МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по практическому заданию №1
по дисциплине «Машинное обучение»

Студент гр. 6304 Доброхвалов М. О.

Преподаватель Жангиров Т. Р.

Санкт-Петербург
2020

Задание 1

Запись исходных данных в датафрейм

```
X = [69, 74, 68, 70, 72, 67, 66, 70, 76, 68, 72, 79, 74, 67, 66, 71,
74, 75, 75, 76]
Y = [153, 175, 155, 135, 172, 150, 115, 137, 200, 130, 140, 265, 185,
112, 140,  150, 165, 185, 210, 220]
df = pd.DataFrame({'X': X, 'Y': Y})
```

A. Среднее, медиана, мода величины X

```
df.X.describe()
df.X.mode()
```

среднее	медиана	мода
71.45	71.50	74

B. Дисперсия величины Y

```
df.Y.var()
```

Y.var = 1441.27

C. Графики гистограммы частот и плотности распределения для X (рис. 1 - 2).

```
df.X.plot.hist()
df.X.plot.density()
```

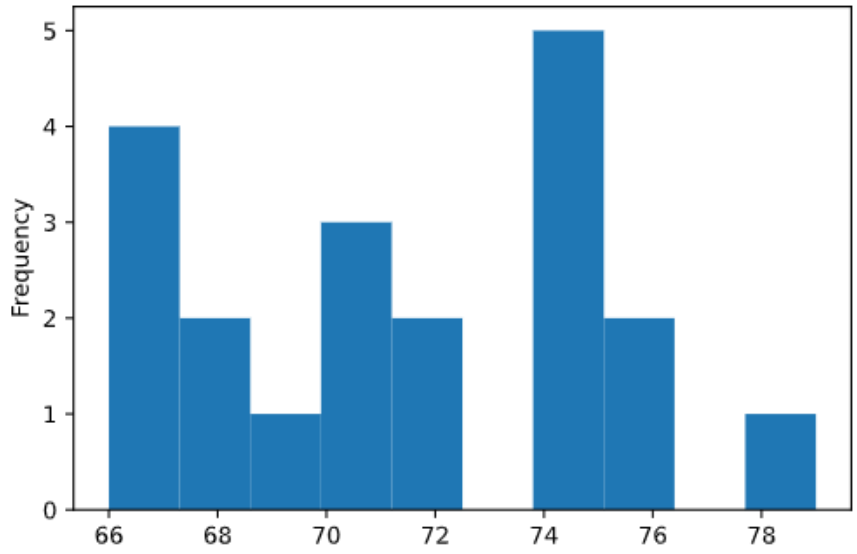

Рисунок 1 — гистограмма частот величины X.

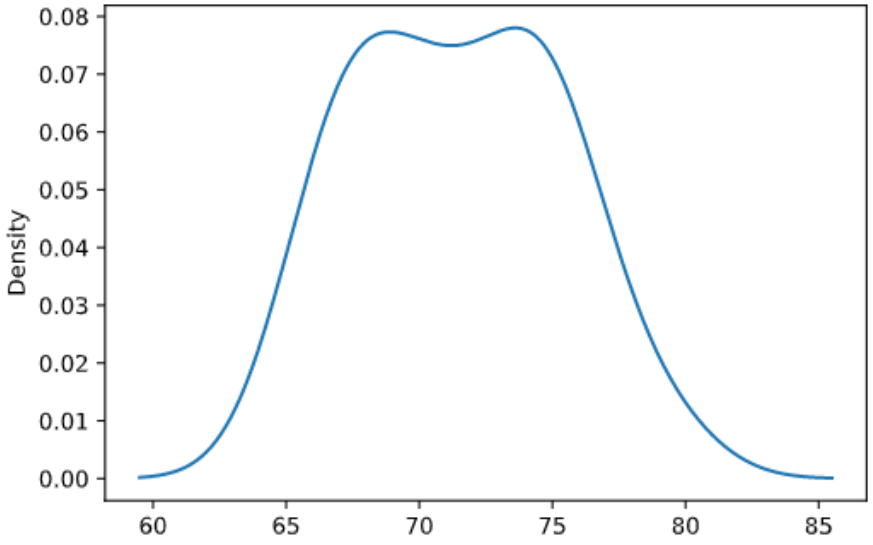

Рисунок 2 — Плотность распределения величины X.

D. Вероятность того, что возраст > 80.

```
scaler =
preprocessing.StandardScaler().fit(df.X.to_numpy().reshape((-1,1)))
threshold = 80
prob = t.sf(x=scaler.transform([[threshold]])[0][0], df=df.X.size-1)
```

Значение p-value Шапиро-Уилка теста = 0.35. Можно считать, что распределение соответствует нормальному закону. После этого выполняется стандартизация значений величины выборки, а также

значения для поверки (80). После чего с помощью T-теста считается вероятность.

```
P(X > 80) = 0.017
```

E. Двумерное математическое ожидание и ковариационная матрица

```
df.mean()
df.cov()
```

$$\mu = \begin{bmatrix} 71.45 \\ 164.70 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 14.58 & 128.88 \\ 128.88 & 1441.27 \end{bmatrix}$$

F. Корреляция

```
df.corr().loc['X','Y']
```

$r_{XY} = 3.82$

G. Диаграмма рассеяния, отображающая зависимость между возрастом и весом (рис. 3)

```
df.plot.scatter(x='X', y='Y')
```

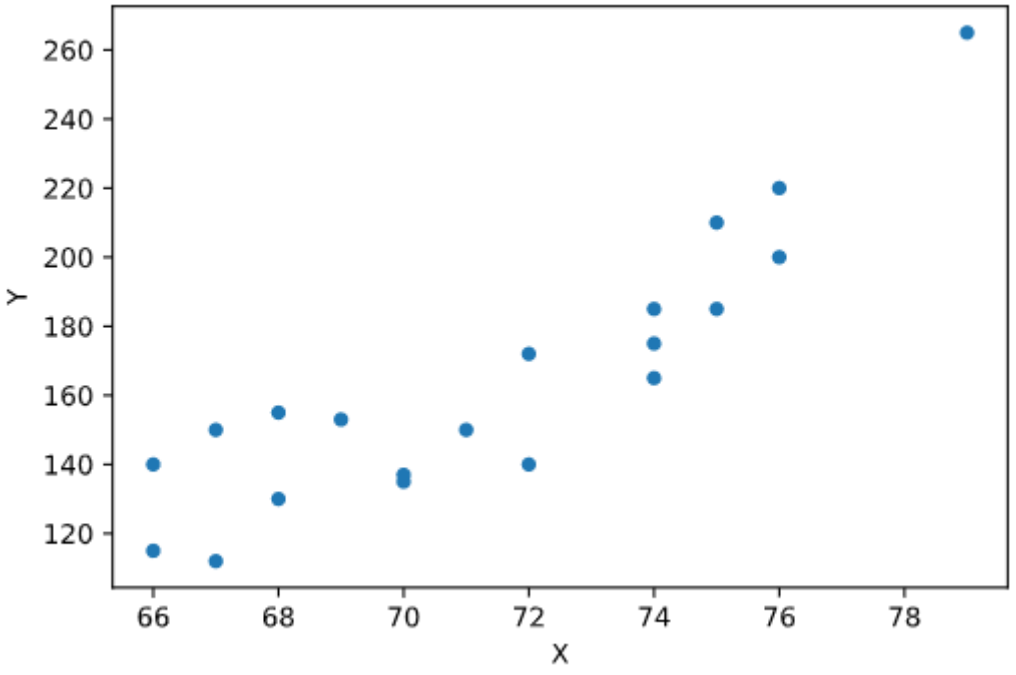

Рисунок 3 — Диаграмма рассеяния возраста и веса.

Задание 2

```
df_2 = pd.DataFrame({'X1': [17,11,11], 'X2': [17,9,8], 'X3':
[12,13,19]})
df_2_cov = df_2.cov()
np.linalg.det(df_2_cov)
```

$$\Sigma = \begin{bmatrix} 12.00 & 17.00 & -8.00 \\ 17.00 & 24.33 & -12.83 \\ -8.00 & -12.83 & 14.33 \end{bmatrix}$$

$det(\Sigma) = 0.0$

Задание 3

A.

```
mean_a, mean_b = 4, 8
std_a, std_b = 1, 2
tested_values = np.array([5,6,7])
for val in tested_values:
    val_std_a = (val - mean_a) / std_a
    val_std_b = (val - mean_b) / std_b
    print('Значение {}, отклонение от среднего N_a = {} std_a,
отклонение от среднего N_b = {} std_b => более вероятно
{}'.format(val, val_std_a, val_std_b, 'N_a' if abs(val_std_a) <
abs(val_std_b) else 'N_b'))
```

Для каждого значения в `tested_values` вычислено удаленность от среднего значения в СКО каждого из распределений. Чем большее значение по модулю, тем меньше вероятность. Результат:

```
Значение 5, отклонение от среднего N_a = 1.0 std_a, отклонение от
среднего N_b = -1.5 std_b => более вероятно N_a
Значение 6, отклонение от среднего N_a = 2.0 std_a, отклонение от
среднего N_b = -1.0 std_b => более вероятно N_b
Значение 7, отклонение от среднего N_a = 3.0 std_a, отклонение от
среднего N_b = -0.5 std_b => более вероятно N_b
```

B. Поиск значения, которое может быть сгенерировано обоими распределениями с равной вероятностью

```
from scipy.optimize import minimize_scalar

test_fun = lambda x: abs((x - mean_a) / std_a + (x - mean_b) / std_b)
res = minimize_scalar(test_fun)
res.x
```

Выполняется минимизация функции, которая считает сумму абсолютных значений удаленности от среднего значения каждого из распределений в СКО этого распределения. График минимизируемой функции (рис. 4)

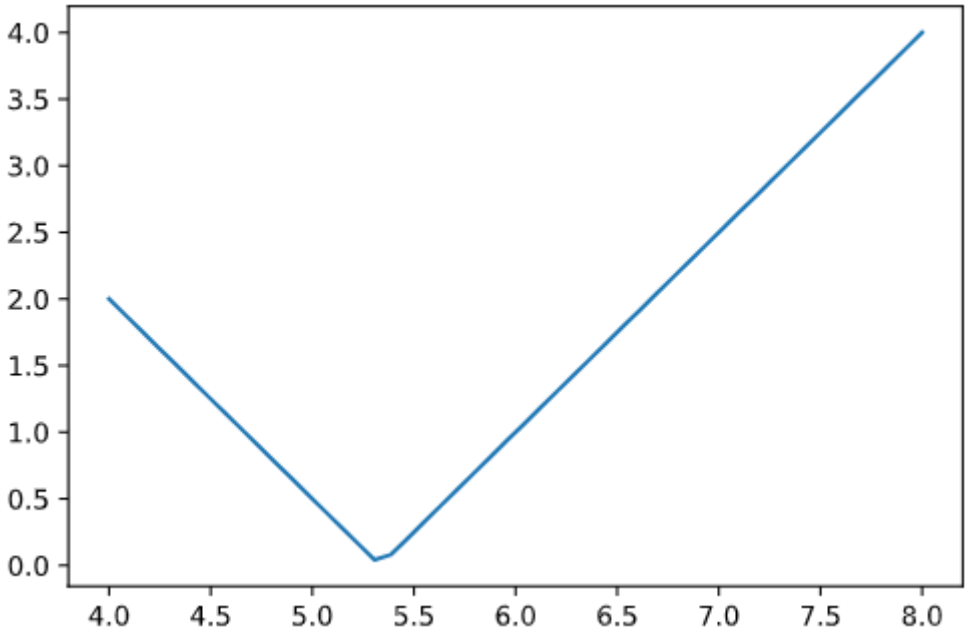

Рисунок 4 — зависимость суммы отклонений от значения аргумента