COMP30027: IMDB Movie Rating Prediction (P2)

Keith Menezes (1269040)

1. Introduction

The goal of this project was to build and critically analyse supervised machine learning models for predicting IMDB movie ratings based on various predictor variables. I aimed to develop three machine learning models, including a baseline model, to classify the binned IMDB rating of movies. This report provides a comprehensive overview of the methodology, results, and critical analysis of the models' performance.

2. Methodology

2.1 Data exploration & Pre-Processing

I began by loading and exploring the datasets to understand their structure, the distribution of each feature, and the target variable. The datasets used in this project comprised 3004 training instances and 752 test instances, each with 27 columns (Yueming, 2017). The predictor variables included categorical and numerical features.

2.1.1 Data Cleaning and Imputation

Initial data exploration revealed many values represented as '0' for some variables. Facebook likes for directors appeared to have missing values replaced with 0 values. I believed this to cause some issues for my classifier and skew the results, so the '0' values were imputed with median values in the dataset.

2.1.2 Feature Engineering and Scaling

Feature engineering was a critical part of my pre-processing. We handled categorical variables using different encoding techniques based on their dimensionality and cardinality. The goal was to maximise the data usage whilst managing model complexity and prevent overfitting.

Categorical encoding was used to convert text-based variables into numerical features for my models. Low-dimensional categorical features (language, country, content rating) were one-hot encoded. High-dimensional features (director name, actor names) were frequency encoded to manage dimensionality. The genre column was binary encoded due to relatively low dimensionality and cardinality.

Additionally, numerical features were scaled using MinMaxScaler to standardise the range, which I found particularly effective for models like logistic regression. The normalisation method was decided by analysing the distribution of the features. I wanted to maintain the original feature distributions, so selected MinMaxScaler. New binary features were created for some columns, such as director_facebook_likes. Finally, features like movie title and plot keywords were dropped due to their high dimensionality and tendency to cause overfitting on testing data.

2.2 Model Development

2.2.1 Selecting Models

The model development phase involved selecting, training, and evaluating various machine learning models to predict the target variable: binned IMDB movie ratings. The first model implemented was a ZeroR (0R) model, serving as a baseline by predicting the most frequent class in the training data. This simplistic approach allowed us to establish a performance benchmark.

Next, more sophisticated models were chosen based on their theoretical strengths and practical applicability to the dataset. The selected models included Decision Tree, Random Forest, Logistic Regression, and XGBoost. Each model was trained using the training set and evaluated through cross-validation to ensure robust performance estimates.

The Decision Tree model was chosen for its ability to handle both numerical and categorical data. Random Forest, an ensemble learning method, was selected to leverage its capability to reduce overfitting through the aggregation of multiple decision trees. Logistic

Regression was included due to its simplicity and effectiveness for binary and multi-class classification problems. Lastly, XGBoost, a powerful gradient boosting framework, was incorporated for its training/testing efficiency and superior performance on the actual test data (Brownlee, 2021).

Neural network models and other deep learning techniques were avoided due to initial poor performance on the test data. Based on my research, I believed that the train dataset was too small to effectively gain the benefits that can be obtained from deep learning methods (Montantes, 2024).

2.2.2 Hyperparameter Tuning

Hyperparameter tuning was conducted using the grid search algorithm for Random Forest, Decision Tree, and XGBoost models. This involved searching for the optimal combination of hyperparameters to maximise model performance. The hyperparameters for Random Forest included the number of estimators, maximum depth, and minimum samples split. For Decision Tree, the maximum depth and minimum samples split were tuned. XGBoost parameters such as the number of estimators, learning rate, and maximum depth were optimised.

For the logistic regression model, an iterative approach was used to uncover the best performing parameters for regularisation, number of iterations and the solver selected.

The final hyperparameters selected on the XGBoost classifier resulted in an approximate 3% increase in accuracy on Kaggle test results.

2.2.3 Final Model Selection

Cross-validation results and real test performance on Kaggle dataset guided the final selection of the best-performing model. The model training process was iterative, with continuous refinement based on validation performance and error analysis. This included trying different scaling, feature selection and preprocessing. The final selected model was then trained on the entire training dataset before being used to predict the test set labels.

3. Results

Each model was trained using an 80-20 train-validation split, and their performance was evaluated using cross-validation to ensure robustness. The performance of the selected classifiers was evaluated using accuracy, precision, recall, and F1-score. The results of the various models are summarised in the following tables (see Tables 1-5).

ZeroR (0R) Baseline	
Metrics	Values (%)
Accuracy	62.7%
Precision	N/A
Recall	N/A
F1-Score	N/A

Table 1- Model evaluation results on the ZeroR baseline classifier on the test dataset using the 'most common' strategy.

Logistic Regression	
Metrics	Values (%)
Accuracy	70.0
Precision	27.0
Recall	27.0
F1-Score	26.0

Table 2- Model evaluation results on the Logistic Regression classifier on the test dataset with an 80-20 train-test split.

Decision Tree	
Metrics	Values (%)
Accuracy	69.6
Precision	46.0
Recall	42.0
F1-Score	43.0

Table 3- Model evaluation results on the Decision Tree classifier on the test dataset with an 80-20 train-test split.

Random Forest	
Metrics	Values (%)
Accuracy	74.7
Precision	43.0
Recall	35.0
F1-Score	36.0

Table 4- Model evaluation results on the Random Forest classifier on the test dataset with an 80-20 train-test split.

XGBoost	
Metrics	Values (%)
Accuracy	75.5
Precision	51.0
Recall	46.0
F1-Score	47.0

Table 5- Model evaluation results on the XGBoost classifier on the test dataset with an 80-20 train-test split.

The XGBoost model achieved the highest accuracy of 75.5% (Table 5), followed by the Random Forest model with 74.7% accuracy (Table 4). The Decision Tree and Logistic Regression models performed moderately well but suffered from relatively poor performance on the Kaggle test dataset (60-70% accuracy). The baseline ZeroR model demonstrated the expected lowest performance by predicting the most frequent class.

On Kaggle test data, the XGBoost classifier trained using all train data consistently outperformed all other models. This eventually achieved a top-8 subject result of 75% on the public leaderboard.

Final XGBoost Classifier	
Tuning	Change on Test Performance (%)
Target Encoding	-15
Hyperparameter Tuning	+3
Median Imputation	+2
MinMaxScaler	+2

Table 6- Approximate changes in test accuracy using the XGBoost classifier on the test dataset after various tuning changes. Assumes base XGBoost classifier as benchmark.

Final XGBoost Classifier		
Macro Average Precision = 65% (601 instances)		
IMDB_Score_Bin	Precision (%)	Instances
0	50.0	5
1	60.0	48
2	80.0	377
3	73.0	152
4	63.0	19

Table 7- Model precision results by IMDB Score Bin using the XGBoost classifier on the train dataset with an 80-20 train-test split.

4. Discussion and Critical Analysis

The superior performance of the XGBoost model could be attributed to its ability to handle complex relationships in the data through boosting. Boosting sequentially improves weak learners, resulting in a strong overall model. The hyperparameter tuning further optimised the model, leading to significant performance improvements. XGBoost's effectiveness in managing various data distributions and its robustness to overfitting were evident from the Kaggle results.

Random Forest also performed well due to its ensemble nature, which reduces overfitting by averaging the results of multiple decision trees. This model's robustness to variance makes it a reliable choice for this classification task. However, its performance was slightly lower than XGBoost, possibly due to its inability to capture certain intricate patterns that a boosting algorithm can identify.

The Decision Tree model, while simpler, provided a reasonable performance but was more prone to overfitting, which limited its generalisability. Hyperparameter tuning mitigated this to some extent, but the model still lagged the ensemble method on Kaggle test data.

Logistic Regression showed decent performance, benefiting from the scaling of numerical features. However, its linear nature made it less effective in capturing possible nonlinear relationships present in the data.

The impact of feature engineering and encoding on model performance was impressive. Initial target encoding of categorical variables led to multicollinearity and overfitting. This presented as near 95% train set accuracy and less than 40% test accuracy on the Kaggle set using the same code. This was resolved by switching to frequency encoding and dropping highly correlated features (Table 6). This demonstrated the importance of careful preprocessing to ensure the integrity and effectiveness of classifier models on real data.

Error analysis revealed that misclassified instances often involved movies with borderline ratings, suggesting that these cases were inherently difficult to predict. The models also generally performed poorly on the top and bottom binned IMDB scores. This is likely due to comparatively low incidence of datapoints in the train dataset for the 0 and 4 bins (Table 7).

5. Conclusion

In conclusion, this project successfully demonstrated an approach to the process of building and evaluating supervised machine learning models for predicting IMDB movie ratings. The XGBoost model emerged as the best performer, showcasing the power of boosting algorithms in handling complex datasets. The importance of thorough data preprocessing, careful feature engineering, and hyperparameter tuning was highlighted throughout the project.

My findings highlight the significance of iterative testing and error analysis in developing robust machine learning models. Future work could explore additional data sources, more sophisticated feature engineering techniques, and advanced algorithms to further enhance classifier performance.

6. References

Brownlee, J. (2021, February 16). *A gentle introduction to XGBoost for applied machine learning*. MachineLearningMastery.com. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Montantes, J. (2024, April 19). *3 reasons to use random forest over a neural network–comparing machine learning versus deep...* Medium. https://towardsdatascience.com/3-reasons-to-use-random-forest-over-a-neural-network-comparing-machine-learning-versus-deep-f9d65a154d89

Yueming. (2017, December 16). *IMDB 5000 movie dataset*. Kaggle. https://www.kaggle.com/datasets/carolzhangdc/imdb-5000-movie-dataset