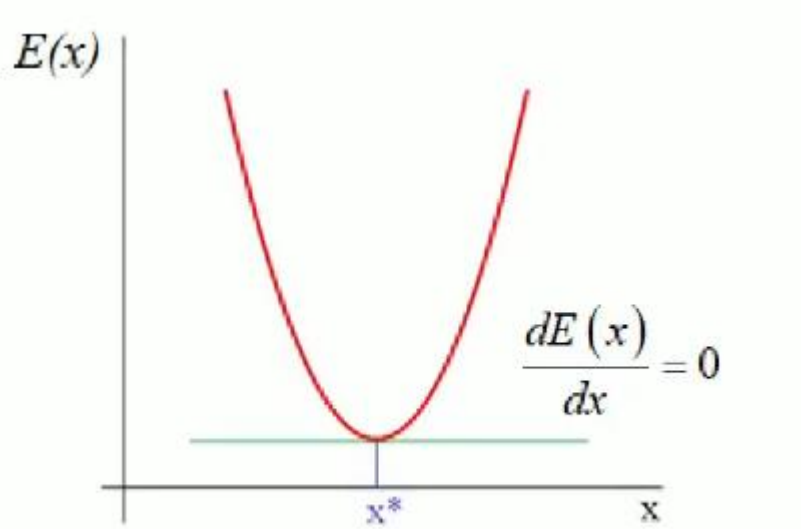
Язык программирования – Python 3.9

Дополнительные подключенные библиотеки – numpy, pandas, matplotlib

Мы работаем в следующих условиях: нам даны значения некоторых “измерений” и мы должны построить их линейную аппроксимацию функцией $f(x) = kx + b$. То есть, нужно подобрать такие коэффициенты k и b, чтобы функция наилучшим образом описывала входные данные. Для этого используем Метод Наименьших Квадратов:

$$E = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

Получается, нам нужно, чтобы данный критерий качества E принимал наименьшее значение. Если рассмотреть его график (квадратичной функции), то становится понятно, что минимум будет находится в точке x^*, где производная функции равна нулю.



В нашем случае мы имеем линейную функцию $f(x; k, b) = kx + b$, а значит нужно решить следующую систему уравнений:

$$\begin{cases} \dfrac{\partial E(k,b)}{\partial k} = \sum_{i=1}^{N} (y_i - f(x_i; k, b)) * \dfrac{\partial f(x_i)}{\partial k} = 0 \\ \dfrac{\partial E(k,b)}{\partial b} = \sum_{i=1}^{N} (y_i - f(x_i; k, b)) * \dfrac{\partial f(x_i)}{\partial b} = 0 \end{cases}$$

$$\frac{\partial f(x_i)}{\partial k} = x_i;\ \frac{\partial f(x_i)}{\partial b} = 1$$

$$\begin{cases} \sum_{i=1}^{N} (y_i - kx_i - b) * x_i = 0 \\ \sum_{i=1}^{N} (y_i - kx_i - b) * 1 = 0 \end{cases}$$

$$\begin{cases} \sum_{i=1}^{N} y_i * x_i - k * \sum_{i=1}^{N} x_i^2 - b * \sum_{i=1}^{N} x_i = 0 \\ \sum_{i=1}^{N} y_i - k * \sum_{i=1}^{N} x_i - b * N = 0 \end{cases}$$

$$\begin{cases} \frac{1}{N}\sum_{i=1}^{N} y_i * x_i - k * \frac{1}{N}\sum_{i=1}^{N} x_i^2 - b * \frac{1}{N}\sum_{i=1}^{N} x_i = 0 \\ \frac{1}{N}\sum_{i=1}^{N} y_i - k * \frac{1}{N}\sum_{i=1}^{N} x_i - b = 0 \end{cases}$$

Из теории вероятности мы знаем:

$\alpha_{1,1} = \frac{1}{N}\sum_{i=1}^{N} y_i x_i$, – первый смешанный начальный момент

$\alpha_2 = \frac{1}{N}\sum_{i=1}^{N} x_i^2$, – второй начальный момент

$m_x = \frac{1}{N}\sum_{i=1}^{N} x_i$, – мат. ожидание для велечины икс

$m_y = \frac{1}{N}\sum_{i=1}^{N} y_i$, – мат. ожидание, соответственно для велечины игрек

Подставляем в систему и получаем $\begin{cases} \alpha_{1,1} - k * \alpha_2 - b * m_x = 0 \\ m_y - k * m_x - b = 0 \end{cases}$

Отсюда выражаем искомые коэффициенты $k = \frac{\alpha_{1,1} - m_x * m_y}{\alpha_2 - m_x^2}$,

$$b = m_y - k * m_x$$

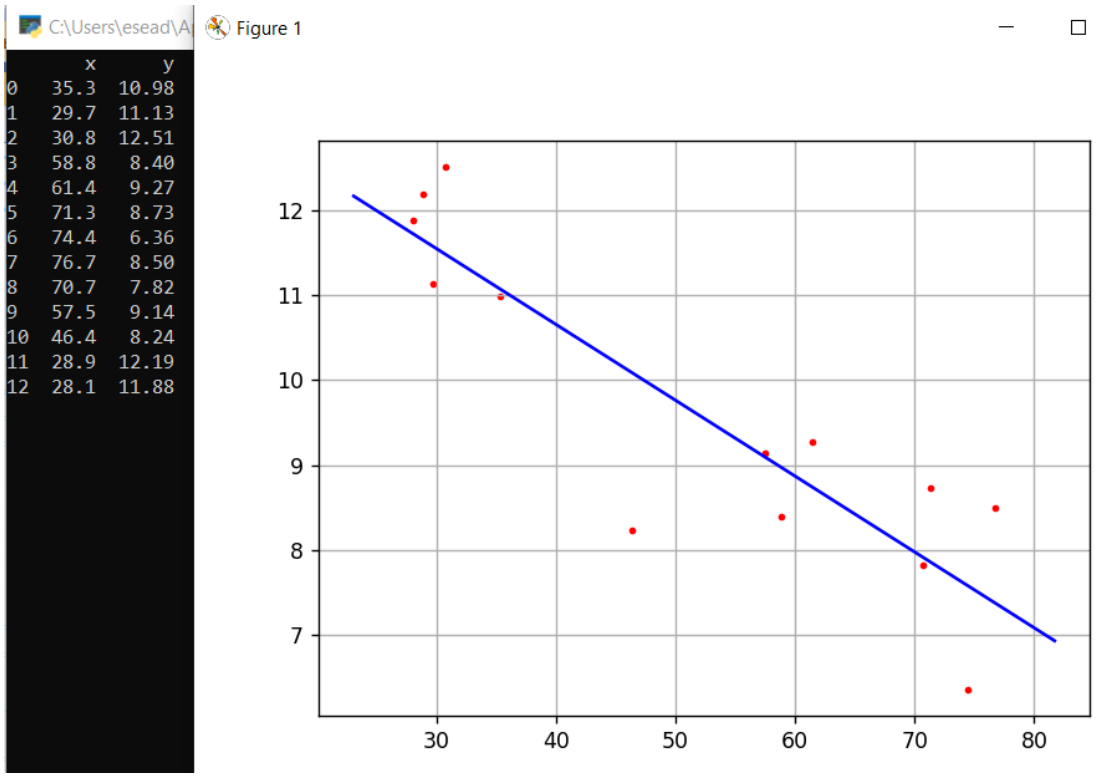
В интернете нашел подходящий датасет, заполнил таблицу эксель и приступил к импорту данных в программу с помощью pandas:

	x	y
2	35,30	10,98
3	29,70	11,13
4	30,80	12,51
5	58,80	8,40
6	61,40	9,27
7	71,30	8,73
8	74,40	6,36
9	76,70	8,50
10	70,70	7,82
11	57,50	9,14
12	46,40	8,24
13	28,90	12,19
14	28,10	11,88

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

filePath = 'dataset.xlsx'
data = pd.read_excel(filePath)
print(data, '\n')


x = np.array(data['x'])
y = np.array(data['y'])
N = len(x)

a1 = np.dot(x.T ,y)/N
a2 = np.dot(x.T, x)/N
mx = x.sum()/N
my = y.sum()/N

k = (a1-mx*my)/(a2-mx**2)
b = my - k*mx

xMax = np.max(x)
xMin = np.min(x)
f = np.array([k*(xMin-5)+b, k*xMin+b, k*xMax+b, k*(xMax+5)+b])
X = np.array([xMin-5, xMin, xMax, xMax+5])

plt.plot(X, f, c='blue')
plt.scatter(x, y, c = 'red', s=5)
plt.grid(True)
plt.show()
```

После этого “вытащил” из датафрейма массивы данных x и y и приступил к работе с ними. По вышевыведенным формулам нашел a1, a2, mx и my, а затем и заветные коэффициенты. Далее начал строить графики, при этом немного расширив границы на 5 ед., чтобы было более нагляднее. Построил график.

Результат программы:



Затем решил проверить, насколько точно работает код, для этого пришлось немного изменить код:

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

kk=0.5
bb=2
input_data = {"x" : np.arange(80),
              "y" : np.array([kk*z+bb for z in range (80)]) + np.random.normal(0, 3, 80)}

data = pd.DataFrame(input_data)
print(data, '\n')


x = np.array(data['x'])
y = np.array(data['y'])
N = len(x)

a1 = np.dot(x.T ,y)/N
a2 = np.dot(x.T, x)/N
mx = x.sum()/N
my = y.sum()/N
```

```
k = (a1-mx*my)/(a2-mx**2)
b = my - k*mx

xMax = np.max(x)
xMin = np.min(x)
f = np.array([k*(xMin-5)+b, k*xMin+b, k*xMax+b, k*(xMax+5)+b])
X = np.array([xMin-5, xMin, xMax, xMax+5])

plt.plot(X, f, c='blue')
plt.plot(np.array([kk*z+bb for z in range (80)]), c = 'red')
plt.scatter(x, y, c = 'red', s=5)
plt.grid(True)
plt.show()
```

Входные данные теперь – просто массив от 0 до 79 по иксу, и выбранные по закону нормального распределения значения для игреков, которые отталкивались от заранее заданной функции с коэффициентами k=0.5 и b=2. Далее идет практически неизмененный код, а в конце выводит два графика – синим цветом – вычесленная прямая, и красным – известная.

Результат: