Практическое задание №7

1. **Даны значения величины заработной платы заемщиков банка (zp) и значения их поведенческого кредитного скоринга (ks): zp = [35, 45, 190, 200, 40, 70, 54, 150, 120, 110], ks = [401, 574, 874, 919, 459, 739, 653, 902, 746, 832]. Используя математические операции, посчитать коэффициенты линейной регрессии, приняв за X заработную плату (то есть, zp - признак), а за y - значения скорингового балла (то есть, ks - целевая переменная). Произвести расчет как с использованием intercept, так и без.**

1.Вариант – расчёт алгебраическим методом.

a.) С intercept-ом

$$b = \frac{n * \sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n * \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

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

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

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook
```

In [2]:
```
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])
```

In [3]:
```
b = (np.mean(zp*ks)-np.mean(zp)*np.mean(ks))/(np.mean(zp**2)-np.mean(zp)**2)
print('Коэффициент b =',b)
```

```
Коэффициент b = 2.620538882402765
```

In [4]:
```
a = np.mean(ks)-b*np.mean(zp)
print('Коэффициент a =',a)
```

```
Коэффициент a = 444.1773573243596
```

b.) Без intercept-а

$$\hat{y} = bx,$$

$$SSE(b) = \sum_{i=1}^{n} (y_i - bx_i)^2$$

$$\frac{\partial}{\partial b} SSE(b) = -2 \sum_{i=1}^{n} (y_i - bx_i) * x_i = 0$$

$$\sum_{i=1}^{n} (y_i x_i) = b * \sum_{i=1}^{n} (x_i^2)$$

$$b = \frac{\sum_{i=1}^{n} (y_i\, x_i)}{\sum_{i=1}^{n} (x_i^2)}$$

In [5]:
```
b_ = np.sum(zp*ks)/np.sum(zp**2)
print('Коэффициент b_ =',b_)
```

```
Коэффициент b_ = 5.889820420132689
```

2.Вариант – расчёт методом матриц.

$$\hat{B} = (X^T * X)^{-1} * X^T * Y$$

a.) С intercept-ом

In [11]:
```
ZP = zp.reshape(10,1)
ZP = np.hstack([np.ones((10,1)),ZP])
ZP
```

Out[11]:
```
array([[  1.,  35.],
       [  1.,  45.],
       [  1., 190.],
       [  1., 200.],
       [  1.,  40.],
       [  1.,  70.],
       [  1.,  54.],
       [  1., 150.],
       [  1., 120.],
       [  1., 110.]])
```

In [13]:
```
KS = ks.reshape(10,1)
KS
```

Out[13]:
```
array([[401],
       [574],
       [874],
       [919],
       [459],
       [739],
       [653],
       [902],
       [746],
       [832]])
```

In [17]:
```
B = np.dot(np.linalg.inv(np.dot(ZP.T, ZP)),ZP.T@KS)
print('Коэффициент a =', B[0,0])
print('Коэффициент b =', B[1,0])
```

```
Коэффициент a = 444.17735732435904
Коэффициент b = 2.620538882402766
```

b.) Без intercept-a

In [18]:
```
ZP = zp.reshape(10,1)
ZP
```

Out[18]:
```
array([[ 35],
       [ 45],
       [190],
       [200],
       [ 40],
       [ 70],
       [ 54],
       [150],
       [120],
       [110]])
```

In [19]:
```
KS = ks.reshape(10,1)
KS
```

Out[19]:
```
array([[401],
       [574],
       [874],
       [919],
       [459],
       [739],
       [653],
       [902],
       [746],
       [832]])
```

In [21]:
```
B = np.dot(np.linalg.inv(np.dot(ZP.T, ZP)),ZP.T@KS)
print('Коэффициент b_ =', B[0,0])
```

```
Коэффициент b_ = 5.889820420132688
```

2. **Посчитать коэффициент линейной регрессии при заработной плате (zp), используя градиентный спуск (без intercept).**

In [39]:
```
def mse_1(b, x=zp, y=ks, n=10):
    return np.sum((b*x-y)**2)/n
```

In [40]:
```
alpha = 1e-6
```

In [41]:
```
def deff(b, x=zp, y=ks, n=10):
    return np.sum((b*x-y)*x)*2/n
```

In [44]:
```
b = 0.5
for i in range(1000):
    b -= alpha*deff(b)
    if i%50 ==0:
        print("{0:<7}b = {1:<25}mse = {2}".format(i, b, mse_1(b)))
```

```
0      b = 0.6485068               mse = 434978.9132049683
50     b = 4.5934392207880705      mse = 79669.87043050732
100    b = 5.56917480801768        mse = 57933.280375525705
150    b = 5.810512259369029       mse = 56603.510258984556
200    b = 5.870204420018959       mse = 56522.15947864721
250    b = 5.884968618543086       mse = 56517.182716589494
300    b = 5.88862038044425        mse = 56516.87825533567
350    b = 5.889523603532249       mse = 56516.85962943917
400    b = 5.889747005815525       mse = 56516.85848997063
450    b = 5.88980226190988        mse = 56516.85842026184
500    b = 5.88981592889626        mse = 56516.858415997296
550    b = 5.889819309274822       mse = 56516.858415736395
600    b = 5.889820145374228       mse = 56516.85841572044
650    b = 5.889820352174209       mse = 56516.858415719485
700    b = 5.889820403323906       mse = 56516.8584157194
750    b = 5.8898204159752225      mse = 56516.85841571941
800    b = 5.889820419104385       mse = 56516.85841571941
850    b = 5.889820419878349       mse = 56516.8584157194
900    b = 5.889820420069781       mse = 56516.8584157194
950    b = 5.88982042011713        mse = 56516.85841571941
```

3. **Произвести вычисления как в пункте 2, но с вычислением intercept. Учесть, что изменение коэффициентов должно производиться**
на каждом шаге одновременно (то есть изменение одного коэффициента не должно влиять на изменение другого во время одной итерации).

In [23]:
```
def mse_2(b,a, x=zp, y=ks, n=10):
    return np.sum((b*x+a-y)**2)/n
```

In [24]:
```
def deff(a, b, x=zp, y=ks, n=10):
    return np.sum((b*x+a-y))*2/n, np.sum((b*x+a-y)*x)*2/n
```

In [25]:
```
alpha1 = 5e-5
alpha2 = 5e-3
```

In [38]:
```
b = 0.5
a = 100
for i in range(10000):
    da, db = deff(a, b)
    b -=alpha1*db
    a -=alpha2*da
    if i%500 ==0:
        print("{0:<7}.b = {1:<25}a = {2:<25}mse = {3}".format(i, b, a, mse_2(b,a)))
```

```
0      .b = 6.91134                a = 105.592               mse = 80122.753258527
500    .b = 3.3292992011090963     a = 348.05876072909433    mse = 8813.992878800733
1000   .b = 2.820964930012659      a = 416.9965602213816     mse = 6657.822609908893
1500   .b = 2.6772161532661265     a = 436.4910639605049     mse = 6485.400645625436
2000   .b = 2.6365663053391852     a = 442.00379687094807    mse = 6471.612618679562
2500   .b = 2.6250711804839875     a = 443.5627092981924     mse = 6470.510034749204
3000   .b = 2.6218205435892545     a = 444.0035447232523     mse = 6470.421864677572
3500   .b = 2.6209013156248333     a = 444.1282059092122     mse = 6470.4148140020425
4000   .b = 2.620641372703509      a = 444.1634580926841     mse = 6470.414250182321
4500   .b = 2.62056786501558       a = 444.173426844625      mse = 6470.414205095484
5000   .b = 2.6205470782207083     a = 444.1762458477464     mse = 6470.414201490034
5500   .b = 2.6205412000485984     a = 444.1770430166042     mse = 6470.41420120172
6000   .b = 2.6205395377958265     a = 444.17726844316866    mse = 6470.4142011786635
6500   .b = 2.6205390677374054     a = 444.1773321901851     mse = 6470.414201176818
7000   .b = 2.6205389348124286     a = 444.17735021681824    mse = 6470.414201176677
7500   .b = 2.620538897223375      a = 444.17735531446124    mse = 6470.414201176658
8000   .b = 2.6205388865937986     a = 444.1773567559922     mse = 6470.414201176661
8500   .b = 2.6205388835879235     a = 444.17735716363427    mse = 6470.414201176659
9000   .b = 2.620538882737911      a = 444.1773572789087     mse = 6470.414201176662
9500   .b = 2.6205388824975393     a = 444.1773573115068     mse = 6470.414201176657
```

4. Вычисление коэффициентов линейной регрессии с помощью библиотеки sklearn

a.) С intercept-ом

In [47]:
```
from sklearn.linear_model import LinearRegression
import pandas as pd

X=pd.DataFrame(zp)
y=pd.DataFrame(ks)
reg = LinearRegression().fit(X, y)

print('Коэффициент a =', reg.intercept_[0])
print('Коэффициент b =', reg.coef_[0,0])
```

```
Коэффициент a = 444.17735732435955
Коэффициент b = 2.6205388824027653
```

b.) Без intercept-a

In [84]:
```
from sklearn.linear_model import LinearRegression
import pandas as pd

X=pd.DataFrame(zp)
y=pd.DataFrame(ks)
reg = LinearRegression(fit_intercept=False).fit(X, y)

print('Коэффициент b =', reg.coef_[0,0])
```

```
Коэффициент b = 5.889820420132688
```