Supervised Learning on Car Buying Report

MCDA 5580

Team member:

Mitkumar Patel A00444857

Chirag Panasuriya A00442907

Wensho Li A00445457

Contents

1. Executive Summary

An analysis of Buying cars data was performed to better understand customers purchasing behaviors. The analysis focused on criteria of cars, which include price, maintenance costs, doors, seats, storage, and safety. Based on these conditions combining with opinions of customers (acceptable/unacceptable/good/ very good), writers finally abstract the behavior rules and classification of customers. This analysis very useful for car companies and even customers to decide which kind of car should be manufactured and which kind of car should be bought.

The decision tree algorithm may not be an optimal solution, but it helps at some point to predict rules for car company and customers. Study can derive based on criteria and decision can be made. Below are the solutions extracted from rules.

1. Generally, a family consist of 3 to 4 people nowadays, so most of customers preferred seats 3 more than it. Even car manufacturers companies are targeting these potential customers with Sedan car. Which are available in a range of sizes like small, compacts, mid-size, full-size.

2. Safety concern is also matter for all customers while making the decision. From the decision we can say customers' expectations regarding safety is medium to high based on their budget. Even car companies are providing new safety features in budget cars like such as automatic emergency braking, blind spot warning, lane departure warning, rear cross-traffic assist, and even rear automatic braking.

3. Most customers fall into low to medium range budget. Even car compacts are manufacturing in budget cars to target these customers. Luxury and sports car customers are very few compared to in budget car customers.

4. Most of customers are not expecting big storage size as storage comes letter in priority and they can adjust in storage as per their budget.

During 2019-20 in Canada, Sedan cars like Honda Civic and Toyota Corolla were the Best-Selling Cars. Our decision tree can predict this type of car selling.

2. Objective

Car, an efficient transportation facility, has become an important part of daily life. In such a case, car selling could contribute to the profit for enterprises which are relevant to vehicle industry. Specially, vehicle retail and product enterprise both want to understand more about criteria of their customers buying cars so they can better align corporate initiatives, such as marketing and revenue generation, producing plan on specific class of vehicles. It is the report authors' intent to perform various data preparation, segmentation, rules generation to produce an analysis that will provide beneficial information to the business owner. In the absence of specific requirements from the business, the reports' authors will consider the goal of car buying classification and rules generation to be knowledge generation for criteria of purchasing cars behavior. A successful analysis will produce delineated and communicable profiles for car buying that will aid in decision making for both

customers, sellers and enterprise of vehicle producing.

3. About Data

To execute the analysis, authors got data of acceptance questionnaire for vehicles which have various fittings and costs. The acceptance data provides purchasing motivation as it relates to costs and accessories of cars. A brief description of this acceptance table is provided herein:

Acceptance Questionnaire Table

Provides acceptance information for each kind of car base on multiple costs and accessories, such a description of the price, maintenance, number of doors, number of seats, storage, and safety. And each row also contains information of customers' opinion on acceptance or not.

Number of records: 1728
Number of unique records: 1728

4. Design/Methodology/Approach

When performing a large analysis with foreign data, it is important to devise a technique that support the overall objective, is communicable and repeatable. When classifying the criteria of purchasing cars, the most frequently used algorithm is R Part and Random Forest. Using the questionnaire data as an example.

In our dataset, we have few input columns and one output column. So, we have implemented supervised learning. We have two options in supervised learning, linear regression, and classification. Here output column is categorical variable, so it is classification problem. We can solve classification problem using various methods like decision tree, Neural Network, Support vector machine, and so on. We have implemented Decision tree and Random Forest. Both have their own benefits and drawbacks. Decision tree is simple and easy to interpret but it's not flexible as random forest. Before classification we have done feature selection using statistical techniques.

5. Decision Tree

5.1 Summary of Decision Tree

The Decision Tree algorithm generate rules, which is the conditional statement that can easily use within a database and easily be understood by humans and to identify a set of records. Rules show the basis for the model's predictions.

5.2 Steps of Building a Decision Tree

All these steps are produced in R Script, the following steps are just for demonstrating the procedure.

Step 1: Determine the Root of the Tree.

Since decision trees are used for classification, you need to determine the classes which are the basis for the decision.

In this case, it is the last column, that is “Should Buy” column with classes acceptance and unacceptance.

To determine the root Node, we need to compute the entropy.

To do this, we need create a frequency table for the classes (the acc/unacc column).

Should Buy	
acceptance	unacceptance
N	M

Step 2: Calculate Entropy for The Classes.

In this step, you need to calculate the entropy for the “Should Buy” column and the calculation step is given below.

Entropy(ShouldBuy) = E (N, (N+M))

Step 3: Calculate Entropy After Split for Each Attribute.

For the other four attributes, we need to calculate the entropy after each of the split.

- E(ShouldBuy, price)
- E(ShouldBuy, maintance)
- E(ShouldBuy, doors)
- E(ShouldBuy,seats)
- E(ShouldBuy,storage)
- E(ShouldBuy,safety)

The entropy for two variables is calculated using the formula as following.

$$Entropy(\mathrm{S},\mathrm{T}) = \sum_{c \in T} P(c)E(c)$$

P: Probability of c event happens

E: Entropy

Step 4: Calculate Information Gain for each split.

The next step is to calculate the information gain for each of the attributes. The information gain is calculated from the split using each of the attributes. Then the attribute with the largest information gain is used for the split.

The information gain is calculated using the formula:

Gain (S, T) = Entropy(S) – Entropy (S, T)

For example, the information gain after splitting using the price attribute is given by:

Gain (ShouldBuy, Outlook) = Entropy (ShouldBuy) – Entropy (ShouldBuy, price)

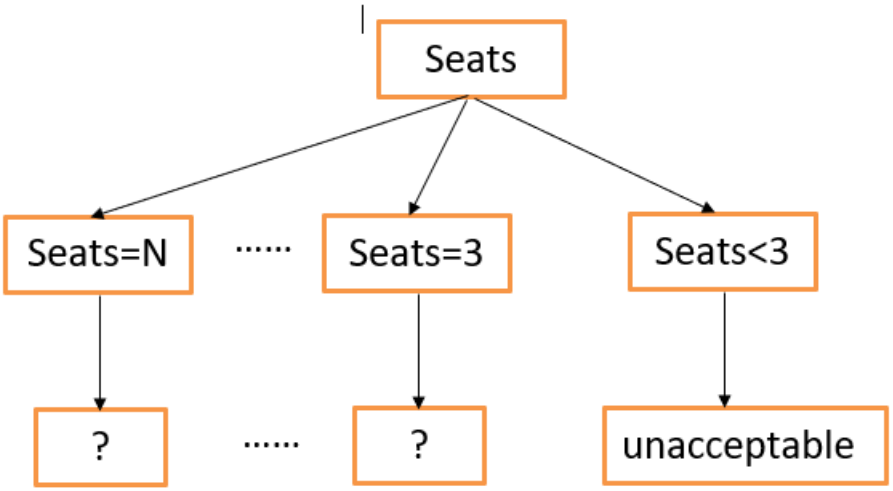
Having calculated all the information gain, we now choose the attribute that gives the highest information gain after the split.

Step 5: Perform the Split.

Draw the First Split of the Decision Tree

Now that we have all the information gain, we then split the tree based on the attribute with the highest information gain.

From our calculation, the highest information gain comes from seats. Therefore, the split will look like this:



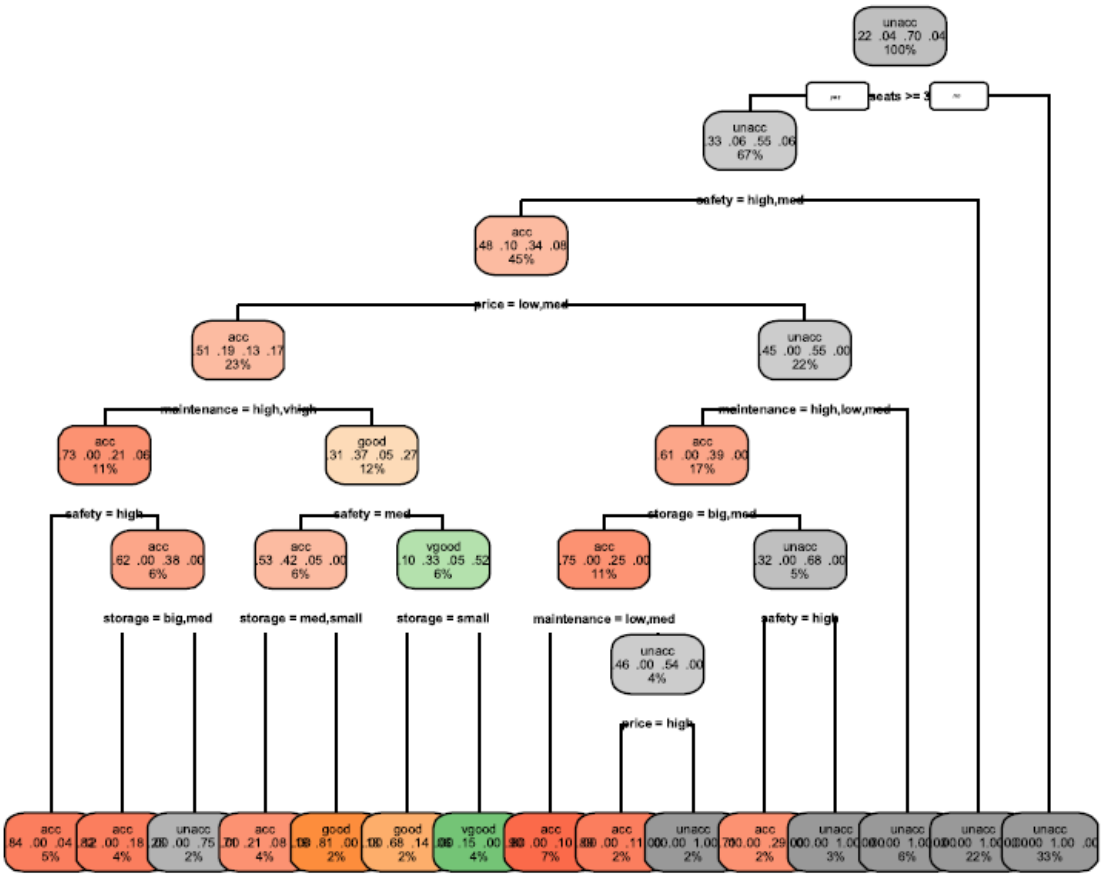
Step 6: Perform Further Splits.

Follow the same procedure of splitting until we split all the attributes.

Step 7: Complete the Decision Tree.

Got the final decision tree.

5.3 Diagrams of Decision Tree



5.4 Analysis about Some of Rules

The Decision Tree algorithm generate rules, which is the conditional statement that can easily use within a database and easily be understood by humans and to identify a set of records. Rules show the basis for the model's predictions.

It generates conditional statement with set of IF-THEN rules. We can express a rule in the following from −

IF condition **THEN** conclusion

- The IF part of the rule is called rule precondition.

  The precondition consist of one or more attribute tests and these tests are logically ANDed

- The THEN part of the rule is called rule consequent.

  The consequent part consists of class prediction.

If the condition holds true for a given tuple, then the antecedent is satisfied. Outcome can be predicted using this.

To extract a rule from a decision tree −

- A rule is generated for each path from the root to the leaf node.
- To form a rule precondition, each splitting criterion is logically ANDed.

- The leaf node holds the class prediction, forming the rule consequent.

1. IF (seats < 3) THAN **Unacceptable**

Decision tree is predicting that with less than 3 seats customer is not ready to buy the car

2. IF (seats >= 3) & (safety is high) & (price is low OR med) & (maintenance is low) & (storage is small) THAN **good**

Here in this rule, seats are 3 or more, safety more concern to customer, price can be low to medium, maintenance should be low and with less storage also customer is good to buy the car.

3. IF (seats >= 3) & (safety is high OR med) & (price is high) & (maintenance is high) THAN **acceptable**

This rule explained, with 3 or more seats, medium to high safety, even with high price and high maintenance customer is accepted to buy the car

4. IF (seats >= 3) & (safety is = low) THAN **Unacceptable**

Now with 3 or more seats and low safety customer is not ready to buy the car

5. IF (seats >= 3) & (safety is high) & (price is high OR vhigh) & (maintenance is low OR med) THAN **acceptable**

This rule explains, seats are 3 or more, safety more concern to customer, price can be high to extremely high, maintenance should be low to medium customer is accepted to buy the car.

6. IF (seats >= 3) & (safety is high OR med) & (price is vhigh) & (maintenance is high) THAN **Unacceptable**

When Seats are 3 or more, safety medium to high, price exceedingly high, maintenance high, customer is not ready to buy the car.

7. IF (seats >= 3) & (safety is med) & (price is high OR vhigh) & (maintenance is low OR med) & (storage is big OR med) THAN **acceptable**

This rule explained, with 3 or more seats, medium safety, even with high to very high price and medium to big storage, customers are accepted to buy the car

8. IF (seats >= 3) & (safety is high) & (price is low OR med) & (maintenance is low OR med) & (storage is big OR med) THAN **vgood**

Here, with 3 or more seats, high safety, with low to medium price, low to medium maintenance and medium to big storage, it's incredibly good to buy the car

9. IF (seats >= 3) & (safety is med) & (price is low OR med) & (maintenance is low OR med) & (storage is big) THAN **good**

In this rule, with 3 or more seats, medium safety, with low to medium price, low to medium maintenance and with big storage, customer is good to buy the car

10. IF (seats >= 3) & (safety is high OR med) & (price is low OR med) & (maintenance is high OR vhigh) & (storage is big OR med) THAN **acceptable**

This rule predicts, with 3 or more seats, medium to high safety, with low to medium price, high to very high maintenance and with medium to big storage customer is accepted to buy the car

6. Evaluation of Model (Decision Tree)

6.1 Accuracy & Recall

Classification Accuracy is what we usually mean when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ Inputs}$$

$$Accuracy = \frac{64 + 12 + 242 + 14}{346} = 0.9595$$

The recall is the measure of our model correctly identifying True Positives. Thus, for all the patients who actually have heart disease, recall tells us how many we correctly identified as having a heart disease. Mathematically:

$$Recall = \frac{True\ Positive}{False\ Negative + True\ Positive}$$

$$Recall\ for\ ACC = \frac{74}{74 + 4 + 1 + 0} = 0.9367$$

Accuracy and all recalls are shown below.

```
> treeCM

predSonar acc good unacc vgood
    acc    74    0     5     1
    good    4   14     1     0
    unacc   1    0   235     0
    vgood   0    2     0     9
> print(paste("accuracy",sum(diag(treeCM))/sum(treeCM),sep=" "))
[1] "accuracy 0.959537572254335"
>
> treeCM=as.data.frame.matrix(table(predSonar,ValidSet$shouldBuy))
> for(i in colnames(treeCM)){
+   print(paste("Recall for", i," is",treeCM[i,i]/sum(treeCM[i]),sep=" "))
+
+ }
[1] "Recall for acc  is 0.936708860759494"
[1] "Recall for good  is 0.875"
[1] "Recall for unacc  is 0.975103734439834"
[1] "Recall for vgood  is 0.9"
```

Decision tree has good accuracy on validation set but it has 8 level in the three. As number of level increase in the tree, complexity increase and hard to interpret the rules. All recall values are above 0.85 so we can say that performance of this model is good.

6.2 AUC

Area Under Curve (AUC) is one of the most widely used metrics for evaluation. It is used for binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. Before defining AUC, let us understand two basic terms:

- True Positive Rate (Sensitivity): True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.

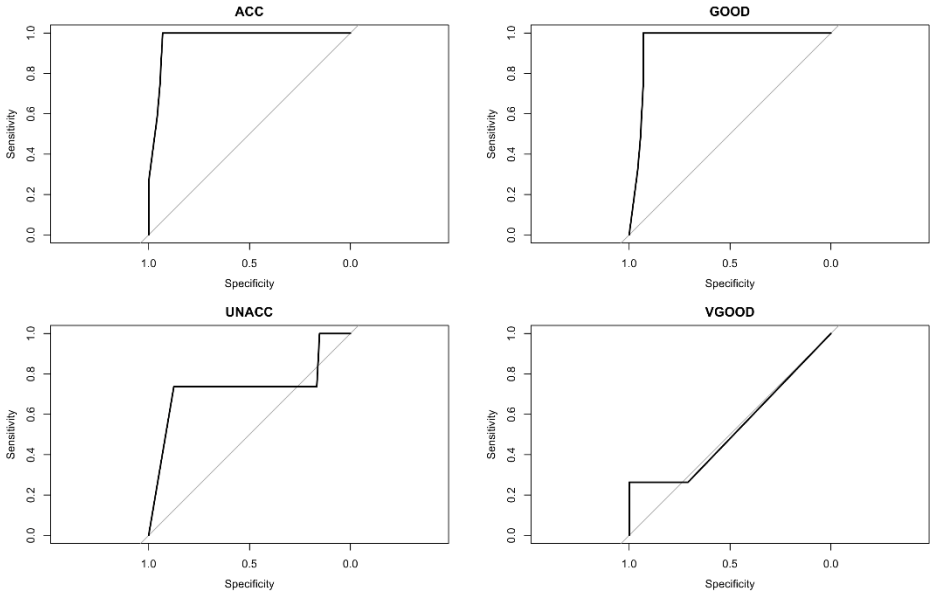
$$True\ Positive\ Rate = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- True Negative Rate (Specificity): False Positive Rate corresponds to the proportion of negative data points that are correctly considered as negative, with respect to all negative data points.

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive}$$

- False Positive Rate: False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

$$False\ Positive\ Rate = 1 - Specificity$$

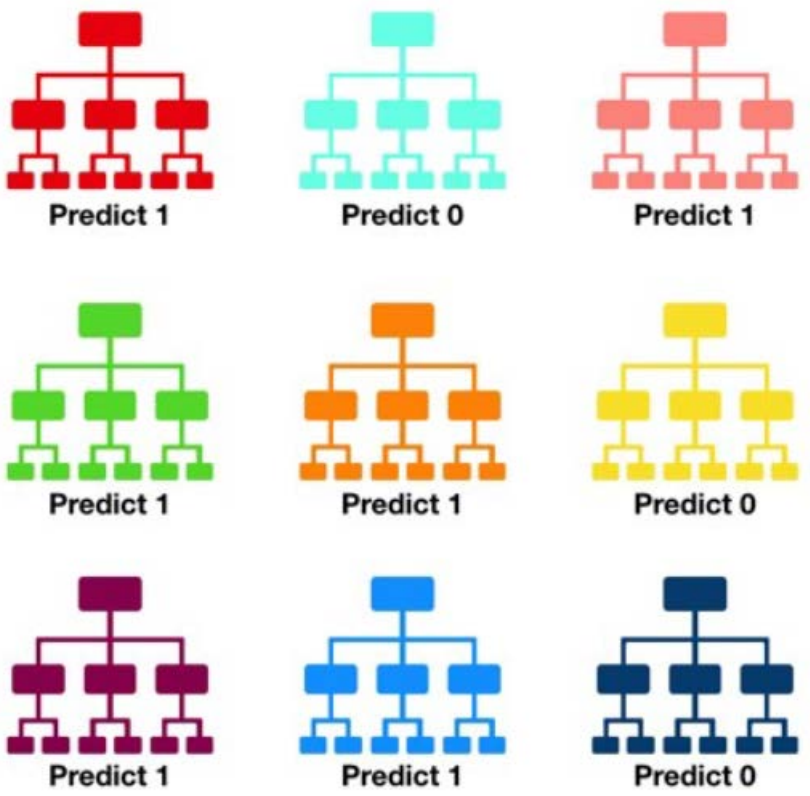


An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has AUC near to the 0 which means it has the worst measure of separability. ACC and GOOD have perfect graph for Area Under Curve, which means True Negative and True Positive are separable from each other. When two distributions overlap, we introduce type 1 and type 2 errors. Depending upon the threshold, we can minimize or maximize them. Model can be classified car falsely into another class for UNACC. Final case of VGOOD is the worst situation. When AUC is approximately 0.5 to 0.7, the model has no discrimination capacity to distinguish between positive class and negative class.

7. Random Forest

7.1 Summary of Random Forest

Random forest, like its name implies, consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model's prediction (see figure below).

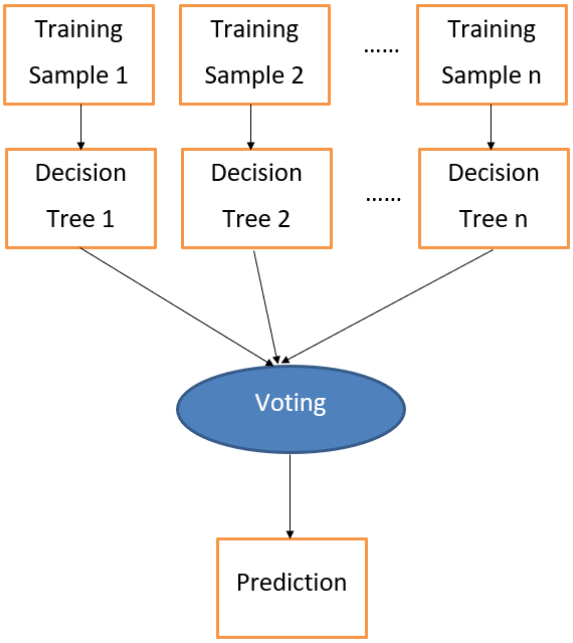


After generating all decision trees, these decision trees will work as a committee and voting for the prediction. The way random forest working protects each individual decision tree falling in errors and achieve a wonderful effect on accuracy for prediction as long as they do not constantly all err in the same direction.

7.2 Steps of Random Forest Working

- **Step 1** − First, start with the selection of random samples from a given dataset.
- **Step 2** − Next, this algorithm will construct a decision tree for every sample.
- **Step 3** - Then it will get the prediction result from every decision tree.
- **Step 4** − In this step, voting will be performed for every predicted result.
- **Step 5** − At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working –



8. Evaluation of Model (Random Forest)

8.1 Accuracy & Recall

Random forest has 95.37% accuracy. We can calculate accuracy and recall from confusion matrix. Confusion matrix, accuracy and recall are mentioned in the image.

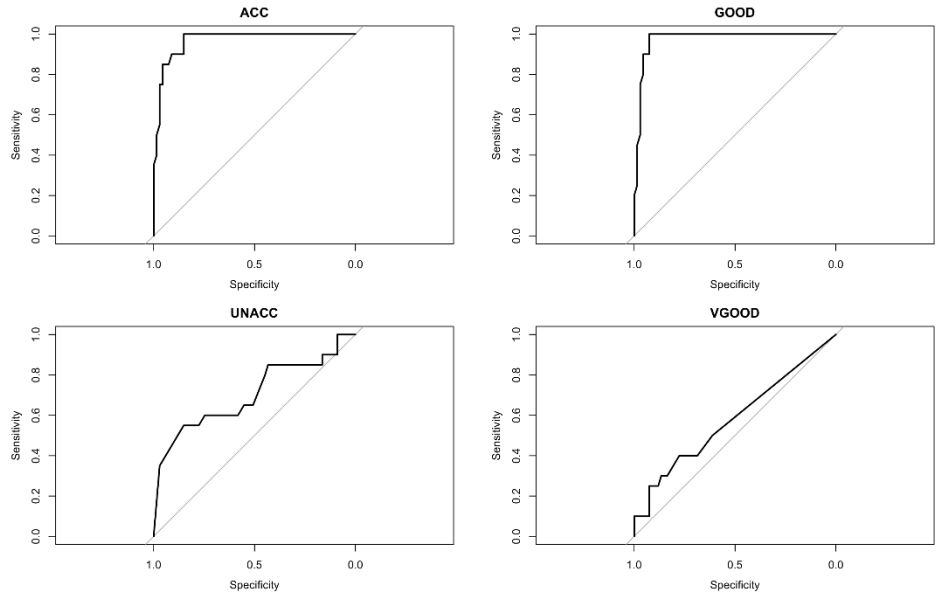
```
> treeCM

predValid acc good unacc vgood
    acc    72    0     8     1
    good    2   10     0     2
    unacc   1    0   241     0
    vgood   1    1     0     7
> print(paste("accuracy",sum(diag(treeCM))/sum(treeCM),sep=" "))
[1] "accuracy 0.953757225433526"
> treeCM=as.data.frame.matrix(table(predValid,ValidSet$shouldBuy))
> for(i in colnames(treeCM)){
+   print(paste("Recall for", i," is",treeCM[i,i]/sum(treeCM[i]),sep=" "))
+
+ }
[1] "Recall for acc  is 0.947368421052632"
[1] "Recall for good  is 0.909090909090909"
[1] "Recall for unacc  is 0.967871485943775"
[1] "Recall for vgood  is 0.7"
>
```

Overall accuracy of random forest is less compared to decision tree. Recall value of ACC, GOOD and UNACC are very high, but VGOOD is misclassified many times, so it has low recall. For this dataset random forest preform worse than Decision tree, it's because rules are simple, and model is overfitted in the Random Forest.

8.2 AUC

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. AUC for all categories is shown below:



ACC and GOOD have almost similar graph and both have AUC equals to one. Random forest can effectively discriminate negative and positive classes. ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds. UNACC has low false positive rate and high True positive rate compared to VGOOD. Random forest id overfit to data that's why VGOOD and UNACC have low AUC.

9. Feature Selection

There are multiple ways in which we can do feature selection for decision tree. In the case of classification problems where input variables are also categorical, we can use statistical tests to determine whether the output variable is dependent or independent of the input variables. If independent, then the input variable is a candidate for a feature that may be irrelevant to the problem and removed from the dataset. A categorical variable is a variable that may take on one of a set of labels. Here in this dataset all variables are categorical so we can impalement Chi-Squared Test. We have compared all the input variables to the output variable and computed P-value and compare with the alpha value. We choose alpha is 0.1. Alpha is a threshold value used to judge whether a test statistic is statistically significant. If P-value is less than alpha value which means variables have relation with each other. We found only "doors" variable, which has high p-value so we decided not to choose in out input.

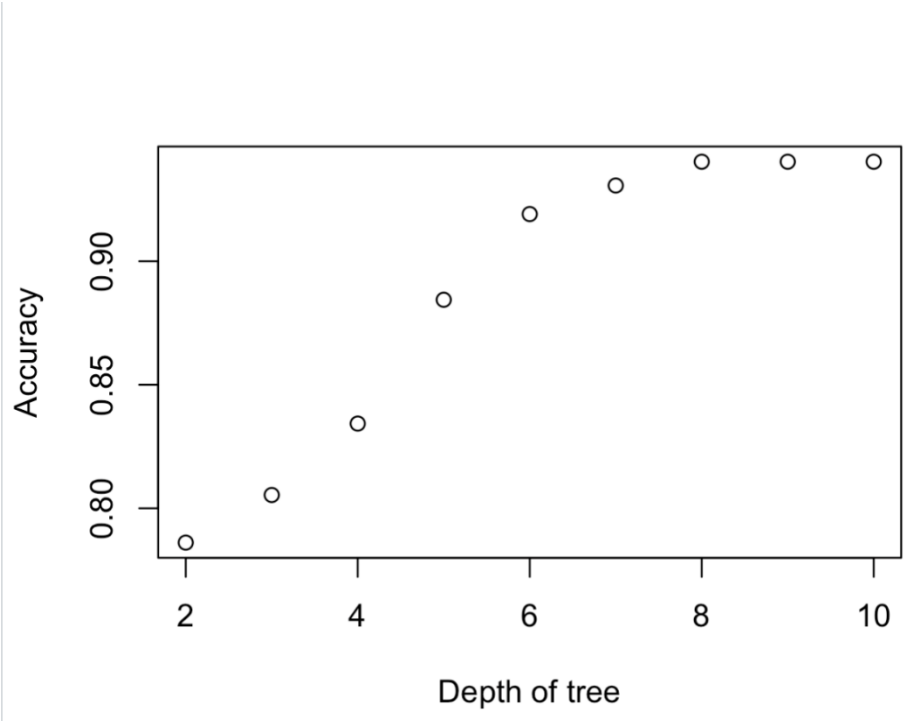
chi-squire:

```
[1] "p value for price is 5.92806259921337e-36"
[1] "p value for maintenance is 2.54765198450779e-26"
[1] "p value for doors is 0.320242159900306"
[1] "doors has less p-value so we'll drop"
[1] "p value for seats is 4.03996804727073e-77"
[1] "p value for storage is 1.02944027531348e-09"
[1] "p value for safety is 2.38915539904397e-100"
```

10. Parameters Tune Procession

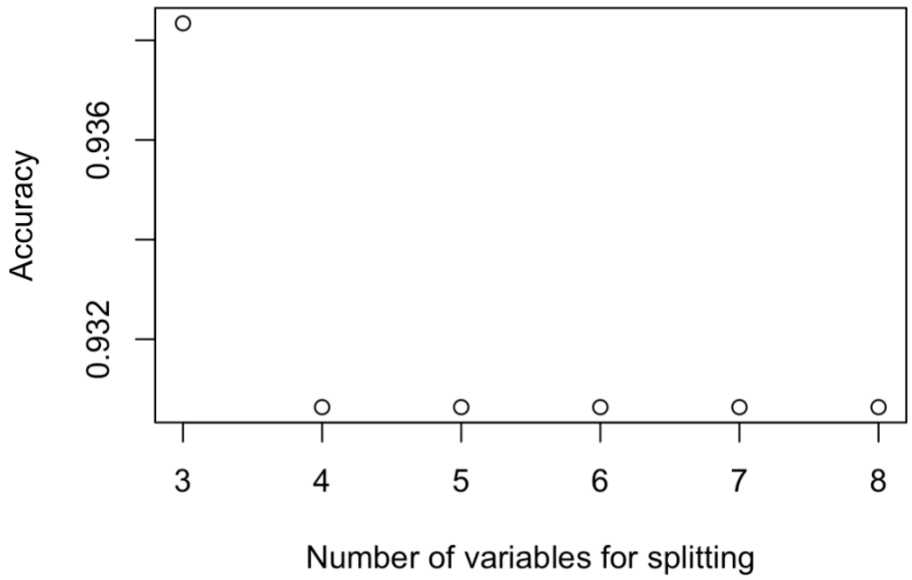
10.1 Depth of Decision Tree:

Recursive Partitioning and Regression Trees has many parameters using which we can create effective decision tree. Depth of decision tree is hyperparameter, so we check our model on various depth using test data. We found that after 8th level of decision tree accuracy is not increased so that is the best value for maximizing accuracy.



10.2 RandomForest ntree and mtry

We used RandomForest: Classification and Regression with Random Forest for implementing Random Forest. This function allows us to choose various parameters such as number of trees, number of parameters for splitting, replacement of data values for training, leaf node size and so on. We checked number of trees to grow in forest and we found that ntree =500 for better model. Number of variables randomly sampled as candidates at each split is denoted as mtry in the parameter. We checked accuracy of random forest for mtry between 3 to 8. Below graph shows that at mtry=3 we got highest accuracy.



11. Appendix

R Script for Decision Tree:

```
library(rpart)
library(caret)
library(pROC)
library(rpart.plot)

sonarData=read.csv("car.csv",header=T)
train <- sample(nrow(sonarData), 0.8*nrow(sonarData), replace = FALSE)
TrainSet <- sonarData[train,]
ValidSet <- sonarData[-train,]


#chi square test useful to find relationship between two categorical variables.
Carheader=colnames(sonarData)[which( colnames(sonarData)!="shouldBuy" )]
for (i in Carheader){
  x=chisq.test(sonarData[i],sonarData$shouldBuy)
  print(paste("p value for", toString(i), "is", toString(x$p.value),sep=" "))
  if(x$p.value>0.1){
    print(paste(i, "has less p-value so we'll drop",sep=" "))
  }
}
#choose depth of the Tree
acc=0
a=c()
depth=1
for (i in 2:10) {
  treeSonar                                                                                                    =
rpart(shouldBuy~price+maintenance+safety+seats+storage,data=TrainSet,method="class",parms
=list(split='information'),maxdepth=i)
  predSonar=predict(treeSonar,newdata=ValidSet,type="class")
  treeCM=table(predSonar,ValidSet$shouldBuy)
```

```
  a[i-1]=sum(diag(treeCM))/sum(treeCM)
  if(acc<a[i-1]){
    acc=a[i-1]
    depth=i
  }
}
plot(2:10,a,xlab="Depth of tree",ylab = "Accuracy")

treeSonar                                                                                                    =
rpart(shouldBuy~price+maintenance+safety+seats+storage,data=TrainSet,method="class",parms
=list(split='information'),maxdepth=depth)
predSonar=predict(treeSonar,newdata=ValidSet,type="class")
treeCM=table(predSonar,ValidSet$shouldBuy)
treeCM
print(paste("accuracy",sum(diag(treeCM))/sum(treeCM),sep=" "))

treeCM=as.data.frame.matrix(table(predSonar,ValidSet$shouldBuy))
for(i in colnames(treeCM)){
  print(paste("Recall for", i," is",treeCM[i,i]/sum(treeCM[i]),sep=" "))

}

rpart.plot(treeSonar)

predSonarProb = predict(treeSonar,newdata=ValidSet,type="prob")
attach(mtcars)
par(mfrow=c(2,2))
plot(roc(ValidSet[,7],predSonarProb[,1]),main="ACC",xlab="Specificity",ylab="Sensitivity")
plot(roc(ValidSet[,7],predSonarProb[,2]),main="GOOD",xlab="Specificity",ylab="Sensitivity")
plot(roc(ValidSet[,7],predSonarProb[,3]),main="UNACC",xlab="Specificity",ylab="Sensitivity")
plot(roc(ValidSet[,7],predSonarProb[,4]),main="VGOOD",xlab="Specificity",ylab="Sensitivity")
```

R Script for Random Forest:

```
library(caret)
library(randomForest)
sonarData=read.csv("car.csv",header=T)
train <- sample(nrow(sonarData), 0.8*nrow(sonarData), replace = FALSE)
TrainSet <- sonarData[train,]
ValidSet <- sonarData[-train,]
summary(TrainSet)
summary(ValidSet)
set.seed(7)

# Pick best "mtry" value for Random forest
a=c()
p=2
max=0
for (i in 3:8) {
  model3 <- randomForest(as.factor(shouldBuy) ~ price+maintenance+safety+seats+storage, data
= TrainSet, ntree = 500, mtry = i, importance = TRUE)
  predValid <- predict(model3, ValidSet, type = "class")
```

```
  a[i-2] = mean(predValid == ValidSet$shouldBuy)
  if(max<a[i-2]){
    max=a[i-2]
    p=i
  }
}
plot(3:8,a ,xlab="Number of variables for splitting",ylab = "Accuracy")


model1 <- randomForest(as.factor(shouldBuy) ~ price+maintenance+safety+seats+storage, data
= TrainSet, ntree = 500, mtry = p, importance = TRUE)

predValid <- predict(model1, ValidSet, type = "class")
# Checking classification accuracy
treeCM=table(predValid,ValidSet$shouldBuy)
treeCM
print(paste("accuracy",sum(diag(treeCM))/sum(treeCM),sep=" "))
treeCM=as.data.frame.matrix(table(predValid,ValidSet$shouldBuy))
for(i in colnames(treeCM)){
  print(paste("Recall for", i," is",treeCM[i,i]/sum(treeCM[i]),sep=" "))

}

predValid <- predict(model1, ValidSet, type = "prob")
attach(mtcars)
par(mfrow=c(2,2))
plot(roc(ValidSet[,7],predValid[,1]),main="ACC",xlab="Specificity",ylab="Sensitivity")
plot(roc(ValidSet[,7],predValid[,2]),main="GOOD",xlab="Specificity",ylab="Sensitivity")
plot(roc(ValidSet[,7],predValid[,3]),main="UNACC",xlab="Specificity",ylab="Sensitivity")
plot(roc(ValidSet[,7],predValid[,4]),main="VGOOD",xlab="Specificity",ylab="Sensitivity")
```