IMDB Movie Rating Prediction: A Crew-Centric Approach

Ketki Patankar
University of California, San Diego
kpatankar@ucsd.edu

Varun Sankar Moparthi
University of California, San Diego
vmoparthi@ucsd.edu

ACM Reference Format:
Ketki Patankar and Varun Sankar Moparthi. 2025. IMDB Movie Rating Prediction: A Crew-Centric Approach. In . ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The task at hand involves building a movie rating prediction model using the IMDB dataset, which includes movie-related metadata such as title ID, actors, actresses, producers, directors, number of votes, genres, start year, and average rating. While typical movie recommendation models rely heavily on user-based information (such as user ratings and preferences), this dataset lacks user-specific data, presenting a unique challenge. Instead of utilizing user interaction data, this project will focus on leveraging available movie attributes to predict ratings. By employing collaborative filtering techniques between crew and movie and other machine learning algorithms, we aim to uncover underlying patterns in the movie metadata that correlate with higher or lower ratings. This approach will enable the creation of a robust recommendation system capable of predicting movie ratings for new or unseen films based on their intrinsic features.

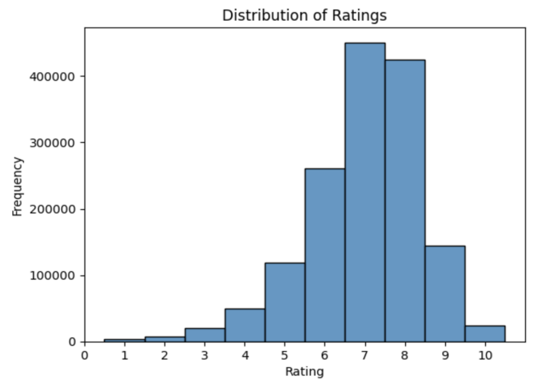
2 Dataset

For our task we are using the IMDB dataset - **https://www.kaggle.com/datasets/ashirwadsangwan/imdb-dataset/data**.

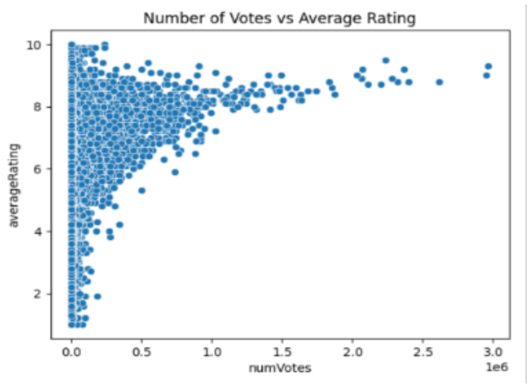
This dataset is in the form of 5 different TSV files. Each file and its contents are clearly listed in the Kaggle webpage. A brief description of what data each file contains is given below:
title.ratings – It contains all the movies/tv shows on IMDB for which we have an average rating estimate available. A user vote count is also provided for the same.
title.basics – It contains movie related details/movie features.
title.akas – It contains additional details about movies like alternate name and other languages in which that movie is available.
name.basics – It contains information about the crew/cast members.
title.principals – This is our primary file, which links various crew and cast members to movies.

2.1 Exploratory Analysis

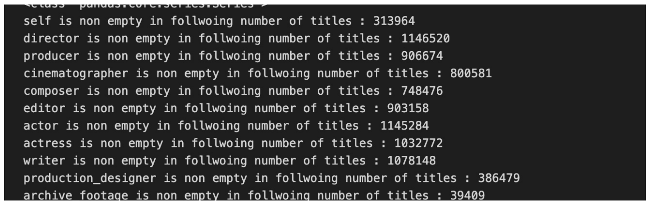
2.1.1 Analysis of Ratings TSV:. The Ratings range from 0 to 10, with a total of 1,502,631 movies rated. Therefore, this dataset is quite large and optimal for our predictive task.



The number of votes per movie ranges from 5 to 2,966,204, with an average of 1,027.92 votes. Out of the 1,502,631 movies, only 90,952 have vote counts exceeding the average. This suggests that the majority of movies have vote counts below the average. This observation is further confirmed by the scatter plot given below.

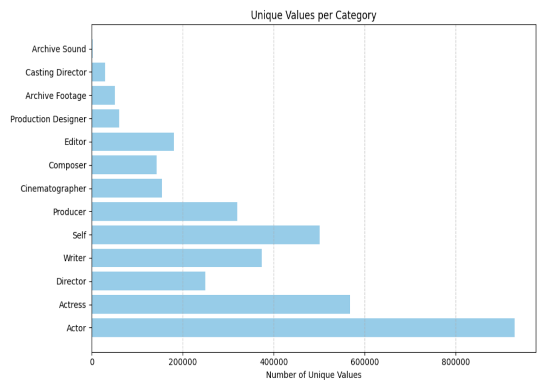


2.1.2 Analysis of Principles TSV:. Out of the 13 different categories presented in the dataset, we have focused on the actor, actress, director, and writer category for the crew/cast, as they have fewer missing values. The peculiarity of these categories is that they can include multiple entities (e.g., several actors for a single movie). From the figure given below, we can confirm that the above mentioned four categories have lesser missing data.



```
self is non empty in follwoing number of titles : 313964
director is non empty in follwoing number of titles : 1146520
producer is non empty in follwoing number of titles : 906674
cinematographer is non empty in follwoing number of titles : 800581
composer is non empty in follwoing number of titles : 748476
editor is non empty in follwoing number of titles : 903158
actor is non empty in follwoing number of titles : 1145284
actress is non empty in follwoing number of titles : 1032772
writer is non empty in follwoing number of titles : 1078148
production_designer is non empty in follwoing number of titles : 386479
archive_footage is non empty in follwoing number of titles : 39409
```

Additionally, the figure below confirms that these selected categories also represent the top groups with the highest number of crew and cast members.



2.1.3 Analysis of Title Basics TSV. The dataset comprises information about movie samples like titleId, PrimaryTitle, StartYear, IsAdult, genres.

We will utilize features such as numVotes, startYear, genres, and crew/cast details, as these are expected to have a significant impact on the rating. These features will play a key role in the models we build for predicting ratings in subsequent tasks.

3 Predictive Task

The goal of this task is to predict the average movie rating based on features like crew/cast (directors, writers, producers, actors), and movie metadata (e.g., release year, runtime, genre). Since there are no explicit user ratings, the task focuses on inferring ratings from these movie characteristics and relationships between movie and its crew.

3.1 Evaluation Metrics

For evaluating our model's performance, we have used the Mean Squared Error as our evaluation metric.

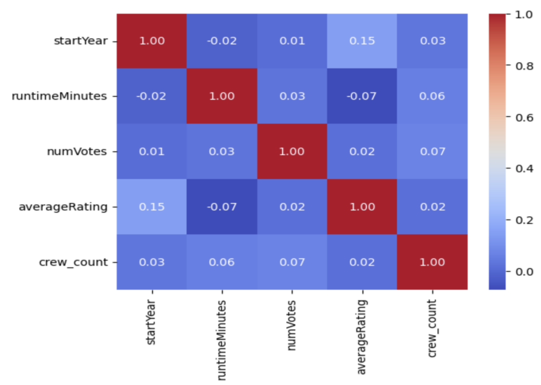
3.2 Feature Engineering

3.2.1 Feature Correlation analysis (linear): Using correlation matrix, we analyzed the correlation of four continuous valued features with average ratings.

The correlation between average rating and crew count (**0.017606**) is slightly negative but very weak. This suggests that having more people in the crew is weakly associated with slightly lower average ratings.

The correlation between average rating and movie runtime (**-0.074047**) is weakly negative. This suggests that, on average, longer movies tend to have slightly lower ratings.

The correlation between average rating and release year (**0.146346**) is weakly positive. This suggests that there is a slight tendency for movies released in more recent years to have slightly higher average ratings.

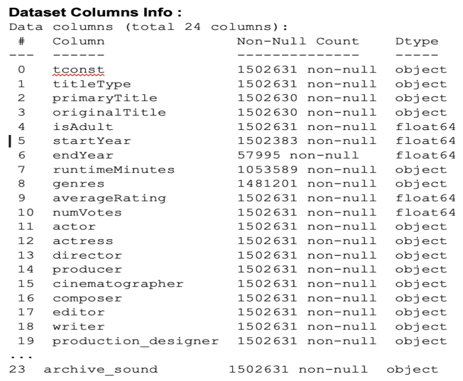


The correlation between average rating and number of user votes per movie (**0.017127**) is very weakly positive. This suggests that the number of votes a movie receives is almost uncorrelated with its average rating.

3.2.2 Data Merging/Consolidation. To get the inclusion of rating and number of votes a movie has received, the **titles.basics** and **title.rating** datasets are merged keeping the titleid as a common key.

To include the crew that worked in a movie based on job categories to the above consolidated dataset 13 disjoint job category columns are added where each titleId takes a set of crewIds that worked in a job for a given movie. This information is given by dataset **title.principals** where titleId is used as a common key to find out all the categories crewIds in the **title.principals** and laterthose crew details are added to the above newly created job columns.

The final merged dataset contains information coupled from all the following datasets **titles.basics**, **titles.rating** and **titles.principles**, which are later used for building the recommendation models. Final Dataset contains around 1.5 million data samples with the following columns.



Dataset Columns Info :

```
Data columns (total 24 columns):
 #   Column               Non-Null Count    Dtype
---  ------               --------------    -----
 0   tconst               1502631 non-null  object
 1   titleType            1502631 non-null  object
 2   primaryTitle         1502630 non-null  object
 3   originalTitle        1502630 non-null  object
 4   isAdult              1502631 non-null  float64
 5   startYear            1502383 non-null  float64
 6   endYear              57995 non-null    float64
 7   runtimeMinutes       1053589 non-null  object
 8   genres               1481201 non-null  object
 9   averageRating        1502631 non-null  float64
 10  numVotes             1502631 non-null  float64
 11  actor                1502631 non-null  object
 12  actress              1502631 non-null  object
 13  director             1502631 non-null  object
 14  producer             1502631 non-null  object
 15  cinematographer      1502631 non-null  object
 16  composer             1502631 non-null  object
 17  editor               1502631 non-null  object
 18  writer               1502631 non-null  object
 19  production_designer  1502631 non-null  object
...
23  archive_sound        1502631 non-null  object
```

3.2.3 Handling missing data. Given the dataset that is reasonably large, for the datasample that has **NaN**'s in the following columns startYear, genres, runtimeMinutes the data samples has been discarded and removed from the dataset. The final data set has around 1 Million datasamples.

3.3 Baseline Comparisions

Traditional collaborative filtering models rely on the user-item interaction matrix. One of the main goals of these techniques is to understand the interaction between the user and items in the form of the similarity between them. We have extended this idea to our model which lacks user-specific data. We have used four different baselines that follow collaborative filtering approaches as listed below:

3.3.1 Jaccard crew-to-crew similarity: In this case, the model behaves like a mean predictor. This occurs because the training and test sets are disjoint, meaning the movies in the test set are not present in the training set. As a result, the model always predicts the average movie ratings.

3.3.2 Jaccard crew-to-crew similarity with shuffled dataset: On shuffling the data before test train split, we ensure that the movies are all not accumulated together and are shuffled up and likely to be in both the training and test sets rather than just one. While some movies in the test set still don't appear in the training set, the shuffling improves the model's performance, leading to more accurate predictions and a reduced MSE.

3.3.3 Cosine similarity for crew-to-crew: Replacing the Jaccard similarity with the Cosine similarity in our model, we observed there is a little improvement in Mean Squared Error. This is because cosine similarity helps normalize the effect of crew members by focusing on commonality between the crew members irrespective of number of crew members for a movie.

3.3.4 Pearson Similarity for crew-to-crew: Measures the similarity between two movie crews based on the shared characteristics or collaborations between them. In this context, we treat each crew member (director, actor, etc.) as an entity, and calculate the similarity between different crew members (or sets of crew members) based on their involvement in movies.

4 Models

For our prediction task, we have selected two models that demonstrate significantly better performance compared to the baseline models as our final choices.

Latent Factor Model

Sparse Matrix Random Forest with Hyper parameter Tuning

4.1 Latent Factor Model

A latent factor model predicts ratings by learning hidden factors that explain the interaction between items and features such as crew IDs (e.g., actors, actresses, directors). Instead of using explicit user IDs, it decomposes the data into latent factors capturing complex relationships between crew-related features and movie ratings.
Matrix Factorization is a type of latent factor model where the observed data (e.g., ratings matrix) is factorized into latent user and item matrices. Our model is a Matrix factorization model whose latent factors are then optimized using gradient descent.

4.1.1 Feature Vector of datasample: Here, actor, actress, director, genres are the set of features used along with numMinutes, isAdult for the rating prediction task.

To build the features of actor, actress,directors, genres, first they are one hot encoded and later added with numMinutes, isAdult to make a complete feature vector. The same is repeated for all the data samples in the dataset. This will create a very large sparse matrix of features.

Later sparse matrix is decomposed into latent factors of dimension 1 and parameters are learned via gradient descent keeping the learning rate at 0.01 and regularizer at 0.1.
Latent Factor Models are better suited for handling large datasets, sparse matrices, and can incorporate additional features into the model. They are more efficient, scalable, and generalize well to unseen data. Collaborative Filtering, which is effective in some cases, tends to struggle with large, sparse datasets, and cold-start problems. It is also more limited in terms of incorporating additional metadata.

4.1.2 Issues. When all the job categories are considered for feature vector, the sparseness of the feature vector created, causes the issue of kernel crashing when the feature matrix is build for the whole data samples.
Later after looking at the analysis of **title.principles** dataset we observed that actor, actress, director, and producer categories are most common in a data sample and all other categories has been mostly an empty set. Hence, when creating the one hot encoded vector we have considered only actor, actress and director as features. But when features are created for whole dataset, it again raised the issue of kernel crashing. To overcome this issue we took a part of the dataset and used 95% for training and 5% for testing.

4.2 Sparse Matrix Random Forest with Hyperparameter Tuning

Random forests are ensemble models that combine multiple decision trees to make predictions. Each tree uses different feature subsets, helping the model capture complex, non-linear relationships between input features (e.g., crew members) and target (e.g., ratings). In this case, the relationship between crew members (e.g., which actor, director, or writer worked on a movie) and the ratings may not be linear or straightforward, and Random Forest can model these complexities better than other models that we have seen.

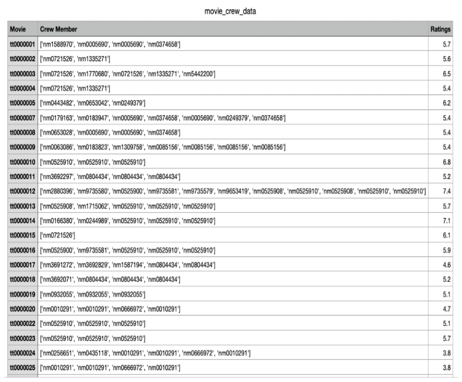
In this approach, we create a sparse matrix where each row represents a movie and each column corresponds to a unique crew member (actor, actress, director, or writer). For each movie, we mark the relevant crew members with a 1 in the matrix, indicating their involvement in that movie. We make use of this matrix to make predictions on movie ratings.

This data is then split into a train and test set with a split of 80:20. These are then converted into dense matrix as RandomForestRegressor typically requires a dense matrix as input.

We have used GridSearchCV to perform an exhaustive search over the hyperparameter grid. We have used a crossvalidation of 3 to evaluate different combinations of hyperparameters. The bestperforming set of parameters are then selected based on the neg_mean_ squared_error scoring metric.
The hyperparameter for our tuned model are: **param_grid** = { **'n_estimators'**: 50 , **'max_depth'**: 150 , **'random_state'**: 42 , **'n_jobs'**: -1 , **'warm_start'**: True }

4.3 Unsuccessful attempts:

In collaborative filtering methods where we try to predict the rating based on similarities of the title and crew. When we use the pearson based similarity in rating prediction observed that mse shooted to 2.2231 which is higher than the baseline comparison of predicting average rating all the time.
When implementing the Jaccard Similarity based ratings prediction task, we initially considered a dataset that looks like the below image:

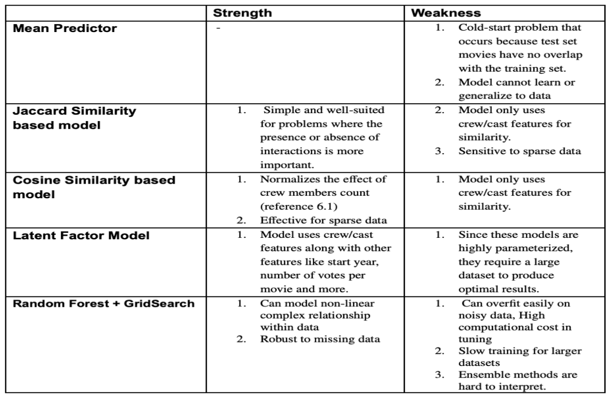


movie_crew_data

Movie	Crew Member	Ratings
tt0000001	['nm1588970', 'nm0005690', 'nm0005690', 'nm0374658']	5.7
tt0000002	['nm0721526', 'nm1335271']	5.6
tt0000003	['nm0721526', 'nm1770680', 'nm0721526', 'nm1335271', 'nm5442200']	6.5
tt0000004	['nm0721526', 'nm1335271']	5.4
tt0000005	['nm0443482', 'nm0653042', 'nm0249379']	6.2
tt0000007	['nm0179163', 'nm0183947', 'nm0005690', 'nm0374658', 'nm0005690', 'nm0249379', 'nm0374658']	5.4
tt0000008	['nm0653028', 'nm0005690', 'nm0005690', 'nm0374658']	5.4
tt0000009	['nm0063086', 'nm0183823', 'nm1309758', 'nm0085156', 'nm0085156', 'nm0085156', 'nm0085156']	5.4
tt0000010	['nm0525910', 'nm0525910', 'nm0525910']	6.8
tt0000011	['nm3692297', 'nm0804434', 'nm0804434', 'nm0804434']	5.2
tt0000012	['nm2880396', 'nm9735580', 'nm0525900', 'nm9735581', 'nm9735579', 'nm9653419', 'nm0525908', 'nm0525910', 'nm0525908', 'nm0525910', 'nm0525910']	7.4
tt0000013	['nm0525908', 'nm1715062', 'nm0525910', 'nm0525910', 'nm0525910']	5.7
tt0000014	['nm0166380', 'nm0244989', 'nm0525910', 'nm0525910', 'nm0525910']	7.1
tt0000015	['nm0721526']	6.1
tt0000016	['nm0525900', 'nm9735581', 'nm0525910', 'nm0525910', 'nm0525910']	5.9
tt0000017	['nm3691272', 'nm3692829', 'nm1587194', 'nm0804434', 'nm0804434']	4.6
tt0000018	['nm3692071', 'nm0804434', 'nm0804434', 'nm0804434']	5.2
tt0000019	['nm0932055', 'nm0932055', 'nm0932055']	5.1
tt0000020	['nm0010291', 'nm0010291', 'nm0666972', 'nm0010291']	4.7
tt0000022	['nm0525910', 'nm0525910', 'nm0525910']	5.1
tt0000023	['nm0525910', 'nm0525910', 'nm0525910']	5.7
tt0000024	['nm0256651', 'nm0435118', 'nm0010291', 'nm0010291', 'nm0666972', 'nm0010291']	3.8
tt0000025	['nm0010291', 'nm0010291', 'nm0666972', 'nm0010291']	3.8

However, predicting ratings for a given movie and a list of crew/cast members—rather than just a single member—caused significant slowdowns. This was primarily because, for each Jaccard comparison, instead of using the movie set for a single crew member, we were considering the union of movies for all crew members in the list and calculating the unique set.

4.4 Strength and Weakness – Baseline vs Our Model



	Strength	Weakness
Mean Predictor	-	1. Cold-start problem that occurs because test set movies have no overlap with the training set. 2. Model cannot learn or generalize to data
Jaccard Similarity based model	1. Simple and well-suited for problems where the presence or absence of interactions is more important.	2. Model only uses crew/cast features for similarity. 3. Sensitive to sparse data
Cosine Similarity based model	1. Normalizes the effect of crew members count (reference 6.1) 2. Effective for sparse data	1. Model only uses crew/cast features for similarity.
Latent Factor Model	1. Model uses crew/cast features along with other features like start year, number of votes per movie and more.	1. Since these models are highly parameterized, they require a large dataset to produce optimal results.
Random Forest + GridSearch	1. Can model non-linear complex relationship within data 2. Robust to missing data	1. Can overfit easily on noisy data, High computational cost in tuning 2. Slow training for larger datasets 3. Ensemble methods are hard to interpret.

5 Literature Review

5.1 Similar Datasets & Previous Work

Imdb-dataset - https://www.kaggle.com/datasets/yukyungchoi/imdb-dataset
This dataset is predominantly used for Sentimental Analysis on movie reviews. Many contributors on kaggle have used this dataset to predict if a particular review is a positive or negative sentiment. A bag-of-words approach is used where we convert each token to a numerical feature vector using CountVectorizer. An Stochastic Gradient Descent Classifier (SGDClassifier) with hinge loss IS used for binary classification of the review into positive and negative sentiment.
IMDB Dataset of 50K Movie Reviews - https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
This dataset is used for Natural Language Processing related tasks or for text analysis. A popular model that can be implemented on this dataset is the LSTM-based text classification model.

5.2 State-of-the-art models

5.2.1 Content-Based Filtering with Deep Learning - NCF:. It combines the strengths of traditional collaborative filtering with deep neural networks. Instead of relying on the linear assumption (like traditional CF), NCF learns complex, non-linear relationships between user and item interactions.Uses neural network architecture that takes user and item embeddings (learned representations) as input and outputs the predicted rating or interaction score.

5.2.2 Gradient boosting algorithms (XGBoost): XGBoost (Extreme Gradient Boosting) is a powerful algorithm that builds a strong predictive model by combining multiple weak decision trees. Each tree is added sequentially to correct the errors (residuals) of the previous ones. Unlike collaborative filtering, which relies on user-item data, XGBoost is ideal for structured/tabular data and can handle missing user information effectively by capturing complex, non-linear relationships.

6 Results and Conclusions

The quantitative comparisons of MSE's in Table1 shows that Latent factor model and Random Forest + GridSearch has very low MSE when compared to collaborative approaches because the models have the ability to find the underlying interactions between movies and crew providing the best estimated rating there by outperforming the similarity based approaches.

Table 1: Quantative Comparisions

Model	MSE
`Mean Predictor`	1.81
`Jaccard Similarity model`	1.0335
`Cosine Similarity model`	1.0326
`Random Forest + GridSearch*`	0.9005
`Latent Factor Model*`	0.75