Random Forest

The dataset is taken from UCI website and can be found on this link. The data contains 7 variables – six explanatory (Buying Price, Maintenance, NumDoors, NumPersons, BootSpace, Safety) and one response variable (Condition). All the variables are categorical in nature and have 3-4 factor levels in each.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
# Load the dataset and explore
carsData <- read.table("C:/Users/zhuwe/Desktop/Visualization/Dataset/car.data.txt", sep= ',',header = F)
colnames(carsData) = c('BuyingPrice','Maintenance','NumDoors','NumPersons','BootSpace','Safety','Condit
head(carsData)
```

```
##   BuyingPrice Maintenance NumDoors NumPersons BootSpace Safety Condition
## 1       vhigh       vhigh        2          2     small    low     unacc
## 2       vhigh       vhigh        2          2     small    med     unacc
## 3       vhigh       vhigh        2          2     small   high     unacc
## 4       vhigh       vhigh        2          2       med    low     unacc
## 5       vhigh       vhigh        2          2       med    med     unacc
## 6       vhigh       vhigh        2          2       med   high     unacc
```

```
str(carsData)
```

```
## 'data.frame':    1728 obs. of  7 variables:
##  $ BuyingPrice: 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 ...
##  $ NumDoors   : Factor w/ 4 levels "2","3","4","5more": 1 1 1 1 1 1 1 1 1 1 ...
##  $ NumPersons : Factor w/ 3 levels "2","4","more": 1 1 1 1 1 1 1 1 1 2 ...
##  $ BootSpace  : 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 ...
##  $ Condition  : Factor w/ 4 levels "acc","good","unacc",..: 3 3 3 3 3 3 3 3 3 3 ...
```

```
summary(carsData)
```

```
##  BuyingPrice Maintenance  NumDoors   NumPersons BootSpace    Safety
##  high :432   high :432   2    :432   2   :576   big  :576   high:576
##  low  :432   low  :432   3    :432   4   :576   med  :576   low :576
##  med  :432   med  :432   4    :432   more:576   small:576   med :576
##  vhigh:432   vhigh:432   5more:432
##  Condition
##  acc  : 384
##  good :  69
##  unacc:1210
##  vgood:  65
```

Implement multiple Random Forest models with different hyper parameters. Split the dataset into train and validation set in the ratio 70:30. We can also create a test dataset, but for the time being we will just keep train and validation set.

```
set.seed(1)
train <- sample(nrow(carsData), 0.7*nrow(carsData), replace = FALSE)
TrainSet <- carsData[train,]
ValidSet <- carsData[-train,]
summary(TrainSet)
```

```
##  BuyingPrice Maintenance  NumDoors   NumPersons BootSpace    Safety
##  high :312   high :290   2    :291   2   :410   big  :402   high:403
##  low  :307   low  :316   3    :312   4   :409   med  :409   low :399
##  med  :300   med  :295   4    :287   more:390   small:398   med :407
##  vhigh:290   vhigh:308   5more:319
##  Condition
##  acc  :273
##  good : 52
##  unacc:840
##  vgood: 44
```

```
summary(ValidSet)
```

```
##  BuyingPrice Maintenance  NumDoors   NumPersons BootSpace    Safety
##  high :120   high :142   2    :141   2   :166   big  :174   high:173
##  low  :125   low  :116   3    :120   4   :167   med  :167   low :177
##  med  :132   med  :137   4    :145   more:186   small:178   med :169
##  vhigh:142   vhigh:124   5more:113
##  Condition
##  acc  :111
##  good : 17
##  unacc:370
##  vgood: 21
```

We will create a Random Forest model with default parameters and then fine tune the model by changing 'mtry'. We can tune the random forest model by changing the number of trees (ntree) and the number of variables randomly sampled at each stage (mtry).

Ntree: Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.

Mtry: Number of variables randomly sampled as candidates at each split. The default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3).

```
rfmodel.1 <- randomForest(Condition ~ ., data = TrainSet, importance = TRUE)
rfmodel.1
```

```
##
## Call:
##  randomForest(formula = Condition ~ ., data = TrainSet, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
```

```
##
##         OOB estimate of  error rate: 3.72%
## Confusion matrix:
##       acc good unacc vgood class.error
## acc   261    5     6     1  0.04395604
## good    8   42     0     2  0.19230769
## unacc  14    1   825     0  0.01785714
## vgood   7    1     0    36  0.18181818
```

```r
# Fine tuning parameters of Random Forest model
rfmodel.2 <- randomForest(Condition ~ ., data = TrainSet, ntree = 500, mtry = 6, importance = TRUE)
rfmodel.2
```

```
##
## Call:
##  randomForest(formula = Condition ~ ., data = TrainSet, ntree = 500,      mtry = 6, importance = TRUI
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 2.4%
## Confusion matrix:
##       acc good unacc vgood class.error
## acc   263    3     6     1  0.03663004
## good    3   48     0     1  0.07692308
## unacc   9    1   830     0  0.01190476
## vgood   4    1     0    39  0.11363636
```

By default, number of trees is 500 and number of variables tried at each split is 2 in this case. Error rate is 3.7%.When we have increased the mtry to 6 from 2, error rate has reduced from 3.7% to 2.4%. Now predict on the train dataset first and then predict on validation dataset.

```r
# Predicting on train set
predTrain <- predict(rfmodel.2, TrainSet, type = "class")
# Checking classification accuracy
table(predTrain, TrainSet$Condition)
```

```
##
## predTrain acc good unacc vgood
##     acc   273    0     0     0
##     good    0   52     0     0
##     unacc   0    0   840     0
##     vgood   0    0     0    44
```

```r
# Predicting on Validation set
predValid <- predict(rfmodel.2, ValidSet, type = "class")
# Checking classification accuracy
table(predValid,ValidSet$Condition)
```

```
##
## predValid acc good unacc vgood
##     acc   107    0     2     1
```

```
##     good    0   17     2     2
##     unacc   4    0   366     0
##     vgood   0    0     0    18
```

```r
mean(predValid == ValidSet$Condition)
```

```
## [1] 0.9788054
```

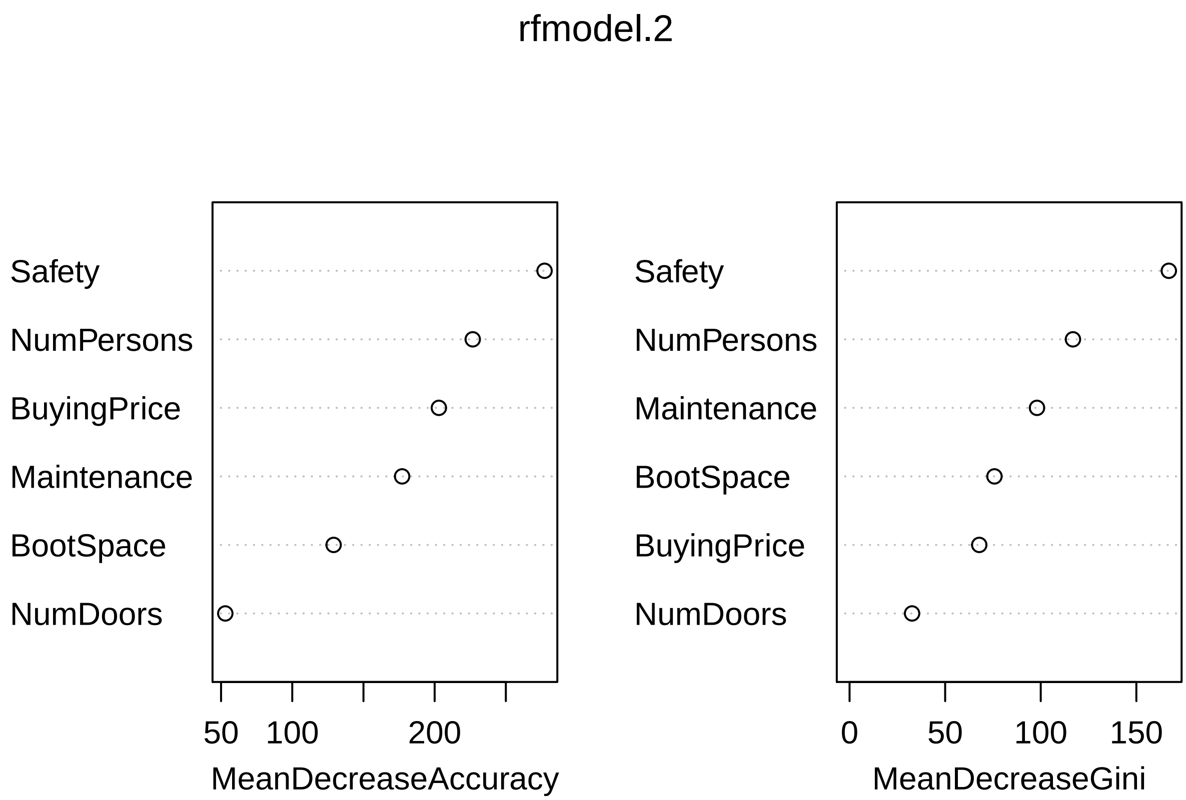
In case of prediction on train dataset, there is zero misclassification; however, in the case of validation dataset, 6 data points are misclassified and accuracy is 97.88%.

We can also check important variables. The below functions show the drop in mean accuracy for each of the variables.

```r
importance(rfmodel.2)
```

```
##                   acc     good     unacc    vgood MeanDecreaseAccuracy
## BuyingPrice 153.77631 78.97809 103.83523 73.11487            202.97207
## Maintenance 134.09001 81.29378  94.76347 51.46012            177.14774
## NumDoors     32.25721 14.99436  32.52565 25.60455             52.96216
## NumPersons  150.11131 53.59957 191.33989 54.35895            226.76044
## BootSpace    84.09127 58.90421  74.13733 56.27642            129.04147
## Safety      169.72484 94.25444 192.57213 95.89891            277.19561
##             MeanDecreaseGini
## BuyingPrice         67.83934
## Maintenance         98.03959
## NumDoors            32.69449
## NumPersons         116.83957
## BootSpace           75.78782
## Safety             166.98739
```

```r
varImpPlot(rfmodel.2)
```



Use 'for' loop and check for different values of mtry.

```
accuracy=c()

for (i in 3:8) {
  rfmodel.3 <- randomForest(Condition ~ ., data = TrainSet, ntree = 500, mtry = i, importance = TRUE)
  predValid <- predict(rfmodel.3, ValidSet, type = "class")
  accuracy[i-2] = mean(predValid == ValidSet$Condition)
}
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within
## valid range

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within
## valid range
```

```
accuracy
```

```
## [1] 0.9730250 0.9768786 0.9768786 0.9788054 0.9788054 0.9788054
```

Compare with decision tree

```
# Compare with Decision Tree
library(rpart)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

```
library(e1071)
# We will compare model 1 of Random Forest with Decision Tree model

dtmodel = train(Condition ~ ., data = TrainSet, method = "rpart")
dtpred = predict(dtmodel, data = TrainSet)
table(dtpred, TrainSet$Condition)
```

```
##
## dtpred  acc good unacc vgood
##   acc   252   52   131    44
##   good    0    0     0     0
##   unacc  21    0   709     0
##   vgood   0    0     0     0
```

```
mean(dtpred == TrainSet$Condition)
```

```
## [1] 0.7948718
```

```
# Running on Validation Set
dtpred.vs = predict(dtmodel, newdata = ValidSet)
table(dtpred.vs, ValidSet$Condition)
```

```
##
## dtpred.vs acc good unacc vgood
##     acc    96   17    59    21
##     good    0    0     0     0
##     unacc  15    0   311     0
##     vgood   0    0     0     0
```

```
mean(dtpred.vs == ValidSet$Condition)
```

```
## [1] 0.7842004
```

On training set we obtain 79.48% accuracy. On validation set we get 78.4%