



МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий
Кафедра управления и интеллектуальных технологий

Отчет по лабораторной работе №1
По курсу «Интеллектуальный анализ данных»
«Построение и анализ регрессионных зависимостей»

Выполнили студенты: Михайловский Михаил, Озеров Сергей

Группа: А-03-21

Проверил: Назаров Николай Алексеевич

Москва 2023

Датасет ирисы Фишера

Подготовка датасета

Импортируем датасет с ирисами из *sklearn.datasets*. Столбцы *sl*, *sw*, *pl*, *pw* - это некоторые параметры различных ирисов, а *target* - это тип ириса.

```
In [ ]: from sklearn.datasets import load_iris
import numpy as np
import pandas as pd

iris = load_iris()
iris_pd=pd.DataFrame(data=np.c_[iris['data'], iris['target']], columns=iris['feature_names'], index=iris['target'])
iris_pd = iris_pd.rename(columns={'sepal length (cm)': 'sl', 'sepal width (cm)': 'sw', 'petal length (cm)': 'pl', 'petal width (cm)': 'pw'})

print(iris_pd)
print(iris_pd.describe())
```

| | sl | sw | pl | pw | target |
|-----|-----|-----|-----|-----|--------|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | 0.0 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | 0.0 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | 0.0 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | 0.0 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | 0.0 |
| .. | ... | ... | ... | ... | ... |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | 2.0 |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | 2.0 |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | 2.0 |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | 2.0 |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | 2.0 |

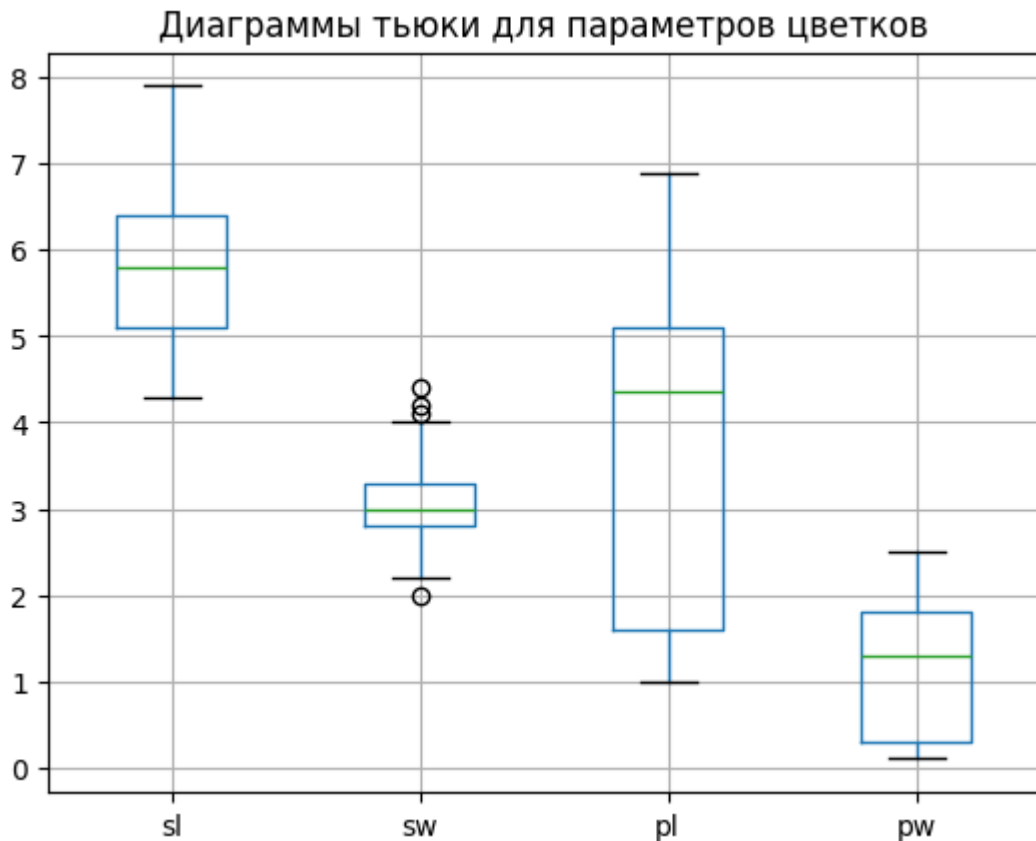
| | sl | sw | pl | pw | target |
|-------|------------|------------|------------|------------|------------|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 5.843333 | 3.057333 | 3.758000 | 1.199333 | 1.000000 |
| std | 0.828066 | 0.435866 | 1.765298 | 0.762238 | 0.819232 |
| min | 4.300000 | 2.000000 | 1.000000 | 0.100000 | 0.000000 |
| 25% | 5.100000 | 2.800000 | 1.600000 | 0.300000 | 0.000000 |
| 50% | 5.800000 | 3.000000 | 4.350000 | 1.300000 | 1.000000 |
| 75% | 6.400000 | 3.300000 | 5.100000 | 1.800000 | 2.000000 |
| max | 7.900000 | 4.400000 | 6.900000 | 2.500000 | 2.000000 |

1. Разведочный анализ данных

Построить диаграмму Тьюки, оценить диапазон изменения данных.

```
In [ ]: import matplotlib.pyplot as plt

iris_pd.iloc[:, :-1].boxplot()
plt.title('Диаграммы тьюки для параметров цветков')
plt.show()
```



Значения всех переменных сравнимы по величине, но находятся в нескольких разных диапазонах и имеют разный разброс. Также стоит отметить, что для значений sw имеем большое количество выбросов.

Проанализировать корреляционные зависимости между исследуемыми переменными. Необходимо построить тепловую карту.

Парная корреляция между i и j переменными можно найти по следующей формуле:

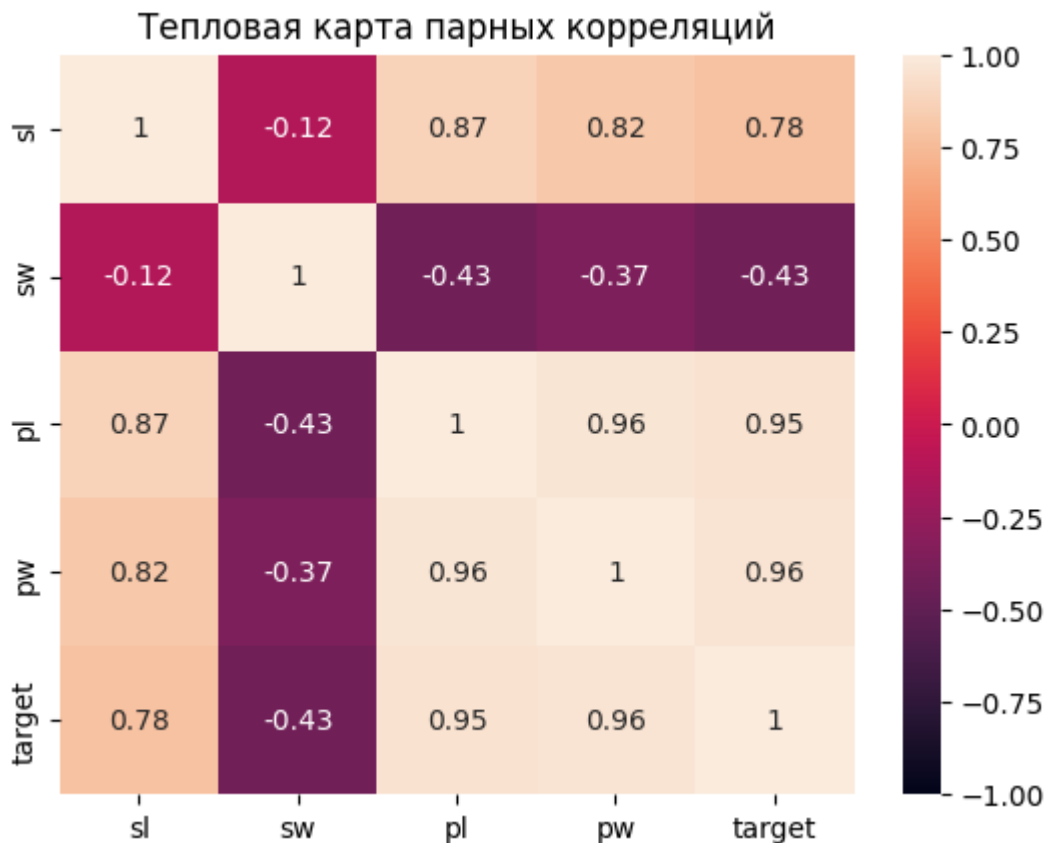
$$\hat{r}_{ik} = \frac{\hat{\sigma}_{ik}}{\hat{\sigma}_{ii}\hat{\sigma}_{kk}}$$

In []: `import seaborn as sns`

```
corr_matr = iris_pd.corr()
print(corr_matr)

sns.heatmap(corr_matr, annot=True, vmin=-1, vmax=1)
plt.title('Тепловая карта парных корреляций')
plt.show()
```

| | sl | sw | pl | pw | target |
|--------|-----------|-----------|-----------|-----------|-----------|
| sl | 1.000000 | -0.117570 | 0.871754 | 0.817941 | 0.782561 |
| sw | -0.117570 | 1.000000 | -0.428440 | -0.366126 | -0.426658 |
| pl | 0.871754 | -0.428440 | 1.000000 | 0.962865 | 0.949035 |
| pw | 0.817941 | -0.366126 | 0.962865 | 1.000000 | 0.956547 |
| target | 0.782561 | -0.426658 | 0.949035 | 0.956547 | 1.000000 |



Практически все переменные имеют высокие линейные связи друг с другом, кроме пар содержащих sw. Еще можно обратить внимание, что парные корреляции отдельных переменных со всеми остальными имеют похожие значения.

Рассчитать частные коэффициенты корреляции, сравнить их со значениями парных коэффициентов корреляции. Необходимо построить тепловую карту.

Частная корреляция между i и j переменными можно найти по следующей формуле:

$$\hat{r}_{ik|1,\dots,L} = -\frac{\hat{R}_{ik}}{\sqrt{\hat{R}_{ii}\hat{R}_{kk}}}, \text{ где } \hat{R}_{xy} \text{ -- алгебраическое дополнение корреляционной матрицы}$$

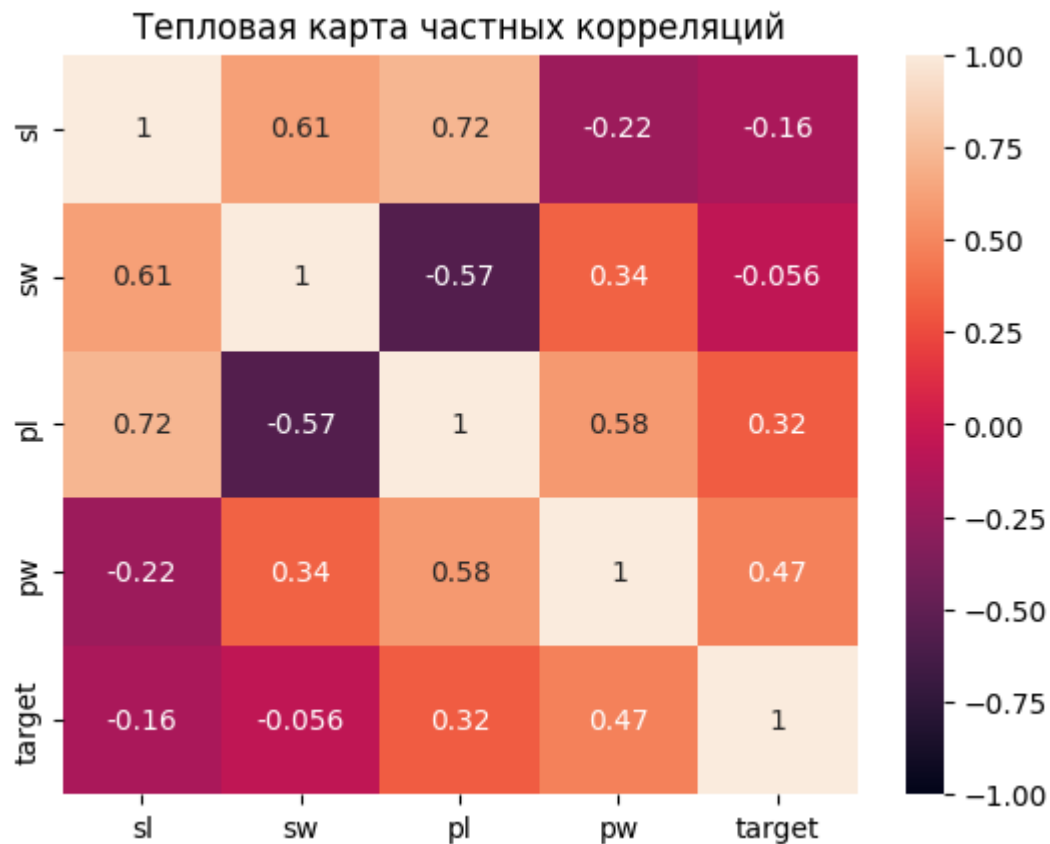


```
In [ ]: import pingouin

pcorr_matr = iris_pd.pcorr()
print(pcorr_matr)

sns.heatmap(pcorr_matr, annot=True, vmin=-1, vmax=1)
plt.title('Тепловая карта частных корреляций')
plt.show()
```

| | sl | sw | pl | pw | target |
|--------|-----------|-----------|-----------|-----------|-----------|
| sl | 1.000000 | 0.610735 | 0.723756 | -0.220409 | -0.159156 |
| sw | 0.610735 | 1.000000 | -0.565057 | 0.336649 | -0.055676 |
| pl | 0.723756 | -0.565057 | 1.000000 | 0.578470 | 0.316796 |
| pw | -0.220409 | 0.336649 | 0.578470 | 1.000000 | 0.472174 |
| target | -0.159156 | -0.055676 | 0.316796 | 0.472174 | 1.000000 |



По сравнению с матрицей парных корреляций частные корреляции отдельных переменных со всеми остальными значительно различаются, в том числе могут менять знак. Можно отметить, что тип цветка, судя по значениям коэффициентов частной корреляции, в большей мере определяется параметрами *pl* и *pw*.

Расчитаем коэффициент множественной корреляции для *target*. Это можно сделать по следующей формуле:

$$\hat{R}_{y|x^{(1)} \dots x^{(M-1)}} = \sqrt{1 - \frac{|\hat{R}|}{\hat{R}_{ii}}}$$

```
In [ ]: #Количество параметров
n = len(iris_pd)
k = len(iris_pd.iloc[0]) - 1

def minor(A, i, j):
    a_shape = A.shape[0]
    M = np.eye(a_shape - 1)
    M[:i, :j] = A[:i, :j]
    M[:i, j:] = A[:i, j+1:]
    M[i:, :j] = A[i+1:, :j]
    M[i:, j:] = A[i+1:, j+1:]

    return M
```

```
def alg_dop(A, i, j):
    M = minor(A, i, j)
    return (-1)**(i+j) * np.linalg.det(M)

R = corr_matr.values
#Множественная корреляция у к x-ам
R_y_x = np.sqrt(1 - np.linalg.det(R)/alg_dop(R, k, k))
print(f'Коэффициент множественной корреляции target: {R_y_x:.3f}')
```

Коэффициент множественной корреляции target: 0.965

По большому значению коэффициента множественной корреляции, можно сказать, что тип цветка имеет сильную линейную связь с его параметрами.

Проверить предположение о распределении признаков по нормальному закону критерием Колмогорова-Смирнова. Необходимо рассчитать значения статистик.

```
In [ ]: import scipy.stats

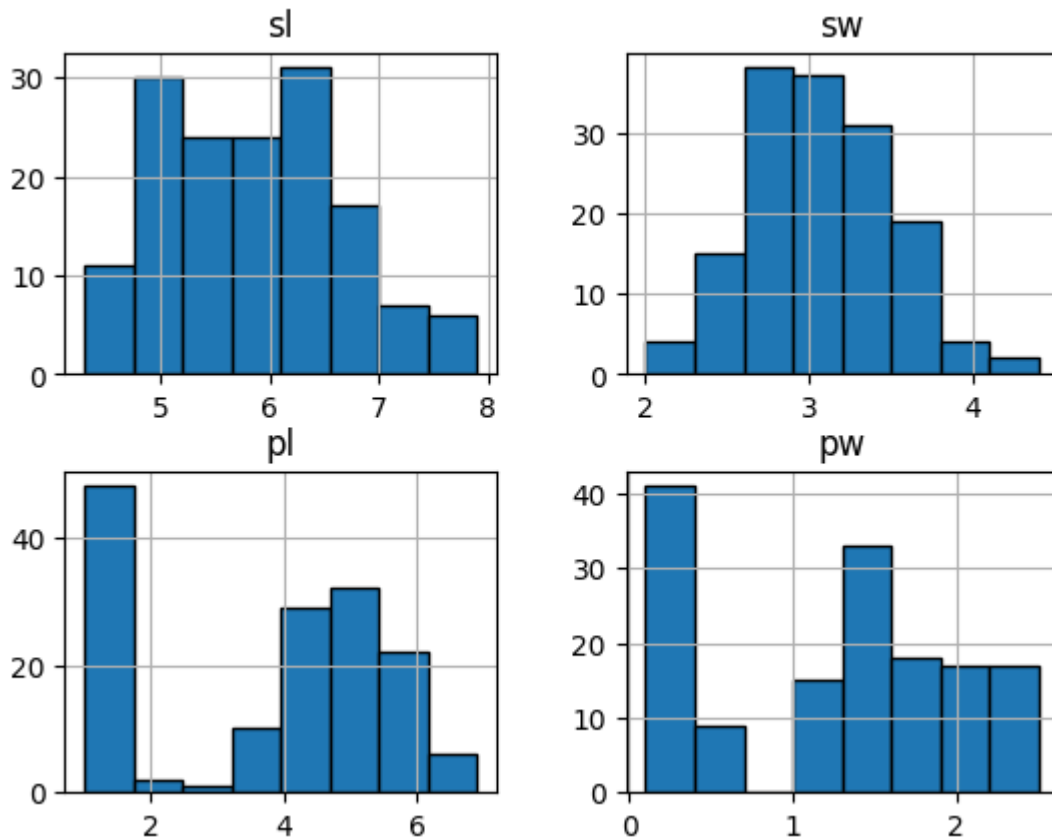
kstest_res = [ scipy.stats.kstest(iris_pd.iloc[:,i], 'norm') for i in range(len(
for i in range(len(kstest_res))):
    print(f'Значение статистики для {iris_pd.columns[i]}: {kstest_res[i].statist
```

Значение статистики для sl: 1.000, pvalue: 0.00e+00
 Значение статистики для sw: 0.979, pvalue: 1.93e-253
 Значение статистики для pl: 0.877, pvalue: 1.40e-136
 Значение статистики для pw: 0.546, pvalue: 1.88e-42
 Значение статистики для target: 0.508, pvalue: 2.10e-36

По результатам критерия Колмогорова-Смирнова данные столбцы противоречат предположению о нормальности закона распределения данных величин, поскольку значение pvalue в каждой из них практически нулевое.

Сделать выводы о степени однородности данных, силе зависимости между переменными, виде функции распределения, наиболее информативных переменных.

```
In [ ]: iris_pd.iloc[:, :-1].hist(bins = 8, edgecolor='black')
plt.show()
```



Построим гистограммы имеющихся данных. По ним видно, что оценки законов распределения данных выборок весьма различны и отличаются от нормального закона распределения, что подтвердил критерий Колмогорова-Смирнова. Заметим, что характер оценок дифференциального закона распределения для pl и pw визуально похож.

Судя по корреляционным коэффициентам, параметры ирисов имеют, как минимум, небольшую линейную связь. Есть связи как прямые так и обратные и их силы различны. Смотря на коэффициенты частной корреляции, можно предположить, что наиболее информативными для определения типа ириса являются параметры pl и pw , поскольку эти параметры имеют наибольшие оценки частной корреляции с типом цветка.

Построение регрессии

**Определить входные и выходные переменные.
Построить парную регрессию и множественную регрессию (Используем sklearn).**

При построении парной линейной регрессии мы рассматриваем модель вида: $\hat{y} = \hat{b}_0 + \hat{b}_1 \cdot x$, где коэффициенты \hat{b}_0 , \hat{b}_1 - это неизвестные, которые находятся по выборке. Модель множественной линейной регрессии имеет вид: $\hat{y} = \hat{b}_0 + \widehat{\vec{b}}^T \vec{x}$

Для оценки качества модели будем использовать коэффициенты детерминации и скорректированной детерминации:

$$R^2 = 1 - \frac{Q_{\text{ост}}}{Q_{\text{общ}}}, R_{\text{корр}}^2 = 1 - \frac{S_{\text{ост}}^2}{S_{\text{общ}}^2}$$

$$\text{Здесь } Q_{\text{общ}} = \sum_{j=1}^N (y_j - \bar{y}_j)^2, Q_{\text{ост}} = \sum_{j=1}^N (\hat{y}_j - y_j)^2, S_{\text{общ}}^2 = \frac{Q_{\text{общ}}}{N-1}, S_{\text{ост}}^2 = \frac{Q_{\text{ост}}}{N-k}$$

В нашем наборе данных прикладной задачей было бы определение типа цветка ирисов по его параметрам. Поэтому, в качестве входных переменных, можно взять параметры цветков sl, sw, pl, pw , а выходной переменной выбрать тип цветка.

```
In [ ]: from sklearn import linear_model
#Получение парной линейной регрессии
def pair_regress(data_pd, x_name, y_name):
    x = data_pd[x_name].values.reshape(-1, 1)
    y = data_pd[y_name].values.reshape(-1, 1)

    reg = linear_model.LinearRegression()
    reg.fit(x,y)

    return reg

#Построить графики результатов регрессии
def plot_regr_results(x, y, y_pred):
    residues = y - y_pred
    plt.figure(figsize=(12.8,6))
    plt.subplot(1, 2, 1)
    plt.grid()
    plt.plot(x, y, 'ko', label='iris data')
    plt.plot(x, y_pred, label='Предсказание')
    plt.title(f'Парная регрессия по {column}')
    plt.xlabel(column)
    plt.ylabel('target')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.grid()
    plt.plot(x, y_pred, label='Предсказание')
    plt.plot(x, residues, 'r+', label='Остатки')
    plt.title('Остатки')
    plt.xlabel(column)
    plt.show()

#Получение коэффициентов детерминации и скорректированной детерминации
def calc_r_squared(y, y_pred, k, return_Q = False):
    res = []
    n = len(y)
    delta_y = (y - y.mean()).reshape(-1)
    residues = (y - y_pred).reshape(-1)
    Q = np.dot(delta_y, delta_y)
    S = Q / (n-1)
    Q_res = np.dot(residues, residues)
    S_res = Q_res / (n-k)
    res.append(1 - Q_res/Q)
    res.append(1 - S_res/S)
    if return_Q:
        res.append(Q)
        res.append(Q_res)
```



```

    return res

reg = dict()
for column in list(iris_pd.columns)[: -1]:
    reg[column] = pair_regress(iris_pd, column, 'target')

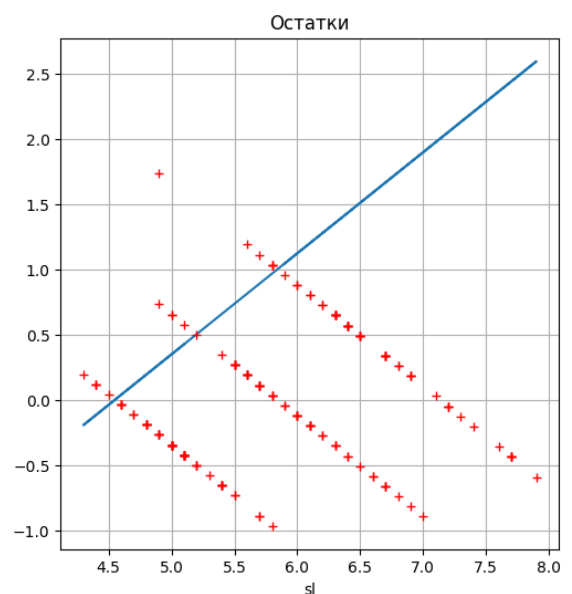
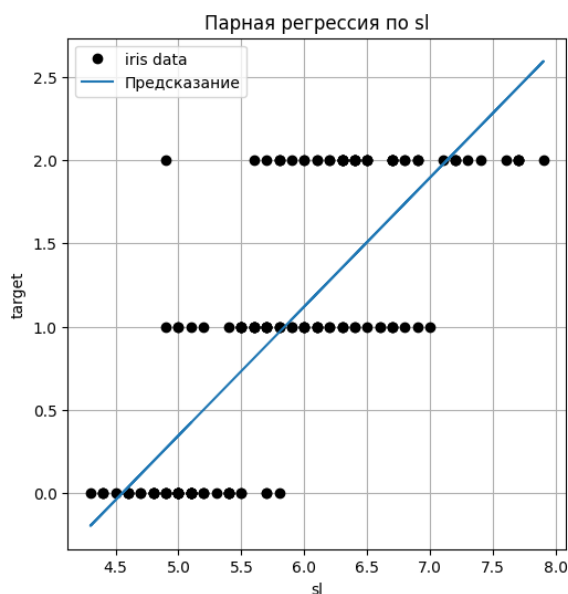
R_squared = dict()
R_adj_squared = dict()
Q = None
Q_res = dict()
RMSE = dict()
iris_y = iris_pd['target'].values.reshape(-1, 1)
delta_y = (iris_y - iris_y.mean()).reshape(-1)
for column in reg:
    iris_x = iris_pd[column].values.reshape(-1, 1)
    iris_y_pred = reg[column].predict(iris_x)
    plot_regr_results(iris_x, iris_y, iris_y_pred)

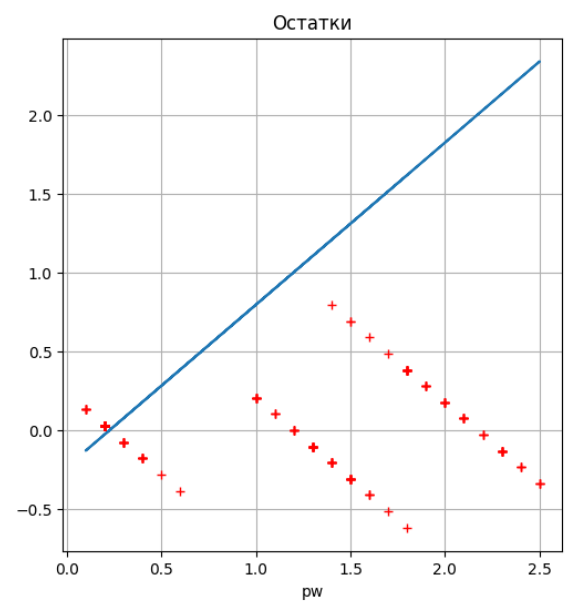
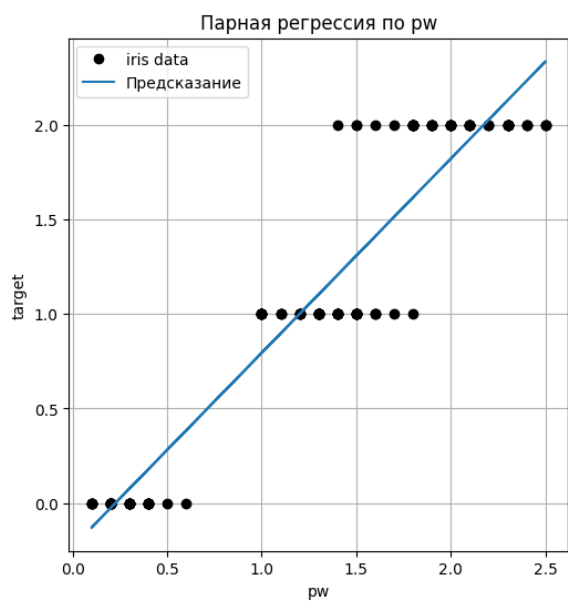
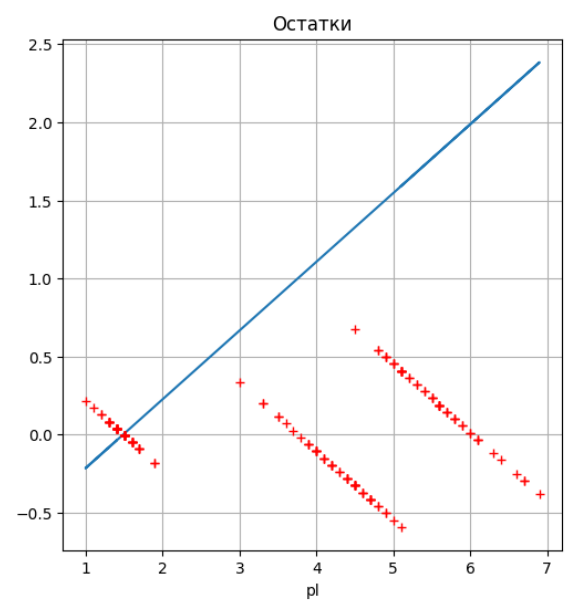
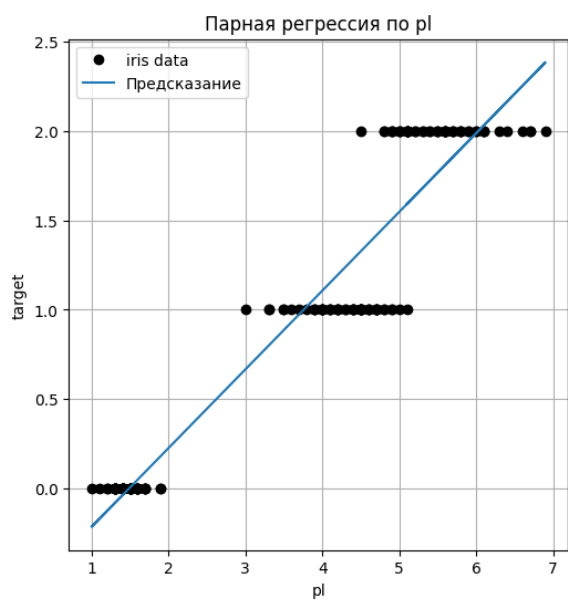
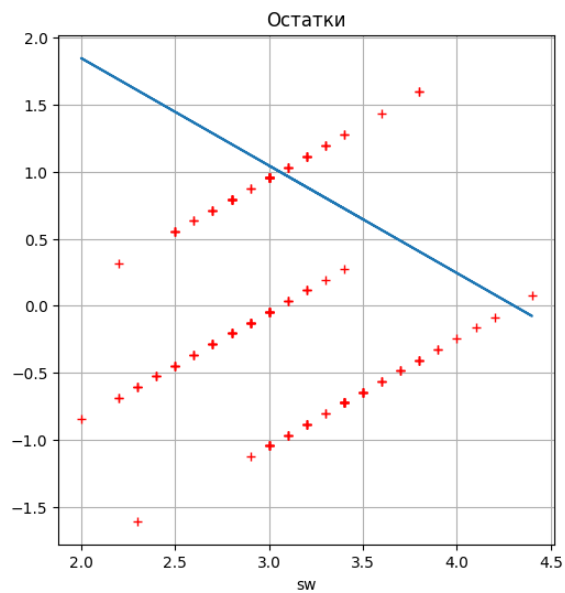
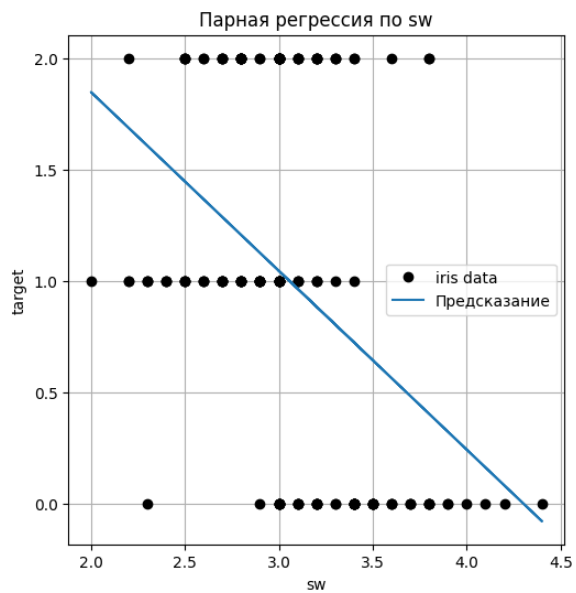
    R_sq, R_adj_sq, Q, Q_res[column] = calc_r_squared(iris_y, iris_y_pred, 1, reg[column])
    R_squared[column] = (R_sq)
    R_adj_squared[column] = (R_adj_sq)
    RMSE[column] = np.sqrt(((iris_y - iris_y_pred) ** 2).mean())

def print_num_dict(dic, round_to):
    for column, val in dic.items():
        print(f'{column}: {val:.{round_to}f}', end='\t')
    print()

print('Коэффициенты детерминации')
print_num_dict(R_squared, 3)
print('Коэффициенты скорректированной детерминации')
print_num_dict(R_adj_squared, 3)
print('Стандартная ошибка')
print_num_dict(RMSE, 3)
print(f'Q общее: {Q:.3f}')
print(f'Q остаточное')
print_num_dict(Q_res, 3)

```





Коэффициенты детерминации
 sl: 0.612 sw: 0.182 pl: 0.901 pw: 0.915
 Коэффициенты скорректированной детерминации
 sl: 0.612 sw: 0.182 pl: 0.901 pw: 0.915
 Стандартная ошибка
 sl: 0.508 sw: 0.738 pl: 0.257 pw: 0.238
 Q общее: 100.000
 Q остаточное
 sl: 38.760 sw: 81.796 pl: 9.933 pw: 8.502

Визуально полученные модели парных линейных регрессий подходят к данным и неплохо их оценивают. Модель построенная по второму параметру (sw) получилась некачественной, что видно и визуально, и по модулям значений остатков.

Смотря на коэффициенты детерминации, можно сказать, что модели построенные на первых двух параметрах хуже, чем модели на последних двух. В частности, вторая модель вовсе не пригодна, первая не очень хорошая, а последние две можно считать удачными.

Стоит отметить, что качество моделей получилось соответствующим порядку их рассчитанной корреляции со значением *target*.

Теперь построим модель множественной линейной регрессии.

```
In [ ]: m_reg = linear_model.LinearRegression()
iris_m_x = iris_pd.iloc[:, :-1].values.reshape(-1, k)

m_reg.fit(iris_m_x, iris_y)
reg['multiple'] = m_reg
iris_m_y_pred = m_reg.predict(iris_m_x)
m_residues = (iris_y - iris_m_y_pred).reshape(-1)
Q_res['multiple'] = np.dot(m_residues, m_residues)

m_R_squared, m_R_adj_squared = calc_r_squared(iris_y, iris_m_y_pred, k)
m_SE = np.std(iris_m_y_pred, ddof=k) / np.sqrt(n)

print(f'Детерминация: {m_R_squared:.3f}\nСкорректированная детерминация: {m_R_ad
print(f'Стандартная ошибка: {m_SE:.3f}')
```

Детерминация: 0.930
 Скорректированная детерминация: 0.929
 Стандартная ошибка: 0.065

Для модели множественной линейной регрессии получилось весьма большое значение скорректированного коэффициента детерминации. Оно больше, чем соответствующий коэффициент для любой из построенных ранее парных регрессий, поэтому можно предполагать, что данная модель лучше их.

Сравнить результаты парной и множественной регрессии.

| | Детерминация | Скорректированная детерминация | Стандартная ошибка |
|---------------------|--------------|--------------------------------|--------------------|
| Парная регрессия sl | 0.612 | 0.612 | 0.052 |

| | Детерминация | Скорректированная детерминация | Стандартная ошибка |
|-------------------------|--------------|--------------------------------|--------------------|
| Парная регрессия sw | 0.182 | 0.182 | 0.029 |
| Парная регрессия pl | 0.901 | 0.901 | 0.063 |
| Парная регрессия pw | 0.915 | 0.915 | 0.064 |
| Множественная регрессия | 0.930 | 0.929 | 0.065 |

Исходя из скорректированного коэффициента детерминации, можно считать адекватными последние 3 модели. Самая большая детерминация у модели, построенной на множественной линейной регрессии, поэтому можно считать, что она является наиболее адекватной.

3. Проверить гипотезу о нормальном распределении остатков. Рассчитать статистику Дурбина-Уотсона

Проверим гипотезу о нормальности распределения остатков с помощью критерия Колмогорова-Смирнова. Также построим QQ графики для наглядности.

```
In [ ]: import statsmodels.api as sm

pred = dict()
for column in iris_pd.columns[:-1]:
    pred[column] = reg[column].predict(iris_pd[column].values.reshape(-1, 1))

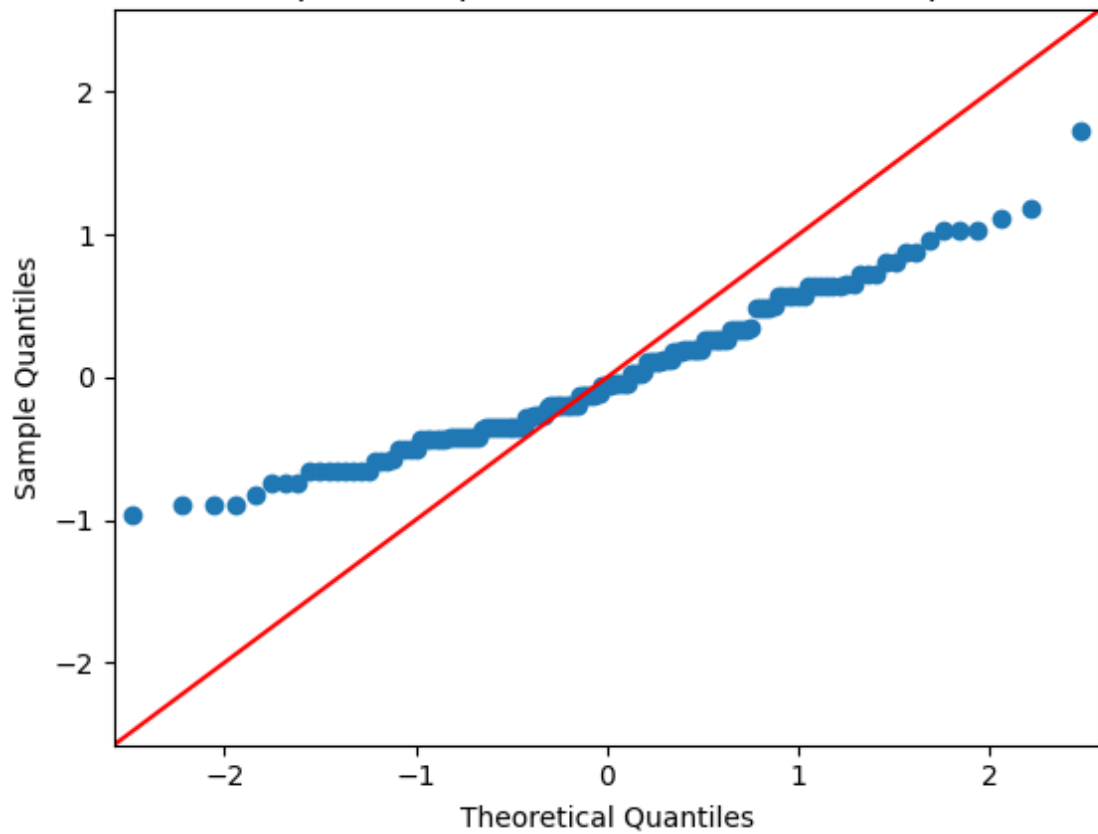
pred['multiple'] = reg['multiple'].predict(iris_m_x)

residues = dict()
for column in pred.keys():
    y_pred = pred[column]
    residues[column] = (iris_y - y_pred).reshape(-1)

    sm.qqplot(residues[column], line='45')
    ks_res = scipy.stats.kstest(residues[column], 'norm')
    plt.title(f'QQ график для {column}. \nТест Колмогорова-Смирнова - статистика')
    plt.show()
```

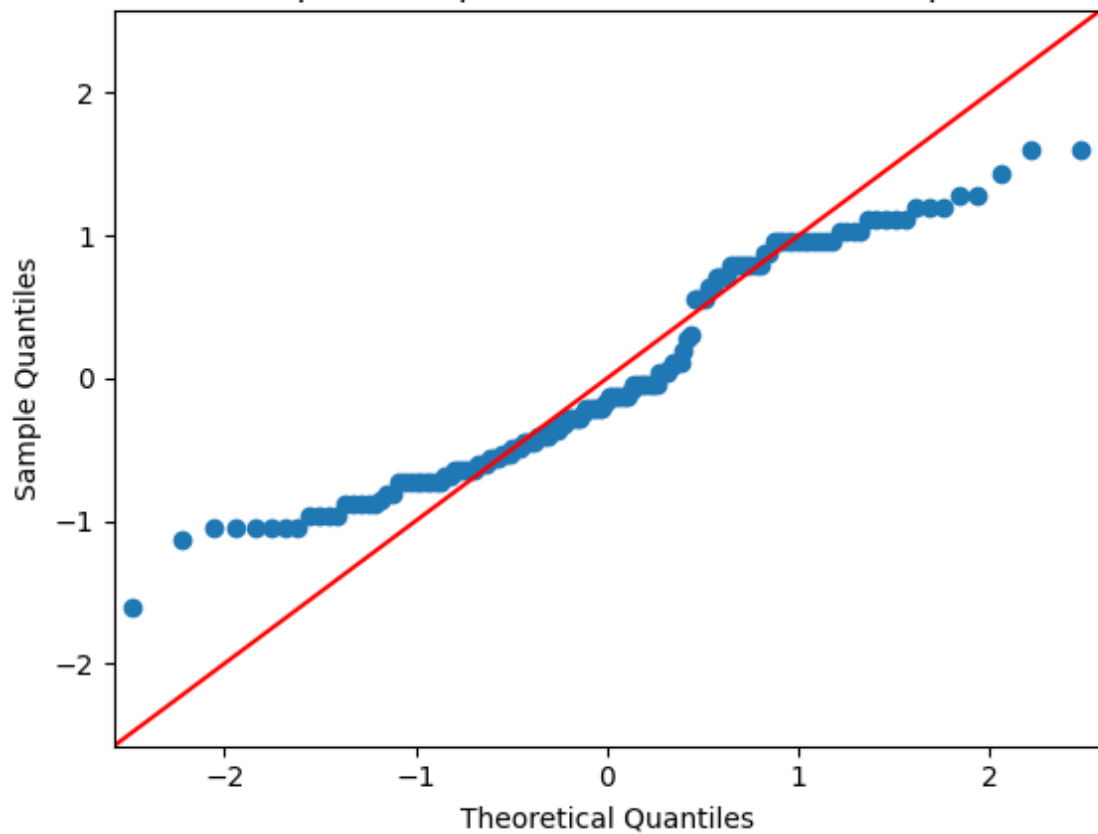
QQ график для sl.

Тест Колмогорова-Смирнова - статистика: 0.200, pvalue: 0.000



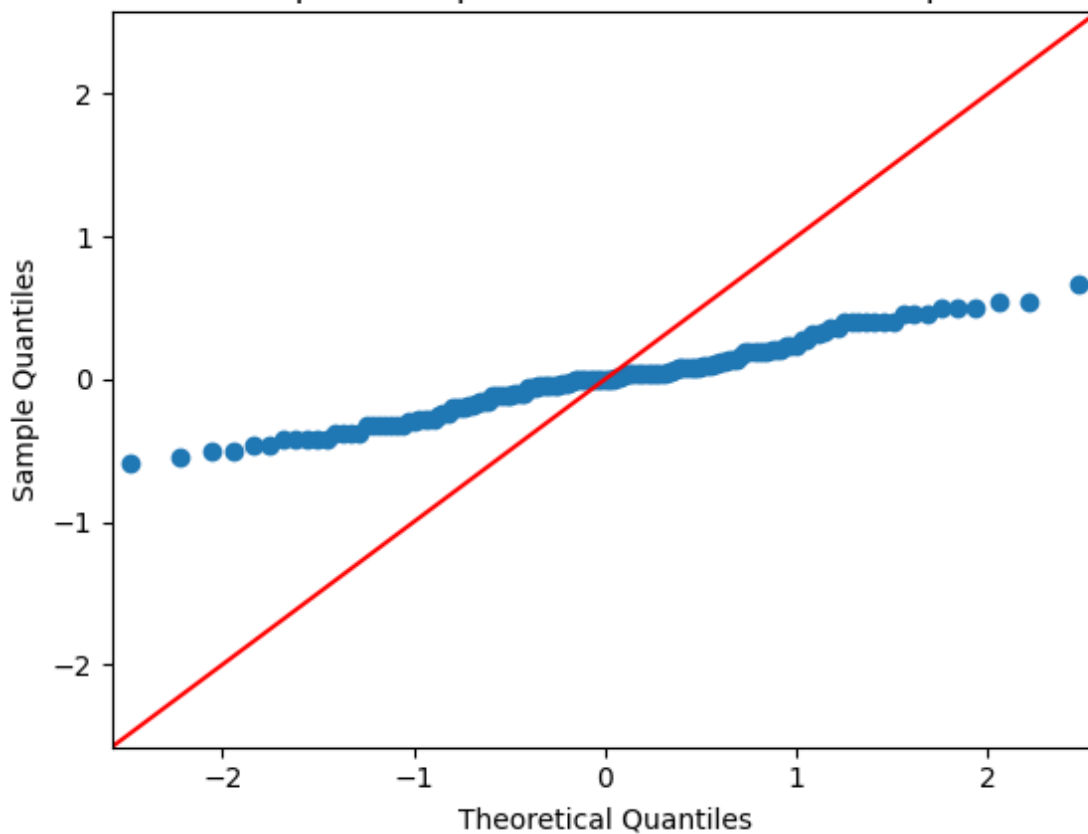
QQ график для sw.

Тест Колмогорова-Смирнова - статистика: 0.134, pvalue: 0.008



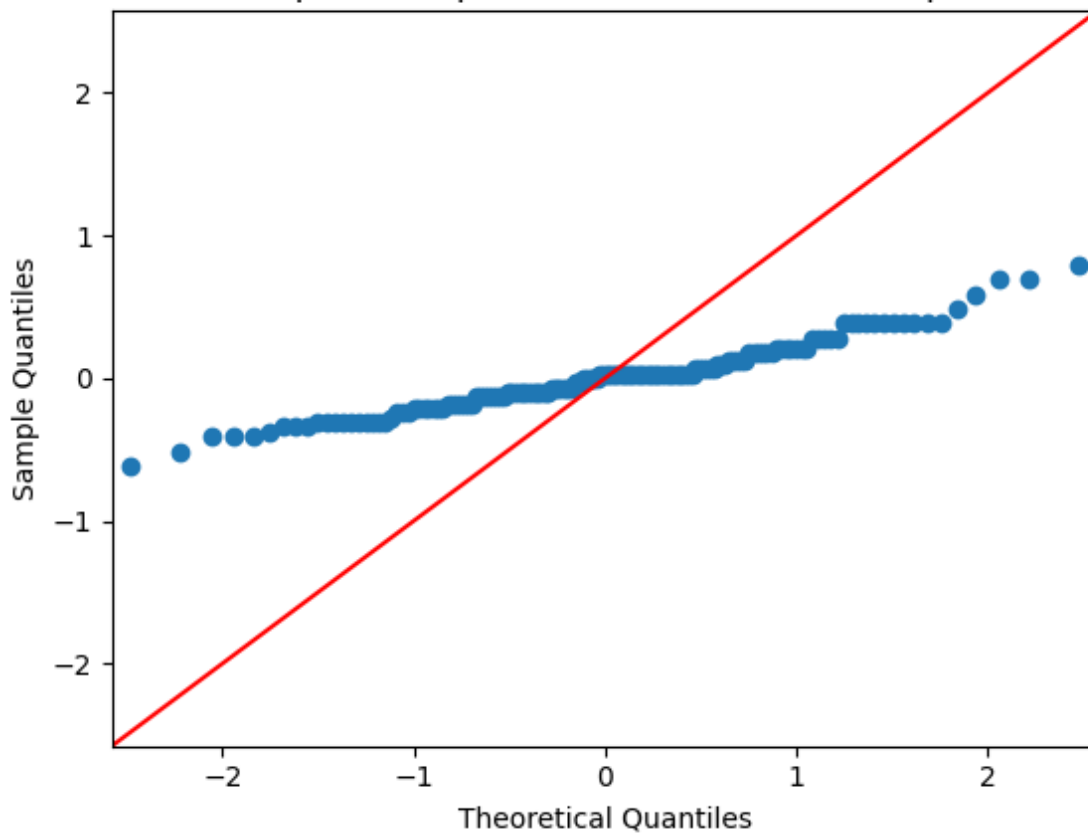
QQ график для p_l .

Тест Колмогорова-Смирнова - статистика: 0.299, pvalue: 0.000



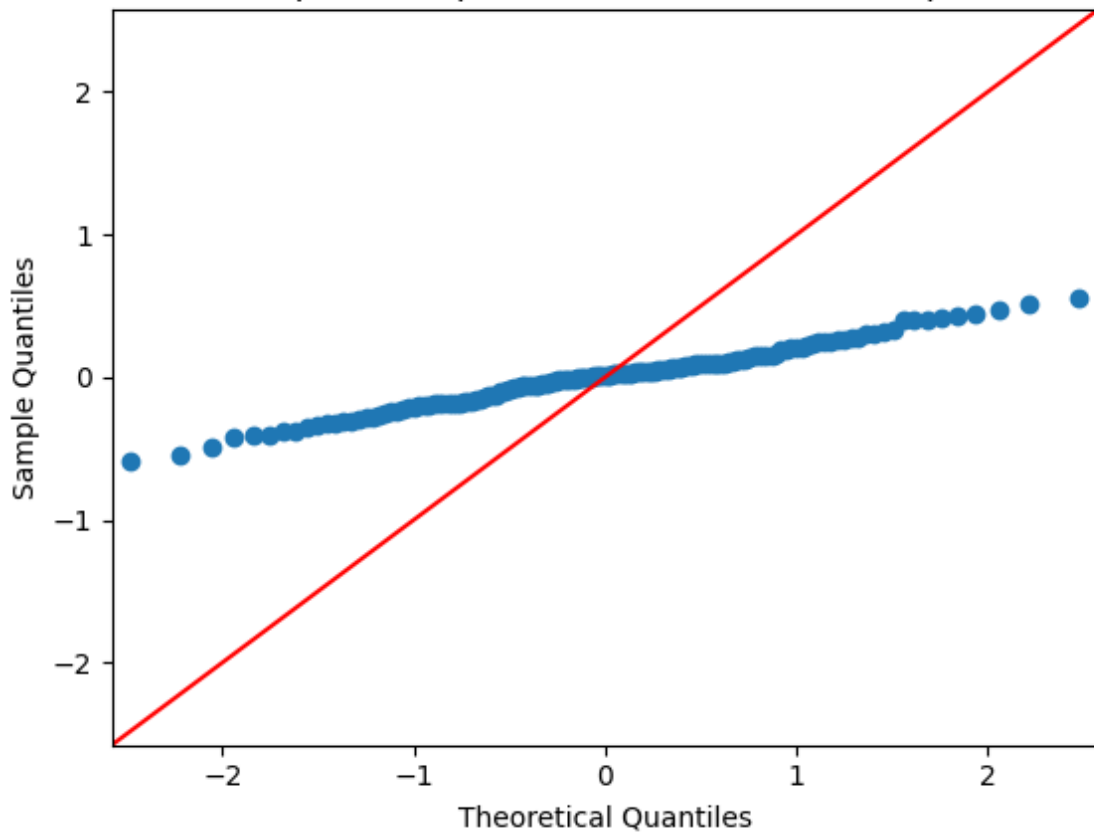
QQ график для p_w .

Тест Колмогорова-Смирнова - статистика: 0.328, pvalue: 0.000



QQ график для multiple.

Тест Колмогорова-Смирнова - статистика: 0.319, pvalue: 0.000



Проверим гипотезу об отсутствии автокорреляции помехи ε по критерию Дарбина-Уотсона. В нём качестве статистики используется:

$$\gamma = \frac{\sum_{i=2}^N (\varepsilon_i - \varepsilon_{i-1})}{Q_{\text{общ}}} \approx 2(1 - \rho) \in (0, 4)$$

Зона допустимых и критических значений разделяется двумя симметричными относительно центра области определения статистики интервалами. В них критерий не даёт информации об автокорреляции. Если значение статистики находится между этими интервалами недопустимости, то выборка не противоречит гипотезе об отсутствии автокорреляции помехи. Если значение статистики находится на краях интервала $(0, 4)$, то следует отвергнуть гипотезу об отсутствии автокорреляции.

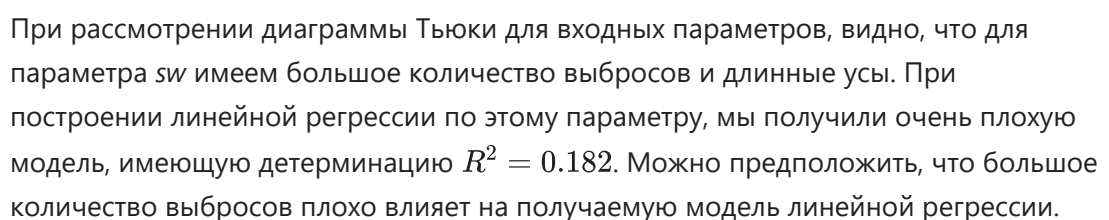
```
In [ ]: def DW(res, Qres):
    result = 0
    for i in range(len(res)-1):
        result += (res[i+1]-res[i]) ** 2
    return result/Qres

DW_val = dict()
for column in residues:
    if column != 'multiple':
        S_res = Q_res[column]/(n-1)
    else:
        S_res = Q_res[column]/(n-k)

    DW_val[column] = DW(residues[column], Q_res[column])
```

Значение статистики Дарбина-Уотсона
sl: 1.228 sw: 0.282 pl: 1.073 pw: 1.510 multiple: 1.077
Области неопределённости теста: m = 1: (1.720, 1.746) | (2.254, 2.280)
m = 4: (1.679, 1.788) | (2.212, 2.321)

4. Проанализировать наличие выбросов. Сделать вывод о наличии (отсутствии) выбросов (с учетом результатов, полученных в предыдущих пунктах).



Сравнивая оставшиеся модели, видно, что для sl длина услов в отношении к самому ящику больше, чем для pl и pw . Здесь можно предположить, что модели, построенные по параметрам, имеющим "длинноусые" диаграммы Тьюки, хуже, чем по параметрам имеющим "короткоусые" диаграммы Тьюки.

5. Провести предсказание зависимой переменной по вычисленному уравнению множественной регрессии (Predict Dependent Variable), оценить точность. Для этого делим выборку на тестовую и обучающую. После чего рассчитываем СКО, коэффициент детерминации и строим график ошибок.

```
In [ ]: from sklearn.model_selection import train_test_split

train, test = train_test_split(iris_pd, test_size=0.4)

column_train = ['sl', 'sw', 'pl', 'pw']
k_reg = len(column_train)
test_y = test['target'].values.reshape(-1, 1)
train_y = train['target'].values.reshape(-1, 1)
train_x = train[column_train].values.reshape(-1, k_reg)
test_x = test[column_train].values.reshape(-1, k_reg)

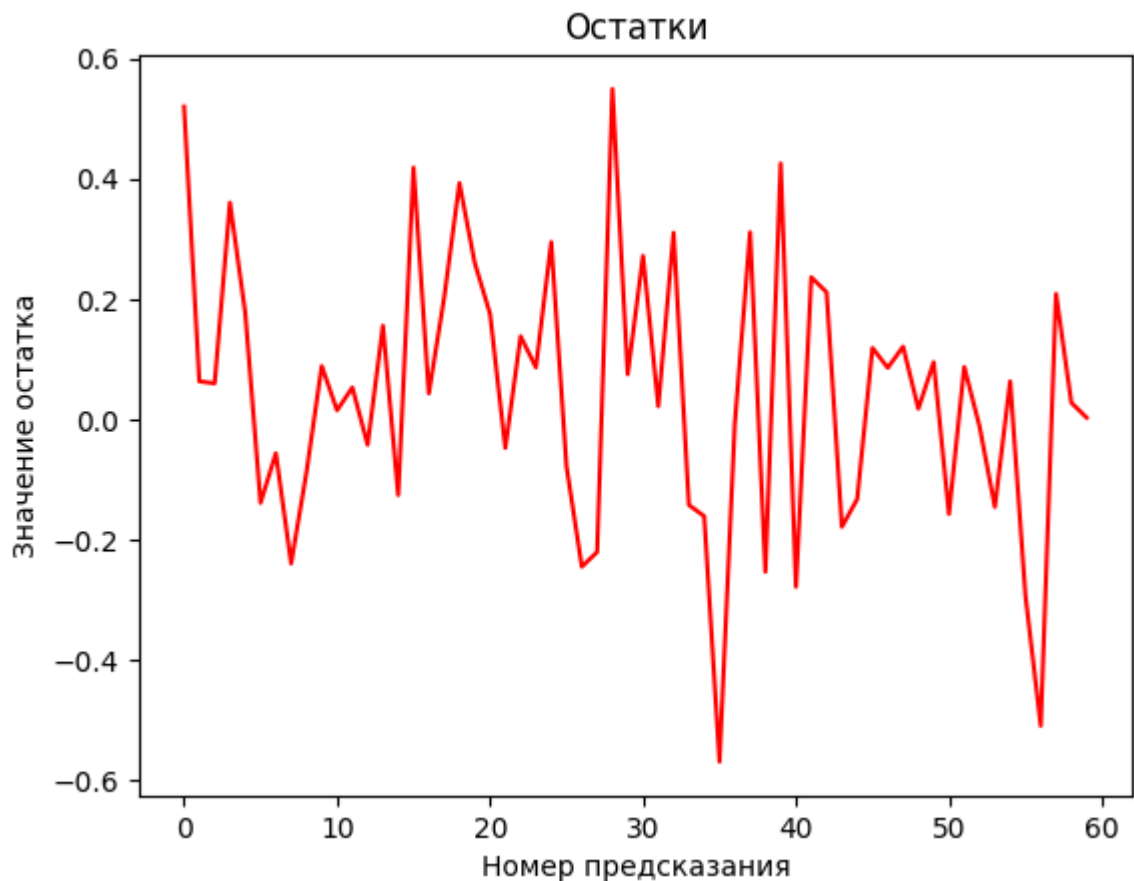
regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)

test_pred = regr.predict(test_x)
test_residues = (test_y - test_pred).reshape(-1)
test_MAE = (test_residues ** 2).mean()
test_Q = test_y.var()*len(test_y)
test_Qres = np.dot(test_residues, test_residues)
test_R_squared = 1 - (test_Qres/(n-k))/(test_Q/(n-1))

print(f'MAE: {test_MAE:.3f}, R_squared={test_R_squared:.3f}')

plt.title(f'Остатки')
plt.plot(test_residues, 'r')
plt.xlabel('Номер предсказания')
plt.ylabel('Значение остатка')
plt.show()
```

MAE: 0.053, R_squared=0.925



На этот раз модель оценивалась по её качеству предсказания на выборке, которая не участвовала в обучении. Судя по коэффициенту детерминации модель получилась весьма хорошей. Учитывая то, что в этот раз модель обучалась на меньшей части данных, её коэффициент детерминации практически не изменился и остался больше 0.9.

6. Сформулировать предложения по улучшению регрессионной модели.

Для улучшения модели можно сделать следующие вещи:

- Провести анализ выбросов параметра sw . Модель парной регрессии построенная по нему получилась плохой, возможно он оказывает не лучшее влияние на множественную регрессию;
- Можно попробовать увеличить обучающую выборку.

Датасет Бейсбол

Подготовка датасета

Обозначим роли следующими метками: Pitcher - 1, Catcher - 2, Baseman - 3, Outfielder - 4, Relief_P - 5

```
In [ ]: import numpy as np
import pandas as pd

data_pd=pd.read_csv('Baseball_formatted.csv', sep=';', decimal=',')

print(data_pd)
print(data_pd.describe())
```

| | Position | Weight | Height | Age |
|-----|----------|--------|--------|-------|
| 0 | 2 | 74 | 180 | 22.99 |
| 1 | 2 | 74 | 215 | 34.69 |
| 2 | 2 | 72 | 210 | 30.78 |
| 3 | 3 | 72 | 210 | 35.43 |
| 4 | 3 | 73 | 188 | 35.71 |
| .. | ... | ... | ... | ... |
| 110 | 4 | 76 | 192 | 25.37 |
| 111 | 4 | 74 | 235 | 29.57 |
| 112 | 4 | 72 | 185 | 27.33 |
| 113 | 3 | 72 | 194 | 26.26 |
| 114 | 3 | 74 | 220 | 27.56 |

[115 rows x 4 columns]

| | Position | Weight | Height | Age |
|-------|------------|------------|------------|------------|
| count | 115.000000 | 115.000000 | 115.000000 | 115.000000 |
| mean | 3.200000 | 73.895652 | 204.191304 | 28.552174 |
| std | 1.440029 | 2.157703 | 20.234977 | 4.128190 |
| min | 1.000000 | 68.000000 | 160.000000 | 22.020000 |
| 25% | 2.000000 | 73.000000 | 188.000000 | 25.110000 |
| 50% | 3.000000 | 74.000000 | 205.000000 | 27.940000 |
| 75% | 4.000000 | 75.000000 | 220.000000 | 31.475000 |
| max | 5.000000 | 81.000000 | 270.000000 | 37.740000 |

Разведочный анализ данных

```
In [ ]: import matplotlib.pyplot as plt

data_columns = tuple(data_pd.columns)
print(data_columns)

for i in range(len(data_pd.iloc[0]) - 1):
    plt.figure(figsize=(4.0,3))
    plt.title(f'Диаграмма тьюки для {data_columns[i+1]}')
    data_pd[[data_columns[i+1]]].boxplot()
    plt.show()

plt.show()
```

('Position', 'Weight', 'Height', 'Age')

Диаграмма тьюки для Weight

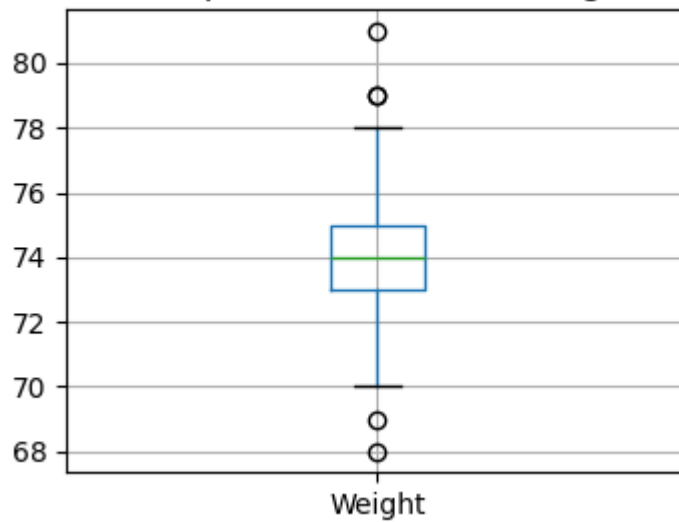


Диаграмма тьюки для Height

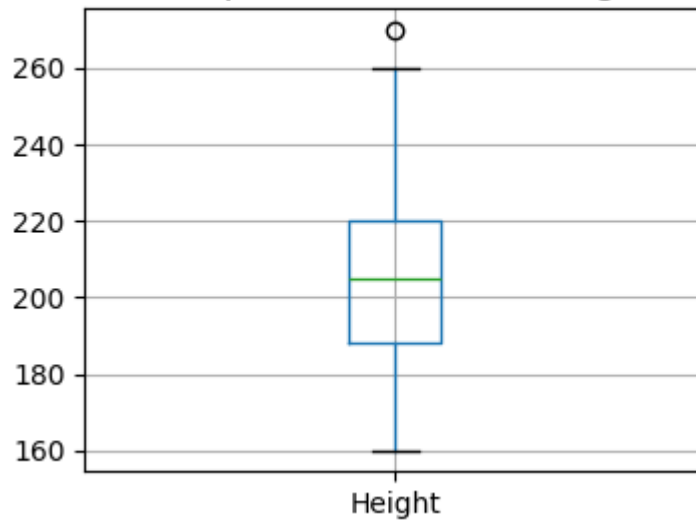
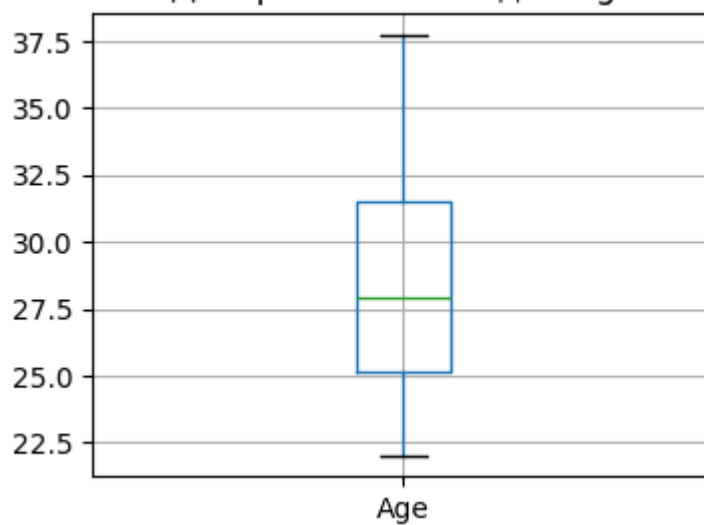


Диаграмма тьюки для Age



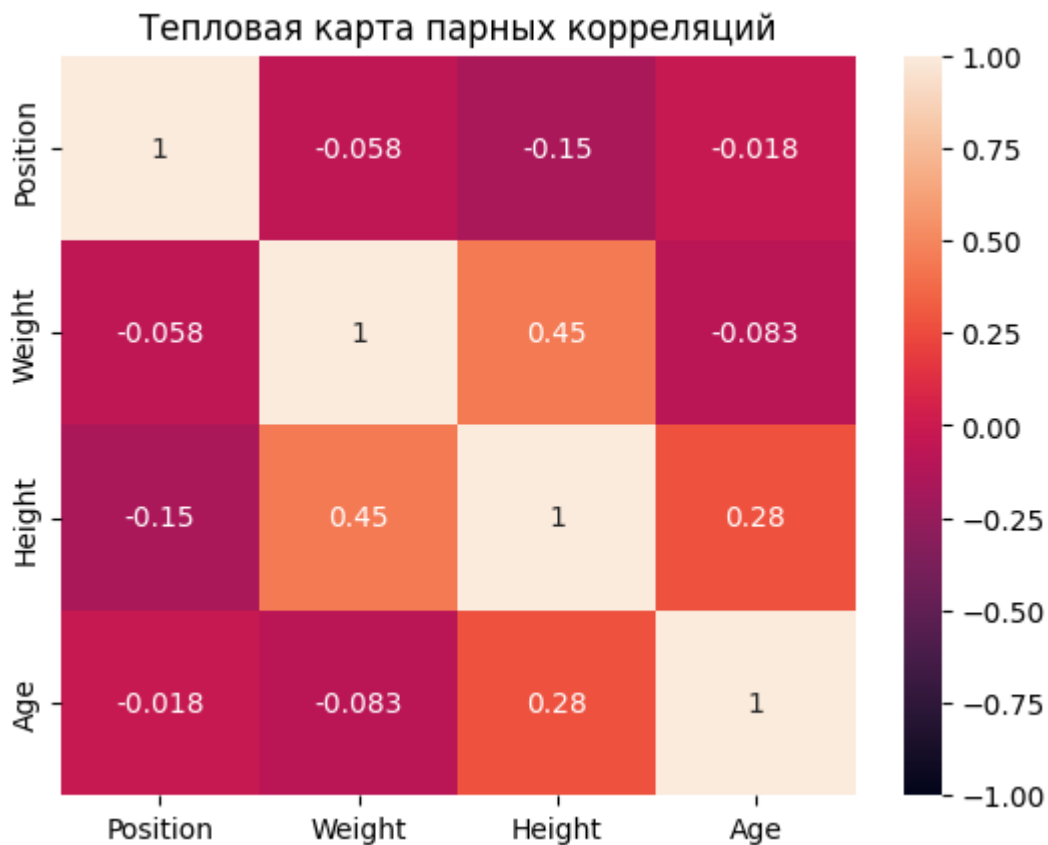
Диаграммы Тьюки для веса и роста имеют длинные усы. Также имеются выбросы. Для возраста диаграмма имеет усы не такие длинные.

```
In [ ]: import seaborn as sns

corr_matr = data_pd.corr()
print(corr_matr)

sns.heatmap(corr_matr, annot=True, vmin=-1, vmax=1)
plt.title('Тепловая карта парных корреляций')
plt.show()
```

```
      Position  Weight  Height  Age
Position  1.000000 -0.058157 -0.152747 -0.018371
Weight    -0.058157  1.000000  0.445074 -0.082647
Height    -0.152747  0.445074  1.000000  0.277046
Age        -0.018371 -0.082647  0.277046  1.000000
```



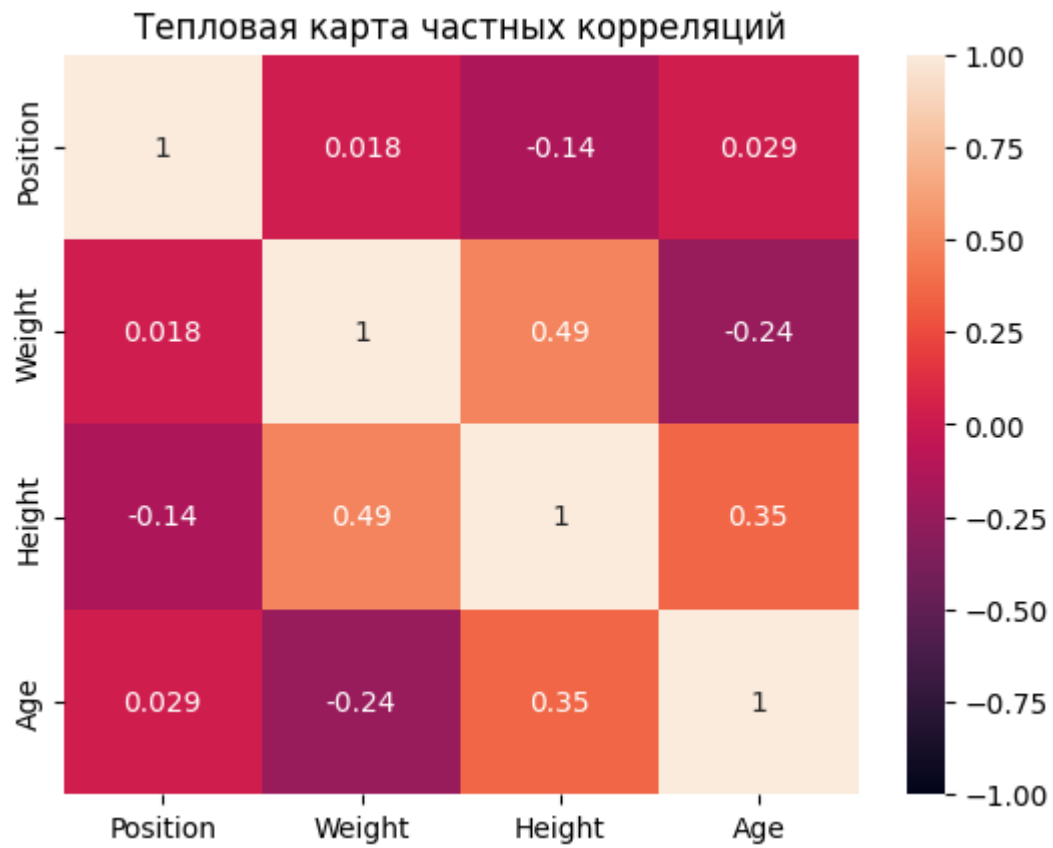
По парным корреляциям видно, что вес коррелирует с ростом, и рост коррелирует с возрастом. Это правда и это очевидно. Однако, можно заметить, что значение позиции имеет малую корреляцию с данными параметрами. Это влияет на качество модели линейной регрессии, которую можно построить по ним.

```
In [ ]: import pingouin

pcorr_matr = data_pd.pcorr()
print(pcorr_matr)

sns.heatmap(pcorr_matr, annot=True, vmin=-1, vmax=1)
plt.title('Тепловая карта частных корреляций')
plt.show()
```

| | Position | Weight | Height | Age |
|----------|-----------|-----------|-----------|-----------|
| Position | 1.000000 | 0.017659 | -0.142878 | 0.028713 |
| Weight | 0.017659 | 1.000000 | 0.486141 | -0.239729 |
| Height | -0.142878 | 0.486141 | 1.000000 | 0.352009 |
| Age | 0.028713 | -0.239729 | 0.352009 | 1.000000 |



```
In [ ]: #Количество параметров
n = len(data_pd)
k = len(data_pd.iloc[0]) - 1

def minor(A, i, j):
    a_shape = A.shape[0]
    M = np.eye(a_shape - 1)
    M[:,j] = A[:,j]
    M[:,j:] = A[:,j+1:]
    M[i:,j] = A[i+1:,j]
    M[i:,j:] = A[i+1:,j+1:]

    return M

def alg_dop(A, i, j):
    M = minor(A, i, j)
    return (-1)**(i+j) * np.linalg.det(M)

R = corr_matr.values
#Множественная корреляция y к x-ам
R_y_x = np.sqrt(1 - np.linalg.det(R)/alg_dop(R, 0, 0))
print(f'Коэффициент множественной корреляции Position: {R_y_x:.3f}')
```

Коэффициент множественной корреляции Position: 0.156

Коэффициент множественной корреляции получился тоже малым. Это значит, что данные параметры имеют слабую линейную связь с позицией.

```
In [ ]: import scipy.stats
```

```
kstest_res = [ scipy.stats.kstest(data_pd.iloc[:,i], 'norm') for i in range(len(
for i in range(len(kstest_res)):
    print(f'Значение статистики для {data_columns[i]}: {kstest_res[i].statistic:
```

Значение статистики для Position: 0.841, pvalue: 5.51e-92

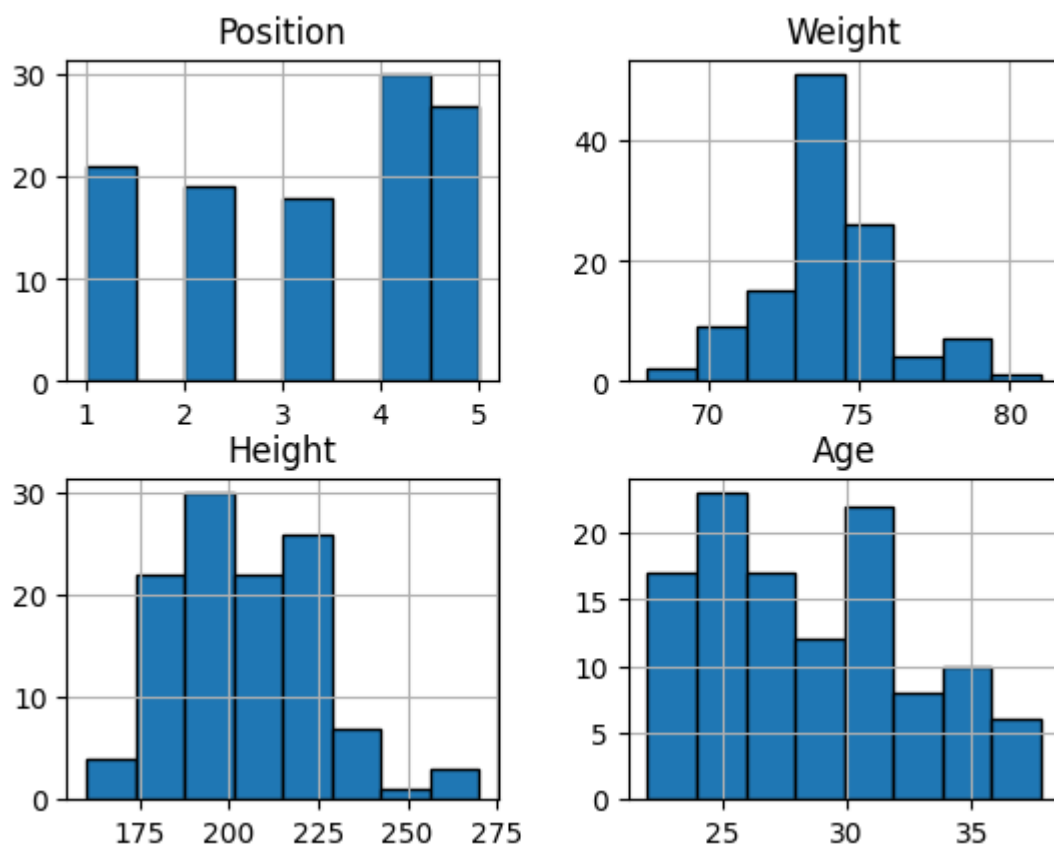
Значение статистики для Weight: 1.000, pvalue: 0.00e+00

Значение статистики для Height: 1.000, pvalue: 0.00e+00

Значение статистики для Age: 1.000, pvalue: 0.00e+00

Все имеющиеся параметры почти наверняка не распределены нормально.

```
In [ ]: data_pd.iloc[:,:].hist(bins = 8, edgecolor='black')
plt.show()
```



Ненормальность распределений параметров видна на данных оценках распределений.

Датасет Barotrop

Подготовка датасета

Обозначим классы следующими метками: BARO - 0, TROP - 1.

```
In [ ]: import numpy as np
import pandas as pd

data_pd=pd.read_csv('Barotrop_formatted.csv', sep=';', decimal=',')

print(data_pd.head(10))
print(data_pd.tail(10))
print(data_pd.describe())
```

| | LONGITUD | LATITUDE | CLASS |
|---|----------|----------|-------|
| 0 | 59.0 | 17 | 0 |
| 1 | 59.5 | 21 | 0 |
| 2 | 60.0 | 12 | 0 |
| 3 | 60.5 | 16 | 0 |
| 4 | 61.0 | 13 | 0 |
| 5 | 61.0 | 15 | 0 |
| 6 | 61.5 | 17 | 0 |
| 7 | 61.5 | 19 | 0 |
| 8 | 62.0 | 14 | 0 |
| 9 | 63.0 | 15 | 1 |

| | LONGITUD | LATITUDE | CLASS |
|----|----------|----------|-------|
| 27 | 68.0 | 14 | 0 |
| 28 | 68.5 | 18 | 0 |
| 29 | 69.0 | 13 | 0 |
| 30 | 69.0 | 15 | 0 |
| 31 | 69.5 | 17 | 0 |
| 32 | 69.5 | 19 | 0 |
| 33 | 70.0 | 12 | 0 |
| 34 | 70.5 | 16 | 0 |
| 35 | 71.0 | 17 | 0 |
| 36 | 71.5 | 21 | 0 |

| | LONGITUD | LATITUDE | CLASS |
|-------|-----------|-----------|-----------|
| count | 37.000000 | 37.000000 | 37.000000 |
| mean | 65.432432 | 16.216216 | 0.486486 |
| std | 3.438269 | 2.709399 | 0.506712 |
| min | 59.000000 | 12.000000 | 0.000000 |
| 25% | 63.000000 | 14.000000 | 0.000000 |
| 50% | 65.500000 | 16.000000 | 0.000000 |
| 75% | 68.000000 | 18.000000 | 1.000000 |
| max | 71.500000 | 21.000000 | 1.000000 |

Разведочный анализ данных

```
In [ ]: import matplotlib.pyplot as plt

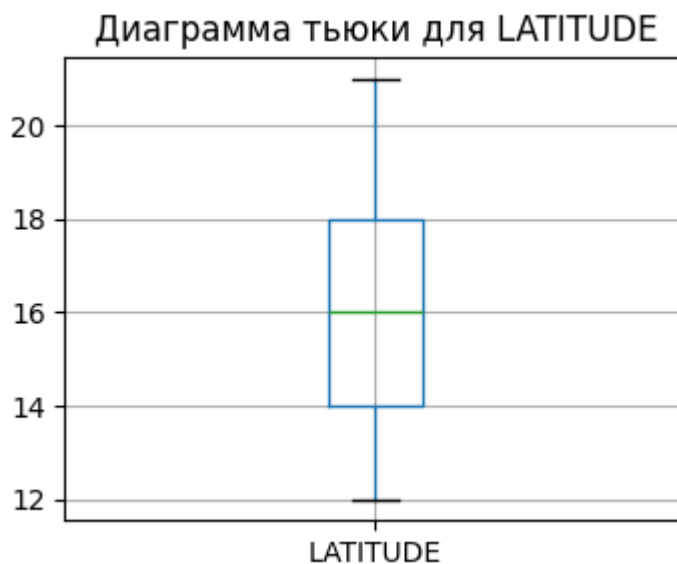
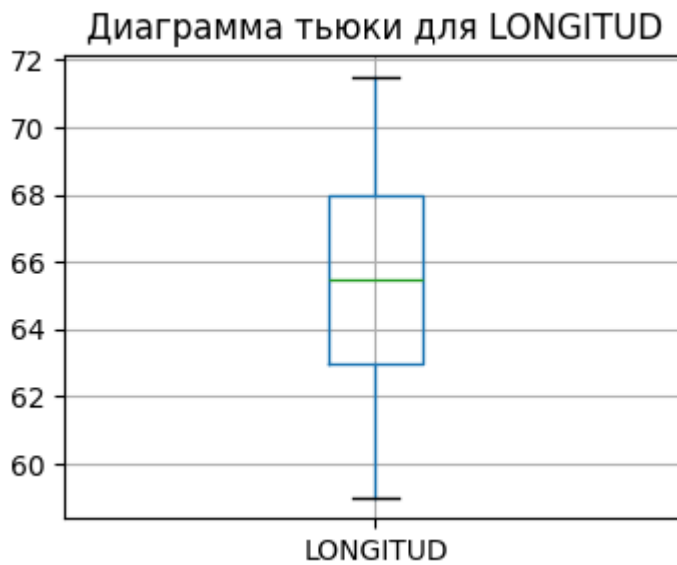
data_columns = tuple(data_pd.columns)

for i in range(len(data_pd.iloc[0]) - 1):
```



```
plt.figure(figsize=(4.0,3))
plt.title(f'Диаграмма тыюки для {data_columns[i]}')
data_pd[[data_columns[i]]].boxplot()
plt.show()

plt.show()
```



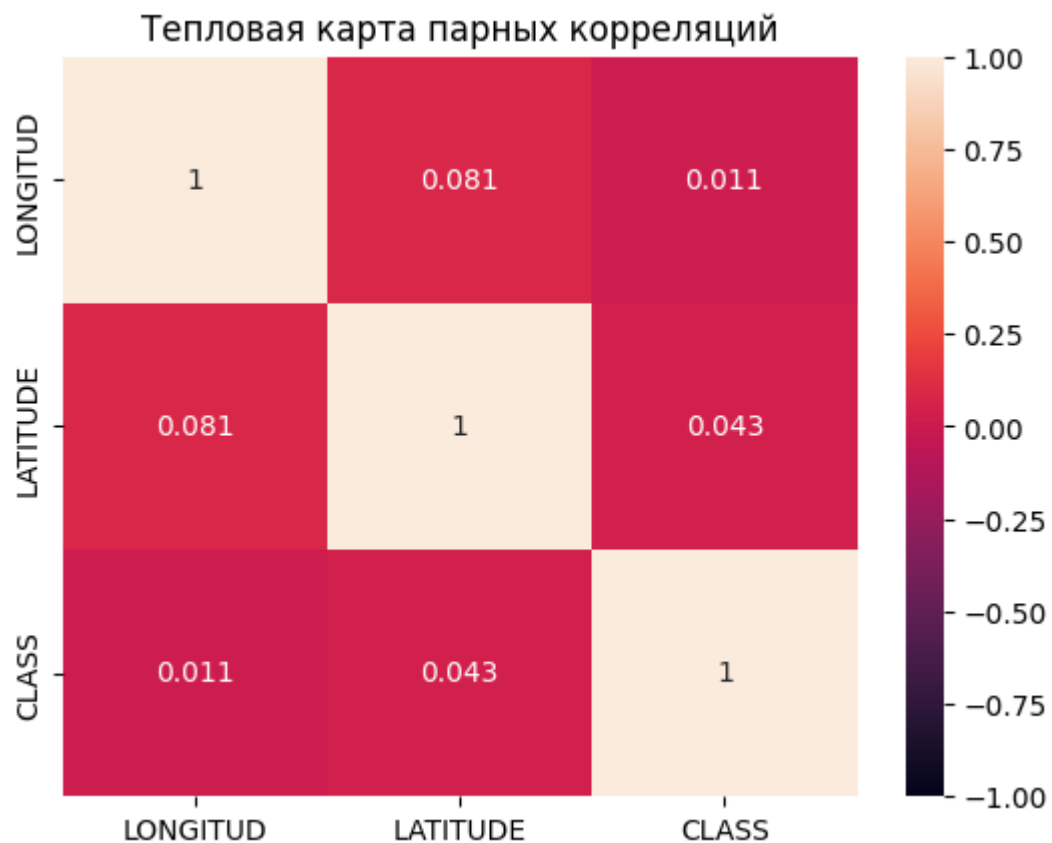
Диаграммы Тыюки выглядят симметричными, а медиана находится ровно в середине ящика. Вполне возможно, что данные параметры распределены нормально. Также выбросы не наблюдаются, а усы не выглядят большими.

```
In [ ]: import seaborn as sns

corr_matr = data_pd.corr()
print(corr_matr)

sns.heatmap(corr_matr, annot=True, vmin=-1, vmax=1)
plt.title('Тепловая карта парных корреляций')
plt.show()
```

| | LONGITUD | LATITUDE | CLASS |
|----------|----------|----------|----------|
| LONGITUD | 1.000000 | 0.080631 | 0.011419 |
| LATITUDE | 0.080631 | 1.000000 | 0.042654 |
| CLASS | 0.011419 | 0.042654 | 1.000000 |



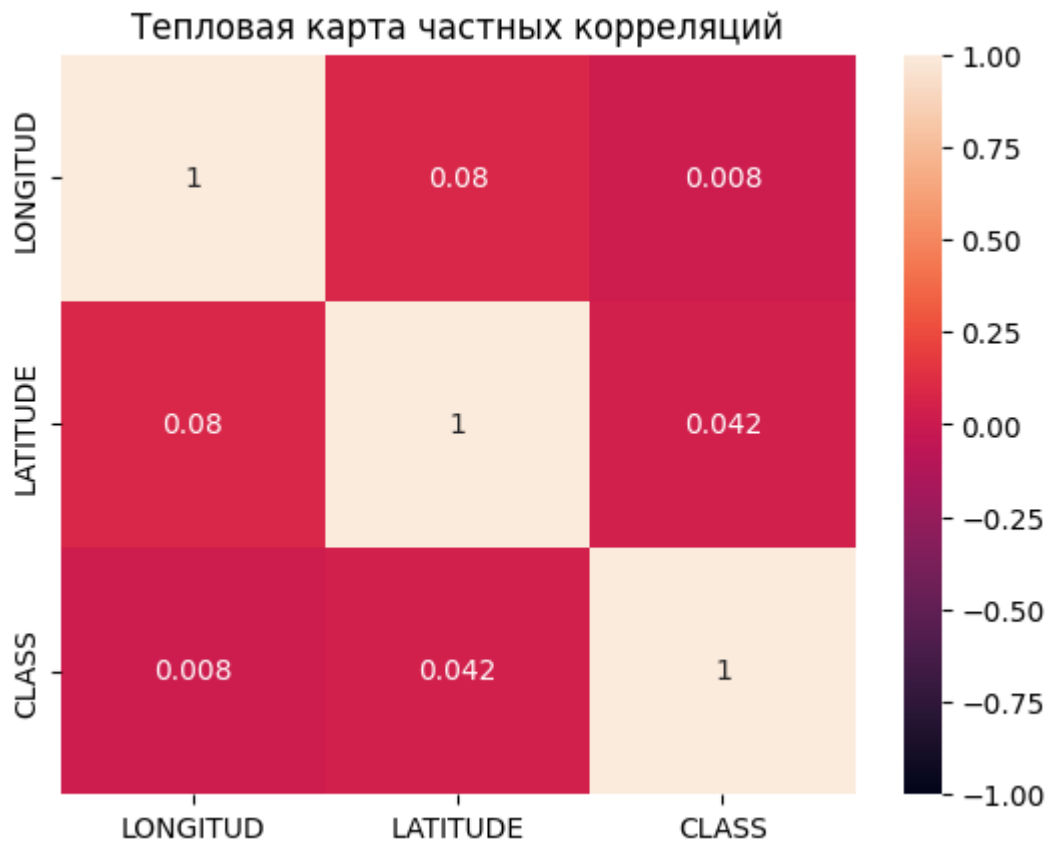
Судя по парным корреляциям, между всеми переменными корреляции не наблюдаются.

```
In [ ]: import pingouin

pcorr_matr = data_pd.pcorr()
print(pcorr_matr)

sns.heatmap(pcorr_matr, annot=True, vmin=-1, vmax=1)
plt.title('Тепловая карта частных корреляций')
plt.show()
```

| | LONGITUD | LATITUDE | CLASS |
|----------|----------|----------|----------|
| LONGITUD | 1.000000 | 0.080222 | 0.008014 |
| LATITUDE | 0.080222 | 1.000000 | 0.041872 |
| CLASS | 0.008014 | 0.041872 | 1.000000 |



```
In [ ]: #Количество параметров
n = len(data_pd)
k = len(data_pd.iloc[0]) - 1

def minor(A, i, j):
    a_shape = A.shape[0]
    M = np.eye(a_shape - 1)
    M[:i,:j] = A[:i,:j]
    M[:i,j:] = A[:i,j+1:]
    M[i,:j] = A[i+1,:j]
    M[i:,j:] = A[i+1,j+1:]

    return M

def alg_dop(A, i, j):
    M = minor(A, i, j)
    return (-1)**(i+j) * np.linalg.det(M)

R = corr_matr.values
#Множественная корреляция у к х-ам
R_y_x = np.sqrt(1 - np.linalg.det(R)/alg_dop(R, k, k))
print(f'Коэффициент множественной корреляции Position: {R_y_x:.3f}')
```

Коэффициент множественной корреляции Position: 0.043

Коэффициент множественной корреляции получился тоже малым. Это значит, что данные параметры имеют слабую линейную связь с классом.

```
In [ ]: import scipy.stats

kstest_res = [ scipy.stats.kstest(data_pd.iloc[:,i], 'norm') for i in range(len(
for i in range(len(kstest_res)):
    print(f'Значение статистики для {data_columns[i]}: {kstest_res[i].statistic:')
```

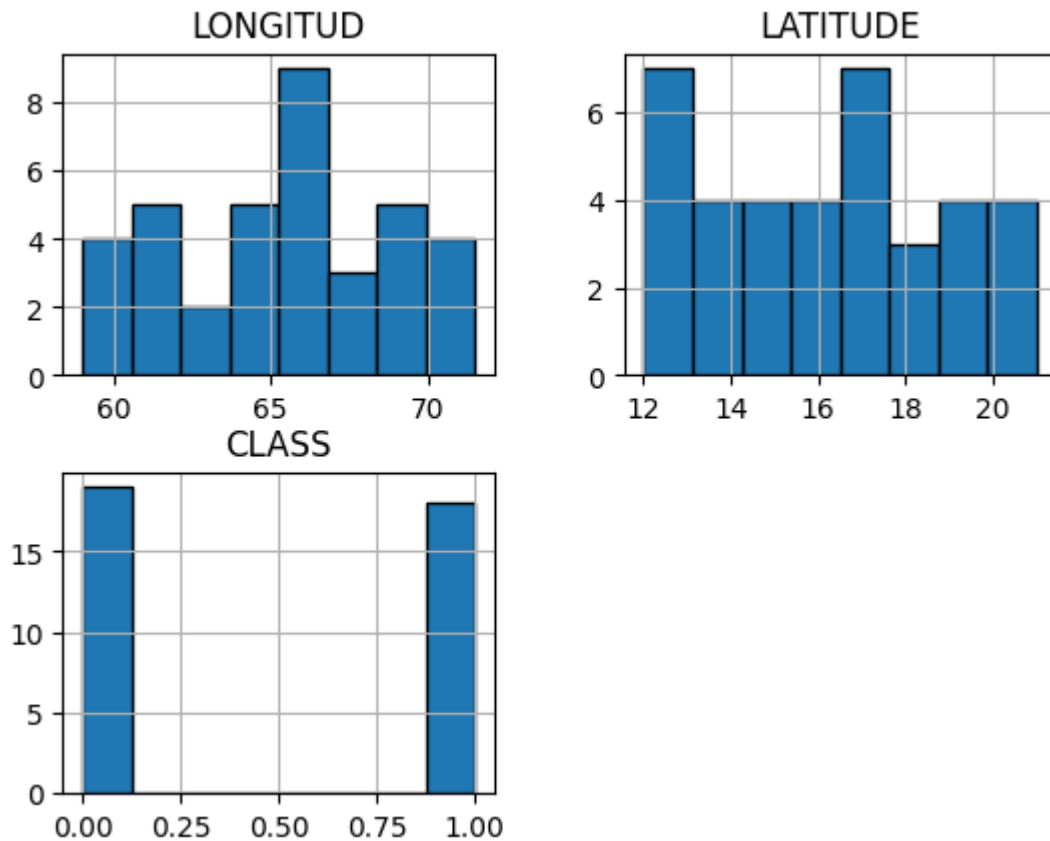
Значение статистики для LONGITUD: 1.000, pvalue: 0.00e+00

Значение статистики для LATITUDE: 1.000, pvalue: 0.00e+00

Значение статистики для CLASS: 0.500, pvalue: 4.45e-09

Все имеющиеся параметры почти наверняка не распределены нормально.

```
In [ ]: data_pd.iloc[:, :].hist(bins = 8, edgecolor='black')  
plt.show()
```



Ненормальность распределений параметров видна на данных оценках распределений.

В данном датасете корреляция между параметрами не наблюдается. Это значит, что скорее всего модели основанные на линейной регрессии будут неэффективны, и строить их здесь не целесообразно.