

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №41

ЗАЩИЩЕНА С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

К.т.н., доц.

Е.Л. Турнецкая

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

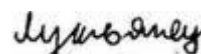
Кластеризация данных

по курсу: Методология и технология проектирования информационных
систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

М320М



П. Е. Лукьянец

подпись, дата

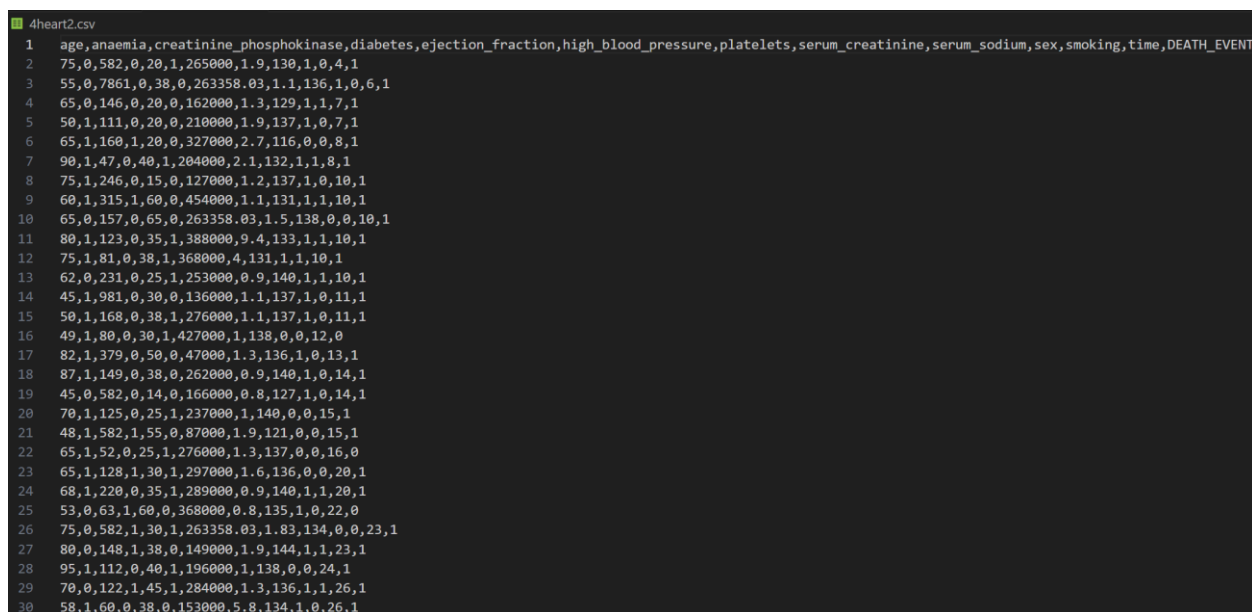
инициалы, фамилия

Санкт-Петербург 2023

Цель работы: изучение связи между признаками двумерного набора данных.

Ход выполнения работы

Для работы был выбран четвёртый датасет из списка под названием «4 heart2». В данном датасете представлена информация о пациентах с сердечными проблемами: возраст, наличие анемии, значение Креатинкиназа, наличие диабета, повышенного давления, процент фракции выброса, количество тромбоцитов, пол, уровень креатинин сыворотки, уровень натрия сыворотки, курит пациент или нет, время - период наблюдения в днях и умер ли пациент в течение периода наблюдения. Чтобы начать работу, импортируем библиотеку, а затем считываем CSV файл. Выведем первые 20 строк с помощью метода `head`. Часть данных этого датасета представлена на рисунке 1.



```
1 age,anaemia,creatinine_phosphokinase,diabetes,ejection_fraction,high_blood_pressure,platelets,serum_creatinine,serum_sodium,sex,smoking,time,DEATH_EVENT
2 75,0,582,0,20,1,265000,1.9,130,1,0,4,1
3 55,0,7861,0,38,0,263358.03,1.1,136,1,0,6,1
4 65,0,146,0,20,0,162000,1.3,129,1,1,7,1
5 50,1,111,0,20,0,210000,1.9,137,1,0,7,1
6 65,1,160,1,20,0,327000,2.7,116,0,0,8,1
7 90,1,47,0,40,1,204000,2.1,132,1,1,8,1
8 75,1,246,0,15,0,127000,1.2,137,1,0,10,1
9 60,1,315,1,60,0,454000,1.1,131,1,1,10,1
10 65,0,157,0,65,0,263358.03,1.5,138,0,0,10,1
11 80,1,123,0,35,1,388000,0.4,133,1,1,10,1
12 75,1,81,0,38,1,368000,4,131,1,1,10,1
13 62,0,231,0,25,1,253000,0.9,140,1,1,10,1
14 45,1,981,0,30,0,136000,1.1,137,1,0,11,1
15 50,1,168,0,38,1,276000,1.1,137,1,0,11,1
16 49,1,80,0,30,1,427000,1,138,0,0,12,0
17 82,1,379,0,50,0,47000,1.3,136,1,0,13,1
18 87,1,149,0,38,0,262000,0.9,140,1,0,14,1
19 45,0,582,0,14,0,166000,0.8,127,1,0,14,1
20 70,1,125,0,25,1,237000,1,140,0,0,15,1
21 48,1,582,1,55,0,87000,1.9,121,0,0,15,1
22 65,1,52,0,25,1,276000,1.3,137,0,0,16,0
23 65,1,128,1,30,1,297000,1.6,136,0,0,20,1
24 68,1,220,0,35,1,289000,0.9,140,1,1,20,1
25 53,0,63,1,60,0,368000,0.8,135,1,0,22,0
26 75,0,582,1,30,1,263358.03,1.83,134,0,0,23,1
27 80,0,148,1,38,0,149000,1.9,144,1,1,23,1
28 95,1,112,0,40,1,196000,1,138,0,0,24,1
29 70,0,122,1,45,1,284000,1.3,136,1,1,26,1
30 58,1,60,0,38,0,153000,5.8,134,1,0,26,1
```

Рисунок 1 – используемый датасет

Для работы с ним использовалась библиотека Pandas.

Работа была выполнена при помощи Visual Studio Code, а также Jupyter Notebook.

Ссылка на GitHub репозиторий с файлами:
<https://github.com/NinjaCaratist/MTPIS>

Чтобы начать работу, импортируем библиотеку, а затем считываем CSV файл.

```
import pandas as pd

df = pd.read_csv("4heart2.csv")
```

Рисунок 2 – скриншот кода

Выведем первые 20 строк с помощью метода head. На рисунке 3 показан код, а на рисунке 4 – результат его работы.

```
#2
print(df.head(20))
```

Рисунок 3 – скриншот кода

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction
0	75.0	0	582	0	20
1	55.0	0	7861	0	38
2	65.0	0	146	0	20
3	50.0	1	111	0	20
4	65.0	1	160	1	20
5	90.0	1	47	0	40
6	75.0	1	246	0	15
7	60.0	1	315	1	60
8	65.0	0	157	0	65
9	80.0	1	123	0	35
10	75.0	1	81	0	38
11	62.0	0	231	0	25
12	45.0	1	981	0	30
13	50.0	1	168	0	38
14	49.0	1	80	0	30
15	82.0	1	379	0	50
16	87.0	1	149	0	38
17	45.0	0	582	0	14
18	70.0	1	125	0	25
19	48.0	1	582	1	55

	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex
0	1	265000.00	1.9	130	1
1	0	263358.03	1.1	136	1
2	0	162000.00	1.3	129	1
3	0	210000.00	1.9	137	1
4	0	327000.00	2.7	116	0
5	1	204000.00	2.1	132	1
6	0	127000.00	1.2	137	1
7	0	454000.00	1.1	131	1
8	0	263358.03	1.5	138	0
9	1	388000.00	9.4	133	1
10	1	368000.00	4.0	131	1
11	1	253000.00	0.9	140	1
12	0	136000.00	1.1	137	1
13	1	276000.00	1.1	137	1
14	1	427000.00	1.0	138	0
15	0	47000.00	1.3	136	1
16	0	262000.00	0.9	140	1
17	0	166000.00	0.8	127	1
18	1	237000.00	1.0	140	0
19	0	87000.00	1.9	121	0

	smoking	time	DEATH_EVENT
0	0	4	1
1	0	6	1
2	1	7	1
3	0	7	1
4	0	8	1
5	1	8	1
6	0	10	1
7	1	10	1
8	0	10	1
9	1	10	1
10	1	10	1
11	1	10	1
12	0	11	1
13	0	11	1
14	0	12	0
15	0	13	1
16	0	14	1
17	0	14	1
18	0	15	1
19	0	15	1

Рисунок 4 – результат вывода

Как уже было сказано, данная таблица содержит информацию о пациентах с сердечными заболеваниями. Предметная область – медицина.

Опишем колонки подробнее:

1. возраст: возраст пациента (лет)
2. -анемия: снижение количества эритроцитов или гемоглобина (логическое значение)
3. -высокое кровяное давление: если у пациента гипертония (логическое значение)
4. креатининфосфокиназа (КФК): уровень фермента КФК в крови (мкг/л)
5. диабет: если у пациента диабет (логическое значение)
6. фракция выброса: процент крови, покидающей сердце при каждом сокращении (в процентах)
7. тромбоциты: тромбоциты в крови (килотромбоциты/ мл)
8. пол: женщина или мужчина (бинарный)
9. креатинин сыворотки: уровень креатинина сыворотки в крови (мг/дл)
10. натрий сыворотки: уровень натрия сыворотки в крови (мэкв/л)
11. курение: если пациент курит или нет (логическое)
12. время: период наблюдения (дни)
13. событие смерти: если пациент умер в течение периода наблюдения (логическое значение)

Теперь с помощью метода «.info» оценим данные. Этот метод возвращает название столбцов, типы данных, количество ненулевых объектов каждом столбце. Этот метод возвращает название столбцов, типы данных, количество ненулевых объектов каждом столбце. Результат работы метода представлен на рисунке 5.

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   age                                   299 non-null    float64
1   anaemia                              299 non-null    int64
2   creatinine_phosphokinase             299 non-null    int64
3   diabetes                             299 non-null    int64
4   ejection_fraction                   299 non-null    int64
5   high_blood_pressure                 299 non-null    int64
6   platelets                           299 non-null    float64
7   serum_creatinine                     299 non-null    float64
8   serum_sodium                        299 non-null    int64
9   sex                                  299 non-null    int64
10  smoking                             299 non-null    int64
11  time                                299 non-null    int64
12  DEATH_EVENT                         299 non-null    int64
dtypes: float64(3), int64(10)
memory usage: 30.5 KB

```

Рисунок 5 – результат вывода

Теперь выведем на экран названия столбцов с помощью `df.columns`.
Названия всех колонок приемлимы.

```
print(df.columns)
```

Рисунок 6 – скриншот кода

```

Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
      'ejection_fraction', 'high_blood_pressure', 'platelets',
      'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
      'DEATH_EVENT'],
      dtype='object')

```

Рисунок 7 – результат вывода

Найдём пропуски и устраним их. При помощи метода «`isna`» найдём все пропуски в таблице, а также при помощи `sum` выведем количество пропусков в каждом столбце. Пропусков в данном наборе данных нет. Код представлен на рисунке 8.

```
print(df.isna().sum())
```

Рисунок 8 – скриншот кода

```
age 0
anaemia 0
creatinine_phosphokinase 0
diabetes 0
ejection_fraction 0
high_blood_pressure 0
platelets 0
serum_creatinine 0
serum_sodium 0
sex 0
smoking 0
time 0
DEATH_EVENT 0
dtype: int64
```

Рисунок 9 – результат вывода

Проверим данные на наличие дубликатов – рисунок 10. Полностью повторяющихся строк нет, результат представлен на рисунке 11. Также выведем уникальные значения каждого столбца, они все также в порядке.

```
print("Количество дубликатов: " + str(df.duplicated().sum()))
df.info()
```

Рисунок 10 – скриншот кода

```
Количество дубликатов: 0
```

Рисунок 11 – результат вывода

Проверим все ли типы данных соответствуют действительности. Все столбцы, кроме возраста соответствуют своему типу. Поэтому при помощи метода «astype» изменяем тип на int. Выведем информацию.

```
df = df.astype({"age": 'int'})
df.info()
print(df['age'])
```

Рисунок 12 – скриншот кода

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   age                                    299 non-null    int32
1   anaemia                               299 non-null    int64
2   creatinine_phosphokinase              299 non-null    int64
3   diabetes                              299 non-null    int64
4   ejection_fraction                     299 non-null    int64
5   high_blood_pressure                    299 non-null    int64
6   platelets                             299 non-null    float64
7   serum_creatinine                       299 non-null    float64
8   serum_sodium                           299 non-null    int64
9   sex                                    299 non-null    int64
10  smoking                                299 non-null    int64
11  time                                   299 non-null    int64
12  DEATH_EVENT                            299 non-null    int64
dtypes: float64(2), int32(1), int64(10)
memory usage: 29.3 KB
0      75
1      55
2      65
3      50
4      65
..
294    62
295    55
296    45
297    45
298    50
Name: age, Length: 299, dtype: int32
```

Рисунок 13 – результат вывода

Выполним кластеризацию объектов. Для начала, чтобы веса можно было сравнивать, признаки приводят к единому масштабу. Стандартизация приводит значение признака именно к такому виду: для каждого наблюдения из исходного значения признака вычитается среднее, а полученная разность делится на стандартное отклонение. На практике необязательно знать, по какой формуле исходные признаки получают значение из этих интервалов. Тем не менее, для решения задач методами линейной регрессии и кластеризации данные обязательно стандартизируют. В sklearn для

нормализации и стандартизации данных в модуле `preprocessing` есть готовые классы `MinMaxScaler()` и `StandardScaler()` соответственно.

Они напоминают модели: их также нужно обучать — показывать, какие значения принимает признак на примере обучающей выборки, — и только после этого применять к любым новым наблюдениям. Также нарисуем график со всеми точками для наглядности.

Оценим средние значения каждой характеристики, чтобы определить какие три характеристики повлияли более всего на выделения кластеров. А также нарисуем график распределения по значениям для каждой характеристики. Характеристики, которые принимают значение 1 или 0, можно объединить в одну группу, она и повлияла на выделение одного из кластеров. Также характеристика "platelets" или тромбоциты имеет очень большое среднее значение, что видно на графике. Эта характеристика также повлияла на выделение кластера. Третьей характеристикой можно выбрать "creatinine_phosphokinase", она также будет иметь влияние на выделение кластера.

```
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
print(df.mean(axis=0))
scaler = StandardScaler() # создаём объект класса scaler
scaler.fit(df) # обучаем стандартизатор
X_sc = scaler.transform(df) # преобразуем набор данных
n = 299
plt.plot([1]*n, df['age'], 'bo', label='age')
plt.plot([2]*n, df['anaemia'], 'co', label='anaemia')
plt.plot([3]*n, df['creatinine_phosphokinase'], 'mo', label='creatinine_phosphokinase')
plt.plot([4]*n, df['diabetes'], 'co', label='diabetes')
plt.plot([5]*n, df['ejection_fraction'], 'go', label='ejection_fraction')
plt.plot([6]*n, df['high_blood_pressure'], 'co', label='high_blood_pressure')
plt.plot([7]*n, df['platelets'], 'ro', label='platelets')
plt.plot([8]*n, df['serum_creatinine'], 'yo', label='serum_creatinine')
plt.plot([9]*n, df['serum_sodium'], 'ko', label='serum_sodium')
plt.plot([10]*n, df['sex'], 'co', label='sex')
plt.plot([11]*n, df['smoking'], 'co', label='smoking')
plt.plot([12]*n, df['time'], 'bo', label='time')
plt.plot([13]*n, df['DEATH_EVENT'], 'co', label='DEATH_EVENT')
plt.legend(loc=0)
plt.show()
```

Рисунок 14 – скриншот кода

```

age                60.833893
anaemia            0.431438
creatinine_phosphokinase  581.839465
diabetes           0.418060
ejection_fraction  38.083612
high_blood_pressure  0.351171
platelets          263358.029264
serum_creatinine   1.393880
serum_sodium       136.625418
sex                0.648829
smoking            0.321070
time              130.260870
DEATH_EVENT        0.321070
dtype: float64

```

Рисунок 15 – результат вывода

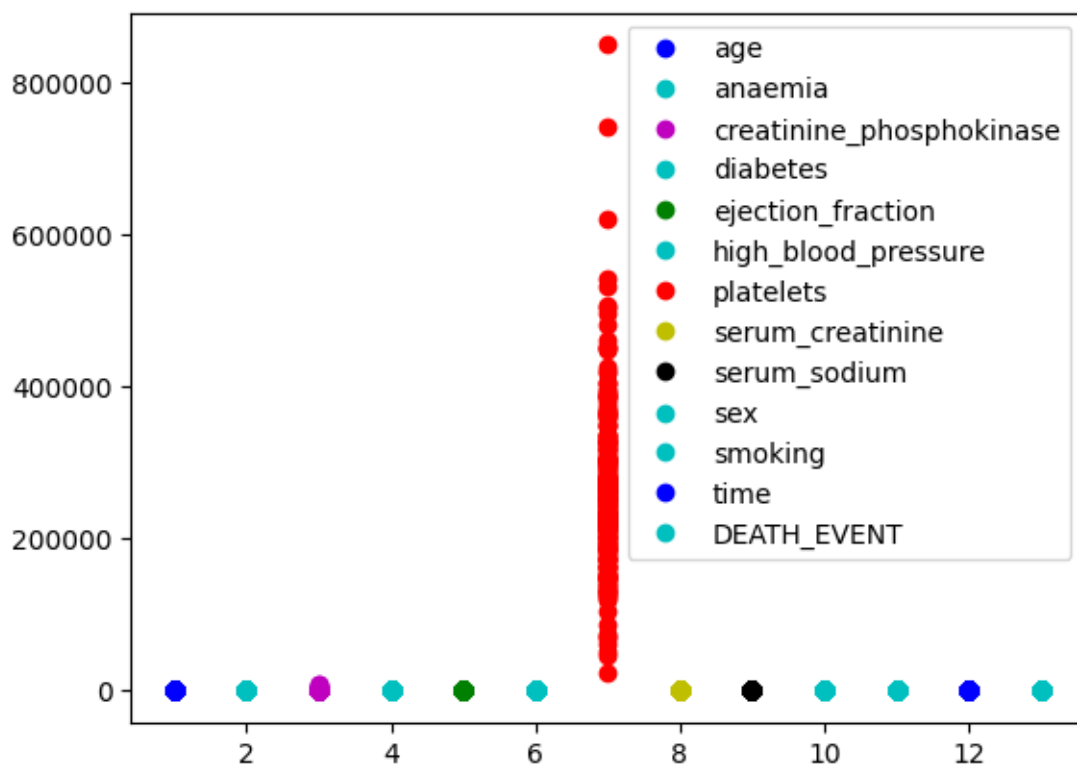


Рисунок 16 – построенный график

По дендрограмме можно визуальнo оценить, сколько кластеров должно быть. А также прикинуть расстояние, после которого мы перестаём объединять объекты. Сперва из модуля для иерархической кластеризации `hierarchy` импортируем классы модели кластеризации `linkage()` и `dendrogram()`. Передадим получившуюся стандартизированную таблицу в качестве

параметра функции `linkage()`. Чтобы диаграмма получилась показательной, лучше передать параметру `method` значение 'ward'.

```
from scipy.cluster.hierarchy import dendrogram, linkage
linked = linkage(X_sc, method = 'ward')
plt.figure(figsize=(15, 10))
dendrogram(linked, orientation='top')
plt.title('Hierarchial clustering for heart ')
plt.show()
```

Рисунок 17 – скриншот кода

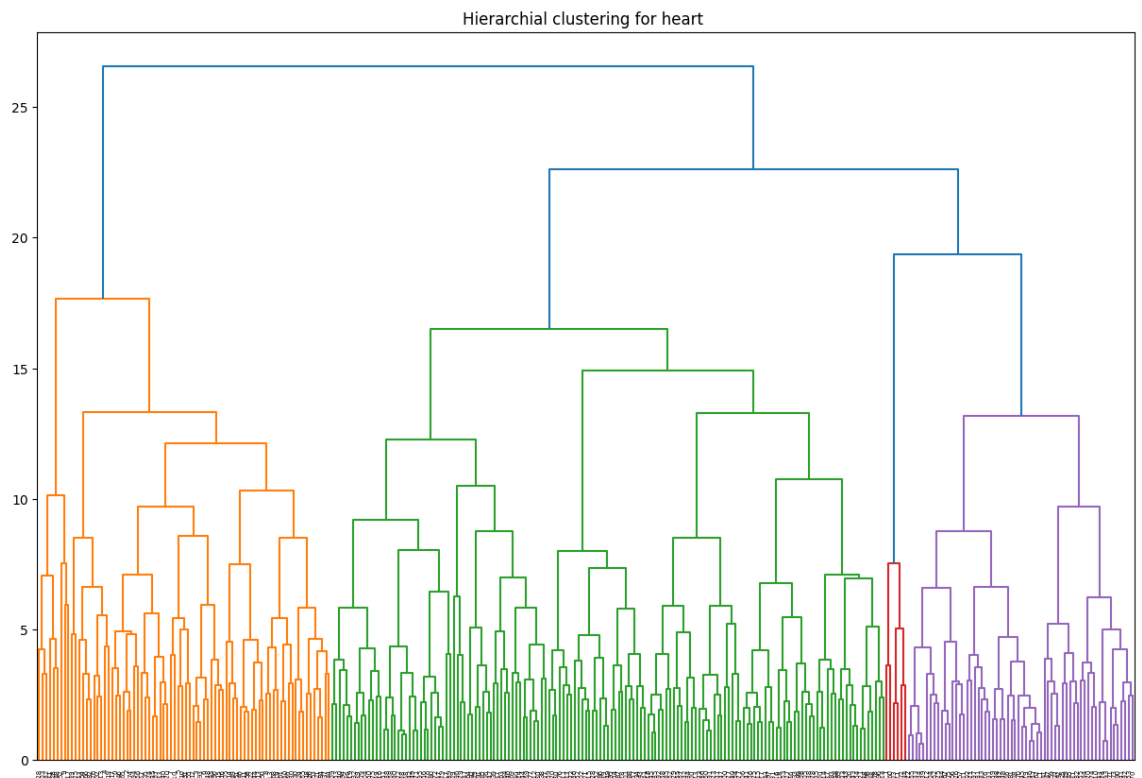


Рисунок 18 – построенная дендрограмма

Метрика силуэта показывает, насколько объект своего кластера похож на свой кластер больше, чем на чужой. На вход передаём нормализованную или стандартизованную матрицу признаков и метки, которые спрогнозировал алгоритм кластеризации, в виде списка. Значение метрики силуэта принимает значения от -1 до 1. Чем ближе к 1, тем качественнее кластеризация.

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
# обязательная стандартизация данных перед работой с алгоритмами
km = KMeans(n_clusters = 3, random_state=0) # задаём число кластеров, равное 5, и фиксируем
labels = km.fit_predict(X_sc) # применяем алгоритм к данным и формируем вектор кластеров
print(silhouette_score(X_sc, labels))
```

Рисунок 19 – скриншот кода

```
super()._check_params_vs_input(X, default_n_init=10)
0.10230814919678868
```

Рисунок 20 – построенные гистограммы

В отличие от задачи классификации или регрессии, в случае кластеризации сложнее выбрать критерий, с помощью которого было бы просто представить задачу кластеризации как задачу оптимизации. В случае kMeans распространен вот такой критерий – сумма квадратов расстояний от точек до центроидов кластеров, к которым они относятся. Для решения этого вопроса (выбора числа кластеров) часто пользуются такой эвристикой: выбирают то число кластеров, начиная с которого описанный функционал $J(C)$ падает "уже не так быстро". Также построим график при помощи цикла вычисления силуэта для нахождения оптимального числа кластеров. По графикам можно прийти к выводу, что это 4 кластера, как было и в дендрограмме.

```

import numpy as np
inertia = []
for k in range(1, 8):
    kmeans = KMeans(n_clusters=k, random_state=0,).fit(X_sc)
    inertia.append(np.sqrt(kmeans.inertia_))

plt.plot(range(1, 8), inertia, marker='s')
plt.xlabel('$k$')
plt.ylabel('$J(C_k)$')
plt.show()

inertia = []
for k in range(2, 8):
    kmeans = KMeans(n_clusters=k, random_state=0)
    labels = kmeans.fit_predict(X_sc)
    inertia.append(silhouette_score(X_sc, labels))

plt.plot(range(2, 8), inertia, marker='s')
plt.xlabel('$k$')
plt.ylabel('$Silhouette$')
plt.show()

```

Рисунок 21 – скриншот кода

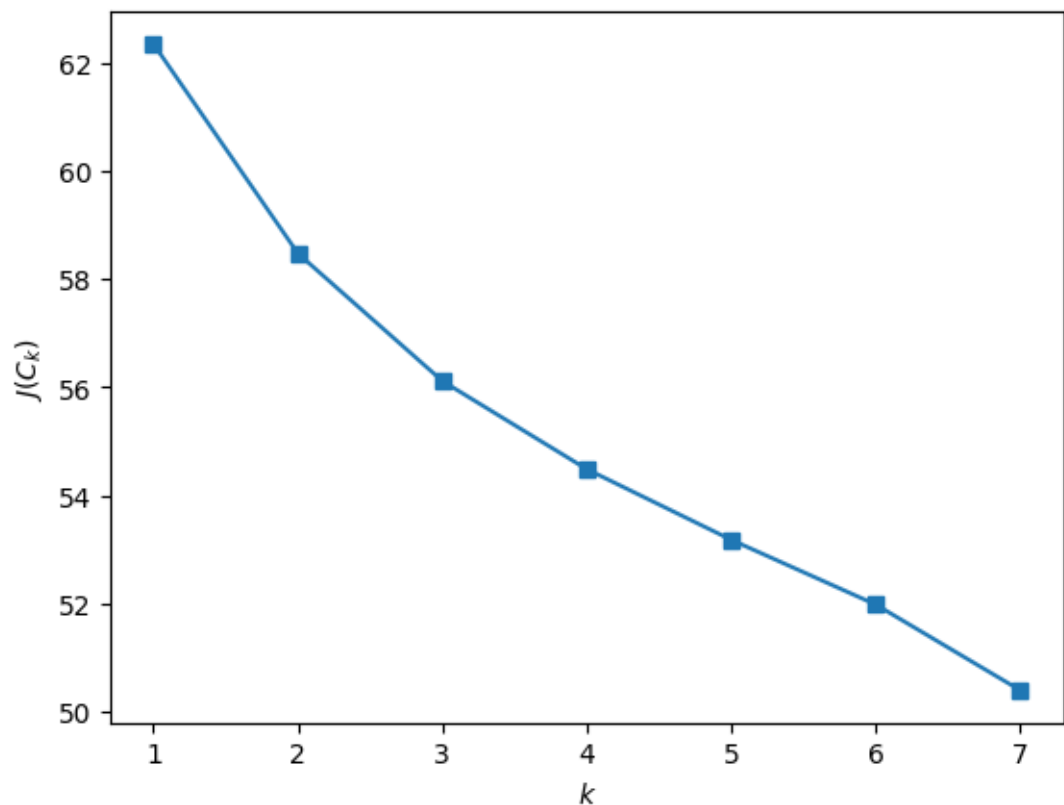


Рисунок 22 – построенный график

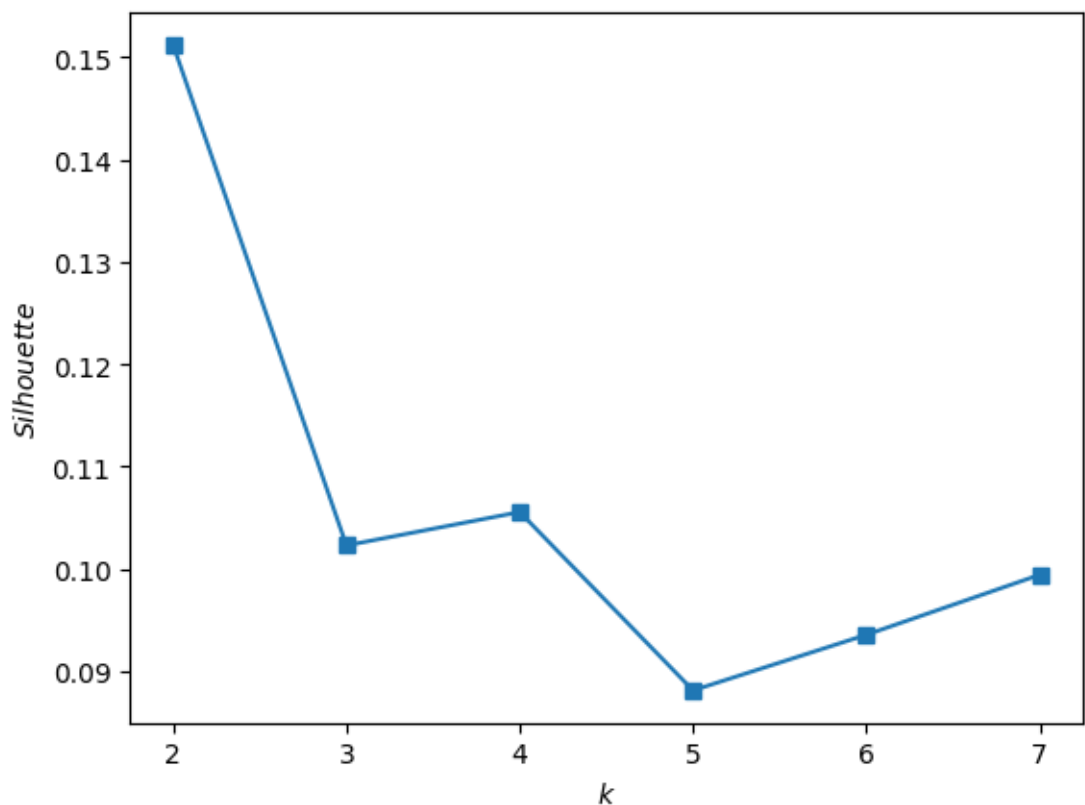


Рисунок 23 – построенный график

Вывод: Таким образом, в ходе выполнения лабораторной работы был выбран и описан выбранный датасет про пациентов с болезнью сердца, изучен интерфейс и возможности Jupyter Notebook, изучены базовые функции библиотеки Pandas и разработана программа, которая считывает данные, выводит о них информацию, удаляет дубликаты, пропуски, изменяет тип данных. Также были изучены методы кластеризации, стандартизированы данные, построена дендограмма и вычислен силуэт при помощи библиотеки K-means. Было вычислено, что оптимальное число кластеров - 4.