Смешанная кластеризация

1. Задача

Рассмотрев популярные видео YouTube, выделить по соотношению “нравится-не нравится” группы популярности видео; оценить противоречивость восприятия популярного контента.

2. Постановка задачи:

Реализовать распределение набора двумерных данных по кластерам с помощью комбинированного метода: иерархическая кластеризация для малой выборки данных; кластеризация по методу k-means для всех данных с помощью центроидов полученных кластеров.

3. Используемые данные:

Датасет из открытых источников с популярными видео Youtube на 14 ноября 2017 года. Данные - название, теги, просмотры, количество меток “нравится-не нравится”.

4. Решение:

Импорт используемых библиотек и методов:

import numpy as np

import matplotlib.pyplot as plt - для вывода данных в визуальном виде

import pandas as pd - для работы с данными таблицы

from random import random - для генерации малой выборки данных

import scipy.cluster.hierarchy as sch - для построения дендрограммы кластеризации

from sklearn.cluster import AgglomerativeClustering - для иерархической кластеризации

Задание числа N данных в малой выборке, числа K рассматриваемых данных вообще.

```
N = 100

K = 2000

colors = ['red', 'orange', 'yellow', 'green', 'blue', 'pink', 'black']
```

Обработка данных из таблицы для последующего использования

```
dataset = pd.read_csv('CAvideos.csv')

X = dataset.iloc[:K, [8, 9]]

# X = X[X['likes'] >= 10000]

X = X.values
```

Задание функций для последующего использования:

```
def dist(n1, n2, array1, array2):  -евклидово расстояние между двумя точками

   return ((array1[n1][0]-array2[n2][0])**2 + (array1[n1][1]-array2[n2][1])**2)**(1/2)
```

def recreate_centroids(array_clusters, array_centroids): - перерассч1т центроидов по кластерам

```
n = len(array_clusters)
for ja in range(n):
    sum_x = 0
    sum_y = 0
    if array_clusters[ja]:
        for element in array_clusters[ja]:
            sum_x += element[0]
            sum_y += element[1]
        array_centroids[ja] = [sum_x / len(array_clusters[ja]), sum_y / len(array_clusters[ja])]
```

def distribute(X0, array_clusters, array_centroids): - распределение данных по кластерам

```
for cluster in clusters:
    cluster.clear()
for num_point in range(len(X0)):
    distances = [float('inf') for z in range(len(array_centroids))]
    for num_centroid in range(len(centroids)):
        distances[num_centroid] = dist(num_point, num_centroid, X0, array_centroids)
    array_clusters[distances.index(min(distances))].append(X0[num_point])
```

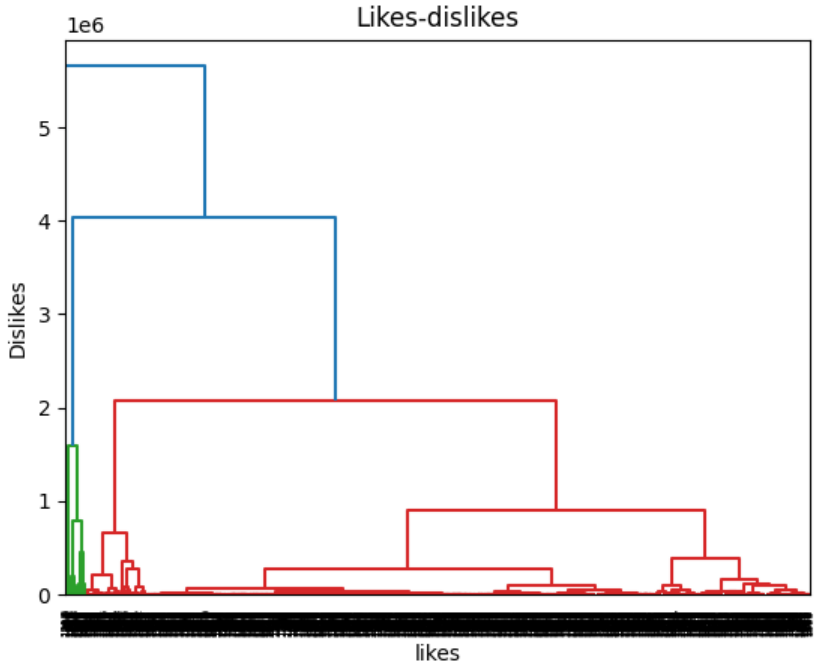
def get_hash(array_2d): - функция преобразования массива центроидов к некоторому числу для последующего ускорения процесса кластеризации

```
summa = 0
for tochka in array_2d:
    summa += (tochka[0]+tochka[1])
return summa
```

Шаг 1 - Определение числа кластеров.

```
dendrogram = sch.dendrogram(sch.linkage(X, method="ward"))
plt.title('Likes-dislikes')
plt.xlabel('likes')
plt.ylabel('Dislikes')
plt.show()
```

Рисунок 1: Дендрограмма распределения данных.



n_clusters = 7

Шаг 2 - Иерархическая кластеризация для малой выборки данных:

Создание малой выборки:

```
little_sample = []
for i in range(N):
    j = round(K * random())
    little_sample.append([X[j][0], X[j][1]])
```

Применение иерархической кластеризации:

```
hc = AgglomerativeClustering(n_clusters=n_clusters, affinity='euclidean', linkage='ward')
y_hc = hc.fit_predict(little_sample)
clusters = [[] for i in range(n_clusters)]
for i in range(len(y_hc)):
    clusters[y_hc[i]].append(little_sample[i])
# show clusters for little_sample
```

Шаг 3 - Кластеризация по методу k-means для полученных кластеров.

Получение центроидов имеющихся предварительных кластеров:

```
centroids = [[] for i in range(n_clusters)]
recreate_centroids(clusters, centroids)
```

Первое аспределение точек по кластерам:

```
distribute(X, clusters, centroids)

hash_old = get_hash(centroids)
recreate_centroids(clusters, centroids)
distribute(X, clusters, centroids)
hash_new = get_hash(centroids)
i = 0
```

Повторение цикла “определение центроидов для каждого кластера, определение кластеров для каждого центроида” либо достаточно большое число раз, либо до постоянного определения положения центроидов (то есть остановки изменений кластеров):

```
while i < 50 and hash_old != hash_new:
    i += 1
    hash_old = hash_new
    recreate_centroids(clusters, centroids)
    distribute(X, clusters, centroids)
    hash_new = get_hash(centroids)
```

Шаг 4 - демонстрация результатов:

```
for i in range(n_clusters):
```

```
    for point in clusters[i]:
        plt.scatter(point[0], point[1], s=3, c=colors[i])
plt.show()
```

Рисунок 2: визуализация полученных кластеров.

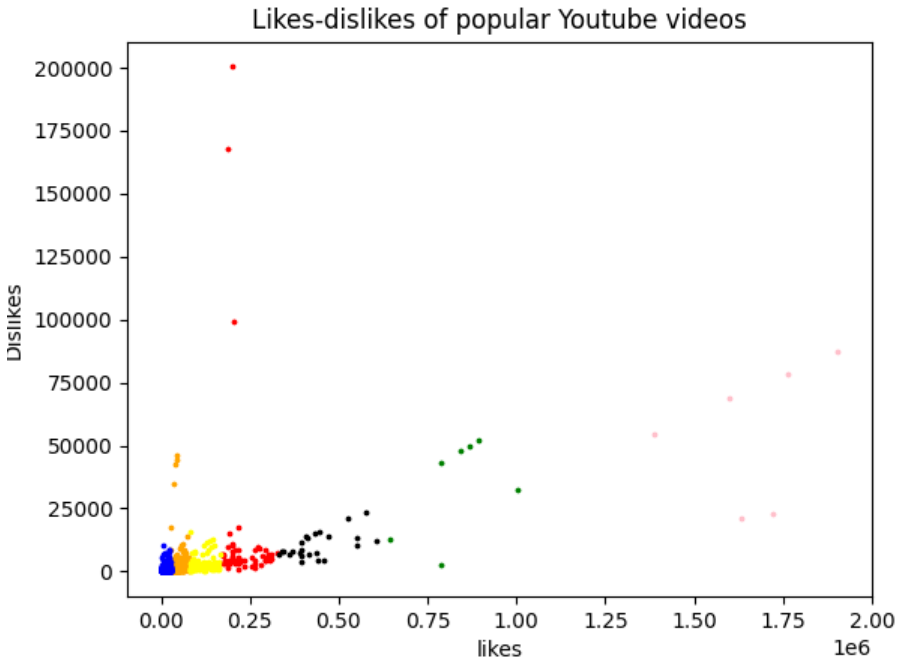
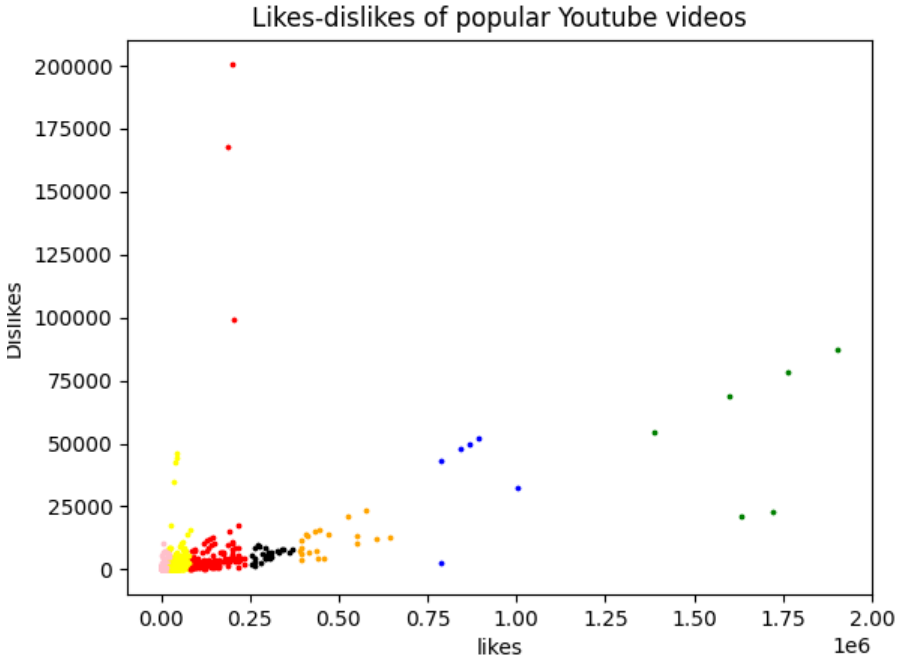


Рисунок 3: визуализация кластеров, полученных с помощью иерархической кластеризации.



5. Выводы:

Реализованная кластеризация демонстрирует несколько другие результаты в области близкого расположения большого числа точек, нежели классические методы. При подобного рода данных минусом является неспособность справиться с аномальными точками (см. верхнюю часть красного кластера).

Для проверки наличия общих точек кластеров при тестах было выведено значение i (переменной подсчёта количества перерасчётов кластеров) на момент остановки программы. За 10 тестов (имеет смысл т.к. выбор первичных центроидов в определённой степени случаен) i ни разу не превысило 35, что позволяет говорить об отсутствии таких точек (порог остановки перерасчётов - i=50).

Тем не менее, в обоих случаях видна градация относительного уровня одобрения видео.