МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики и инворматики
Кафедра математического моделирования и анализа данных

Отчет по лабораторной работе 5
Кластерный анализ неоднородных данных

Выполнил:
Карпович Артём Дмитриевич
студент 3 курса 7 группы

Преподаватель:
Малюгин Владимир Ильич

Минск, 2023

Опираясь на процесс предварительного анализа в лабораторной работе 3, можно точно сказать, что мы имеем четкое разделение на два кластера.

[1]:
```
import matplotlib.pyplot as plt
import statsmodels.api as sm
import pandas as pd
import numpy as np
import seaborn as sns
import graphviz

from sklearn.model_selection import train_test_split, RepeatedStratifiedKFold,␣
↪cross_val_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,␣
↪QuadraticDiscriminantAnalysis
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_samples, silhouette_score
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from sklearn import datasets, tree
from matplotlib import pyplot
```

Подгрузили необходимый dataset, создали dataframe с необходимыми нам видами ириса “setosa” и “virginica”.

[2]:
```
ds = datasets.load_iris()

ext_target = ds.target[:, None]
df = pd.DataFrame(
    np.concatenate((ds.data, ds.target_names[ext_target]), axis=1),
    columns=ds.feature_names + ['target_name'])

df = df.loc[df['target_name'] != 'versicolor'].reset_index(drop=True)

df = df.rename(columns={
    'sepal width (cm)': 'sepalwid',
    'sepal length (cm)': 'sepallen',
    'petal length (cm)': 'petallen',
    'petal width (cm)':'petalwid'})

df
```

[2]:

	sepallen	sepalwid	petallen	petalwid	target_name
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
..	...	...	...	...	...

```
95      6.7      3.0      5.2      2.3   virginica
96      6.3      2.5      5.0      1.9   virginica
97      6.5      3.0      5.2      2.0   virginica
98      6.2      3.4      5.4      2.3   virginica
99      5.9      3.0      5.1      1.8   virginica

[100 rows x 5 columns]
```

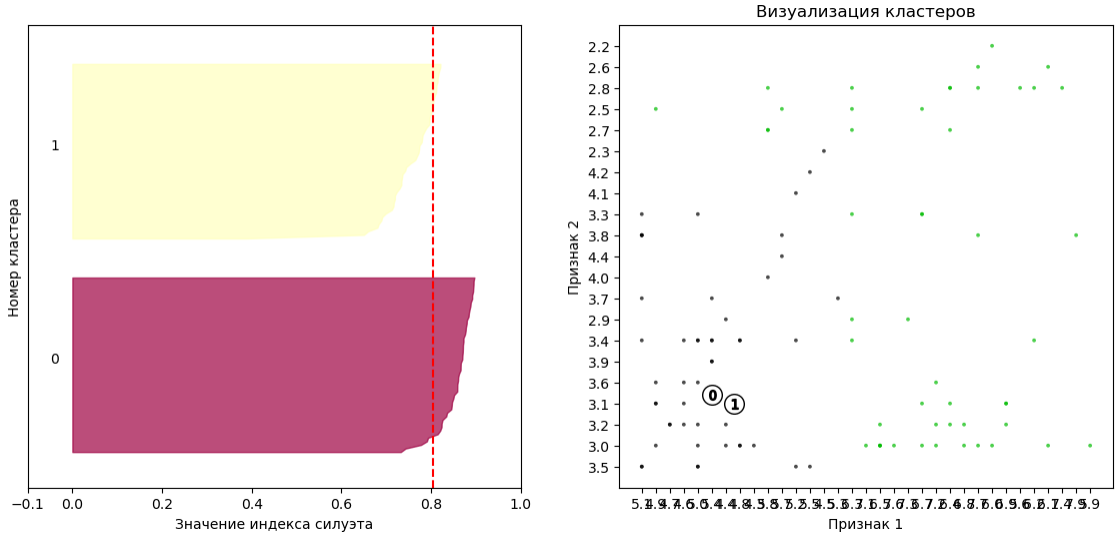
Алгоритм k-средних

Предположим, что оптимальное число кластеров в нашем случае равно двум, это можно заметить по матрице диаграмм рассеяния в лабораторной работе номер 3. Проверим этот факт, рассмотрев количество кластеров от двух до пяти.

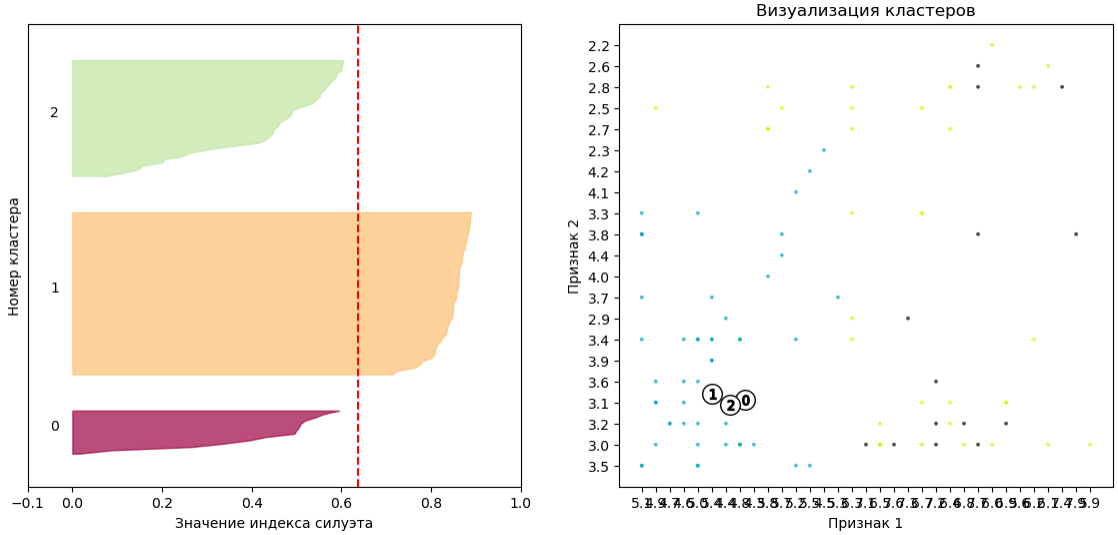
[3]:
```
X = df.drop(columns=['target_name'], axis=1).values

for i in range(2, 6):

    n_clusters = i

    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(X)

    silhouette_avg = silhouette_score(X, cluster_labels)
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(14, 6)

    ax1.set_xlim([-0.1, 1])
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    y_lower = 10
    for i in range(n_clusters):
        ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels␣
↪== i]
        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = plt.cm.get_cmap("Spectral")(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
```

```
        y_lower = y_upper + 10

    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_xlabel("Значение индекса силуэта")
    ax1.set_ylabel("Номер кластера")

    ax1.set_yticks([])
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    colors = plt.cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(X[:, 0], X[:, 1], marker=".", s=30, lw=0, alpha=0.7, c=colors,␣
↪edgecolor="k")

    centers = kmeans.cluster_centers_
    ax2.scatter(
        centers[:, 0],
        centers[:, 1],
        marker="o",
        c="white",
        alpha=1,
        s=200,
        edgecolor="k",
    )

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker="$%d$" % i, alpha=1, s=50, edgecolor="k")

    ax2.set_title("Визуализация кластеров")
    ax2.set_xlabel("Признак 1")
    ax2.set_ylabel("Признак 2")

    print(f'Значение индекса силуэтта для {i+1}-кластеров: {silhouette_avg}')

    plt.show()
```

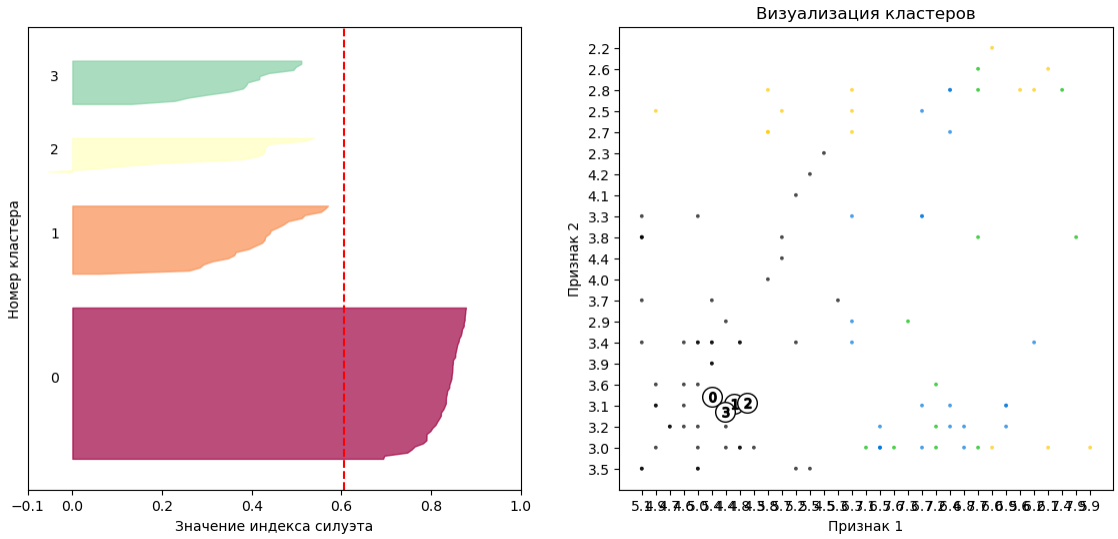
```
Значение индекса силуэтта для 2-кластеров: 0.8045671428949102
```



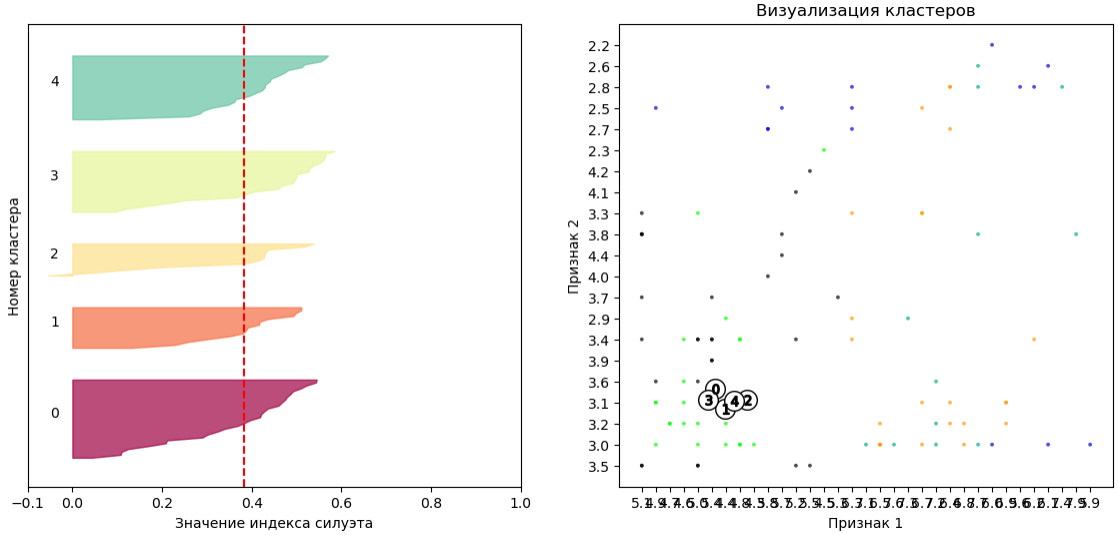
Значение индекса силуэтта для 3-кластеров: 0.6365670305909423



Значение индекса силуэтта для 4-кластеров: 0.6048361411020379




```
Значение индекса силуэтта для 5-кластеров: 0.38366413399968347
```

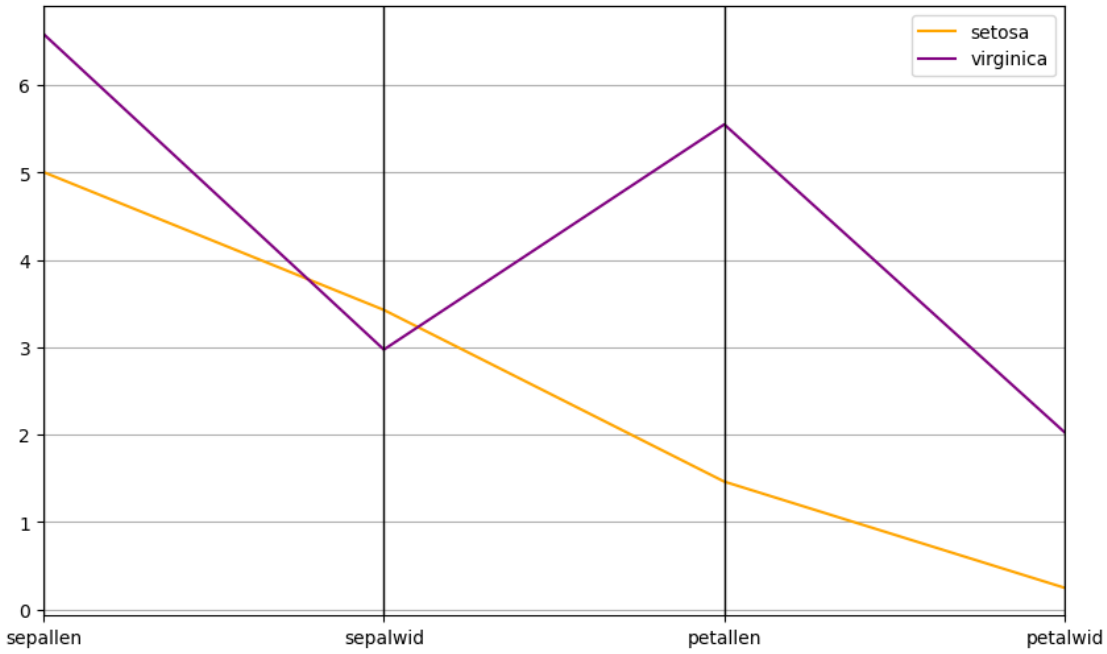


Видно, что значение IS-индекса >= 0.6 для количества 2-4, однако для 2 он достигает наибольшего значения.

Построим график центров для двух кластеров.

[4]:
```
kmeans = KMeans(n_clusters=2, random_state=42)
cluster_labels = kmeans.fit_predict(X)

centers = pd.DataFrame(kmeans.cluster_centers_[:, 0:4], columns=['sepallen',␣
↪'sepalwid', 'petallen', 'petalwid'])
```

```
centers['label'] = pd.Series(['setosa', 'virginica'])

plt.figure(figsize=(10, 6))

pd.plotting.parallel_coordinates(centers, 'label', color=('orange', 'purple'))

plt.show()
```



Видно, что модель слабо различает кластер “setosa” по переменной SEPALWID.

Проведем дисперсионный анализ переменных по кластерам.

[5]:
```
df_ = df.drop(columns=['target_name'], axis=1)
df_['labels'] = kmeans.labels_

model = sm.formula.ols('labels ~ sepallen + sepalwid + petalwid + petallen',␣
 ↪data=df_).fit()
anovatable = sm.stats.anova_lm(model, typ=2)

print(anovatable)
```

```
                sum_sq    df             F         PR(>F)
sepallen  2.922601e-01  32.0  2.653232e+26  3.868132e-144
sepalwid  9.756169e-02  20.0  1.417115e+26  1.758411e-142
petalwid  2.243182e-01  17.0  3.833286e+26  8.677922e-145
petallen  1.036017e+00  28.0  1.074892e+27  1.928173e-147
Residual  3.786491e-28  11.0           NaN            NaN
```

У каждой переменной уровень значимости ниже 0.05, что позволяет отглонить гипотезу о равенстве межгрупповых средних, то есть все признаки оказались информативными для модели.

Рассмотрим показатель точности наших предсказаний для 2 кластеров.

[6]:
```
le = LabelEncoder()

kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X)
y_pred = kmeans.labels_
y = le.fit_transform(df['target_name'])

accuracy_score = sum(y_pred==y) / y.size
print(f'Точность: {accuracy_score}')
```

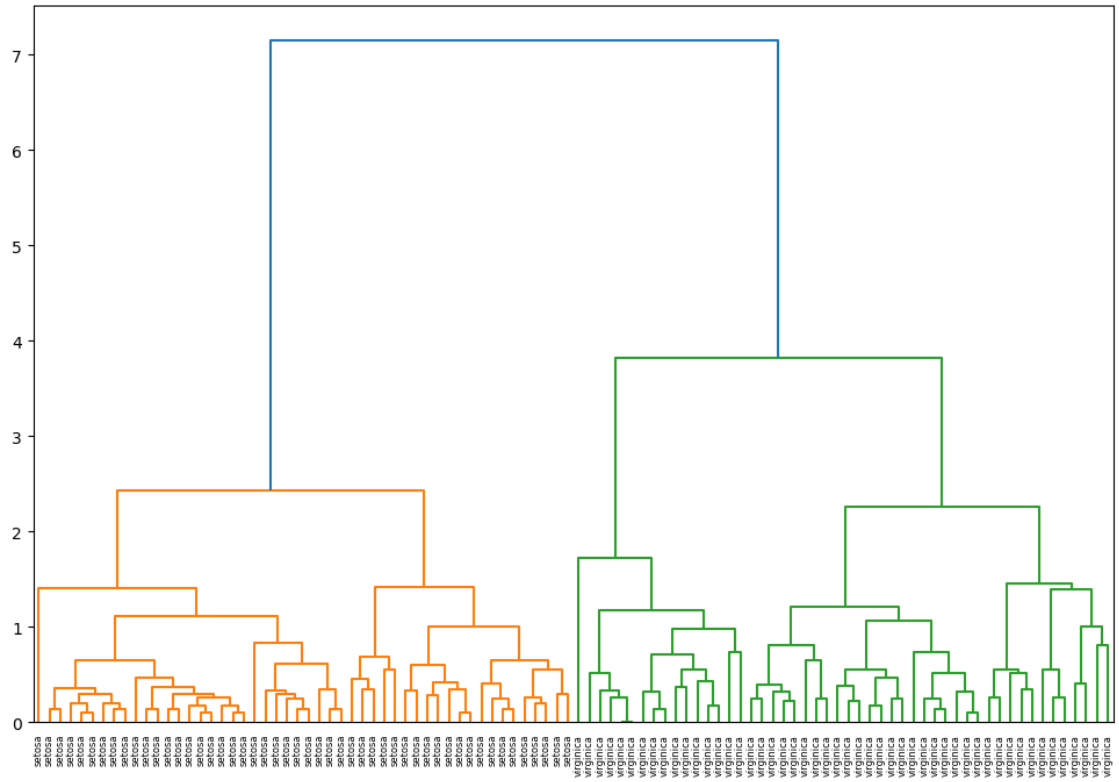
```
Точность: 1.0
```

Показатель точности равен 1.0, что является идеальным результатом для алгоритма. То есть мы в 100% случаях правильно определили метку класса для рассматриваемого объекта.

Иерархический кластерный анализ

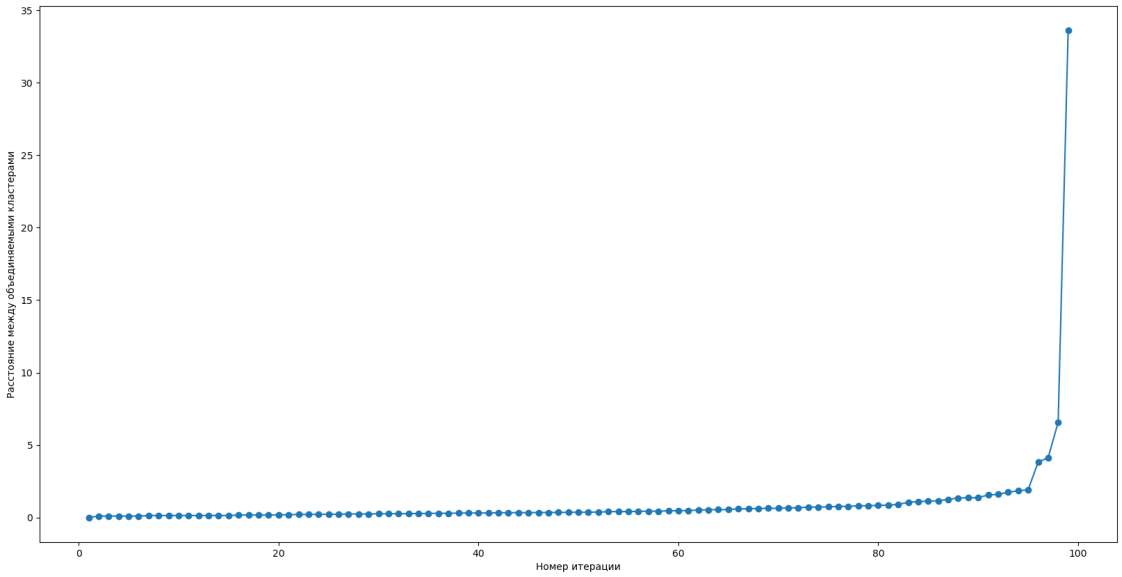
Рассматриваем ту же задачу, но с применением иерархического кластерного анализа. Рассмотрим дендограмму.

[9]:
```
ac = AgglomerativeClustering(n_clusters=2, compute_distances=True)

df['target_values'] = le.fit_transform(df['target_name'])
samples = df.drop(columns = ['target_name'], axis = 1).values
mergings = linkage(samples, method='complete')
labels = fcluster(mergings, 2, criterion='maxclust')

plt.figure(figsize=(12, 8))

dendrogram(mergings,
           labels=le.inverse_transform(labels - 1),
           leaf_rotation=90,
           leaf_font_size=6,
           )

plt.show()
```



Видно, что в итоге мы получили два кластера. Рассмотрим график расстояния между кластерами.

[10]:
```
plt.figure(figsize=(20, 10))
plt.plot(range(1, len(X)), ac.fit(X).distances_, marker='o')
plt.xlabel('Номер итерации')
plt.ylabel('Расстояние между объединяемыми кластерами')
plt.show()
```



На этом графике видно, что он резко устремляется вверх примерно на показателе расстояния равном 4, что, опираясь на первый график, примерно соответствует моменту разделения на два кластера.

Проведем дисперсионный анализ.

[11]:
```
df_ = df.drop(columns=['target_name'], axis=1)
df_['labels'] = ac.labels_

model = sm.formula.ols('labels ~ sepallen + sepalwid + petalwid + petallen',␣
 ↪data=df_).fit()
anovatable = sm.stats.anova_lm(model, typ=2)

print(anovatable)
```

```
               sum_sq    df             F         PR(>F)
sepallen  2.061048e-01  32.0  1.691247e+26  4.603900e-143
sepalwid  9.622608e-02  20.0  1.263374e+26  3.306915e-142
petalwid  2.239569e-01  17.0  3.459270e+26  1.526311e-144
petallen  1.036705e+00  28.0  9.722235e+26  3.349177e-147
Residual  4.189129e-28  11.0           NaN            NaN
```

У каждой переменной уровень значимости ниже 0.05, что позволяет отглонить гипотезу о равенстве межгрупповых средних, то есть все признаки оказались информативными для модели.

[12]:
```
clusters = ac.fit_predict(X)

silhouette_avg = silhouette_score(X, clusters)
print("Средний индекс силуэта:", silhouette_avg)
```

```
Средний индекс силуэта: 0.8045671428949102
```

Вывод

Оба алгоритма справились с задачей кластеризации в рассматриваемой выборке, определив два кластера. Алгоритм k-средних с идеальной точностью справился с этой задачей. Поскольку ИАК не гарантирует одинаковое присвоение меток кластеров между разными запусками, нельзя применить accuracy_score для оценки его качества, но можно использовать средний индекс силуэта, который равен индексу силуэта для k-средних.