Week10Assignment

February 16, 2025

create a recommender system

[2]:
```
from surprise.prediction_algorithms.matrix_factorization import SVD
from surprise import Reader, Dataset
from surprise import accuracy
import pandas as pd
from sklearn.preprocessing import LabelEncoder, MultiLabelBinarizer
from sklearn.model_selection import train_test_split
```

[3]:
```
#save the csv files to dataframes
movies_df = pd.read_csv('movies.csv')
ratings_df = pd.read_csv('ratings.csv')
```

[4]:
```
#join the two datasets
df = pd.merge(ratings_df, movies_df[['movieId','genres']], on = 'movieId', how␣
↪= 'left')
```

[5]:
```
df.head()
```

[5]:
```
   userId  movieId  rating  timestamp  \
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                               Comedy|Romance
2                        Action|Crime|Thriller
3                             Mystery|Thriller
4                       Crime|Mystery|Thriller
```

[6]:
```
#create the encoders and multilabelbinarizer, we encode the data to eliminate␣
↪gaps between id variables and make the data cleaner
user_encoder = LabelEncoder()
movie_encoder = LabelEncoder()
mlb = MultiLabelBinarizer()
```

```
df['userId'] = user_encoder.fit_transform(df['userId'])
df['movieId'] = movie_encoder.fit_transform(df['movieId'])

df = df.join(pd.DataFrame(mlb.fit_transform(df.pop('genres').str.split('|')),␣
 ↪columns = mlb.classes_, index = df.index ))
```

[7]:
```
df.head()
```

[7]:
```
   userId  movieId  rating  timestamp  (no genres listed)  Action  Adventure  \
0       0        0     4.0  964982703                   0       0          1
1       0        2     4.0  964981247                   0       0          0
2       0        5     4.0  964982224                   0       1          0
3       0       43     5.0  964983815                   0       0          0
4       0       46     5.0  964982931                   0       0          0

   Animation  Children  Comedy  …  Film-Noir  Horror  IMAX  Musical  \
0          1         1       1  …          0       0     0        0
1          0         0       1  …          0       0     0        0
2          0         0       0  …          0       0     0        0
3          0         0       0  …          0       0     0        0
4          0         0       0  …          0       0     0        0

   Mystery  Romance  Sci-Fi  Thriller  War  Western
0        0        0       0         0    0        0
1        0        1       0         0    0        0
2        0        0       0         1    0        0
3        1        0       0         1    0        0
4        1        0       0         1    0        0

[5 rows x 24 columns]
```

[8]:
```
df.drop(columns = "(no genres listed)", inplace = True)
```

[ ]:
```

```

[9]:
```
train_df, test_df = train_test_split(df, test_size = 0.2)
train_df
```

[9]:
```
       userId  movieId  rating   timestamp  Action  Adventure  Animation  \
5744       40     3617     4.0  1458933682       0          0          0
48363     312     1131     4.0  1030556287       0          0          0
23752     162     1290     3.0   894217532       0          0          0
52880     346        0     5.0   847645986       0          1          1
```

```
94446     598     5701     3.0  1498521577       0          0          0
…       …       …      …          …      …        …         …
69852     447     7263     3.0  1312638561       1          0          0
25344     176     3880     2.5  1435526025       1          0          0
33717     228      412     3.0   838144001       1          0          0
21009     138     6453     1.0  1453924666       1          1          0
73345     473      756     3.0  1060105286       0          0          0

       Children  Comedy  Crime  …  Film-Noir  Horror  IMAX  Musical  \
5744          0       1      0  …          0       0     0        0
48363         0       0      0  …          0       0     0        0
23752         0       0      0  …          0       0     0        0
52880         1       1      0  …          0       0     0        0
94446         0       0      1  …          0       0     0        0
…           …     …     …   …        …      …    …     …
69852         0       0      0  …          0       0     0        0
25344         0       1      0  …          0       0     0        0
33717         0       0      0  …          0       0     0        0
21009         0       0      0  …          0       0     1        0
73345         0       0      0  …          0       0     0        0

       Mystery  Romance  Sci-Fi  Thriller  War  Western
5744         0        1       0         0    0        0
48363        0        0       0         1    0        0
23752        0        1       0         0    0        0
52880        0        0       0         0    0        0
94446        0        0       0         1    0        0
…          …      …      …       …   …     …
69852        0        0       0         1    1        0
25344        0        0       1         0    0        0
33717        0        0       0         1    0        0
21009        0        0       1         1    0        0
73345        0        0       0         0    0        0

[80668 rows x 23 columns]
```

[10]:
```
#create a reader to read the ratings provided
reader = Reader(rating_scale = (0.5, 5))
#load the data
data = Dataset.load_from_df(train_df[['userId', 'movieId', 'rating']], reader)
#build the trainset to store the user item interactions
trainset = data.build_full_trainset()
```

[ ]:

[ ]:

[11]:
```
#creat the svd model and fit it to our tainset
model_svd = SVD()
model_svd.fit(trainset)

predictions_svd = model_svd.test(trainset.build_anti_testset())
#RMSE is .4763
accuracy.rmse(predictions_svd)
```

```
RMSE: 0.4763
```

[11]:
```
0.47629372223372624
```

[ ]:
```

```

[56]:
```
#function to get the recommendations for a user
def get_top_n_recommendations(user_id, n=5):
  #match the userid to the df, then pull the corrisponding movie ids to that␣
 ↪user
  user_movies = df[df['userId'] == user_id]['movieId'].unique()
  all_movies = df['movieId'].unique()
  #predict agains the remaining movies the user has not watched/reviewed
  movies_to_predict = list(set(all_movies) - set(user_movies))
  #loop through the predictions and pull the top n predictions
  user_movie_pairs = [(user_id, movie_id, 0) for movie_id in movies_to_predict]
  predictions_cf = model_svd.test(user_movie_pairs)

  top_n_recommendations = sorted(predictions_cf, key = lambda x: x.est)[:n]

  for pred in top_n_recommendations:
    predicted_rating = pred.est
    print(predicted_rating)

#save the movie ids and return the list of top predictions
  top_n_movie_ids = [int(pred.iid) for pred in top_n_recommendations]

  top_n_movies = movie_encoder.inverse_transform(top_n_movie_ids)

  return top_n_movies
```

[58]:
```
#take user imput for the userid, pass to the previous function and return the␣
 ↪list of results
user_id = input("What userid would you like recomendations for? ")
recommendations = get_top_n_recommendations(user_id)
top_n_movies_titles = movies_df[movies_df['movieId'].
 ↪isin(recommendations)]['title'].tolist()
print(f"Top 5 Recommendations for User {user_id}:")
for i, title in enumerate(top_n_movies_titles, 1):
  print(f"{i}.{title}")
```

```
What userid would you like recomendations for?  123

2.275002968714635
2.349780146197188
2.3609299347617214
2.4386771660129947
2.444613671593667
Top 5 Recommendations for User 123:
1.Speed 2: Cruise Control (1997)
2.Batman & Robin (1997)
3.Godzilla (1998)
4.Wild Wild West (1999)
5.Battlefield Earth (2000)
```

[ ]:

[ ]: