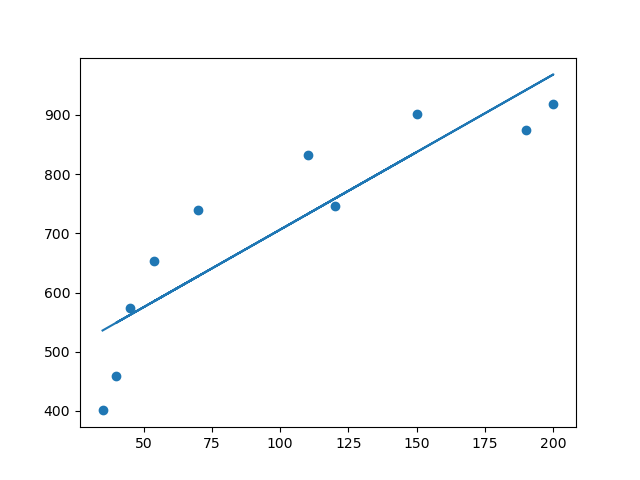
**Домашняя работа №7**

**Задача №1**

**Код Python:**

**import** inline **as** inline  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
  
zp = np.array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])  
ks = np.array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])  
  
*#линейная регрессия*b=(np.mean(zp\*ks)-np.mean(zp)\*np.mean(ks))/(np.mean(zp\*\*2)-np.mean(zp)\*\*2)  
a=np.mean(ks)-b\*np.mean(zp)  
T=[a,b]  
print(**"Коэффициенты линейной регрессии "**, T)  
  
y\_hat=444.18+2.62054\*zp  
  
plt.scatter(zp,ks)  
plt.plot(zp, 444.18+2.62054\*zp)  
plt.show()  
  
*#решаем матричным методом с intercept*zp=zp.reshape(10, 1)  
  
ks=ks.reshape(10, 1)  
  
  
zp=np.hstack([np.ones((10,1)),zp])  
*#print(zp)*B=np.dot(np.linalg.inv(np.dot(zp.T,zp)),zp.T@ks)  
print(**"Коэффициенты с интерсептом"**, B)  
  
*#решаем матричным методом без intercept*zp = np.array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])  
ks = np.array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])  
  
zp=zp.reshape(10, 1)  
  
ks=ks.reshape(10, 1)  
B=np.dot(np.linalg.inv(np.dot(zp.T,zp)),zp.T@ks)  
print(**"Коэффициенты без интерсепта"**, B)

**Выдача:**

****

Коэффициенты линейной регрессии [444.1773573243596, 2.620538882402765]

Коэффициенты с интерсептом [[444.17735732]

[ 2.62053888]]

Коэффициенты без интерсепта [[5.88982042**]]**

**Задача №2**

*#Задача 2. Посчитать коэффициент линейной регрессии при заработной плате (zp), используя градиентный спуск (без intercept).***import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
  
zp = np.array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])  
ks = np.array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])  
  
*#Градиентный спуск*X=zp  
y=ks  
  
*#функция потерь***def** mse\_(B1, y=y, X=X, n=8):  
 **return** np.sum(((B1\*X)-y)\*\*2/n)  
  
alpha=1e-6  
alpha  
  
B1=0.01  
n=10  
  
  
**for** i **in** range (3000):  
 B1 -=alpha\*(2/n)\*np.sum((B1\*X-y)\*X)  
 **if** i%500==0:  
 print(**'Iteration: {i}, B1={B1}, mse={mse}'**.format(i=i, B1=B1, mse=mse\_(B1)))

**Вывод:**

Iteration: 0, B1=0.172007868, mse=633650.6032213713

Iteration: 500, B1=5.889815520588435, mse=70646.07302006266

Iteration: 1000, B1=5.8898204201284905, mse=70646.07301964928

Iteration: 1500, B1=5.889820420132673, mse=70646.07301964928

Iteration: 2000, B1=5.889820420132673, mse=70646.07301964928

Iteration: 2500, B1=5.889820420132673, mse=70646.07301964928

**Задача №3**

В каких случаях для вычисления доверительных интервалов и проверки статистических гипотез используется таблица значений функции Лапласа, а в каких - таблица критических точек распределения Стьюдента?

1. Таблица значений функции Лапласа используется в случае, когда известно среднее квадратичное отклонение генеральной совокупности и генеральная совокупность распределена нормально;
2. Таблица критических распределений Стьюдента в случае, если среднее квадратичное отклонение генеральной совокупности не известно. При этом существует выборка, которая распределена нормально.

**Задача 4**

**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
  
zp = np.array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])  
ks = np.array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])  
  
*#Градиентный спуск*X=zp  
y=ks  
  
*#будем искать через условие равенства градинта 0*alpha, w1, w0, eps = 5e-5, 0.1, 0.1, 1e-10  
w1\_, w0\_ = w1 + 2 \* eps, w0 + 2 \* eps  
unconditional\_stop, i = 1000000, 0  
**while** i <= unconditional\_stop **and** abs(w1 - w1\_) \* abs(w0 - w0\_) > eps\*\*2:  
 w0\_, w1\_, i = w0, w1, i + 1  
 w1 -= alpha \* (2/len(X)) \* np.sum(((w0 + w1 \* X) - y) \* X)  
 w0 -= alpha \* (2/len(X)) \* np.sum((w0 + w1 \* X) - y)  
a, b = w0, w1  
print(**f'Линейная регрессия y = {**round(b, 4)**}x + {**round(a, 4)**}'**)  
print(**f'{**i**} - итераций'**)

**Вывод:**

Линейная регрессия y = 2.6205x + 444.1773

634045 - итераций