НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

Отчет по лабораторной работе №4

по курсу: «Специальные технологии баз данных»

Выполнил: *студент группы С20-702* *Нуритдинходжаева А.*

(подпись) (Фамилия И.О.)

Проверил: *Манаенкова Т.А.*

(оценка) (подпись) (Фамилия И.О.)

Москва 2024 г

Условие задания

Вариант 3

Задание 1

1. Загрузите с сайта https://sci2s.ugr.es/keel/datasets.php набор статистических данных, указанный в вашем варианте. Разберитесь, какие данные приведены в наборе и какой атрибут является меткой класса.

2. На основе загруженного файла создайте Pandas DataFrame, подобрав правильные типы данных столбцов.

3. Выполните стандартизацию полученного дата фрейма.

4. Оцените число кластеров для метода K-means с помощью методов локтя на основе инерции и искажения и с помощью метрики силуэта. Сравните рекомендуемые числа кластеров с реальным числом классов.

5. Выполните кластеризацию методом K-means с реальным числом кластеров. Постройте кросс-таблицу для сравнения оригинальных и предсказанных классов.

6. Постройте дендрограмму по методу Уорда для вашего набора данных. Оцените, соответствует ли дендрограмма реальному числу классов.

7. Выполните кластеризацию методом Уорда с реальным числом кластеров.

8. Постройте три раза проекцию по двум первым координатам точек набора данных, раскрасив их в различные цвета в соответствии с реальными классами, классами, предсказанными k-means, и классами, предсказанными методом Уорда.

9. Примените к исходному набору данных (после стандартизации) метод главных компонент, выбрав компоненты, соответствующие собственным числам, большим 1.

10. Повторите для модифицированного набора данных шаги 4-9.

11. Дайте оценку, стала ли кластеризация точнее после применения метода главных компонент.

Решение

Задание 1

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn import metrics
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans, AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram
from sklearn.metrics import silhouette_score
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.model_selection import ParameterGrid
from sklearn.metrics import classification_report
from sklearn.datasets import make_classification
import re
import seaborn as sns
from imblearn.over_sampling import RandomOverSampler
from sklearn.decomposition import PCA

f = open('segment.dat', 'r')
skip_rows = 0
COLUMNS = []
for line in f:
  if line[0] == '@':
    string = line.split(' ')
    if string[0] == '@attribute':
      COLUMNS.append(string[1])
    skip_rows += 1
COLUMNS = ['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-
density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-
mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-
mean','Class']
f.close()

df = pd.read_table('segment.dat', skiprows=skip_rows, sep=',', names=COLUMNS)

#числовой тип данных
numeric_columns = ['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-
density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-
mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-
mean','Class']
df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric)

pd.set_option('display.max_columns', 2000)
pd.set_option('display.width', 20000)
```

```
print(df)

scaler_std = preprocessing.StandardScaler()
x = scaler_std.fit_transform(df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-mean']])
df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-mean']] = x

print(df)

rg = range(2, 9)

inertia=[]
distortion=[]

for k in rg:
  kmeans = KMeans(n_clusters = k, random_state = 44444).fit(
    df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-mean']])
  inertia.append(kmeans.inertia_)
  distortion.append(sum(np.min(cdist(
    df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-mean']],
    kmeans.cluster_centers_, 'euclidean'),axis = 1))/df.shape[0])

print('\n', "инерция:", '\n', inertia, '\n')
print("искажение:", '\n', distortion)

plt.title("Метод локтя для инерции\n",fontsize=16)
plt.scatter(x=[i for i in rg],y=inertia,s=150,edgecolor='k')
plt.plot(rg, inertia);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Инерция",fontsize=15)
plt.xticks([i for i in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.show()

plt.title("Метод локтя для искажения\n",fontsize=16)
plt.scatter(x=[i for i in rg],
y=distortion,s=150,edgecolor='k')
plt.plot(rg, distortion);
plt.grid(True)
```

```
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Искажение",fontsize=15)
plt.xticks([i for i in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.show()

silhouette = []

for k in rg:
  kmeans = KMeans(n_clusters = k, random_state = 44444).fit(
    df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-density-5',
'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-
mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-mean']])
  preds = kmeans.predict(
    df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-density-5',
'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-
mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-mean']])
  silhouette.append(silhouette_score(
    df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-density-5',
'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-
mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean','Hue-mean']],
    preds))

print('\n', "Метод силуэта", '\n', silhouette, '\n')

plt.title("Метод силуэтов\n",fontsize=16)
plt.scatter(x=[r for r in rg],y=silhouette,edgecolor='k')
plt.plot(rg, silhouette);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Инерция",fontsize=15)
plt.xticks([r for r in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.show()

print(df['Class'].max())

kmeans = KMeans(n_clusters = 7, random_state = 42)
df['Class_label_new'] = kmeans.fit_predict(df[['Region-centroid-col', 'Region-centroid-row',
'Region-pixel-count', ' Short-line-density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
        'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-
mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean',
                    'Hue-mean']])

print(df)

cross_tab = pd.crosstab(df['Class'], df['Class_label_new'], margins = False)
print(cross_tab)
```

```
def plot_dendrogram(model, **kwards):
  counts = np.zeros(model.children_.shape[0])
  n_samples = len(model.labels_)
  for i, merge in enumerate(model.children_):
    current_count = 0
    for child_idx in merge:
      if child_idx < n_samples:
        current_count += 1
      else:
        current_count += counts[child_idx - n_samples]
    counts[i] = current_count

  linkage_matrix = np.column_stack([model.children_, model.distances_, counts]).astype(float)

  dendrogram(linkage_matrix, **kwards)

ag = AgglomerativeClustering(linkage = 'ward', n_clusters = None, distance_threshold = 0)
model = ag.fit(df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-count', ' Short-line-
density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
         'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-
mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean',
                 'Hue-mean']])
plot_dendrogram(model, truncate_mode = 'level', p =7)
plt.show()

ward = AgglomerativeClustering(n_clusters=7, linkage='ward')
df['ward_pred'] = ward.fit_predict(df[['Region-centroid-col', 'Region-centroid-row', 'Region-pixel-
count', ' Short-line-density-5', 'Short-line-density-2', 'Vedge-mean', 'Vedge-sd',
         'Hedge-mean', 'Hedge-sd', 'Intensity-mean','Rawred-mean','Rawblue-mean','Rawgreen-
mean','Exred-mean','Exblue-mean','Exgreen-mean','Value-mean','Saturatoin-mean',
                              'Hue-mean']])
print(df)

plt.subplots(figsize=(7, 6))
sns.scatterplot(data=df, x='Region-centroid-col', y='Region-centroid-row',
hue="Class",palette="crest")
plt.show()

plt.subplots(figsize=(7, 6))
sns.scatterplot(data=df, x='Region-centroid-col', y='Region-centroid-row',
hue="Class_label_new",palette="crest")
plt.show()

plt.subplots(figsize=(7, 6))
sns.scatterplot(data=df, x='Region-centroid-col', y='Region-centroid-row',
hue="ward_pred",palette="crest")
plt.show()

df1=df.drop('Class',axis=1)
```

```
pca=PCA()
pca.fit(df1)
print(pca.explained_variance_)

pca = PCA(n_components=5)
Principal_components=pca.fit_transform(df1)
pca_df = pd.DataFrame(data = Principal_components, columns = ['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC
5'])
print(pca_df)

rg = range(2, 9)

inertia=[]
distortion=[]

for k in rg:
  kmeans = KMeans(n_clusters = k, random_state = 44444).fit(
    pca_df[['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5']])
  inertia.append(kmeans.inertia_)
  distortion.append(sum(np.min(cdist(
      pca_df[['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5']],
       kmeans.cluster_centers_, 'euclidean'),axis = 1))/pca_df.shape[0])

print('\n', "инерция:", '\n', inertia, '\n')
print("искажение:", '\n', distortion)

plt.title("Метод локтя для инерции\n",fontsize=16)
plt.scatter(x=[i for i in rg],y=inertia,s=150,edgecolor='k')
plt.plot(rg, inertia);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Инерция",fontsize=15)
plt.xticks([i for i in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.show()

plt.title("Метод локтя для искажения\n",fontsize=16)
plt.scatter(x=[i for i in rg],
y=distortion,s=150,edgecolor='k')
plt.plot(rg, distortion);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Искажение",fontsize=15)
plt.xticks([i for i in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.show()

silhouette = []

for k in rg:
  kmeans = KMeans(n_clusters = k, random_state = 44444).fit(
```

```
    pca_df[['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5']])
  preds = kmeans.predict(
    pca_df[['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5']])
  silhouette.append(silhouette_score(
    pca_df[['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5']],
    preds))

print('\n', "Метод силуэта", '\n', silhouette, '\n')

plt.title("Метод силуэтов\n",fontsize=16)
plt.scatter(x=[r for r in rg],y=silhouette,edgecolor='k')
plt.plot(rg, silhouette);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Инерция",fontsize=15)
plt.xticks([r for r in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.show()

kmeans = KMeans(n_clusters = 7, random_state = 42)
pca_df['Class_label_new'] = kmeans.fit_predict(pca_df[['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5']])

print(pca_df)

cross_tab1 = pd.crosstab(df['Class'], df['Class_label_new'], margins = False)
print(cross_tab1)

ag = AgglomerativeClustering(linkage = 'ward', n_clusters = None, distance_threshold = 0)
model = ag.fit(pca_df[['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5']])
plot_dendrogram(model, truncate_mode = 'level', p =7)
plt.show()

ward = AgglomerativeClustering(n_clusters=7, linkage='ward')
pca_df['ward_pred'] = ward.fit_predict(pca_df[['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5']])
print(pca_df)

plt.subplots(figsize=(7, 6))
sns.scatterplot(data=pca_df, x='PC 1', y='PC 2', hue="Class_label_new",palette="crest")
plt.show()

plt.subplots(figsize=(7, 6))
sns.scatterplot(data=pca_df, x='PC 1', y='PC 2', hue="ward_pred",palette="crest")
plt.show()
```