# NextFlick: Movie Recommendation System

## I. Introduction

In the digital era, personalized recommendations have become an indispensable aspect of the online experience. From Amazon and Google to Netflix and Etsy, major companies are leveraging sophisticated recommender systems to guide users toward content and experiences that align with their preferences. This advancement is fueled by the exponential growth in consumer data collection and accessibility. These algorithms have become revenue generators, accounting for a significant portion of these companies' earnings and user engagement. For instance, Netflix's recommender system drives around 80% of the traffic on their website, generating an estimated $1 billion annually.

Over the years, recommender systems have employed two primary strategies: content-based and interaction-based approaches. Content-based systems create user and item profiles to capture their characteristics and behaviors. For movie recommendations, movies can be modeled by genre, runtime, cast, language, and other attributes, while users can be profiled based on age, demographics, and gender. However, this approach is highly data-intensive, requiring a vast collection of information. Interaction-based approaches, also known as collaborative filtering, rely solely on user-product interaction history to generate recommendations. They suggest products similar to what a user has previously interacted with or products that similar users have interacted with. Consequently, the data requirement is significantly reduced. Recently, hybrid approaches like Factorization Machines and learning-based methods from Natural Language Processing, Deep Learning, and Reinforcement Learning have emerged.

This report presents a comparative analysis of various recommendation systems designed for movie recommendations using the Amazon Movie Reviews Dataset.[1] The dataset comprises 1 million user reviews for approximately 350,000 users and 32,000 movies. We demonstrate how various collaborative filtering techniques, including similarity-based, latent factors, Factorization Machines, and Collaborative Filtering, address previous limitations and enhance recommendation performance. [1]

[1]The dataset can be found at: https://snap.stanford.edu/data/web-Movies.html

## II. Problem Statement

### A. Dataset Statistics

We utilized the Amazon movie review dataset, encompassing approximately 8 million reviews spanning 15 years (1997-2012). Each data record within the Amazon movie review dataset contains pertinent information, including -

TABLE I: Product Review Dataset

| **Column Descriptions** | |
|---|---|
| productId | Unique identifier for the product |
| userId | Unique identifier for the user |
| profileName | Name of the user's profile |
| helpfulness | Fraction of users who found the review helpful |
| score | Rating given by the user |
| time | Timestamp of the review |
| summary | Brief summary of the review |
| text | Full text of the review |

To accommodate computational limitations, our analysis focuses on the initial 1 million reviews, ensuring a manageable and insightful analysis within the specified constraints. Table 1 describes the statistics for the first 1 million rows.

TABLE II: Data Statistics

| **Summary** | |
|---|---|
| Total Users | 352,169 |
| Total Movies | 31,838 |

| **Averages** | |
|---|---|
| Review Length | 168 words |
| Global Rating | 4.08 |

Instances, where two reviews shared identical values for productId, userId, and score, were considered duplicates. We identified 15,580 duplicates and retained only the first occurrence of each duplicate review. Following duplicate removal, we meticulously divided this dataset into three sets: a training set comprising 90% of the reviews, a validation set with 10%, and a testing set also accounting for 10%. This sequential split ensures a balanced distribution for training, validation, and testing, facilitating comprehensive model evaluation and development. We fine-tuned our models to maximize performance on the validation set and report the metrics obtained on the testing set.

## B. Predictive Task

The dataset consists of user reviews, where each review includes a rating provided by a user for a specific movie. Our primary objective is to predict these ratings for novel user-movie pairs. By successfully predicting ratings, we can leverage this information to recommend movies to users. The concept is to suggest movies that we anticipate the user will rate highly based on our predictive model. This process enables us to enhance the user experience by offering personalized movie recommendations.

Our goal is to learn a function $f$ such that the following equation holds

$$f(u, i) = \hat{r}_{u,m} \tag{1}$$

Here $\hat{r}_{u,m}$ represents the predicted rating for user $u$ and movie $m$. We aim to minimize the difference between the predicted value $\hat{r}_{u,m}$ and the true value $r_{u,m}$. These errors are quantified by calculating the mean squared error (MSE), which measures the squared difference between the true and predicted values, serves as the loss metric. Minimizing this loss allows us to enhance the accuracy of our rating predictions..

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(r^i_{u,m} - \hat{r}^i_{u,m})^2 \tag{2}$$

where $N$ is the total number of interactions and $i$ is the specific interaction under consideration.

The loss function value plays a pivotal role in evaluating different models and selecting the most effective approach. Optimal generalization to unseen data, as indicated by the lowest Mean Squared Error (MSE) on the testing set, guides the model selection process.

Despite its widespread adoption, MSE has limitations in evaluating recommender systems. It encounters difficulties in distinguishing between higher-rated and lower-rated items when squared errors are similar. This poses a particular challenge in discerning the influence of predicted ratings on higher-rated items. Additionally, MSE assumes normally distributed errors, which may not always be the case. The dominance of more popular items in the calculation also presents challenges, as a lower MSE does not necessarily equate to a superior recommendation model.

To address these issues, we employ a different metric - the Area under the ROC curve (AUC). AUC is a ranking-based heuristic that evaluates a model's ability to rank positive items effectively. Unlike MSE, AUC is more intuitive and circumvents the pitfalls associated with rating distribution imbalances. The equation for AUC is formulated to quantify the model's effectiveness in ranking purchased or positive items. This shift to AUC enhances our ability to assess and optimize the recommender system's performance, providing a more robust and meaningful evaluation metric.

$$AUC(u) = \frac{1}{|I_u||I - I_u|}\sum_{i\in I_u}\sum_{j\in I-I_u} R_{ij} \tag{3}$$

where,

$$R_{ij} = \delta(Rank_u(i) < Rank_u(j)) \tag{4}$$

and $I_u$ is the set of items the user $u$ has interacted with, $I$ is the set of all items and $\delta$ is the dirac-delta function.

Further, the AUC across all users U is,

$$AUC = \frac{1}{|U|}\sum_{u\in U} AUC(u) \tag{5}$$

where $U$ is the set of all users.

Beyond user ID and movie ID, we also incorporated temporal features, such as the month and year of the review timestamps, after one-hot encoding. We additionally explored the inclusion of features like review text length, summary length, and review helpfulness. However, these features did not yield noticeable performance improvements. Therefore, we decided to exclude them from the final models to preserve computational efficiency.

## III. Exploratory Data Analysis

In this section, we present actionable insights and interesting findings derived from the exploratory data analysis. We exploited these insights to incorporate different features in our models to optimize performance.

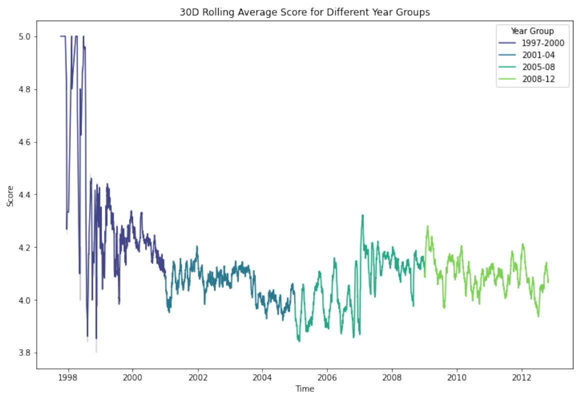

Fig. 1: 30D Rolling Average Score for Different Year Groups

Fig. 1 The plot illustrates the 30-day rolling average score for different year groups (each spanning 4 years) over time. A noticeable trend emerges, indicating a decreasing trend from 1997 to 2004, followed by an increasing trend from 2005 to 2008 characterized by significant fluctuations. The period from 2008 to 2012 exhibits a relatively constant trend with mild fluctuations. It's important to note that the larger fluctuations at the beginning of the timeline may be attributed to a lower number of reviews during that period.

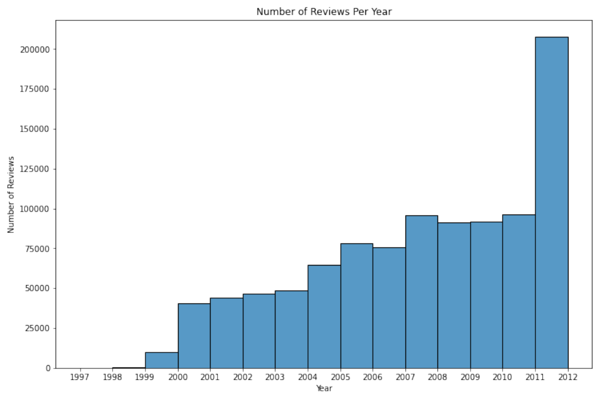

Fig. 2: Histogram of Number of Reviews Per Year

The histogram of the number of reviews per year in Fig. 2 depicts a clear increasing trend over the years. This indicates a rising volume of reviews submitted on the platform, reflecting a growing user engagement or an expanding catalog of movies attracting more user interactions.

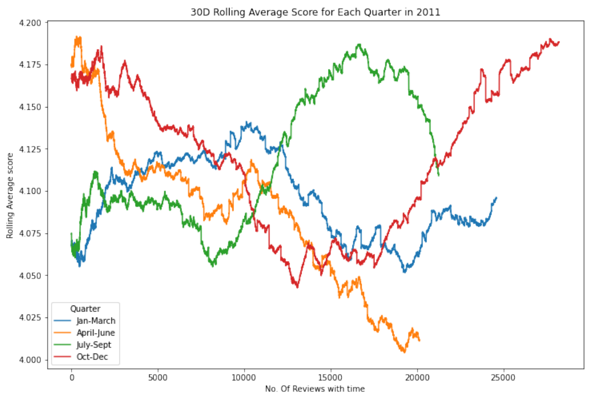

Fig. 3: 30D Rolling Average Score for Each Quarter in 2011

Fig. 3 illustrates the 30-day rolling average score for each quarter of the year 2011. The x-axis represents the sequential number of reviews over time within each quarter, while the y-axis shows the corresponding rolling average score. The rolling average is calculated with a 30-day time window, excluding the first 7 days to allow for a maturity period. The plot reveals distinctive patterns in the 30-day rolling average scores for different quarters of the year 2011. Specifically, for the July-Sept and Oct-Dec quarters, the ratings show an initial decrease but start to increase towards the end of the respective quarters. On the other hand, the other two quarters (Jan-March and April-June) display a contrasting pattern, where ratings begin relatively high and then gradually decrease over time. The temporal insights gained from the various analyses, such as the rolling average scores for different year groups, the quarterly rolling averages, and the distribution of reviews across months, highlight discernible patterns in user behavior over time. These patterns, characterized by fluctuations, peaks, and troughs, suggest a temporal dimension that can be exploited to enhance the performance of the recommendation system.

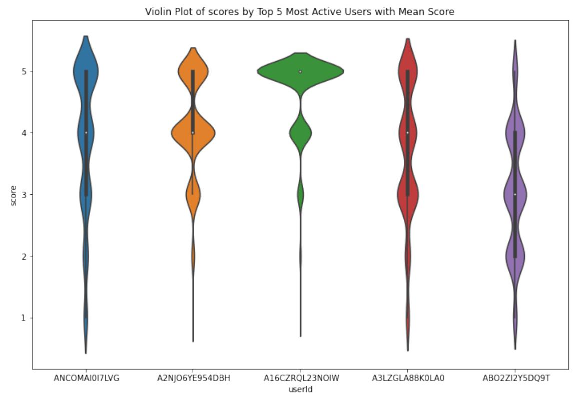

Fig. 4: Violin Plot of scores by Top 5 Most Active

Fig. 4 visualizes the distribution of scores for the top 5 most active users. Each "violin" represents the

TABLE III: Top 5 most active users by number of reviews

| User ID | Review Count |
|---|---|
| A16CZRQL23NOIW | 1307 |
| ANCOMAI0I7LVG | 1271 |
| A3LZGLA88K0LA0 | 1268 |
| A2NJO6YE954DBH | 1102 |
| ABO2ZI2Y5DQ9T | 1037 |

distribution of review scores for a specific user, with the width indicating the density of scores at different values. The observation that the third user tends to give most films a 5-star rating, while the others exhibit a more diverse range of ratings, suggests distinct reviewing behaviors among the top users. This insight emphasizes the importance of considering individual user reviewing patterns. Table III shows the review counts for the top 5 most active users.

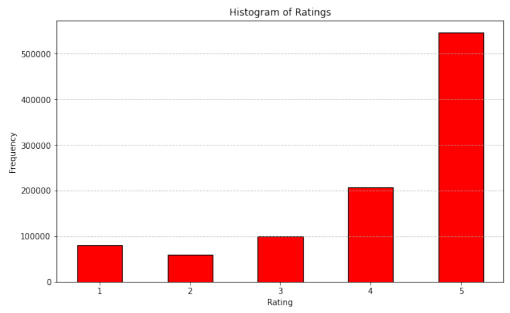

Fig. 5: Histogram of Number of Reviews with rating

The distribution of reviews across different star ratings in Fig. 5 reveals that 5-star ratings dominate, while 2-star ratings are the least prevalent. In general, most movies are highly rated.

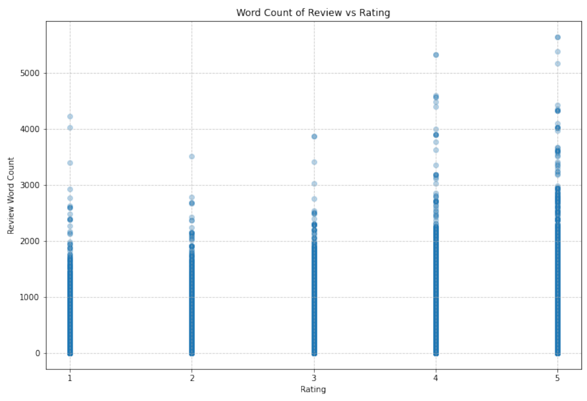

Fig. 6: Word Count of Review vs Rating

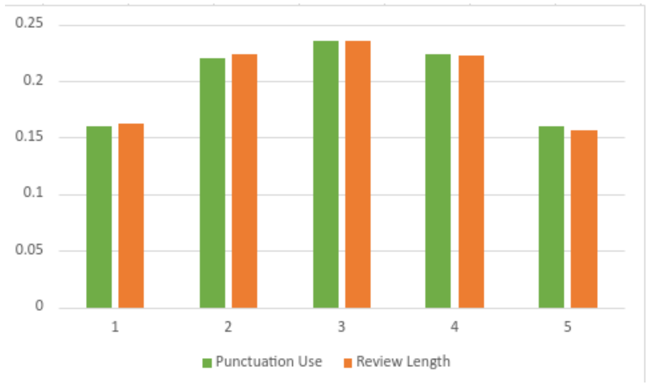

Fig. 7: Variation of punctuation use and review length vs rating

Examining word count via a scatter plot in Fig. 6 unveils an interesting pattern: both 5-star and 1-star reviews tend to be shorter on average compared to reviews with other ratings. This trend persists when analyzing the average punctuation use across various ratings in Fig. 7. These findings collectively highlight distinct characteristics in the lengths and linguistic features of reviews across different star ratings.

An intriguing observation within the dataset reveals that approximately 337k reviews are identified as duplicates. This suggests a notable occurrence where users submit identical reviews for distinct movies. Table IV shows the top 5 users contributing the highest number of duplicate reviews. The table includes both the raw count and the percentage of duplicates for each user. This phenomenon could be attributed to various factors, such as consistent preferences, reviewing habits, or potentially automated processes.

TABLE IV: Users with highest number of duplicate reviews

| User Id | Duplicate Count | Duplicate (%) |
|---|---|---|
| ANCOMAI0I7LVG | 508 | 39.97% |
| A16CZRQL23NOIW | 504 | 38.56% |
| A3LZGLA88K0LA0 | 446 | 35.17% |
| A39CX0EE4BZCZC | 431 | 43.93% |
| A10ODC971MDHV8 | 411 | 42.28% |

## IV. Literature Review

Recommender systems have become especially popular today due to the increase in online shopping on websites such as Amazon, social networking and online entertainment on Facebook and Netflix, etc. Especially since the famous Netflix Prize competition commenced years ago, collaborative filtering (CF) techniques have been very successful in building recommender systems.

In [1], the authors have started out with the concept of latent feature representation to accurately try to predict the item ratings based on the user, and item interactions. The model introduced consists of user and item biases and latent features for both users and items. Taking inspiration from the results and reception of this paper we have introduced a similar model with separate regularizers for the biases and latent features. As in [1], a comparative analysis has been performed on multiple datasets, we rather pick one dataset - 'Amazon Movie Reviews Dataset' and try to improve our predictions.

Many CF-based approaches are built on matrix factorization [2], in which the user, item interaction matrix is broken down into two low-rank matrices corresponding to latent features of users and items. The inner product between the two user and item features is then used to estimate the rating which a user might give to an item.

In recent times, dense neural networks have received immense appreciation and success in domains like computer vision and natural language processing. However, deep neural networks have not been elaborately explored in depth and range in the domain of recommendation systems. In [4], the authors have replaced the dot product of latent features with a dense architecture which learns an arbitrary function from the interactions. They have presented a framework named Neural Collaborative Filtering (NCF). NCF is a general approach and expresses and generalizes the concept of matrix factorization in its framework. To deal with and model non-linearities, a multi-layer dense neural net is leveraged to learn the user-item interactions.

In [6], the authors have discussed the recent setting in recommendation systems where, more users, items and rating data are consistently being added to the model, causing many variations and shifts in the underlying relation between users and items to be rated/recommended. This problem is known as 'concept drift' or 'temporal dynamics in recommendation systems'. CF also includes methods like neighborhood models which utilize the collaboration of the ratings by similar users to come up with recommendations [5]. In general, a neighborhood model can adapt in order to model dynamic changes in Recommendation Systems over a period of time, by integrating either time-dependent algorithms [7], [8] or by making use of time-independent algorithms [9], [10], [11]. In [6], the authors have presented several studies on time-dependent and time-independent algorithms for neighborhood-based RS. They have used popular methods/techniques like K Nearest Neighbours(KNN) and Long Short-Term Memory(LSTM) models. Taking inspiration from this, we have also managed to incorporate 'Timestamp' as a feature and were able to outperform all the other models on the basis of MSE and AUC scores.

## V. Model Exploration

In this section, we provide a comprehensive overview of the models and methodologies employed in our recommendation system. We delve into the specifics of the experimental setup, feature engineering and hyperparameter tuning, and to optimize model performance.

### A. Always Mean Predictor (AMP)

The Always Mean Predictor (AMP) serves as our baseline model. This simplistic predictor adopts a straightforward strategy of consistently forecasting the mean rating, denoted as $\mu$, for any user and movie pair. The lack of adaptability to individual user preferences or specific movie characteristics limits its performance, making it a rudimentary predictor in our model hierarchy. We calculate the $\mu$ from the training data as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} r^{i}_{u,m} \tag{6}$$

### B. Collaborative Filtering (CF)

Collaborative Filtering stands out as an essential approach for capturing user preferences in our recommendation system. This method leverages both item similarity and user similarity to make predictions. We employed the Item-Based Collaborative Filtering to model recommendations. The item-based collaborative filtering approach predicts outcomes by evaluating the similarity of the current item with all other items that a user has previously interacted with. This method recommends movies or predicts higher ratings for movies that share similarities with those a user has already consumed. In our implementation of item-based collaborative filtering, we employ a model that predicts ratings using the following equation:

$$\hat{r}_{u,i} = \overline{R}_i + \frac{\sum_{j\in I_u\setminus\{j\}}(r_{u,j} - \overline{R}_j)\cdot Sim(i,j)}{\sum_{j\in I_u\setminus\{j\}} Sim(i,j)} \tag{7}$$

where $I_u$ is the set of movies the user $u$ has already rated, $\overline{R}_i$ is the average rating for the movie $i$ and $Sim(i,j)$ is the measure of similarity between movie $i$ and movie $j$. We used the Jaccard similarity, defined as

$$Sim(i,j) = \frac{U_i \cap U_j}{U_I \cup U_j} \tag{8}$$

where $U_i$ and $U_j$ are the set of users who have interacted with movie $i$ and $j$ respectively. We constructed a sparse interaction matrix by retaining only the movies that each user has rated and the users who have rated each movie. This approach enabled us to efficiently store all interactions in memory. Subsequently, we applied the heuristics outlined in Equation 7 to make predictions based on the similarity between items and users.

## C. Latent Factor Models (LF)

Latent factor models enhanced recommendation systems by utilizing supervised learning to model both users and items in a latent space. This approach captured user behavior and movie traits in a compact, hidden, and low-dimensional latent space. The inspiration for this method came from recognizing that singular value decomposition (SVD) of the interaction matrix could yield low-dimensional representations of users and movies. Due to the computational challenges associated with performing SVD on large, sparse, and partially observed matrices, latent factor models employed supervised learning to learn these low-dimensional representations. The models aimed to predict higher ratings for users and movies exhibiting similarity in the latent space. The latent factor models can be described as

$$\hat{r}_{u,i} = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i \tag{9}$$

where $\alpha$ is a constant, $\beta_u$ and $\beta_i$ are the bias terms associated with the user $u$ and item $i$, $\gamma_u$ and $\gamma_i$ are $k$-dimensional vectors representing the user $u$ and item $i$ in the smaller $k$ latent dimension. The dot product between $\gamma_u$ and $\gamma_i$ captures the similarity among them. These parameters are learned by minimizing the regularized mean squared error, defined as follows:

$$loss = \sum_{u,i} (\alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i - r_{u,i})^2 +$$
$$\lambda \left[ \sum_u \beta_u^2 + \sum_i \beta_i^2 + \sum_u ||\gamma_u||_2^2 + \sum_i ||\gamma_i||_2^2 \right]$$

We computed the gradients for each parameter and employed gradient descent using the scipy library to minimize the loss. Tuning the hyperparameters involved careful experimentation, and the values were selected based on minimizing the error on the validation set. This approach helped prevent overfitting, as the performance on the validation set is indicative of the model's generalization capability. For our experiments we defined two latent factor models

*1) LF Bias only (LFB):* We create a latent factor model with only the bias terms for users and movies.

$$\hat{r}_{u,i} = \alpha + \beta_u + \beta_i \tag{10}$$

The regularization parameter $\lambda$ was set to $10^{-5}$. The latent dimension size, $K = 5$, yielded optimal results on the validation set.

*2) LF standard (LFS):* We also train a standard latent factor with all its terms. During the training process, it became evident that the bias terms and the latent factor terms in the complete latent factor model had different scales. To address this, we introduced independent regularization terms for both bias and latent factor terms in the loss function. The modified loss function is defined as:

$$loss = \sum_{u,i} (\alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i - r_{u,i})^2 +$$
$$\lambda_1 \left[ \sum_u \beta_u^2 + \sum_i \beta_i^2 \right] + \lambda_2 \left[ \sum_u ||\gamma_u||_2^2 + \sum_i ||\gamma_i||_2^2 \right]$$

The regularization parameters $\lambda_1$ and $\lambda_2$ were chosen as $10^{-5}$ and $5 \times 10^{-5}$, respectively. The latent dimension size, $K = 5$, yielded optimal results on the validation set.

## D. Factorization Machines (FM)

Factorization Machines (FMs) are versatile techniques that extend latent factor models to incorporate features beyond user-item interactions. They enable modeling interactions between any pair of features by associating each feature with a low-dimensional latent vector. The interactions between features are then represented by the dot product of these latent vectors, weighted by their dot product in the latent space. The model can be described by the following equation:

$$\hat{r}_{u,i} = w_0 + \sum_{i=1}^{F} w_i x_i + \sum_{i=1}^{F} \sum_{j=i+1}^{F} < \gamma_i, \gamma_j > x_i x_j \tag{11}$$

where $x_i$ and $x_j$ are arbitrary features, $w_i$'s are the weights associated with each features, $\gamma_i$ and $\gamma_j$ are the low dimensional representation of features $x_i$ and $x_j$, and $F$ is the total number of features.

Factorization Machines provide a framework for capturing second-order interactions among various features, representing a more expansive model class and a generalization of latent factor models. When considering only user interactions, we created two features: $x_1$, a one-hot encoded vector representing users, and $x_2$, a one-hot encoded vector representing items. Additionally, we trained an FM model that incorporated timestamps as an extra feature. This was achieved by one-hot encoding the month and year information. If an interaction occurred in a specific month and year, the corresponding bit in the one-hot binary vector was activated. The relevance of temporal features, was identified during the Exploratory Data Analysis which suggested that leveraging temporal information could enhance the model's performance, thereby influencing the recommendations made by the system.

In the data preparation process for FM, feature vectors were created. For each unique user (totaling $352,169$), a one-hot encoded vector of size $352,169$ was generated, with all values set to $0$ except for the current user, which was set to $1$. Similarly, a one-hot encoded feature was created for each movie, with a dimension of $31,842$, representing the total number of unique movies in the dataset. The timestamp of each interaction was binned into one-month spans, resulting in $180$ bins covering the time range from October 1997 to October 2012. The one-hot encoded vector for the timestamp was then created by setting the corresponding month index to $1$. This vector, along with the user and movie vectors, was employed to train FM. The optimal validation performance was achieved with $K = 100$ as the latent dimension size, and regularization constants set to $10^{-2}$ and $5 \times 10^{-4}$ for the weights and latent factors, respectively.

## VI. Results and Conclusion

As detailed in the preceding section, a variety of models, including Always Mean Predictor, Collaborative Filtering, Latent Factor Models, and Factorization Machines, have been developed to predict user ratings for items. These models are evaluated based on metrics such as Mean Squared Error (MSE) and Area Under Curve (AUC).

Table V presented a comparative analysis of the performance of each model on the test set. Notably, FM, which incorporated an additional feature 'timestamp,' outperformed all other models in terms of both the MSE and the AUC. The results revealed a non-linear relationship between users and movies, as models leveraging an extra feature to capture such interactions, like FM, demonstrated superior performance. In contrast, similarity-based models such as CF, LFB, and LFS were too simplistic to capture these intricate interactions.

TABLE V: Comparison of different model performance

| **Model** | **MSE** | **AUC** |
|---|---|---|
| $AMP$ | 1.597 | 0.682 |
| $CF$ | 1.518 | 0.695 |
| $LFB$ | 1.440 | 0.703 |
| $LFS$ | 1.427 | 0.714 |
| $FM$ | 1.104 | 0.742 |

Despite improving upon the baseline set by AMP, latent factor models faced the "Cold Start" problem, resulting in poor predictions for users or items with very few interactions. Additionally, these models lacked interpretability and failed to account for temporal evolution in the data. Collaborative filtering methods shared similar drawbacks.

The findings emphasized the importance of efficiently modeling user interactions for building robust recommendation systems. Matrix Factorization, coupled with naive feature engineering, outperformed similarity-based methods.

## References

[1] McAuley, Julian John, and Jure Leskovec. "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews." In Proceedings of the 22nd international conference on World Wide Web, pp. 897-908. 2013.

[2] Yehuda Koren, Robert Bell, Chris Volinsky, and others. 2009. Matrix factorization techniques for recommender systems. Computer 42, 8 (2009),30–37.

[3] Y. Shi, "An improved collaborative filtering recommendation method based on timestamp," 16th International Conference on Advanced Communication Technology, 2014, pp. 784-788, doi: 10.1109/ICACT.2014.6779069

[4] He, Xiangnan, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural collaborative filtering." In Proceedings of the 26th international conference on world wide web, pp. 173-182. 2017.

[5] Liu, X.; Aberer, K. Towards a dynamic top-N recommendation framework. In Proceedings of the 8th ACM Conference on Web Science—WebSci '16, Hannover, Germany, 22–25 May 2016; pp. 217–224.

[6] Li, L.; Zheng, L.; Yang, F.; Li, T. Modeling and broadening temporal user interest in personalized news recommendation. Expert Syst. Appl. 2014, 41, 3168–3177.

[7] Wang, Y.; Yuan, Y.; Ma, Y.; Wang, G. Time-Dependent Graphs: Definitions, Applications, and Algorithms. Data Sci. Eng. 2019, 4, 352–366.

[8] 121. Park, Y.; Park, S.; Jung, W.; Lee, S.-G. Reversed CF: A fast collaborative filtering algorithm using a k-nearest neighbor graph. Expert Syst. Appl. 2015, 42, 4022–4028.

[9] K. Shah, A. Salunke, S. Dongare and K. Antala, "Recommender systems: An overview of different approaches to recommendations," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 2017, pp. 1-4, doi: 10.1109/ICIIECS.2017.8276172.

[10] Moghaddam, S.; Jamali, M.; Ester, M. ETF: Extended tensor factorization model for personalizing prediction of review helpfulness. In Proceedings of the WSDM 2012–the 5th ACM International Conference on Web Search and Data Mining, Seattle, WA, USA, 8–12 February 2012.

[11] Shi, Y.; Karatzoglou, A.; Baltrunas, L.; Larson, M.; Hanjalic, A.; Oliver, N. TFMAP: Optimizing MAP for top-n context-aware recommendation. In Proceedings of the SIGIR '12: The 35th International ACM SIGIR conference on research and development in Information Retrieval, Portland, OR, USA, 12–16 August 2012.