Machine Learning IMDB Movie Classification:

1. Introduction

The aim of this task is to engineer a machine learning classifier which can predict the rating of movies on IMDB.

After first applying feature engineering, feature selection and feature creation techniques, three different combination classifiers will be created with optimal hyperparameters found through tuning. Their classification performance will subsequently be critically evaluated.

It was found that whilst the three classifiers all had an accuracy of 0.60 or greater after cross-validation and could reliably predict the quality of more than half of the movies in the test set, their respective training methods and classification behaviour impacted their ability to make more reliable predictions.

2. Methodology

Before classification, feature engineering, feature selection and feature combination occurred to pre-process the data and select useful features for prediction.

2.1 Feature Engineering

2.1.1 One-Hot Encoding

One-hot encoding was applied to the "content_rating", "language", and "country" categorical features so they could be analysed later by feature selection and classification algorithms later.

2.1.2 Standardisation

The original dataset had several numerical features (e.g. "num_voted_users", "cast_total_facebook_likes") which were not on the same scale. It was hence decided that all numerical features in the dataset would be standardised to have a mean of 0 and a standard deviation of 1 to reduce the performance impact that unscaled features might have on some classifiers.

2.2 Feature Evaluation/Selection

To evaluate the correlation of each feature in the dataset to movie ratings, the Mutual Information (MI) score of each feature was calculated.

During this stage, it was discovered that several of the continuous features, such as "num_voted_users" had the highest MI scores with scores of approximately 0.177 (correlation is shown in Figure 1). It was also noticed that out of all the categorical features, "genre" and "title_embeddings" were also among the higher correlated features, with scores of 0.049 and 0.022 respectively.

However, it was noticed that most features in the dataset had low MI scores which would have limited their usefulness in classification as their correlation is low.

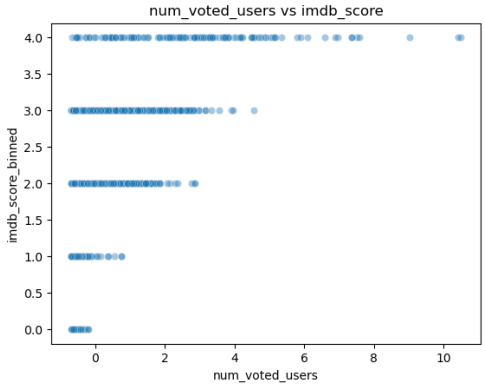

Figure 1 – Scatter plot of standardised num_voted_users vs imdb_score, showing a correlation.

2.3 Feature Combination

To try and overcome the issue of features having low MI scores, SKLearn's PolynomialFeatures library was used to create new features which are polynomial combinations of the existing features. New features were created which included up to degree-6 combinations of the six-highest MI features of a particular set of the testing data. Degree-6 was selected as degrees larger than this took too long to combine.

The set of original continuous features along with the pre-processed "genre" and "title_embedding" feature arrays had the highest average MI scores previously, so these sets were selected for the Feature Combination stage.

After creating feature combinations, many new features with equivalent or higher MI scores than those in the original dataset were produced (e.g. "director_facebook_likes^2 * num_voted_users" was a feature used in

classification).

In the end, 35 features with the highest MI scores were selected to ensure the most useful classifiers were used for predictions. The top 20 polynomial features from the original continuous dataset, the top 10 polynomial features from the "genre" dataset and the top 5 polynomial features from the "title_embedding" dataset were selected.

2.4 Classifier Creation

Three unique classifiers, fitted to the 35 features mentioned above, were created as part of this investigation. An AdaBoost Classifier (ABC), a Random Forest Classifier (RFC) as well as a Bagging Classifier (BC) were extensively tuned and evaluated. These three classifiers were selected as the predictions from combination classifiers are regarded to be as good, if not better, than the predictions from the best base classifiers themselves. They also use multiple base classifiers, meaning multiple classifiers are used in delivering predictions instead of a sole classifier.

3. Results of Classifiers

3.1 Hyperparameter Tuning

To improve the performance of the classifiers, the hyperparameters were tuned and optimised through iterations of 10-fold cross validation.

For the AdaBoost Classifier, the learning rate and number of base Decision Tree estimators was tuned, and a learning rate of 2.0 with 25 estimators was found to be the most accurate with an average score of 0.600. Compared to the other two classifiers, it was noticed that the accuracy of ABC greatly varied depending on the hyperparameters. For example, an ABC classifier with a learning rate of 2.25 and 75 estimators had an accuracy of 0.144, whilst the same model which decreased the learning rate by 0.25 had a much higher accuracy of 0.573. This indicates that depending on the learning rate (as well as the number of base estimators), the base Decision Trees are more likely to overfit and higher weights could be assigned to trees which learn features which do not generalise well to test sets.

For the Random Forest Classifier, the number of random trees in the forest was tuned, with 75 found to be the optimal number with an accuracy of 0.693. The Bagging Classifier had the base classifier tuned (between Decision Tree, SVM and Gaussian Naïve Bayes) as well as the number of estimators. An SVM base classifier with 40 estimators was found to be the most accurate at 0.695, however all models tested had an accuracy score of at least 0.689.

Classifier	Hyperparameters	Acc.
ABC	2.0 learning rate 25 estimators	0.600
RFC	75 random trees	0.693
BC	SVM base 40 estimators	0.695

Table 1 - Average accuracy of the three final classifiers after 10-fold cross validation.

3.2 Performance Evaluation

A confusion matrix was produced for each of the three classifiers listed in Table 1. The training data was separated into an 80-20 train-test split for the purposes of creating the confusion matrix. The results shown by the confusion matrices will be analysed in the next section.

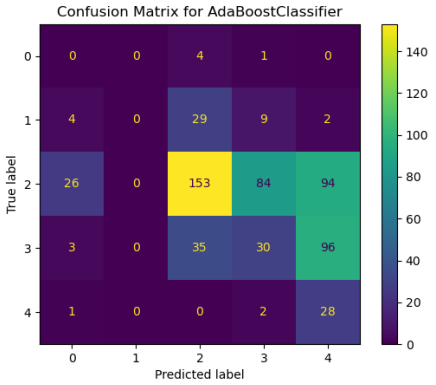

Figure 2 - Confusion Matrix for the AdaBoost Classifier with 25 estimators and 2.0 learning rate.

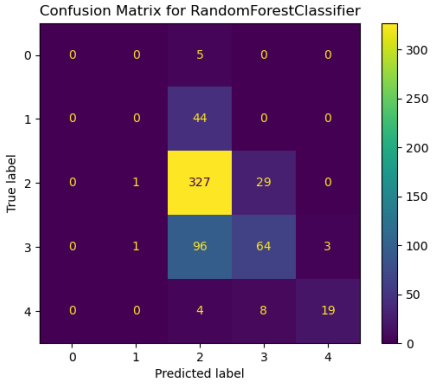

Figure 3 - Confusion Matrix for the Random Forest Classifier with 75 random trees.

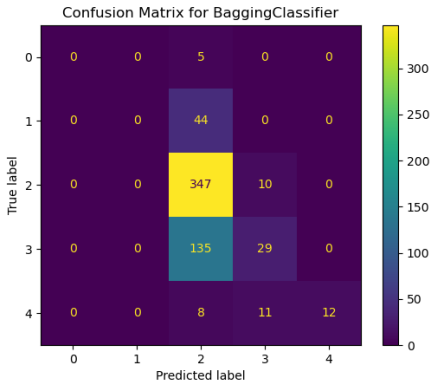

Figure 4 - Confusion Matrix for the Bagging Classifier using SVM as a base classifier and with 40 estimators.

4. Discussion and Critical Analysis

Whilst the three classifiers had average accuracy scores above 50%, they still had some issues classifying movie ratings correctly. These issues are explored in more detail below.

4.1 Uneven Class Distribution

Observing the confusion matrices shown above, the "2" quality label is the most common in this dataset, with over 350 instances being included in the 20% testing split used for confusion matrix creation. All three classifiers are seen to incorrectly predict minority class instances as being of quality "2". This demonstrates that the classifiers, especially BC and RFC, have generalised to the "2" label more than the other classes as there are more instances, and hence have a bias towards it.

For instance, none of the three classifiers predicted the 5 "0" instances correctly, instead classifying them as "2" (or "2" or "3" for AdaBoost as shown in Figure 2).

4.2 Differing Bias and Variance

As seen in Figure 4, the Bagging Classifier appears to have the highest bias towards the "2" label. Compared to the Random Forest Classifier in Figure 3 which correctly detects approximately 39% of the true "3" instances and 61% of the true "4" instances, the Bagging Classifier only detects about 18% and 38% of these true instances correctly respectively. As a result, the Bagging Classifier has a significantly lower recall than the Random Forest Classifier, however both classifiers have very low variance in their predictions. This is explored in more detail in 4.3 below.

Observing the AdaBoost Classifier's confusion matrix in Figure 2, it has a lower bias towards the "2" label compared to other classifiers. It instead has higher variance compared to the other classifiers; it misclassifies over half of the true "2" instances as being either "0", "3" or "4". This is explored more in section 4.4.

4.3 Bagging Classifier Lower Recall

The Bagging Classifier was shown to have lower precision than the Random Forest Classifier, which can be linked to the different methods which both models use to make predictions.

The bagging classifier used in this task took 40 bootstrapped samples of the dataset and trained 40 different SVM models on the data. By comparison, the random forest classifier constructed 75 random trees. In these trees, only some of the attributes for each instance were considered at each split, meaning that each tree considered different attributes and could hence capture less prominent aspects of the data. This can help avoid overfitting as the trees could more easily learn features which could distinguish some minority classes from more prominent ones, allowing it to identify minority classes more effectively.

4.4 AdaBoost Lower Precision

In comparison to the other two classifiers, the AdaBoost Classifier makes very different predictions. As can be seen in Figure 2, the AdaBoost Classifier has a much lower precision compared to the other classifiers. For example, for the "4" label, BC has a precision of 1, RFC has a precision of approximately 0.86 whilst ABC has a precision of approximately 0.13.

This can be linked to ABC's use of weighted voting to predict. During training, AdaBoost assigns higher sampling weights to instances which it predicts incorrectly to focus on classifying them correctly in future iterations; assigning lower weights to base classifiers which make incorrect predictions in the process. As shown in Figure 2, the minority classes (e.g. "0", "3" and "4" labels) have been more extensively trained on by ABC, meaning that trees which predict well for "0", "3" and "4" are likely weighted higher, whilst the trees which more accurately classify "2" labels do not have as high of a sampling weight in the weighted voting.

As a result, the AdaBoost Classifier has overfitted to these labels and has hence

predicted that these labels occur more frequently than they do in the test set, leading to lower precision.

5. Conclusions

Whilst all models were evaluated to provide correct predictions to over half of the dataset, each model had several issues which prevented it from delivering more accurate predictions. The class distribution was uneven, the RFC and BC models had a high bias towards the majority class, whilst the ABC model's use of weighted voting meant that it assigned higher weights to trees which predicted minority classes better.

Overall, the RFC model was determined to be the better and more-balanced model out of the three, with it having a higher recall than the BC model and a higher precision than the ABC model meaning it performed well on the majority and minority classes in comparison.

In future, multiple improvements, such as bootstrapping or resampling the original data multiple times to oversample minority classes, as well as implementing some more complex classifier algorithms such as Neural Networks could help deliver more accurate models.

6. References

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.