Movie Recommendation System

Balaji Sundar Anand Babu
Computer Science
Northeastern University
Boston, United States
anandbabu.b@northeastern.edu

***Abstract*—For people who love watching movies and posting ratings on the Internet, movie recommendation is the need of the hour. It is also vital that these ratings play a critical role in a viewer's movie-watching experience.Recent works have fallen prey to restricting their predictions, relying solely on the metadata given in the dataset. This primary failure of such works is the root of the problem statement, and the primary motivation of this proposal is to address this problem.There were also works that took other factors into consideration and proceeded further with the prediction, but even these failed to address or consider the movie-watching experience. This resulted in movies being suggested as per their category, irrespective of the user's choice and without much consideration given to the user's wish. These works also failed to be genre-based; rather, they focused somewhat on the timeline and used text prediction for a content-based analysis. The need for this recommender is to enhance the user's experience and, collectively, suggest quality movies. Yes, the user may not want to watch only quality movies, but it is mandatory to customize the movie-watching experience for the user and make it viable enough for them to have a good watch.This proposal lays theS the foundation for further works that can be carried out, taking various factors into consideration to enhance the watcher's experience.**

I. Introduction

The core need of the movie-watching experience is to give the user a wholesome experience, satisfying the entertainment factor and also keeping all the values checked. This is contributed to by the cast, crew, and also the genre the user specifies to watch. Though a movie which is exceptional in a particular genre will still be at stake if an individual user chooses not to sit through that genre. According to the user, it is still a bad movie and it still does not offer the content the user wishes to watch. So, this becomes a task for any particular recommender to predict and suggest movies to the user.

Just as mentioned in [1], it is mandatory to take the user's previous watch history, consider the amount of movies watched per user in a year, and take all of the other factors that influenced the user into consideration. This might or might not work, but it would still give the recommender enough feature space to play around with to make a better prediction. The recommender's primary motive has to be to predict what rating the user will give to one particular movie, taking external factors and also the film's inbuilt quality into consideration. This makes the recommendation more specific about the movies it recommends. For streaming platforms like Netflix and Prime, if they lose out on suggesting proper movies to the user at any given point, or if they end up suggesting a bad movie, the user's overall movie-watching experience or engagement factor with the application will reduce. They might end up not liking the application on the whole. Maybe not with one bad suggestion, but collectively if the application fails, then the engagement factor will eventually reduce. In order to enhance this, we need to consider a set of features along with the user's previous interaction to arrive at a proper recommendation. There was also a [2] journal which addressed the voluminous content out there on Netflix and how it has challenged society with some poor recommendations at times, which influence the user, change their view, and make their life a bit stressful. So, as mentioned, it can also become a stressful environment if the recommendation is incorrect.

The dataset which we have used is from the MovieLens review system, which is likely the best dataset that could be used for this recommendation problem as it deals with a variety of data; a description will be given below. As suggested, we get a vast amount of data with respect to each user's movie interaction and the reviews submitted by each user. This provides another scope of improvement compared to the other works mentioned below, enhancing the prediction for the user. As suggested by [1], it is mandatory to resort to collaborative filtering. It is also mandatory to consider other users' interactions and group similar sets of users to see the movies watched and determine if something common emerges in each other's approach. This can then be used while predicting the ratings. Groupings can happen based on the set of movies liked across users, grouping them to figure out what the likeability factor of each user might be for each movie.

Apart from this, it is also mandatory to resort to employing a content-based filtering wherein the movies are analyzed genre-wise, with further other factors taken into consideration to arrive at a final result.

More light on this will be provided in the further sections.

II. Recent Works

There are various other disadvantages in not recommending a proper movie to a user; this drives the motivation to build a recommender system focused more on the user's personal choice along with the content of the movie. [3] also states the business advantages and the extent of a user's engagement contribution towards a streaming platform's success. It throws further light on a theoretical methodology for building a strategic recommender system. The theoretical analysis states that content-based filtering is equally vital as the model adhering to collaborative filtering. That is, the recommender system from the third-party provider must be useful, which does take business insights and platform limitations into consideration. Integrating this content with the user's choice will therein provide a better recommendation, which adds value to the user and also drives them to invest in a specific streaming platform. Though this analysis is very much platform-oriented, it still gives an edge to prove theoretically the need for both content-based and collaborative filtering integration in order to arrive at a better recommender system. The edge that the proposed solution in this paper provides is a practical and an applicative-oriented approach, taking all the theoretical and practical insights gained and providing a better recommendation system.

Apart from the business disadvantages, it is also mandatory to take personalized health hazards for an individual user into consideration. [2] discusses in detail the stress associated with streaming platforms, which addresses the negative effects of having a poor recommendation for a specific user. This gets them into a different zone altogether, and it becomes a chained reaction that affects them mentally. Furthermore, the engagement factor with any incorrect recommendation can drive the user to keep rooting for content that is not a better solution. Apart from that, it is also mandatory to keep in check that the user should not miss out on content, which forms the base for any recommender system. This [2] also talks about not providing a particular content to the user according to their wish, which can lead to a reduction in the excitement factor the user carries. Considering all this theoretical understanding, it is mandatory to also keep the emotions of the users in check before we can proceed with any recommender system. The solution addressed in this paper is, however, restricted to considering the user's likeability factor and the content of the movies into consideration and predicting the rating at a given particular time.

The practical difficulty in the prediction of the rating given by the user is the model's inefficiency to take external factors or content-oriented factors into consideration. [1] gives a theoretical approach of how a model could be achieved by having a hybrid model in check. But it does fail to address or build a model to predict the accurate rating for any particular movie. However, it does take the MovieLens dataset into consideration to give a better approach towards working with the recommendation problem. The insights drawn from this work were useful to plug the needed features into the model; more in this regard is discussed in further sections. The model suggested in our proposed solution is where we would practically employ and try to arrive at a better performance metric, thereby enhancing the prediction.

The practical approach to the recommender system is to employ a hybrid model and proceed with the recommendation. [4] proposes a solution that did focus on the same and tried building a model that employed a hybrid approach of combining both the collaborative filtering and content-based filtering approaches to arrive at a recommendation. The paper also induced a genetic algorithm, which cost them huge memory requirements to arrive at a consideration. This had multiple disadvantages, starting from occupying most of the memory to compute this recommendation. To arrive at a better result, we do not require the model to have an accurate prediction; for Example (eg), a rating of 4 and 4.2 are the same, which yields an accurate result, also ensuring the film falls under the likeability zone. So the level of prediction in rating can be determined. There are use cases, and there are various other users, who proceed further with movies if it holds a rating above 4. This imbalance in prediction will also be balanced with the lower rating, if the movie has received any, while computing the average rating like how our proposal does. These do have an effect, which will be discussed further.

III. Data Description

The data for this project was taken from from "The Movies Dataset" on Kaggle. It's a large collection that combines information from two main sources: The Movie Database (TMDB) and the MovieLens dataset. It includes details on over 45000 movies from 671 users.

This information is split into several files. The main one, moviesmetadata.csv, has all the basic facts for the 45,000 movies. This includes important details like the film's budget, how much money it made (revenue), its genre (like comedy or drama), a plot summary, the production company, and its release date. It also has the movie's average rating and the number of votes it received, which helps in judging its quality.

Other files add more detail. The credits.csv file lists the cast and crew, and keywords.csv lists the plot keywords for the films. This information was stored in a complex text format (JSON in string format), which means we had to process it first to get the useful data out.

For building a movie recommendation system, the most important file was ratings.csv. This file is huge, contains 26 million ratings (on a 1-5 star scale) from different users. To make testing our system faster and more manageable, a much smaller file (ratings_small.csv) was also provided. It has 100,000 ratings from 700 users for 9,000 movies. For convinence we consider the data in the small csv and proiceed further with the prediction of rating.

Finally, it is important to understand that machine learning models cannot use raw data directly, like reading a plot summary or a list of actors. Instead, we have to process that raw data to extract useful, organized information (called "features"). For eg, we have analyzed the text from the metadata, also ratings provided by different user to arrive at a recommendation.

A sample look at the dataset is given below.

TABLE I
SAMPLE DISPLAY OF MOVIE DATASET

	userId	movieId	rating	timestamp	tmdbId
0	1	31	2.5	2009-12-14 02:52:24	9909
1	1	1029	3.0	2009-12-14 02:52:59	11360
2	1	1061	3.0	2009-12-14 02:53:02	819
3	1	1129	2.0	2009-12-14 02:53:05	1103
4	1	1172	4.0	2009-12-14 02:53:25	11216

IV. DATA PREPROCESSING

For the predictive analysis, we keep track of and select the useful columns from the dataset. This includes IDs (userId, movieId), rating information, and movie metadata.

A key preprocessing step involved the genre_list. The genre field, which is a JSON object, was passed to a function to extract the name field for each genre. This was then assigned to a new field as a list called genre_list. Following this, we are using the | character as a pipeline to segregate the genres, which allows for the creation of dummy variables. To reduce the feature space, the rarest genres (appearing in 20 or fewer rows) were dropped.

Regarding outlier removal, based on the literature survey, it was clearly stated that the rating has contributed to most of the movies; therefore, dropping data isn't the solution. Instead, we are removing the inappropriate columns to proceed further.

Based on the result, the number of movies the dataset has is 9,025, and the number of users is 671. The useful columns have been taken up, including the IDs, the engineered genre_list, votecount, and vote_mean. The features will be engineered well, with necessary normalization being applied as needed.

V. DATA ANALYSIS:

From the initial analysis, which was done for the basic numerical data like Rating, Voting, and other popularity metrics, it is bought to light that the standard deviation of vote_count is larger. In contrast, the standard deviation for rating and vote_average is lesser.

Talking about the distribution, there are more movies which have ratings of 4, and the majority of the ratings fall under the 3-5 range. There are very few movies which have ratings lesser than 3.

For the Popularity distribution, the data is concentrated and more towards 0 - 100. Vote_average, however, is between the range of 4 - 8, and the data has a uniform spread over this range.

A. Rating Distribution

1) Graph Analysis: The "Rating distribution" bar chart shows the count for each rating value. It's clear that the data is not normally distributed. The highest bar is for a rating of 4.0, with almost 30,000 counts. The next most common ratings are 3.0 (around 20,000 counts) and 5.0 (around 15,000 counts). This confirms the note that the majority of the ratings falls under 3-5. The counts for ratings of 2.5 and below are much smaller, which also supports the observation that there are very few movies which have lesser than 3. The distribution is skewed to the left, as the tail of lower ratings is longer.

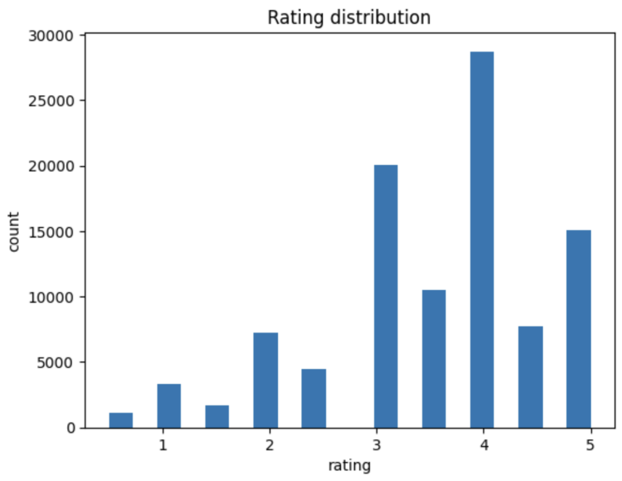


Fig. 1. Rating Distribution

B. Vote Count Distribution

1) Graph Analysis: The "vote_count distribution" histogram is heavily skewed to the right. There is a very large bar at the far left, showing that a massive number of movies (over 35,000) have a very low vote_count (likely between 0 and 500). The rest of the data forms a very long chain to the right, showing that a few movies have an extremely high vote_count, but these are rare.

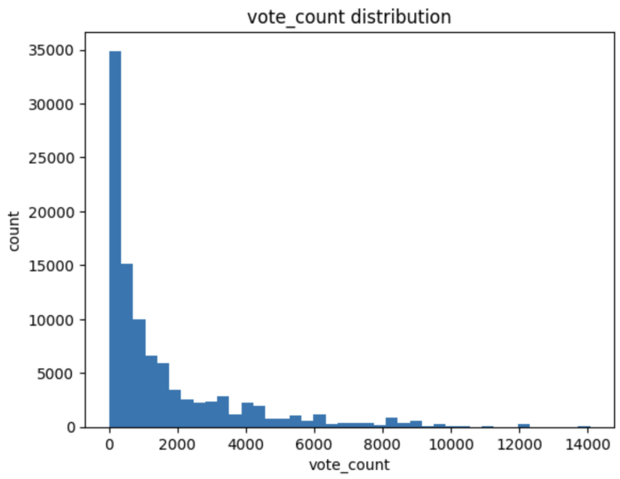
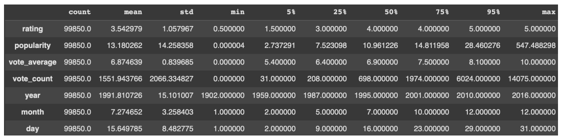


Fig. 2. Vote distrubition

C. Other Numerical Data

1) vote_average: The mean (6.87) and median (6.90) are almost the same, showing a symmetric, non-skewed distribution. The std (0.83) is very small, confirming your note that it is "lesser." The 25% mark is 6.4 and the 75% mark is 7.5, which shows that the middle 50% of all movie vote averages are packed into that small range.
2) popularity: This distribution acts similarly to vote_count but is less extreme. The mean (13.18) is higher than the median (10.96), which again points to a right-skew. This matches your note that the data is concentrated and more towards 0 - 100 (the table shows 75% of movies have a popularity of 14.8 or less)



	count	mean	std	min	5%	25%	50%	75%	95%	max
rating	99850.0	3.542979	1.057967	0.500000	1.500000	3.000000	4.000000	4.000000	5.000000	5.000000
popularity	99850.0	13.180262	14.258358	0.000004	2.737291	7.523098	10.961226	14.811958	28.460276	547.488298
vote_average	99850.0	6.874639	0.839685	0.000000	5.400000	6.400000	6.900000	7.500000	8.100000	10.000000
vote_count	99850.0	1551.943766	2066.334827	0.000000	31.000000	208.000000	698.000000	1974.000000	6024.000000	14075.000000
year	99850.0	1991.810726	15.101007	1902.000000	1959.000000	1987.000000	1995.000000	2001.000000	2010.000000	2016.000000
month	99850.0	7.274652	3.258403	1.000000	2.000000	5.000000	7.000000	10.000000	12.000000	12.000000
day	99850.0	15.649785	8.482775	1.000000	2.000000	9.000000	16.000000	23.000000	29.000000	31.000000

Fig. 3. Tabular data

To understand how the users have actually contributed to the rating, a plot is made. This is done by grouping the ratings together by userId and plotting the number of ratings the user has made. From this, it is seen that most of the users do provide ratings, but most of the users provide less than 500 reviews. There are only a few users who have rated movies more than 1000 reviews, but not more than that.

A popularity plot, which shares the details of how much movies are rated, was also generated. For the Number of Reviews, the number of reviews written for each movie is less, which is in the range between 0-50. The number of ratings between the rest of the ranges, wherein the number of ratings a movie holds is lesser. But this can still be helpful for the regression problem we hold, with which the likeability factor can be determined.

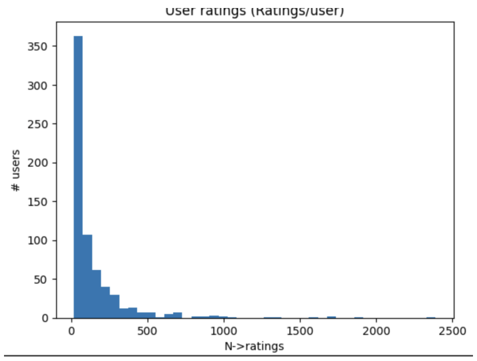


Fig. 4. User rating

For the genre distribution, all the genres were taken in hand. After calculating movies per genre and printing the sorted values just to visualize the count, we can see which genre has higher movies. Talking about the movie genre, the number of movies in the genre is Drama; the least is TV movie. After calculating the average rating per genre, it was found that, however, TV Movie category movies is the highly rated movie category. (This could also be a reason because this has a lesser number of movies).

A brief addition mentioned in the abstract is also made:

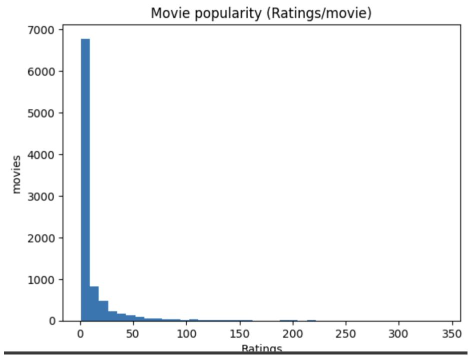


Fig. 5. Movie Popularity Graph

apart from calculating the rating, in order to proceed further with the recommendation, we are determining the likeability factor. We are adding a new column where ratings which exceed 4.0 fall into the likeability zone. And that is the movie we are recommending to the user, so as to enhance the movie experience.

This is the like rate per genre, calculating what will be the likes per genre. We are sorting the like rates to find out which user has liked which genre the most. Also, talking about the likeability factor, it is determined that the movies belonging to the category of history is the most liked movie category.

Finally, we are seeing the correlation between numeric features and also checking for skewed data.

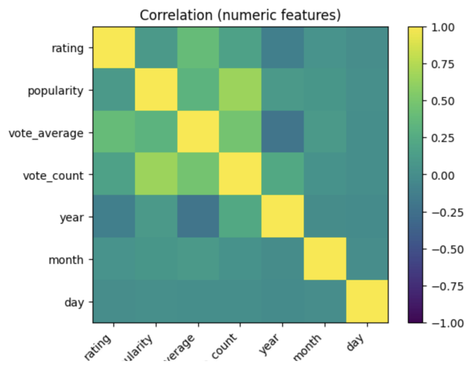


Fig. 6. Correlation Graph

Just to confirm the numerical result we achieved in the previous steps, the users hold a lower number of ratings, and the ratings for any specific movie are contributed by different users. This also supports the fact that there were a lot of movie ratings spread across 4 and 5, just as we figured from the numeric values.

The plots given above also talk about how the review and rating trend has changed over the years. Since 1980, the reviewer and rating trend has changed, just as to enhance the movie-watching experience for any given user.

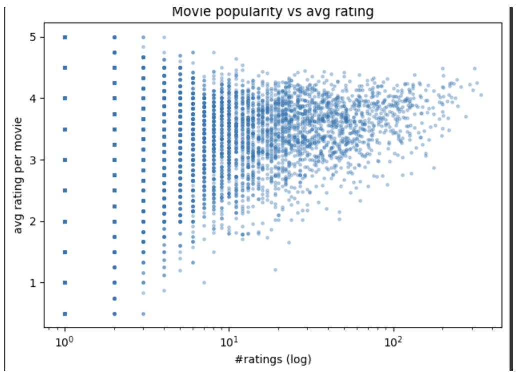


Fig. 7. Average rating graph/user

There are multiple factors for a particular user to choose a movie at a given time. But this time, we will proceed further with the features we have in hand and predict the rating, and based on the rating, we will determine the likeability rating.

VI. Proposed Methodology:

The approach in this study is focused on how the hybrid model is being built and how that hybrid model is enforced. This is done using a couple of things. The first part is processing the ratings that are there for the movies, from which we calculate the voted average. This voted average is just with respect to the movies themselves, providing a movie-centric baseline.

The second part is analyzing how this varies for different users, which is with respect to the specific user interaction with the respective movies. By combining both the movie's average data and the data from user interactions, the hybrid model is built. We also provide in enough data to get insights for this process. A detailed explanation of these components is given in further subsections.

The below is the Prototype of the model built.

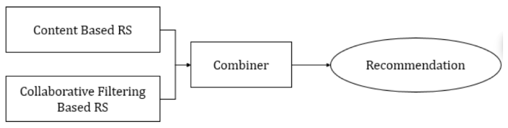


Fig. 8. Model Protype

A. Feature Extraction

As mentioned in previous subsections this dataset has a lot of irrelevant fields, which includes the tagline and the collection it belongs to which is needed, but then more than half of the samples lack collection so having that to fill null values proceeding further doesn't make sense to the problem statement taken so columns which tends similar have been dropped, since the focus is not about the textual analysis of the content that is skipped as well. The meta data however is been extracted and using pipelines the genre of a particular mvie based on the given movie ids is extracted and placed in different fields.

Following this the count of each genre is made for further model training , and the model is fed in with enough insights so as to ensure there is no overfit also that ensures the model does not suffer from underfit with the features provided.

A feature space of different statistical measures are employed : (Minimum value, Mean, Variance,Max Value, Standard Deviation, and Percentile). The result of the feature computed from the feature set is grouped by user eg Ratings. For instance, when the rating or the voting average for a particular individual change, the data will be taken into account, the measures mentioned above are computed.

For each and every numeric feature/Column we created standardized versions (z-scores). For heavy-tailed columns (popularity, vote_count) we applied log1p first, then applied standardization. Scaled columns are followed with __z; and the log versions are followed by _log1p as column names.

B. Implementation of ML models

After the preprocessing pipeline (ID joins, type fixes, date parsing, genre one-hots, and z-scaling of numeric metadata), we trained four supervised regression models—Linear Regression (OLS), KNN Regressor, Random Forest Regressor, and XGBoost Regressor—on the selected feature set (z-scored numerics, genre indicators, and optional user/movie means). We adopted an 80/20 holdout methodology: 80% of the data was used for training and 20% was reserved strictly for final evaluation. Models were run with sensible default settings (no exhaustive tuning) to provide a fair baseline comparison. Performance was assessed using four metrics: RMSE (primary error metric), MAE, R², and Explained Variance. The model achieving the lowest RMSE / MAE and highest R² / Explained Variance on the held-out set was selected as the best performer. In our runs, XGBoost achieved the strongest generalization.

C. Hyper Parameter tuning

Modeling setup (5-fold cross-validation with hyperparameter tuning). After the preprocessing mentioned above (ID joins, type fixes, date parsing, genre one-hots, and z-scaling of numeric metadata), four supervised regression algorithms were executed: linear regression (OLS), KNN regression, random forest regression, and XGBoost regression on the selected feature set. Thirty percent of the feature set is not isolated in this phase; instead, model selection was performed through K-fold cross-validation with K = 5 to obtain a robust generalization estimate. For the tunable models (KNN, Random Forest, XGBoost), we applied randomized hyperparameter search within the cross-validation loop to identify configurations that balance bias and variance.

Performance was analyzed using four metrics :RMSE (primary), MAE, R², and Explained Variance which is computed on each fold and summarized as mean ± standard deviation across the five folds. The model that minimized RMSE/MAE and maximized R²/Explained Variance under this 5-fold protocol was considered the best fit, maintaining harmony by being neither an overfit nor an underfit model; in our experiments, the tuned XGBoost configuration emerged as the top performer and was subsequently refit on the full dataset with the selected hyperparameters.

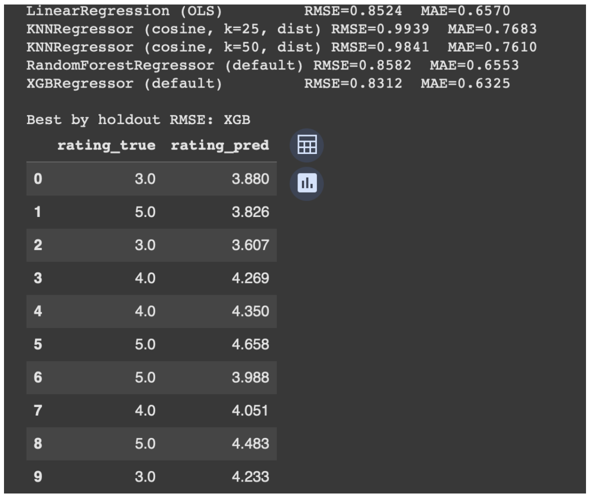
VII. Result

A. Evaluation metric used for model evaluation:

- **RMSE (Root Mean Squared Error):** chosen as the primary error measure since it goes against larger deviations more heavily, thus ensuring that the model is overall fit thereby ensuring mispredictions. So we will use this term as the primary metric to evaluate model efficiency.
- **MAE (Mean Absolute Error):** included to report an average absolute deviation that is prone or identify outliers and easy to interpret on the original rating scale.
- **R^2 (Coefficient of Determination):** reported to quantify the proportion of variance in ratings explained by the model, thereby indicating explanatory power.
- **Explained Variance:** used alongside R^2 to summarize how much variance from the original target is captured by the moel and this evaluated.

B. Model Evaluation

Below is the model's performance before hyper parameter tuning : As seen in the result, it is taken to notice that with



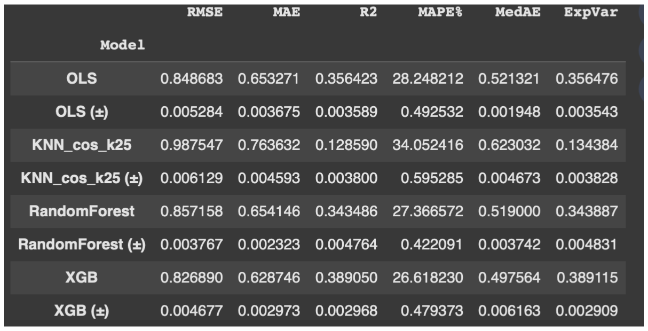
```
LinearRegression (OLS)         RMSE=0.8524  MAE=0.6570
KNNRegressor (cosine, k=25, dist) RMSE=0.9939  MAE=0.7683
KNNRegressor (cosine, k=50, dist) RMSE=0.9841  MAE=0.7610
RandomForestRegressor (default) RMSE=0.8582  MAE=0.6553
XGBRegressor (default)         RMSE=0.8312  MAE=0.6325

Best by holdout RMSE: XGB
```

	rating_true	rating_pred
0	3.0	3.880
1	5.0	3.826
2	3.0	3.607
3	4.0	4.269
4	4.0	4.350
5	5.0	4.658
6	5.0	3.988
7	4.0	4.051
8	5.0	4.483
9	3.0	4.233

Fig. 9. 80-20 Split

the 80-20 split XGBoost behaves the best, with RMSE 0.83.

Below is th evaluation result after hyper parameter tuning which is done using K cross Validation with k as 5 reason stated above.



	RMSE	MAE	R2	MAPE%	MedAE	ExpVar
Model						
OLS	0.848683	0.653271	0.356423	28.248212	0.521321	0.356476
OLS (±)	0.005284	0.003675	0.003589	0.492532	0.001948	0.003543
KNN_cos_k25	0.987547	0.763632	0.128590	34.052416	0.623032	0.134384
KNN_cos_k25 (±)	0.006129	0.004593	0.003800	0.595285	0.004673	0.003828
RandomForest	0.857158	0.654146	0.343486	27.366572	0.519000	0.343887
RandomForest (±)	0.003767	0.002323	0.004764	0.422091	0.003742	0.004831
XGB	0.826890	0.628746	0.389050	26.618230	0.497564	0.389115
XGB (±)	0.004677	0.002973	0.002968	0.479373	0.006163	0.002909

Fig. 10. 80-20 Split

After hyper paramter tuning the result has further improved with the betterment of RMSE and the deviation is comparativey is less which is also indicated in the xGBoost .

C. Overview of results

It is to bring out the result that with the given result stated above, the xGBoost algorithm, being a supervised regressor, performs the best, outshining RandomForest, Linear Regression, and KNN. With just a deviation of 0.004, which is proved to be the better result on the same line of interests. [4] also takes ML algorithms into account and it chose to fail, with KNN's RMSE of 0.95.

It is then that they employed evolutionary computing to enhance the rating prediction, which still proves to be the best, but the introduction of likeability as one more target variable from here will further scale down the problem and make it better.

VIII. Analysis and Discussion

The model's efficiency is influenced by several other factors like user's personal choice to look out for movies.

A. Impact of content

So the general rating given by the user is dependent on the personal choice but it also takes into consideration the user's collaborative choice being made while picking a movie to watch. The content starts from picking the right genre for the user that he/she wishes to watch, which also takes the other content material such as the year it got released, followed by its voted average which contributes to the overall rating prediction. This combined together with the user preference will yield a better result.

B. Impact of Machine learning Algorithm

Four distinct algorithms were implemented with the taken dataset, and it was proven that XGBoost performs the best amongst all the other algorithms. XGBoost worked the best while determining the personalized rating state which also determines the recommendation threshold for an individual and intimate them accordingly to start watching the suggested movies. But since the comparative performance of determining the preference state XGBoost proved to be predominant over the other algorithms. Since XGBoost employs boosted decision trees that get divided to different subtrees and evaluate them with a gradient-based objective in order to overcome any loss of information while computing. There was an option to prefer deep learning models but considering the space and time constraints the solution is confined and proved to be better with the XGBoost algorithm for the movie recommendation problem.

IX. Conclusion

Movie recommendation is essential for every individual in day-to-day life just for the fact that it meaningfully reduces search cost and improves viewing satisfaction. However, the available solutions are either restricted with their approach towards the problem by relying solely on popularity heuristics or by using a pre-determined formula for scoring reviews, which have failed to address the problem with a proper evaluation metric for diverse users. In the solution that is been proposed in this paper, it is an automated prediction of user ratings with a proper consideration of both content and colllaborative filtering approach. The ratings data are combined with movie metadata (genres, release year, vote statistics) across many users and titles to analyze the characteristics, though not over-generalizing the result the aim was to find out if there were any consistent trends for particular feature configurations. It is explored that XGBoost has performed the best overall, with an RMSE closer to 0.82 on the held-out set, accompanied by a strong MAE and positive R^2/Explained Variance, and the model predicts the rating with minimal train–test gap which strengthens the solution and moves the recommendations towards higher-scored candidates while avoiding false promotion of poorly matched items.

X. Future Work

This solution can still be extended by expanding the feature space to various other factors wherein we can take the content or the premise into consideration, and provide an overall rating system, also taking other external factors which might influence a user to resort to movies of a specific genre, which could also be added to the list. This could be achieved by integrating IOT sensors to capture other external climatic factors, findings did prove it has an edge over such data.

References

[1] Mahesh Goyani, and Neha Chaurasiya "A Review of Movie Recommendation System" , 6th of May 2020.

[2] un Kim1, and Jong-Hwan Choi1, and Gyeongju, and Tan-Tan Bao , A Study on the Effect of Content Choice Deferral on Stress .

[3] Marc Bourreau, and Germain Gaudin ,Streaming platform and strategic recommendation bias.

[4] Shreya Agrawal, and Pooja Jain ,An Improved Approach for Movie Recommendation System.