SMM636 Machine Learning
Individual Coursework

Kejia Tang

March 24, 2021

Contents

1 Question 1

`GermanCredit` from `caret` package is used to apply decision tree and random forest algorithm in this problem set. The two classes for a total number of 1000 credit records are good (700) and bad (300). There are 61 predictors related to people attributes in the original data set. Close investigation reveals that there are 2 predictors that have the same values for both of two classes. Therefore, the 2 variables `Purpose.Vacation` and `Personal.Female.Single` are deleted and the remaining 59 predictors are used to conduct the classification task. In the subsequent analysis, 70% of the data are used to train the classification model and the final model is tested on the remaining 30% data.

1.1 Decision tree model

A decision tree model is trained using 10-fold cross-validation via `rpart` function from `rpart` package. The parameters specified in `rpart.control` refer to the number of cross-validation `xval=10`, the minimum number of instances in terminal leaf `minbucket=2` and the threshold of complexity parameter `cp = 0.01`. The relative cross-validation error rates are plotted against tree sizes and cp values in Figure 1. The output `cptable` of the fit, which contains the error rates in the cross-validation, indicates that a good choice of cp for pruning the tree is the value for which the error rate is below the horizontal line that is 1SE above the minimum error. Hence, a cp value of 0.024 which produces the minimum cross-validated error is chosen to prune the tree of 6 terminal leaves.

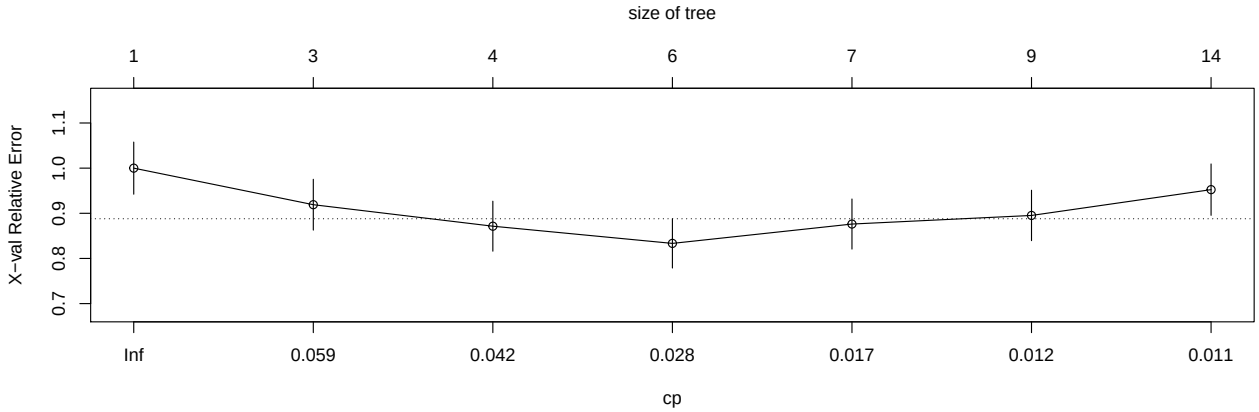

Figure 1: The cross-validated errors against tree size and cp for decision tree.

The pruned tree is shown using `rpart.plot` package in Figure 2. In this binary classification model, the green node represents a majority of `Good` instances in that specific node while the red node refers to a majority of `Bad` in the node. There are 70% of instances being `Good` in the root node and the first predictor to split the tree is `CheckingAccountStatus.none`. At each split, an instance with the answer `yes` to the splitting rule would fall in the left branch while `no` would lead to the right branch. A total number of 5 splits are conducted, leading to the 6 terminal leaves, among which the darker colour indicates a purer node that has a higher probability of the majority class. The "purest" `Good` node accounts for 40% of all the training instances, among which 88% are `Good` classes, while only 2% of the observations fall in the "purest" `Bad` node, which has a probability of 82% belonging to `Bad` classes. It seems that people without checking account status tend to have `good` credit record, while young people (less than 26 years old) who have credit history of not paying in this bank or not paying all the credits, are more likely to have `bad` credit.

By applying the pruned tree model to the test set, a test error of 0.297 is obtained, which is higher than the training error of 0.224. The outcome indicates that the decision tree performs slightly better in the training data rather than in the test set, suggesting a potential problem of overfitting in decision tree algorithm, which can be improved by using methods such as bagging and random forest.

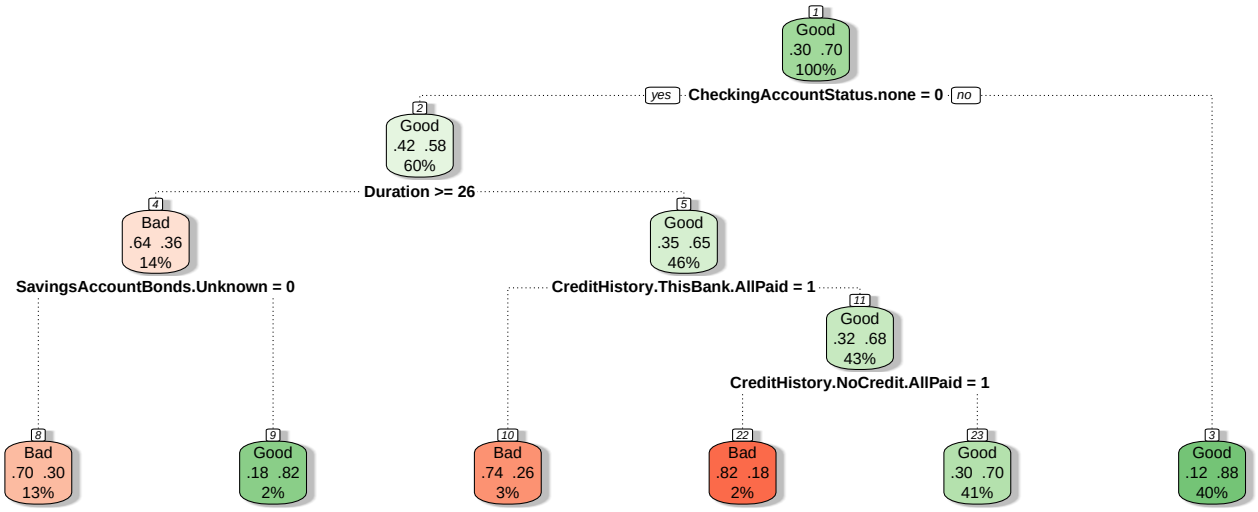

Figure 2: The pruned decision tree of 6 terminal leaves.

1.2 Random forest model

The `caret` package is used to tune random forest models via 10-fold cross-validation. The tuning process is focus on one parameter `mtry`, which refers to the number of variables randomly selected to be sampled at each split. The tuning grid for `mtry` is specified as `mtry=c(2,10,20,30,40,50,59)`. Besides, the number of trees to grow is by default `ntree=500`. The `rf` method is used in `caret` training process and the metric to compare the models is `Accuracy`. The relationship between cross-validated accuracy and `mtry` is shown in Figure 3. The `mtry=10` is selected as the final model produces the highest classification accuracy of 0.761 on the training set. The curve indicates that the cross-validated accuracy fluctuates when including more predictors at each split. It is also noticed that having all the 59 predictors at each split, which is equivalent to bagging method, generates an accuracy slightly lower than the optimal value. The outcome is not surprising as random forest, which draws random subsets of features to train the trees that are more independent, often results in better classification performance. Additionally, the number of predictors selected at each split is usually recommended to be $\sqrt{p} \approx 8$, which is close to the `mtry=10` chosen in this model.

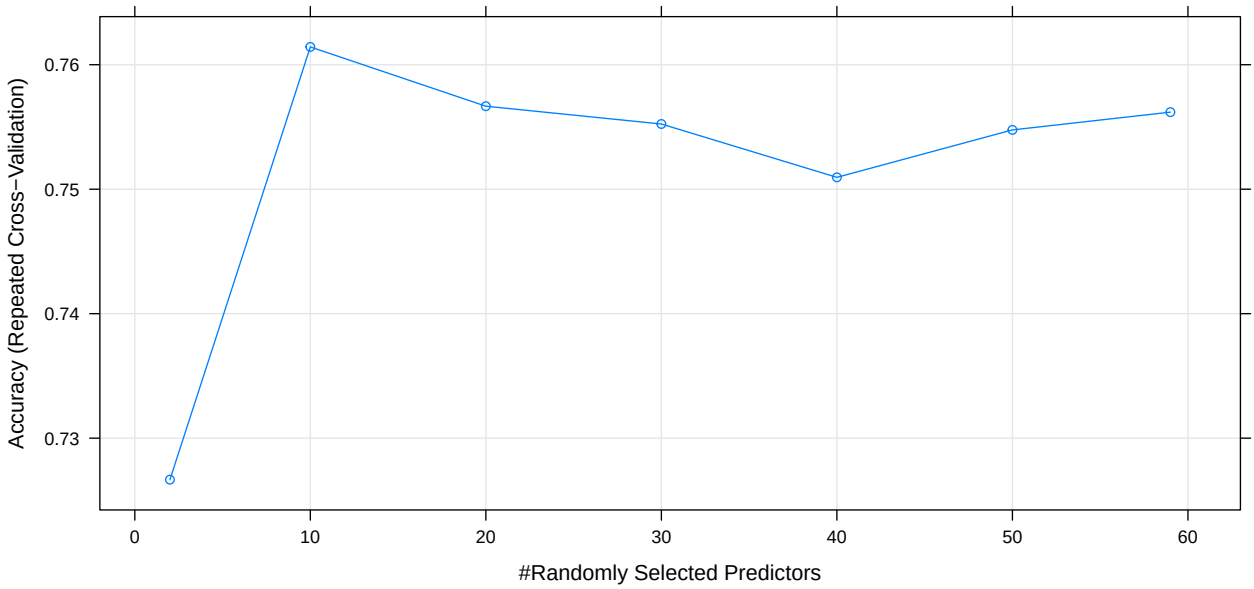

Figure 3: The cross-validated errors against mtry for random forest.

The final random forest model is used to classify the test data and the corresponding error

rate is 0.257, which is lower than the decision tree test error of 0.297. It indicates that random forest algorithm slightly improves the classification accuracy of the model on the test set, as random forest is often used to fix the overfitting problem of decision tree and the model becomes more generalised. To figure out which variables have more significant impact on the classification outcome, variable importance is calculated and the top 10 predictors are plotted with importance value in Figure 4. The importance of each predictor is computed based on the mean decrease of accuracy and it shows that the accuracy decreases the most when excluding the variable `Amount`, followed by `Age` and `Duration`. It makes sense that older people with small loan amount and longer duration tend to have `Good` credit records. The variable importance helps to visualise the drivers of the outcome and provides insights for the bank that these top 10 attributes should be carefully checked to avoid loaning money to people with high probability of `bad` credit.

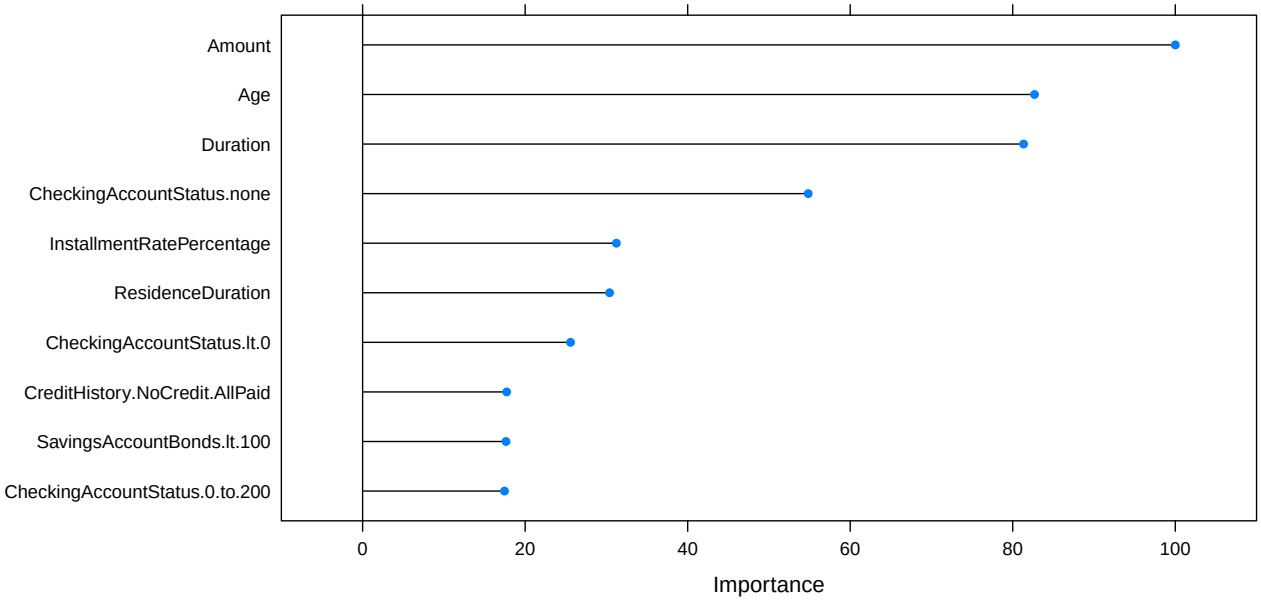

Figure 4: The top 10 variable importance for random forest model.

1.3 ROC curves

Figure 5 presents two ROC curves for classification on the test set using decision tree and random forest models that are tuned above. Random forest model achieves a larger area under ROC curve (AUC) of 0.780, compared to decision tree which produces AUC of 0.700. The result that random forest performs better in classifying test data corresponds to the outcome

that it achieves lower test error computed above. To conclude, there is a small improvement in the classification of test set when applying random forest algorithm, confirming the fact that random forest can be used to fix the overfitting problem of decision tree.

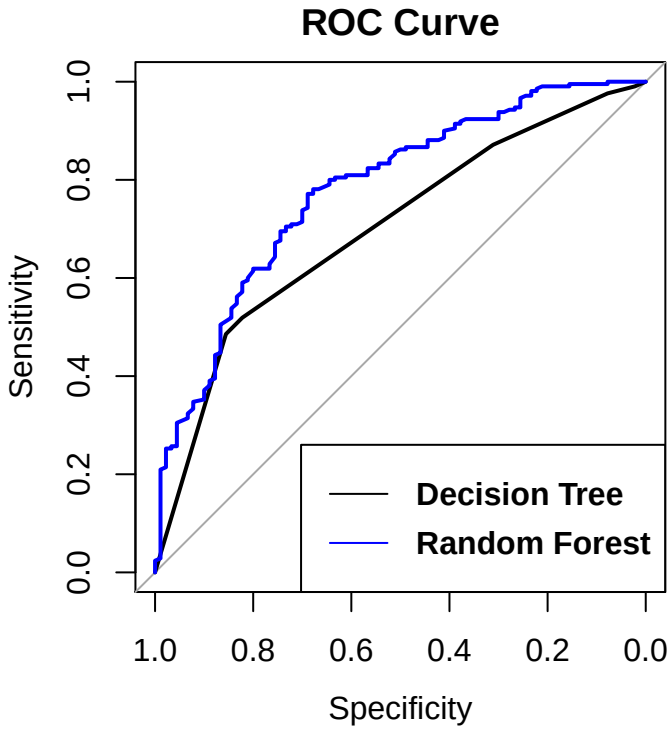

Figure 5: The ROC curves for decision tree and random forest.

2 Question 2

2.1 Simulate data set

To apply support vector machine (SVM) in this question, a data set is simulated and three "true" classes are constructed. In the simulated two-dimensional data, the first two classes are a mixture of Gaussian distributions with the same size 50, different mean vector `c(-1,0)`, `c(1,0)` and the same covariance matrix of diagonal elements equal to 0.5 and off-diagonal equal to 0.1. The R function `rmvnorm` from the package `mvtnorm` is used to generate such multivariate normal distributions. The third class consists of 50 points randomly lying on a circle of `radius=3` with the circle center located in `(0,0)` and angles randomly generated from `runif` function. Thus, a simulated data of `n=150` observations in total are presented in a two-dimensional space as shown in Figure 6. As indicated in the plot, the three classes are not linearly separable.

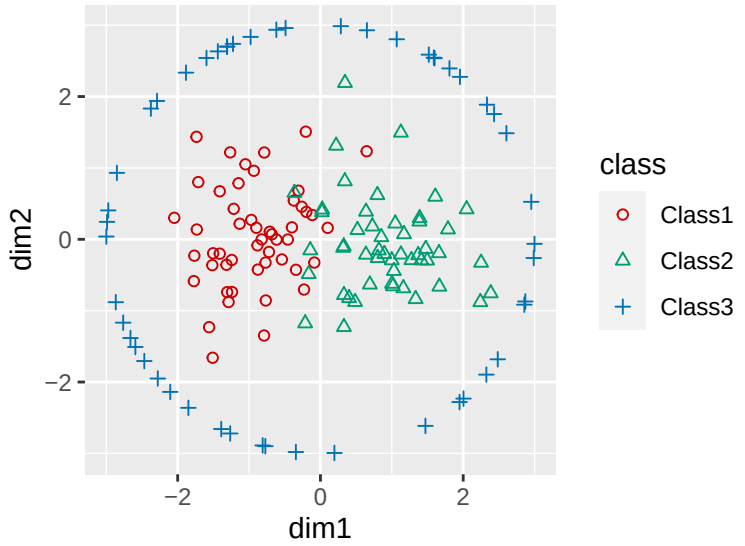

Figure 6: Simulated data represented in a two-dimensional space.

Subsequently, the support vector machine with a polynomial kernel and a radial basis function (RBF) kernel, as well as a support vector classifier which is equivalent to SVM with a linear kernel, are trained on 50% of the simulated data. The `caret` package is used to carry out the 5-fold cross-validation which is repeated 3 times. Additionally, the data is scaled to have zero mean and unit variance such that the two features have the same influence on the

distance metric. Based on the corresponding tuning parameters, a final model is fitted with the highest training accuracy in each repeated cross-validation process. The fitted model is then predicted on the remaining 50% simulated data and the associated test error rates are obtained. Finally, the three-class AUC values resulted from SVM with different kernels on the test set are computed to compare the models.

2.2 Train SVM with polynomial kernel

There are three parameters trained in SVM model with polynomial kernel. The `method='svmPoly'` is specified in the training process using `caret` package, and the three associated parameters are `degree`, `scale` and `C`, which indicate respectively the polynomial degree, the scaling parameter and the cost of constraints violation. A grid of parameters with `degree=c(2,3,4,5)`, `scale=c(0.01,0.1,1)` and `C=c(0.01,0.1,1,10)` are tuned and the training accuracy against different parameter values are plotted in Figure 7. Generally speaking, it seems that higher scaling value leads to higher accuracy; besides, larger cost, which means that samples inside the margins are more penalised, tends to improve the accuracy except when `scale=1`. Additionally, higher polynomial degrees seem to result in higher accuracy except when `scale=1`, in which case the influence of polynomial degree is relatively small. The optimal model is selected that achieves the highest training accuracy of 0.929 with parameters as `degree=2`, `scale=1` and `C=1`.

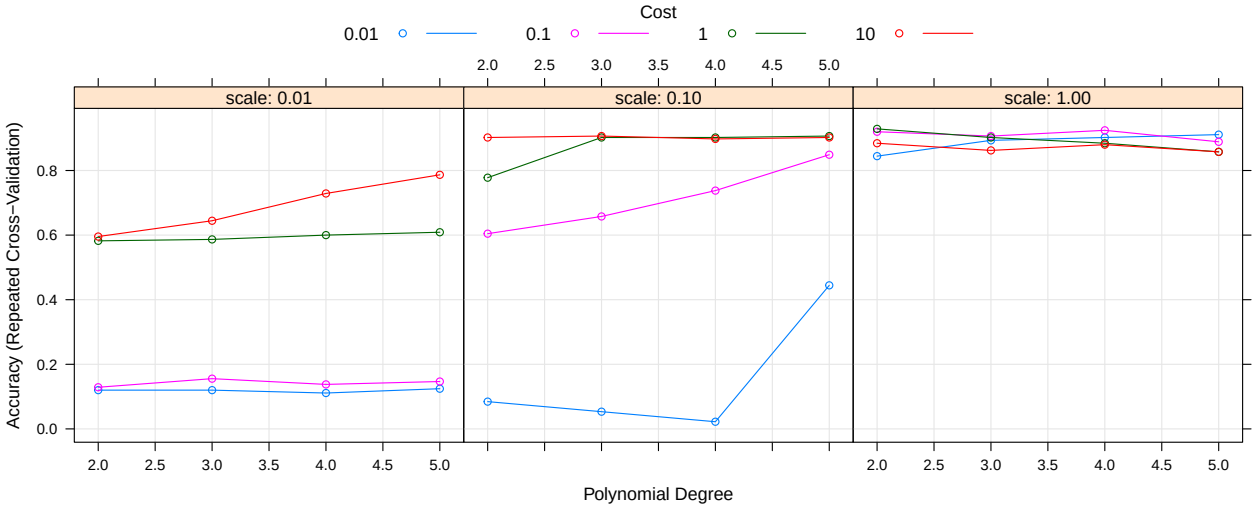

Figure 7: Plot of training accuracy against tuning parameters for SVM with polynomial kernel using 5-fold cross-validation.

2.3 Train SVM with RBF kernel

The `method='svmRadial'` is specified using `caret` package to apply SVM with RBF kernel, and there are two parameters to tune. A tuning grid of parameters are defined as `sigma=c(0.01,0.1,1,2,4,6,8,10)` and `C=c(0.01,0.1,1,10)`, which refer to searching for an optimal combination of the smoothing parameter sigma and the cost of constraints violation. Figure 8 demonstrates the relationship between training accuracy and the tuning parameters. It is noticeable that a small value of cost (i.e. `C=0.01` means that samples within the margins are barely penalised) produces significantly low accuracy. By holding constant the cost, higher value of sigma tends to generally improve accuracy when $sigma \leq 1$, while increasing sigma values has relatively small influence on accuracy when $sigma > 1$. As sigma increases, the decision boundary would become more flexible as the boundary is simply determined by points that are close to it and ignores points that are far away. The final model with `sigma=1` and `C=1` is chosen according to the highest accuracy of 0.902 it achieves.

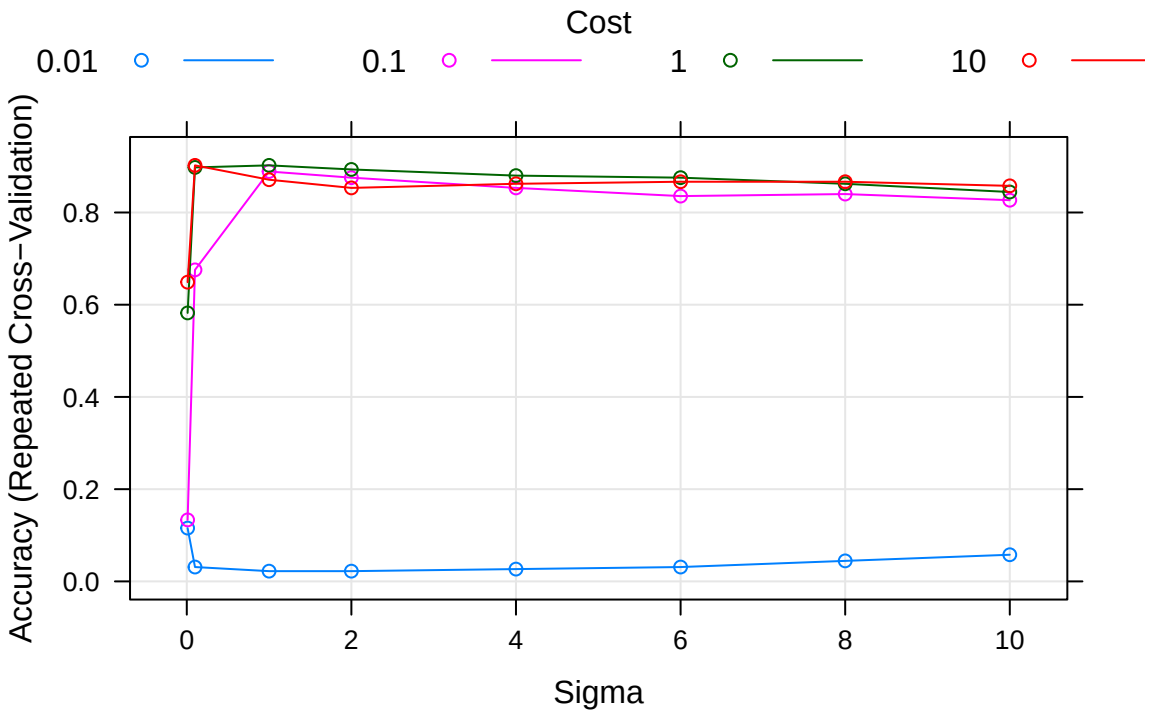

Figure 8: Plot of training accuracy against tuning parameters for SVM with RBF kernel using 5-fold cross-validation.

2.4 Train SVC

To train a support vector classifier on the training data, the `method='svmLinear'` is used in `caret` package as SVM with linear kernel is equivalent to SVC. One parameter cost of constraints violation is tuned and a grid of tuning values `C=c(0.01,0.1,1,10,20,30,40,50)` is specified. The relationship between accuracy on the training set and different values of cost is shown in Figure 9. Generally speaking, as cost increases, the accuracy becomes lower. That being said, the more penalised the samples inside the margins, the less accurate the SVC is to classify the training data. The optimal cost value `C=0.01` is selected for the final model as it achieves the highest accuracy of 0.631. Compared with nonlinear classification models fitted above, the linear kernel produces a significantly low accuracy in the training data set.

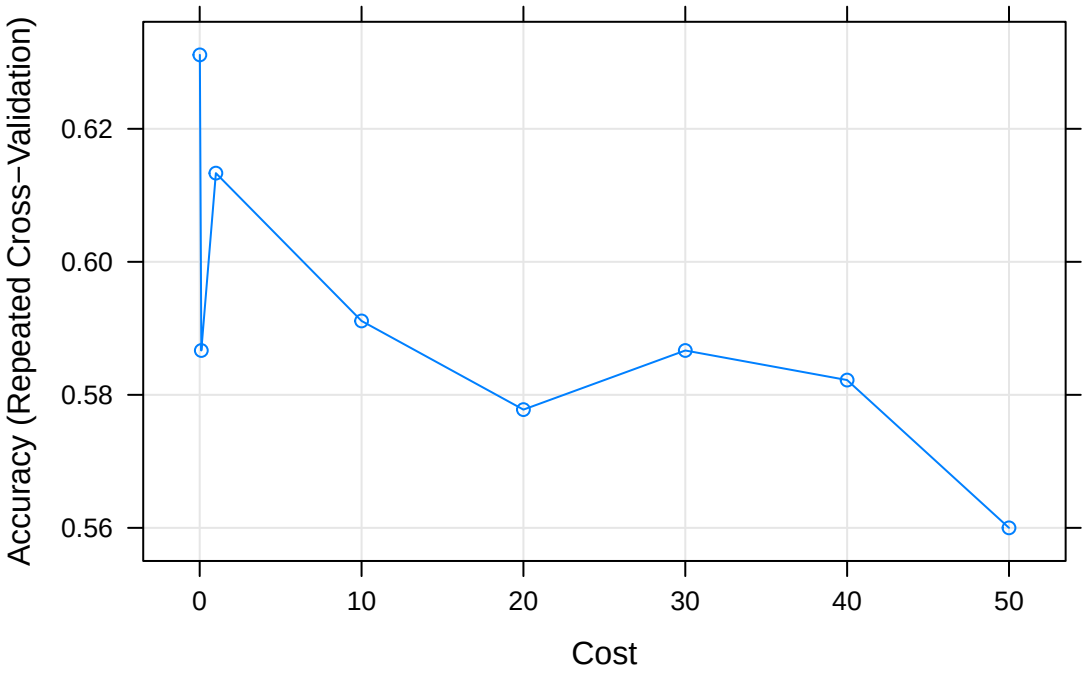

Figure 9: Plot of training accuracy against tuning parameters for SVC using 5-fold cross-validation.

2.5 Comment on the test results

The tuned SVM with polynomial, RBF and linear kernel are fitted on the test set, which is distinct from the training data. The test error rates for the three models are presented in

Table 1. SVM with 2-degree polynomial kernel produces the lowest test error rate of 0.013, slightly lower than the error rate of 0.04 that is achieved by SVM with `sigma=1` RBF kernel. SVC has the highest test error rate equal to 0.373. The outcome that SVM with linear kernel performs the worst on test set corresponds to the fact that the simulated data are not linearly separable. The nonlinear decision boundary, therefore, tends to achieve more accurate classification of the test data.

Table 1: Test error rates for SVM with polynomial, RBF and linear kernel.

	polynomial	RBF	linear
test error rate	0.013	0.04	0.373

To further compare the classification performance of the three SVM, the area under the ROC curve (AUC) is computed. As there are three classes in the data, `multiclass.roc` function from `pROC` package is used to build the ROC curves and extract AUC values. The three-class AUC values on the test data are indicated in Table 2. SVM with either polynomial or RBF kernel produces high AUC value of 0.999 or 0.998, indicating that it generates nonlinear decision boundaries that classify well the test data points into three classes. In contrast, SVM with linear boundary does not perform well in this classification task, with AUC of only 0.734. To conclude, it is recommended to apply SVM with polynomial or RBF kernel, thus generating nonlinear classification boundary in this nonlinearly separated data set.

Table 2: The AUC results on test set for SVM with polynomial, RBF and linear kernel.

	polynomial	RBF	linear
3-class AUC (test)	0.999	0.998	0.734