Netflix Recommendation Study

UTRGV CSCI6370 Machine Learning with Dr. Lei HanSheng

Steven Bowler 20562494

This study uses the Kaggle Netflix dataset (https://www.kaggle.com/netflix-inc/netflix-prize-data) as the basis for creating a collaborative filtering model that predicts which movies would be most preferred by a customer based on that customer's previous movie ratings.

The project repo is here (https://github.com/stevenbowler/netflixstudy) on github. The project is in Cookiecutter (https://drivendata.github.io/cookiecutter-data-science/) Data Science project structure.

The study is structured as follows:

1. Data Wrangling (https://github.com/stevenbowler/netflixstudy/blob/master/notebooks/netflixstudyDataWrangling.ipynb)
2. Exploratory Data Analysis - EDA part 1 (https://github.com/stevenbowler/netflixstudy/blob/master/notebooks/netflixstudyEDA.ipynb) which includes homework submission #1
3. Exploratory Data Analysis - EDA part 2 (https://github.com/stevenbowler/netflixstudy/blob/master/notebooks/netflixstudyEDAv3.ipynb) which includes homework submission #2
4. Exploratory Data Analysis - EDA part 3 (https://github.com/stevenbowler/netflixstudy/tree/master/reports) were various attempts to successfully build an SQL netflixstudy database, however, it was decided to continue with pandas and scikitlearn since it afforded better tools for the analysis. Therefore, none of the SQL development was used for this study.
5. Model (https://github.com/stevenbowler/netflixstudy/blob/master/notebooks/netflixstudyModel.ipynb) this same file, the final project submission including the prediction model and predictions below.

Attribution:

1. D. Lao Data Wrangling and Collaborative Filtering (https://www.kaggle.com/stevenbowler/netflix-movie-recommendation/edit)
2. Anjana Tiha Collaborative Filtering (https://github.com/anjanatiha/Movie-Recommendation-Engine-using-User-Based-Collaborative-Filtering)
3. Rhys Shea K-Means Clustering (https://programming.rhysshea.com/K-means_movie_ratings/)

The final prediction model is presented below

In [30]:

```
# Import necessary modules
import pandas as pd
import numpy as np
from pandas_profiling import ProfileReport
import math
import re
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
from sklearn.cluster import KMeans
from scipy.sparse import csr_matrix
import seaborn as sns
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
sns.set_style("darkgrid")
```

Load Movie Titles Dataframe

In [31]:

```
df_title = pd.read_csv('../data/raw/movie_titles.csv', encoding = "ISO-8859-1"
, header = None, names = ['Movie_Id', 'Year', 'Name'])
df_title.set_index('Movie_Id', inplace = True)
print (df_title.head(10))
```

```
            Year                          Name
Movie_Id                                      
1         2003.0               Dinosaur Planet
2         2004.0    Isle of Man TT 2004 Review
3         1997.0                     Character
4         1994.0  Paula Abdul's Get Up & Dance
5         2004.0      The Rise and Fall of ECW
6         1997.0                          Sick
7         1992.0                         8 Man
8         2004.0    What the #$*! Do We Know!?
9         1991.0      Class of Nuke 'Em High 2
10        2001.0                       Fighter
```

Load the full dataset

Load the inline, cleaned, dataset. Will not use pivot table version (df_p.csv) for this study.

This file df.csv was created in the Data-wrangling (https://github.com/stevenbowler/netflixstudy/blob/master/notebooks/netflixstudyDataWrangling.ipynb) and EDA (https://github.com/stevenbowler/netflixstudy/blob/master/notebooks/netflixstudyEDA.ipynb) phases of this study.

In [32]:

```
# load straight cleaned dataset, will not use pivot table version (df_p.csv) f
or this study.
# This file was created in the [Data-wrangling and EDA
df = pd.read_csv('../data/processed/df.csv')
```

In [33]:

```
# drop the bottom 30% of movies with fewest number of ratings to speed things
 up
f = ['count','mean']

df_movie_summary = df.groupby('Movie_Id')['Rating'].agg(f)
df_movie_summary.index = df_movie_summary.index.map(int)
movie_benchmark = round(df_movie_summary['count'].quantile(0.7),0)
drop_movie_list = df_movie_summary[df_movie_summary['count'] < movie_benchmark
].index
```

Collaborative Filtering Recommendation Model

Use collaborative filtering (https://en.wikipedia.org/wiki/Collaborative_filtering), with reduced number of records to test the model, say 250,000 instead of the full 75million in the reduced dataset (eliminated zero ratings and fewest 30% ratings)

NOTE: 250,000 records in the model takes 15 minutes on my PC to make a prediction, so using all 75million records of course 450 minutes appx 7.5 hours for one prediction.

In [34]:

```
df_short = df.head(250000)
```

In [35]:

```
reader = Reader()
data = Dataset.load_from_df(df_short[['Cust_Id', 'Movie_Id', 'Rating']][:], re
ader)
svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'])
```

Out[35]:

```
{'test_rmse': array([0.98569376, 0.98552318, 0.98421558, 0.97671291, 0.980733
38]),
 'test_mae': array([0.78785978, 0.78826079, 0.79064036, 0.78267065, 0.7725084
5]),
 'fit_time': (16.44412636756897,
  15.239561080932617,
  15.418436765670776,
  14.791906833648682,
  16.456958293914795),
 'test_time': (0.6165235042572021,
  0.4303755760192871,
  0.38679981231689453,
  0.41698360443115234,
  98.9626636505127)}
```

Show some customer Ids and run some predictions on what those customers might like to see

```
In [36]: df.head(10)
```

Out[36]:

	Unnamed: 0	Cust_Id	Rating	Movie_Id
0	696	712664	5.0	3
1	697	1331154	4.0	3
2	698	2632461	3.0	3
3	699	44937	5.0	3
4	700	656399	4.0	3
5	701	439011	1.0	3
6	703	1644750	3.0	3
7	704	2031561	4.0	3
8	705	616720	4.0	3
9	706	2467008	4.0	3

Enter Customer_Id of the 1st customer to be used for predictions

```
In [37]: Customer_Id = 1331154
```

Show the above customer's favorite movies

```
In [38]: Customer = df[(df['Cust_Id'] == Customer_Id) & (df['Rating'] == 5)]
         Customer = Customer.set_index('Movie_Id')
         Customer = Customer.join(df_title)['Name']
         print(Customer)
```

```
Movie_Id
143                                      The Game
270                       Sex and the City: Season 4
361            The Phantom of the Opera: Special Edition
457                                 Kill Bill: Vol. 2
482                                             Frida
                                ...
16860                          Law & Order: Season 1
16954                Indiana Jones and the Last Crusade
17085                                   24: Season 2
17627              Harry Potter and the Sorcerer's Stone
17709                           A River Runs Through It
Name: Name, Length: 158, dtype: object
```

Predict which movies customer would like:

In [39]:

```
Customer = df_title.copy()
Customer = Customer.reset_index()
Customer = Customer[~Customer['Movie_Id'].isin(drop_movie_list)]

data = Dataset.load_from_df(df_short[['Cust_Id', 'Movie_Id', 'Rating']], reader)

trainset = data.build_full_trainset()
svd.fit(trainset)

# Customer['Estimate_Score'] = Customer['Movie_Id'].apply(lambda x: svd.predict(785314, x).est)
Customer['Estimate_Score'] = Customer['Movie_Id'].apply(lambda x: svd.predict(Customer_Id, x).est)

Customer = Customer.drop('Movie_Id', axis = 1)

Customer = Customer.sort_values('Estimate_Score', ascending=False)
print(Customer.head(10))
```

```
         Year                                       Name  Estimate_Score
27     2002.0                             Lilo and Stitch        3.929959
29     2003.0                      Something's Gotta Give        3.854963
57     1996.0                                 Dragonheart        3.829985
82     1983.0                                    Silkwood        3.692269
0      2003.0                              Dinosaur Planet        3.583042
11802  2005.0                                       Zeher        3.583042
11804  2004.0                              The Big Bounce        3.583042
11805  1998.0  The Secret of N-I-M-H 2: Timmy to the Rescue        3.583042
11806  1995.0                  Chinese Odyssey 2: Cinderella        3.583042
11807  1977.0                                 Eaten Alive        3.583042
```