ПРАКТИЧЕСКАЯ РАБОТА №4

Корреляция, линейная регрессия и дисперсионный анализ

Корреляция

Строгие (функциональные) зависимости – каждому значению одной переменной соответствует единственное значение другой.

Между экономическими переменными часто нет строгой зависимости: цена – спрос, доход – потребление, стаж работы – производительность труда.

Частая *задача* – определение зависимости между случайными величинами, которые являются признаками одних и тех же объектов:

- затраты компании на рекламу – доход от продаж;
- уровень инфляции – безработица и т.д.

В таких случаях говорят не о функциональных, а о *корреляционных*, либо *статистических* зависимостях.

Статистическая зависимость: изменение одной из величин влечет изменение распределения другой.

Корреляционная зависимость: при изменении одной из величин изменяется среднее значение другой.

Корреляции помогают увидеть характер связи между двумя непрерывными переменными. Исследуют такую связь построением диаграммы рассеяния, в которой независимую переменную откладывают по оси x, зависимую – по оси y. Когда нет данных по поводу взаимоотношений переменных, то ось не имеет значения. Каждому значению выборки соответствует точка на графике с координатами (x, y). Диаграммы рассеяния помогают почувствовать свойства связи между переменными: форму (линейная, квадратичная и др.), направление (положительное, отрицательное), силу (сильная, слабая). С помощью диаграмм рассеяния можно получить общее впечатление о разбросе данных, увидеть выбросы, если они имеются.

Два подхода к оценке силы корреляции:

1. Первый опирается только на абсолютную величину коэффициента корреляции

Таблица 1. Оценка силы корреляции по абсолютной величине коэффициента корреляции

Абс. величина коэффициента корреляции |r|	**Сила корреляции**
$\|r\| \geq 0{,}70$	сильная
$0{,}50 \leq \|r\| \leq 0{,}69$	средняя
$0{,}30 \leq \|r\| \leq 0{,}49$	умеренная

0,20≤ |r| ≤0,29	слабая
|r| ≤ 0,19	очень слабая

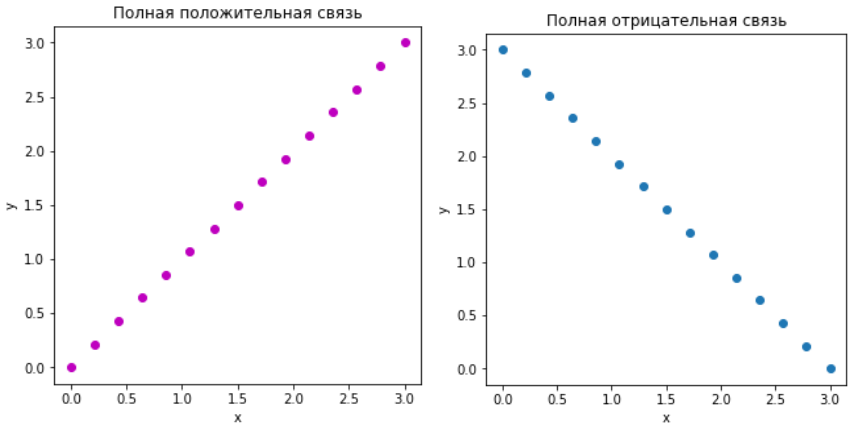


Рисунок 1 Полная и отрицательная связь

2. Второй (см. таб. 2) основан на уровне статистической значимости коэффициента корреляции при данном объеме выборки. Чем больше объем выборки, тем меньший коэффициент корреляции, взятый по абсолютной величине, признается показателем достоверности корреляционной связи.

Таблица 2. Оценка силы корреляции по уровню статистической значимости

Уровень статистической значимости коэффициента корреляции	**Сила корреляции**
p ≤ 0,01	высокая значимая
0,01< p ≤ 0,05	значимая
0,05 <p ≤ 0,10	тенденция достоверной связи
коэффициент корреляции не достигает уровня статистической значимости	незначимая

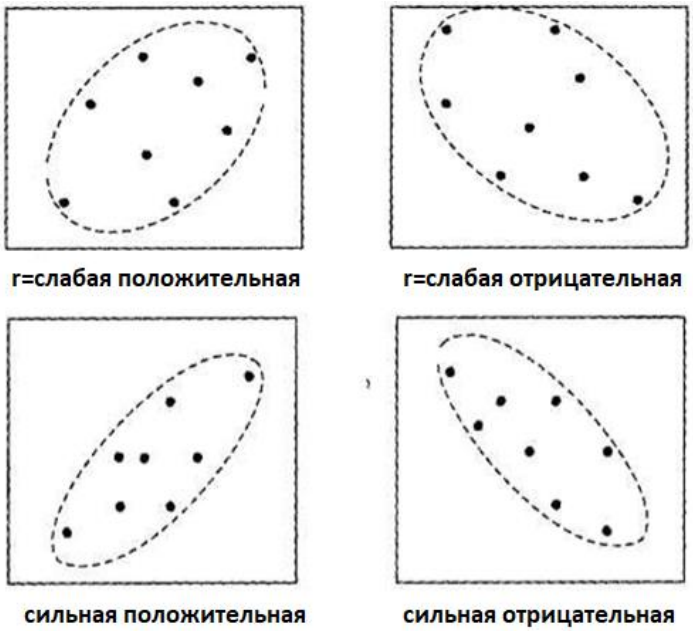


Рисунок 2 Схематическое изображение силы и направления корреляции

Наличие значимой корреляции между переменными дает основание не отвергать гипотезу о причинно-следственной связи между переменными, но не служит подтверждением такой гипотезы.

Коэффициент корреляции Пирсона

Коэффициент корреляции Пирсона (линейный коэффициент корреляции) – мера связи для двух непрерывных или характеризующих отношения переменных.

Обозначение:

- для выборки – r,
- для генеральной совокупности – ρ.

Принимает значения от –1 до +1.

0 (нуль) – отсутствие связи между переменными.

Частая *нулевая гипотеза* для корреляционного анализа: переменные не связаны (r = 0).

Альтернативная гипотеза: $r \neq 0$.

Как рассчитать корреляцию в Python

Самая простая функция, которая рассчитывает коэффициент корреляции только для двух переменных: **corrcoef(**var1,var2**)**.

*Пример***.** Определить два вектора, представляющих число изделий шести видов, изготовленных в первом цехе (ceh1) и во втором (ceh 2):

```
import numpy as np
var1 = np.array([120, 140, 132, 160, 110,131])
var2 = np.array([200, 180, 191, 160, 210,188])


np.corrcoef(var1, var2)
```

```
array([[ 1.        , -0.99685226],
       [-0.99685226,  1.        ]])
```

Рисунок 3 Корреляция с помощью numpy

По умолчанию эта функция создает матрицу коэффициентов корреляции. Если бы мы только хотели вернуть коэффициент корреляции между двумя переменными, мы могли бы используйте следующий синтаксис:

```
np.corrcoef(var1, var2)[0,1]
```

```
-0.9968522579863439
```

Рисунок 4 Особенности синтаксиса

Коэффициент корреляции практически равен –1, что означает сильную отрицательную корреляцию.

Построим диаграмму рассеяния:

```
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt

x = np.array([120, 140, 132, 160, 110,131])
y = np.array([200, 180, 191, 160, 210,188])

plt.grid(True)
plt.title('Диаграмма рассеяния', fontsize = 20)
plt.xlabel('Число изделий, изготовленных цехом №1')
plt.ylabel('Число изделий, изготовленных цехом №2')
plt.scatter(x, y, marker = 'o', color = 'crimson')
```

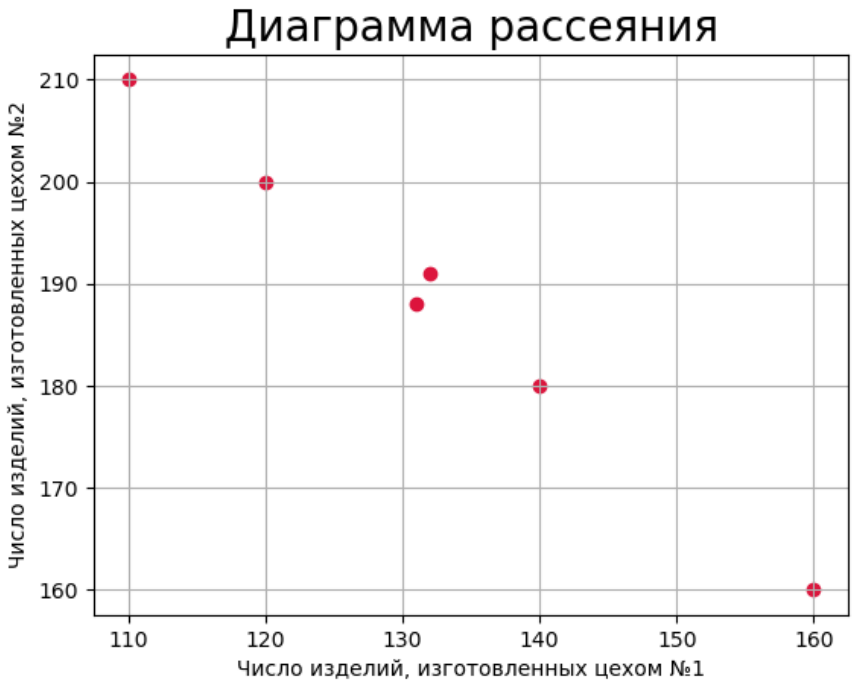


Рисунок 5 Диаграмма рассеяния по двух цехам

Расчет корреляции между двумя конкретными переменными в DataFrame: **data['A'].corr(data['B'])**

Пример: Расчет коэффициент корреляции площади квартиры к цене.

```
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd


os.chdir('C:/Users/User/Desktop/Work/ЛН и КР')
data = pd.read_csv('housePrice.csv')

data['Area'] = pd.to_numeric(data['Area'],errors = 'coerce')   #преобразования аргумента в числовой тип

cr = data['Area'].corr(data['Price(USD)'])    #коэффициент корреляции
print('Коэффициент корреляции = ', cr)
```

```
Коэффициент корреляции =  0.7226470263552708
```

Рисунок 6 Расчет коэффициент корреляции площади квартиры к цене

Коэффициент примерно равен 0.73, что означает высокую корреляцию.

Построим диаграмму рассеивания:

```
fig = plt.figure('Диаграмма рассеяния', figsize = (8,7))    #название и размер внешних границ границ рисунка
plt.grid(True)    #включаем сетку у графика
plt.title('Стоимость и площадь квартир', fontsize = 20)    #название графика и его размер
plt.xlabel('Площадь', size = 12)    #подпись оси x и её размер
plt.ylabel('Цена, $', size = 12)    #подпись оси y  и её размер
plt.yticks(size = 9)    #изменение размера для оси Y
plt.xticks(size = 9)    #изменение размера для оси x


x = data.head(300)['Area']    #проводим выборку - первые 300 строк
y = data.head(300)['Price(USD)']


plt.scatter(x, y, color = 'crimson', #создание графика, определение данных для осей
            alpha = 0.3)    #плотность
plt.show() #вывод графика
```

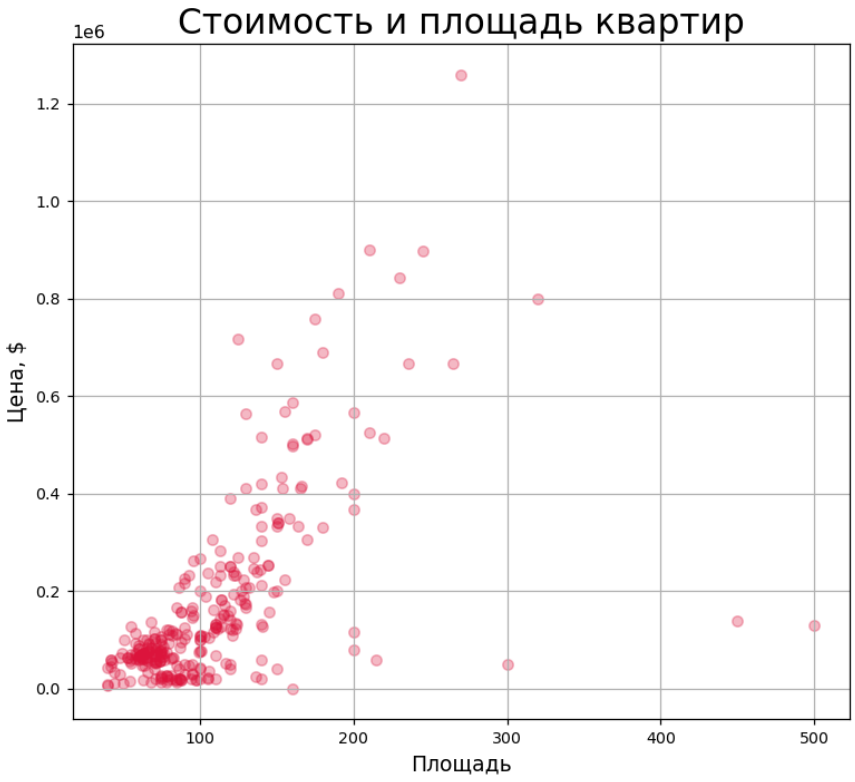


Рисунок 7 Диаграмма рассеяния по стоимости и площади квартир

В некоторых случаях мы хотим понять корреляцию между более чем одной парой переменных. В этих случаях мы можем создать матрицу корреляции, представляющая собой квадратную таблицу, которая показывает коэффициенты корреляции между несколькими попарными комбинациями переменных.

```
os.chdir('C:/Users/User/Desktop/Work/ЛН и КР')
sl = pd.read_csv('Salaries.csv')
df = pd.DataFrame(sl, columns=['yrs.since.phd','yrs.service','salary'])
df.corr()    #создать корреляционную матрицу
df.corr().round(3)    #создайте ту же матрицу корреляции с коэффициентами, округленными до 3 знаков после запятой
```

	yrs.since.phd	yrs.service	salary
yrs.since.phd	1.000	0.910	0.419
yrs.service	0.910	1.000	0.335
salary	0.419	0.335	1.000

Рисунок 8 Матрица корреляции

Все коэффициенты корреляции по диагонали таблицы равны 1, потому что каждая переменная совершенна коррелирует сам с собой. Все остальные коэффициенты корреляции указывают на корреляцию между различными попарными комбинациями переменных.

Вы можете визуализировать матрицу корреляции с помощью параметра стиля в pandas:

```
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	yrs.since.phd	yrs.service	salary
yrs.since.phd	1.000000	0.910000	0.419000
yrs.service	0.910000	1.000000	0.335000
salary	0.419000	0.335000	1.000000

Рисунок 9 Добавления стиля

Линейная регрессия

Одним из наиболее фундаментальных статистических методов, используемых для прогнозирования в науке о данных, является линейная регрессия. Линейная регрессия позволяет нам моделировать и понимать отношения между переменными, а также делать прогнозы на основе этих отношений. Используя принципы линейности, мы можем извлекать значимые закономерности и делать выводы из имеющихся данных.

Итак, еще в старших классах мы узнали об этом уравнении для рисования прямой на графике: y = ax + b. **«a» в уравнении — это наклон**, а **«b» — точка пересечения с осью y или сдвиг**. Мы используем эту же идею в линейной регрессии, где мы пытаемся создать модель, которая может предсказывать вещи. По сути, у нас есть набор точек данных на графике, где значения x являются независимыми переменными, а значения y — зависимыми переменными. Что мы хотим сделать, так это найти линию, которая лучше всего соответствует этим точкам и максимально приближается к фактическим данным, которые у нас есть. Чтобы найти наилучшее соответствие, необходимо выбрать подходящие значения «a» и «b».

Линейная регрессия — модель зависимости переменной от одной или нескольких других переменных (факторов, регрессоров, независимых переменных) с линейной функцией зависимости.

Ниже на графике представлена линейная регрессия переменной y от переменной x (См. рис. 10).

Некоторые условия при построении графиков.

Коэффициент наклона:

1. Если у нас не будет коэффициента наклона, то линяя будет параллельна оси Ox.
2. Если коэффициент наклона больше 0, то линяя идет на увеличение, при этом чем больше коэффициент, тем более наклон крутой.
3. Если коэффициент наклона меньше 0, то линяя идет на уменьшение, при этом чем меньше коэффициент, тем более наклон крутой.

Сдвиг:

1. Если у нас не будет сдвига (коэффициента b), то линяя будет проходить через точку (0, 0).
2. Если коэффициент сдвига не равен 0, а к примеру, равен 2, то линяя будет проходить через точку (0, 2).

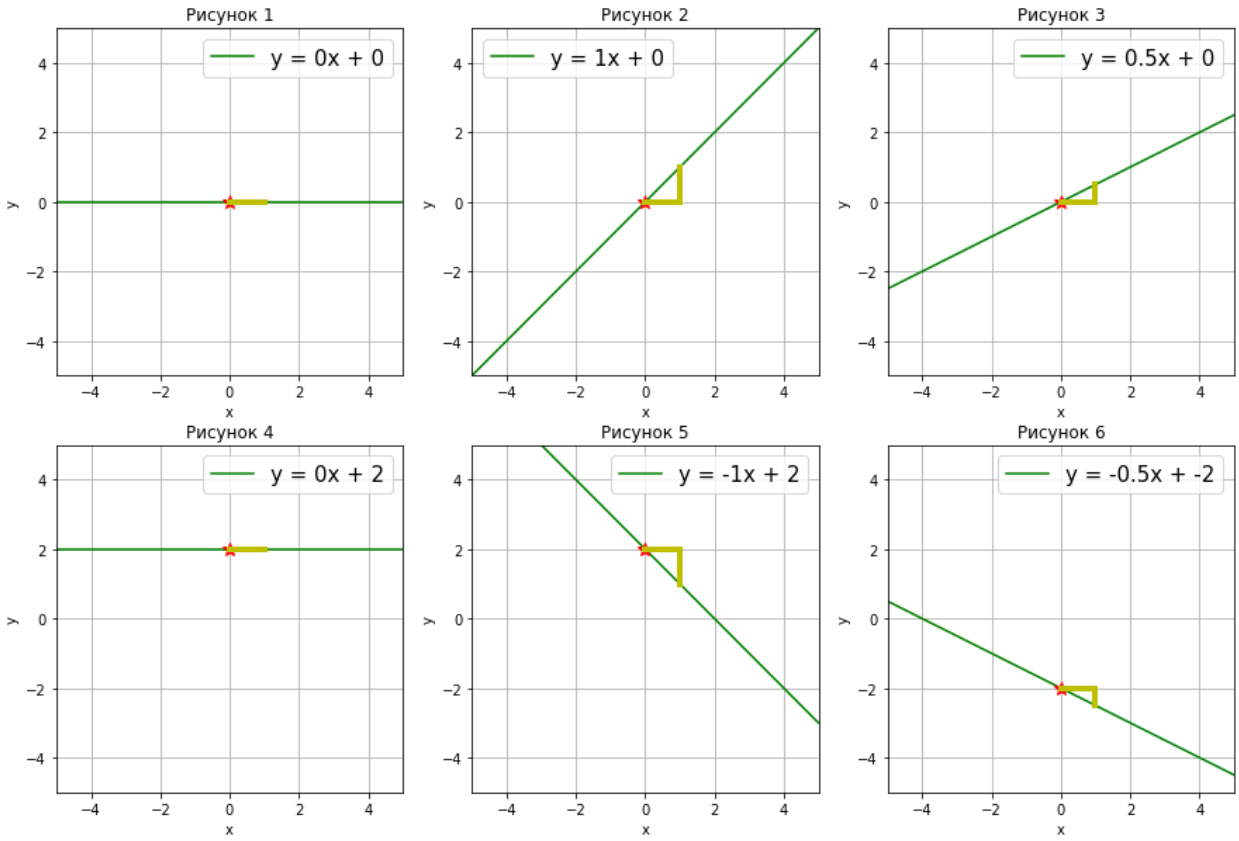


Рисунок 10 Пример функции с разными коэффициентами

Рассмотрим примеры. На рисунке 1 коэффициент a и b нулевые, сдвиг и наклон отсутствуют. Рисунок 2, появляется наклон, коэффициент наклона равен 1, что значит, смещение, один к одному. Рисунок 3, по оси x шагаем на 1, по оси y на 0.5, так как коэффициент наклона 0.5. Рисунок 4, коэффициент наклона равен 0, а сдвиг равен 2, линия не вертится, но сдвигается от начала координат, отклоняется вверх на две единицы. Рисунок 5, совмещается сдвиг и наклон, наклон имеет отрицательную силу, а также сдвиг на 2 единицы от начала координат. Рисунок 6, коэффициент наклона равен -0.5, шагаем в меньшую сторону, сдвиг отрицательный, идем вниз на 2 единицы от начала координат.

Реализация линейной регрессии в sklearn

С самим уравнением прямой разобрались, теперь давайте обучим линейную регрессию. Для начала загрузим данные о недвижимости в Бостоне:

```
boston_df = pd.read_csv('C:/Users/User/Desktop/Work/session 1/Lr reg/boston.csv')
boston_df
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

Рисунок 11 Датасет о недвижимости в Бостоне

Информация об атрибутах:

- CRIM - преступность на душу населения в разбивке по городам;

- ZN - доля жилой земли, зонированной для участков площадью более 25 000 квадратных футов;

- INDUS - Доля промышленных площадей, не связанных с розничной торговлей, в каждом городе;

- CHAS - (1, если тракт граничит с рекой, 0 в противном случае);

- NOX - Концентрация оксидов азота NOX (частей на 10 миллионов);

- RM - Среднее количество комнат в жилом помещении;

- AGE - доля занимаемых владельцами квартир, построенных до 1940 года;

- DIS - Взвешенные расстояния до пяти бостонских центров занятости;

- RAD -Индекс доступности радиальных магистралей;

- TAX - ставка налога на недвижимость по полной стоимости за 10 000 долларов;

- PTRATIO - Соотношение числа учеников и учителей в разбивке по городам;

- B 1000(Bk - 0,63)^2, где Bk - доля чернокожих в разбивке по городам;

- LSTAT - % более низкий статус населения;

- MEDV Средняя стоимость домов, занимаемых владельцами, в 1000 долларов.

Проверим пропущенные значения:

```
boston_df.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

Рисунок 12 Проверка пропущенных значений

Пропущенные значения отсутствуют. Как вы вероятно заметили, все признаки в этом датасете количественные, за исключением переменной CHAS (1, если тракт граничит с рекой, 0 в противном случае) (См. рис.13). Это нужно учитывать при построении корреляционной матрицы.

```
boston_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

```
boston_df.CHAS.value_counts()
```

```
0.0    471
1.0     35
Name: CHAS, dtype: int64
```

Рисунок 13 Общая информация о датасете

На данном этапе от нас требуется понять какие взаимосвязи мы можем выявить между переменными, чтобы потом построить модель. Визуализируем корреляционную матрицу по одной целевой переменной MEDV:

```
corr_matrix = boston_df.corr().MEDV.to_frame().round(2)
corr_matrix.style.background_gradient(cmap='coolwarm')
```

	MEDV
CRIM	-0.390000
ZN	0.360000
INDUS	-0.480000
CHAS	0.180000
NOX	-0.430000
RM	0.700000
AGE	-0.380000
DIS	0.250000
RAD	-0.380000
TAX	-0.470000
PTRATIO	-0.510000
B	0.330000
LSTAT	-0.740000
MEDV	1.000000

Рисунок 14 Корреляция по целевой переменной

Мы видим, что переменные RM и LSTAT имеют достаточно сильную корреляцию с целевой переменной MEDV, 0.70 и −0.74, соответственно. Также умеренная корреляция наблюдается у переменных PTRATIO, TAX и INDUS. Переменную CHAS можно удалить. Поработаем с наиболее коррелирующей переменной LSTAT.

Возьмем модель LinearRegression из sklearn из модуля linear_model:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model
```

```
LinearRegression()
```

Рисунок 15 LinearRegression из sklearn

Создадим две переменных, X – признак «LSTAT» и y – целевое значение:

```
X = boston_df[['LSTAT']]
y = boston_df['MEDV']
X = np.array(X,type(float))
y = np.array(y,type(float))
```

Рисунок 16 Создание двух переменных

Скормим данные нашей модели:

```
model.fit(X,y)
```

```
LinearRegression()
```

Рисунок 17 Передача данных с помощью "fit"

Всё, модель обучилась, это происходит очень быстро. Обучение линейной модели заключается в поиске коэффициентов, конкретно в нашей

задаче — это коэффициент сдвига и наклона. Можем эти коэффициента отобразить, если возьмем у модели атрибут coef_ и intercept_.

```
model.coef_, model.intercept_
```

```
(array([-0.95004935]), 34.55384087938311)
```

Рисунок 18 Сдвиг и наклон

Угол наклона = -0.95

Коэффициент сдвига = 34.55

Теперь отрисуем линию с нашими коэффициентами:

```
import matplotlib.pyplot as plt

model_a = model.coef_[0]
model_b = model.intercept_

model_y_sk = model_a * X + model_b

fig = plt.figure(figsize=(10, 6))
plt.plot(X, model_y_sk, linewidth=2, color = "r", label=f'linear_model = {model_a:.2f}x + {model_b:.2f}')
plt.scatter(X, y, alpha=0.7)
plt.grid()
plt.xlabel('feature')
plt.ylabel('target')
plt.legend(prop={'size': 16})
plt.show()
```

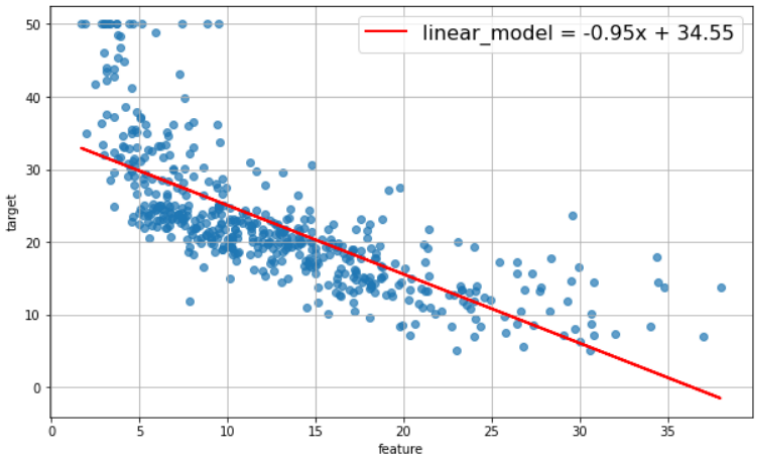


Рисунок 19 Визуализация линии регрессии

Представленная прямая наилучшим образом прошла вдоль точек из обучающей выборки.

Давайте теперь попробуем как-нибудь оценить работу модели регрессии из sklearn. В этом нам помогут **метрики** – это показатели, говорящие нам, насколько точны наши прогнозы и какова величина отклонения от фактических значений. Возьмем MSE - mean squared error.

$$MSE = \frac{1}{n}\sum_{i}^{n}(y - y\_pred)^2$$

где $(y - y\_pred)^2$ — это отклонение истины от предсказания в квадрате;

$\frac{1}{n}\sum_i^n$ – это усредненная сумма всех отклонений в квадрате, поделенная на их количество.

```
mean_squared_error(model_y_sk, y)
```

```
38.48296722989414
```

Рисунок 20 MSE

MSE или среднее квадратичное отклонение составило 38.48

Как же модель из sklearn умудрилась построить такие качественные предсказания? Ведь у неё было множество вариантов построения, можно менять коэффициенты наклона и сдвига как вздумается. На самом деле линейная регрессия обучается с помощью **методов оптимизации**. Они занимаются тем, что берут какую-то функцию и находят в ней минимальные значения или так называемый экстремум функции. Выше мы вывели среднее квадратичное отклонение (См. рис. 20), которое в свою очередь тоже является функцией, а значит мы можем ее оптимизировать, найти самые маленькие значения ошибки для нашей задачи.

Модель sklearn основывается на методе наименьших квадратов, но мы рассмотрим один из лучших методов оптимизации **– градиентный спуск**.

Градиентный спуск

Обсудим, что такое градиент и зачем надо спускаться. Градиентом функции f называется n-мерный вектор из частных производных.

$$\nabla f(x_1, \dots, x_n) = \left(\frac{\partial f}{\partial x_i}\right)_{i=1}^{n}$$

К примеру, если функция зависит от трех переменных: $F(x, y, z)$, то её градиент будет равен:

$$\nabla f(x, y, z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

При этом, градиент задает направление наискорейшего роста функции. Значит, антиградиент будет показывать направление ее скорейшего убывания, что будет полезно нам в нашей задаче минимизации функционала ошибки.

Градиентный спуск — метод нахождения локального минимума с помощью движения вдоль градиента.

Подробно рассмотрим, как рассчитывается метод градиентного спуска на небольшом примере. Реализуем параболу и найдем точку минимума. Зададим две функции:

1. функция параболы $f(x)=x2$
2. производная функции параболы $\nabla f(x)=2x$

Отобразим функцию на графике (См. рис. 21):

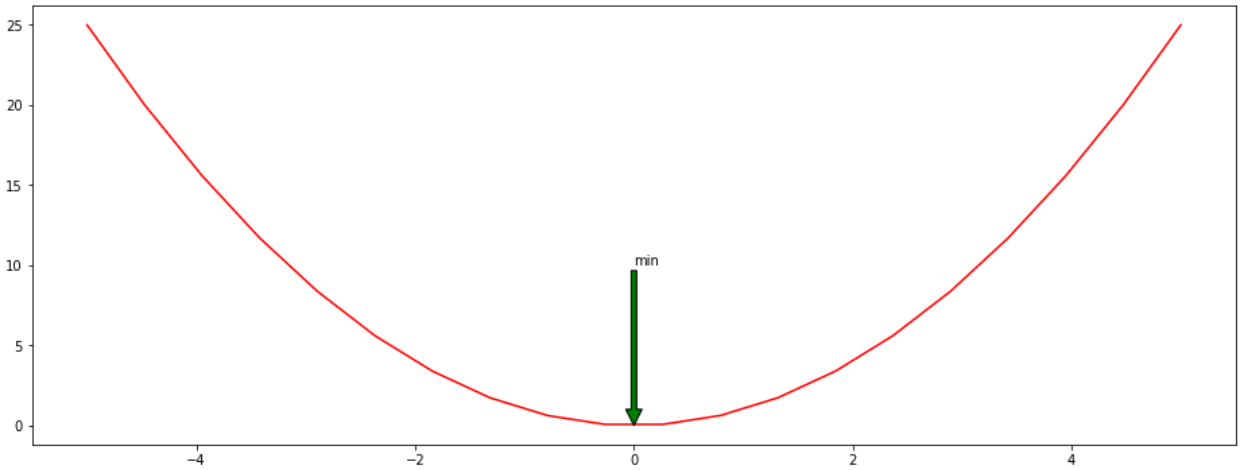


Рисунок 21 Парабола начальная

Чтобы найти минимум этой функции мы можем воспользоваться методом оптимизации - градиентный спуск, для этого нужно задать начальную точку, например 5, откуда будем считать градиенты и скатываться в минимум.

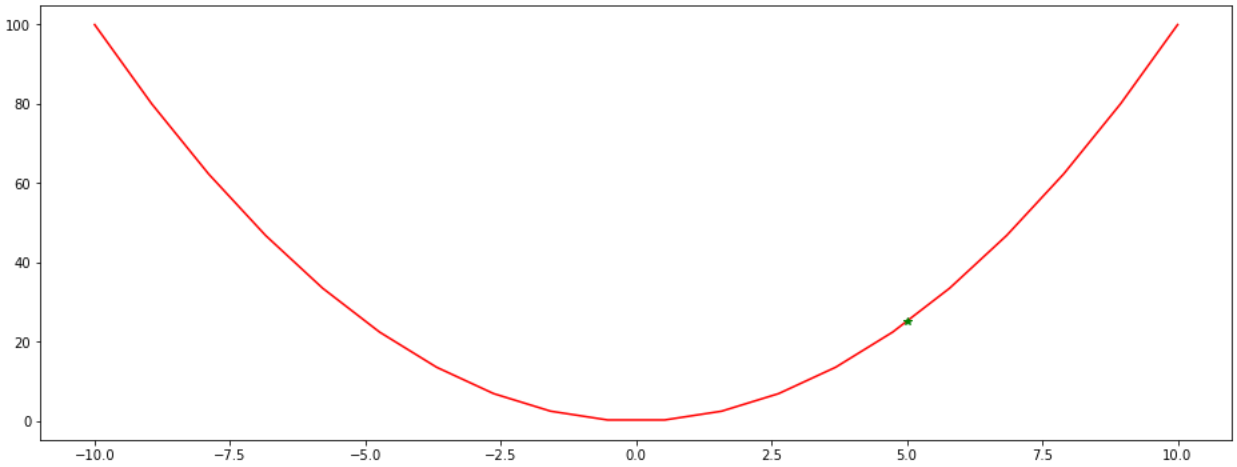


Рисунок 22 Парабола, начальная точка

Точка старта – зеленая звездочка (см. рис. 22). Теперь в этой точке можем посчитать градиент. Он равняется 10, т.к. начальная точка равна 5, а производная будет равняться $\nabla f(x) = 2 \cdot x = 2 \cdot 5 = 10$

Можем отрисовать направление градиента, он показывает наискорейший рост функции и действительно видим, зеленый вектор идет вверх (См. рис. 23).

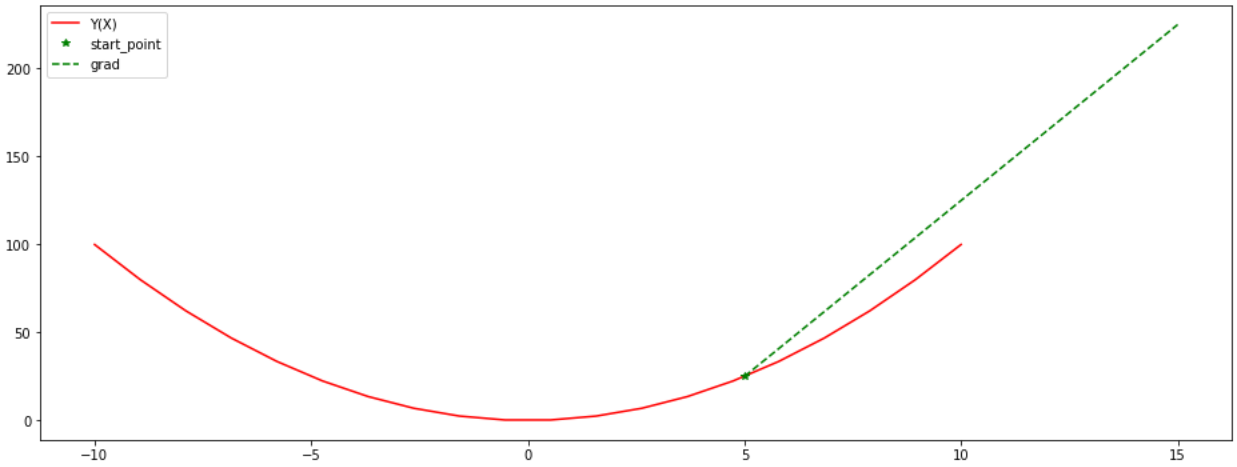


Рисунок 23 Отображение наискорейшего роста функции

Но если будем двигаться по этому вектору, то к минимуму функции не придем, поэтому нужно идти в противоположном направлении, а значит брать **антиградиент,** в нашем случае -10. Но если мы пойдем от текущей точке 5 в сторону антиградиента −10, то окажемся в точке −5, а это так же удалено от минимума, как и наша стартовая точка (См. рис. 24).

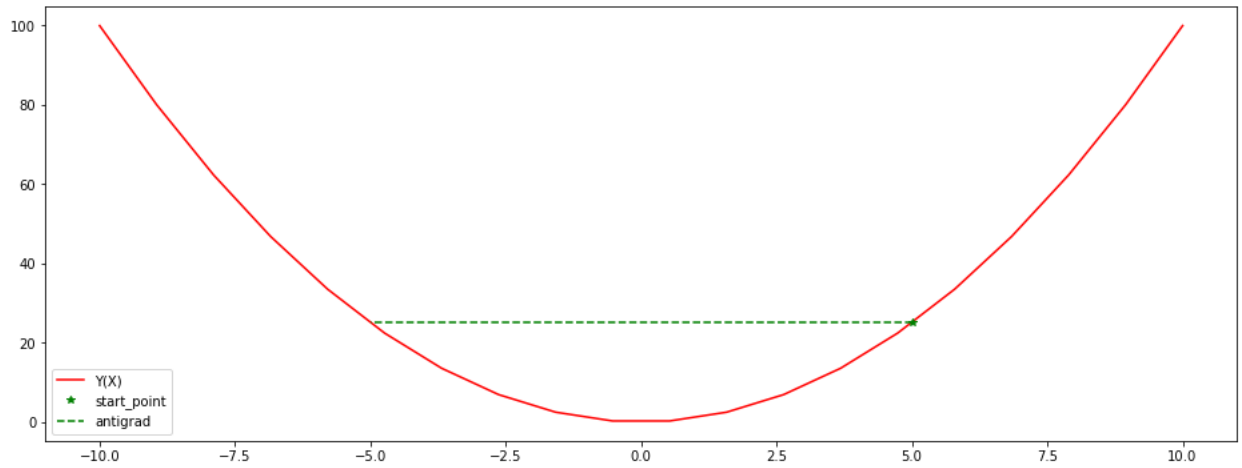


Рисунок 24 Отображение слишком большого шага

Поэтому чтобы не перескакивать минимальное состояние функции мы можем делать шаг в сторону антиградиента не полностью, а только на какую-то долю, для этого нужно ввести значения **шага обучения** (скорость обучения) – это значения, замедляющее шаги градиентного спуска, чтобы не пропустить локальный минимум. В нашем случае, скорость обучения составит 0,1.

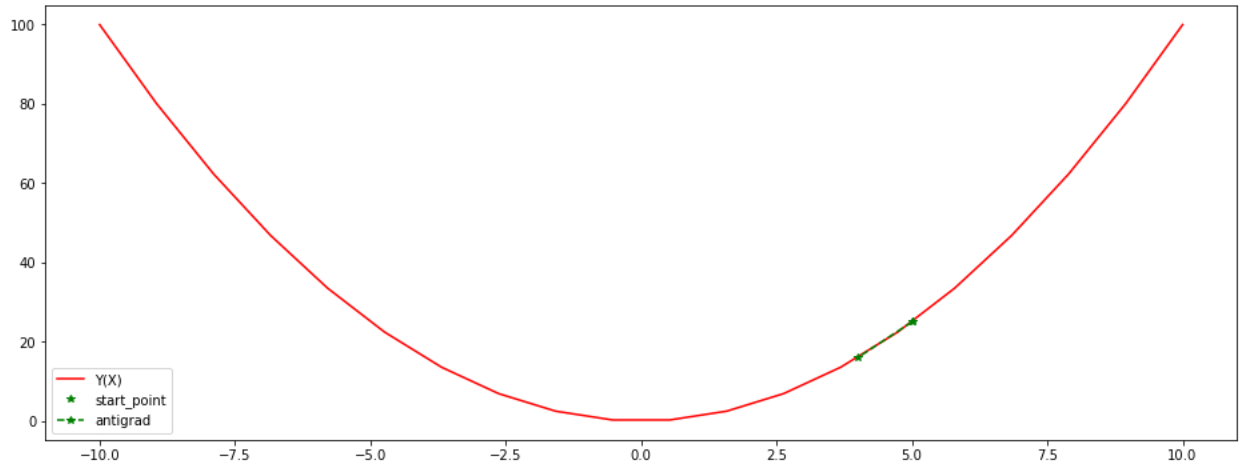


Рисунок 25 Первый шаг градиентного спуска

Вот мы и получили новую точку с координатой x=4 (См. рис. 25).

Теперь в этой точке можем снова рассчитать значение градиента. Производная будет равняться 8. Отрисуем направление градиента, который показывает наискорейший рост функции. Синим пометим уже пройденный шаг (См. рис. 25).

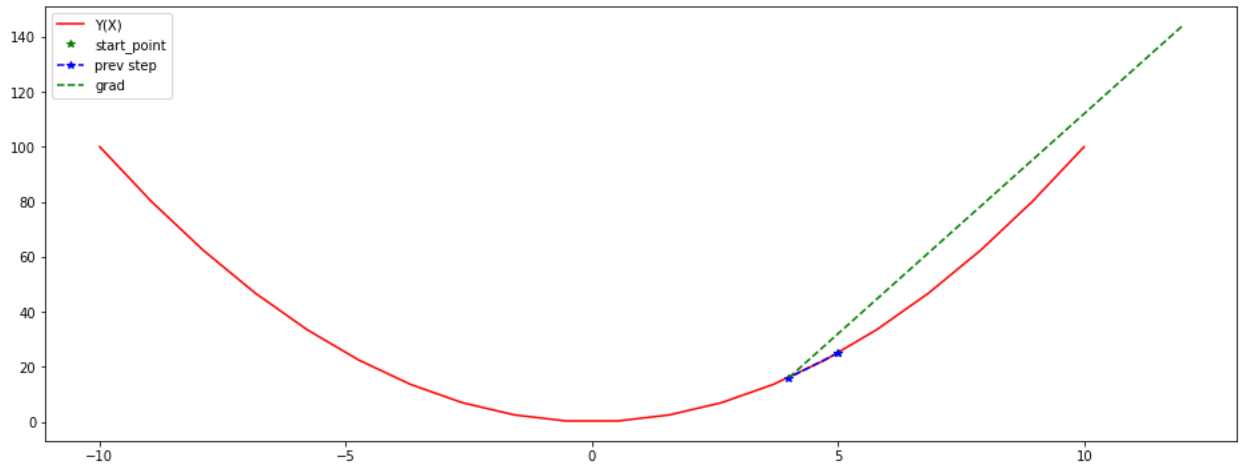


Рисунок 26 Наискорейший рост функции

Но если будем двигаться по этому вектору, то к минимуму функции не придем, поэтому нужно идти в противоположном направлении, а значит брать **антиградиент**.

Но при этом помним, что если сходить на полный антиградиент, то можем перелететь минимум, поэтому домножим на скорость обучения. И получаем еще одну точку, которая уже ближе к минимуму функции (См. рис. 27)

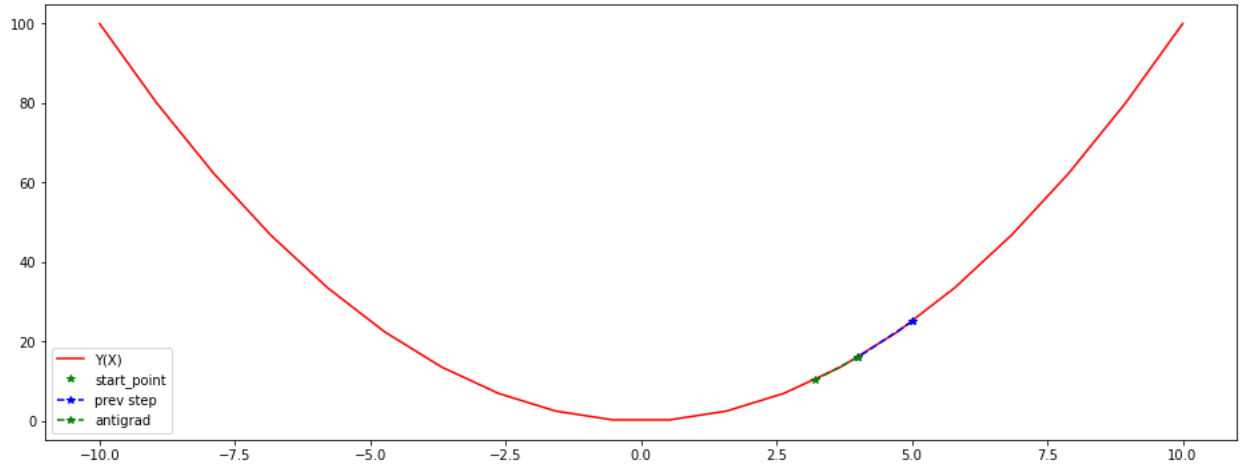
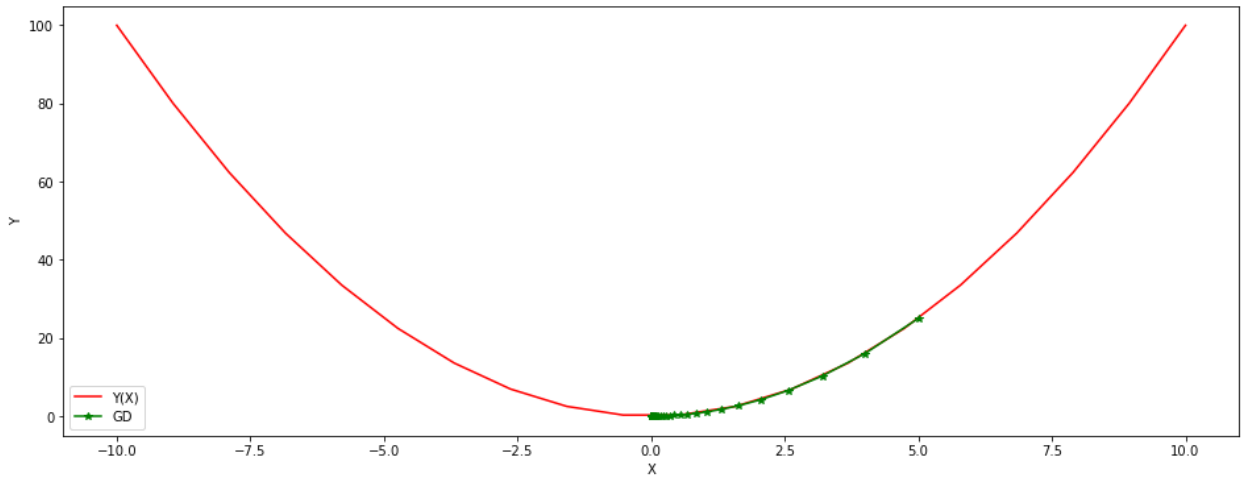


Рисунок 27 Отображение второй точки

Чтобы облегчить себе жизнь, выполним цикл на 100 итераций, график станет выглядеть таким образом (См. рис. 28).

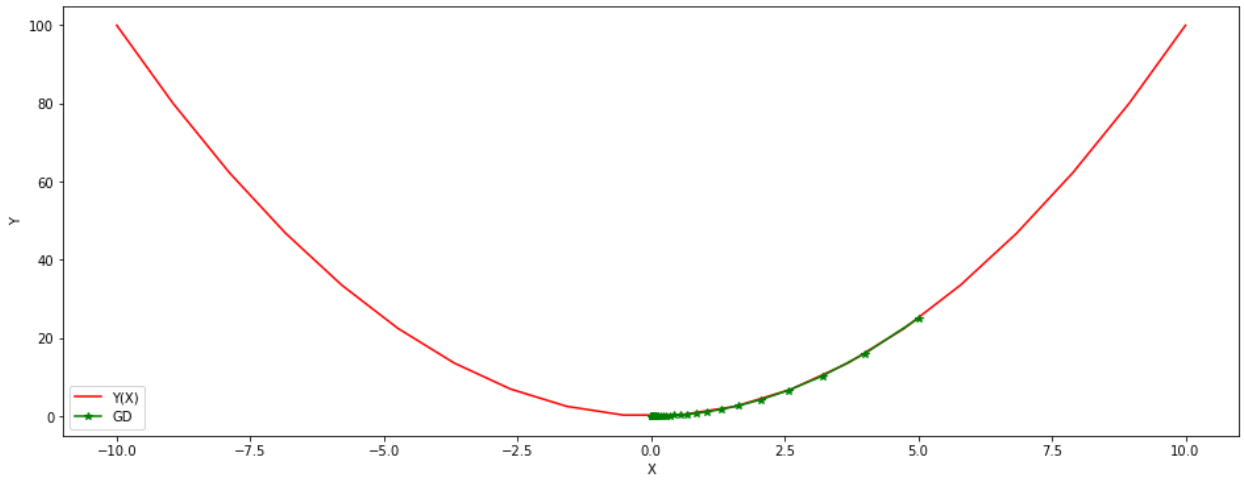



```
Итерация: 95
Текущая точка 3.108270227561169e-09| Следующая точка 2.4866161820489353e-09
----------------------------------------------------------
Итерация: 96
Текущая точка 2.4866161820489353e-09| Следующая точка 1.989292945639148e-09
----------------------------------------------------------
Итерация: 97
Текущая точка 1.989292945639148e-09| Следующая точка 1.5914343565113183e-09
----------------------------------------------------------
Итерация: 98
Текущая точка 1.5914343565113183e-09| Следующая точка 1.2731474852090548e-09
----------------------------------------------------------
Итерация: 99
Текущая точка 1.2731474852090548e-09| Следующая точка 1.0185179881672439e-09
----------------------------------------------------------
минимум 1.0185179881672439e-09, количество затраченных итераций: 99
```

Рисунок 28 100 итераций цикла

Но здесь значения самой лучшей минимальной точки на последних шагах очень похожи и на самом деле мы могли не ждать столько итераций и выйти из цикла раньше. Итоговые точки отличатся друг от друга совсем немного, получается мы топчемся на месте, хоть и были относительно близки к нулю еще 50 итераций назад.

Для этого введем значение eps, равной 0.0001, с помощью которого будем проверять разницу между текущей точкой и следующей точкой и если она меньше eps (а значит точки очень близки), то можем выйти из алгоритма.




```
Итерация: 39
Текущая точка 0.0008307674973655728| Следующая точка 0.0006646139978924582
----------------------------------------------------------
Итерация: 40
Текущая точка 0.0006646139978924582| Следующая точка 0.0005316911983139665
----------------------------------------------------------
Итерация: 41
Текущая точка 0.0005316911983139665| Следующая точка 0.00042535295865117324
----------------------------------------------------------
Итерация: 42
Текущая точка 0.00042535295865117324| Следующая точка 0.0003402823669209386
----------------------------------------------------------
минимум 0.0003402823669209386, количество затраченных итераций: 42
```

Рисунок 29 – 42 итоговых итераций с eps = 0.0001

Алгоритму понадобилось всего лишь 42 итерации, разница между двумя точками оказалась меньше eps, а значит можем выйти из цикла схождения алгоритма — это называется критерий останова.

Градиентный спуск вручную

Как работает градиентный спуск разобрались, теперь применим его к данным о недвижимости в Бостоне. Попробуем добиться тех же значений наклона, сдвига и среднего квадратичного отклонения (MSE) как у модели sklearn.

Загружаем данные, создаем две переменные:

```
boston_df = pd.read_csv('C:/Users/User/Desktop/Work/session 1/Lr reg/boston.csv')
X = boston_df[['LSTAT']]
y = boston_df['MEDV']
X = np.array(X,type(float))
y = np.array(y,type(float))
```

Рисунок 30 Загрузка данных

Реализуем две функции:

1. mserror - функция среднеквадратичной ошибки
$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y_i - y\_pred_i)^2 = \frac{1}{n}\sum_{i=0}^{n}(y_i - (w_1 * X_i + w_o))^2 = \frac{1}{n}\sum_{i=0}^{n}(y_i - w_1 * X_i - w_o)^2$$

2. gr_mserror - градиент функции MSE. Распишем его отдельно для коэффициента сдвига и коэффициента наклона:

Сдвиг: $\frac{\partial MSE}{\partial w_0} = \frac{1*2}{n}\left(y_i - y\_pred_i\right) * -1$

Наклон: $\frac{\partial MSE}{\partial w_1} = \frac{1*2}{n}\left(y_i - y\_pred_i\right) * -X$

```
# функция, определяющая среднеквадратичную ошибку
def mserror(X, w1, w0, y):
    y_pred = w1 * X[:, 0] + w0 #w1 - наклон, X - признак "LSTAT", w0 - сдвиг
    return np.sum((y - y_pred) ** 2) / len(y_pred)

# функция градиента
def gr_mserror(X, w1, w0, y):
    y_pred = w1 * X[:, 0] + w0
    return np.array([2/len(X)*np.sum((y - y_pred)) * (-1), #Сдвиг
                     2/len(X)*np.sum((y - y_pred) * (-X[:, 0]))]) #Наклон
```

Рисунок 31 Функции MSE и градиента

Запустим цикл градиентного спуска. В начале инициализировали коэффициенты, затем на каждом шаге считаем градиент, умножаем его на шаг обучения и вычитаем его из предыдущих значений коэффициентов и так далее пока не поймем, что точки коэффициентов очень похожи друг на друга на соседних итерациях.

```python
# установка минимального значения, на которое должны изменяться веса
eps = 0.0001

# первоначальное точка
w1 = 0
w0 = 0

# размер шага (learning rate)
learning_rate = 0.001

next_w1 = w1
next_w0 = w0
# количество итерация
n = 100000
for i in range(n):
    cur_w1 = next_w1
    cur_w0 = next_w0

    # движение в негативную сторону вычисляемого градиента
    next_w0 = cur_w0 - learning_rate * gr_mserror(X, cur_w1, cur_w0, y)[0]
    next_w1 = cur_w1 - learning_rate * gr_mserror(X, cur_w1, cur_w0, y)[1]

    # остановка когда достигнута необходимая степень точности
    print(f"Итерация: {i}")
    print(f"Текущая точка {cur_w1, cur_w0}| Следующая точка {next_w1, next_w0}")
    print(f"MSE {mserror(X, cur_w1, cur_w0, y)}")
    print("------------------------------------------------------------")

    if (abs(cur_w1 - next_w1) <= eps) and (abs(cur_w0 - next_w0) <= eps):
        break
```

```
Текущая точка (-0.9375398353804576, 34.34547816703025)| Следующая точка (-0.9375458485867428, 34.345578325000666)
MSE 38.49343943424087
------------------------------------------------------------
Итерация: 10627
Текущая точка (-0.9375458485867428, 34.345578325000666)| Следующая точка (-0.9375518589025372, 34.3456784348261)
MSE 38.493429368882914
------------------------------------------------------------
Итерация: 10628
Текущая точка (-0.9375518589025372, 34.3456784348261)| Следующая точка (-0.9375578663292301, 34.34577849652969)
MSE 38.49341931319927
------------------------------------------------------------
Итерация: 10629
Текущая точка (-0.9375578663292301, 34.34577849652969)| Следующая точка (-0.9375638708682101, 34.345878510134575)
MSE 38.49340926718063
------------------------------------------------------------
Итерация: 10630
Текущая точка (-0.9375638708682101, 34.345878510134575)| Следующая точка (-0.9375698725208655, 34.34597847566387)
MSE 38.49339923081784
------------------------------------------------------------
```

Рисунок 32 10630 итераций при размере шага = 0,001 и eps = 0,0001

С шагом 0.001 пришлось пройти много итераций, но результат схож с моделью sklearn. Это можно увидеть в таблице (См. рис. 33):

	Наклон вручную	Сдвиг вручную	MSE вручную	Наклон модели sklearn	Сдвиг модели sklearn	MSE модели sklearn
0	-0.93757	34.345978	38.493399	-0.950049	34.553841	38.482967

Рисунок 33 Сравнение двух подходов

Визуализируем регрессию, которую реализовали вручную с помощью градиентного спуска и через модуль sklearn с использованием метода наименьших квадратов:

```
fig = plt.figure(figsize=(10, 6))

x = np.arange(0, 40)
our_model_y = next_w1 * x + next_w0

plt.plot(X, model_y_sk, linewidth=2, color = "r", label=f'Модель sklearn = {model_a:.2f}x + {model_b:.2f}')
plt.plot(x, our_model_y, '--g', linewidth=2, label=f'Вручную = {next_w1:.2f}x + {next_w0:.2f}')
plt.scatter(X, y)
plt.grid()
plt.xlabel('feature')
plt.ylabel('target')
plt.legend(prop={'size': 16})
plt.show()
```

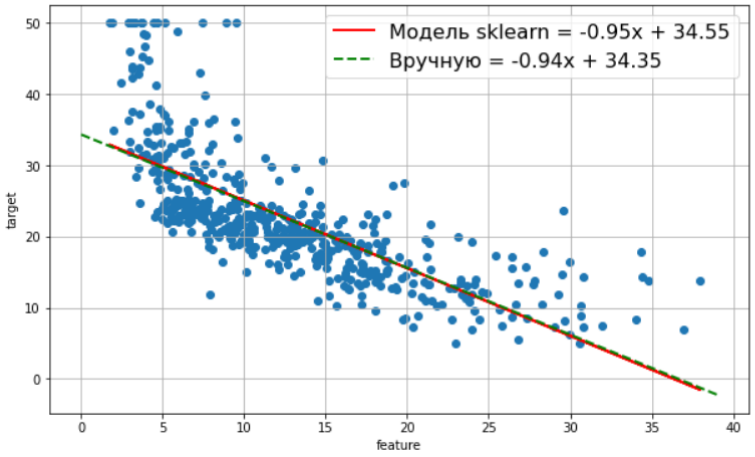


Рисунок 34 Визуализация двух подходов по реализации линейной регрессии

Визуализированные линии наслаиваются друг на друга. В целом, оба похода приводят к хорошим результатам.

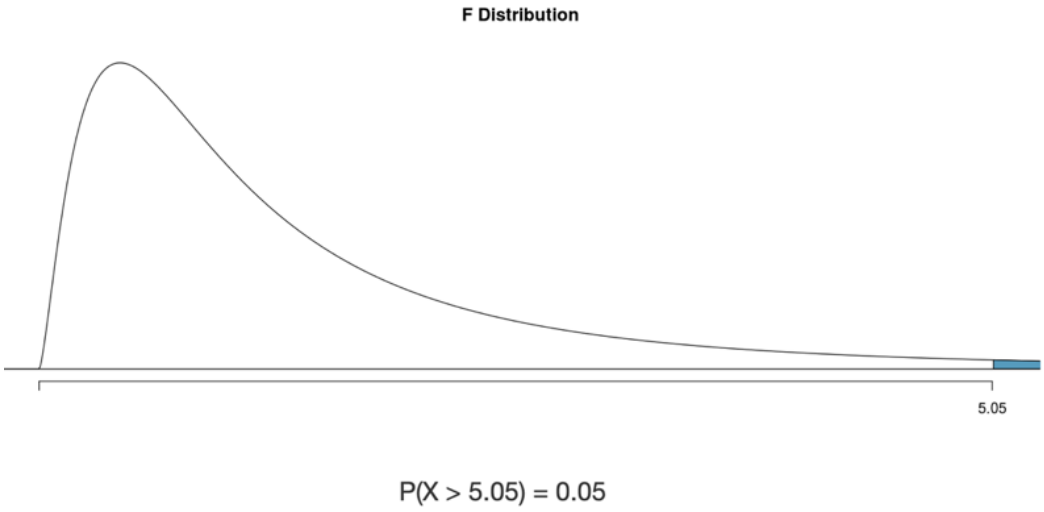
Дисперсионный анализ

Регрессия и дисперсионный анализ (ANOVA) – два статистических метода, использующих общую линейную модель (GLM), в основе которых лежит предположение о том, что зависимая переменная представляет собой функцию от одной или более независимых переменных.

Дисперсия – характеристика рассеивания данных вокруг их среднего значения.

Дисперсионный анализ (ANOVA) – статистическая процедура, используемая для сравнения средних значений определенной переменной в двух и более независимых группах.

Основная статистика в дисперсионном анализе – F-отношение, используемое для выявления статистической значимости различий между группами.



F-распределение для числа групп – 5 и числа степеней свободы – 5

В дисперсионном анализе теоретическое распределение F-отношения не является нормальным, оно подчиняется распределению Фишера.

F-значение всегда является положительным, потому вероятность отклонения рассчитывается только в правую сторону.

В дисперсионном анализе рассматривается отношение двух дисперсий: **межгрупповой** и **внутригрупповой**.

Общая сумма квадратов SST (общая изменчивость данных) **–** показатель, характеризующий степень изменчивости данных без учета разделения их на группы. Вычисляется общая сумма квадратов следующим образом:

- для каждого наблюдения рассчитывается насколько оно отклонится от среднего значения,
- складывается сумма квадратов полученных отклонений.

Общая сумма квадратов SST получена из двух источников: **межгрупповая сумма квадратов** SSB (характеристика, показывающая насколько групповые средние отклоняются от общего среднего) и **внутригрупповая сумма квадратов** SSW (сумма квадратов отклонений от среднего внутри каждой из групп).

Межгрупповая дисперсия MSB, объяснённая влиянием фактора, характеризует рассеивание значений между градациями (группами) вокруг средней всех данных.

Внутригрупповая дисперсия MSW, необъяснённая, характеризует рассеивание данных внутри градаций фактора (групп) вокруг средних значений этих групп.

Отношение межгрупповой и внутригрупповой дисперсий – **фактическое отношение Фишера**. Его сравнивают с **критическим значением отношения Фишера**. В случае, когда фактическое отношение Фишера превышает критическое, то средние классов градации различны, а исследуемый фактор оказывает существенное влияние на изменение данных. В обратном случае: средние классов градации друг от друга не отличаются, а фактор не оказывает существенного влияния на изменение данных.

Целью дисперсионного анализа является исследование наличия/отсутствия существенного влияния некоторого количественного/качественного фактора на изменения исследуемого признака.

Фактор, предположительно имеющий/не имеющий существенное влияние, делят на группы и на основе исследования значимости средних в наборах данных, соответствующих группам фактора, выясняют одинаково ли влияние фактора.

Пример 1. Исследование зависимости прибыли предприятия от типа используемого сырья. В данном случае группы – типы сырья.

Пример 2. Исследование зависимости себестоимости выпуска единицы продукции от размера предприятия. Здесь группы – величины предприятий (малое, среднее, большое).

Минимальное число групп в дисперсионном анализе – две. Группы могут быть количественные и качественные.

В дисперсионном анализе вычисляется удельный вес суммарного воздействия одного/нескольких факторов. Насколько влияние фактора существенно, исследуется с помощью гипотез:

Нулевая гипотеза H_0 утверждает, что все a классов градации имеют одинаковые значения средних: $\mu_1 = \mu_2 = ... = \mu_a$.

Альтернативная гипотеза H_1: не все классы градации имеют одно значение средних.

Однофакторный дисперсионный анализ

При формировании групп для сравнения в однофакторном дисперсионном анализе используется только одна переменная (фактор).

Пример. Исследуется эффективность работы нового станка по обработке металлов с помощью дисперсионного анализа. Сравнение проводится с работой старого станка, который уже используется в производстве. В данном исследовании фактор – используемый станок. У него два уровня: новый, старый станки.

В дисперсионном анализе фактор может иметь более двух уровней.

Однофакторный дисперсионный анализ с двумя уровнями аналогичен t-критерию. Нулевая гипотеза обычно говорит о равенстве средних двух групп, альтернативная – о различии средних (двусторонний тест) или различии в определенном направлении (односторонний тест).

Основные условия проведения дисперсионного анализа:

1. Зависимая переменная должна быть непрерывной, неограниченной/изменяющейся в широком интервале и представлена интервальными/характеризующими отношения данными; факторы должны быть дихотомическими/категориальными.
2. Каждое значение зависимой переменной не должно зависеть от других ее значений.

Исключения: рассматривается временная зависимость или значения были измерены у объектов, которые объединены в группы (члены одной семьи, учащиеся в одном классе) и это повлияло на зависимую переменную.

3. В каждой группе непрерывная переменная имеет приблизительно нормальное распределение. Нормальность распределения можно

проверить, используя гистограмму («на глаз») или статистические тесты на нормальность.

4. Дисперсии изучаемых групп должны быть приблизительно одинаковыми. Проверить похожесть дисперсий можно с помощью теста Левина, в котором нулевая гипотеза гласит, что дисперсия однородна, и если результат теста Левина статистически не значим (при применении критерия $\alpha < 0{,}05$), то дисперсии достаточно похожи.

Некоторые условия проведения дисперсионного анализа могут нарушаться, например, F-статистика надежна в случае, когда распределение непрерывной переменной отлично от нормального, а размеры групп одинаковы. Одинаковый размер обеспечивает и устойчивость F-статистики к нарушениям однородности дисперсии. А нарушение условия независимости может сильно исказить результаты.

Однофакторный дисперсионный анализ основан на том, что общая сумма квадратов SST получена из двух компонент: межгрупповой суммы квадратов SSB и внутригрупповой суммы квадратов SSW:

SST= SSB + SSW

Пример 3.

Группа 1	Группа 2	Группа 3
1	3	5
2	4	6
3	5	7

Сравниваем 3 группы, в каждой из которых по 3 значения.

Нулевая гипотеза: в генеральной совокупности нет значимых различий между средними, все средние трёх групп равны друг другу. Альтернативная гипотеза: хотя бы пара средних значимо различается между собой.

H_0: $\mu_1 = \mu_2 = \mu_3$

H_1: $\mu_1 \neq \mu_2 = \mu_3$ или $\mu_1 = \mu_2 \neq \mu_3$ или $\mu_1 \neq \mu_2 \neq \mu_3$

Вычислим среднее значение всех наблюдений:

$$\bar{\bar{x}} == \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{ij}}{n} = \frac{1+2+3+3+4+5+5+6+7}{9} = 4$$

Вычислим общую сумму квадратов:

$$SST = \sum_{i=1}^{m}\sum_{j=1}^{n_i}\left(x_{ij} - \bar{\bar{x}}\right)^2 (1-4)^2 + (2-4)^2 + (3-4)^2 +$$

$$+(3-4)^2 + (4-4)^2 + (5-4)^2 + (5-4)^2 + (6-4)^2 + (7-4)^2 = 30$$

Степени свободы для общей суммы квадратов:

$$dF_{SST} = n - 1 = 9 - 1 = 8$$

Вычислим средние значения внутри каждой из групп:

$$\bar{x}_i = \frac{\sum_{j=1}^{n_i} x_j}{n_i}, i = \overline{1,m}$$

$$\bar{x}_1 = \frac{1+2+3}{3} = 2$$

$$\bar{x}_2 = \frac{3+4+5}{3} = 4$$

$$\bar{x}_3 = \frac{5+6+7}{3} = 6$$

Внутригрупповая сумма квадратов:

$$SSW = \sum_{i=1}^{m}\sum_{j=1}^{n_i}\left(x_{ij} - \bar{x}_i\right)^2 = (1-2)^2 + (2-2)^2 + (3-2)^2 +$$

$$+(3-4)^2 + (4-4)^2 + (5-4)^2 +$$

$$+(5-6)^2 + (6-6)^2 + (7-6)^2 = 6$$

Степени свободы для внутригрупповой суммы квадратов:

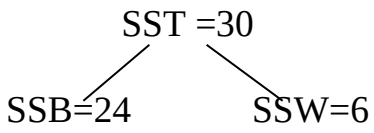
$$dF_{SSW} = n - m = 9 - 3 = 6$$

Межгрупповая сумма квадратов:

$$SSB = \sum_{i=1}^{m} n_i(\bar{x}_i - \bar{\bar{x}})^2 = 3(2-4)^2 + 3(4-4)^2 + 3(6-4)^2 = 24$$

Степени свободы для межгрупповой суммы квадратов:

$$dF_{SSB} = m - 1 = 3 - 1 = 2$$



Получили, что большая часть общей изменчивости обеспечивается благодаря межгрупповой сумме квадратов, значит группы значительно различаются между собой.

Межгрупповая дисперсия:

$$MS_B = \frac{SSB}{dF_{SSB}} = \frac{24}{2} = 12$$

Внутригрупповая дисперсия:

$$MS_W = \frac{SSW}{dF_{SSW}} = \frac{6}{6} = 1$$

Вычислим F-значение:

$$F = \frac{MS_B}{MS_W} = \frac{12}{1} = 12$$

Критическое значение отношения Фишера:

$F_{0,05;\ 2;6} = 5,14$

Так как фактическое отношение Фишера меньше критического:

$$F = 12 > 5,14 = F_{\ 0,05;\ 2;6}$$

можно сделать вывод, что есть существенные различия между группами.

Пример 4.

Мы хотим проверить, отличается ли возраст избирателей на основе какой-либо категориальной переменной, например от расы избирателя. Для этого сгенерируем данные с различными параметрами, что позволит продемонстрировать выполнения дисперсионного анализа в Python.

Сгенерируем выборку из 1000 элементов, в которую включены следующие расы: asian, black, hispanic, white, other и возраста избирателей.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
```

```
np.random.seed(12)
races = ["asian", "black", "hispanic", "white", "other"]

# генерируем случайные данные
voter_race = np.random.choice(a = races, p = [0.05, 0.15, 0.25, 0.05, 0.5], size = 1000) # категориальная переменная (раса избирателей)
voter_age = stats.poisson.rvs(loc = 18, mu = 30, size = 1000) # числовая переменная (возраст избирателей)
```

Так как все возраста генерируются одинаково, то это говорит нам о том, что они все из одной генеральной совокупности, поэтому ANOVA должна дать результат, что существенной разницы нет.

```
# группируем данные возраста по рассе
voter_frame = pd.DataFrame({"race":voter_race, "age":voter_age})
groups = voter_frame.groupby("race").groups

# добавляем конкретные группы
asian = voter_age[groups["asian"]]
black = voter_age[groups["black"]]
hispanic = voter_age[groups["hispanic"]]
white = voter_age[groups["white"]]
other = voter_age[groups["other"]]
voter_frame.head()
```

	race	age
0	black	51
1	other	49
2	hispanic	51
3	other	48
4	asian	56

```
#выполняем ANOVA
stats.f_oneway(asian, black, hispanic, white, other)
```

```
F_onewayResult(statistic=1.7744689357329695, pvalue=0.13173183201930463)
```

Рассмотрим альтернативный способ, используем функцию anova_lm() из библиотеки statsmodels:

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

```
model = ols('age ~ race', data = voter_frame).fit()
anova_result = sm.stats.anova_lm(model, typ = 2)
print(anova_result)
```

```
             sum_sq     df         F    PR(>F)
race        199.369    4.0  1.774469  0.131732
Residual  27948.102  995.0       NaN       NaN
```

Попробуем сгенерировать чуть измененные данные. Сгенерируем возраста для белых людей отдельно. В качестве среднего возраста возьмем 32 года. Это изменение должно отразиться на результатах ANOVA. Он должен показать различность возрастов белых людей и остальных.

```
np.random.seed(12)

# генерируем случайные данные
voter_race = np.random.choice(a = races, p = [0.05, 0.15, 0.25, 0.05, 0.5], size = 1000)

white_ages = stats.poisson.rvs(loc = 18, mu = 32, size = 1000)
voter_age = stats.poisson.rvs(loc = 18, mu = 30, size = 1000)
voter_age = np.where(voter_race == "white", white_ages, voter_age)

# группируем данные возраста по расе
voter_frame = pd.DataFrame({"race":voter_race, "age":voter_age})
groups = voter_frame.groupby("race").groups

# добавляем конкретные группы
asian = voter_age[groups["asian"]]
black = voter_age[groups["black"]]
hispanic = voter_age[groups["hispanic"]]
white = voter_age[groups["white"]]
other = voter_age[groups["other"]]
```

```
#выполняем ANOVA
stats.f_oneway(asian, black, hispanic, white, other)
```

```
F_onewayResult(statistic=3.6470318084857154, pvalue=0.00586731196131632)
```

```
model = ols('age ~ race', data = voter_frame).fit()
anova_result = sm.stats.anova_lm(model, typ = 2)
print(anova_result)
```

```
                sum_sq     df         F    PR(>F)
race        472.278126    4.0  3.647032  0.005867
Residual  32212.272874  995.0       NaN       NaN
```

ANOVA нашел различие, поскольку p-значение меньше 0,05. Это означает, что фактор раса оказывает статистически значимое влияние на возраст избирателей, но было бы интересно узнать в каких именно группах есть влияние. Для этого необходимо вернуться на шаг назад. Можно использовать t критерий Стьюдента для всех пар рас, но такой метод при большом разнообразии групп может дать слишком большую ошибку.

Метод Бонферрони является одним из наиболее простых и известных способов контроля над групповой вероятностью ошибки.

Предположим, что мы применили определенный статистический критерий 3 раза (например, сравнили при помощи критерия Стьюдента средние значения групп A и B, A и C, и B и C) и получили следующие три P-значения: 0.01, 0.02 и 0.005. Если мы хотим, чтобы групповая вероятность ошибки при этом не превышала определенный уровень значимости $\alpha = 0.05$, то, согласно методу Бонферрони, мы должны сравнить каждое из полученных P-значений не с α, а с $\frac{\alpha}{m}$, где m – число проверяемых гипотез. Деление исходного уровня значимости α на m – это и есть поправка Бонферрони. В рассматриваемом примере каждое из полученных P-значений необходимо было бы сравнить с $\frac{0.05}{3} = 0.017$. В результате мы выяснили бы, что P-значение для второй гипотезы (0.02) превышает 0.017 и, соответственно, у нас не было бы оснований отвергнуть эту гипотезу.

Вместо деления изначально принятого уровня значимости на число проверяемых гипотез, мы могли бы умножить каждое из исходных P-значений

на это число. Сравнив такие скорректированные P-значения (англ. adjusted P-values; обычно обозначаются буквой q) с α, мы пришли бы к точно тем же выводам, что и при использовании поправки Бонферрони.

- 0.01*3 = 0.03 < 0.05: гипотеза отклоняется;
- 0.02*3 = 0.06 > 0.05: гипотеза принимается;
- 0.005*3 = 0.015 < 0.05: гипотеза отклоняется.

Вернемся к нашему примеру с избирателями. Выполним попарные сравнения.

```
# перебираем все пары
race_pairs = []

for race1 in range(4):
  for race2 in range (race1 + 1, 5):
    race_pairs.append((races[race1], races[race2]))

# t-test
for race1, race2 in race_pairs:
  print(race1, race2)
  print(stats.ttest_ind(voter_age[groups[race1]], voter_age[groups[race2]]))
```

```
asian black
Ttest_indResult(statistic=0.8386446909747979, pvalue=0.4027281369339345)
asian hispanic
Ttest_indResult(statistic=-0.42594691924932293, pvalue=0.6704669004240726)
asian white
Ttest_indResult(statistic=-2.235132300024921, pvalue=0.027828801627453537)
asian other
Ttest_indResult(statistic=0.3687230802619566, pvalue=0.712474249112879)
black hispanic
Ttest_indResult(statistic=-1.9527839210712925, pvalue=0.05156197171952594)
black white
Ttest_indResult(statistic=-3.4490459390086468, pvalue=0.0006893463707824467)
black other
Ttest_indResult(statistic=-0.9244438185606086, pvalue=0.3555931499524523)
hispanic white
Ttest_indResult(statistic=-2.1309216040170442, pvalue=0.033930889763891824)
hispanic other
Ttest_indResult(statistic=1.6450276425039192, pvalue=0.10037925272137736)
white other
Ttest_indResult(statistic=3.306112013211683, pvalue=0.0010062493632570478)
```

Мы имеем 10 сравнений, поэтому m = 10. Поэтому α необходимо уменьшить в 10 раз. То есть можно сказать, что критическое значение α у нас становится 0.005.

Сделаем выводы.

Группы	p-значение	гипотеза
asian - black	0.4027281369339345	принимается
asian - hispanic	0.6704669004240726	принимается
asian - white	0.027828801627453537	принимается
asian - other	0.712474249112879	принимается
black - hispanic	0.05156197171952594	принимается
black - white	0.0006893463707824467	отклоняется
black - other	0.3555931499524523	принимается
hispanic - white	0.033930889763891824	принимается
hispanic - other	0.10037925272137736	принимается
white - other	0.0010062493632570478	отклоняется

Видим, что различия в возрастах есть у следующих пар рас: black-white и white-other. Так же у asian-white и hispanic-white p-значение достаточно небольшое, но всё же больше 0.005. Эти результаты говорят о том что возраст светлокожих имеет отличие от остальных, что собственно мы и реализовали при генерации данных.

Также для такого рода анализа можно использовать пост-хок тесты.

Пост-хок тесты

После проведения дисперсионного анализа получаем данные о том, значимо ли влияние изучаемого фактора на данные: различаются ли между группами средние значения зависимой переменной. Однако результаты анализа не дают ответа на вопрос: благодаря каким различиям это влияние оказалось значимым?

Для решения данной задачи предназначены пост-хок тесты.

Свойства пост-хок тестов

- post-hoc тесты применяются когда влияние фактора значимо;
- тесты делают поправку для снижения вероятности ошибки I рода;
- они учитывают величину различий между средними значениями и количество сравниваемых между собой пар;
- тесты отличаются по степени консервативности (разумный компромисс – пост-хок тест Тьюки).

Пост-хок тест Тьюки

- строго контролирует значимость критерия α (0.05)
- одновременно проверяет все парные гипотезы;
- чувствителен к неравенству дисперсий;
- если размер групп имеет сильные различия, работает плохо.

Выполним пост-хок тест Тьюки для нашего примера с данными об избирателях, а также построим график с доверительными интервалами.

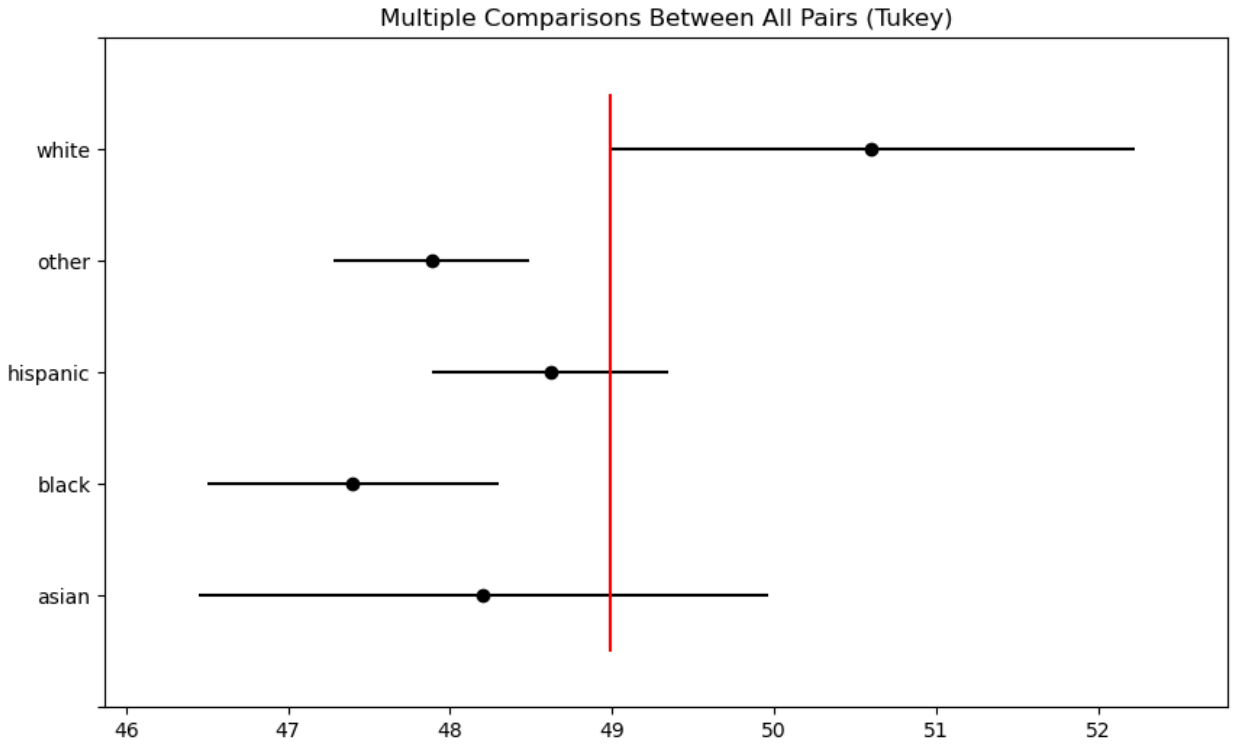
```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
```

```
tukey = pairwise_tukeyhsd(endog = voter_age, groups = voter_race, alpha = 0.05)
tukey.plot_simultaneous()
plt.vlines(x = 49.57, ymin = -0.5, ymax = 4.5, color = "red")
tukey.summary()
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
asian	black	-0.8032	0.924	-3.4752	1.8688	False
asian	hispanic	0.4143	0.9919	-2.1324	2.961	False
asian	other	-0.3191	0.9965	-2.7613	2.1231	False
asian	white	2.3955	0.2492	-0.8186	5.6095	False
black	hispanic	1.2175	0.2433	-0.406	2.8409	False
black	other	0.4841	0.8932	-0.9699	1.9381	False
black	white	3.1986	0.0056	0.653	5.7443	True
hispanic	other	-0.7334	0.4603	-1.9419	0.475	False
hispanic	white	1.9811	0.1649	-0.4326	4.3949	False
other	white	2.7146	0.0115	0.4113	5.0178	True

Заметим, что, как и в прошлом случае различие в возрастах есть у black-white и other-white. И можно сделать аналогичные выводы. Посмотрим на график со средними значениями и их доверительными интервалами.



Видим, что доверительные интервалы white-hispanic и white-asian перекрываются, поэтому пост-хок тесты показали что различия между ними не существенные.

Двухфакторный дисперсионный анализ без повторений

Двухфакторный дисперсионный анализ применяется для проверки возможной зависимости результативного признака от двух факторов.

Пусть m – число градаций первого фактора и k – число градаций второго фактора.

Двухфакторный дисперсионный анализ основан на том, что общая сумма

квадратов SST получена из трёх компонент: объяснённой влиянием фактора *A* суммы квадратов отклонений SSB_A, объяснённой влиянием фактора *B* суммы квадратов отклонений SSB_B и необъяснённой суммы квадратов отклонений (суммы квадратов отклонений ошибки):

$$SST = SSB_A + SSB_B + SSW$$

Рассмотрим следующий пример:

Ботаник хочет знать, влияет ли на рост растений воздействие солнечного света и частота полива. Она сажает 30 семян и позволяет им расти в течение двух месяцев при различных условиях солнечного света и частоты полива. Через два месяца она записывает высоту каждого растения в дюймах.

Используйте следующие шаги, чтобы выполнить двусторонний дисперсионный анализ, чтобы определить, оказывают ли частота полива и воздействие солнечного света существенное влияние на рост растений, а также определить, есть ли какой-либо эффект взаимодействия между частотой полива и воздействием солнечного света.

- вода: как часто поливалось каждое растение: ежедневно или еженедельно
- солнце: сколько солнечного света получило каждое растение: низкое, среднее или высокое
- высота: высота каждого растения (в дюймах) через два месяца

```
import numpy as np
import pandas as pd
```

```
# создаем данные
df = pd.DataFrame({'water': np.repeat(['daily', 'weekly'], 15),
 'sun': np.tile(np.repeat(['low', 'med', 'high'], 5), 2),
 'height': [6, 6, 6, 5, 6, 5, 5, 6, 4, 5,
 6, 6, 7, 8, 7, 3, 4, 4, 4, 5,
 4, 4, 4, 4, 4, 5, 6, 6, 7, 8]})
```

```
df.head()
```

	water	sun	height
0	daily	low	6
1	daily	low	6
2	daily	low	6
3	daily	low	5
4	daily	low	6

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

```
# выполняем ANOVA
model = ols('height ~ C(water) + C(sun) + C(water):C(sun)', data=df).fit()
sm.stats.anova_lm(model, typ=2)
```

	sum_sq	df	F	PR(>F)
C(water)	8.533333	1.0	16.0000	0.000527
C(sun)	24.866667	2.0	23.3125	0.000002
C(water):C(sun)	2.466667	2.0	2.3125	0.120667
Residual	12.800000	24.0	NaN	NaN

Мы можем видеть следующие p-значения для каждого из факторов в таблице:

- вода: p-значение = 0,000527
- солнце: p-значение = 0,0000002
- вода * солнце: p-значение = 0,120667

Поскольку p-значения для воды и солнца меньше 0,05, это означает, что оба фактора оказывают статистически значимое влияние на высоту растений.

А поскольку p-значение для эффекта взаимодействия (0,120667) составляет не менее 0,05, это говорит нам об отсутствии значительного эффекта взаимодействия между воздействием солнечного света и частотой полива.

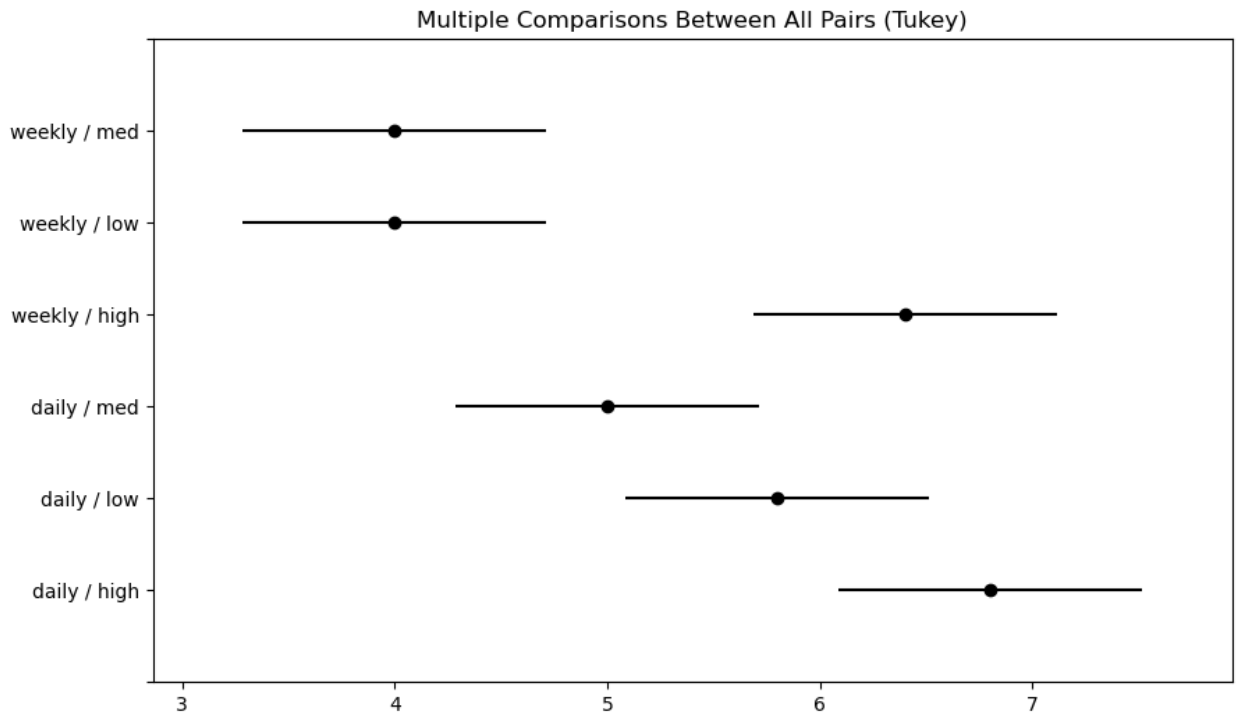
Выполним пост-хок тест Тьюки для нашего примера, а также построим график с доверительными интервалами.

```
df['combination'] = df.water + " / " + df.sun
```

```
tukey = pairwise_tukeyhsd(endog = df['height'], groups = df['combination'], alpha = 0.05)
tukey.plot_simultaneous()
#plt.vlines(x = 49.57, ymin = -0.5, ymax = 4.5, color = "red")
tukey.summary()
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
daily / high	daily / low	-1.0	0.2898	-2.4281	0.4281	False
daily / high	daily / med	-1.8	0.0079	-3.2281	-0.3719	True
daily / high	weekly / high	-0.4	0.951	-1.8281	1.0281	False
daily / high	weekly / low	-2.8	0.0	-4.2281	-1.3719	True
daily / high	weekly / med	-2.8	0.0	-4.2281	-1.3719	True
daily / low	daily / med	-0.8	0.5252	-2.2281	0.6281	False
daily / low	weekly / high	0.6	0.7827	-0.8281	2.0281	False
daily / low	weekly / low	-1.8	0.0079	-3.2281	-0.3719	True
daily / low	weekly / med	-1.8	0.0079	-3.2281	-0.3719	True
daily / med	weekly / high	1.4	0.057	-0.0281	2.8281	False
daily / med	weekly / low	-1.0	0.2898	-2.4281	0.4281	False
daily / med	weekly / med	-1.0	0.2898	-2.4281	0.4281	False
weekly / high	weekly / low	-2.4	0.0003	-3.8281	-0.9719	True
weekly / high	weekly / med	-2.4	0.0003	-3.8281	-0.9719	True
weekly / low	weekly / med	0.0	1.0	-1.4281	1.4281	False



Двухфакторный дисперсионный анализ с повторениями

Двухфакторный дисперсионный анализ с повторениями применяется для проверки не только возможной зависимости результативного признака от двух факторов – *A* и *B*, но и возможного взаимодействия факторов *A* и *B*.

Пусть *m* – число градаций фактора *A*, *k* – число градаций фактора *B*, *r* – число повторений.

В данном статистическом комплексе общая сумма квадратов SST получена из четырех компонент:

$$SST = SSB_A + SSB_B + SSB_{AB} + SSW$$

Практическая работа

1. Определить два вектора, представляющие собой число автомобилей, припаркованных в течении 5 рабочих дней у бизнес-центра на уличной стоянке и в подземном гараже.

День	Улица	Гараж
Понедельник	80	100
Вторник	98	82
Среда	75	105
Четверг	91	89
Пятница	78	102

1.1. Найти и интерпретировать корреляцию между переменными «Улица» и «Гараж» (подсчитать корреляцию по Пирсону).
1.2. Построить диаграмму рассеяния для вышеупомянутых переменных.

2. Найти и выгрузить данные. Вывести, провести предобработку и описать признаки.
2.1. Построить корреляционную матрицу по одной целевой переменной. Определить наиболее коррелирующую переменную, продолжить с ней работу в следующем пункте.
2.2. Реализовать регрессию вручную, отобразить наклон, сдвиг и MSE.
2.3. Визуализировать регрессию на графике.

3. Загрузить данные: 'insurance.csv'. Вывести и провести предобработку. Вывести список уникальных регионов.
3.1. Выполнить однофакторный ANOVA тест, чтобы проверить влияние региона на индекс массы тела (BMI), используя первый способ, через библиотеку Scipy.
3.2. Выполнить однофакторный ANOVA тест, чтобы проверить влияние региона на индекс массы тела (BMI), используя второй способ, с помощью функции anova_lm() из библиотеки statsmodels.
3.3. С помощью t критерия Стьюдента перебрать все пары. Определить поправку Бонферрони. Сделать выводы.
3.4. Выполнить пост-хок тесты Тьюки и построить график.
3.5. Выполнить двухфакторный ANOVA тест, чтобы проверить влияние региона и пола на индекс массы тела (BMI), используя функцию anova_lm() из библиотеки statsmodels.
3.6. Выполнить пост-хок тесты Тьюки и построить график.

4. Оформить отчет о проделанной работе, написать выводы.