RateMyMovie - A Movie Rating Predictor

Sanjeev Penupala, Gautam Sapre

The University of Texas at Dallas
sanjeev.penupala@utdallas.edu, gss1700001@utdallas.edu

Abstract

In this study, we aimed to develop a machine-learning model to predict the rating of a movie based on various features such as genre, director, and stars. To achieve this goal, we used a data set of over 6000 movies with ratings ranging from 0 to 10 (ratings were binned into 3 categories for classification). We explored several machine learning algorithms including decision trees, random forests, and gradient boosting, and conducted hyperparameter tuning to optimize their performance. Our best model achieved an F1-macro average score of 0.54. Our results suggest that machine learning models are slightly better than random guessing, due to the nature of the imbalance, when it comes to distinguishing the different movie rating classes.

Introduction

In recent years, machine learning has become an increasingly popular approach to solving complex problems in a variety of fields. One area in which machine learning has shown great promise is the prediction of consumer preferences. With the explosion of streaming services and online movie platforms, understanding what factors influence user ratings has become an important challenge for movie studios and content providers.

Predicting movie ratings can be a valuable tool for film professionals to use. For example, movie studios and production companies can benefit from accurate predictions of how well a movie in production will be received by audiences. This can help film professionals make key decisions regarding the production, budget, and marketing plan for their upcoming movies.

Furthermore, streaming services like Netflix and Amazon Prime want to provide better recommendations to their users to keep them engaged on their platforms. However, with new films, it's hard to choose whether to recommend it to a user or not until there have been enough user ratings on it, which takes time. With a movie rating predictor, it would be easy to filter out good from bad movies and keep users engaged with fresh films.

In this paper, we present our results aimed at predicting movie ratings using a dataset of movie features and user ratings. Our approach involved training several machine learning models on a subset of the data, tuning hyperparameters, and evaluating the performance of the models on a held-out test set.

The results of our project showed that machine learning algorithms can barely or somewhat predict movie ratings, with models having an F1 score between 0.3 to 0.6. Our project highlights the potential of machine learning to provide valuable insights into user preferences and to inform decision-making in the entertainment industry.

Data & Design

We used the "IMDb Movie Dataset: All Movies by Genre" [1] Kaggle dataset for our problem. Due to the size of the genre datasets, we decided to choose one genre dataset for our problem, the action genre. The dataset contained 50k+ examples, with the following main columns: year, certificate, runtime, genre, director, stars, and rating. After performing EDA, we found out there were over 189k null values and other challenging data parts, so we decided to impose the following changes to the dataset:

- **year**: from years 2000 to 2022 (modern films should be trained on modern data),
- **certificate**: remove it, since the majority of films didn't have certificates (null entries).
- **genre**: remove it, since we know our action is our genre.
- **director**: choose the main director only, since there were 1-2 per example.
- **stars**: choose the main star only, since there were 3-4 per example.

Even after the changes above, we still had over 60k null values, so we ran a movie scraping script to fill in the null values. We removed the remaining null values as the last resort, which left us with 6.6k examples. We also binned the ratings into three categories: below average, average, and above average, to turn it from regression to a classification problem. We wanted to choose between 3-5 bins since that range gives the most user-friendly understanding of the model, and after preliminary analysis, chose 3 bins, due to it having better accuracy across various models. However, we noticed this forced the classes to be imbalanced, with the average class being the majority class, since the ratings followed approximately a normal distribution. For the categorical encoding, we initially used CountEncoding, as OneHotEncoding wasn't an option due to its high dimensionality. However, we observed subpar results for our first iteration of testing, so we eventually adopted CatBoostEncoding, a variation of TargetEncoding that helps mitigate target leakage and keeps the same dimensions of the data. After choosing our models, we also performed hyperparameter tuning to get the best models we could.

Models

We implemented six different machine learning models to predict movie ratings: k-NN, Decision Trees, Random Forest, Logistic Regression, Naive Bayes, and Gradient Boosting. For each model, we evaluated its performance using several metrics, including the confusion matrix, classification report, and ROC curve.

K-NN

The K-NN model achieved an F1 Score of 0.52 on the test set. The confusion matrix for the model is shown in Figure 1. We also generated a classification report and ROC curve for the model, which are shown in Table 2 and Figure 2, respectively.

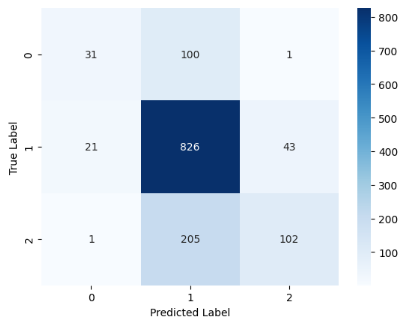

Figure 1: Confusion Matrix for K-NN model

Table 1: k-NN: TPR and FPR for each class

Class	TPR	FPR
0	0.2348	0.0184
1	0.9281	0.6932
2	0.3312	0.0431

Table 2: Classification Report for K-NN

0	0.58	0.23	0.34	132
1	0.73	0.93	0.82	890
2	0.70	0.33	0.45	308
accuracy			0.72	1330
macro avg	0.67	0.50	0.53	1330
weighted avg	0.71	0.72	0.68	1330

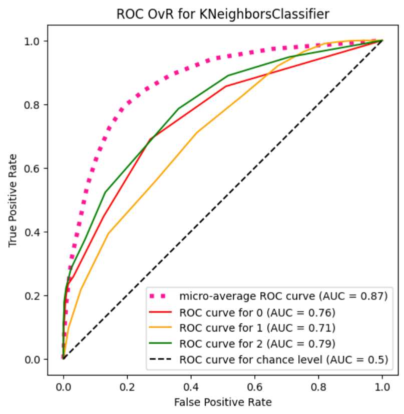

Figure 2: ROC curve for K-NN model

Decision Trees

The Decision Trees model achieved an F1 Score of 0.55 on the test set. The confusion matrix for the model is shown in Figure 3. We also generated a classification report and ROC curve for the model, which are shown in Tables 4 and Figure 4, respectively.

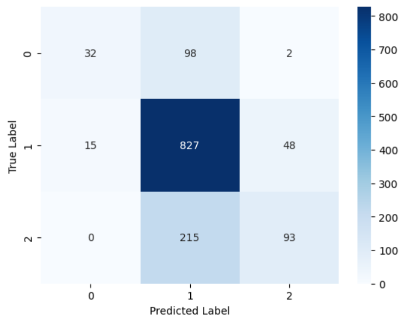

Figure 3: Confusion Matrix for Decision Tree model

Table 3: Decision Trees: TPR and FPR for each class

Class	TPR	FPR
0	0.2500	0.0134
1	0.9281	0.7091
2	0.3019	0.0489

Table 4: Classification Report for Decision Trees

	precision	recall	f1-score	support
0	0.67	0.25	0.36	132
1	0.74	0.92	0.82	890
2	0.66	0.35	0.46	308
accuracy			0.72	1330
macro avg	0.69	0.51	0.55	1330
weighted avg	0.71	0.72	0.69	1330

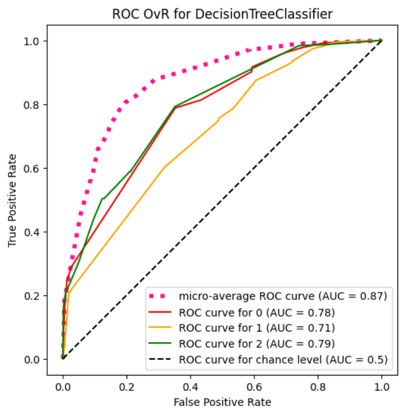

Figure 4: ROC curve for Decision Trees model

Logistic Regression

The Logistic Regression model achieved an F1 Score of 0.48 on the test set. The confusion matrix for the model is shown in Figure 5. We also generated a classification report and ROC curve for the model, which are shown in Table 6 and Figure 6, respectively.

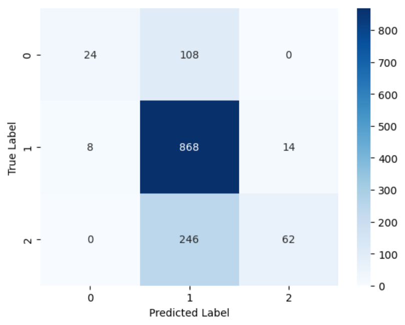

Figure 5: Confusion Matrix for Logistic Regression model

Table 5: Logistic Regression: TPR and FPR for each class

Class	TPR	FPR
0	0.1742	0.0058
1	0.9528	0.7341
2	0.2987	0.0362

Table 6: Classification Report for Logistic Regression

	precision	recall	f1-score	support
0	0.67	0.25	0.36	132
1	0.74	0.92	0.82	890
2	0.66	0.35	0.46	308
accuracy			0.72	1330
macro avg	0.69	0.51	0.55	1330
weighted avg	0.71	0.72	0.69	1330

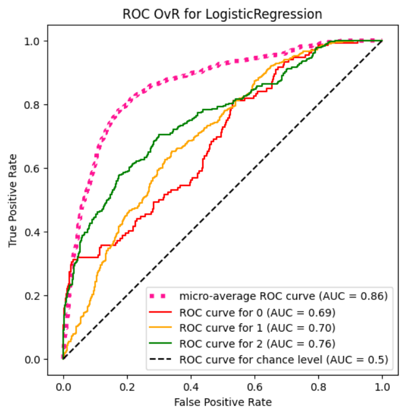

Figure 6: ROC curve for Logistic Regression model

Gaussian Naive Bayes

The Gaussian Naive Bayes model achieved an F1 Score of 0.48 on the test set. The confusion matrix for the model is shown in Figure 7. We also generated a classification report and ROC curve for the model, which are shown in Table 8 and Figure 8, respectively.

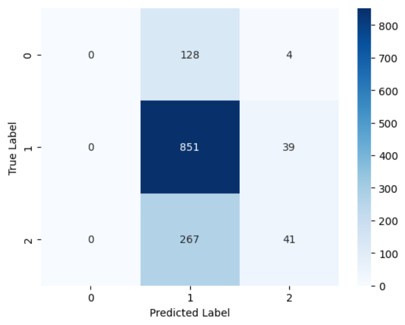

Figure 7: Confusion Matrix for Gaussian Naive Bayes model

Table 7: G-NB: TPR and FPR for each class

Class	TPR	FPR
0	0.0000	0.0000
1	0.9562	0.8977
2	0.1331	0.0421

Table 8: Classification Report for Gaussian Naive Bayes

	precision	recall	f1-score	support
0	0.00	0.00	0.00	132
1	0.68	0.96	0.80	890
2	0.49	0.13	0.21	308
accuracy			0.67	1330
macro avg	0.39	0.36	0.34	1330
weighted avg	0.57	0.67	0.58	1330

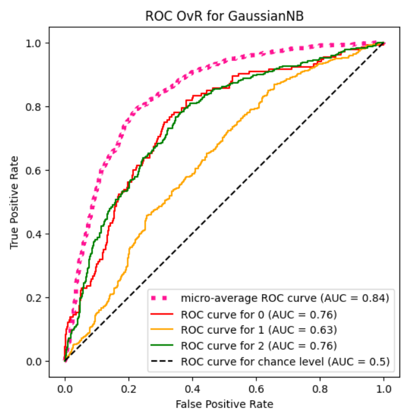

Figure 8: ROC curve for Gaussian Naive Bayes model

Random Forest

The Random Forest model achieved an F1 Score of 0.54 on the test set. The confusion matrix for the model is shown in Figure 9. We also generated a classification report and ROC curve for the model, which are shown in Table 10 and Figure 10, respectively.

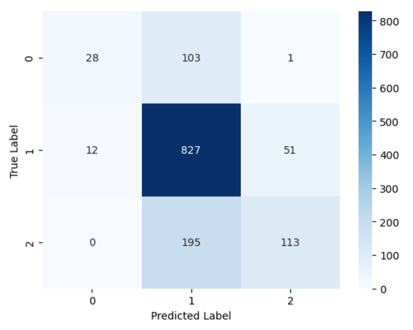

Figure 9: Confusion Matrix for Random Forest model

Table 9: Random Forest: TPR and FPR for each class

Class	TPR	FPR
0	0.2121	0.0100
1	0.9348	0.6750
2	0.3701	0.0459

Table 10: Classification Report for Random Forest

	precision	recall	f1-score	support
0	0.70	0.21	0.33	132
1	0.74	0.93	0.82	890
2	0.68	0.37	0.48	308
accuracy			0.73	1330
macro avg	0.71	0.50	0.54	1330
weighted avg	0.72	0.73	0.69	1330

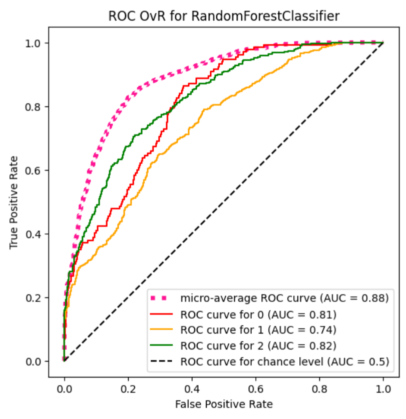

Figure 10: ROC curve for Random Forest model

Gradient Boosting

The Gradient Boosting model achieved an F1 Score of 0.53 on the test set. The confusion matrix for the model is shown in Figure 11. We also generated a classification report and ROC curve for the model, which are shown in Table 12 and Figure 12, respectively.

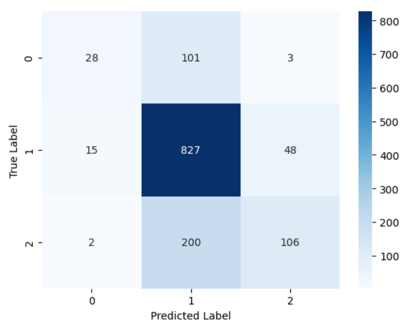

Figure 11: Confusion Matrix for Gradient Boosting Model

Table 11: Gradient Boosting: TPR and FPR for each class

Class	TPR	FPR
0	0.2121	0.0142
1	0.9303	0.6818
2	0.3474	0.0489

Table 12: Classification Report for Gradient Boosting

	precision	recall	f1-score	support
0	0.62	0.21	0.32	132
1	0.73	0.93	0.82	890
2	0.68	0.34	0.46	308
accuracy			0.72	1330
macro avg	0.68	0.50	0.53	1330
weighted avg	0.71	0.72	0.69	1330

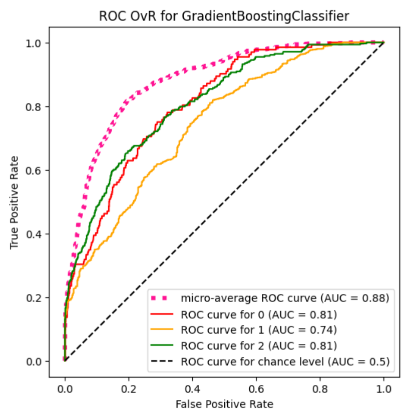

Figure 12: ROC curve for Gradient Boosting model

Discussion

Knowing the class labels are imbalanced, accuracy wasn't a good measure to use and neither was ROC-AUC since the FP rate for highly imbalanced datasets is pulled down due to a large TN rate. So, we decided to primarily look at the F1-macro average score, since we didn't have a preference for one class over the other for our prediction.

Our worst classifier was the GaussianNaiveBayes with an F1-macro average score of 0.34 and our best classifier was the RandomForest with an F1-macro average score of 0.54 after hyperparameter tuning. Even with k-cross-fold validation, our scores didn't change much. These results are slightly underwhelming to receive, as we expected our models to learn the data better. This leads us to speculate it wasn't entirely the models' fault, but rather the data's.

The biggest key contributor to par performance was class imbalance. The nature of the problem came from user ratings being normally distributed around a mean of 5-6. Most models become biased toward the average class, in this case. We also, unfortunately, had a relatively small dataset to work with from our original dataset, due to lots of null values. Information loss causes models to overfit since there is not enough data to capture the true relationships. If we were able to have genre and certificate columns in our dataset, without many null values, or even found a better way to encode more directors and actors for each movie, we might've been able to capture a better relationship, as we believe those would play a vital role in movie ratings.

Feature selection tells us runtime is the most important factor for movie rating, with a positive correlation, however, EDA also tells us 8% of the runtime data are outliers. While 8% is not a lot, it could serve as noise that the models overfit to, which can take down the overall accuracy of the model by a couple of points.

Using CatBoostEncoding may have been one of the best encodings we could have used for a dataset to avoid the curse of dimensionality, but it could also pose other issues. One is because it is using target variable information, there is still a risk of overfitting the data, even if we are encoding the train and test sets separately. Second, the range of the CatBoosted columns is between 0-2. For directors/stars that are extremely popular vs. ones who are not, there is not a lot of numeric room to differentiate between both of them, which could be hard for those categorical variables to contribute significantly towards the movie ratings.

Another thing is the scaling of our features. With year, runtime, and our CatBoosted director/stars, they all have different scales, which can make a model believe one feature is more important than the other, when in reality, we want to treat every feature equally as important. Gradient descent-based algorithms like GradientBoost and Distance-based algorithms like kNN could suffer from different feature scales.

Going back to the models, we see GaussianNaiveBayes performed the poorest, which can indicate our data violates the Naive Bayes assumption of our features being independent of each other, thus underfitting. DecisionTree, GradientBoost, kNN, and LogisticRegression, and RandomForest all had similar F1-macro scores, confusion matrices, and ROC curves (*although this is not a good indicator for imbalanced datasets*), even though they are all learning different decision boundaries, even after hyperparameter tuning. As discussed thoroughly above, there's a good chance all of these models are overfitting to the data, due to noise and the lack of more training examples. It's also possible the dataset is not complex enough to distinguish different learnings by these models, as the features are not strong predictors of the target variable, thus performing similarly. Since RandomForest did 1% percent better than our other models' scores, we can say that RandomForest was able to learn the boundaries slighlty better, due to the nature of the ensemble method, since bagged decision trees can learn a variety of decision boundaries and cancel out random errors from each other. With enough examples and features, it's possible to see RandomForest outperforming the other models, given we fix the data issues described above also.

Despite these limitations, our project provides valuable insights into the potential of machine learning for predicting movie ratings. Our findings suggest that while machine learning models can be slightly better than random guessing in this problem, careful consideration must be given to issues such as data issues and model selection to ensure accurate and reliable results to build an effective and efficient movie rating predictor.

Conclusion

In the film business, predicting movie ratings can be a win-win. Movie makers can pick content liked by the audience and as an audience, we get content that we actually enjoy watching. Film professionals came to make informed decisions about budget, cast, director, and marketing plans for upcoming movies. Our goal was to find a correlation between certain predictors and the rating of the movie, and we find that our models can somewhat predict movie ratings, given an F1 score between 0.3 and 0.6. Our biggest issue which blocked us from getting a more accurate result was the class imbalance in the dataset. Our future work would include working with a stronger and more reliable dataset. We would also like to use features such as certificates and genre, which could potentially play a part in the output. Also, we would like to figure out a way to use all the stars and directors given to us, we will have to play a little bit with alternative encoding methods for this to work. Overall, our project highlights the need for continued research in this area to improve the performance of machine learning models for predicting movie ratings and to identify the factors that contribute to their success or failure.

References

[1] G, C. R. (2023). IMDb Movie Dataset: All Movies by Genre. Retrieved April 15, 2023, from https://www.kaggle.com/datasets/rajugc/imdb-movies-dataset-based-on-genre