ДЗ по "Теория вероятностей и математическая статистика (семинары)"

Семинар 9. Линейная регрессия Логистическая регрессия

from statistics import mean

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

ROUND\_NUMBER = 5

# Задание 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.

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])

# модель линейной регрессии y = b1\*x

zp\_copy = zp.copy()

ks\_copy = ks.copy()

zp\_len = len(zp\_copy)

zp\_copy = zp\_copy.reshape(-1,1)

ks\_copy = ks\_copy.reshape(-1,1)

b = np.dot(np.linalg.inv(np.dot(zp\_copy.T,zp\_copy)), zp\_copy.T @ ks\_copy)

b\_not\_intercept = b[0][0]

print(f'Задание 1.1, линейная регрессия без intercept:   y = x \* {round(b[0][0],ROUND\_NUMBER)}')

# модель линейной регрессии y = b0 + b1\*x

zp\_copy = np.hstack([np.ones((zp\_len,1)),zp\_copy])

b = np.dot(np.linalg.inv(np.dot(zp\_copy.T,zp\_copy)), zp\_copy.T @ ks\_copy)

print(f'Задание 1.2, линейная регрессия с intercept:     y = {round(b[0][0],ROUND\_NUMBER)} + x \* {round(b[1][0],ROUND\_NUMBER)}')

# проверка

model = LinearRegression()

zp\_proverka = zp.copy()

# транспонируем одномерный массив

zp\_proverka=zp\_proverka.reshape(-1,1)

# подбираем коэффициенты

regres = model.fit(zp\_proverka,ks)

print(f'Задание 1, проверка вычислений:           intercep = {round(regres.intercept\_,ROUND\_NUMBER)}, b1 = {round(regres.coef\_[0],ROUND\_NUMBER)}')

# отрисовка графика

plt.scatter(zp,ks)

plt.plot(zp, b[0][0] + zp\*b[1][0], label='y = b0 + x\*b1')

plt.plot(zp, zp\*b\_not\_intercept, label='y = x\*b')

plt.title("Задание 1")

plt.xlabel('zp')

plt.ylabel('ks')

plt.legend()

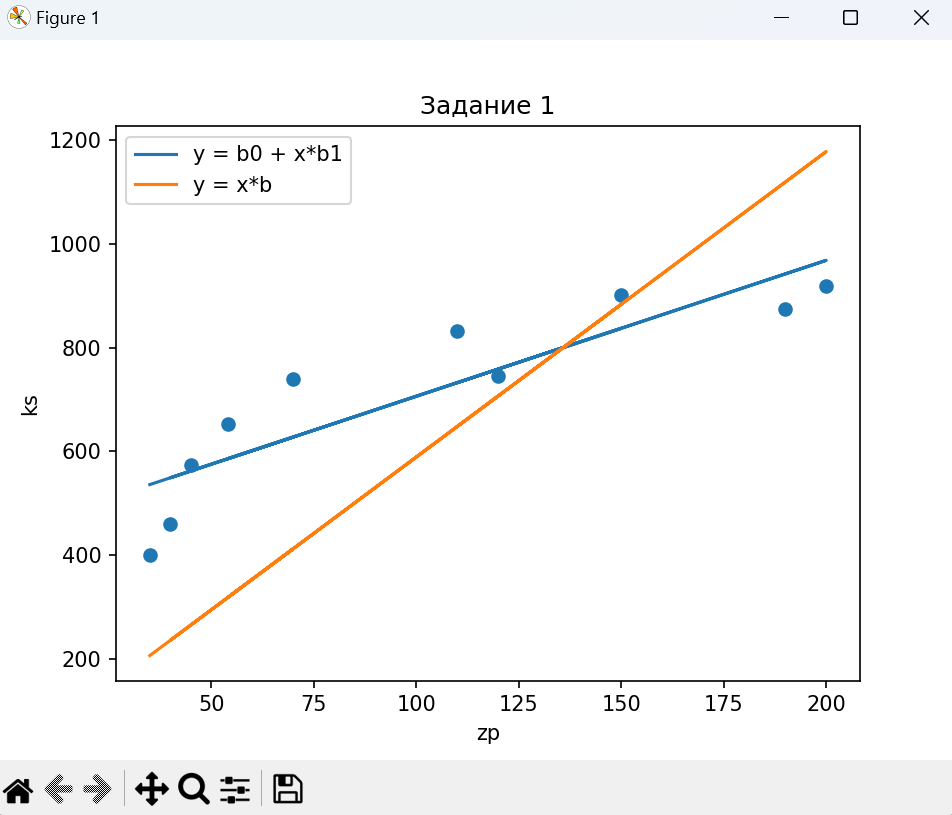
plt.show()

**Результат:**

Задание 1.1, линейная регрессия без intercept: y = x \* 5.88982

Задание 1.2, линейная регрессия с intercept: y = 444.17736 + x \* 2.62054

Задание 1, проверка вычислений: intercep = 444.17736, b1 = 2.62054



# Задание 2

# Посчитать коэффициент линейной регрессии при заработной плате (zp),

# используя градиентный спуск (без intercept).

def mse\_(B1, x, y, n):

    return np.sum((B1\*x-y)\*\*2)/n

# расчет коэффициента методом градиентного спуска

# y = b1\*x

alpha = 1e-6

b\_gradient = 0.1

n = len(zp)

for i in range(2000):

    b\_gradient -= alpha\*(2/n)\*np.sum((b\_gradient\*zp-ks)\*zp)

    if i % 100 == 0:

        print(f'Задание 2 (y=b\*x): iteration: {i}, b: {b\_gradient}, mse: {mse\_(b\_gradient, zp, ks, len(zp))}')

# отрисовка графика

plt.scatter(zp,ks)

# plt.plot(zp, b[0][0] + zp\*b[1][0], label='y = b0 + x\*b1')

plt.plot(zp, zp\*b\_gradient, label='y = x\*b')

plt.title("Задание 2")

plt.xlabel('zp')

plt.ylabel('ks')

plt.legend()

plt.show()

**Результат:**

Задание 2 (y=b\*x): iteration: 0, b: 0.25952808, mse: 493237.7212546963

Задание 2 (y=b\*x): iteration: 100, b: 5.54537842245223, mse: 58151.31823171113

Задание 2 (y=b\*x): iteration: 200, b: 5.868748638669329, mse: 56522.97550129376

Задание 2 (y=b\*x): iteration: 300, b: 5.888531320728348, mse: 56516.88130936019

Задание 2 (y=b\*x): iteration: 400, b: 5.8897415574471985, mse: 56516.85850140053

Задание 2 (y=b\*x): iteration: 500, b: 5.889815595583751, mse: 56516.858416040064

Задание 2 (y=b\*x): iteration: 600, b: 5.889820124983314, mse: 56516.85841572062

Задание 2 (y=b\*x): iteration: 700, b: 5.889820402076462, mse: 56516.85841571941

Задание 2 (y=b\*x): iteration: 800, b: 5.88982041902807, mse: 56516.8584157194

Задание 2 (y=b\*x): iteration: 900, b: 5.889820420065112, mse: 56516.85841571941

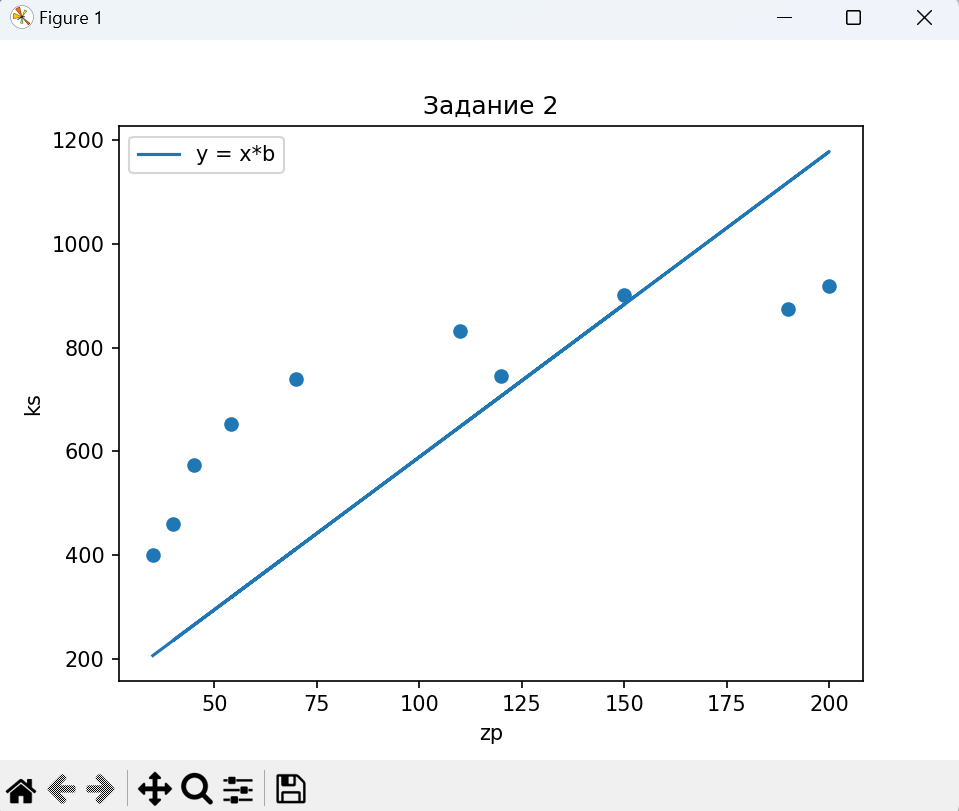
Задание 2 (y=b\*x): iteration: 1000, b: 5.8898204201285544, mse: 56516.85841571941

Задание 2 (y=b\*x): iteration: 1100, b: 5.889820420132437, mse: 56516.85841571943

Задание 2 (y=b\*x): iteration: 1200, b: 5.889820420132673, mse: 56516.85841571943

Задание 2 (y=b\*x): iteration: 1300, b: 5.889820420132673, mse: 56516.85841571943

Задание 2 (y=b\*x): iteration: 1400, b: 5.889820420132673, mse: 56516.85841571943



# Задание 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).

alpha = 1e-6

b\_gradient = 0.01

n = len(zp)

b0 = np.mean(ks)-(np.mean(zp\*ks)-np.mean(zp)\*np.mean(ks))/(np.mean(zp\*\*2)-np.mean(zp)\*\*2)\*np.mean(zp)

for i in range(2000):

    b\_gradient -= alpha\*np.sum((b0 + b\_gradient\*zp-ks)\*zp)

    if i % 100 == 0:

        print(f'Задание 3 (y=b0+b1\*x): iteration: {i}, b0: {b0}, b1: {b\_gradient}, mse: {mse\_(b\_gradient, zp, ks, len(zp))}')

# отрисовка графика

plt.scatter(zp,ks)

# plt.plot(zp, b[0][0] + zp\*b[1][0], label='y = b0 + x\*b1')

plt.plot(zp, b0+zp\*b\_gradient, label='y = b0 + x\*b')

plt.title("Задание 3")

plt.xlabel('zp')

plt.ylabel('ks')

plt.legend()

plt.show()

**Результат:**

Задание 3 (y=b0+b1\*x): iteration: 0, b0: 444.1773573243596, b1: 0.36964349967309934, mse: 476322.2799678742

Задание 3 (y=b0+b1\*x): iteration: 100, b0: 444.1773573243596, b1: 2.6205380604067248, mse: 203764.0130055573

Задание 3 (y=b0+b1\*x): iteration: 200, b0: 444.1773573243596, b1: 2.6205388824024647, mse: 203763.93896085554

Задание 3 (y=b0+b1\*x): iteration: 300, b0: 444.1773573243596, b1: 2.6205388824027636, mse: 203763.93896082864

Задание 3 (y=b0+b1\*x): iteration: 400, b0: 444.1773573243596, b1: 2.6205388824027636, mse: 203763.93896082864

Задание 3 (y=b0+b1\*x): iteration: 500, b0: 444.1773573243596, b1: 2.6205388824027636, mse: 203763.93896082864

Задание 3 (y=b0+b1\*x): iteration: 600, b0: 444.1773573243596, b1: 2.6205388824027636, mse: 203763.93896082864

