IMDB Movie Rating

Duc Huy Le

1 Introduction

Since the 20th century, movies have been an essential and important kind of entertainment to human beings. It is one of the major sources of commerce and marketing, not only movie making companies care about the success of movies but also general people and business. Social media is primarily the place for people to share their insight about the movies they just watch. Therefore, analysis of social media data about movie ratings has gotten attention from the Machine Learning department in recent years, and IMDB (Internet Movie Database) which is the most popular online platform for people to share their rating about movies is the perfect candidate for movie rating prediction. The movie dataset is derived from the IMDB 5000 Movie Dataset2, focusing on movie rating prediction based on various predictors associated with movies. This research will discuss the procedures of conducting movie rating analysis, which include preprocessing, choosing suitable Machine Learning models, hyperparameter tuning and applying those models to the dataset, and analyze the result to gain insights of the data and training processes.

2 Methodology

2.1 Preprocess - Feature Engineering

The dataset contains some nominal data columns include director and actor name, genres, etc. Machine learning model require numeric input to perform classification task. Thus, we implement Count Vectorizer and One Hot Encoding to handle such data.

2.1.1 Label Encoding

There are tremendous amount of distinct value for nominal features in the dataset, we choose to label encode 'director_name', 'actor_1_name', 'actor_2_name', 'actor_3_name', 'movie_title' for efficiency and preventing the increasing in dimension of the dataset. As the amount of distinct people's names is vast, encountering a name in testing data that is not present in the training data is inevitable. In order to solve this problem, we decided to fit the label encoded model with both training dataset and test dataset, however, we are risking that our model will be overfitting as the model may learn patterns specific to the test data rather than generalise patterns.

2.1.2 One Hot Encoding

Even though the amount of distinct value for genres, content rating, language, and country is also large, they are considerably small compared to other nominal features. We decided to do One Hot Encoding for these features to extract valuable information about the relationship between them and labels. It allows us to represent these categorical variables in a way that preserves their distinctiveness without introducing a hierarchical relationship, thereby enhancing the model's ability to learn from these features.

2.1.3 Count Vectorizer

Since plot is the core of every movie, we want to investigate the effect of 'plot_keywords' on our model. We chose to Count Vectorized the 'plot_keywords' features, however, it dramatically increases our dataset dimension. Thus, before we concatenate it to our dataset, we performed Principal Component Analysis (PCA) to reduce the dimensionality while retaining as much as variability of the dataset. This step allowed us to incorporate the essential information from the 'plot_keywords' into our dataset without significantly increasing its complexity.

2.2 Classification Model

2.2.1 Zero R - baseline

Zero R is simply choosing the classifier that appears the most in the training dataset and default to that whenever a testing instance is presented. Due to its simplicity and naive nature, we will use Zero R as our baseline for evaluating the performance of other models.

2.2.2 Random Forest

Since we have a diverse set of features which contains non-linear relationships we decided to use Random Forest as the first method of classifying movie rating. The preprocessed dataset using Label Encode and One Hot Encode increase the risk of model overfitting and increase the dimensionality of the dataset, both problems can be handled well by Random Forest. Random Forest is well-suited to manage both issues effectively, making it an ideal choice for our classification task.

2.2.3 Bagging with Decision Tree

Bagging (Bootstrap Aggregating) with Decision Tree was chosen as a second classifier. It works by training multiple models on different subsets of the training data then averaging all the predictions. Bagging reduces the risk of overfitting which is increased by the process of label encoding and one hot encoding. By averaging the predictions of multiple models, Bagging stabilizes the output and improves generalization, making it a robust choice for our classification task. Additionally, Bagging can handle high variance in the data effectively, further enhancing the model's performance and reliability.

2.2.4 Fine Tuning

To optimize the performance of the model, fine tuning is performed using grid search with cross-validation. Fine-tuning is the process of finding the optimal value for the hyper- parameter to ensure optimal condition of the training of the model. We decided to perform fine tuning using the Random Forest classifier and applied the same result for Bagging with Decision Tree because of the long run time. The following hyperparameters were tuned:

- Number of Trees (n_estimators): the number of tree in the forest
- Maximum depth (max_depth): the maximum depth of each tree
- Minimum sample split (min_samples_split): the minimum number of sample required to split an internal node
- Minimum sample leaf (min_samples_leaf): the minimum number of samples to be at leaf node

2.3 Evaluation

We will use k-fold cross validation to our model to test the model performance consistency across different subset of data, and evaluate based on the metrics below:

- Accuracy
- Precision
- Recall
- F1 score

3 Result

3.1 Zero R

Guessing all instances to have a rating of 2 yield a classifying accuracy of 0.60, this result indicates the dataset likely has a class imbalance and is not likely to be generalized. Class imbalance negatively affects the machine learning model and makes it harder to correctly classify the minority classes.

3.2 Random Forest

After applying Random Forest on the processed dataset, we have the results below, for conciseness, precision, recall and f1-score will be labeled as Pre, Rec and F1, respectively:

Class	Precision	Recall	F1 Score
0	0.0000	0.0000	0.0000
1	0.3000	0.0208	0.0382
2	0.7337	0.9456	0.8262
3	0.6902	0.5058	0.5830
4	0.9240	0.4032	0.5431

Average accuracy: 0.7287

Table 1: Evaluation metric of each label with Random Forest after data preprocessing

Label	Precision	Recall	F1 Score
0	0.0000	0.0000	0.0000
1	0.0000	0.0000	0.0000
2	0.7323	0.9494	0.8267
3	0.6884	0.4993	0.5769
4	0.9636	0.3788	0.5212

Average accuracy: 0.7267

Table 2: Evaluation metric of each label with Random Forest after data preprocessing and fine tuning

3.3 Bagging Decision Tree

Class	Precision	Recall	F1 Score
0	0.0000	0.0000	0.0000
1	0.5509	0.2022	0.2363
2	0.7614	0.8635	0.8056
3	0.6706	0.6048	0.6317
4	0.8592	0.5885	0.6901

Average accuracy = 0.7321

Table 3: Evaluation metric of each label with Bagging Decision Tree after data preprocessing

Label	Precision	Recall	F1 Score
0	0.0000	0.0000	0.0000
1	0.5526	0.1895	0.2224
2	0.7591	0.8532	0.7988
3	0.6619	0.6112	0.6305
4	0.8524	0.5731	0.6749

Average accuracy: 0.7200

Table 4: Evaluation metric of each label with Bagging Decision Tree after data preprocessing and fine tuning

3.4 Synthetic Minority Over-sampling Technique - SMOTE

Since the result of Zero R shows that the label '2' dominates the dataset, we considered adding an extra step which is SMOTE to address the class imbalance problem. SMOTE works by generating synthetic samples for the minority class to balance the class distribution. Below are result of Random Forest and Bagging Decision Tree on balanced dataset

Label	Precision	Recall	F1 Score
0	0.9898	0.9984	0.9941
1	0.8355	0.9163	0.8724
2	0.7633	0.5470	0.5945
3	0.7197	0.7743	0.7372
4	0.9549	0.9815	0.9679

Average accuracy: 0.8435

Table 5: Random Forest on resample dataset

Label	Precision	Recall	F1 Score
0	0.9644	0.9962	0.9800
1	0.7468	0.9081	0.8193
2	0.4698	0.2012	0.2707
3	0.5930	0.7852	0.6736
4	0.9636	0.9766	0.9700

Average accuracy: 0.7735

Table 6: Bagging Decision Tree on resample dataset

4 Discussion

4.1 Fine tuning

Before we performed fine tuning to find the best hyperparameter for the 2 models, we constructed a manual k-fold cross validation tuning. We observe that 3 fold-cross validation typically performed worse than 10 fold does, however, the difference is not significant. A small fold number offers quicker training time, larger fold can even out extreme cases.

As the number of folds increases the training set gets larger, which helps to smooth out the effect of outliers and provide more reliable estimation. However, if the number of folds is too large, it can significantly increase the training time, and leave fewer instances in the test set to evaluate the model's performance effectively. Therefore, there is a tradeoff between time and robustness of the model evaluation when choosing the number of folds for cross-validation.

Both our models' performance is slightly increased by the fine tuning process using grid search. Performance increase is primarily due to increase in the number of trees in the forest which is computationally intensive. We observed that the trade off between computational efficiency and accuracy was not favorable, the marginal gains in accuracy did not justify the substantial increase in computational resources required.

4.2 Random Forest

We can see that the model fails to predict the rating of '0' and '1' in the dataset, as shown in the result the precision of label '0' and '1' are 0. Part of the reason is the lack of instances in the training dataset; there are only 193 instances of '1' and 21 instances of '0'. This indicates that the model is biased toward the majority class and fails to predict minority classes accurately. After applying SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset, the overall accuracy of the model improved.

To investigate the effect of data preprocessing on our model, we performed Random Forest on the all numeric features dataset which is the raw dataset with nominal and text data removed. The model returns the accuracy of 0.7304 which is higher than the performance after data preprocessing. This indicates that Random Forest struggles to eliminate the risk of overfitting created by data preprocessing and negatively affected by it.

The One Hot Encoding, while capturing valuable categorical information, introduces many new features which represent information of four features in the raw dataset, thus it makes the dataset have few very strong predictors. Random Forest, which selects a random subset of features for each split, may not always use these strong predictors, leading to less optimal splits and potentially lower performance.

4.3 Bagging Decision Tree

This model is better than Random Forest at classifying label '1', however, the problem of lacking of label '0' is also affecting Bagging Decision Tree model. The reason that Bagging Decision Tree can classify label '1' is that it does not introduce randomness in the feature selection process. This allows the model to fully utilize all available features at each split, which can be particularly beneficial when certain features are highly predictive.

4.4 Synthetic Minority Over-sampling Technique

The result after applying SMOTE on the dataset shows a dramatic increase in accuracy for both models. SMOTE generates new instances for the minority class by interpolating between existing instances using k-nearest neighbors. Even though it helps balance out the dataset to improve the performance of both models, it leads to overfitting. The synthetic instances cause the models to learn patterns specific to the training data, rather than generalization to unseen data. Both of our models performed well on other labels, but struggled with recall for label '2' with recall values of 0.5470 for Random Forest and 0.2012 for Bagging Decision Tree. This leads to a problem because label '2' dominates our dataset with approximately 60% of the whole training dataset.

5 Conclusions

This report has demonstrated the procedures of predicting movie rating based on data collected in social media using Random Forest and Bagging Decision Tree models. We used data preprocessing, feature encoding, and handling imbalanced classes to support our model learning process. We have found out how the model would behave had we omit preprocessing, and investigate, compare the performance metrics to gain insights of our training process.

References