MCDA5580 Assignment 2

Team Member

* Hemalatha Srinivasan A00452621
* Ajay Jain A00455849
* Kin Wa, Chan A00467755

**Table of Content**

[Executive Summary 2](#_Toc1931939426)

[Objectives 2](#_Toc1842396402)

[Data Analysis 3](#_Toc1417429087)

[Design/ Methodology/ Approach 5](#_Toc813638736)

[Overview 5](#_Toc619127367)

[Decision tree Analysis 5](#_Toc1661445177)

[Random Forest Analysis 11](#_Toc1296239987)

[Caret (Classification and Regression Training) 20](#_Toc1448260209)

[Conclusion 25](#_Toc156222827)

[Appendix 26](#_Toc971297243)

[R Script 26](#_Toc1702867774)

[Decision Tree 26](#_Toc1826171047)

[Random Forest 28](#_Toc594581366)

[Caret Package 33](#_Toc746236435)

[Reference/ Citation 41](#_Toc38441738)

# Executive Summary

We are given dataset of customer feedback regarding the desirability of buying the car with features like safety, price etc. We explored supervised classification algorithms to develop models to predict desirability of buying cars given their features. We evaluated the model performance using different hyperparameters and performance metrics. The final selected model is the model created from Caret package with 98% accuracy.

Our ultimate goal is to deploy optimized model as a recommendation system on the website of car dealers in the Halifax area. Also, collect online feedback from customers of dealers and use it to evaluate and update the model. However, the current result tends to have overfitting problem which may affect the ability of describing actual customer behaviors. So, to fine tune model with a larger data set is suggested before deploying the final solution.

# Objectives

The main **o**bjective of this process is to create a data model to classify the datasets into different classes using the results obtained and group together as different sets for easy understanding. This will create a model which will have high accuracy and less error. The obtained result will give clarity in determining the best and fewer performing categories which may help to understand the user preference. This will also provide an overview of where the growth of the organization is affected. Overall, the aim of the model is to give a quality result for identifying where the improvement should be made to increase the future result.

There are clear classified results (“shouldBuy”) in this problem, and it had type of classes > 2. Therefore, supervised classification models which support multiple class should be selected. In report, Decision Tree & Random Forecast models will be used, which have advantages of having good predictive power (especially for Random Forest), relatively easy to build and tune and the result more able to be visualized.

# Data Analysis

Dataset has 1798 rows with 6 independent and 1 dependent variable. All the data is categorical. There are no missing or invalid values.

Below is description of independent variables. All the data is evenly distributed across different categories.

price - 4 categories (low, med, high, vhigh)

maintenance - 4 categories (low, med, high, vhigh)

doors - 4 categories (2, 3, 4, 5more)

seats - 3 categories (2, 4, more)

storage - 3 categories (small, med, big)

safety - 3 categories (low, med, high)

shouldBuy is dependent and categorical variable. It can take 4 values which describe desirability of buying the car in ascending order.

unacc - Unacceptable (don't buy)

acc - Acceptable to buy

good - Good to buy

vgood - Very good to buy

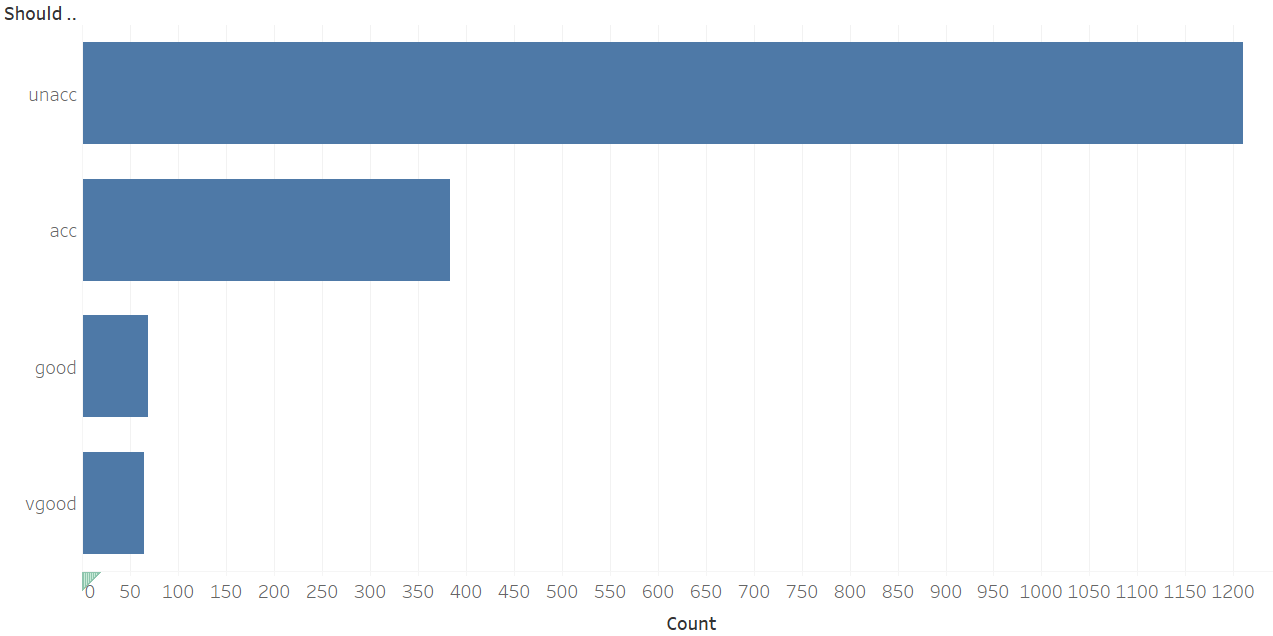
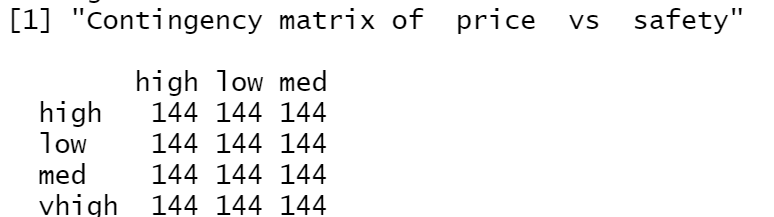


Figure 1 Count Plot - Output Classes

**Correlation** - Contingency matrix for each combination of independent variables is having same values which rules out any possibility of any correlation. No further analysis or plotting is required in this regard.

Screen shot of price vs safety is shown below:



# Design/ Methodology/ Approach

## Overview

Decision Tree & Random Forecast models is used in this report, differences between Decision Tree & Random Forecast models are the result of Random Forecast is generated from the result a set of decision trees. Compare with single decision tree, it reduces overfitting and improved accuracy. The two models will be implemented by the rpart (for Decision Tree) and randomForest (for Random Forecast) package in R. In addition, Caret package will also be used to cross checking which is a comprehensive framework for model development in R. Finally, the Area Under Curve (AUC) value will be used to evaluate the performance of the tested models.

## Decision tree Analysis

Decision Tree Algorithm is one of the classification algorithms for understanding the dataset and predicting the future results by recursively partitioning the input dataset into smaller sets. Since the decision trees visualization is effective which will enable the user to view and understand how the predictions are obtained by the algorithm. They provide results fast that will make them easy to train. They are good in handling both numerical and categorical datasets.

**Methodology**

1. Dataset has been loaded and split into two parts namely, Training set (80%) and Testing set (80%).

head(carData)

price maintenance doors seats storage safety shouldBuy1 vhigh vhigh 2 2 small low unacc2 vhigh vhigh 2 2 small med unacc3 vhigh vhigh 2 2 small high unacc4 vhigh vhigh 2 2 med low unacc5 vhigh vhigh 2 2 med med unacc6 vhigh vhigh 2 2 med high unacc

1. Decision tree is formed using rpart function with minsplit value as 6 for control parameter. Training set is used for creating the decision tree.

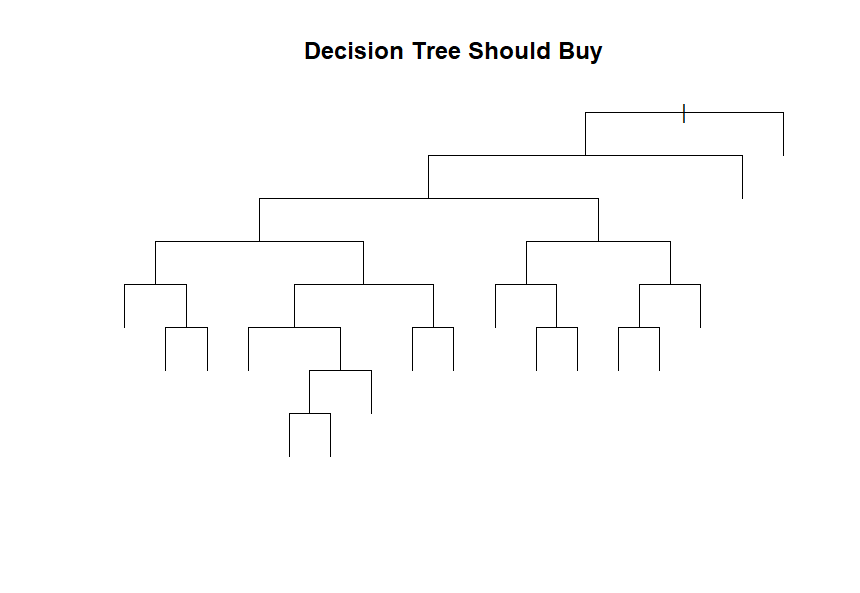


Figure 2 Decision Tree

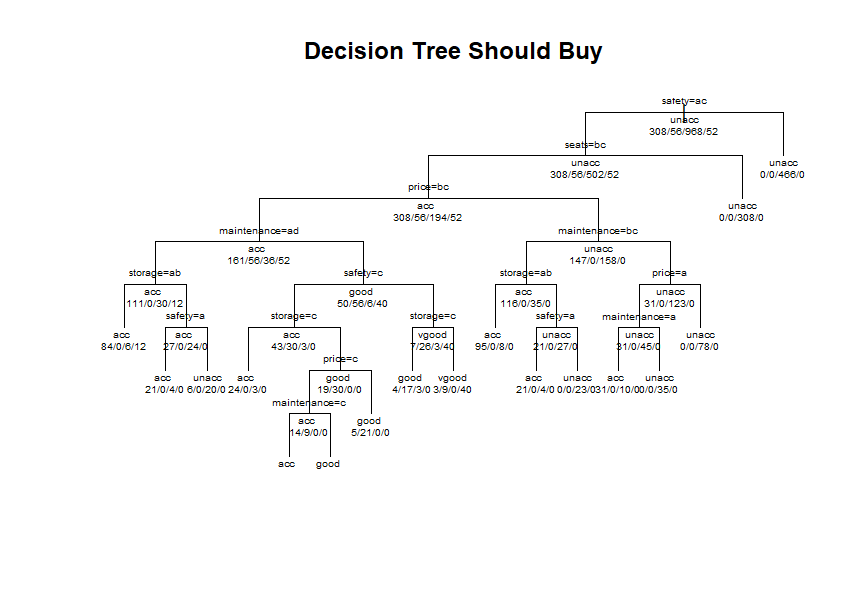
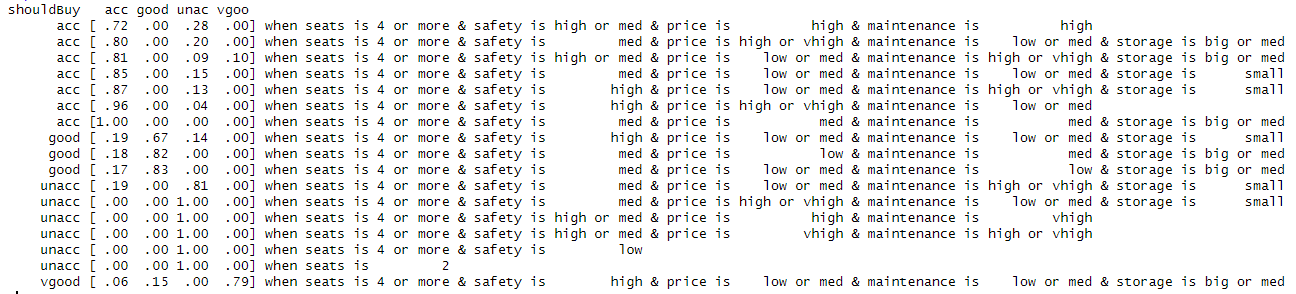


Figure 3 Decision Tree with Labels

Rules of Decision Tree



1. Minsplit is decided based on the reached AUC result with the best closer value.
2. Car prediction is done using decision tree and test data with predict () method
3. Predict data is created for both class and problem.



1. Confusion matrix and statistics (Accuracy, Sensitivity (Recall), Specificity etc.) are created based on test data and predicted class results.

一張含有 文字 的圖片

自動產生的描述

1. Multiclass ROC is created using test data and Problem Prediction result
2. Using Multiclass ROC result, Area under Curve is determined

sum(diag(treeCarCM)/sum(treeCarCM))[1] 0.9505814

auc(roc.multi)Multi-class area under the curve: 0.8878

1. ROC graph is plotted for each individual classes namely, ACC, UNACC, GOOD, VGOOD

.

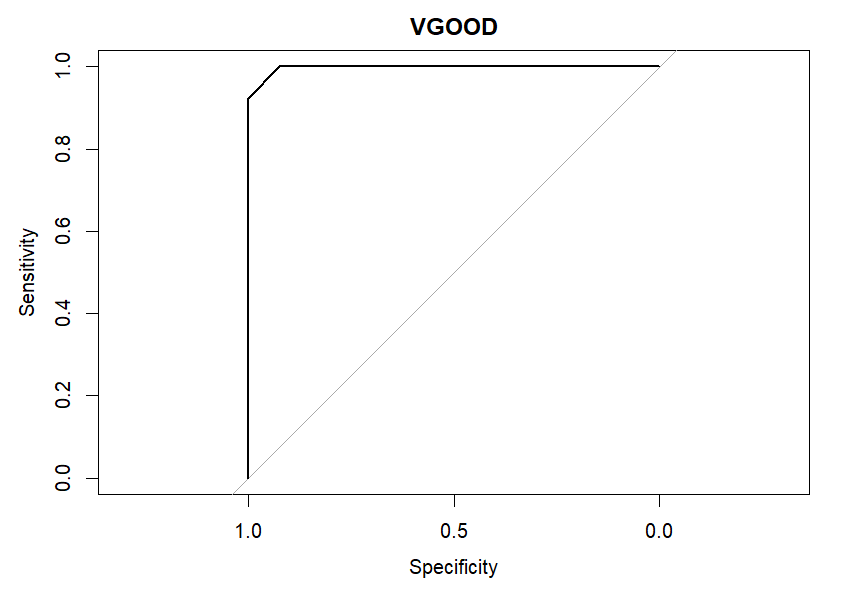


Figure 4 ROC Plot

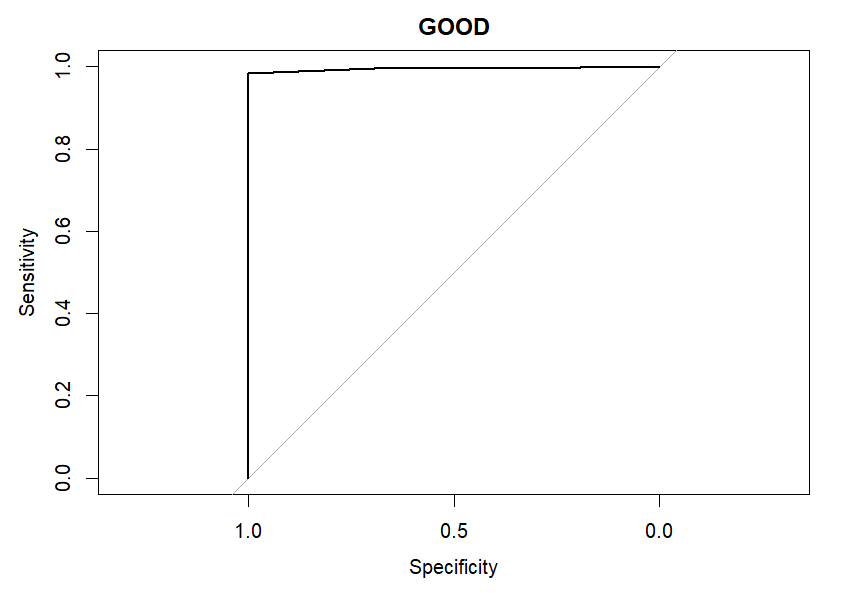


Figure 5 ROC Plot

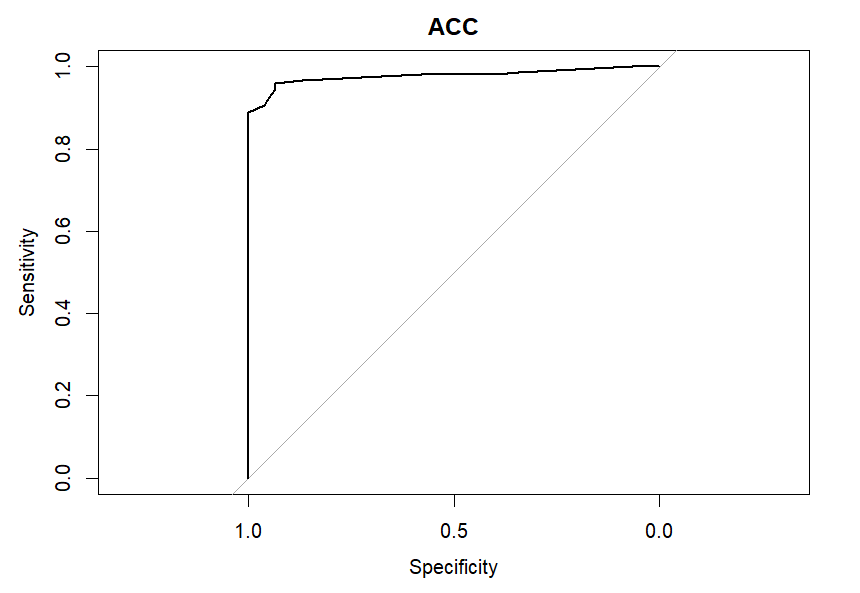


Figure 6 ROC Plot

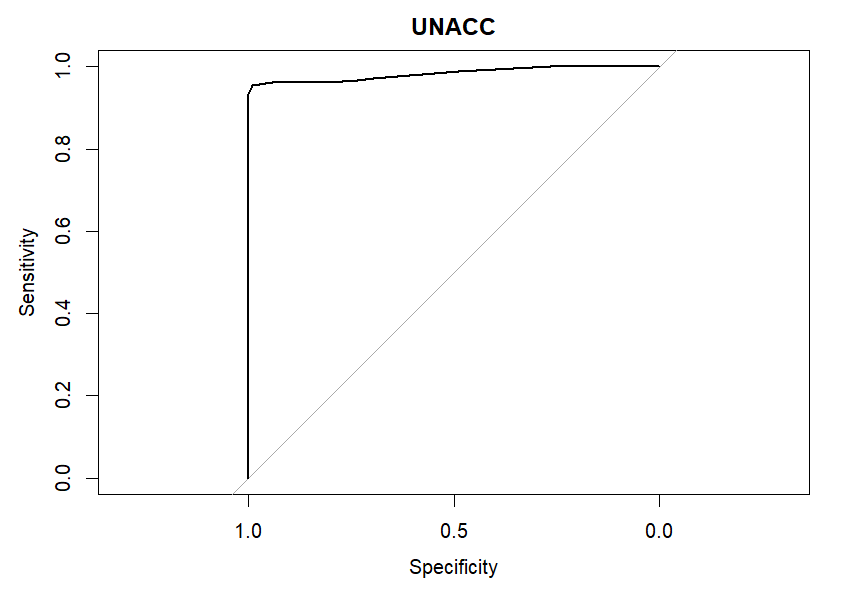


Figure 7 ROC Plot

Summary

Final model gave accuracy of 94.19% and AUC of 0.8878 with parameter minsplit=6

## Random Forest Analysis

Random Forest algorithm is derived from decision tree algorithm. It contains a collection of decision trees trained on random subsets of features and observations and produce the final result by average the predictions of all the trees.

Compare with original decision tree algorithm, it has advantage of lesser chance to overfit data and better to handle smaller data set. However, it lose the interpretability of result but generally yield a better performance compared with a single decision tree. It is the reason to practice Random Forest algorithm here to discover potential performance gain of prediction model.

**Methodology**

* Data handling

Data will be split into training (80 %) and testing (20%) set.

* Hyperparameter tuning

“mtry” parameter

It controls the number of randomly selected features that are used to determine the best split at each node. i.e., the level of fitting of the algorithm.

tuneRF() is used in tuning the parameter:

> model <- tuneRF(train[,1:6], train[,7] , mtryStart = 2)

mtry = 2 OOB error = 5.56%

Searching left ...

mtry = 1 OOB error = 26.45%

-3.753247 0.05

Searching right ...

mtry = 4 OOB error = 2.75%

0.5064935 0.05

mtry = 6 OOB error = 2.67%

0.02631579 0.05

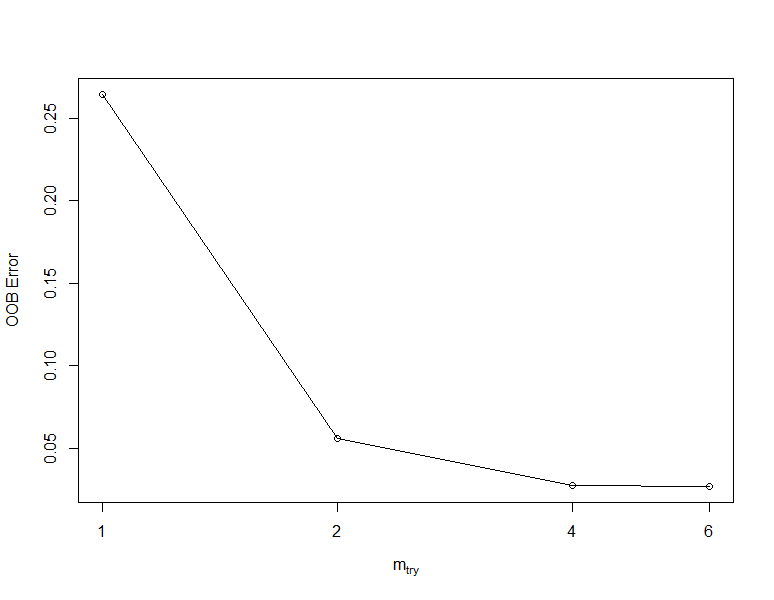


Figure 8 Error Plot vs mtry

We see mtry = 6 have the lowest OOB Error, so it is the optimal value.

* + “nodesize” parameter

This parameter specifies the minimum of terminal nodes in the tree. Smaller “nodesize” value results higher predictive accuracy but may cause overfitting.

Random Forest models with different setting are built and tried to predict data in the test data set. The table above contains the confusion matrices and accuracy of these models. We can see node size =1 have the best accuracy, so it will become the optimal parameter.

|  |  |  |
| --- | --- | --- |
| Node size = 1 | Node size = 5 | Node size = 10 |
| 一張含有 文字 的圖片  自動產生的描述 | 一張含有 文字 的圖片  自動產生的描述 | 一張含有 文字 的圖片  自動產生的描述 |

**Optimal model**

The hyperparameter tuning yield mtry = 6 and node size = 1 the optimal value for the random forest model. Here are the details of model:

Confusion Matrix and Statistics (Accuracy, Sensitivity (Recall), Specificity etc.)

一張含有 文字 的圖片

自動產生的描述

Variable Importance

一張含有 文字 的圖片

自動產生的描述

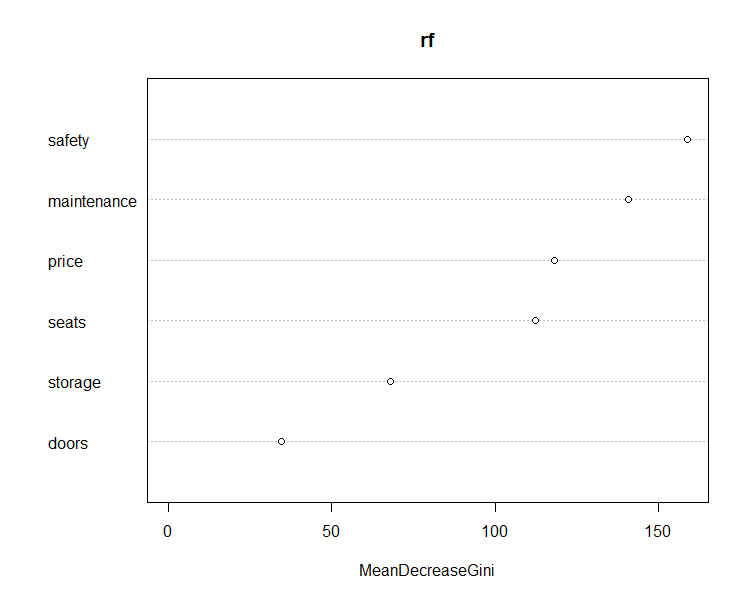


Figure 9 Variable Importance

Plot of a Single Tree in the Random Forecast model

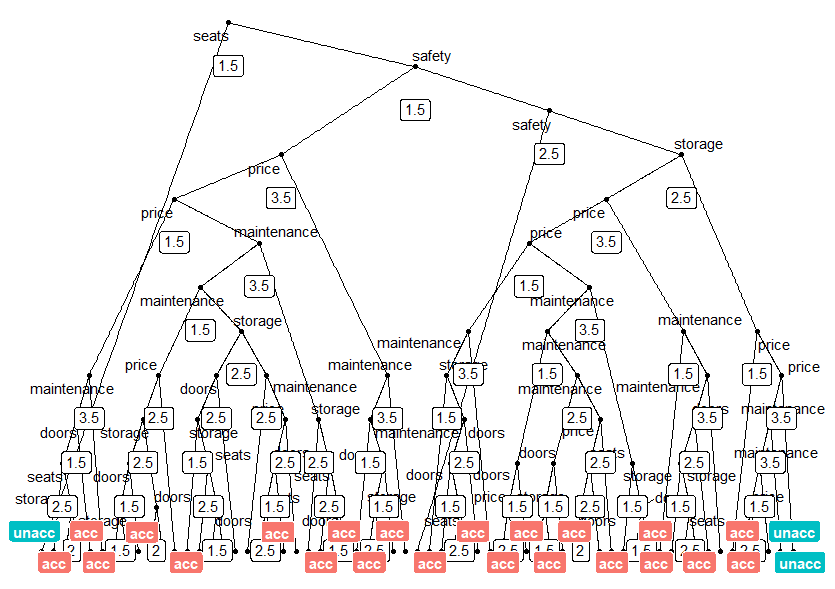


Figure 10 Sample Tree Diagram

ROC (Receiver Operating Characteristics) Curve

ROC for Unacc Class

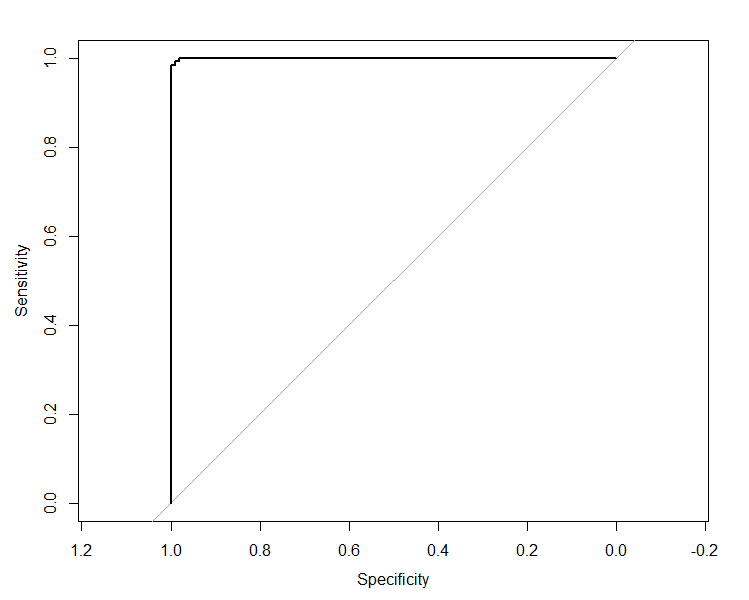


Figure 11 ROC Plot

ROC for Acc Class

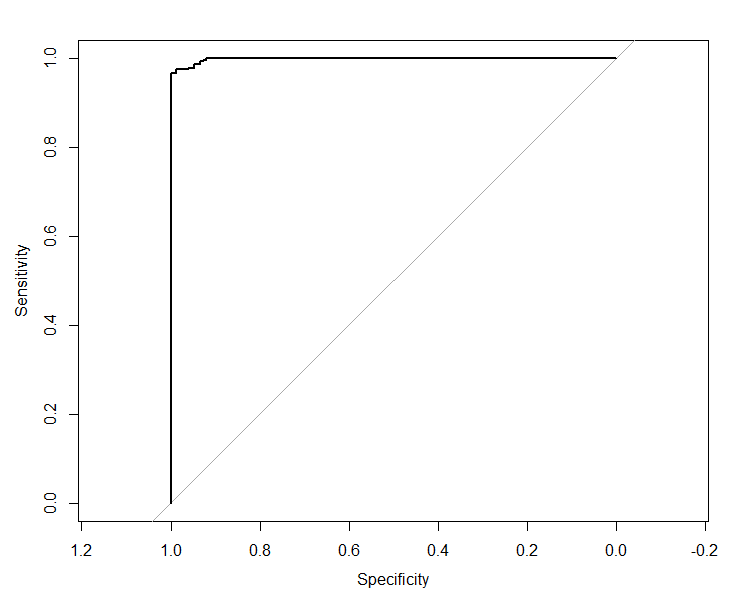


Figure 12 ROC Plot

ROC for Good Class

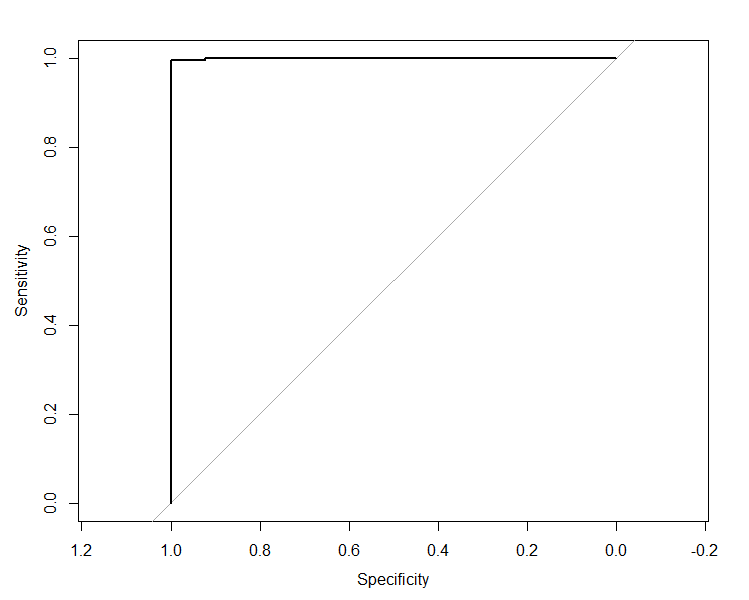


Figure 13 ROC Plot

ROC for Vgood Class

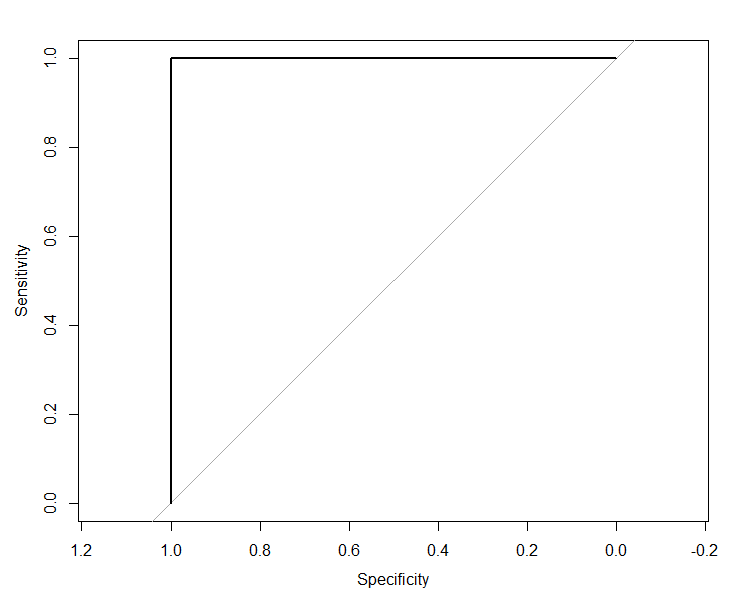


Figure 14 ROC Plot

AUC (Area under Curve)



Summary

Final model gave accuracy of 97.67% and AUC of 0.9921 with parameters mtry=6, nodesizes =1

## Caret (Classification and Regression Training)

Caret is versatile package in R which has lot of tools for

* data splitting
* pre-processing
* feature selection
* model tuning using resampling
* variable importance estimation

This package provides a good interface to perform tasks like parameter tuning and variable importance. Hyperparameter tuning is done using grid search where list of parameters and its values are given in matrix, and it fits the model with all combinations of parameters and then evaluates the best amongst them.

Random forest algorithm in this package allows us to tune only mtry parameter. But it can be customized to tune other features as well. We used this customization approach to tune ntree, minsplit.

**Methodology**

* Data handling

Data will be split into training (80 %) and testing (20%) set.

* Hyperparameter tuning

Below is the plot of “Accuracy” for mtry, ntree and minsplit combination:

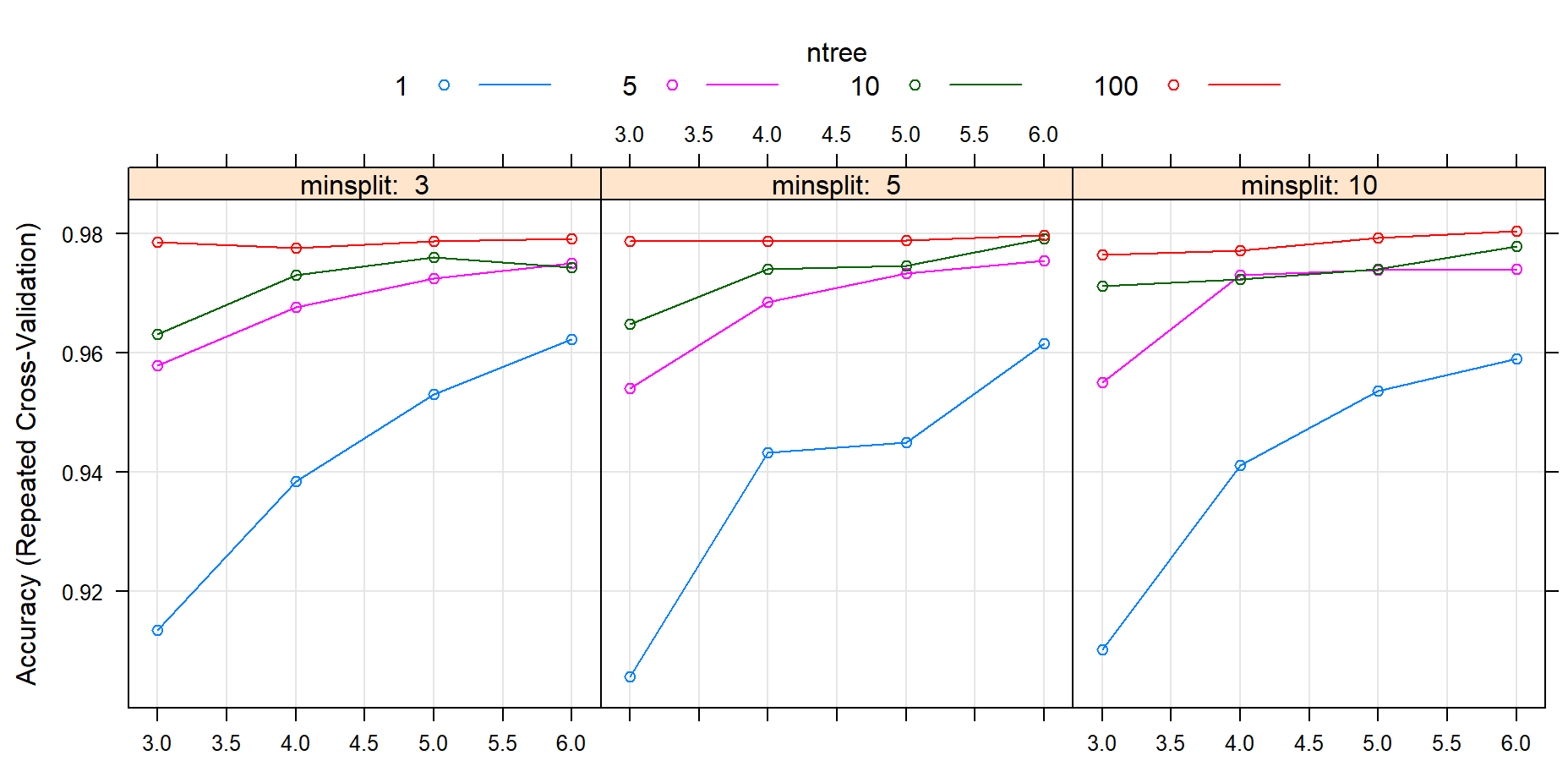


Figure 15 Tuning Plot

Optimal model chosen in the grid search had parameters mtry = 6, ntree = 100, maxdepth = 5

and minsplit = 10.

Variable importance plot for the model:

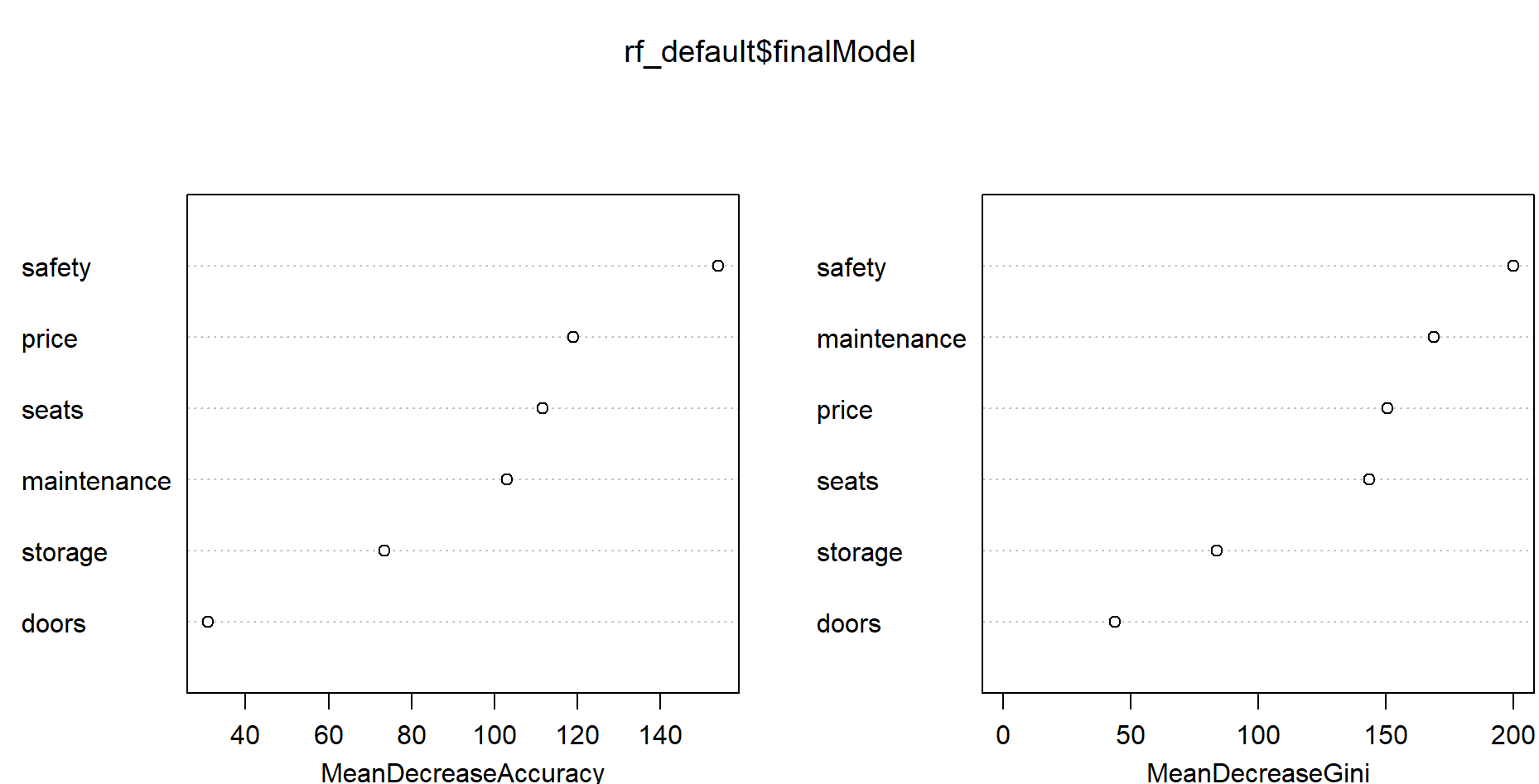


Figure 16 Variable Importance

ROC plot for the final model:

Unacc class

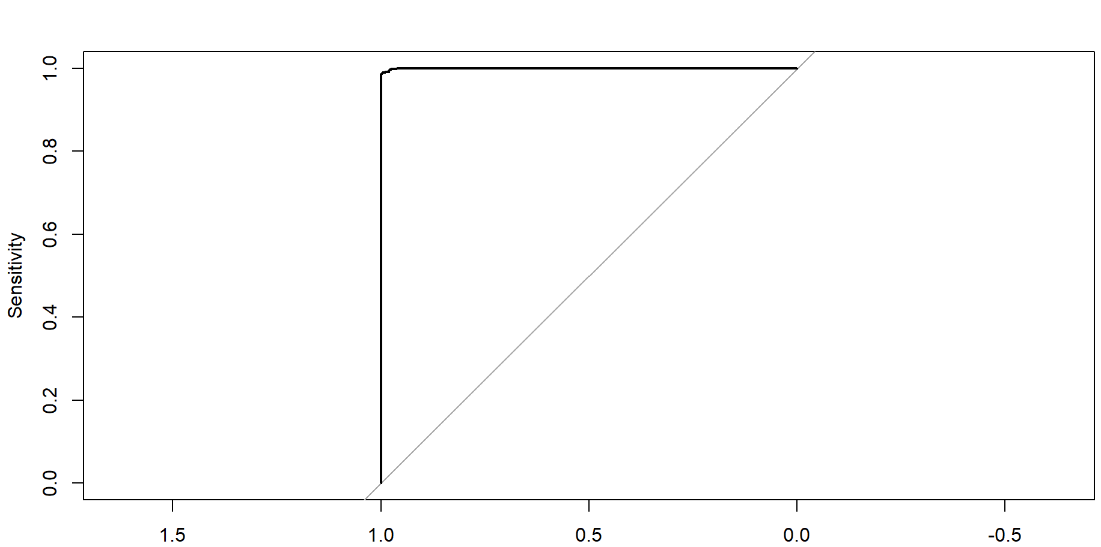


Figure 17 ROC Plot

Acc class

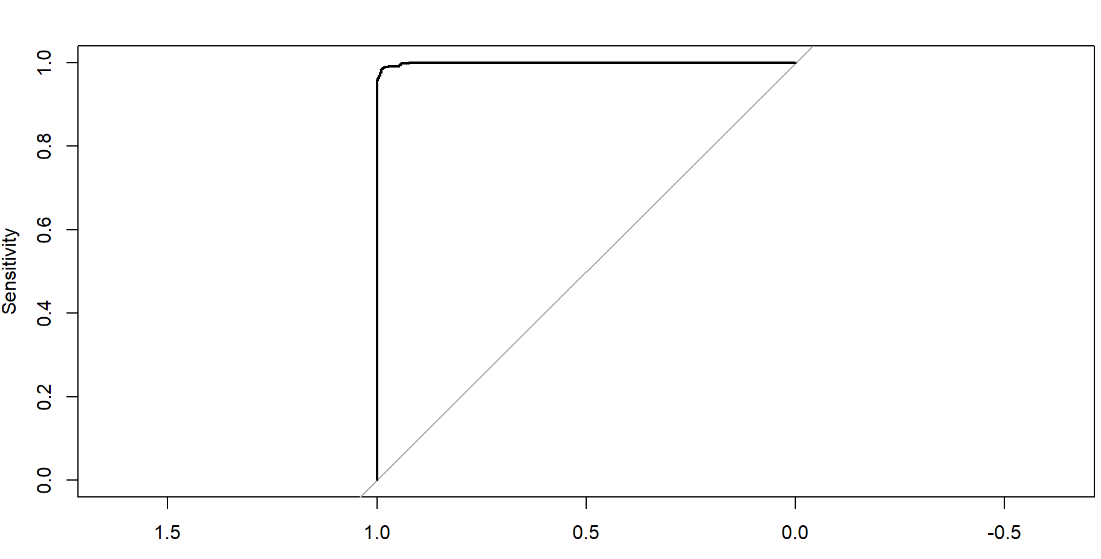


Figure 18 ROC Plot

Good class

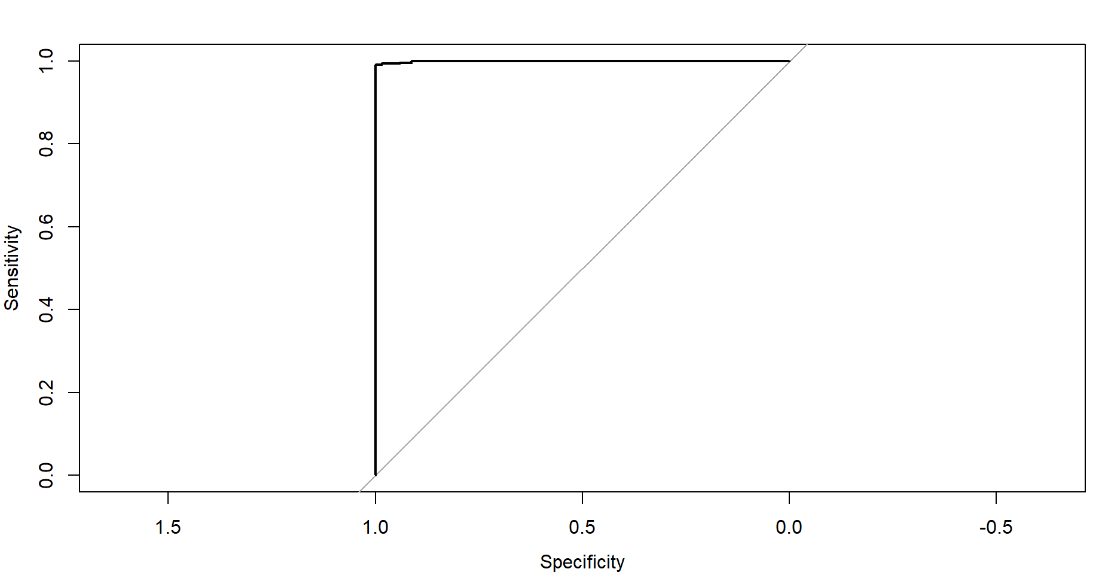


Figure 19 ROC Plot

Vgood class

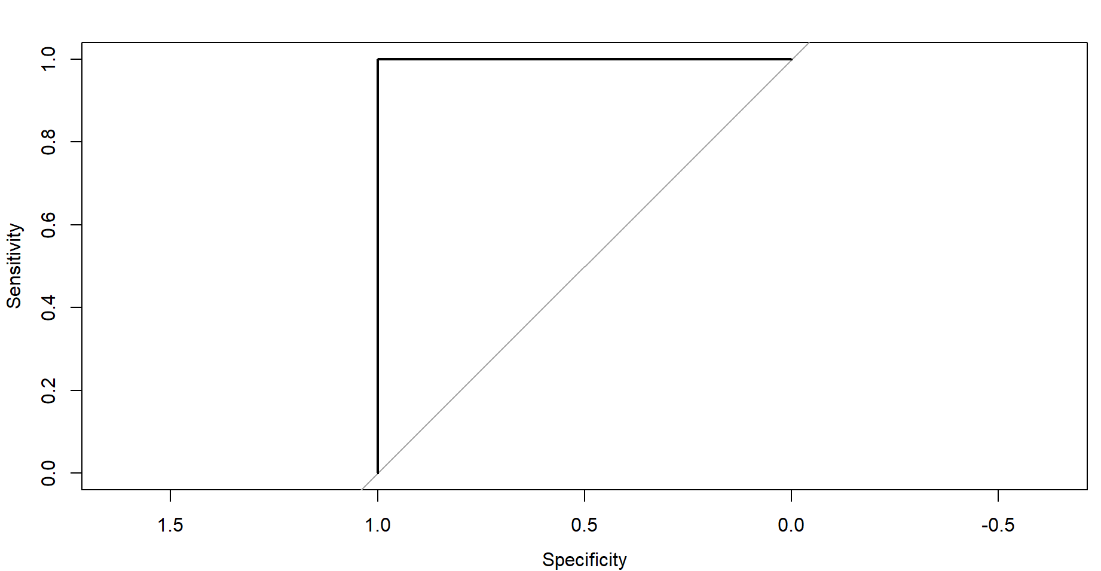
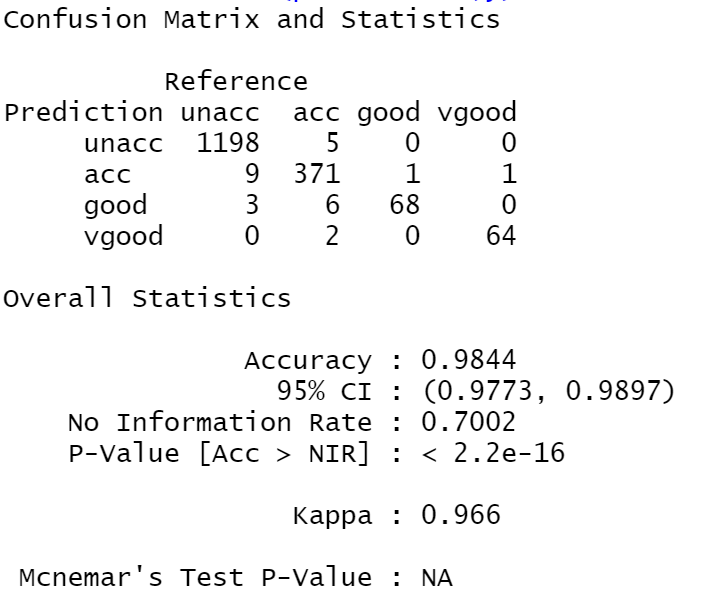
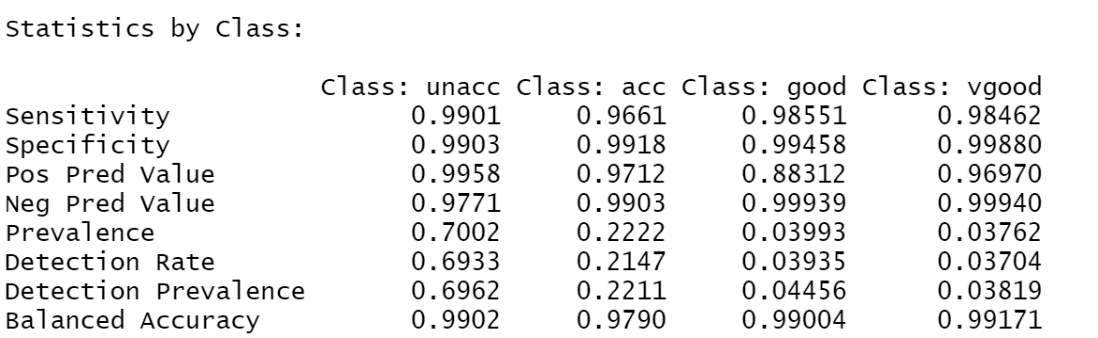


Figure 20 ROC Plot

Confusion Matrix and Statistics





Plot of one tree from random forest

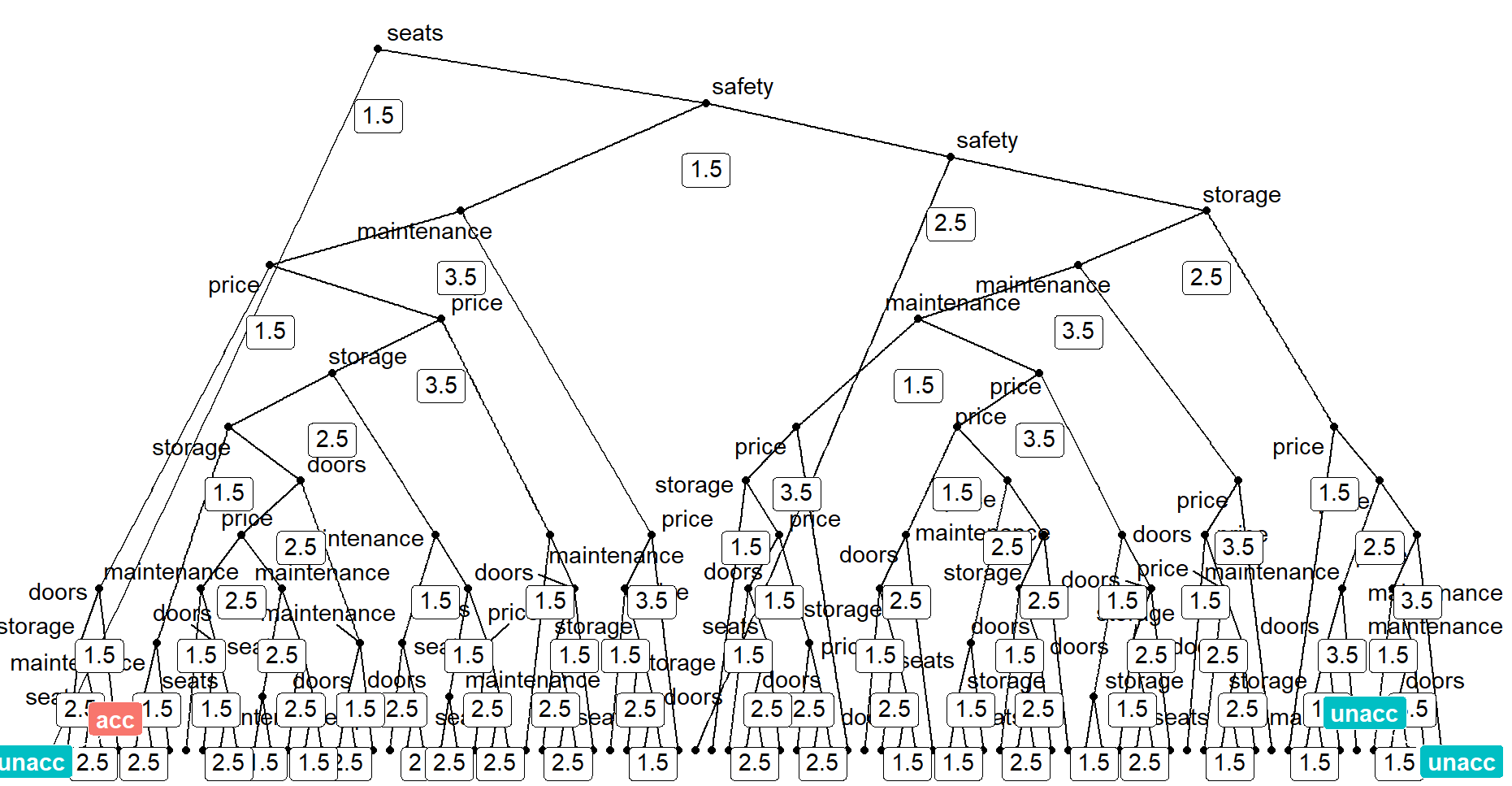


Figure 21 Sample Tree

AUC

Logo

Description automatically generated

Summary

Final model gave accuracy of 98.44% and AUC of 1 with parameters mtry=6, minsplit=10 and ntree=100.

# Conclusion

The result analysis aligned with objective stated, optimized models are created for classification problem and have high accuracy (AUC for Decision Tree Analysis, Random Forest Analysis and Caret package are 0.8878, 0.9921, and 1 respectively). Model generated from Caret package shows the best predictive power. However, the accuracy of the models trends too high, which probability may not reflect the characteristics of the actual problem. It may cause by small data set resulting overfitting problem.

On the other hand, an interesting finding from Variable Importance Plots (Figure 9 & 16) is safety played the most important factor in customer making the purchase decision. The next important factors are price, maintenance and no of seats depending on the model referenced.

For next step actions, the first thing is to increase size of data set to refine the model. It is crucial to overcome the overfitting problem. Moreover, when time and resource are allowed more complex model like Neural Networks and Gradient Boosting can be test in future to see if any performance gain can be achieved.

# Appendix

## R Script

### Decision Tree

library(rpart)

library(randomForest)

library(caret)

carData<-read.csv("D:/MSc CDA/Semester 2/3. Data and Text Mining\_MCDA5580/Assignment 2/car.data")

head(carData)

trainIndex<-createDataPartition(carData$shouldBuy, p=0.8, list=FALSE)

train<-carData[trainIndex,]

test<-carData[-trainIndex,]

treeCar<-rpart(shouldBuy~price+maintenance+doors+seats+storage+safety,data=train,method="class",control=rpart.control(minsplit=6))

#Plot Tree

plot(treeCar,uniform=TRUE,main="Decision Tree Should Buy")

text(treeCar,use.n=TRUE,all=TRUE,cex=.5,main="Decision Tree Should Buy")

plot(treeCar,uniform=TRUE,main="Decision Tree Should Buy")

# Plot Tree rules

rpart.rules(treeCar)

#Preduction

predCar<-predict(treeCar,newdata=test,type="class")

predCarProb<-predict(treeCar,newdata = test, type="prob")

head(predCar)

treeCarCM<-table(test[,"shouldBuy"],predCar)

treeCarCM

plot(predCar)

plot(treeCarCM)

sum(diag(treeCarCM)/sum(treeCarCM))

#Print Confusion Matrix & stat

test$shouldBuy <- factor(test$shouldBuy , levels=c("unacc", "acc", "good", "vgood"),

ordered=TRUE)

confusionMatrix(predCar, test[,"shouldBuy"])

#AUC

library(pROC)

roc.multi=multiclass.roc(test$shouldBuy, predCarProb[,2])

auc(roc.multi)

# Plot ROC

predictions<-as.data.frame(predict(treeCar, newdata = test, type = "prob"))

predictions$predict<-names(predictions)[1:4][apply(predictions[,1:4], 1, which.max)]

predictions$observed<-test$shouldBuy

head(predictions)

roc.unacc<-roc(ifelse(predictions$observed=="unacc", "unacc", "non-unacc"), as.numeric(predictions$unacc))

roc.acc<-roc(ifelse(predictions$observed=="acc", "acc", "non-acc"), as.numeric(predictions$acc))

roc.good<-roc(ifelse(predictions$observed=="good", "good", "non-good"), as.numeric(predictions$good))

roc.vgood<-roc(ifelse(predictions$observed=="vgood", "vgood", "non-vgood"), as.numeric(predictions$vgood))

plot(roc.unacc,main="UNACC")

plot(roc.acc,main="ACC")

plot(roc.good,main="GOOD")

plot(roc.vgood,main="VGOOD")

### Random Forest

library(rpart.plot)

library(caret)

library(randomForest)

library(pROC)

carData=read.csv('D:/#Spring 2023/5580 - Text Mining/Assignment2/car.csv',header = TRUE)

trainIndex <- createDataPartition(carData$shouldBuy, p = 0.8, list = FALSE)

train <- carData[trainIndex,]

test <- carData[-trainIndex,]

train$shouldBuy <- factor(train$shouldBuy , levels=c("unacc", "acc", "good", "vgood"),

ordered=TRUE)

test$shouldBuy <- factor(test$shouldBuy , levels=c("unacc", "acc", "good", "vgood"),

ordered=TRUE)

set.seed(123)

# Perform hyperparameter tuning

# 1. mtry

model <- tuneRF(train[,1:6], train[,7] , mtryStart = 2)

# 2. nodesize

# Try for the best value of node size

rf=randomForest(shouldBuy~price+maintenance+doors+seats+storage+safety,data=train, mtry =6, nodesize= 1)

rfp <- predict(rf, newdata = test)

rfCM = table(rfp,test$shouldBuy)

rfProb =predict(rf, newdata = test,type="prob")

roc.multi <-multiclass.roc(test$shouldBuy, rfProb[,2])

rfCM

sum(diag(rfCM))/sum(rfCM)

rf=randomForest(shouldBuy~price+maintenance+doors+seats+storage+safety,data=train, mtry =6, nodesize= 5)

rfp <- predict(rf, newdata = test)

rfCM = table(rfp,test$shouldBuy)

rfProb =predict(rf, newdata = test,type="prob")

roc.multi <-multiclass.roc(test$shouldBuy, rfProb[,2])

rfCM

sum(diag(rfCM))/sum(rfCM)

rf=randomForest(shouldBuy~price+maintenance+doors+seats+storage+safety,data=train, mtry =6, nodesize= 10)

rfp <- predict(rf, newdata = test)

rfCM = table(rfp,test$shouldBuy)

rfProb =predict(rf, newdata = test,type="prob")

roc.multi <-multiclass.roc(test$shouldBuy, rfProb[,2])

rfCM

sum(diag(rfCM))/sum(rfCM)

# Build the optimized model

rf=randomForest(shouldBuy~price+maintenance+doors+seats+storage+safety,data=train, mtry =6, nodesize= 1)

# Make predictions on the testing set

rfp <- predict(rf, newdata = test)

rfCM = table(rfp,test$shouldBuy)

rfProb=predict(rf, newdata = test,type="prob")

# plot ROC for different classes

predictions <- as.data.frame(predict(rf, newdata = test, type = "prob"))

predictions$predict <- names(predictions)[1:4][apply(predictions[,1:4], 1, which.max)]

predictions$observed <- test$shouldBuy

roc.unacc <- roc(ifelse(predictions$observed=="unacc", "unacc", "non-unacc"), as.numeric(predictions$unacc))

roc.acc <- roc(ifelse(predictions$observed=="acc", "acc", "non-acc"), as.numeric(predictions$acc))

roc.good <- roc(ifelse(predictions$observed=="good", "good", "non-good"), as.numeric(predictions$good))

roc.vgood <- roc(ifelse(predictions$observed=="vgood", "vgood", "non-vgood"), as.numeric(predictions$vgood))

plot(roc.unacc)

plot(roc.acc)

plot(roc.good)

plot(roc.vgood)

#Print Confusion Matrix & stat

confusionMatrix(rfp, test$shouldBuy)

#Evaluate variable importance

importance(rf)

varImpPlot(rf)

#Print AUC

rfProb <- as.numeric(predict(rf, newdata = test, type = 'response'))

roc.multi <- multiclass.roc(test$shouldBuy, rfProb)

roc.multi$auc

#Plot of a Single Tree in the Random Forecast model

# Plot function

library(dplyr)

library(ggraph)

library(igraph)

tree\_func <- function(final\_model,

tree\_num) {

# get tree by index

tree <- randomForest::getTree(final\_model,

k = tree\_num,

labelVar = TRUE) %>%

tibble::rownames\_to\_column() %>%

# make leaf split points to NA, so the 0s won't get plotted

mutate(`split point` = ifelse(is.na(prediction), `split point`, NA))

# prepare data frame for graph

graph\_frame <- data.frame(from = rep(tree$rowname, 2),

to = c(tree$`left daughter`, tree$`right daughter`))

# convert to graph and delete the last node that we don't want to plot

graph <- graph\_from\_data\_frame(graph\_frame) %>%

delete\_vertices("0")

# set node labels

V(graph)$node\_label <- gsub("\_", " ", as.character(tree$`split var`))

V(graph)$leaf\_label <- as.character(tree$prediction)

V(graph)$split <- as.character(round(tree$`split point`, digits = 2))

# plot

plot <- ggraph(graph, 'dendrogram') +

theme\_bw() +

geom\_edge\_link() +

geom\_node\_point() +

geom\_node\_text(aes(label = node\_label), na.rm = TRUE, repel = TRUE) +

geom\_node\_label(aes(label = split), vjust = 2.5, na.rm = TRUE, fill = "white") +

geom\_node\_label(aes(label = leaf\_label, fill = leaf\_label), na.rm = TRUE,

repel = TRUE, colour = "white", fontface = "bold", show.legend = FALSE) +

theme(panel.grid.minor = element\_blank(),

panel.grid.major = element\_blank(),

panel.background = element\_blank(),

plot.background = element\_rect(fill = "white"),

panel.border = element\_blank(),

axis.line = element\_blank(),

axis.text.x = element\_blank(),

axis.text.y = element\_blank(),

axis.ticks = element\_blank(),

axis.title.x = element\_blank(),

axis.title.y = element\_blank(),

plot.title = element\_text(size = 18))

print(plot)

}

# Plot tree

tree\_func(final\_model = rf, 1)

### Caret Package

#rm(list = ls()) # Clear Environment

#cat("\014") # Clear Console

library(randomForest)

library(caret)

library(pROC)

library(matrixStats)

library(Hmisc)

# Customized model over caret to add ntree, maxdepth and minsplit to grid search

# Reference https://rpubs.com/phamdinhkhanh/389752

customRF <- list(type = "Classification",

library = "randomForest",

loop = NULL)

customRF$parameters <- data.frame(parameter = c("mtry", "ntree", "maxdepth", "minsplit"),

class = rep("numeric", 4),

label = c("mtry", "ntree", "maxdepth", "minsplit"))

customRF$grid <- function(x, y, len = NULL, search = "grid") {}

customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs)

{

randomForest(x, y,

importance = TRUE,

mtry = param$mtry,

ntree=param$ntree,

maxdepth=param$maxdepth,

minsplit=param$minsplit)

}

#Predict label

customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)

predict(modelFit, newdata)

#Predict prob

customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)

predict(modelFit, newdata, type = "prob")

customRF$sort <- function(x) x[order(x[,1]),]

customRF$levels <- function(x) x$classes

# Main script starts here

carData=read.csv('car.csv',header = TRUE)

x=carData[,1:6]

y=carData[,7]

# Describe data

describe(x)

describe(y)

y <- factor(y, levels=c("unacc", "acc", "good", "vgood"),

ordered=TRUE)

# Find Correlation between features

# Loop over all pairs of categorical variables

cols <- colnames(x)

for (i in 1:(length(cols) - 1))

{

for (j in (i + 1):length(cols))

{

# Create a contingency table of the two categorical variables

contingency\_table <- table(x[, cols[i]], x[, cols[j]])

print(paste('Contingency matrix of ', cols[i], ' vs ', cols[j]))

print(contingency\_table)

}

}

# Modeling

set.seed(123)

# Training control for train-test split

control <- trainControl(method = "repeatedcv",

number = 5,

repeats = 3)

# Tuning grid search

tuning\_grid <- expand.grid(mtry = c(3:6),

ntree = c(1, 5, 10, 100),

maxdepth = c(5),

minsplit = c(3, 5, 10))

# Model fitting and tuning search

rf\_default <- train(x,y,

method=customRF,

tuneGrid=tuning\_grid,

metric="Accuracy",

trControl=control)

# Print the model search results and final model

plot(rf\_default)

print(rf\_default$results)

print(rf\_default)

# Importance plot

varImpPlot(rf\_default$finalModel)

# Predict and print confusion matrix

predictions <- predict(rf\_default$finalModel, data = x)

confusionMatrix(predictions,y)

#Print AUC

rfProb <- as.numeric(predict(rf\_default$finalModel,

newdata = x, type = 'response'))

roc.multi <- multiclass.roc(y, rfProb)

roc.multi$auc

# Predict and print roc curve

predictions\_prob <- predict(rf\_default$finalModel,

data = x,

type = "prob")

predictions <- as.data.frame(predictions\_prob)

predictions$predict <- names(predictions)[max.col(predictions[, 1:4], ties.method = "first")]

predictions$observed <- y

head(predictions)

roc.unacc <- roc(ifelse(predictions$observed=="unacc", "unacc", "non-unacc"), as.numeric(predictions$unacc))

roc.acc <- roc(ifelse(predictions$observed=="acc", "acc", "non-acc"), as.numeric(predictions$acc))

roc.good <- roc(ifelse(predictions$observed=="good", "good", "non-good"), as.numeric(predictions$good))

roc.vgood <- roc(ifelse(predictions$observed=="vgood", "vgood", "non-vgood"), as.numeric(predictions$vgood))

plot(roc.unacc, title="Class uncc")

plot(roc.acc, title="Class acc")

plot(roc.good, title="Class good")

plot(roc.vgood, title="Class vgood")

#Plot of a Single Tree in the Random Forecast model

# Plot function

library(dplyr)

library(ggraph)

library(igraph)

tree\_func <- function(final\_model,tree\_num)

{

# get tree by index

tree <- randomForest::getTree(final\_model,

k = tree\_num,

labelVar = TRUE) %>%

tibble::rownames\_to\_column() %>%

# make leaf split points to NA, so the 0s won't get plotted

mutate(`split point` = ifelse(is.na(prediction), `split point`, NA))

# prepare data frame for graph

graph\_frame <- data.frame(from = rep(tree$rowname, 2),

to = c(tree$`left daughter`, tree$`right daughter`))

# convert to graph and delete the last node that we don't want to plot

graph <- graph\_from\_data\_frame(graph\_frame) %>%

delete\_vertices("0")

# set node labels

V(graph)$node\_label <- gsub("\_", " ", as.character(tree$`split var`))

V(graph)$leaf\_label <- as.character(tree$prediction)

V(graph)$split <- as.character(round(tree$`split point`, digits = 2))

# plot

plot <- ggraph(graph, 'dendrogram') +

theme\_bw() +

geom\_edge\_link() +

geom\_node\_point() +

geom\_node\_text(aes(label = node\_label), na.rm = TRUE, repel = TRUE) +

geom\_node\_label(aes(label = split), vjust = 2.5, na.rm = TRUE, fill = "white") +

geom\_node\_label(aes(label = leaf\_label, fill = leaf\_label), na.rm = TRUE,

repel = TRUE, colour = "white", fontface = "bold", show.legend = FALSE) +

theme(panel.grid.minor = element\_blank(),

panel.grid.major = element\_blank(),

panel.background = element\_blank(),

plot.background = element\_rect(fill = "white"),

panel.border = element\_blank(),

axis.line = element\_blank(),

axis.text.x = element\_blank(),

axis.text.y = element\_blank(),

axis.ticks = element\_blank(),

axis.title.x = element\_blank(),

axis.title.y = element\_blank(),

plot.title = element\_text(size = 18))

print(plot)

}

# Plot tree

tree\_func(final\_model = rf\_default$finalModel, 1)

## Reference/ Citation

Supervised Classification Model

<https://emeritus.org/in/learn/types-of-supervised-learning/>

<https://machinelearningmastery.com/types-of-classification-in-machine-learning/>

Random Forest Model

<https://cran.r-project.org/web/packages/randomForest/randomForest.pdf>

<https://www.listendata.com/2014/11/random-forest-with-r.html#id-35df3b>

Caret Package

<https://www.machinelearningplus.com/machine-learning/caret-package/>

[The caret Package (topepo.github.io)](https://topepo.github.io/caret/index.html)

Random Forest Model tuning

<https://afit-r.github.io/random_forests>

Plot ROC Curve

<https://stackoverflow.com/questions/46124424/how-can-i-draw-a-roc-curve-for-a-randomforest-model-with-three-classes-in-r>

Plot tree in Random Forest

<https://shiring.github.io/machine_learning/2017/03/16/rf_plot_ggraph>

Customize Caret package for additional tuning parameter

<https://rpubs.com/phamdinhkhanh/389752>

Overfitting

<https://crunchingthedata.com/random-forest-overfitting/>