Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа №3

по дисциплине «Машинное обучение»

Выполнил
студент гр. 3530904/00103 Плетнева А.Д.

Руководитель Селин И.А.

Санкт-Петербург
2022

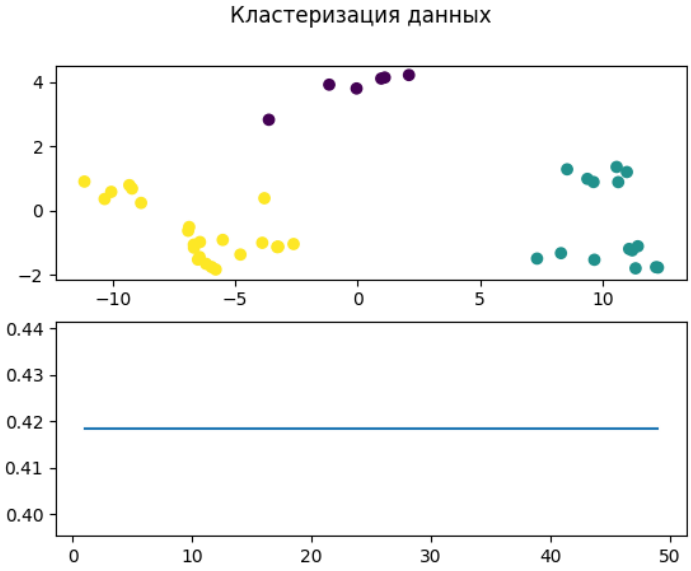
Оглавление

Задание

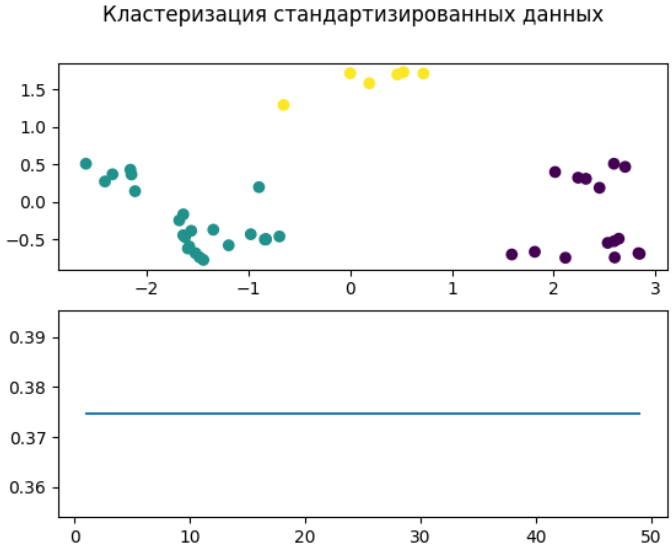
1. Разбейте множество объектов из набора данных pluton.csv на 3 кластера с помощью k-means. Сравните качество разбиения в зависимости от максимального числа итераций алгоритма и использования стандартизации.
2. Разбейте на кластеры множество объектов из наборов данных clustering_1.csv, clustering_2.csv и clustering_3.csv с помощью k-means, DBSCAN и иерархической кластеризации. Определите оптимальное количество кластеров (где это применимо). Какой из методов сработал лучше и почему?
3. Осуществите сжатие цветовой палитры изображения (любого, на ваш выбор). Для этого выделите n кластеров из цветов всех пикселей изображения и зафиксируйте центра этих кластеров. Создайте изображение с цветами из сокращенной палитры (цвета пикселей только из центров выделенных кластеров). Покажите исходное и сжатое изображения.
4. Постройте дендрограмму для набора данных votes.csv (число голосов, поданных за республиканцев на выборах с 1856 по 1976 год). Строки представляют 50 штатов, а столбцы - годы выборов (31). Проинтерпретируйте полученный результат.

Задание 1

Разбили множество объектов из набора данных pluton.csv на 3 кластера с помощью k-means, для оценки качества разбиения использовали **Индекс Дэвиса–Булдина**, оказалось, что он не меняется при увеличении максимального числа итераций

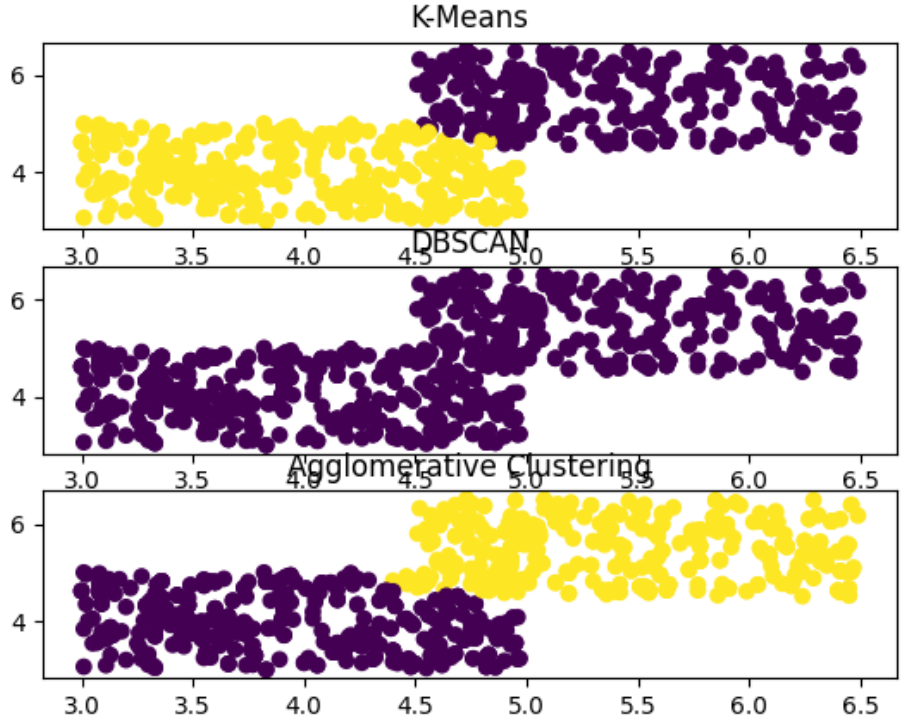


Далее данные были стандартизированы, для оценки качества разбиения также использовали индекс Дэвиса-Булдина, он также не меняется, однако для стандартизированных данных его значение меньше, то есть в этом случае качество разбиения лучше

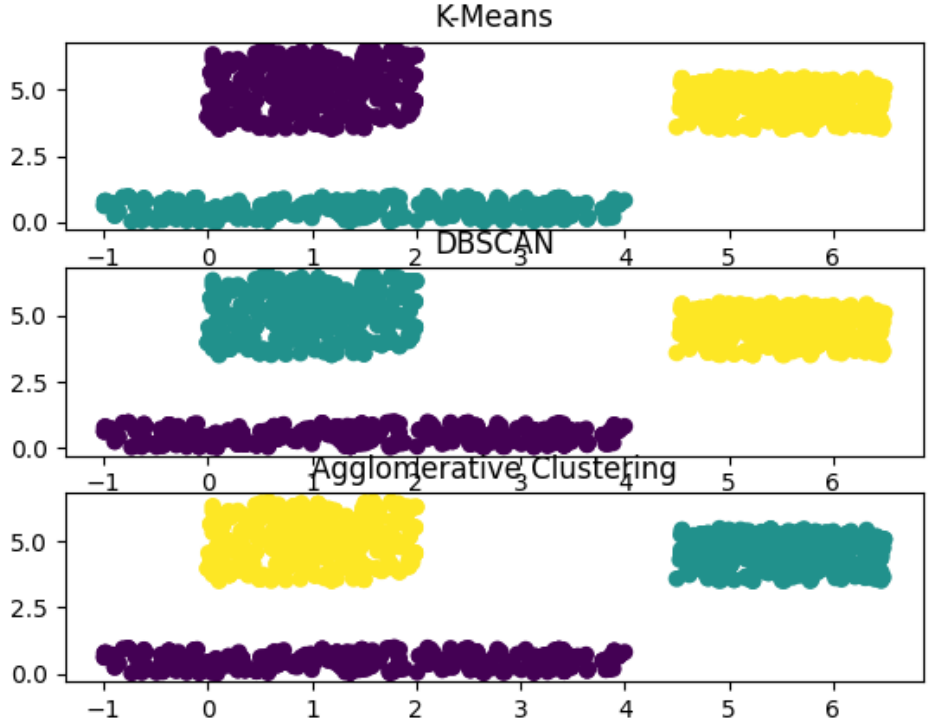


Задание 2

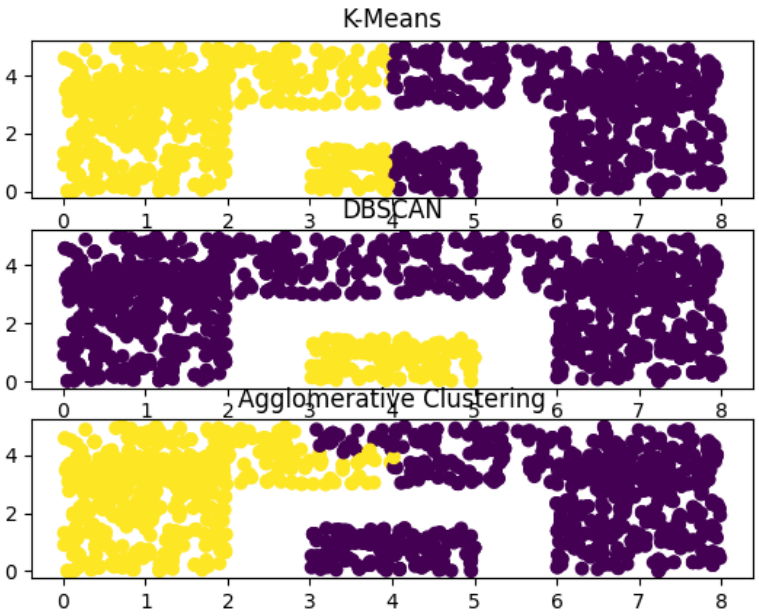
На первом наборе данных лучше всего себя показал метод к средних, а хуже всех справился метод DBSCAN, это связано с тем, что объекты, относящиеся к разным кластерам очень плотно расположены и метод распознает их как объекты одного кластера



Данные второго датасета все алгоритмы кластеризовали верно



В третьем наборе данные линейно не разделимы, лучше всего с таким набором справился DBSCAN. Алгоритм K-Means плохо справился с задачей, это связано с тем, что идея алгоритма заключается в том, что необходимо находить точки вокруг центров.



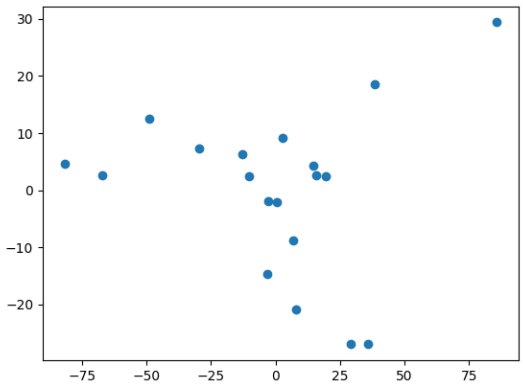
Задание 3

Quantized image (16 colors, K-Means)

Quantized image (4 colors, K-Means)

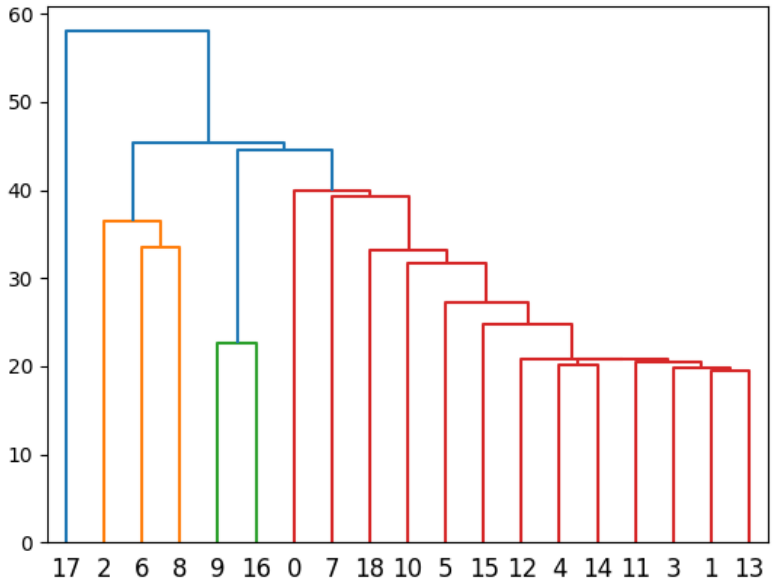
Задание 4

Точечная диаграмма, иллюстрирующая данные из votes.csv



В дендрограмме по вертикали откладывается минимальное расстояние между кластерами, а по горизонтали – исходные данные (объекты). Глядя на этот график, мы сразу видим, в каком порядке происходило объединение данных в группы и насколько сильно кластеры отделены друг от друга по минимальному расстоянию.

Кроме оценки качества этот график показывает нам, где можно провести уровень отсечения для получения определенного числа выходных кластеров. Это очень удобно, особенно, если мы наперед не знаем, на сколько таксонов следует разбить входные данные. Например, если мы хотим, чтобы минимальное расстояние между кластерами было 50, то получится 2 кластера (в одном 17 штат, в другом – все остальные)



Текст программы

ex1

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import cluster, decomposition
from sklearn.metrics import davies_bouldin_score
from sklearn.preprocessing import StandardScaler


def clustering(data, max_iteration, title):
    scores = {}
    for num_iter in range(1, max_iteration):
        kmeans = cluster.KMeans(3, max_iter=num_iter)
        kmeans.fit(data)
        scores[num_iter] = davies_bouldin_score(data, kmeans.labels_)
    max_key = max(scores, key=lambda k: scores[k])
    kmeans = cluster.KMeans(3, max_iter=max_key)
    kmeans.fit(data)
    pca = decomposition.PCA(n_components=2)
    X_reduced = pca.fit_transform(data)
    fig, ax = plt.subplots(2, 1)
    fig.suptitle(title)
    ax[0].scatter(X_reduced[:, 0], X_reduced[:, 1], c=kmeans.labels_)
    ax[1].plot(scores.keys(), scores.values())
    plt.show()


data = pd.read_csv('pluton.csv')

scaler = StandardScaler()
scale_data = scaler.fit_transform(data)

clustering(data, 50, "Кластеризация данных")

clustering(scale_data, 50, "Кластеризация стандартизированных данных")
```

ex2

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import cluster
from sklearn.cluster import DBSCAN, AgglomerativeClustering


def visualized_fun(data, cluster_num):
    kmeans = cluster.KMeans(cluster_num)
    dbscan = DBSCAN()
    kmeans.fit(data)
    dbscan.fit(data)
    agg_clustering = AgglomerativeClustering(cluster_num)
    agg_clustering.fit(data)
    fig, ax = plt.subplots(3, 1)
    ax[0].scatter(data[0], data[1], c=kmeans.labels_)
    ax[0].set_title('K-Means')
    ax[1].scatter(data[0], data[1], c=dbscan.labels_)
    ax[1].set_title('DBSCAN')
    ax[2].scatter(data[0], data[1], c=agg_clustering.labels_)
    ax[2].set_title('Agglomerative Clustering')
    plt.show()


data_1 = pd.read_csv("clustering_1.csv", header=None, sep="\t")
data_2 = pd.read_csv("clustering_2.csv", header=None, sep="\t")
```

```
data_3 = pd.read_csv("clustering_3.csv", header=None, sep="\t")

visualized_fun(data_1, 2)
visualized_fun(data_2, 3)
visualized_fun(data_3, 2)
```

ex3

```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin
from sklearn.utils import shuffle

n_colors = 4

img = Image.open("image.png")

# Convert to floats instead of the default 8 bits integer coding. Dividing by
# 255 is important so that plt.imshow behaves works well on float data (need
to
# be in the range [0-1])
img = np.array(img, dtype=np.float64) / 255

# Load Image and transform to a 2D numpy array.
w, h, d = original_shape = tuple(img.shape)
assert d == 3
image_array = np.reshape(img, (w * h, d))

image_array_sample = shuffle(image_array, random_state=0, n_samples=1_000)
kmeans = KMeans(n_clusters=n_colors, random_state=0).fit(image_array_sample)


labels = kmeans.predict(image_array)


def recreate_image(codebook, labels, w, h):
    """Recreate the (compressed) image from the code book & labels"""
    return codebook[labels].reshape(w, h, -1)


# Display all results, alongside original image
plt.figure(1)
plt.clf()
plt.axis("off")
plt.title("Original image")
plt.imshow(img)

plt.figure(2)
plt.clf()
plt.axis("off")
plt.title(f"Quantized image ({n_colors} colors, K-Means)")
plt.imshow(recreate_image(kmeans.cluster_centers_, labels, w, h))

plt.show()
```

ex4

```
import pandas as pd
from matplotlib import pyplot as plt
from numpy import array
from scipy.cluster import hierarchy
from sklearn import decomposition

data = pd.read_csv('votes.csv')
data = data.dropna(how='any')
```

```
temp = hierarchy.linkage(array(data))
plt.figure()
dn = hierarchy.dendrogram(temp)
plt.show()

pca = decomposition.PCA(n_components=2)
X_reduced = pca.fit_transform(data)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1])
plt.show()
```