Assignment_3

Barini Simhadri

10-16-2023

Jump to -
Problem 1
Problem 2
Problem 3
Problem 4

Importing necessary libraries

```
library(rpart)
library(tidyverse)
library(caret)
library(ggplot2)
library(dplyr)
library(rattle)
```

```
data =
read.csv2("C:/Users/bunty/Desktop/funda/week_5/breast_cancer_updated.csv" ,
header = T, sep = ",")
d = data
```

Problem 1

For this problem, you will perform a straightforward training and evaluation of a decision tree, as well as generate rules by hand. Load the breast_cancer_updated.csv data. These data are visual features computed from samples of breast tissue being evaluated for cancer1. As a preprocessing step, remove the IDNumber column and exclude rows with NA from the dataset.

```
summary(d)
```

```
##     IDNumber          ClumpThickness    UniformCellSize   UniformCellShape
##  Min.   :   61634   Min.   : 1.000    Min.   : 1.000    Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000    1st Qu.: 1.000    1st Qu.: 1.000
##  Median : 1171710   Median : 4.000    Median : 1.000    Median : 1.000
##  Mean   : 1071704   Mean   : 4.418    Mean   : 3.134    Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000    3rd Qu.: 5.000    3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000    Max.   :10.000    Max.   :10.000
##
##  MarginalAdhesion  EpithelialCellSize   BareNuclei       BlandChromatin
```

```
##  Min.   : 1.000   Min.   : 1.000    Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000    1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000    Median : 1.000   Median : 3.000
```

```
##  Mean   : 2.807   Mean   : 3.216    Mean   : 3.545   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000    3rd Qu.: 6.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000    Max.   :10.000   Max.   :10.000
##                                     NA's   :16
##  NormalNucleoli      Mitoses           Class
##  Min.   : 1.000   Min.   : 1.000   Length:699
##  1st Qu.: 1.000   1st Qu.: 1.000   Class :character
##  Median : 1.000   Median : 1.000   Mode  :character
##  Mean   : 2.867   Mean   : 1.589
##  3rd Qu.: 4.000   3rd Qu.: 1.000
##  Max.   :10.000   Max.   :10.000
##
```

Removing IDNumber column

```
d = select(d, -c(IDNumber))
```

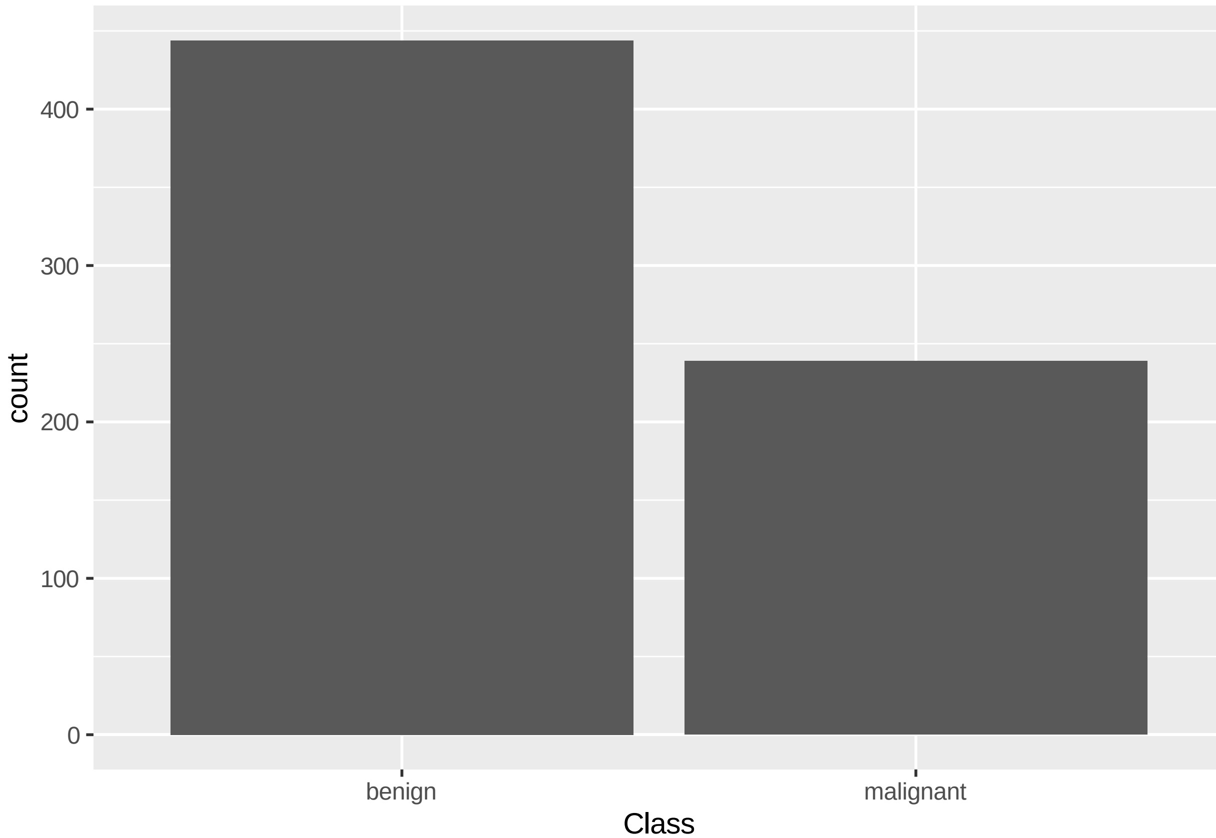
dropping rows with all NA's and it shows 0 after dropping all.

```
d = na.omit(d)
d[rowSums(is.na(d)) > 0, ]
```

```
##  [1] ClumpThickness      UniformCellSize     UniformCellShape    MarginalAdhesion
##  [5] EpithelialCellSize  BareNuclei          BlandChromatin      NormalNucleoli
##  [9] Mitoses             Class
## <0 rows> (or 0-length row.names)
```

a. Apply decision tree learning (use rpart) to the data to predict breast cancer malignancy (Class) and report the accuracy using 10-fold cross validation.

```
ggplot(d,aes(x=Class)) + geom_bar()
```




```r
#evaluation  method
train_control = trainControl(method = "cv", number = 10)
# Fit the model
tree1 <- train(Class ~., data = d, method = "rpart", trControl = train_control)
# Evaluate fit
tree1
```

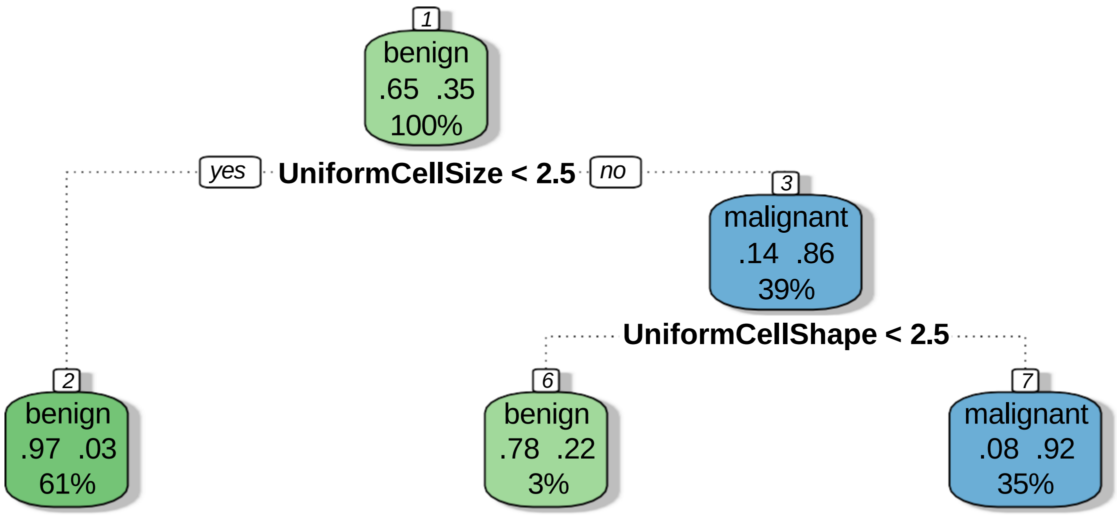
```
## CART
##
## 683 samples
##   9 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 614, 615, 615, 615, 615, 614, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02510460  0.9370205  0.8621292
##   0.05439331  0.9209292  0.8297487
##   0.79079498  0.8287084  0.5567141
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0251046.
```

```r
pred_tree <- predict(tree1, d)

# Generate confusion matrix for the test set
confusionMatrix(as.factor(d$Class), pred_tree)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       424        20
##   malignant     17       222
##
##                Accuracy : 0.9458
##                  95% CI : (0.9261, 0.9616)
##     No Information Rate : 0.6457 ##
P-Value [Acc > NIR] : <2e-16 ##
##                   Kappa : 0.8813
##
##  Mcnemar's Test P-Value : 0.7423
##
##             Sensitivity : 0.9615
##             Specificity : 0.9174
##          Pos Pred Value : 0.9550
##          Neg Pred Value : 0.9289
##              Prevalence : 0.6457
##          Detection Rate :  0.6208
##    Detection Prevalence : 0.6501 ##
       Balanced Accuracy : 0.9394
##
##        'Positive' Class : benign
##
```

b. Generate a visualization of the decision tree.

```r
fancyRpartPlot(tree1$finalModel, caption = "")
```



generated result using if-then and added the result in separate column.

```
n=1
for(n in 1:nrow(d)){
  x <- d$UniformCellSize[n]
  y <- d$UniformCellShape[n]
  if(x & y >= 2.5){
            result="malignant"
        } else if(x >= 2.5 & y < 2.5){
            result="Bening"
        }else {
            result="Bening"
        }
  d[n,"ifthenresult"]<- result
  n=n+1
}
```

```
head(d)
```

```
##   ClumpThickness UniformCellSize UniformCellShape MarginalAdhesion
## 1              5               1                1                1
## 2              5               4                4                5
## 3              3               1                1                1
## 4              6               8                8                1
## 5              4               1                1                3
## 6              8              10               10                8
```

```
##   EpithelialCellSize BareNuclei BlandChromatin NormalNucleoli Mitoses     Class
## 1                  2          1              3              1       1    benign
## 2                  7         10              3              2       1    benign
## 3                  2          2              3              1       1    benign
## 4                  3          4              3              7       1    benign
## 5                  2          1              3              1       1    benign
## 6                  7         10              9              7       1 malignant
##   ifthenresult
## 1       Bening
## 2    malignant
## 3       Bening
## 4    malignant
## 5       Bening
## 6    malignant
```

Problem 2

In this problem you will generate decision trees with a set of parameters. You will be using the storms data, a subset of the NOAA Atlantic hurricane database2 , which includes the positions and attributes of 198 tropical storms (potential hurricanes), measured every six hours during the lifetime of a storm. It is part of the dplyr library, so load the library and you will be able to access it. As a preprocessing step, view the data and make sure the target variable (category) is converted to a factor (as opposed to character string).

```
head(storms)
```

```
## # A tibble: 6 x 13
##   name   year month   day  hour   lat  long status       category  wind pressure
##   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct>           <dbl> <int>    <int>
## 1 Amy    1975     6    27     0  27.5 -79   tropical de~       NA    25     1013
## 2 Amy    1975     6    27     6  28.5 -79   tropical de~       NA    25     1013
## 3 Amy    1975     6    27    12  29.5 -79   tropical de~       NA    25     1013
## 4 Amy    1975     6    27    18  30.5 -79   tropical de~       NA    25     1013
## 5 Amy    1975     6    28     0  31.5 -78.8 tropical de~       NA    25     1012
## 6 Amy    1975     6    28     6  32.4 -78.7 tropical de~       NA    25     1012
## # i 2 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

```
storm_data <- storms
```

```
summary(storm_data)
```

```
##      name                year          month             day       
##  Length:19066       Min.   :1975   Min.   : 1.000   Min.   : 1.00  
##  Class :character   1st Qu.:1993   1st Qu.: 8.000   1st Qu.: 8.00  
##  Mode  :character   Median :2004   Median : 9.000   Median :16.00  
##                     Mean   :2002   Mean   : 8.699   Mean   :15.78  
##                     3rd Qu.:2012   3rd Qu.: 9.000   3rd Qu.:24.00  
##                     Max.   :2021   Max.   :12.000   Max.   :31.00  
##                                                                    
##       hour             lat             long                          status    
##  Min.   : 0.000   Min.   : 7.00   Min.   :-109.30   tropical storm      6684
```

```
##  1st Qu.: 5.000   1st Qu.:18.40   1st Qu.: -78.70   hurricane              4684
##  Median :12.000   Median :26.60   Median : -62.25   tropical depression:3525
##  Mean   : 9.094   Mean   :26.99   Mean   : -61.52   extratropical          2068
##  3rd Qu.:18.000   3rd Qu.:33.70   3rd Qu.: -45.60   other low              1405
##  Max.   :23.000   Max.   :70.70   Max.   :  13.50   subtropical storm  :  292
##                                                     (Other)            :  408
##     category          wind            pressure      tropicalstorm_force_diameter
##  Min.   :1.000   Min.   : 10.00   Min.   : 882.0   Min.   :   0.0
##  1st Qu.:1.000   1st Qu.: 30.00   1st Qu.: 987.0   1st Qu.:   0.0
##  Median :1.000   Median : 45.00   Median :1000.0   Median : 110.0
##  Mean   :1.898   Mean   : 50.02   Mean   : 993.6   Mean   : 146.3
##  3rd Qu.:3.000   3rd Qu.: 65.00   3rd Qu.:1007.0   3rd Qu.: 220.0
##  Max.   :5.000   Max.   :165.00   Max.   :1024.0   Max.   :1440.0
##  NA's   :14382                                     NA's   :9512
##  hurricane_force_diameter
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 14.81
##  3rd Qu.:  0.00
##  Max.   :300.00
##  NA's   :9512
```

```r
str(storm_data)
```

```
## tibble [19,066 x 13] (S3: tbl_df/tbl/data.frame)
##  $ name                        : chr [1:19066] "Amy" "Amy" "Amy" "Amy" ...
##  $ year                        : num [1:19066] 1975 1975 1975 1975 1975 ...
##  $ month                       : num [1:19066] 6 6 6 6 6 6 6 6 6 6 ...
##  $ day                         : int [1:19066] 27 27 27 27 28 28 28 28 29 29 ...
##  $ hour                        : num [1:19066] 0 6 12 18 0 6 12 18 0 6 ...
##  $ lat                         : num [1:19066] 27.5 28.5 29.5 30.5 31.5 32.4 33.3 34 34.4 34 ...
##  $ long                        : num [1:19066] -79 -79 -79 -79 -78.8 -78.7 -78 -77 -75.8 -74.8 ...
##  $ status                      : Factor w/ 9 levels "disturbance",..: 7 7 7 7 7 7 7 7 8 8 ...
##  $ category                    : num [1:19066] NA NA NA NA NA NA NA NA NA NA ...
##  $ wind                        : int [1:19066] 25 25 25 25 25 25 25 30 35 40 ...
##  $ pressure                    : int [1:19066] 1013 1013 1013 1013 1012 1012 1011 1006 1004 1002 ...
##  $ tropicalstorm_force_diameter: int [1:19066] NA NA NA NA NA NA NA NA NA NA ...
##  $ hurricane_force_diameter    : int [1:19066] NA NA NA NA NA NA NA NA NA NA ...
```

```r
storm_data$category = as.factor(storm_data$category)
str(storm_data$category)
```

```
## Factor w/ 5 levels "1","2","3","4",..: NA NA NA NA NA NA NA NA NA NA ...
```

```r
storm_data = na.omit(storm_data)
```

a. Build a decision tree using the following hyperparameters, maxdepth=2, minsplit=5 and minbucket=3. Be careful to use the right method of training so that you are not automatically tuning the cp parameter, but you are controlling the aforementioned parameters specifically. Use cross validation to report your accuracy score. These parameters will result in a relatively small tree.

Set hyper parameters to controls minsplit, maxdepth, and minbucket

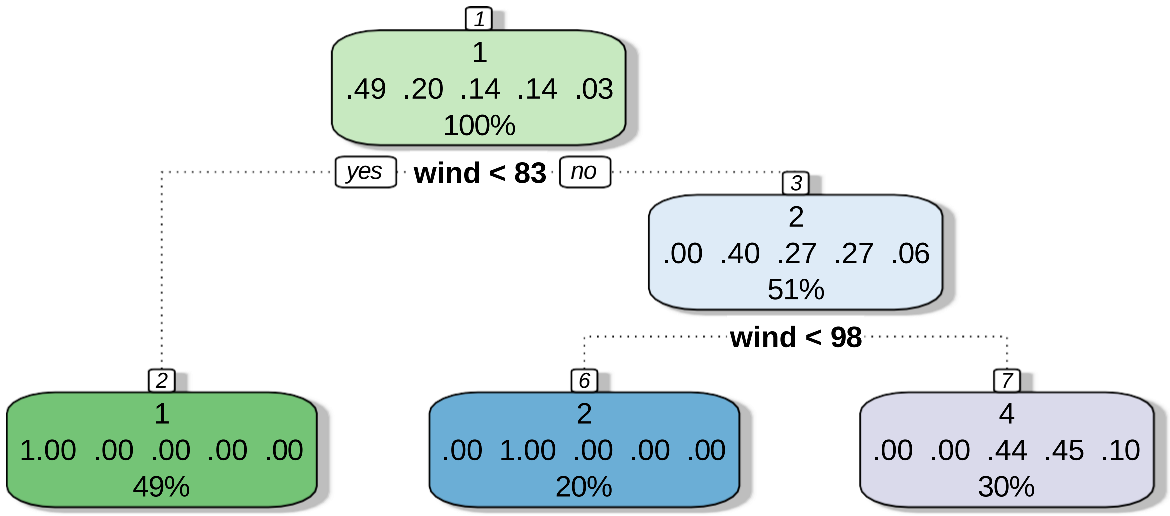
```r
hypers = rpart.control(minsplit =  5, maxdepth = 2, minbucket = 3)
```

We can see the performance of our model.

```r
#fit the model
tree_storm <- train(category ~., data = storm_data, control = hypers, trControl =
    train_control, method = "rpart1SE")

#evaluate
tree_storm
```

```
## CART
##
## 2051 samples
##   12 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1846, 1846, 1847, 1845, 1846, 1846, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8337486  0.7530905
```

```r
fancyRpartPlot(tree_storm$finalModel, caption = "")
```




```
pred_tree_storm = predict(tree_storm,storm_data)
confusionMatrix(storm_data$category,pred_tree_storm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4    5
##          1 1013    0    0    0    0
##          2    0  414    0    0    0
##          3    0    0    0  277    0
##          4    0    0    0  283    0
##          5    0    0    0   64    0
##
## Overall Statistics
##
##                Accuracy : 0.8337
##                  95% CI : (0.8169, 0.8496)
##     No Information Rate : 0.4939
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7531
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000   1.0000       NA   0.4535       NA
## Specificity            1.0000   1.0000   0.8649   1.0000   0.9688
## Pos Pred Value         1.0000   1.0000       NA   1.0000       NA
## Neg Pred Value         1.0000   1.0000       NA   0.8071       NA
## Prevalence             0.4939   0.2019   0.0000   0.3042   0.0000
## Detection Rate         0.4939   0.2019   0.0000   0.1380   0.0000
## Detection Prevalence   0.4939   0.2019   0.1351   0.1380   0.0312
## Balanced Accuracy      1.0000   1.0000       NA   0.7268       NA
```

b. To see how this performed with respect to the individual classes, we could use a confusion matrix. We also want to see if that aspect of performance is different on the train versus the test set. Create a train/test partition. Train on the training set. By making predictions with that model on the train set and on the test set separately, use the outputs to create two separate confusion matrices, one for each partition. Remember, we are testing if the model built with the training data performs differently on data used to train it (train set) as opposed to new data (test set). Compare the confusion matrices and report which classes it has problem classifying. Do you think that both are performing similarly and what does that suggest about overfitting for the model?

partitioning of data.

```r
index = createDataPartition(y=storm_data$category, p=0.7, list=FALSE)
```

```r
train_set = storm_data[index,]
test_set = storm_data[-index,]
```

```r
#fit the model
tree3 <- train(category ~., data = train_set, control = hypers, trControl =
    train_control, method = "rpart1SE")

tree3
```

```
## CART
##
## 1438 samples
##   12 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1294, 1295, 1295, 1294, 1295, 1293, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8338124  0.7532243
```

```r
pred_tree_train = predict(tree3,train_set)
pred_tree_test = predict(tree3,test_set)
```

```r
confusionMatrix(train_set$category,pred_tree_train)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1 710   0   0   0   0
##          2   0 290   0   0   0
##          3   0   0   0 194   0
##          4   0   0   0 199   0
##          5   0   0   0  45   0
##
## Overall Statistics
##
##                Accuracy : 0.8338
##                  95% CI : (0.8135, 0.8527)
##     No Information Rate : 0.4937
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7532
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000   1.0000       NA   0.4543       NA
## Specificity            1.0000   1.0000   0.8651   1.0000  0.96871
## Pos Pred Value         1.0000   1.0000       NA   1.0000       NA
## Neg Pred Value         1.0000   1.0000       NA   0.8071       NA
## Prevalence             0.4937   0.2017   0.0000   0.3046  0.00000
## Detection Rate         0.4937   0.2017   0.0000   0.1384  0.00000
## Detection Prevalence   0.4937   0.2017   0.1349   0.1384  0.03129
## Balanced Accuracy      1.0000   1.0000       NA   0.7272       NA
```

```
confusionMatrix(test_set$category,pred_tree_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1 303   0   0   0   0
##          2   0 124   0   0   0
##          3   0   0   0  83   0
##          4   0   0   0  84   0
##          5   0   0   0  19   0
##
## Overall Statistics
##
##                Accuracy : 0.8336
##                  95% CI : (0.8017, 0.8622)
##     No Information Rate : 0.4943
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7528
##
```

```
##    Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000   1.0000       NA   0.4516       NA
## Specificity            1.0000   1.0000   0.8646   1.0000    0.969
## Pos Pred Value         1.0000   1.0000       NA   1.0000       NA
## Neg Pred Value         1.0000   1.0000       NA   0.8072       NA
## Prevalence             0.4943   0.2023   0.0000   0.3034    0.000
## Detection Rate         0.4943   0.2023   0.0000   0.1370    0.000
## Detection Prevalence   0.4943   0.2023   0.1354   0.1370    0.031
## Balanced Accuracy      1.0000   1.0000       NA   0.7258       NA
```

From above two confusion matrices , we can see the accuracy is almost same for predicting both on train and set set which indicates there is no overfitting. The model misclassify to predict some intances of class 3 and 5.

Problem 3

This is will be an extension of Problem 2, using the same data and class. Here you will build many decision trees, manually tuning the parameters to gain intuition about the tradeoffs and how these tree parameters affect the complexity and quality of the model. The goal is to find the best tree model, which means it should be accurate but not too complex that the model overfits the training data. We will achieve this by using multiple sets of parameters and creating a graph of accuracy versus complexity for the training and the test sets (refer to the tutorial). This problem may require a significant amount of effort because you will need to train a substantial number of trees (at least 10).

a. Partition your data into 80% for training and 20% for the test data set

```
index = createDataPartition(y=storm_data$category, p=0.8, list=FALSE)
train_set = storm_data[index,]
test_set = storm_data[-index,]
```

```
str(test_set)
```

```
## tibble [407 x 13] (S3: tbl_df/tbl/data.frame)
##  $ name                        : chr [1:407] "Alex" "Charley" "Danielle" "Danielle" ...
##  $ year                        : num [1:407] 2004 2004 2004 2004 2004 ...
##  $ month                       : num [1:407] 8 8 8 8 8 8 8 8 8 8 ...
##  $ day                         : int [1:407] 5 14 15 15 17 18 27 28 30 30 ...
##  $ hour                        : num [1:407] 0 12 6 18 6 0 6 12 6 18 ...
##  $ lat                         : num [1:407] 38.5 32.3 14.1 15.2 21.7 25.9 14.2 17.2 19 19.4 ...
##  $ long                        : num [1:407] -66 -79.7 -30.8 -33.5 -39.6 -40.6 -47.8 -51.6 -56.8 -59
##  $ status                      : Factor w/ 9 levels "disturbance",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ category                    : Factor w/ 5 levels "1","2","3","4",..: 3 1 1 2 2 1 1 3 3 3 ...
##  $ wind                        : int [1:407] 105 65 75 85 90 75 75 105 100 110 ...
##  $ pressure                    : int [1:407] 957 988 981 975 970 981 980 958 958 948 ...
##  $ tropicalstorm_force_diameter: int [1:407] 225 140 195 195 150 150 135 180 260 280 ...
##  $ hurricane_force_diameter    : int [1:407] 70 40 45 50 40 40 30 50 70 90 ...
##  - attr(*, "na.action")= 'omit' Named int [1:17015] 1 2 3 4 5 6 7 8 9 10 ...
##   ..- attr(*, "names")= chr [1:17015] "1" "2" "3" "4" ...
```

b. Train at least 10 trees using different sets of parameters, through you made need more. Create the graph described above such that you can identify the inflection point where the tree is overfitting and pick a high-quality decision tree. Your strategy should be to make at least one very simple model and at least one very complex model and work towards the center by changing different parameters. Generate a table that contains all of the parameters (maxdepth, minsplit, minbucket, etc) used along with the number of nodes created, and the training and testing set accuracy values. The number of rows will be equal to the number of sets of parameters used. You will use the data in the table to generate the graph. The final results to be reported for this problem are the table and graph.

Implementing this question as I learned from the tutorial 6.

Tree 1

```
# Initialize cross validation
train_control = trainControl(method = "cv", number = 10)

# tree 1
hypers = rpart.control(minsplit =  2, maxdepth = 1, minbucket = 2)
tree1 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree1, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree1, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree1$finalModel$frame)

comp_tbl <- data.frame("Nodes" = nodes, "TrainAccuracy" = a_train, "TestAccuracy" = a_test,
                       "MaxDepth" = 1, "Minsplit" = 2, "Minbucket" = 2)
```

Tree 2

```
hypers = rpart.control(minsplit =  5, maxdepth = 2, minbucket = 5)
tree2 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree2, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree2, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree2$finalModel$frame)
```

```r
comp_tbl   <-   rbind(comp_tbl,c(nodes,a_train,a_test,2,5,5))
```

Tree 3

```r
hypers = rpart.control(minsplit =   50, maxdepth = 3, minbucket = 50)
tree3 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree3, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree3, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree3$finalModel$frame)



comp_tbl   <-   rbind(comp_tbl,c(nodes,a_train,a_test,3,50,50))
```

Tree 4

```r
hypers = rpart.control(minsplit =   100, maxdepth = 4, minbucket = 100)
tree4 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree4, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree4, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree4$finalModel$frame)



comp_tbl   <-   rbind(comp_tbl,c(nodes,a_train,a_test,4,100,100))
```

Tree 5

```r
hypers = rpart.control(minsplit =   100, maxdepth = 5, minbucket = 100)
tree5 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")
```

```
# Training set
pred_tree <- predict(tree5, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree5, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree5$finalModel$frame)



comp_tbl  <-  rbind(comp_tbl,c(nodes,a_train,a_test,5,100,100))
```

Tree 6

```
hypers = rpart.control(minsplit =  1000, maxdepth = 4, minbucket = 1000)
tree6 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree6, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree6, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree6$finalModel$frame)



comp_tbl  <-  rbind(comp_tbl,c(nodes,a_train,a_test,4,1000,1000))
```

Tree 7

```
hypers = rpart.control(minsplit =  2000, maxdepth = 5, minbucket = 2000)
tree7 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree7, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree7, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
```

```r
a_test <- cfm_test$overall[1]
nodes <- nrow(tree7$finalModel$frame)

comp_tbl  <-  rbind(comp_tbl,c(nodes,a_train,a_test,5,2000,2000))
```

Tree 8

```r
hypers = rpart.control(minsplit =  5000, maxdepth = 8, minbucket = 5000)
tree8 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree8, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree8, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree8$finalModel$frame)



comp_tbl  <-  rbind(comp_tbl,c(nodes,a_train,a_test,8,5000,5000))
```

Tree 9

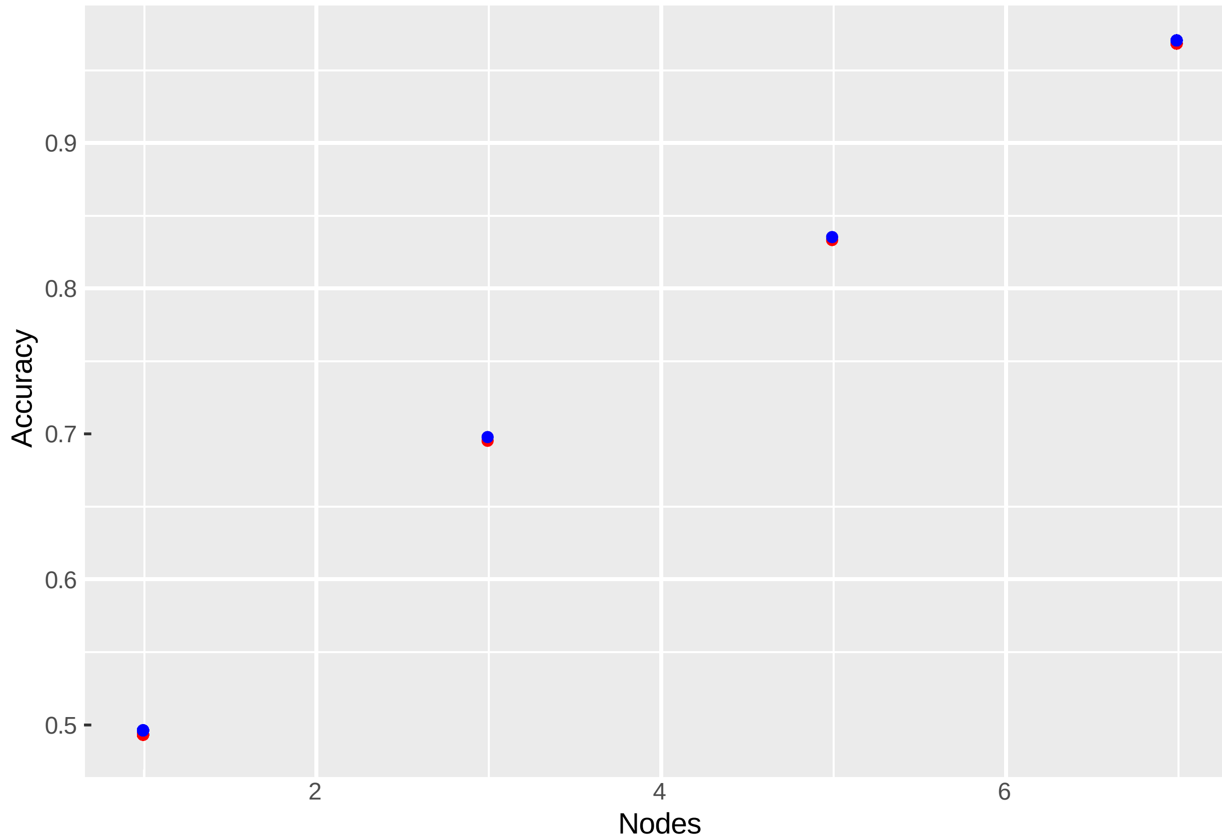
```r
hypers = rpart.control(minsplit =  10000, maxdepth = 12, minbucket = 10000)
tree9 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree9, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree9, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree9$finalModel$frame)



comp_tbl  <-  rbind(comp_tbl,c(nodes,a_train,a_test,12,10000,10000))
```

Tree 10

```r
hypers = rpart.control(minsplit =   10000, maxdepth = 25, minbucket = 10000)
tree10 <- train(category ~., data = train_set, control = hypers,
                  trControl = train_control, method = "rpart1SE")

# Training set
pred_tree <- predict(tree10, train_set)
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test set
pred_tree <- predict(tree10, test_set)
cfm_test <- confusionMatrix(test_set$category, pred_tree)

a_train <- cfm_train$overall[1]
a_test <- cfm_test$overall[1]
nodes <- nrow(tree10$finalModel$frame)



comp_tbl <- rbind(comp_tbl,c(nodes,a_train,a_test,25,10000,10000))

comp_tbl
```

```
##            Nodes TrainAccuracy TestAccuracy MaxDepth Minsplit Minbucket
## Accuracy       3     0.6952555    0.6977887        1        2         2
## 2              5     0.8333333    0.8353808        2        5         5
## 3              7     0.9683698    0.9705160        3       50        50
## 4              7     0.9683698    0.9705160        4      100       100
## 5              7     0.9683698    0.9705160        5      100       100
## 6              1     0.4933090    0.4963145        4     1000      1000
## 7              1     0.4933090    0.4963145        5     2000      2000
## 8              1     0.4933090    0.4963145        8     5000      5000
## 9              1     0.4933090    0.4963145       12    10000     10000
## 10             1     0.4933090    0.4963145       25    10000     10000
```

```r
#plot the graph
ggplot(comp_tbl,  aes(x=Nodes))  +
       geom_point(aes(y = TrainAccuracy), color = "red") +
       geom_point(aes(y = TestAccuracy), color="blue") +
       ylab("Accuracy")
```



c. Identify the final choice of model, list it parameters and evaluate with a the confusion matrix to make sure that it gets balanced performance over classes. Also get a better accuracy estimate for this tree using cross validation.

From the above, we can see the Model Tree3 has the highest accuracy with maxdepth = 3, minsplit = 50 and minbucket =50.

```
#tree 3
hypers = rpart.control(minsplit =  50, maxdepth = 3, minbucket = 50)
tree3 <- train(category ~., data = train_set, control = hypers, trControl = train_control,  method = "rp

tree3
```

```
## CART
##
## 1644 samples
##   12 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1479, 1480, 1480, 1479, 1480, 1478, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9683735  0.9530949
```

```r
# Test set
pred_tree <- predict(tree3, test_set)
confusionMatrix(test_set$category, pred_tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1 202   0   0   0   0
##          2   0  82   0   0   0
##          3   0   0  55   0   0
##          4   0   0   0  56   0
##          5   0   0   0  12   0
##
## Overall Statistics
##
##                Accuracy : 0.9705
##                  95% CI : (0.9491, 0.9847)
##     No Information Rate : 0.4963
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9561
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000   1.0000   1.0000   0.8235       NA
## Specificity            1.0000   1.0000   1.0000   1.0000  0.97052
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000       NA
## Neg Pred Value         1.0000   1.0000   1.0000   0.9658       NA
## Prevalence             0.4963   0.2015   0.1351   0.1671  0.00000
## Detection Rate         0.4963   0.2015   0.1351   0.1376  0.00000
## Detection Prevalence   0.4963   0.2015   0.1351   0.1376  0.02948
## Balanced Accuracy      1.0000   1.0000   1.0000   0.9118       NA
```

Problem 4

In this problem you will identify the most important independent variables used in a classification model. Use the Bank_Modified.csv data. As a preprocessing step, remove the ID column and make sure to convert the target variable, approval, from a string to a factor.

```r
bank = read.csv("C:/Users/bunty/Desktop/funda/week_5/Bank_Modified.csv",
header = T, sep = ",")
```

```
##   X cont1 cont2 cont3 bool1 bool2 cont4 bool3 cont5 cont6 approval credit.score
## 1 1 30.83 0.000  1.25     t     t     1     f   202     0        +       664.60
## 2 2 58.67 4.460  3.04     t     t     6     f    43   560        +       693.88
## 3 3 24.50 0.500  1.50     t     f     0     f   280   824        +       621.82
## 4 4 27.83 1.540  3.75     t     t     5     t   100     3        +       653.97
```

```
## 5 5 20.17 5.625    1.71     t     f     0     f    120    0      +      670.26
## 6 6 32.08 4.000    2.50     t     f     0     t    360    0      +      672.16
##   ages
## 1   58
## 2   54
## 3   62
## 4   51
## 5   58
## 6   37
```

```
bank = select(bank , -c("X"))
```

converting to factor

```
bank$approval = as.factor(bank$approval)
str(bank$approval)
```

```
##  Factor w/ 2 levels "-","+": 2 2 2 2 2 2 2 2 2 2 ...
```

Removing NA's

```
bank = na.omit(bank)
bank[rowSums(is.na(bank)) > 0, ]
```

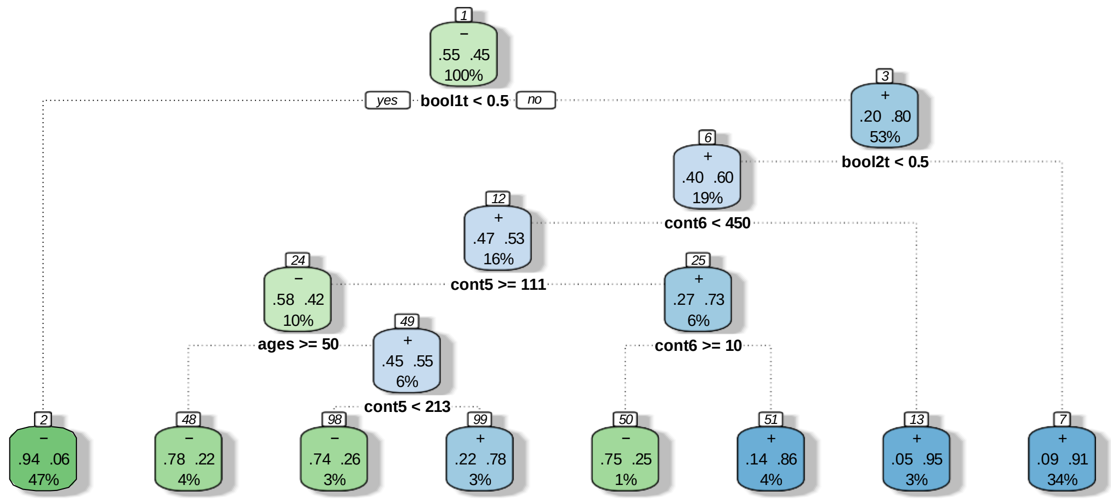
```
##  [1]  cont1        cont2        cont3        bool1        bool2
##  [6]  cont4        bool3        cont5        cont6        approval
## [11]  credit.score  ages
## <0 rows> (or 0-length row.names)
```

a. Build your initial decision tree model with minsplit=10 and maxdepth=20

```
#set hyperparameters
hypers = rpart.control(minsplit = 10, maxdepth = 20)
tree <- train(approval ~., data = bank, control = hypers, trControl =
    train_control, method = "rpart1SE")
tree
```

```
## CART
##
## 666 samples
##  11 predictor
##   2 classes: '-', '+'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 600, 599, 600, 599, 599, 600, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8604025  0.7178918
```

```r
fancyRpartPlot(tree$finalModel, caption= "")
```



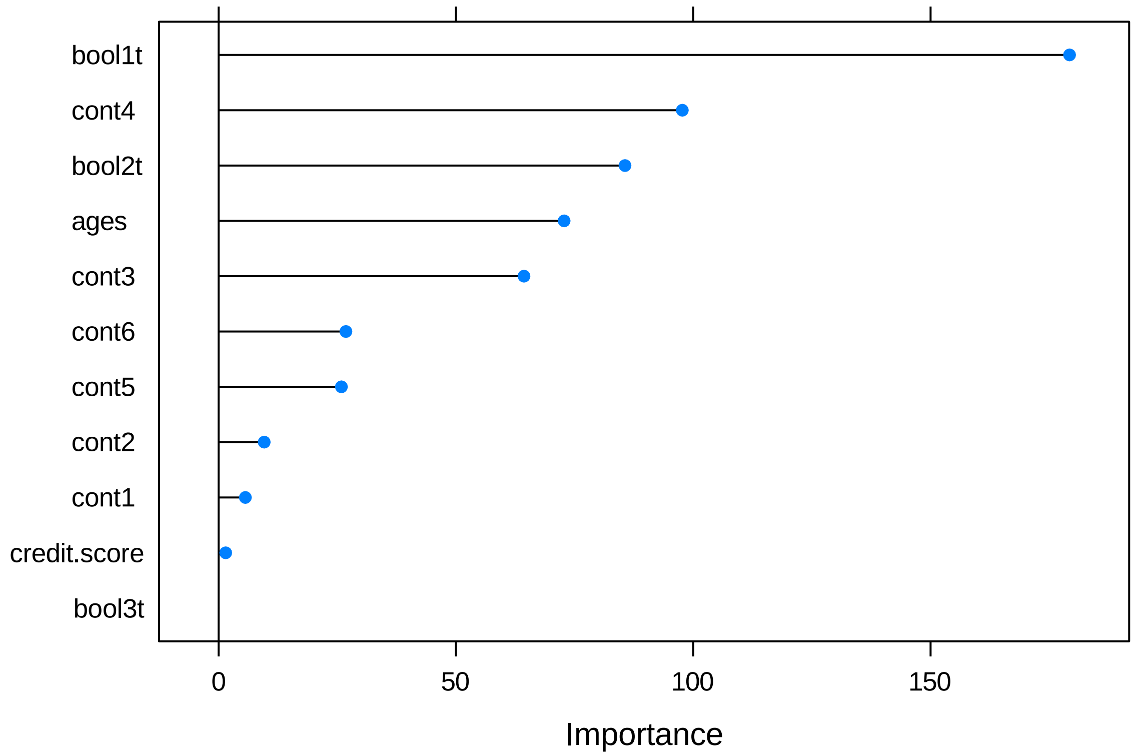
b. Run variable importance analysis on the model and print the result.

```r
var_imp <- varImp(tree, scale= FALSE)
var_imp
```

```
## rpart1SE variable importance
##
##               Overall
## bool1t        179.282
## cont4          97.700
## bool2t         85.622
## ages           72.800
## cont3          64.343
## cont6          26.828
## cont5          25.878
## cont2           9.620
## cont1           5.646
## credit.score    1.504
## bool3t          0.000
```

c. Generate a plot to visualize the variables by importance.

```
plot(var_imp)
```



d . Rebuild your model with the top six variables only, based on the variable relevance analysis. Did this change have an effect on the accuracy?

```
new_bank = select(bank,c("approval","bool1","cont4","bool2","ages","cont6","cont3"))
head(new_bank)
```

```
##   approval bool1 cont4 bool2 ages cont6 cont3
## 1        +     t     1     t   58     0  1.25
## 2        +     t     6     t   54   560  3.04
## 3        +     t     0     f   62   824  1.50
## 4        +     t     5     t   51     3  3.75
## 5        +     t     0     f   58     0  1.71
## 6        +     t     0     f   37     0  2.50
```

```
str(new_bank)
```

```
## 'data.frame':    666 obs. of  7 variables:
##  $ approval: Factor w/ 2 levels "-","+": 2 2 2 2 2 2 2 2 2 2 ...
##  $ bool1   : chr  "t" "t" "t" "t" ...
##  $ cont4   : int  1 6 0 5 0 0 0 0 0 0 ...
##  $ bool2   : chr  "t" "t" "f" "t" ...
##  $ ages    : int  58 54 62 51 58 37 47 67 61 62 ...
```

```
## $ cont6 : int 0 560 824 3 0 0 31285 1349 314 1442 ...
## $ cont3 : num 1.25 3.04 1.5 3.75 1.71 ...
## - attr(*, "na.action")= 'omit' Named int [1:24] 72 84 87 93 98 203 207 244 255 271 ...
## ..- attr(*, "names")= chr [1:24] "72" "84" "87" "93" ...
```

```
# split the data
index1 = createDataPartition(y=new_bank$approval, p=0.7, list=FALSE)
train_set = new_bank[index1,]
test_set = new_bank[-index1,]
```

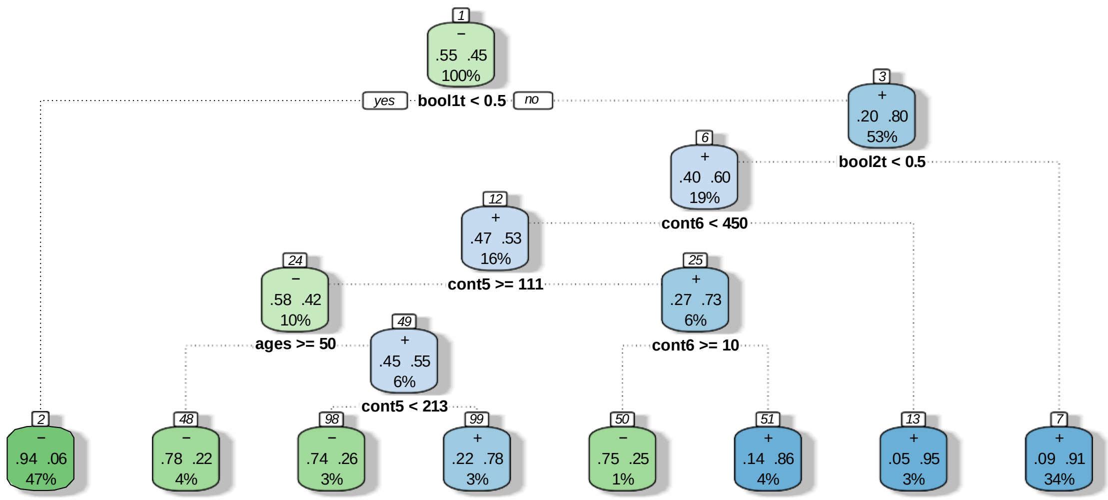
```
tree2 <- train(approval ~., data = train_set, method = "rpart1SE",
               trControl = train_control)
tree2
```

```
## CART
##
## 467 samples
##   6 predictor
##   2 classes: '-', '+'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 420, 421, 420, 420, 421, 421, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8586031  0.7149616
```

we can observe that accuracy has increased after selecting relevant predictors (first 6).

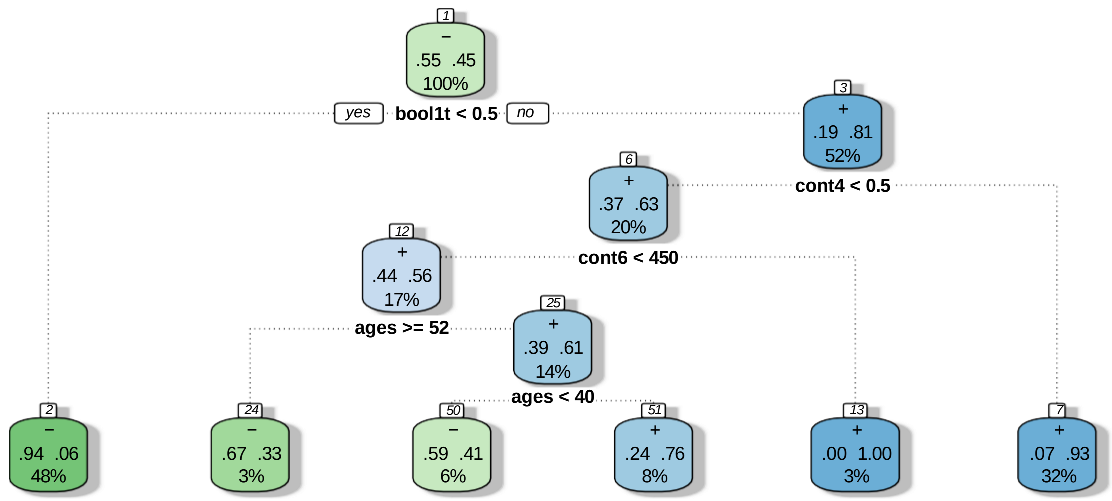
e. Visualize the trees from (a) and (d) and report if reducing the number of variables had an effect on the size of the tree?

```
fancyRpartPlot(tree$finalModel, caption= "tree1")
```



tree1

```
fancyRpartPlot(tree2$finalModel, caption= "tree2")
```



tree2

reducing the number of variables will reduce the graph.