Leveraging Machine Learning Algorithms for User-Centric Movie Recommendations

Afonso Simão, Mikael Johnatan and Pedro Figueira

***Abstract*— The rapid growth of digital streaming platforms has increased the need for advanced recommendation systems capable of providing personalized content suggestions. This paper develops and compares seven machine learning algorithms for predicting user preferences in a movie recommendation system, using the MovieLens 20M dataset. Models evaluated include Singular Value Decomposition (SVD), SVD++, Non-Negative Matrix Factorization (NMF), Slope One, and Co-Clustering. The study compares their performance based on Root Mean Squared Error (RMSE) and computational efficiency. Surprisingly, the Slope One algorithm delivered the best balance of performance, achieving an RMSE of 0.855 with minimal computational complexity. While SVD and SVD++ provided strong results, they required significantly more computational power, making them less practical for large-scale, real-time applications. Slope One's simplicity and efficiency make it a compelling choice for real-world streaming platforms. The study's findings suggest that simpler algorithms can outperform complex methods in practical environments.**

I. INTRODUCTION

In recent years, the exponential growth of digital streaming platforms, such as Netflix, has revolutionized the way people consume media. With vast catalogs of movies and TV series available at the click of a button, the challenge of providing users with relevant and personalized content has become increasingly complex. As a result, recommendation systems have become essential components of these platforms, aiming to deliver suggestions tailored to individual user preferences.

One of the key strategies in building effective recommendation systems is recommendation prediction models, a machine learning technique that organizes users based on shared characteristics. By applying these algorithms, we can identify patterns in user behavior and preferences, enabling the creation of personalized movie and series recommendations that enhance the viewing experience. However, given the variety of such methods available, it is crucial to understand which techniques perform best in different contexts, particularly in predicting user preferences in the entertainment domain.

This work focuses on developing a movie recommendation algorithm based on scores, aiming to suggest titles to users based on their previous ratings and liking's. The goal of this system is to identify which movies or series a user is likely to enjoy, assigning a score to each title based on their behavior and preferences.

By comparing the performance of these algorithms in the context of personalized recommendations, this research seeks to provide insights into the most effective strategies for improving user experience on streaming platforms. In this scenario, recommendation systems have become critical components, aiming to deliver suggestions that align with individual user preferences.

This paper focuses on analyzing, testing, and comparing various predictions algorithms, such as Matrix Factorization algorithms (SVD, SVD++, NMF), the Slope One algorithm and Co-Clustering, aiming to assess their effectiveness in identifying user preferences for movies and series. These methods will be applied to a Netflix-like application that simulates the recommendation systems used by major streaming services. The objective is to develop an efficient and accurate recommendation engine capable of better predicting user interests, enhancing their overall experience.

By comparing the performance of the models in the context of personalized recommendations, this research aims to provide insights into the most suitable methods for increasing user satisfaction on streaming platforms. The findings of this study could contribute to optimizing content delivery and production, as well as significantly improving the user experience in modern entertainment environments. Therefore, the development of an efficient recommendation engine is essential to elevate the level of user engagement.

The paper is organized as follows: Section II presents the chosen dataset's information. Section III will presents the methodology used. Section IV presents the model's results. Finally, Section V discusses the conclusions and future work.

II. ABOUT THE DATASET

Data selection is a crucial aspect for any analysis work using predictive algorithms, such as machine learning models. So a large and diverse dataset was chosen, the MovieLens 20M Dataset [1], it contains 20 million ratings (ranging from 0.5 to 5) originated by 130 thousand users and 27 thousand films. The extensiveness of the dataset not only provides a significant number of interactions between users and films, but it also spans a wide variety of titles, from classic to recent releases.

Such magnitude and diversity enables the algorithm to learn from different taste profiles and preferences, enriching the accuracy of personalized recommendations and expanding its ability to cater to a broad audience.

Moreover, retaining to the structure of the data set, it is composed of 2 files, `movies.csv` and `ratings.csv` offering a solid foundation for analysis. The `ratings.csv` is where the main data was derived, containing 20 million entries for columns for the id for the current user, id for the film rated, a score from 0.5 to 5 for the film and the movie's timestamp. The timestamp column was removed as it related to the date and time of the score submission and its removal

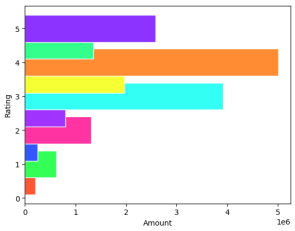

Fig. 1. Amount of each rating

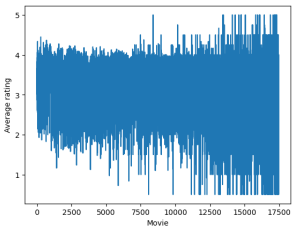

Fig. 2. Average rating per movie

would simplify the dataset, making it more streamlined and focusing on more relevant attributes.

Therefore, the models were trained using the `ratings.csv` file and the results come as film ids, which are then cross-referenced to get their names in the `movies.csv` file, which contains the movie's id and name.

III. Methodology

A total of 7 different models, both unsupervised and semi-supervised, were trained and tested under models built within a Python SciKit (add-on package for the SciPy library), the Surprise toolkit [3]. It provides easy-to-use algorithms for building and analyzing recommender systems that deal with explicit rating data.

All the models were created by following the same code pipeline. Firstly the algorithm is computed for various accuracy metrics on various combinations of parameters, over a cross-validation procedure (with 5-fold segments, 80% training and 20% testing), making it easy to find the best set of parameters for the model.

The K-Nearest Neighbors (*KNN*) based algorithms, while conceptually simple and easy to implement, suffer from significant scalability issues, especially when applied to large datasets. The main drawback of *KNN* algorithms is that they require the entire dataset to be stored in memory, as they rely on computing the distance between each pair of users or items to find the nearest neighbors. This means that the computational cost increases dramatically as the size of the dataset grows.

Moreover, *KNN* based algorithms must be rerun each time a new user or item is introduced, as the distances between all users or items must be recalculated. This necessity to recompute distances with every update results in further computational overhead, making *KNN* impractical for dynamic or large-scale systems where data is constantly changing or expanding.

In contrast to models like *SVD* or *SVD++*, which learn a fixed set of parameters and can make predictions based on these parameters without needing to store or recompute the full dataset, *KNN* does not benefit from such efficiencies. As a result, while *KNN*-based algorithms can perform well on smaller datasets, their usage becomes prohibitive in real-world applications that require frequent updates or deal with vast amounts of data such as streaming-services recommendation systems.

Furthermore, the more complex models (*SVD++*, *NMF* and *Co-Clustering*), with a bigger need for computational power, were composed of a random 10% segment of the whole dataset (200 thousand entries).

Thus the following prediction models from the Surprise toolkit were used:

A. Basic Algorithms

1) Normal Predictor: Is an algorithm that predicts a random rating based on the distribution of the training set, assumed to have a normal distribution. This prediction is generated using the Maximum Likelihood Estimation over the normal distribution.

For this model there were no hyperparameters to tune.

2) Baseline Only Predictor: Algorithm predicting the baseline estimate for any given user and item.

The hyperparameters changed were a set of options for baseline estimates computation; 2 methods for the baseline calculation were used, the Alternating Least Squares (*ALS*) method and Stochastic Gradient Descendent (*SGD*) method, with each requiring a different set of hyperparameters to tune, with both changing the regularization values, helping to avoid overfitting by ensuring the user bias terms do not become too extreme, the higher regularization values (e.g., 15) make the prediction mode conservative, while lower values (e.g., 5) might fit the data more closely, although there is a higher risk of overfitting.

The Stochastic Gradient Descendent method was tested under various regularization values (0.001, 0.002, 0.003, 0.004, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05 and 0.1) and different learning rates (0.0001, 0.0005, 0.001, 0.002, 0.003, 0.004 and 0.005).

The Alternating Least Squares method was also tested under various regularization values, for users (3, 4, 5, 7, 10 and 15) and for movies (1, 2, 3, 4, 5, 10 and 15), and epochs (5, 10, 15, 20, 25 and 30).

B. Matrix Factorization-based Algorithms

1) Singular Value Decomposition: The Singular Value Decomposition algorithm (*SVD*), popularized by Simon Funk during the Netflix Prize competition, when no baseline is used, is equivalent to Probabilistic Matrix Factorization method; and for unknown predictions, the minimized regularized squared error is used.

The hyperparameters modified were the number of epochs (15, 20, 22, 25, 30, 35 and 40), the learning rate for all parameters (0.001, 0.002, 0.003, 0.004 and 0.005) and the regularization term for all parameters (0.05, 0.1, 0.15, 0.2,

0.3, 0.4 and 0.6). The smaller range for the hyperparameters values is due to the high computational power to run these models on large datasets.

2) Single Value Decomposition Plus Plus: The Single Value Decomposition Plus Plus algorithm (*SVD++*) is an extension of the *SVD* algorithm taking into account implicit ratings. The model takes into account the fact that an user *u* rated an item *i*, regardless of the rated value. Keep in mind this model was trained under 10% of the total dataset size.

The *SVD++* model may initially seem superior due to its ability to incorporate implicit feedback, which allows it to account for items that a user interacted with, even without providing explicit ratings. This feature can enhance the accuracy of predictions, particularly in cases where users have limited explicit feedback but substantial implicit behavior data (e.g., clicks or views).

However, the *SVD++* model comes with a significant drawback: its computational complexity. Because the algorithm processes both explicit and implicit feedback, the additional data points and interactions require far more resources. As a result, *SVD++* can have compute times that are 2x to 10x slower than the regular *SVD* algorithm. This substantial increase in computational cost makes *SVD++* less practical for large-scale datasets or time-sensitive applications unless sufficient computational power is available to offset these limitations.

The hyperparameters changed were the same as the *SVD* algorithm, the number of epochs (15, 20, 25, 30, 32, 35, 40, 45, 50, 55, 60, 65, 70 and 75), the learning rate for all parameters (0.001, 0.002, 0.003, 0.004, 0.005, 0.006 and 0.007) and the regularization term for all parameters (0.1, 0.2, 0.4 and 0.6).

3) Non-Negative Matrix Factorization: The Non-Negative Matrix Factorization (*NMF*) is a filtering algorithm very similar to the *SVD* algorithm. The optimization procedure for the model is a (regularized) stochastic gradient descendent with a specific choice of step size that will ensure non-negativity of the factors for the formula, provided the initial values are also positive. Furthermore, the algorithm is very highly dependent on initial values.

The hyperparameters modified were the number of epochs (10, 15, 20, 22, 25, 30 and 35), the learning rates for both user and item biases (0.001, 0.002, 0.003, 0.004 and 0.005), and the regularization terms for users bias (0.1, 0.2, 0.3, 0.4, 0.5 and 0.6).

C. Slope One Algorithm

The Slope One Algorithm is composed by 3 related slope one schemes composed by predictors of the form $f(x) = x + b$, which precompute the average difference between the ratings of one item and another for users who rated both. The algorithm is known for being simple to implement, while still maintaining a good accuracy. By separating the factoring of the items, by items the user liked and item the user disliked, the algorithm is competitive with slower, memory-based schemes; such as K-Nearest-Neighbors (*KNN*) based algorithms.

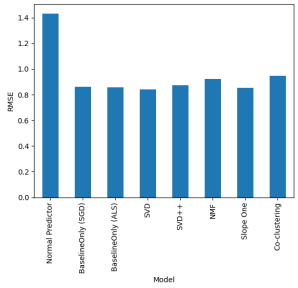

Fig. 3. Model Results by RMSE

The were no changed hyperparameters.

D. Co-Clustering Algorithm

The Co-clustering Algorithm is a weighted simultaneous clustering of the users and items, the resulting prediction is built on top of a items and users cluster that is obtained by the average rating on the user-only cluster and item-only cluster. Moreover, a accuracy comparable to that of the correlation and matrix factorization based approaches at a much lower computational cost.

The hyperparameters that were changed were the number of both the users and items clusters (1, 2, 3, 4, 5, 10, 20, 25 and 30) and the number of epochs (5, 10, 15, 20, 30, 35 and 40).

IV. Results

The performance of each algorithm was evaluated based on their ability to predict user ratings accurately. The following metrics were used to compare the performance of the models:

- Root Mean Squared Error (*RMSE*): A widely used metric for evaluating recommendation algorithms, RMSE measures the average magnitude of error between predicted and actual ratings. Lower values indicate better performance.
- Computational Efficiency: Measured by the time taken to train and test each model, this was a critical factor in determining the practicality of algorithms for large-scale applications like streaming services.

A. Normal Predictor

The Normal Predictor model, as expected, had a relatively poor compared to more advanced methods, with a *RMSE* of 1.433. The model's lack of sophistication makes it impractical for real-world recommendation tasks, but it provides a useful reference point for understanding the effectiveness of more complex algorithms.

B. Baseline Only

The Baseline Only model performed significantly better than the Normal Predictor, although much slower (2x slower). The *RMSE* of the best model was 0.857, with usage of the Alternating Least Squares (*ALS*) method, regularization for the items was 3 and for the users was 5, under 20 epochs. For the Stochastic Gradient Descendent model,

the best model achieved had a RMSE of 0.8615, with a regularization value of 0.001 and learning rate of 0.002.

C. Singular Value Decomposition

The *SVD* model performed closely to the Baseline Only model, with a best model *RMSE* being 0.842, achieving better results due to the increase in complexity of the model. The model was highly sensitive to the change in hyperparameters, the best model was obtained with 35 epochs, learning rate for all parameters of 0.003 and regularization term for all parameters of 0.1.

D. Singular Value Decomposition Plus Plus

The *SVD++* model performed just slightly worse than the *SVD* model, which was maybe caused by the need to reduce the dataset size to 10% of it's original size. Also, the model has a significantly higher need for computational power, making it completely ineffectual for real-world big data cases; it ran for 1.5x the time of the *SVD* model. The best *RMSE* found for the model was 0.875, with the hyperparameters being 60 epochs, learning rate of all parameters of 0.007, regularization term for users bias of 0.1.

E. Non-Negative Matrix Factorization

The *NMF* model performed worse than expected, with a final best result of *RMSE* of 0.923, with the following hyperparameters for the model constructions: 10 epochs, learning rate for users bias of 0.002, learning rate for items bias of 0.002, regularization term for users bias of 0.1. The non-negativity constraints made it slower to converge compared to *SVD* model.

F. Slope One

The Slope One model, known for its simplicity, delivered surprisingly competitive results, the best found from all models. The *RMSE* of the best model was 0.855, and no hyperparameters were tuned for the model. Computational efficiency was also very good, with the models running faster than the more complex models (*SVD*, *SVD++* and *NMF*).

G. Co-Clustering

The best found model's *RMSE* was of 0.948, with the following hyperparameters: 1 user cluster, 2 items clusters and 10 epochs. But at the cost of the high computational power needed due to it making a cluster for items and users, making the model the slowest to train, taking about 3x the time of the *SVD* model. Also, it is notable that lower cluster sizes (1-2) were preferred by the model as well as lower epochs were needed to train (10), which may make the model viable in cases such as recommendation systems.

V. Conclusions

In this study, several machine learning algorithms were evaluated for their ability to predict user preferences in a movie recommendation system using the MovieLens 20M dataset. The primary goal was to identify which models perform best in terms of accuracy and computational efficiency, particularly in the context of large-scale applications like streaming platforms. Although matrix factorization algorithms like *SVD* and *SVD++* were initially expected to deliver the best results, a surprising outcome was observed: the Slope One algorithm emerged as the most balanced and effective model for this specific dataset.

Slope One, despite its simplicity, performed exceptionally well, offering competitive accuracy (*RMSE*: 0.855) and excellent computational efficiency. Its straightforward nature, requiring fewer computational resources and no complex hyperparameter tuning, makes it a practical choice for real-world applications. This simplicity combined with solid performance makes the model an attractive option for building efficient recommendation systems on large-scale platforms where both speed and accuracy are critical.

Although other models may have provided better *RMSE* results, the high computational complexity and longer training times make them less ideal for environments where quick updates, scalability and handling of enormous amounts of data are required.

In conclusion, while advanced matrix factorization techniques may offer higher accuracy, the Slope One algorithm strikes an optimal balance between performance, computational simplicity and training time making it the most suitable model for practical implementations of recommendation systems in the entertainment industry. Its ability to scale efficiently while maintaining solid prediction accuracy positions it as a viable solution for improving user satisfaction on streaming platforms. Future work could focus on enhancing the Slope One algorithm with additional features, such as incorporating implicit feedback, to further improve its performance in real-world streamlined applications.

References

[1] "GroupLens." https://grouplens.org/datasets/movielens/20m. Accessed on 1 October 2024.

[2] "ScikitLearn." https://scikit-learn.org. Accessed on 1 October 2024.

[3] "Surprise Documentation." https://surprise.readthedocs.io/en/stable. Accessed on 1 October 2024.

[4] L. Danniel and M. Anna. Slope One Predictors for Online Rating-Based Collaborative Filtering. In: Proceedings of the 2005 SIAM International Conference on Data Mining. SIAM, pp. 471–475.

[5] S. Merugu and T. George. A Scalable Collaborative Filtering Framework Based on Co-Clustering. In: Proceedings. Fifth IEEE International Conference on Data Mining, Houston, TX, 2005 pp. 625-628.

[6] X. Luo, M. Zhou, Y. Xia and Q. Zhu. An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems. In: IEEE Transactions on Industrial Informatics, 2014, vol. 10, no. 2, pp. 1273-1284.

[7] S. Ruslan and M. Andriy. Learning from Incomplete Ratings Using Non-negative Matrix Factorization. In: Proceedings of the Sixth SIAM International Conference on Data Mining, 2006.