|  |
| --- |
| Classification with car data |
| MCDA 5580 Data Mining Assignment 2 Report |

|  |
| --- |
|  |

****

|  |  |
| --- | --- |
| ­­ | |
| Contents  [Executive Summary 2](#_Toc10577740)  [About the Data 2](#_Toc10577741)  [Classification Analysis 2](#_Toc10577742)  [Conclusion 10](#_Toc10577743)  [Appendix 11](#_Toc10577744)  [Captions 16](#_Toc10577745) | |

# Executive Summary

This report will introduce the car data we will use for classification analysis, we will use Open Refine to number coding the dataset and import into R. In R, we use the scorecard library to split the full dataset into 75% training data and 25% testing data. To find out the best minsplit value to training the decision tree we choose confusion matrix, ROC and AUC to calculate the accuracy. At the end of analysis, we introduce k-fold and random forest to check the importance of each attribute with different k value and iteration.

# About the Data

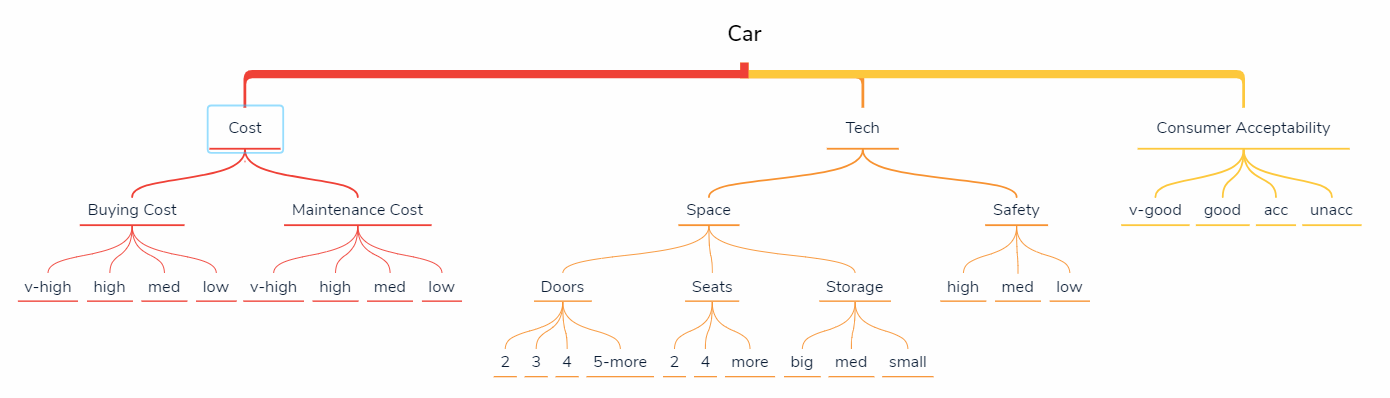


Figure Car Data Structure

The car data set included three parts: Cost, Tech and Consumer Acceptability. The Cost is split to Buying Cost and Maintenance Cost, each cost rated as v-high, high, med and low. Tech included two different components: Space and Safety. Space has Doors, Seats and Storage, for Doors, it can be 2, 3, 4 and 5-more; for Seats, it can be 2, 4 and more; for Storage, it can be big, med and small. In Safety part, the car was rated by high, med and low. The last attribute for the car is Consumer Acceptability which was rated as v-good, good, acc and unacc.

# Classification Analysis

Step.1:

We edit the data to make all columns numerical except for the “shouldBuy” column and then import the data into R.

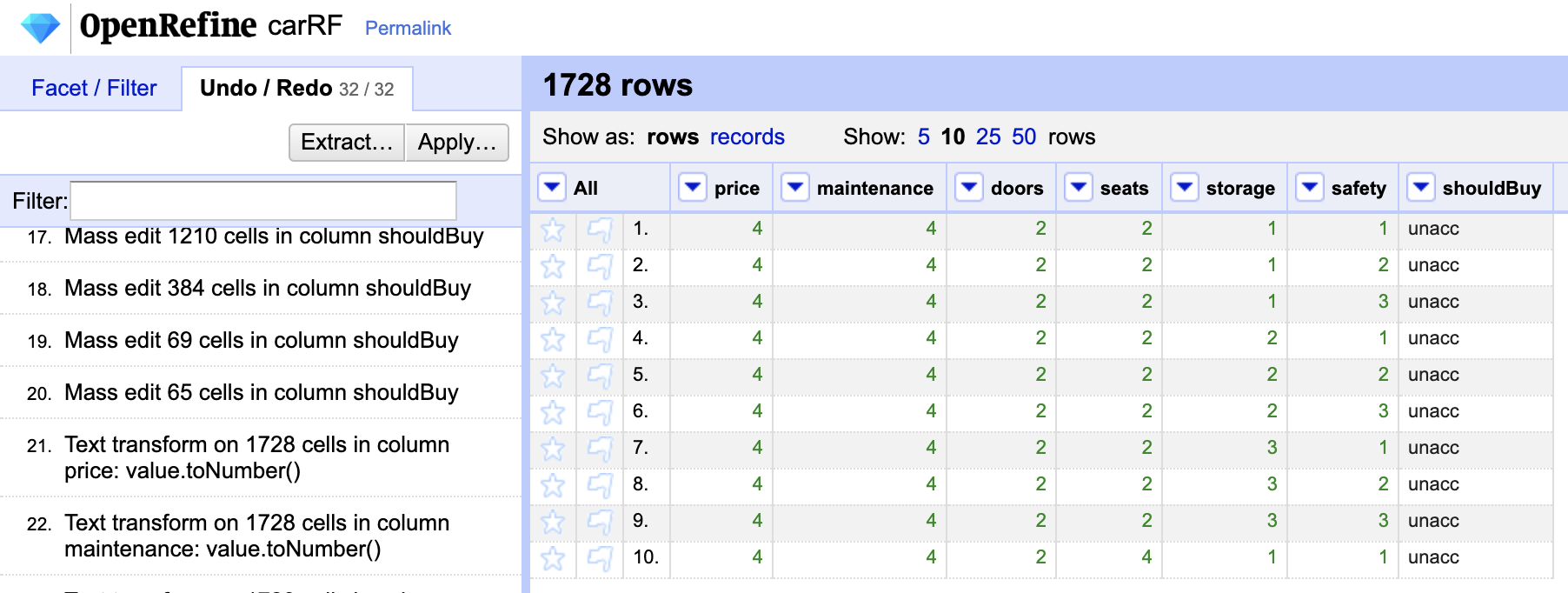
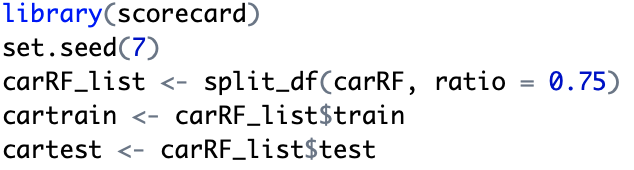


Figure Number Coding with Open Refine



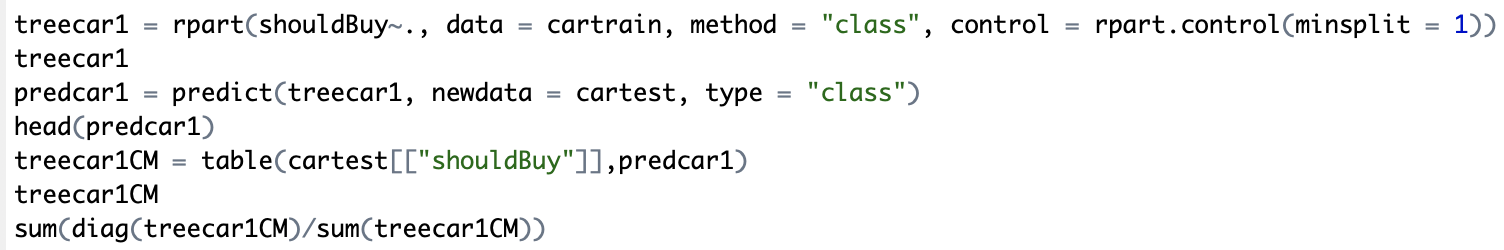
Step.2:

We divided the data into two: 75% car data into training data and 25% car data into testing data.



Step.3:

Now we using training data to train the decision tree and adjust the rpart control with different minsplit for 1, 10,50, 100, 130(which is near the 10% of training data size):



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Minsplit | 1 | 10 | 50 | 100 | 130 |
| Accuracy | 0.9407745 | 0.929385 | 0.9271071 | 0.8747153 | 0.8747153 |

Figure Minsplit Accuracy Table

Now we can see that with the minsplit of 1, which means the model is definitely over fed, the accuracy is 0.94. From the minsplit value from 10 to 50, the accuracy ay doesn’t change much. But between 50 and 100, there is a significant drop and then a fixed value between the minsplit of 100 and 130.

Step.4:

By checking the decision tree, we can find with which minsplit value the decision tree is not over fed. At first, we know for sure that with the minsplit value of 1, the decision tree is over fed for sure.



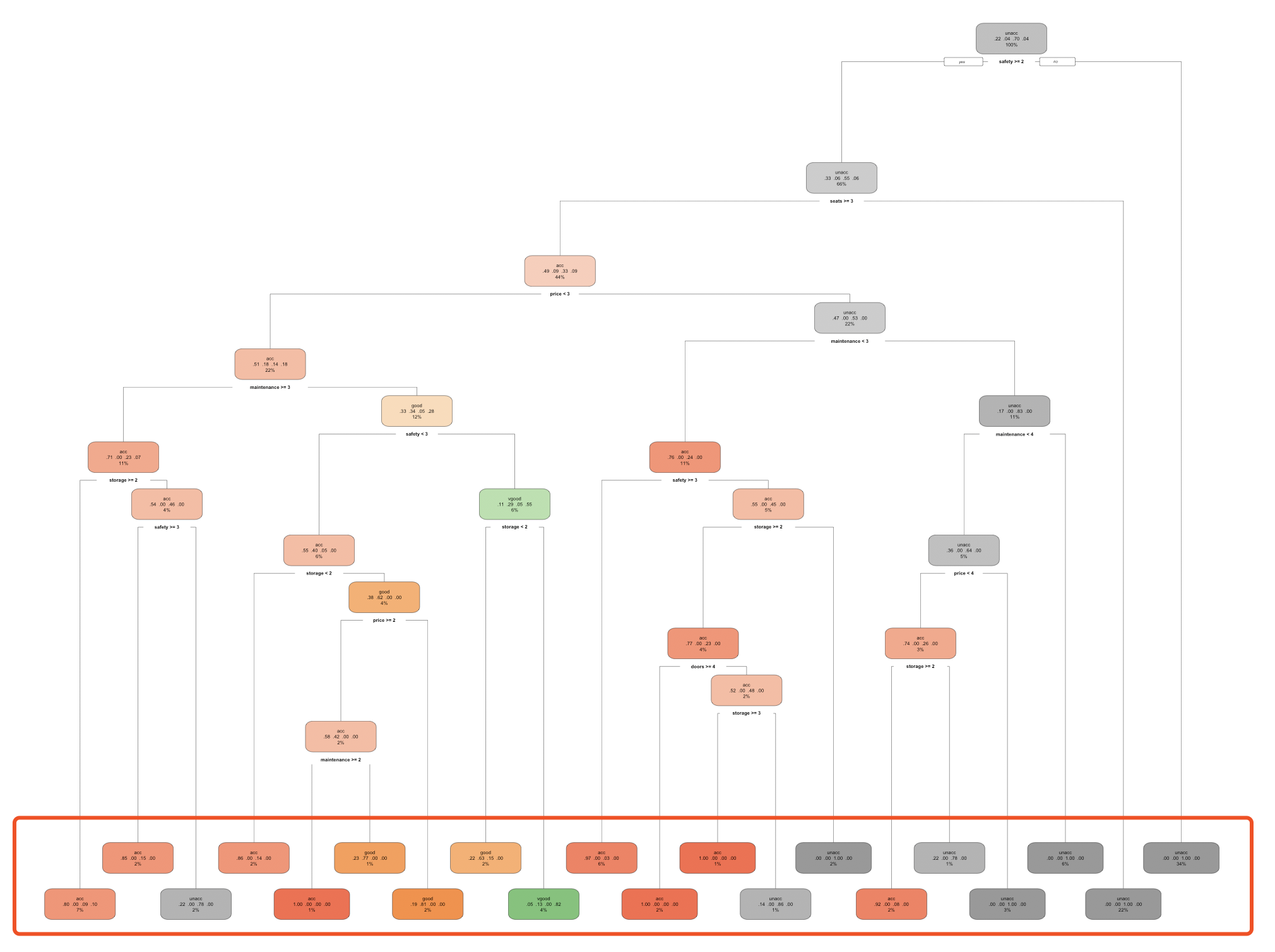


Figure Plot of Decision Tree Minsplit = 1

Plot of Decision Tree with Minsplit Value of 1

The decision trees with minsplit value of 10 are exactly the same with the decision tree with minsplit value of 1, so it’s definitely over fed. Although the decision tree with minsplit value of 50 is defferent, we can still assume it’s over fed because of the similar accuracy.

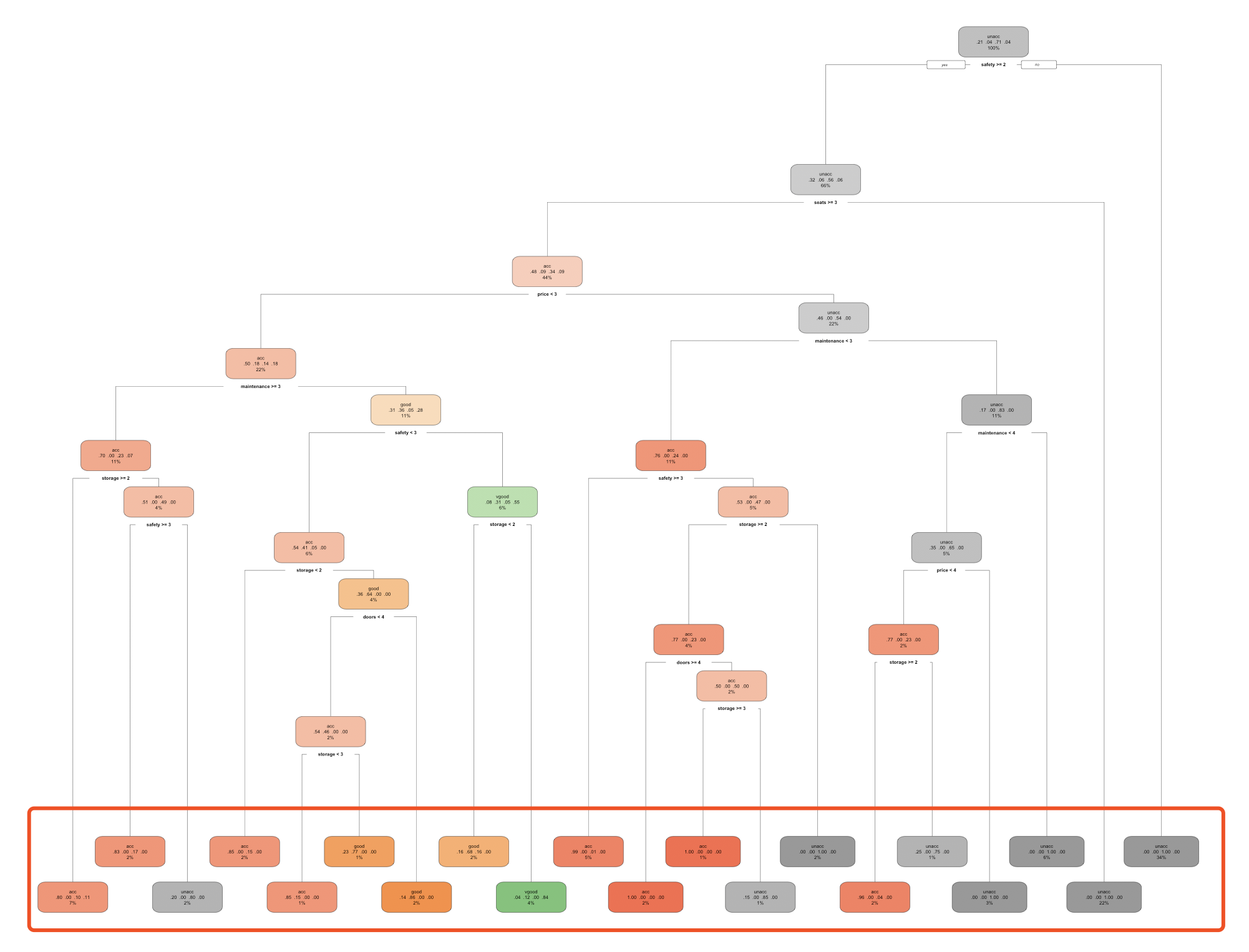


Figure PLOT OF DECISION TREE MINSPLIT = 10

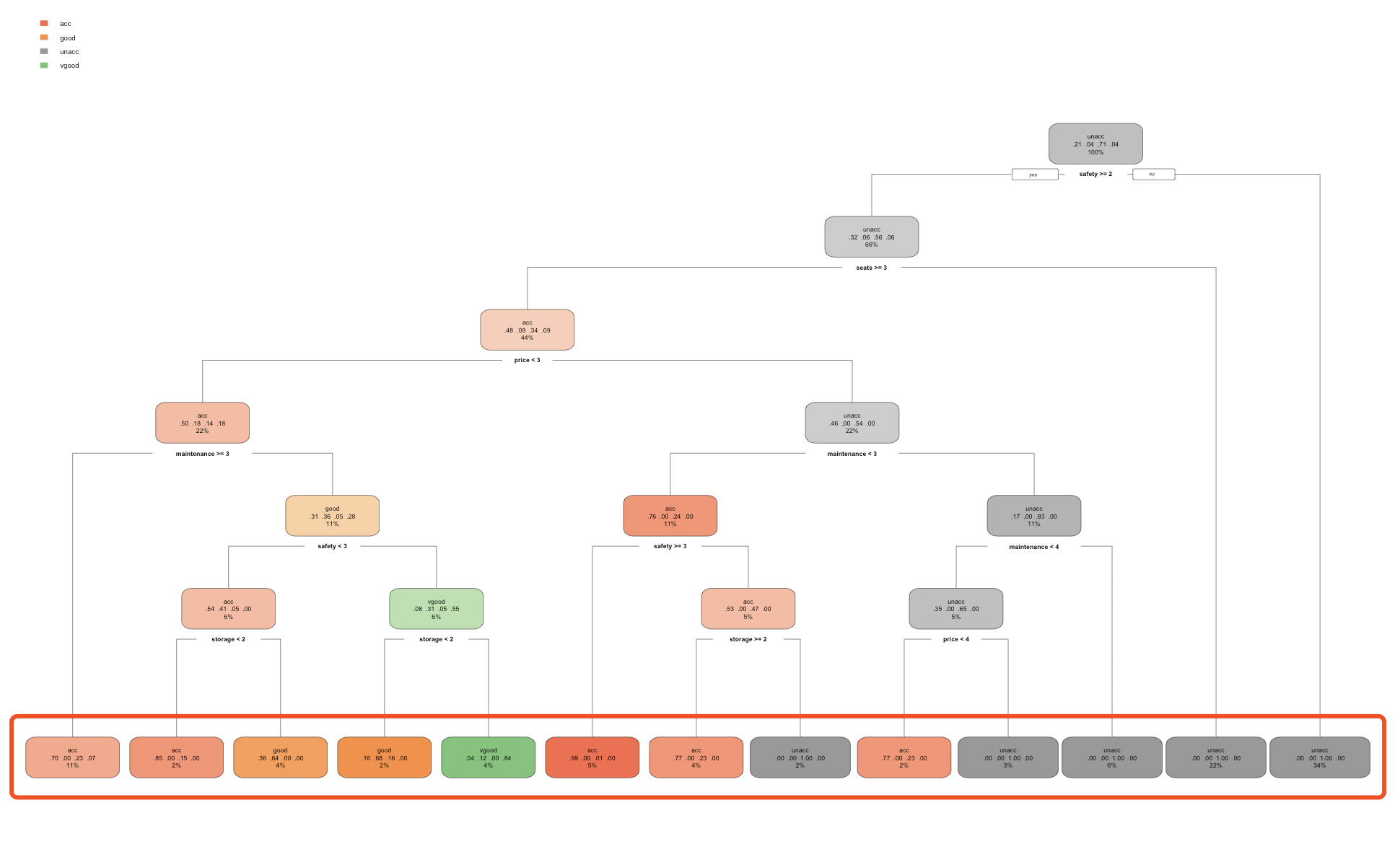


Figure PLOT OF DECISION TREE MINSPLIT = 50

Plots of Decision Trees with Minsplit Values of 10 (Figure 5) and 50 (Figure 6)

The decision trees with minsplit values of 100 and 130 are the same and the max depth is significant smaller than the former ones. So we can come to the conclusion that these models are not over feeded.

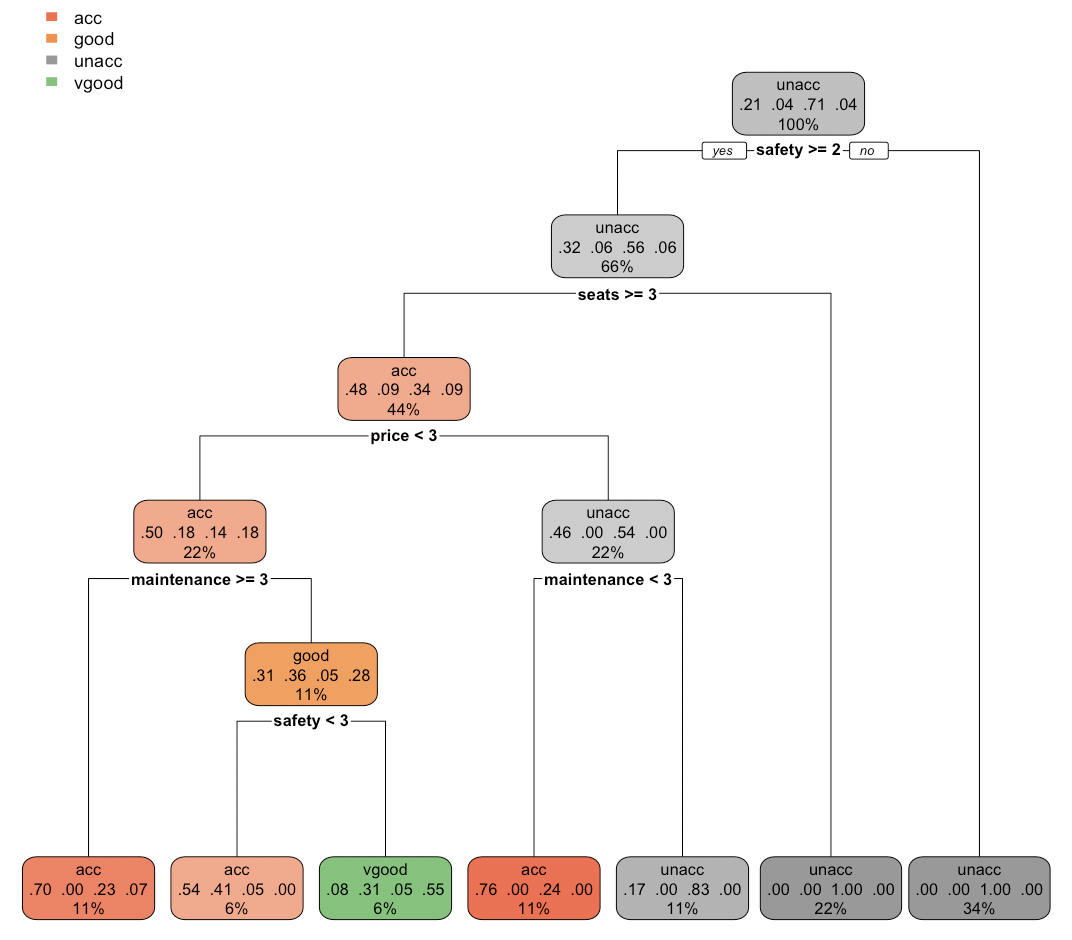


Figure PLOT OF DECISION TREE MINSPLIT = 100 and 130

Plots of Decision Trees with Minsplit Values of 100 and 130

The decision tree shows safety and seats are the top two important attributes in predicting whether we should buy the car or not. And if the safety is low, then the car is totally unacceptable.

Step.5:

Since the accuracy and decision tree is the same with the minsplit values of 100 and 130, we want to find out from where it becomes stable. So we write a function to plot the minsplit and the accuracy.



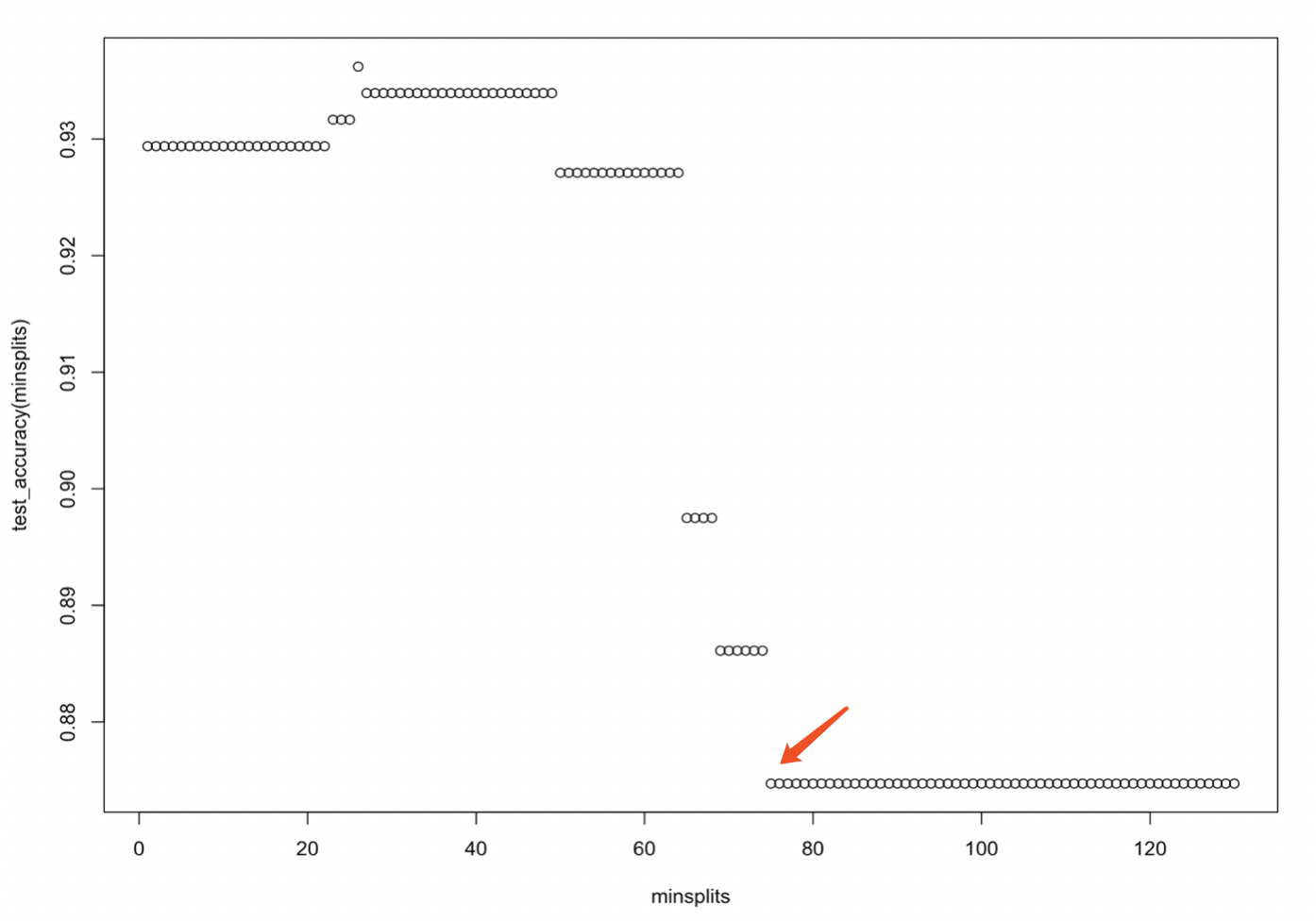
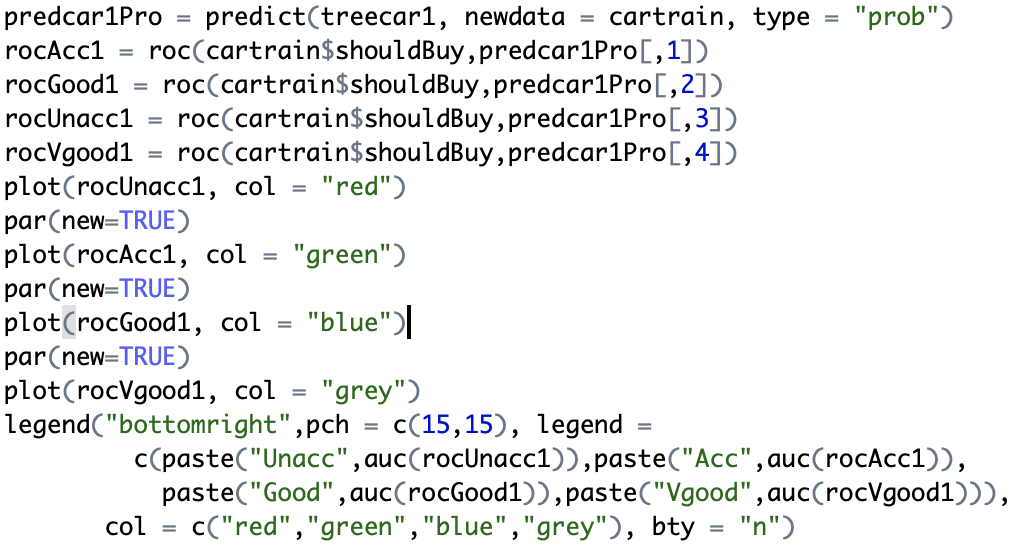


Figure Minsplit Accuracy Plot

From the graph, we can see with the minsplit value of 75, the accuracy becomes stable.

Step.5:

We also want to check the best minsplit value using ROC and AUC. Because there are four values in our predicting attribute, we plot the ROC of each value separately.



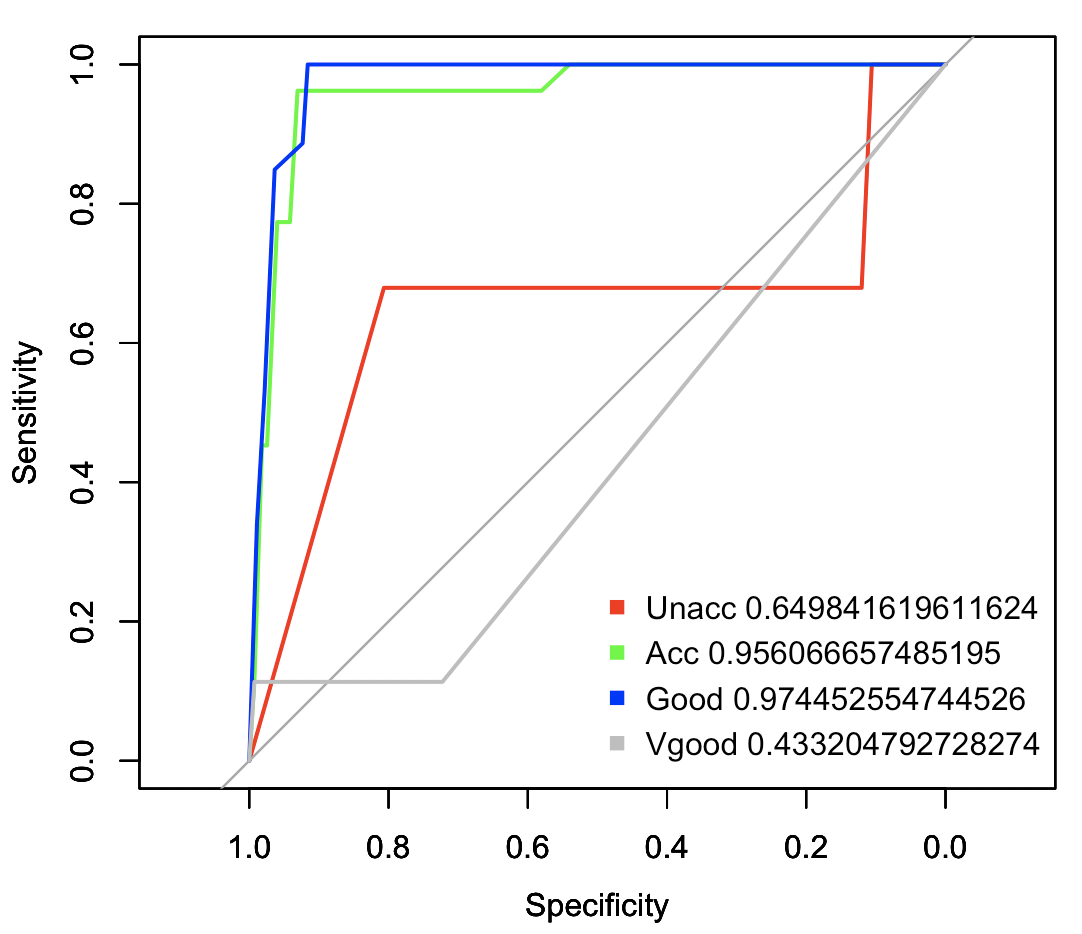
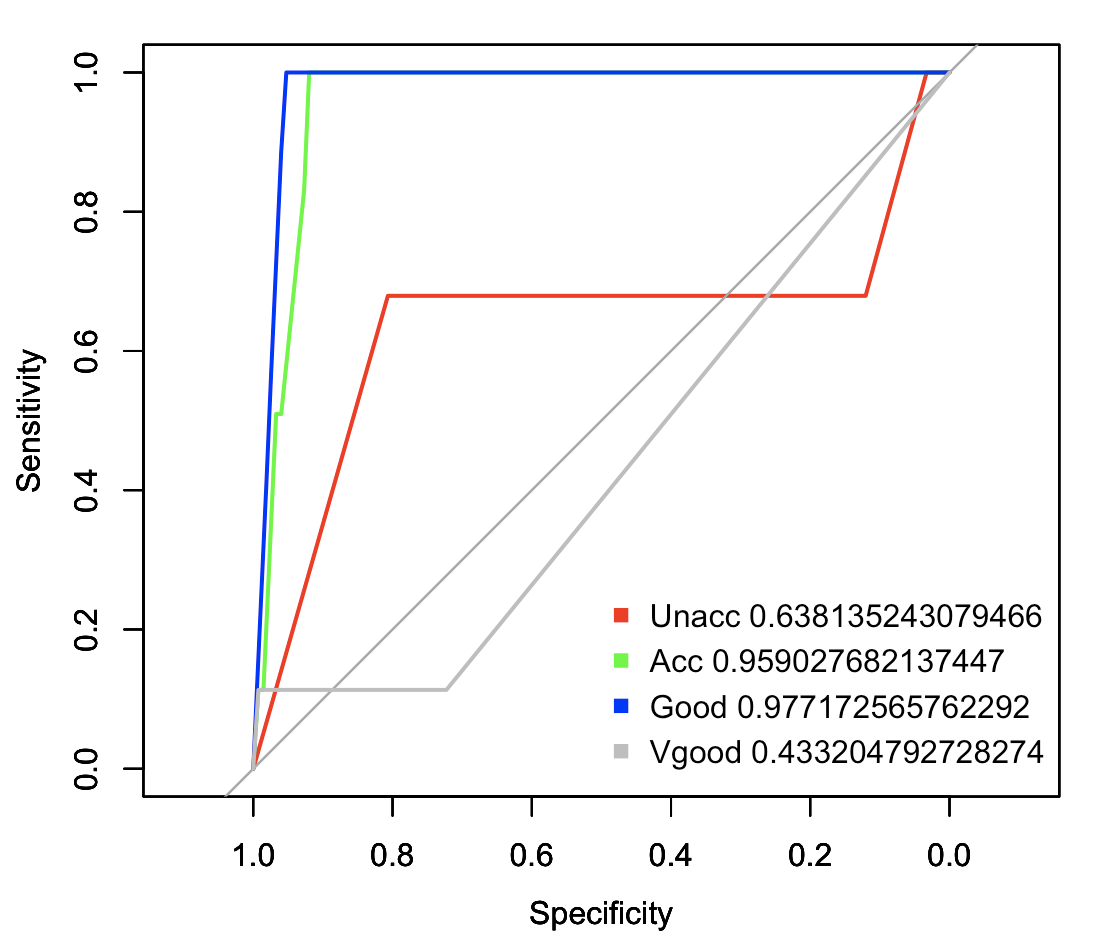


Figure ROC & AUC When Minsplit = 1 Figure 10 ROC & AUC When Minsplit = 10

ROC and AUC Values with Minsplit Values of 1 (Left) and 10 (Right)

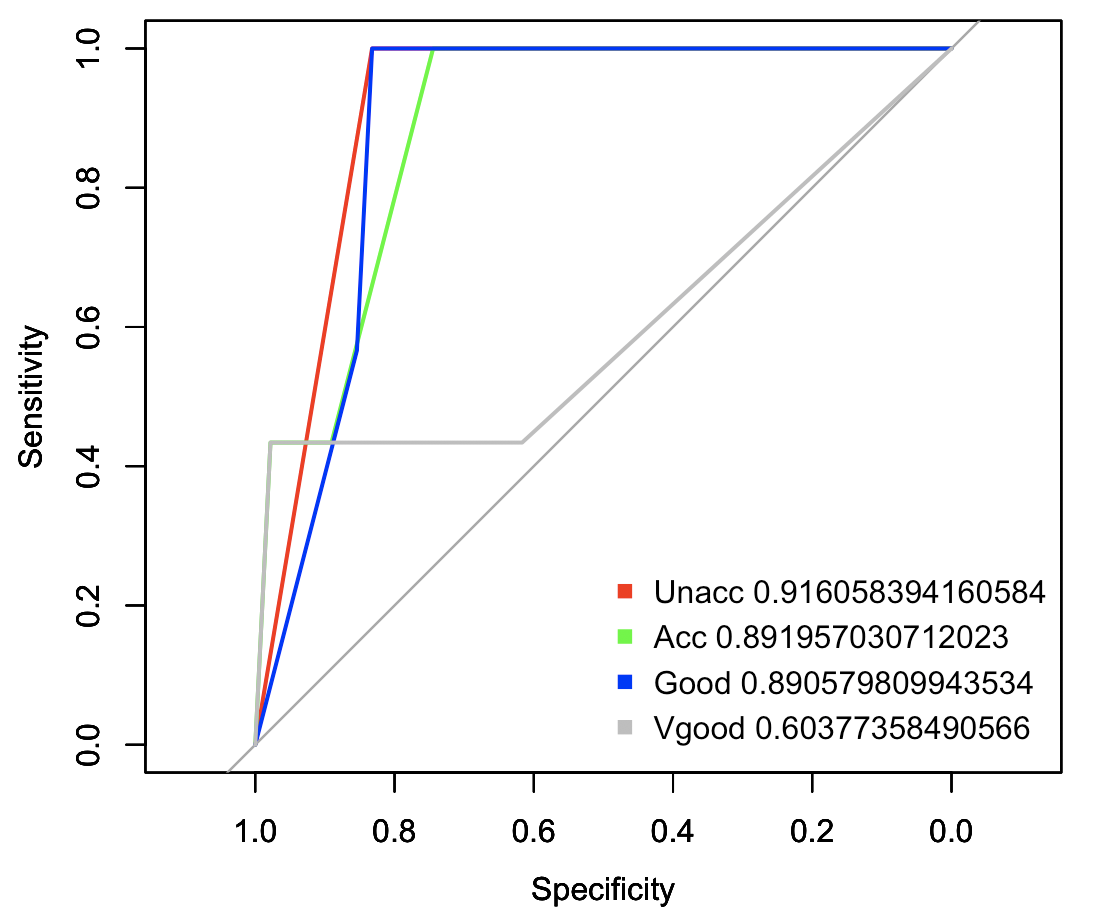
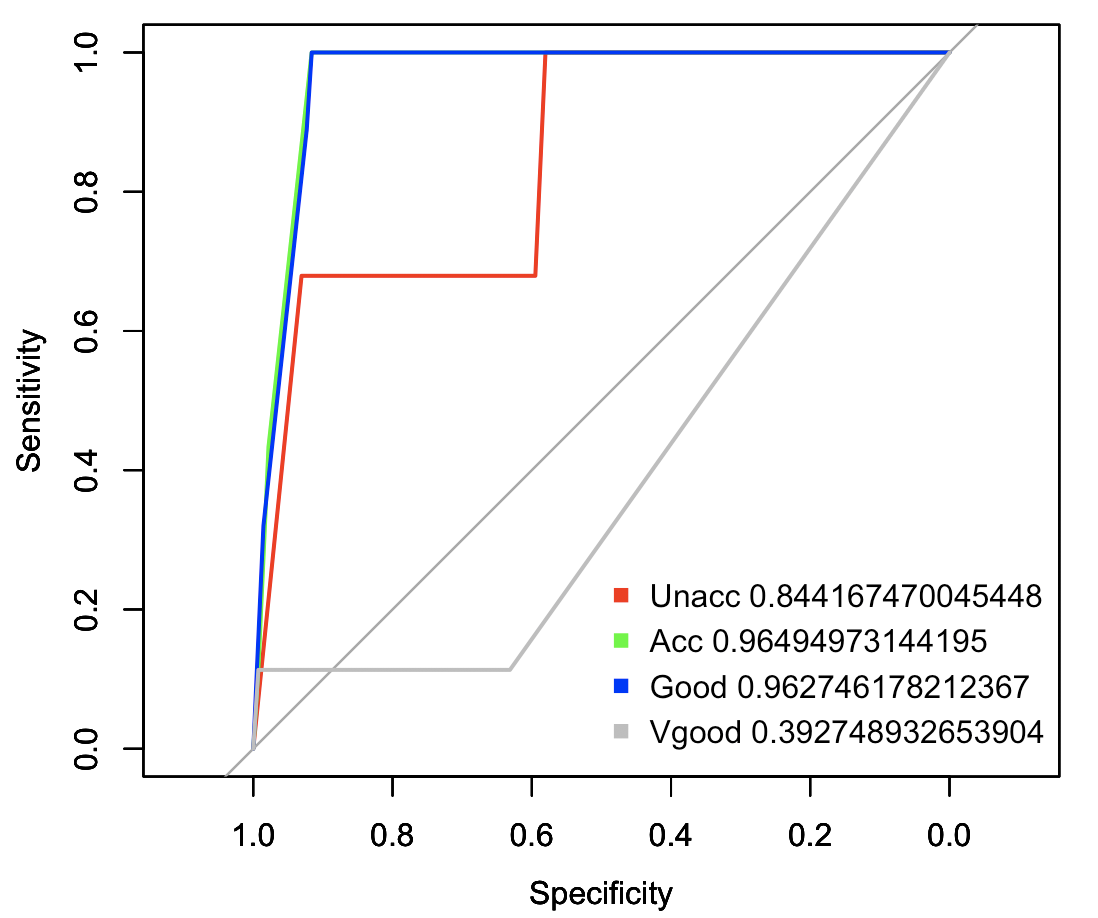


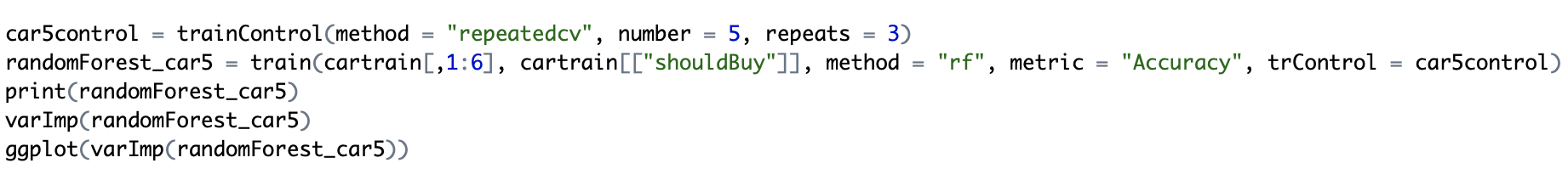
Figure ROC & AUC WHEN MINSPLIT = 50 Figure 12 ROC & AUC WHEN MINSPLIT = 75

ROC and AUC Values with Minsplit Values of 50 (Left) and 75 (Right)

We can see from the graphs that with minsplit value of 75, the overall AUC values and the curves are the most satisfactory ones. The AUC values of Unacc and Vgood significantly increase with the minsplit values from 1 to 75. From 75 on, the ROC graphs won’t change with increasing minsplit values.

Step.6:

We still want to check which attribute is the most important one. As a result, we introduce k-fold and random forest. We tried different k value and iteration to check if there is a significant change in the importance of each attribute.



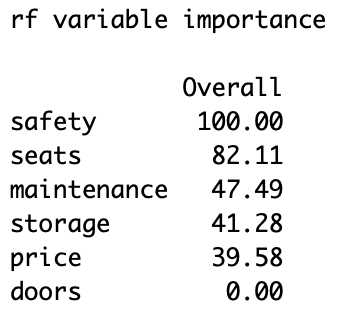
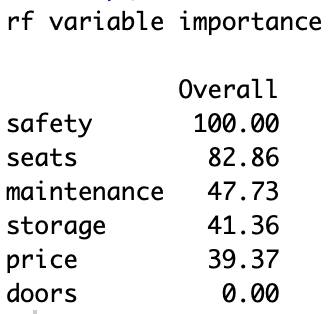
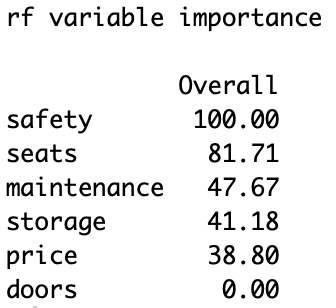
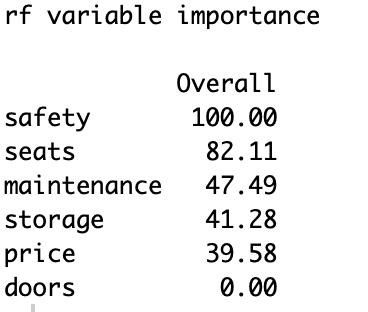
 

Figure Attribute importance When K in 5, 7, 10, 15 and Iteration = 3

Importance of Each Attribute with K Value of 5,7,10,15 (Left to Right) and 3 Iteration

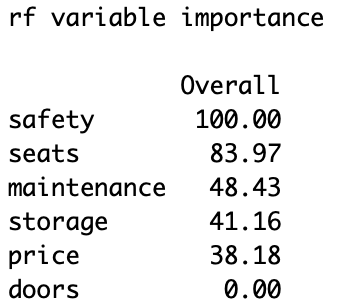
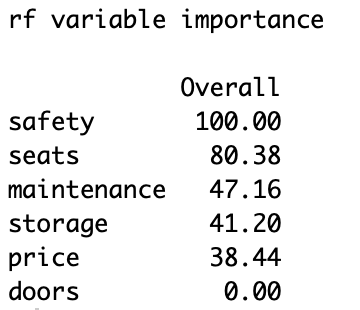
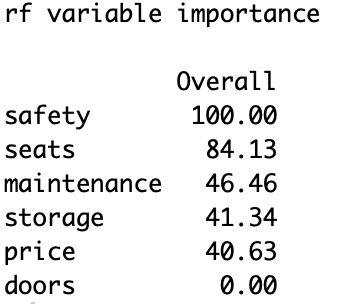
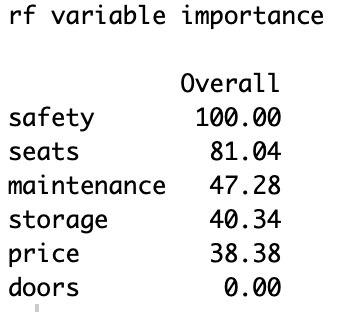


Figure Attribute Importance When K in 5, 7, 9, 11 with 10 Folds

Importance of Each Attribute with 10 Folds and Iteration Value of 5,7,9,11 (Left to Right)

With different k values and iteration numbers, the importance of attributes fluctuates but with the same ranking and similar values. We can come to the conclusion that safety is the most important attribute and the seats are the second. Also, the attribute of doors is of the least significance and even make no difference in predicting the attribute of shouldBuy.

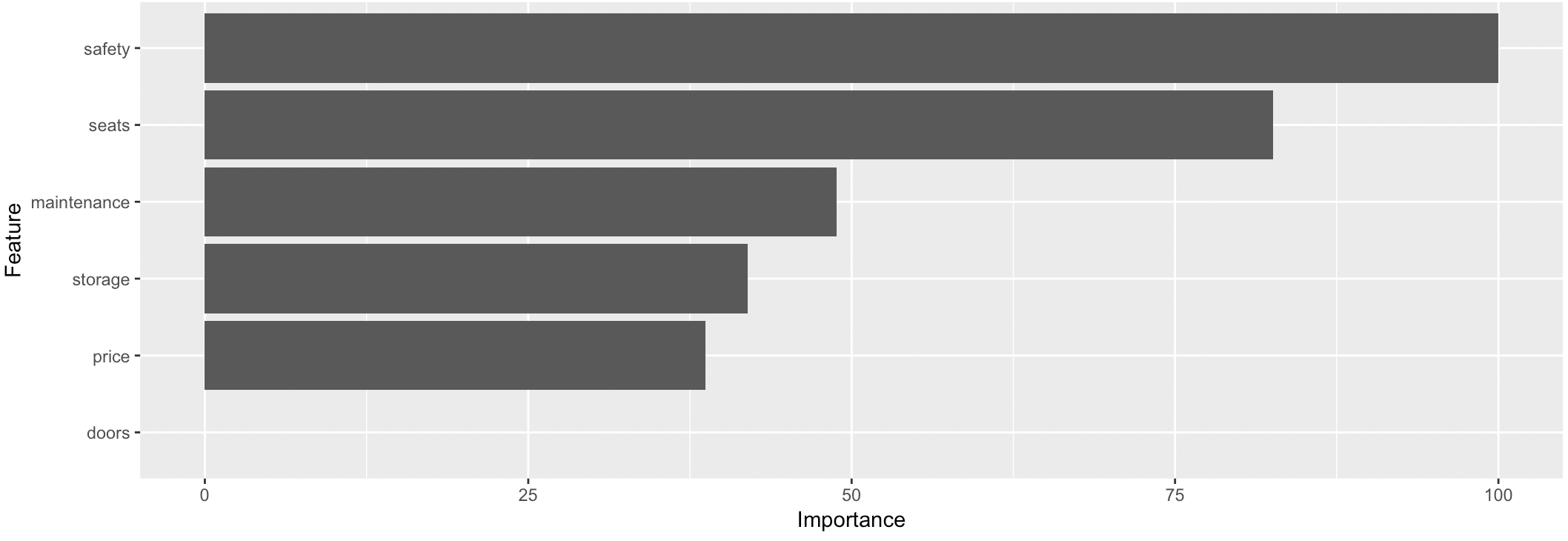


Figure Attribute importance

# Conclusion

In this analysis we found by confusion matrix the best minsplit value for decision tree is around 50 to 100, continue to graph decision tree we can see when the minsplit value set to 50, the decision tree did not over fed. Minsplit Accuracy (Figure 8) given the result when minsplit is 75 and larger, the accuracy becomes stable, we also use ROC and AUC graph to prove this. K-fold and random forest weighting the importance of all attributes in this order (from important to unimportant): Safety > Seats > Maintenance > Storage > Price > Doors.

# Appendix

library(rpart)

library(rpart.plot)

library(pROC)

library(caret)

carRF <- read.csv("/Users/jiyewang/Desktop/Courses/Data and Text Mining/A2/carRF.csv")

library(scorecard)

set.seed(7)

carRF\_list <- split\_df(carRF, ratio = 0.75)

cartrain <- carRF\_list$train

cartest <- carRF\_list$test

treecar1 = rpart(shouldBuy~., data = cartrain, method = "class", control = rpart.control(minsplit = 1))

treecar1

predcar1 = predict(treecar1, newdata = cartest, type = "class")

head(predcar1)

treecar1CM = table(cartest[["shouldBuy"]],predcar1)

treecar1CM

sum(diag(treecar1CM)/sum(treecar1CM))

rpart.plot(treecar1)

predcar1Pro = predict(treecar1, newdata = cartrain, type = "prob")

rocAcc1 = roc(cartrain$shouldBuy,predcar1Pro[,1])

rocGood1 = roc(cartrain$shouldBuy,predcar1Pro[,2])

rocUnacc1 = roc(cartrain$shouldBuy,predcar1Pro[,3])

rocVgood1 = roc(cartrain$shouldBuy,predcar1Pro[,4])

plot(rocUnacc1, col = "red")

par(new=TRUE)

plot(rocAcc1, col = "green")

par(new=TRUE)

plot(rocGood1, col = "blue")

par(new=TRUE)

plot(rocVgood1, col = "grey")

legend("bottomright",pch = c(15,15), legend = c(paste("Unacc",auc(rocUnacc1)),paste("Acc",auc(rocAcc1)),paste("Good",auc(rocGood1)),paste("Vgood",auc(rocVgood1))), col = c("red","green","blue","grey"), bty = "n")

treecar10 = rpart(shouldBuy~., data = cartrain, method = "class", control = rpart.control(minsplit = 10))

treecar10

predcar10 = predict(treecar10, newdata = cartest, type = "class")

head(predcar10)

treecar10CM = table(cartest[["shouldBuy"]],predcar10)

treecar10CM

sum(diag(treecar10CM)/sum(treecar10CM))

rpart.plot(treecar10)

predcar10Pro = predict(treecar10, newdata = cartrain, type = "prob")

rocAcc10 = roc(cartrain$shouldBuy,predcar10Pro[,1])

rocGood10 = roc(cartrain$shouldBuy,predcar10Pro[,2])

rocUnacc10 = roc(cartrain$shouldBuy,predcar10Pro[,3])

rocVgood10 = roc(cartrain$shouldBuy,predcar10Pro[,4])

plot(rocUnacc10, col = "red")

par(new=TRUE)

plot(rocAcc10, col = "green")

par(new=TRUE)

plot(rocGood10, col = "blue")

par(new=TRUE)

plot(rocVgood10, col = "grey")

legend("bottomright",pch = c(15,15), legend = c(paste("Unacc",auc(rocUnacc10)),paste("Acc",auc(rocAcc10)),paste("Good",auc(rocGood10)),paste("Vgood",auc(rocVgood10))), col = c("red","green","blue","grey"), bty = "n")

treecar50 = rpart(shouldBuy~., data = cartrain, method = "class", control = rpart.control(minsplit = 50))

treecar50

predcar50 = predict(treecar50, newdata = cartest, type = "class")

head(predcar10)

treecar50CM = table(cartest[["shouldBuy"]],predcar50)

treecar50CM

sum(diag(treecar50CM)/sum(treecar50CM))

rpart.plot(treecar50)

`predcar50Pro = predict(treecar50, newdata = cartrain, type = "prob")

rocUnacc50 = roc(cartrain$shouldBuy,predcar50Pro[,3])

rocAcc50 = roc(cartrain$shouldBuy,predcar50Pro[,1])

rocGood50 = roc(cartrain$shouldBuy,predcar50Pro[,2])

rocVgood50 = roc(cartrain$shouldBuy,predcar50Pro[,4])

plot(rocUnacc50, col = "red")

par(new=TRUE)

plot(rocAcc50, col = "green")

par(new=TRUE)

plot(rocGood50, col = "blue")

par(new=TRUE)

plot(rocVgood50, col = "grey")

legend("bottomright",pch = c(15,15), legend = c(paste("Unacc",auc(rocUnacc50)),paste("Acc",auc(rocAcc50)),paste("Good",auc(rocGood50)),paste("Vgood",auc(rocVgood50))), col = c("red","green","blue","grey"), bty = "n")

treecar100 = rpart(shouldBuy~., data = cartrain, method = "class", control = rpart.control(minsplit = 100))

treecar100

predcar100 = predict(treecar100, newdata = cartest, type = "class")

head(predcar100)

treecar100CM = table(cartest[["shouldBuy"]],predcar100)

treecar100CM

sum(diag(treecar100CM)/sum(treecar100CM))

rpart.plot(treecar100)

predcar100Pro = predict(treecar100, newdata = cartrain, type = "prob")

rocUnacc100 = roc(cartrain$shouldBuy,predcar100Pro[,3])

rocAcc100 = roc(cartrain$shouldBuy,predcar100Pro[,1])

rocGood100 = roc(cartrain$shouldBuy,predcar100Pro[,2])

rocVgood100 = roc(cartrain$shouldBuy,predcar100Pro[,4])

plot(rocUnacc100, col = "red")

par(new=TRUE)

plot(rocAcc100, col = "green")

par(new=TRUE)

plot(rocGood100, col = "blue")

par(new=TRUE)

plot(rocVgood100, col = "grey")

legend("bottomright",pch = c(15,15), legend = c(paste("Unacc",auc(rocUnacc100)),paste("Acc",auc(rocAcc100)),paste("Good",auc(rocGood100)),paste("Vgood",auc(rocVgood100))), col = c("red","green","blue","grey"), bty = "n")

treecar130 = rpart(shouldBuy~., data = cartrain, method = "class", control = rpart.control(minsplit = 130))

treecar130

predcar130 = predict(treecar130, newdata = cartest, type = "class")

head(predcar130)

treecar130CM = table(cartest[["shouldBuy"]],predcar130)

treecar130CM

sum(diag(treecar130CM)/sum(treecar130CM))

rpart.plot(treecar130)

predcar130Pro = predict(treecar130, newdata = cartrain, type = "prob")

rocUnacc130 = roc(cartrain$shouldBuy,predcar130Pro[,3])

rocAcc130 = roc(cartrain$shouldBuy,predcar130Pro[,1])

rocGood130 = roc(cartrain$shouldBuy,predcar130Pro[,2])

rocVgood130 = roc(cartrain$shouldBuy,predcar130Pro[,4])

plot(rocUnacc130, col = "red")

par(new=TRUE)

plot(rocAcc130, col = "green")

par(new=TRUE)

plot(rocGood130, col = "blue")

par(new=TRUE)

plot(rocVgood130, col = "grey")

legend("bottomright",pch = c(15,15), legend = c(paste("Unacc",auc(rocUnacc130)),paste("Acc",auc(rocAcc130)),paste("Good",auc(rocGood130)),paste("Vgood",auc(rocVgood130))), col = c("red","green","blue","grey"), bty = "n")

treecarT = rpart(shouldBuy~., data = cartrain, method = "class", control = rpart.control(minsplit = 75))

treecarT

predcarT = predict(treecarT, newdata = cartest, type = "class")

head(predcarT)

treecarTCM = table(cartest[["shouldBuy"]],predcarT)

treecarTCM

sum(diag(treecarTCM)/sum(treecarTCM))

rpart.plot(treecarT)

predcarTPro = predict(treecarT, newdata = cartrain, type = "prob")

rocUnaccT = roc(cartrain$shouldBuy,predcarTPro[,3])

rocAccT = roc(cartrain$shouldBuy,predcarTPro[,1])

rocGoodT = roc(cartrain$shouldBuy,predcarTPro[,2])

rocVgoodT = roc(cartrain$shouldBuy,predcarTPro[,4])

plot(rocUnaccT, col = "red")

par(new=TRUE)

plot(rocAccT, col = "green")

par(new=TRUE)

plot(rocGoodT, col = "blue")

par(new=TRUE)

plot(rocVgoodT, col = "grey")

legend("bottomright",pch = c(15,15), legend = c(paste("Unacc",auc(rocUnaccT)),paste("Acc",auc(rocAccT)),paste("Good",auc(rocGoodT)),paste("Vgood",auc(rocVgoodT))), col = c("red","green","blue","grey"), bty = "n")

test\_accuracy <- function(ilists){

a <- list()

for(ilist in ilists){

treecarF = rpart(shouldBuy~., data = cartrain, method = "class", control = rpart.control(minsplit = ilist))

predcarF = predict(treecarF, newdata = cartest, type = "class")

treecarFCM = table(cartest[["shouldBuy"]],predcarF)

b = list(sum(diag(treecarFCM)/sum(treecarFCM)))

a=c(a,b)

}

return(a)

}

minsplits = c(1:130)

test\_function <- test\_accuracy(minsplits)

test\_function

plot(minsplits,test\_accuracy(minsplits))

car5control = trainControl(method = "repeatedcv", number = 5, repeats = 3)

randomForest\_car5 = train(cartrain[,1:6], cartrain[["shouldBuy"]], method = "rf", metric = "Accuracy", trControl = car5control)

print(randomForest\_car5)

varImp(randomForest\_car5)

ggplot(varImp(randomForest\_car5))

car7control = trainControl(method = "repeatedcv", number = 7, repeats = 3)

randomForest\_car7 = train(cartrain[,1:6], cartrain[["shouldBuy"]], method = "rf", metric = "Accuracy", trControl = car7control)

print(randomForest\_car7)

varImp(randomForest\_car7)

ggplot(varImp(randomForest\_car7))

car10control = trainControl(method = "repeatedcv", number = 10, repeats = 3)

randomForest\_car10 = train(cartrain[,1:6], cartrain[["shouldBuy"]], method = "rf", metric = "Accuracy", trControl = car10control)

print(randomForest\_car10)

varImp(randomForest\_car10)

ggplot(varImp(randomForest\_car10))

car15control = trainControl(method = "repeatedcv", number = 15, repeats = 3)

randomForest\_car15 = train(cartrain[,1:6], cartrain[["shouldBuy"]], method = "rf", metric = "Accuracy", trControl = car15control)

print(randomForest\_car15)

varImp(randomForest\_car15)

ggplot(varImp(randomForest\_car15))

carTcontrol = trainControl(method = "repeatedcv", number = 10, repeats = 11)

randomForest\_carT = train(cartrain[,1:6], cartrain[["shouldBuy"]], method = "rf", metric = "Accuracy", trControl = carTcontrol)

print(randomForest\_carT)

varImp(randomForest\_carT)

ggplot(varImp(randomForest\_carT))

# Captions

[Figure 1 Car Data Structure 2](#_Toc10577663)

[Figure 2 Number Coding with Open Refine 3](#_Toc10577664)

[Figure 3 Minsplit Accuracy Table 3](#_Toc10577665)

[Figure 4 Plot of Decision Tree Minsplit = 1 4](#_Toc10577666)

[Figure 5 PLOT OF DECISION TREE MINSPLIT = 10 5](#_Toc10577667)

[Figure 6 PLOT OF DECISION TREE MINSPLIT = 50 5](#_Toc10577668)

[Figure 7 PLOT OF DECISION TREE MINSPLIT = 100 and 130 6](#_Toc10577669)

[Figure 8 Minsplit Accuracy Plot 7](#_Toc10577670)

[Figure 9 ROC & AUC When Minsplit = 1 Figure 10 ROC & AUC When Minsplit = 10 8](#_Toc10577671)

[Figure 11 ROC & AUC WHEN MINSPLIT = 50 Figure 12 ROC & AUC WHEN MINSPLIT = 75 8](#_Toc10577672)

[Figure 13 Attribute importance When K in 5, 7, 10, 15 and Iteration = 3 9](#_Toc10577673)

[Figure 14 Attribute Importance When K in 5, 7, 9, 11 with 10 Folds 9](#_Toc10577674)

[Figure 15 Attribute importance 10](#_Toc10577675)