Introduction:

This report was concerned with identifying the best classification model for predicting whether a Kickstarter project will be successful or fail, based on a number of descriptive features. The data utilized for this analysis contained information relating to Kickstarter projects recorded in 2018. Previously, in Phase 1 of this report, the dataset was thoroughly processed and explored in order to prepare it for classification analysis. Several irrelevant features were removed, and some other features were manipulated or combined into a more useful format. Furthermore, the target feature was redefined as a binary variable in accordance with the aim of this project.

The second phase of this report will focus on the construction and evaluation of four classification models, including naïve Bayes, decision tree, random forest and K-nearest neighbor (KNN) models. Initially, the methodology of the classification analysis is described. The results of the hyper-parameter tuning for each model is then reported, with the prediction threshold adjustment for each model also considered. Subsequently, the most ideal model identified from each classification model type is subjected to model evaluation in order to compare performance across all models. A critique of the methodology utilized in this report is then provided, and finally our findings are summarized in the final section.

Method:

*Classification Preprocessing*

Initially, the cleaned and updated dataset generated in Phase 1 was loaded into R. All character features were redefined as factors. All numeric features were transformed via a log transformation to account for extreme values, and were then normalized to be within the range [0, 1]. On import, the dataset was comprised of 331,465 observations, each relating to a different Kickstarter project. Performing classification analysis – specifically hyper-parameter tuning – on a dataset of this proved almost impossible due to processing capabilities. As such, a random sample of 40,000 was taken from the full dataset (this sample was taken with a set seed so that the same sample would be taken on all following runs). With regards to the representation of the target feature, the “successful” level comprised 40% of the observations, while the remaining 60% related to the “failed” target level. This ratio corresponded to the representation of the target within the full dataset almost perfectly. The sample was then partitioned into training and test sets using a 70:30 split. Once again the representation of the target feature was examined within each partition, and both sets had proportions very close to the sample. The training set was then considered for hyper-parameter tuning.

*Hyper-Parameter Tuning*

In all tuning analyses, 5-fold cross-validation stratified sampling was used to account for the slight target level imbalance, and mean misclassification error (mmce) was employed as the optimization metric.

Firstly, a classification task was defined for the training set, with the target set to the “state” feature. A set of learners was then defined and included the naïve Bayes, decision tree, random forest, and KNN algorithms. For the naïve Bayes classifier, the parameter for tuning was the Laplace smoothing parameter. This parameter dictates the degree of smoothing applied to the conditional probabilities utilised to make predictions, and can help mitigate against the occurrence of zero probabilities. Here, a grid search was carried out for Laplace parameter values equal to 0,10,25,50,100 and 200.

For the decision tree classifier, the minsplit parameter and the minbucket parameter were tuned. These parameters control the minimum amount of observations required to split a root node, and the minimum amount of parameters required in a leaf node, respectively. For the minsplit parameter, values considered included 5,150,300,450 and 600, while for the minbucket parameter, values considered were 5, 100, 200 and 300. A grid search was carried out for all combinations of these values.

The parameters considered for tuning in the random forest classifier were the mtry and ntree parameters. The mtry parameter controls the number of descriptive features utilized with each subtree, while the ntree parameter relates to the amount of subtrees generated. It is suggested by Breiman (2001) that the mtry parameter should correspond with the square root of the number of descriptive features in the dataset – in this case, six. This would suggest the ideal value of this parameter would be approximately 2.4, and as such, the values of 1,2 and 3 were considered for tuning. For the ntree parameter, values of 10, 20 and 30 were considered. Again a gird search was carried out for all combinations of these two parameters.

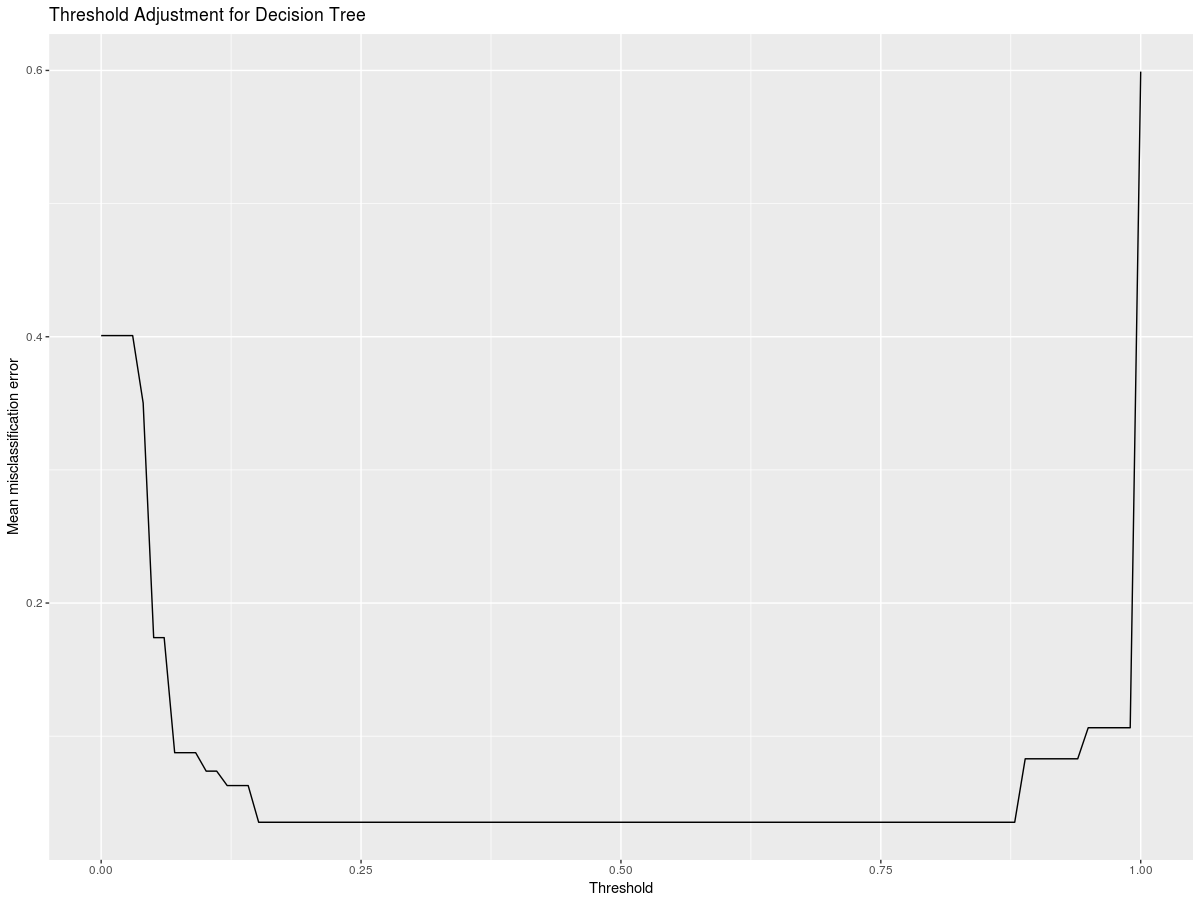
Finally, for the KNN model, the K parameter was considered. This parameter relates to the number of neighbours to consider when classifying a new query. Initially, the values of 1, 3, 5, 10, 20 and 30 were employed for tuning, and subsequently, the values of 4, 6, 8, 10, 12, 14, 16 and 18 were considered.

After tuning, the iterations generated for each model were visualized as appropriate using either line plots or heatmaps. The most ideal parameters was identified as the point which minimized the mmce measure. Tuned learners were then created by fusing the optimized parameters with their associated learners.

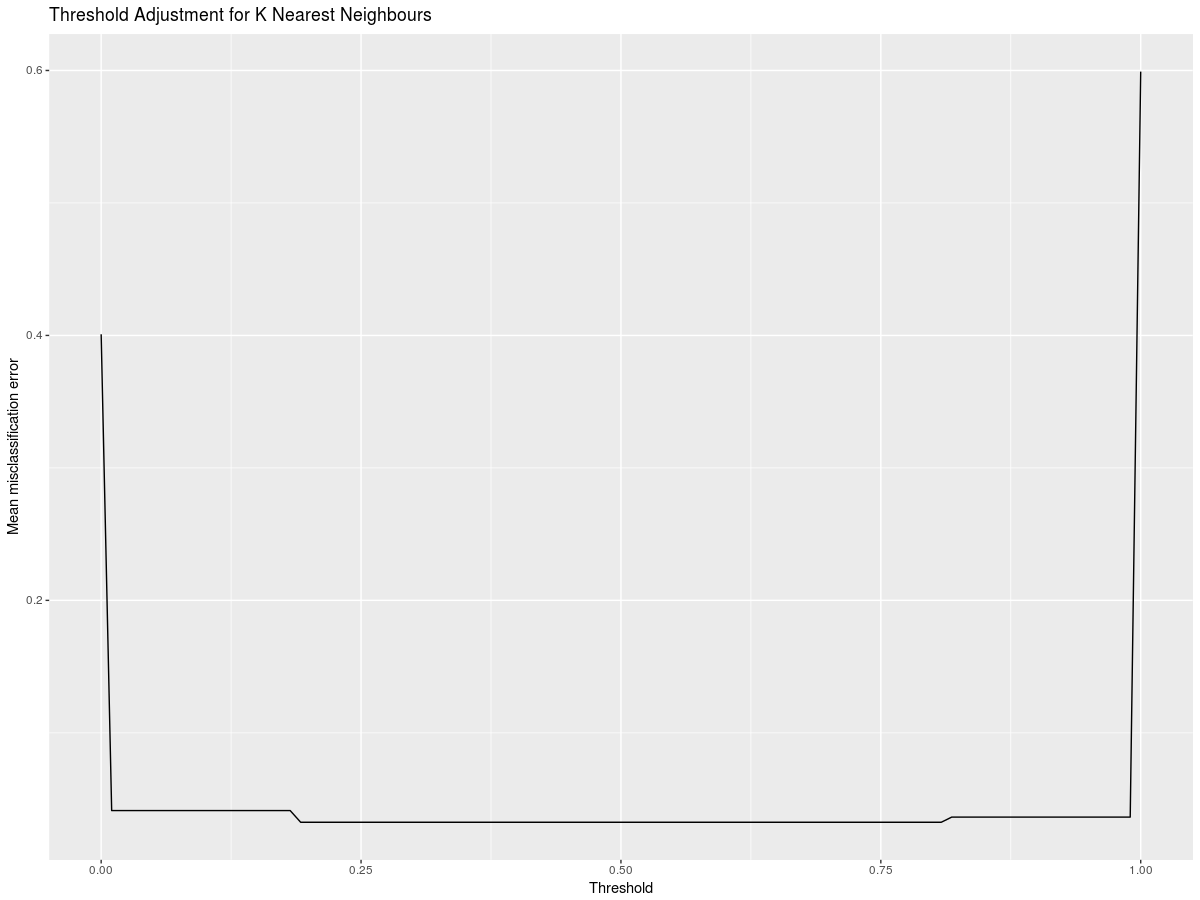
*Prediction Threshold Assessment and Adjustments*

Threshold Adjustment

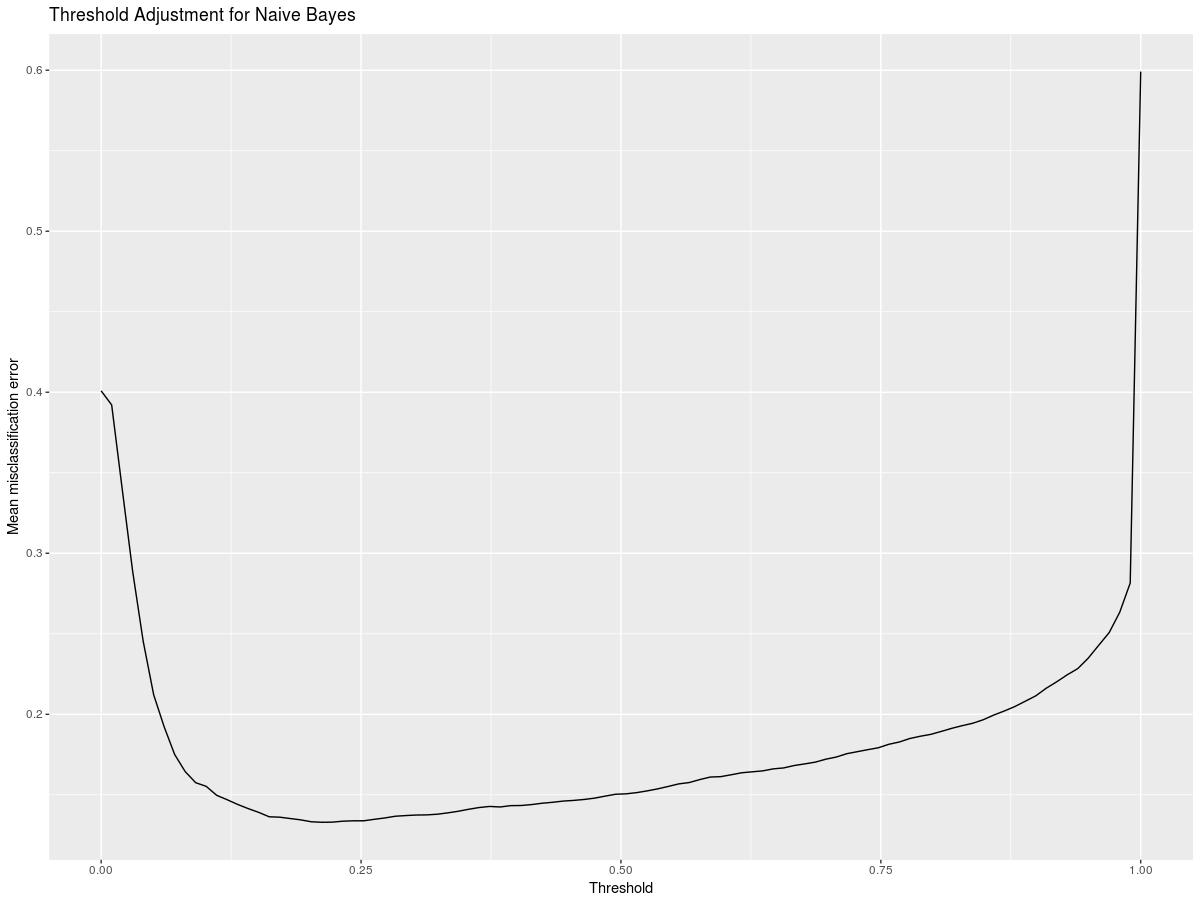
We chose to use the mean value for the threshold value.



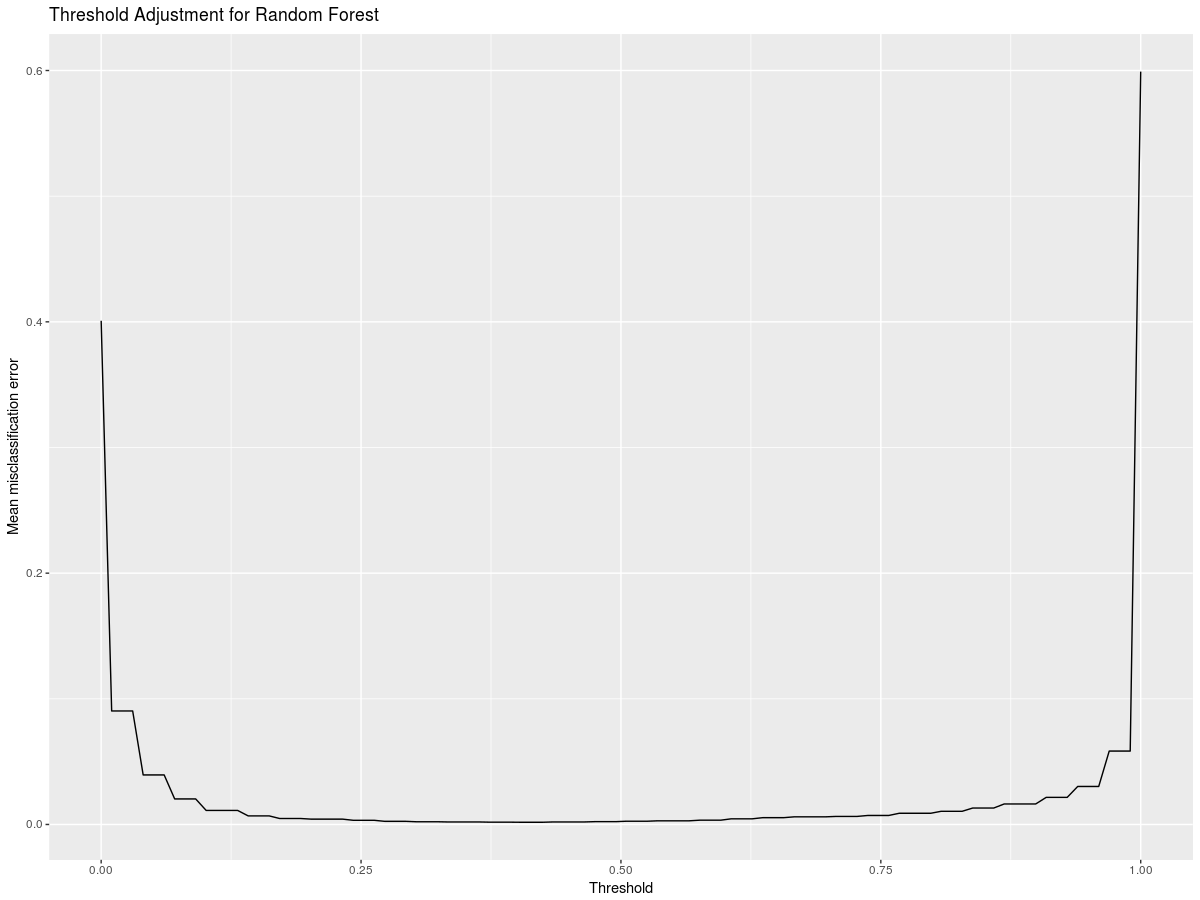
For the decision tree model, a threshold of 0.52 was used.



For the K Nearest Neighbor model, a threshold of 0.5 was used



For the Naive Bayes model a threshold value of 0.21 was used.



For the Random Forest model, a threshold value of 0.41 was used.

**Results and Analysis**

**Decision Tree**

Confusion Matrix:

|  |
| --- |
| predicted |
| true failed successful -err.- |
| failed 6878 311 311 |
| successful 113 4697 113 |
| -err.- 113 311 424 |

Prediction Accuracy:

|  |
| --- |
| F1 Recall Precision |
| 0.970 0.957 0.984 |

**K-Nearest Neighbour**

Confusion Matrix:

|  |
| --- |
| predicted |
| true failed successful -err.- |
| failed 6960 229 229 |
| successful 161 4649 161 |
| -err.- 161 229 390 |

Prediction Accuracy:

|  |
| --- |
| F1 Recall Precision |
| 0.973 0.968 0.977 |

**Naive Bayes**

Confusion Matrix:

|  |
| --- |
| predicted |
| true failed successful -err.- |
| failed 6330 859 859 |
| successful 736 4074 736 |
| -err.- 736 859 1595 |

Prediction Accuracy:

|  |
| --- |
| F1 Recall Precision |
| 0.888 0.881 0.896 |

**Random Forest**

Confusion Matrix:

|  |
| --- |
| predicted |
| true failed successful -err.- |
| failed 7166 23 23 |
| successful 5 4805 5 |
| -err.- 5 23 28 |

Prediction Accuracy:

|  |
| --- |
| F1 Recall Precision |
| 0.998 0.997 0.999 |

Discussion:

* Consider other learners
* Dataset size