Individual Coursework Submission Form

Specialist Masters Programme

Surname: Perera	**First Name:** Joshua
MSc in Business Analytics	**Student ID number:** [redacted]
Module Code: SMM636	
Module Title: Machine Learning	
Lecturer: Rui Zhu	**Submission Date:** 26/03/2021

Declaration:

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the coursework instructions and any other relevant programme and module documentation. In submitting this work, I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

We acknowledge that work submitted late without a granted extension will be subject to penalties, as outlined in the Programme Handbook. Penalties will be applied for a maximum of five days lateness, after which a mark of zero will be awarded.

Marker's Comments (if not being marked on-line):

Deduction for Late Submission:

Final Mark: %

Table of Contents

Question 1

The dataset used in Question 1, involving data collected by Dr. Hans Hofmann of the University of Hamburg, classifies a person who wants to take out credit from a bank into good or bad credit risk based on several attributes such as account status, credit history, purpose of loan, savings accounts and more. In total, there are 1,000 entries and 20 different attributes.

Part 1 – Decision Trees

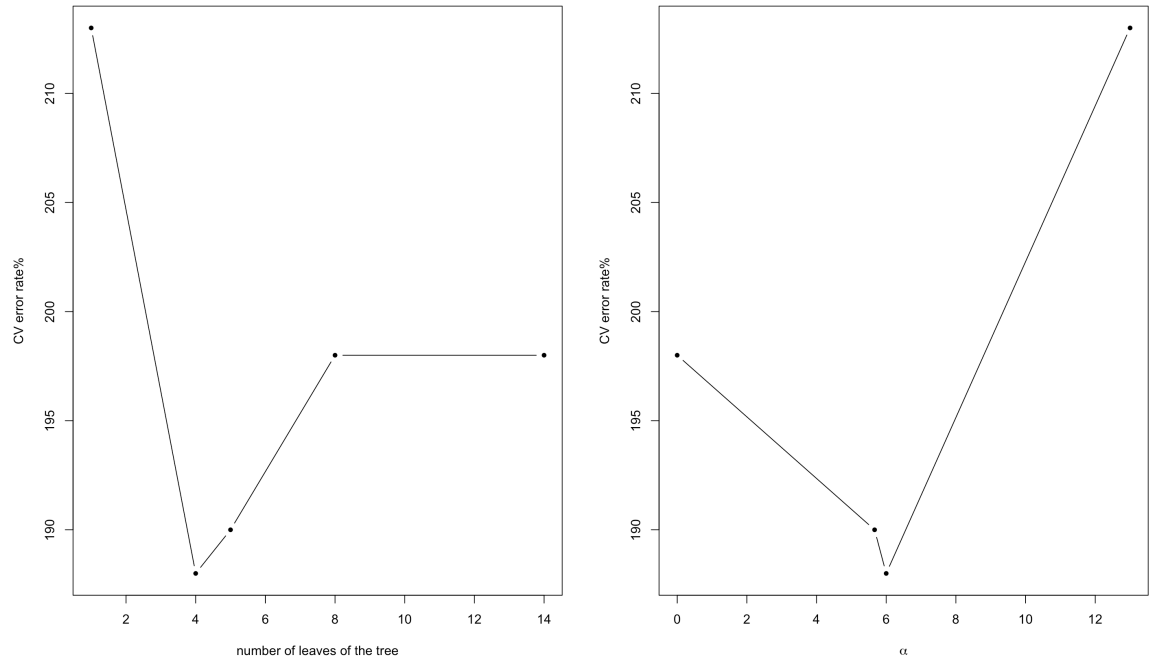
Decision trees are a powerful predictive modelling and classification tool which predict the value of a target variable through learning the specific decision rules from the training data. Moreover, they operate by splitting the feature space into several (often rectangular) regions which do not overlap. In many cases, trees can be very large and complex, containing many terminal nodes or leaves, which may make the model hard to interpret or visualise well. Thus, pruning the tree is useful as it solves the issue of complexity and negates the decision tree from overfitting the training data, which improves predictive accuracy (Nseerullah, 2018). Although, finding the optimal number of terminal nodes/leaves and the size of the tree is vital.

Prior to implementing the decision tree and its pruned version, the German Credit data is split into training and test sets, at 70% and 30% respectively, so that the decision tree can learn from training set and evaluate its predictions based on the test set. The training set is often larger than the test set since there will be less variation in the model parameters if the model is trained with more data.

As evident from Appendix A, which shows the original decision tree for the German Credit data, the results produce a somewhat big and complex tree (containing 14 terminal nodes/leaves) suggesting that the tree has most likely been overfitted. A residual mean deviance of 0.8671 and misclassification error rate of 0.2114 can also be observed. In theory, residual deviance is a measure of how well the target variable is predicted when the predictors are included, as opposed to null deviance where the predictors are excluded and only the intercept is used (datascienceblog.net, 2018). A smaller value or value close to 1 deems the chosen model appropriate. Based on the results produced from the original decision tree, the residual mean deviance is quite close to 1, which is quite good, while the misclassification rate is relatively low (classification rate is above the 50% random guessing threshold) – although, inspecting the confusion matrix when pruning the decision tree is ideal before reaching any conclusions.

As aforementioned, when producing the pruned decision tree, the optimal number of terminal nodes is crucial. To assess the optimal number, 10-fold cross-validation using the *cv.tree()* function is used and the results obtained are below:

Figure 1: Cross-Validation Results



From Figure 1, the optimal number of terminal nodes/leaves is chosen based on which produces the lowest cross-validation error rate (%), which in this case equates to 4 terminal nodes/leaves shown by the graph on the left. The graph on the right shows the cross-validation rate against α, which represents a tuning and complexity parameter. Based on these results, the pruned decision tree is produced using *prune.misclass()*, with the parameter *best = 4* included.

Figure 2: Pruned Decision Tree

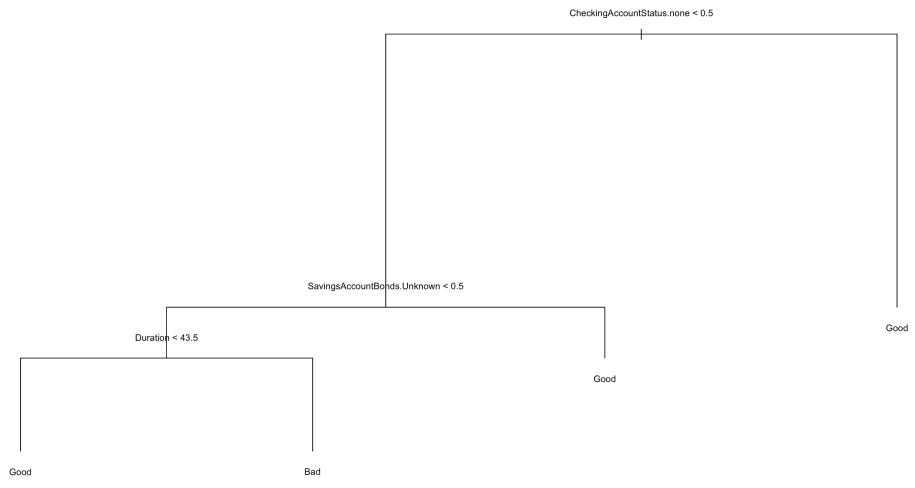


Figure 2 shows the resulting smaller tree with the 4 terminal nodes/leaves which is much simpler to visualise and interpret. When observing the residual mean deviance, a value of 1.051 is obtained – which is closer to 1 than the value from the original decision tree. This suggests that the pruned tree is a better model for the data used. However, in terms of the misclassification error rate, it seems to have increased after pruning to a value of 0.2443 from 0.2114. The change is marginal, but the confusion matrix should be examined.

Table 1: Test Set Confusion Matrix for Pruned Decision Tree Model

	Actual: Bad	Actual: Good
Predicted: Bad	33	34
Predicted: Good	57	176

As observable from Table 1, the predictions for Good credit risk have been mostly correctly classified, whilst the predictions for Bad credit risk have been misclassified. In other words, the model predicted many people with Bad credit risk to have a Good credit risk. In practice, this can be detrimental to a bank as there is a high chance that the people falling into this false positive category will default on their loans/credit. The test error rate gives an idea of how the model performs with data it has not been trained with previously, also known as the test set. **A mean accuracy value of 0.6966667 and test error rate of 0.3033333 is attained** (*test error rate = 1 – mean accuracy*) which suggests that the model performs reasonably well. However, as discernible from when using the random forest model, the decision tree model will be less accurate since one of its characteristics is having lower accuracy but higher interpretability.

Part 2 – Random Forests

Random forests are essentially a collection of several decision trees whereby each tree produces a prediction, which may individually not be accurate, but once averaged with and across all other trees will produce a strong prediction. The main difference between random forests and decision trees is, as previously mentioned, that many decision trees exist within a random forest – however, it is important to consider other differences which can be useful in deciding which model to choose. Due to the complex structure of random forests, they are much harder to interpret and have a higher training time than decision trees, but certainly produce more accurate predictions. As opposed to decision trees, the characteristic that embodies random forests is higher accuracy but lower interpretability. As such, the trade-off is quite clear.

In terms of producing a random forest model for the German Credit data, the train control is first set, with 10-fold cross-validation, to specify the type of resampling and then the model is defined. The results are found in Appendix B. From the accuracy graph and table in Appendix B, the optimal mtry – which is the number of random variables used in each tree – chosen is 16 since it produces the highest accuracy value at 0.7780952. The reason for choosing accuracy as a metric is because it is a universal measure, while kappa specifically works well with more imbalanced data and we do not see significant imbalances in the German Credit data (30% in the Bad class, 70% in the Good class). The performance of the model is again analysed by a confusion matrix.

Table 2: Confusion Matrix for Random Forest Model

	Actual: Bad	Actual: Good
Predicted: Bad	92	118
Predicted: Good	35	455

As evident from Table 2, the model predicts both people with Good and Bad credit risk well since the number of true positives exceeds the false negatives and the number of true negatives exceeds the false positives.

Table 3: Confusion Matrix for Random Forest Model with Optimal mtry

	Actual: Bad	Actual: Good
Predicted: Bad	94	116
Predicted: Good	39	451

The results of the new random forest model in Table 3 show that some observations were classified into the column of *Actual: Bad* from the column of *Actual: Good*. At a glance, there is not much of a change, although an increase from 21.86% to 22.14% is seen in the out-of-bag (OOB) estimate of error rate.

Table 4: Test Set Confusion Matrix for Random Forest Model

	Actual: Bad	Actual: Good
Predicted: Bad	37	22
Predicted: Good	53	188

Furthermore, similar to the approach taken for the decision tree model in Part 1, the test error rate and mean accuracy is obtained. **The test error rate equates to 0.25 whilst the mean accuracy equates to 0.75**. What is clear from these results, in addition to the confusion matrix in Table 5, compared to the decision tree model, is that more observations are classified well to true positives and true negatives. This ultimately shows that the random forest model performs relatively better compared to the decision tree model.

Figure 3: Variable Importance Plots

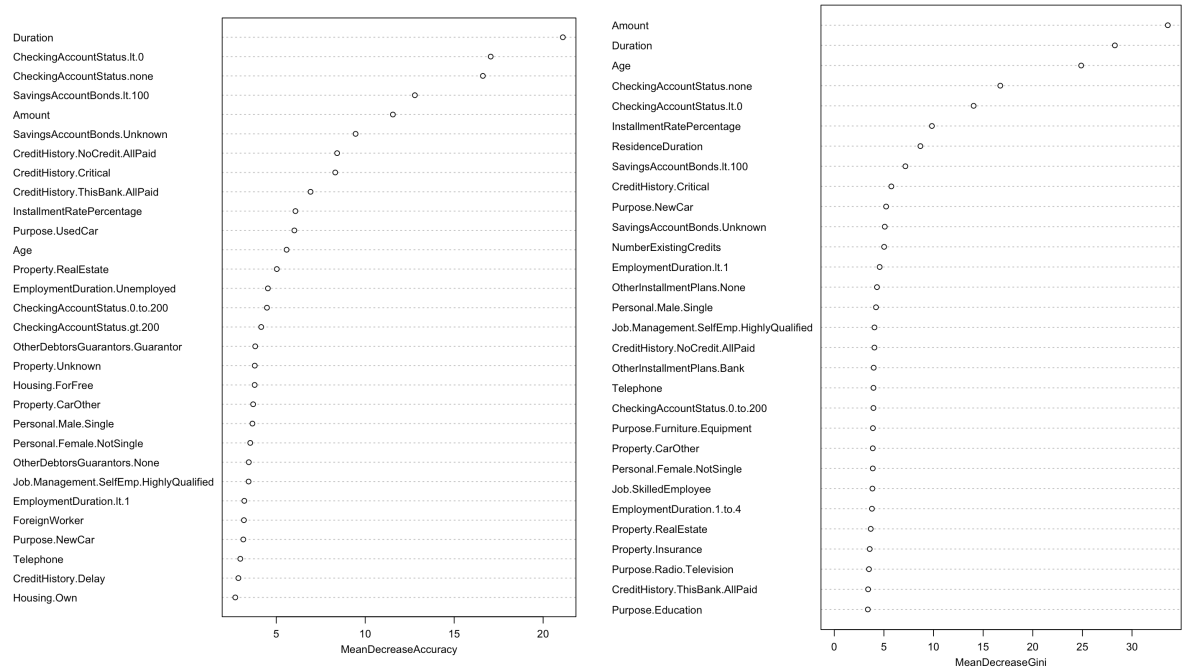
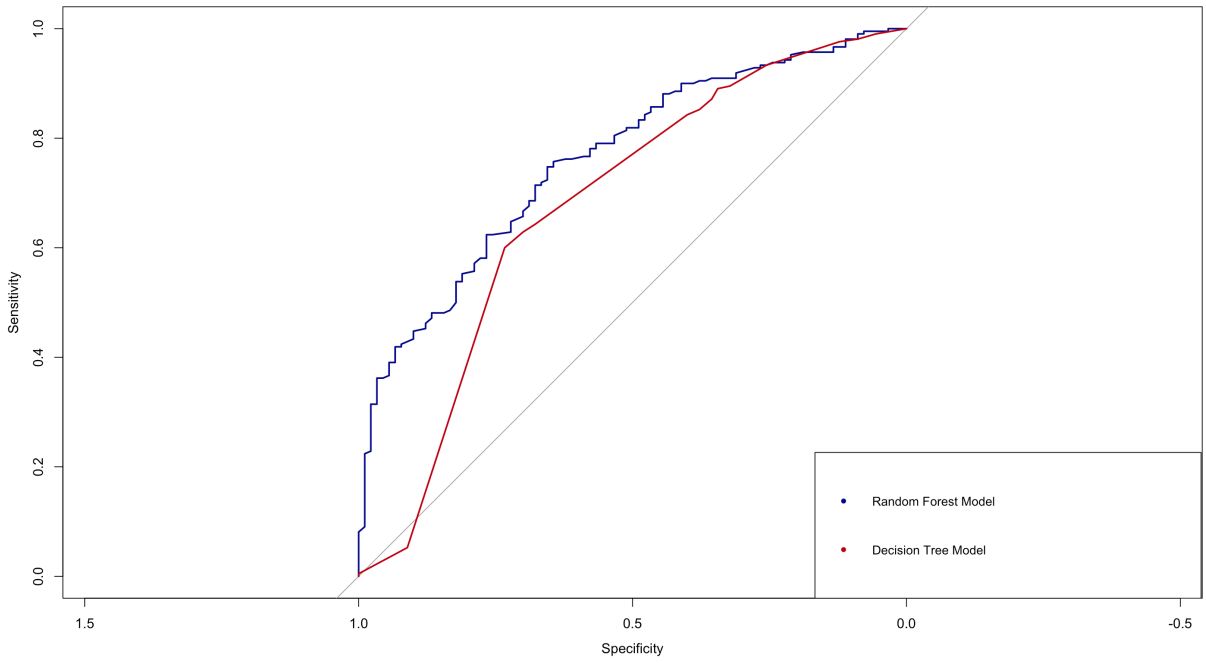


Figure 3 illustrates the variable importance which is measured by the Mean Decrease Accuracy and Mean Decrease Gini Coefficient methods to assess the variables with the most predictive power, and those with the least in order to omit them for better fit. The larger the values of the variables, the more important they are. Firstly, the plot on the left, consisting of the Mean Decrease Accuracy values, assesses by how much the accuracy decreases or changes when a chosen variable is omitted. If the top 3 are selected, *Duration*, *CheckingAccountStatus.lt.0* and *CheckingAccountStatus.none* are the most important variables. Secondly, the plot on the right, consisting of the Mean Decrease Gini values, assesses the gain/decrease of purity when a variable is split. If the top 3 are selected again, *Amount*, *Duration* and *Age* are the most important variables. However, when examining both plots simultaneously, the important variables common in both seem to be *Duration*, *CheckingAccountStatus.lt.0* and *CheckingAccountStatus.none*.

Part 3 – ROC Curves

A receiver operating characteristic (ROC) curve essentially visualises the performance of a classification model at all classification thresholds using two key metrics: the true positive rate (known as sensitivity) and the false positive rate (known as specificity) (Bhandari, 2020). The area under the curve (AUC) is typically the depiction of the ROC which allows the curve to be more interpretable. In theory, the closer the ROC is to the top-left, the better the model is classified, and the closer the ROC is to the 45-degree diagonal, the worse it is classified. These conditions correspond to the AUC values since the higher the AUC, the higher the accuracy and performance of the model.

Figure 4: ROC Curves for Decision Tree and Random Forest Models



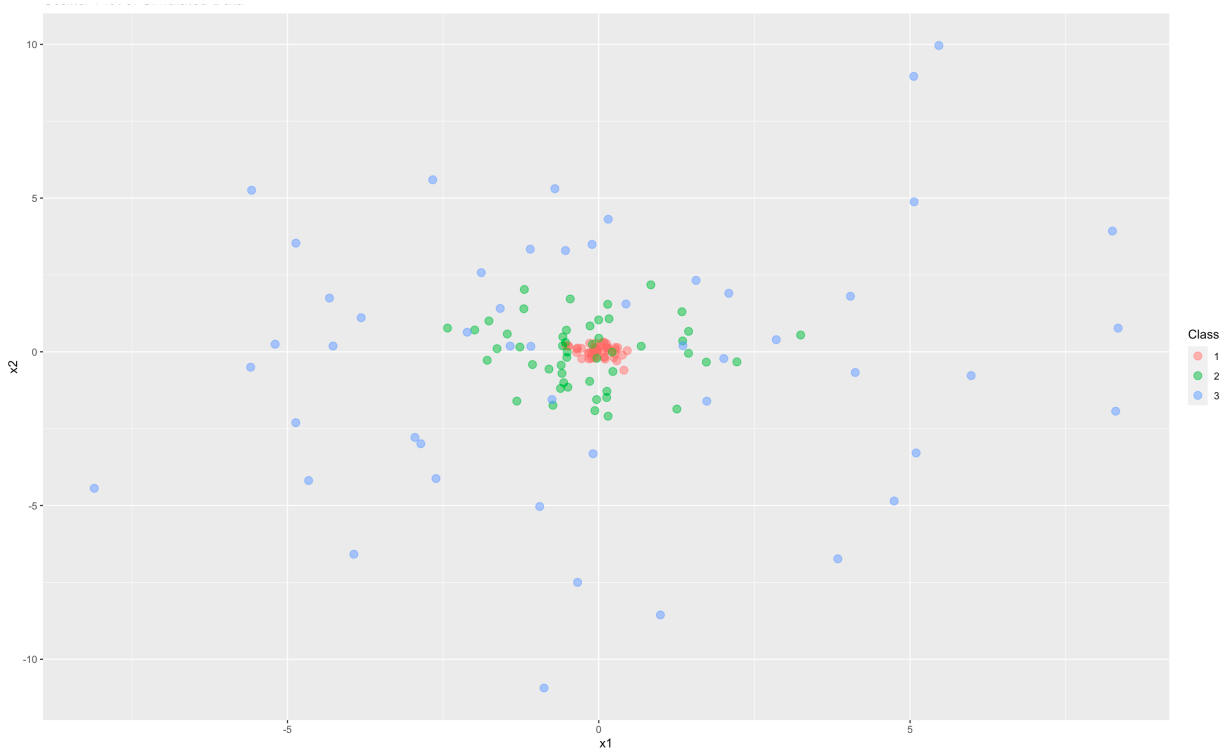
Based on Figure 4, the random forest model seems to perform better, aligning with the advantages random forests have over decision trees mentioned previously, since its curve is closer to the top-left corner of perfect classification than the decision tree model curve. This is further confirmed using the AUC values gained, which are 0.6781 for the decision tree model and 0.7623 for the random forest model. The random forest model is the clear winner, however, it is also important to consider that interpreting it is quite difficult and, after all, it may not necessarily be the best model for the German Credit dataset – hence why other models, such as gradient boosting and neural networks, need to be experimented on this dataset.

Question 2

Simulated Dataset

To generate the three-class dataset with 50 observations in each class and 2 features, the *rnorm()* function is used to produce 3 varying matrices containing the 3 classes – all with mean equal to zero and standard deviations of 0.2, 0.3 and 0.45 respectively – to ensure they are not linearly separable. The labels of the classes (1, 2 and 3) are then assigned to them, and finally, everything is combined into one data frame using a combination of the *rbind()* and *cbind()* functions. The columns are also renamed for simplicity purposes.

Figure 5: Simulated Data Scatter Plot



As evident from Figure 5, it appears quite difficult to linearly separate the different classes, with all 3 of the classes following a circular/ellipse shape, which means that the goal of creating a dataset that is non-linearly separable has been achieved. The data is then split into training and test sets with 50% of the total observations assigned to each set.

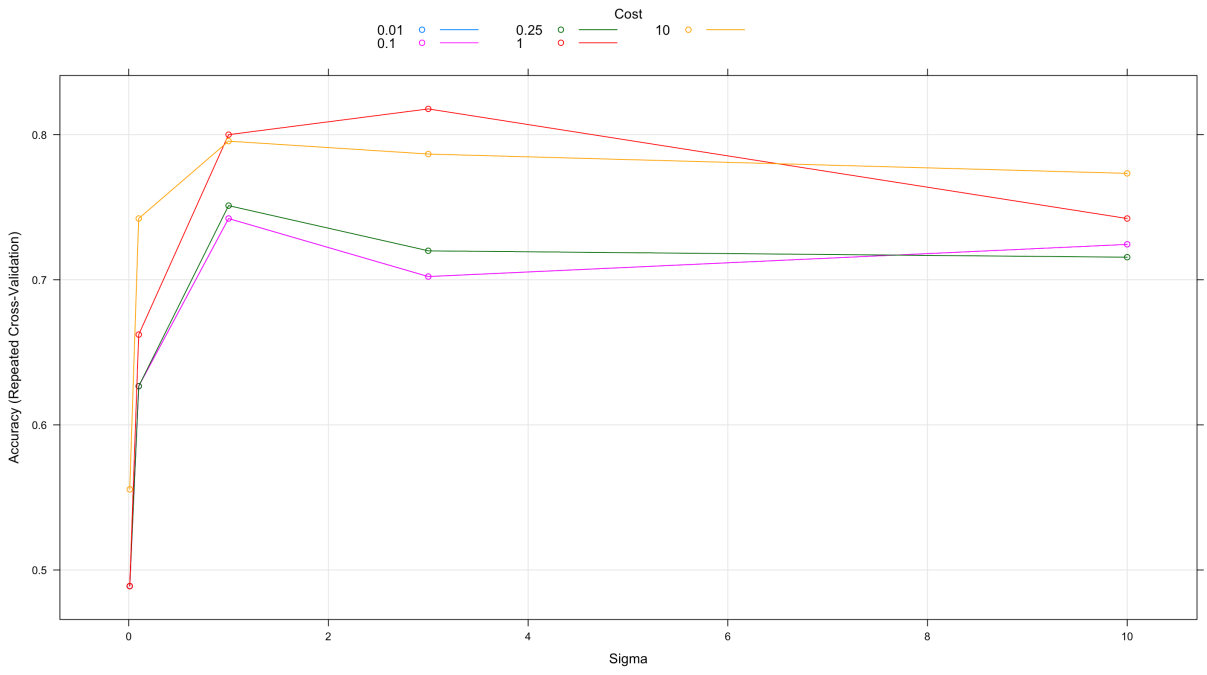
Radial Basis Function (RBF) Kernel

Primarily, kernel functions are essentially a way of using a linear classifier to solve a non-linear task, and this is done by the transformation of the training data and ultimately returning the inner product between two observations in a standard feature space. The Radial Kernel Basis

Function (RBF) is a kernel function where its value depends only on the distance from the origin. The RBF Kernel is commonly used since it has similar characteristics to the k-nearest neighbours methodology and it only requires to store the support vectors during the training process, instead of storing the whole dataset, which solves the problem of space complexity.

When running the RBF Kernel on the simulated dataset, the parameters associated with each model are tuned by 5-fold cross-validation using *fitControl* and the data is also scaled to avoid attributes with extreme means or standard deviations from skewing or influencing the results. The results of the RBF Kernel are shown below, where C and sigma are tuning parameters which represent the severity of the violations to the margin and hyperplane that can be tolerated. In theory, C assesses the bias-variance trade-off which means that if C is a large value, there is more bias but low variance and this means that there will be a wider margin and many observations will violate the margin – whereas if C is a small value, there is less bias but high variance. Generally-speaking, there are more support vectors when C is high. However, it is important to note that when using the *caret* library (as well as most other R libraries), a large C value actually leads to a narrower margin, which is the opposite case to the theoretical aspect. When tuning sigma and C, a grid is tuned such that sigma and C hold values the values: 0.01, 0.1, 1, 3 and 10 (for sigma) and 0.01, 0.1, 0.25, 1 and 10 (for C).

Figure 6: Accuracy of C and Sigma Values for RBF Kernel

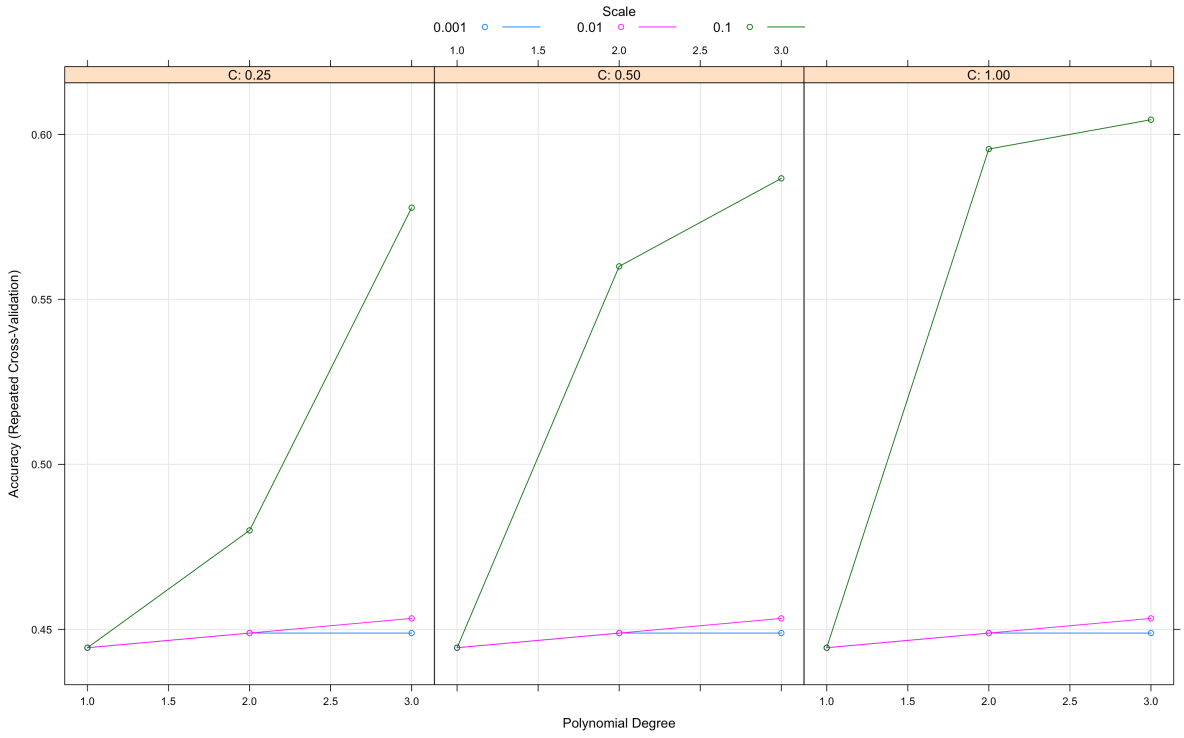


Based on Figure 6, the optimal sigma value chosen is 3 and the optimal C value chosen is 1 since they showed the largest accuracy. To then test the model on the test data, the test error is computed. **A mean accuracy of 0.8533333 and test error rate of 0.1466667 is obtained**, which shows that the RBF Kernel performs well and produces strong predictions.

Polynomial Kernel

Polynomial Kernels differ to RBF Kernels in the sense that they represent the similarity of training observations in a feature space of polynomials of the original variables which, in other words, means that the inner product is calculated by increasing the value of the power of the kernel. Also, Polynomial Kernels fit a support vector classifier in a high-dimensional space. The data is scaled within the *train()* function and the results of training the dataset with the Polynomial Kernel are shown below:

Figure 7: Accuracy of Polynomial Degree and C Values for Polynomial Kernel



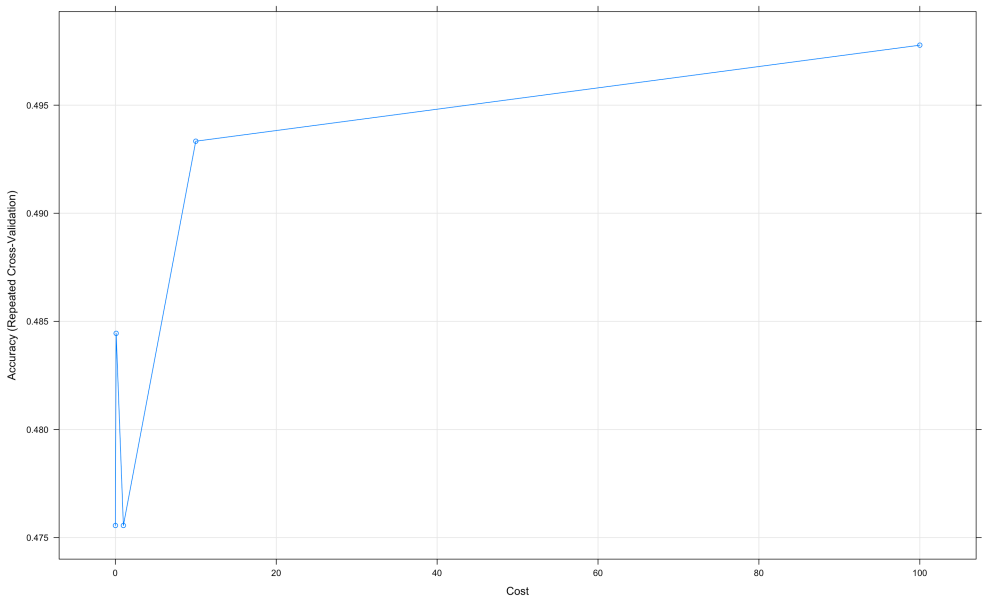
Based on Figure 7, the final values chosen for the model are degree = 3, scale = 0.1 and C = 1 given they were attained by assessing which had the highest accuracy. At all 3 C values (0.25, 0.50 and 1.00), the accuracy of degree = 3 was 0.5644444, 0.5955556 and 0.6044444 respectively. **Moreover, when assessing the mean accuracy and test error rate, the values obtained are 0.6933333 and 0.3066667 respectively**. The results are noticeably worse than

that of the RBF Kernel since the mean accuracy has decreased and the test error rate has increased.

Support Vector Classifier (SVC) / Linear Kernel

The SVC is the generalisation of the maximal margin classifier to a non-separable case and it is a classifier that essentially is based on a hyperplane that does not perfectly separate classes – which means that it allows the Support Vector Machine to misclassify a certain number of training observations. This may seem counterintuitive, but the method may lead to greater robustness and better classification of most of the training data. Also, the SVC measures the similarity of pairwise observations using the Pearson correlation The SVC and Linear Kernel are the same thing, hence why the training procedure involves using the *svmLinear* method. When applying the Linear Kernel to the German Credit data, a similar approach to the RBF Kernel is taken whereby a grid for C is defined with values: 0.01, 0.1, 1, 10 and 100. As with the other kernels, the data is scaled within the *train()* function and the results are shown below:

Figure 8: Accuracy of C Values for SVC/Linear Kernel

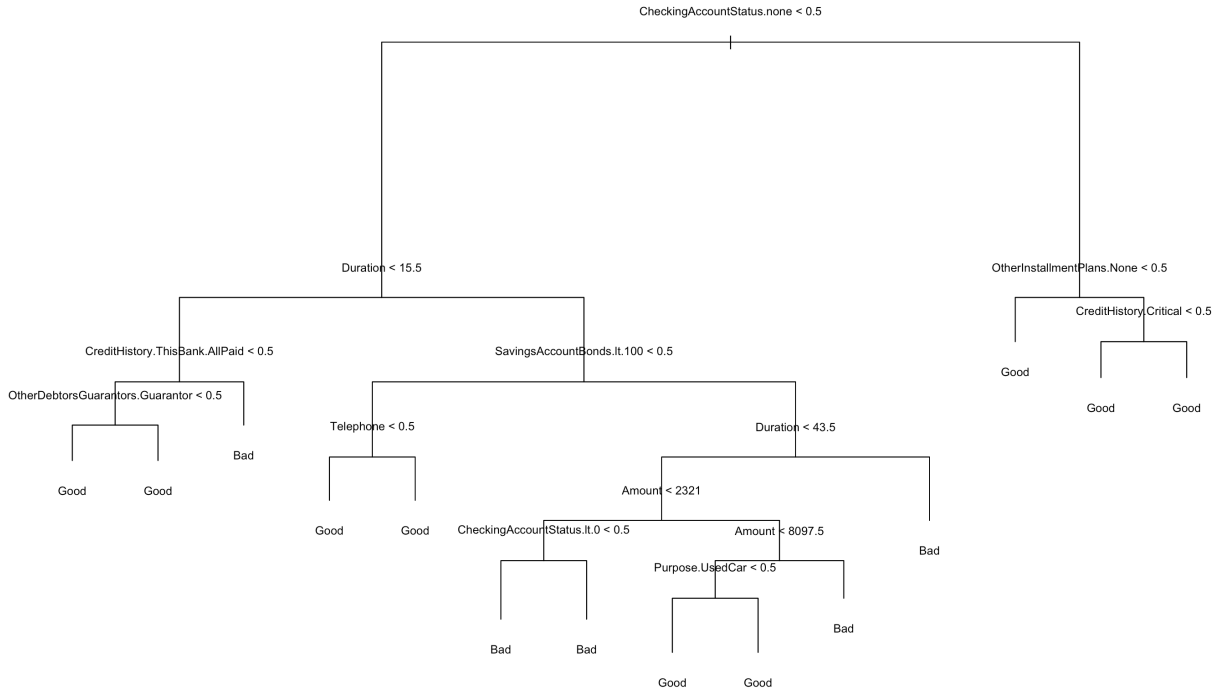


Based on Figure 8, the final value used for the model was C = 100 since it produced the highest accuracy at 0.4977778, which is quite high suggesting that the model allowed for quite a high level of tolerance towards the violation of the margin. **In terms of testing the model on the test data, the mean accuracy and test error rate obtained are 0.5466667 and 0.4533333 respectively**. The values are relatively worse than the values produced from the RBF Kernel and Polynomial Kernel models, meaning that out of all the 3 kernels, the RBF Kernel performed best when tested on the test data.

Appendix

Appendix A – Decision Tree of German Credit Data

Original Decision Tree

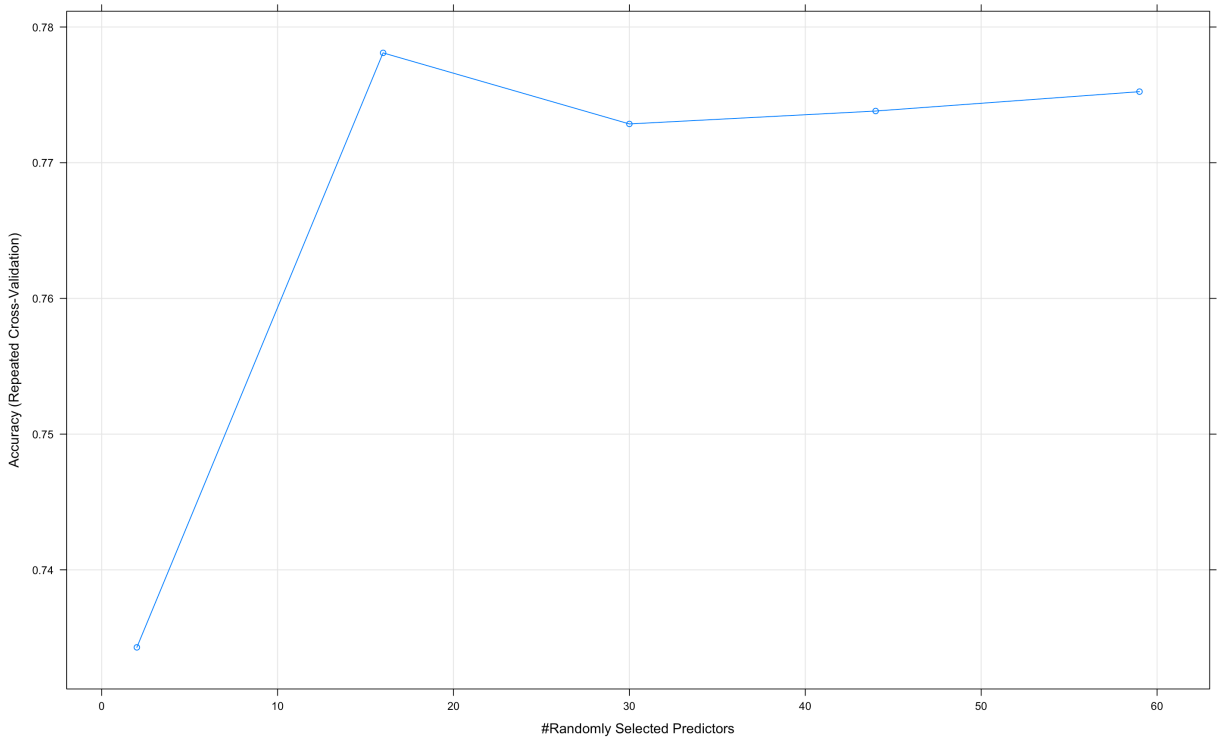


Appendix B – Choosing the Optimal mtry Value

Accuracy and Kappa Values for Random Forest

mtry	Accuracy	Kappa
2	0.7342857	0.1766536
16	0.7780952	0.4132262
30	0.7728571	0.4077507
44	0.7738095	0.4137446

Accuracy Graph



References

Nseerullah. (2018). [online] THE PROS AND CONS OF PRUNING IN CLASSIFICATION. Available at: https://www.worldresearchlibrary.org/up_proc/pdf/2005-154323163309-10.pdf [Accessed 26 Mar. 2021].

datascienceblog.net. (2018). *Interpreting Generalized Linear Models*. [online] Available at: https://www.datascienceblog.net/post/machinelearning/interpreting_generalized_linear_models/.

Bhandari. (2020). *AUC-ROC Curve in Machine Learning Clearly Explained*. [online] Available at: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/.