SAINT MARY'S UNIVERSITY SINCE 1802

One University. One World. Yours.

Assignment 2: Classification with car data

PREPARED BY

ABHIJITH SANTHOSH JAYA: A00435857
ARUN RAJ JAVAGAL LAKKASHETTY: A00435847
VIDHYADHARA TEJA KATRAGADDA: A00433626
BIJAY KHADKA: A00437343

Table of Contents

EXECUTIVE SUMMARY

The purpose of the report is to highlight the findings of our analysis performed on the car data set. We were required to predict the acceptability of cars based on the given data. As part of the stipulated requirement for this task, we have employed classification as the primary data mining technique for our analysis. The key factors we took into our prediction model are price, maintenance, doors, seats, storage, safety. The model thus developed hope to serve the business owner to make better decisions.

TOOLS USED

R-STUDIO: For tuning the parameters to predict the acceptability of a car, we can make use of different classification algorithms using R language.

DATASET PREPERATION

We are provided with "Car.csv" dataset and we are considering all the columns to predict an acceptability of car. The details of the given different columns of the dataset can be found below.

price	maintenance	doors	seats	storage	safety	shouldBuy
vhigh	vhigh	2	2	small	low	unacc
vhigh	vhigh	2	2	small	med	unacc
vhigh	vhigh	2	2	small	high	unacc
vhigh	vhigh	2	2	med	low	unacc
vhigh	vhigh	2	2	med	med	unacc
vhigh	vhigh	2	2	med	high	unacc
vhigh	vhigh	2	2	big	low	unacc
vhigh	vhigh	2	2	big	med	unacc
vhigh	vhigh	2	2	big	high	unacc
vhigh	vhigh	2	4	small	low	unacc
vhigh	vhigh	2	4	small	med	unacc
vhigh	vhigh	2	4	small	high	unacc
vhigh	vhigh	2	4	med	low	unacc
vhigh	vhigh	2	4	med	med	unacc

Fig: Column of "Car.csv" dataset

Each of the 7 columns corresponds to a variable that is taken as input to our analysis. The primary variable is "shouldBuy". The variable "shouldBuy" comprises of 4 categorical variables acc, good, unacc, vgood

The predictor variables are Safety, Seats, Price, Maintenance, Doors, Storage. A brief description of theses variable can be found below:

Price: Price has categorical values: high, low, med, vhigh
Maintenance: Maintenance has categorical values: high, low, med, vhigh
Doors: Doors is having mixture of numeric and categorical values: 2, 3, 4, 5more

Safety: Safety has categorical values high, low, med.
Seats: Safety is having mixture of numeric and categorical values: 2,4, more

Storage: Storage has categorical values big, med, small.

The data set has been loaded into the R environment for analysis using the command below:

- CAR_DATA = read.table("C:/Users/arunr/Desktop/MCDA/data mining/data_mining_assignments/data_mining_assignments/data_mining_assignment2/car.csv ", sep=',', header=T)

ANALYSIS

In order to analyse the data and to train our predictive model, we have used decision trees and random forest.

DECISION TREE(RPART)

We have used rpart library[3] in R to generate the segment of the decision tree. We have split the CAR_DATA into training data and test data. We have taken 2/3 of data as training data and the remaining 1/3rd of data as test data. The training data is given as input to rpart function.

We found that for the given dataset, rpart is giving good accuracy for minsplit = 10. The accuracy is found to decrease for values of minsplit over 50. It is remaining the same for 10 < minsplit <50. Hence, we have taken the minimum value of 10 as minsplit.

```
       predcar
        acc good unacc vgood
  acc   119    7     9     2
  good    0   19     0     3
  unacc  11    2   394     0
  vgood   0    0     0    18

> sum(diag(treeCM)/sum(treeCM))
[1] 0.9417808
```

Fig: Accuracy of 0.94178 for minsplit = 10

Now, we are plotting the decision of the tree for minsplit = 10 using the command below.

- car_DTree = rpart(formula=shouldBuy~., data=CAR_DATA, method="class", control=rpart.control(minsplit=10))

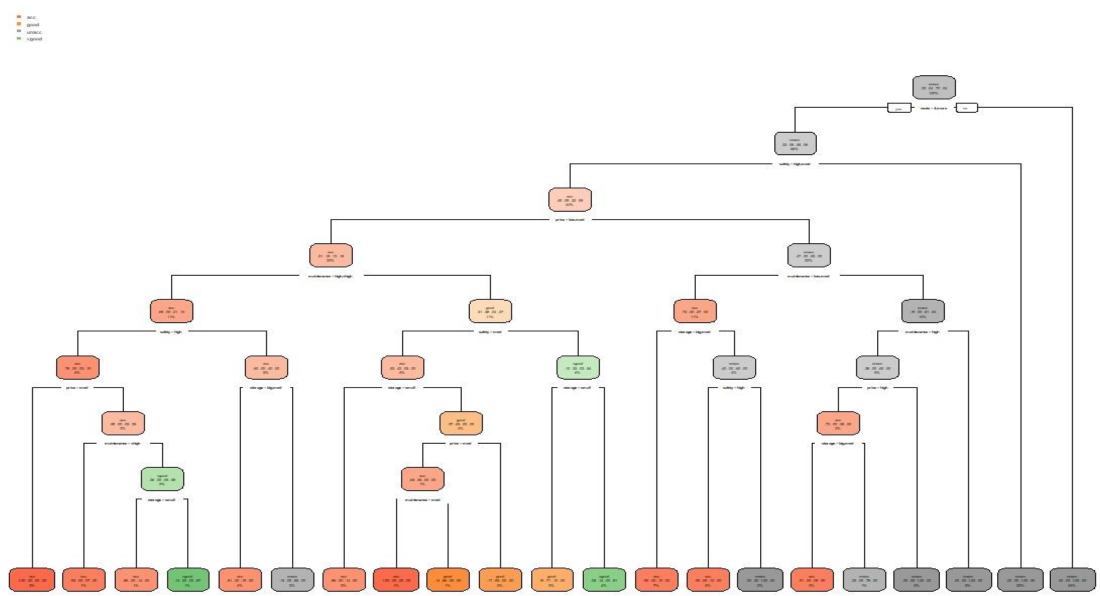


Fig: Decision tree graph

List of some of the rules:

```
> rpart.rules(carTree)
 shouldBuy   acc good unac vgoo
      acc [ .81  .00  .19  .00] when seats is 4 or more & safety is        med & price is    low or med & maintenance is high or vhigh & storage is big or med
      acc [ .86  .00  .14  .00] when seats is 4 or more & safety is       high & price is          low & maintenance is        high & storage is      small
      acc [ .86  .00  .14  .00] when seats is 4 or more & safety is        med & price is    low or med & maintenance is    low or med & storage is      small
      acc [ .90  .00  .10  .00] when seats is 4 or more & safety is       high & price is high or vhigh & maintenance is    low or med & storage is      small
      acc [ .90  .00  .10  .00] when seats is 4 or more & safety is high or med & price is high or vhigh & maintenance is    low or med & storage is big or med
      acc [ .91  .00  .09  .00] when seats is 4 or more & safety is high or med & price is         high & maintenance is        high & storage is big or med
      acc [ .93  .00  .07  .00] when seats is 4 or more & safety is       high & price is          low & maintenance is       vhigh
      acc [1.00  .00  .00  .00] when seats is 4 or more & safety is       high & price is          med & maintenance is high or vhigh
      acc [1.00  .00  .00  .00] when seats is 4 or more & safety is        med & price is          med & maintenance is         med & storage is big or med
     good [ .19  .71  .10  .00] when seats is 4 or more & safety is       high & price is    low or med & maintenance is    low or med & storage is      small
     good [ .17  .83  .00  .00] when seats is 4 or more & safety is        med & price is          low & maintenance is    low or med & storage is big or med
     good [ .14  .86  .00  .00] when seats is 4 or more & safety is        med & price is          med & maintenance is          low & storage is big or med
    unacc [ .22  .00  .78  .00] when seats is 4 or more & safety is high or med & price is         high & maintenance is        high & storage is      small
    unacc [ .15  .00  .85  .00] when seats is 4 or more & safety is        med & price is    low or med & maintenance is high or vhigh & storage is      small
    unacc [ .00  .00 1.00  .00] when seats is 4 or more & safety is        med & price is high or vhigh & maintenance is    low or med & storage is      small
    unacc [ .00  .00 1.00  .00] when seats is 4 or more & safety is high or med & price is        vhigh & maintenance is        high
    unacc [ .00  .00 1.00  .00] when seats is 4 or more & safety is high or med & price is high or vhigh & maintenance is       vhigh
    unacc [ .00  .00 1.00  .00] when seats is 4 or more & safety is        low
    unacc [ .00  .00 1.00  .00] when seats is        2
    vgood [ .05  .14  .00  .81] when seats is 4 or more & safety is       high & price is    low or med & maintenance is    low or med & storage is big or med
    vgood [ .13  .00  .00  .87] when seats is 4 or more & safety is       high & price is          low & maintenance is        high & storage is big or med
```

Making predictions using the below R command:

predBuy = predict(car_DTree,newdata=carData,type="class")

CONFUSION MATRIX

We have plotted the confusion matrix to get the accuracy of our model. The test data was given as input to validate the accuracy of the mode. The confusion matrix data can be found below.

```
> confusionMatrix(predictCars,CAR.test$shouldBuy)
Confusion Matrix and Statistics

          Reference
Prediction acc good unacc vgood
     acc   131    0     6     1
     good    3   21     2     0
     unacc   2    0   399     0
     vgood   1    1     0    17

Overall Statistics

               Accuracy : 0.9726
                 95% CI : (0.9559, 0.9843)
    No Information Rate : 0.6969
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9409

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: acc Class: good Class: unacc Class: vgood
Sensitivity              0.9562     0.95455       0.9803      0.94444
Specificity              0.9843     0.99110       0.9887      0.99647
Pos Pred Value           0.9493     0.80769       0.9950      0.89474
Neg Pred Value           0.9865     0.99821       0.9563      0.99823
Prevalence               0.2346     0.03767       0.6969      0.03082
Detection Rate           0.2243     0.03596       0.6832      0.02911
Detection Prevalence     0.2363     0.04452       0.6866      0.03253
Balanced Accuracy        0.9703     0.97282       0.9845      0.97046
```

Fig: Confusion matrix and accuracy decision tree

We decided to obtain the importance of variables to get the conclusion. We calculated important variables contributing to targeted variable “shouldBuy”. We got result as below:

```
> varImp(carTree1)
              Overall
doors        32.83426
maintenance 139.82254
price       112.05515
safety      293.72942
seats        86.18631
storage     145.76243
```

Fig: Important variables information for decision tree

It is very clear that ‘safety’ tops all the variables taken to form the decision tree. Hence, it is the most important variable

ESTIMATING QUALITY OF OUR CLASSIFIER USING ROC AND AUC

We are plotting the ROC[3] curves for the 4 classes which we will use to find the AUC. AUC helps to estimate the accuracy of our classifier. Typically, if the AUC value lies between 0.5 to 1, where 0.5 denotes a bad classifier and 1 denotes an excellent classifier. Furthermore, the sensitivity and specificity indicate the following.

- True positive rate = Sensitivity = Recall
- False positive rate = Specificity

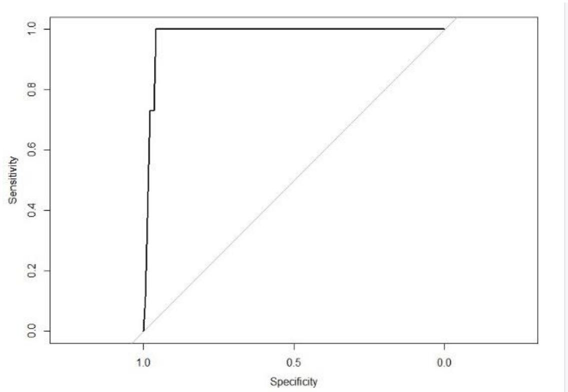


Fig: Roc for Class 'acc'

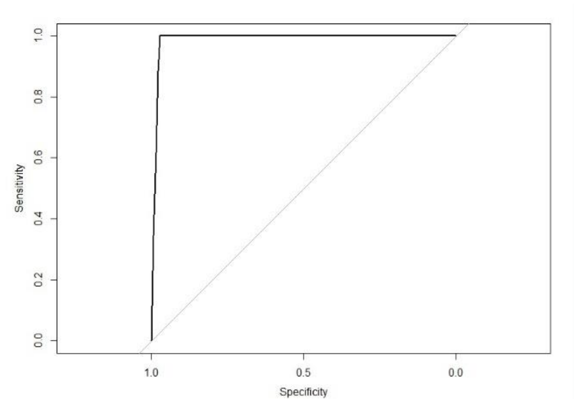


Fig: Roc for Class 'good'

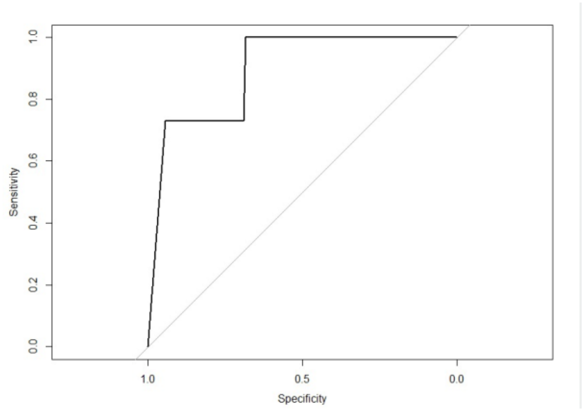


Fig: Roc for Class 'unacc'

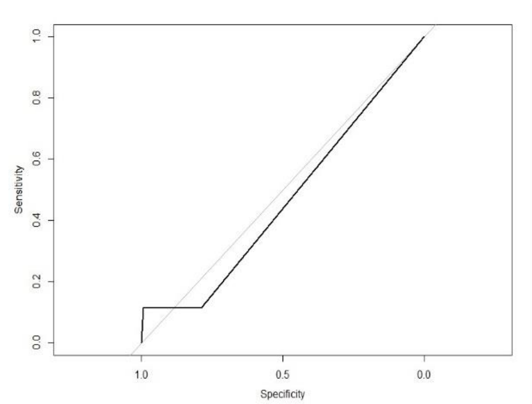


Fig: Roc for Class 'vgood'

RANDOM FOREST

"RandomForest" library [2] has been used for creating the model. We used the car dataset for the targeted variable "shouldBuy" to obtain the model. We used three different ntree's to get confusion matrix, ntree = 50 gave the more accuracy compare to ntree = 100 and ntree = 500. We predicted using model and the data. The confusion matrix for ntree = 50 is as below:

CONFUSION MATRIX

We have plotted the confusion matrix to get the accuracy of our model. The test data was given as input to validate the accuracy of the mode. The confusion matrix data can be found below.

```
> confusionMatrix(predictCars,CAR.test$shouldBuy)
Confusion Matrix and Statistics

          Reference
Prediction acc good unacc vgood
     acc   133    0     4     1
     good    3   21     2     0
     unacc   1    0   401     0
     vgood   0    1     0    17

Overall Statistics

               Accuracy : 0.9795
                 95% CI : (0.9644, 0.9893)
    No Information Rate : 0.6969
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9555

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: acc Class: good Class: unacc Class: vgood
Sensitivity              0.9708     0.95455       0.9853      0.94444
Specificity              0.9888     0.99110       0.9944      0.99823
Pos Pred Value           0.9638     0.80769       0.9975      0.94444
Neg Pred Value           0.9910     0.99821       0.9670      0.99823
Prevalence               0.2346     0.03767       0.6969      0.03082
Detection Rate           0.2277     0.03596       0.6866      0.02911
Detection Prevalence     0.2363     0.04452       0.6884      0.03082
Balanced Accuracy        0.9798     0.97282       0.9898      0.97134
>
```

Fig: Confusion matrix for ntree = 50

```
> confusionMatrix(predictCars,CAR.test$shouldBuy)
Confusion Matrix and Statistics

          Reference
Prediction acc good unacc vgood
     acc   131    0     6     1
     good    3   21     2     0
     unacc   2    0   399     0
     vgood   1    1     0    17

Overall Statistics

               Accuracy : 0.9726
                 95% CI : (0.9559, 0.9843)
    No Information Rate : 0.6969
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9409

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: acc Class: good Class: unacc Class: vgood
Sensitivity              0.9562     0.95455       0.9803      0.94444
Specificity              0.9843     0.99110       0.9887      0.99647
Pos Pred Value           0.9493     0.80769       0.9950      0.89474
Neg Pred Value           0.9865     0.99821       0.9563      0.99823
Prevalence               0.2346     0.03767       0.6969      0.03082
Detection Rate           0.2243     0.03596       0.6832      0.02911
Detection Prevalence     0.2363     0.04452       0.6866      0.03253
Balanced Accuracy        0.9703     0.97282       0.9845      0.97046
>
```

Fig: Confusion matrix for ntree = 100

```
Confusion Matrix and Statistics

          Reference
Prediction acc good unacc vgood
     acc   132    0     4     1
     good    3   21     2     0
     unacc   1    0   401     0
     vgood   1    1     0    17

Overall Statistics

               Accuracy : 0.9777
                 95% CI : (0.9622, 0.9881)
    No Information Rate : 0.6969
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9519

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: acc Class: good Class: unacc Class: vgood
Sensitivity              0.9635     0.95455       0.9853      0.94444
Specificity              0.9888     0.99110       0.9944      0.99647
Pos Pred Value           0.9635     0.80769       0.9975      0.89474
Neg Pred Value           0.9888     0.99821       0.9670      0.99823
Prevalence               0.2346     0.03767       0.6969      0.03082
Detection Rate           0.2260     0.03596       0.6866      0.02911
Detection Prevalence     0.2346     0.04452       0.6884      0.03253
Balanced Accuracy        0.9762     0.97282       0.9898      0.97046
>
```

Fig: Confusion matrix for ntree = 500

We decided to obtain the importance of variables to get the conclusion. We calculated important variables contributing to targeted variable “shouldBuy”. We got result as below:

```
> varImp(rf)
               Overall
price         73.57622
maintenance   82.75970
doors         31.46479
seats        115.57676
storage       62.31788
safety       151.94355
```

Fig: Important variables information for decision tree

It is very clear that ‘safety’ and ‘seats’ top all the variables taken to form the decision tree. Hence, it is the most important variable.

We have plotted the ROC curves to identify the AUC. AUC has been generated for the 4 classes of targeted variable (1: acc, 2:good, 3:unacc, 4:vgood) and plotted ROC graphs are as shown below.

```
predictCars = predict(rf, newdata= CAR.test, type = "prob")
buyacc <- roc(CAR.test$shouldBuy, predictCars[,1])
auc(buyacc)
plot(buyacc)
buygood <- roc(CAR.test$shouldBuy, predictCars[,2])
auc(buygood)
plot(buygood)
buyunacc <- roc(CAR.test$shouldBuy, predictCars[,3])
auc(buyunacc)
plot(buyunacc)
buyvgood <- roc(CAR.test$shouldBuy, predictCars[,4])
auc(buyvgood)
plot(buyvgood)
```

Fig: R command to plot AUC and ROC

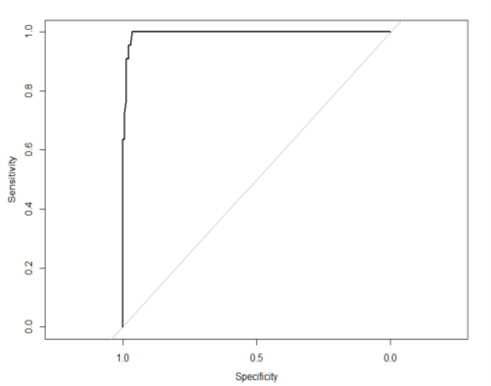


Fig: Roc for Class ‘acc’

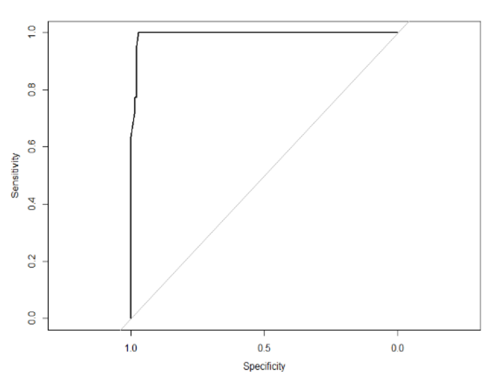
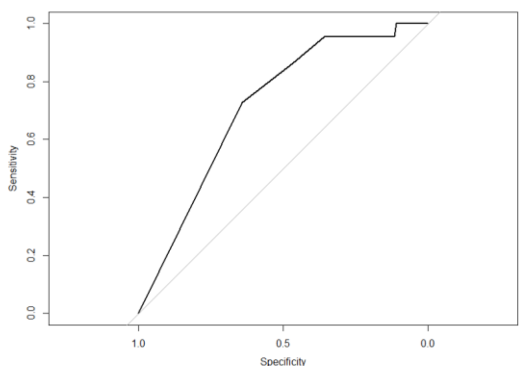


Fig: Roc for Class ‘good’



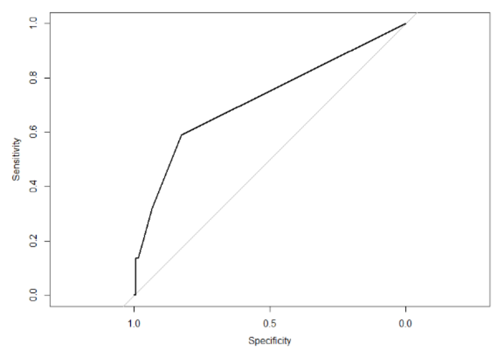


Fig: Roc for Class 'unacc' *Fig: Roc for Class 'vgood'*

Based on the above ROC's, the corresponding AUC's has been calculated from random forest method. AUCs for 'acc' and 'good' is above 95% which means our model is able to predict more accurately when compared to other methods.

K – FOLD CROSS VALIDATION

We validated model using K-fold cross validation [4][1], we performed validation on random forest model.

- Number of folds = 5 and repetition = 3 times. The values of accuracy and kappa are as follows:

```
> print(randomForest_default)
Random Forest

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

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 1382, 1382, 1383, 1382, 1383, 1383, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9681679  0.9309646
  4     0.9778118  0.9520051
  6     0.9822490  0.9613955

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

Fig - K-fold values for n=5, rep =3

The important variables that are contributing towards acceptability of car are shown below:

```
> varImp(randomForest_default)
rf variable importance

            Overall
safety       100.00
seats         73.31
maintenance   52.43
storage       33.27
price         30.83
doors          0.00
```

Fig - Important variables after k-fold n=5 rep=3

From the above Fig, safety and seats are the important variables and below is the graph for the same.

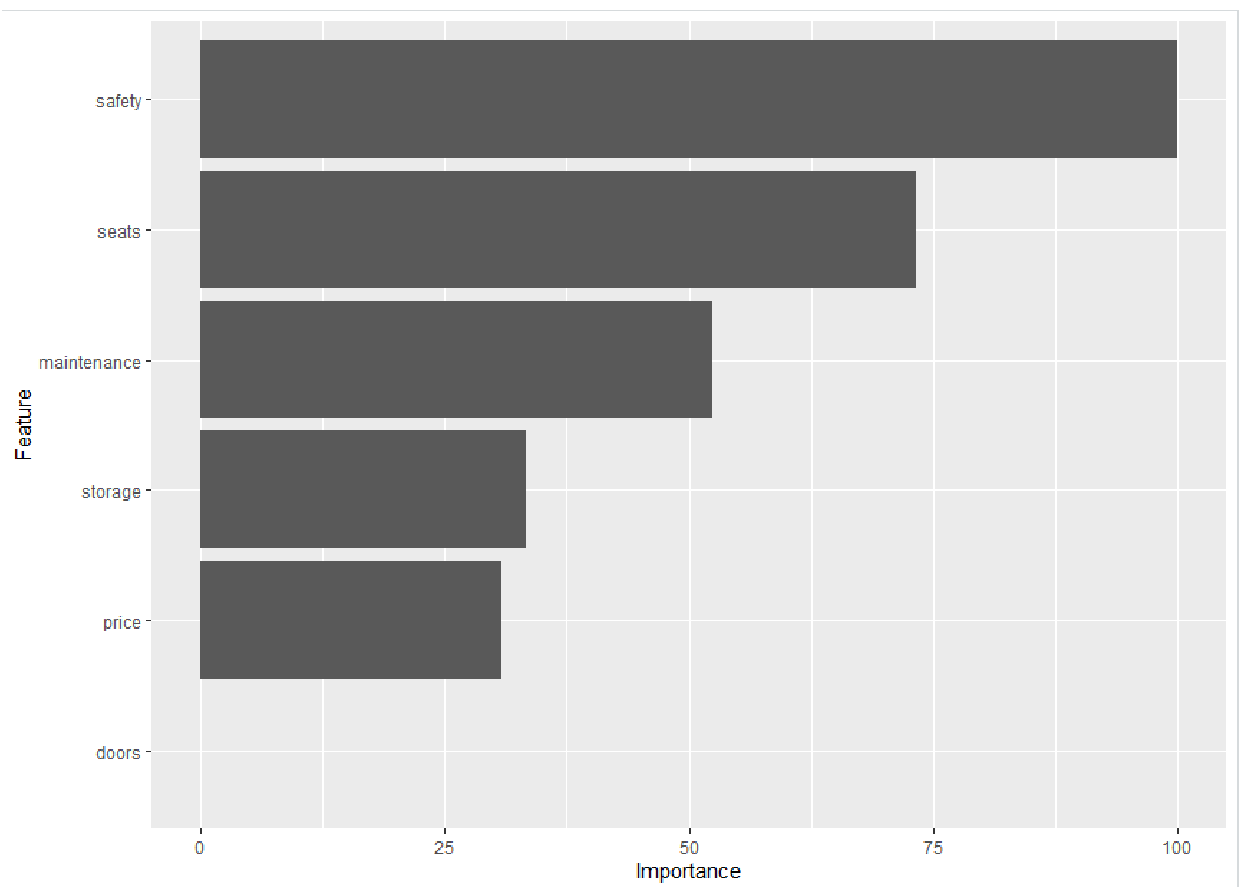


Fig – Graph of important variables

- Number of folds = 10 and repetition = 3 times. The values of accuracy and kappa are as follows:

```
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: 1556, 1556, 1555, 1555, 1555, 1555, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9747319  0.9454202
  4     0.9845642  0.9666293
  6     0.9861079  0.9698344

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

Fig - K-fold values for n=10, rep =3

The important variables that are contributing towards acceptability of car are shown below:

```
> varImp(randomForest_default)
rf variable importance

            Overall
safety       100.00
seats         72.45
maintenance   51.85
storage       32.43
price         30.45
doors          0.00
```

Fig - Important variables after k-fold n=10 rep=3

From the above Fig, safety and seats are the important variables and below is the graph for the same.

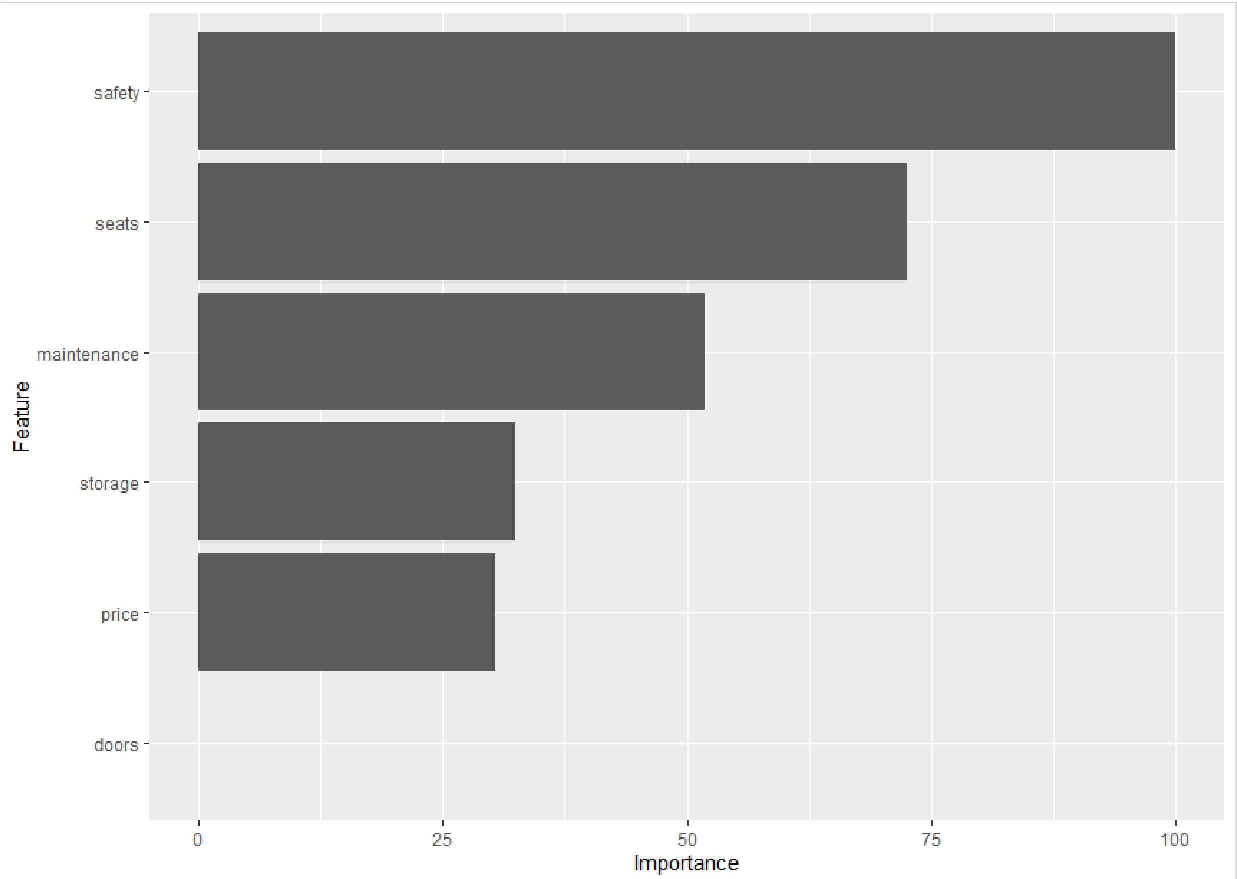


Fig – Graph of important variables

CONCLUSION

- From the above analysis and graphs, we can say that all variables are contributing to the acceptability of cars.
- Based on ROC and AUC we can conclude that "Random forest method" gives more accurate model which fits appropriately.
- The variables safety and seats are playing important role that majorly contribute to cars acceptability compared to other variables.
- The variable doors play very less role in cars acceptability.

APPENDIX A(REFERENCES)

[1] https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Classification/Decision_Trees

[2] https://www.youtube.com/watch?v=zmO3-7_I2zU&feature=youtu.be

[3] https://www.youtube.com/watch?v=cu-qaHDohU8&t=62s

[4] https://www.youtube.com/watch?v=YXr-KniXXp0&t=4s

APPENDIX B (R CODE)

RPART:

```
CAR_DATA = read.table("C:/Users/arunr/Desktop/MCDA/data mining/data_mining_assignments/data_mining_assignments/data_mining_assignment2/car.csv", sep=',', header=T)

View(CAR_DATA)

summary(CAR_DATA)

CAR.split <- sample(2, nrow(CAR_DATA),replace = TRUE, prob = c(2/3,1/3))

#trainig and testing data

CAR.train <- CAR_DATA[CAR.split == 1,1:7]

CAR.test <- CAR_DATA[CAR.split == 2,1:7]

#library('rpart')

#summary(X)

library(rpart)

#install.packages("rpart.plot")

library(rpart.plot)

carTree1 <- rpart(shouldBuy~., data = CAR.train, method = "class", control = rpart.control(minsplit=10))

carTree1

rpart.plot(carTree1)

#prediction

predcar1 <- predict(carTree1, newdata= CAR.test, type = "class")

head(predcar1)

treeCM = table(CAR.test[["shouldBuy"]],predcar1)

treeCM

sum(diag(treeCM)/sum(treeCM))

#drawing AOC

library(pROC)

predcar1 <- predict(carTree1, newdata= CAR.test, type = "prob")

buyROCAcc1 = roc(CAR.test$shouldBuy, predcar1[,1])

plot(buyROCAcc1)
```

```
auc(buyROCAcc1)
buyROCGOOD = roc(CAR.test$shouldBuy, predcar1[,2])
plot(buyROCGOOD)
auc(buyROCGOOD)W
buyRocUnacc = roc(CAR.test$shouldBuy, predcar1[,3])
plot(buyRocUnacc)
auc(buyRocUnacc)
buyRocVgood = roc(CAR.test$shouldBuy, predcar1[,4])
plot(buyRocVgood)
auc(buyRocVgood)
carTree1$importance
varImp(carTree1)
rpart.rules(carTree1)
```

RANDOM FOREST:

```
#Random forest
set.seed(100)
#Here y variable is category.we use classification
library(randomForest)
set.seed(222)
#passing data to RF model
rf<-randomForest(shouldBuy~.,data=CAR.train,ntree=50,mtry=4,importtance = TRUE,proximity=TRUE)
rf
plot(rf)
library("randomForestSRC")
help("predict")
predictCars = predict(rf1, newdata= CAR.test, type = "CLASS")
confusionMatrix(predictCars,CAR.test$shouldBuy)
#ROC
```

```
predictCars = predict(rf, newdata= CAR.test, type = "prob")
buyacc <- roc(CAR.test$shouldBuy, predictCars[,1])
auc(buyacc)
plot(buyacc)
buygood <- roc(CAR.test$shouldBuy, predictCars[,2])
auc(buygood)
plot(buygood)
buyunacc <- roc(CAR.test$shouldBuy, predictCars[,3])
auc(buyunacc)
plot(buyunacc)
buyvgood <- roc(CAR.test$shouldBuy, predictCars[,4])
auc(buyvgood)
plot(buyvgood)
rf$importance
varImp(rf)
ggplot(varImp(rf))
```

K-FOLD CROSS VALIDATION:

```
# K-fold Cross Validation
install.packages("caret")
library(caret)
#Define control (simillar to rpart) repeatedcv means repeated crossvalidation. It is a resampling method.
# 5 folds and 3 iterations.
control = trainControl(method="repeatedcv", number = 5, repeats = 3)
help("trainControl")
```

#To Produce the same result again

set.seed(1801)

#random forest method and k-fold cross validation.

randomForest_default = train(CAR_DATA[,1:6], CAR_DATA$shouldBuy,method = "rf",metric = "Accuracy", trControl = control)

#install.packages("e1071")

print(randomForest_default)

#Print important variables

varImp(randomForest_default)

#Plot important variables

ggplot(varImp(randomForest_default))