

Figure 1: Correlation plot for all variables in the provided data (chatter)

The above *Figure 1* gives a pairs plot of all variables contained in *chatter.csv* file, the output provides the distribution, density plot of each variable as well as their scattered plot and correlation against each other. The file comprises 8 variables and for the purpose of the classification analysis the column “Resolved” is used as the target variable. The target variable consists of 2 groups (“Yes” and “No”) which indicates if the customer considered an issue resolved. The plots in *Figure 2* shows the boxplot distribution of other variables categorized by the groups in the target variable. This presents the visual measures of central tendencies.

The final *figure 3* shows the overall distribution of all the variables on a density plot, some distributions are skewed at different sides while some can be assumed to be bimodal; therefore, for the sake of this output, normality cannot be assumed for the variables of this data.

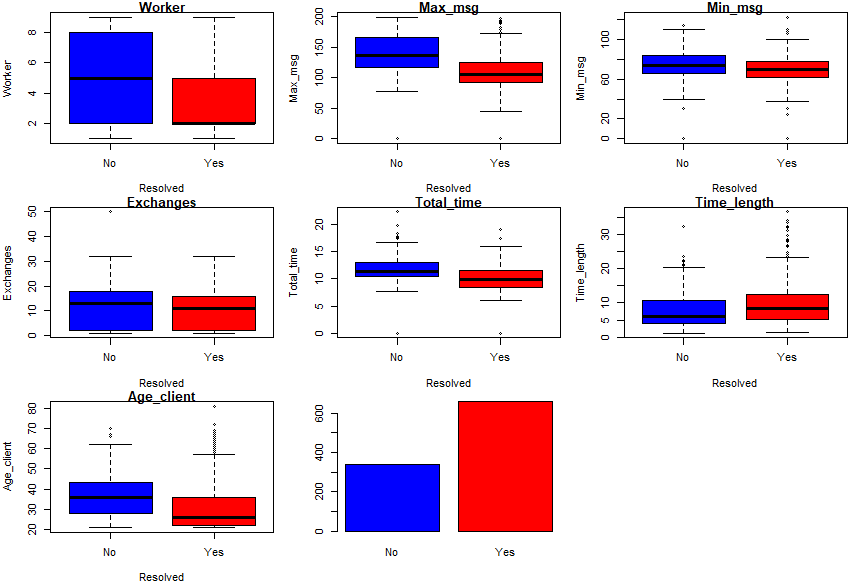


Figure 2: Variable Visual Representation 1

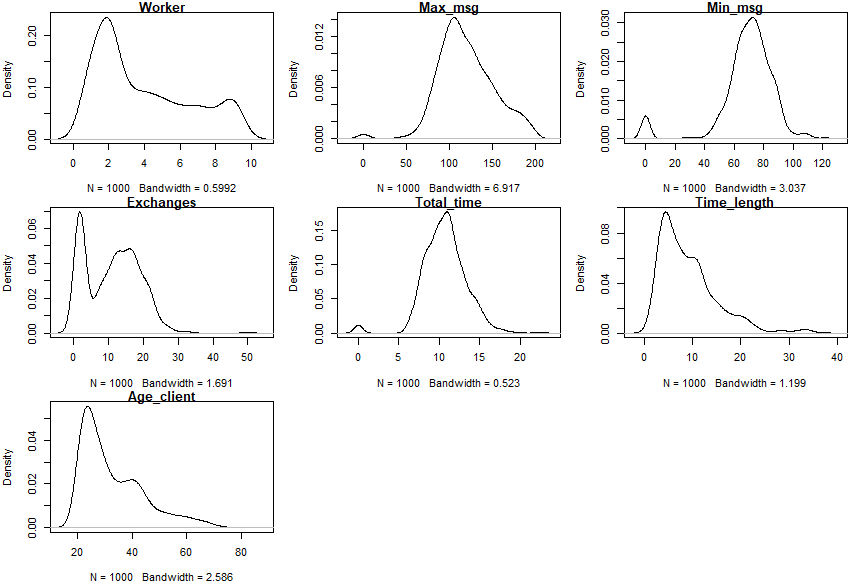


Figure 3:Variable Visual Representation 2

**Normal Tree model (unpruned)**

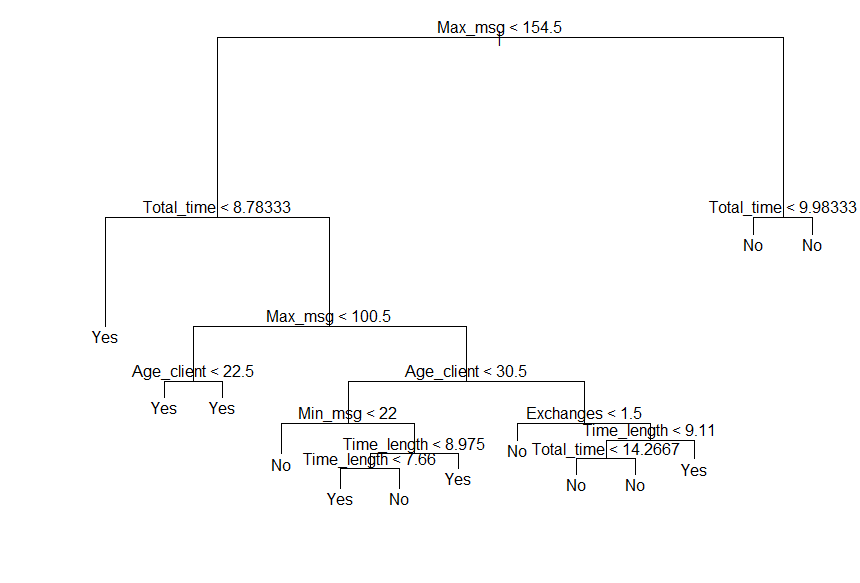


Figure 4:Unpruned Decision Tree

Classification tree:

tree(formula = Resolved ~ ., data = train)

Variables actually used in tree construction:

[1] "Max\_msg" "Total\_time" "Age\_client" "Min\_msg" "Time\_length" "Exchanges"

Number of terminal nodes: 13

Residual mean deviance: 0.7548 = 594 / 787

Misclassification error rate: 0.1688 = 135 / 800

The diagram in *Figure 4* shows the final decision tree derived from an unpruned tree model, the model makes use of 6 variables out of 7 provided variables, these variables provide a better classification model in determining if the issues are considered resolved or not. The “worker” variable is not considered in the design of this tree and can be assumed to be irrelevant in determining the resolution of the issues. As stated in the data description given, the worker column data consists of values gotten from the self-assessment given by other employees; therefore, it less likely for the variable to directly affect the resolution of customer issues.

The unpruned tree has a misclassification error of 16.9% on the training data and also, there are 13 terminal nodes on this tree model created. And it can be observed that some of these nodes give a single resolution output irrespective of the conditions attached to them. Therefore, we can try pruning to take care of such redundant responses.

**After Prediction**

While testing with the testing dataset, the following metric output were gotten: -

**Misclassification Error:** 17.5%

**Accuracy:** 82.5%

**Pruned Model**

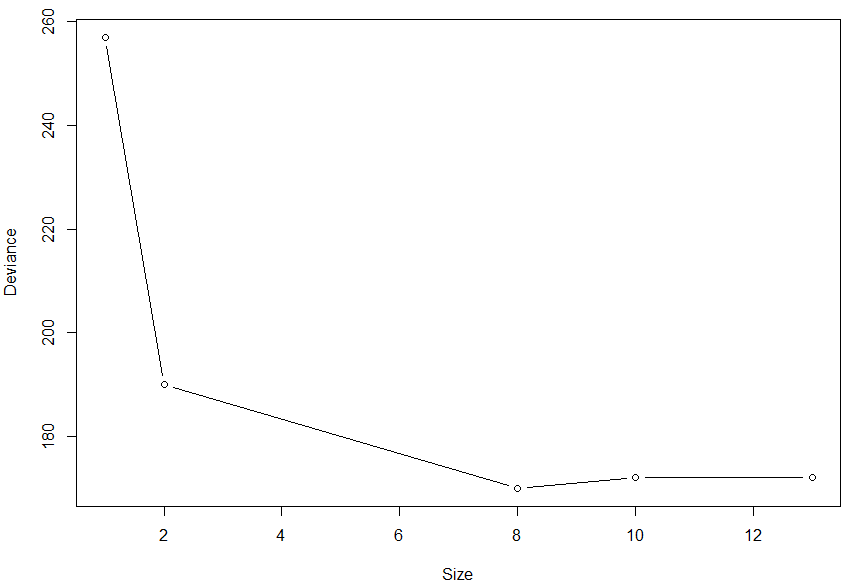


Figure 5: Prune Parameter size vs Deviance

Classification tree:

snip.tree(tree = tree\_mod, nodes = c(3L, 10L, 94L, 45L))

Variables actually used in tree construction:

[1] "Max\_msg" "Total\_time" "Age\_client" "Min\_msg" "Exchanges" "Time\_length"

Number of terminal nodes: 8

Residual mean deviance: 0.8303 = 657.6 / 792

Misclassification error rate: 0.1775 = 142 / 800

In the attempt to prune the tree model, the best tuning value for the hyperparameter is a tree size of 8 terminal nodes as indicated in *Figure 5* as the value that gives the least deviance. For further analysis, we proceed to check if the pruned tree model has an effect on the accuracy of its prediction of the test dataset and the result is as follows: -

After Prediction using Prune model:

**Misclassification Error:** 17.5%

**Accuracy:** 82.5%

As it can be observed that the pruned tree has no effect on the accuracy of prediction of this data as its accuracy before and after pruning remains unchanged. Therefore, pruning this tree model only improved the readability of the tree by reducing the number of terminal nodes, but did not improve the prediction accuracy on the testing data. *Figure 6* shows the visual representation of the pruned tree model.

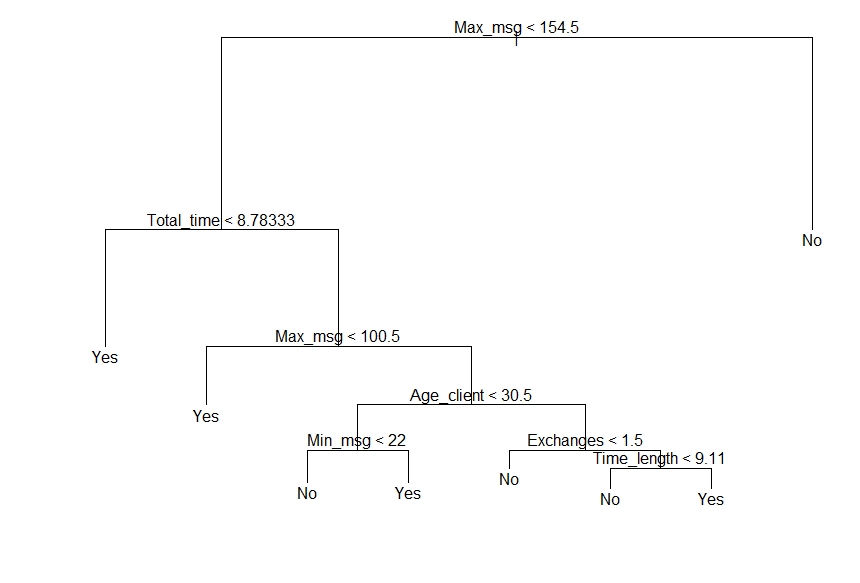


Figure 6: Pruned Decision Tree

**Random Forest**

Random Forest

800 samples

7 predictor

2 classes: 'No', 'Yes'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 720, 720, 720, 720, 719, 721, ...

Resampling results across tuning parameters:

mtry Accuracy Kappa

1 0.8451303 0.6200512

2 0.8476619 0.6301817

3 0.8413645 0.6191973

6 0.8301449 0.5942385

7 0.8288946 0.5903641

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was mtry = 2

The randomForest method was applied to the same data as used for the tree model, and a cross validation method is used in selecting the best value for the tuning parameter (mtry) to give the best accuracy of the model. The best value selected for mtry is 2 as this hives the highest accuracy of over 83%. *Figure 7* below shows the plot for number of randomly selected predictors against their accuracy when applied to form a model with the training data.

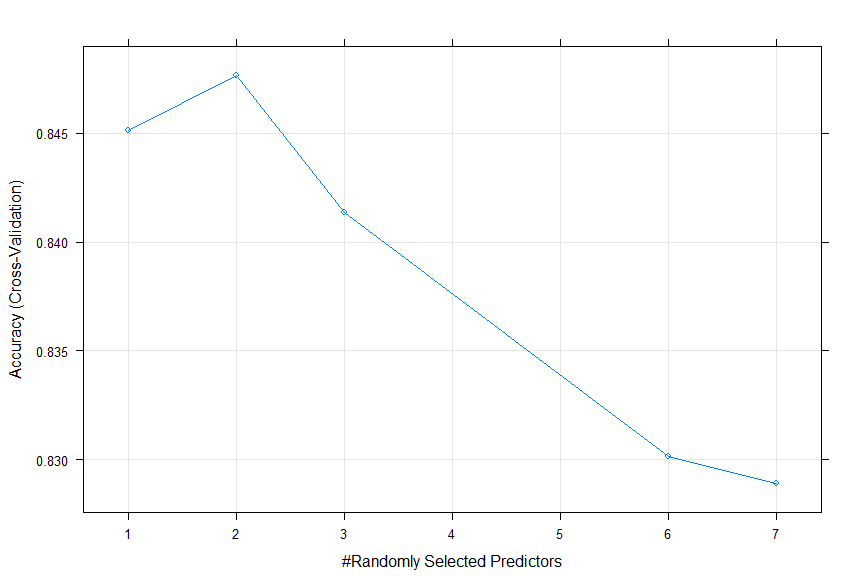


Figure 7:Accuracy of Randomly selected Predictors

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 2

OOB estimate of error rate: 15.12%

Confusion matrix:

No Yes class.error

No 178 79 0.30739300

Yes 42 501 0.07734807

The output of the final model created is shown above, and it shows that 500 trees were used with 2 variables tried at each split. The OOB error is 15.12%, also, it is seen that the model performs better at predicting if an issue is resolved with a classification error as low as 7.7%. But on the other hand, it has a classification error of 30.7% when predicting issues that remain unresolved.

This model Is used to predict the target the target result of the test dataset and after predicting, the following metric was gotten: -

**Misclassification Error**: 16%

**Accuracy**: 84%

It can be seen that the randomForest model performed better in making prediction when compared to the normal and pruned tree method.

**Support Vector Machine (SVM)**

Untuned model

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 1

Number of Support Vectors: 388

(192 196)

Number of Classes: 2

Levels:

No Yes

A support vector method is used to classify the same *chatter.csv* and a linear kernel with a cost of 1 is used for this model creation. The output shows that 388 support vectors were used with 192 and 196 for the resolution status of “No” and “Yes” respectively.

The accuracy of this model is 74.5%, and this is lower than both randomForest and the tree models

Finding the best tuning parameter

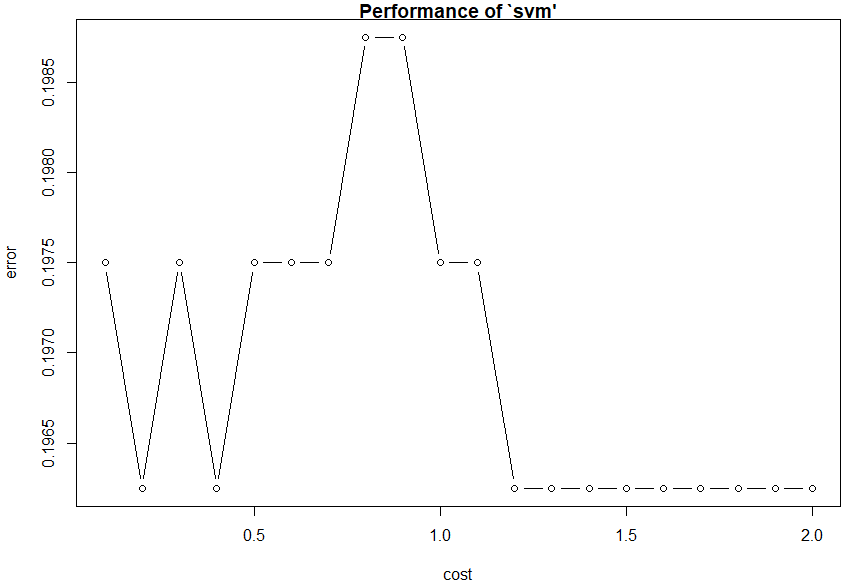


Figure 8: Cost parameter tuning

*Figure 8* above shows a plot different cost values against their respective errors, the cost sequence is created with a constant increase of 0.1. It will be seen that the least error can be gotten from a number of different cost value, so therefore, we assume the cost vale to be 0.2 which is the first value giving the least error. The output below shows the result of the tuning parameter which indicates that a cost value of 0.2 is the pest parameter to use when creating the model and a 10-fold cross validation sampling method is used to determine this value. The selected cost value gives a minimal error rate of 19.6%

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:

cost

0.2

- best performance: 0.19625

Best Model

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 0.2

Number of Support Vectors: 391

(194 197)

Number of Classes: 2

Levels:

No Yes

The new model created from the tuned cost value of 0.2 uses 391 support vectors with 194 and 197 used for “No” and “Yes” resolution output respectively. The accuracy of this model is 75% which is slightly higher than the untuned model.

Comparison

Accuracy

unprunnedTree.Accuracy 0.825

prunedTree.Accuracy 0.825

randomForest.Accuracy 0.840

svm\_untuned.Accuracy 0.745

SVM\_tuned.Accuracy 0.750

From the above tabled output, it is seen that the best model to use for the classification of this data is the randomForest model as it gives the highest accuracy compared to other models. The randomForest performed better as it is robust to outliers, and it will be observed that in the boxplot grouped by their various resolution result that there is a presence of outliers in the distribution of the various variables. And also, the randomForest has a lower risk of overfitting as it is learns using different samples and at each node, a random set of features are selected for splitting.

The SVM model performed the least as it provided a lesser accuracy as compared to other models, the SVM model maximises margin and therefore depends on the distance between different point thus, it performs better on a sparcely distributed data, but it will be observed that in *Figure 1* that the data is densely populate with minimal spacing, thus, this gives a reasonable explanation to why if could not preform better.