Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Прикладные интеллектуальные системы и экспертные системы»

Кластеризация данных

Студент Коровайцев А.А.

Группа М-ИАП-23-1

Руководитель Кургасов В.В.

Доцент

Липецк 2023 г.

Цель работы

Получить практические навыки решения задачи кластеризации фактографических данных в среде Jupiter Notebook. Научиться настраивать параметры методов и оценивать точность полученного разбиения.

Задание кафедры

1) Загрузить выборки согласно варианту задания.

2) Отобразить данные на графике в пространстве признаков. Поскольку решается задача кластеризации, то подразумевается, что априорная информация о принадлежности каждого объекта истинному классу неизвестна, соответственно, на данном этапе все объекты на графике должны отображаться одним цветом, без привязки к классу.

3) Провести иерархическую кластеризацию выборки, используя разные способы вычисления расстояния между кластерами: расстояние ближайшего соседа (single), дальнего соседа (complete), Уорда (Ward). Построить дендрограммы для каждого способа. Размер графика должен быть подобран таким образом, чтобы дендрограмма хорошо читалась.

4) Исходя из дендрограмм выбрать лучший способ вычисления расстояния между кластерами.

5) Для выбранного способа, исходя из дендрограммы, определить количество кластеров в имеющейся выборке. Отобразить разбиение на кластеры и центроиды на графике в пространстве признаков (объекты одного кластера должны отображаться одним и тем же цветом, центроиды всех кластеров – также одним цветом, отличным от цвета кластеров).

6) Рассчитать среднюю сумму квадратов расстояний до центроида, среднюю сумму межкластерных расстояний для данного разбиения. Сделать вывод о качестве разбиения.

7) Провести кластеризацию выборки методом k-средних. для k [1, 10].

8) Сформировать три графика: зависимость средней суммы квадратов расстояний до центроида, средней суммы средних внутрикластерных расстояний и средней суммы межкластерных расстояний от количества кластеров. Исходя из результатов, выбрать оптимальное количество кластеров.

9) Составить сравнительную таблицу результатов разбиения иерархическим методом и методом k-средних.

Вариант №7

n_samples = 100

Вид классов: classification

Random_state = 55

class_sep = 1.5

Для всех вариантов:

n_features = 2

n_redundant = 0

n_informative = 2

n_cluster_per_class = 1

n_classes = 4

Ход работы

Загрузим выборки согласно варианту, код для генерации данных представлен на рисунке 1.

```
X, y = make_classification(n_samples=100,
                           n_features=2,
                           n_redundant=0,
                           n_informative=2,
                           n_clusters_per_class=1,
                           n_classes=4,
                           random_state=55,
                           class_sep=1.5)
```

Рисунок 1 – Код для генерации данных

Отобразим на графике сгенерированные данные, для этого воспользуемся библиотекой matplotlib.pyplot. Полученный график представлен на рисунке 2.

```
plt.scatter(X[:, 0], X[:, 1])
```

```
<matplotlib.collections.PathCollection at 0x14507c790>
```

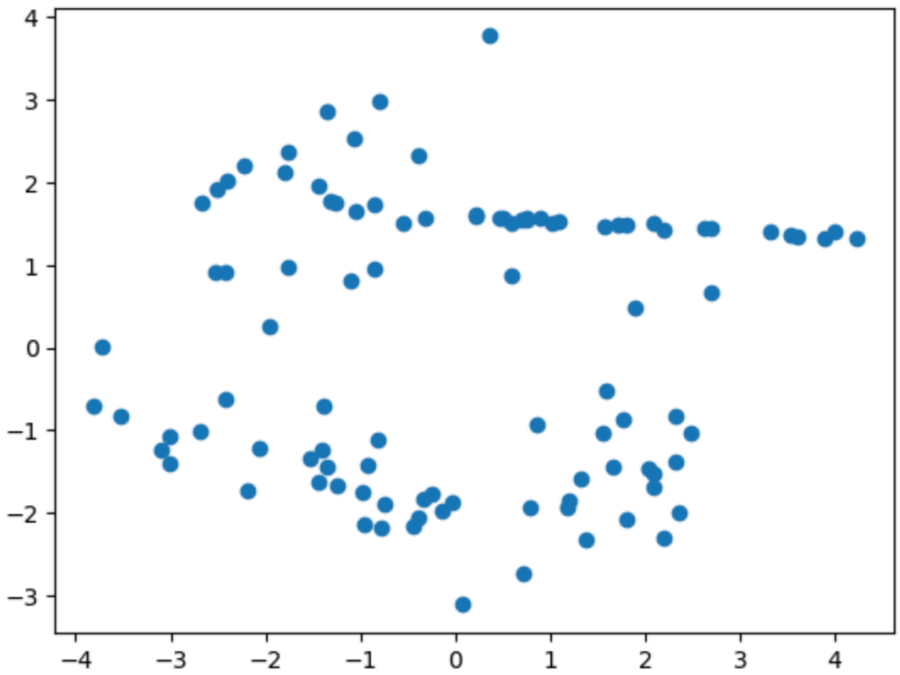

Рисунок 2 – Визуализация сгенерированных данных

Воспользуемся иерархической кластеризацией выборки с использованием различных методов вычисления расстояния.

Метод вычисления расстояния ближайшего соседа (single) и код для этого представлен на рисунке 3. Полученная дендограмма представлена на рисунке 4.

```
1 mergings_single = linkage(X, method='single')
2 mergings_single
```

```
array([[5.50000000e+01, 6.80000000e+01, 1.52210151e-02, 2.00000000e+00],
       [3.00000000e+00, 4.30000000e+01, 3.67553174e-02, 2.00000000e+00],
       [3.80000000e+01, 7.20000000e+01, 4.96617655e-02, 2.00000000e+00],
       [2.60000000e+01, 9.50000000e+01, 5.48418269e-02, 2.00000000e+00],
       [1.60000000e+01, 1.01000000e+02, 5.89391463e-02, 3.00000000e+00],
       [3.60000000e+01, 6.20000000e+01, 6.55950412e-02, 2.00000000e+00],
       [2.30000000e+01, 8.60000000e+01, 7.32334192e-02, 2.00000000e+00],
       [4.20000000e+01, 5.10000000e+01, 7.68651510e-02, 2.00000000e+00],
       [1.20000000e+01, 3.50000000e+01, 8.23646761e-02, 2.00000000e+00],
       [7.80000000e+01, 8.40000000e+01, 8.60879071e-02, 2.00000000e+00],
       [1.90000000e+01, 5.90000000e+01, 8.95418736e-02, 2.00000000e+00],
       [4.90000000e+01, 1.02000000e+02, 9.07691161e-02, 3.00000000e+00],
       [5.20000000e+01, 5.60000000e+01, 1.08135006e-01, 2.00000000e+00],
       [7.00000000e+00, 9.80000000e+01, 1.08767626e-01, 2.00000000e+00],
       [7.90000000e+01, 9.40000000e+01, 1.15887692e-01, 2.00000000e+00],
       [1.04000000e+02, 1.11000000e+02, 1.17926478e-01, 6.00000000e+00],
```

Рисунок 3 – Вычисленные расстояния ближайшего соседа (single)

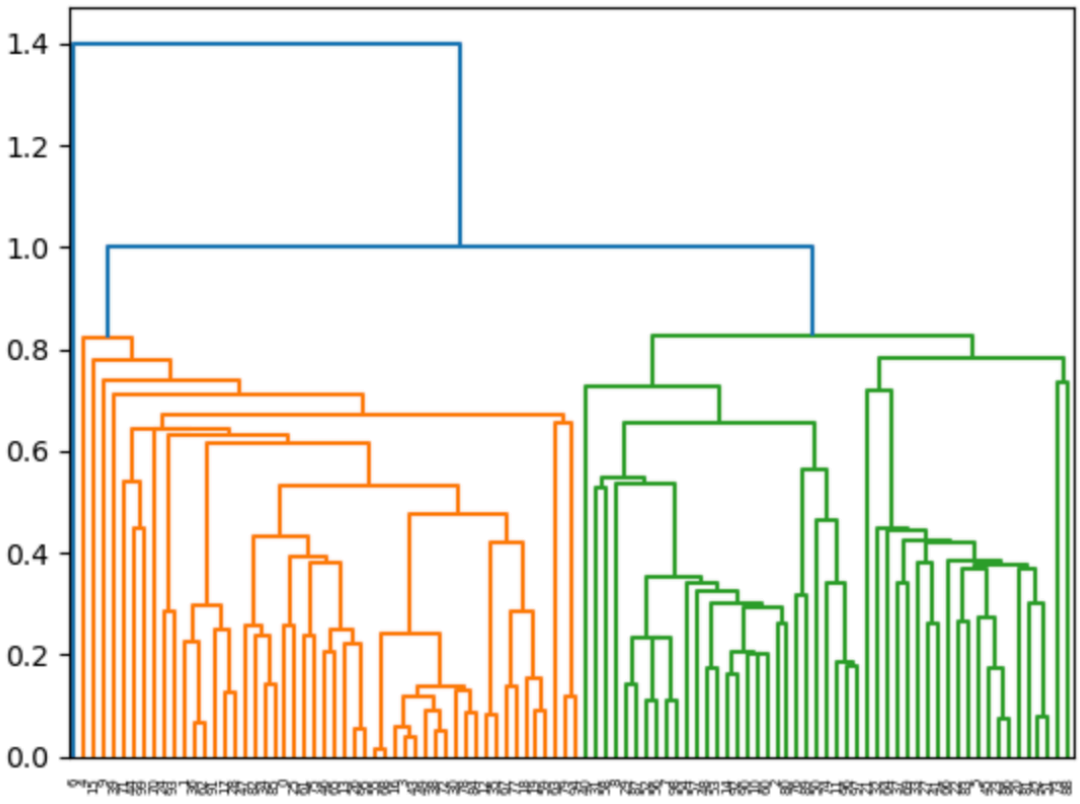

Рисунок 4 – Дендограмма для расстояния ближайшего соседа (single)

Метод вычисления расстояния дальнего соседа (complete) и код для этого представлен на рисунке 5. Полученная дендограмма представлена на рисунке 6.

```
1 mergings_complete = linkage(X, method='complete')
2 mergings_complete
```

```
array([[5.50000000e+01, 6.80000000e+01, 1.52210151e-02, 2.00000000e+00],
       [3.00000000e+00, 4.30000000e+01, 3.67553174e-02, 2.00000000e+00],
       [3.80000000e+01, 7.20000000e+01, 4.96617655e-02, 2.00000000e+00],
       [2.60000000e+01, 9.50000000e+01, 5.48418269e-02, 2.00000000e+00],
       [1.60000000e+01, 1.01000000e+02, 6.04297320e-02, 3.00000000e+00],
       [3.60000000e+01, 6.20000000e+01, 6.55950412e-02, 2.00000000e+00],
       [2.30000000e+01, 8.60000000e+01, 7.32334192e-02, 2.00000000e+00],
       [4.20000000e+01, 5.10000000e+01, 7.68651510e-02, 2.00000000e+00],
       [1.20000000e+01, 3.50000000e+01, 8.23646761e-02, 2.00000000e+00],
       [7.80000000e+01, 8.40000000e+01, 8.60879071e-02, 2.00000000e+00],
       [1.90000000e+01, 5.90000000e+01, 8.95418736e-02, 2.00000000e+00],
       [5.20000000e+01, 5.60000000e+01, 1.08135006e-01, 2.00000000e+00],
       [7.00000000e+00, 9.80000000e+01, 1.08767626e-01, 2.00000000e+00],
       [7.90000000e+01, 9.40000000e+01, 1.15887692e-01, 2.00000000e+00],
       [1.70000000e+01, 2.80000000e+01, 1.24046354e-01, 2.00000000e+00],
       [4.90000000e+01, 1.02000000e+02, 1.36751150e-01, 3.00000000e+00],
       [6.70000000e+01, 7.70000000e+01, 1.38861079e-01, 2.00000000e+00],
       [3.40000000e+01, 8.50000000e+01, 1.39770591e-01, 2.00000000e+00],
       [2.90000000e+01, 8.70000000e+01, 1.42955875e-01, 2.00000000e+00],
       [1.40000000e+01, 9.20000000e+01, 1.63194672e-01, 2.00000000e+00],
       [4.80000000e+01, 5.30000000e+01, 1.71764333e-01, 2.00000000e+00],
```

Рисунок 5 – Вычисленные расстояния дальнего соседа (complete)

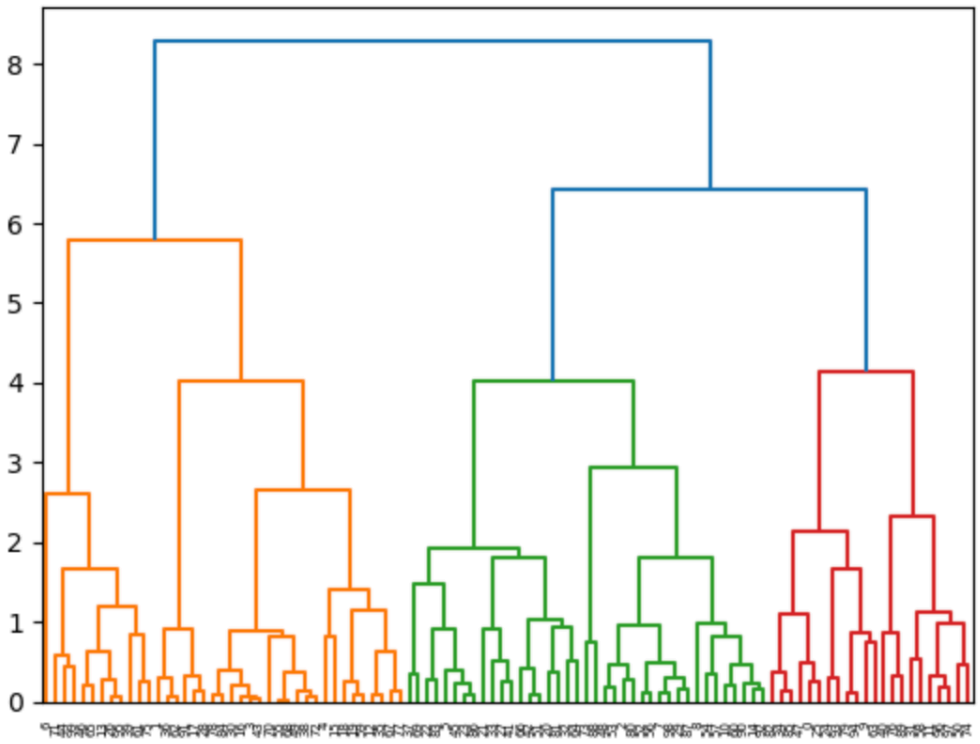

Рисунок 6 – Дендограмма для расстояния дальнего соседа (complete)

Метод вычисления расстояния Уорда (ward) и код для этого представлен на рисунке 7. Полученная дендограмма представлена на рисунке 8.

```
1 mergings_ward = linkage(X, method='ward')
2 mergings_ward
```

```
array([[5.50000000e+01, 6.80000000e+01, 1.52210151e-02, 2.00000000e+00],
       [3.00000000e+00, 4.30000000e+01, 3.67553174e-02, 2.00000000e+00],
       [3.80000000e+01, 7.20000000e+01, 4.96617655e-02, 2.00000000e+00],
       [2.60000000e+01, 9.50000000e+01, 5.48418269e-02, 2.00000000e+00],
       [1.60000000e+01, 1.01000000e+02, 6.55748874e-02, 3.00000000e+00],
       [3.60000000e+01, 6.20000000e+01, 6.55950412e-02, 2.00000000e+00],
       [2.30000000e+01, 8.60000000e+01, 7.32334192e-02, 2.00000000e+00],
       [4.20000000e+01, 5.10000000e+01, 7.68651510e-02, 2.00000000e+00],
       [1.20000000e+01, 3.50000000e+01, 8.23646761e-02, 2.00000000e+00],
       [7.80000000e+01, 8.40000000e+01, 8.60879071e-02, 2.00000000e+00],
       [1.90000000e+01, 5.90000000e+01, 8.95418736e-02, 2.00000000e+00],
       [5.20000000e+01, 5.60000000e+01, 1.08135006e-01, 2.00000000e+00],
       [7.00000000e+00, 9.80000000e+01, 1.08767626e-01, 2.00000000e+00],
       [7.90000000e+01, 9.40000000e+01, 1.15887692e-01, 2.00000000e+00],
       [1.70000000e+01, 2.80000000e+01, 1.24046354e-01, 2.00000000e+00],
       [4.90000000e+01, 1.02000000e+02, 1.30911584e-01, 3.00000000e+00],
       [6.70000000e+01, 7.70000000e+01, 1.38861079e-01, 2.00000000e+00],
       [3.40000000e+01, 8.50000000e+01, 1.39770591e-01, 2.00000000e+00],
       [2.90000000e+01, 8.70000000e+01, 1.42955875e-01, 2.00000000e+00],
       [1.40000000e+01, 9.20000000e+01, 1.63194672e-01, 2.00000000e+00],
       [4.80000000e+01, 5.30000000e+01, 1.71764333e-01, 2.00000000e+00],
```

Рисунок 7 – Вычисленные расстояния с помощью метода Уорда (ward)

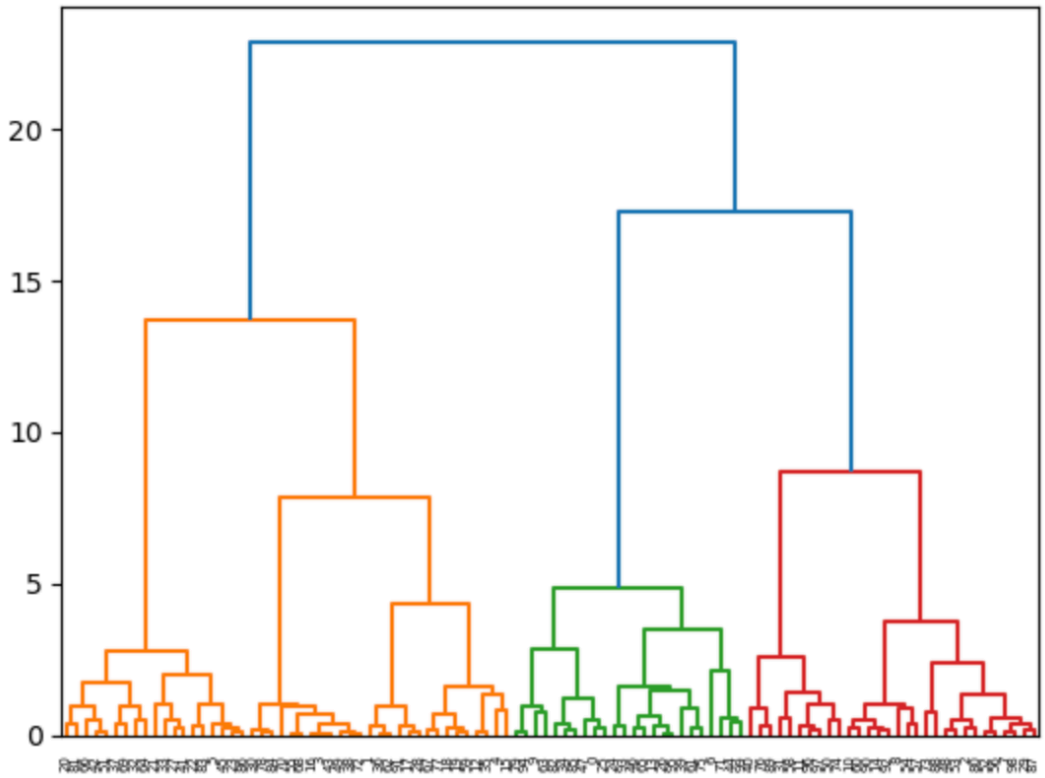

Рисунок 8 – Дендограмма для расстояния Уорда (ward)

Лучшим способом вычисления расстояния между кластерами является дальнего соседа (complete). Определим количество кластеров в имеющейся выборке с использованием данного способа и отобразим разбиение на кластеры и центроиды на графике в пространстве признаков. Полученное разбиение представлено на рисунке 9.

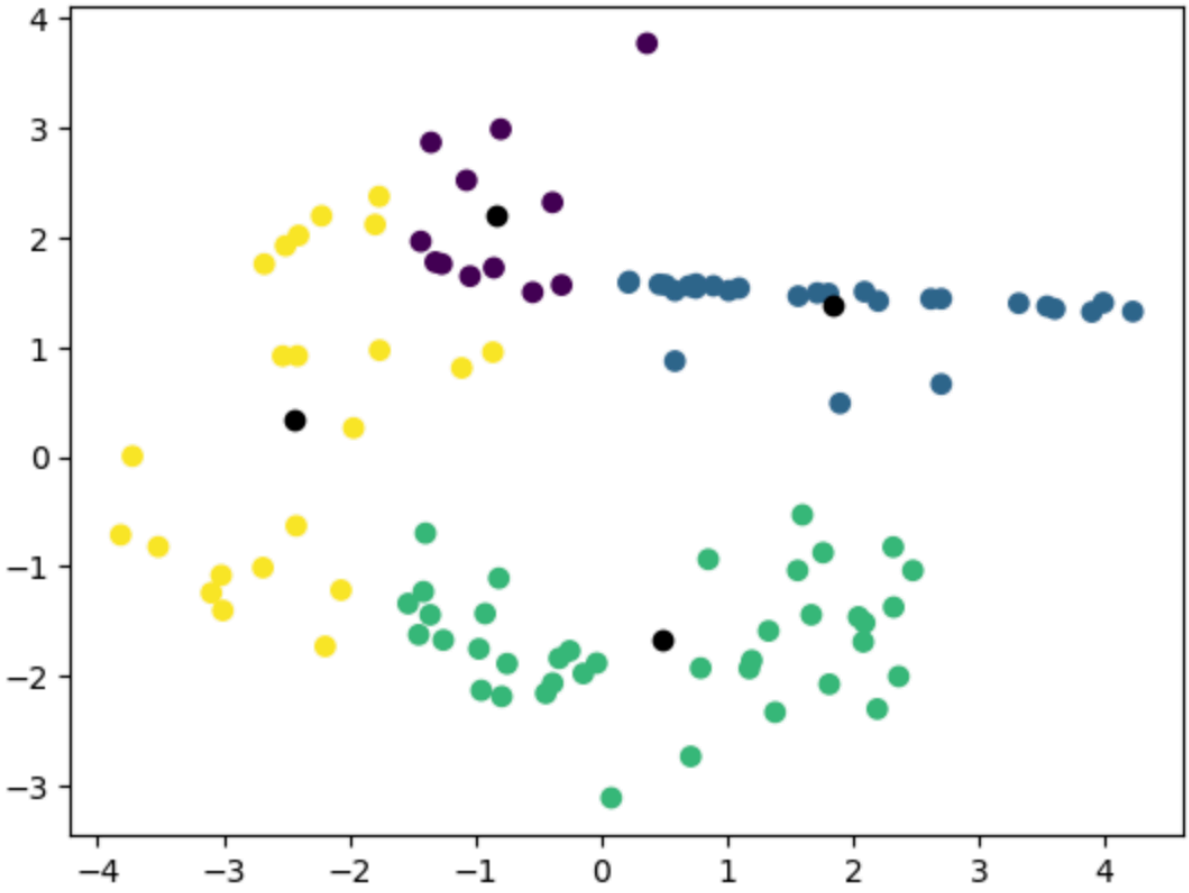

Рисунок 9 – График разбиения данных на кластеры

Рассчитаем среднюю сумму квадратов расстояний до центроида, среднюю сумму средних внутрикластерных расстояний и среднюю сумму межкластерных расстояний для данного разбиения.

Рассчитанное значение суммы квадратов расстояний до центроида и код для этого представлен на рисунке 10. Рассчитанное значение средних внутрикластерных расстояний и код для этого представлен на рисунке 11. Рассчитанное значение суммы межкластерных расстояний и код для этого представлен на рисунке 12.

```
# Сумма квадратов расстояний до центроида (inertia)
sum_sq_dist = np.zeros(4)
for i in range(1, 5):
    ix = np.where(T == i)
    sum_sq_dist[i - 1] = np.sum(euclidean_distances(*X[ix, :], [clusters[i - 1]]) ** 2)
sum_sq_dist = np.sum(sum_sq_dist) / 4
sum_sq_dist
```

```
46.816598779185185
```

Рисунок 10 – Сумма квадратов расстояний до центроида

```
# Сумма средних внутрикластерных расстояний
sum_avg_intercluster_dist = np.zeros(4)
for i in range(1, 5):
    ix = np.where(T == i)
    sum_avg_intercluster_dist[i - 1] = np.sum(euclidean_distances(*X[ix, :], [clusters[i - 1]]) ** 2) / len(*X[i
sum_avg_intercluster_dist = np.sum(sum_avg_intercluster_dist) / 4
sum_avg_intercluster_dist
```

```
1.708069605190309
```

Рисунок 11 – Сумма средних внутрикластерных расстояний

```
# Сумма межкластерных расстояний
sum_intercluster_dist = np.sum(euclidean_distances(clusters, clusters))
sum_intercluster_dist
```

```
41.22177711444111
```

Рисунок 12 – Сумма межкластерных расстояний

Проведем кластеризацию выборки методом k-средних для k = [1, 10], а после построим три графика: зависимость средней суммы квадратов расстояний до центроида, средней суммы средних внутрикластерных расстояний и средней суммы межкластерных расстояний от количества кластеров.

Код для расчета средней суммы квадратов расстояний до центроида представлен на рисунке 13, а построенный график на рисунке 14.

```
# Средней суммы квадратов расстояний до центроида
sum_sq_dist_avg = []
for it, kmean in enumerate(models):
    sum_sq_dist_avg.append(kmean.inertia_ / (it + 1))
sum_sq_dist_avg
```

```
[646.3560779131652,
 189.2156519671821,
 73.24361867072788,
 33.804278557782894,
 20.06091255860947,
 11.025843176738787,
 8.177674127178951,
 6.137711546018515,
 4.592302738432533,
 3.6597898015447585]
```

Рисунок 13 – Код для вычисления средней суммы квадратов расстояний до центроида

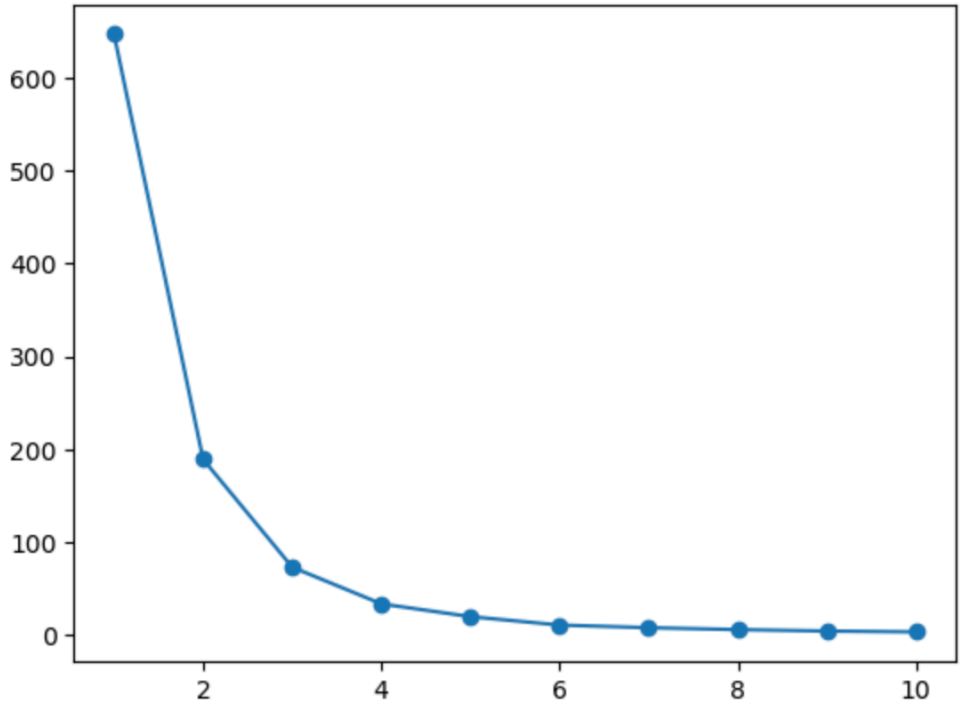

Рисунок 14 – График средней суммы квадратов расстояний до центроида

Код для расчета средней суммы средних внутрикластерных расстояний представлен на рисунке 15, а построенный график на рисунке 16.

```python
# Средней суммы средних внутрикластерных расстояний
new_centers = [kmean.cluster_centers_ for kmean in models]

sum_avg_intercluster_dist_avg = []
for k, kmean in enumerate(models):
    intercluster_sum = np.zeros(4)
    for i in range(4):
        ix = np.where(predicted_values[k] == i)
        if len(ix[0]) == 0:
            intercluster_sum[i - 1] = 0
        else:
            intercluster_sum[i - 1] = np.sum(euclidean_distances(*X[ix, :], [kmean.cluster_centers_[i - 1]]) **
    sum_avg_intercluster_dist_avg.append(np.sum(intercluster_sum) / (k + 1))
sum_avg_intercluster_dist_avg
```

```
[6.463560779131652,
 14.51201641242747,
 15.040195140980373,
 16.8298205683454,
 16.74816927126883,
 10.387129950728175,
 9.380204127185065,
 8.290426189777811,
 8.52405446812152,
 9.587885075326412]
```

Рисунок 15 – Код для вычисления средней суммы средних внутрикластерных расстояний

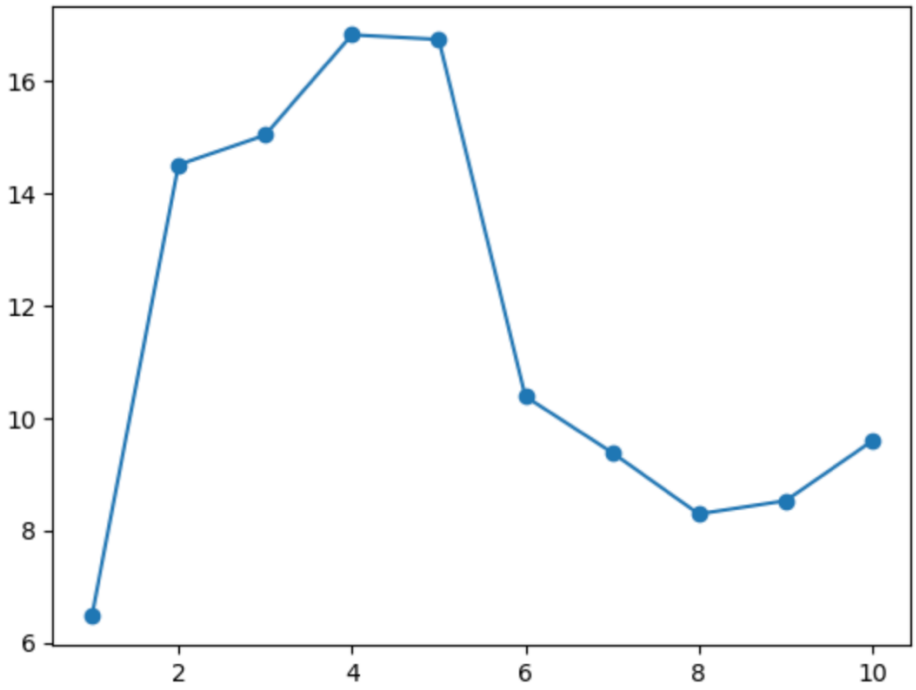

Рисунок 16 – График средней суммы средних внутрикластерных расстояний

Код для расчета средней суммы межкластерных расстояний от количества кластеров представлен на рисунке 17, а построенный график на рисунке 18.

```
# Средней суммы межкластерных расстояний от количества кластеров
sum_intercluster_dist_avg = []

for k, kmean in enumerate(models):
    value = np.sum(euclidean_distances(kmean.cluster_centers_, kmean.cluster_centers_))
    sum_intercluster_dist_avg.append(value / (k + 1))
sum_intercluster_dist_avg
```

```
[0.0,
 3.276303652616876,
 7.1540075209680625,
 10.929546574895468,
 14.574045430504816,
 18.582365920079223,
 22.39623761884071,
 25.52919592854157,
 29.677760874925777,
 33.95642870846736]
```

Рисунок 17 – Код для вычисления средней суммы межкластерных расстояний от количества кластеров

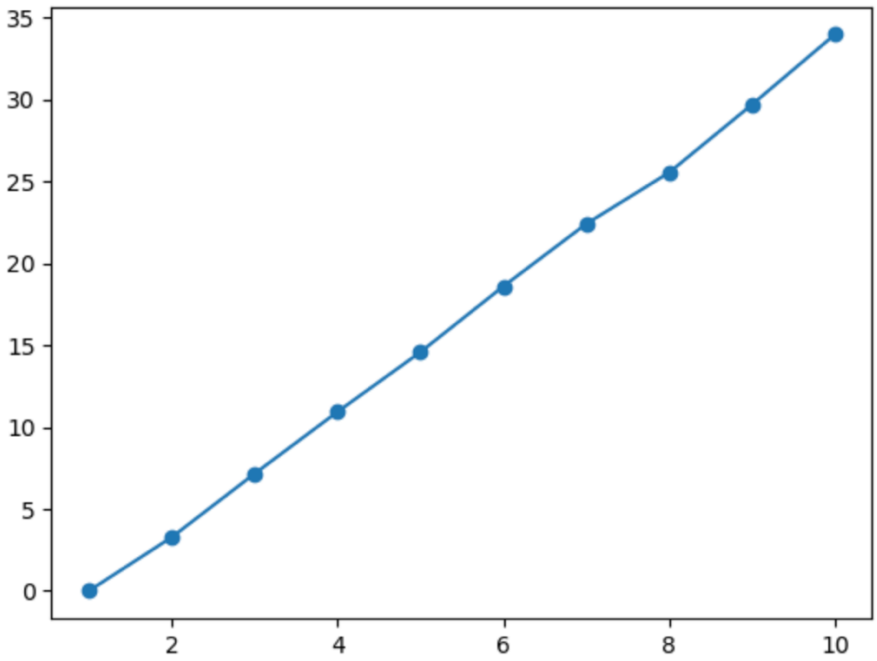

Рисунок 18 – График средней суммы межкластерных расстояний от количества кластеров

Как видно из рисунка 14 оптимальное количество кластеров составляет от двух до четырех.

Составим сравнительную таблицу для ранее описанных метрик качества моделей для иерархического метода и метода k-средних. Составленная таблица представлена на рисунках 19 – 21.

	Иерархический метод			Метод k-средних		
	Сумма квадратов расстояний до центроида	Сумма средних внутрикластерных расстояний	Сумма межкластерных расстояний	Сумма квадратов расстояний до центроида	Сумма средних внутрикластерных расстояний	Сумма межкластерных расстояний
0	46,81659878	1,708069605	41,22177711	646,3560779	6,463560779	0
1	46,81659878	1,708069605	41,22177711	189,215652	14,51201641	3,276303653
2	46,81659878	1,708069605	41,22177711	73,24361867	15,04019514	7,154007521
3	46,81659878	1,708069605	41,22177711	33,80427856	16,82982057	10,92954657
4	46,81659878	1,708069605	41,22177711	20,06091256	16,74816927	14,57404543
5	46,81659878	1,708069605	41,22177711	11,02584318	10,38712995	18,58236592
6	46,81659878	1,708069605	41,22177711	8,177674127	9,380204127	22,39623762
7	46,81659878	1,708069605	41,22177711	6,137711546	8,29042619	25,52919593
8	46,81659878	1,708069605	41,22177711	4,592302738	8,524054468	29,67776087
9	46,81659878	1,708069605	41,22177711	3,659789802	9,587885075	33,95642871

Рисунок 19 – Сводная таблица сравнения

Иерархический метод		
Сумма квадратов расстояний до центроида	Сумма средних внутрикластерных расстояний	Сумма межкластерных расстояний
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711
46,81659878	1,708069605	41,22177711

Рисунок 20 – Таблица данных для иерархического метода

Метод k-средних		
Сумма квадратов расстояний до центроида	Сумма средних внутрикластерных расстояний	Сумма межкластерных расстояний
646,3560779	6,463560779	0
189,215652	14,51201641	3,276303653
73,24361867	15,04019514	7,154007521
33,80427856	16,82982057	10,92954657
20,06091256	16,74816927	14,57404543
11,02584318	10,38712995	18,58236592
8,177674127	9,380204127	22,39623762
6,137711546	8,29042619	25,52919593
4,592302738	8,524054468	29,67776087
3,659789802	9,587885075	33,95642871

Рисунок 21 – Таблица данных для метода k-средних

Вывод

В ходе выполнения данной лабораторной работы мною были получены навыки кластеризации данных.

В рамках данной работы были применены различные методы кластеризации: иерархический метод и метод k-средних.

В ходе анализа метрик было определено, что оптимальное значение кластеров от двух до четырех.

Также была составлена таблица сравнения метрик иерархическим методом кластеризации и методом k-средних.

Приложение А

Исходный код

```
#!/usr/bin/env python
# coding: utf-8

# # Лабораторная работа №4
# ### Вариант №7
#
# ### Вид классов: `classification`
# ### Random state: `55`
# ### Class sep: `1.5`
#
# ### Для всех:
# ### `n_features = 2`
# ### `n_redundant = 0`
# ### `n_informative = 2`
# ### `n_clusters_per_class = 1`
# ### `n_classes = 4`
# ### `n_samples = 100`

# In[1]:


from sklearn.datasets import make_classification


# ### Загрузка выборки согласно варианту №7

# In[2]:


X, y = make_classification(n_samples=100,
                           n_features=2,
                           n_redundant=0,
                           n_informative=2,
                           n_clusters_per_class=1,
                           n_classes=4,
                           random_state=55,
                           class_sep=1.5)


# ### Отображение выборки на графике

# In[3]:


import matplotlib.pyplot as plt


# In[4]:


plt.scatter(X[:, 0], X[:, 1])


# ### Иерархическая кластеризация выборки

# In[5]:


from scipy.cluster.hierarchy import linkage, dendrogram
```

```
# ### Расстояние ближайшего соседа (single)

# In[6]:


mergings_single = linkage(X, method='single')
mergings_single


# In[7]:


dendrogram(mergings_single)
plt.show()


# ### Расстояние дальнего соседа (complete)

# In[8]:


mergings_complete = linkage(X, method='complete')
mergings_complete


# In[9]:


dendrogram(mergings_complete)
plt.show()


# ### Расстояние Уорда (Ward)

# In[10]:


mergings_ward = linkage(X, method='ward')
mergings_ward


# In[11]:


dendrogram(mergings_ward)
plt.show()


# ### Выбор лучшего разбиения

# In[12]:


mergings_complete = linkage(X, method='complete')
mergings_complete


# In[13]:


dendrogram(mergings_complete)
plt.show()
```

```
# In[14]:


import numpy as np

def update_cluster_centers(X, c):
    centers = np.zeros((4, 2))
    for i in range(1, 5):
        ix = np.where(c == i)
        centers[i - 1, :] = np.mean(X[ix, :], axis=1)
    return centers


# In[15]:


from scipy.cluster.hierarchy import fcluster


# In[16]:


T = fcluster(mergings_complete, 4, criterion='maxclust')
clusters = update_cluster_centers(X, T)
clusters


# In[17]:


plt.scatter(X[:, 0], X[:, 1], c=T)
plt.scatter(clusters[:, 0], clusters[:, 1], c='black')


# ### Вычисление характеристик

# In[18]:


from sklearn.metrics.pairwise import euclidean_distances


# In[34]:


# Сумма квадратов расстояний до центроида (inertia)
sum_sq_dist = np.zeros(4)
for i in range(1, 5):
    ix = np.where(T == i)
    sum_sq_dist[i - 1] = np.sum(euclidean_distances(*X[ix, :], [clusters[i -
1]]) ** 2)
sum_sq_dist = np.sum(sum_sq_dist) / 4
sum_sq_dist


# In[20]:


# Сумма средних внутрикластерных расстояний
sum_avg_intercluster_dist = np.zeros(4)
for i in range(1, 5):
    ix = np.where(T == i)
    sum_avg_intercluster_dist[i - 1] = np.sum(euclidean_distances(*X[ix, :],
```

```
[clusters[i - 1]]) ** 2) / len(*X[ix, :])
sum_avg_intercluster_dist = np.sum(sum_avg_intercluster_dist) / 4
sum_avg_intercluster_dist


# In[21]:


# Сумма межкластерных расстояний
sum_intercluster_dist = np.sum(euclidean_distances(clusters, clusters))
sum_intercluster_dist


# ### Кластеризация выборки методом k-средних

# In[22]:


from sklearn.cluster import KMeans


# In[23]:


models = []
predicted_values = []

for k in range(1, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X)
    models.append(kmeans)
    predicted_values.append(kmeans.predict(X))


# In[24]:


# Средней суммы квадратов расстояний до центроида
sum_sq_dist_avg = []
for it, kmean in enumerate(models):
    sum_sq_dist_avg.append(kmean.inertia_ / (it + 1))
sum_sq_dist_avg


# In[25]:


plt.plot(range(1, 11), sum_sq_dist_avg, '-o')


# In[26]:


# Средней суммы средних внутрикластерных расстояний
new_centers = [kmean.cluster_centers_ for kmean in models]

sum_avg_intercluster_dist_avg = []
for k, kmean in enumerate(models):
    intercluster_sum = np.zeros(4)
    for i in range(4):
        ix = np.where(predicted_values[k] == i)
        if len(ix[0]) == 0:
            intercluster_sum[i - 1] = 0
        else:
```

```
            intercluster_sum[i - 1] = np.sum(euclidean_distances(*X[ix, :],
[kmean.cluster_centers_[i - 1]]) ** 2) / len(*X[ix, :])
    sum_avg_intercluster_dist_avg.append(np.sum(intercluster_sum) / (k + 1))
sum_avg_intercluster_dist_avg


# In[27]:


plt.plot(range(1, 11), sum_avg_intercluster_dist_avg, '-o')


# In[28]:


# Средней суммы межкластерных расстояний от количества кластеров
sum_intercluster_dist_avg = []

for k, kmean in enumerate(models):
    value = np.sum(euclidean_distances(kmean.cluster_centers_,
kmean.cluster_centers_))
    sum_intercluster_dist_avg.append(value / (k + 1))
sum_intercluster_dist_avg


# In[29]:


plt.plot(range(1, 11), sum_intercluster_dist_avg, '-o')


# ### Составление сравнительной таблицы

# In[30]:


import pandas as pd


# In[31]:


columns = pd.MultiIndex.from_product([['Иерархический метод', 'Метод k-
средних'],
                                      ['Сумма квадратов расстояний до
центроида', 'Сумма средних внутрикластерных расстояний', 'Сумма межкластерных
расстояний']])
df = pd.DataFrame(columns=columns)
df


# In[32]:


df['Иерархический метод', 'Сумма квадратов расстояний до центроида'] =
[sum_sq_dist for _ in range(len(sum_sq_dist_avg))]
df['Иерархический метод', 'Сумма средних внутрикластерных расстояний'] =
[sum_avg_intercluster_dist for _ in
range(len(sum_avg_intercluster_dist_avg))]
df['Иерархический метод', 'Сумма межкластерных расстояний'] =
[sum_intercluster_dist for _ in range(len(sum_intercluster_dist_avg))]

df['Метод k-средних', 'Сумма квадратов расстояний до центроида'] =
sum_sq_dist_avg
```

```
df['Метод k-средних', 'Сумма средних внутрикластерных расстояний'] =
sum_avg_intercluster_dist_avg
df['Метод k-средних', 'Сумма межкластерных расстояний'] =
sum_intercluster_dist_avg

df


# In[33]:


df.to_excel('result.xlsx')
```