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 more appropriate features were introduced. 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 feature selection and hyper-parameter tuning for each model is then reported. Prediction threshold adjustment for each model is then described, and subsequently the most ideal model identified from each classification type is subjected to evaluation in order to compare performancs. 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 impossible due to processing capabilities. As such, a random sample of 40,000 was taken from the full dataset. All further analyses utilised this sample. 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 feature selection and parameter tuning.

*Feature Selection*

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. Here, 3-fold cross-validation stratified sampling was used to account for the slight target level imbalance and mean misclassification error (mmce) was used as the optimization measure. To get an idea of the relative importance of the descriptive features, the information gain and chi-squared values for each feature was plotted via bar charts. Simultaneous Pertubation Stochastic Approximation (SPSA) was then performed on the training set to gain insight into the ideal number of features to include in later analysis. Using this information, a general set of potential feature values was defined, which included values 2, 3 and 4. Feature tuning was then carried out using the filter approach based on feature chi-squared values. Here, a grid search was used over the potential feature values set. For each classifier, a new fused learner was then defined with the number of features to use set to the optimal value identified in the grid search.

*Hyper-Parameter Tuning*

The fused learners defined in the previous section were then subjected to hyper-parameter tuning. In all hyper-parameter tuning analyses, 5-fold cross-validation stratified sampling was used and mmce was employed as the optimization measure.

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 observations required in a leaf node, respectively. For the minsplit parameter, values considered were 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. As will be shown in the results section, the feature selection tuning applied previously ultimately found that the random forest classifier was optimized when the number of features used was 3. As such, for the mtry parameter 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 a more specified search was conducted using the values of 1-8.

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

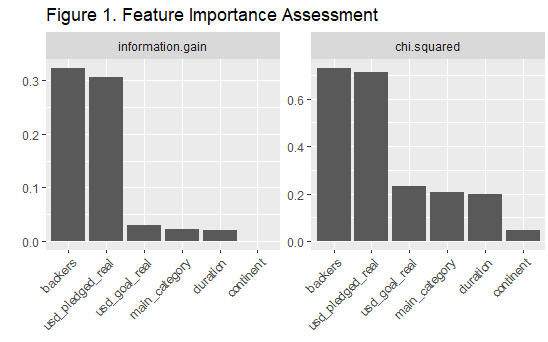
*Evaluation*

The tuned learners were then trained on the training classification task to create prediction models. A new classification task was defined for the test set, and the models were used to make predictions on this task. Using this information, the optimal probability threshold was determined by comparing the mmce values across variations of the prediction threshold and a plot was used to help visualize this. The optimal threshold predictions were used for further performance evaluation. Here, confusion matrices were constructed for each model and their associated mmce scores were compared. Furthermore, the precision, recall and F-1 measures for each model were also compared.

**Results:**

*Feature Selection*

The information gain and chi-squared feature value plots are shown below in Figure 1.

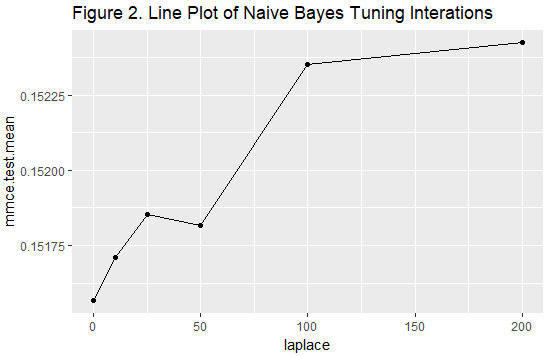


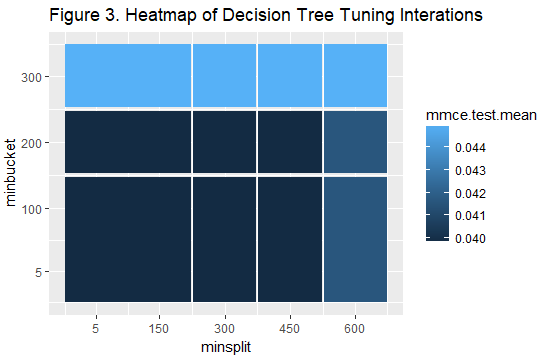
Here it is clear that the backers and the amount pledged are apparently very important features in relation to the target. All other features appear to have similar importance, except for continent, which is very low.

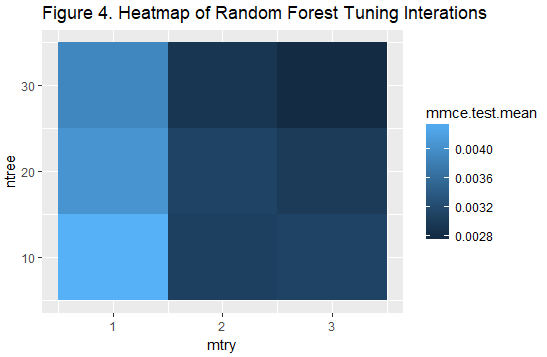
Running feature selection using the filer approach showed that the optimal number of features to use for the naïve Bayes classifier was 4, and had an average mmce of 0.152. The associated features included backers, usd\_pledged\_real, usd\_goal\_real and main\_category. The remaining classifiers were all shown to be optimized with 3 features: backers, usd\_pledged\_real and usd\_goal\_real. The average mmce of the optimized decision tree, random forest and KNN classifiers were 0.040, 0.004 and 0.006, respectively.

*Hyper-Parameter Tuning*

The results of the hyper-parameter tuning for each of the fused learners defined from the feature selection are shown in the plots below.







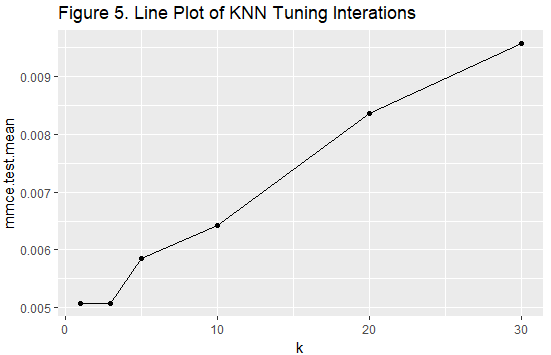
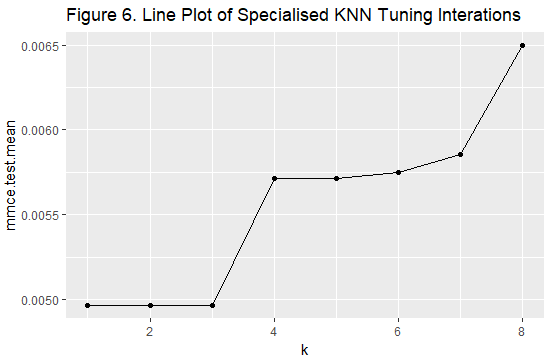


Figure 2 above highlights that the ideal value of the laplace parameter here is 0. This suggests that smoothing is not necessary for the dataset being analyzed. The mmce at laplace = 0 was 0.152.

For the decision tree parameters, Figure 3 shows that there is little variation in mmce for minbucket values between 5-200 and for minsplit values between 5-450. Values greater than the maximum of these ranges results in poorer mmce performance. The optimal values for the minbucket and minsplit parameters was shown to be 200 and 300, respectively, and the associated mmce for these values was 0.040.

The random forest heatmap in Figure 4 shows that the mtry value of 4 and ntree value of 30 was able to produce the best mmce out of all other combination tested. The associated mmce of the optimized random forest was 0.003.

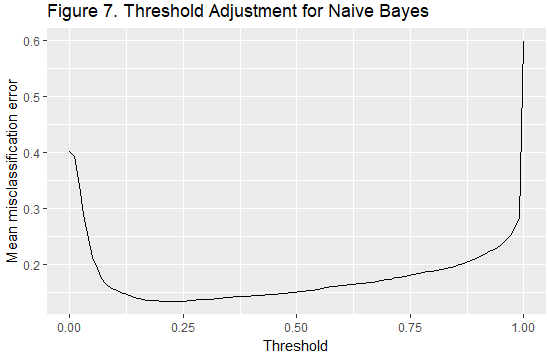
Finally, the plot of K parameter tuning shown in Figure 5 shows that the classifier is optimized for smaller values of k - apparently in the range 1-5. As such further investigation into the optimal k was conducted via a more specified grid search for k values between 1-8. The result plot is shown below in Figure 6.



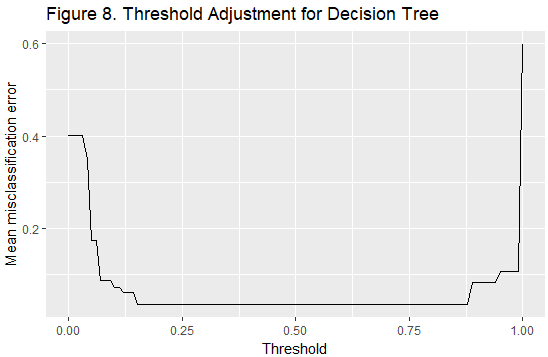
The above plot clearly shows that k values from 1-3 reduce mmce below other higher values. The median, 2, was taken as the optimal k value, which had an associated mmce of 0.005.

*Threshold Adjustment*

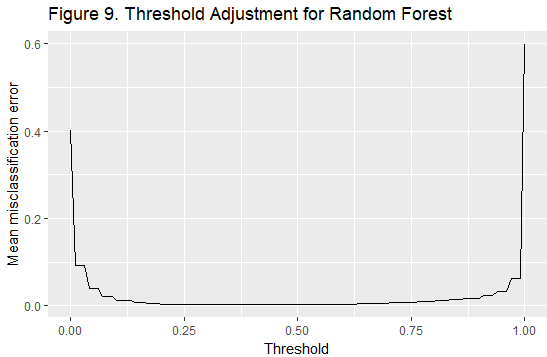
The plots below demonstrate variation in mmce for each model for different prediction thresholds on the test set.



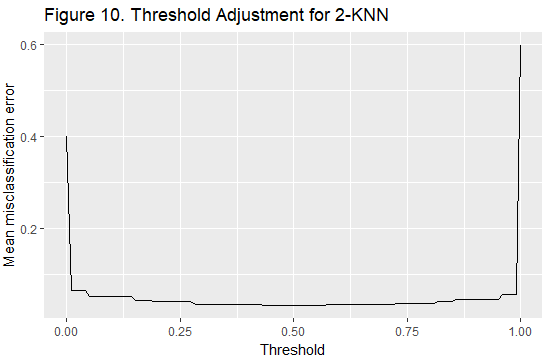
For the Naive Bayes model, a threshold value of 0.21 was found to be ideal.



For the decision tree model a threshold value of 0.51 was used.



A threshold value of 0.4 was used for the random forest.



A threshold value of 0.5 was used for the KNN model.

The optimal threshold values for minimizing mmce were then extracted and test predictions were remade using the new thresholds. This change did not affect the decision tree or KNN classifiers’ performances, which retained their unadjusted mmce values of 0.035 and 0.033, respectively. This is not surprising given the long plateau evidenced in Figures 8 and 10. The other two models, however, did show improvement upon threshold readjustment, with the naïve Bayes’ mmce dropping to 0.133 and the random forest mmce dropping to 0.002.

*Confusion Matrices and Summary Table*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 1. Naïve Bayes Confusion Matrix** | | | | |
|  |  | Predicted | | |
|  |  | Failed | Successful | err. |
| Observed | Failed | 6330 | 859 | 859 |
| Successful | 736 | 4074 | 736 |
| err. | 736 | 859 | 1595 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 2. Decision Tree Confusion Matrix** | | | | |
|  |  | Predicted | | |
|  |  | Failed | Successful | err. |
| Observed | Failed | 6878 | 311 | 311 |
| Successful | 113 | 4697 | 113 |
| err. | 113 | 331 | 424 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 3. Random Forest Confusion Matrix** | | | | |
|  |  | Predicted | | |
|  |  | Failed | Successful | err. |
| Observed | | Failed | 7171 | 18 | 18 |
| Successful | 12 | 4798 | 12 |
| err. | 12 | 18 | 30 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 4. KNN Confusion Matrix** | | | | |
|  |  | Predicted | | |
|  |  | Failed | Successful | err. |
| Observed | Failed | 6960 | 229 | 229 |
| Successful | 161 | 4649 | 161 |
| err. | 161 | 229 | 390 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 5. Performance Summary Table** | | | | |
| Model | F1 | Recall | Precision | MMCE |
| KNN | 0.973 | 0.968 | 0.977 | 0.033 |
| Naive Bayes | 0.888 | 0.881 | 0.896 | 0.133 |
| Random Forest | 0.997 | 0.997 | 0.998 | 0.002 |
| Decision Tree | 0.970 | 0.957 | 0.984 | 0.035 |

Clearly, all classifiers performed very well at predicting on the test data. The random forest model can be seen to have almost perfect performance across all measures, while the KNN and decision tree models had very similar performances to each other, but were slightly worse than the random forest. The naïve Bayes model performed the worst out of all models, however its accuracy is still objectively quite good.

Discussion:

This analysis aimed to identify the best classification model to use for prediction whether a Kickstarter project would be successful. The results presented here ultimately suggest that while most model