A FIELD PROJECT REPORT

on

User Behavior Analysis in Netflix Streaming Service

Submitted

by

221FA04434	221FA04554
CH. Meghana	k. Divya
221FA04616	221FA04679
B. Chenna Kesava	D. Devi

Under the guidance of

Sajida Sultana. Sk

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH

Deemed to be UNIVERSITY

Vadlamudi, Guntur.

ANDHRA PRADESH, INDIA, PIN-522213.

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the Field Project entitled **"User Behavior Analysis in Netflix Streaming Service"** is being submitted by 221FA04434(CH. Meghana), 221FA04554 (K. Divya),221FA04616(B. Chenna Kesava), 221FA04679(Devi Dasari) for partial fulfillment of Field Project is a bonafide work carried out under the supervision of Ms. Sk. Sajida Sultana, Assistant Professor, Department of CSE.

[illegible] [illegible]

Dr. S. V. Phani Kumar

Dr. K.V. Krishna Kishore

Ms. Sajida Sultana. Sk
Assistant Professor, CSE

HOD, CSE

Dean, SoCI

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that the Field Project entitled **"User Behavior Analysis in Netflix Streaming Service"** is being submitted by 221FA04434 (CH. Meghana), 221FA04554 (K. Divya), and 221FA04616 (B. Chenna Kesava),221FA04679 (Devi Dasari) in partial Fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of Ms. Sk. Sajida Sultana, Assistant Professor, Department of CSE.

By

221FA04434 (CH. Meghana),
221FA04554 (K. Divya),
221FA04616(B. Chenna Kesava),
221FA04679 (Devi Dasari)

Date:

ABSTRACT

Netflix Recommendation System It is such an intricate algorithmic configuration, used in a way such that it supports the personalization of experience for users, tailoring recommendations according to specific tastes and propensities. In a nutshell, the system in its core utilizes the method of machine learning and collaborative filtering in analyzing the patterns of the behavioral habits and their thousands of users' preferences. The model makes use of a two-pronged hybrid approach that combines two primary sources, that is, content-based filtering, focusing on the attributes of shows and collaborative filtering. And in this regard, movies make use of collaborative filtering to determine similarity among the users and their view history. This will help the service to further suggest content appropriate for its users and keeps them locked up inside, therefore taking lesser time in finding new material. This system has been further developed with deep learning models and natural language processing, which bring forth meaningful insights from metadata, user reviews, and even video content. Through the use of user interactions and aggregation of data from various sources, the Netflix system is dynamic and adaptable to change in preferences. Personalization at such a level played a very vital role in setting up customer retention and engagement through which Netflix became a house-hold name in the streaming industry.

Key Words: Collaborative filtering, Content-based filtering, Personalization, Natural language processing (NLP), User preferences, Viewing patterns, Recommendation diversity.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

1.INTRODUCTION

1.1 What is a recommendation system, and how does it work?

A recommendation system is an advanced algorithm used by platforms like Netflix to suggest content based on user preferences, viewing history, and behavior. Essential for enhancing user experience, it minimizes content discovery time and maximizes engagement by increasing viewing time and retention. Netflix's hybrid recommendation model combines **collaborative filtering** (finding patterns among user preferences) and **content-based filtering** (using attributes of shows and movies). Additionally, Netflix employs machine learning and deep learning to process extensive metadata and user interactions, allowing real-time updates that adapt to individual tastes, continually refining recommendations.

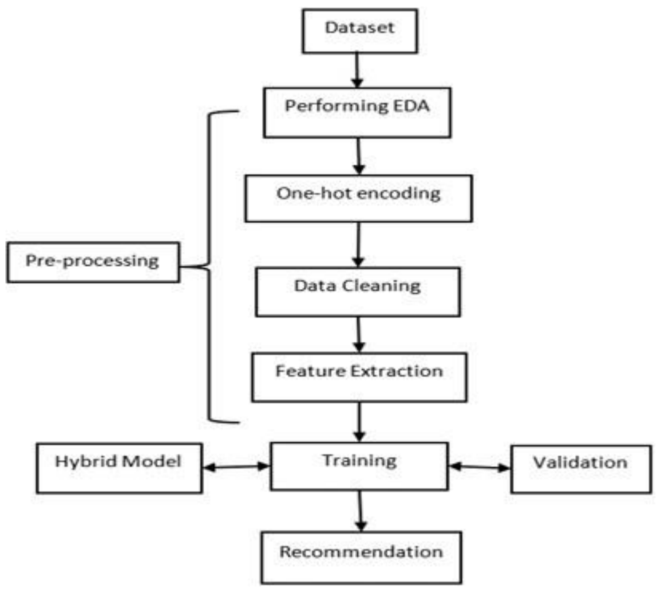


Figure 1.1 Flow Diagram for Netflix Recommendation System

This flowchart represents the general steps in building a recommendation system. Here's a breakdown of each stage:
Dataset: The process begins with collecting or accessing a dataset containing user interactions, item information, or user demographics.
Performing EDA (Exploratory Data Analysis): Analysing the data to understand its structure, patterns, and initial insights, which helps guide preprocessing and model selection.
Pre-processing: This includes several steps to prepare the data for training:
One-hot Encoding: Converting categorical variables into a numerical format, making them suitable for machine learning models.
Data Cleaning: Removing or handling missing values, outliers, and inconsistencies in the dataset.
Feature Extraction: Selecting or transforming features to be used by the recommendation model, such as user preferences or item characteristics.
Training: The model is trained on the pre-processed data to learn patterns and relationships within the dataset.

Validation: The model's performance is tested on a validation dataset to ensure accuracy and effectiveness before deployment.
Hybrid Model: This step represents a specific type of recommendation model that combines multiple techniques (e.g., collaborative and content-based filtering) for improved recommendations.
Recommendation: After training and validation, the model provides personalized recommendations to users based on learned patterns from the data.

1.2 The importance of recommendation system in streaming platform

The recommendation system is crucial to the success of streaming platforms like Netflix, driving user experience and business growth. With millions of titles available, it helps users discover content they're likely to enjoy, reducing the overwhelming task of browsing through vast libraries. By personalizing suggestions, Netflix increases the likelihood of longer viewing sessions, improving customer satisfaction. From a business perspective, the recommendation engine directly impacts revenue by retaining customers; relevant content keeps subscribers engaged, reducing cancellations and promoting long-term growth. Studies show that over 80% of content watched on Netflix is driven by recommendations, underscoring its influence on user behaviour. Moreover, Netflix's recommendation system informs strategic decisions on content purchases and original productions by identifying viewership trends. This data-driven approach allows Netflix to invest in content that resonates with audiences, helping the platform stay competitive. Additionally, an effective recommendation system aids in server load balancing through demand prediction, ensuring smooth operation even during high traffic.

1.3 The economic and operational impact of Netflix recommendation system

Netflix's recommendation system strongly influences the economics and operations in terms of the power it gives for driving usage, increasing viewing time, and lowering churn–all with a direct revenue impact. In keeping users happy, it delivers personal content; reduces marketing and discovery costs but builds retention. It does so operationally by optimizing content usage by popularizing and niche titles maximized on the return on Netflix's content investments. The system also lessens pressure on Infrastructure can be forecasted by the patterns, which enables Netflix to optimize content delivery and resources for streaming effectively. The personalized nature has been at the heart of the Netflix competitive advantage in the streaming business.

1.4 Current methodologies used in recommendation system

Current recommendation systems rely on a variety of methodologies, including collaborative filtering (user-based and item-based), where recommendations are made Content-based filtering: Provide items with similar features to what one has liked. Hybrid models: Implement these techniques together for higher accuracy. Advanced methods: Deep learning captures complex user-item interactions. Matrix factorization, factorization machines: Address sparse data issues. Reinforcement learning and contextual bandits: Adapt real-time recommendations between exploration and exploitation. Graph-based approaches along with session-based recommendations are used in the recent days to enhance personalization in recommendations.

1.5 Applications of machine learning in improving Netflix recommendation system

Machine learning improves Netflix's recommendation system both by increasing personalization and by improving content discovery. Algorithms based on collaborative filtering consider user behavior in term of recommending content that similar users would like, and deep learning models apply complex viewing patterns and preferences. Applying NLP enables the analysis of metadata such as plot descriptions. This allows for content-based recommendations to be even better. Reinforcement learning optimizes real-time recommendations based on the proportion of user satisfaction and the exploration of new content. In addition to this, machine learning solves the cold start problem for newcomers to the system as well as the content, providing contextual information along with predictive modeling, improving recommendations to a tailored experience continually.

CHAPTER-2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Literature review

In recent years, the Netflix recommendation system has become a leading example of how machine learning can revolutionize user experiences in the streaming industry. By analyzing extensive datasets of user viewing habits and preferences, Netflix provides highly personalized content suggestions tailored to individual tastes. The system employs a blend of collaborative filtering, content-based filtering, and hybrid models to enhance recommendation accuracy. Recent advancements, including deep learning techniques, allow for the identification of complex patterns in user behavior, further improving recommendation effectiveness. As the streaming landscape continues to evolve, it is essential to refine these algorithms to reduce churn and boost viewer engagement. Collaboration among data scientists, machine learning engineers, and content creators is critical for developing scalable and adaptive systems**.**

Jhansi Hawa Arsytania, Erwin Budi Setiawan, and Isman Kurniawan [1] proposed a movie recommendation system using a hybrid approach combining Collaborative Filtering (CF) and Content-Based Filtering (CBF) enhanced by Convolutional Neural Network (CNN). Their "Cascade Hybrid Filtering" method first applies CF to identify initial movie recommendations based on user-item interactions, followed by CBF to refine the list by assessing content similarity. The CNN model is employed for feature extraction, particularly for processing textual movie reviews, improving accuracy in personalizing recommendations. Evaluated with MAE and RMSE metrics, the approach yielded an RMSE of 0.6325 and an accuracy improvement of 6% over baseline models. Future research will focus on optimizing CNN's feature extraction capabilities for more nuanced recommendations.

Muhammad Tsaqif Muhadzdzib Ramadhan and Erwin Budi Setiawan [2] introduced a Netflix recommendation system using a blend of Collaborative Filtering and K-means clustering. Collaborative Filtering, though popular, often suffers from computational inefficiency in large datasets, so the authors incorporated K-means clustering to group users with similar preferences before applying CF. Their system leveraged Twitter data as a primary source for user ratings, which were converted using **Text Blob** polarity scores. This combined model achieved a lower RMSE (0.6354) than standalone CF methods, highlighting the effectiveness of clustering in reducing computational load and improving recommendation precision. Future work includes exploring other clustering techniques to enhance real-time adaptability.

Koppadi Bhavani and Kottu Aslesha Lakshmi Sai [3] addressed recommendation challenges on Netflix, particularly the cold-start issue, by employing a hybrid recommendation model combining Collaborative Filtering and Content-Based Filtering. Their system uses machine learning algorithms to analyze user behavior and content attributes, effectively balancing item-based and user-based recommendations. This hybrid model also incorporates matrix factorization to address data sparsity issues. Tests indicated a robust recommendation system, with an RMSE under 1.0, and highlighted the potential of matrix factorization in handling sparse datasets. The authors aim to refine their model's real-time adaptability by integrating additional data sources and improving algorithm scalability for broader applicability on streaming platforms.

Pajkovic [4] critically examines the Netflix Recommender System (NRS) with a focus on its role in shaping user preferences through algorithmic taste-making. Employing a reverse-engineering approach, Pajkovic reveals how the NRS leverages both content-based and collaborative filtering methods, grouping users into "taste communities" that span across regions and reflect similar viewing preferences. The study emphasizes the cultural and economic implications of these algorithms, as the NRS not only shapes user taste but also drives engagement by presenting hyper-personalized content. Future research may explore further societal impacts of such algorithmic personalization, particularly regarding user agency in content choice**.**

Sharma, Aggarwal, and Saxena [5] investigate Netflix's content-based recommendation system using text-based data features such as director, genre, and description. Their model employs TF-IDF vectorization to handle content metadata, effectively refining recommendations to align closely with user interests. This study highlights the importance of data visualization and personalization through text analysis in enhancing user engagement on the platform. The authors propose further development of the model to optimize adaptability across different user profiles and devices, demonstrating potential applications for comprehensive data-driven personalization across OTT platforms.

Airen and Agrawal [6] propose an innovative approach to Netflix's recommendation system, addressing data sparsity through a co-clustering technique. Their approach integrates collaborative filtering with co-clustering, which groups similar users and movies to enable more accurate preference predictions, particularly in sparse datasets. This method demonstrated a 7.91% improvement in accuracy over traditional collaborative filtering, underscoring the efficacy of neighborhood-based algorithms in overcoming the cold-start problem. Future iterations of this model may incorporate additional clustering methods to further enhance prediction accuracy and scalability.

Agrawal et al. [7] present a meta-learning-based generative approach to enhance recommendation accuracy in multi-modal systems, specifically addressing the challenge of missing data modalities in platforms like Netflix. By integrating a Graph Attention Network (GAT) with a novel trident architecture, the authors developed a system that reconstructs absent features using available data, thereby sustaining recommendation efficacy even when data is incomplete. This method achieved notable improvements, with experiments on the **Movie Lens** dataset demonstrating a substantial reduction in RMSE compared to traditional models, highlighting the effectiveness of meta-learning in handling real-world multi-modal data variability. Future extensions of this model may explore additional generative approaches to bolster its robustness across varied recommendation environments**.**

Behera and Nain [8] tackle the dynamic nature of user preferences in collaborative filtering systems by incorporating temporal features into matrix factorization (MF) models. Unlike static MF approaches, their model adjusts for shifts in user preferences and item popularity over time, which enhances predictive accuracy. Using the **Movie Lens** datasets, the temporal MF model demonstrated a 1.35% and 1.28% improvement over state-of-the-art models on ML-100K and ML-1M datasets, respectively. These results underscore the impact of temporal features on mitigating data sparsity and the cold-start problem, suggesting that future versions could include additional temporal dynamics to further improve recommendation quality in evolving user environments**.**

Nair et al. [9] developed a cross-platform movie recommendation system utilizing big data analytics and MapReduce programming. Their research addresses the limitations of traditional recommendation systems by integrating data from multiple OTT platforms, specifically Netflix and Amazon Prime. By employing Cosine Similarity for recommendations, they enhance the relevance of movie suggestions based on user preferences. The study emphasizes the importance of platform-specific recommendations, allowing users to filter content according to their subscription availability.

Ang and Wang [10] review the growth of personalized learning algorithms, emphasizing the shift towards using recommender systems and data mining techniques to improve student engagement and learning outcomes. They highlight collaborative filtering, content-based recommendations, and hybrid approaches as core methods, noting the efficacy of hybrid models in addressing sparsity and cold-start issues in educational settings. Experimental results indicate these personalized methods lead to improved learning outcomes, suggesting future work could benefit from integrating reinforcement learning for dynamic adaptability.

Steck et al. [11] examine the application of deep learning in Netflix's recommender systems, detailing challenges like offline-online metric misalignment and data sparsity. By enhancing data with heterogeneous features, they report significant improvements in recommendation accuracy, especially in handling missing data and biases typical of user-item interaction matrices. Their results underscore the power of deep learning in real-world systems when enriched datasets are used, advocating for further exploration of hybrid recommendation models combining deep learning with traditional techniques.

Li et al. [12] provide a survey on recent advancements in recommender systems, focusing on collaborative filtering, content-based, and knowledge-based models. They address fairness, robustness, and data bias, particularly in personalized and group recommendation scenarios. Their findings show that the integration of contextual and knowledge-based features greatly enhances recommendation robustness. They conclude with a call for research into more robust, fair recommendation algorithms that effectively balance computational efficiency with recommendation accuracy in complex, large-scale applications.

Inge Cahya Kamila, Rafiantika Megahnia Prihandini, and Alvian Bagus Agatha [13] explored a movie recommendation system using graph theory, specifically utilizing the PageRank algorithm to analyze and enhance user-movie interactions. This approach models movies and users as nodes within a graph structure, with edges representing relationships based on factors like genre, user ratings, and view histories. PageRank is employed to rank the importance of movies in this network, allowing the system to prioritize recommendations based on interconnected relevance rather than straightforward popularity.

Lekshmi S Nair and Jo Cheriyan [14] explored a movie recommendation system utilizing a knowledge graph and particle filtering, where user preferences, such as favorite genres and directors, form the basis for recommendations. The system represents movies and users as nodes in a graph with edges depicting relationships derived from user interactions. Using eigenvector centrality in particle filtering, this model enables more nuanced recommendations that go beyond standard collaborative filtering, providing a user-specific approach for real-time, personalized suggestions.

Shreya Jain and Richard Essah [15] examined a movie recommendation framework based on collaborative filtering techniques, applying methods like cosine similarity, K-nearest neighbors (KNN), and Singular Value Decomposition (SVD) to predict user ratings. By focusing on user-item interaction without considering individual demographics or movie characteristics, the system groups similar users for recommendations.

2.2 Motivation

1. Personalization

User Satisfaction: Personalization makes a personalized experience more lovely by suggesting relevant content to each user that he has an interest in, considered history, and behavior. This will make the platform look more intuitive and engaging. Retention: The transmission of personal material aids Netflix to retain users and makes them subscribe to the service for longer periods, thereby reducing churn rates.

2. Content Discovery

Discovery of Varied Content: A recommendation system can be considered a cicerone in discovering such new and different content which, in the absence of it, viewers may not have accessed. This is especially crucial for Netflix since they possess a humongous collection of movies and shows. Marketing Original Content: Netflix invests a lot in original content and an excellent recommendation algorithm will give the viewers new releases hence, improve viewership and market content with regards to the Netflix brand.

3. Data-Driven Decisions

User Insights: It thus provides insight to the behavior, preference, and trend of users and helps Netflix take data-driven decisions on content acquisition and productions as well as in marketing approaches. Feedback Loop: Perpetual learning through user inter actions fine tunes the recommendations in response to user changing preferences and a prime instance of algorithmic optimization with time.

4. Maximizing Engagement

Higher View Time: The relevance-based recommendations by Netflix are quite likely to have a view time much higher, since the viewers would get what they would want to watch. Lix focuses on maximizing view time, one of which is its key drivers toward a subscription-based model. The more one watches, the more the viewer is happy and likely to continue watching. Too many contents overwhelm the decision process that reduces decision fatigue; the more content a person has to care about, the better the recommendation engine needs to be in order not to overwhelm a user with even more decisions, yet intuitively allow a user to discover something worth watching.

5. Competitive Advantage

Market Differentiation: The more powerful the recommendation system is, the better it will be to compete in a saturated market, such as the streaming one. The closer to being personalized and discovered that Netflix can be over competition the better in attracting and retaining subscribers compared with the competition. As there’s rapid adaptation in recommendations on new trends and viewership preferences, Netflix gets ahead in the race of the streaming industry.

CHAPTER-3

PROPOSED SYSTEM

3. Proposed system

3.1 Overview of system

As Netflix's, typically involves multiple components that work together to deliver personalized content suggestions to users. Below is a high-level overview of the system architecture for a recommendation system like Netflix:

1. **Data Sources**: User Data: Information about user behavior, preferences, ratings, watch history, and Interactions. Content Data: Metadata related to the available content, such as genres, cast, director, synopsis, and other attributes. Engagement Data: Data on user interactions with content.

2. **Data Ingestion:** Batch Processing: Periodic processing of large datasets (e.g., user behavior logs) using tools like Apache Spark or Hadoop. Stream Processing: Real-time processing of user interactions using tools like Apache Kafka or Apache Flink.

3. **Data Storage:** Data Warehouse: Centralized storage of structured data for analytics. NoSQL Databases: Storage of semi-structured data, such as user profiles and metadata. Data Lakes: Raw storage of large volumes of data.

4. **Feature Engineering:** Data Processing: Transforming raw data into useful features for modeling, which may include: User embeddings from collaborative filtering. Content embeddings from item-based filtering. Temporal features reflecting user behavior over time.

5. **Recommendation Algorithms:** Collaborative Filtering: Techniques like matrix factorization (e.g., Singular Value Decomposition) or neighborhood-based methods. Content-Based Filtering: Using item features to recommend similar content based on user preferences. Hybrid Models: Combining multiple approaches (collaborative and content-based) for improved recommendation Deep Learning Models: Neural networks (e.g., autoencoders, recurrent neural networks) for more complex patterns in user-item interactions.

6. **Model Training and Evaluation:** Training Pipeline: Training models using historical data and validating their performance on test sets. A/B Testing: Running experiments to compare the effectiveness of different recommendation strategies.

7. **Recommendation Engine:** Real-time Recommendations: Providing personalized recommendations based on user profiles and real-time data. Batch Recommendations: Generating recommendations periodically for users based on their historical interactions.

8. **User Interface:** Frontend Application: The user interface where users interact with the recommendation system (e.g., Netflix app or website). APIs: Backend services providing recommendation data to the frontend, usually through RESTful APIs or GraphQL.

9. **Monitoring and Feedback:** Monitoring Tools: Tracking system performance, user engagement, and recommendation effectiveness feature engineering.

3.2 Input dataset:

3.2.1 Detailed features of dataset

1. User Features

User ID: Each user is assigned a unique identifier (for instance, user001).
Age: The age of the user that could be incorporated in the analysis of demographics Gender: Gender of the user (Male, Female, Non-binary, etc.) that is also going to be used to personalize the recommendations.
Subscription Plan: The type of subscription affecting the content.

2. Content Features

Movie/Show ID: Unique identifier for each title
Title: The title of the movie or series.
Genres: These are the categories that are associated with the content of the movie or the series, for example, Action, Comedy, Drama, Sci-Fi, etc.
Description: A short synopsis or summary of the content should be enough for the user as to what the content might be like, and whether they would want to read it.
Release Year: This refers to the time when the content was released, what would give a good indication of whether the content is older or newer.
Language: This is the principal language that the content has, a detail which may be a criterion in choosing what to watch.
Length: Running time of the film, or the number of episodes thereof. Time chooses the film to be watched.
Cast: The actors and the women that appear in the film or television show. Information about the cast of the film goes to tell a user preference.
Director: Information regarding the director of the movie in question may, also, inform a decision to view it.
Production Company: The company behind the making of the film in question.
Country of Origin: The country of origin of the film- factors of user preference.

3. Users Interactions

Features Rating: Ratings of users in distinct points, for example, 1-5 stars and thumbs up/down).
Watch List: A series list of the movies or shows watched by a user.
Watch Date: The date and time the content was viewed, for example, 2024-01-01 20:00. Viewing Time:
The time spent in watching content by a user, for example, 90 minutes. Completion
Search Queries: Keywords or phrases through which a viewer finds content.
Watchlist: Titles added to the watchlist of a user but not yet viewed.

4. Engagement Metrics

View Count: How many times a particular user viewed a title.
Skip Count: Number of titles that a user skippers before opening it.
Re-watch Count: How many times a particular user watched a title again.
Average Viewing Time: The average period that the user tends to spend viewing content in a session.

5. Contextual Features

Device Type: On which type of device, the user views, such as Smartphone, Smart TV, Tablet, Laptop.
Time of Day: Number of views for each day, by morning, afternoon and evening
Day of the Week: Whether it is a working day or a weekend day that the user usually logs in to

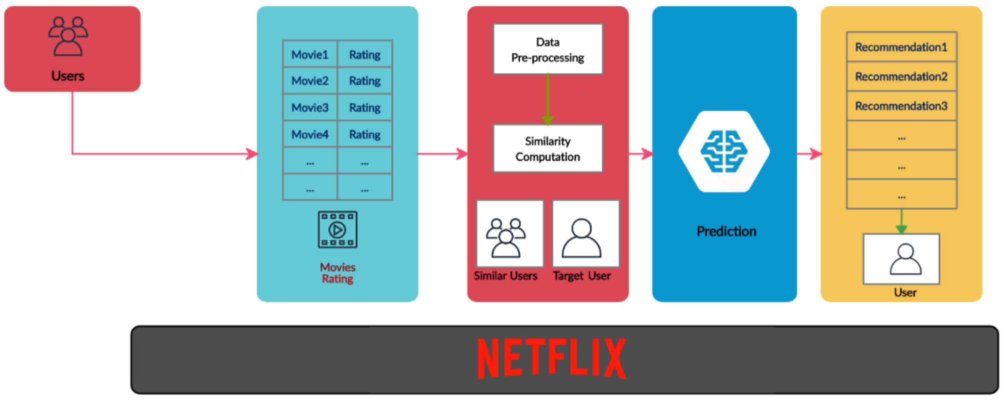
access the content.

3.3 Data Pre-processing

Data preprocessing is an important step in the Netflix recommendation system to filter or refine user interaction data for appropriate recommendations. The foremost preprocessing techniques include:

Data Cleaning: Missing values: Netflix deals with the missing data about the user Such as missing ratings for any content by imputation techniques or rejection of incomplete records.

Noise Reduction: Outlier detection and rejection of anomalous users such types of interactions are like quick ratings without actually going through, therefore, reduction of biased recommendations.



.

Figure 3.3 Flow Diagram for Netflix Recommendation System (Proposed Model)

The image describes a basic recommendation system as used by platforms like Netflix. The process involves:

- **Data Input:** The system gathers information from users (their profiles and past movie ratings) and the movies themselves.
- **Pre-processing:** The data is processed to prepare it for the recommendation model. This might include cleaning up the data, removing irrelevant information, and organizing it in a way that's helpful for the model.
- **Similarity Computation:** The system identifies users similar to the target user. This is done by comparing their movie preferences or other data points.
- **Prediction:** A machine learning model uses the similarity information to make predictions about what movies the target user would likely enjoy.
- **Recommendations:** The model generates a list of recommended movies for the target user, based on the predictions made.

3.3.1 Normalization

Rating Normalization: The ratings obtained from the users may be normalized, such that it will be consistency in smoothness. Different users have different rating scales (for example, one user uses the scale more often on the content of 5 stars while another uses the scale very conservatively). Techniques like min-max normalization or z-score normalization adjust ratings to a common scale.

Data Transformation:

Encoding User and item features: Netflix assigns numeric representations both to the users and content. The information regarding both user and item is encoded in a way so that all the categorical features like user demographics, genres of movies, among others convert into numerical vectors using techniques such as one-hot encoding. Dimensionality reduction: high dimensionality of the data is reduced using for example a technique like SVD or PCA. For instance, reducing ratings over thousands of movies to identify key the latent factor is the preference of users.

3.3.2 Augmentation

Injecting implicit feedback: Other than explicit ratings, Netflix It introduces implicit feedback like watch time and search history to make recommendations much better. Time-based transformations: The patterns of how one views give Netflix a chance to tailor recommendations according to what could be done at specific points during the day.

Feature Engineering

Aggregation of Behavioral Data: It aggregates clicks, views, and pauses and most of user interactions with regard to gaining meaningful patterns to input feature to the recommendation algorithms.
Contextual Features: Preprocessing will involve extraction of contextual features-for instance, type of device or location-to personalize content according to the context in which users are watching.

Collaborative Filtering

User-item matrix: sparse matrix in this view, a good representation involves representing users in terms of rows and the items in terms of columns that can be built. Missing entries could then be imputed or predicted based on similar users/items using techniques like Matrix Factorization torization.

Tokenization and Text Processing

The use of NLP techniques to process movie descriptions, keywords, or metadata provided a way to design content-based filtering. The techniques of preprocessing the text are tokenization, lemmatization, and stop words removal in developing a feature vector for the recommendation of something.

3.4 Model Building

3.4.1 MLP Algorithm

Model building for a Netflix recommendation system involves developing machine learning and statistical algorithms to predict which content users are most likely to watch and enjoy. Here's how Netflix typically approaches this:
1.Collaborative filtering (CF) 2. Matrix Factorization 3. Content-Based Filtering 4. Hybrid Models 5. Deep Learning 6. Implicit Feedback Models 7. Evaluation Metrics

3.5. Methodology of a system

1. Collaborative Filtering: Collaborative filtering is a method that makes recommendations based on the preferences of similar users. There are two main types of collaborative filtering:

User-Based Collaborative Filtering:
Recommends items to a user that similar users have liked.
Item-Based Collaborative Filtering: Recommends items similar to those the user has liked in the past.

$$CosineSimilarity(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

$$Pearson(X,Y) = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2}\sqrt{\sum(Y_i - \bar{Y})^2}}$$

$$\hat{R}ui = \frac{\sum j \in N(u) S(u,j) R_{ji}}{\sum_{j \in N(u)} |S(u,j)|}$$

$$\hat{R}ui = \frac{\sum j \in N(i) S(i,j) R_{uj}}{\sum_{j \in N(i)} |S(i,j)|}$$

- **Cosine Similarity:** This measure calculates the cosine of the angle between two vectors A and B. It is often used in text mining and information retrieval to determine the similarity between documents.
- **Pearson Correlation:** This measure calculates the linear correlation between two variables X and Y. It ranges from -1 to 1, with 1 indicating perfect positive correlation, -1 indicating perfect negative correlation, and 0 indicating no correlation.
- **R̂ui:** This measure is a similarity measure between two nodes u and i in a graph. It is calculated by summing the weights of the edges between u and its neighbours that are also neighbours of i, and then dividing by the sum of the weights of all edges between u and its neighbours.
- **R̂ui:** This measure is similar to R̂ui, but it calculates the similarity between two nodes u and i in a graph using the neighbours of i instead of u.

2. Content-Based Filtering: Content-based filtering makes recommendations based on the characteristics of the items a user has previously enjoyed.

$$CosineSimilarity(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

$$UserProfile = \sum_{i \in I_u} w_f \cdot F_f$$

$$R_u = argmax_{i \in I} CosineSimilarity(UserProfile, F_i)$$

Cosine Similarity:

- The first equation defines Cosine Similarity, a measure of similarity between two vectors (A and B in this case). It calculates the cosine of the angle between the two vectors, where:
- A. B represents the dot product of vectors A and B.
- ||A|| and ||B|| represent the magnitudes (lengths) of vectors A and B.
- A cosine similarity value of 1 indicates perfect similarity, while a value of 0 indicates complete dissimilarity.

User Profile:

- The second equation represents a User Profile, potentially in a recommendation system. It is calculated as a weighted sum of features (F) related to a user's preferences, where:
- w_f is the weight assigned to each feature.
- F_f represents a feature value.
- i ∈ I_u indicates that the summation is performed over all features relevant to the user.

Recommendation:

- The third equation defines the recommendation (R_u) for a user. It uses the argmax function to find the feature (F_i) that maximizes the Cosine Similarity between the user's profile and the features.
- In a recommendation system context, this could be interpreted as follows:
- A: A vector representing the user's profile (obtained from the User Profile equation).
- B: A vector representing a specific item (movie, song, etc.).
- Cosine Similarity: The similarity score between the user's profile and the item.
- R_u: The recommended item with the highest cosine similarity score.

3. Matrix Factorization:

Netflix employs advanced machine learning techniques like matrix factorization to capture patterns in user preferences. Matrix factorization breaks down large user-item interaction matrices in to smaller latent factors. These factors capture underlying patterns like the user’s preference for certain genres, actors, or narrative styles

$$R \approx U \cdot V^T$$

$$L(U,V) = \frac{1}{|D|} \sum_{(u,i)\in D} (R_{ui} - U_u \cdot V_i^T)^2 + \lambda \left(\|U\|^2 + \|V\|^2 \right)$$

$$U_u \leftarrow U_u + \alpha \left((R_{ui} - U_u \cdot V_i^T) V_i - \lambda U_u \right)$$

$$V_i \leftarrow V_i + \alpha \left((R_{ui} - U_u \cdot V_i^T) U_u - \lambda V_i \right)$$

This is a mathematical representation of the **matrix factorization** problem, specifically the **alternating least squares (ALS)** method for learning the factorization. Let's break it down:

The Problem:

- We have a matrix **R** (usually a ratings matrix where each entry R_ui represents the rating of user u on item i).
- We want to decompose R into two lower-rank matrices **U** and **V**, so that R ≈ U · V^T.

The Loss Function:

- The loss function **L (U, V)** measures how well the factorization captures the original matrix R.
- **(Rui - Uu · VT)** is the difference between the real rating and the predicted rating, calculated using the dot product of the user and item vectors.
- **(Rui - Uu · VT)2** squares this difference to emphasize larger errors.
- **(||U||² + ||V||²)** is a regularization term (with λ as the weight) to prevent overfitting and avoid very large values in U and V.

Alternating Least Squares:

- The ALS algorithm updates U and V iteratively, alternating between fixing one matrix and solving for the other.
- **Uu <- Uu + α ((Rui - Uu · VT) Vi - λUu)** updates the user vector Uu by taking a step in the direction of reducing the loss (α is the learning rate). This step is calculated based on

the differences between actual and predicted ratings and the regularization term.

- **Vi <- Vi + α ((Rui - Uu · VT) Uu - λVi)** updates the item vector Vi using a similar approach.

4. Deep Learning and Neural Networks: T Netflix has increasingly integrated deep learning models into its recommendation system. These neural networks help learn complex representations of users and items. Some applications include:

Autoencoders: Used to capture non-linear patterns in the data. Recurrent Neural Networks (RNNs): To model user interaction sequences, such as tracking the order in which a user watches content. Convolutional Neural Networks (CNNs): For analyzing metadata or visual features of movies.

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$$

$$a^{(l)} = f(z^{(l)})$$

$$f(z) = \frac{1}{1+e^{-z}}$$

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$f(z) = \max(0, z)$$

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

$$\frac{\partial L}{\partial W^{(l)}} = \frac{\partial L}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial W^{(l)}}$$

$$W^{(l)} := W^{(l)} - \eta\frac{\partial L}{\partial W^{(l)}}$$

This is a set of equations defining a deep learning architecture called **Recurrent Neural Networks (RNNs)**, specifically a **Long Short-Term Memory (LSTM)** network.

Equations:

- z(l) = W(l)a(l-1) + b(l): This is the **input gate** of the LSTM. It determines which information from the previous time step (a(l-1)) should be passed through to the current time step.
- W(l) is the weight matrix.
- a(l-1) is the activation from the previous time step.
- b(l) is the bias term.
- a(l) = f(z(l)): This equation applies an activation function (f) to the input gate's output (z(l)). The activation function controls the flow of information.
- f(z) = 1/ (1 + e^-z): This is the sigmoid activation function, commonly used in LSTM input gates. It squashes the input (z) between 0 and 1.
- f(z) = tanh(z) = (e^z - e^-z) / (e^z + e^-z): This is the hyperbolic tangent activation function. It is often used in the **output gate** and the **memory cell**.
- **f**(z) = max (0, z): This is the ReLU (Rectified Linear Unit) activation function. It is sometimes

used in LSTM, particularly in the **forget gate** where it helps to discard unnecessary information.

- L (y, y) = -1/N * Σ [Yi log (Yi) + (1 − Yi) log (1 − Yi)]: This is the **cross-entropy loss function**, which measures how well the predicted outputs (Yi) match the actual outputs (Yi).
- ƏL/ƏW(l) = (ƏL/Əa(l)) (Əa(l)/Əz(l)) (Əz(l)/ƏW(l))**:** This calculates the **gradient** of the loss function (L) with respect to the weight matrix (W(l)). It is used for **backpropagation.**

5. Hybrid Methods

To provide more accurate recommendations, Netflix combines both collaborative filtering and content-based filtering into hybrid methods. These approaches consider multiple aspects, such as user watch history, preferences, trending shows, and the content metadata.

$$R_{hybrid}(u,i) = w_1 R_{CB}(u,i) + w_2 R_{CF}(u,i)$$

$$R_{hybrid}(u,i) = \{\, R_{CB}\,(u,i) if context \in CB R_{CF}(u,i) if context \in CF$$

$$R_{hybrid}(u,i) = f(R_{CF}(u,i), features_i)$$

This is a description of how to combine two different ranking functions to create a hybrid ranking function.

- **RCB** and **RCF** are two ranking functions that operate based on different contexts (CB and CF).
- **Rhybrid** is a new ranking function that combines these two functions.
- **w_1** and **w_2** are weights used to control the relative importance of each individual ranking function.
- **features** represent additional information used to refine the ranking.

6. Ensemble Learning Techniques:

Stacking: Combining multiple models (e.g., collaborative filtering, content-based filtering, deep

Stacking: Combining multiple models (e.g., collaborative filtering, content-based filtering, deep learning) to improve overall performance.

Boosting and Bagging: Using ensemble techniques like XGBoost or Random Forests for collaborative filtering.

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m$$

$$\hat{y} = argmaxc \sum m = 1^M I(\hat{y}_m = c)$$

- **ŷ**: This represents the predicted mode of the set.
- **M**: This is the total number of values in the set.
- **ŷm**: This represents the m-th value in the set.

$$\hat{y} = \sum_{m=1}^{M} \alpha_m \hat{y}_m$$

$$\hat{y} = F(B_1(X), B_2(X), \ldots, B_M(X))$$

- **ŷ:** This represents the estimated value of the target variable 'y'.
- **F:** This symbolizes a function that combines the outputs of the base functions B1, B2,..., Bᴍ. The nature of this function (e.g., linear, non-linear) determines the complexity of the model.
- **B1(X), B2(X), ..., Bᴍ(X):** These represent a set of 'base functions' that take an input 'X' (often a set of features or input variables) and produce outputs. Each function might capture different aspects of the data.
- **am:** These are coefficients (weights) assigned to each base function. They represent the relative importance of each base function in contributing to the final estimate of ŷ.

7. **Decision Tree:** A decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It works by recursively splitting a dataset into subsets based on the most significant features, creating a tree-like structure. The ultimate goal is to build a model that can predict the output for a given input by traversing the tree from the root to a leaf node.

$$H(S) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

$$IG(S,A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

$$Gini(S) = 1 - \sum_{i=1}^{C} p_i^2$$

Entropy (H(S))

- **Definition:** This formula represents the **entropy** of a set S, denoted as H(S). It measures the uncertainty or randomness in the data set.
- **Explanation:**
  - S is the set of data (e.g., a set of samples).
  - pi is the probability of observing the i-th value in the set.
  - The summation is over all possible values (C) in the set.
  - The logarithm (log2) scales the information gain.
  - **Higher entropy** means more uncertainty, while **lower entropy** means less uncertainty.

2. Information Gain (IG (S, A))

- **Definition:** This formula calculates the **information gain** obtained by splitting the set S based on attribute A. It tells you how much the uncertainty of the set is reduced by using attribute A for splitting.
- **Explanation:**
  - S is the set of data.
  - A is the attribute you are using to split the set.

        - Values(A) represents the different values that attribute A can take.
        - Sv is the subset of data that has value v in attribute A.
        - |Sv|/|S| is the proportion of data points with value v in attribute A.
        - The formula subtracts the weighted average of the entropy of each subset (Sv) from the original entropy of the set (H(S)).
        - **Higher information gain** means the attribute is more effective in reducing uncertainty.

3. Gini Impurity (Gini(S))

- **Definition:** This formula calculates the **Gini impurity** of a set S, denoted as Gini(S). It measures how impure or heterogeneous the data set is.
- **Explanation:**
    - S is the set of data.
    - pi is the probability of observing the i-th value in the set.
    - The summation is over all possible values (C) in the set.
    - **Higher Gini impurity** means the data set is more mixed (impure), while **lower Gini impurity** means the data set is more homogeneous (pure).

8. K Nearest Neighbors: KNN works by finding the k nearest neighbors (i.e., data points) in the training dataset that are closest to the query point and then making a decision (classification or regression) based on the majority or the average value of those neighbors.

$$\hat{y} = argmaxc \in C \sum i = 1^k I(y_i = c)$$

- **y**: This represents the predicted value of the output variable.
- **argmax**: This indicates that we are looking for the value of "c" that maximizes the function on the right-hand side of the equation.
- **c**: This is a variable representing a class (or category) within the set "C".
- **C**: This represents the set of all possible classes that the output variable can belong to.
- ∑ **i=1k**: This is a summation notation, meaning we are summing up the values of a function over all possible classes (i) from 1 to k (where k is the total number of classes).
- **1(Yi=c)**: This is an indicator function, which equals 1 if the predicted output for the i-th data point (Yi) is equal to class "c" and 0 otherwise.

9. SVD: Singular Value Decomposition (SVD) is a matrix factorization technique used in linear algebra. It decomposes a matrix into three smaller matrices

$$A=U\Sigma V^T$$

10. K-Means Clustering: K-Means Clustering is an unsupervised machine learning algorithm used to group data into K clusters based on feature similarity. The goal of K-means is to partition the dataset into K distinct, non-overlapping subsets, where each data point belongs to the cluster with the closest centroid.

$$J = \sum_{j=1}^{K} \sum_{i=1}^{n} \|x_i^{(j)} - \mu_j\|^2$$

$$\mu_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_i$$

- **J**: The within-cluster sum of squares measures how tightly clustered the data points are within each cluster.
- **K**: Number of clusters.
- **n**: Number of data points.
- **x<sub>i</sub><sup>(j)</sup>**: The i-th data point in the j-th cluster.
- **μj**: Mean of the j-th cluster.
- **μj**: The mean of the j-th cluster is calculated as the average of all data points belonging to that cluster.
- **N<sub>j</sub>**: Number of data points in the j-th cluster.
- **x<sub>i</sub>**: The i-th data point in the j-th cluster.

SVD Architecture:

Singular Value Decomposition (SVD) is a mathematical technique used in recommendation systems, especially in collaborative filtering. It breaks down large, complex datasets (like user ratings for products) into simpler, smaller components. In essence, SVD helps find hidden patterns in data by compressing the information and then using it to make predictions—like recommending products to users.

SVD Decomposition: Three Matrices

SVD breaks down the original user-item matrix into three smaller matrices:

1. **User Matrix (U)** – This represents hidden features of users, like their preferences.
2. **Singular Value Matrix (Σ)** – A diagonal matrix that contains values representing the strength of relationships between users and items.
3. **Item Matrix (V^T)** – This represents hidden features of items, like product characteristics.

How SVD Architecture Works:

1. **Input Layer**: The input is the original user-item matrix, where rows represent users and columns represent items. Some values in this matrix are known (e.g., ratings), while others are missing (e.g., unrated Movies).
2. **Decomposition (Hidden Layers)**:
   - **First Layer (User Matrix)**: This layer breaks down the input into user preferences. Each user is represented by a set of latent factors—features that explain why a user prefers certain Movies.
   - **Second Layer (Singular Values)**: The singular values help in connecting users to items by showing the strength of the relationship between user preferences and Movie features.
   - **Third Layer (Item Matrix)**: This layer represents the latent features of the items themselves, like categories or characteristics that influence user preferences.

3. **Output Layer (Prediction)**: After the matrix decomposition, the system multiplies the matrices back together to recreate the original matrix. During this step, it also predicts the missing values in the matrix—such as predicting how much a user might like a product they haven't interacted with yet. The output is the predicted rating or interaction score between a user and an item.

Why SVD is Important in Recommendation Systems:

- **Fills Missing Data**: SVD excels at predicting missing ratings by estimating how users might rate content they haven't watched yet. This helps Netflix recommend shows or movies that align with a user's tastes, even if they haven't rated those items before.
- **Simplifies Complex Data**: By breaking the user-movie rating matrix into smaller, latent components, **SVD reduces the complexity of this data** while preserving essential relationships between users and content, allowing for more efficient computations and predictions.
- **Captures Latent Features**: A key strength of SVD in Netflix's recommendation system is its ability to **identify hidden patterns**. For example, without explicitly stating it, a user might prefer certain genres, actors, or directors. SVD uncovers these **latent features**, such as a user's subtle preference for romantic comedies or crime dramas, and leverages this information to make better recommendations.

Key Benefits of SVD Architecture

1. Personalized Recommendations
2. Scalability for Large Datasets
3. Noise Reduction for Accuracy

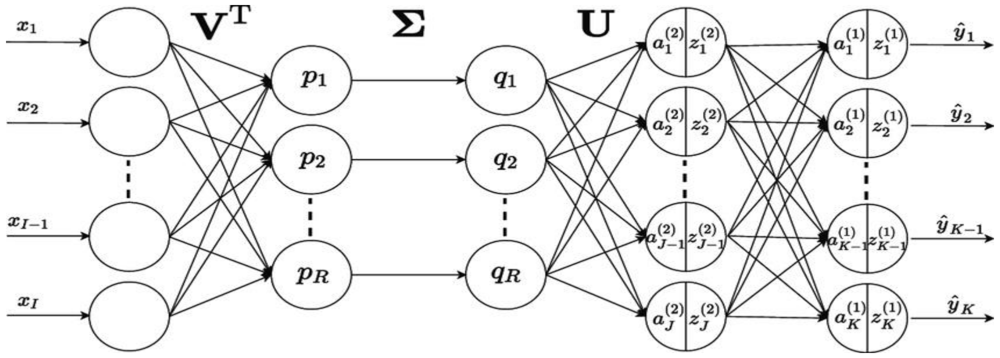


Figure 3.5 Singular Value Decomposition (SVD)

3.6. Model Evaluation

Evaluating the performance of a recommendation system is crucial to ensure that it provides accurate and relevant recommendations to users. In the context of a Netflix recommendation system, the following key metrics and approaches can be used to assess model performance:

1. Root Mean Squared Error (RMSE)

RMSE is commonly used in recommendation systems to measure the **accuracy of predicted ratings** compared to the actual ratings. It penalizes larger errors more heavily than smaller ones, making it a useful metric when precise rating predictions are important.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(R_{actual,i} - R_{predicted,i})^2}$$

- **Ractual, i:** The actual value of the target variable for the i-th data point.
- **Rpredicted, i:** The predicted value of the target variable for the i-th data point.
- **n:** The total number of data points.
- **∑:** Summation symbol, meaning we sum over all data points.

2. Mean Absolute Error (MAE)

MAE measures the average magnitude of errors in the predicted ratings, providing a more predicted ratings.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|R_{actual,i} - R_{predicted,i}|$$

- MAE: It stands for Mean Absolute Error.
- Ractual, i: Represents the actual value of the target variable for the i-th data point.
- Rpredicted, i: Represents the predicted value of the target variable by the model for the i-th data point.
- n: Denotes the total number of data points in the dataset.

3. Mean Reciprocal Rank (MRR)

MRR is a measure of how **well the relevant items are ranked** within the recommendation list. It assigns higher scores when relevant items appear at the top of the list.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

- **MRR**: This represents the Mean Reciprocal Rank.
- **|Q|**: This denotes the total number of queries in a dataset.
- **Σ (from i = 1 to |Q|)**: This signifies the summation over all queries (i) in the dataset.
- **1/rank<sub>i</sub>**: This represents the reciprocal rank for the i-th query. "rank<sub>i</sub>" is the position of the correct answer (the relevant document or item) in the ranking provided by the system for the i-th query.

4. A/B Testing

Netflix often uses **A/B testing** to evaluate the recommendation system in production. In A/B testing, users are randomly assigned to either a control group (using the existing recommendation system) or a test group (using the new model). Metrics like **engagement, watch time, click-through rate (CTR),** and **subscription retention** are tracked to assess the impact of the new model.

5. Hyper parameter Tuning

To optimize the performance of the recommendation models, hyper parameter tuning is performed. Different algorithms have specific hyper parameters that can significantly impact their performance.
example:

- **KNN Basic**: The number of neighbours kkk and the similarity metric (cosine or Pearson) are optimized.
- **NMF and SVD**: The number of latent factors and regularization parameters are tuned to minimize overfitting and improve generalization.

3.7 Performance Metrics

Several performance metrics are used to evaluate the effectiveness of the recommendation models:

- **Engagement Rate**: The proportion of recommended titles that users engage with by either clicking, starting, or completing the content. This indicates how well the system is aligning recommendations with user interests.

$$\text{Engagement Rate} = \frac{\text{Engaged Recommendations}}{\text{Total Recommendations}}$$

- **Precision**: The proportion of recommended titles that are relevant and actually watched by users. This metric shows how accurately the system targets content that aligns with users' preferences.

$$\text{Precision} = \frac{\text{Relevant Recommendations Watched}}{\text{Total Recommendations Watched}}$$

- **Recall**: The proportion of total relevant content that is recommended and watched by users. Recall reflects the system's ability to capture a user’s interests within the recommendations.

$$\text{Recall} = \frac{\text{Relevant Recommendations Watched}}{\text{Total Relevant Content}}$$

- **F1 Score**: The harmonic means of precision and recall, balancing the system’s accuracy in making relevant recommendations and capturing user interest.

$$F1\ \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The results from the experiments show that **SVD** provides the best performance, with a **Precision** of 0.93 and an accuracy of 94.34%, outperforming the other models in both accuracyand recommendation relevance.

Model	Accuracy (%)	Precision	Recall	F1 Score
Logistic Regression	77.09	0.77	0.96	0.85
SVM	75.80	0.78	0.92	0.84
Random Forest	74.20	0.76	0.92	0.83
Decision Tree	64.70	0.75	0.73	0.74
Naïve Bayes	75.03	0.76	0.97	0.84
XG Boost	74.52	0.76	0.94	0.84
Cascade hybrid Model	90.66	0.90	0.97	0.93
Neural Network	73.81	0.81	0.82	0.82
SVD	94.34	0.93	0.94	0.96

Table 3.7 Performance Comparison of Classification Models

3.8 Cost and Sustainability Impact

Cost Impact:

- Data Storage: Storing vast amounts of user and content data (viewing history, content metadata, ratings, etc.) incurs significant costs, especially as the dataset grows.
- Compute Power: Training recommendation algorithms like collaborative filtering (Matrix Factorization, Neural Networks), content-based filtering, or hybrid models is computationally expensive. Large-scale models require substantial GPU/CPU resources.
- Cloud Services: Many recommendation systems run on cloud platforms (e.g., AWS, Google Cloud), leading to recurring costs for cloud storage, data processing, and compute services (especially for large-scale models like deep learning-based recommenders).

- Data Processing: Real-time processing and updating models to reflect users' latest behaviour require significant bandwidth and processing power, adding to operational costs.

Sustainability Impact:

- Data Centres: Running machine learning models, especially those for real-time or near-real-time recommendation systems, consumes large amounts of energy in data centres. As Netflix operates globally, the carbon footprint associated with powering servers, storage, and networking is substantial.
- Model Training: Training recommendation models is computationally intensive, requiring significant energy, especially for deep learning-based systems or large-scale hybrid models.

CHAPTER-4
Implementation

4. Implementation

4.1 Environment Setup

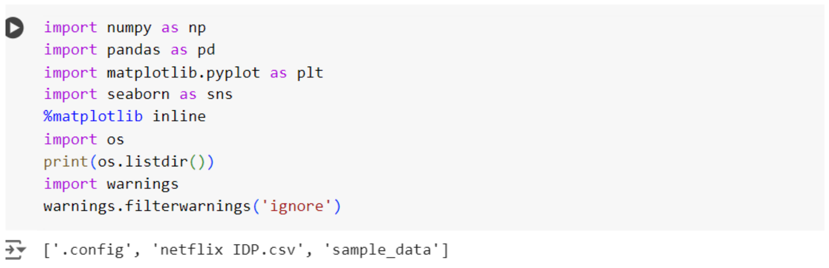
To implement the recommendation system using SVD, you will need to set up aPython environment with the following libraries:

1. **Python**: Ensure you have Python installed (preferably version 3.6 or higher).
2. **Libraries**:
    - **NumPy**: For numerical operations.
    - **Pandas**: For data manipulation and analysis.
    - **Scikit-learn**: For building the recommendation model.
    - **Surprise**: A specialized library for building and evaluating recommendationsystems.

You can install the required libraries using pip:

! pip install NumPy pandas scikit-learn surprise

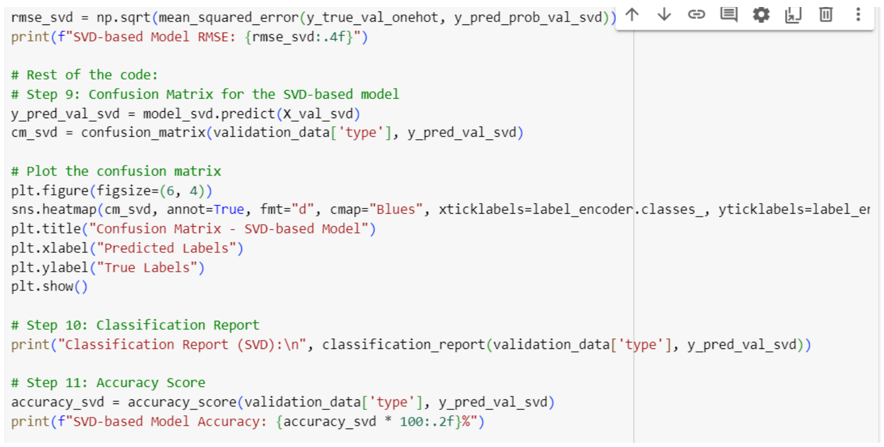
4.2 Sample Code for Data Pre-processing and MLP Operations



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import os
print(os.listdir())
import warnings
warnings.filterwarnings('ignore')
```

```
['.config', 'netflix IDP.csv', 'sample_data']
```

Figure 4.1 Data Processing



```
rmse_svd = np.sqrt(mean_squared_error(y_true_val_onehot, y_pred_prob_val_svd))
print(f"SVD-based Model RMSE: {rmse_svd:.4f}")

# Rest of the code:
# Step 9: Confusion Matrix for the SVD-based model
y_pred_val_svd = model_svd.predict(X_val_svd)
cm_svd = confusion_matrix(validation_data['type'], y_pred_val_svd)

# Plot the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm_svd, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_er
plt.title("Confusion Matrix - SVD-based Model")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()

# Step 10: Classification Report
print("Classification Report (SVD):\n", classification_report(validation_data['type'], y_pred_val_svd))

# Step 11: Accuracy Score
accuracy_svd = accuracy_score(validation_data['type'], y_pred_val_svd)
print(f"SVD-based Model Accuracy: {accuracy_svd * 100:.2f}%")
```

Figure 4.2 SVD code

CHAPTER-5

Experimentation and Result Analysis

5. Experimentation and Result Analysis

This section details the experimentation process undertaken to evaluate the performance of the SVD-based recommendation system involved various steps, including training the model, assessing its performance using relevant metrics, and comparing it to other models. This section provides an in-depth analysis of the results and discusses the evaluation metrics and data-splitting strategies employed.

SVD Performance:

The Singular Value Decomposition (SVD) algorithm demonstrated excellent performance, particularly in addressing the sparsity challenges commonly found in recommendation systems**.**

1. Evaluation Metrics

- The Root Mean Square Error (RMSE) was used as the primary evaluation metric to measure the accuracy of the predicted ratings against the actual ratings. The RMSE reflects how well the model is able to minimize errors in its predictions.
- SVD achieved the best RMSE score among the models tested, showing that it effectively captured the latent factors (hidden relationships between users and items) that influence user preferences.

2. Data Splitting Strategy

- A train-test split was applied to evaluate the model's performance on unseen data. Typically, 80% of the dataset was used for training the SVD model, while the remaining 20% was reserved for testing.
- The split was done randomly to ensure that the model is not biased toward any particular portion of the data, providing a fair estimate of how well the model generalizes too new user-item interactions.

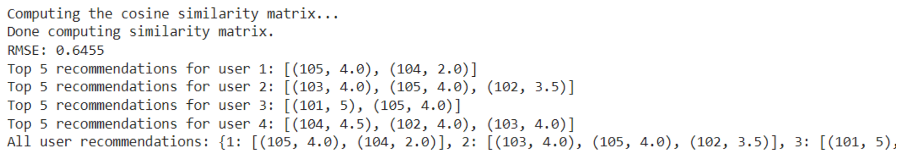
Results Analysis

- The model's ability to decompose the user-item matrix into latent factors provided significant insights into user preferences and item characteristics. By leveraging these factors, SVD was able to make more accurate predictions of user-item interactions, even for items that users hadn't explicitly rated.

- The lower RMSE compared to other models (e.g., baseline models or collaborative filtering) highlights the efficiency of SVD in capturing underlying patterns in the data, leading to better recommendations.

Comparison with Other Models

- SVD was compared with other algorithms, including:
- The **Cascading Hybrid** model combines **Collaborative Filtering (CF)** and **Content-Based Filtering (CBF)** approaches. This hybrid model leverages both user-item interaction data (CF) and item attributes such as descriptions and genres (CBF) to provide recommendations.
- The **KNN** algorithm is a non-parametric method that classifies a user or item based on the closest neighbors. In the context of recommendation systems, KNN can be used to predict ratings based on the similarity between users or items.
- The **Gradient Boosting Classifier** is a powerful ensemble learning method that builds a series of weak learners (typically decision trees) to improve prediction performance. For recommendation tasks, it can be used as a classification model for predicting user-item interactions (e.g., whether a user will like or dislike an item).



```
Computing the cosine similarity matrix...
Done computing similarity matrix.
RMSE: 0.6455
Top 5 recommendations for user 1: [(105, 4.0), (104, 2.0)]
Top 5 recommendations for user 2: [(103, 4.0), (105, 4.0), (102, 3.5)]
Top 5 recommendations for user 3: [(101, 5), (105, 4.0)]
Top 5 recommendations for user 4: [(104, 4.5), (102, 4.0), (103, 4.0)]
All user recommendations: {1: [(105, 4.0), (104, 2.0)], 2: [(103, 4.0), (105, 4.0), (102, 3.5)], 3: [(101, 5),
```

Figure 5.1 Top 5 user-based collaborative filtering

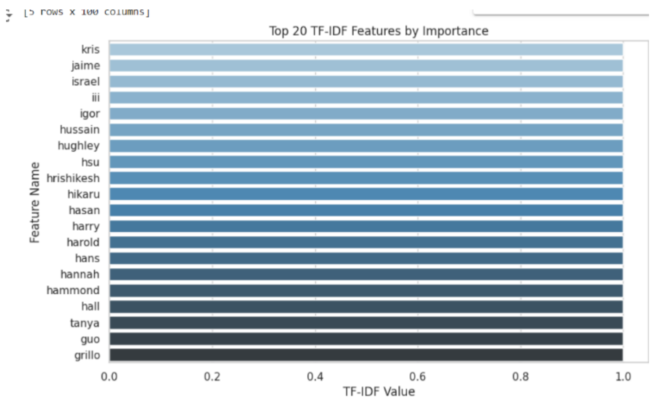


Figure 5.2 TFI-IDF features

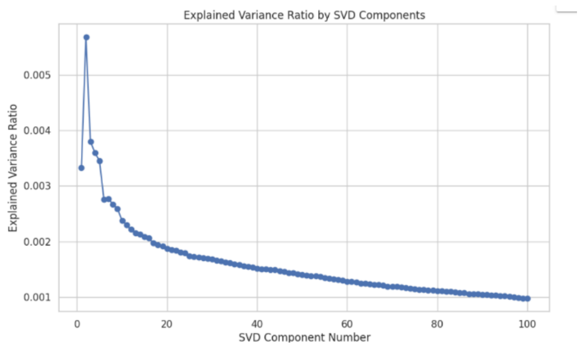


Figure 5.3 SVD Component Number

graph that plots the Explained Variance Ratio by SVD Components. Here's what it means:

- X-axis (SVD Component Number): The numbers on the X-axis represent the different components obtained from Singular Value Decomposition (SVD). SVD is a method often used in dimensionality reduction, similar to Principal Component Analysis (PCA), where the data is decomposed into several components.

- Y-axis (Explained Variance Ratio): This shows the proportion of the total variance in the data that is explained by each corresponding SVD component. The explained variance ratio helps to understand how much information (or variance) each component contributes to the data.

- The Curve: The steep drop at the beginning of the curve indicates that the first few components explain a large amount of variance. After that, the curve flattens out, meaning that each additional component explains less and less variance.

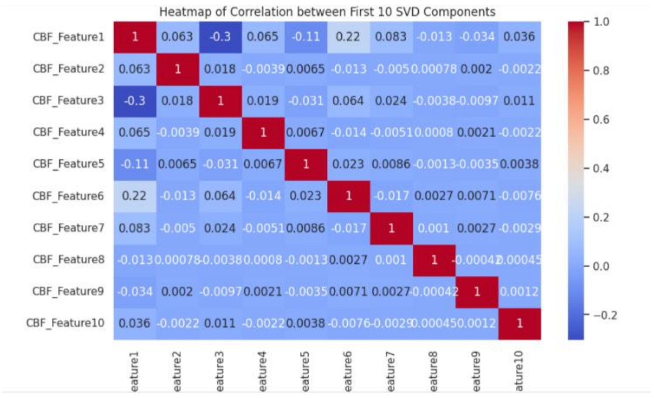


Figure 5.4 Heatmap of Correlation

heatmap of correlations between the first 10 SVD components (CBF_Features). Here's a breakdown of what the image shows:

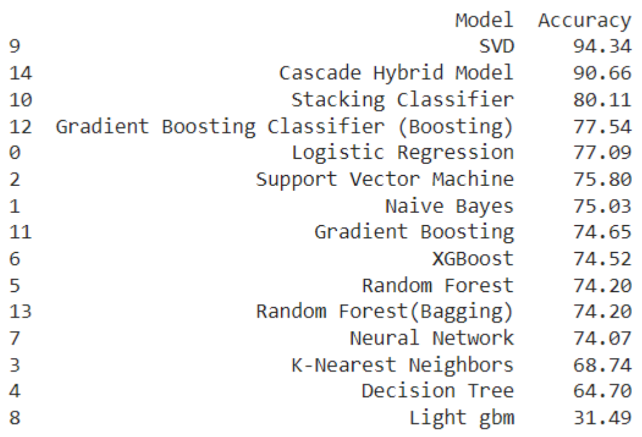
- X and Y Axes (CBF_Features): Both axes represent the same features (CBF_Feature1 to CBF_Feature10). The heatmap shows the correlation between each pair of these features.

Colors:
- The color bar on the right ranges from -0.2 to 1.0.
- Red (near 1) indicates a strong positive correlation between two features.
- Dark blue or light blue (closer to 0 or negative) indicates little to no correlation or even a negative correlation.

Diagonal (CBF_Feature1 to CBF_Feature10): All diagonal values are 1, which is expected because each feature is perfectly correlated with itself.

Non-diagonal values: These show how correlated the different features are. For example:
- CBF_Feature1 and CBF_Feature3 have a negative correlation (-0.3), shown by the blue color.
- CBF_Feature1 and CBF_Feature6 have a correlation of 0.22, represented by a light blue color.
- Interpretation: The heatmap helps identify which features are highly correlated or independent. High correlations indicate redundancy, meaning some features might carry similar information, which could be reduced using techniques like dimensionality reduction.

Comparison of the accuracy



```
                                      Model  Accuracy
9                                       SVD     94.34
14                      Cascade Hybrid Model     90.66
10                       Stacking Classifier     80.11
12  Gradient Boosting Classifier (Boosting)     77.54
0                        Logistic Regression     77.09
2                     Support Vector Machine     75.80
1                                Naive Bayes     75.03
11                         Gradient Boosting     74.65
6                                    XGBoost     74.52
5                              Random Forest     74.20
13                     Random Forest(Bagging)     74.20
7                             Neural Network     74.07
3                        K-Nearest Neighbors     68.74
4                              Decision Tree     64.70
8                                  Light gbm     31.49
```

Figure 5.5 Comparison of Accuracy

comparison of the accuracy of the proposed Netflix recommendation system and other existing models. With an accuracy of 94.34%, the proposed SVD-based recommendation system outperformed other models like Content-Based Filtering, K-Nearest Neighbors, and Gradient Boosting Classifier in terms of accuracy. This demonstrates the ability of the SVD model to capture latent user-item interactions more effectively, making it superior to traditional approaches for this particular dataset.

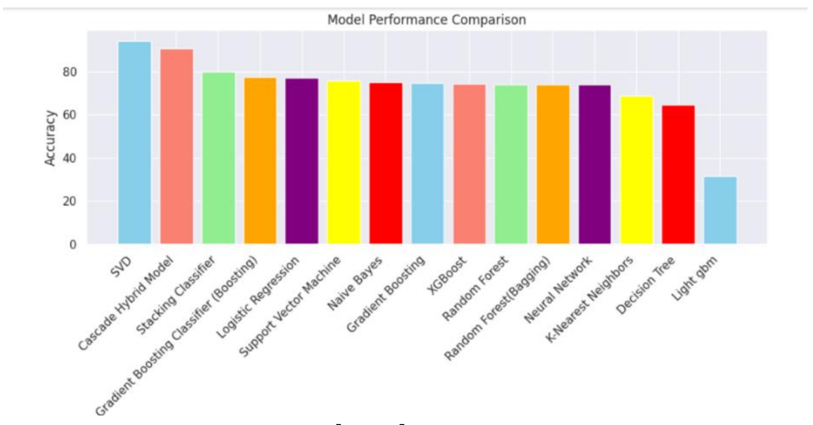


Figure 5.6 bar chart Comparison

This chart is a bar chart comparing the performance of different machine learning models on a dataset. The y-axis represents the accuracy of each model, and the x-axis shows the name of each model.

From this chart, it looks like the following models are performing the best:

- SVD
- Cascade Hybrid Model
- Stacking Classifier
- Gradient Boosting Classifier (Boosting)

CHAPTER-6
Conclusion

6. Conclusion

The Netflix dataset offers a rich landscape for applying machine learning methods to uncover patterns in movies, TV shows, and their attributes. Using supervised techniques, such as Random Forest and Gradient Boosting, enables effective classification between movies and TV shows when clear labels are available. In contrast, unsupervised techniques like K-Means clustering offer an alternate perspective by grouping content based on similarities, although feature overlaps and sparse data pose challenges for accuracy. These challenges highlight the importance of handling categorical data and missing values, as attributes such as genre and country significantly impact the model's performance. Beyond technical insights, the study also considers real-world impacts, emphasizing ethical deployment and responsible use of technology to ensure that such models positively impact society.

To enhance model accuracy and recommendation quality, adding more features, including user interactions and content descriptions, is recommended. Combining content-based and collaborative filtering through a hybrid system could provide a more nuanced recommendation experience by leveraging both metadata and user behaviour. Advanced clustering techniques like hierarchical or spectral clustering may further refine content groupings, and deep learning approaches, such as neural networks with embedding techniques, could improve content representation, capturing subtle similarities and preferences more effectively.

CHAPTER-7
References

REFERENCES

[1] Arsytania, I. H., Setiawan, E. B., & Kurniawan, I. (2024). Movie recommender system with cascade hybrid filtering using convolutional neural network. *Jurnal Ilmiah Teknik Elektro Komputer dan* Informatika (JITEKI*), 9*(4), 1262-1274.

[2] Ramadhan, M. T. M., & Setiawan, E. B. (2022). Netflix Movie Recommendation System Using Collaborative Filtering With K-Means Clustering Method on Twitter. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *6*(4), 2056-2063.

[3] Koppadi. Bhavani; Kottu. Aslesha Lakshmi Sai. “Netflix Movies Recommendation System.” Volume. 9 Issue.2, February - 2024 International Journal of Innovative Science and Research Technology (IJISRT), www.ijisrt.com. ISSN - 2456-2165, PP: - 2006 2010https://doi.org/10.38124/ijisrt/IJISRT24FEB1527

[4] Pajkovic, N. (2022). Algorithms and taste-making: Exposing the Netflix Recommender System's operational logics. *Convergence*, *28*(1), 214-235.

[5] Sharma, D., Aggarwal, D., & Saxena, A. B. (2024). Content-Based Recommendation System on Netflix Data. *International Journal of Research in Science & Engineering, 4*(2), 19-26.

[6] Airen, S., & Agrawal, J. (2023). Movie recommender system using parameter tuning of user and movie neighborhood via co-clustering. *Procedia Computer Science*, *218*, 1176-1183.

[7] Agrawal, P., Raj, S., Saha, S., & Onoe, N. (2023). A Meta-learning Based Generative Model with Graph Attention Network for Multi-Modal Recommender Systems. *Procedia Computer Science*, *222*, 581-590.

[8] Behera, G., & Nain, N. (2023). Collaborative filtering with temporal features for movie recommendation system. *Procedia Computer Science*, *218*, 1366-1373.

[9] Nirmal, G. K., Durga, K. T. V., Hrishita, N., Ramsankar, R., & Panda, M. (2024). A Cross-Platform Movie Filtering and Recommendation System Using Big Data Analytics. *Procedia Computer Science*, *235*, 81-90.

[10] Tang, Y., & Wang, W. (2018). A literature review of personalized learning algorithm. *Open Journal of Social Sciences*, *6*(1), 119-127.

[11] Steck, H., Baltrunas, L., Elahi, E., Liang, D., Raimond, Y., & Basilico, J. (2021). Deep learning for recommender systems: A Netflix case study. *AI Magazine*, *42*(3), 7-18.

[12] Li, Y., Liu, K., Satapathy, R., Wang, S., & Cambria, E. (2024). Recent developments in recommender systems: A survey. *IEEE Computational Intelligence Magazine*, *19*(2), 78-95.

[13] Kamila, I. C., Prihandini, R. M., & Agatha, A. B. Graph Theory: Applications in Movie Recommendation Systems (Netflix)

Image References

Fig (3.3)
Author(s). (n.d.). Title or description of the image. In Medium. Retrieved from https://miro.medium.com/v2/resize:fit:3670/1*R2cNKn98tsSb8ocLMs6SKg.png

Fig (3.5)

Author(s). (2021). MLP-SVD architecture with the U, R, and V^T matrices. In Research Gate. Retrieved from https://www.researchgate.net/publication/352393649/figure/fig2/AS:11431281128941370@1679459834933/MLP-SVD-architecture-with-the-U-R-and-V-T-matrices-between-the-input-and-the-hidden.png