

ПРИМЕНЕНИЕ СТАТИСТИЧЕСКИХ МЕТОДОВ НА ЯЗЫКЕ PYTHON

- ❖ расчет основных статистических показателей
- ❖ корреляционный анализ
- ❖ регрессия

ПОЛЕЗНЫЕ ССЫЛКИ И ИСТОЧНИКИ

- Плас Дж. Вандер Python для сложных задач: наука о данных и машинное обучение. — СПб.: Питер, 2018. — 576 с.: ил. — (Серия «Бестселлеры O'Reilly»).
- Мартин О. Байесовский анализ на Python / пер. с англ. А. В. Снастина. — М.: ДМК Пресс, 2020. — 340 с.: ил.
- 50 оттенков matplotlib — The Master Plots (с полным кодом на Python) : <https://habr.com/ru/articles/468295/>

Статистика занимается сбором, организацией (упорядочением), анализом и интерпретацией данных, следовательно, знание статистики чрезвычайно важно для анализа данных. При анализе данных используются два основных статистических метода:

- ✓ **разведочный анализ данных** (РАД) (Exploratory Data Analysis – EDA) – используются числовые обобщающие характеристики, такие как среднее значение, мода, стандартное отклонение и вероятные отклонения (этот раздел разведочного анализа данных также называют описательной статистикой). Кроме того, при разведочном анализе данные исследуются визуально с применением широко известных инструментальных средств, таких как гистограммы и диаграммы рассеяния;
- ✓ **статистический вывод** (inferential statistics) – вывод утверждений на основе текущих данных. Это может быть необходимо для понимания некоторых конкретных явлений, или для прогнозирования в будущем точек данных (которые ранее не наблюдались), либо для выбора одного из не скольких альтернативных объяснений результатов наблюдений. **Статистический вывод** – это набор методов и инструментов, которые помогают ответить на типы вопросов, перечисленные выше.

Данные – это важнейший ингредиент в статистике и науке о данных (даталогии). Данные поступают из различных источников, таких как эксперименты, компьютерные имитации, опросы и полевые наблюдения.

Если мы являемся ответственными за генерацию или сбор данных, то всегда в первую очередь не обходимо тщательно **продумать и сформулировать вопросы**, на которые нужно получить ответы, и **определить** используемые для этого **методы**, и только после этого **приступить к обработке данных**.

Всегда **необходимо интерпретировать данные в контексте используемых моделей**, включая ментальные и формальные. Данные без моделей ни о чем не говорят.

Наука о данных является междисциплинарным предметом. Наука о данных охватывает три отдельные, но пересекающиеся сферы:

- ✓ специалиста по математической статистике, умеющего моделировать наборы данных и извлекать из них основное;
- ✓ навыки специалиста в области компьютерных наук, умеющего проектировать и использовать алгоритмы для эффективного хранения, обработки и визуализации этих данных;
- ✓ экспертные знания предметной области, полученные в ходе традиционного изучения предмета, — умение как формулировать правильные вопросы, так и рассматривать ответы на них в соответствующем контексте.

Очень полезной способностью при анализе данных является умение написать код на каком-либо языке программирования, например на Python. Обработка данных является неизбежной необходимостью с учетом того, что мы живем в беспорядочном мире с еще более беспорядочными данными, поэтому умение писать программный код помогает решать эти задачи. Даже если вы настолько удачливы, что находящиеся в вашем распоряжении данные предварительно обработаны и очищены, умение писать код все равно останется полезным навыком, потому что современная статистика реализована в основном с помощью языков программирования, таких как Python или R

Язык программирования Python пригоден для науки о данных в основном благодаря большой и активно развивающейся экосистеме пакетов, созданных сторонними разработчиками:

- ❖ **библиотеки NumPy** — для работы с однородными данными в виде массивов;
- ❖ **библиотеки Pandas** — для работы с неоднородными и поименованными данными;
- ❖ **SciPy** — для общих научных вычислительных задач;
- ❖ **библиотеки Matplotlib** — для визуализаций типографского качества;
- ❖ Seaborn позволяет быстро создавать красивые графики на основе pandas-структур.
- ❖ **оболочки IPython** — для интерактивного выполнения и совместного использования кода;
- ❖ **библиотеки Scikit-Learn** — для машинного обучения и множества других инструментов, которые будут упомянуты в дальнейшем.

- Область статистики часто понимают неправильно, однако она играет важную роль в повседневной жизни. Корректно составленная статистика позволяет извлечь знания из неопределённого и сложного реального мира, однако при неправильном применении она может нанести вред или ввести в заблуждение. Для того, чтобы отличить правду от лжи, важно чётко понимать методы статистики и значение различных статистических измерений.

Что именно представляет собой статистика?

Ключевые идеи при ответе на этот сложный вопрос:

- ❖ статистика — наука о данных;
- ❖ данные — набор наблюдений за интересующей нас генеральной совокупностью;
- ❖ статистика предоставляет конкретный способ сравнения генеральных совокупностей с помощью чисел, а не неоднозначных описаний.

ОПИСАТЕЛЬНАЯ СТАТИСТИКА

✓ Когда у нас есть набор наблюдений, полезно свести признаки наших данных в одно определение. Этим занимается описательная статистика. Как следует из названия, описательная статистика описывает конкретное свойство данных, которые она обобщает. Такую статистику можно разделить на две категории:

- ❖ меры центральной тенденции
- ❖ меры разброса.

МЕРЫ ЦЕНТРАЛЬНОЙ ТЕНДЕНЦИИ

Меры центральной тенденции — показатели, представляющие собой ответ на вопрос: «На что похожа середина данных?».

Среднее значение

- ✓ Данная характеристика описывает среднее значение в наборе данных. Вычислить её довольно просто: сложите все значения и разделите полученную сумму на количество значений.
- ✓ В случае со средним значением «серединой» датасета будет среднее арифметическое его значений. Среднее значение отражает типичный показатель в наборе данных. Если мы случайно выберем один из показателей, то, скорее всего, получим значение, близкое к среднему.

Загрузили данные

```
import csv
with open("wine-data.csv", "r", encoding="latin-1") as f:
    wines = list(csv.reader(f))
```

index	country	description	designation	points	price	province	region_1	region_2	variety	winery
0	US	"This tremendous 100%..."	Martha's Vineyard	96	235	California	Napa Valley	Napa	Cabernet Sauvignon	Heitz
1	Spain	"Ripe aromas of fig..."	Carodorum Selecci Especial Reserva	96	110	Northern Spain	Toro		Tinta de Toro	Bodega Carmen Rodriguez
2	US	"Mac Watson honors..."	Special Selected Late Harvest	96	90	California	Knights Valley	Sonoma	Sauvignon Blanc	Macauley
3	US	"This spent 20 months..."	Reserve	96	65	Oregon	Willamette Valley	Willamette Valley	Pinot Noir	Ponzi
4	France	"This is the top wine..."	La Brelade	95	66	Provence	Bandol		Provence red blend	Domaine de la Begude

МЕРЫ ЦЕНТРАЛЬНОЙ ТЕНДЕНЦИИ

Вычислить среднее значение на Python просто. Чему равна средняя оценка вина в нашем датасете:

```
# Извлекаем оценки из датасета
scores = [float(w[4]) for w in wines]

# Складываем все оценки
sum_score = sum(scores)

# Ищем количество оценок
num_score = len(scores)

# Считаем среднее значение
avg_score = sum_score/num_score

print(avg_score) # выводит 87.8884184721394
```

Это среднее значение говорит нам, что «типичная» оценка в датасете равна примерно 87,8. Соответственно, большинство вин имеют высокий рейтинг, если предположить, что оценивают по шкале от 0 до 100. Тем не менее нужно учесть, что Wine Enthusiast не публикует отзывы с рейтингом ниже 80. Есть разные типы среднего значения, но это — наиболее распространённая форма. Оно называется средним арифметическим, так как интересующие нас значения складываются.

МЕРЫ ЦЕНТРАЛЬНОЙ ТЕНДЕНЦИИ

Медиана

- ✓ Следующая мера центральной тенденции, о которой пойдёт речь, — медиана. Медиана, как и среднее значение, нужна для определения типичного значения в наборе данных, но при этом не требует вычислений.
- ✓ Чтобы найти медиану, данные нужно расположить в порядке возрастания. Медианой будет значение, которое совпадает с серединой набора данных. Если количество значений чётное, то берётся среднее двух значений, которые «окружают» середину.

Возьмите ваши наблюдения: 80, 87, 95, 83, 92

Расположите их в
возрастающем порядке: 80, 83, 87, 92, 95

Среднее значение и есть
медиана

80, 83, **87**, 92, 95

Если значений чётное кол-во, то
медианой будет среднее
арифметическое двух средних
значений

80, 83, **89.5**, 95, 98

МЕРЫ ЦЕНТРАЛЬНОЙ ТЕНДЕНЦИИ

Стандартной библиотекой Python не предусмотрен поиск медианы. Найти медиану цен на вина, следуя описанному алгоритму :

```
# Извлекаем цены
prices = [float(w[5]) for w in wines if w[5] != ""]

# Находим их количество
num_wines = len(prices)

# Сортируем в порядке возрастания
sorted_prices = sorted(prices)

# Ищем индекс среднего элемента
middle = (num_wines / 2) + 0.5

# Находим медиану
print(sorted_prices[middle]) # 24
```

С версии Python 3.4 есть встроенный способ поиска медианного значения.

Медианная цена бутылки вина составляет 24\$. Это предполагает, что как минимум у половины вин в датасете цена равна или ниже 24\$. Неплохо! А что насчёт среднего значения? Учитывая, что и медиана, и среднее значение отражают типичное значение, можно предположить, что они должны быть примерно

```
print(sum(prices)/len(prices)) # 33.13
```

Средняя цена в 33,13\$ на порядок выше медианной. Как это произошло? Разница между медианой и средним значением существует из-за робастности (выбросоустойчивости).

МЕРЫ ЦЕНТРАЛЬНОЙ ТЕНДЕНЦИИ

Проблема выбросов

- ✓ Как вы помните, среднее значение можно найти, сложив все значения и разделив сумму на их количество, в то время как медиана ищется простой перестановкой значений. Если в данных есть выбросы — значения, которые гораздо выше или ниже остальных, — это может негативно повлиять на среднее значение. Таким образом, среднее значение не робастно, а медиана — напротив, выбросоустойчива.

Давайте взглянем на максимальную и минимальную цену в наших данных:

```
min_price = min(prices)
max_price = max(prices)
print(min_price, max_price) # 4.0, 2300.0
```

Теперь мы знаем, что в данных есть выбросы. Выбросы могут отражать интересные события или ошибки в нашем наборе данных, поэтому важно уметь определять их наличие. Сравнение медианы и моды — один из способов определить наличие выбросов, хотя визуализация обычно позволяет сделать это быстрее.

МЕРЫ ЦЕНТРАЛЬНОЙ ТЕНДЕНЦИИ

Мода

- ✓ Это последняя мера центральной тенденции, о которой пойдёт речь. Мода определяется как значение, которое наиболее часто встречается в наборе данных. Мода не так очевидно соответствует понятию «середины» как среднее значение или медиана, но это соответствие абсолютно обосновано: если значение появляется в данных неоднократно, оно приблизит среднее значение к моде. Чем чаще появляется значение, тем сильнее оно влияет на среднее. Таким образом, мода показывает наиболее значимый фактор, формирующий среднее значение.

Функции для поиска моды у Python нет. Вычислим, посчитав количество повторений различных цен и выбрав самую частую:

Мода относительно близка к медиане, поэтому можно уверенно сказать, что и мода, и медиана отражают средние значения цен на вино.

```
# Создаём пустой словарь, в котором будем считать количество появлений цен
price_counts = {}
for p in prices:
    if p not in price_counts:
        counts[p] = 1
    else:
        counts[p] += 1

# Проходимся по словарю и ищем максимальное количество повторений
maxp = 0
mode_price = None
for k, v in counts.items():
    if maxp < v:
        maxp = v
        mode_price = k
print(mode_price, maxp) # 20.0, 7860
```

Меры центральной тенденции полезны для описания среднего значения данных. Тем не менее они не показывают, насколько большой разброс присутствует в данных. Здесь на помощь приходят меры разброса данных.

МЕРЫ РАЗБРОСА ДАННЫХ

- Меры разброса отвечают на вопрос: «Как сильно варьируются мои данные?». В мире существует не так много вещей, которые остаются в одном и том же состоянии при каждом наблюдении. Эта изменчивость делает мир нечётким и неопределённым, поэтому полезно иметь показатели, которые могут обобщить эту «нечёткость».

МЕРЫ РАЗБРОСА ДАННЫХ

Размах

- ✓ Наша первая мера разброса — размах. Из всех измерений, которые мы рассмотрим далее, его вычислить проще всего. Для этого нужно просто вычесть из наибольшего значения в наборе данных наименьшее.

Мы нашли максимальную и минимальную цены, когда искали медиану, поэтому сейчас можем использовать их.

```
price_range = max_price - min_price  
print(price_range) # 2296.0
```

Итак, размах равен 2296, но что это значит? Когда мы рассматриваем результаты различных измерений, очень важно делать это в контексте наших данных. Наша медианная цена была 24\$, а размах равен 2296\$. Размах на два порядка больше медианы, что указывает на сильный разброс данных. Возможно, будь у нас ещё один винный датасет, мы могли бы сравнить размахи, чтобы понять, как они отличаются. В ином случае сам по себе размах не слишком полезен.

МЕРЫ РАЗБРОСА ДАННЫХ

Мы скорее хотели бы узнать, как сильно данные отличаются от типичного значения. Здесь нам помогут стандартное отклонение и дисперсия случайной величины.

Стандартное отклонение

- ✓ Стандартное отклонение тоже является мерой разброса данных. Оно помогает узнать, как сильно данные отличаются от типичного значения. Иными словами, оно говорит о том, как сильно данные отличаются от среднего арифметического. Отношение к среднему арифметическому хорошо видно при расчё

Греческая буква «сигма» используется для обозначения стандартного отклонения

1. Вычтите каждое наблюдение из среднего значения

2. Возведите каждую разность в квадрат

3. Сложите все разности

4. Разделите сумму на количество наблюдений минус 1

5. Из результата извлеките квадратный корень

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

МЕРЫ РАЗБРОСА ДАННЫХ

Надо посчитать стандартное отклонение, чтобы более полно описать цены вин и их оценки, поэтому пишем свою функцию. Поиск кумулятивной суммы вручную выглядел бы довольно громоздко, но циклы **for** в Python всё упрощают. Мы пишем свою функцию, чтобы показать, что на Python легко заниматься такой статистикой. Тем не менее в библиотеке **numpy** тоже реализовано вычисление стандартного отклонения через функцию **std**:

```
def stdev(nums):  
    diffs = 0  
    avg = sum(nums)/len(nums)  
    for n in nums:  
        diffs += (n - avg)**(2)  
    return (diffs/(len(nums)-1))**(0.5)  
  
print(stdev(scores)) # 3.2223917589832167  
  
print(stdev(prices)) # 36.32240385925089
```

Такие результаты вполне ожидаемы. Оценки варьируются от 80 до 100, поэтому можно предположить, что стандартное отклонение будет небольшим. С другой стороны, отклонение в ценах гораздо выше из-за выбросов. Чем больше стандартное отклонение, тем больше рассеяны данные вокруг среднего значения, и наоборот.

МЕРЫ РАЗБРОСА ДАННЫХ

Дисперсия

- ✓ Часто стандартное отклонение и дисперсию связывают вместе и делают это не без причины. Вот уравнение дисперсии, ничего не напоминает?

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}$$

Дисперсия и стандартное отклонение — почти одно и то же! Дисперсия — просто квадрат стандартного отклонения. Более того, обе величины отражают одну и ту же вещь — меру разброса, хотя стоит отметить, что единицы измерения разные. В каких бы единицах ни измерялись ваши данные, единицы измерения отклонения будут такими же, а у дисперсии они будут возведены в квадрат.

НЕКОТОРЫЕ СТАТИСТИЧЕСКИЕ ОПЕРАЦИИ В PYTHON3 КОМАНДОЙ В ОДНУ СТРОКУ

- Для демонстрации таких операций мы будем использовать *набор данных о выживших на «Титанике»*.

```
# Import Pandas Library
import pandas as pd

# Load Titanic Dataset as Dataframe
dataset = pd.read_csv('train.csv')

# Show dataset
# head() by default show
# 5 rows of the dataframe
dataset.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

НЕКОТОРЫЕ СТАТИСТИЧЕСКИЕ ОПЕРАЦИИ В PYTHON3 КОМАНДОЙ В ОДНУ СТРОКУ

- Вычисляет среднее значение с помощью

```
# Calculate the Mean  
# of 'Age' column  
mean = dataset['Age'].mean()
```

```
# Print mean  
print(mean)
```

```
29.69911764705882
```

ies.mean()

Вычисляет медианное значение с помощью метода *DataFrame/Series.median()*

```
# Calculate Median of 'Fare' column  
median = dataset['Fare'].median()
```

```
# Print median  
print(median)
```

```
14.4542
```

НЕКОТОРЫЕ СТАТИСТИЧЕСКИЕ ОПЕРАЦИИ В PYTHON3 КОМАНДОЙ В ОДНУ СТРОКУ

- Вычисляет модальное или наиболее часто встречающееся значение с помощью метода *DataFrame.mode()*

```
# Calculate Mode of 'Sex' column
mode = dataset['Sex'].mode()

# Print mode
print(mode)
```

```
0    male
dtype: object
```

Вычисляет количество или частоту ненулевых значений с помощью метода *DataFrame/Series.count()*

```
# Calculate Count of 'Ticket' column
count = dataset['Ticket'].count()

# Print count
print(count)
```

```
891
```

НЕКОТОРЫЕ СТАТИСТИЧЕСКИЕ ОПЕРАЦИИ В PYTHON3 КОМАНДОЙ В ОДНУ СТРОКУ

- Вычисляет стандартное отклонение значений с помощью метода **DataFrame/Series.std()**

```
# Calculate Standard Deviation  
# of 'Fare' column  
std = dataset['Fare'].std()  
  
# Print standard deviation  
print(std)
```

```
49.693428597180905
```

Вычисляет максимальное значение, используя метод *DataFrame/Series.max()*

```
# Calculate Maximum value in 'Age' column  
maxValue = dataset['Age'].max()  
  
# Print maxValue  
print(maxValue)
```

```
80.0
```

НЕКОТОРЫЕ СТАТИСТИЧЕСКИЕ ОПЕРАЦИИ В PYTHON3 КОМАНДОЙ В ОДНУ СТРОКУ

- Вычисляет минимальное значение с помощью метода **DataFrame/Series.min()**

```
# Calculate Minimum value in 'Fare' column
minValue = dataset['Fare'].min()

# Print minValue
print(minValue)
```

0.0000

Подводит итоги общей описательной статистики с помощью метода **DataFrame/Series.describe()**

```
# Statistical summary
dataset.describe()
```

	PassengerId	Survived	Pclass	Age	Sib Sp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

НЕКОТОРЫЕ СТАТИСТИЧЕСКИЕ ОПЕРАЦИИ В PYTHON3 КОМАНДОЙ В ОДНУ СТРОКУ

- Функция Pandas **Series.cov()** используется для вычисления ковариации двух рядов данных. В следующем примере ковариация вычисляется с помощью метода Pandas и вручную, а затем результаты сравниваются.

В этом примере создаются два списка и преобразуются в ряды с помощью метода Pandas `.Series()`. Находится среднее значение обоих рядов, и создаётся функция для ручного вычисления ковариации. Также применяется метод Pandas `.cov()`, и результаты обоих способов сохраняются в переменных и выводятся на печать для сравнения результатов.

```
Results from Pandas method:  2.8499999999999996
Results from manual function method:  2.8499999999999996
```

Как видно из результатов, они одинаковы в обоих случаях. Следовательно, этот метод полезен при поиске ковариации для больших рядов данных.

```
import pandas as pd

# list 1
a = [2, 3, 2.7, 3.2, 4.1]

# list 2
b = [10, 14, 12, 15, 20]

# storing average of a
av_a = sum(a)/len(a)

# storing average of b
av_b = sum(b)/len(b)

# making series from list a
a = pd.Series(a)

# making series from list b
b = pd.Series(b)

# covariance through pandas method
covar = a.cov(b)

# finding covariance manually
def covarfn(a, b, av_a, av_b):
    cov = 0

    for i in range(0, len(a)):
        cov += (a[i] - av_a) * (b[i] - av_b)
    return (cov / (len(a)-1))

# calling function
cov = covarfn(a, b, av_a, av_b)

# printing results
print("Results from Pandas method: ", covar)
```

РАСЧЕТ ОСНОВНЫХ СТАТИСТИЧЕСКИХ ПОКАЗАТЕЛЕЙ

```
import numpy as np
import pandas as pd
from scipy import stats # Для расширенной статистики

data = [23, 45, 67, 12, 89, 45, 34, 56, 72, 31]
df = pd.DataFrame({'Age': [25, 30, 35, 40, 45],
                  'Salary': [50000, 60000, 70000, 80000, 90000]})

# Среднее значение
mean = np.mean(data) # Или df['Age'].mean()

# Медиана
median = np.median(data) # Или df['Age'].median()

# Стандартное отклонение
std_dev = np.std(data, ddof=1) # Или df['Age'].std()

# Квартили
q1, q3 = np.percentile(data, [25, 75]) # Или df['Age'].quantile([0.25, 0.75])

print(f"Среднее: {mean:.2f}, Медиана: {median}, STD: {std_dev:.2f}")
print(f"Q1: {q1}, Q3: {q3}")
```

```
→ Среднее: 47.40, Медиана: 45.0, STD: 23.81
Q1: 31.75, Q3: 64.25
```

КОРРЕЛЯЦИОННЫЙ АНАЛИЗ

```
from scipy.stats import pearsonr
```

```
r, p_value = pearsonr(df['Age'], df['Salary'])
```

```
print(f"Коэффициент Пирсона: {r:.2f}, p-value: {p_value:.4f}")
```

```
# Интерпретация:
```

```
#  $|r| > 0.7$  – сильная корреляция
```

```
#  $0.3 < |r| < 0.7$  – умеренная
```

```
#  $|r| < 0.3$  – слабая
```

```
Коэффициент Пирсона: 1.00, p-value: 0.0000
```

ПОСТРОЕНИЕ МОДЕЛИ РЕГРЕССИИ

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

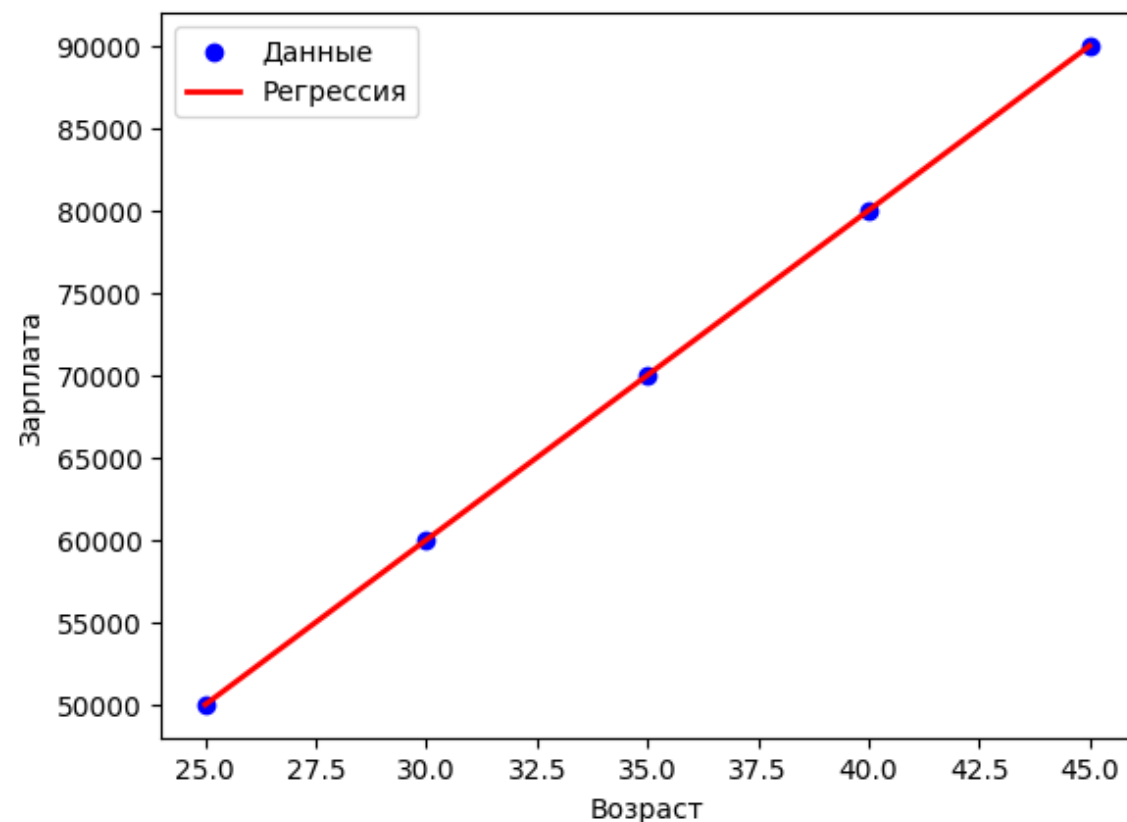
# Подготовка данных
X = df[['Age']] # Признак
y = df['Salary'] # Целевая переменная

# Разделение на тренировочную и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Создание и обучение модели
model = LinearRegression()
model.fit(X_train, y_train)

# Прогнозирование
y_pred = model.predict(X_test)

plt.scatter(X, y, color='blue', label='Данные')
plt.plot(X, model.predict(X), color='red', linewidth=2, label='Регрессия')
plt.xlabel('Возраст')
plt.ylabel('Зарплата')
plt.legend()
plt.show()
```



ПРИМЕРЫ ВИЗУАЛИЗАЦИИ РЕЗУЛЬТАТОВ АНАЛИЗА ДАННЫХ

- Для примеров использования методов библиотек Python взят готовый датасет с информацией о пользователях онлайн-кинотеатра, о фильмах и их рейтинге.

Последовательность уже известных по лекциям 2-4 операций обработки и очистки данных смотрите здесь

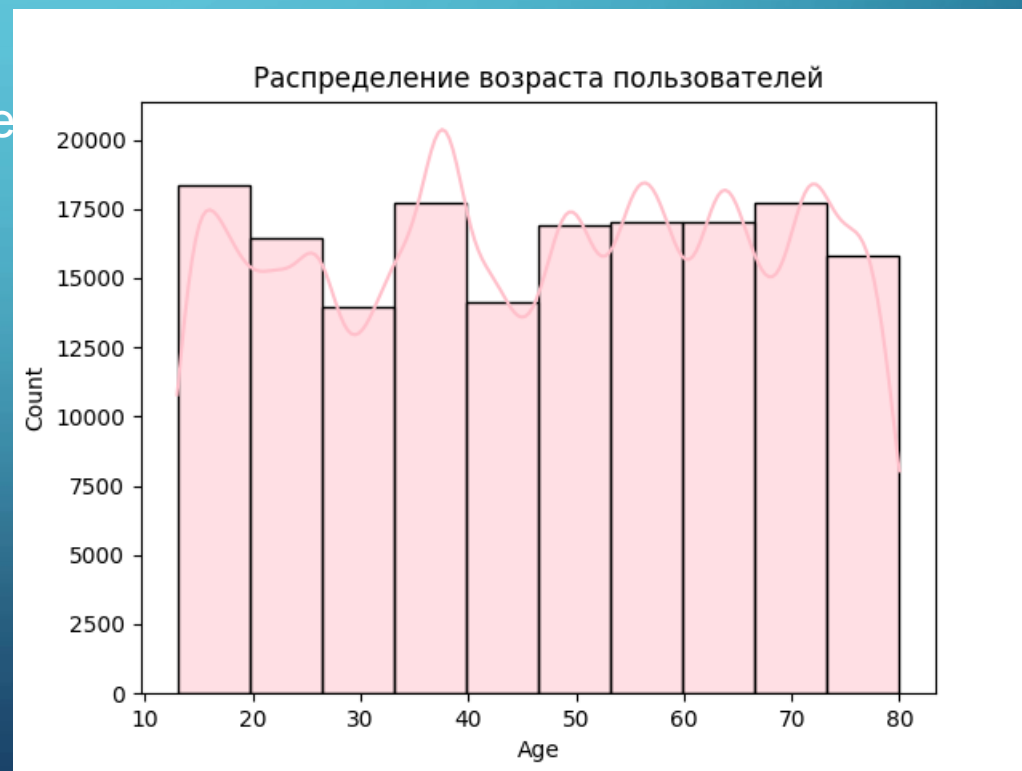
[Инструменты Python для анализа данных на примере данных стриминг-сервиса / Хабр](#)

ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ АНАЛИЗА ДАННЫХ

- Гистограмма распределения возраста пользователей

```
sns.histplot(rating_user['Age'], bins = 10, kde = True, color='pink')  
plt.title('Распределение возраста пользователей')  
plt.show()
```

С помощью метода `sns.histplot()` была построена гистограмма распределения возраста пользователей



ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ АНАЛИЗА ДАННЫХ

- Кривая плотности рейтинга

```
sns.kdeplot(rating_user['Rating'], color='blue', fill=True)  
plt.title('Кривая плотности рейтинга')  
plt.show()
```

Можно увидеть, что пользователи больше склонны ставить фильмам оценку '4'



ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ АНАЛИЗА ДАННЫХ

- Распределение типов подписок по разным возрастным группам

Сгруппируем данные по возрастной группе и типу подписки:

```
subscription_counts = rating_user.groupby(['Age_Group',  
'Subscription_Type'], observed=True).size().reset_index(name='Count')  
print(subscription_counts)
```

output:

	Age_Group	Subscription_Type	Count
0	18-30	Basic	8753
1	18-30	Premium	9662
2	18-30	Standard	8254
3	30-45	Basic	14185
4	30-45	Premium	11211
5	30-45	Standard	11167
6	45-60	Basic	11977
7	45-60	Premium	12598
8	45-60	Standard	12310
9	60-90	Basic	17929
10	60-90	Premium	14991
11	60-90	Standard	16373

Метод `.size()` подсчитывает количество записей в каждой группе.

Метод `.reset_index()` преобразует результат группировки обратно в DataFrame.

Параметр `name='Count'` присваивает имя новому столбцу, содержащему количество подписчиков для каждой комбинации 'Age_Group' и 'Subscription_Type'.

ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ АНАЛИЗА ДАННЫХ

- Распределение типов подписок по разным возрастным группам

Построение гистограммы barplot:

```
sns.barplot(x='Age_Group', y='Count', hue='Subscription_Type',  
data=subscription_counts, palette='viridis')  
plt.title('Распределение типов подписок по возрастным группам')  
plt.xlabel('Возрастная группа')  
plt.ylabel('Количество подписок')  
plt.legend(title='Тип подписки')  
plt.show()
```

По оси абсцисс выводится возрастная группа, по оси ординат количество подписчиков. Из данной гистограммы видно, что пользователи от 18 до 30 лет чаще выбирают подписку Premium, а пользователи более старшего возраста предпочитают подписку Basic.

