Homework3

Kavana Manvi KrishnaMurthy

2024-05-06

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.

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(rpart)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(ggplot2)
breast_cancer <- read.csv(
  "/Users/kavanamanvi/Desktop/FDS/HW3/breast_cancer_updated.csv")
#Remove IDNumber coloumn
library(dplyr)
breast_cancer <- breast_cancer %>%   select(-IDNumber)
#exclue missing values
breast_cancer <- na.omit(breast_cancer)
#view summary
summary(breast_cancer)
```

```
##  ClumpThickness   UniformCellSize  UniformCellShape MarginalAdhesion
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.00
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.00
##  Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.00
##  Mean   : 4.442   Mean   : 3.151   Mean   : 3.215   Mean   : 2.83
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.00
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.00
##  EpithelialCellSize   BareNuclei     BlandChromatin   NormalNucleoli
##  Min.   : 1.000     Min.   : 1.000   Min.   : 1.000   Min.   : 1.00
##  1st Qu.: 2.000     1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.00
##  Median : 2.000     Median : 1.000   Median : 3.000   Median : 1.00
##  Mean   : 3.234     Mean   : 3.545   Mean   : 3.445   Mean   : 2.87
##  3rd Qu.: 4.000     3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 4.00
##  Max.   :10.000     Max.   :10.000   Max.   :10.000   Max.   :10.00
##     Mitoses           Class
##  Min.   : 1.000   Length:683
##  1st Qu.: 1.000   Class :character
##  Median : 1.000   Mode  :character
##  Mean   : 1.603
##  3rd Qu.: 1.000
##  Max.   :10.000
```

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.

```
# Evaluation method
train_control = trainControl(method = "cv", number = 10)
# Fit the model
bc_tree <- train(Class ~., data = breast_cancer, method = "rpart",
                 trControl = train_control)
#Evaluate fit
bc_tree
```

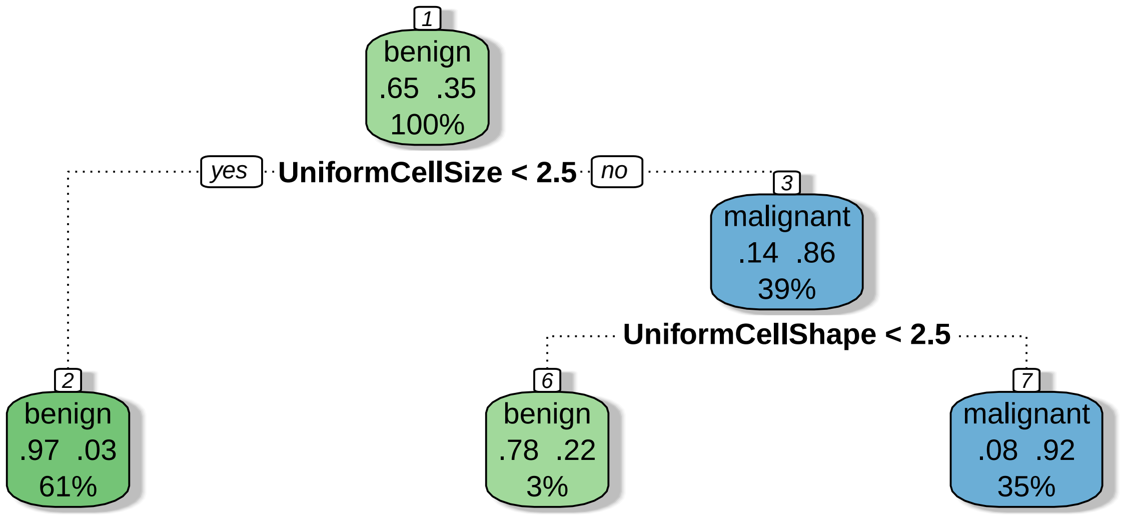
```
## CART
##
## 683 samples
##   9 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 615, 614, 614, 615, 615, 614, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02510460  0.9399585  0.8689443
##   0.05439331  0.9281279  0.8453237
##   0.79079498  0.7967554  0.4606411
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0251046.
```

b. Generate a visualization of the decision tree.

```r
library(rattle)
```

```
## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

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



c. Generate the full set of rules using IF-THEN statements.

IF UniformCellSize < 2.5 THEN Class = "benign"

IF UniformCellSize >= 2.5 AND UniformCellShape<2.5 THEN Class = "benign"

IF UniformCellSize >= 2.5 AND UniformCellShape>=2.5 THEN Class = "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).

```
data(storms)
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>
```

```
# Convert 'category' to a factor
raw_storms <-  as.data.frame (storms)
raw_storms$category <- as.factor(raw_storms$category)
# data pre-process
storms  <- raw_storms  %>% mutate (name =  as.factor (name))  %>% drop_na
summary(storms)
```

```
##        name           year          month             day
##  Maria    :  68   Min.   :2004   Min.   : 1.000   Min.   : 1.00
##  Danielle:  62   1st Qu.:2008   1st Qu.: 8.000   1st Qu.: 7.00
##  Irma     :  52   Median :2012   Median : 9.000   Median :15.00
##  Katia    :  49   Mean   :2013   Mean   : 8.951   Mean   :15.13
##  Helene   :  48   3rd Qu.:2018   3rd Qu.: 9.000   3rd Qu.:23.00
##  Jose     :  48   Max.   :2022   Max.   :12.000   Max.   :31.00
##  (Other) :1843
##       hour             lat             long
##  Min.   : 0.000   Min.   : 9.50   Min.   :-119.30
##  1st Qu.: 5.000   1st Qu.:19.10   1st Qu.: -78.20
##  Median :10.500   Median :25.40   Median : -65.20
##  Mean   : 9.112   Mean   :25.59   Mean   : -65.13
##  3rd Qu.:16.750   3rd Qu.:31.20   3rd Qu.: -53.35
##  Max.   :23.000   Max.   :48.80   Max.   : -14.10
##
##                     status      category      wind           pressure
##  hurricane              :2170   1:1083   Min.   : 65.00   Min.   : 882.0
##  disturbance            :   0   2: 434   1st Qu.: 70.00   1st Qu.: 954.0
##  extratropical          :   0   3: 291   Median : 85.00   Median : 969.0
##  other low              :   0   4: 297   Mean   : 88.53   Mean   : 965.6
##  subtropical depression:   0   5:  65   3rd Qu.:100.00   3rd Qu.: 981.0
##  subtropical storm      :   0            Max.   :160.00   Max.   :1001.0
##  (Other)                :   0
##  tropicalstorm_force_diameter hurricane_force_diameter
##  Min.   : 50.0                Min.   :  0.00
##  1st Qu.:175.0                1st Qu.: 35.00
##  Median :232.5                Median : 50.00
##  Mean   :254.1                Mean   : 62.87
##  3rd Qu.:310.0                3rd Qu.: 85.00
##  Max.   :870.0                Max.   :300.00
```

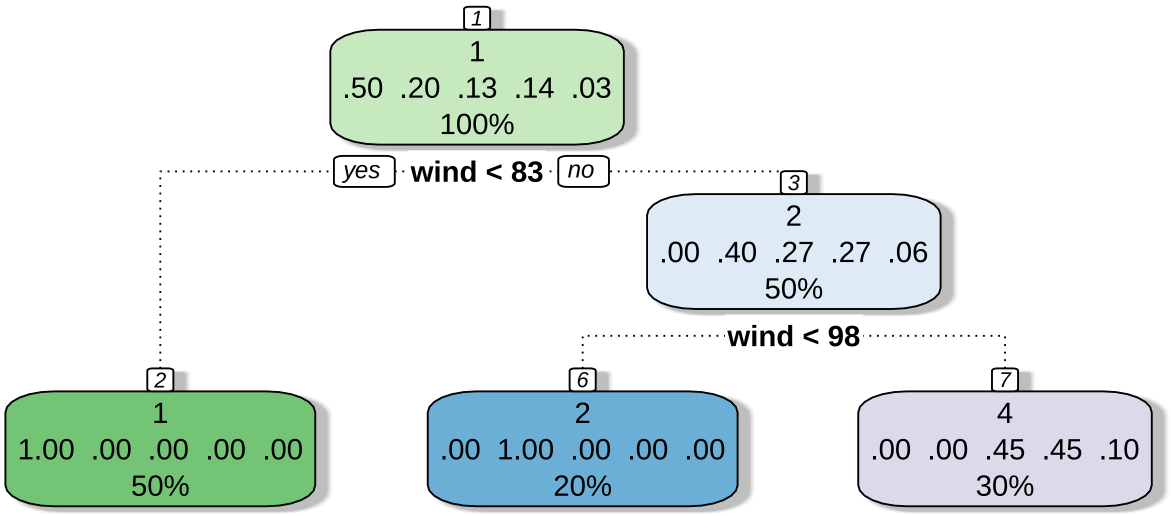
```
##
```

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.

```r
set.seed(94)
train_control = trainControl(method = "cv", number = 10)

#set hyperparameters
hypers = rpart.control(minsplit =  5, maxdepth = 2, minbucket = 3)

# Fit the model
tree2 <- train(category ~., data = storms, control = hypers,
               trControl = train_control, method = "rpart1SE")
tree2
```

```
## CART
##
## 2170 samples
##   12 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1953, 1953, 1953, 1953, 1954, 1952, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8359475  0.7550503
```

```r
# Visualize your decision tree
fancyRpartPlot(tree2$finalModel, caption = "")
```



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?

Let's start by partitioning our data into 70% train and 30% test sets.

```r
# Partition the data
index = createDataPartition(y=storms$category, p=0.7, list=FALSE)
# Everything in the generated index list
train_set = storms[index,]
# Everything except the generated indices
test_set = storms[-index,]

set.seed(94)
train_control = trainControl(method = "cv", number = 10)

# Tree 1
hypers = rpart.control(minsplit =  5, maxdepth = 2, minbucket = 3)
tree1 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")
tree1
```

```
## CART
##
## 1521 samples
##   12 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1370, 1367, 1370, 1370, 1370, 1369, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8356584  0.7546652
```

```
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree1, train_set)
# Confusion Matrixcfm_train
cfm_train <- confusionMatrix(train_set$category, pred_tree)
cfm_train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1 759   0   0   0   0
##          2   0 304   0   0   0
##          3   0   0   0 204   0
##          4   0   0   0 208   0
##          5   0   0   0  46   0
##
## Overall Statistics
##
##                Accuracy : 0.8356
##                  95% CI : (0.816, 0.8539)
##     No Information Rate : 0.499
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7546
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity             1.000   1.0000       NA   0.4541       NA
## Specificity             1.000   1.0000   0.8659   1.0000  0.96976
## Pos Pred Value          1.000   1.0000       NA   1.0000       NA
## Neg Pred Value          1.000   1.0000       NA   0.8096       NA
## Prevalence              0.499   0.1999   0.0000   0.3011  0.00000
## Detection Rate          0.499   0.1999   0.0000   0.1368  0.00000
## Detection Prevalence    0.499   0.1999   0.1341   0.1368  0.03024
## Balanced Accuracy       1.000   1.0000       NA   0.7271       NA
```

```r
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree1, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)
cfm_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1 324   0   0   0   0
##          2   0 130   0   0   0
##          3   0   0   0  87   0
##          4   0   0   0  89   0
##          5   0   0   0  19   0
##
## Overall Statistics
##
##                Accuracy : 0.8367
##                  95% CI : (0.8059, 0.8643)
##     No Information Rate : 0.4992
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.756
##
##  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   0.4564       NA
## Specificity            1.0000   1.0000   0.8659   1.0000  0.97072
## Pos Pred Value         1.0000   1.0000       NA   1.0000       NA
## Neg Pred Value         1.0000   1.0000       NA   0.8107       NA
## Prevalence             0.4992   0.2003   0.0000   0.3005  0.00000
## Detection Rate         0.4992   0.2003   0.0000   0.1371  0.00000
## Detection Prevalence   0.4992   0.2003   0.1341   0.1371  0.02928
## Balanced Accuracy      1.0000   1.0000       NA   0.7282       NA
```

```r
# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree1$finalModel$frame)
```

The decision tree model built on the storms dataset shows consistent performance across the training and test sets. The overall accuracy of the model is similar for both sets, with the test set slightly outperforming the training set (83.67% vs. 83.56%). However, the model struggles with correctly classifying Category 4 storms, as indicated by a lower sensitivity for this class in both the training (45.41%) and test (45.64%) sets. Despite this challenge, the model demonstrates balanced accuracy, suggesting that it is not biased towards any particular class. Additionally, the similar performance on both sets indicates that the model is not overfitting, as it generalizes well to unseen data by capturing underlying patterns rather than noise from the training data.

##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

```
set.seed(124)
# Partition the data
index = createDataPartition(y=storms$category, p=0.8, list=FALSE)
# Everything in the generated index list
train_set = storms[index,]
nrow(train_set)
```

```
## [1] 1738
```

```
# Everything except the generated indices
test_set = storms[-index,]
nrow(test_set)
```

```
## [1] 432
```

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.

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

# Tree 0
hypers = rpart.control(minsplit =  1, maxdepth = 1, minbucket = 1)
tree0 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree0, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
```

```r
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree0, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree0$finalModel$frame)

# Form the table
comp_tbl <- data.frame("Nodes" = nodes, "TrainAccuracy" = a_train,
                       "TestAccuracy" = a_test,
                       "MaxDepth" = 1, "Minsplit" = 1, "Minbucket" = 1)

# 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
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree1, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree1, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree1$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 1, 2, 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
# Evaluate the fit with a confusion matrix
```

```r
pred_tree <- predict(tree2, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree2, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree2$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 2, 5, 5))


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

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree3, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree3, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree3$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 3, 50, 50))


#Tree4
hypers = rpart.control(minsplit = 5, maxdepth = 4, minbucket = 3)
tree4 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")
```

```
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree4, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree4, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree4$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 4, 5, 3))

# Tree 5
hypers = rpart.control(minsplit = 100, maxdepth = 4, minbucket = 100)
tree5 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree5, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree5, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree5$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 4, 200, 200))


# Tree 6
hypers = rpart.control(minsplit = 300, maxdepth = 8, minbucket = 300)
tree6 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree6, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree6, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree6$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 8, 300, 300))

# Tree 7
hypers = rpart.control(minsplit = 500, maxdepth = 30, minbucket = 500)
tree7 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree7, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree7, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree7$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 30, 500, 500))
```

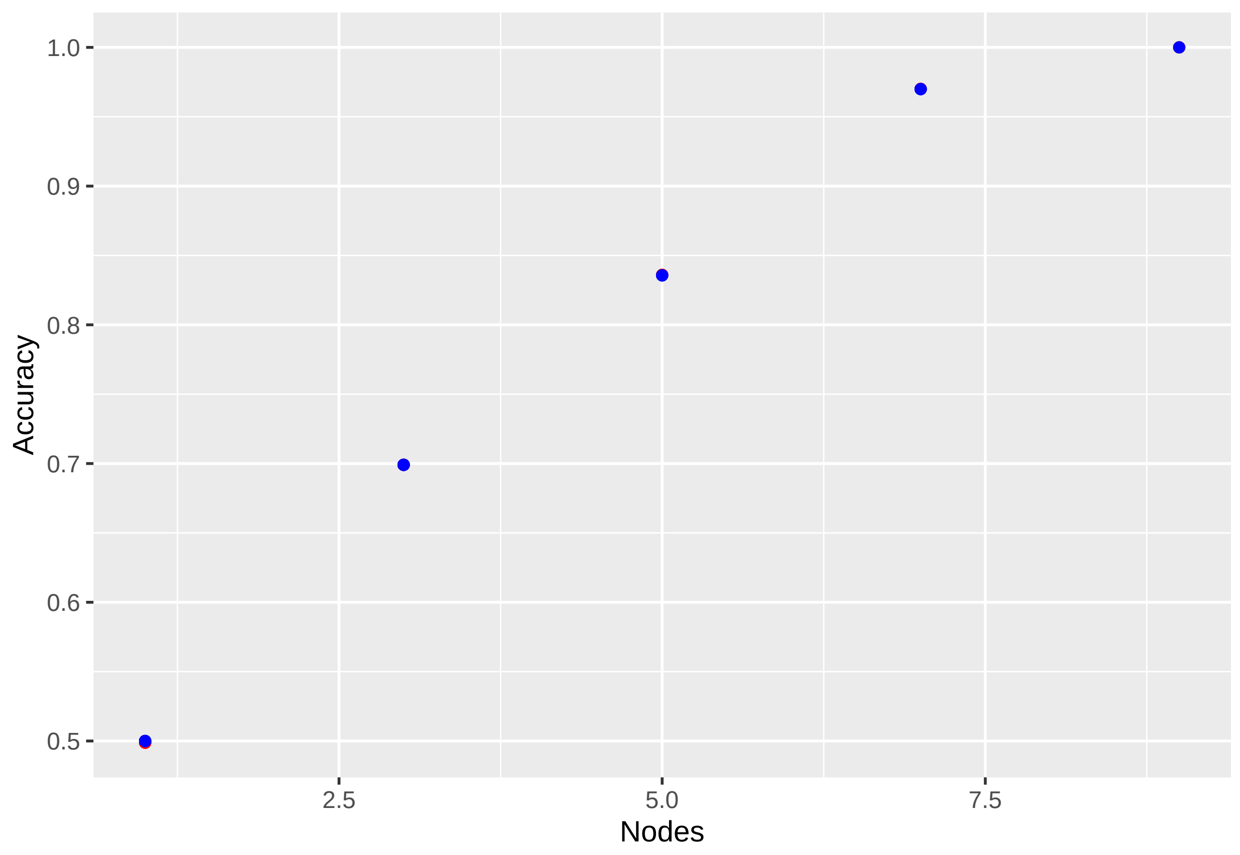
```r
# Tree 8
hypers = rpart.control(minsplit = 150, maxdepth = 10, minbucket = 800)
tree8 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree8, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree8, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree8$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 10, 150, 800))

# Tree 9
hypers = rpart.control(minsplit = 800, maxdepth = 12, minbucket = 1000)
tree9 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree9, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree9, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree9$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 20, 800, 1000))
```

```r
# Tree 10
hypers = rpart.control(minsplit = 1250, maxdepth = 16, minbucket = 1300)
tree10 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree10, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree10, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree10$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 16, 1250, 1300))

# Tree 11
hypers = rpart.control(minsplit = 1500, maxdepth = 20, minbucket = 1800)
tree11 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree11, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree11, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree11$finalModel$frame)

# Add rows to the table - Make sure the order is correct
```

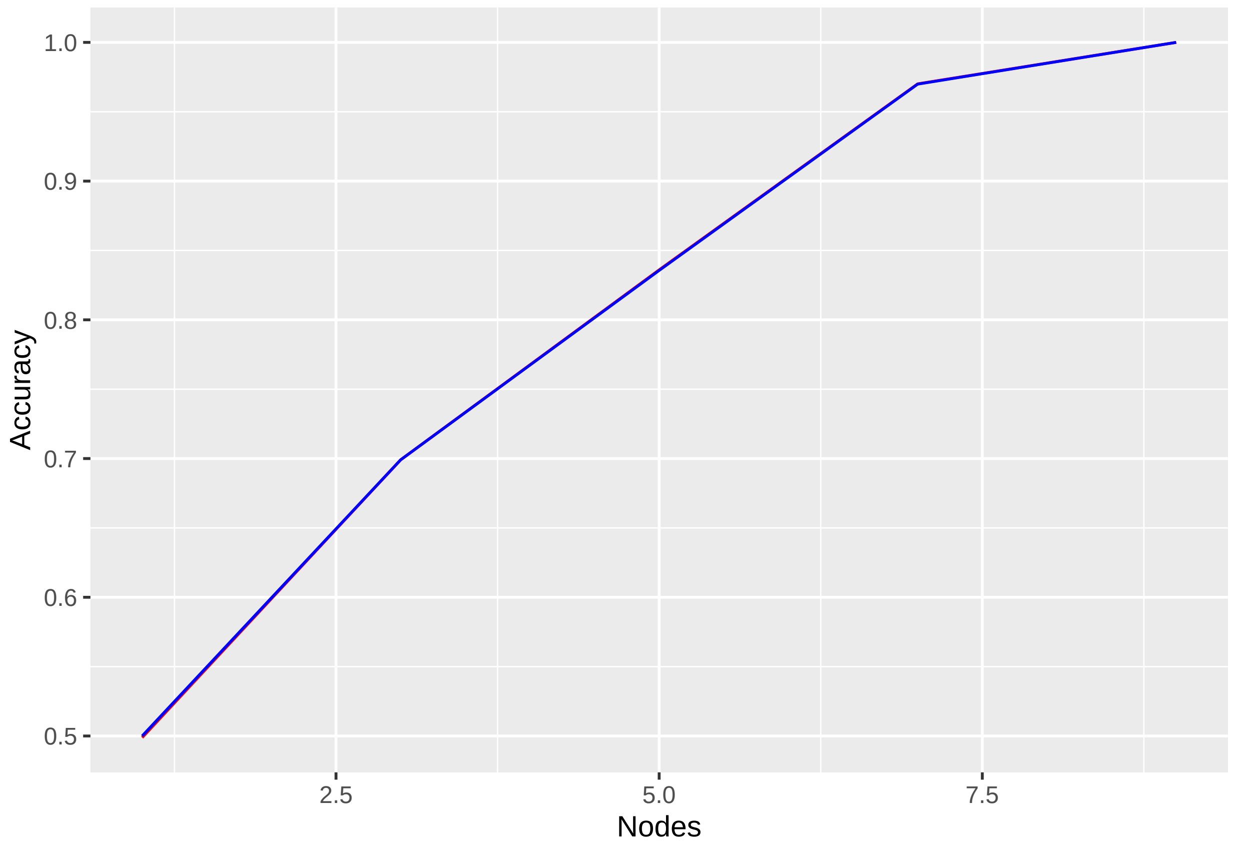
```r
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 20, 1500, 1800))

# Tree 12
hypers = rpart.control(minsplit = 2000, maxdepth = 30, minbucket = 2000)
tree12 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree12, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree12, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree12$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 30, 2000, 2000))
comp_tbl
```

```
##          Nodes TrainAccuracy TestAccuracy MaxDepth Minsplit Minbucket
## Accuracy     3     0.6990794    0.6990741        1        1         1
## 1            3     0.6990794    0.6990741        1        2         2
## 11           5     0.8360184    0.8356481        2        5         5
## 12           7     0.9700806    0.9699074        3       50        50
## 13           9     1.0000000    1.0000000        4        5         3
## 14           7     0.9700806    0.9699074        4      200       200
## 15           5     0.8360184    0.8356481        8      300       300
## 16           3     0.6990794    0.6990741       30      500       500
## 17           3     0.6990794    0.6990741       10      150       800
## 18           1     0.4988493    0.5000000       20      800      1000
## 19           1     0.4988493    0.5000000       16     1250      1300
## 110          1     0.4988493    0.5000000       20     1500      1800
## 111          1     0.4988493    0.5000000       30     2000      2000
```

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



```
# Visualize with line plot
ggplot(comp_tbl, aes(x=Nodes)) +
  geom_line(aes(y = TrainAccuracy), color = "red") +
  geom_line(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.

```
#Tree4
#Final model
set.seed(95)
train_control = trainControl(method = "cv", number = 10)

hypers = rpart.control(minsplit = 5, maxdepth = 4, minbucket = 3)
tree4 <- train(category ~., data = train_set, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree4, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$category, pred_tree)
cfm_train
```

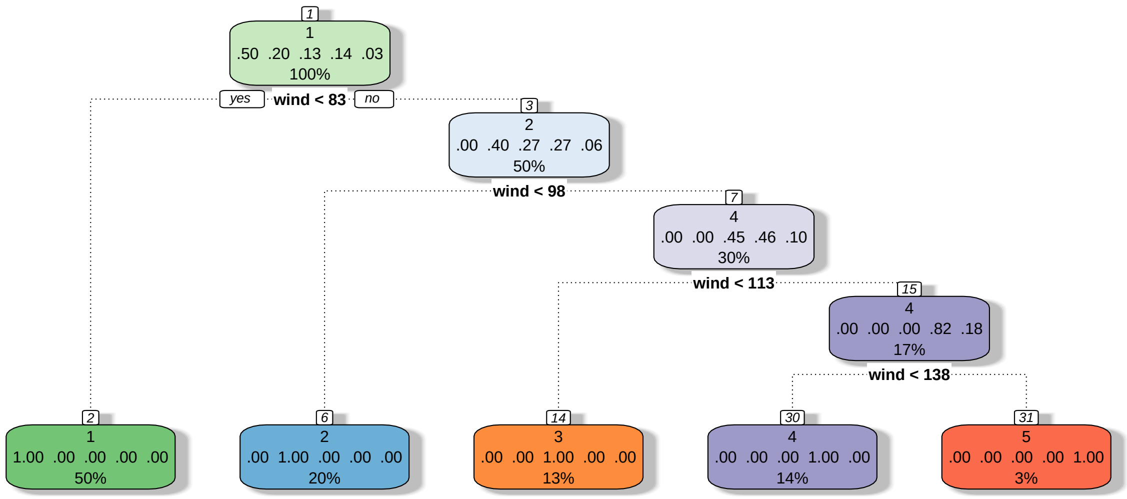
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1 867   0   0   0   0
##          2   0 348   0   0   0
##          3   0   0 233   0   0
##          4   0   0   0 238   0
##          5   0   0   0   0  52
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9979, 1)
##     No Information Rate : 0.4988
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  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   1.0000  1.00000
## Specificity            1.0000   1.0000   1.0000   1.0000  1.00000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000  1.00000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000  1.00000
## Prevalence             0.4988   0.2002   0.1341   0.1369  0.02992
## Detection Rate         0.4988   0.2002   0.1341   0.1369  0.02992
## Detection Prevalence   0.4988   0.2002   0.1341   0.1369  0.02992
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000  1.00000
```

```r
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree4, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$category, pred_tree)
cfm_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1 216   0   0   0   0
##          2   0  86   0   0   0
##          3   0   0  58   0   0
##          4   0   0   0  59   0
##          5   0   0   0   0  13
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9915, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity               1.0   1.0000   1.0000   1.0000  1.00000
## Specificity               1.0   1.0000   1.0000   1.0000  1.00000
## Pos Pred Value            1.0   1.0000   1.0000   1.0000  1.00000
## Neg Pred Value            1.0   1.0000   1.0000   1.0000  1.00000
## Prevalence                0.5   0.1991   0.1343   0.1366  0.03009
## Detection Rate            0.5   0.1991   0.1343   0.1366  0.03009
## Detection Prevalence      0.5   0.1991   0.1343   0.1366  0.03009
## Balanced Accuracy         1.0   1.0000   1.0000   1.0000  1.00000
```

```r
# Visualize your decision tree
fancyRpartPlot(tree4$finalModel, caption = "")
```



##Problem 4 (25 points) 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("/Users/kavanamanvi/Desktop/FDS/HW3/Bank_Modified.csv")
bank <-  as.data.frame (bank)
#Covert target variable approval as a factor
bank$approval <- as.factor(bank$approval)
#remove missing values
```

```
bank <- na.omit(bank)
#remove ID column
bank <- select(bank, -X)
#view summary
summary(bank)
```

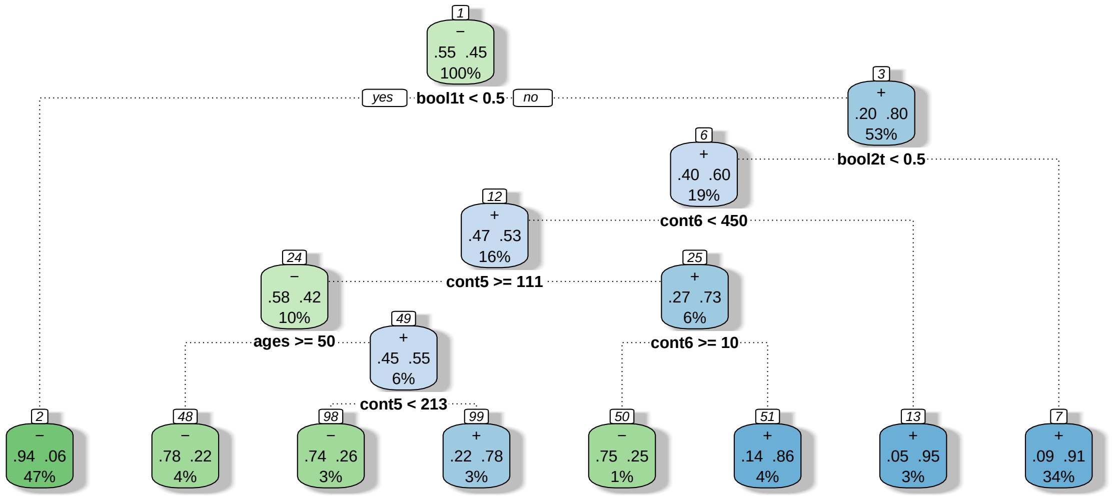
```
##      cont1           cont2            cont3           bool1
##  Min.   :13.75   Min.   : 0.000   Min.   : 0.000   Length:666
##  1st Qu.:22.60   1st Qu.: 1.010   1st Qu.: 0.165   Class :character
##  Median :28.50   Median : 2.750   Median : 1.000   Mode  :character
##  Mean   :31.57   Mean   : 4.798   Mean   : 2.222
##  3rd Qu.:38.25   3rd Qu.: 7.207   3rd Qu.: 2.585
##  Max.   :80.25   Max.   :28.000   Max.   :28.500
##     bool2               cont4           bool3               cont5
##  Length:666         Min.   : 0.000   Length:666         Min.   :   0.00
##  Class :character   1st Qu.: 0.000   Class :character   1st Qu.:  75.25
##  Mode  :character   Median : 0.000   Mode  :character   Median : 160.00
##                     Mean   : 2.459                      Mean   : 182.12
##                     3rd Qu.: 3.000                      3rd Qu.: 271.00
##                     Max.   :67.000                      Max.   :2000.00
##      cont6          approval  credit.score        ages
##  Min.   :     0.0   -:367    Min.   :585.1   Min.   :17.00
##  1st Qu.:     0.0   +:299    1st Qu.:666.4   1st Qu.:31.00
##  Median :     5.0            Median :697.1   Median :38.00
##  Mean   :   998.6            Mean   :696.3   Mean   :39.67
##  3rd Qu.:   399.0            3rd Qu.:726.4   3rd Qu.:47.00
##  Max.   :100000.0            Max.   :806.0   Max.   :84.00
```

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

```
# Set hyperparameters
hypers = rpart.control(minsplit =  10, maxdepth = 20)
#Fit the model
train_control = trainControl(method = "cv", number = 10)
bank_tree <- train(approval ~., data = bank, control = hypers,
                   trControl = train_control, method = "rpart1SE")
bank_tree
```

```
## CART
##
## 666 samples
##  11 predictor
##   2 classes: '-', '+'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 600, 599, 599, 600, 599, 599, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8602442  0.7185038
```

```r
#Visualize the descision tree
fancyRpartPlot(bank_tree$finalModel, caption = "")
```



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

```r
# View the predictors present in the model
colnames(bank)
```

```
##  [1] "cont1"        "cont2"        "cont3"        "bool1"        "bool2"
##  [6] "cont4"        "bool3"        "cont5"        "cont6"        "approval"
## [11] "credit.score" "ages"
```

```r
# View the variable importance scores
var_imp <- varImp(bank_tree, scale = FALSE)
var_imp
```

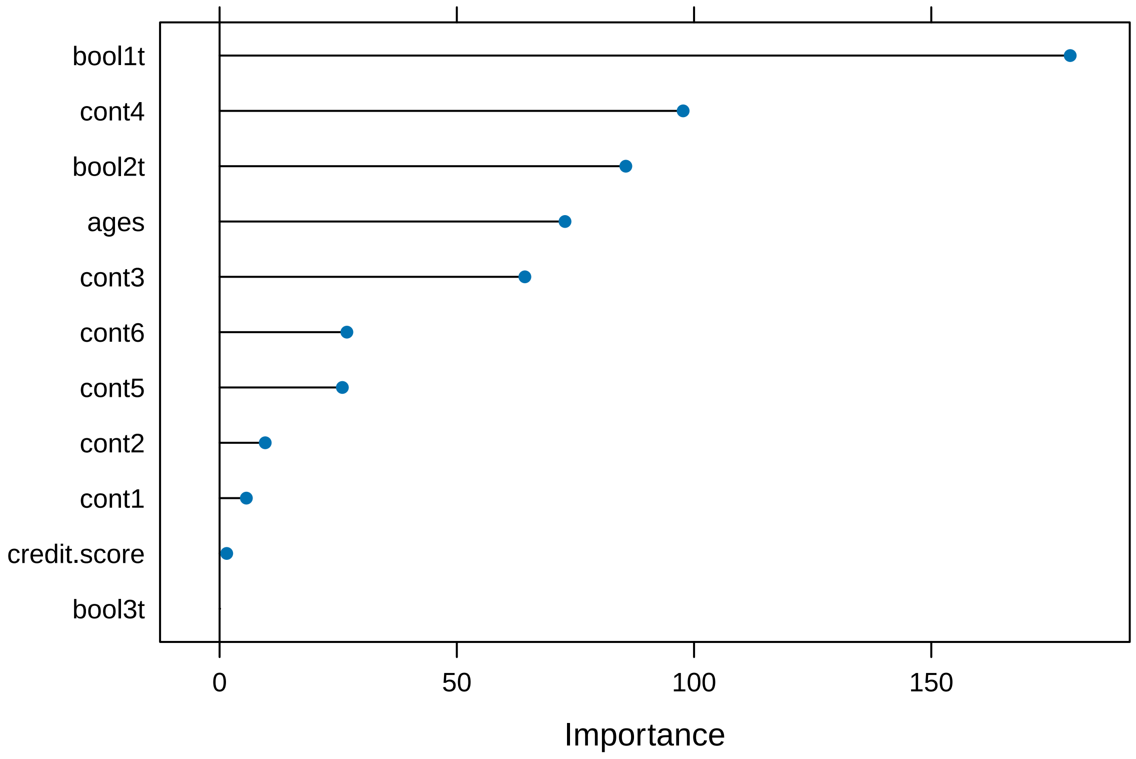
```
## rpart1SE variable importance
##
##              Overall
## bool1t       179.282
## cont4         97.700
## bool2t        85.622
## ages          72.800
## cont3         64.343
## cont6          26.828
## cont5          25.878
## cont2           9.620
## cont1           5.646
## credit.score  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

```
# Estimate variable importance
importance <- varImp(bank_tree, scale=FALSE)
# Summarize importance
print(importance)
```

```
## 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.

```
# Visualize
plot(importance)
```



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?

```
# Extract the top six variable names
top_vars <- c("bool1", "cont4", "bool2", "ages", "cont3", "cont6", "approval")
# Create a subset of the bank dataset with only the specified columns
bank_top_var <- bank[, top_vars]
head(bank_top_var)
```

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

```
# Build the model with top six variables
bank_tree2 <- train(approval ~., data = bank_top_var, control = hypers,
                    trControl = train_control, method = "rpart1SE")
bank_tree2
```

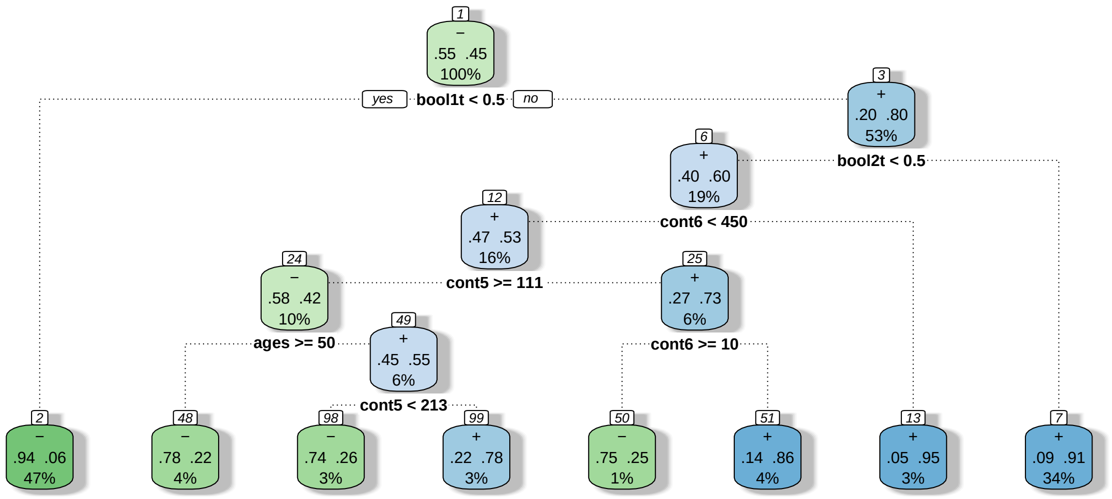
```
## CART
##
```

```
## 666 samples
##   6 predictor
##   2 classes: '-', '+'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 599, 601, 600, 599, 599, 599, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8693734  0.7367019
```

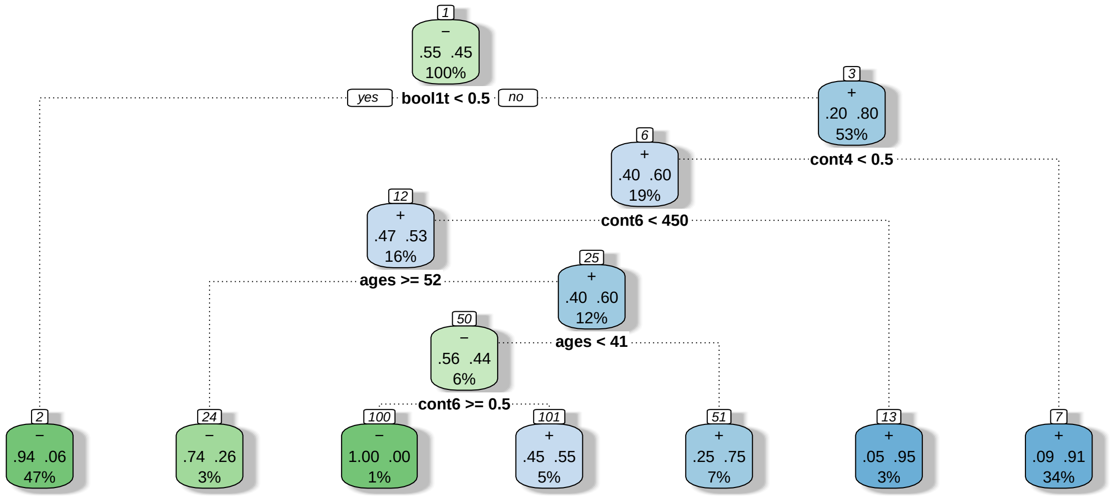
The accuracy of second model, bank_tree2(0.8543872) is lower than the first model, bank_tree(0.8768431) after rebuilding the model with the top six variablesThe first model, bank_tree might be overly complex and fitting too closely to the training data, capturing noise rather than genuine patterns. By using only the top six variables in bank_tree2, we are simplifying the model and reducing the risk of overfitting. However, this simplification could also lead to a loss of some predictive ability.

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?

```
#Visualize the first descision tree
fancyRpartPlot(bank_tree$finalModel, caption = "")
```



```r
#Visualize the second descision tree
fancyRpartPlot(bank_tree2$finalModel, caption = "")
```



Reducing the number of variables had a noticeable impact on the size of the decision tree in the image. The tree on the left (a) is significantly larger than the tree on the right (b). The left tree exhibits more branches and leaves, indicating that it considered a greater number of variables in its decision-making process. The labels adjacent to the leaves of the decision tree represent the proportion of data points that belong to that particular category. For example, in the left tree, the far-left branch leads to a leaf labeled "2180" with a "100%" label. This indicates that all 2180 data points following that branch are classified into the same category.

In summary, the decision tree with fewer variables is simpler and more compact compared to the decision tree with more variables. This suggests that reducing the number of variables can result in a more straightforward and potentially faster decision tree. However, it's essential to note that a smaller tree may sacrifice some accuracy.