Comparing Logistic Regression and Naïve Bayes Algorithms for Movie Rating Predictions

Matthew Lorette Anaya
Dept. of Computer Science
University of Massachusetts Lowell
matthew_loretteanaya@student.uml.edu

***Abstract*— In a world where media consumption is central to many people's lives, this study focuses on comparing the effectiveness of Logistic Regression and Naive Bayes algorithms in predicting movie ratings, a task for which these algorithms are not typically optimized. Working with categorical movie data, the project delves into a meticulous data preprocessing phase to transform a raw movie dataset into a format ready for machine learning application. The study further enhances the capabilities of these algorithms through hyperparameter optimization and the use of encoders and vectorizers. The performance of the models is rigorously evaluated using accuracy scores and confusion matrices, shedding light on the predictive power of each model. The results highlight the substantial impact of data preprocessing and hyperparameter tuning on improving model accuracy, demonstrating the feasibility of applying these machine learning techniques for movie rating prediction, even when using algorithms not inherently suited for the task. The ultimate goal of this study is to incorporate the better performing model into an Android application, showcasing the real-world application of this research.**

I. Introduction

As of 2023, IMDb stands as a colossal storehouse of movie, TV, and celebrity content, hosting an astounding amount of data. The database lists over 7.5 million titles, including episodes, and boasts over 10.4 million personalities, with an engaged user base of 83 million registered users [1]. This massive aggregation of data necessitates the application of effective data analysis techniques to extract valuable insights. A particularly intriguing area of application is the prediction of movie ratings based on a variety of movie features such as title, genre, director, cast, runtime, and movie descriptions.

In this context, the project embarks on a comparative study of two machine learning algorithms, Logistic Regression and Naive Bayes, to predict movie ratings. While these algorithms are not traditionally optimized for handling categorical movie data, this research pushes their boundaries by employing meticulous data preprocessing, hyperparameter optimization, and the application of encoders and vectorizers [2][3]. The performance of these models is evaluated, paving the way to better understand their suitability for such a task.

The ultimate goal of this project is not only to extract insights from the vast IMDb data but also to translate these insights into a practical tool in the form of an Android application. Thus, this project strives to bridge the gap between theoretical machine learning applications and practical real-world usage.

II. Related Works

Several research works have applied machine learning techniques to predict movie ratings. For instance, [4] used different regression techniques, while [5] leveraged collaborative filtering for recommendations. This study differs by employing a comprehensive data preprocessing strategy and using classification algorithms - Logistic Regression and Naive Bayes - to predict ratings.

III. Data Preprocessing

The original dataset comprises various movie attributes, including title, genre, director, cast, runtime, and a brief description. We begin by loading the dataset and removing unnecessary columns, i.e., 'id', 'released_year', and 'movie_link'. I then handle missing data in the 'ratings' column and filter out unusable descriptions. The dataset is then reordered by moving the 'ratings' column to the end. The second step involves creating a balanced dataset to mitigate the potential bias due to class imbalance in ratings. I split the 'ratings' into bins and sample an equal number of instances from each bin. This balanced dataset is then saved for further use. Next, I encode categorical variables, including 'title', 'genre', 'director', 'cast', 'runtime', and 'description', using a Label Encoder, which converts each category to a unique integer. However, for the 'Description' field, I later opt for a

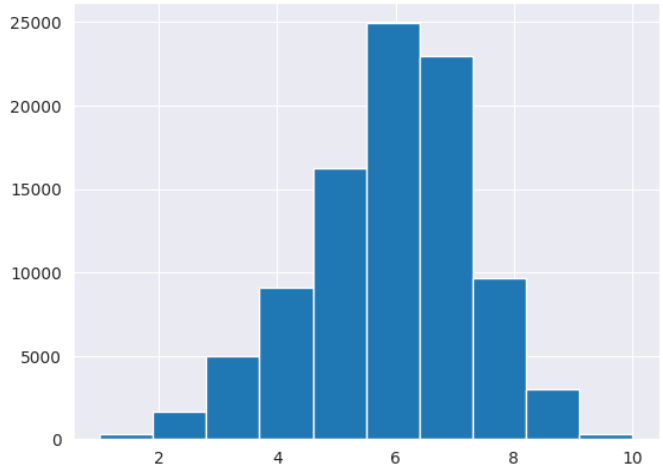

Figure 1: Ratings Data Prior to Balancing

TF-IDF Vectorizer due to its higher performance with text data in hopes of furthering the accuracy of the. models. The TF-IDF Vectorizer converts the 'Description' column into a matrix of TF-IDF features and captures more semantic information than a simple encoding. Though, this method greatly increases the memory needed to service it.

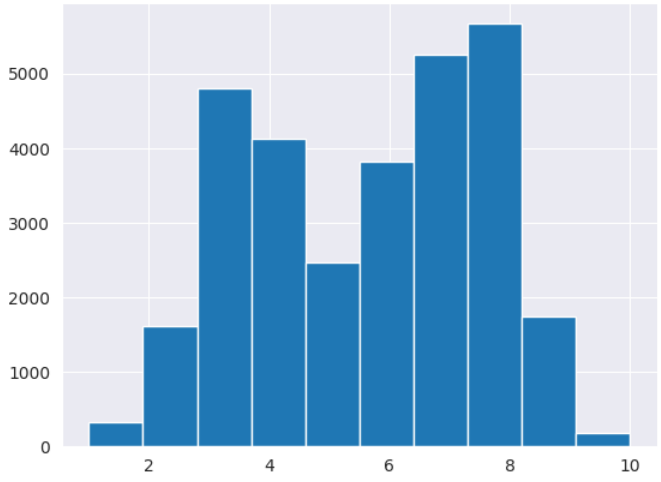

Figure 2: Ratings Data After Balancing

IV. Training Algorithms and Process

The balanced and preprocessed dataset forms the cornerstone of the machine learning models. It's divided into two subsets: a training set to train the models, and a testing set to evaluate their predictive accuracy.

The exploration involves the implementation of two prominent machine learning algorithms: Logistic Regression and Naive Bayes. Logistic Regression, a popular algorithm for binary classification problems, is adapted here for multi-class classification, and Naive Bayes is chosen for its efficiency and simplicity when dealing with high-dimensional datasets.

Both models are trained utilizing the same training set, ensuring consistency in the learning environment. The initial approach involved dividing the dataset without considering the distribution of classes in the training and testing subsets.

After the implementation of the former I also employed stratified sampling during the dataset splitting process. Stratified sampling is a method of sampling that involves dividing a population into homogeneous subgroups known as strata, which ensures that each stratum is adequately represented within the whole sample population[6]. This technique is particularly beneficial in this case, where I aim to ensure that the training and testing datasets are a well-balanced distribution of samples from each bin.

V. Experimental Evaluation

A comprehensive experimental evaluation was conducted to assess the performance of two machine learning models - Logistic Regression and Multinomial Naive Bayes - on the preprocessed and balanced movie dataset. The performance of these models was compared based on their accuracy and confusion matrix [4].

In the initial rounds of model training, a split of the dataset into training and testing sets was performed without stratification. The Logistic Regression model achieved an accuracy of 0.407 while the Naive Bayes model reported an accuracy of only 0.362. The confusion matrices of both models were analyzed to understand the type of errors (false positives and false negatives) made by the models.

Later, the same models were trained with stratified data, ensuring an equal representation of each rating bin in the training and testing sets. The accuracy of the Logistic Regression model improved to 0.425, and the Naive Bayes model's accuracy actually had a decrease of 0.003 to 0.359. The confusion matrices indicated fewer misclassifications, demonstrating the value of stratifying the data.

At this point it was pretty much clear that the better performing of the two was the Logistic Regression model and I made a pivot to pouring the rest of the resources into this previously mentioned model. This led to the hyperparameter tuning on the Logistic Regression model using GridSearchCV. The best parameters were identified as 'C': 0.1 and 'solver': 'newton-cg'. After tuning, the accuracy of the Logistic Regression only improved by 0.003 from the stratified non-optimized parameter model to 0.428.

In the final experiment, the 'Description' column was vectorized using TF-IDF instead of simple label encoding, testing both stratify and non using only the Logistic Regression model. This modification led to a noticeable decrease in accuracy from the stratified encoded and the hyperparameter tuned models at 0.4075. The stratified version of the vectorized description categorial data had about the same accuracy of the stratified encoded model at 0.4255, invalidating the advantage of using TF-IDF for textual data in this specific instance, which is contradictory to[7]. This is also ectremely memory instensive depending on the tools used to do so making it not very suitable for this project as it can make it difficult to integrate into the Android application. In Fig 3 we can see the difference in performance between all tested models.

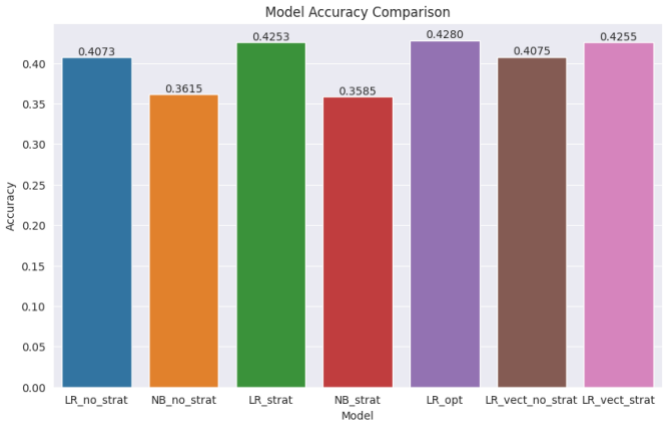

Figure 3: Model Accuracy Comparison

In summary, the experiments revealed that both Logistic Regression and Naive Bayes models can provide valuable

insights into movie rating predictions when properly trained and optimized. Although the raw categorical data posed a significant challenge, even through careful preprocessing, the incorporation of encoders and vectorizers, and hyperparameter optimization, the models achieved less than reasonable performance. The systematic approach employed in this study demonstrates the necessity for the correct model depending on the data being processed.

VI. Android App Model Integration

Incorporating the encoded and stratified version of the Logistic Regression model into the Android app using Android Studio presented several challenges, which were addressed using the Chaquopy plugin. Chaquopy is a plugin for Android Studio that allows seamless integration of Python code into Android applications[8]. By using Chaquopy, it was possible to maintain the integrity of the original Python-based machine learning model without the need for multiple conversions into different formats that might compromise the model's performance and accuracy.

One of the primary challenges faced during the integration process was ensuring compatibility between the Python libraries used for the model and the Android app. Chaquopy made it possible to use the original Python code, along with the required libraries like NumPy, pandas, scikit-learn, and joblib, in the Android application. This eliminated the need to retrain or recreate the model using different tools and ensured that the app's performance was consistent with the original model.

Another challenge was managing the efficient loading of the saved model, encoders, and feature names from the joblib files. These files had to be placed in a location accessible by the Android app. Once the necessary files were placed, the Chaquopy plugin facilitated the loading of these files directly into the app, allowing for the seamless use of the trained Logistic Regression model.

In addition to handling the saved model and encoders, it was also necessary to adapt the user interface and input handling to work with the Android app. This involved creating input fields for the movie features and implementing the necessary code to preprocess the input data, ensuring it was in the correct format for the model to make predictions.

Overall, integrating the encoded and stratified Logistic Regression model into the Android app using Android Studio and Chaquopy proved to be a challenging yet effective solution. This approach preserved the model's integrity while enabling a smooth transition from Python-based development to an Android application, providing valuable insights into movie rating predictions for users on the go.

VII. Conclusion

The project concluded that both Logistic Regression and Multinomial Naive Bayes models performed less than desirable on the movie dataset, with Logistic Regression marginally outperforming the latter. This was to be expected as both these models were not suited for the dataset, which presented the challenge of this project. It was found that preprocessing the dataset by balancing the rating bins, encoding categorical variables, and vectorizing the 'Description' column using TF-IDF only minimally enhanced models' performance. Additionally, hyperparameter tuning using GridSearchCV further improved the accuracy of the Logistic Regression model, but not by much.

The project emphasized that the choice of data preprocessing techniques and machine learning models largely depends on the nature and distribution of the data. The findings underscore the importance of exploratory data analysis and visualization in understanding the dataset and making informed decisions about the machine learning pipeline.

Future work could include exploring other machine learning models like Linear Regression, which would be much better suited given the dataset. Possibly applying different feature selection techniques or using different text vectorization techniques for the 'Description' column. The project also suggested the inclusion of more movie features such as gross earnings, country of origin, or production company for more nuanced predictions.

In summary, the project demonstrated how machine learning can be applied to predict movie ratings, which could be used by streaming platforms to personalize user recommendations or by movie producers to evaluate potential audience reception. The approach and findings of this study could be applicable to other domains where sentiment prediction is important, such as product reviews or social media sentiment analysis.

References

[1] IMDb. (2023). IMDb Statistics. Retrieved from https://www.imdb.com/

[2] S. Raschka. (2021). Python Machine Learning. Packt Publishing.

[3] C. M. Bishop. (2006). Pattern Recognition and Machine Learning. Springer.

[4] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, pp. 426-434.

[5] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in Advances in neural information processing systems, 2008, pp. 1257-1264.

[6] G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning: with Applications in R. New York, NY, USA: Springer, 2013.

[7] S. Robertson, "Understanding Inverse Document Frequency: On theoretical arguments for IDF," Journal of Documentation, vol. 60, no. 5, pp. 503-520, 2004.

[8] Chaquopy. (n.d.). Chaquopy: Python for Android. Chaquopy. Retrieved from https://chaquo.com/chaquopy/