Untitled115

July 4, 2023

1 Module 9: Recommender

System Case Study Contact us: support@intellipaat.com / © Copyright Intellipaat / All rights reserved Intel iPaat Python for Data Science Certification Course Problem Statement: Sam's next exam would be to build a "Recommender System" using the Singular Value Decomposition (SVD) algorithm. Questions would be asked on the basis of what you've learnt in the respective module. Tasks To Be Performed: 1. Implementing User-Based Recommender System using SVD (Singular Value Decomposition) method: a. Load the 'ratings' and 'movies' datasets which is a part of 'MovieLense' b. Find the unique number of users and movies in the 'ratings' dataset c. Create a rating matrix for the 'ratings' dataset and store it in 'Ratings' d. Load the 'ratings' dataset as SVD's Dataset object and compute 3-fold cross-validation using the SVD object e. Find all the movies rated as 5 stars by user id '5' and store it in 'ratings_1' data frame f. Create a shallow copy of the 'movies' dataset and store the result in 'user_5' g. Train a recommender system using the SVD object and predict the ratings for user id '5' h. Print the top10 movie recommendations for the user id '5'

[1]:
```
## import the requirred libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
##import the datasets
movies=pd.read_csv(r'C:/Users/Vikas/Downloads/movies.csv',sep=',')
```

[4]:
```
movies.head(10)
```

[4]:
```
   movieId                              title  \
0        1                   Toy Story (1995)
1        2                     Jumanji (1995)
2        3            Grumpier Old Men (1995)
3        4           Waiting to Exhale (1995)
4        5 Father of the Bride Part II (1995)
5        6                        Heat (1995)
6        7                     Sabrina (1995)
7        8                Tom and Huck (1995)
8        9                Sudden Death (1995)
9       10                   GoldenEye (1995)
```

```
                                  genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                  Adventure|Children|Fantasy
2                              Comedy|Romance
3                        Comedy|Drama|Romance
4                                      Comedy
5                       Action|Crime|Thriller
6                              Comedy|Romance
7                          Adventure|Children
8                                      Action
9                  Action|Adventure|Thriller
```

[6]:
```
## import another one datase
ratings=pd.read_csv(r'C:/Users/Vikas/Downloads/ratings.csv',sep=',')
```

[7]:
```
ratings.head(10)
```

[7]:
```
   userId  movieId  rating   timestamp
0       1        2     3.5  1112486027
1       1       29     3.5  1112484676
2       1       32     3.5  1112484819
3       1       47     3.5  1112484727
4       1       50     3.5  1112484580
5       1      112     3.5  1094785740
6       1      151     4.0  1094785734
7       1      223     4.0  1112485573
8       1      253     4.0  1112484940
9       1      260     4.0  1112484826
```

[8]:
```
##Find the unique number of users and movies in the ‘ratings’ dataset
df=ratings.groupby ('movieId') ['userId'].nunique ().
 ↪reset_index(name='userIdCount')
df
```

[8]:
```
       movieId  userIdCount
0            1         2569
1            2         1155
2            3          685
3            4          138
4            5          657
…          …          …
14021   130073            1
14022   130219            1
14023   130462            1
14024   130490            2
14025   130642            1
```

```
[14026 rows x 2 columns]
```

[9]:
```
# b. Find the unique number of users and movies in the 'ratings' dataset
num_users = ratings['userId'].nunique()
num_movies = ratings['movieId'].nunique()
```

[10]:
```
num_users
```

[10]:
```
7120
```

[11]:
```
num_movies
```

[11]:
```
14026
```

[12]:
```
# c. Create a rating matrix for the 'ratings' dataset and store it in 'Ratings'
Ratings = ratings.pivot(index='userId', columns='movieId', values='rating').
  ↪fillna(0)
Ratings
```

[12]:
```
movieId  1       2       3       4       5       6       7       8       \
userId
1            0.0     3.5     0.0     0.0     0.0     0.0     0.0     0.0
2            0.0     0.0     4.0     0.0     0.0     0.0     0.0     0.0
3            4.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
4            0.0     0.0     0.0     0.0     0.0     3.0     0.0     0.0
5            0.0     3.0     0.0     0.0     0.0     0.0     0.0     0.0
…          …     …     …     …     …     …     …     …
7116         4.0     0.0     0.0     0.0     3.5     0.0     0.0     0.0
7117         4.0     0.0     4.0     0.0     0.0     5.0     3.0     0.0
7118         0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
7119         5.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
7120         4.5     4.0     0.0     0.0     0.0     0.0     4.0     0.0

movieId  9       10      …  129350  129354  129428  129707  130052  130073  \
userId                   …
1            0.0     0.0 …     0.0     0.0     0.0     0.0     0.0     0.0
2            0.0     0.0 …     0.0     0.0     0.0     0.0     0.0     0.0
3            0.0     0.0 …     0.0     0.0     0.0     0.0     0.0     0.0
4            0.0     4.0 …     0.0     0.0     0.0     0.0     0.0     0.0
5            0.0     0.0 …     0.0     0.0     0.0     0.0     0.0     0.0
…          …     … …     …     …     …     …     …     …
7116         0.0     0.0 …     0.0     0.0     0.0     0.0     0.0     0.0
7117         1.0     3.0 …     0.0     0.0     0.0     0.0     0.0     0.0
7118         0.0     0.0 …     0.0     0.0     0.0     0.0     0.0     0.0
7119         0.0     0.0 …     0.0     0.0     0.0     0.0     0.0     0.0
7120         0.0     0.0 …     0.0     0.0     0.0     0.0     0.0     0.0
```

```
movieId  130219  130462  130490  130642
userId
1           0.0     0.0     0.0     0.0
2           0.0     0.0     0.0     0.0
3           0.0     0.0     0.0     0.0
4           0.0     0.0     0.0     0.0
5           0.0     0.0     0.0     0.0
…         …     …     …     …
7116        0.0     0.0     0.0     0.0
7117        0.0     0.0     0.0     0.0
7118        0.0     0.0     0.0     0.0
7119        0.0     0.0     0.0     0.0
7120        0.0     0.0     0.0     0.0

[7120 rows x 14026 columns]
```

[13]:
```
!pip install surprise
import pandas as pd
from surprise import Dataset
from surprise import Reader
from surprise import SVD
from surprise.model_selection import cross_validate
```

```
Requirement already satisfied: surprise in c:\users\vikas\anaconda3\lib\site-
packages (0.1)
Requirement already satisfied: scikit-surprise in
c:\users\vikas\anaconda3\lib\site-packages (from surprise) (1.1.3)
Requirement already satisfied: joblib>=1.0.0 in
c:\users\vikas\anaconda3\lib\site-packages (from scikit-surprise->surprise)
(1.1.1)
Requirement already satisfied: numpy>=1.17.3 in
c:\users\vikas\anaconda3\lib\site-packages (from scikit-surprise->surprise)
(1.23.5)
Requirement already satisfied: scipy>=1.3.2 in
c:\users\vikas\anaconda3\lib\site-packages (from scikit-surprise->surprise)
(1.10.0)
```

[19]:
```
##conda install -c conda-forge scikit-surprise
## import the library from surprice package
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate

## load the reader library
reader = Reader()
data = Dataset.load_from_df(ratings[['userId','movieId','rating']],reader)

## use the SVD algorithm
svd = SVD()
```

```
## compute the RMSE of the SVD algorithm
cross_validate(svd,data,measures=['RMSE','MAE','MSE'],cv=3,verbose=True)
```

```
Evaluating RMSE, MAE, MSE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.8440  0.8444  0.8445  0.8443  0.0002
MAE (testset)     0.6466  0.6467  0.6470  0.6468  0.0002
MSE (testset)     0.7123  0.7129  0.7132  0.7128  0.0004
Fit time          8.73    9.82    8.56    9.03    0.56
Test time         3.21    3.35    2.85    3.14    0.21
```

```
[19]: {'test_rmse': array([0.84400049, 0.8443516 , 0.8445323 ]),
       'test_mae': array([0.64659188, 0.64673123, 0.64702161]),
       'test_mse': array([0.71233682, 0.71292963, 0.7132348 ]),
       'fit_time': (8.731149196624756, 9.815208435058594, 8.558051824569702),
       'test_time': (3.2115299701690674, 3.349113941192627, 2.853925943374634)}
```

2 above what iam mentioned press shift and tab u will open documentation

```
[20]: ## here i want to predict user one for diffirent movies rated at reccoment␣
       ↪those movies the reccoment all those movies prediction
      ##highest perticular movies by this users
      ratings.head(10)
```

```
[20]:    userId  movieId  rating   timestamp
      0       1        2     3.5  1112486027
      1       1       29     3.5  1112484676
      2       1       32     3.5  1112484819
      3       1       47     3.5  1112484727
      4       1       50     3.5  1112484580
      5       1      112     3.5  1094785740
      6       1      151     4.0  1094785734
      7       1      223     4.0  1112485573
      8       1      253     4.0  1112484940
      9       1      260     4.0  1112484826
```

```
[21]: # Find all the movies rated as5 stars by user id ‘5’ and store it in␣
       ↪‘ratings_1’ data frame

      ratings_1= ratings[(ratings['userId']==5)&(ratings['rating']==5)]
      ratings_1=ratings_1.set_index('movieId')
      ratings_1=ratings_1.join(movies)['title']
      ratings_1.head(10)
```

[21]:
```
movieId
11                       Dracula: Dead and Loving It (1995)
62       Don't Be a Menace to South Central While Drink…
141                                           Gospa (1995)
150                                  Addiction, The (1995)
260                               Ladybird Ladybird (1994)
318      Strawberry and Chocolate (Fresa y chocolate) (…
364                                        Maverick (1994)
368                                   Reality Bites (1994)
377                        When a Man Loves a Woman (1994)
380                                     Bad Company (1995)
Name: title, dtype: object
```

[30]:
```
## Create a shallow copy of the 'movies' dataset and store the result in␣
 ↪'user_5'
user_5= movies.copy()
user_5= user_5.reset_index()
```

[31]:
```
user_5
```

[31]:
```
       index  movieId                               title  \
0          0        1                    Toy Story (1995)
1          1        2                      Jumanji (1995)
2          2        3             Grumpier Old Men (1995)
3          3        4            Waiting to Exhale (1995)
4          4        5  Father of the Bride Part II (1995)
…          …        …                                 …
27273  27273   131254         Kein Bund für's Leben (2007)
27274  27274   131256        Feuer, Eis & Dosenbier (2002)
27275  27275   131258                  The Pirates (2014)
27276  27276   131260                 Rentun Ruusu (2001)
27277  27277   131262                   Innocence (2014)

                                            genres
0      Adventure|Animation|Children|Comedy|Fantasy
1                       Adventure|Children|Fantasy
2                                   Comedy|Romance
3                             Comedy|Drama|Romance
4                                           Comedy
…                                              …
27273                                       Comedy
27274                                       Comedy
27275                                    Adventure
27276                           (no genres listed)
27277                     Adventure|Fantasy|Horror

[27278 rows x 4 columns]
```

```
[33]: user_5= movies.copy()
      user_5= user_5.reset_index()
      data = Dataset.load_from_df(ratings[['userId','movieId','rating']],reader)
      trainset=data.build_full_trainset()
      svd.fit(trainset)
      user_5['Estimate_score']=user_5['movieId'].apply(lambda x:svd.predict(1,x).est)
      user_5=user_5.drop(['movieId','genres','index'],axis=1)
      user_5=user_5.sort_values('Estimate_score',ascending=False)
      print(user_5.head(10))
```

```
                                                   title  Estimate_score
4897   Lord of the Rings: The Fellowship of the Ring,…        4.820224
7041   Lord of the Rings: The Return of the King, The…        4.803595
5853       Lord of the Rings: The Two Towers, The (2002)        4.776067
6501                                Umberto D. (1952)        4.605551
6859                         Europa (Zentropa) (1991)        4.509855
6873   Passion of Joan of Arc, The (Passion de Jeanne…        4.503719
6667                     Judgment at Nuremberg (1961)        4.478136
8937                  Decalogue, The (Dekalog) (1989)        4.457044
10407                             Why We Fight (2005)        4.455664
10286                                 Serenity (2005)        4.450090
```

```
[ ]:
```