Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине «Прикладные интеллектуальные системы и экспертные системы»

Кластеризация данных

Студент Сухоруков К.О.

Группа М-ИАП-22

Руководитель Кургасов В.В.

Липецк 2022 г.

Цель работы:

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

Задание кафедры

Задание:

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 средних внутрикластерных расстояний и 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	3
Вид классов	Blobs
Random_state	41
Cluster_std	2
Noise	-
Centers	7

Ход работы

Импортируем необходимые для работы библиотеки и модули.

- pandas — программная библиотека на языке Python для обработки и анализа данных;

- numPy (сокращенно от Numerical Python)— библиотека с открытым исходным кодом для языка программирования Python. Возможности: поддержка многомерных массивов (включая матрицы); поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами;

- matplotlib — библиотека на языке программирования Python для визуализации данных двумерной и трёхмерной графикой;

- библиотека NLTK — пакет библиотек и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python. Содержит графические представления и примеры данных;

- itertools стандартизирует основной набор быстрых эффективных по памяти инструментов, которые полезны сами по себе или в связке с другими инструментами;

- scikit-learn – это библиотека Python, которая является одной из самых полезных библиотек Python для машинного обучения. Она включает все алгоритмы и инструменты, которые нужны для задач классификации, регрессии и кластеризации. Она также включает все методы оценки производительности модели машинного обучения.

1) Импортировать необходимые для работы библиотеки и модули;

```
In [154]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          from sklearn.datasets import make_blobs
          from sklearn.metrics import classification_report
          from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
          from sklearn.cluster import KMeans
```

Генерация выборки

Вид класса	random_state	cluster_std	noise	centers
blobs	41	2	-	7

Рисунок 1 – Импорт библиотек

2) Загрузить данные и отобразить данные;

```
In [155]: centers_count = 7
          X, y = make_blobs(centers=7, random_state=41, cluster_std=2)
```

Сгенерированная выборка

```
In [156]: plt.scatter(X[:,0], X[:,1])
Out[156]: <matplotlib.collections.PathCollection at 0x7f69efcc92b0>
```

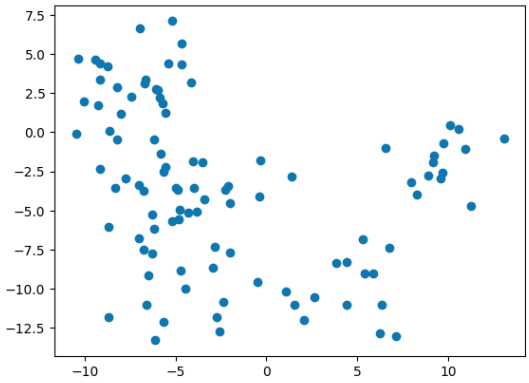

Рисунок 2 – Загрузка выборки

3) Кластеризация;

Иерархическая класстеризация

```
In [157]: def cluster_center(X, c):
              centers = np.zeros((centers_count,2))
              for i in range(1,8):
                  ix = np.where(c == i)
                  centers[i-1,:] = np.mean(X[ix,:], axis=1)

              return centers
```

```
In [158]: mergings = linkage(X, method='single')
          T = fcluster(mergings, centers_count, criterion='maxclust')
          c = cluster_center(X, T)
```

```
In [159]: dendrogram(mergings)
          plt.show()
```

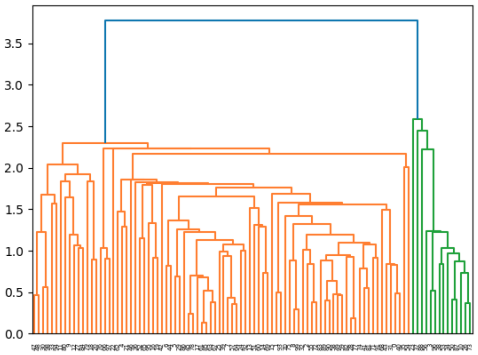

Рисунок 3 – Расстояние ближайшего соседа

```
In [160]: mergings = linkage(X, method='complete')
          dendrogram(mergings)
          plt.show()
```

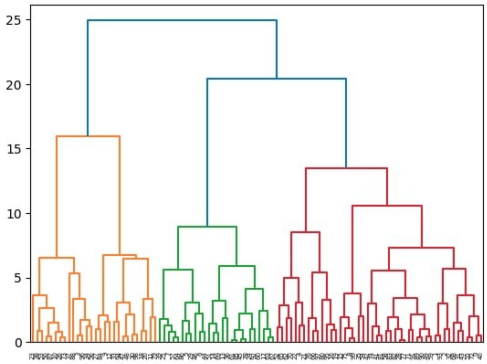

Рисунок 4 – Расстояние дальнего соседа

```
In [161]: mergings = linkage(X, method='ward')
          dendrogram(mergings)
          plt.show()
```

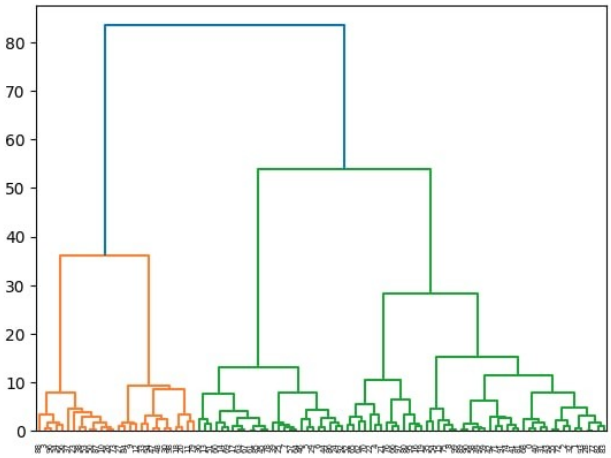

Рисунок 5 – Метод Уорда

4) Выбор лучшего способа;

Лучшее разбиение

```
In [162]: mergings = linkage(X, method='complete')
          dendrogram(mergings)
          plt.show()
```

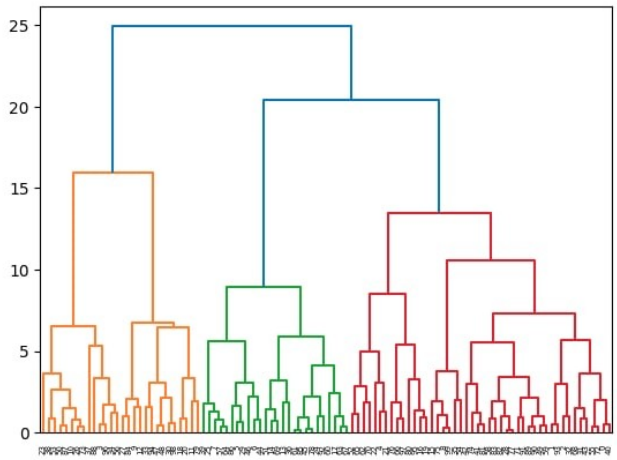

Рисунок 6 – Лучший способ разбиений

5) Отобразить разбиение на кластеры и центроиды:

```
In [163]: T = fcluster(mergings, centers_count, criterion='maxclust')
          c = cluster_center(X, T)
          plt.scatter(X[:,0], X[:,1], c=T)
          plt.scatter(c[:,0], c[:,1], c='black')

Out[163]: <matplotlib.collections.PathCollection at 0x7f69f0c99760>
```

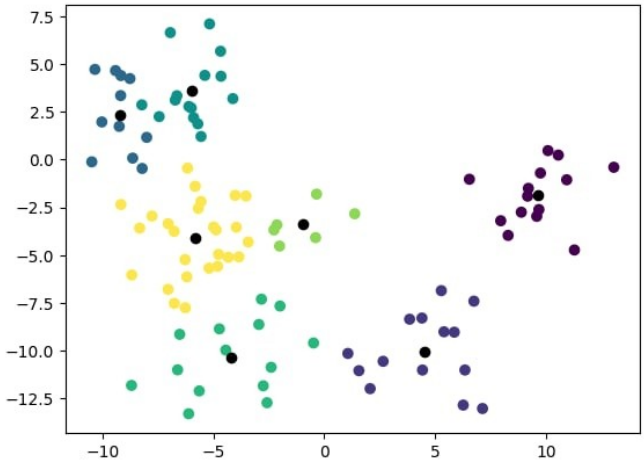

Рисунок 7 – Разбиение на кластеры и центроиды

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

```
In [164]: from sklearn.metrics.pairwise import euclidean_distances

In [169]: def sum_sq_dist(X, clust, cent):
              sums = np.zeros(centers_count)
              for i in range(1,centers_count+1):
                  ix = np.where(clust == i)
                  sums[i-1] = np.sum(euclidean_distances(*X[ix,:], [cent[i-1]])**2)
              return np.sum(sums) / centers_count

          def sum_av_sq_dist(X, clust, cent):
              sums = np.zeros(centers_count)
              for i in range(1,centers_count+1):
                  ix = np.where(clust == i)
                  sums[i-1] = np.sum(euclidean_distances(*X[ix,:], [cent[i-1]])**2)/len(*X[ix,:])
              return np.sum(sums) / centers_count

          def sum_cl_dist(cent):
              sums = np.sum(euclidean_distances(cent, cent))
              return sums / centers_count

In [170]: sum_sq_dist(X, T, c)

Out[170]: 79.32752133273394

In [171]: sum_av_sq_dist(X, T, c)

Out[171]: 5.23442770388245

In [172]: sum_cl_dist(c)

Out[172]: 67.67978464403674
```

Рисунок 8 – Среднее расстояние до центроидов, средняя сумма внутрикластерных расстояний и средняя сумма междукластерных расстояний

7) Провести кластеризацию методом к-средних

Метод k-средних

```
In [186]: from sklearn.cluster import KMeans
          from matplotlib.pyplot import xticks

In [187]: clussters = []
          dist = []
          cent = []
          for k in range(1,11):
              kmeans = KMeans(n_clusters=k)
              kmeans.fit(X)
              clussters.append(kmeans.predict(X))
              cent.append(kmeans.cluster_centers_)
              dist.append(kmeans.inertia_ / k)

In [189]: xticks(np.arange(1, 11, step=1))
          plt.plot(range(1, 11), dist, '-bo')

Out[189]: [<matplotlib.lines.Line2D at 0x7f69efeaa610>]
```

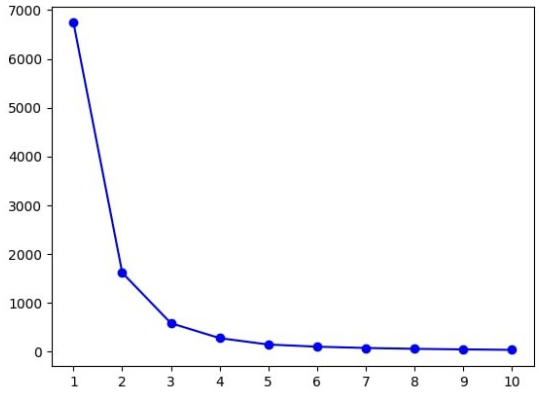

Рисунок 9 – Метод к-средних

8) Сформировать графики: зависимость средней суммы квадратов расстояний до центроида, средней суммы средних внутрикластерных расстояний и средней суммы межкластерных расстояний от количества кластеров.

```
In [201]: metric = []
          for k in range(10):
              sums = np.zeros(k)
              for i in range(k):
                  ix = np.where(clussters[k] == i)
                  sums[i-1] = np.sum(euclidean_distances(*X[ix,:], [cent[k][i-1]])**2)/len(*X[ix,:])
              metric.append(np.sum(sums) / (k+1))

In [202]: xticks(np.arange(1, 11, step=1))
          plt.plot(range(1, 11), metric, '-bo')

Out[202]: [<matplotlib.lines.Line2D at 0x7f69eb91ceb0>]
```

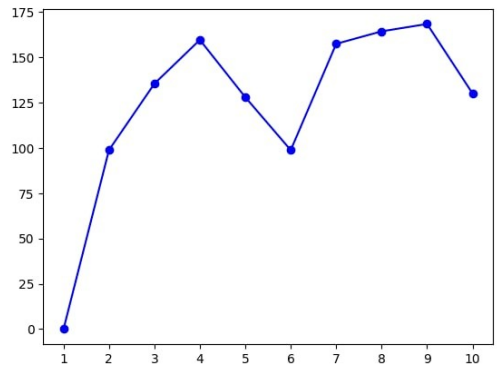

Рисунок 10 – Зависимость средней суммы квадратов расстояний до центроида

```
In [204]: metric = []
          for k in range(10):
              sums = np.sum(euclidean_distances(cent[k], cent[k]))
              metric.append(sums / (k+1))

In [205]: xticks(np.arange(1, 11, step=1))
          plt.plot(range(1, 11), metric, '-bo')

Out[205]: [<matplotlib.lines.Line2D at 0x7f69eba38a90>]
```

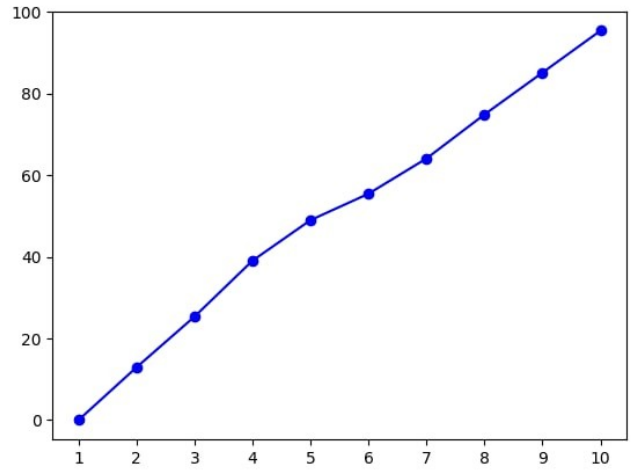

Рисунок 11 – Зависимость средней суммы средних внутрикластерных расстояний

Вывод

В ходе выполнения лабораторной работы были изучены и реализованы методы кластеризации данных. Лучшим оказалось разбиение, для которого расстояние между кластерами вычислялось методом дальнего соседа. Количество кластеров при этом подходе составило 6. При этом метод локтя нашёл оптимальное количество кластеров в размере трёх.

Код программы

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.metrics import classification_report
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from sklearn.cluster import KMeans


# ## Генерация выборки
# | Вид класса | random_state | cluster_std | noise | centers |
# |:----------:|:------------:|:-----------:|:-----:|:-------:|
# |    blobs   |      41      |      2      |   -   |    7    |

# In[2]:


centers_count = 7
X, y = make_blobs(centers=7, random_state=41, cluster_std=2)


# ## Сгенерированная выборка

# In[3]:


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


# ## Иерархическая класстеризация

# In[4]:


def cluster_center(X, c):
    centers = np.zeros((centers_count,2))
    for i in range(1,8):
        ix = np.where(c == i)
        centers[i-1,:] = np.mean(X[ix,:], axis=1)
```

```
    return centers


# In[5]:


mergings = linkage(X, method='single')
T = fcluster(mergings, centers_count, criterion='maxclust')
c = cluster_center(X, T)


# In[6]:


dendrogram(mergings)
plt.show()


# In[ ]:




# In[7]:


mergings = linkage(X, method='complete')
dendrogram(mergings)
plt.show()


# In[ ]:




# In[8]:
```

```
mergings = linkage(X, method='ward')
dendrogram(mergings)
plt.show()


# ## Лучшее разбиение

# In[9]:


mergings = linkage(X, method='complete')
dendrogram(mergings)
plt.show()


# In[10]:


T = fcluster(mergings, centers_count, criterion='maxclust')
c = cluster_center(X, T)
plt.scatter(X[:,0], X[:,1], c=T)
plt.scatter(c[:,0], c[:,1], c='black')


# In[11]:


from sklearn.metrics.pairwise import euclidean_distances


# In[12]:


def sum_sq_dist(X, clust, cent):
    sums = np.zeros(centers_count)
    for i in range(1,centers_count+1):
        ix = np.where(clust == i)
        sums[i-1] = np.sum(euclidean_distances(*X[ix,:], [cent[i-1]])**2)
    return np.sum(sums) / centers_count

def sum_av_sq_dist(X, clust, cent):
```

```
    sums = np.zeros(centers_count)
    for i in range(1,centers_count+1):
        ix = np.where(clust == i)
                          sums[i-1]   =   np.sum(euclidean_distances(*X[ix,:],
[cent[i-1]])**2)/len(*X[ix,:])
    return np.sum(sums) / centers_count

def sum_cl_dist(cent):
    sums = np.sum(euclidean_distances(cent, cent))
    return sums / centers_count


# In[13]:


sum_sq_dist(X, T, c)


# In[14]:


sum_av_sq_dist(X, T, c)


# In[15]:


sum_cl_dist(c)


# ## Метод k-средних

# In[16]:


from sklearn.cluster import KMeans
from matplotlib.pyplot import xticks


# In[17]:
```

```
clussters = []
dist = []
cent = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X)
    clussters.append(kmeans.predict(X))
    cent.append(kmeans.cluster_centers_)
    dist.append(kmeans.inertia_ / k)


# In[18]:


xticks(np.arange(1, 11, step=1))
plt.plot(range(1, 11), dist, '-bo')

metric = []
for k in range(10):
    sums = np.zeros(k)
    for i in range(k):
        ix = np.where(clussters[k] == i)
               sums[i-1]  =  np.sum(euclidean_distances(*X[ix,:],  [cent[k][i-
1]])**2)/len(*X[ix,:])
    metric.append(np.sum(sums) / (k+1))

xticks(np.arange(1, 11, step=1))
plt.plot(range(1, 11), metric, '-bo')

metric = []
for k in range(10):
    sums = np.sum(euclidean_distances(cent[k], cent[k]))
    metric.append(sums / (k+1))


xticks(np.arange(1, 11, step=1))
plt.plot(range(1, 11), metric, '-bo')
```