Корреляция и линейная регрессия

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.

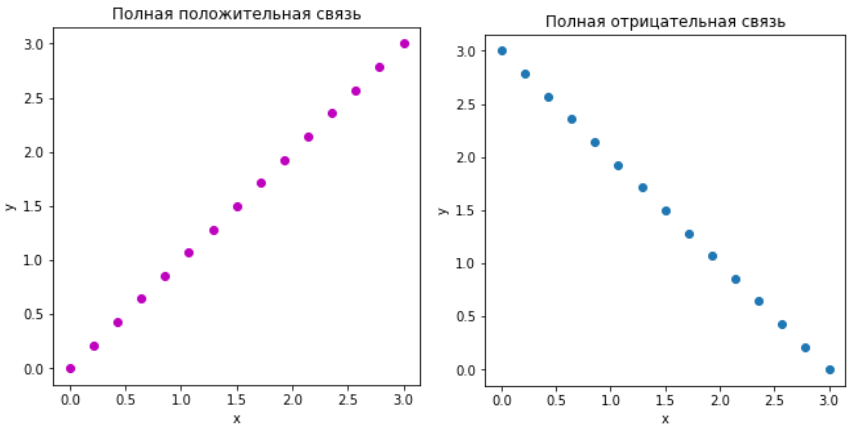
Корреляции помогают увидеть характер связи между двумя непрерывными переменными. Исследуют такую связь построением диаграммы рассеяния, в которой независимую переменную откладывают по оси x, зависимую – по оси y. Когда нет данных по поводу взаимоотношений переменных, то ось не имеет значения. Каждому 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
$0{,}20 \leq \|r\| \leq 0{,}29$	слабая
$\|r\| \leq 0{,}19$	очень слабая



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**
$p \leq 0{,}01$	высокая значимая
$0{,}01 < p \leq 0{,}05$	значимая
$0{,}05 < p \leq 0{,}10$	тенденция достоверной связи
коэффициент корреляции не достигает уровня статистической значимости	незначимая

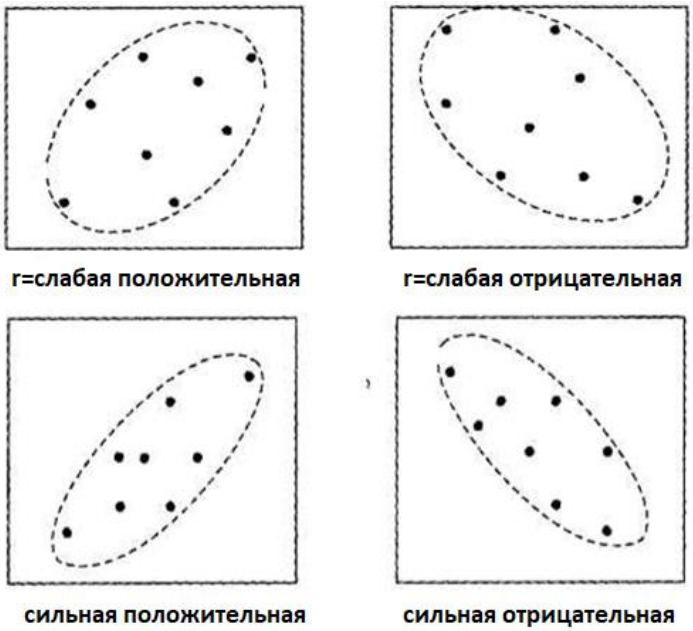


Рис. 1 – Схематическое изображение силы и направления корреляции

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 ≠ 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.        ]])
```

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
```

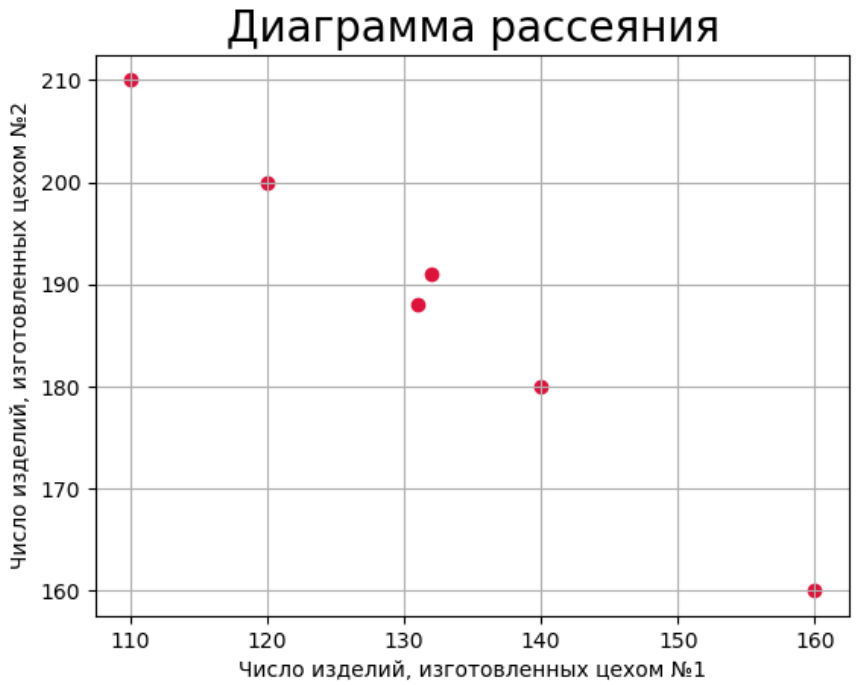
Коэффициент корреляции практически равен –1, что означает сильную отрицательную корреляцию.

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')
```



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
```

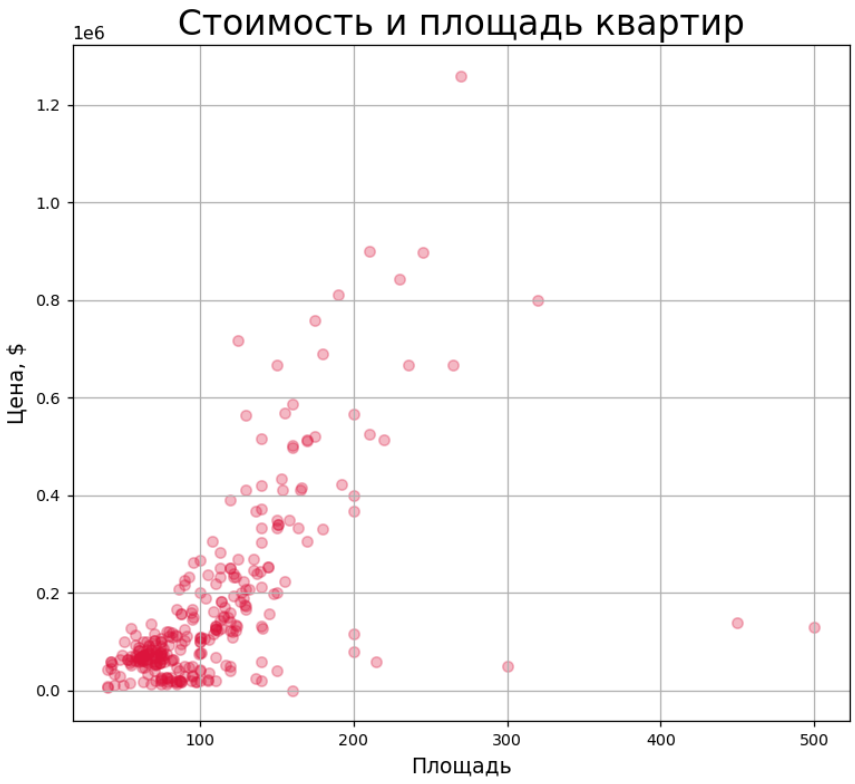
Коэффициент примерно равен 0.73, что означает высокую корреляцию.

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
```



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

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 параметры стиля доступны в pandas:

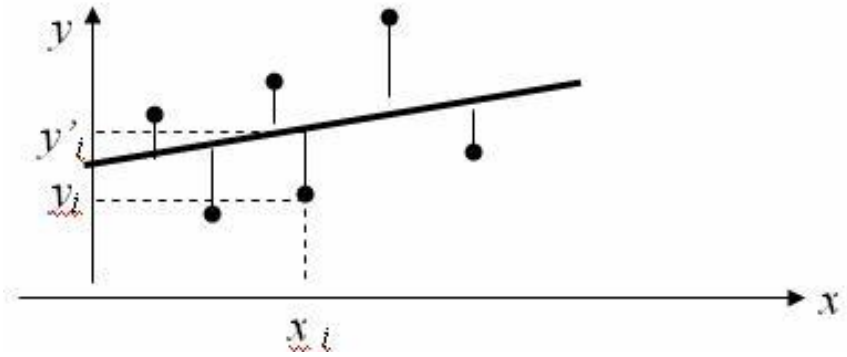
```
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	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

Метод наименьших квадратов (МНК)

Метод наименьших квадратов – принцип поиска коэффициентов регрессии путём минимизации суммы квадратов отклонений между реальными значениями признака и прогнозируемыми согласно предполагаемой форме зависимости (в нашем случае – линейной).

Таким образом, задача метода наименьших квадратов: подобрать прямую линию, которая ближе всего расположена к точкам корреляционного поля. Согласно МНК, линия выбирается так, чтобы сумма квадратов расстояний по вертикали между точками корреляционного поля и этой линией была бы минимальной.



Математическая запись данной задачи:

$$S = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - a + bx_i)^2 \to min$$

Значения y_i и x_i $(i = \overline{1,n})$ – данные наблюдений. В целевой функции задачи S они представляют собой константы. Переменными в данной функции являются искомые оценки параметров $\tilde{a}$ и $\tilde{b}$. Чтобы найти минимум целевой функции двух переменных необходимо вычислить частные производные данной функции по

каждому из параметров, приравнять их нулю:

$$\frac{\partial S}{\partial \tilde{a}} = 0, \frac{\partial S}{\partial \tilde{b}} = 0$$

В результате получим систему линейных уравнений:

$$\begin{cases} \sum_{i=1}^{n} y_i = \tilde{a} \cdot n + \tilde{b} \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} y_i \cdot x_i = \tilde{a} \sum_{i=1}^{n} x_i + \tilde{b} \sum_{i=1}^{n} {x_i}^2 \end{cases}$$

Введем обозначения:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \ , \qquad \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \ , \qquad \overline{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i\, y_i \ , \qquad \overline{x^2} = \frac{1}{n}\sum_{i=1}^{n} {x_i}^2$$

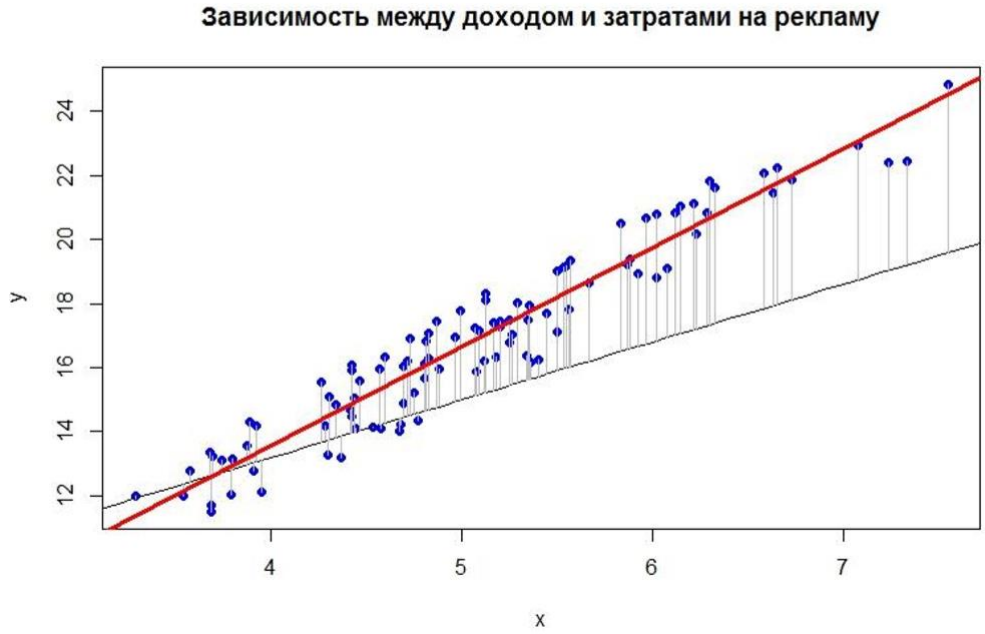
Решая данную систему, найдем искомые оценки параметров:

$$\tilde{b} = \frac{\overline{x} \cdot \overline{y} - \overline{xy}}{\overline{x}^2 - \overline{x^2}}$$

$$\tilde{a} = \overline{y} - \tilde{b}\overline{x}$$

Правильность подсчета параметров уравнения регрессии можно проверить сравнением сумм $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \tilde{y_i}$ (расхождение возможно из–за округления расчетов).

Учитывая свойства функции S нетрудно показать, что это решение является точкой минимума функции. Иными словами, значения $\tilde{a}$ и $\tilde{b}$ обеспечивают получение наилучшей линейной функции, отражающей зависимость переменной отклика *y* от фактора *x*. График этой линейной зависимости называется прямой регрессии (*y* на *x*). На приведённом ниже рисунке эта прямая имеет красный цвет.



Найденная линейная функция позволяет прогнозировать значение зависимого признака (y) по заданным значениям независимого фактора (x).

Простая линейная регрессия

Простая линейная регрессия — это подход к прогнозированию ответа с использованием одной функции.

Предполагается, что две переменные линейно связаны. Следовательно, мы пытаемся найти линейную функцию, которая максимально точно предсказывает значение отклика (y) как функцию функции или независимой переменной (x).

Давайте рассмотрим набор данных, в котором у нас есть значение ответа y для каждой функции x:

x	0	1	2	3	4	5	6	7	8	9
y	1	3	2	5	7	8	8	9	10	12

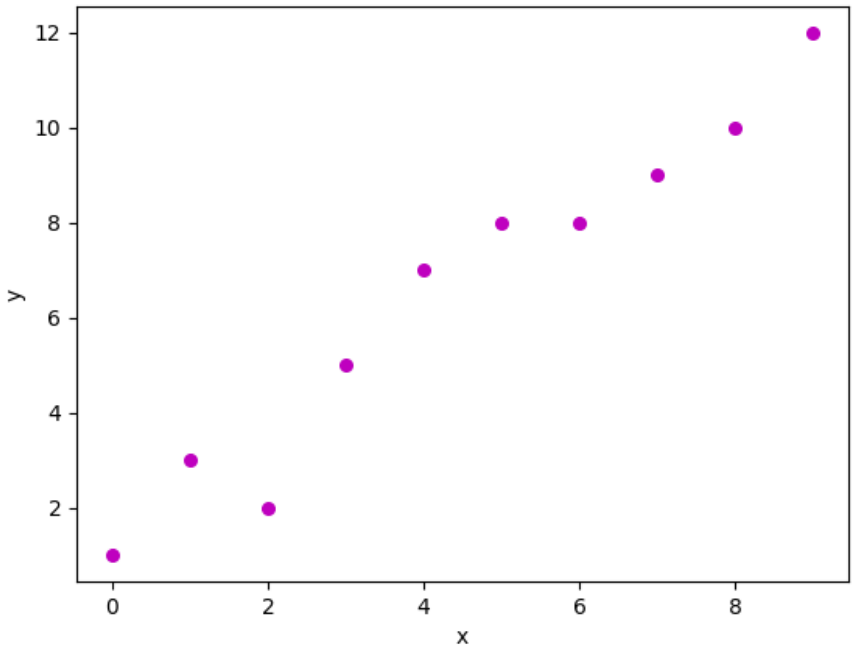
Для общности мы определяем:

x как вектор признаков, т.е. x = [x_1, x_2, ..., x_n],

y как вектор отклика, т.е. y = [y_1, y_2, ..., y_n]

для n наблюдений (в приведенном выше примере n= 10).

Точечный график приведенного выше набора данных выглядит следующим образом:



Теперь задача состоит в том, чтобы найти линию, которая лучше всего подходит для приведенного выше точечного графика, чтобы мы могли предсказать реакцию для любых новых значений объектов (т.е. значение x, отсутствующее в наборе данных).

Эта линия называется линией регрессии.

Уравнение линии регрессии представлено в виде:

$$h(x_i) = \beta_0 + \beta_1 x_i$$

Здесь,

h(x_i) представляет прогнозируемое значение ответа для i-го наблюдения.

b_0 и b_1 являются коэффициентами регрессии и представляют собой и наклон линии регрессии соответственно.

Чтобы создать нашу модель, мы должны “выучить” или оценить значения коэффициентов регрессии b_0 и b_1. И как только мы оценим эти коэффициенты, мы можем использовать модель для прогнозирования ответов.

Используем принцип наименьших квадратов. Рассмотрим:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i = h(x_i) + \varepsilon_i \Rightarrow \varepsilon_i = y_i - h(x_i)$$

Здесь e_i - остаточная ошибка в i-м наблюдении.

Наша цель - минимизировать общую остаточную ошибку.

Мы определяем квадрат ошибки или функцию затрат, J как:

$$J(\beta_0, \beta_1) = \frac{1}{2n} \sum_{i=1}^{n} \varepsilon_i^2$$

и наша задача - найти значения b_0 и b_1, для которых J(b_0,b_1) минимально.

Результатом является:

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

где SS_xy – это сумма перекрестных отклонений y и x:

$$SS_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} y_i x_i - n\bar{x}\bar{y}$$

а SS_xx — это сумма квадратов отклонений от x:

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

Реализация на python:

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

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

# Построение линейной регрессии
estimate_coef(x, y)
n = np.size(x)

m_x = np.mean(x)    # среднее значение вектора x и y
m_y = np.mean(y)

SS_xy = np.sum(y*x) - n*m_y*m_x    # вычисление перекрестного отклонения и отклонения около x
SS_xx = np.sum(x*x) - n*m_x*m_x

b_1 = SS_xy / SS_xx    # вычисление коэффициентов регрессии
b_0 = m_y - b_1*m_x

print("Коэффициенты: ", "b_1 = ", b_1, ' b_0 = ', b_0)
```

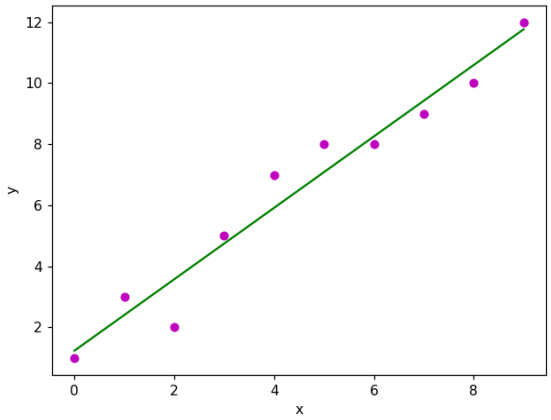
```
Using matplotlib backend: nbAgg
Коэффициенты:  b_1 =  1.1696969696969697  b_0 =  1.2363636363636363
```

```
plt.scatter(x, y, color = "m", marker = "o", s = 30)

y_pred = b[0] + b[1]*x    # Прогнозируемый вектор

plt.plot(x, y_pred, color = "g")    # построение линии регрессии

plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Реализация простой линейной регрессии на основе библиотеки:

```
import pandas
from pandas import DataFrame
import matplotlib.pyplot as plt
import numpy as np
import os

from sklearn.linear_model import LinearRegression
```

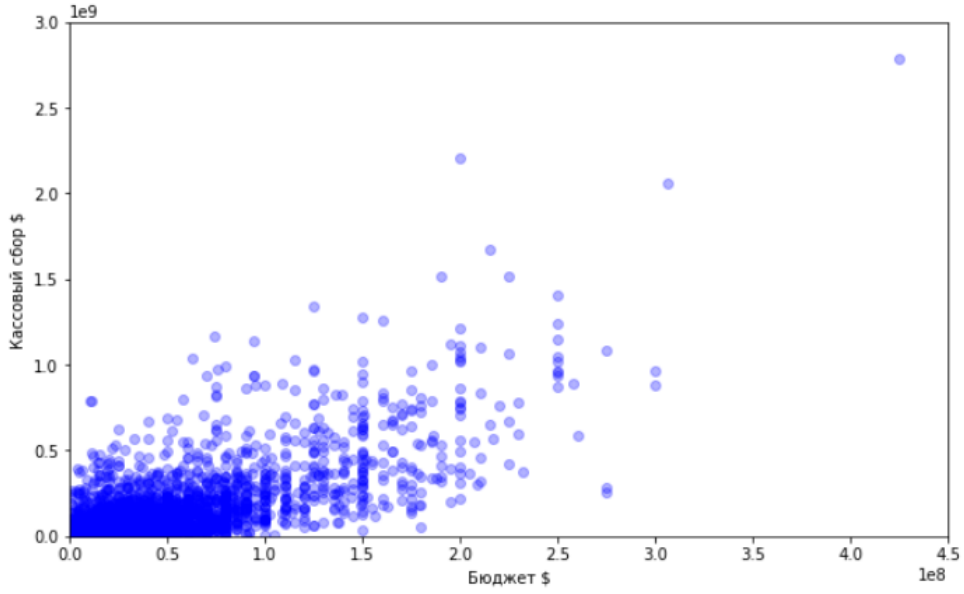
```
os.chdir('C:/Users/User/Desktop/Work/ЛН и КР')
data = pandas.read_csv('cost_revenue_clean.csv').astype(float)
```

```
data.describe()
```

	production_budget_usd	worldwide_gross_usd
count	5.034000e+03	5.034000e+03
mean	3.290784e+07	9.515685e+07
std	4.112589e+07	1.726012e+08
min	1.100000e+03	2.600000e+01
25%	6.000000e+06	7.000000e+06
50%	1.900000e+07	3.296202e+07
75%	4.200000e+07	1.034471e+08
max	4.250000e+08	2.783919e+09

```
X = DataFrame(data,columns=['production_budget_usd'], )
y = DataFrame(data,columns=['worldwide_gross_usd'])
X = np.array(X,type(float))    # Нормализация для нормального среза
y = np.array(y,type(float))
```

```
plt.figure(figsize=(10,6))
plt.scatter(X, y, color = 'blue',
            alpha=0.3)      # Плотность
plt.xlabel('Бюджет $')
plt.ylabel('Кассовый сбор $')
plt.ylim(0, 3000000000)
plt.xlim(0, 450000000)
plt.show()
```




```
regression = LinearRegression() #Модель необходимо обучить
regression.fit(X, y)    #    подаем данные
# X - Признак, на основе которого мы будем предсказывать, y - # Признак, который нужно предсказать
```

```
LinearRegression()
```

```
regression.coef_      # Наклон линии регрессии
```

```
array([[3.11150918]])
```

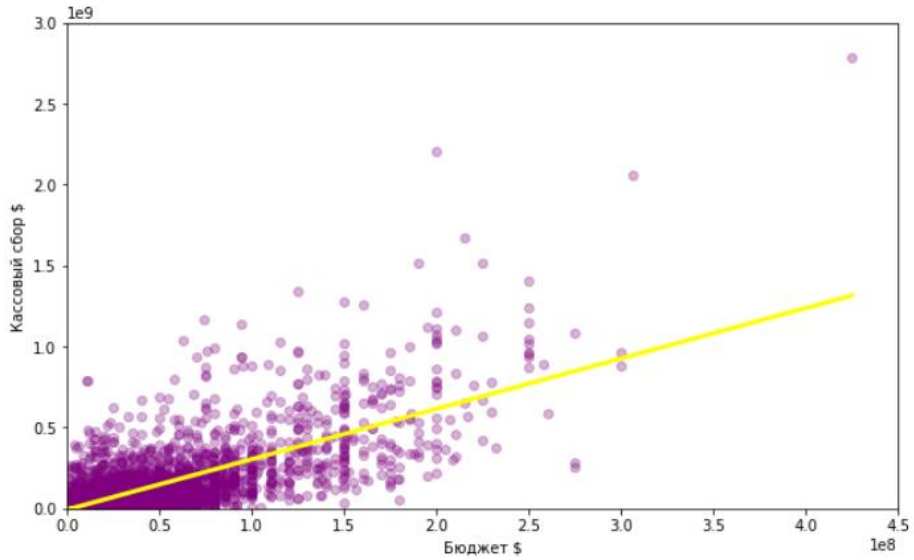
```
regression.intercept_     # y-перехват
```

```
array([-7236192.72913963])
```

```
plt.figure(figsize=(10,6))
plt.scatter(X, y, alpha=0.3, color = 'purple')

plt.plot(X, regression.predict(X), color='yellow', linewidth=3)

plt.xlabel('Бюджет $')
plt.ylabel('Кассовый сбор $')
plt.ylim(0, 3000000000)
plt.xlim(0, 450000000)
plt.show()
```



Практическое задание

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

Найти и интерпретировать корреляцию между переменными «Улица» и «Гараж» (подсчитать корреляцию по Пирсону).

2. Построить диаграмму рассеяния.
3. Загрузить bitcoin.csv.
4. Убрать из данных для обучения модели последние 14 дней.
5. Предсказать стоимость криптовалюты на следующие 14 дней с помощью линейной регрессии.
6. С помощью коэффициента детерминации сравнить исходные данные 14-ти последних дней, которые были обрезаны перед обучением модели, и 14 дней, которые предсказала модель.
7. Вывести угол наклона и y-перехват.
8. Построить диаграмму рассеяния (ось x – это close, ось y – это предсказанные моделью close).
9. Загрузить housePrice.csv
10. Произвести предобработку данных.
11. Реализовать линейную регрессию вручную, без использования библиотеки.
12. Вывести угол наклона и y-перехват.
13. Построить диаграмму.
14. Оформить отчет о проделанной работе, написать выводы.