DATA-DRIVEN MOVIE RECOMMENDATION SYSTEM

Ruiyuan Xu

ABSTRACT

In the era of information overload, personalized recommendation systems play a crucial role in filtering and delivering relevant content to users. This project presents a baseline Movie Recommendation System built using the Netflix Prize data. The system employs collaborative filtering, a common approach in recommendation systems, to predict the rating or preference that a user would give to an item.

The project also explores the use of hybrid methods with correlations for generating recommendations. The system aims to enhance the quality of search results and provide items that are more relevant to the user's search history or preferences.

OBJECTIVE

•Learn from Netflix Prize data and recommend best TV shows or movies to users, based on self & others behavior.

METHODS

•Collaborative filtering;

•Correlation or other hybrid methods.

•The recommendation algorithm is based on the Surprise library, a Python scikit for building and analyzing recommender systems.

Proposal

DATA SOURCE

NETFLIX PRIZE DATA

Movie ID

range from 1 to 17770 sequentially.

Customer ID

range from 1 to 2649429, with gaps. There are 480189 users.

Rating

on a five-star (integral) scale from 1 to 5.

Date they gave the ratings

have the format YYYY-MM-DD.

DATA MANIPULATION

Data loading:

Read data from dataset.

Data viewing:

Understand user rating trends and try to find reasons.

Data cleaning:

Efficiently add Movie ID column to the data frame, and wash out the invalid data.

Data slicing:

Improve data quality by removing unpopular movies and less active customers.

RECOMMENDATION MODELS

- Mainly use collaborative filtering,
  a common approach in recommendation systems

Evaluation:
Use cross-validation to evaluate the model.
(RMSE & MAE)

- Also use hybrid methods with correlations

TIMELINE

DAY 1~2 :

data preprocessing and exploratory data analysis.

DAY 3~4 :

implementing and adjusting recommendation models.

DAY 5~6 :

evaluating results and preparing the final report.

MILESTONES

Data Manipulation (Understanding, Processing and Warehousing)

Data Loading

Our dataset is comprised of four distinct files, each containing the following attributes: Movie ID, Customer ID, Rating, and Date. An additional file provides a mapping of the Movie ID to its corresponding background information, such as the name of the movie and its release year. We successfully loaded these four datasets and merged them with the background dataset to create a comprehensive data frame.

```
Dataset 1 shape: (24058263, 2)
-Dataset examples-
          Cust_Id  Rating
0               1:     NaN
5000000   2560324     4.0
10000000  2271935     2.0
15000000  1921803     2.0
20000000  1933327     3.0
```

```
Full dataset shape: (24058263, 2)
-Dataset examples-
          Cust_Id  Rating
0               1:     NaN
5000000   2560324     4.0
10000000  2271935     2.0
15000000  1921803     2.0
20000000  1933327     3.0
```

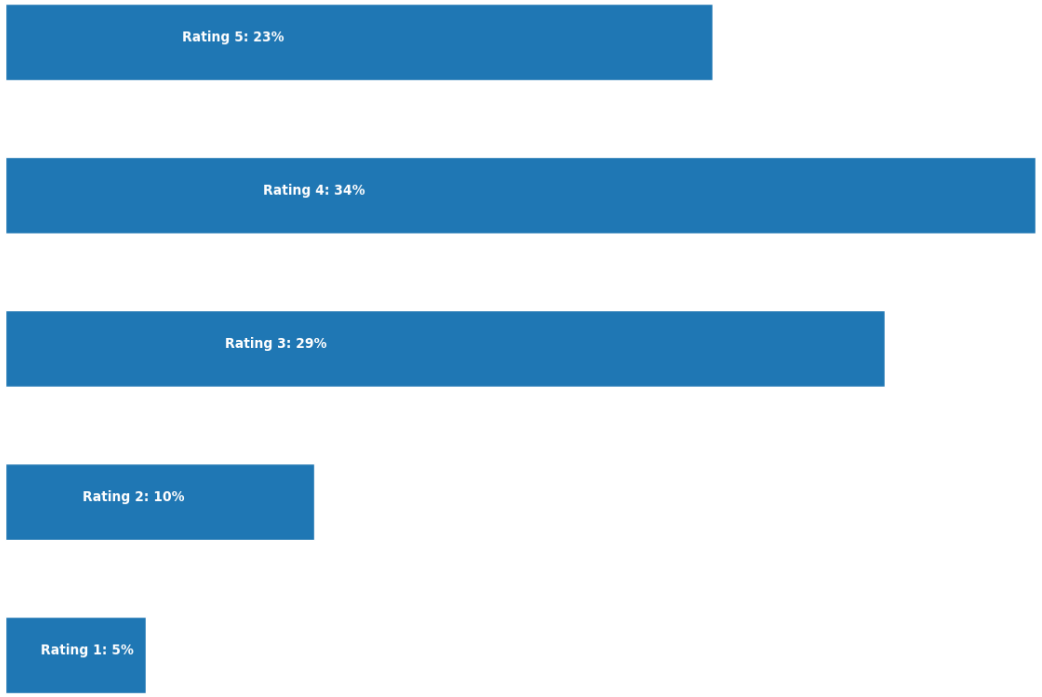
MILESTONES

Data Manipulation (Understanding, Processing and Warehousing)

Data Visualization & Understanding

Upon reviewing the data, we noted a tendency for the ratings to skew positively, with most ratings exceeding 3. This observation could be attributed to the likelihood that dissatisfied customers often choose not to rate at all, rather than providing a low rating. This is an important consideration for our analysis, as it suggests that movies with low ratings are generally of poor quality.

Total: 17,770 Movies; 480,189 customers; 100,480,507 ratings given



MILESTONES

Data Manipulation (Understanding, Processing and Warehousing)

Data Cleaning

The importation of the Movie ID presented a significant challenge due to its disorganized nature. A direct loop through the dataframe to add the Movie ID column proved to be inefficient and exceeded the memory capacity of the Kernel. To overcome this, we devised an alternative approach: we first created a numpy array of the correct length, then added this array as a column to the main dataframe. This method proved to be both efficient and effective.

```
# Iterate through consecutive NaN index pairs to assign movie IDs
for i,j in zip(df_nan['index'][1:],df_nan['index'][:-1]):
    # numpy approach
    temp = np.full((1,i-j-1), movie_id)
    movie_np = np.append(movie_np, temp)
    movie_id += 1

# Account for the last record and its corresponding length
# Numpy approach: Create an array of movie_id with length (len(df) - df_nan.iloc[-1, 0] - 1)
last_record = np.full((1, len(df) - df_nan.iloc[-1, 0] - 1), movie_id)
movie_np = np.append(movie_np, last_record)

# Display the resulting movie numpy array and its length
print('Movie numpy: {}'.format(movie_np))
print('Length: {}'.format(len(movie_np)))
```

```
Movie numpy: [1.000e+00 1.000e+00 1.000e+00 ... 4.499e+03 4.499e+03 4.499e+03]
Length: 24053764
```

MILESTONES

Data Manipulation (Understanding, Processing and Warehousing)

Data Slicing

To address memory limitations and potential bias, we implemented data slicing. We removed movies and customers with minimal reviews, optimizing space utilization and increasing the statistical significance of our results.

```
# Determining the review count threshold for movies
movie_benchmark = round(df_movie_summary['count'].quantile(0.7), 0)

# Identifying movies with review counts below the threshold
drop_movie_list = df_movie_summary[df_movie_summary['count'] < movie_benchmark].index

# Displaying the movie review count threshold
print('Movie minimum times of review: {}'.format(movie_benchmark))

# Grouping by Cust_Id and calculating aggregated statistics
df_cust_summary = df.groupby('Cust_Id')['Rating'].agg(f)

# Converting the index to integers
df_cust_summary.index = df_cust_summary.index.map(int)

# Determining the review count threshold for customers
cust_benchmark = round(df_cust_summary['count'].quantile(0.7), 0)

# Identifying customers with review counts below the threshold
drop_cust_list = df_cust_summary[df_cust_summary['count'] < cust_benchmark].index

# Displaying the customer review count threshold
print('Customer minimum times of review: {}'.format(cust_benchmark))
```

```
Movie minimum times of review: 1799.0
Customer minimum times of review: 52.0
```

MILESTONES

Recommendation Models:
Collaborative Filtering

We employed Collaborative Filtering for our recommendation model. The model was trained using the Surprise library's SVD algorithm.

Model Training

The model was trained using 3-fold cross-validation. For each fold, the model was fitted with the training set and then predictions were made on the test set. The root mean square error (RMSE) and mean absolute error (MAE) were calculated for each fold to evaluate the model's performance. The results were as follows:

•RMSE: 0.9893, MAE: 0.7988

•RMSE: 0.9948, MAE: 0.7993

•RMSE: 0.9901, MAE: 0.7955

User Preference Analysis

We analyzed the past preferences of user 401047 by identifying the movies they rated highly (5 stars). This provided us with an understanding of the user's preferences, which included movies such as "The Game", "Michael Collins", "Speed", and "Jaws", among others.

Movie Recommendations

We then used our trained model to predict which movies user 401047 would enjoy. The model estimated a score for each movie, and the top 10 movies with the highest scores were selected as recommendations. The recommended movies included "I Love Lucy: Season 2", "Lilo and Stitch", "Richard III", and "Aqua Teen Hunger Force: Vol. 1", among others.


```
# cross_validate(svd, data, measures=['RMSE', 'MAE'])
trainset = data.build_full_trainset()
svd.fit(trainset)
```

```
RMSE: 0.9893
MAE:  0.7988
RMSE: 0.9948
MAE:  0.7993
RMSE: 0.9901
MAE:  0.7955
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7aa1e0e26a70>
```

```
df_401047 = df[(df['Cust_Id'] == 401047) & (df['Rating'] == 5)]
df_401047 = df_401047.set_index('Movie_Id')
df_401047 = df_401047.join(df_title)['Name']
print(df_401047)
```

```
Movie_Id
143                                    The Game
445                              Michael Collins
607                                       Speed
798                                        Jaws
886                                         Ray
985                                   The Mummy
1073                              Coach Carter
1173                     Walking with Dinosaurs
1220                                  Man on Fire
1625                 Aliens: Collector's Edition
```

```
print(user_401047.head(10))
```

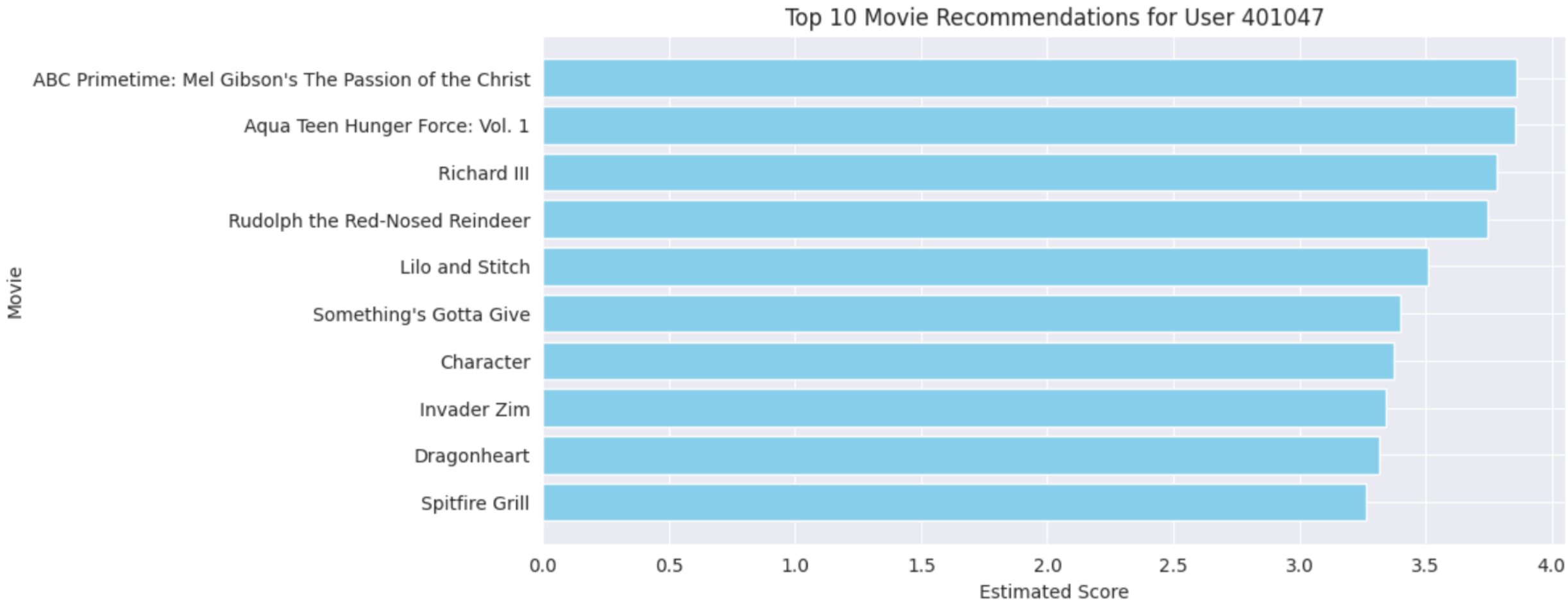
```
          Year                                             Name  \
75      1952.0                          I Love Lucy: Season 2
27      2002.0                                  Lilo and Stitch
56      1995.0                                    Richard III
32      2000.0                     Aqua Teen Hunger Force: Vol. 1
31      2004.0  ABC Primetime: Mel Gibson's The Passion of the...
57      1996.0                                     Dragonheart
12894   1997.0                                         Ivanhoe
12895   1994.0                          Bullets Over Broadway
12893   1960.0                              The Time Machine
12896   2002.0                                    Unspeakable

       Estimate_Score
75           3.957108
27           3.834872
56           3.651427
32           3.629310
31           3.508568
57           3.388175
12894        3.281208
12895        3.281208
12893        3.281208
12896        3.281208
```

MILESTONES

Recommendation Models:

Collaborative Filtering



MILESTONES

Recommendation Models: Pearson's R Correlations

In this approach, we employed Pearson's R correlation as a metric to quantify the linear relationship between the review scores of all possible pairs of movies. We then presented a list of the top ten movies that exhibit the highest correlations. This method allowed us to identify films that are highly regarded in a similar manner, thereby providing valuable insights for recommendations.

We defined a function, recommend(movie_title, min_count), that prints the top 10 movies recommended based on Pearson's R correlation for a given movie. This function retrieves the similarity scores of the given movie with all other movies, computes the pairwise correlation, and then sorts the movies based on the correlation scores.

We used this function to generate recommendations for two movies: "Die Hard 2: Die Harder" and "Braveheart". The top recommended movies for these two were found to be highly correlated in terms of review scores.

For "Die Hard 2: Die Harder", the top recommended movies included "Lethal Weapon 3", "Missing in Action 2: The Beginning / Missing in Action 3: Braddock", and "Under Siege 2: Dark Territory", among others.

For "Braveheart", the top recommended movies included "The Last Samurai", "Lethal Weapon", and "Operation Pacific", among others.

```
recommend("Die Hard 2: Die Harder", 0)
```

```
For movie (Die Hard 2: Die Harder)
- Top 10 movies recommended based on Pearsons'R correlation -
 PearsonR                                                  Name  count      mean
 1.000000                                  Die Hard 2: Die Harder  62809 3.660065
 0.503048                                       Lethal Weapon 3  75148 3.813807
 0.455355 Missing in Action 2: The Beginning / Missing in Action 3: Braddock   2297 3.057466
 0.443854                                Under Siege 2: Dark Territory   9063 3.296591
 0.442952                                       Lethal Weapon 113377 3.971317
 0.440787                                       Out for Justice   6660 3.392042
 0.434394                                Rambo: First Blood Part II  19240 3.432952
 0.431696                                                 Rocky V   8170 3.033660
 0.420941                      The Substitute/The Substitute 3   2260 2.684513
 0.419061                                       Double Impact   6697 2.862476
```

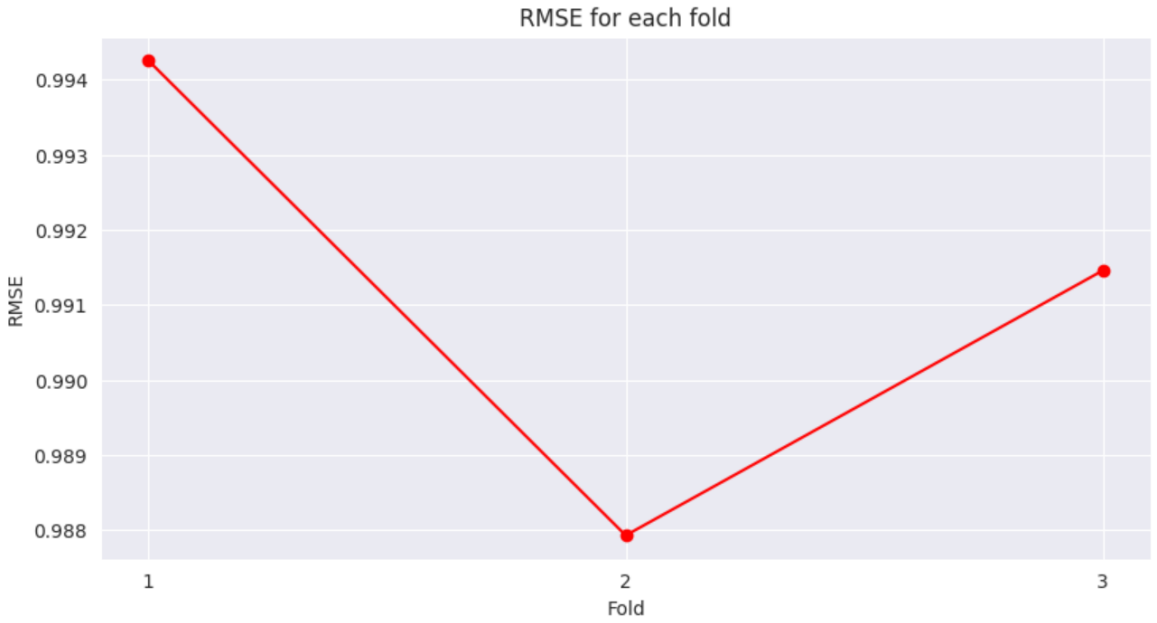
```
recommend("Braveheart", 0)
```

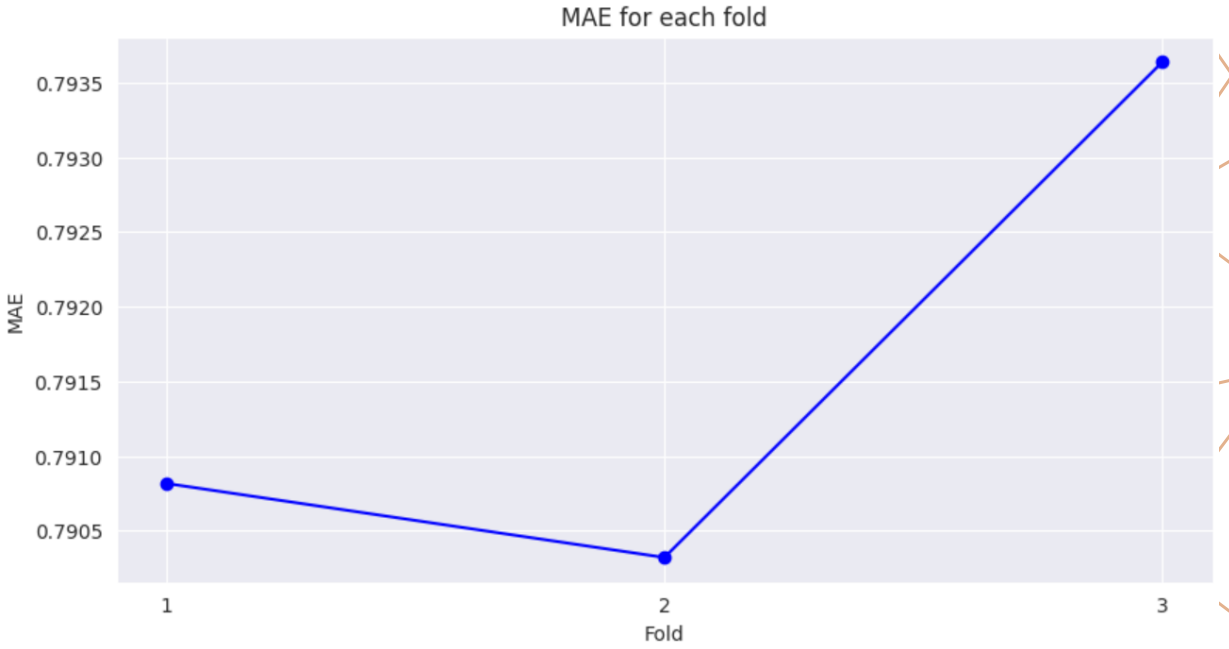
```
For movie (Braveheart)
- Top 10 movies recommended based on Pearsons'R correlation -
 PearsonR                          Name  count      mean
 1.000000                     Braveheart 135601 4.294423
 0.292888                The Last Samurai 139428 3.946596
 0.289452                   Lethal Weapon 113377 3.971317
 0.288157              Operation Pacific   1839 3.762915
 0.285428          The Fighting Seabees   4155 3.514320
 0.280299                        Rio Lobo   3693 3.832656
 0.279053 Dragon Ball Z: Super Android 13   2426 3.428689
 0.275931                        Rob Roy  11409 3.589009
 0.271564          The Devil's Brigade   2236 3.538909
 0.261248   Robin Hood: Prince of Thieves  44001 3.632917
```

MILESTONES

Recommendation Models:
Evaluation

After training our model, we use it to make predictions. The model estimates a score for each movie, which represents the predicted rating or preference that a user would give to that movie. These scores are then used to generate recommendations.





CONCLUSION

Through this project, we have successfully built a movie recommendation system that can predict the rating or preference that a user would give to an item. The system is capable of making recommendations based on both collaborative filtering and Pearson's R correlations. The project has demonstrated the effectiveness of these methods in creating a recommendation system and has provided valuable insights for further improvements and refinements.

As we move forward, we aim to refine our model by incorporating more sophisticated techniques and exploring other types of recommendation systems. We believe that the future of recommendation systems lies in the ability to effectively combine different methods and to adapt to the unique preferences and behaviors of individual users.

THANK YOU!

References

- https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75
- https://www.kaggle.com/code/rounakbanik/movie-recommender-systems/notebook
- **Netflix Prize Data：** Netflix Prize data | Kaggle
- 协同过滤 | Machine Learning | Google for Developers
- https://en.wikipedia.org/wiki/Collaborative_filtering
- Pearson correlation in R | R-bloggers
- [1901.03888] Hybrid Recommender Systems: A Systematic Literature Review (arxiv.org)