ГУАП

КАФЕДРА № 41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ст. преподаватель |  |  |  | В.В. Боженко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| КЛАСТЕРИЗАЦИЯ |
| по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4216 |  |  |  | С.Д. Комолова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**Цель лабораторной работы:** изучение алгоритмов и методов кластеризации на практике.

**Ход работы:**

Ссылка на репозиторий github: <https://github.com/sufferix/data-analysis>

Были установлены библиотеки, среди которых есть библиотека sklearn, с помощью которой будет выполнена кластеризация в данной лабораторной работе

import numpy as np  
import pandas as pd  
from sklearn.preprocessing import StandardScaler  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_score  
from sklearn import metrics  
import matplotlib.pyplot as plt

Была продублирована функция из задания, которая задаёт определенный датасет из рандомных значений

def make\_data(n, seed):  
    np.random.seed(seed)  
    shift\_matrix = np.array([[3, 3], [6, 9], [9, 3]])  
    data = np.random.randn(3, 2, n) + shift\_matrix.reshape((3, 2, 1))  
    data = np.swapaxes(data, 1, 2)  
    data = data.reshape((-1, 2))  
    data \*= np.array([[20, 0.5]])  
    df = pd.DataFrame(data, columns=['x', 'y'])  
    df = df.sample(frac=1.0).reset\_index(drop=True)  
    return df

Были созданы тренировочные и валидационные выборки из них

train = make\_data(50, seed=31)  
test = make\_data(90, seed=265)

Обе выборки были стандартизированы, чтобы сохранять одинаковую размерность

scaler = StandardScaler()  
train\_scaled = scaler.fit\_transform(train)  
test\_scaled = scaler.transform(test)

Далее была создана функция, которая принимает на вход выборки и количество кластеров, которые необходимо создать в графике. Также данная функция выводить коэффициент силуэта, отображающий, насколько точно кластеры разделены друг от друга. Коэффициент нормализованный, поэтому может принимать значения от 0 до 1, чем ближе к 1, тем более четко кластеры разделены.

В данной функции используется кластеризация методом k-means.

Метод k-means делит данные на k кластеров, минимизируя внутрикластерные расстояния. Он работает итеративно: сначала случайно выбираются центры кластеров, затем каждому объекту данных назначается ближайший кластер на основе расстояния до центра. После этого центры кластеров пересчитываются как среднее всех точек в кластере. Процесс повторяется, пока центры кластеров не перестанут изменяться или не будет достигнут максимум итераций

def clustering(train\_scaled, test\_scaled, n\_clusters):  
  
    kmeans = KMeans(n\_clusters=n\_clusters, random\_state=42)  
    train\_labels = kmeans.fit\_predict(train\_scaled)  
  
    plt.figure(figsize=(10, 6))  
    for cluster in range(n\_clusters):  
        plt.scatter(train\_scaled[train\_labels == cluster, 0], train\_scaled[train\_labels == cluster, 1], label=f'Cluster {cluster}')  
    plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], c='black', marker='X', s=200, label='Centroids')  
    plt.title(f"Кластеризация тренировочных данных с {n\_clusters} кластерами")  
    plt.legend()  
    plt.show()  
  
    test\_labels = kmeans.predict(test\_scaled)  
  
  
    plt.figure(figsize=(10, 6))  
    for cluster in range(n\_clusters):  
        plt.scatter(test\_scaled[test\_labels == cluster, 0], test\_scaled[test\_labels == cluster, 1], label=f'Кластер {cluster}')  
    plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], c='black', marker='X', s=200, label='Центр')  
    plt.title(f"Кластеризация тестовых данных с {n\_clusters} кластерами")  
    plt.legend()  
    plt.show()  
  
    silhouette\_avg = silhouette\_score(train\_scaled, train\_labels)  
    print(f"Коэффициент силуэта для {n\_clusters} кластеров: {silhouette\_avg}")

Необходимо вывести графики с 3 кластерами, также были выведены с 2 и 4 кластерами для сравнения (см. Рисунки 1-6)

clustering(train\_scaled, test\_scaled, n\_clusters=3)  
  
clustering(train\_scaled, test\_scaled, n\_clusters=2)  
clustering(train\_scaled, test\_scaled, n\_clusters=4)

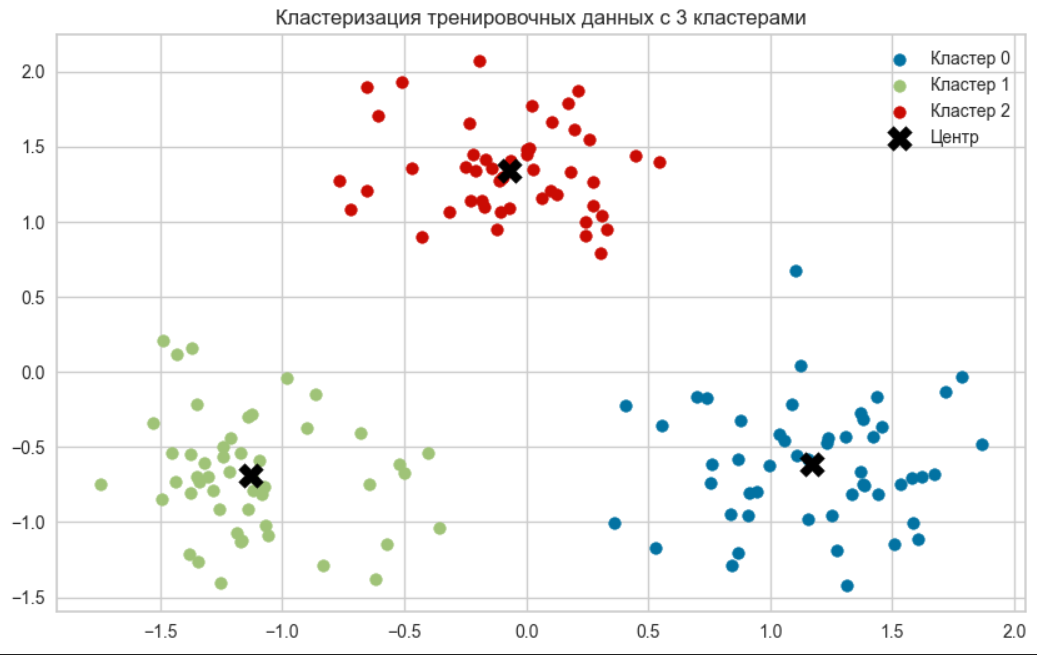


Рисунок 1 – Кластеризация тренировочных данных с 3 кластерами

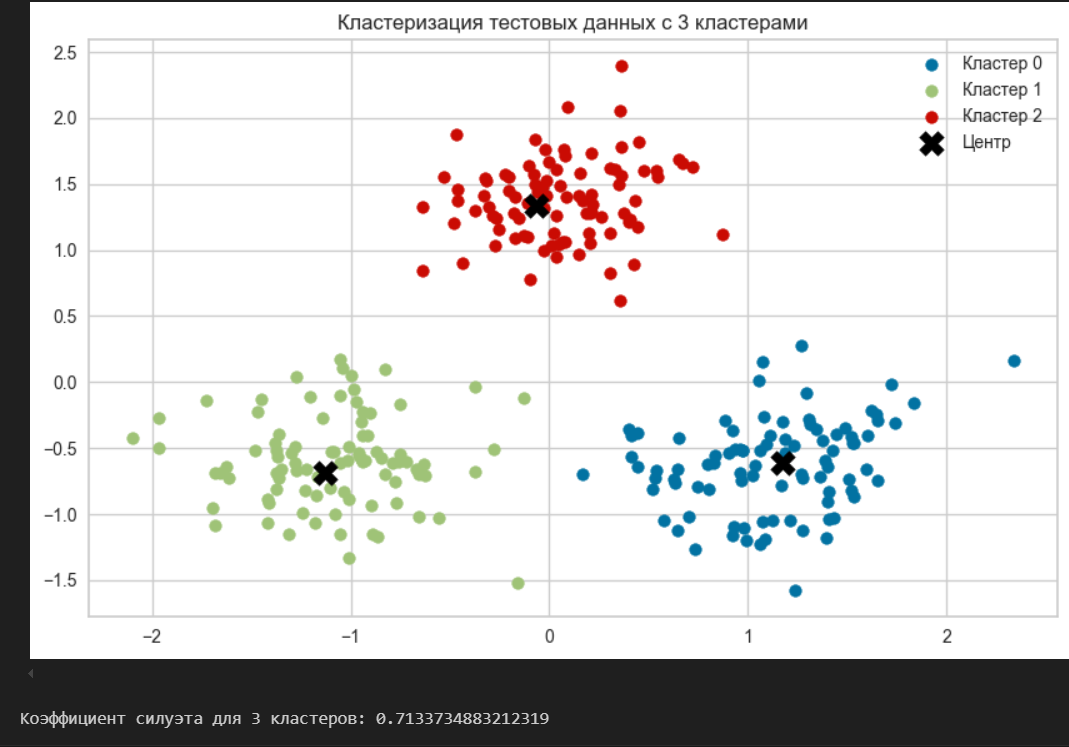


Рисунок 2 – Кластеризация тестовых данных с 3 кластерами и коэффициент силуэта

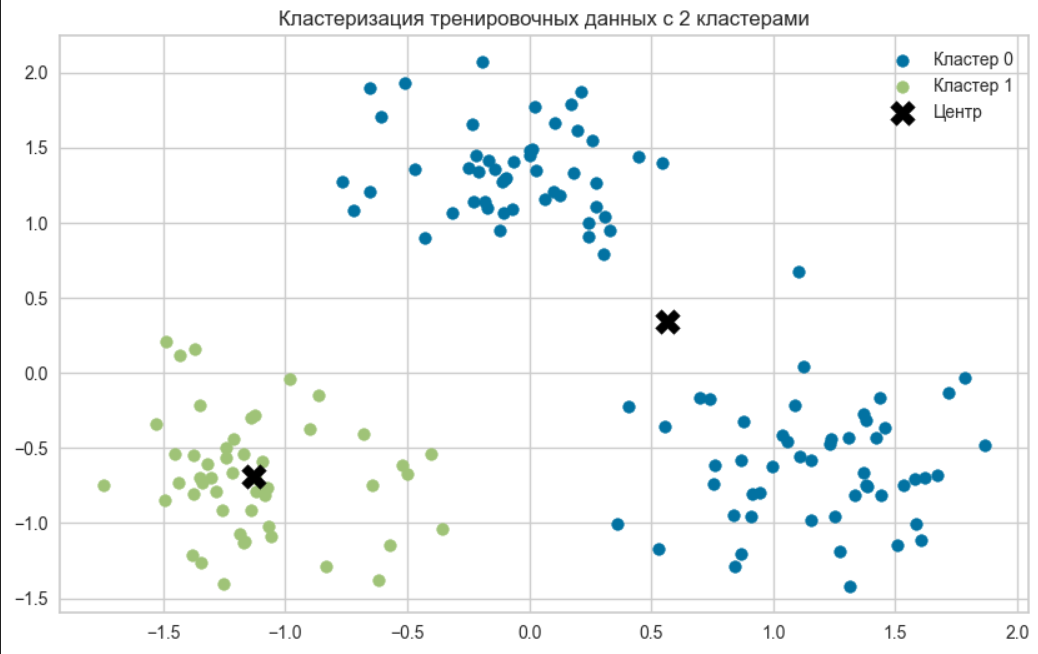


Рисунок 3 - Кластеризация тренировочных данных с 2 кластерами

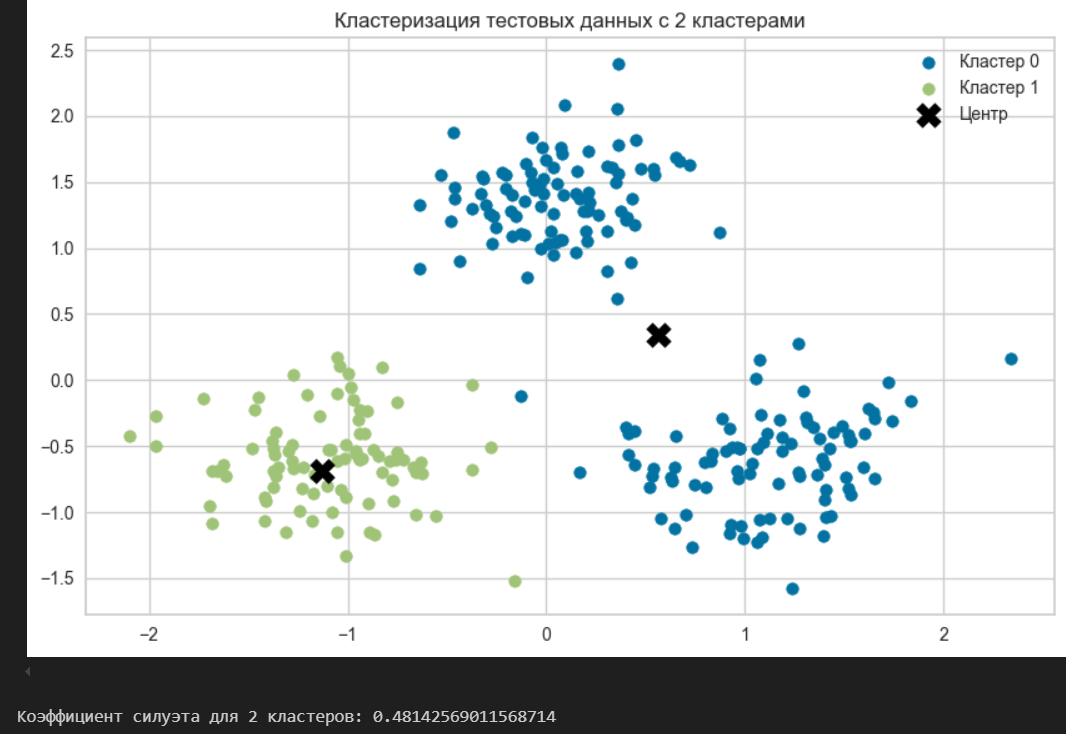


Рисунок 4 – Кластеризация тестовых данных с 2 кластерами и коэффициент силуэта

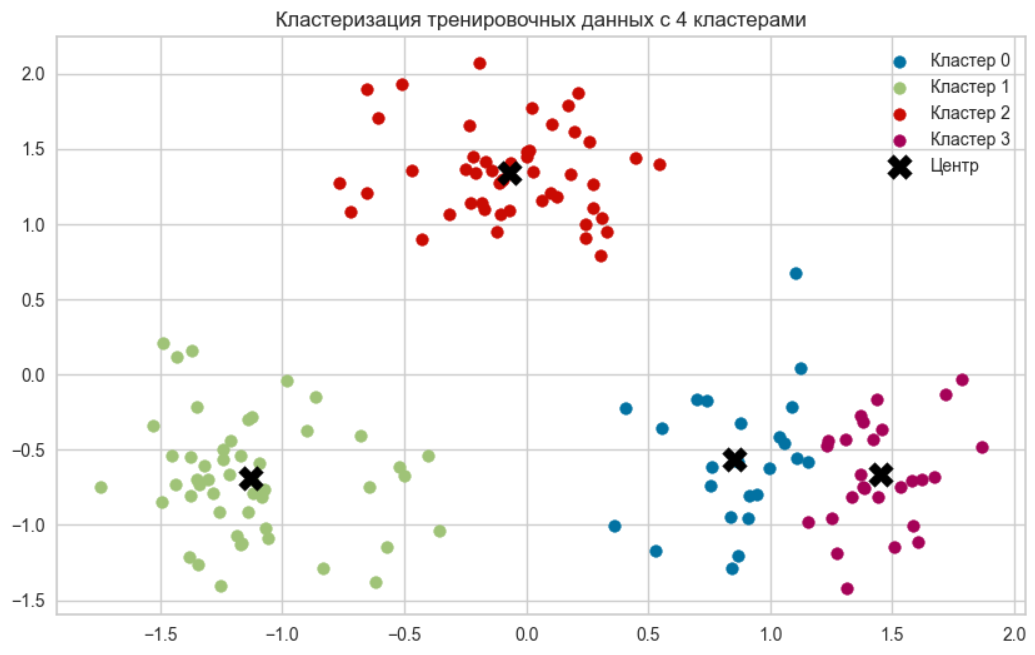


Рисунок 5 – Кластеризация тренировочных данных с 4 кластерами

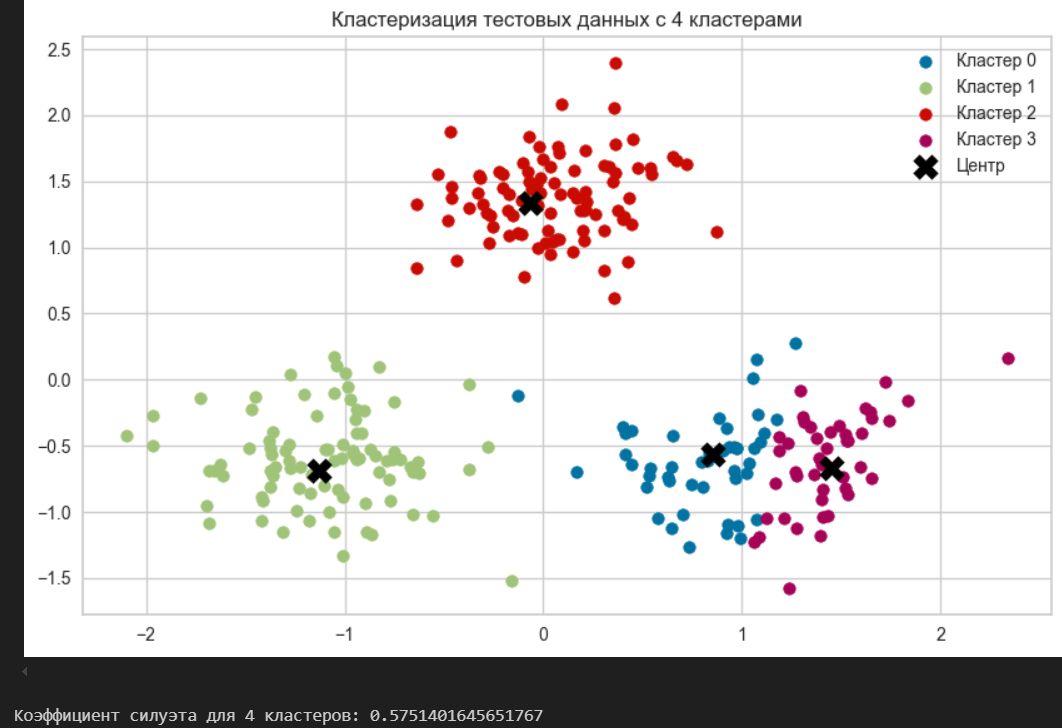


Рисунок 6 - Кластеризация тестовых данных с 4 кластерами и коэффициент силуэта

* Кластеризация с тремя кластерами:

Хорошая кластеризация с коэффициентом силуэта 0.724, что указывает на чётко разделённые кластеры.

* Кластеризация с двумя кластерами:

Коэффициент силуэта снизился до 0.499, что говорит о менее чётком разделении данных.

* Кластеризация с четырьмя кластерами:

Коэффициент силуэта составил 0.610, что лучше, чем для двух кластеров, но хуже, чем для трёх.

На основе данного анализа можно сказать, что не всегда чем больше кластеров, тем лучше, поэтому есть необходимость искать оптимальное количество кластеров для данного датасета - для этого применяется "метод локтя".

Метод локтя анализирует график зависимости инерции (внутрикластерной суммы квадратов расстояний) от числа кластеров. По мере увеличения числа кластеров инерция уменьшается, но после определённого момента снижение становится незначительным. Точка "перелома" на графике напоминает локоть — это и есть оптимальное число кластеров.

Для поиска оптимального количеста кластеров методом локтя используется метод KElbowVisualizer библиотеки yellowbrick. Метод принимает на вход обученную модель методом k-means, количество кластеров, которые необходимо визуализировать на графике и метрику, по которой будет составлен сам график. Был выбран коэффициент силуэта в качестве выборки (см. Рисунок 7)

from yellowbrick.cluster import KElbowVisualizer  
model = KMeans(random\_state=42)  
visualizer = KElbowVisualizer(model, k=(2, 10), metric='silhouette', timings=False)  
  
visualizer.fit(test\_scaled)  
visualizer.show()

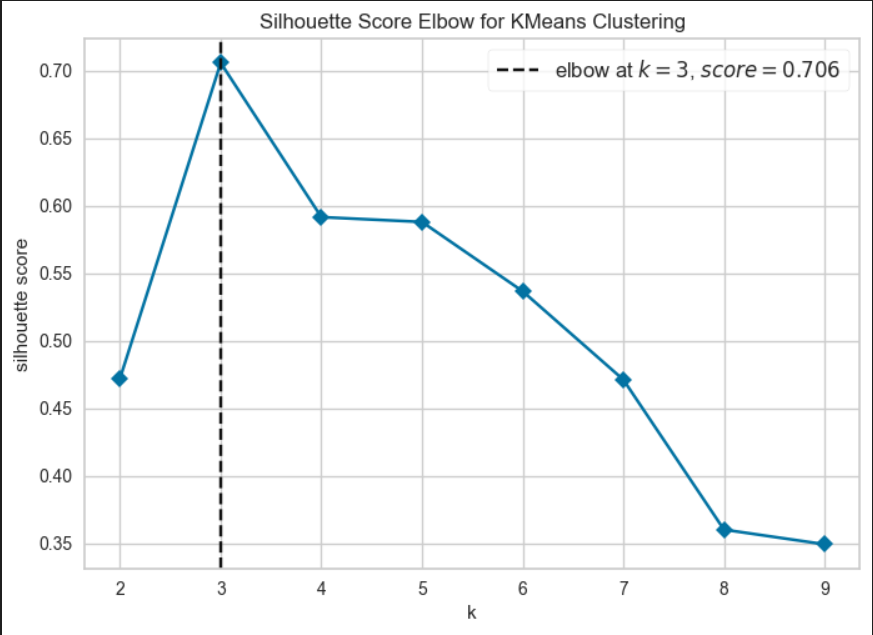


Рисунок 7 – Метод локтя для кластеризации

По данному графику можно определить, что оптимальное количество кластеров для данной модели является 3, так как имеет наибольший коэффициент силуэта.

Часть 2

Был загружен датасет 1heart.csv, в котором находится информация о пациентах с анализами для определения наличия заболевания сердца. Были выведены 5 записей, чтобы оценить датасет. (см. Рисунок 8)

data = pd.read\_csv('1heart.csv', sep=',')  
  
data.head(5)

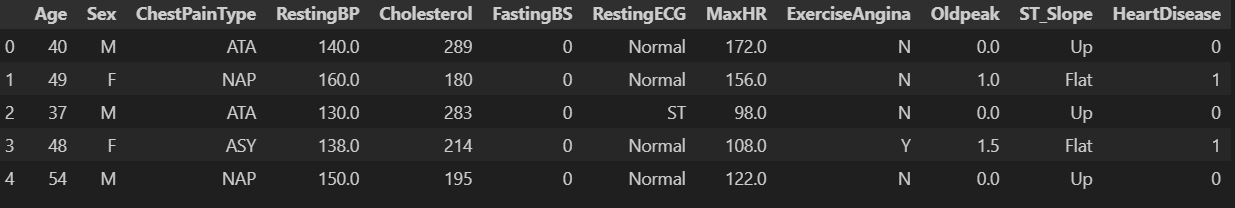


Рисунок 8 – Первые 5 записей датасета

Была выведена информация о всех столбцах (см. Рисунок 9)

data.info()

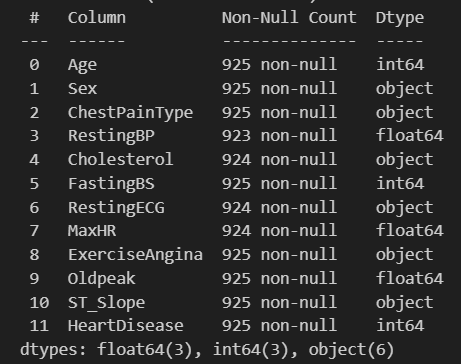


Рисунок 9 – Вывод информации о всех столбцах

Как можно увидеть, есть записи с нулевыми значениями, а также столбец Cholesterol имеет тип object, что не соответствует числовым значениям внутри него. Также можно заметить, что столбцы FastingBS и HeartDisease имеют двоичные значения, в то время как ExerciseAngina также показывает значения да/нет, но написано это текстом. Ниже эти моменты будут исправлены

data['ExerciseAngina'] = data['ExerciseAngina'].map({'N': 0, 'Y': 1})  
  
data['ExerciseAngina'].value\_counts()

nans = data[data.isnull().any(axis=1)]  
display(nans)

data = data.dropna()  
data.isna().sum()

Было определено, что в столбце Cholesterol есть некорректное значение, которое было выведено и переделано в корректное, после этого столбец был преобразован в числовой (см. Рисунок 10)

invalid\_rows = data[pd.to\_numeric(data['Cholesterol'], errors='coerce').isna()]  
  
invalid\_rows

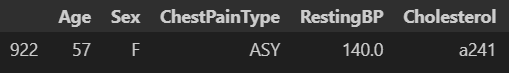


Рисунок 10 – Вывод записи с некорректными данными

data['Cholesterol'] = data['Cholesterol'].replace('a241','241')  
data['Cholesterol'] = pd.to\_numeric(data['Cholesterol'])

Был выведен метод .describe() для всех числовых столбцов (см. Рисунок 11)

data.describe()

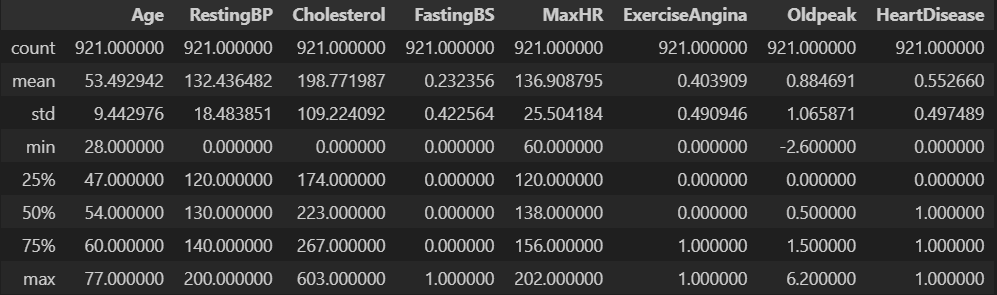


Рисунок 11 – Вывод метода .describe()

Отсюда видно, что столбцы RestingBP и Cholesterol имеют записи со значением 0, что в данном контексте является некорректным. Поэтому все нули были заменены на средние значения (см. Рисунок 12)

data['RestingBP'] = data['RestingBP'].replace(0, data['RestingBP'].median())  
data['Cholesterol'] = data['Cholesterol'].replace(0, data['Cholesterol'].median())  
  
data.describe()

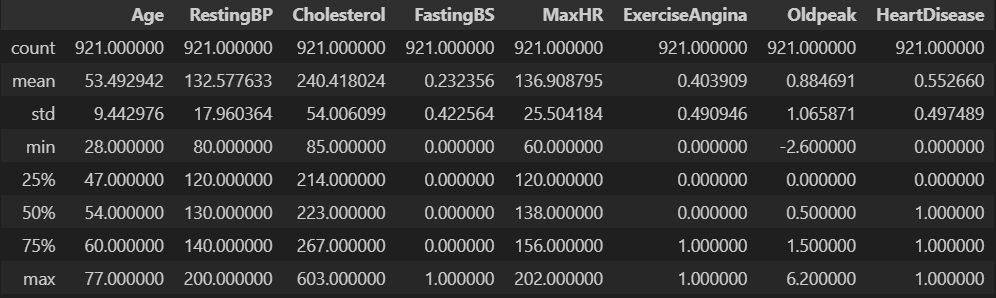


Рисунок 12 – Повторный вывод метода .describe()

Предобработка завершена, можно анализировать данный датасет.

Была выделена целевая переменная HeartDisease, по которой была выведена матрица диаграмм (см. Рисунок 13)

import seaborn as sns  
  
sns.pairplot(data, hue= 'HeartDisease')

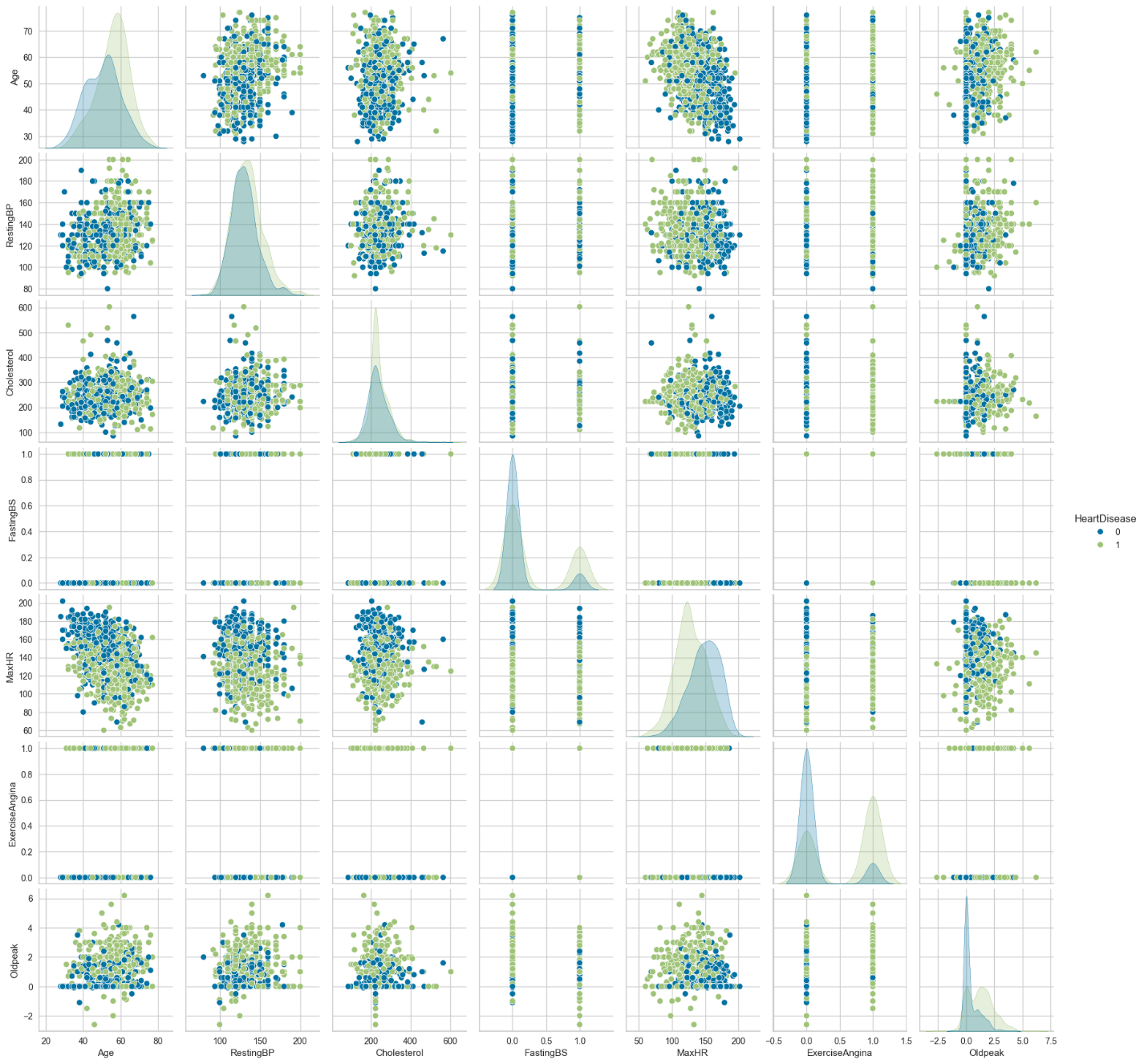


Рисунок 13 – Матрица диаграмм с целевой переменной HeartDisease

По данной матрице диаграмм видно, что данные имеют высокую степень перекрытия между группами, что затрудняет чёткое разделение классов. Наиболее отличающиеся признаки — это Oldpeak, ExerciseAngina, и MaxHR. Для пациентов с больным сердцем значения Oldpeak и ExerciseAngina смещены вверх, а MaxHR наоборот вниз.

Был создан датасет только из числовых столбцов, чтобы можно было рассчитывать количество кластеров. Также все значения были нормализованы

num\_data = data.select\_dtypes(exclude=['object'])  
scaler = StandardScaler()  
scaled = scaler.fit\_transform(num\_data)

Для определения оптимального количества кластеров был выведен метод локтя по метрике коэффициента силуэта (см. Рисунок 14)

kmeans\_elbow = KMeans(random\_state= 42)  
visualizer = KElbowVisualizer(kmeans\_elbow, k=(2, 10), metric='silhouette', timings=False)  
  
visualizer.fit(scaled)  
visualizer.show()

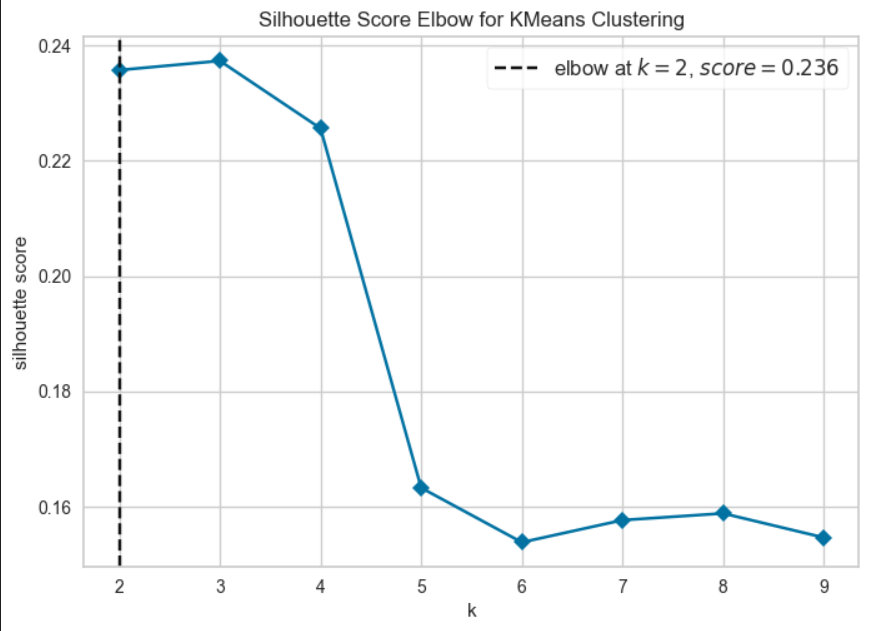


Рисунок 14 – Метод локтя

Данный график показывает, что лучше всего использовать два кластера для дальнейшей работы, которые были выведены на графике с центрами этих кластеров. (см. Рисунок 15)

kmeans = KMeans(n\_clusters=2, random\_state=42)  
kmeans.fit(scaled)  
  
labels = kmeans.labels\_  
  
plt.scatter(scaled[:, 0], scaled[:, 1], c=labels, cmap='rainbow', alpha=0.7)  
plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=200, marker= "X", c='black', label='Центры')  
plt.title("K-Means кластеризация с 2 кластерами")

plt.legend()  
plt.show()

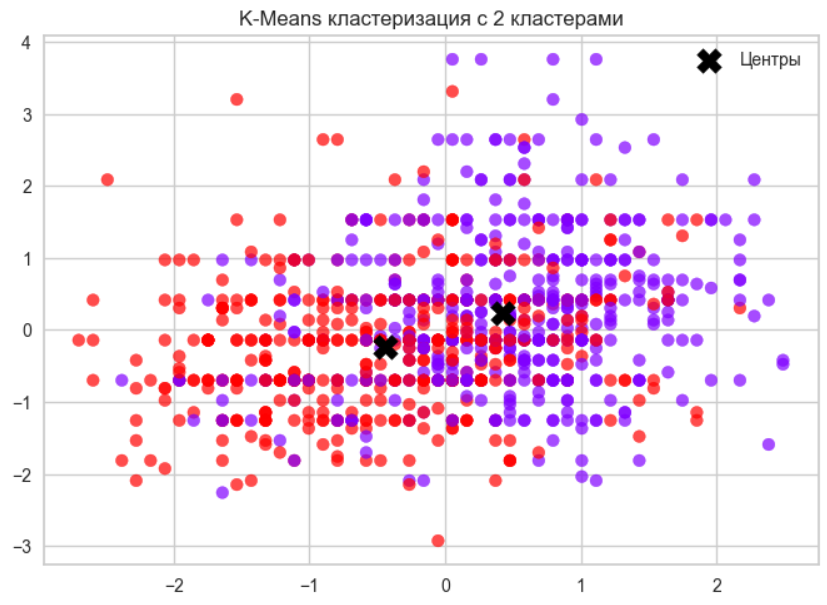


Рисунок 15 – K-Means кластеризация с 2 кластерами

По данному графику видно, что точки двух кластеров частично перекрываются, особенно в центральной части графика, что указывает на слабую отделимость кластеров. Чёрные крестики обозначают центры кластеров, вокруг которых данные распределены с некоторой плотностью.

Также был выведен график силуэтного анализа с помощью метода SilhouetteVisualizer библиотеки yellowbrick (см. Рисунок 16)

from yellowbrick.cluster import SilhouetteVisualizer  
visualizer = SilhouetteVisualizer(kmeans, colors='yellowbrick')  
visualizer.fit(scaled)  
visualizer.show()  
plt.show()

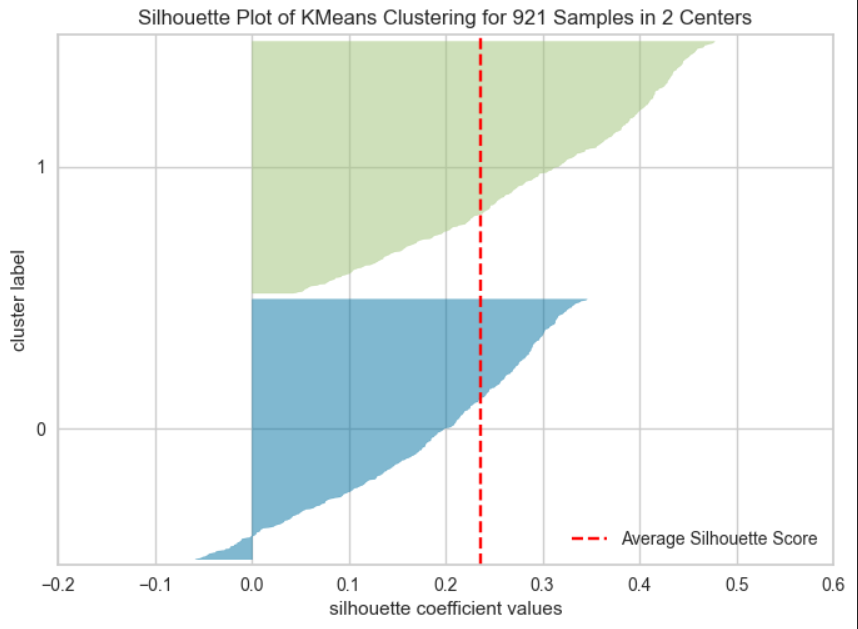


Рисунок 16 – График силуэтного анализа

График силуэтного анализа показывает, что средний коэффициент силуэта около 0.236, указывая на слабую отделимость кластеров. Наличие отрицательных значений силуэта в синем кластере свидетельствует о том, что некоторые точки ближе к другому кластеру, чем к своему. Это подтверждает пересечение кластеров и сложность в их чётком разделении.

Далее были выведены средние значения по каждому кластеру (см. Рисунок 17)

num\_data['Cluster'] = labels  
  
cluster\_means = num\_data.groupby('Cluster').mean()  
  
display(cluster\_means)



Рисунок 17 – Средние значения по каждому кластеру

Под данной таблице можно сказать, что кластеры поделились примерно по целевой переменной HeartDisease. Первый кластер показывает средние значения пациентов, которые в большинстве своём были признаны как люди с больным сердцем, второй кластер отвечает за здоровых людей. Однако в этих распределениях могут находиться здоровые люди среди больных, так и наоборот, но процент таких "выбросов" довольно низок для данного датасета. Также по данной таблице можно сказать, что у больных людей все значения выше, чем у здоровых, кроме максимальной частоты сердцебиения, что выглядит довольно правдоподобно для данного контекста.

Были просчитаны метрики для получившихся кластеров

silhouette\_avg = silhouette\_score(scaled, labels)  
inertia = kmeans.inertia\_   
  
print(silhouette\_avg, inertia)

true = data['HeartDisease']  
  
pred = labels

ari = metrics.adjusted\_rand\_score(true, pred)  
hs = metrics.homogeneity\_score(true, pred)  
cs = metrics.completeness\_score(true, pred)  
vms = metrics.v\_measure\_score(true, pred)  
  
print(ari,hs,cs,vms)

Значение метрики ARI 0,56 указывает на то, что кластеры не идеально соответствуют истинным группам, но всё же есть некоторая согласованность. Метрики гомогенности и полноты указывают на то, что в кластерах находятся объекты, которые принадлежат одному кластеру, но не полностью соответствуют одной истинной группе. Это стало ясно по прошлой таблице, так как в обоих кластерах находятся элементы, которые должны относиться к другому кластеру. Метрика V-measure является средним между гомогенностью и полнотой, поэтому у него такое же среднее значение

Далее была сделана дендрограмма по тем же значениям. Дендрограмма — это визуальное представление результатов иерархической кластеризации. Она показывает, как данные объединяются в кластеры на разных уровнях иерархии. Дендрограмма была выполнена с помощью библиотек scipy и sklearn (см. Рисунок 18)

from scipy.cluster.hierarchy import dendrogram, linkage  
from sklearn.cluster import AgglomerativeClustering  
  
linked = linkage(scaled, method='ward')  
  
plt.figure(figsize=(10, 7))  
dendrogram(linked, truncate\_mode='lastp', p=30, show\_leaf\_counts=True)  
plt.title('Dendrogram')  
plt.xlabel('Data Points')  
plt.ylabel('Distance')  
plt.show()  
  
agg\_clustering = AgglomerativeClustering(n\_clusters=2, linkage='ward')  
agg\_labels = agg\_clustering.fit\_predict(scaled)  
  
num\_data['Agg\_Cluster'] = agg\_labels  
  
agg\_cluster\_means = num\_data.groupby('Agg\_Cluster').mean()  
  
agg\_silhouette\_avg = silhouette\_score(scaled, agg\_labels)  
print(agg\_silhouette\_avg)

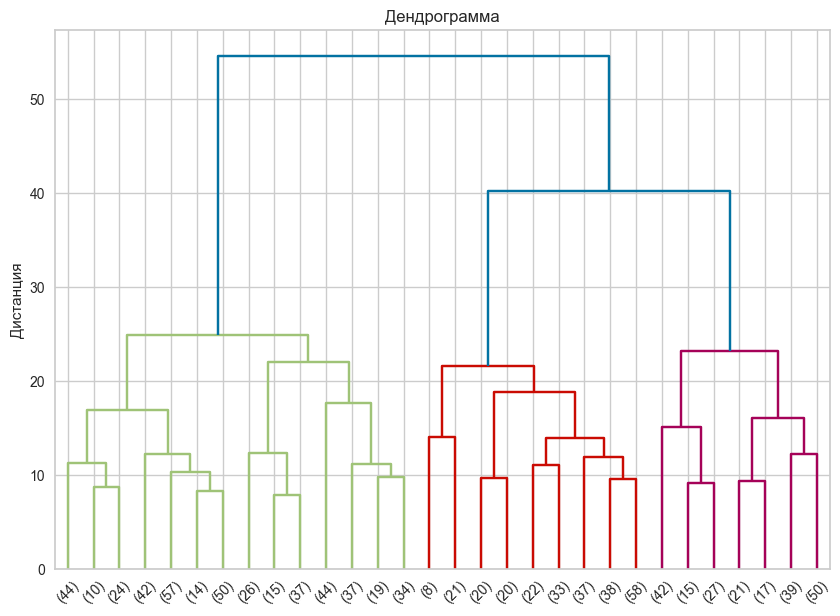


Рисунок 18 - Дендрограмма

Дендрограмма показывает структуру иерархической кластеризации. Длинная горизонтальная линия на уровне расстояния около 50 указывает на разделение данных на два основных кластера. Если спуститься ниже, на уровень около 30, можно выделить три кластера. Зеленый кластер (слева) демонстрирует высокую однородность, так как его объекты объединяются на малых расстояниях, в то время как красный и фиолетовый кластеры (справа) имеют большее разнообразие. Это указывает на возможность дальнейшего деления данных.

По результатам работы можно заключить, что оба метода кластеризации, K-Means и иерархическая кластеризация, справились с задачей, но качество разделения данных оказалось слабым из-за сложности структуры данных и перекрытия кластеров. K-Means показал более высокое значение силуэтного коэффициента (0.236) по сравнению с иерархической кластеризацией (0.207), что указывает на его лучшее качество разделения в данной задаче. Однако иерархическая кластеризация предоставила больше информации о структуре данных за счёт дендрограммы, что позволяет визуально анализировать иерархию кластеров.

Вывод

В ходе работы были выполнены два больших этапа: кластеризация сгенерированных данных, а также кластеризация реального медицинского датасета 1heart.csv.

На первом этапе была реализована функция для генерации набора данных с заданным количеством объектов в группах и фиксируемым зерном случайных значений. Созданные данные были разделены на тренировочные и тестовые выборки, после чего выполнена их стандартизация. Метод K-Means был применён для кластеризации тренировочных данных на три кластера. Метки кластеров и центры визуализированы, а для тестовых данных была выполнена дополнительная кластеризация с последующей визуализацией. Коэффициент силуэта, рассчитанный для n\_clusters = 3, 2 и 4, показал, что лучшее качество кластеризации достигается при трёх кластерах. Метод локтя подтвердил это, демонстрируя минимальную инерцию при разумном числе кластеров.

На втором этапе проводилась кластеризация медицинских данных. Они были очищены от пропусков, некорректных значений и выбросов, а числовые признаки масштабированы. Метод K-Means показал силуэтный коэффициент 0.236, а иерархическая кластеризация — 0.207, что указывает на слабую отделимость кластеров в обоих случаях. Дендрограмма позволила проанализировать структуру данных, показывая возможность разбиения на два или три кластера. Метрики, сравнивающие истинные метки кластеров (HeartDisease) с предсказанными, указали на умеренное соответствие (Adjusted Rand Index — 0.56)