Кластеризация пользователей по кинопредпочтениям на базе датасета Movielens с помощью алгоритма KMeans из библиотеки Scikit-learn

immediate

December 14, 2021

Выполнила студент 431 группы Алтынова Анна

Цель: кластеризация пользователей сервиса IMDB по кинопредпочтениям

шаг 0: импорт и вывод примеров данных

шаг 1: очистка данных: удаляем теги(их почти не ставят), таймстемпы, жанры, названия, удаляем фильмы, которым ставили рейтинг не больше 30 раз и пользователей, которые ставили рейтинг не больше 30 раз, NAN замещаем средними по колонке, объединяем нужные данные в один датафрейм

шаг 2: запускаем алгоритм кластеризации при k=10, выводим пример кластера

шаг 3: запускаем алгоритм на k от 1 до 30, для каждого k считаем "ошибку": сумма по кластерам средних квадратов отклонений от центра кластера (по пользователям из каждого кластера), представляем на графике

шаг 4: выводим пример кластера при $k > 1$ с минимальной "ошибкой"

результат: получили кластеризацию пользователей по кинопредпочтениям, то есть, в каждом кластере находятся пользователи, оценивающие каждый фильм похожим образом

```
<import pandas as pd
import numpy as np
from sklearn.cluster import KMeans

movies = pd.read_csv('/content/drive/MyDrive/movielens/movies.csv')
ratings = pd.read_csv('/content/drive/MyDrive/movielens/ratings.csv')
tags = pd.read_csv('/content/drive/MyDrive/movielens/tags.csv')

ratings= ratings.drop(columns=['timestamp'])
movies = movies.drop(columns=['genres', 'title'])

data = pd.merge(ratings, movies, on='movieId')
data1 = pd.pivot_table(data, index='userId', columns='movieId', values='rating')


movies30 = ratings['movieId'].value_counts()[ratings['movieId'].value_counts() > 30]

ratings30 = ratings[ratings['movieId'].isin(movies30.index)]

users30 = ratings30['userId'].value_counts()[ratings30['userId'].value_counts() > 30]


ratings30 = ratings30[ratings30['userId'].isin(users30.index)]
data2 = pd.merge(ratings30, movies, on='movieId')

data2 = pd.pivot_table(data2, index='userId', columns='movieId', values='rating')


dataset_mean = data2.fillna(data2.mean())

k_means = KMeans(n_clusters=10, random_state=0)

labels = k_means.fit_predict(X=dataset_mean)>


<array([0, 9, 0, 0, 0, 0, 9, 0, 0, 9, 9, 0, 0, 9, 0, 9, 9, 0, 0, 0, 9, 0,
       0, 0, 0, 9, 9, 9, 0, 0, 5, 3, 0, 0, 0, 0, 9, 0, 9, 0, 0, 0, 0, 0,
       0, 0, 0, 9, 0, 0, 0, 4, 0, 0, 0, 0, 0, 9, 9, 0, 0, 0, 0, 9, 0, 0,
       0, 9, 9, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9, 0, 0, 0, 0, 9, 0, 9, 0, 0,
       0, 0, 9, 0, 8, 0, 9, 9, 7, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       9, 9, 0, 0, 0, 0, 9, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0,
       0, 0, 9, 9, 0, 0, 9, 9, 0, 0, 0, 0, 0, 9, 0, 9, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 9, 0, 0, 9, 0, 0, 0, 9, 0, 0, 0, 0, 0, 9, 0, 0, 0, 9,
       0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 9, 9, 0, 9, 9, 9, 0, 0, 0, 0, 9, 1,
       0, 0, 0, 9, 9, 0, 0, 0, 0, 9, 9, 0, 0, 0, 9, 9, 0, 0, 9, 0, 0, 0,
       0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 9, 0, 0, 9, 9,
       9, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0, 9, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 9, 0, 0, 0, 0, 0, 9, 9, 0, 0,
       9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 9, 0, 0, 0, 0, 0, 0,
       0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 9, 0,
```

```
       9, 9, 0, 0, 0, 0, 9, 0, 0, 0, 9, 0, 9, 9, 0, 0, 9, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 9,
       9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 9, 0, 9, 0,
       9, 2, 0, 0, 9, 0, 0, 0, 0, 9, 0], dtype=int32)>
<
clusters = [[data2.index[i] for i in range(len(labels)) if labels[i] == j] for j in range(10)]


centers = k_means.cluster_centers_

def error(data, clusters, centers): # NANs ignored
  errsum = 0
  for i in range(len(centers)):
    cluster_data = data[data.index.isin(clusters[i])]
    err = ((cluster_data - centers[i])**2).sum(axis=1).sum() / len(clusters[i])
    errsum += err
  return errsum


errors = []
for k in range(1, 30):
  k_means = KMeans(n_clusters=k, n_init = 20, random_state=0)
  labels = k_means.fit_predict(X=dataset_mean)
  clusters = [[data2.index[i] for i in range(len(labels)) if labels[i] == j] for j in range(k)]
  centers = k_means.cluster_centers_
  err_k = error(data2, clusters, centers)
  errors.append(err_k)

import matplotlib.pyplot as plt

plt.plot([i for i in range(1, 30)], errors, label='errors')
plt.legend()
plt.show()>
```

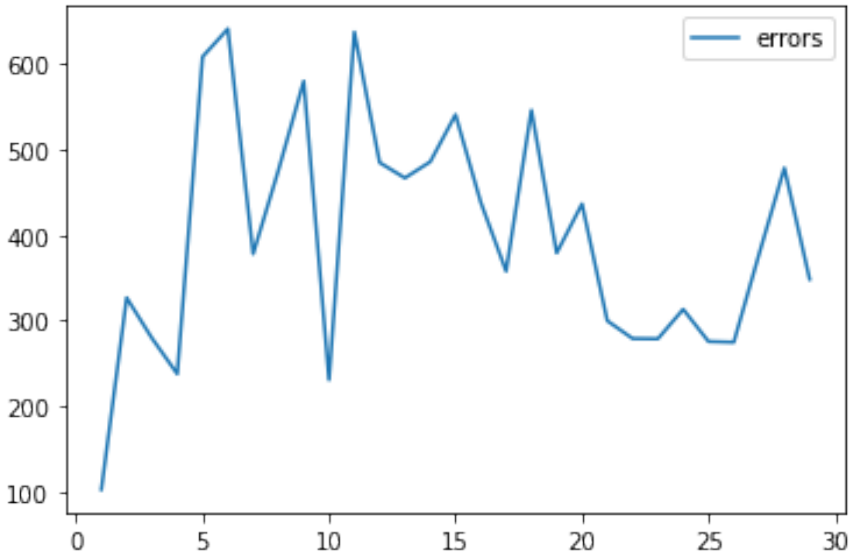

Figure 1: This is a caption

```
<k_means = KMeans(n_clusters=10,n_init=20, random_state=0) # n_clusters = index(min(errors)) + 1
labels = k_means.fit_predict(X=dataset_mean)
clusters = [[data2.index[i] for i in range(len(labels)) if labels[i] == j] for j in range(10)]
centers = k_means.cluster_centers_

error(data2, clusters, centers)
230.5456166290174

labels


array([0, 9, 0, 0, 0, 0, 9, 0, 0, 9, 9, 0, 0, 9, 0, 9, 9, 0, 0, 0, 9, 0,
       0, 0, 0, 9, 9, 9, 0, 0, 5, 3, 0, 0, 0, 0, 9, 0, 9, 0, 0, 0, 0, 0,
       0, 0, 0, 9, 0, 0, 0, 4, 0, 0, 0, 0, 0, 9, 9, 0, 0, 0, 0, 9, 0, 0,
       0, 9, 9, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9, 0, 0, 0, 0, 9, 0, 9, 0, 0,
       0, 0, 9, 0, 8, 0, 9, 9, 7, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       9, 9, 0, 0, 0, 0, 9, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0,
       0, 0, 9, 9, 0, 0, 9, 9, 0, 0, 0, 0, 0, 9, 0, 9, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 9, 0, 0, 9, 0, 0, 0, 9, 0, 0, 0, 0, 0, 9, 0, 0, 0, 9,
       0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 9, 9, 0, 9, 9, 9, 0, 0, 0, 0, 9, 1,
       0, 0, 0, 9, 9, 0, 0, 0, 0, 9, 9, 0, 0, 0, 9, 9, 0, 0, 9, 0, 0, 0,
       0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 9, 0, 0, 9, 9,
       9, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0, 9, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 9, 0, 0, 0, 0, 0, 9, 9, 0, 0,
       9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 9, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 9, 0,
       9, 9, 0, 0, 0, 0, 9, 0, 0, 0, 9, 0, 9, 9, 0, 0, 9, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 9,
       9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 9, 0, 9, 0,
```

```
9, 2, 0, 0, 9, 0, 0, 0, 0, 9, 0], dtype=int32)
```