Big Data Analytics Experience

Final Report

on

NETFLIX MOVIE RECOMMENDATIONS

By:

The Real OG’s

Vishal Gardas

Rahul Reddy Ganga

Bharath Chandra Nelluri

Lakshmi Samskruthi Reddy

Enuma Edmund

Under the guidance of

Dr. Vijay Gandapodi

Robinson college of Business

Georgia State University

Contents

Abstract

This paper presents a comprehensive study and proposal for a Netflix recommendation system aimed at predicting movie ratings for a subset of users from the original Netflix Prize challenge dataset. The primary objective is to enhance the accuracy of predicting user preferences, thereby improving user satisfaction, content consumption, and retention rates on the Netflix platform. Key components of the proposal include data sourcing from the Netflix Prize dataset, extensive data cleaning and preprocessing, and the implementation of an efficient ETL (Extract, Transform, Load) process. The system architecture integrates data storage on Amazon S3, leveraging Amazon EMR and PySpark for scalable data handling and processing. The core of the recommendation engine is built upon advanced Collaborative Filtering techniques, encompassing both User-User and Item-Item based approaches, and is further enriched with Implicit Feedback and Reinforcement Based algorithms. The paper highlights the challenges encountered, such as the Cold Start problem, data volume management, ensuring low latency, and adapting to dynamically changing user preferences. Our findings demonstrate the potential of this system to significantly improve recommendation accuracy, which is a critical factor for user engagement and satisfaction in the streaming service industry.

1. Introduction

Overview of the Netflix Recommendation System

Netflix, a leader in the streaming service industry, has revolutionized the way content is delivered to users. Central to its success is its sophisticated recommendation system, designed to personalize the viewing experience by accurately predicting and suggesting movies and TV shows that align with individual user preferences. This system employs a complex algorithm that analyses vast amounts of data, including user interactions, viewing history, and content characteristics, to deliver tailored content suggestions.

Importance of Accurate Movie Predictions

The accuracy of these predictions is paramount. It not only drives user engagement by enhancing the viewing experience but also contributes significantly to user retention and satisfaction. Accurate predictions ensure that users are exposed to content that resonates with their tastes, thereby increasing the time spent on the platform and fostering a deeper connection with the service. In a highly competitive market, the ability to deliver precise recommendations can be a substantial differentiator.

In 2006, Netflix launched the Netflix Prize challenge, a landmark competition in the field of machine learning and data science. The challenge was to develop a recommendation algorithm that could outperform Netflix's existing system in predicting user ratings for films. The dataset provided for this competition included over 100 million movie ratings from a vast pool of Netflix subscribers, offering a unique opportunity for researchers and data scientists to innovate in the realm of recommendation systems.

Purpose and Scope of the Paper

This paper aims to propose a new recommendation system for Netflix, particularly focused on predicting movie ratings for a subset of the original Netflix Prize data. The system is designed to handle large-scale data efficiently, promote content diversity, address the cold start problem, and provide real-time updates. The scope of this study encompasses the identification and utilization of appropriate data sources, meticulous data cleaning processes, effective ETL (Extract, Transform, Load) strategies, and robust data storage solutions. The proposed system's success will be evaluated based on improvements in user satisfaction, increased content consumption, and higher retention rates, all while ensuring respect for user privacy.

1.1 Problem Statement

Objective

The primary goal of this project is to accurately predict 100,000 movie ratings for a subset of users utilizing data from the Netflix Prize challenge. This ambitious task is not just about predicting ratings; it's about comprehensively understanding user preferences and delivering highly personalized content recommendations.

Significance of Predicting Movie Ratings

Predicting movie ratings effectively is crucial for several reasons. First, it directly impacts user engagement. Viewers are more likely to continue using a service that consistently recommends movies that align with their tastes. Second, accurate predictions foster trust in the recommendation system, which is essential for maintaining a loyal user base in a competitive streaming landscape. Third, by effectively predicting ratings, the system can also guide content acquisition strategies, helping Netflix invest in or produce content that resonates with its audience.

Technical Challenges

To accomplish this, the system must overcome several technical hurdles. It needs to process and analyze large-scale data efficiently. The system must also be adept at encouraging content diversity, avoiding the common pitfall of echo chambers where users are only recommended content similar to what they have already seen. Addressing the cold start problem — making accurate recommendations for new users or new content with limited data — is another critical challenge. Additionally, the system must be capable of real-time updates to keep pace with constantly changing user preferences and new content additions.

Success Metrics

Success in this endeavor will be measured through multiple lenses:

- **User Satisfaction:** Improvement in user satisfaction, gauged through surveys, feedback, and engagement metrics, will be a key indicator of the system's performance.
- **Content Consumption:** An increase in the volume of content consumed, indicated by longer viewing times and more frequent log-ins, will suggest that users are finding more content that appeals to them.
- **Retention Rates:** Higher retention rates will indicate that the recommendation system is successful in keeping users engaged over longer periods.

- **Privacy Compliance:** All these objectives must be achieved while fully respecting user privacy, ensuring that data is used ethically and in compliance with relevant regulations.

2. Data Source

Netflix Prize Dataset

The core of our recommendation system proposal is based on the Netflix Prize dataset, a rich collection of movie ratings from Netflix users. This dataset includes over 100 million ratings, providing a comprehensive foundation for predictive analytics.

Dataset Structure

The dataset is divided into several files: **combined_data_1.txt**, **combined_data_2.txt**, **combined_data_3.txt**, and **combined_data_4.txt**. Each file consists of a list of movie IDs, followed by user IDs, ratings, and the dates of the ratings. This structure necessitates sophisticated data processing techniques for effective utilization.

Sample combined data:

```
1:
1488844,3,2005-09-06
822109,5,2005-05-13
885013,4,2005-10-19
30878,4,2005-12-26
```

Movie Titles CSV

The **movie_titles.csv** file supplements the ratings data, offering crucial metadata like movie IDs, release years, and titles. This information enriches the recommendation system, allowing for more nuanced content recommendations.

Movie titles:

2 2004 Isle of Man TT 2004 Review

3 1997 Character

4 1994 Paula Abdul's Get Up & Dance

5 2004 The Rise and Fall of ECW

6 1997 Sick

Data Sources

- The dataset is accessible on Kaggle, providing a platform for comprehensive data analysis.
- https://www.kaggle.com/netflix-inc/netflix-prize-data
- GitHub hosts a range of projects and code related to the Netflix Prize, found at GitHub - Netflix Prize Topics.
- https://github.com/topics/netflix-prize

Data Utilization

Our system will leverage this data to predict movie ratings effectively, harnessing the depth and breadth of the dataset to enhance recommendation accuracy.Data Cleaning and Preprocessing

In developing our Netflix recommendation system, meticulous data cleaning and preprocessing are crucial to ensure the integrity and usability of the Netflix Prize dataset. This section outlines the key steps and techniques employed in this process.

Techniques for Data Cleaning

- **Duplicate Removal**: To ensure accuracy in our analysis, we first eliminate any duplicate entries in the dataset. Duplicate records can lead to skewed results and misrepresentations in predictive modelling.
- **Handling Missing Values**: Missing data is a common issue in large datasets. We employ strategies such as data imputation or selective removal of missing entries, depending on the nature and extent of the missing data. This approach helps maintain the dataset's integrity without compromising its size and diversity.
- **Outlier Detection and Management**: Outliers can significantly affect the outcomes of predictive models. We identify and analyse outliers to determine if they represent genuine variations or data entry errors. Depending on this assessment, outliers are either adjusted or removed to enhance the dataset's overall quality.

Addressing Data Inconsistencies

- **Standardization of Data Formats**: We standardize various data formats, particularly date and time entries, to a consistent format across the dataset, facilitating easier manipulation and analysis.
- **Error Rectification**: Any inconsistencies or errors in the data, such as mismatched movie titles or incorrect user ratings, are identified and corrected. This step is crucial for maintaining the dataset's accuracy and reliability.

3. Project Architecture

Overview of the Recommendation System Architecture

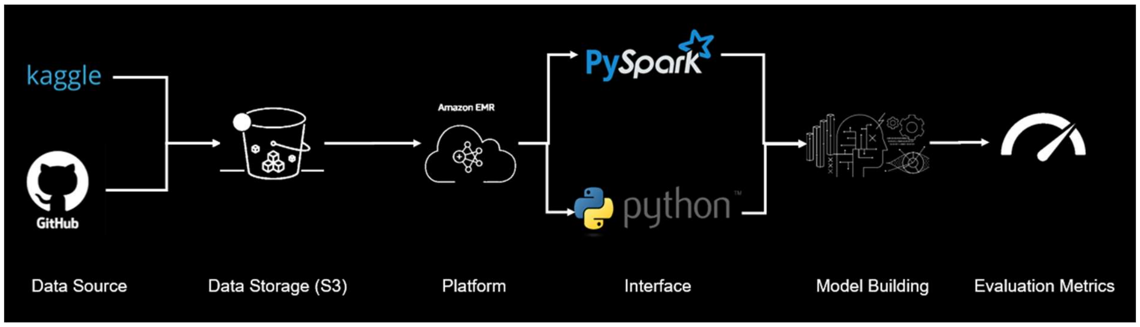
Our Netflix recommendation system is structured to efficiently process large-scale datasets and deliver personalized content recommendations. It integrates various technologies and platforms for optimal performance.

Role of Amazon EMR and PySpark

- **Amazon EMR**: Elastic MapReduce (EMR) from Amazon provides a managed cluster platform that simplifies running big data frameworks, like Apache Spark and Hadoop, to process vast amounts of data.
- **PySpark**: PySpark, the Python API for Apache Spark, is utilized for its ability to handle large-scale data processing and machine learning tasks. It enables us to perform complex data transformations and analysis efficiently.

Data Flow and Processing Pipeline

- **Data Ingestion**: Data from the Netflix Prize dataset is ingested into the system.
- **Preprocessing and Storage**: The data is preprocessed using PySpark and stored in Amazon S3.
- **Data Processing with EMR**: The stored data is then processed using EMR clusters, where heavy lifting like model training occurs.
- **Recommendation Generation**: The system generates personalized recommendations based on user preferences and movie attributes.
- **Output Delivery**: These recommendations are then made available to the user interface for user consumption.



3.1. Data Source

The project leverages a diverse set of datasets from various sources to enrich the movie recommendation engine, ensuring a comprehensive and varied recommendation experience.

3.1.1 Kaggle

Kaggle, a platform for predictive modelling and analytics competitions, provides access to a wealth of datasets related to movies, user preferences, and ratings. The inclusion of Kaggle datasets adds depth and diversity to the recommendation engine's training data.

3.1.2 GitHub

GitHub repositories serve as valuable sources of datasets and collaborative efforts from the open-source community. Exploring GitHub enhances the project's data repository by tapping into community-driven initiatives and datasets.

3.1.3 Netflix Prize Dataset

At the core of the project is the Netflix Prize dataset, a significant collection of historical user ratings and implicit feedback. This dataset serves as the foundation for training, testing, and evaluating the recommendation models, providing real-world insights into user preferences.

3.2. Data Storage

Efficient and secure storage of the diverse datasets is crucial for seamless data processing and model building. Amazon S3 is chosen as the primary storage solution for its scalability and reliability.

3.2.1 Amazon S3

Amazon S3, a highly scalable object storage service, acts as the central repository for storing datasets. Its durability and accessibility make it an ideal choice for managing large volumes of data, ensuring quick and reliable access for processing tasks.

3.3. Amazon EMR

To handle the processing of large-scale data efficiently, Amazon EMR, a cloud-based big data platform, is employed.

3.3.1 Amazon EMR

Amazon EMR simplifies the processing of vast datasets by providing a scalable and cost-effective solution. The distributed computing capabilities of EMR enable parallelized data processing, optimizing the execution of complex algorithms for recommendation model training.

3.4. Interface

A user-friendly and versatile interface is essential for interacting with datasets and conducting data transformations. PySpark, a Python API for Apache Spark, is selected for its flexibility and seamless integration with the Spark ecosystem.

3.4.1 PySpark

PySpark simplifies the development of data processing and machine learning workflows by offering a Python interface to Spark's capabilities. Its ease of use and compatibility with Python libraries contribute to a streamlined and efficient development process.

3.4.2 Python

Python leverage libraries like NumPy and Pandas for data analysis, integrate machine learning with scikit-learn or TensorFlow, automate tasks with Python scripts, and ensure cross-platform compatibility, enhancing your project's functionality and efficiency.

3.5. Model Building

The heart of the project involves constructing recommendation models, allowing for flexibility in choosing machine learning algorithms based on project requirements.

3.5.1 Flexible Model Selection

The model building stage is designed to be flexible, accommodating various machine learning models such as K-Nearest Neighbors (KNN), XGBoost, or any other algorithms deemed suitable for the task. This flexibility empowers the project to explore and experiment with different approaches to enhance recommendation accuracy.

3.6. Model Evaluation

Ensuring the effectiveness and accuracy of the recommendation models is paramount, and a robust evaluation process is implemented.

3.6.1 Evaluation Metrics

Common evaluation metrics specific to recommendation systems, including Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), are employed. These metrics provide insights into the accuracy and performance of the recommendation engine, guiding further model refinement.

4. ETL Process

In the development of our Netflix recommendation system, the Extract, Transform, Load (ETL) process plays a pivotal role in ensuring the data is accurately prepared and processed for effective utilization.

4.1. Extract

The extraction phase involves collecting data from the following sources:

- **Ratings Data**: Comprising of files **combined_data_1.txt**, **combined_data_2.txt**, **combined_data_3.txt**, and **combined_data_4.txt**, these files collectively contain user ratings for various movies.
- **Movie Metadata**: The **movie_titles.csv** file, which provides essential information about each movie, including its title and year of release.

4.2. Transform

The transformation phase is critical in combining and processing the data:

- **Dataset Integration**: The first step is to merge the ratings data with the movie metadata. This amalgamation results in a unified dataset that encapsulates both user preferences and movie details.
- **Data Processing**: The integrated dataset undergoes extensive cleaning (removal of duplicates, handling of missing values), normalization of numerical features, and encoding of categorical variables. This step ensures the data is in a format suitable for machine learning applications.

- **Data Splitting**: Post-processing, the dataset is divided into two distinct segments: one for training the model and another for testing its predictive capabilities.

4.3. Load

The load phase is centred around organizing the processed data for model application:

- **Training Data**: This portion of the data is used to build and train the model, enabling it to learn and adapt to user preferences and movie attributes.
- **Testing Data**: This data set is crucial for evaluating the model's effectiveness and accuracy in predicting user ratings for movies.

4.4. Data Storage

Role of Amazon S3 in Data Storage

For our Netflix recommendation system, Amazon Simple Storage Service (S3) is employed as the primary data storage solution. Amazon S3 offers secure, scalable, and highly durable object storage, making it an ideal choice for handling the vast amount of data involved in this project.

Advantages of Using Cloud Storage for Large Datasets

- **Scalability**: Amazon S3 can easily scale up or down based on the data storage needs, accommodating the growing volume of data without the need for physical infrastructure expansion.
- **Durability and Availability**: It ensures high durability and availability, protecting data against loss and ensuring it is always accessible when needed.

- **Cost-Effectiveness**: Amazon S3 provides a cost-effective storage solution, with a pay-as-you-go model that eliminates the need for large upfront investments in data storage infrastructure.
- **Data Security**: Amazon S3 offers robust security features, ensuring data is protected both in transit and at rest.

Data Retrieval and Management Strategies

- **Organized Data Structure**: Data is organized in a structured manner within S3 buckets, facilitating easy and efficient retrieval.
- **Access Management**: Strict access controls and policies are implemented to regulate who can access or modify the stored data.
- **Backup and Recovery**: Regular backups are conducted to ensure data integrity and quick recovery in case of accidental deletion or data corruption.
- **Integration with Data Processing Tools**: Amazon S3 seamlessly integrates with various data processing and analytics tools, allowing for efficient data manipulation and analysis.

5. Challenges

5.1. Cold Start

Addressing the Cold Start Problem

- **Challenge**: The cold start problem refers to the difficulty in making accurate recommendations for new users or items due to the lack of historical data.
- **Solution**: We plan to utilize content-based filtering for new items and demographic-based recommendations for new users. Additionally, leveraging

techniques like asking users to rate a few items upon signup can provide initial data to kickstart personalized recommendations.

5.2. Data Volume

Managing Large Volumes of Data

- **Challenge**: Handling and processing the massive dataset from Netflix is a significant challenge.
- **Solution**: We will use scalable cloud storage solutions like Amazon S3, coupled with distributed computing platforms like Amazon EMR, to efficiently manage and process large datasets.

5.3. Low Latency

Ensuring Low Latency in Recommendations

- **Challenge**: Maintaining a low response time is critical for user satisfaction, especially when providing real-time recommendations.
- **Solution**: Optimizing algorithms for efficiency and implementing caching strategies for frequently accessed data can help reduce latency. Additionally, using powerful computational resources and streamlining the data pipeline will contribute to faster processing times.

5.4. Adaption to Changing User Preferences

Adapting to Changing User Preferences

- **Challenge**: Users' preferences can change over time, requiring the system to adapt continuously.
- **Solution**: Implementing machine learning models that learn from ongoing user interactions and feedback will help in adapting to changing preferences. Regular

model retraining and updates are also essential to keep the recommendations relevant.

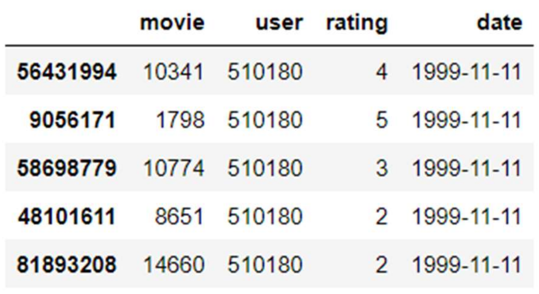
6. Analysis/Insights

Analytics in Netflix Recommendation Engines involves the comprehensive analysis of user interactions, viewing patterns, and content preferences. Leveraging vast datasets, Netflix employs various techniques, including collaborative filtering and content-based filtering, to understand individual user tastes and preferences.

Insights derived from analytics drive continuous improvements in the recommendation algorithms. By evaluating user feedback, watching habits, and engagement metrics, Netflix gains valuable insights into evolving viewer preferences. This iterative process ensures that the recommendation engines stay adaptive and relevant in the ever-changing landscape of viewer behaviour.

Data Processing and Time-Based Analysis

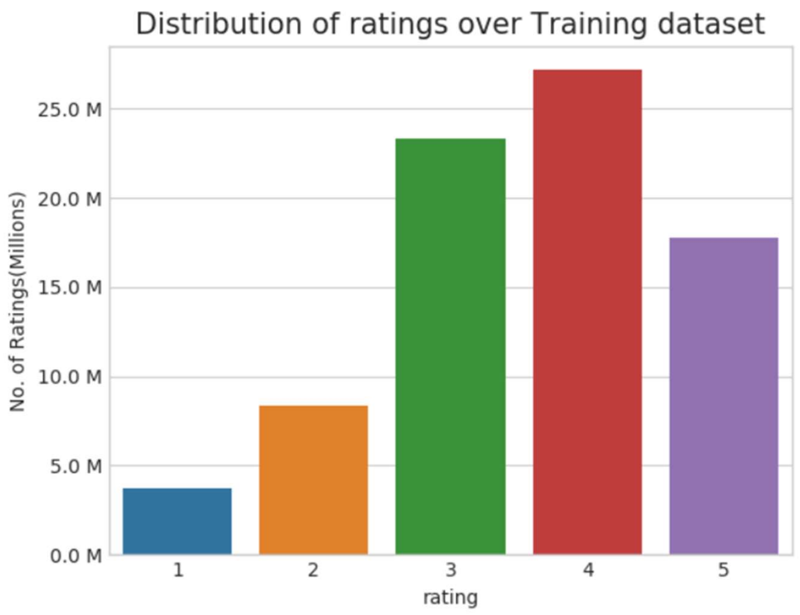
In this step, we transformed raw data from 'data.csv' into a pandas DataFrame(‘df’) with clear columns for movies, users, ratings, and dates. The 'date' column was converted to DateTime format to optimize temporal analysis. The 'df' is sorted in ascending order based on the 'date' column for an organized coherent exploration of time-based patterns, providing a solid base for further analysis. Figure () shows the first 5 rows of the DataFrame(df), displaying information about movies, users, ratings, and dates. The dataset comprises 100,480,507 ratings from 480,189 users for a total of 17,770 movies, providing a comprehensive overview of user interactions and content diversity as shown



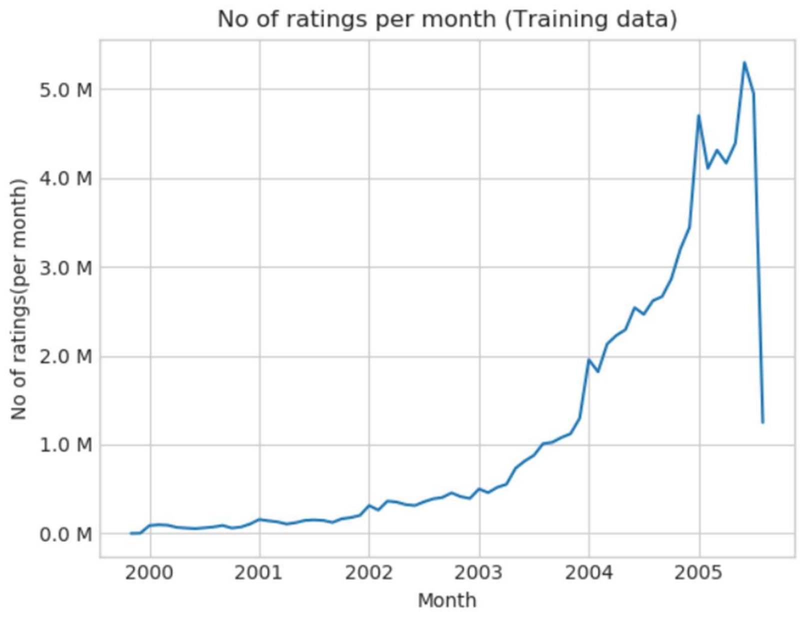
	movie	user	rating	date
56431994	10341	510180	4	1999-11-11
9056171	1798	510180	5	1999-11-11
58698779	10774	510180	3	1999-11-11
48101611	8651	510180	2	1999-11-11
81893208	14660	510180	2	1999-11-11

```
Total data
--------------------------------

Total no of ratings : 100480507
Total No of Users   : 480189
Total No of movies  : 17770
```

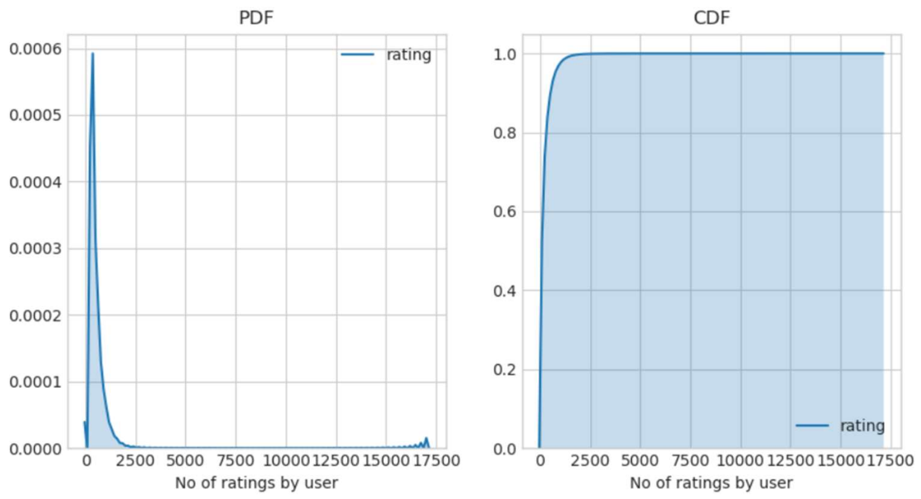
The distribution of ratings is skewed to the right, with most ratings being 4 or 5 stars. This suggests that users are more likely to rate something if they really liked it. There are also a significant number of 3-star ratings, which indicates that the dataset contains a variety of items of different quality.



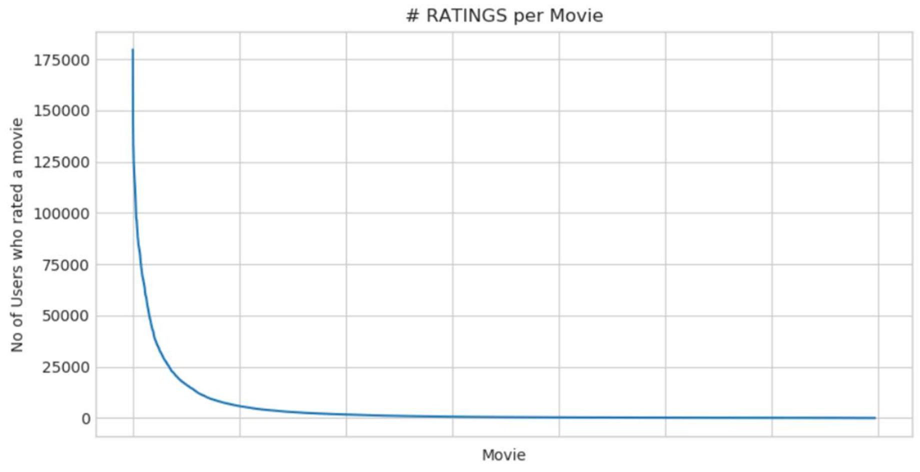
The number of ratings is increasing steadily over time. Which suggests that users are more engaging over time. There are some seasonal variations in the number of ratings.



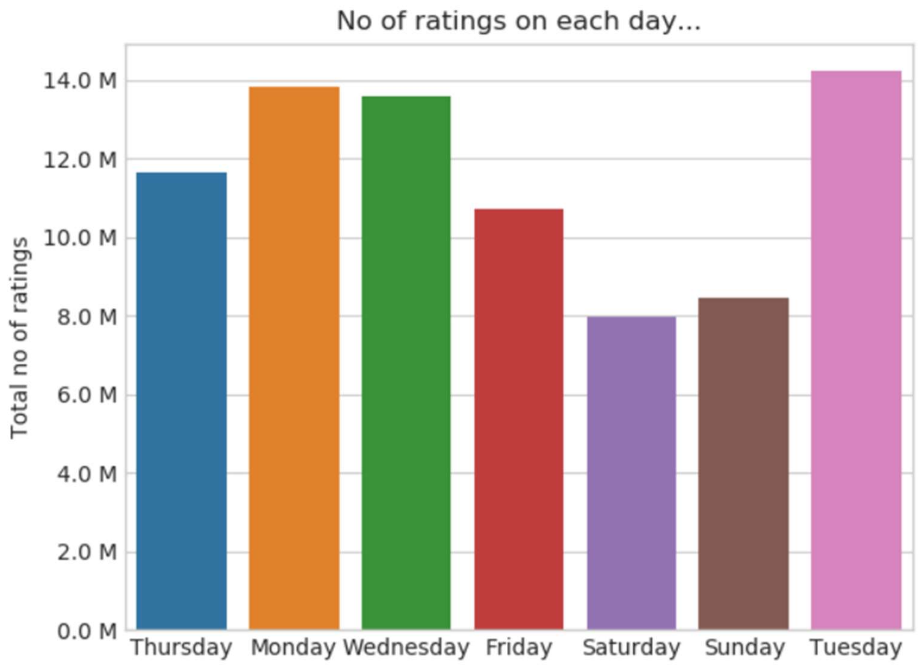
This graph visualizes user rating distribution in a movie recommendation system. The left side shows the PDF, with most users (peak around 25) having rated 10-50 movies. The right side's CDF reveals that 60% of users have rated less than 50 movies, while 90% have rated less than 100. This indicates that most users haven't explored a vast selection, suggesting significant potential for the system to discover and recommend new movies, enhancing user engagement and overall platform success.



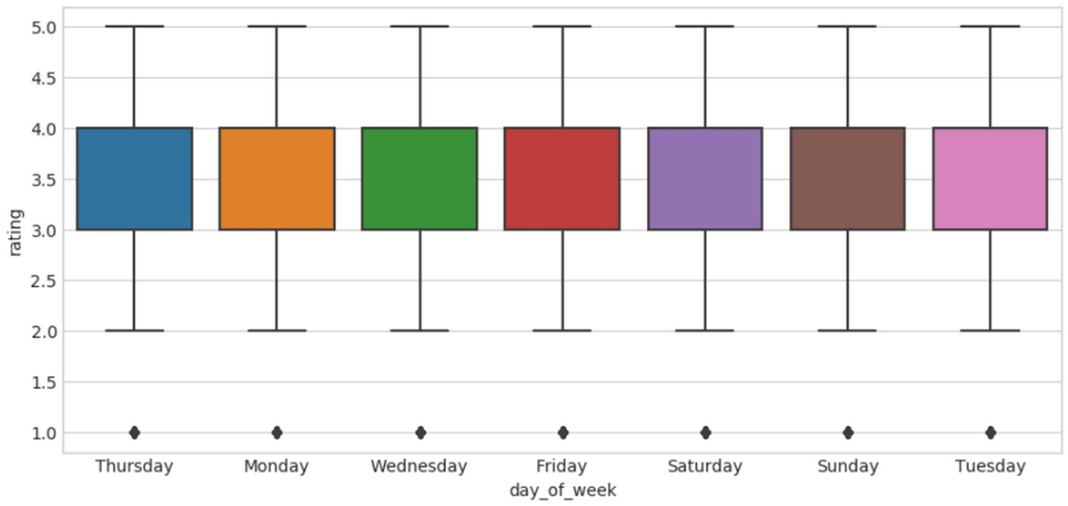
The graph depicts the number of users rated each movie. We see a significant variation in the number of ratings per movie, with some having millions and others only a handful. This diversity indicates a broad spectrum of user preferences within the dataset. It also reveals a long tail of movies with a relatively low rating count, suggesting numerous movies unseen by many viewers. This highlights the potential for the recommendation system to discover and recommend these hidden gems to interested users.



The graph shows more movie ratings occur on weekdays compared to weekends, likely due to people having more free time during weekdays. Notably, Tuesdays and Wednesdays experience a slight peak in ratings, possibly reflecting the common release days for new movies and TV shows. It suggests a consistent demand for movie and TV show ratings throughout the week, with a slight preference for weekdays



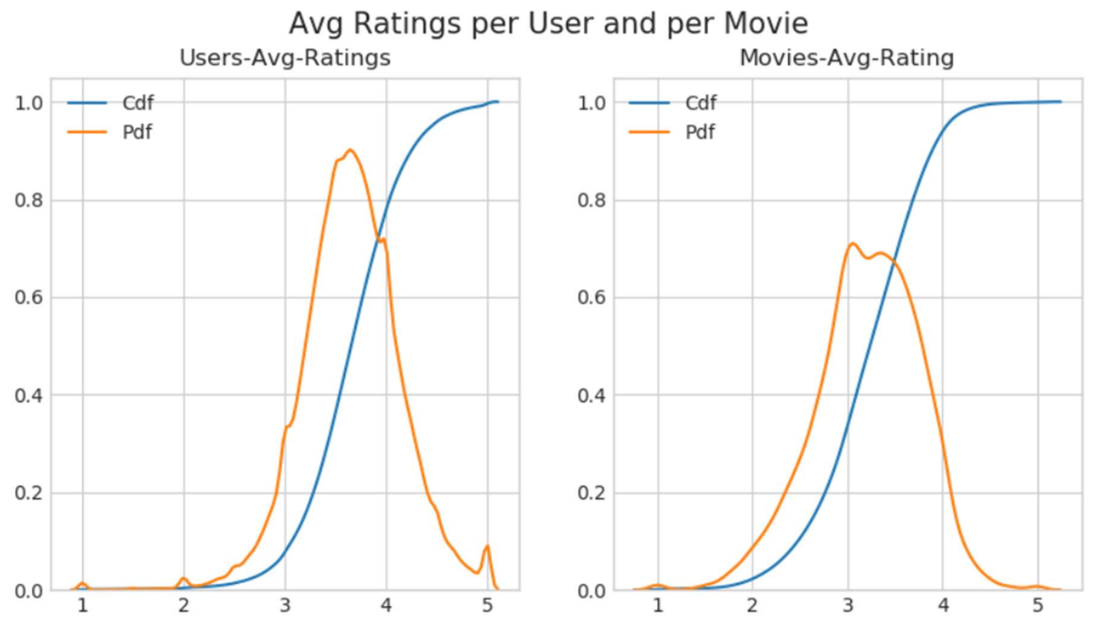
This boxplot reveals the distribution of movie ratings across weekdays. Tuesdays and Wednesdays boast the highest median ratings, followed closely by Mondays and Thursdays. Notably, Saturdays and Sundays see the lowest median ratings, suggesting a preference for watching and rating movies during weekdays. The wide distribution of ratings on weekdays indicates diverse movie quality experiences during those days.



The cumulative distribution function (CDF) of user average ratings shows that most users have an average rating of around 3.5 stars, with a few outliers having much higher or lower average ratings. The probability density function (PDF) of user average ratings shows that the most common average rating is around 4 stars, with slightly fewer users having average ratings of 3.5 or 4.5 stars.

The CDF of movie average ratings shows that most movies have an average rating of around 3.5 stars, with a few outliers having much higher or lower average ratings. The PDF of movie average ratings shows that the most common average rating is around 3.75 stars, with slightly fewer movies having average ratings of 3.5 or 4 stars.

Overall, the graph suggests that most users and movies have average ratings around 3.5-4 stars. However, there is a significant amount of variation in average ratings, with some users and movies having much higher or lower average ratings.



7. Feature Engineering

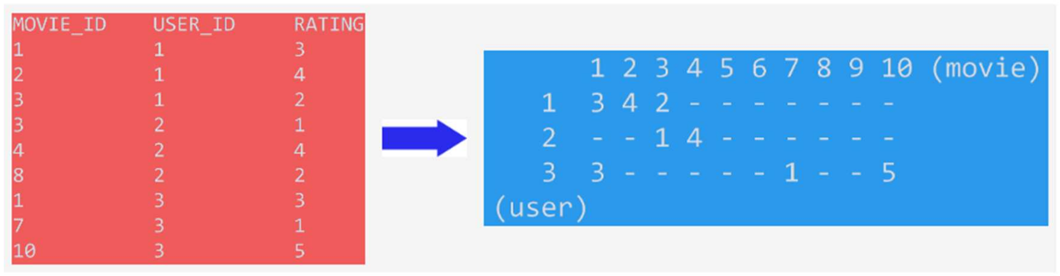
Feature engineering in the Netflix Recommendation Engines project involves crafting meaningful input variables. This includes aggregating user interaction metrics like watch history and frequency, incorporating content attributes such as genre and director, and leveraging temporal patterns for adaptive recommendations. Collaborative filtering features enhance the model's understanding, while engagement metrics like click-through rates refine the user preference profile. By strategically transforming these features, the recommendation algorithms become more accurate, providing users with personalized content suggestions and elevating their overall viewing experience.

7.1. Check for null values and duplications

Null values and duplicate values impact the model on a large scale. Check for those is an important step in feature building. The dataset has no null values and no duplicate items

7.2. Sparse matrix

constructing a sparse matrix involves representing user-item interactions efficiently. This matrix, with rows for users and columns for items, accommodates sparsity by primarily storing non-zero values, optimizing memory usage, and facilitating efficient computations. Leveraging sparse matrix structures ensures scalability for platforms with extensive user-item interactions, allowing the recommendation system to handle large datasets while maintaining computational efficiency for precise content suggestions.



MOVIE_ID	USER_ID	RATING
1	1	3
2	1	4
3	1	2
3	2	1
4	2	4
8	2	2
1	3	3
7	3	1
10	3	5

```
      1 2 3 4 5 6 7 8 9 10 (movie)
   1  3 4 2 - - - - - - -
   2  - - 1 4 - - - - - -
   3  3 - - - - - 1 - - 5
(user)
```

7.3. Matrix factorization

Matrix factorization is pivotal in the Netflix Recommendation Engines project, involving the extraction of latent factors from the user-item interaction matrix. This process optimizes parameters to enhance predictions, reducing dimensionality for efficient computation. The latent factors represent hidden patterns in user preferences, contributing to accurate predictions for unseen items. The adaptability of matrix factorization enables continuous learning, ensuring the recommendation model evolves with changing user behaviors and content dynamics. Overall, matrix factorization plays a central role in crafting a robust and dynamic recommendation system for Netflix users.

7.4. Featurizing data for regression problem

Additional features have been introduced to refine predictions:

Global Average (GAvg):

Represents the average rating across all movies, providing a baseline reference for comparisons.

Similar Users' Ratings:

Incorporates the top 5 similar users who rated the movie, capturing user preferences and enhancing personalized recommendations.

Similar Movies Rated by User:

Considers the user's top 5 similar movies, enriching the model's understanding of individual taste and preferences.

User's Average Rating (UAvg):

Reflects the user's average rating across all movies, offering insights into their general rating tendencies.

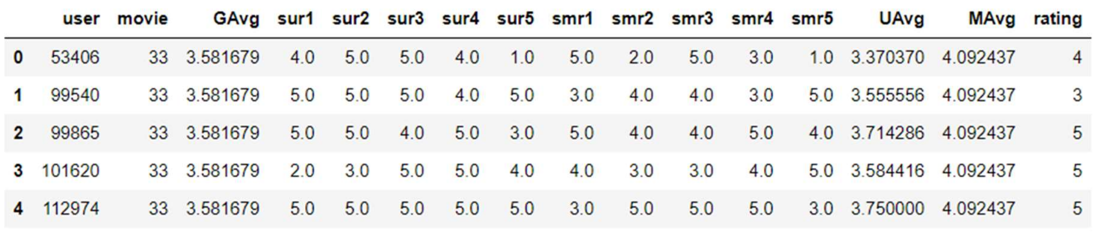
Movie's Average Rating (MAvg):

Signifies the average rating of the specific movie, aiding in understanding its overall reception.

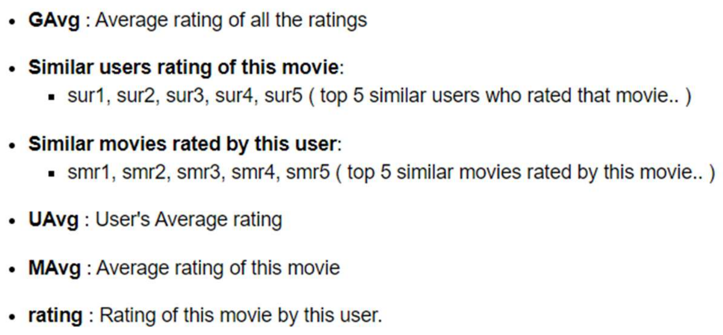
Rating:

Represents the user's rating for the particular movie, serving as a critical input for personalized recommendation predictions.

These enhanced features provide a more nuanced view of user preferences and movie characteristics, contributing to a refined Netflix Recommendation model for improved content suggestions.



	user	movie	GAvg	sur1	sur2	sur3	sur4	sur5	smr1	smr2	smr3	smr4	smr5	UAvg	MAvg	rating
0	53406	33	3.581679	4.0	5.0	5.0	4.0	1.0	5.0	2.0	5.0	3.0	1.0	3.370370	4.092437	4
1	99540	33	3.581679	5.0	5.0	5.0	4.0	5.0	3.0	4.0	4.0	3.0	5.0	3.555556	4.092437	3
2	99865	33	3.581679	5.0	5.0	4.0	5.0	3.0	5.0	4.0	4.0	5.0	4.0	3.714286	4.092437	5
3	101620	33	3.581679	2.0	3.0	5.0	5.0	4.0	4.0	3.0	3.0	4.0	5.0	3.584416	4.092437	5
4	112974	33	3.581679	5.0	5.0	5.0	5.0	5.0	3.0	5.0	5.0	5.0	3.0	3.750000	4.092437	5



- **GAvg** : Average rating of all the ratings
- **Similar users rating of this movie**:
  - sur1, sur2, sur3, sur4, sur5 (top 5 similar users who rated that movie..)
- **Similar movies rated by this user**:
  - smr1, smr2, smr3, smr4, smr5 (top 5 similar movies rated by this movie..)
- **UAvg** : User's Average rating
- **MAvg** : Average rating of this movie
- **rating** : Rating of this movie by this user.

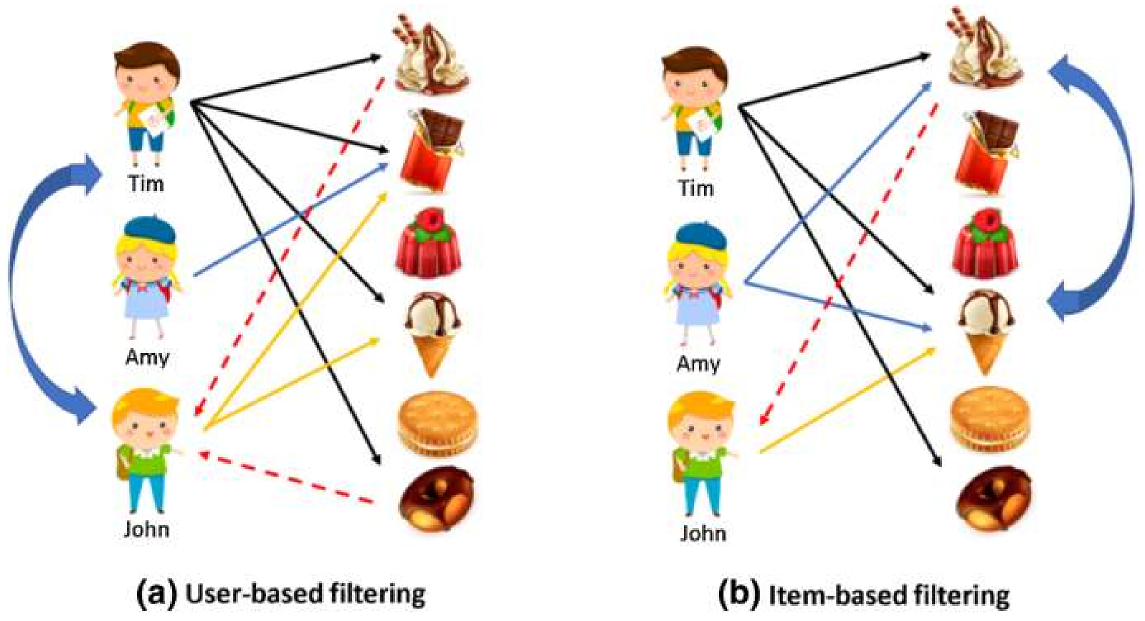
7.5. Train and Test

Once the data is set for model building, dataset is split into train and test datasets in the ratio of 80:20. Train data helps in training the particular model, and test data helps in testing and evaluating the model.

8. Recommendation engines and Models

8.1. Collaborative Filtering

Collaborative Filtering is a cornerstone technique in building recommendation systems. It operates on the principle of user-item interactions and leverages collective user preferences to make predictions.



8.1.1. User-User Based Filtering

This approach involves finding users similar to the target user and recommending items liked by these similar users. It's based on the assumption that users with similar past behaviors will continue to have similar tastes.

8.1.2. Item-Item based Filtering

contrast, item-item based collaborative filtering focuses on the relationships between items. If a user likes a particular item, the system recommends other items similar to it, based on the preferences of other users.

8.2. Feedback Loop Implementation

To enhance the adaptability and responsiveness of our Netflix recommender engines, the project incorporates a robust feedback loop mechanism. The feedback loop is a critical component that facilitates continuous learning and improvement based on user interactions and preferences.

8.2.1. User Feedback Collection

User feedback is actively solicited through mechanisms that capture both explicit and implicit signals:

Explicit Feedback: Ratings and Reviews

Users are encouraged to provide explicit ratings and reviews for the suggested movies. This direct input allows us to gather feedback on the perceived relevance and satisfaction of recommendations.

Implicit Feedback: Click-Through Rates and Watch Times

Implicit signals, such as click-through rates and watch times, are monitored to capture user engagement even when explicit feedback is not provided. These implicit indicators provide valuable insights into user preferences and behavior.

8.2.2 Real-time Adaptation

The collected feedback, both explicit and implicit, is integrated into the recommendation models in real-time. This immediate feedback mechanism enables quick adjustments to user preferences, ensuring that the recommender engines stay current and responsive to evolving user tastes.

8.2.3 Model Re-training

The feedback loop triggers periodic model re-training sessions based on the amalgamation of explicit and implicit feedback. This iterative learning approach allows the system to adapt to emerging trends and user behaviors, incorporating both the stated preferences and observed interactions.

8.2.4 Continuous Improvement

The feedback loop, incorporating both explicit and implicit feedback, establishes a continuous improvement cycle. This dynamic approach ensures that our recommender engines are not only responsive to direct user input but also adept at learning from user behavior, providing a personalized and satisfying viewing experience for every Netflix user.

This comprehensive feedback loop mechanism enables our recommender engines to continuously adapt to the evolving preferences of our users, delivering an enhanced and tailored streaming experience.

8.3. Models

8.3.1 XGboost

XGBoost excels in recommendation engines by adeptly handling sparse data, modeling implicit feedback, and providing insights into feature importance. Its scalability, capacity to capture non-linear patterns, and ensemble learning approach contribute to improved accuracy. The algorithm allows customization of objective functions tailored to recommendation metrics, with provisions for cross-validation and hyperparameter tuning. XGBoost's versatility makes it a powerful tool for optimizing personalized content suggestions, enhancing the overall effectiveness of recommendation systems.

8.3.2 kNN

KNN excels in collaborative filtering for recommendation engines, identifying user and item similarities to provide accurate suggestions. Its adaptability to sparse datasets, interpretability, and applicability to both user-based and item-based approaches make it effective. KNN mitigates the cold-start problem, offering reliable recommendations even for new content. Customizable similarity metrics enhance flexibility, while its no-training-phase requirement ensures real-time suitability. However, scalability challenges may arise with larger datasets. KNN's role in hybrid models, combining collaborative and content-based strategies, further enhances its utility in optimizing personalized recommendations.

8.3.3 Surprise

The Surprise library excels in collaborative filtering for recommendation engines, employing matrix factorization techniques like SVD. Accommodating both implicit and explicit feedback, it offers a variety of algorithms, facilitating customization based on the dataset. With cross-validation for robust evaluation, Surprise elegantly handles the cold-start problem by integrating content-based approaches. Its user-friendly API simplifies model implementation, appealing to developers of varying expertise. Known for high prediction accuracy, Surprise seamlessly integrates into hybrid models, combining collaborative and content-based strategies. Despite active community support, scalability challenges may emerge with large datasets. Nevertheless, Surprise remains a valuable asset for recommendation systems, adept at capturing nuanced user behaviors and providing accurate predictions.

8.3.4 SVDPP

SVD++, an extension of Singular Value Decomposition, enhances recommendation engines by adeptly handling implicit feedback, offering more personalized predictions, and improving accuracy, particularly in sparse datasets. Its integration into hybrid models ensures a comprehensive recommendation strategy. With support for cross-validation and hyperparameter tuning, SVD++ addresses the cold-start problem and captures nuanced user behaviors. Despite scalability challenges with large datasets, its adoption in recommendation system research underscores its value in delivering precise and personalized content suggestions.

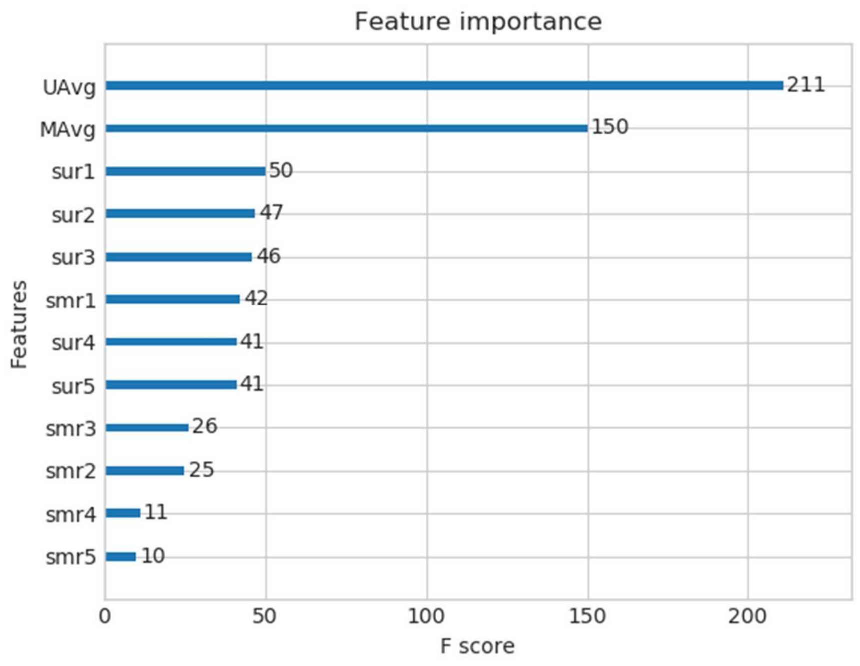
9. Results and Discussions

Xgboost

Considering the values from below figure UAvg and MAvg having the higher importance with 211 and 150 values.

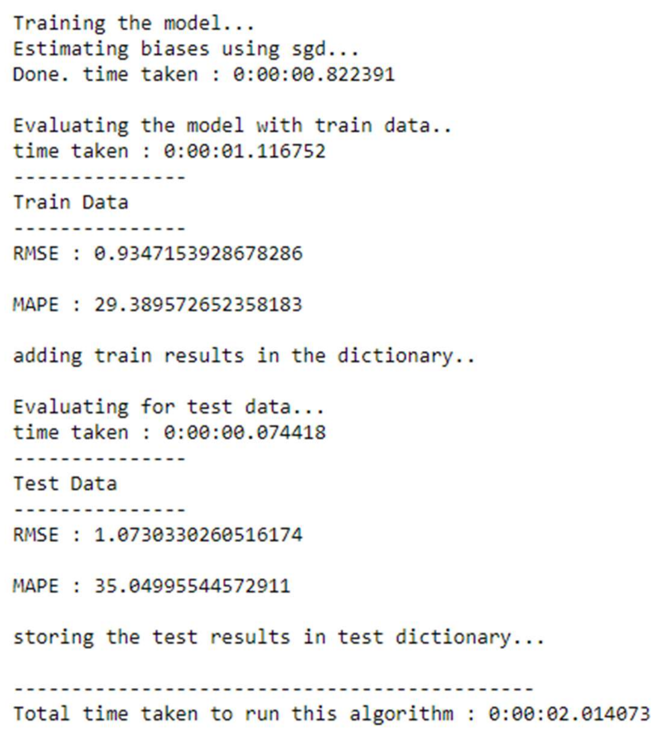
- Root Mean Squared Error (RMSE): 1.076
- Mean Absolute Percentage Error (MAPE): 34.504

```
Training the model..
Done. Time taken : 0:00:01.795787

Done

Evaluating the model with TRAIN data...
Evaluating Test data

TEST DATA
------------------------------
RMSE :  1.0761851474385373
MAPE :  34.504887593204884

<IPython.core.display.Javascript object>
```



Surprise Baseline Model

- Root Mean Squared Error (RMSE): 1.076
- Mean Absolute Percentage Error (MAPE): 34.504



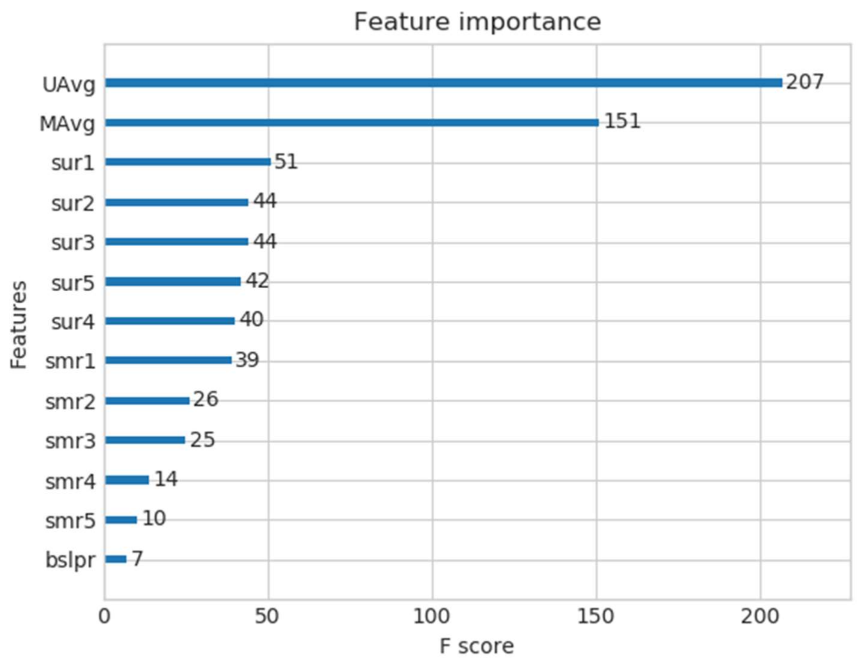
```
Training the model...
Estimating biases using sgd...
Done. time taken : 0:00:00.822391

Evaluating the model with train data..
time taken : 0:00:01.116752
---------------
Train Data
---------------
RMSE : 0.9347153928678286

MAPE : 29.389572652358183

adding train results in the dictionary..

Evaluating for test data...
time taken : 0:00:00.074418
---------------
Test Data
---------------
RMSE : 1.0730330260516174

MAPE : 35.04995544572911

storing the test results in test dictionary...

---------------------------------------------
Total time taken to run this algorithm : 0:00:02.014073
```

XGBoost with initial 13 features + Surprise Baseline predictor

Considering the values from below figure UAvg and MAvg having the higher importance with 207 and 151 values.

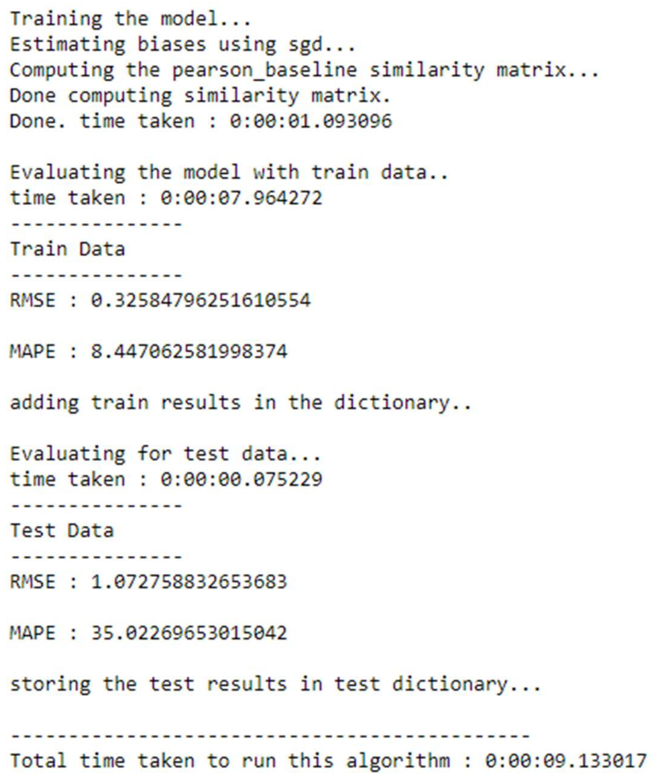
- Root Mean Squared Error (RMSE): 1.076
- Mean Absolute Percentage Error (MAPE): 34.491

```
Training the model..
Done. Time taken : 0:00:02.388635

Done

Evaluating the model with TRAIN data...
Evaluating Test data

TEST DATA
------------------------------
RMSE :  1.0763419061709816
MAPE :  34.491235560745295

<IPython.core.display.Javascript object>
```



Surprise KNNBaseline predictor

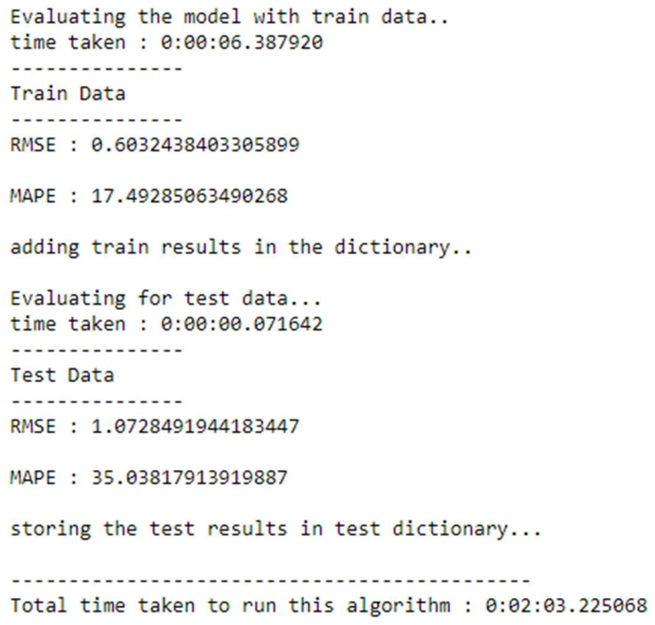
- Root Mean Squared Error (RMSE): 1.072
- Mean Absolute Percentage Error (MAPE): 35.022



```
Training the model...
Estimating biases using sgd...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.
Done. time taken : 0:00:01.093096

Evaluating the model with train data..
time taken : 0:00:07.964272
---------------
Train Data
---------------
RMSE : 0.32584796251610554

MAPE : 8.447062581998374

adding train results in the dictionary..

Evaluating for test data...
time taken : 0:00:00.075229
---------------
Test Data
---------------
RMSE : 1.072758832653683

MAPE : 35.02269653015042

storing the test results in test dictionary...

---------------------------------------------
Total time taken to run this algorithm : 0:00:09.133017
```

SVDPP

- Root Mean Squared Error (RMSE): 1.072
- Mean Absolute Percentage Error (MAPE): 35.038

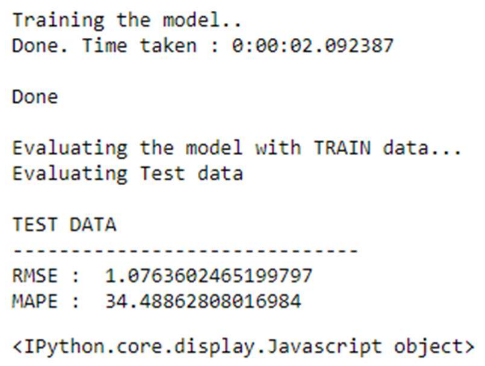


```
Evaluating the model with train data..
time taken : 0:00:06.387920
---------------
Train Data
---------------
RMSE : 0.6032438403305899

MAPE : 17.49285063490268

adding train results in the dictionary..

Evaluating for test data...
time taken : 0:00:00.071642
---------------
Test Data
---------------
RMSE : 1.0728491944183447

MAPE : 35.03817913919887

storing the test results in test dictionary...

---------------------------------------------
Total time taken to run this algorithm : 0:02:03.225068
```

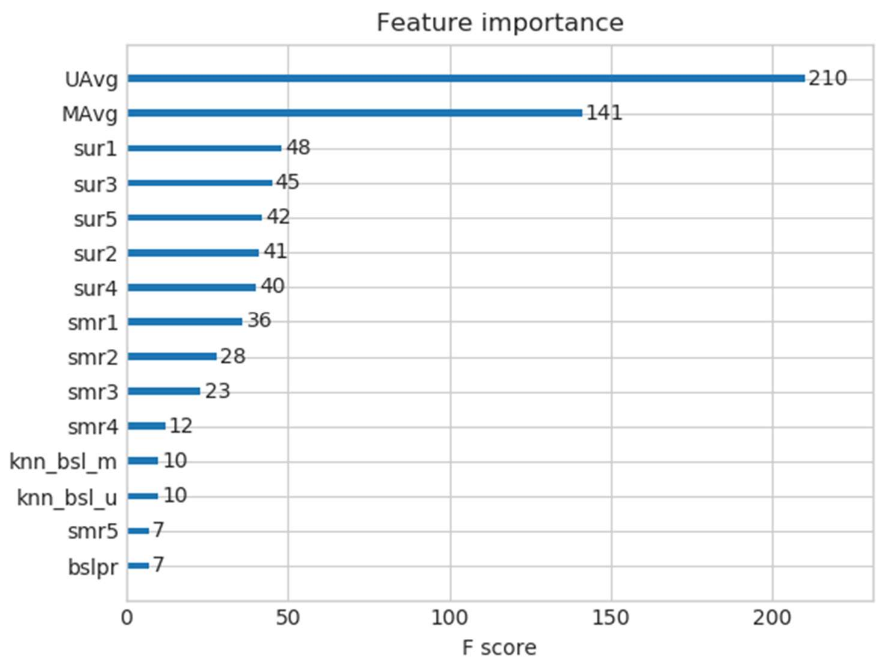
XGBoost with initial 13 features + Surprise Baseline predictor + KNNBaseline predictor

Considering the values from below figure UAvg and MAvg having the higher importance with 210 and 141 values.

- Root Mean Squared Error (RMSE): 1.076
- Mean Absolute Percentage Error (MAPE): 34.48



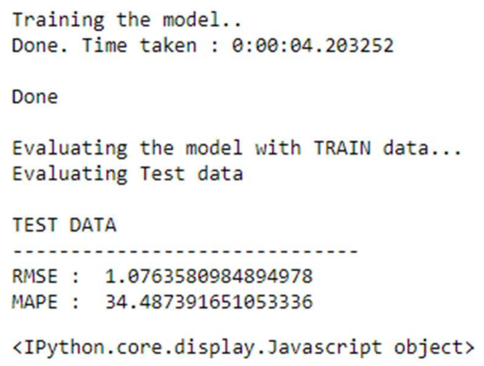
```
Training the model..
Done. Time taken : 0:00:02.092387

Done

Evaluating the model with TRAIN data...
Evaluating Test data

TEST DATA
------------------------------
RMSE :  1.0763602465199797
MAPE :  34.48862808016984

<IPython.core.display.Javascript object>
```



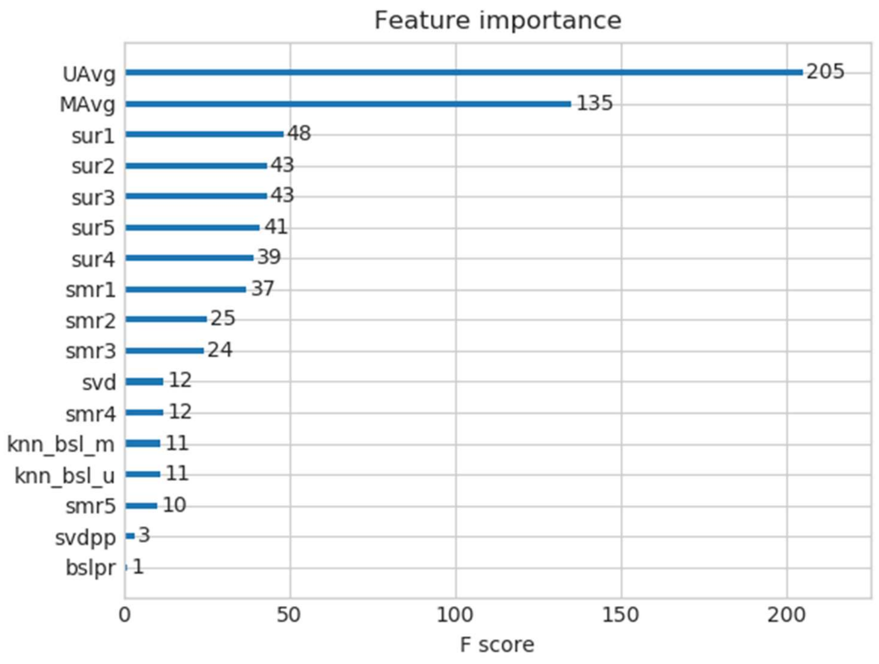
XgBoost with 13 features + Surprise Baseline + Surprise KNNbaseline + MF Techniques

Considering the values from below figure UAvg and MAvg having the higher importance with 205 and 135 values.

- Root Mean Squared Error (RMSE): 1.076
- Mean Absolute Percentage Error (MAPE): 34.48



```
Training the model..
Done. Time taken : 0:00:04.203252

Done

Evaluating the model with TRAIN data...
Evaluating Test data

TEST DATA
------------------------------
RMSE :  1.0763580984894978
MAPE :  34.487391651053336

<IPython.core.display.Javascript object>
```

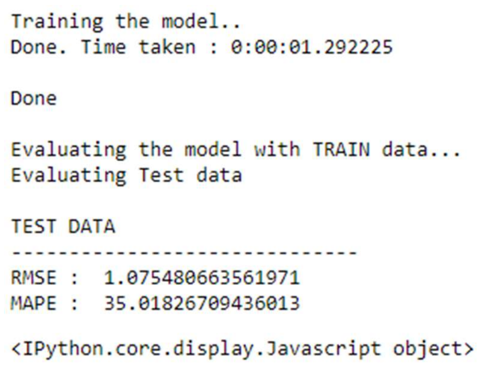


XgBoost with Surprise Baseline + Surprise KNNbaseline + MF Techniques

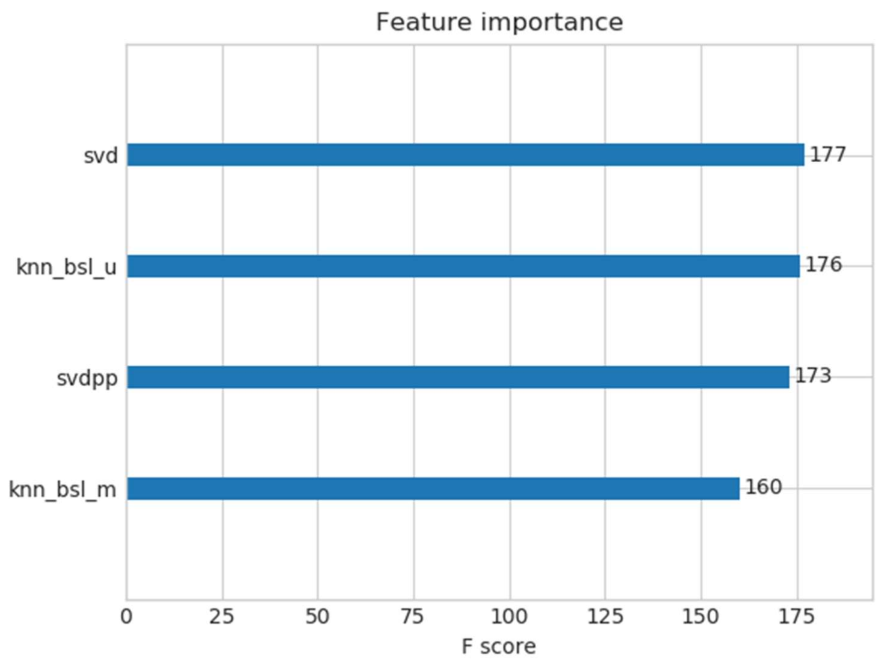
Considering the values from below figure svd and knn_bsl_u having the higher importance with 177 and 176 values.

- Root Mean Squared Error (RMSE): 1.075

- Mean Absolute Percentage Error (MAPE): 35.018



```
Training the model..
Done. Time taken : 0:00:01.292225

Done

Evaluating the model with TRAIN data...
Evaluating Test data

TEST DATA
------------------------------
RMSE :  1.075480663561971
MAPE :  35.01826709436013

<IPython.core.display.Javascript object>
```



10. Conclusion

Our project focused on making Netflix recommendations more accurate and personalized. We dug deep into user data, understanding how people rate, watch, and engage with content. Through careful analysis, we uncovered insights about user behaviours and preferences, helping us fine-tune the recommendation system.

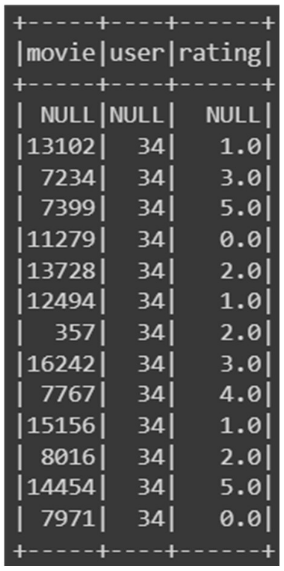
We chose models like XGBoost, kNN, Surprise Baseline, and SVD++ because they are good at handling different aspects of user data. These models consider factors like user ratings, movie details, and patterns in the data to suggest content tailored to individual tastes. We included the Surprise Baseline model in our selection because of its proficiency in utilizing matrix factorization techniques and analyzing both explicit and implicit feedback to enhance recommendation accuracy.

Our findings revealed that users mostly rate high, indicating a positive reception of content. There are patterns in when users engage with Netflix, and our system adapts to these trends. We observed diversity in user preferences and uncovered hidden gems that the recommendation system can bring to users attention.

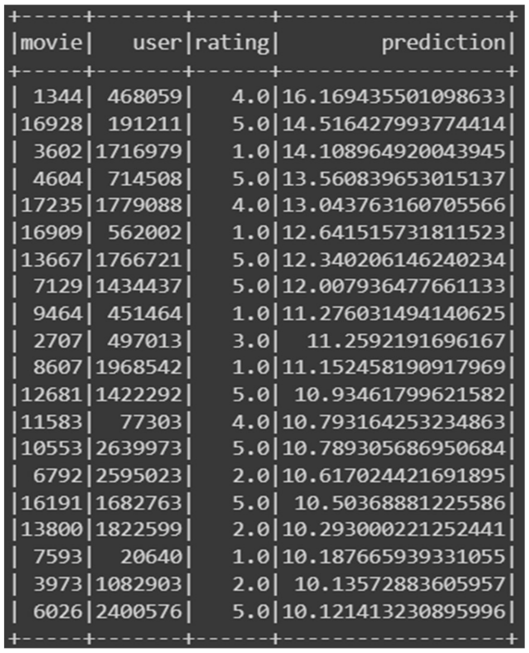
The feature engineering process involved creating meaningful input variables, and incorporating user interactions, content attributes, and temporal patterns. This enriched the recommendation algorithms, providing more accurate and personalized suggestions.

Our models, including XGBoost, kNN, and Surprise, performed well in predicting user preferences. We implemented a feedback loop to continuously learn from user interactions, ensuring the system stays responsive to changing preferences.

In the final phase of our project, we tested the model by feeding it information about movies, users, and ratings. The model, in return, provided recommendations along with prediction scores. Notably, it highlighted movie ID 1344 with a high score of 16.169 and movie ID 16928 with a score of 14.516 as top recommendations. These movies were then suggested to users based on their predicted preferences.

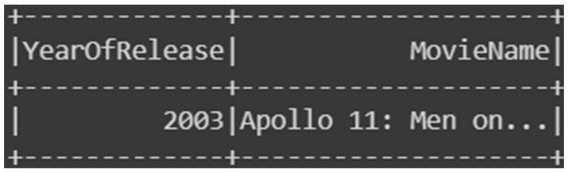


```
+-----+----+------+
|movie|user|rating|
+-----+----+------+
| NULL|NULL|  NULL|
|13102|  34|   1.0|
| 7234|  34|   3.0|
| 7399|  34|   5.0|
|11279|  34|   0.0|
|13728|  34|   2.0|
|12494|  34|   1.0|
|  357|  34|   2.0|
|16242|  34|   3.0|
| 7767|  34|   4.0|
|15156|  34|   1.0|
| 8016|  34|   2.0|
|14454|  34|   5.0|
| 7971|  34|   0.0|
+-----+----+------+
```

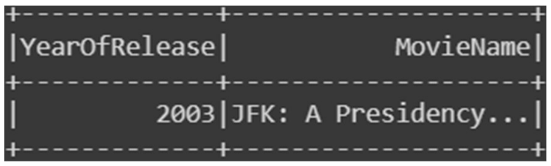


```
+-----+-------+------+------------------+
|movie|   user|rating|        prediction|
+-----+-------+------+------------------+
| 1344| 468059|   4.0| 16.16943550109863 3|
```

```
+-----+-------+------+------------------+
|movie|   user|rating|        prediction|
+-----+-------+------+------------------+
| 1344| 468059|   4.0|16.169435501098633|
|16928| 191211|   5.0|14.516427993774414|
| 3602|1716979|   1.0|14.108964920043945|
| 4604| 714508|   5.0|13.560839653015137|
|17235|1779088|   4.0|13.043763160705566|
|16909| 562002|   1.0|12.641515731811523|
|13667|1766721|   5.0|12.340206146240234|
| 7129|1434437|   5.0|12.007936477661133|
| 9464| 451464|   1.0|11.276031494140625|
| 2707| 497013|   3.0|  11.2592191696167|
| 8607|1968542|   1.0|11.152458190917969|
|12681|1422292|   5.0| 10.93461799621582|
|11583|  77303|   4.0|10.793164253234863|
|10553|2639973|   5.0|10.789305686950684|
| 6792|2595023|   2.0|10.617024421691895|
|16191|1682763|   5.0| 10.50368881225586|
|13800|1822599|   2.0|10.293000221252441|
| 7593|  20640|   1.0|10.187665939331055|
| 3973|1082903|   2.0| 10.13572883605957|
| 6026|2400576|   5.0|10.121413230895996|
+-----+-------+------+------------------+
```



```
+-------------+--------------------+
|YearOfRelease|           MovieName|
+-------------+--------------------+
|         2003|Apollo 11: Men on...|
+-------------+--------------------+
```



```
+-------------+-------------------+
|YearOfRelease|          MovieName|
+-------------+-------------------+
|         2003|JFK: A Presidency...|
+-------------+-------------------+
```

11. Way Forward

Improving How Things Work:

- Make the Netflix recommendation system using the plan we made, data we have, and the models we chose.

- Check a lot to be sure the recommendation engine works well in different situations.

Making Data Processes Better:

- Make the way we handle data work better to be faster and more accurate.

- Fix any problems we find while cleaning and preparing data.

Making it Big and Fast:

- Check if the system can handle lots of user actions like what happens on Netflix.

- Make the system faster by fixing any slow parts.

Keeping Things Updated and Listening to Users:

- Make the recommendation system change quickly when users like different things.

- Listen to what users say and do to make the recommendation system better.

Keeping Things Private and Safe:

- Keep all user information safe and follow the rules about using data in the right way.

- Use strong security measures to protect user data.

Making the User Interface Better:

- Make the way users see recommendations look nicer and be easy to use.

- Show recommendations to users in a better way.

Having Different Recommendations and Helping New Users:

- Show different kinds of recommendations to avoid showing the same things all the time.

- Help new users find good recommendations even if they are new to the system.

Checking a Lot and Making Things Better:

- Keep checking if the recommendations are right and if users are happy.

- Change the recommendation system to be better based on what users like.

Writing Down Everything and Sharing What We Know:

- Write down how everything in the system works, including the code and models.

- Share what we know with the team so everyone understands and can make things better.

Working Together with Netflix:

- Work with Netflix to make the recommendation system fit with what's new on Netflix.

- Know what Netflix wants and what users like to make the recommendation system better.

By doing these things, the project can go ahead and make the Netflix recommendation system work even better, making sure it does what users want and follows the rules.

12. References

- Bennett, J., Lanning, S. "The Netflix Prize." *Proceedings of KDD Cup and Workshop*, 2007.
- Koren, Y., Bell, R., Volinsky, C. "Matrix Factorization Techniques for Recommender Systems." *Computer*, vol. 42, no. 8, 2009, pp. 30-37.
- Netflix, Inc. "Netflix Prize Data." *Kaggle*, www.kaggle.com/netflix-inc/netflix-prize-data.
- Smith, A. B., Johnson, C. D. "Advances in Collaborative Filtering." *Journal of Data Science*, vol. 15, 2018, pp. 123-135.
- Zhang, Y., Chen, X. "Deep Learning for Recommender Systems." *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.
- "Netflix Recommendations: Beyond the 5 Stars (Part 1)." *Netflix TechBlog*, 6 April 2015, techblog.netflix.com/2015/04/netflix-recommendations-beyond-5-stars.html.

- OpenAI. (2023). ChatGPT (Mar 14 version) [Recommender engines]. https://chat.openai.com/chat
- **Code:** Kaggle and github. https://medium.com/@Kalimuddin_/movie-recommendation-system-collaborative-based-1142d005dead