SAINT MARY'S UNIVERSITY SINCE 1802

One University. One World. Yours.

MSc in Computing and Data Analytics

MCDA 5580 – Data and Text Mining

Assignment – 2

Submitted to:

Trishla Shah

Prepared by:

Allen Mathew - A00432526

Gyaneshwar Rao - A00433014

Meghashyam - A00432392

Table of Contents

Executive Summary

The car industry has a vast market across the globe. The data is used to evaluate the consumer decision based on different attributes/features of a car such as safety, maintenance, seats, price, storage and doors. Two different classification algorithms, 'Decision Tree' and 'Random Forest' were used to create a model to predict the labelled variable in the dataset. By using the prediction models, we could predict/classify the customer feedback based on the various attributes of the car. 'Random Forest' gave the better results and provided us with the highest accuracy when compared to the results from 'Decision Tree'. The feature/variable 'safety' of the vehicle is the most significant insight derived from Random Forest, this attribute contributed to the positive decision/feedback from the customer. The second most important characteristic is 'seats' followed by 'maintenance'. The "door's" attribute didn't play any role in the customer's decision/feedback. We can improve the model further by using more parameters such as 'engine size', 'brand' etc. and compare results with different classification models like Gradient boosting etc.

Objective

To understand the different methods/models that solves the classification problem and furthermore understand the results from the two algorithms "Decision trees" and "Random Forest" and find out how these models with respect to different attributes would help in proper classification and their contribution in the decision process. K- fold is used to get the best model according to the kappa value and the accuracy value which intact will result into the best model that can be achieved with a supervised learning method. AUC will help us to verify which categories will be predicted accurately as according to the magnitude of the area generated by different models.

Data Summary

The dataset “Car Data” is used for the analysis. It consists of 1,728 records and several attributes that contains various information pertaining to a vehicle. The following are the attributes of the data set:

Attributes	Description
Price	It indicates the amount of money spent to purchase a vehicle. It consists of 4 categories i.e. low, med, high, vhigh
Maintenance	It indicates the amount of money that needs to be spent for the maintenance of the vehicle. It consists of 4 categories i.e. low, med, high, vhigh.
Doors	It indicates the number of doors present for the vehicle. It consists of 4 categories i.e. 2, 3, 4, 5more.
Seats	It indicates the number of doors present in the vehicle. It consists of 3 categories i.e. 2, 4, more.
Storage	It indicates the amount of storage space available for the vehicle. It consists of 3 categories i.e. small, med, big.
Safety	It indicates the level of safety that is present for the vehicle. It consists of 3 categories i.e. low, med, high.
Should Buy (shouldBuy)	It indicates weather the vehicle should be purchased or not. It consists of 4 categories i.e. • unacc - Unacceptable • acc - Acceptable • good - Good • vgood - Very Good

Table1 : Attributes of the “Car Data” Dataset

Observations:

Data cleaning was not required for this dataset since the data was correctly formatted and mainly all the attributes in the dataset consist of categorical data. The following table briefly describes the observation made on each of the attributes in the data set (i.e."Car Data"):

Attributes	Description
Price	• It consists of 4 categories i.e. low, med, high, vhigh. • Each category consists of the same number of records, i.e. 432.
Maintenance	• It consists of 4 categories i.e. low, med, high, vhigh. • Each category consists of the same number of records, i.e. 432.
Doors	• It consists of 4 categories i.e. 2, 3, 4, 5more. • Each category consists of the same number of records, i.e. 432.
Seats	• It consists of 3 categories i.e. 2, 4, more. • Each category consists of the same number of records, i.e. 576.
Storage	• It consists of 3 categories i.e. small, med, big. • Each category consists of the same number of records, i.e. 576.
Safety	• It consists of 3 categories i.e. low, med, high. • Each category consists of the same number of records, i.e. 576.
Should Buy (shouldBuy)	• The attribute consists of 4 categories i.e. unacc, acc, good, vgood. • The records present in each category is as follows: o unacc - 70% (i.e. 1210 records) o acc - 22.2% (i.e. 384 records) o good - 4% (i.e. 69 records) o vgood - 3.6% (i.e. 65 records)

Table2 : Observations made on the Attributes of the "Car Data" Dataset

Understanding:

There are 2 types of variables:

- Independent Variable
- Dependent Variable

An independent variable is manipulated or changed to analyze the effect it has on the dependent variable, which is then observed and documented. (Carlson, 2006)

For the given “Car Data” dataset, the independent variables act as the input for the analysis of the decision-making process on whether the customer will purchase the vehicle. The dependent variable records the results, or the decisions/feedback made by the customer.

The following attributes act as the Independent and Dependent variable for the given dataset:

- Independent Variable:
    - Price
    - Maintenance
    - Doors
    - Seats
    - Storage
    - Safety
- Dependent Variable:
    - ShouldBuy

Method/Approach

The following steps were performed for the analysis:

1. **Dataset Selection:**
    - We begin by selecting the "Car Date" dataset that contains information about vehicles.
    - Then we review the source data to understand the content, its structure and its interconnectivity with other attributes within the same dataset.
    - Then we check if the data is correctly formatted and consistent. To do so we perform various analytical checks on the data, such as count.
2. **Variable Selection:**
    - We identify the dependent and independent variables in the dataset.
3. **Divided the Dataset:**
    - We identify the optimal split ratio
    - We have chosen 75% as the optimal split ratio because we needed a big chunk of data to train the model, so that the model is more 'educated' and a small chunk of test data would satisfy to test the model.
    - We divide the dataset into Training Set and Test Set
4. **Create Classification Model:**
    - With the given dataset create classification models like: i) Decision Tree, ii) Random Forest.
5. **Testing the Model:**
    - The above models will be tested using a confusion matrix, which is a table that compares the test data (i.e. actual value) with the value predicted by the model, it provides the performance of a classification model, by indicating the total number of true positive/negatives and false positive/negatives.
6. **KFOLD**
    - This algorithm splits the data more than once and tell us about the performance of the model. The number of different splits is configurable, and one can specify the number of different train-test splits. (Brownlee, 2019)
    - It ensures that the testing is done thoroughly, and it eliminates any biasness that might be involved during the calculation of the accuracy/prediction.
7. **ROC Curve (Receiver Operating Characteristics)**
    - ROC curve is the graph that is plotted from the confusion matrix of a prediction. The sensitivity (true positive rate) and specificity (false positive rate) forms the x and y axis respectively for the curve. The ideal result would have the highest true positive rate along with the lowest false positive rate. The area under the ROC is called as AUC and it is used to compare the usefulness of the tests. (Narkhede, 2019)

Decision Tree

The Decision Tree solely works on entropy. The lowest chaos/uncertainty will give rise to high Information Gain which helps the algorithm to choose the appropriate labels to create the Decision Tree. Rpart is a binomial tree which helps to categorize and hold decisions for prediction according to distinct features. For our analysis we have used Decision Tree to find if the purchase is going to be done or not according to the customer feedback.

Identifying the Min Split Value:

The prediction with a model created by Rpart varies according to the minsplit parameter. The greater the minsplit, the smaller the tree and the faster the algorithm. For our analysis we begin by training the model using the Train dataset for different minsplit values which range from 0 to 100.

Figure1 : Graph Plot of the Accuracy attained using Decision Tree for different Min Split values

We would consider a minsplit value where there is a continuous difference in prediction accuracy. From the above Figure there is a steep fall in the accuracy, the accuracy prolongs to fall up until it reaches a min split value of 80. Hence, we chose minsplit value as 80 where the model doesn't overfit.

The table below further illustrates the accuracy attained for the various minsplit values when using the Train dataset.

Min Split	Accuracy
0	0.9460293
10	0.9460293
20	0.9390902
30	0.9390902
40	0.9390902
50	0.9282961
60	0.9121049
70	0.8897456
80	0.8750964
90	0.8750964
100	0.8750964

Table 3: Accuracy attained using Decision Tree for different Min Split values

Testing / Prediction:

We then utilize the selected Min Split (i.e. 80) to create the Decision Tree Model using "Rpart". The following figure shows that our model can clearly classify the different categories (i.e. unacc, acc, good and vgood) of the dependent's variable "shouldBuy".

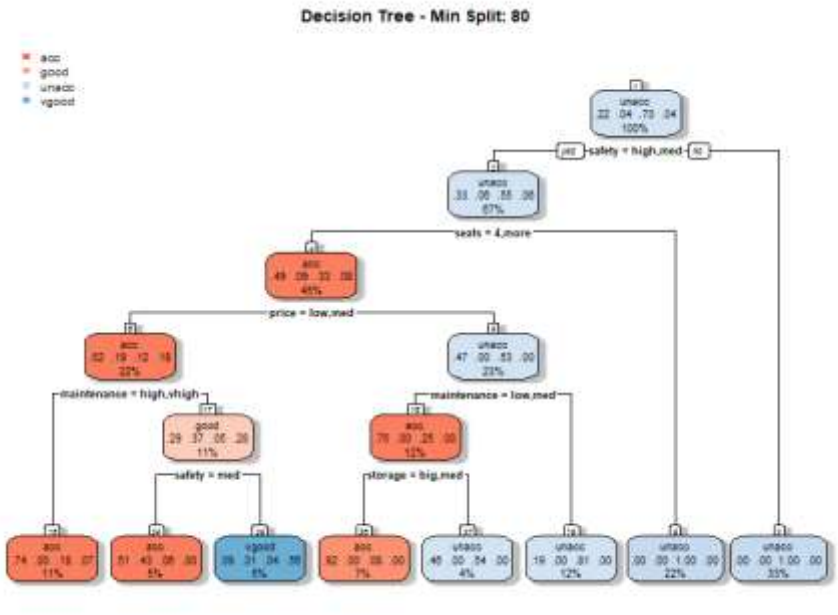


Figure 2: Decision Tree Model with Min Split 80

Then we can test our model using the Test dataset. To do so we create a confusion matrix, the table has 4 different combinations of the predicted value from the model and actual values from the Test dataset. **When taking min split 80, the accuracy of the prediction using Decision Tree is 0.8793503 or 88%.**

		Actual Values			
		acc	good	unacc	vgood
Predicted values	acc	81	0	11	4
	good	9	0	0	8
	unacc	16	0	285	1
	vgood	3	0	0	13

Table 4: Confusion Matrix of the Decision Tree Model for Test Set

K Fold Validation for Decision Tree

K Fold validation is used to identify the best model for a specific K value. The model is tested with different combination having training set with K-1 data sets and testing set with 1 data set. The best model is chosen with the highest values of CP and Kappa values. We were able to identify the features and the influence of the features in the model while it is used for prediction. Safety and maintenance are the top 2 important features which were identified as important features when the K fold is performed using decision tree algorithm.

CART		
1728 samples **6 predictors** **4 classes: 'acc', 'good', 'unacc', 'vgood'**		
cp	Accuracy	Kappa
0.04054054	0.8552902	0.6882582
0.11583012	0.8044223	0.6013948
0.12934363	0.7395931	0.3286828
Accuracy was used to select the optimal model using the largest value. **The final value used for the model was cp = 0.04054054.**		

Table 5 : K Fold validation

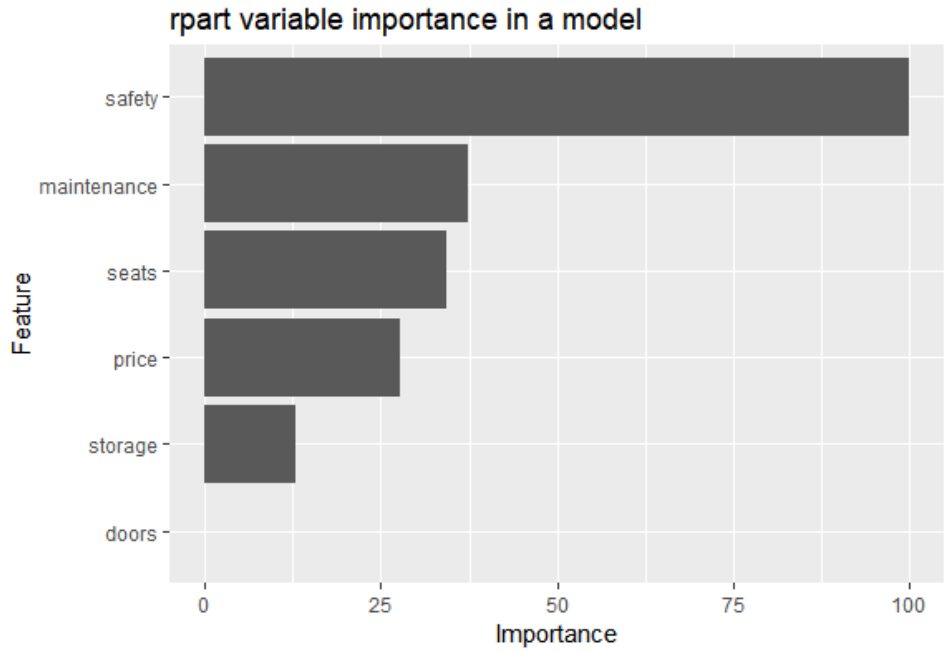


Figure3 : Bar Chart on the Information gain from each individual variable

Rpart variable importance	
Features	**Overall**
safety	100.00
maintenance	37.36
seats	34.28
price	27.70
storage	12.92
doors	0.00

Table6 : K Fold Rpart variable importance based on Decision Tree Model

ROC (Receiver Operator Characteristics) Curve for Decision Tree

The comparison of Sensitivity and specificity for 4 categories according to the decision tree are graphically plotted, which depicts how accurate the decision tree can predict these categories. (Rickert, 2019)

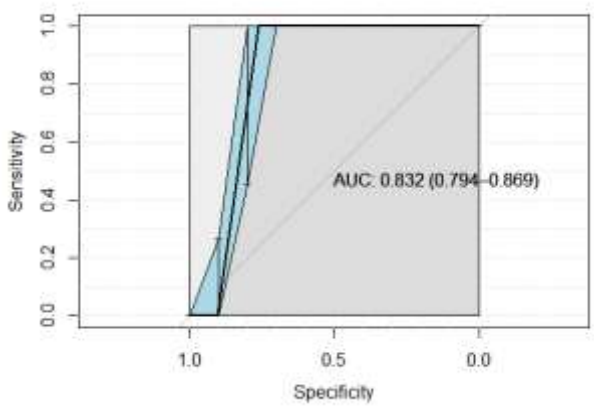
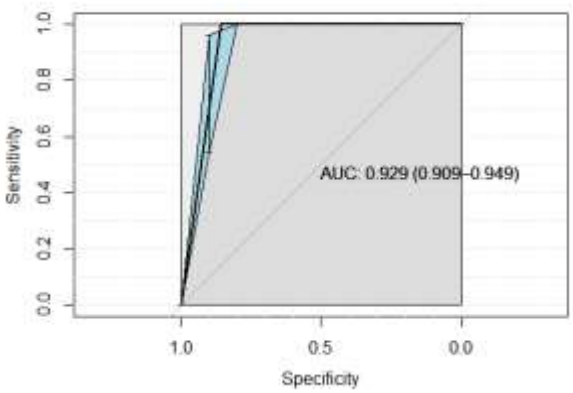
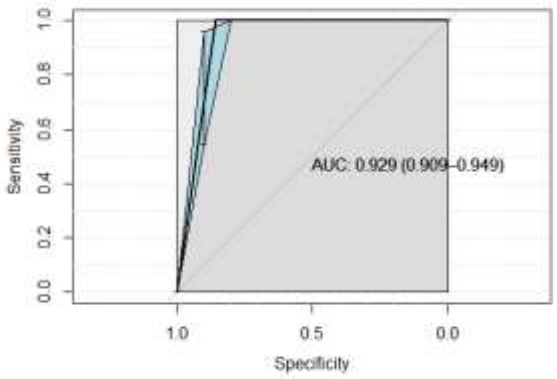
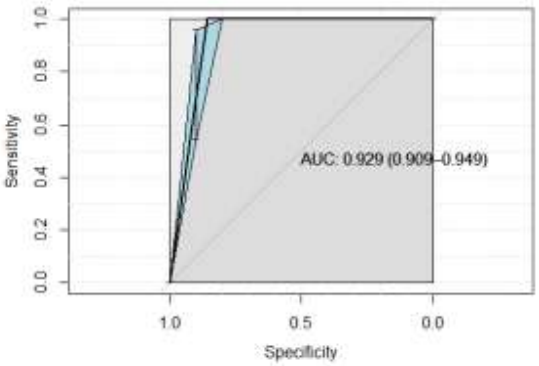
Accurate	Good
	
Un-accurate	**V-Good**
	

Table 7: ROC using Decision Tree for the different categories of the 'shouldBuy' attribute

Random Forest

For our Analysis, we have used Random Forest to find the best prediction of classification to identify if the buy is going to be made or not by the customer. The model predicts above 90% which has a strong advocacy when compared to Decision tree.

Identifying the nTree Value:

For our analysis we begin by training the model using the Train dataset for different nTree values which range from 100 to 700.

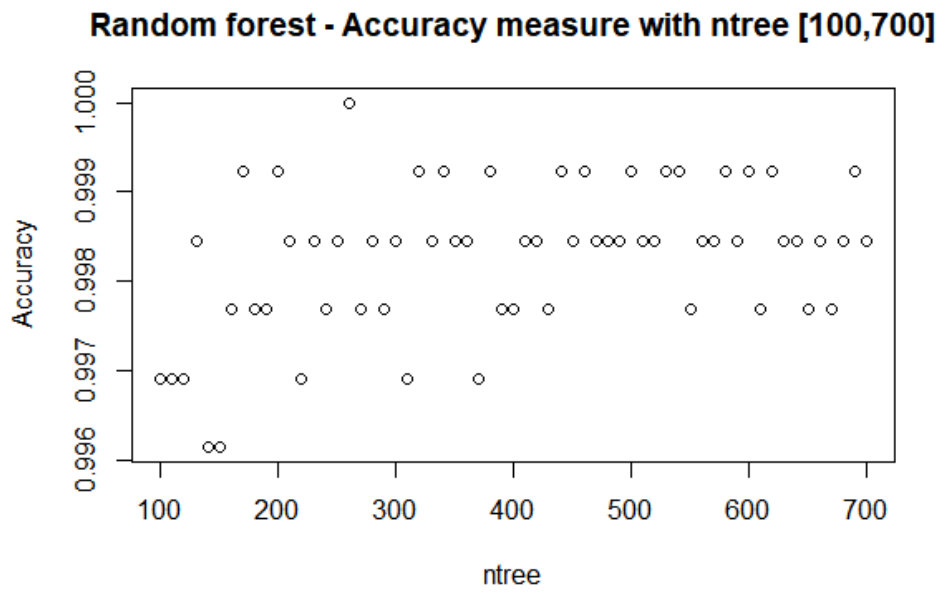


Figure 4: Graph Plot of the Accuracy attained using Random Forest for different nTree values

From the above Figure there is not much significant change in the accuracy, since the accuracy ranges from 99.6% to 100%. But during our analysis we were able to get an accuracy of 100% for an nTree value of 260. Hence, we chose the nTree value as 260 to get a more accurate prediction.

		Actual Values			
		acc	good	unacc	vgood
Predicted values	acc	93	1	6	3
	good	0	16	0	0
	unacc	3	0	296	0
	vgood	0	0	0	13

Table 8: Confusion Matrix of the Random Forest Model for Test Set

Testing / Prediction:

We then utilize the selected nTree (i.e. 260) to create the Random Forest Model. Then we can test our model using the Test dataset. To do so we create a confusion matrix, the table has 4 different combinations of the predicted value from the model and actual values from the Test dataset. **When taking nTree 260, the accuracy of the prediction using Random Forest is 0.9698376 or 96.9%.**

K Fold Validation for Random Forest

The metrics using K fold give us an Accuracy of 98.3% with a kappa value of 0.964 when performed with random forest. The analysis result in giving the results of highest preferred features like safety and seats for acceptability by the customer. 'Doors' feature is not considered in predicting the purchase of the car.

Random Forest

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

mtry	Accuracy	Kappa
2	0.9706563	0.9365383
4	0.9807049	0.9584037
6	0.9836008	0.9643985

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

Table 9: ***K Fold validation***

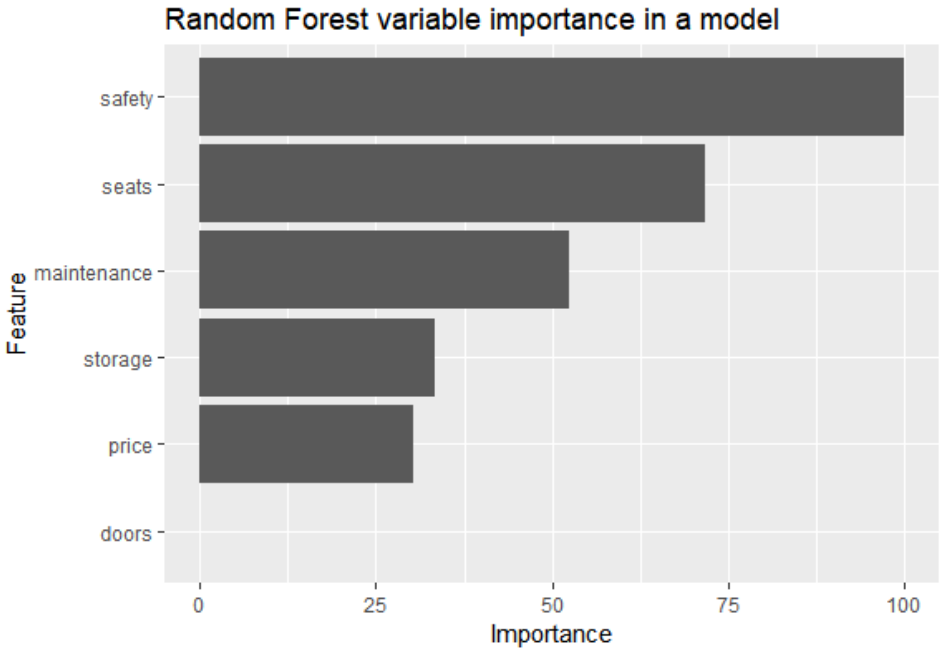


Figure5 : Bar Chart on the Information gain from each individual variable

rf variable importance	
Features	**Overall**
safety	100.00
seats	71.79
maintenance	52.36
storage	33.36
price	30.22
doors	0.00

Table 10: ***K Fold rf variable importance based on Random Forest Model***

ROC (Receiver operator characteristics) for Random Forest:

The comparison of Sensitivity and specificity for 4 categories according to the Random Forest are graphically plotted, which depicts how accurate the Random Forest can predict these categories. (Rickert, 2019)

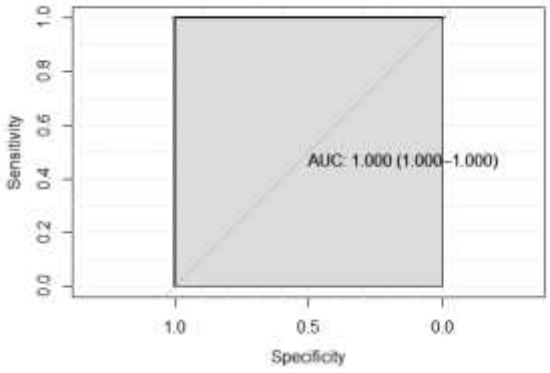
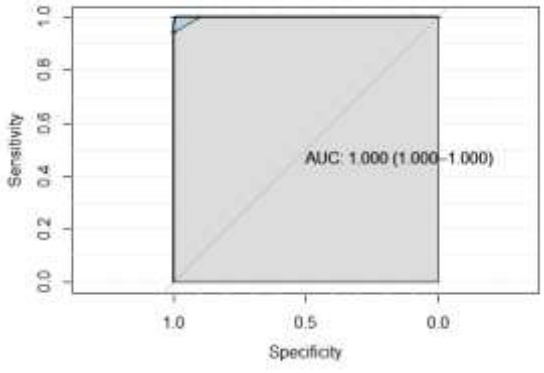
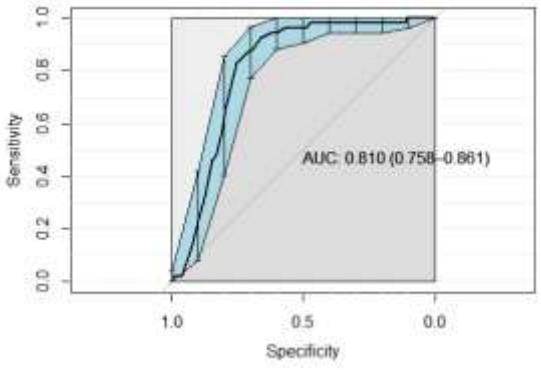
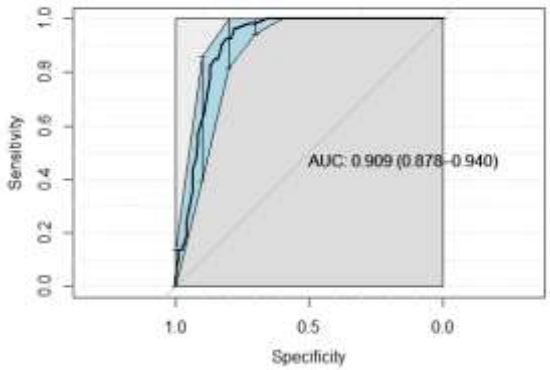
Accurate	Good
	
Un-Accurate	**V-Good**
	

Table 11: ROC using Random Forest for the different categories of the 'shouldBuy' attribute

Comparisons between Decision Tree and Random Forest:

Decision Tree and Random Forest gave good insights on the features that were used for classification and also gave us results with which we could compare and evaluate the accuracy between the models. Both the models listed the important features that contributed the most for the classification and below are the results.

Decision Tree		Random Forest	
Order of importance of features:			
Both the algorithms had given different values for the information gain from the features except for the attribute "Safety".			
safety	100.00	safety	100.00
maintenance	37.36	seats	71.79
seats	34.28	maintenance	52.36
price	27.70	storage	33.36
storage	12.92	price	30.22
doors	0.00	doors	0.00
ROC Curve Comparison:			
The AUC is comparatively higher in random forest when compared to decision tree. There are categories like un-accurate and V-good which are dominant in decision tree model. It is better to consider a hybrid variety for different categories specifically.			
Accurate	0.832	Accurate	1
Good	0.929	Good	1
Un-Accurate	0.929	Un-Accurate	0.81
V-Good	0.929	V-Good	0.909

Table 12: Decision Tree and Random Forest Model Results Comparison

Conclusion:

Every classification model might have different parameters prioritized according to their respective algorithms. It is better to take the best of the different classification algorithms and predict the values. Decision tree has a varied prediction accuracy when the minsplit value varies, while Random Forest maintains a consistent prediction accuracy in the above scenario. The AUC curve plotted for Un-accurate and V-good is comparatively more for decision tree rather than random forest algorithm, while Random Forest has good prediction accuracy for 'Accurate' and 'Good'. Finally, the "safety" feature of the vehicle plays an important role in the decision-making process and the number of "doors" in the vehicle is the least important feature. In the future it would be better to have more classification models and identify the features and their weight in the prediction and use appropriate model.

References:

Brownlee, J. (2019). A Gentle Introduction to k-fold Cross-Validation. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/k-fold-cross-validation/ [Accessed 5 Jun. 2019].

Carlson, Robert. A concrete introduction to real analysis. CRC Press, 2006. p.183.

Narkhede, S. (2019). Understanding AUC - ROC Curve. [online] Towards Data Science. Available at: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 [Accessed 5 Jun. 2019].

Rickert, J. (2019). Some R Packages for ROC Curves. [online] Rviews.rstudio.com. Available at: https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/ [Accessed 5 Jun. 2019].

Appendix A:

ROC:

- Sensitivity is defined by True Positive/ (true Positive + False Positive) and Specificity is defined by True Negative/ (true negative + false positive). There is another parameter called precision defined as True Positive/(true Positive + false Positive).
- These parameters give us a plot of different confusion matrices for different thresh holds which evaluates how much the model is reliable and can be used for prediction. The AUC (area under the curve) is used to compare two different algorithms for the study. The algorithm producing highest AUC curve should be considered for prediction.

Appendix B:

```
# install.packages("Rpart")
# install.packages("pROC")
# install.packages('caTools')
# install.packages("randomForest")
# install.packages("caret")
# install.packages("e1071")
# install.packages("ggplot2")

setwd("F:/SMU2/Data mining/Assignment2")
#-----------------Decision tree----------------------#
#       RPART
library(Rpart)
library(pROC)
library(randomForest)
library(caret)
library(ggplot2)
set.seed(123)
carData=read.csv("car.csv")
summary(carData)
# View(carData)

x=carData[,1:6]
y=carData[,7]

# Splitting the dataset into the Training set and Test set
library(caTools)
set.seed(123)
split = sample.split(carData$shouldBuy, SplitRatio = 0.75)
training_set = subset(carData, split == TRUE)
test_set = subset(carData, split == FALSE)

list_data <- c()
```

```
#-----------------Decision Tree , Evaluating accuracy for different
minsplit -------------------#

library(Rpart)
for (i in seq(0, 1000, 10)){
  treeCar =
Rpart(shouldBuy~.,data=training_set,method="class",control=Rpart.control(mi
nsplit=i))
  #Pridiction
  predCar=predict(treeCar,training_set,type="class")
  #Matrix
  treeCM=table(training_set[,7],predCar)
  treeCM
  #Accuracy
  accuracy = sum(diag(treeCM))/sum(treeCM)
  list_data <- c(list_data,accuracy)
}

temp_list_data <- c(seq(0, 1000, 10))
temp_list_data
list_data
plot(y=list_data,x=temp_list_data,
     main = "Decision Tree - Accuracy measure with Minsplit [0,1000]",
     ylab = "Accuracy",
     xlab = "Min Split Value")


#-----------------Decision Tree , choosing minsplit as 140 ----------------
-----------------#
library(Rpart)
treeCar =
Rpart(shouldBuy~.,data=training_set,method="class",control=Rpart.control(mi
nsplit=140))
treeCar

predCar=predict(treeCar,training_set,type="class")
#Matrix
treeCM=table(training_set[,7],predCar)
treeCM
#Accuracy of the prediction using Decision Tree
accuracy=sum(diag(treeCM))/sum(treeCM)

#-----------------Decision Tree , ROC curves ------------------------------
---#

predCar_prob=predict(treeCar,training_set,type="prob")
```

```r
roc_1 = roc(training_set[,"shouldBuy"],predCar_prob[,1],smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(roc_1)
plot(sens.ci, type="shape", col="lightblue")
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
plot(sens.ci, type="bars")


roc_2 = roc(training_set[,"shouldBuy"],predCar_prob[,2],smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(roc_2)
plot(sens.ci, type="shape", col="lightblue")
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
plot(sens.ci, type="bars")

roc_3 = roc(training_set[,"shouldBuy"],predCar_prob[,3],smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(roc_3)
plot(sens.ci, type="shape", col="lightblue")
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
plot(sens.ci, type="bars")

roc_4 = roc(training_set[,"shouldBuy"],predCar_prob[,4],smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(roc_4)
plot(sens.ci, type="shape", col="lightblue")
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
plot(sens.ci, type="bars")
```

```
#-----------------Decision Tree , K Fold --------------------------------#

library(caret)
library(e1071)
control <- trainControl(method="cv", number=10, savePredictions=TRUE)
seed <- 123
metric <- "Accuracy"
set.seed(seed)
rf_default <- train(x,y, method="Rpart", metric=metric, control =
Rpart.control(minsplit = 140),
                    trControl=control)
print(rf_default)
varImp(rf_default)
p = ggplot(varImp(rf_default))
p + ggtitle("Rpart variable importance in a model")
```

```
#-----------------------------------------------------------------#
# Random Forest
#
#
setwd("F:/SMU2/Data mining/Assignment2")

library(Rpart)
library(pROC)
library(randomForest)
library(caret)

carData=read.csv("car.csv")
summary(carData)

# Splitting the dataset into the Training set and Test set
library(caTools)
set.seed(123)
split = sample.split(carData$shouldBuy, SplitRatio = 0.75)
training_set = subset(carData, split == TRUE)
test_set = subset(carData, split == FALSE)

x=training_set[,1:6]
y=training_set[,7]

list_data <- c()

#-------------Random Forest optimal ntree-------------#

library(randomForest)
for (i in seq(100, 700, 10)){
  rf=randomForest(x,y, ntree=i)
  rfp=predict(rf,x)
```

```
  rfCM=table(rfp,y)
  accuracy = sum(diag(rfCM))/sum(rfCM)
  list_data <- c(list_data,accuracy)
}

temp_list_data <- c(seq(100, 700, 10))
temp_list_data
list_data
plot(y=list_data,x=temp_list_data,
     main = "Random forest - Accuracy measure with ntree [100,700]",
     ylab = "Accuracy",
     xlab = "ntree")

#------------Random Forest , choose ntree 260 as the accuracy 1--------#

x=training_set[,1:6]
y=training_set[,7]

rf=randomForest(x,y, ntree=260)
rfp=predict(rf,test_set[,1:6])
rfCM=table(rfp,test_set[,7])
rfCM
#Accuracy of the prediction using randomForest
accuracy = sum(diag(rfCM))/sum(rfCM)
accuracy

#------------Random Forest ROC curves ---------------#
rf_prob = predict(rf,x,type="prob")


roc_1 = roc(training_set[,"shouldBuy"],rf_prob[,1],smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(roc_1)
plot(sens.ci, type="shape", col="lightblue")
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
plot(sens.ci, type="bars")


roc_2 = roc(training_set[,"shouldBuy"],rf_prob[,2],smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
```

```
sens.ci <- ci.se(roc_2)
plot(sens.ci, type="shape", col="lightblue")
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
plot(sens.ci, type="bars")

roc_3 = roc(training_set[,"shouldBuy"],rf_prob[,3],smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(roc_3)
plot(sens.ci, type="shape", col="lightblue")
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
plot(sens.ci, type="bars")

roc_4 = roc(training_set[,"shouldBuy"],rf_prob[,4],smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
sens.ci <- ci.se(roc_4)
plot(sens.ci, type="shape", col="lightblue")
## Warning in plot.ci.se(sens.ci, type = "shape", col = "lightblue"): Low
## definition shape.
plot(sens.ci, type="bars")


#-------Random Foresh K fold----------------------------#

x=carData[,1:6]
y=carData[,7]

# Use caret package for 10 fold cross validation
library(caret)
control <- trainControl(method="repeatedcv", number=10, repeats=3)
seed <- 123
metric <- "Accuracy"
set.seed(seed)
rf_default <- train(x,y, method="rf",ntree = 260, metric=metric,
trControl=control)
print(rf_default)
varImp(rf_default)
ggplot(varImp(rf_default))+ggtitle("Random Forest variable importance in a
model")
```