MCDA5580 - Data and Text Mining

Assignment – 2

Classification with car data

Submitted By:
Nikhil Bhat (A00434789)
Mehar Singh (A00434701)
Samarth Gupta (A00433096)

Contents

1. Executive Summary

The analysis is carried out to study the acceptability of a car based on various parameters such as price, maintenance, doors, seats, safety and storage. The study was carried out on 1728 total observations to best classify the acceptability of a car to customers based on the following classes by creating models using Decision Tree and Random Forest classification algorithms:

- acc
- good
- unacc
- vgood

Based on the developed models, the Random Forest algorithm with 10-fold cross validation results in the best accuracy. The study and models are explained in detail in the report.

2. About the data

The given data consists of 1728 records of cars along with their features and customer's decisions to buy the car for each record. The data consists of features like "price", "maintenance", "doors", "seats", "storage" and "safety". The last column "shouldBuy" gives the customer's decision for each car.

The Summary of the given data is as follows:

```
  price      maintenance    doors       seats       storage      safety     shouldBuy
high :432   high :432   2    :432   2    :576   big   :576   high:576   acc  : 384
low  :432   low  :432   3    :432   4    :576   med   :576   low :576   good :  69
med  :432   med  :432   4    :432   more:576   small:576   med :576   unacc:1210
vhigh:432   vhigh:432   5more:432                                       vgood:  65
```

The below figure details the structure of input data:

```
'data.frame':   1728 obs. of  7 variables:
 $ price      : Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ maintenance: Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ doors      : Factor w/ 4 levels "2","3","4","5more": 1 1 1 1 1 1 1 1 1 1 ...
 $ seats      : Factor w/ 3 levels "2","4","more": 1 1 1 1 1 1 1 1 1 2 ...
 $ storage    : Factor w/ 3 levels "big","med","small": 3 3 3 2 2 2 1 1 1 3 ...
 $ safety     : Factor w/ 3 levels "high","low","med": 2 3 1 2 3 1 2 3 1 2 ...
 $ shouldBuy  : Factor w/ 4 levels "acc","good","unacc",..: 3 3 3 3 3 3 3 3 3 3 ...
```

As observed, there are in total 1728 observations with 7 columns or variables. The number of levels in each variable is also mentioned in the above figure. These variables can be split as dependent and independent variables as shown below.

2.1 Independent Variables:

S.No.	Variable Name	Levels
1.	price	high, low, med, vhigh
2.	maintenance	high, low, med, vhigh
3.	doors	2, 3, 4, 5more
4.	seats	2, 4, more
5.	storage	big, med, small
6.	safety	high, low, med

2.2 Dependent Variable:

S.No.	Variable Name	Levels
1.	shouldBuy	acc, good, unacc, vgood

2.3 Splitting Data

We have initially split the dataset in 80:20 ratio and assigned them into separate train and test datasets respectively.

For the purpose of better understanding and compatibility with syntax of randomForest method in R we have also defined and stored the independent and dependent variables in train dataset as “independent_train” and “dependent_train”. Similarly, in test dataset we have stored the independent and dependent variables in “independent_test” and “dependent_test”.

3. Decision Tree

To create decision tree in R we use the “rpart” library. Initially we try out different “minsplit” values for creating the decision trees and note the accuracy for each “minsplit” to decide the best suited “minsplit” value.

The table below gives the accuracy for the chosen “minsplit” values:

S.No.	Minsplit Value	Accuracy
1.	10	0.9421965
2.	50	0.9277457
3.	75	0.867052
4.	100	0.8583815

We choose minsplit=10 as it provides the highest accuracy.

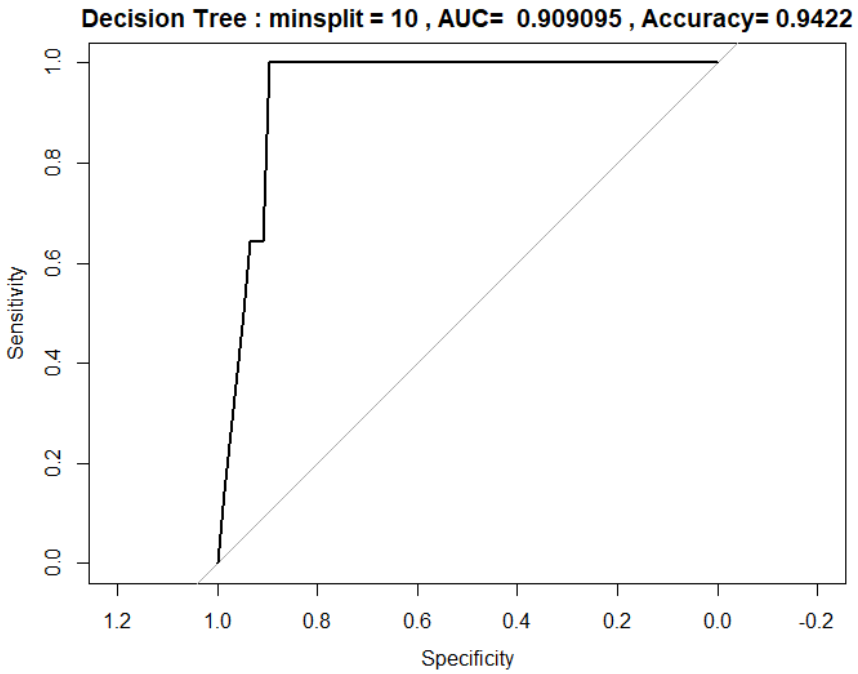
The confusion matrix for minsplit value 10 is as follows:

```
> treeCM
        predCar
         acc good unacc vgood
  acc     69    5     2     1
  good     0   12     0     2
  unacc    7    0   235     0
  vgood    3    0     0    10
```

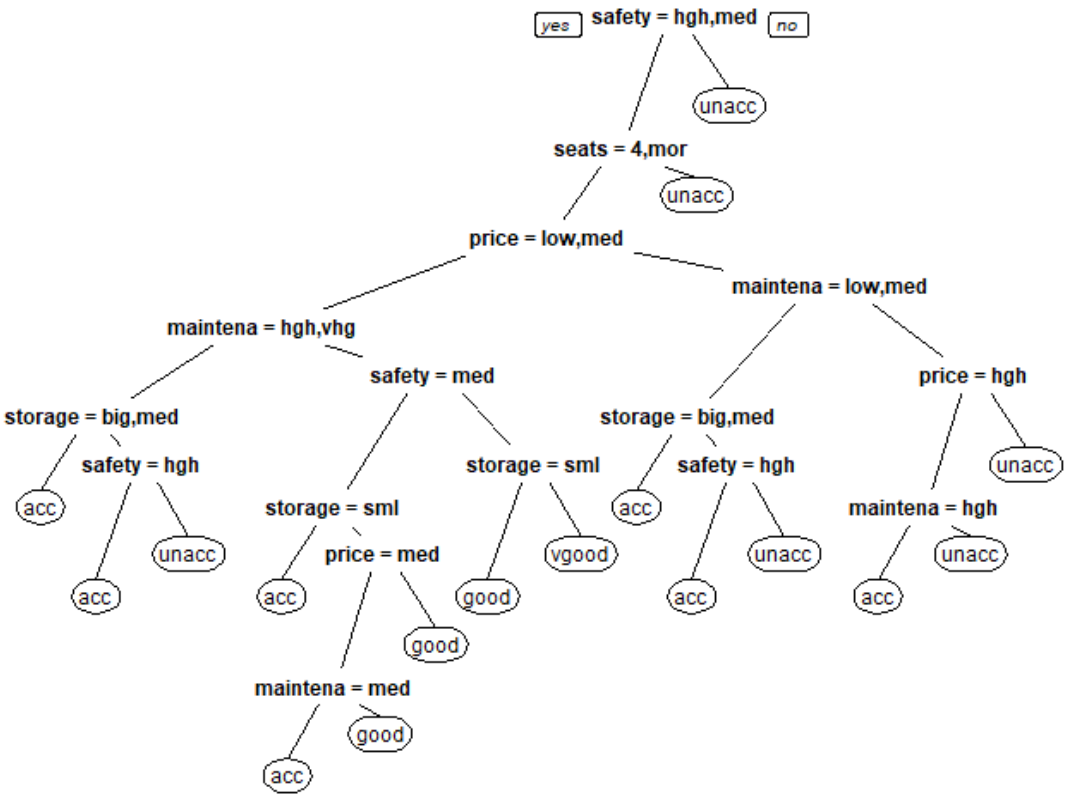
The accuracy is calculated by the sum of diagonal values by the sum of all values in the confusion matrix.

```
> #Finding Accuracy
> sum(diag(treeCM))/sum(treeCM)
[1] 0.9421965
```

On plotting the ROC curve and AUC values we get the below figure:



On plotting with minsplit=10, we get the following decision tree:



4. Random Forest

In case of the Random Forest Algorithm, the "randomForest" library in R builds a collection of decision trees and uses prediction results from all these trees to get a more accurate and stable prediction. For the purpose of deciding the best value for "ntree", we run the random forest algorithm on the training data set with ntree = 500, 100, 50, 10 and select the model based on the highest accuracy.

4.1 Number of trees/nodes = 500

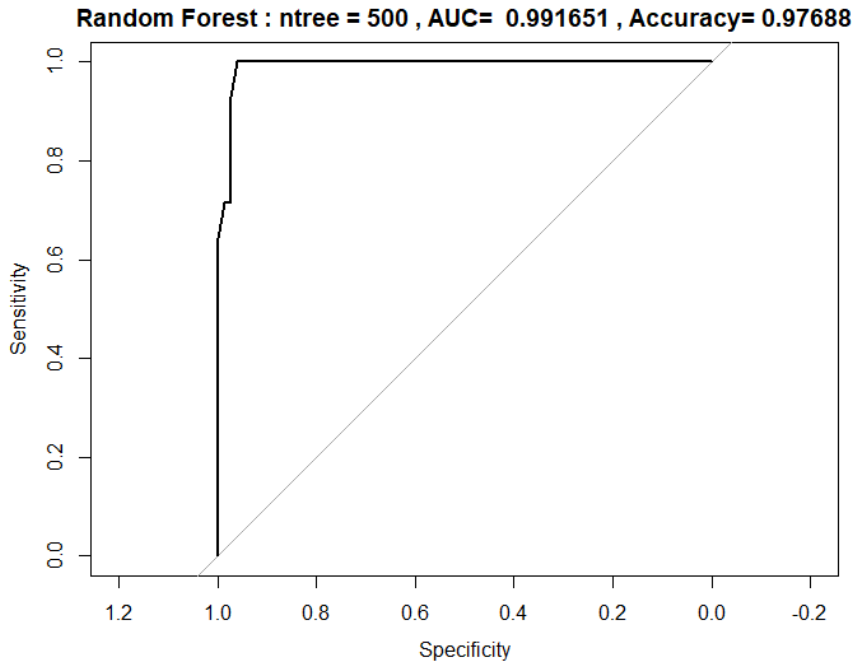
On applying the "randomForest" method in R with "ntree" value 500, we get the following confusion matrix:

```
> random_confM_500
                dependent_test
random_fp_500 acc good unacc vgood
        acc    74    0     2     2
        good    2   13     0     0
        unacc   1    0   240     0
        vgood   0    1     0    11
```

The accuracy calculated by dividing the sum of diagonal values by the sum of all values is as follows:

```
> #finding accuracy
> sum(diag(random_confM_500))/sum(random_confM_500)
[1] 0.9768786
```

On plotting the ROC curve and AUC values we get the following:



4.2 Number of trees/nodes = 100

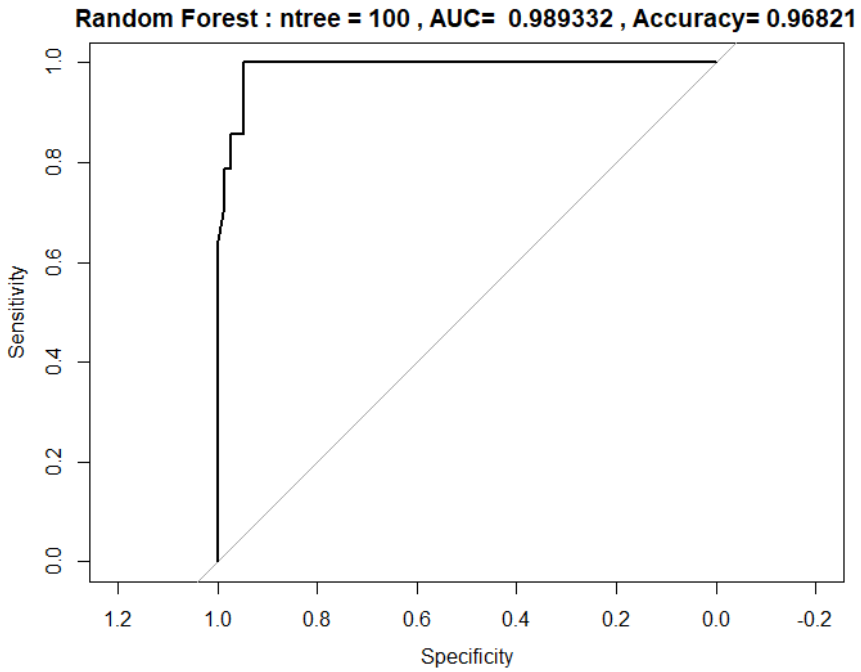
On applying the "randomForest" method in R with "ntree" value 10, we get the following confusion matrix:

```
> random_confM_100
                dependent_test
random_fp_100 acc good unacc vgood
        acc    72    2     2     1
        good    3   11     0     0
        unacc   2    0   240     0
        vgood   0    1     0    12
```

The accuracy calculated by dividing the sum of diagonal values by the sum of all values is as follows:

```
> #finding accuracy
> sum(diag(random_confM_100))/sum(random_confM_100)
[1] 0.9682081
```

On plotting the ROC curve and AUC values we get the following:



4.2 Number of trees/nodes = 50

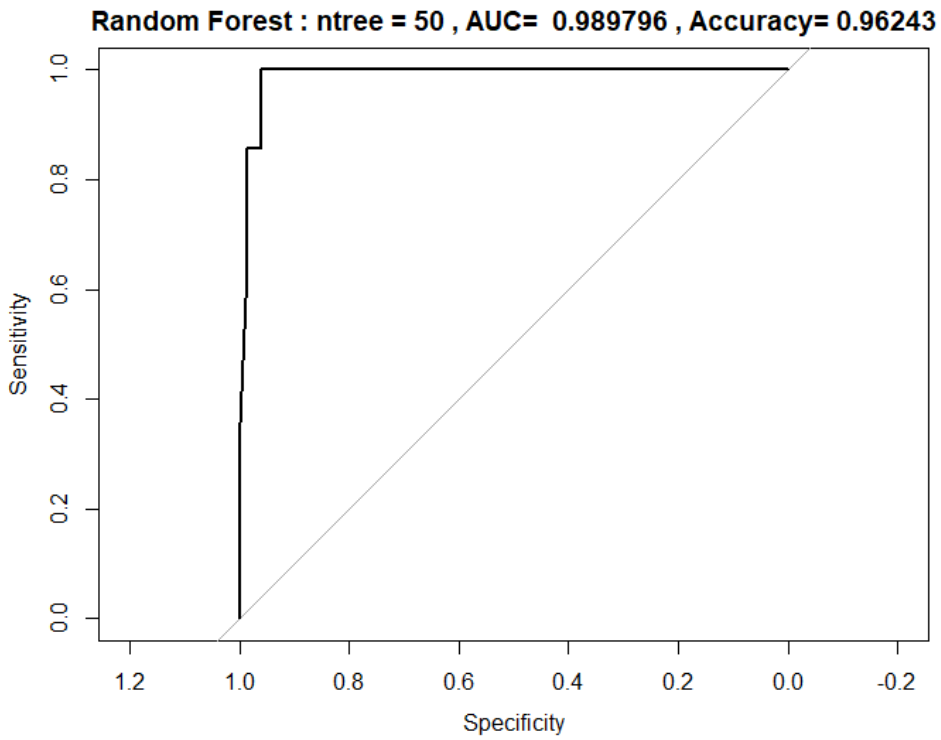
On applying the "randomForest" method in R with "ntree" value 50, we get the following confusion matrix:

```
> random_confM_50
                dependent_test
random_fp_50 acc good unacc vgood
       acc    73    1     3     2
       good    2   12     0     2
       unacc   2    0   239     0
       vgood   0    1     0     9
```

The accuracy calculated by dividing the sum of diagonal values by the sum of all values is as follows:

```
> #finding accuracy
> sum(diag(random_confM_50))/sum(random_confM_50)
[1] 0.9624277
```

On plotting the ROC curve and AUC values we get the following:



4.4 Number of trees/nodes = 10

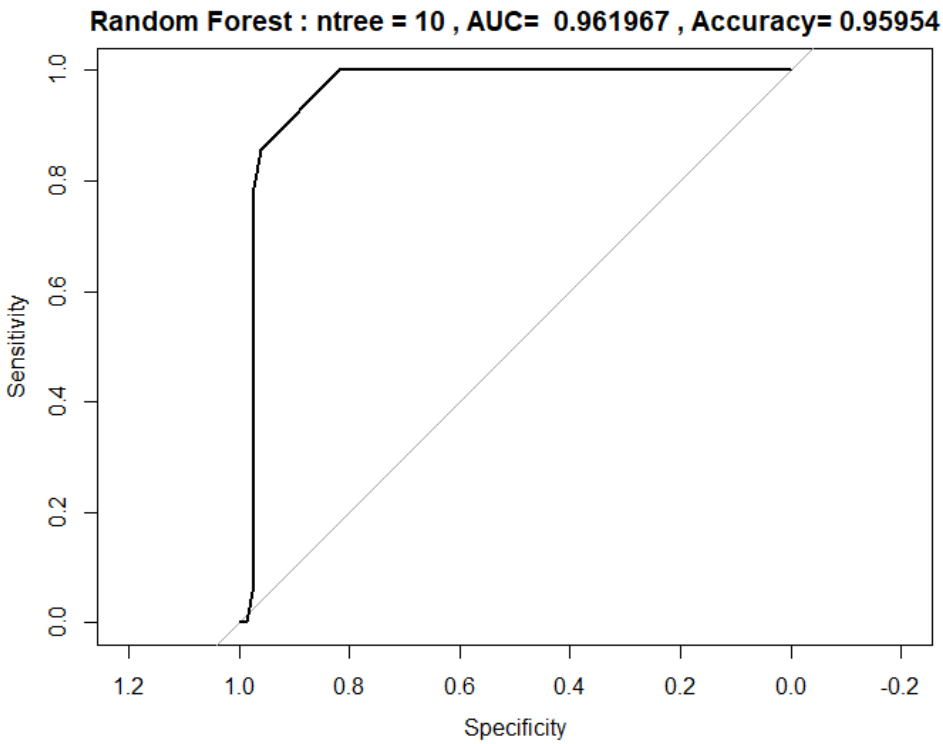
On applying the "randomForest" method in R with "ntree" value 10, we get the following confusion matrix:

```
> random_confM_10
             dependent_test
random_fp_10 acc good unacc vgood
       acc    73    1     5     2
       good    2   11     0     0
       unacc   2    0   237     0
       vgood   0    2     0    11
```

The accuracy calculated by dividing the sum of diagonal values by the sum of all values is as follows:

```
> #finding accuracy
> sum(diag(random_confM_10))/sum(random_confM_10)
[1] 0.9595376
```

On plotting the ROC curve and AUC values we get the following:



It can be concluded from the above observations that higher the value of "ntree" higher is the accuracy and AUC obtained. Thus, random forest with "ntree" value of 500 is the best model for prediction.

5. Data Analysis – Random Forest with K-fold cross validation

The K-fold cross validation process is a re-sampling procedure used to evaluate the ability of a model to correctly classify/predict the dependent variables on multiple test sets. For the purpose of this analysis K=10 is used for splitting the entire dataset into 10 folds or test/train splits randomly and repeats=3 is used to denote the number of iterations. These folds are created in such a way that each data point occurs in exactly one test set resulting in a test set approximately the same size as the training set. For the k-fold cross validation, the entire dataset [1] [2] is used instead of splitting it into training and test data. By doing cross-validation, we can use all 1728 observations for both training and testing while evaluating the algorithm on observations it has never seen before.

Since, the highest accuracy was observed for Random Forest model with ntree=500, we have used the same algorithm and ntree value for cross validation as well.

The below figure shows the confusion matrix for 10-fold cross validation:

```
> confusionMatrix(randomForest_default)
Cross-Validated (10 fold, repeated 3 times) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction  acc good unacc vgood
     acc   21.5  0.1   0.5   0.0
     good   0.3  3.8   0.1   0.0
     unacc  0.2  0.0  69.4   0.0
     vgood  0.1  0.0   0.0   3.7

 Accuracy (average) : 0.985
```

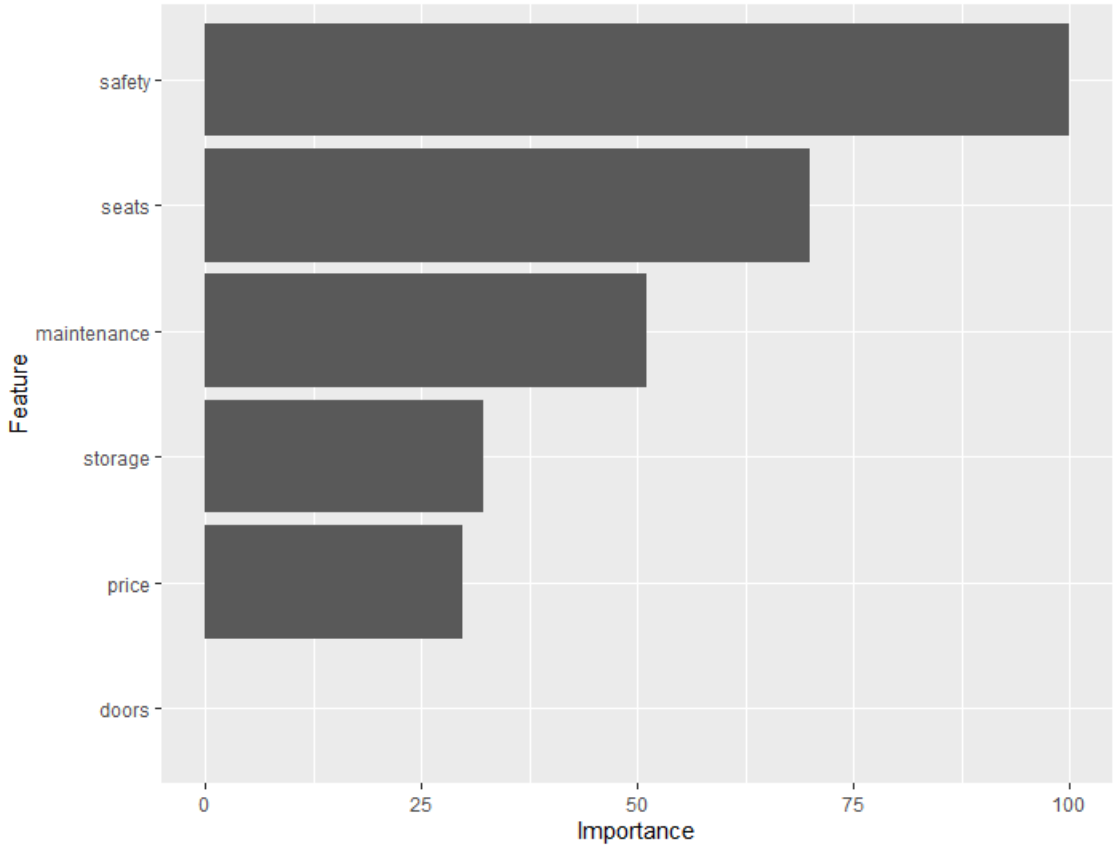
The model achieved an accuracy of 98.5% which is higher than the accuracy of the models developed in previous sections. The figure below showcases the accuracy and kappa values for the analysis:

```
> print(randomForest_default)
Random Forest

1728 samples
   6 predictor
   4 classes: 'acc', 'good', 'unacc', 'vgood'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1555, 1556, 1556, 1555, 1554, 1556, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9716389  0.9389310
  4     0.9820572  0.9612475
  6     0.9849541  0.9673558

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 6.
```

The below figures signify the importance of each independent variable in determining the in dependent variable obtained from the 10-fold cross validation analysis.

```
> #Print important variables
> varImp(randomForest_default)
rf variable importance

            Overall
safety       100.00
seats         70.04
maintenance   51.05
storage       32.29
price         29.80
doors          0.00
```

The below bar chart displays the same information as above in order to better visually signify the importance of each feature in determination of the acceptability of a car.



6. Conclusion

The below table summarizes the classification accuracy for the created models:

S.No.	Algorithm	Control Parameter	Accuracy
1.	Decision Tree	minsplit = 10	0.9421965
		minsplit = 50	0.9277457
		minsplit = 75	0.867052
		minsplit = 100	0.8583815
2.	Random Forest	ntree = 500	0.9768786
		ntree = 100	0.9682081
		ntree = 50	0.9624277
		ntree = 10	0.9595376
3.	Random Forest with K-fold cross validation	ntree=500, K=10, repeats = 3	0.9849541

Thus, from the above table it can be concluded that Random Forest with 10-fold cross validation is the best classifier model generator.

Appendix A (References)

[1] DataCamp, "R tutorial: Cross-validation," [Online]. Available: https://www.youtube.com/watch?v=OwPQHmiJURI.

[2] D. Shulga, "5 Reasons why you should use Cross-Validation in your Data Science Projects," [Online]. Available: https://towardsdatascience.com/5-reasons-why-you-should-use-cross-validation-in-your-data-science-project-8163311a1e79.

Appendix B (R Script)

```
#Getting car Data
carData=read.csv("car_data.csv")
summary(carData)
str(carData)

library(caTools)

set.seed(8008)

#Splitting Cardata in 80:20 ratio for training and testing purposes
split= sample.split(carData$shouldBuy, SplitRatio = 0.8)
trainCarData= subset(carData, split == TRUE)
testCarData= subset(carData, split == FALSE)

#viewing dependent and independent variables in train data
independent_train=trainCarData[,1:6]
dependent_train=trainCarData[,7]

#viewing dependent and independent variables in test data
independent_test=testCarData[,1:6]
dependent_test=testCarData[,7]

#-------------------------------------------------------------------------------------------
#install.packages("rpart")
#install.packages("pROC")
#install.packages("rpart.plot")
library(rpart)
library(pROC)
library(rpart.plot)

#Forming Car Decision Tree with train data

#-------------------------------------------------------------------------------------------
#minsplit = 10
carTree = rpart(shouldBuy~.,data=trainCarData,method="class",control=rpart.control(minsplit=10))
carTree
#Checking Test Data on Car Tree
predCar=predict(carTree,newdata=testCarData[,-7],type="class")
#creating confusion matrix
treeCM=table(testCarData[,7],predCar)
treeCM
#Manually Cross-Checking actual dependent variables
summary(dependent_test)
#Finding Accuracy
sum(diag(treeCM))/sum(treeCM)
```

```
#-------------------------------------------------------------------------------------------
#minsplit = 50
carTree = rpart(shouldBuy~.,data=trainCarData,method="class",control=rpart.control(minsplit=50))
carTree
#Checking Test Data on Car Tree
predCar=predict(carTree,newdata=testCarData[,-7],type="class")
#creating confusion matrix
treeCM=table(testCarData[,7],predCar)
treeCM
#Manually Cross-Checking actual dependent variables
summary(dependent_test)
#Finding Accuracy
sum(diag(treeCM))/sum(treeCM)

#-------------------------------------------------------------------------------------------
#minsplit = 75
carTree = rpart(shouldBuy~.,data=trainCarData,method="class",control=rpart.control(minsplit=75))
carTree
#Checking Test Data on Car Tree
predCar=predict(carTree,newdata=testCarData[,-7],type="class")
#creating confusion matrix
treeCM=table(testCarData[,7],predCar)
treeCM
#Manually Cross-Checking actual dependent variables
summary(dependent_test)
#Finding Accuracy
sum(diag(treeCM))/sum(treeCM)

#-------------------------------------------------------------------------------------------
#minsplit = 100
carTree = rpart(shouldBuy~.,data=trainCarData,method="class",control=rpart.control(minsplit=100))
carTree
#Checking Test Data on Car Tree
predCar=predict(carTree,newdata=testCarData[,-7],type="class")
#creating confusion matrix
treeCM=table(testCarData[,7],predCar)
treeCM
#Manually Cross-Checking actual dependent variables
summary(dependent_test)
#Finding Accuracy
sum(diag(treeCM))/sum(treeCM)

#-------------------------------------------------------------------------------------------
#minsplit = 10
carTree = rpart(shouldBuy~.,data=trainCarData,method="class",control=rpart.control(minsplit=10))
#Checking Test Data on Car Tree
predCar=predict(carTree,newdata=testCarData[,-7],type="class")
#creating confusion matrix
treeCM=table(testCarData[,7],predCar)
```

```
#Finding Accuracy
accuracy <- sum(diag(treeCM))/sum(treeCM)


#Plotting Car Decision Tree
rpart.plot(carTree)
prp(carTree)

predCar=predict(carTree,newdata=testCarData[,-7],type="prob")
#
r1 <- multiclass.roc(dependent_test,predCar[,1])
auc1 <- auc(r1)
print(auc1)
#Plotting ROC
plot(roc(dependent_test,predCar[,1]))

x1<-paste("Decision Tree : minsplit = 10",",","AUC= ",round(auc1,6),",","Accuracy=",round(accuracy,5))
plot(roc(dependent_test,predCar[,1]),main=x1)

#-------------------------------------------------------------------------------------------
#install.packages('randomForest')

#Random forest for n=500
library(randomForest)
#creating random forest
random_f_500=randomForest(independent_train, dependent_train, ntree=500)
#predicting test data by passing through random forest
random_fp_500=predict(random_f_500,independent_test)
#combining to check predicted values with dependent variables in test data
checkData=cbind(random_fp_500, dependent_test)
checkData
#creating confusion matrix
random_confM_500=table(random_fp_500,dependent_test)
random_confM_500
#finding accuracy
sum(diag(random_confM_500))/sum(random_confM_500)
accuracy_500<-sum(diag(random_confM_500))/sum(random_confM_500)
#plotting the random forest
rfProb_500=predict(random_f_500,independent_test,type="prob")
r_500<- roc(dependent_test,rfProb_500[,1])
auc_500<-auc(r_500)
x_500<-paste("Random Forest : ntree = 500",",","AUC=
",round(auc_500,6),",","Accuracy=",round(accuracy_500,5))
plot(r_500,main=x_500)


#-------------------------------------------------------------------------------------------
#Random forest for n=100
#creating random forest
random_f_100=randomForest(independent_train, dependent_train, ntree=100)
```

```
#predicting test data by passing through random forest
random_fp_100=predict(random_f_100,independent_test)
#combining to check predicted values with dependent variables in test data
checkData=cbind(random_fp_100, dependent_test)
checkData
#creating confusion matrix
random_confM_100=table(random_fp_100,dependent_test)
random_confM_100
#finding accuracy
sum(diag(random_confM_100))/sum(random_confM_100)
accuracy_100<-sum(diag(random_confM_100))/sum(random_confM_100)
#plotting the random forest
rfProb_100=predict(random_f_100,independent_test,type="prob")
r_100<- roc(dependent_test,rfProb_100[,1])
auc_100<-auc(r_100)
x_100<-paste("Random Forest : ntree = 100",",","AUC=
",round(auc_100,6),",","Accuracy=",round(accuracy_100,5))
plot(r_100,main=x_100)

#-----------------------------------------------------------------------------------------
#Random forest for n=50
#creating random forest
random_f_50=randomForest(independent_train, dependent_train, ntree=50)
#predicting test data by passing through random forest
random_fp_50=predict(random_f_50,independent_test)
#combining to check predicted values with dependent variables in test data
checkData=cbind(random_fp_50, dependent_test)
checkData
#creating confusion matrix
random_confM_50=table(random_fp_50,dependent_test)
random_confM_50
#finding accuracy
sum(diag(random_confM_50))/sum(random_confM_50)
accuracy_50<-sum(diag(random_confM_50))/sum(random_confM_50)
#plotting the random forest
rfProb_50=predict(random_f_50,independent_test,type="prob")
r_50<- roc(dependent_test,rfProb_50[,1])
auc_50<-auc(r_50)
x_50<-paste("Random Forest : ntree = 50",",","AUC=
",round(auc_50,6),",","Accuracy=",round(accuracy_50,5))
plot(r_50,main=x_50)

#-----------------------------------------------------------------------------------------
#Random forest for n=10
#creating random forest
random_f_10=randomForest(independent_train, dependent_train, ntree=10)
#predicting test data by passing through random forest
random_fp_10=predict(random_f_10,independent_test)
#combining to check predicted values with dependent variables in test data
checkData=cbind(random_fp_10, dependent_test)
```

```
checkData
#creating confusion matrix
random_confM_10=table(random_fp_10,dependent_test)
random_confM_10
#finding accuracy
sum(diag(random_confM_10))/sum(random_confM_10)
accuracy_10<-sum(diag(random_confM_10))/sum(random_confM_10)
#plotting the random forest
rfProb_10=predict(random_f_10,independent_test,type="prob")
r_10<- roc(dependent_test,rfProb_10[,1])
auc_10<-auc(r_10)
x_10<-paste("Random Forest : ntree = 10",",","AUC=
",round(auc_10,6),",","Accuracy=",round(accuracy_10,5))
plot(r_10,main=x_10)

#-----------------------------------------------------------------------------------------------
#Random Forest K-fold cross validation
#install.packages("caret")
#install.packages("e1071")
library(caret)
library(e1071)

#Setup control for cross validation with 10 folds and 3 iterations
ctrl = trainControl(method="repeatedcv", number = 10, repeats = 3)
#Using random forest method and k-fold cross validation with Accuracy as metric and ntree = 500
randomForest_default = train(carData[,1:6], carData[,7],method = "rf",ntree=500,metric = "Accuracy",
trControl = ctrl)
confusionMatrix(randomForest_default)
print(randomForest_default)
#Print important variables
varImp(randomForest_default)
#Plot important variables
ggplot(varImp(randomForest_default))
```