Classification models for movies

Yutong He

1 Introduction

The main objective of this project is to predict whether a movie would win any Oscar and Golden Globes award. This could provide predictive insights to the movie industry prior to the announcement of the annual award. The analysis is done by developing machine learning (ML) models trained on a labeled dataset containing movie features. The project makes use of the Kaggle dataset Movies (IMDb, Earnings and more), and the documented model development processes are available at this Jupyter Notebook.

2 Data description and preparation

2.1 Exploratory analysis

The dataset contains 3974 movies (rows) and 16 features (columns) initially. The features include a mix of object variables (movie names, director, actor(s), genre), and numeric variables (running time, budget, box office, IMDb score, etc.). At the preprocessing stage, we performed the following tasks.

1. We dropped the column containing movie names. This assumed the names would not have effects on movies' award-winning chance as much as other factors in the dataset. A practical reason of dropping this column is if we were to keep it, then encoding the names uniquely to numerical values would create too many extra features to model.
2. We combined four columns in the dataset, ['Director', 'Actor 1', 'Actor 2', 'Actor 3'], into one, and encoded their impact in terms of the number of times any of the crew members won awards from previous years. Apart from the assumed correlation between the past success of a movie crew and the performance of a movie in the future from the same crew, a practical reason of doing so is also to avoid creating too many extra columns.
3. The movie ['Genre'] column was uniquely encoded to integers.
4. Rows with missing values were dropped.
5. We found out the distribution of the dataset was extremely skewed (see figure 1), as the vast majority of movies in the dataset won no award. Even combining all award winning movies into one category would not produce a significantly more balanced dataset (see figure 2).

2.2 Dataset re-balance

We apply three different methods to balance the originally skewed dataset and check their effects via a simple logistic regression model. The three methods are:

1. Class-reweighting. This introduces a weight on one class over another but the overall amount of samples in both classes remain the same. In this analysis, class 1 is assigned 0.8 and class 0 has 0.2.
2. Synthetic Minority Oversampling Technique (SMOTE). This method synthetically produces more samples in class 1 to match the samples in class 0 (see left panel of figure 3).
3. Random undersampling. This achieves the opposite of method 2 above and randomly choose initially more populated class 0 to match class 1 (see right panel of figure 3).

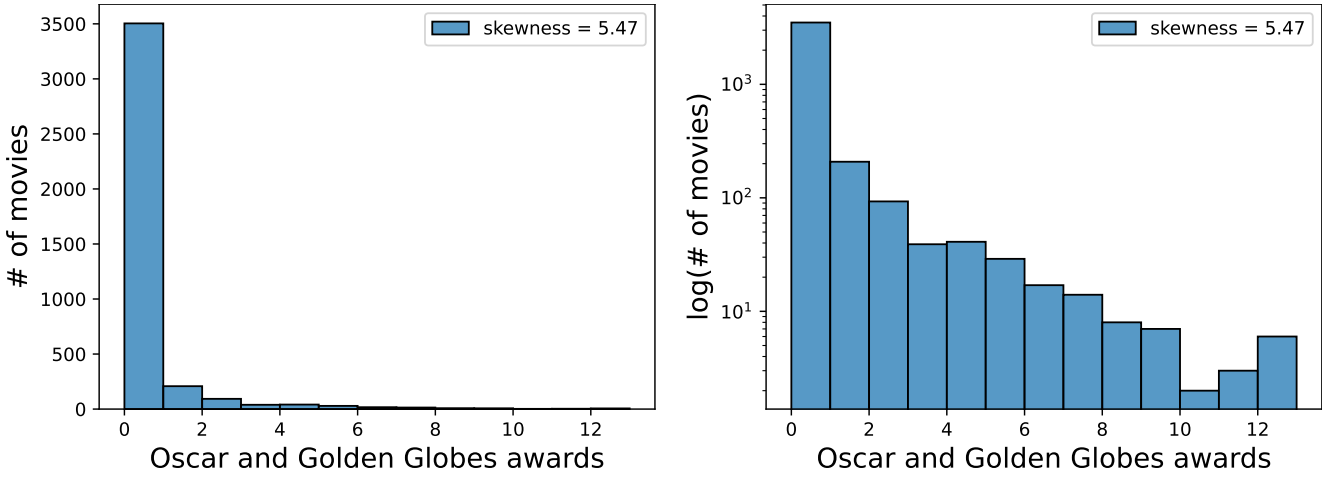

Figure 1: Skewed distribution of the original dataset in terms of the target variable on the horizontal axis. Linear (left) and logarithmic scales (right) are shown on the vertical axis.

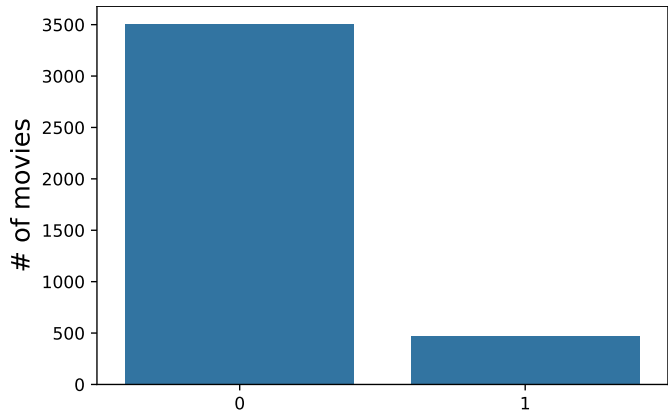

Figure 2: Clumping all award-winning movies in one category encoded with number 1, irrespective of the number of awards, would still not match category 0, i.e., movies that won no award.

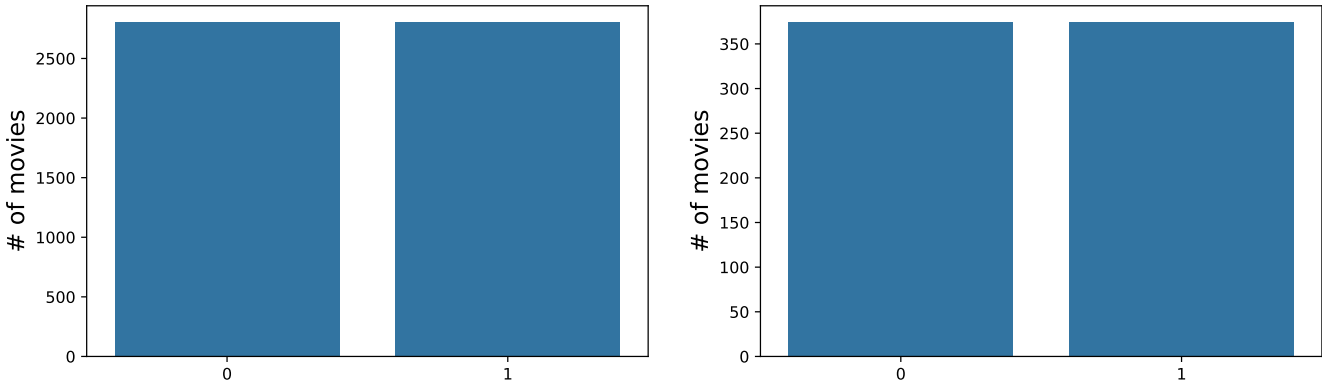

Figure 3: Using methods SMOTE (left) and undersampling (right), parity is achieved between two initially unbalanced classes.

Measuring the accuracy, precision, recall, f score, and auc from the three methods relative to those of the unbalanced dataset (see figure 4), we choose class-reweighting as the method to implement throughout the project. The confusion matrices corresponding to the three methods and for the original dataset are shown in figure 5 to illustrate the effect and importance of balancing the dataset.

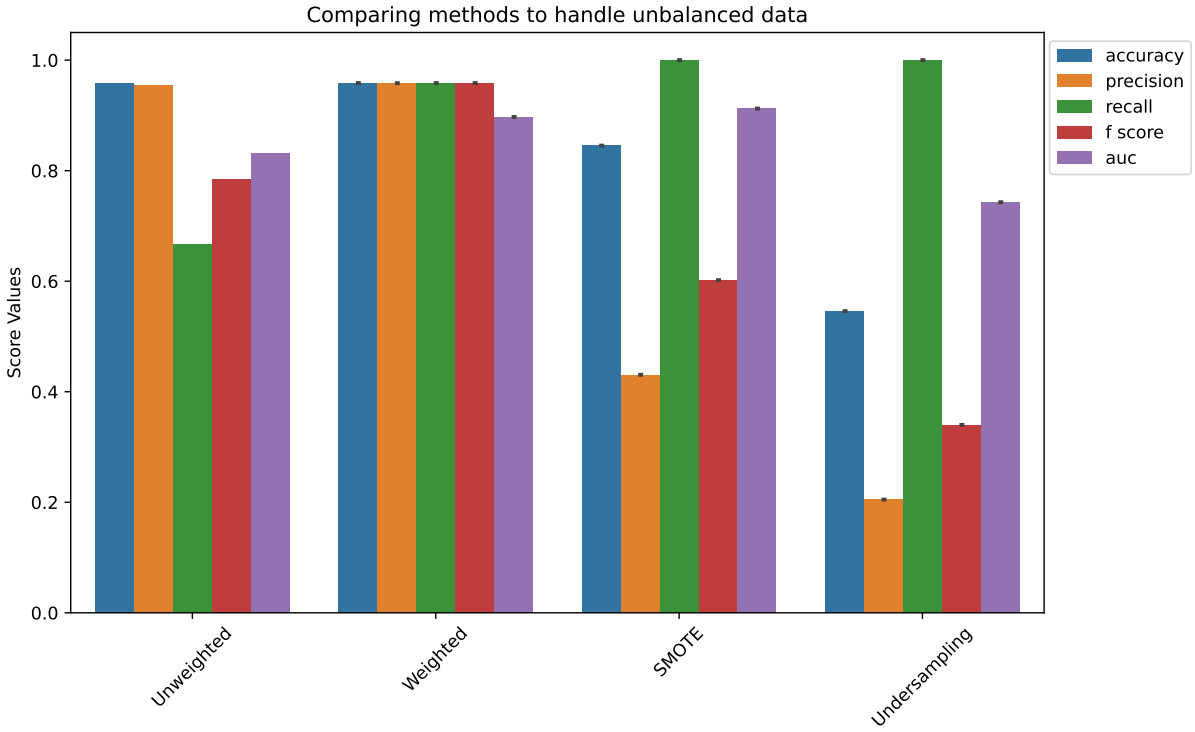

Figure 4: Comparing the effect of different methods to handle unbalanced dataset.

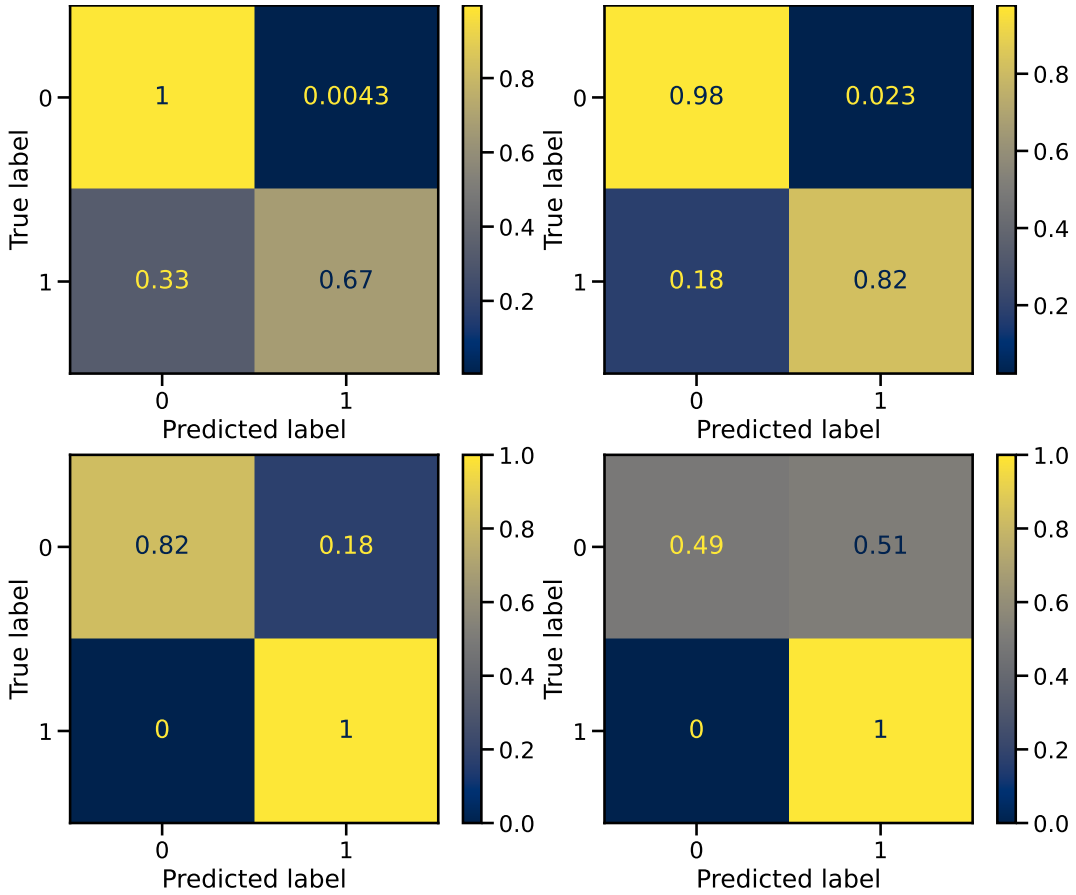

Figure 5: Confusion matrices of a logistic regression model applied to the original dataset without weight (upper left) and with class re-weighting (upper right), and to the SMOTE (lower left) and randomly undersampled dataset (lower right).

3 Classification models

Four classifiers were developed for this project: logistic regression, support vector machine (SVM), decision tree, and random forest. Different sets of model parameters corresponding to the relevant models were searched through for tuning purposes. The final results were obtained from each model using optimally tuned parameters. See the Jupyter Notebook for detailed model development and tuning process.

3.1 Logistic regression

Logistic regression provides an intuitive understanding of the factors contributing positively and negatively to the classification result, shown in figure 6. Some of the coefficients contribute negligibly to the outcome and are now shown in the figure. Of the five features that remain important in the trained logistic regressor, four correlate positively – Oscar and Golden Globes nominations, IMDb score, Budget, Director Box Office %, in descending order of importance. And one correlates negatively – Release year.

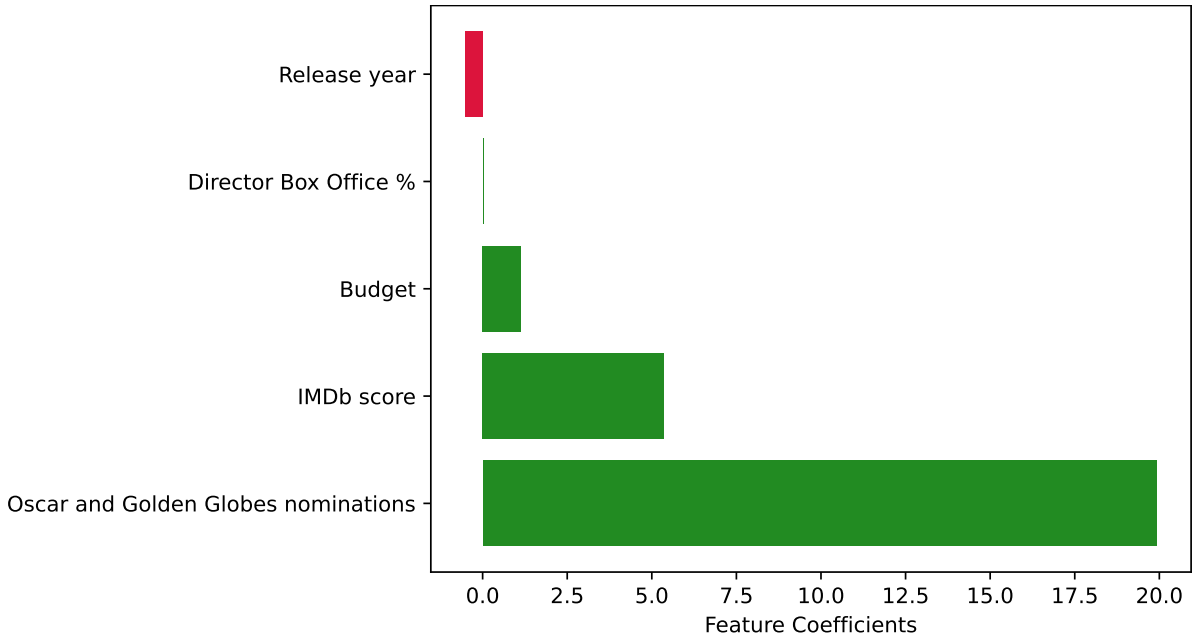

Figure 6: Coefficients of important features contributing positively (green) and negatively (red) to the classification output from the logistic regression model.

3.2 Support vector machine (SVM)

SVM operates by determining a hyperplane boundary separating the two classes that are otherwise rather difficult to disentangle. To illustrate the effect of SVM visually, we show in figure 7 the decision boundary between the two classes in a reduced case where only two features, Oscar and Golden Globes nominations and IMDb score, are selected. The solid blue curve indicates the boundary maximizing the distances away from support vectors (black circles) in each class 0 (red dots) and 1 (blue dots). As a result, most class 0 items (red dots) and class 1 items (blue dots) reside on either side of the decision boundary. This reduced SVM as an example also provides a rather straightforward visual illustration of the decision-making reasoning. But overall, SVM operating on the full dataset in higher dimensions does not output an easily visualizable boundary.

3.3 Decision tree

Decision tree starts from the full dataset and successively splits into two or more branches until the final subsets contain objects within only one of the two classes. The tree is split such that the information

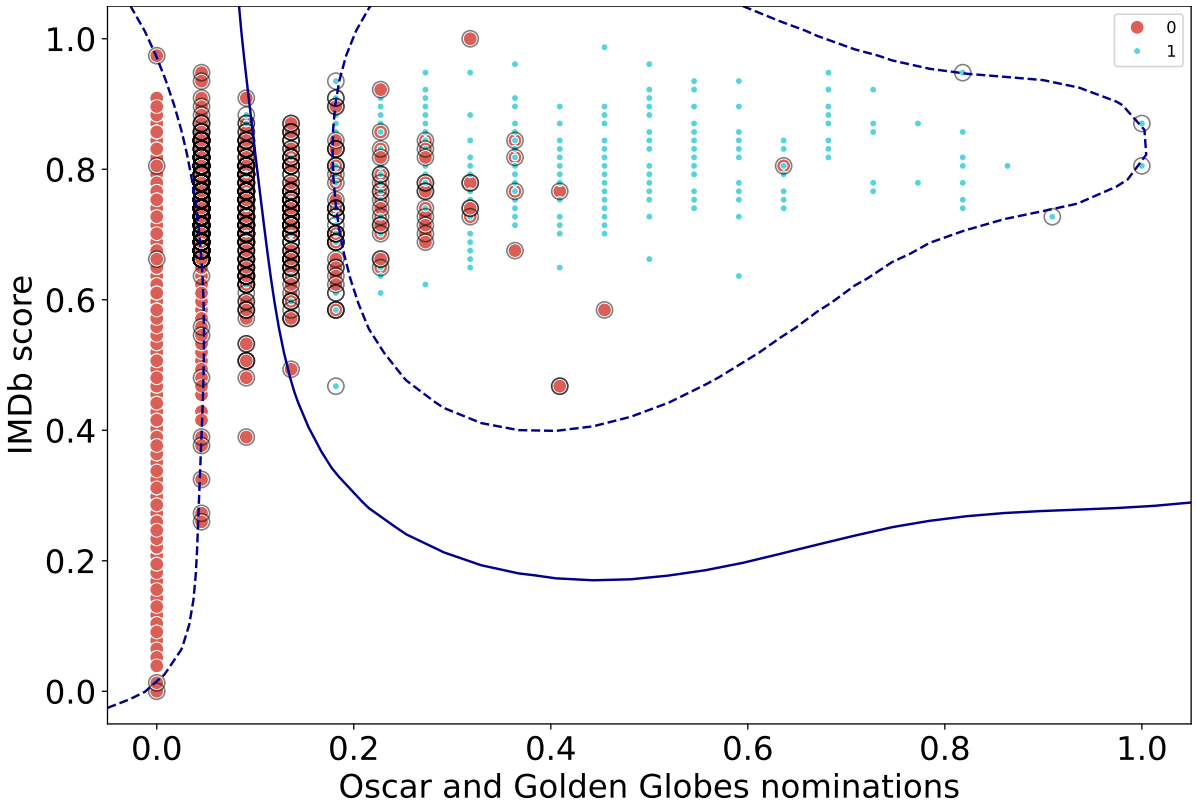

Figure 7: Decision boundary of a reduced SVM with only two features displayed on the horizontal and vertical axes.

purity is increased at every level further from the top. This process is rather logical and easy to follow in principle, although in practice the system becomes quickly complicated as the level goes deeper, such as figure 10 in the Appendix for our example on the movie dataset.

3.4 Random forest

Random forest is an ensemble model based on trees. But unlike the base decision trees, random forest assigns estimators that decorrelate features in the input dataset. The number of estimators is thus a tunable model parameter that does not have to be the same as the number of features. Figure 11 in the Appendix shows the correlation between the 16 random forest estimators selected for this project. This model is less intuitive than others since the estimators are assigned algorithmically and may not always correspond to variables that have real-world meaning. However, random forest can be a powerful classifier precisely because it could construct indicators that are less correlated among themselves.

4 Insights and key findings

We compare the key findings from the four classifiers presented in section 3 by looking at their accuracy, precision, recall, f score, and auc shown in figure 8. Overall SVM has the highest scores across all metric. We also provide the confusion matrix of the SVM in figure 9. Given that SVM also has relatively easy interpretation conceptually, we choose it as the final classifier to predict the award-winning prospect of a movie.

Overall, ML models are project specific and need to be customized to address the individual questions and datasets. Data preprocessing and exploratory analysis presented in section 2 takes up the bulk of the project in terms of time spent. This is not uncommon for ML projects since many model functionalities are readily available from Python packages but real-world datasets can take some time to organize and prepare before feeding them to the ML models.

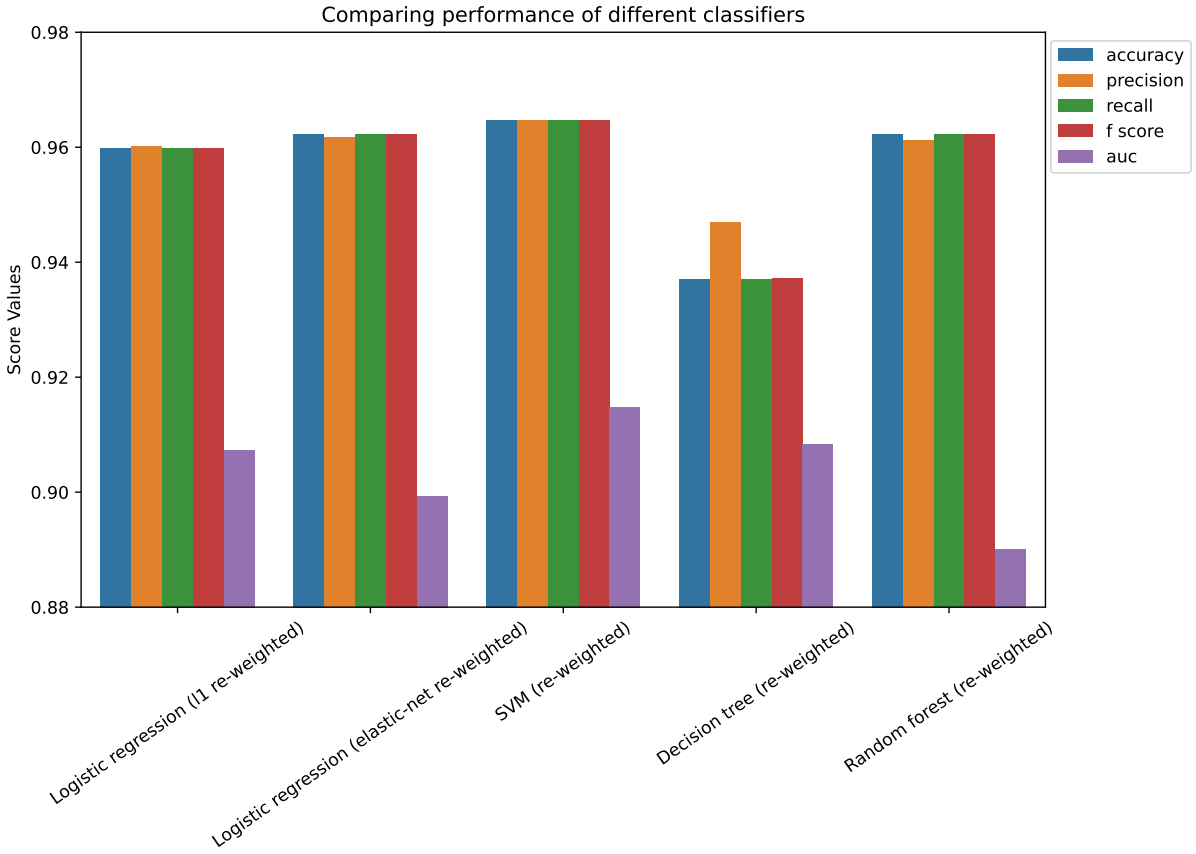

Figure 8: Performance comparison of the classification models presented in section 3.

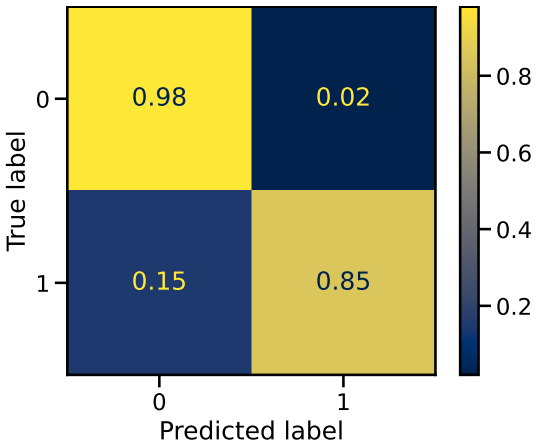

Figure 9: Confusion matrix of the SVM.

5 Outlook and conclusion

In terms of the model development, a next-step improvement would be to sweep through a larger parameter space to fine-tune the models for better performance. A related aspect is to decide a different measure for optimal model parameters. In this project, the weighted f score is chosen as the metric for the grid search but perhaps a different measure would produce better model parameters.
A multiple class predictor would be an even better deliverable than a binary classifier but the input dataset would be extremely skewed and tricky to handle. A larger dataset containing awards other than Oscar and Golden Globes could perhaps compensate for the imbalance to some extent.

Appendix: large figures

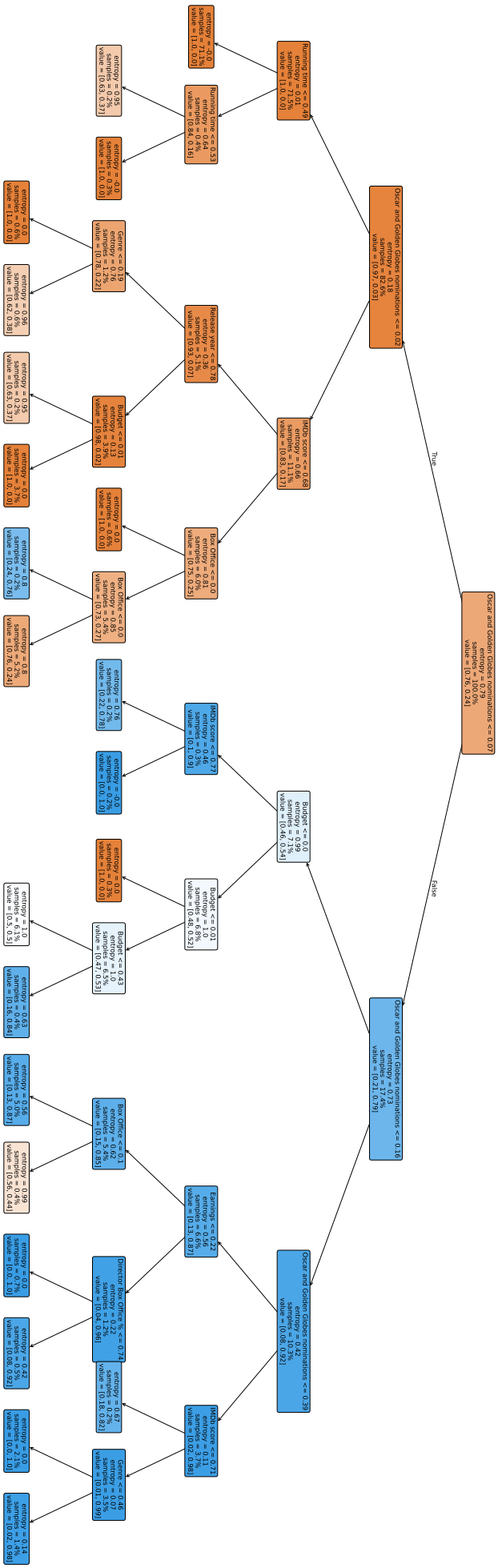

Figure 10: Decision tree visualization.

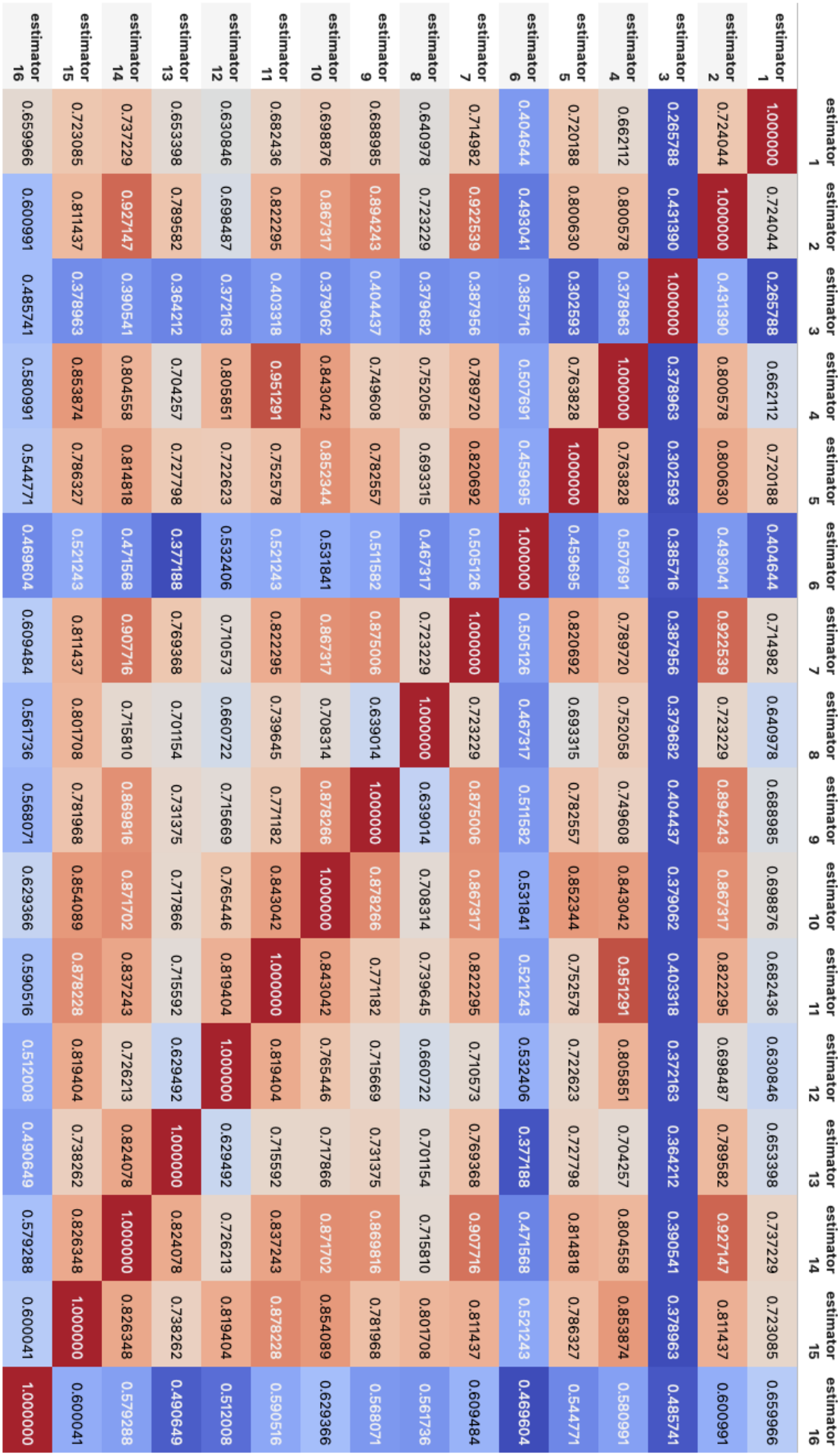

Figure 11: Random forest estimators and their correlations.