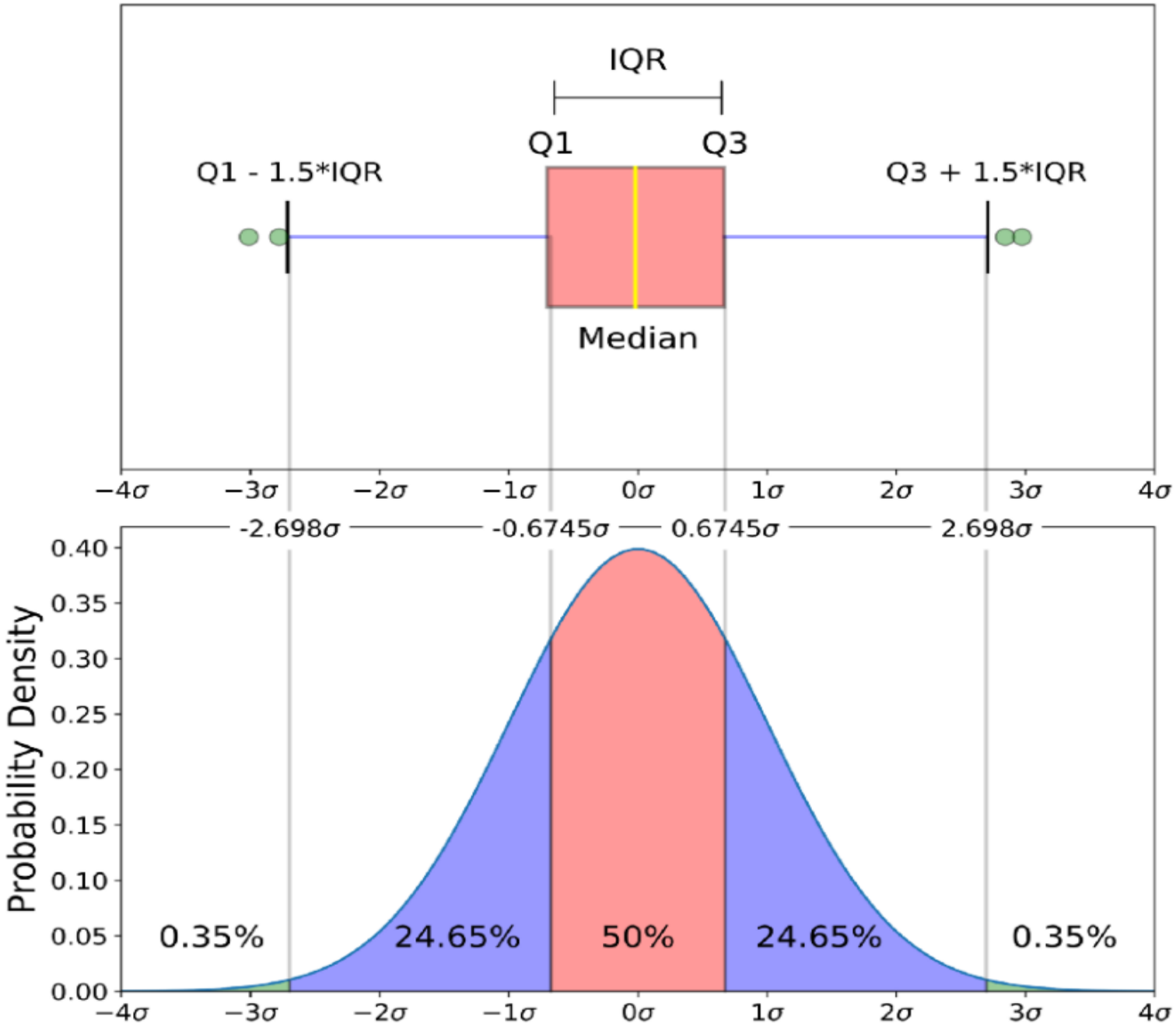
box plot Ящик с усами Диаграмма размаха



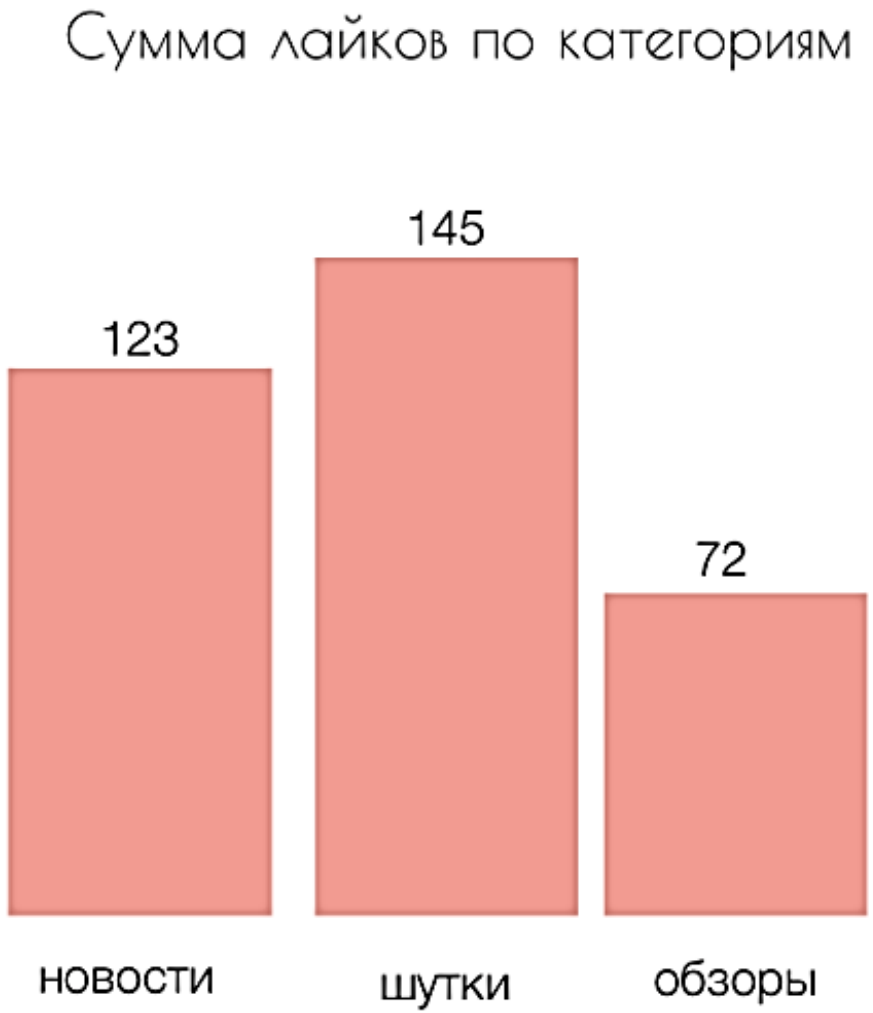
IQR - размах

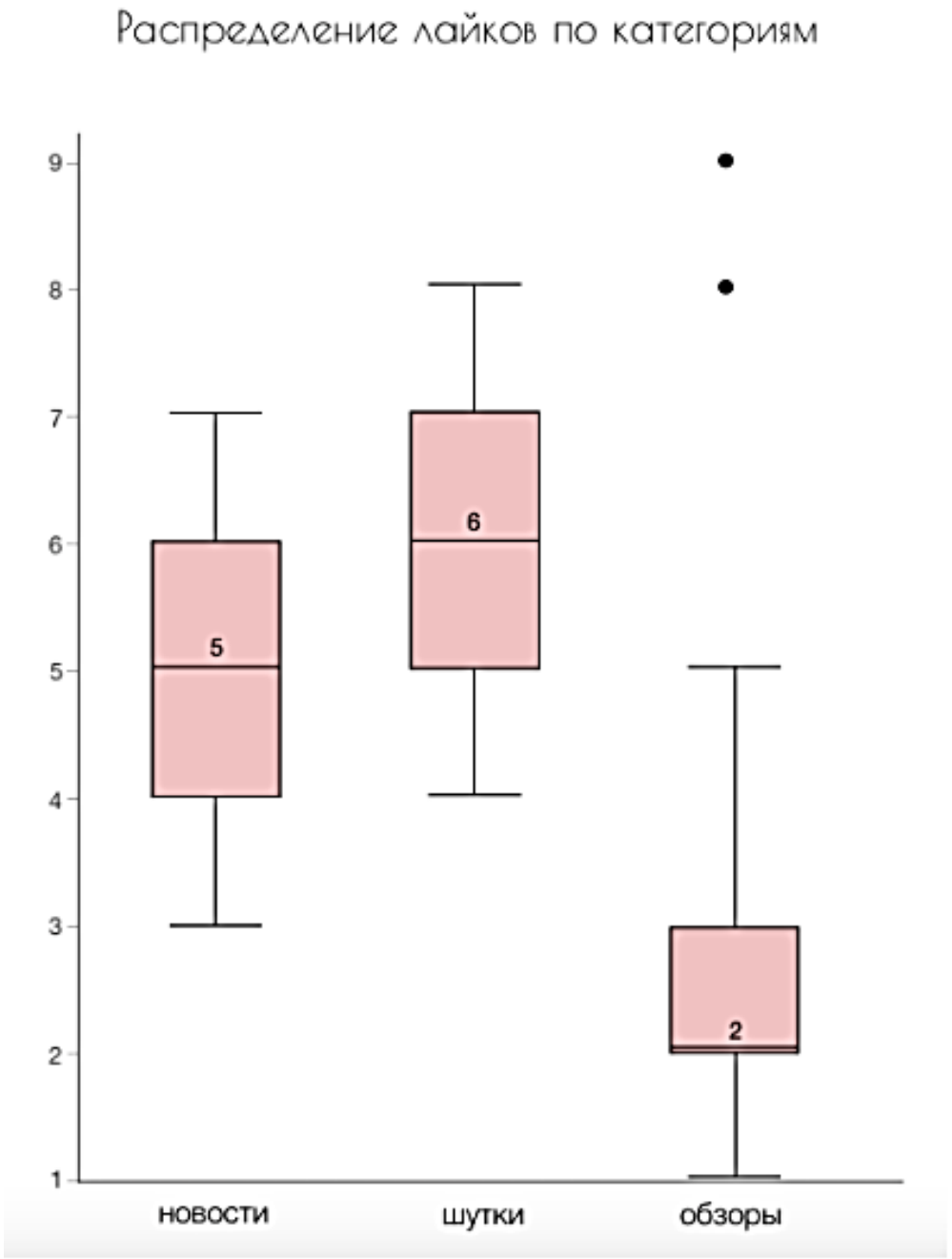
Q1 – 25 перцентиль (1й квартиль)

Q3 – 75 перцентиль (3й квартиль)

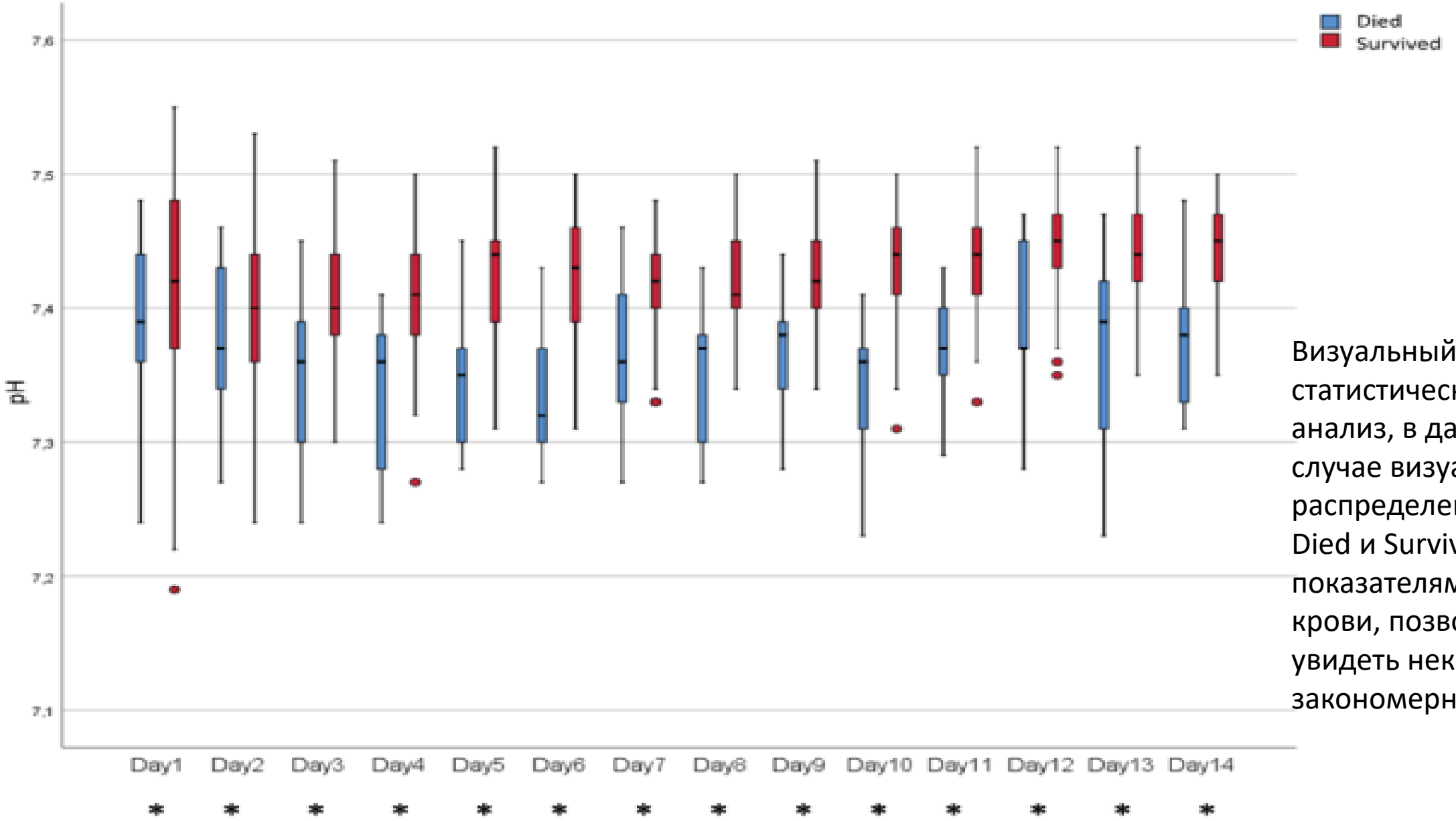

IQR
Q1
Q3
Q1 - 1.5*IQR
Q3 + 1.5*IQR
Median
−4σ
−3σ
−2σ
−1σ
0σ
1σ
2σ
3σ
4σ
-2.698σ
-0.6745σ
0.6745σ
2.698σ
Probability Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.35%
24.65%
50%
24.65%
0.35%

Пример: анализ эффективных постов блога





pH крови позволяет прогнозировать вероятность смерти от ковида



Визуальный статистический анализ, в данном случае визуализация распределения групп Died и Survived по показателям pH крови, позволяет увидеть некоторые закономерности.

Предобработка данных

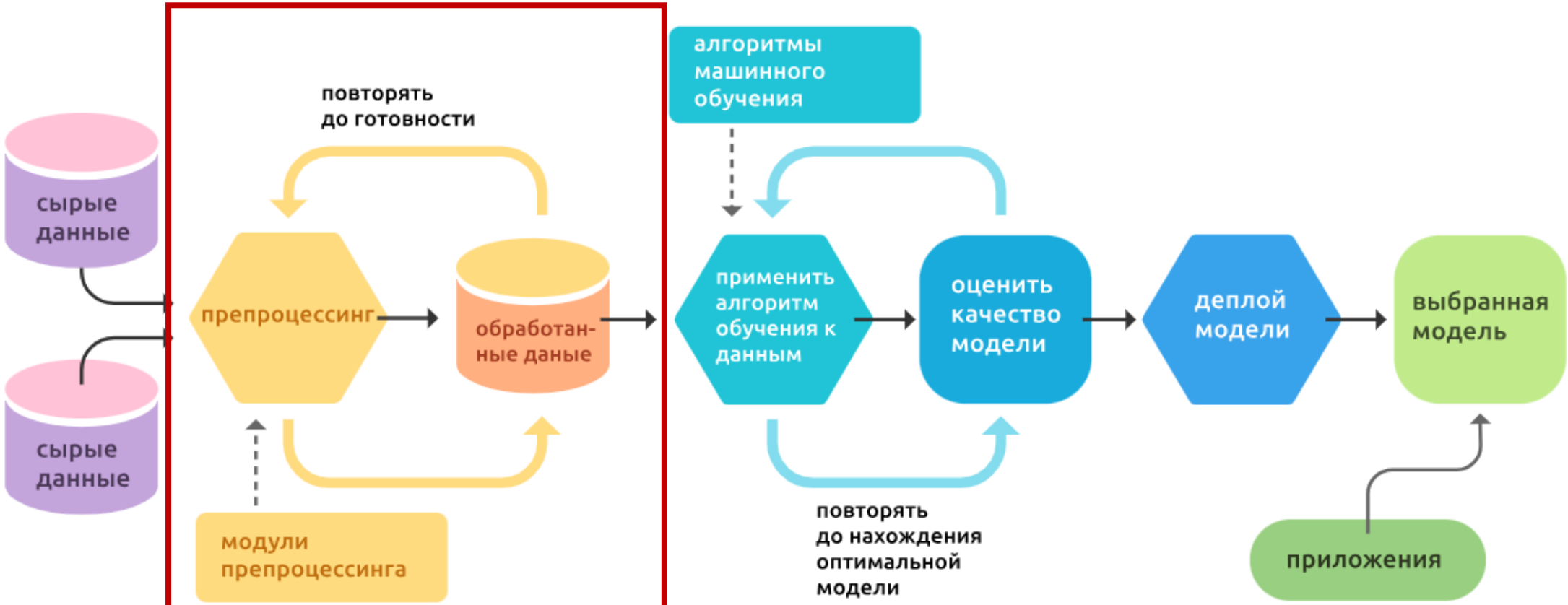
Все в науке о данных начинается с данных.

Предобработка данных является важнейшим этапом, и если она не будет выполнена, то дальнейший анализ в большинстве случаев невозможен из-за того, что аналитические алгоритмы просто не смогут работать или результаты их работы будут некорректными. Иными словами, реализуется принцип:

GIGO — garbage in, garbage out (мусор на входе, мусор на выходе).

По оценкам специалистов 80% времени уходит на подготовку данных**!**

Предобработка данных — это ключевой этап в процессе анализа данных, который позволяет обнаружить и исправить ошибки, преобразовать данные в удобный для анализа формат и создать новые признаки для улучшения качества анализа и моделирования.



Среди проблем, вызывающих снижение качества данных, можно выделить следующие:

- пропущенные значения;
- дубликаты;
- противоречия;
- аномальные значения и выбросы;
- шум;
- некорректные форматы и представления данных;
- ошибки ввода данных.

	0	1	2	3	4	5	6	7
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
5	5	116	74	0	0	25.6	0.201	30
6	3	78	50	32	88	31.0	0.248	26
7	10	115	0	0	0	35.3	0.134	29
8	2	197	70	45	543	30.5	0.158	53
9	8	125	96	0	0	0.0	0.232	54

Пропущенные значения

Реальные наборы данных могут иметь пропуски значений параметров. Это может случиться из-за технических проблем, или если датасет собран из нескольких источников с разными наборами параметров; важно то, что в таблице присутствуют пустые ячейки

Примеры:

- *Отзывы на кинопоиске*

Случайность пропусков

Предполагается, что пропуски в матрице располагаются случайно

Примеры исключений:

- *отказ респондентов отвечать на вопросы*

Для работы с отсутствующими значениями **нужно использовать интуицию**, чтобы выяснить, почему значение отсутствует.

Это значение отсутствует, потому что оно не было записано или потому что оно не существует?

- ✓ Если значение отсутствует, потому что оно не существует (например, наличие детей). Эти значения можно сохранить как NaN.
- ✓ Если значение отсутствует из-за того, что оно не было записано, можно попытаться угадать, что это могло быть, основываясь на других значениях в этом столбце и строке.

Если пропусков много — тренировка на таких данных сильно ухудшит качество модели, а то и окажется вовсе невозможной. **Многие алгоритмы машинного обучение** не только **требуют массив чисел (а не NaN или «missing»),** но и ожидают, что этот массив будет состоять из валидных данных, а в случае наличия пропущенных значений в массиве будет выбрасываться исключение.

Как можно выявить пропущенные данные?

Функция **info()** выводит информацию о количестве ненулевых значений по столбцам

```
Ввод [56]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       768 non-null    int64
 1   1       763 non-null    float64
 2   2       733 non-null    float64
 3   3       541 non-null    float64
 4   4       394 non-null    float64
 5   5       757 non-null    float64
 6   6       768 non-null    float64
 7   7       768 non-null    int64
dtypes: float64(6), int64(2)
memory usage: 48.1 KB
```

Подсчет количества нулевых значений по столбцам

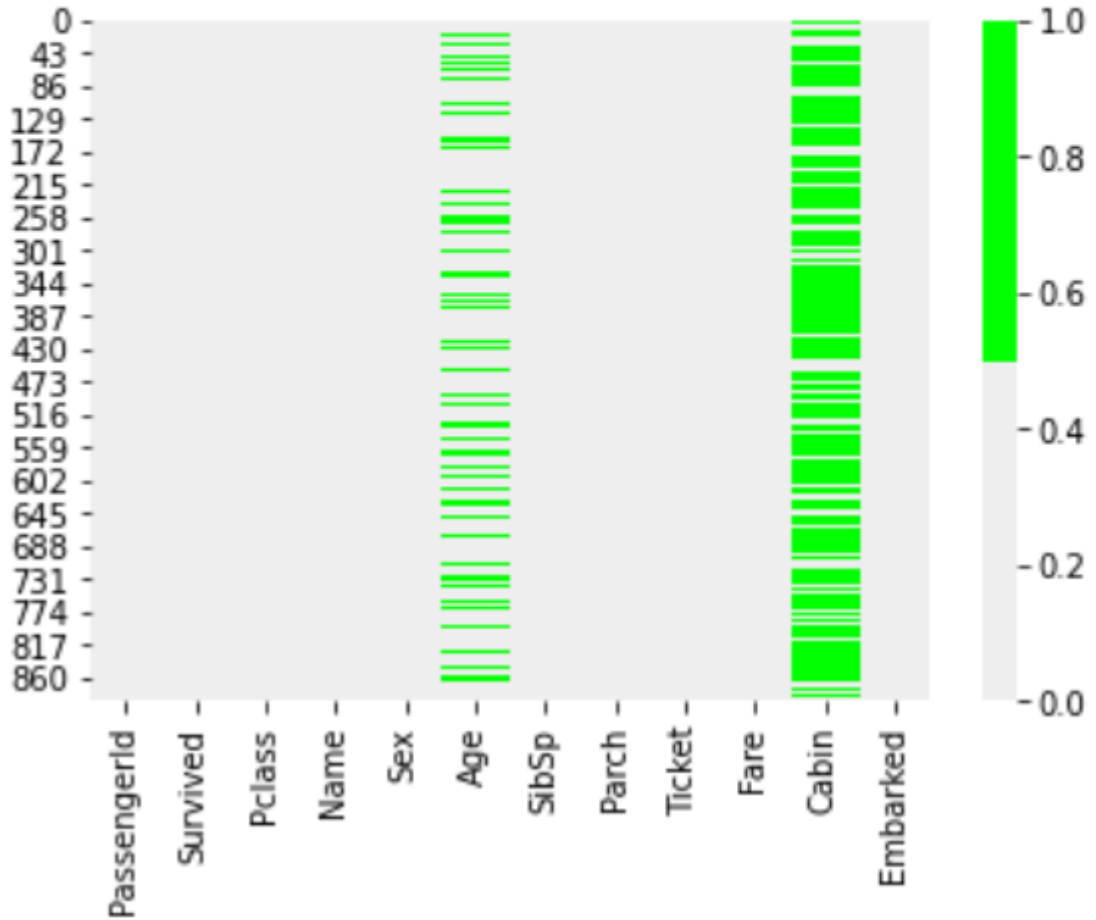
```
Ввод [48]: data[data.eq(0)].count()

Out[48]: 0    111
         1      5
         2     35
         3    227
         4    374
         5     11
         6      0
         7      0
         dtype: int64
```

Как можно выявить пропущенные данные?

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
```

```
cols = data.columns[:]
# определяем цвета
# красный - пропущенные данные
colours = ['#eeeeee', '#00ff00']
sns.heatmap(data[cols].isnull(), cmap=sns.color_palette(colours))
```

```
<AxesSubplot:>
```



Что делать с пропущенными значениями?

1. Исключение строк с пропущенными значениями стало решением по умолчанию в некоторых популярных прикладных пакетах. НО такой метод применим только в том случае, когда малая часть объектов выборки имеет пропущенные значения. Преимуществом данного подхода является простота и невозможность испортить данные путем замены пропусков. В случае достаточно большого размера выборки метод может показывать хорошие результаты.

2. Замена пропусков зачастую и вполне обоснованно кажется более предпочтительным решением. Однако это не всегда так. Неудачный выбор метода заполнения пропусков может не только не улучшить, но и сильно ухудшить результаты.

Существует множество способов замены данных со своими их преимуществами и недостатками.

Замена значений

Методы замены пропусков на основе имеющейся информации часто объединяют в одну группу, называемую ***Single-imputation methods*.**

Замена данных средним/модой

В случае категориального признака все пропуски заменяются на наиболее часто встречающее значение, в случае количественного признака – на среднее значение по признаку. Данный метод позволяет учитывать имеющиеся данные.

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

mean()

	col1	col2	col3	col4	col5
0	2.0	5.0	3.0	6.0	7.0
1	9.0	11.0	9.0	0.0	7.0
2	19.0	17.0	6.0	9.0	7.0

Плюсы:

- Просто и быстро.
- Хорошо работает на небольших наборах численных данных.

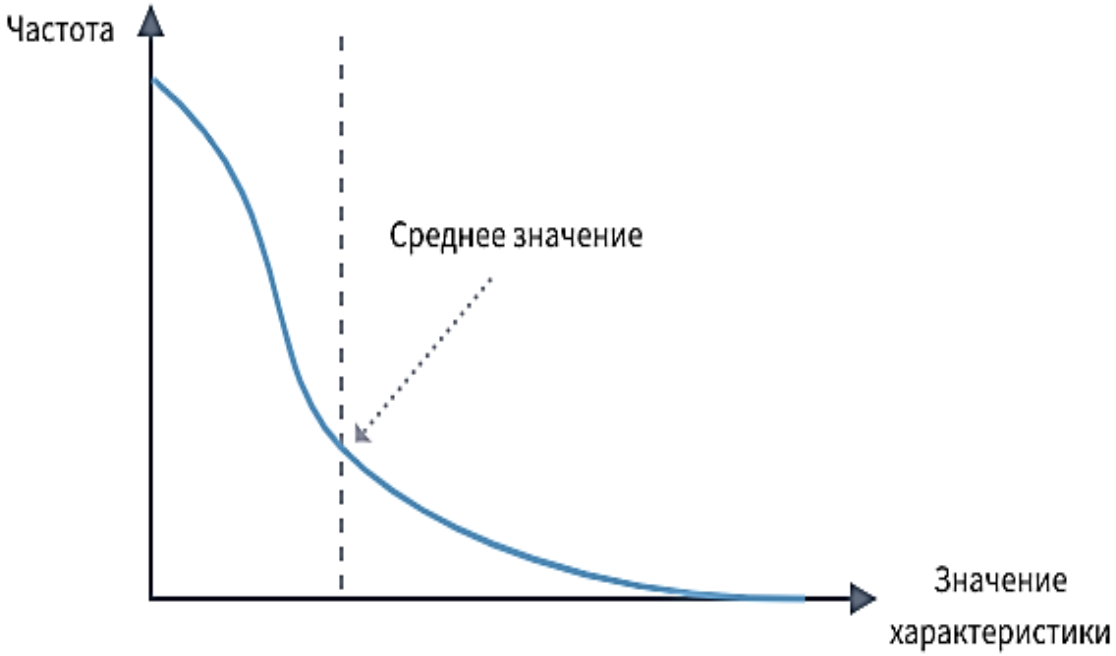
Минусы:

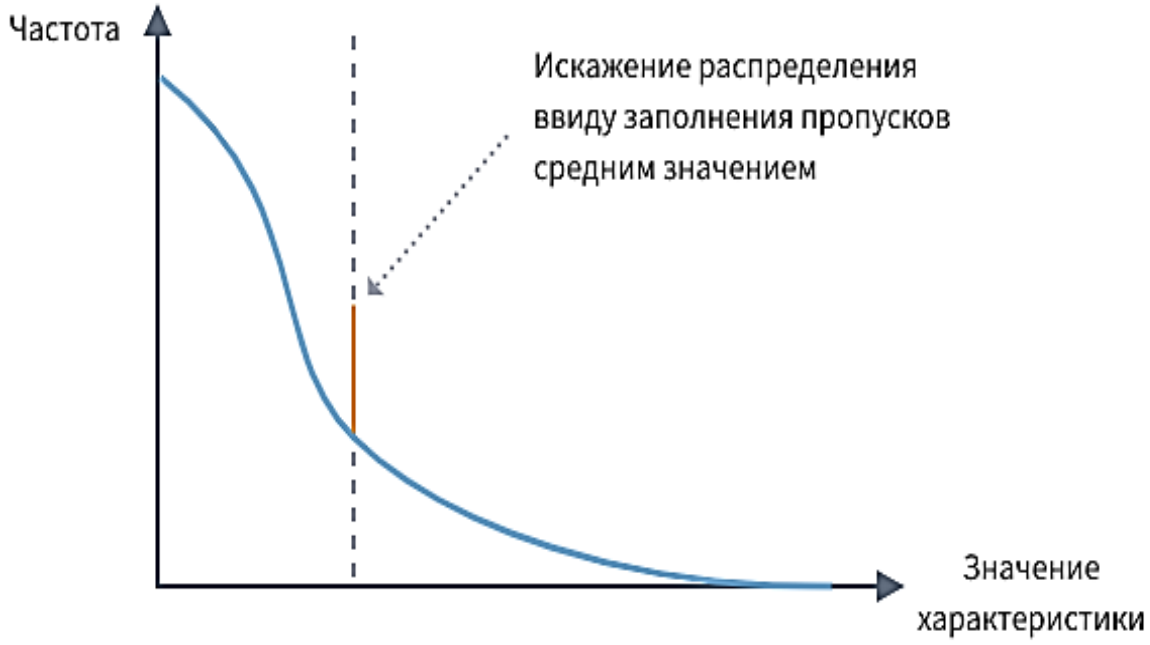
- Значения вычисляются независимо для каждого столбца, так что корреляции между параметрами не учитываются.
- Не работает с качественными переменными.
- Метод не особенно точный.

Замена значений

При заполнении пропусков средним значением, модой, нулем или медианой свойственны одни и те же **недостатки**.

На рисунках: распределение значений непрерывной характеристики до заполнения пропусков **средним значением** и после него.

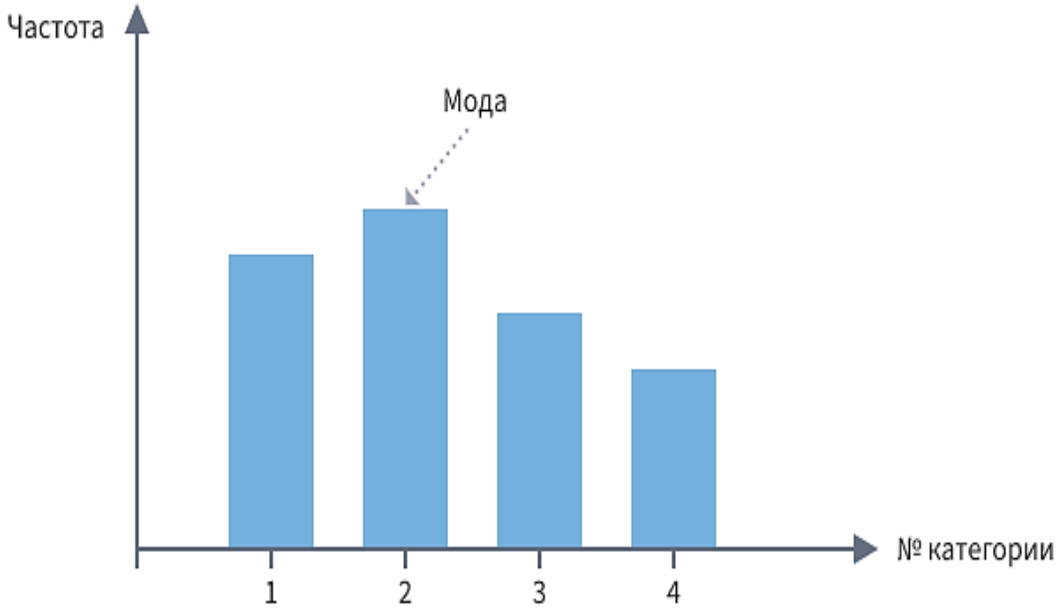


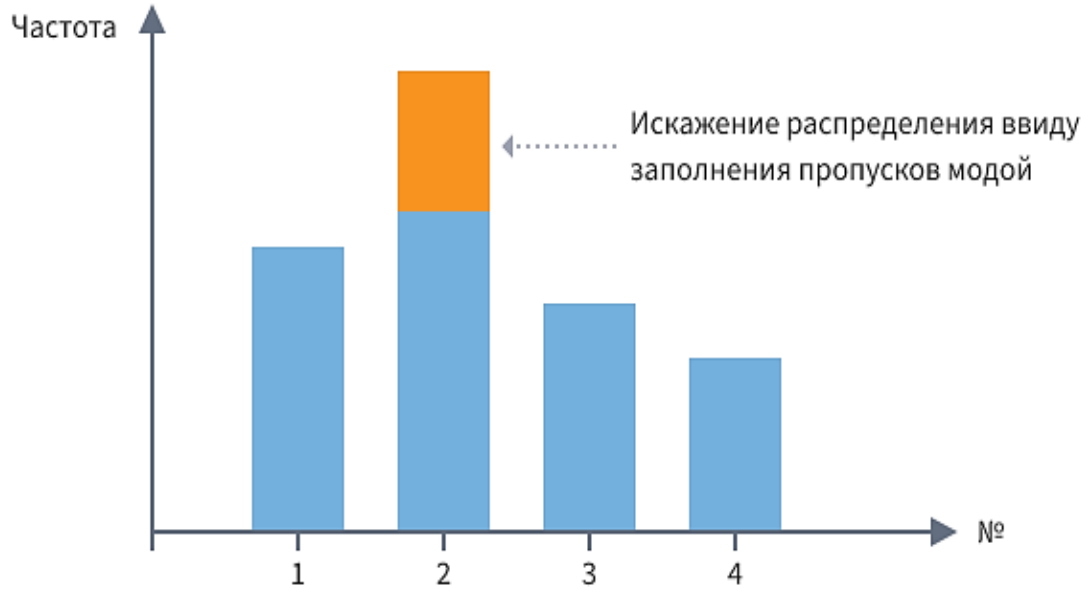


Данный метод приводит к существенному искажению распределения характеристики.
Это в итоге проявляется в искажении всех показателей, характеризующих свойства распределения (кроме среднего значения), заниженной корреляции и завышенной оценке стандартных отклонений.

Замена значений

В случае категориальной дискретной характеристики наиболее часто используется **заполнение модой.**
На рисунке 2 показано распределение категориальной характеристики до и после заполнения пропусков.





При заполнении пропусков категориальной характеристики модой проявляются те же недостатки, что и при заполнении пропусков непрерывной характеристики средним арифметическим (нулем, медианой и тому подобным).ы

Замена значений

Замена NaN средним значением

data.fillna(data.mean())

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	NaN	NaN	NaN	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	NaN	NaN	NaN	0.232	54

	0	1	2	3	4	5	6	7
0	6	148.0	72.000000	35.00000	155.548223	33.600000	0.627	50
1	1	85.0	66.000000	29.00000	155.548223	26.600000	0.351	31
2	8	183.0	64.000000	29.15342	155.548223	23.300000	0.672	32
3	1	89.0	66.000000	23.00000	94.000000	28.100000	0.167	21
4	0	137.0	40.000000	35.00000	168.000000	43.100000	2.288	33
5	5	116.0	74.000000	29.15342	155.548223	25.600000	0.201	30
6	3	78.0	50.000000	32.00000	88.000000	31.000000	0.248	26
7	10	115.0	72.405184	29.15342	155.548223	35.300000	0.134	29
8	2	197.0	70.000000	45.00000	543.000000	30.500000	0.158	53
9	8	125.0	96.000000	29.15342	155.548223	32.457464	0.232	54

Замена значений

Замена NaN следующим значением

data.fillna(value=None, method="bfill")

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	NaN	NaN	NaN	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	NaN	NaN	NaN	0.232	54

	0	1	2	3	4	5	6	7
0	6	148.0	72.0	35.0	94.0	33.6	0.627	50
1	1	85.0	66.0	29.0	94.0	26.6	0.351	31
2	8	183.0	64.0	23.0	94.0	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
5	5	116.0	74.0	32.0	88.0	25.6	0.201	30
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26
7	10	115.0	70.0	45.0	543.0	35.3	0.134	29
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53
9	8	125.0	96.0	23.0	846.0	37.6	0.232	54

Замена пропусков новым «плейсхолдером»

Замена значений

- ✓ Пропуски в категориальных переменных можно закодировать новой категорией (например, MISSING)

- ✓ Пропуски в числовых признаках можно заполнить числом -999
  (*к примеру, при использовании метода решающих деревьев, такие данные попадут в отдельный лист*).

```
# категориальные признаки
df[‘Title1'] = df[' Title1'].fillna('MISSING')

# численные признаки
df['l Title2'] = df[' Title2'].fillna(-999)
```

Таким образом, можно сохранить данные о пропущенных значениях, что тоже может быть ценной информацией.

https://techrocks.ru/2020/04/02/data-cleaning-with-python/

Замена значений

Повторение результата последнего наблюдения

LOCF (Last observation carried forward) — повторение результата последнего наблюдения. Данный метод применяется, как правило, при заполнении пропусков **во временных рядах**, когда последующие значения априори сильно взаимосвязаны с предыдущими.

Когда применение LOCF обосновано.

Пример: если мы измеряем температуру воздуха в некоторой географической точке на открытом пространстве, причем измерения проводятся каждую минуту, то при нормальных условиях — если исключить природные катаклизмы — измеряемая величина априори не может резко (на 10–20 °C) измениться за столь короткий интервал времени между последующими измерениями. Следовательно, заполнение пропусков предшествующим известным значением в такой ситуации обоснованно.

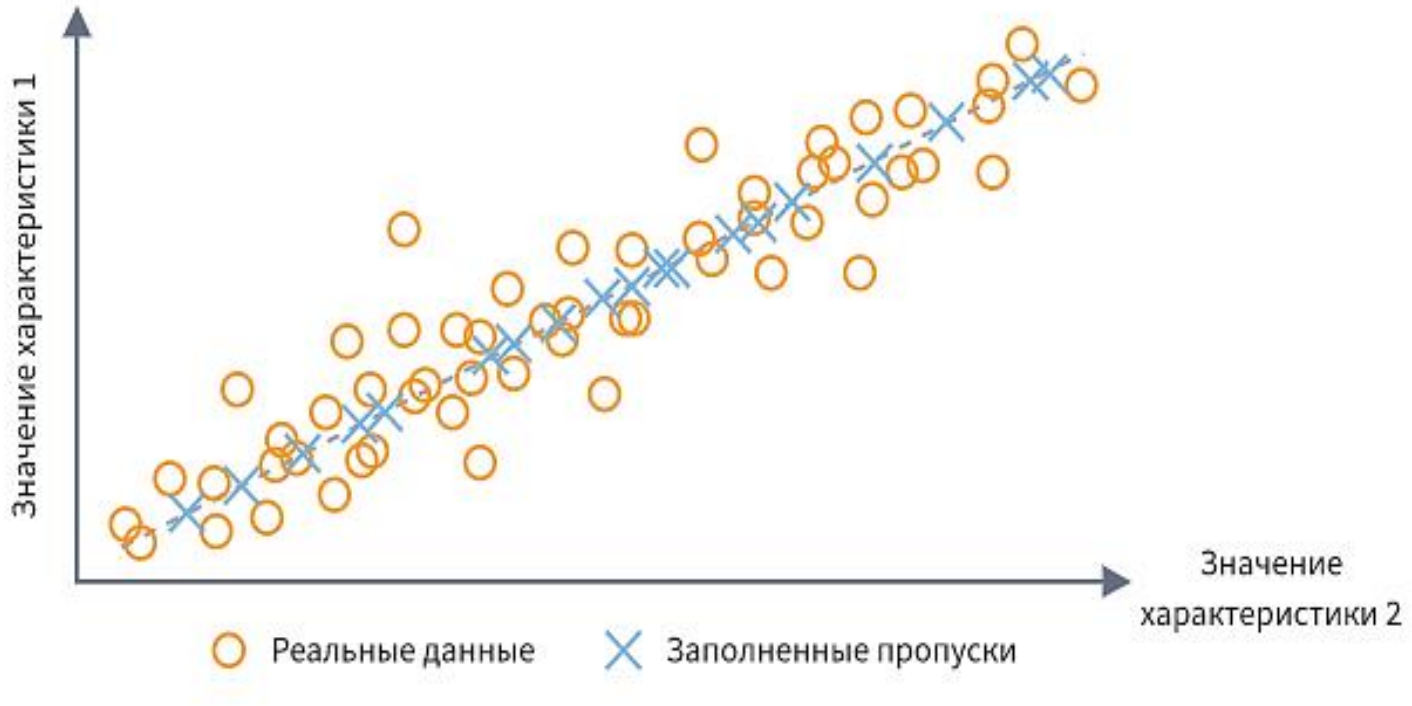
Хотя в описанной выше ситуации метод логичен и обоснован, он тоже может привести к существенным искажениям статистических свойств. Так, возможна ситуация, когда применение LOCF приведет к дублированию выброса (заполнению пропусков аномальным значением). Кроме того, если в данных много последовательно пропущенных значений, то гипотеза о небольших изменениях уже не выполняется и, как следствие, использование LOCF приводит к неправильным результатам.

Восстановление пропусков на основе регрессионных моделей

Данный метод заключается в том, что пропущенные значения заполняются с помощью модели линейной регрессии, построенной на известных значениях набора данных.

На рисунке показан пример результатов заполнения пропущенных значений характеристики 1 на основе известных значений характеристики 2.

Метод линейной регрессии позволяет получить правдоподобно заполненные данные.



Однако реальным данным свойственен некоторый разброс значений, который при заполнении пропусков на основе линейной регрессии отсутствует. Как следствие, вариация значений характеристики становится меньше, а корреляция между характеристикой 2 и характеристикой 1 искусственно усиливается.

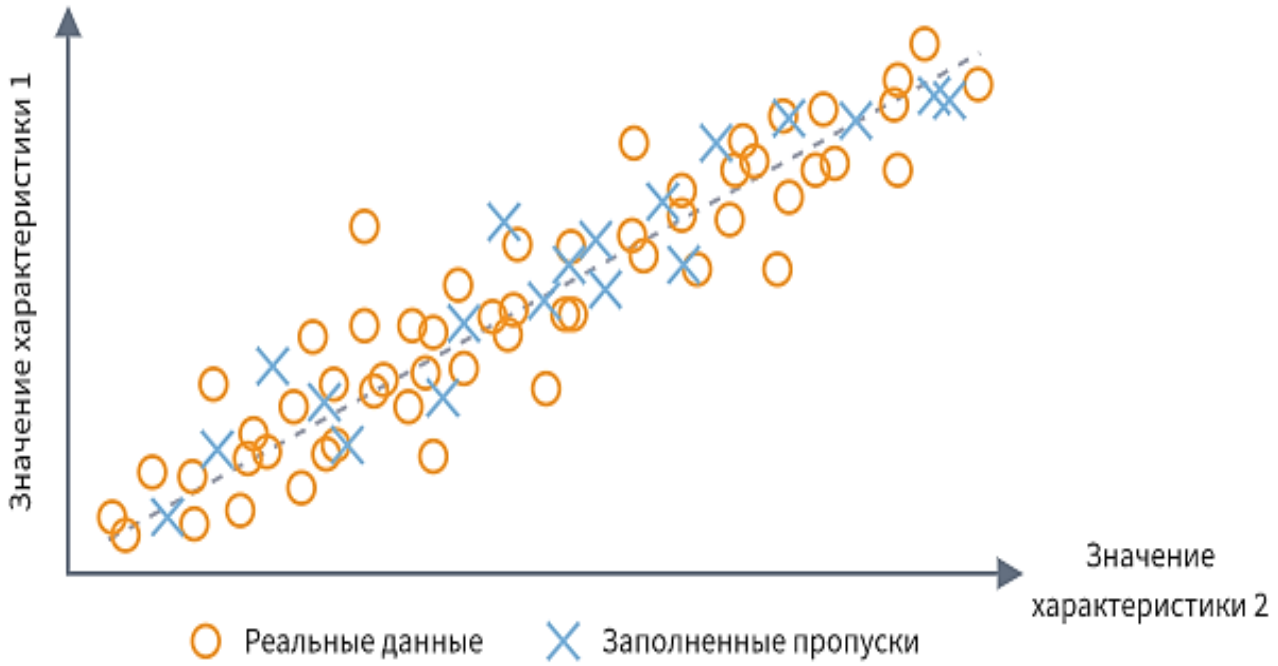
В результате данный метод заполнения пропусков становится тем хуже, чем выше вариация значений характеристики, пропуски в которой мы заполняем, и чем выше процент пропущенных строк.

Замена значений

Есть метод, решающий эту проблему: метод стохастической линейной регрессии.

На рисунке — **заполнение пропусков на основе стохастической линейной регрессии**

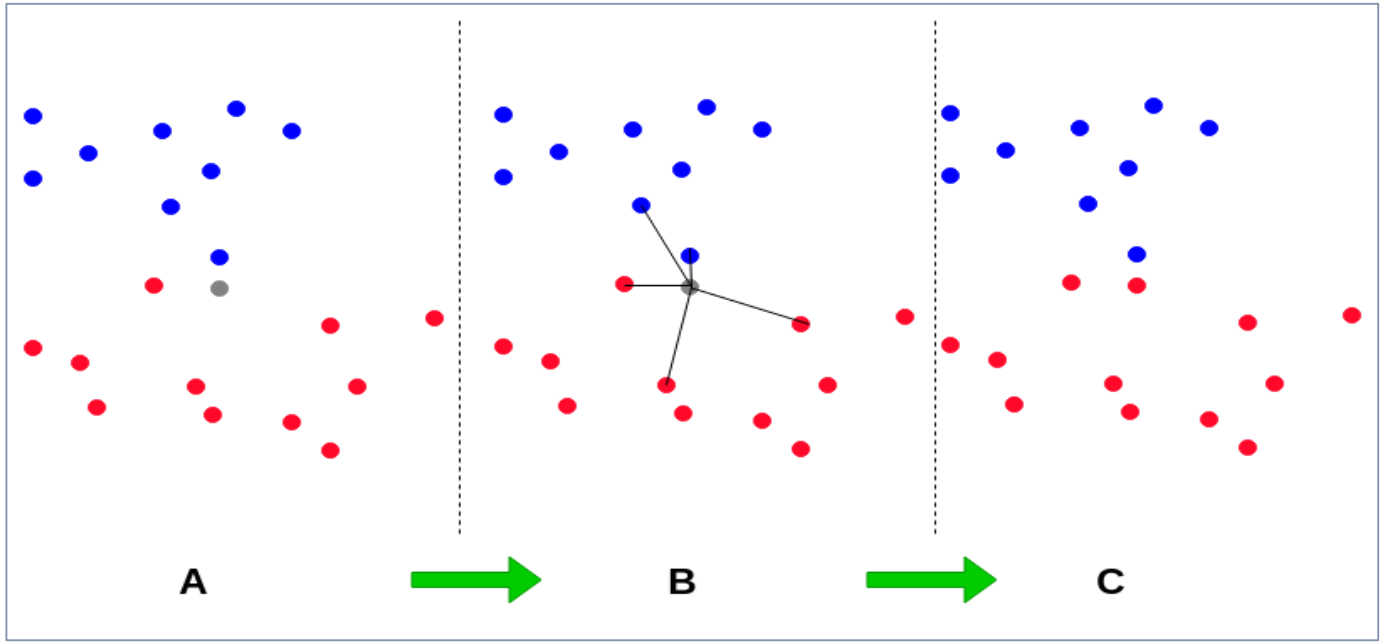
Модель стохастической линейной регрессии отражает не только линейную зависимость между характеристиками, но и отклонения от этой линейной зависимости. Этот метод обладает положительными свойствами заполнения пропусков на основе линейной регрессии и, кроме того, не так сильно искажает значения коэффициентов корреляции.



Заполнение пропусков с помощью стохастической линейной регрессии в общем случае приводит к наименьшим искажениям статистических свойств выборки.
А в случае, когда между характеристиками прослеживаются явно выраженные линейные зависимости, метод стохастической линейной регрессии нередко превосходит даже более сложные методы.

Замена данных с помощью метода k-NN

k-Nearest Neighbour (k ближайших соседей) — простой алгоритм классификации, он использует сходство точек, чтобы предсказать недостающие значения на основании *k* ближайших точек, у которых это значение есть. Иными словами, выбирается *k* точек, которые больше всего похожи на рассматриваемую, и уже на их основании выбирается значение для пустой ячейки.



Плюсы:

- На некоторых датасетах может быть точнее среднего/медианы или константы.
- Учитывает корреляцию между параметрами.

Минусы:

- Вычислительно дороже, так как требует держать весь набор данных в памяти.
- Важно понимать, какая метрика дистанции используется для поиска соседей.
- Может потребоваться предварительная нормализация данных.
- Чувствителен к выбросам в данных.

2. Аномальное значение (Outlier value)

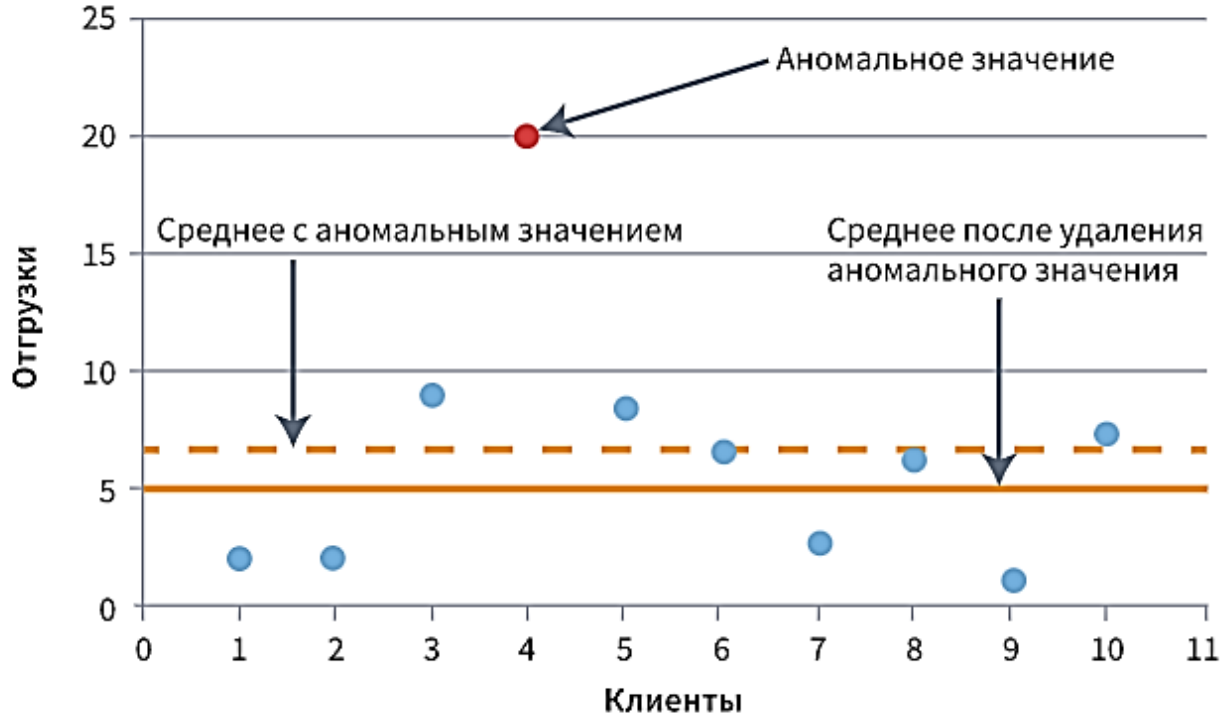
Выбросы – это данные, которые существенно отличаются от других наблюдений. Они могут соответствовать реальным отклонениям, но могут быть и просто ошибками.

Выбросы необходимо подавить или удалить, поскольку они могут вызвать некорректную работу алгоритмов и привести к искажению результатов анализа данных.

Степень устойчивости алгоритма к наличию в данных выбросов называется **робастностью**.

Пример 1: искажение статистических показателей

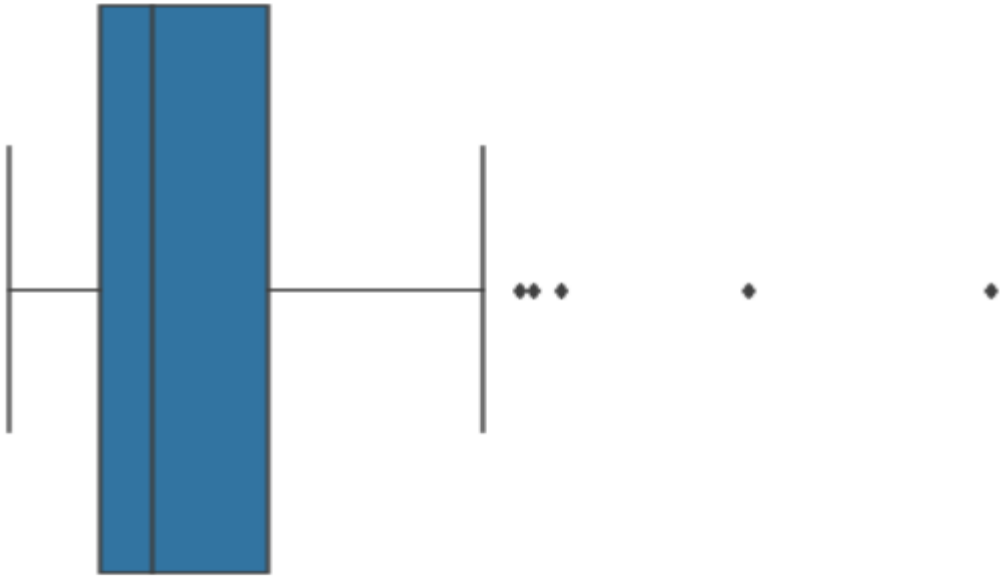
Вопрос: как оценить данные на наличие выбросов



2.1. Как обнаружить выбросы?

Для численных и категориальных признаков используются разные методы изучения распределения, позволяющие обнаружить выбросы.

2.1.1. При помощи box-plot



2.1. Как обнаружить выбросы?

2.1.2. Описательная статистика

Можно проанализировать описательную статистику.

Например, для признака видно, что максимальное значение равно 7478, в то время как 75% квартиль равен только 43. Значение 7478 – выброс.

```
df['life_sq'].describe()
```

```
count    24088.000000
mean        34.403271
std         52.285733
min          0.000000
25%         20.000000
50%         30.000000
75%         43.000000
max       7478.000000
Name: life_sq, dtype: float64
```

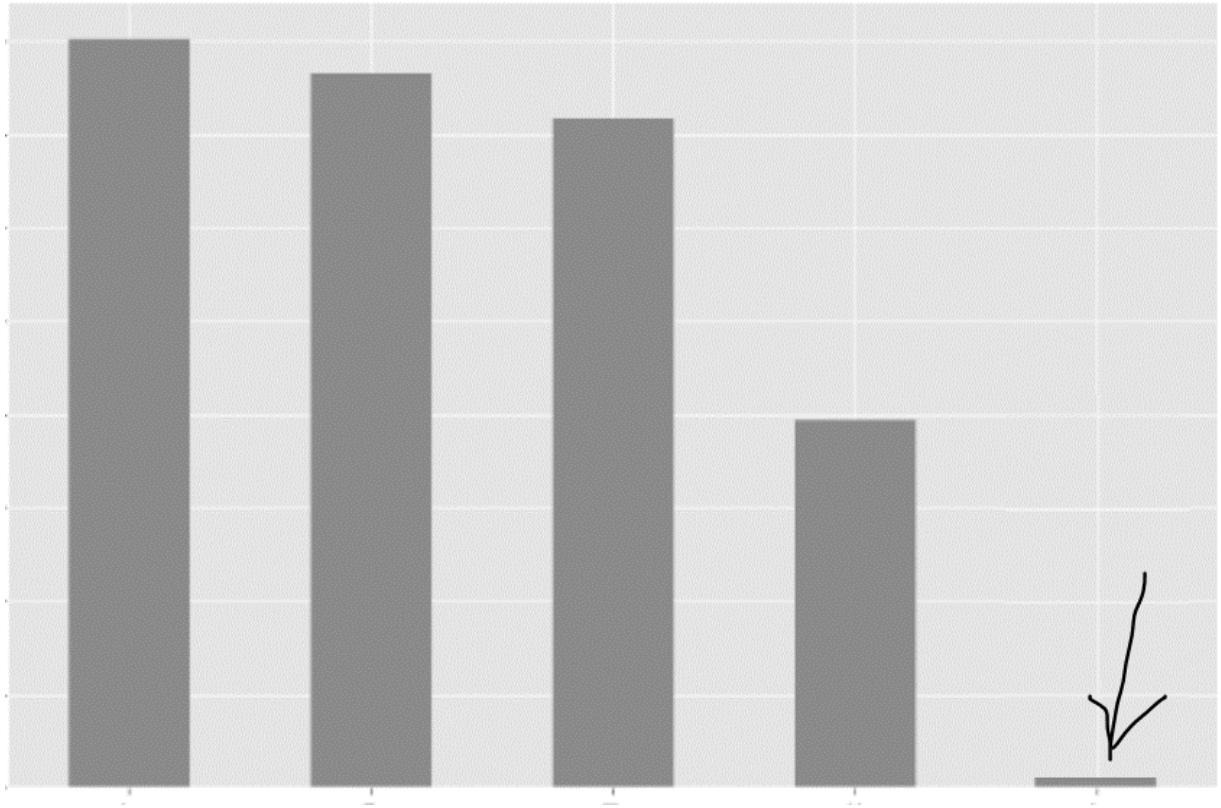
https://techrocks.ru/2020/04/02/data-cleaning-with-python/

2.1. Как обнаружить выбросы?

2.1.3. Столбчатая диаграмма

Для категориальных признаков можно построить столбчатую диаграмму – для визуализации данных о категориях и их распределении.

Например, на рисунке распределение признака вполне равномерно и допустимо. Но если существует категория только с одним значением, то это будет выброс.



3. Неинформативные данные

Вся информация, поступающая в модель, должна служить целям проекта. Если она не добавляет никакой ценности, от нее следует избавиться.

Три основных типа «ненужных» данных:

1. неинформативные признаки с большим количеством одинаковых значений,
2. нерелевантные признаки,
3. дубликаты записей.

3.1 Неинформативные признаки

Если признак имеет слишком много строк с одинаковыми значениями, он не несет полезной информации для проекта.

Как обнаружить?

Можно составить список признаков, у которых более 95% строк содержат одно и то же значение.

```
num_rows = len(df.index)
low_information_cols = []

for col in df.columns:
    cnts = df[col].value_counts(dropna=False)
    top_pct = (cnts/num_rows).iloc[0]

    if top_pct > 0.95:
        low_information_cols.append(col)
        print('{0}: {1:.5f}%'.format(col, top_pct*100))
        print(cnts)
        print()
```

Что делать?

Если после анализа причин получения повторяющихся значений пришли к выводу, что признак не несет полезной информации, используйте drop().

3.2 Нерелевантные признаки (син. *неважный, незначимый*)

Нерелевантные признаки обнаруживаются ручным отбором и оценкой значимости.

Например в датасете по ценам на жилье в России, признак, регистрирующий температуру воздуха в Торонто точно не имеет никакого отношения к прогнозированию цен на российское жилье. Если признак не имеет значения для проекта, его нужно исключить.

Часто бывает так, что признаки довольно сильно зависят друг от друга и их одновременное наличие избыточно. Например, у вас есть две переменные - «время, проведенное на беговой дорожке в минутах» и «сожженные калории». Эти переменные сильно взаимосвязаны: чем больше времени вы проводите на беговой дорожке, тем больше калорий вы сжигаете. Следовательно, нет смысла хранить оба, поскольку только один из них уже содержит необходимую информацию.

3.3 Дубликаты записей

Если значения признаков (всех или большинства) в двух разных записях совпадают, эти записи называются дубликатами.

Как обнаружить повторяющиеся записи?

Способ обнаружения дубликатов зависит от того, что именно мы считаем дубликатами.

- ✓ Например, в наборе данных есть уникальный идентификатор id. Если две записи имеют одинаковый *id*, то считаем, что это одна и та же запись. Удаляем все неуникальные записи.
- ✓ Другой распространенный способ вычисления дубликатов: по набору ключевых признаков. *Например, неуникальными можно считать записи с одной и той же площадью жилья, ценой и годом постройки.*

Что делать с дубликатами?

Очевидно, что повторяющиеся записи нам не нужны, значит, их нужно исключить из набора.

Большая проблема очистки данных – разные форматы записей.

Рассмотрим четыре самых распространенных несогласованности.

4.1 Разные регистры символов

Непоследовательное использование разных регистров в категориальных значениях является очень распространенной ошибкой, которая может существенно повлиять на анализ данных.

Как обнаружить?

Следующий код выводит все уникальные значения признака:

```
df['title'].value_counts(dropna=False)
```

Что делать?

Эта проблема легко решается принудительным изменением регистра:

```
# пусть все будет в нижнем регистре
df['title'] = df['title'].str.lower()
df['title'].value_counts(dropna=False)
```

4.2 Разные форматы данных

Ряд данных в наборе находится не в том формате, с которым нам было бы удобно работать. Например, даты, записанные в виде строки, следует преобразовать в формат DateTime.

Как обнаружить?

В примере признак *timestamp* представляет собой строку, хотя является датой:

Что же делать?

Значения признака *timestamp* следует преобразовать в удобный формат:

По ссылке можно найти код для данного примера

https://techrocks.ru/2020/04/02/data-cleaning-with-python/

	id	timestamp	full_sq	life_sq	floor	m
0	1	2011-08-20	43	27.0	4.0	
1	2	2011-08-23	34	19.0	3.0	
2	3	2011-08-27	43	29.0	2.0	
3	4	2011-09-01	89	50.0	9.0	
4	5	2011-09-05	77	77.0	4.0	

4.2 Разные форматы данных

Адреса

Мало кто следует стандартному формату, вводя свой адрес в базу данных.

Что делать?

Минимальное форматирование включает следующие операции:

- приведение всех символов к нижнему регистру;
- удаление пробелов в начале и конце строки;
- удаление точек;
- стандартизация формулировок: замена street на st, apartment на apt и т. д.

4.3 Опечатки

Опечатки в значениях категориальных признаков приводят к таким же проблемам, как и разные регистры символов. Например, признак city, а его значениями будут *torontoo* и *tronto*. В обоих случаях это опечатки, а правильное значение – *toronto*.

Как обнаружить?

Для обнаружения опечаток требуется особый подход.
Простой способ идентификации подобных элементов – *нечеткая логика или редактирование расстояния.* Суть этого метода заключается в измерении количества букв (расстояния), которые нам нужно изменить, чтобы из одного слова получить другое.

Что делать?

Можно установить критерии для преобразования этих опечаток в правильные значения.
Например, если расстояние некоторого значения от слова *toronto* не превышает 2 буквы, мы преобразуем это значение в правильное – *toronto*.

Несбалансированность классов

В машинном обучении нередко возникают ситуации, когда в обучающем наборе данных **доля примеров некоторого класса оказывается слишком низкой, а другого — слишком большой**. Эта ситуация в теории машинного обучения известна как несбалансированность классов (class imbalance), а классификация в условиях несбалансированности классов называется несбалансированной классификацией (unbalanced classification).

Несбалансированность классов как правило создаёт проблемы при решении задач классификации, поскольку построенные на таких данных модели имеют «перекос» в сторону преобладающего класса, т.е. с большей вероятностью присваивают его метку класса новым наблюдениям при практическом использовании модели.

- ✓ Сокращение числа примеров мажоритарного класса
- ✓ Увеличение числа примеров миноритарного класса

Подробнее https://loginom.ru/blog/imbalance-class

Прочие этапы предобработки данных

Трансформация данных заключается в оптимизации их представлений и форматов с точки зрения решаемых задач и целей анализа. Трансформация не ставит целью изменить информационное содержание данных. Ее задача — представить эту информацию в таком виде, чтобы она могла быть использована наиболее эффективно.
Вообще, трансформация данных — очень широкое понятие, не имеющее четко очерченных границ. Иногда этот термин иногда распространяют на любые манипуляции с данными, независимо от их целей и методов. Однако в контексте анализа данных трансформация данных имеет вполне конкретные цели и задачи, а также использует достаточно стабильный набор методов. К основным из них относятся **нормализация, преобразование типов и форматов, сортировка, группировка, слияние и др**.

https://wiki.loginom.ru/articles/data-transformation.html

Квантование данных (Биннинг) — **это преобразование непрерывной переменной в категориальную переменную.** Например, если мы хотим применить условия к непрерывным столбцам, скажем, к столбцу «вес машины», мы можем создать новый категориальный столбец с помощью:
вес> 1500 и вес <2500 как «Легкий»
вес> 2500 и вес <3500 как «средний»
вес> 3500 и вес <4500 как "Heavy"
вес> 4500 как «очень тяжелый»

https://www.analyticsvidhya.com/blog/2020/09/pandas-speed-up-preprocessing/

Фильтрация данных (спектральный анализ)

https://basegroup.ru/community/articles/data-filtration

Препроцессинг

Препроцессинг описан в разделе Preprocessing data библиотеки scikit-learn. В scikit-learn это трансформация и нормализация данных. Делать это необходимо, так как многие алгоритмы чувствительны к выбросам, а так же распределению данных в выборке.

StandardScaler центрирует данные, удаляет среднее значение для каждого объекта, а затем масштабирует, деля на среднее отклонение. $x_{scaled} = \frac{x-u}{s}$, где u среднее, а s отклонение.

MinMaxScaler трансформирует признаки в выбранном диапазоне. $x_{scaled} = \frac{x-\min(x)}{\max(x)-\min(x)}$

MaxAbsScaler трансформирует в диапазон [-1,1]. Используется для центрированных вокруг нуля или разреженных данных. $x_{scaled} = \frac{x}{\max(abs(x))}$

RobustScaler. Для данных, в которых много выбросов.

Для нормализации данных можно использовать **Normalizer**. Часто это необходимымо, когда алгоритм предсказывает, базируясь на взвешенных значениях, основанных на расстояниях между точками данных. Особенно актуально для классификации текста и кластеризации.

https://konstantinklepikov.github.io/2019/10/08/scikit-learn-preprocessing.html

Newchronik 10 ноября 2020 в 15:01

https://habr.com/ru/post/527334/

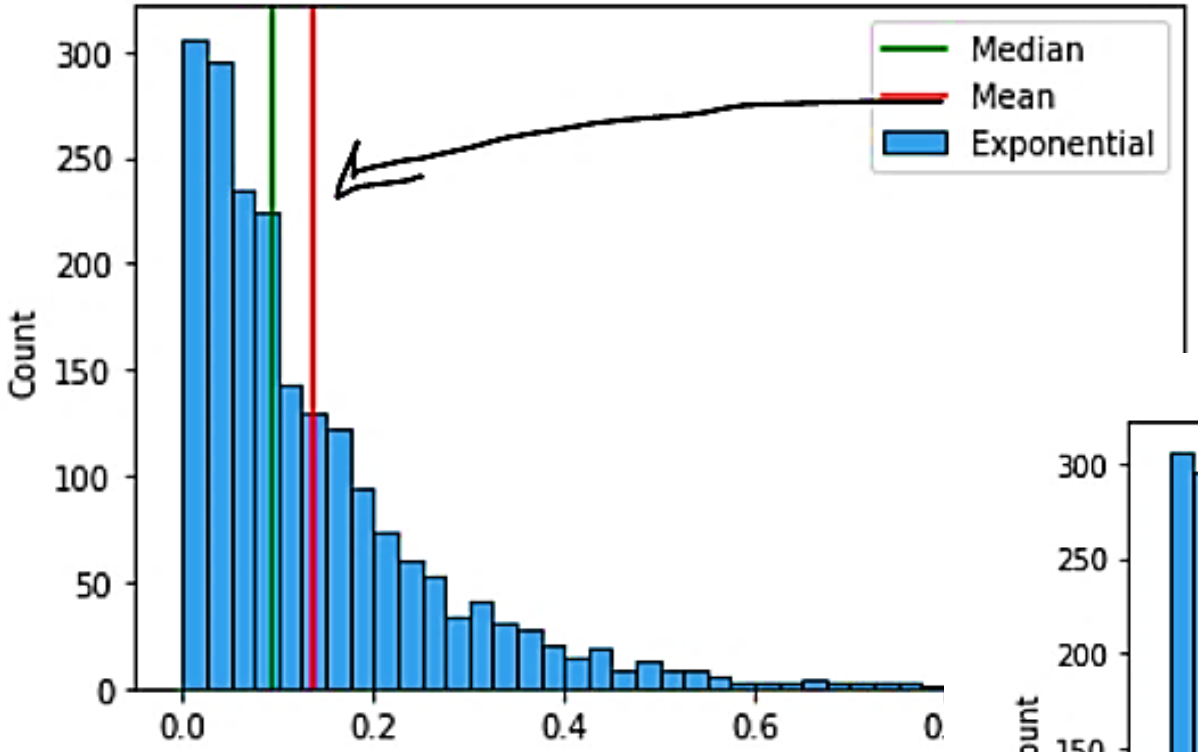
Умная нормализация данных

Python*, Алгоритмы*, Машинное обучение*, Искусственный интеллект, Data Engineering*

Главное условие правильной нормализации — все признаки должны быть равны в возможностях своего влияния.

Что лучше взять за центр? Часто используется среднее арифметическое значение. Здесь проявляется проблема № 1 — различные типы распределений не позволяют применять к ним методы, созданные для нормального распределения.
Если Вы спросите любого специалиста по статистике, какое значение лучше всего показывает “типичного представителя” совокупности, то он скажет, что это — **медиана**, а не среднее арифметическое. Последнее хорошо работает только в случае нормального распределения и совпадает с медианой (алгоритм стандартизации вообще оптимален именно для нормального распределения). А у Вас распределения разных признаков могут (и скорее всего будут) кардинально разные.

Вот, например, различия между медианой и средним арифметическим значением для экспоненциального распределения.



В отличии от среднего значения медиана практически не чувствительна к выбросам и асимметрии распределения. Поэтому её оптимально использовать как “нулевое” значение при центрировании