COMP30027 Machine Learning Assignment 2 Report

(Word count: 1778)

1. Introduction

In this report, experiments were carried out to explore the application of different machine learning models for predicting the rating of movies based on various predictor variables, as well as how different methods of feature engineering and selection can affect the machine learning models' performance.

In particular, this report will investigate Entropy Decision Trees and its embedding variants such as Decision Trees with AdaBoost and Random Forests in predicting ratings using IMDB movie dataset. Throughout the study, the methodology used in this task will be evaluated, including the impact of different feature representations and selection methods on model performance. In particular, the additional text-based features, like those generated by Doc2Vec or FastText features, will be examined to determine their potential contribution to the model accuracy and compared to those generated manually and by One-Hot Encoding. Additionally, the report will explore how AdaBoost and Random Forest work and contribute to more accurate predictions.

2. Methodology

2.1 Machine Learning Algorithms

The first learner is ZeroR, predicting using the most frequent label.

The second one is a Decision Tree. Decision Trees learn optimized if-else rules from features to predict the target variable. Since decision trees struggle with irrelevant features and are prone to overfitting, the task of optimizing the maximum depth is essential. In the end, max_depth was set to 11.

Decision Trees alone are often considered weak learners, however, with AdaBoost, the Decision Tree performance can significantly increase. AdaBoost iteratively changes the training instance distribution based on previous classifier performance, boosting Decision Trees. Similar to Decision Trees, the hyperparameters were carefully considered. The final model was tuned with n_estimators = 20 and learning_rate = 0.1 from GridSearchCV.

The final learner is Random Forest. Many Random Trees are trained using different samples and features from the training dataset. In the final model, n_estimators = 10 and max_depth = None.

2.2 Dataset Cleaning

There is only 1 None value in the language attribute. In addition, I figured out that there were 18 duplicates in the dataset, so I removed them.

2.3 Feature Engineering

The first method is One-Hot Encoding. This method converts each unique value of a feature into a column. When it is applied to the test dataset, it ignores any new values in the data.

The second method is using given additional features: CountVectorizer on director and actor names, Doc2Vec on plot_keywords and genrers, FastText title_embeddings[1].

The last method is manually creating new features. genres and plot_keywords attributes were tokenized by splitting on "|" and converted into columns. facebook_likes from actor 2 and actor 3 were added to create a new feature and cast_total_facebook_likes was removed. Similarly, num_critic_for_reviews was divided by num_user_for_reviews. is_english feature was generated to indicate whether a movie was in English and converted country values to USA, UK, and others. For content_rating, the old ratings were updated to

[1] For more information, visit the documentations, "Get Started · fastText," accessed May 15, 2024, https://fasttext.cc/index.html; "Sklearn.Feature_extraction.Text.CountVectorizer," scikit-learn, accessed May 15, 2024, https://scikit-learn/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html; "Gensim: Topic Modelling for Humans," accessed May 15, 2024, https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html.

align with current standards. M was changed to GP then to PG, and X was replaced by NC-17.

2.4 Evaluation Methods

First, I split the dataset into 67% training data and 33% test data. I used both 10-fold cross-validation on the training set and used a held-out validation set (20% training data). To compare models, I used confusion matrix, accuracy, macro recall and macro precision score.

3. Result

The results were produced by the models after being trained on the training set and tested against the test set.

3.1 ZeroR

Table 1 reports the evaluation scores of ZeroR.

Metrics	Score
Accuracy	0.60
Precision	0.20
Recall	0.12

Table 1- Evaluation scores of ZeroR.

3.2 Decision Tree (max_depth = 11)

Table 2 shows the evaluation scores of the Decision Tree model.

Metrics	Score
Accuracy	0.68
Precision	0.54
Recall	0.49

Table 2- Evaluation scores of Entropy Decision Tree.

Figure 3 displays the confusion matrix of the model.

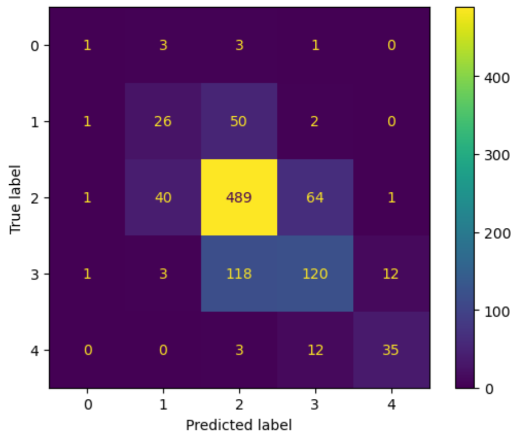

Figure 3- Confusion matrix of Decision Tree tested on 33% data.

3.3 Decision Tree (max_depth=11) with AdaBoost (learning_rate=0.1, n_estimators=20)

Table 3 presents 3 evaluation metrics of the model.

Metrics	Score
Accuracy	0.71
Precision	0.75
Recall	0.47

Table 3- Evaluation scores of AdaBoost Decision Tree.

Figure 4 displays the confusion matrix of Decision Tree boosted by AdaBoost algorithm.

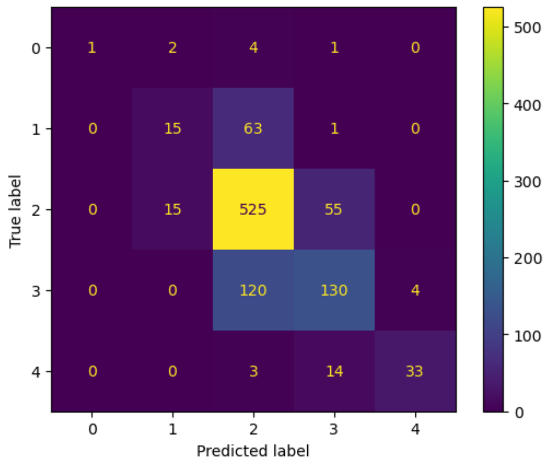

Figure 4- Confusion matrix of Decision Tree with AdaBoost tested on 33% data.

3.4 Random Forest (max_depth=None, n_estimators=10)

Evaluation scores are illustrated in Table 4, while the confusion matrix of the Random Forest is shown in Figure 5.

Metrics	Score
Accuracy	0.66
Precision	0.58
Recall	0.27

Table 4- Evaluation scores of Random Forest

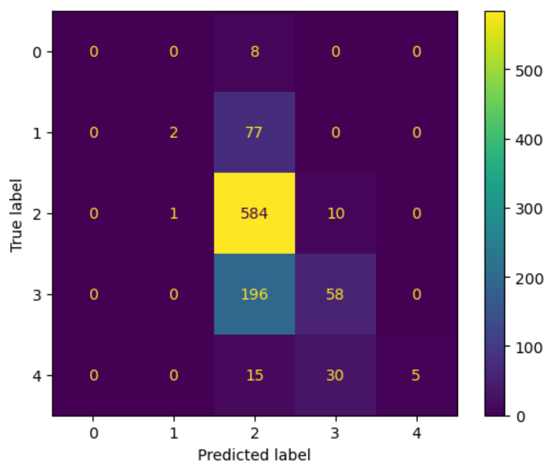

Figure 5- Confusion matrix of Random Forest tested on 33% data.

4. Discussion and Critical Analysis

4.1 Evaluation Strategies

Cross-validation was chosen as it is more powerful and provides a more robust estimate of generalization performance. However, as Decision Trees and its variants tend to be overfitting, it is important to be able to test the model with new data after cross-validation[2]. Therefore, I both applied cross-validation on the training set and estimated the generalization error of final models by retraining the models with the whole held-out training dataset and testing against the held-out test set to prevent models from data leakage and have a better view of the ability of models in generalization.

4.2 Feature Engineering Impacts

In this session, the impact of the additional features and why One-Hot Encoding and manually engineered features were selected will be discussed.

4.2.1 Impacts of Manual Feature Engineering

Most of categorical features are highly branching nominal attributes, also there are some highly correlated numerical features from Figure 6.

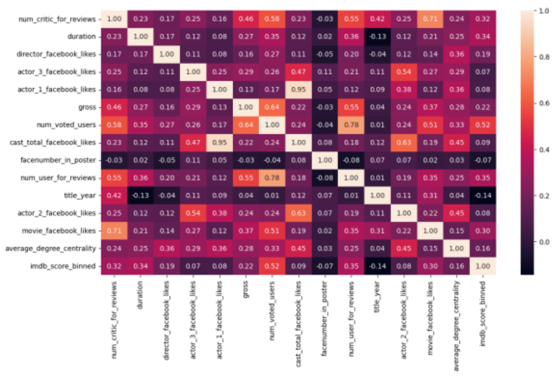

Figure 6- Correlation matrix for numerical features.

This contributes to overfitting and more irrelevant features. Hence, manually generating features and removing highly correlated attributes were carried out. An example is that creating an indicator for movies in English and reducing the values of country reduced the risk of sparse data. A reason for not preferring One-Hot Encoding is that each value of these attributes represents a combination of words, resulting in over 2000 new columns by One-Hot Encoding. Tokenization preserves all information while effectively reducing the dataset's dimensions.

4.2.2 Impacts of Additional Features

To assess additional features' benefits, four Decision Trees without optimal hyperparameter was trained, mainly to see whether adding new features improves the Decision Tree. The scores of cross-validations of the untuned one-hot model are shown in Table 5.

Metrics	Features
Accuracy	0.64
Precision	0.43
Recall	0.42

Table 5- Cross-validation scores of untuned Decision Tree with one-hot encoding.

Then 3 more models were trained, each replacing One-Hot columns with the corresponding additional ones. Table 6 demonstrates the scores when CountVectorize features were integrated.

Metrics	Score
Accuracy	0.604
Precision	0.388
Recall	0.373

Table 6- Cross-validation scores of Decision Tree trained with CountVectorize features.

[2] Adrian Tam, "Training-Validation-Test Split and Cross-Validation Done Right," *MachineLearningMastery.Com* (blog), September 22, 2021, https://machinelearningmastery.com/training-validation-test-split-and-cross-validation-done-right/.

Since the technique is tokenizing actor names by splitting on spaces. They may not be helpful because to identify a director or actor influencing the movie ratings, a whole name is required instead of just a token.

The next two models were trained with Doc2Vec genres, plot_keywords features, and FastText title_embeddings. These features led to performance degradation. Table 7 illustrates the mean accuracy score of a Decision Tree trained with Doc2Vec features.

Metric	Score
Accuracy	0.565
Precision	0.352
Recall	0.358

Table 7- Cross-validation scores of Decision Tree trained with Doc2Vec features.

Some reasons might be the features are not compatible with Decision Trees. Decision Trees select one attribute to split on at each level, which may not effectively capture all dimensions in Doc2Vec necessary for meaningful information. Since points close to each other in the Doc2Vec embedding space share more similar meanings, each individual dimension in Doc2Vec might not be useful when considered separately. The same reasons can be applied to FastText title_embeddings.

To further validate removing these features, Figure 7 displays the confusion matrix of AdaBoost Decision Tree trained with additional features, supporting this decision, as the model's scores were lower compared to those without additional features.

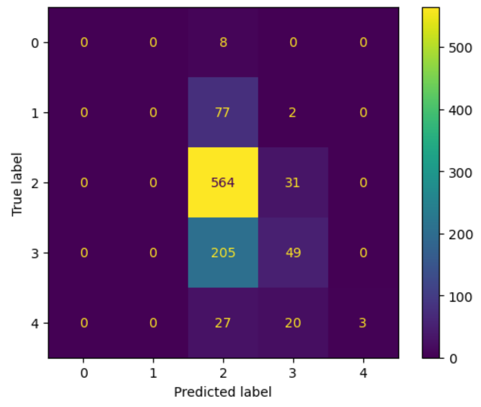

Figure 7- Confusion matrix of AdaBoost Decision Tree trained on held-out training set with all additional features.

4.3 Tuning models

By using One-Hot Encoding, the number of attributes can exceed 10000. It is essential to find an optimal hyperparameters for our models.

For Decision Tree, considering computer resources when many values for max_depth were considered, I split the training set to get a validation set first.

Max_depth	Accuracy	F1 score
3	0.71	0.34
11	0.64	0.42
20	0.63	0.41
100	0.62	0.41
1000	0.62	0.41
None	0.62	0.41

Table 8- Accuracy of Decision Trees with max_depth from 3 to 1000.

The model performance stabilized around max_depth = 20 as can be seen in Table 8, which may indicate most features were not informative, and the tree might be overfitting. After narrowing down potential values, I selected the max_depth = 11 because it generalized well across labels and had high cross-validation accuracy (0.668). Compared to max_depth = 10, despite the higher accuracy (0.673), the model made more errors on label 1 and 4 as depicted in Figure 8, so had a lower recall and precision score, therefore, not generalized as well as max_depth = 11 in Figure 9.

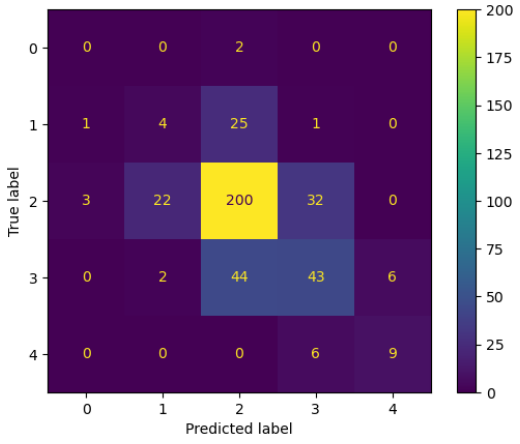

Figure 8- Confusion matrix of a Decision Tree with max_depth = 10 tested on validation set.

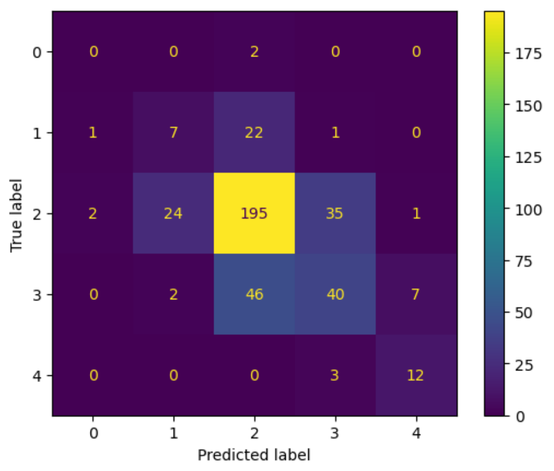

Figure 9- Confusion matrix of a Decision Tree with max_depth = 11 tested on validation set.

For AdaBoost and Random Forest, there is more than one hyperparameter to be optimized, therefore, GridSearchCV was used. Unfortunately, due to the long run time, I could not choose a high number of n_estimators for both AdaBoost and Random Forest.

4.4 Model Selection

In the end, I chose AdaBoost Decision Tree to make predictions on unseen dataset and received an accuracy around 0.70. According to the accuracy of the held-out test set, I would rate the performance of each model as follows: AdaBoost DT > DT > Random Forest > ZeroR.

To know the reasons behind errors in the Decision Tree, the top 3 most contributing features of the DT including num_voted_users, Drama and other_actors_facebook_likes are investigated. According to Figure 10, the distributions of these features in the training set for instances labelled 3, 2 and 1 are quite similar. It could contribute to more errors in predicting these labels.

Although AdaBoost made more mistakes in predicting label 1, it made lots more correct predictions in label 2 and 3, as can be seen in Figures 3 and 4. The recall score of AdaBoost DT is lower than DT, but the accuracy and precision are higher. These behaviours can be explained by AdaBoost algorithm. Wrong-predicted instances have higher weight, so the model pays more attention to them in the next iteration. Furthermore, the dataset is imbalanced, there are more label 2 and 3 in the dataset. As a result, although the Decision Tree made wrong predictions on every label, AdaBoost might be more focused on improving the prediction accuracy of label 2 and 3, so reduce the accuracy of label 1 and 4.

The Random Forest performs quite badly, the accuracy score is lower than Decision Tree. According to Figure 5, the correct predictions mostly came from label 2, which implies the Random Forest seems to not generalize well. This may be caused by the dataset. Random Forest picks features randomly, however, there are quite many irrelevant features in this dataset, which may cause some random trees constructed with uninformative attributes. Furthermore, the dataset is imbalanced, so random trees with irrelevant attributes may tend to predict instances as label 2. Considering a model with max_depth = 20, it has even lower accuracy and mostly chose 2 as predicting classes according to Figure 11, which implies that Decision Trees were built using most of uninformative features. That is the reason why in the final model, max_depth was set as default.

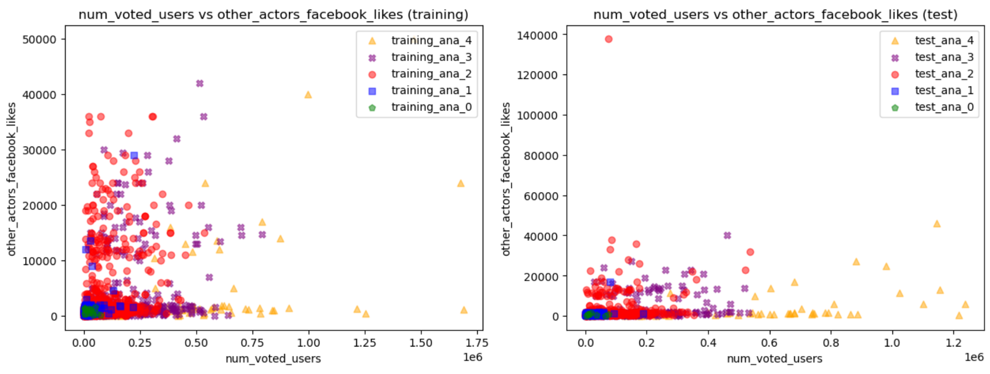

Figure 10- the distribution of num_voted_users and other_actors_facebook_likes in training set and test set.

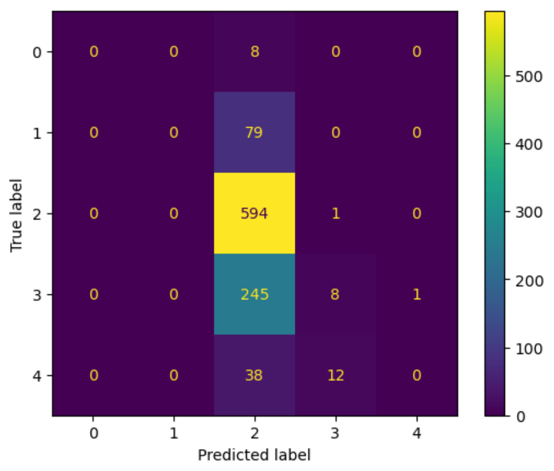

Figure 11- Confusion matrix of a Random Forest with max_depth = 20 tested on validation set.

5. Conclusions

To conclude, due to many highly branching categorical attributes in the IMDB movie dataset, the most difficult task was doing feature engineering and optimizing hyperparameters. The strengths and weaknesses of the models were identified. ZeroR provides a simple baseline, while Decision Trees are easily used and interpreted on any kind of dataset, it is a weak learner and risk overfitting. AdaBoost aims to boost Decision Tree performance but requires careful hyperparameter tuning. Random Forests, although robust to overfitting, may have faced challenges with this specific dataset. There are some more additional steps that I could have done to improve all model performance such as doing feature selection before tuning the model.

6. References

"Gensim: Topic Modelling for Humans." Accessed May 15, 2024. https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html.

"Get Started · fastText." Accessed May 15, 2024. https://fasttext.cc/index.html.

scikit-learn. "Sklearn.Feature_extraction.Text.CountVectorizer." Accessed May 15, 2024. https://scikit-learn/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

Tam, Adrian. "Training-Validation-Test Split and Cross-Validation Done Right." *MachineLearningMastery.Com* (blog), September 22, 2021. https://machinelearningmastery.com/training-validation-test-split-and-cross-validation-done-right/.