CSE 258 - Assignment 2

Shrenik Jain
A69029862
shj017@ucsd.edu

Sammed Kamate
A69032538
skamate@ucsd.edu

Bhavana Avanthi
A69036271
bavanthi@ucsd.edu

ABSTARCT

This report presents the development of a movie recommendation system using the MovieLens dataset, comprising over 45,000 movies and 26 million ratings. We implement and evaluate multiple models, including content-based filtering, Singular Value Decomposition (SVD), Bayesian Parameter Models, and Autoencoders. Exploratory data analysis reveals patterns in genre distribution, rating trends, and user interactions, guiding feature selection and model design. Evaluation metrics such as RMSE, MAE, Precision@K, and Recall@K demonstrate that Autoencoders outperform traditional methods by capturing complex, non-linear user-item relationships. The study highlights the importance of model selection based on data characteristics, addressing challenges like sparsity, scalability, and the cold-start problem.

I. DATASET

A. *Description*

The dataset we will use is The Movies Dataset, which includes metadata for over 45,000 movies and 26 million ratings from more than 270,000 users. It contains several CSV files with different aspects of the data, such as ratings, cast, crew, plot keywords, budget, revenue, release dates, languages, production companies, and countries. The dataset consists of movies released on or before July 2017 and is available from the official GroupLens website.

This dataset was chosen for its rich metadata, which supports both content-based and collaborative filtering approaches, and its large number of user ratings, providing sufficient data for training and evaluating various models. The diversity in genres, languages, and user demographics provides an opportunity to build a generalized recommendation system that can cater to a wide audience We will explore the dataset in more detail in the subsequent section.

B. *Exploratory Data Analysis*

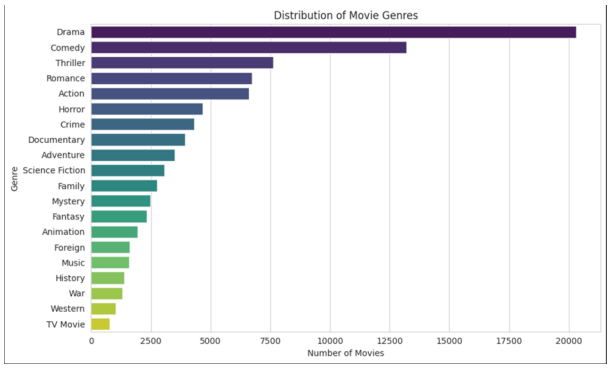

Fig. 1. Distribution of Movie Genres

The bar chart reveals a clear hierarchy in movie genre distribution. Drama dominates, followed by Comedy. Horror, Crime, and Documentary represent a moderate segment. This uneven distribution suggests a potential bias in the dataset.

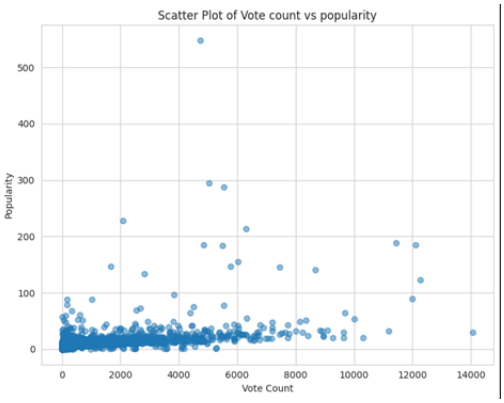

Fig. 2. Distribution Analysis

The scatter plot shows a weak positive correlation between vote count and popularity. While most movies cluster in the lower range (0-2000 votes), some outliers have up to 14,000 votes. The relationship is not strictly

linear, indicating that high vote counts don't always correspond to high popularity.So, it suggests that popularity cannot be solely predicted by vote count. This highlights the need for additional features, such as average rating and release year, to improve predictions.

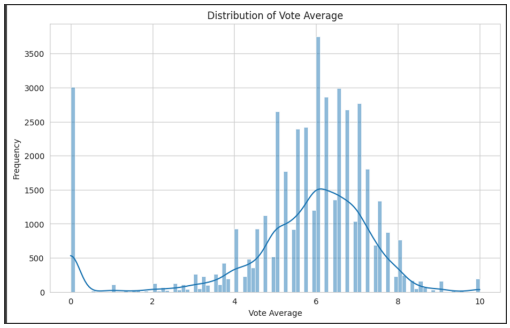

Fig. 3. Distribution of Vote Average

The histogram shows a bimodal distribution of movie ratings. The main peak is around the 6-7 rating range, indicating most movies receive above-average ratings. There is a long tail extending to 10, representing a smaller quantity of exceptionally rated movies. The observed bimodal distribution of ratings (Fig. 3) suggests that a linear model may not capture the full complexity of user preferences, prompting the selection of a non-linear model.

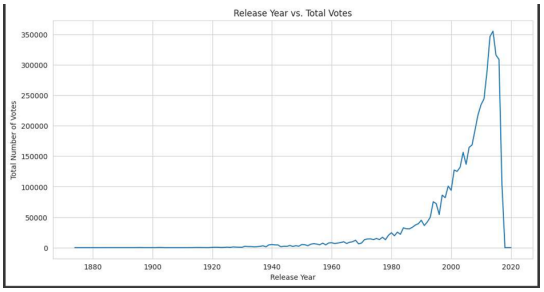

Fig. 4. Release Year vs. Total Votes

The dataset reveals two key patterns in movie ratings over time. It shows minimal activity from 1880-1960, followed by an explosive increase from 1990-2010, peaking at 350,000 votes around 2010-2015, before sharply declining. After the peak, there's a sharp downward trend and it falls to nearly zero by 2020. This pattern likely reflects changes in movie rating platforms, user behavior, and possibly data collection methods rather than actual movie viewership trends.

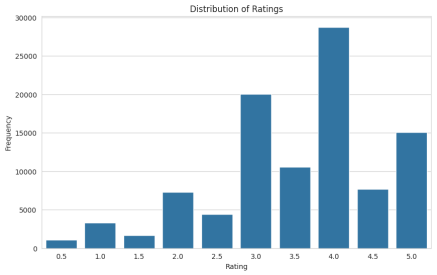

Fig. 5. Distribution of Ratings

The histogram of user ratings reveals the frequency distribution across different rating levels, providing insights into user preferences and potential biases. This information is crucial for selecting appropriate recommendation models. Content-based filtering can leverage this distribution to emphasize highly rated items, while collaborative filtering models, such as SVD and Bayesian approaches, can adjust for rating biases and better predict user preferences. Understanding these patterns ensures that the system can balance popularity and personalization, enhancing overall recommendation accuracy.

Fig. 6. Overview Word Cloud

The most frequently occurring words in movie overviews highlight key themes, plot elements, and genre-specific features central to the narrative. Analyzing these terms helps extract attributes that define each movie, such as tone, setting, and character types. This information is critical for content-based filtering models, which rely on semantic similarities to recommend movies with similar attributes to those a user has previously enjoyed. By incorporating these features, the system can improve recommendation accuracy, particularly for users with limited interaction data, thereby addressing the cold-start problem for new users.

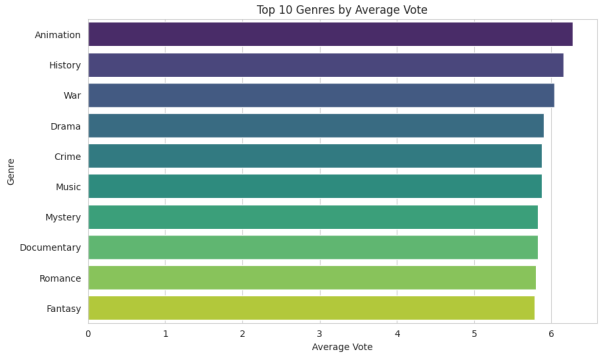

Fig. 7. Top 10 Genres by Average Vote

The skewed distribution of genres indicates a potential bias in the dataset, which we addressed by normalizing the genre representation during feature extraction to ensure more balanced model training. By examining the top 10 genres with the highest average ratings, we can better understand user preferences.

II. Predictive Tasks

The predictive task in this case is to build a movie recommendation system using the MovieLens dataset. We aim to predict the ratings users would give to movies they have not yet rated, a task central to the design of collaborative filtering and content-based recommender systems, which were covered in the course.

We used multiple models, starting with a content-based recommender system, moving to SVD (Singular Value Decomposition), then a Bayesian Parameter Modeling model with regularization, and finally an autoencoder model to optimize the prediction accuracy.

A. Features Used and Data Preprocessing

The dataset contains metadata such as movie titles, genres, and descriptions, along with user ratings. For collaborative filtering approaches, the rating matrix (user-item matrix) was used directly, with appropriate data preprocessing steps:

- **Data Cleaning:** Missing values in the rating matrix were handled by filling them with zeros or using an imputation method.
- **Normalization:** Some methods required normalization of the ratings (scaling ratings to a particular range or centering them around zero) to improve model performance.
- **Splitting the Dataset:** The user-item matrix was split into training and testing sets (80-20 split).

For content-based filtering, we utilize features such as the top three actors, director, genres, production company, and movie plot keywords. This information is combined into a unified into a single string capturing all these elements. This ensures that all relevant features are represented cohesively, allowing for effective analysis of similarities between movies for recommendations.

B. Evaluation Metrics

- **Root Mean Squared Error (RMSE):** Since we are predicting ratings (a continuous variable), RMSE is a good measure of how accurately the model can predict the ratings a user would give to items.
- **Mean Absolute Error (MAE):** This is another metric to assess prediction accuracy, focusing on the average magnitude of the prediction errors.
- **Precision@K:** Measures the proportion of relevant recommendations in the top-K recommended items.
- **Recall@K:** Measures the proportion of relevant items in the top-K recommendations out of all relevant items.

C. Baselines for Comparison

*1) **Content Based Filtering**:* This method uses cosine similarity to identify the most relevant movies by calculating the similarity between unified metadata representations (metadata soup) of each movie.

*2) **Collaborative Filtering**:* This method makes predictions about a user's preferences based on the preferences of other users. It leverages the concept that users who have had similar interactions in the past are likely to have similar preferences in the future. We analyze user-movie interactions (movie ratings) to recommend movies. We have used various models such as SVD, Bayesian Parameter Modeling, and Autoencoders to perform collaborative filtering.

III. Model Description

A. Content Based Filtering

*1) **Model Overview**:* This method suggests items similar to a specific item by analyzing their metadata. For movies, this includes features like genre, director, cast, description, and other relevant features. The key components are:

- **Metadata Soup:** A unified string combining all relevant metadata, such as "jealousy toy boy tomhanks timallen donrickles johnlasseter animation comedy family pixaranimationstudios," which serves as input for vectorization.
- **Vectorization:** Using Python's scikit-learn CountVectorizer, metadata soup is converted into a numerical format. For example, if our corpus

has 3 documents, D_1 = "The movie was fantastic", D_2 = "The plot was predictable", D_3 = "I love the movie". The resulting matrix M of word counts can be written as:

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Where, Rows represent Documents D_1, D_2, D_3
Columns represent words from the Vocabulary V = {"The", "movie", "was", "fantastic", "plot", "predictable", "I", "love"}

To compare the "metadata soup", various similarity metrics like Euclidean, Pearson, and cosine similarity can be employed. For our system, we used cosine similarity due to its computational efficiency and ability to measure similarity independent of vector magnitude, making it ideal for large datasets.

$$\text{sim} = \cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}||\vec{B}|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

*2) **Justification For Model Usage**:* This method can provide highly personalized recommendations by leveraging item-specific attributes. It excels in scenarios where user-item interaction data is sparse, as it focuses on item metadata rather than relying on user behavior patterns. This method avoids the "cold-start" problem for users by recommending items based on their characteristics, making it ideal for applications with rich item metadata, such as movies, books, or products

*3) **Scalability**:* The method is computationally lightweight, especially when using cosine similarity, enabling efficient operation even with extensive datasets. The approach can adapt to diverse datasets and user preferences, providing robust and relevant recommendations.

B. Single Value Decomposition

*1) **Model Overview**:* We implemented a Latent Factor Model using Singular Value Decomposition (SVD) to address recommendation challenges. SVD reduces the dimensionality of the utility matrix by extracting latent factors, and mapping users and items into a shared latent space of dimension r. Hyperparameters, including the number of epochs, learning rate, and regularization, were optimized using GridSearchCV with five-fold cross-validation to mitigate bias and prevent overfitting.

*2) **Justification For Model Usage**:* SVD was chosen for its ability to uncover latent factors that represent hidden relationships between users and items, which enhances recommendation accuracy. Its performance on sparse data makes it a robust choice for recommendation systems.

*3) **Scalability**:* Scalability challenges with a dynamic user base were addressed effectively through the dimensionality reduction offered by SVD. This computational efficiency and the ability to optimize parameters ensured that the model could scale while maintaining accuracy.

C. Bayesian Parameter Model

*1) **Model Overview**:* This is a probabilistic model designed to predict user preferences and item recommendations by incorporating uncertainty and prior knowledge. The key components are:

- **Latent Factors:** The model learns hidden user and item features, representing preferences and characteristics, using a probabilistic approach.
- **Prior Distributions:** It incorporates prior knowledge via prior distributions, which help handle cold-start problems and sparse data.

*2) **Justification For Model Usage**:* Bayesian models are ideal for collaborative filtering tasks because: - They can effectively handle uncertainty in the data, providing probabilistic predictions. - They can incorporate prior information, such as content features, addressing the cold start problem in the SVD model. - Bayesian methods are flexible and can integrate both collaborative filtering and content-based features, making them robust for sparse data scenarios.

*3) **Scalability and Overfitting**:* A challenge with Bayesian models is the computational cost, particularly for large datasets. Additionally, the method naturally includes regularization through priors, but careful tuning is necessary to prevent overfitting, especially with sparse data, otherwise, the model falls into a local minima.

D. Auto Encoder - Optimised Model

*1) **Model Overview**:* In this model, the encoder learns to map the high-dimensional user-item matrix (sparse interactions) to a lower-dimensional latent space. Additionally, the decoder reconstructs the user-item matrix from the latent representation. The goal is for the autoencoder to reconstruct the original ratings as closely as possible, which can then be used to predict ratings for unseen items.

Loss Function: The primary loss function used for training the autoencoder is the Mean Squared Error

(MSE) between the predicted ratings and the actual ratings.

Strengths and Weaknesses

Content-based filtering effectively handles the cold-start problem for new users, as it relies on item attributes rather than prior interactions. However, it struggles with the cold-start problem for new items, as recommendations depend on sufficient metadata. The system often fails to account for user interaction patterns across the entire user base, limiting its ability to identify trends or shared preferences. Additionally, poor metadata can significantly affect the performance of the model.

SVD addresses the lack of user-user interaction by capturing latent factors from user-item interaction data, enabling it to recommend items based on patterns shared across the user base. Additionally, SVD mitigates the problem of poor metadata by uncovering hidden relationships and underlying patterns in the data. However, SVD assumes that the data can be effectively represented in a linear latent space, which may not capture complex, non-linear interactions between users and items.

Bayesian methods can better handle the cold start problem by incorporating side information (e.g., content features, user demographics) to recommend items even with limited interactions. Additionally, they can quantify uncertainty in predictions. However, Bayesian models are more computationally expensive and require more sophisticated techniques (e.g., Markov Chain Monte Carlo) for inference.

Autoencoders can capture more complex, non-linear interactions by adding side inputs, making them more flexible than traditional models. However, autoencoders require significant computational resources for training. Autoencoders are less interpretable due to the complexity of neural networks.

IV. Literature

A. Dataset and Usage

The dataset used in this study is the MovieLens dataset, which contains user ratings for movies and metadata such as genres, release dates, and other attributes. This dataset is widely used in recommendation system research and has been a benchmark for evaluating various recommendation algorithms. In prior studies, the MovieLens dataset has been utilized to evaluate a range of methods, including content-based filtering, collaborative filtering, matrix factorization (such as SVD), and more recently, deep learning models like autoencoders. It is particularly useful for testing algorithms due to its comprehensive and structured nature, making it ideal for building and evaluating models that predict user preferences based on historical ratings.

B. Similar Datasets and Prior Work

Other widely used datasets for recommendation systems include:

- **Netflix Prize Dataset:** A large-scale dataset with over 100 million ratings was used for the Netflix competition, which fostered extensive research in matrix factorization and collaborative filtering techniques.
- **Amazon Product Review Dataset:** This dataset is used for product recommendation tasks, focusing on user reviews and ratings, similar to the MovieLens dataset.
- **Yelp Dataset:** Often used in the context of restaurant or business recommendations, it has been studied using methods such as factorization machines and graph-based models.

C. State-of-the-Art Methods:

The current state-of-the-art methods for recommendation systems include:

- Matrix Factorization: Techniques like SVD and Bayesian Matrix Factorization are widely used to uncover latent factors in user-item interaction matrices, effectively predicting missing ratings.
- Deep Learning Models:
  - **Autoencoders:** Used for learning non-linear representations of user-item interactions, improving accuracy in recommendation tasks.
  - **Neural Collaborative Filtering (NCF):** Combines multi-layer perceptrons with matrix factorization to capture complex patterns.
  - **Hybrid Methods:** Combine content-based and collaborative filtering approaches to tackle cold-start problems and improve recommendation quality.
  - **Reinforcement Learning and Graph-based Methods:** Newer techniques that explore optimal recommendation strategies through user interaction feedback or by capturing user-item relationships in graph structures.

D. Comparison to Our Findings

Our findings align with the existing literature:

- Content-based filtering effectively leveraged movie metadata but struggled with cold-start issues for new users.

- SVD and Bayesian Parameter Model provided more robust recommendations by modeling user-item interaction patterns. The Bayesian approach helped address overfitting through regularization.
- Autoencoders outperformed both content-based and matrix factorization models in our experiments, particularly in capturing complex, non-linear user-item interactions, aligning with recent studies that show deep learning models can excel on large, sparse datasets.

Our results confirm that while simpler models like content-based filtering and SVD are effective, autoencoders offer superior performance, especially when dealing with complex and sparse data.

V. Results and Conclusion

A. *Results*

We have evaluated the performance of collaborative filtering recommender systems using key metrics such as RMSE (Root Mean Square Error), MSE (Mean Square Error), and MAE (Mean Absolute Error). The table below illustrates the progressive improvement achieved by each model, highlighting their increasing accuracy and effectiveness in minimizing prediction errors.

TABLE I
Model Performance Comparison

Model	RMSE	MAE	MSE
SVD	0.872	0.673	0.761
Bayesian Parameter Model	0.852	0.59	0.725
Autoencoder	0.449	0.089	0.201

We evaluated the accuracy of the Autoencoders by calculating additional performance metrics like precision and recall. It demonstrated improved performance with a precision@K and recall@K as follows:

TABLE II
Autoencoder Precision and Recall@K

Metric	Precision@K	Recall@K
K=1	0.4118	0.0063
K=5	0.4941	0.0382
K=10	0.5059	0.0774
Average Precision	**0.4706**	
Average Recall	**0.0406**	

B. *Conclusion*

We explored a range of recommendation techniques, starting with content-based filtering and progressing through collaborative filtering, SVD (Singular Value Decomposition), Bayesian Parameter Model, and finally Autoencoders. Each model was designed to address the limitations of its predecessor, such as metadata dependency, the cold-start problem, and the need to capture non-linear patterns in user-item interactions.

We conducted a comprehensive evaluation of collaborative filtering models using RMSE, MSE, and MAE to assess predictive accuracy. As shown in Table 1, the Autoencoder significantly outperformed traditional approaches like SVD, with a 48% reduction in RMSE. This improvement highlights the advantage of non-linear feature representation in capturing complex user-item interactions, especially in sparse datasets