Predicting ICO Success using various Machine Learning Models

1 Introduction

The growing popularity and value of Bitcoin over the last few years has increased publicity for cryptocurrencies drastically. The sector has been steadily growing with new coins being launched every day. Typically, newly created cryptocurrencies are tied to the funding of a blockchain based project. In an initial coin offering (ICO) money is raised for these projects by issuing coins in exchange for investment. Investors who see value in the project tied to the coins will hope their value will increase over time if the project is successful. However, not all these ICOs succeed in reaching their fundraising goal. In this report we aim to predict whether an ICO will succeed based on various datapoints. We will first explore the dataset used for this project and then apply appropriate cleaning and transformation to the data to prepare it for statistical modelling. We will then try several machine learning classifiers to fit to the data and try predicting ICO success. We will compare these models using the appropriate performance metrics in an effort to find the best model for this task.

2 Data Understanding

The dataset we worked with contains 1606 instances of ICOs, with data for 20 features. The *id* feature stores an id for each ICO. There is also a *tokenName* feature, but not all entries for this are unique.

The target attribute for our project is the *success* feature, which is set as either 1 if the ICO was successful or 0 if not. There are four other binary variables in the dataset. *softcap* indicates if a minimum fundraising goal was set for the ICO. *hardcap* indicates whether a maximum fundraising goal was specified. *whitepaper* indicates whether a whitepaper was published for the cryptocurrency. *video* indicates if the ICO's campaign page included a video.

The dataset also contains six other numerical features the histograms of which can be seen in Figure 1. *tokenNum* is the number of tokens to be issued with the ICO. This variable ranges from 0 to 22,600,000,000,000,000 and is heavily skewed. *teamSize* is the number of team members on the project. It ranges from 0 to 67 and is also skewed. *overallrating* is a rating given by investment experts on a scale of 1 to 5. The data is normally distributed and ranges from 0.8 to 4.8, so there must be

some invalid values included. *offered_ownership* is the percentage of tokens offered to investors for the ICO. Being a percentage, the values should range from 0 to 1, but the actual data ranges from 0.01 to 601.25. This must be an invalid entry which explains the histogram looking off. *tokenPrice* is the price set per token for the ICO. This is technically not numerical in the data as it is encoded as a string with the exchange rate, so we cannot peak at the distribution without further processing. *socialMedia* is an indicator of how actively the ICO was promoted on social media on a scale from 0 to 3, which is reflected in the data.

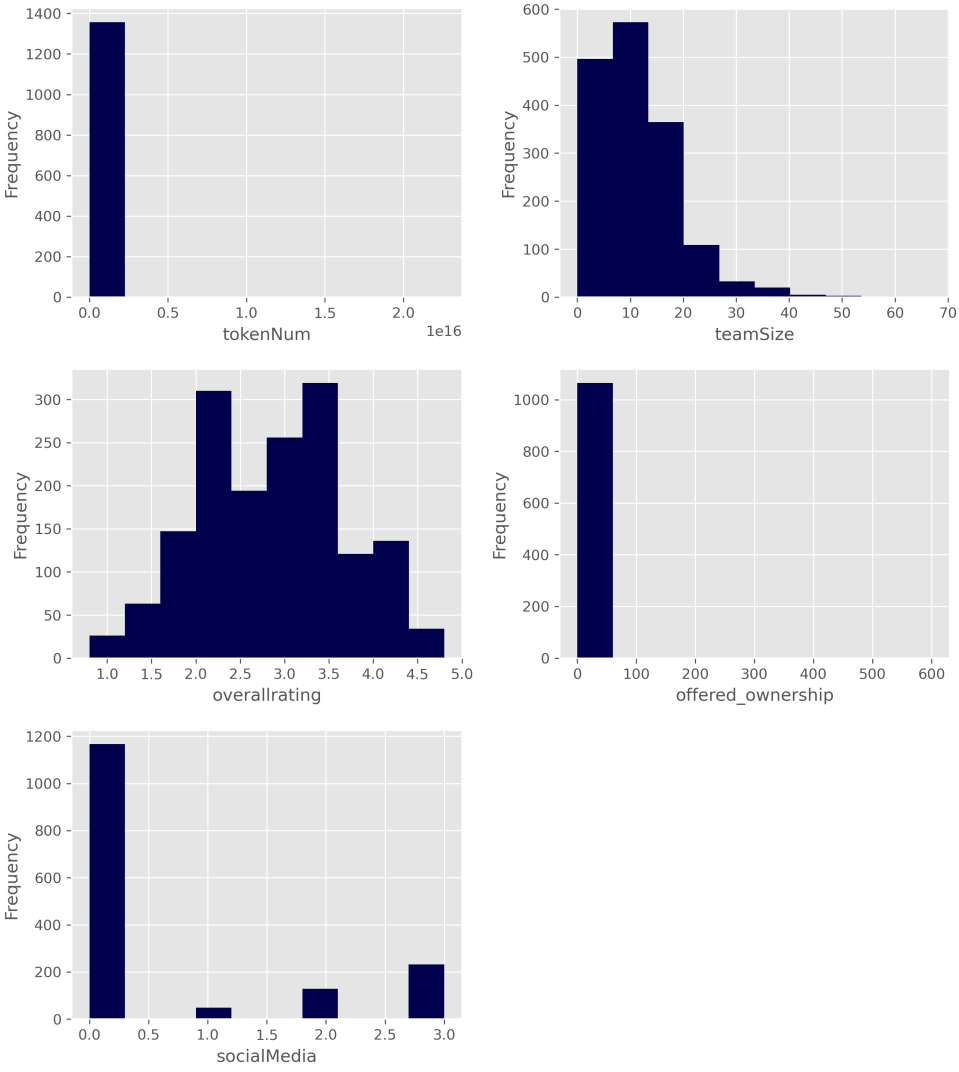

Figure 1: Histograms of numerical features

There are two variables that represent dates in the dataset. *startdate* stores the starting date of the fundraising campaign, while *enddate* stores the ending date.

Finally, there are five categorical features in the dataset. *country* stores the country where the project is located. *categories* is the sectors the cryptocurrency is related

to. *platform* stores which blockchain platform the cryptocurrency is based on. *acceptingCurrency* is the names of all currencies that are accepted for the ICO.

To get some understanding of the influences between the features Figure 2 shows a correlation matrix for the numerical features. As we can see most features are predominantly uncorrelated which is beneficial as this reduces the risk of multicollinearity. Only *teamSize* and *overallrating* are somewhat correlated. We wanted to check the correlation between *tokenType* and *platform* too. For this we performed a chi-squared test, which revealed high correlation with a p-value of basically 0.

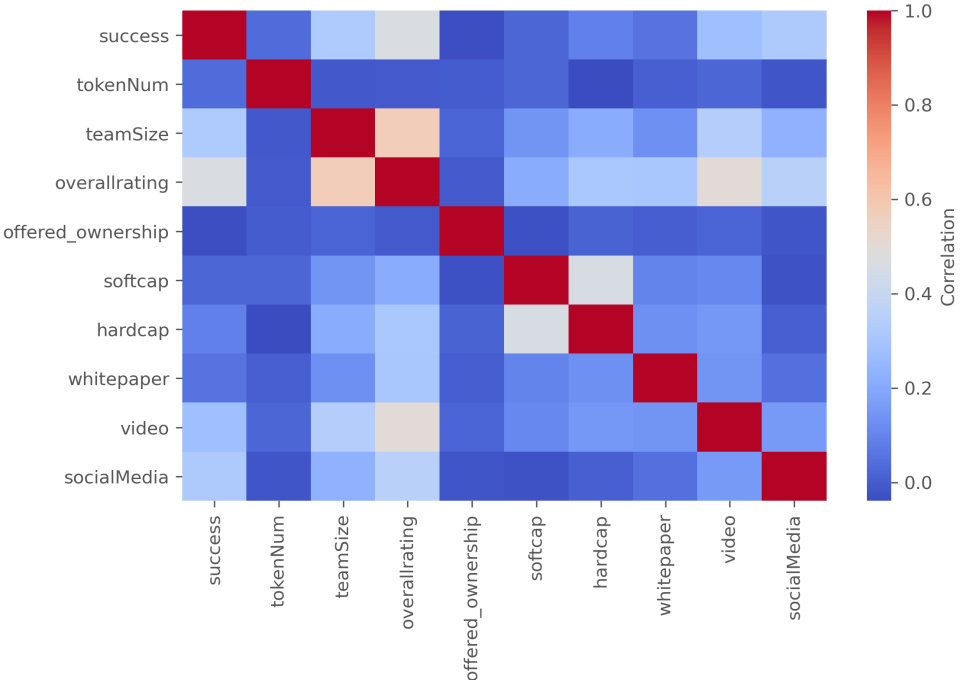

Figure 2: Correlation matrix of numerical features

3 Data Preparation

The first step we took for cleaning the data was to remove features that provided no informational value to predict ICO success. This included the *id* and *tokenName* features. We did check the *tokenName* values first and discovered that there were a lot of duplicates. Since the duplicates values varied across all other features it did not seem like these were invalid instances and rather different attempts using the same token abbreviation. We also decided to remove the *tokenPrice* feature. This probably does have some prediction value, but the format used was not standardised and used many different base currencies for the exchange rate. Converting all

these to use a common base currency would have been difficult and imprecise since exchange rates change over time and there was no temporal information included.

The next step was to gain an overview of the missing values in the dataset. Figure 3 shows a plot of missing values across all features. As we can see the *offered_ownership*, *tokenType*, *tokenNum* and *acceptingCurrency* features needed the most work with 34%, 29%, 15% and 14% missing values respectively. First however we can see that there are several instances where values are missing for the five last features. These are mostly binary features where it is hard to justify filling in values. As these instances represent only 2% of the dataset we decided to remove them.

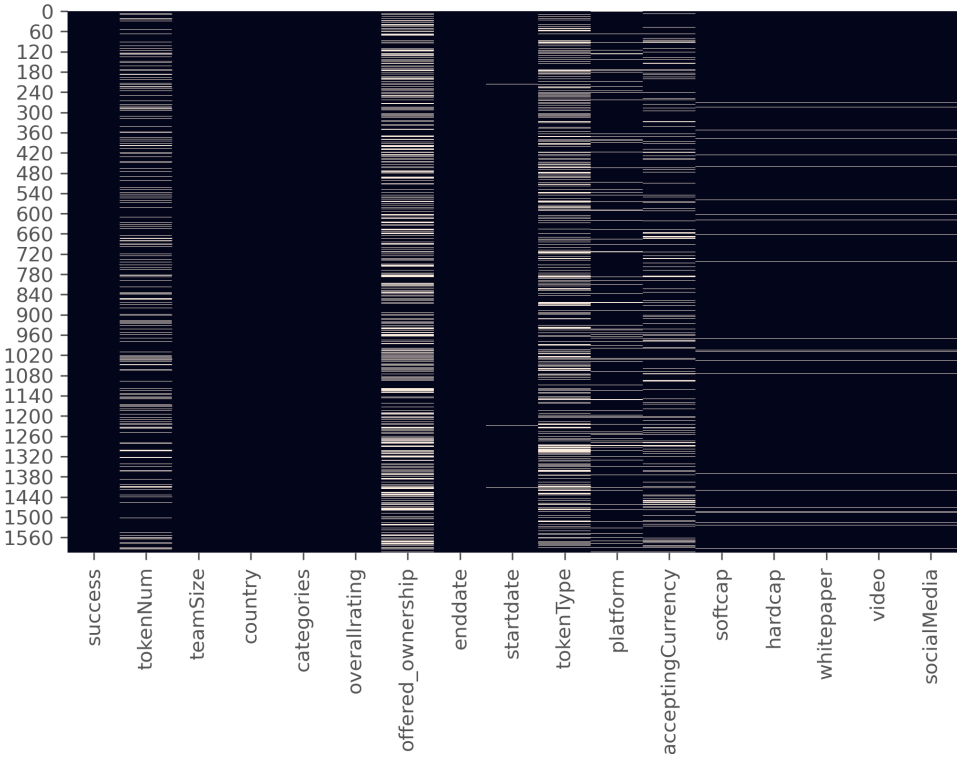

Figure 3: Heatmap showing missing values of all features

startdate and *enddate* cannot be interpreted by our models and provide little information by themselves. Instead, we were interested in the campaign duration, so we created a new feature as the number of days between the *startdate* and *enddate*. We replaced 5 missing values as well as some invalid instances where the *startdate* came after the *enddate* with the median *campaignDuration*. Figure 4 shows a boxplot and linechart representation of the *campaignDuration*.

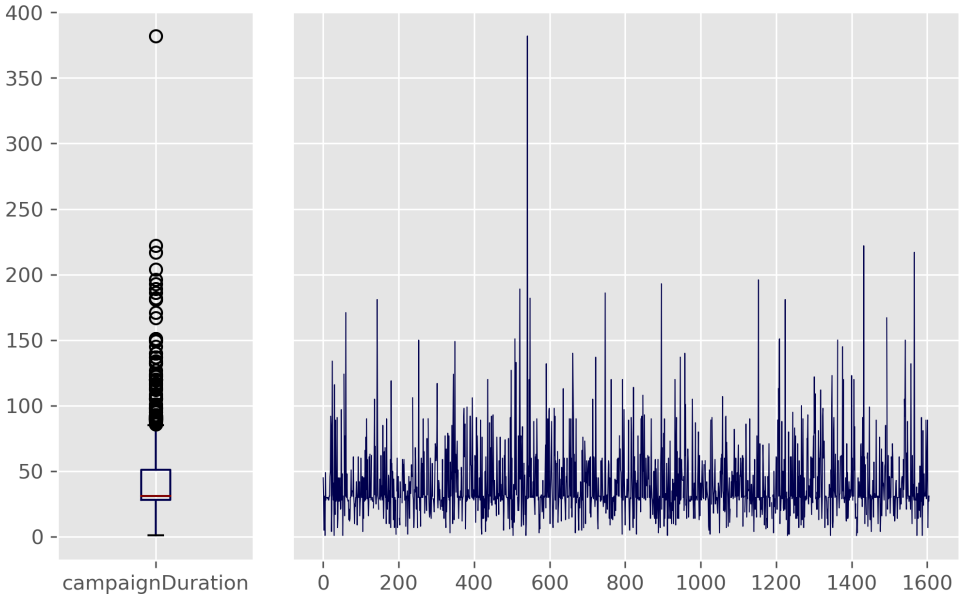

Figure 4: Boxplot and linechart for *campaignDuration*

As we can see there is a clear outlier with a *campaignDuration* of 382. We used min-max scaling to make sure all values ranged between 0 and 1. We did this so the features are weighted the same by our algorithms and exceptionally large values of one feature do not overpower any of the others. To make sure the majority of values for *campaignDuration* are not close to zero and their information is not lost due to some instances with large values, we decided to cap the *campaignDuration* at 80 days and assign every instance larger than that the maximum value of 1. The

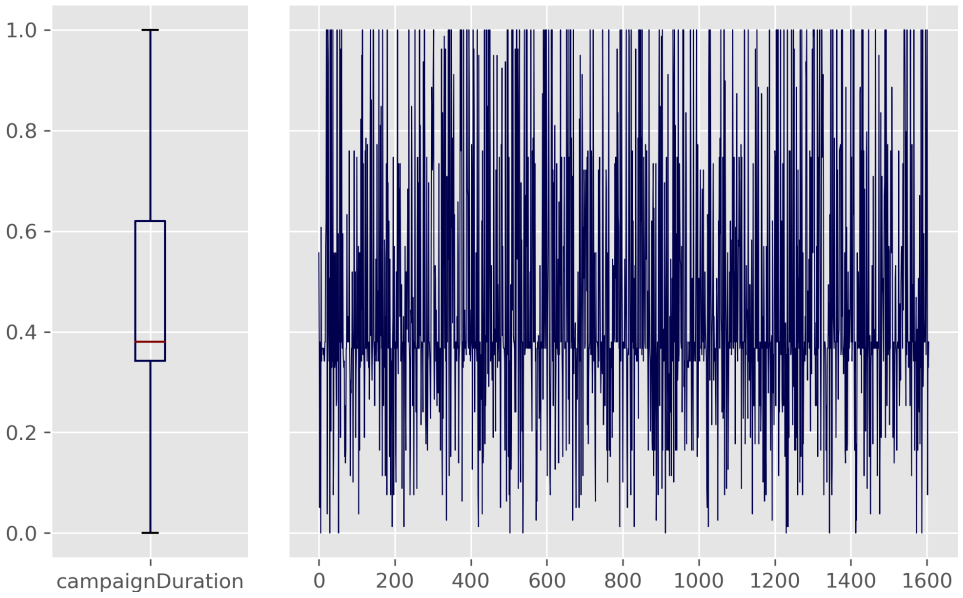

Figure 5: Boxplot and linechart for *campaignDuration* after rescaling

results of this rescaling can be seen in Figure 5 where the better distribution of *campaignDuration* should be clear.

For the remaining numerical features we included the distribution plots in Appendix A.2. Looking at *tokenNum* next, we saw that the scale of values varied drastically. We introduced a cap of 250,000,000 and scaled the data to range from 0 to 1. We also removed all values of zero or lower and replaced all missing values with the median of the distribution. There was an outlier for *offered_ownership* at 600, which is an invalid value as this variable is supposed to be a percentage value and thus should range between 0 and 1. We therefore replaced it with the median. *teamSize* had some higher values distorting the scale so we set a cap at 25 and applied the min-max scaling. For *overallrating* there were some values below 1, which should not be the case according to the rating scale, so we replaced them with 1 and rescaled the data. For *socialMedia* only the rescaling to our target range was required. Appendix A.1 includes histograms for the numerical features after cleaning and rescaling for a comparison to Figure 1.

Finally, we had to clean the categorical features and transform them to a numerical format for use with our models. Starting with *countries* we first cleaned up the data. This included typos, consolidating entries like *United Kingdom* and *UK* and replacing incorrectly occuring cities with their respective country's names. Some of our models cannot deal with categorical values and we therefore used an approach similar to one-hot encoding do generate features for each of the countries, where an instance was assigned value 1 if it the respective country was listed and 0 if not. It should be noted that this is not strictly one-hot encoding, as some instances have multiple countries listed. The same will be true for all other categorical features as well. This is useful as it reduces the risk of multicollinearity between the features. As an added precaution we did not include a feature for non-country entries like *worldwide* or the missing values. The same process was used for the *categories* feature, where no cleaning was necessary. We noticed there was high correlation between *tokenType* and *platform*, so we decided to remove *tokenType* due to it missing more values. We then followed the same cleaning and encoding pipeline. It should be noted that only Ethereum had a significant number of occurrences and the other created features likely do not have much predictive power. However, we decided to include all features and perform feature selection later where appropriate. We dealt with the last feature *acceptingCurrency* in the same way as with the other categorical features.

After the data preparation process we were left with a dataframe of 240 features and 1577 instances. Each feature's values ranged between 0 and 1 and are well distributed.

4 Modelling

The goal of this project was to fit a statistical model that can predict whether an ICO will be successful. This is a classification task with the *success* feature acting as our target variable. The other 239 features can be used as predictor variables. We first tried a few of the most popular classification models without any hyperparameter tuning, to establish a baseline and see which model we wanted to optimise further. First, we tried logistic regression, a linear model which is often the first step in classification tasks. We also wanted to try a decision tree model, as well as an ensemble method. We used a random forest model for this. We also tried support vector machines, for our last basic model implementation. The results of all models will be covered in section 5, but out of the covered models the logistic regression and random forest models were most promising.

We decided to try and improve our logistic regression model by performing feature selection using recursive feature elimination (RFE). Many of our features might not carry that much information about the target variable and removing them can help both training time and model performance. RFE allows us to find an optimal subset of features by recursively trying smaller and smaller sets of features until the solution is found that maximises the model accuracy. We achieved the best results by using only 6 features, *overallrating*, *socialMedia*, *country_CHINA*, *country_BELARUS*, *platform_OMNI* and *platform_BITSHARES'*.

We also wanted to try using a neural network to see if we could improve on the pre-configured models. The final architecture of this network can be seen in Figure 6. We started out with simple fully connected layers. It soon became apparent

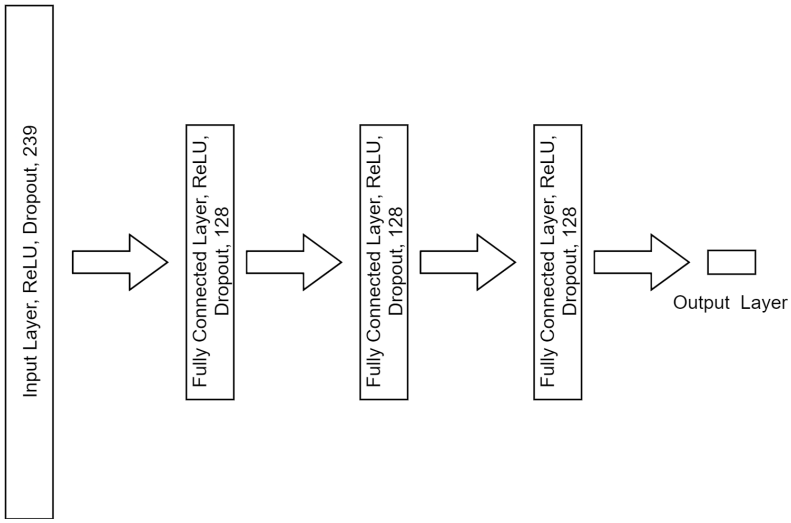

Figure 6: Architecture of our neural network model

however that the model would quickly start overfitting as can be seen in Figure 7 that shows the validation loss and training loss over the training epochs for the network. Both loss curves start decreasing in the beginning as the network begins to train its weights and learn the patterns in the data. However, at around 50 epochs the training loss has flattened out and starts rising again while the training loss continues to decrease. At this point the model is starting to fit to the noise in the data and would get worse at generalising. We therefore only trained our final model for 50 epochs. We also tried adding dropout layers after all fully connected layers and using L2 regularisation to combat the overfitting. Ultimately there was not much else we could do without more data as the 1577 instances in our dataset did not prove sufficient to avoid overfitting. The data is also not well suited to data augmentation so we decided to leave the model as it is shown in Figure 6.

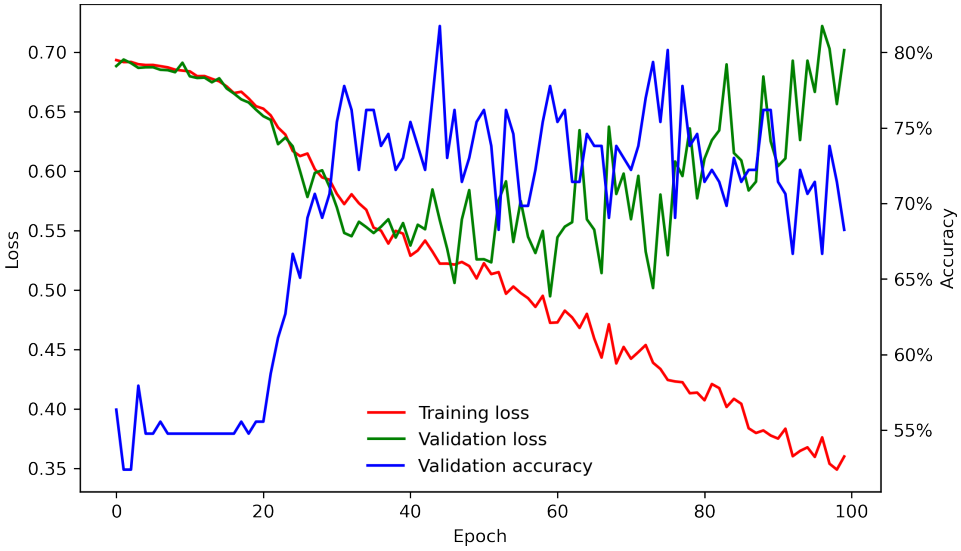

Figure 7: Training loss, validation loss and validation accuracy over epochs

5 Evaluation

In order to analyse the performance of the different models, Table 1 shows various performance metrics for all the tested models. The dataset contains 55.2% instances where the ICO was successful. This is a fairly equal split which means we should be fine with using accuracy as a performance metric. However, Table 1 also shows performance results for the precision, recall and F1 metrics. Precision allows us to judge if there is a tendency towards more false positives (predicting an ICO failure when that instance is actually of a successful ICO), if the precision falls. Similarly Recall allows us to judge if there is a tendency towards more false negatives (predicting a successful ICO when it was actually a failure), if the recall falls.

Metric	LRM	DT	RF	SVM	LRM_RFE	NN
Accuracy	72.17%	64.62%	73.62%	71.47%	74.38%	69.52%
Precision	74.77%	68.50%	75.61%	74.40%	77.34%	71.26%
Recall	74.94%	66.55%	77.13%	73.79%	75.86%	72.08%
F1	74.83%	67.46%	76.35%	74.07%	76.57%	71.73%

Table 1: Performance metrics for the logistic regression model (LRM), decision tree (DT), random forest (RF), support vector machine (SVM), improved logistic regression model after recursive feature elimination (LRM_RFE) and the neural network (NN)

F1 is a holistic performance metric that would be useful if the dataset was skewed and had substantially more instances of one class than the other.

As we can see from the results, the logistic regression model after using RFE to reduce the number of features used and the random forest model performed the best. The results are fairly constant across the different metrics, which is not surprising given the split of the target class. We did not use feature selection for the random forest model because it is an ensemble method that tries many different decision trees using various feature splits and thus performs a type of feature selection on its own. The best performing model in terms of accuracy and F1 score is the logistic regression model with the small feature subset, with an accuracy of 74.4%. However, if we wanted to minimise false negatives, we would prefer the random forest model as its recall at 77.1% is higher than that of the logistic regression model at 75.9%. If we are more concerned about false positives, we should choose the logistic regression model again. This shows quite nicely

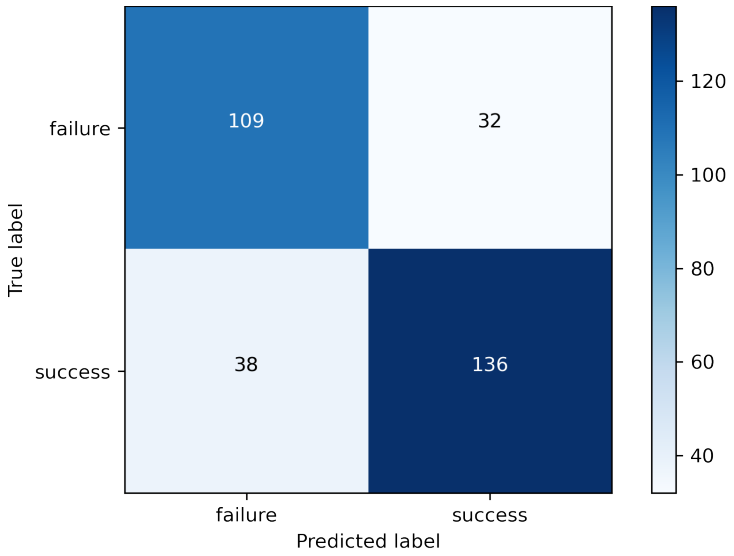

Figure 8: Confusion matrix for the linear regression model after recursive feature elimination

that there is often a trade-off between precision and recall and one must choose which one to optimise for. Figure 8 shows the confusion matrix for the logistic regression model after feature selection. As we can see we have slightly less false positives, counting 32 compared to the 38 false negatives. Almost three quarters of the instances were correctly classified, represented on the main diagonal, with somewhat more true positives than true negatives due to over 55% of the instances being classed as a successful ICO. Confusion matrices for the other modules can be found in Appendix A.3 for comparison.

6 Conclusion

We have found that the dataset on ICOs needed a lot of cleaning and preparation to be used with machine learning models. Predicting ICOs proved hard, with the best model only achieving an accuracy of 74.4%. This model was the logistic regression model using a subset of the available features, deduced through recursive feature elimination. If false negatives are the main concern the random forest model should be used instead, due to its better recall score. Our excursion in building a neural network model has shown us that more data would be required for better performance, as the model begins to overfit rather quickly. Future projects could try more extensive hyperparameter tuning, new models not tested in this report or tackling the problem with a larger dataset. In conclusion we did succeed in finding a model that can predict whether ICOs will be successful more often than not, but there is room for improvement with regards to classification accuracy.

A Appendix

A.1 Histograms for Numerical Features after Cleaning and Rescaling

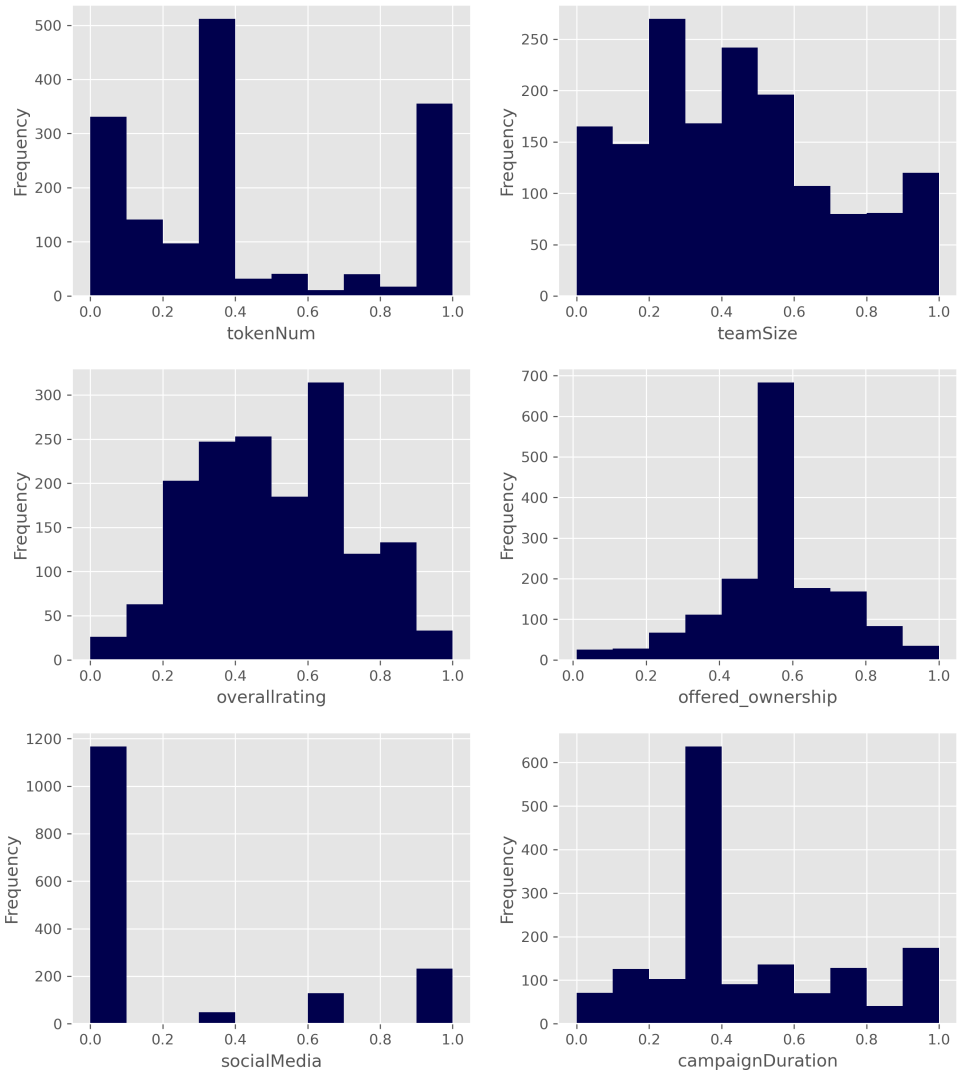

Figure 9: Histograms of all numerical features after rescaling

A.2 Distribution Plots for Numerical Features

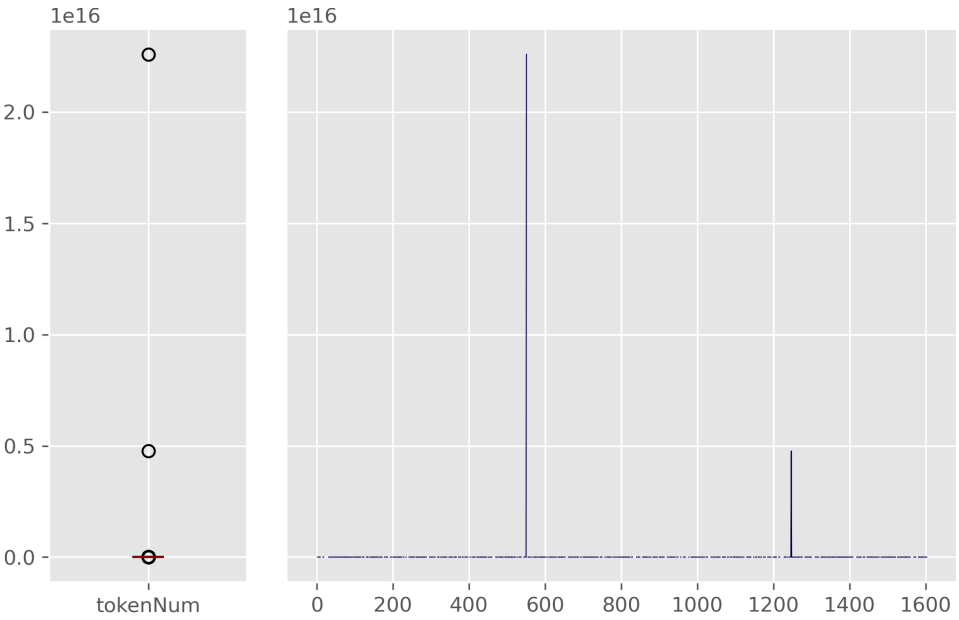

Figure 10: Boxplot and linechart for *tokenNum*

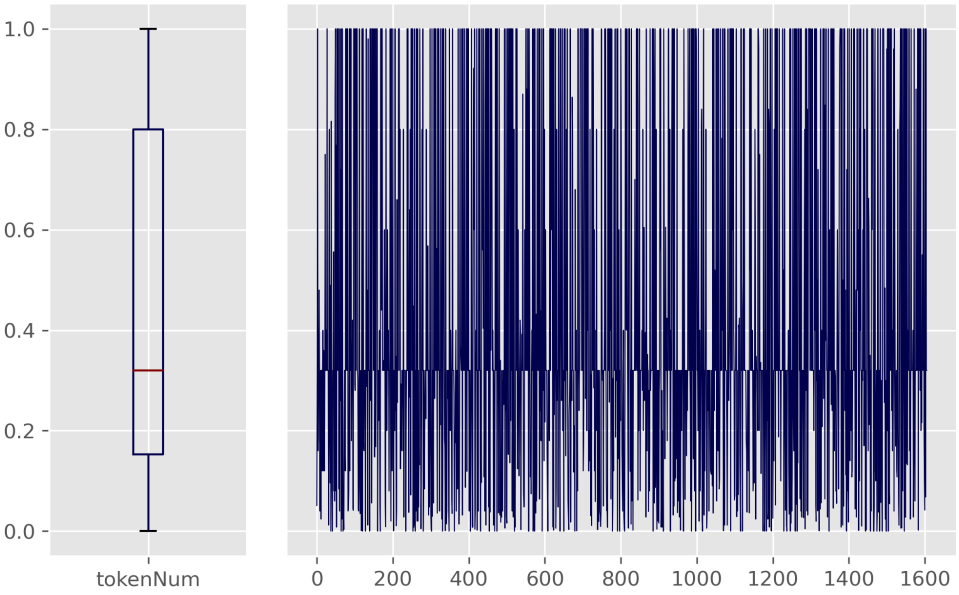

Figure 11: Boxplot and linechart for *tokenNum* after rescaling

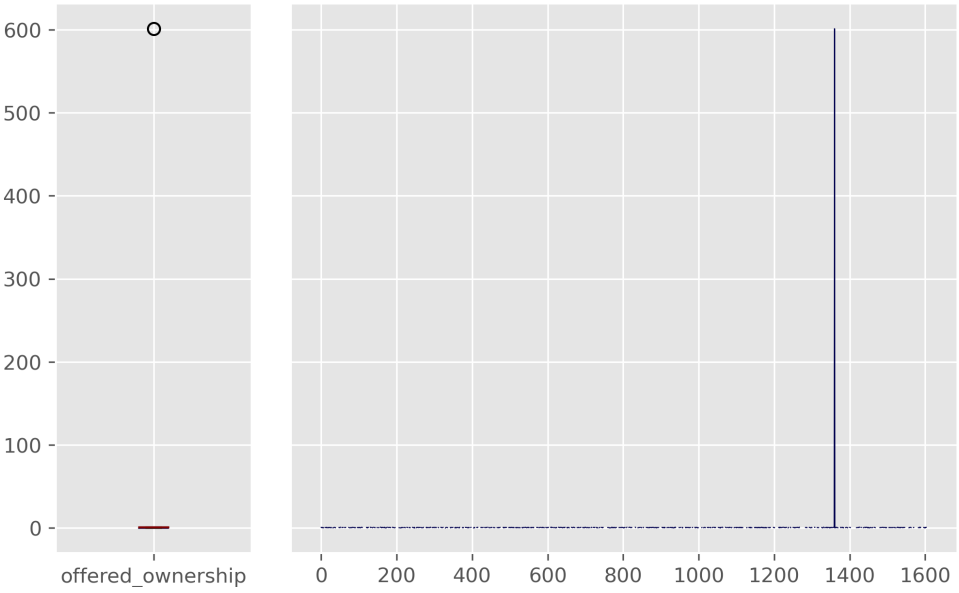

Figure 12: Boxplot and linechart for *offered_ownership*

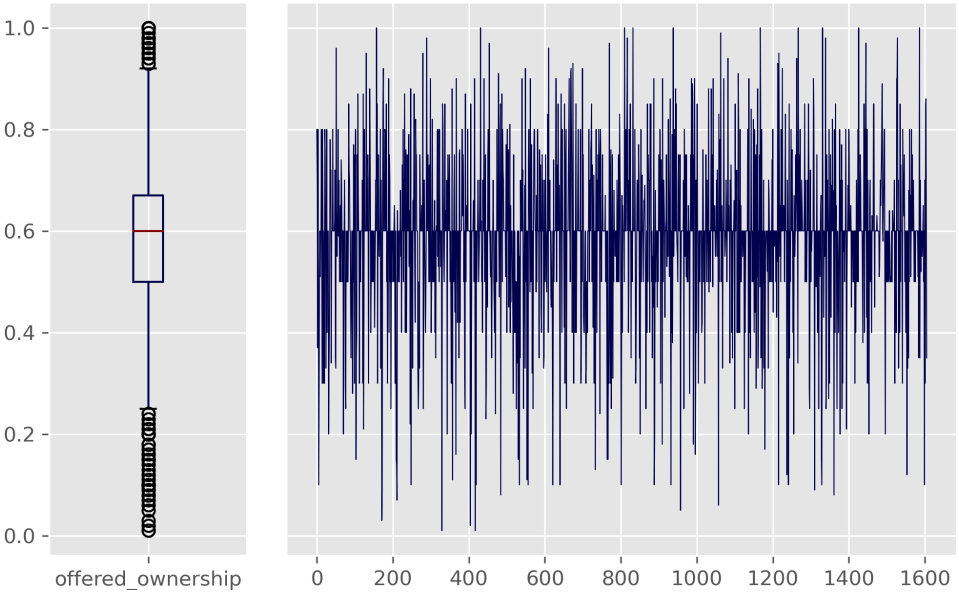

Figure 13: Boxplot and linechart for *offered_ownership* after rescaling

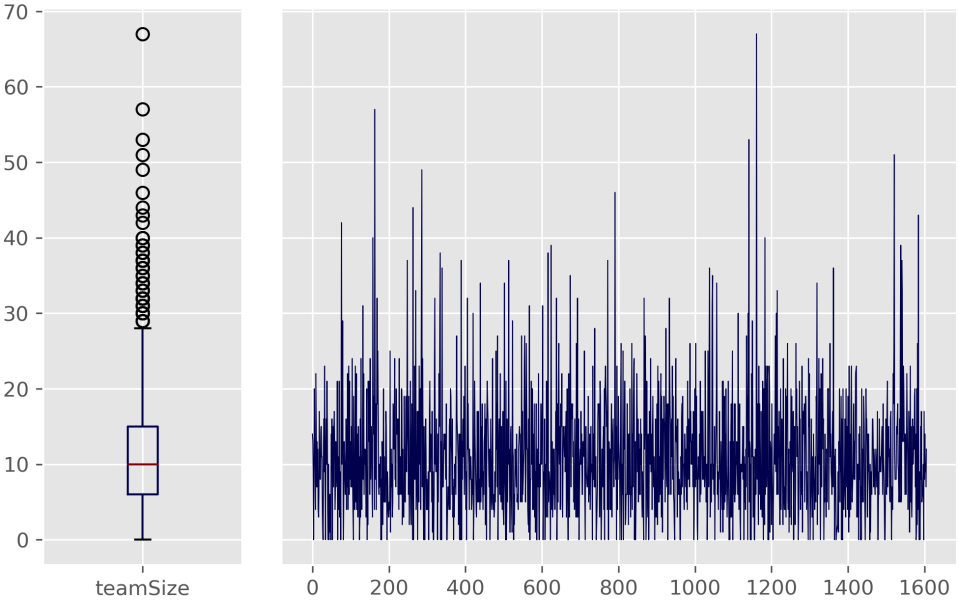

Figure 14: Boxplot and linechart for *teamSize*

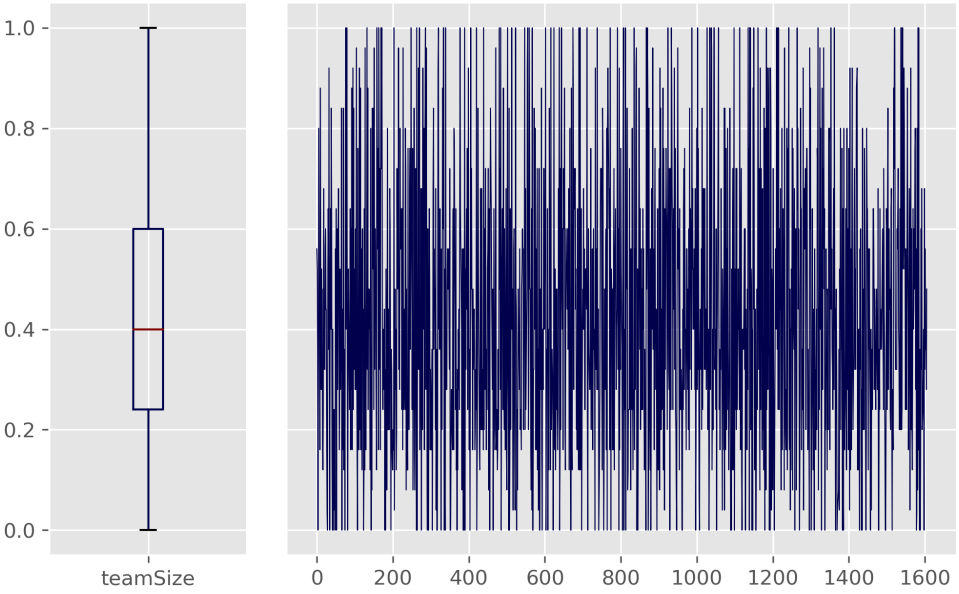

Figure 15: Boxplot and linechart for *teamSize* after rescaling

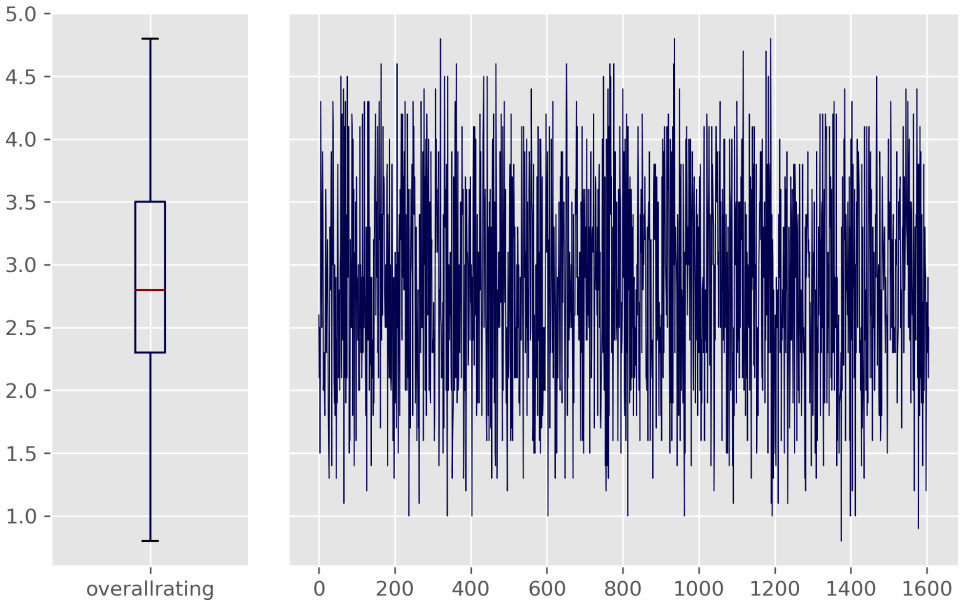

Figure 16: Boxplot and linechart for *overallrating*

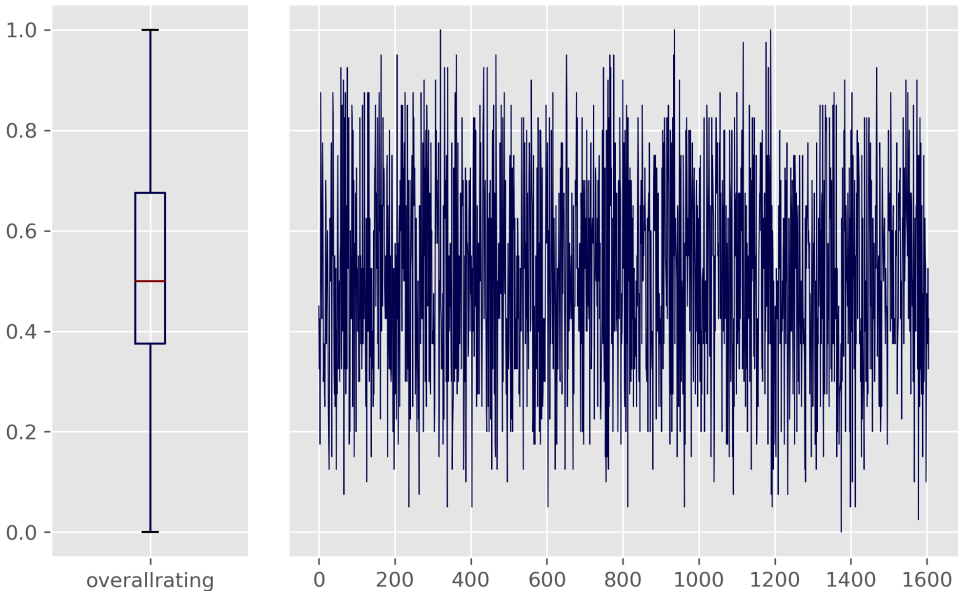

Figure 17: Boxplot and linechart for *overallrating* after rescaling

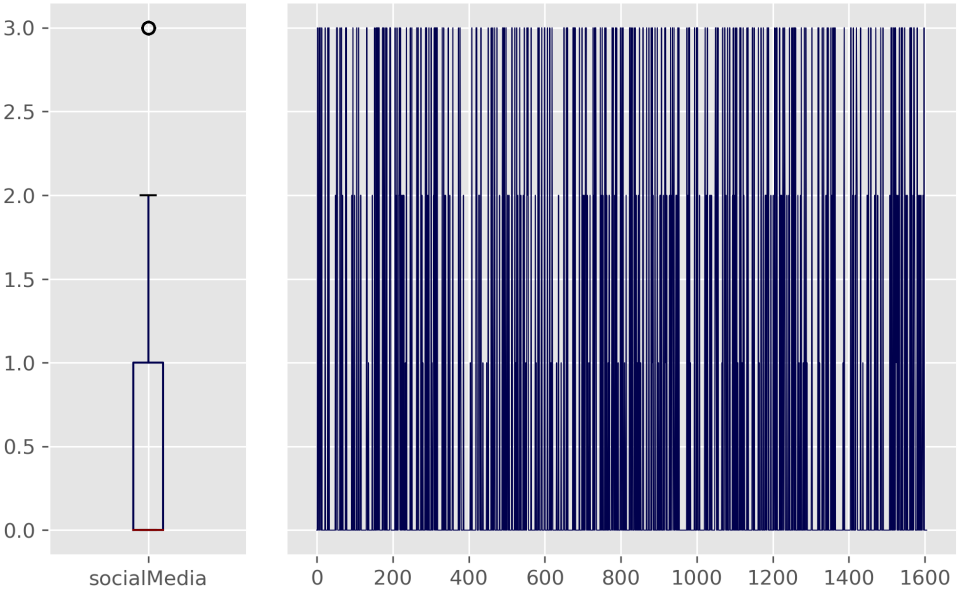

Figure 18: Boxplot and linechart for *socialMedia*

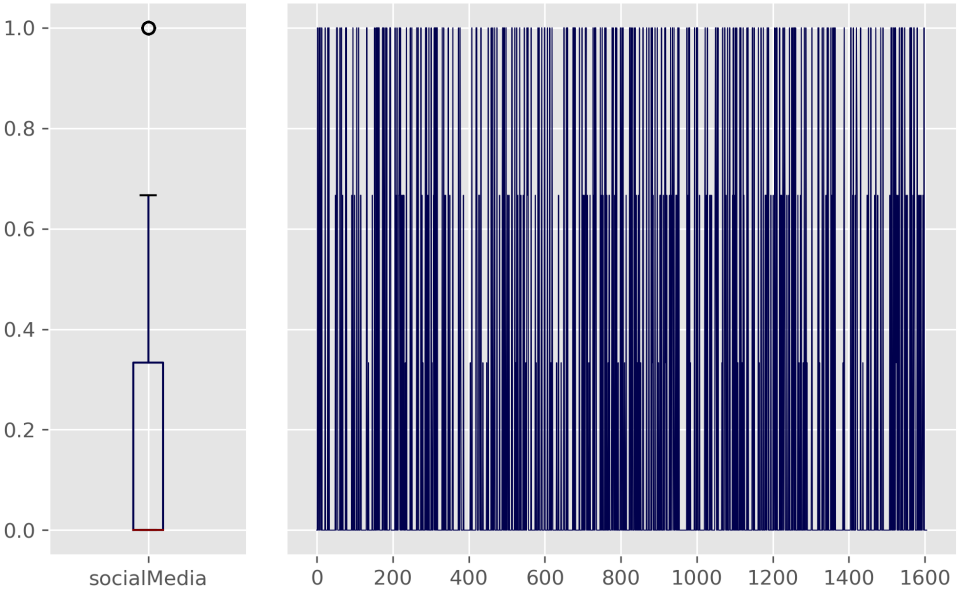

Figure 19: Boxplot and linechart for *socialMedia* after rescaling

A.3 Confusion matrices for remaining models

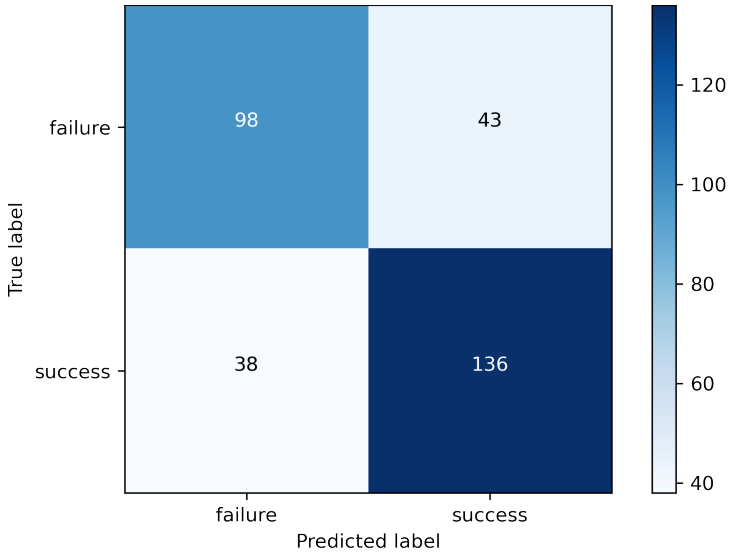

Figure 20: Confusion matrix for the linear regression model

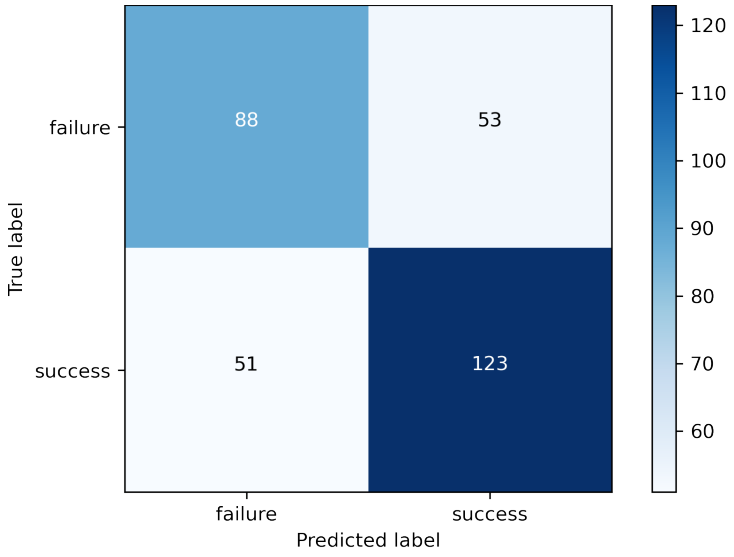

Figure 21: Confusion matrix for the decision tree model

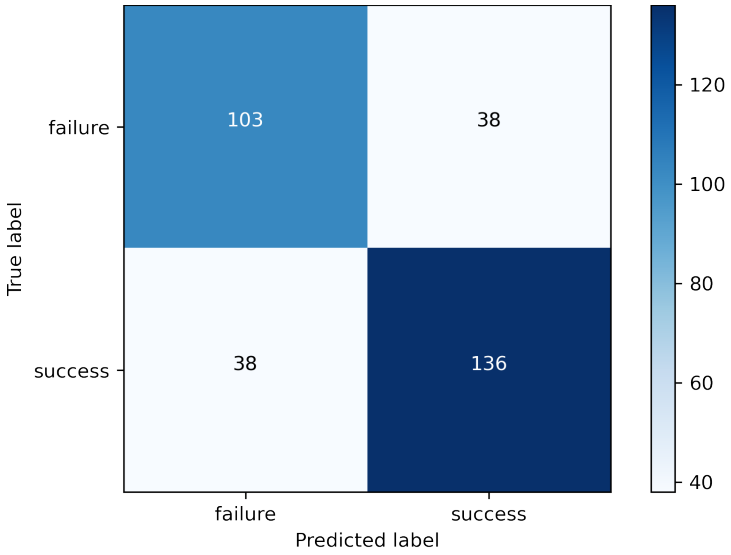

Figure 22: Confusion matrix for the random forest model

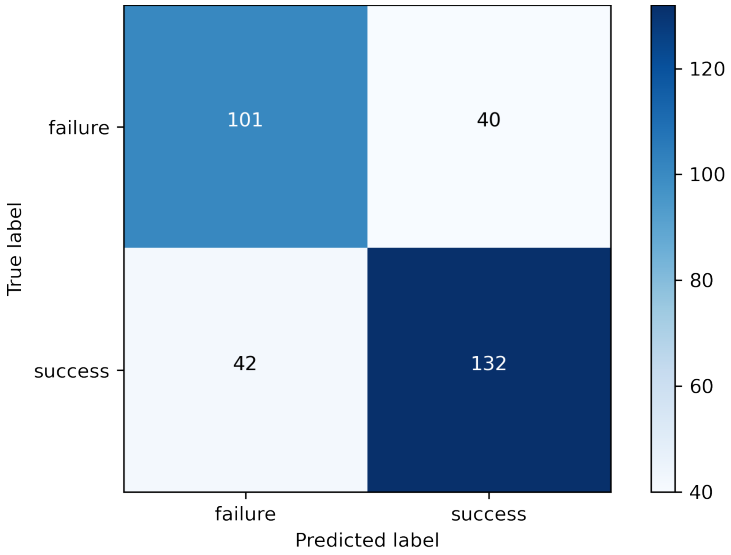

Figure 23: Confusion matrix for the support vector machine