DSC 441- Homework 3

Sanchal Dhurve

1.Problem 1 (15 points):

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.

```r
# Load necessary libraries
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.4.2
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.4.2

## Loading required package: lattice
```

```r
# Load the dataset
breastCancer <- read.csv("C:/Users/SDHURVE/Documents/breast_cancer_updated.csv")
head(breastCancer)
```

```
##   IDNumber ClumpThickness UniformCellSize UniformCellShape MarginalAdhesion
## 1  1000025              5               1                1                1
## 2  1002945              5               4                4                5
## 3  1015425              3               1                1                1
## 4  1016277              6               8                8                1
## 5  1017023              4               1                1                3
## 6  1017122              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
```

```r
summary(breastCancer)
```

```
##     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
##
```

```r
# Remove IDNumber column
breastCancer$IDNumber <- NULL

# Remove rows with NA values
breastCancer <- na.omit(breastCancer)

summary(breastCancer)
```

```
##  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.

```
# Convert 'Class' column to factor (since it's categorical)
breastCancer$Class <- as.factor(breastCancer$Class)

# Set seed for reproducibility
set.seed(123)

# Define 10-fold cross-validation
trainControlSet1 <- trainControl(method = "cv", number = 10)
```

```
# Train decision tree model using rpart
tree1 <- train(Class ~ ., data = breastCancer, method = "rpart", trControl = trainControlSet1)
tree1
```

```
## CART
##
## 683 samples
##   9 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 614, 614, 615, 615, 615, 615, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02510460  0.9415388  0.8714556
##   0.05439331  0.9283461  0.8428032
##   0.79079498  0.8567136  0.6383030
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0251046.
```

```
# Predict using the trained model (return class labels)
predictingMalignance <- predict(tree1, breastCancer, type = "raw")

# Generate and print confusion matrix
confMatrix <- confusionMatrix(predictingMalignance, breastCancer$Class)
confMatrix
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       424        17
##   malignant     20       222
```

```
##
##                Accuracy : 0.9458
##                  95% CI : (0.9261, 0.9616)
##     No Information Rate : 0.6501
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8813
##
##  Mcnemar's Test P-Value : 0.7423
##
##             Sensitivity : 0.9550
##             Specificity : 0.9289
##          Pos Pred Value : 0.9615
##          Neg Pred Value : 0.9174
##              Prevalence : 0.6501
##          Detection Rate : 0.6208
##    Detection Prevalence : 0.6457
##       Balanced Accuracy : 0.9419
##
##        'Positive' Class : benign
##
```

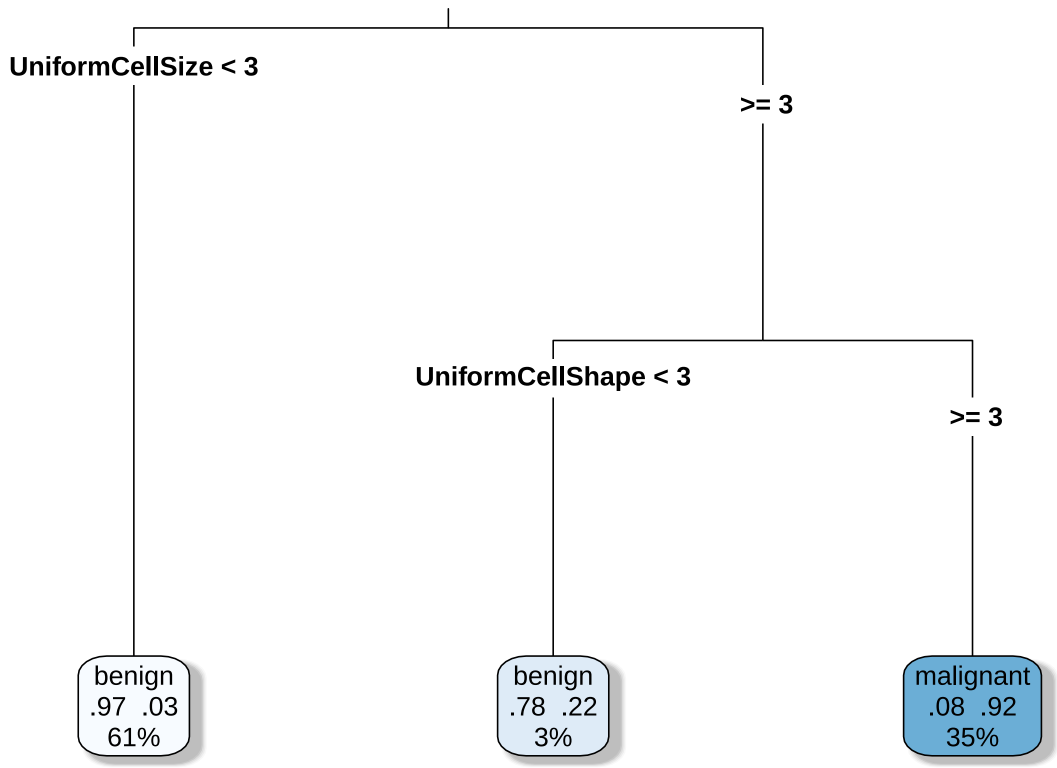
The decision tree model achieved an accuracy of 94.58% using 10-fold cross-validation, demonstrating strong predictive performance. The Kappa value of 0.8813 indicates a high level of agreement between predicted and actual classifications. Sensitivity (95.50%) and specificity (92.89%) confirm that the model effectively distinguishes between benign and malignant cases. These results suggest that the model is highly reliable for predicting breast cancer malignancy.

b.Generate a visualization of the decision tree.

```r
# Load necessary libraries
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.4.2
```

```r
# Plot the decision tree
rpart.plot(tree1$finalModel, type = 3, extra = 104, fallen.leaves = TRUE, box.palette = "Blues", shadow
```



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

```r
# Extract rules from the decision tree
tree_rules <- path.rpart(tree1$finalModel, nodes = row.names(tree1$finalModel$frame))
```

```
##
##  node number: 1
##    root
##
##  node number: 2
##    root
##    UniformCellSize< 2.5
##
##  node number: 3
##    root
##    UniformCellSize>=2.5
##
##  node number: 6
##    root
##    UniformCellSize>=2.5
##    UniformCellShape< 2.5
##
##  node number: 7
##    root
##    UniformCellSize>=2.5
##    UniformCellShape>=2.5
```

```r
# Convert the rules into IF-THEN statements
for (i in seq_along(tree_rules)) {
  # Extract class prediction from the model
  class_label <- ifelse(tree1$finalModel$frame$yval[i] == 1, "benign", "malignant")

  # Skip the root node rule (already covered)
  if (length(tree_rules[[i]]) > 1) {
    cat("\nRule", i, ":\nIF ")
    cat(paste(tree_rules[[i]][-1], collapse = " AND "))  # Remove 'root'
    cat(" THEN Class =", class_label, "\n")
  }
}
```

```
##
## Rule 2 :
## IF UniformCellSize< 2.5 THEN Class = benign
##
## Rule 3 :
## IF UniformCellSize>=2.5 THEN Class = malignant
##
## Rule 4 :
## IF UniformCellSize>=2.5 AND UniformCellShape< 2.5 THEN Class = benign
##
## Rule 5 :
## IF UniformCellSize>=2.5 AND UniformCellShape>=2.5 THEN Class = malignant
```

2.Problem 2 (15 points):

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).

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
# Load necessary libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(rpart)
library(caret)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.4.2

## Loading required package: tibble

## 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
# Load the storms dataset
data("storms")

# Copy dataset to stormData for modification
stormData <- storms

# Print first few rows
head(stormData)
```

```
## # 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>
```

```r
# Convert 'category' to a factor
stormData$category <- as.factor(stormData$category)

# Print summary before cleaning
cat("Summary before excluding the NA values\n\n")
```

```
## Summary before excluding the NA values
```

```r
print(summary(stormData))
```

```
##      name                year           month              day
##  Length:19537       Min.   :1975   Min.   : 1.000   Min.   : 1.00
##  Class :character   1st Qu.:1994   1st Qu.: 8.000   1st Qu.: 8.00
##  Mode  :character   Median :2004   Median : 9.000   Median :16.00
##                     Mean   :2003   Mean   : 8.706   Mean   :15.73
```

```
##                     3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:24.00
##                     Max.   :2022   Max.   :12.000   Max.   :31.00
##
##       hour              lat             long                           status
##  Min.   : 0.000   Min.   : 7.00   Min.   :-136.90   tropical storm      :6830
##  1st Qu.: 5.000   1st Qu.:18.30   1st Qu.: -78.80   hurricane           :4803
##  Median :12.000   Median :26.60   Median : -62.30   tropical depression:3569
##  Mean   : 9.101   Mean   :27.01   Mean   : -61.56   extratropical       :2151
##  3rd Qu.:18.000   3rd Qu.:33.80   3rd Qu.: -45.50   other low           :1453
##  Max.   :23.000   Max.   :70.70   Max.   :  13.50   subtropical storm  : 298
##                                                     (Other)             : 433
##  category          wind           pressure      tropicalstorm_force_diameter
##  1    : 2548   Min.   : 10.00   Min.   : 882.0   Min.   :   0.0
##  2    :  993   1st Qu.: 30.00   1st Qu.: 986.0   1st Qu.:   0.0
##  3    :  593   Median : 45.00   Median :1000.0   Median : 110.0
##  4    :  553   Mean   : 50.05   Mean   : 993.5   Mean   : 147.9
##  5    :  116   3rd Qu.: 65.00   3rd Qu.:1007.0   3rd Qu.: 220.0
##  NA's:14734   Max.   :165.00   Max.   :1024.0   Max.   :1440.0
##                                                  NA's   :9512
##  hurricane_force_diameter
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 14.92
##  3rd Qu.:  0.00
##  Max.   :300.00
##  NA's   :9512
```

```r
# Remove rows with any missing values
stormData <- stormData[complete.cases(stormData), ]

# Print summary after cleaning
cat("\nSummary after excluding the NA values\n\n")
```

```
##
## Summary after excluding the NA values
```

```r
print(summary(stormData))
```

```
##      name                year          month             day
##  Length:2170        Min.   :2004   Min.   : 1.000   Min.   : 1.00
##  Class :character   1st Qu.:2008   1st Qu.: 8.000   1st Qu.: 7.00
##  Mode  :character   Median :2012   Median : 9.000   Median :15.00
##                     Mean   :2013   Mean   : 8.951   Mean   :15.13
##                     3rd Qu.:2018   3rd Qu.: 9.000   3rd Qu.:23.00
##                     Max.   :2022   Max.   :12.000   Max.   :31.00
##
##       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
##
```

```r
# Train the decision tree with fixed hyperparameters
stormTreeUsingHyperParameters <- train(
  category ~ .,
  data = stormData,
  method = "rpart",
  trControl = trainControl(method = "cv", number = 10),  # 10-fold cross-validation
  tuneGrid = data.frame(cp = 0),  # Prevent automatic tuning of cp
  control = rpart.control(
    minsplit = 5,
    maxdepth = 2,
    minbucket = 3
  )
)

# Print model summary
cat("Summary of storm tree using HyperParameters\n\n")
```

```
## Summary of storm tree using HyperParameters
```

```r
print(stormTreeUsingHyperParameters)
```

```
## CART
##
## 2170 samples
##   12 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1952, 1954, 1952, 1952, 1953, 1953, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8359511  0.7550544
```

```
##
## Tuning parameter 'cp' was held constant at a value of 0
```

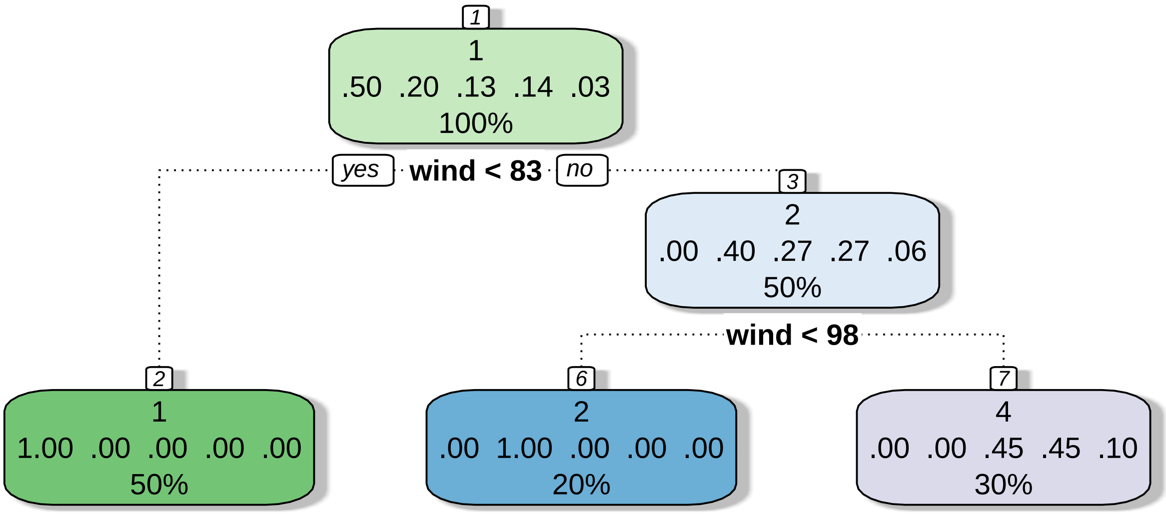
```r
# Predict on the full dataset
predictStormTree <- predict(stormTreeUsingHyperParameters, stormData, type = "raw")

# Generate confusion matrix
cat("\nConfusion Matrix for Storm data using hyperparameters\n")
```

```
##
## Confusion Matrix for Storm data using hyperparameters
```

```r
confMatrix <- confusionMatrix(predictStormTree, stormData$category)
print(confMatrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4    5
##          1 1083    0    0    0    0
##          2    0  434    0    0    0
##          3    0    0    0    0    0
##          4    0    0  291  297   65
##          5    0    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.8359
##                  95% CI : (0.8197, 0.8513)
##     No Information Rate : 0.4991
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.755
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000      1.0   0.0000   1.0000  0.00000
## Specificity            1.0000      1.0   1.0000   0.8099  1.00000
## Pos Pred Value         1.0000      1.0      NaN   0.4548      NaN
## Neg Pred Value         1.0000      1.0   0.8659   1.0000  0.97005
## Prevalence             0.4991      0.2   0.1341   0.1369  0.02995
## Detection Rate         0.4991      0.2   0.0000   0.1369  0.00000
## Detection Prevalence   0.4991      0.2   0.0000   0.3009  0.00000
## Balanced Accuracy      1.0000      1.0   0.5000   0.9050  0.50000
```

```r
# Plot the decision tree
fancyRpartPlot(
  stormTreeUsingHyperParameters$finalModel,
  caption = "Decision tree using hyperparameter values\n maxdepth=2, minsplit=5 and minbucket=3"
)
```



Decision tree using hyperparameter values
maxdepth=2, minsplit=5 and minbucket=3

The cross-validation accuracy score for this decision tree model is 83.59%. Since maxdepth=2, the resulting decision tree is relatively small, meaning it has low complexity while maintaining good predictive performance.

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?

```
# Load necessary libraries
library(caret)
library(rpart)
library(rattle)

# Set seed for reproducibility
set.seed(123)

# Split data into 70% training and 30% testing
trainIndex <- createDataPartition(y = stormData$category, p = 0.7, list = FALSE)
trainSet <- stormData[trainIndex, ]
testSet <- stormData[-trainIndex, ]
```

```r
# Print dataset sizes
cat("Number of rows in Train set:", nrow(trainSet), "\n")
```

```
## Number of rows in Train set: 1521
```

```r
cat("Number of rows in Test set:", nrow(testSet), "\n")
```

```
## Number of rows in Test set: 649
```

```r
# Train decision tree on the training set with correct hyperparameters
treeForTrainSet <- train(
  category ~ .,
  data = trainSet,
  method = "rpart",
  trControl = trainControl(method = "cv", number = 10),  # 10-fold cross-validation (Required for 2.a)
  tuneGrid = data.frame(cp = 0),  # No automatic cp tuning
  control = rpart.control(minsplit = 5, maxdepth = 2, minbucket = 3)
)

# Print model summary
cat("\nSummary of train set decision tree using HyperParameters:\n\n")
```

```
##
## Summary of train set decision tree using HyperParameters:
```

```r
print(treeForTrainSet)
```

```
## CART
##
## 1521 samples
##   12 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1369, 1369, 1369, 1368, 1369, 1368, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.835641  0.7546557
##
## Tuning parameter 'cp' was held constant at a value of 0
```

```r
# Make predictions on training set
predictTrainSet <- predict(treeForTrainSet, trainSet, type = "raw")

# Generate confusion matrix for training data
cat("\nConfusion Matrix for Train Set:\n")
```

```
##
## Confusion Matrix for Train Set:
```

```r
trainConfMatrix <- confusionMatrix(trainSet$category, predictTrainSet)
print(trainConfMatrix)
```

```
## 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
# Make predictions on test set
predictTestSet <- predict(treeForTrainSet, testSet, type = "raw")

# Generate confusion matrix for test data
cat("\nConfusion Matrix for Test Set:\n")
```

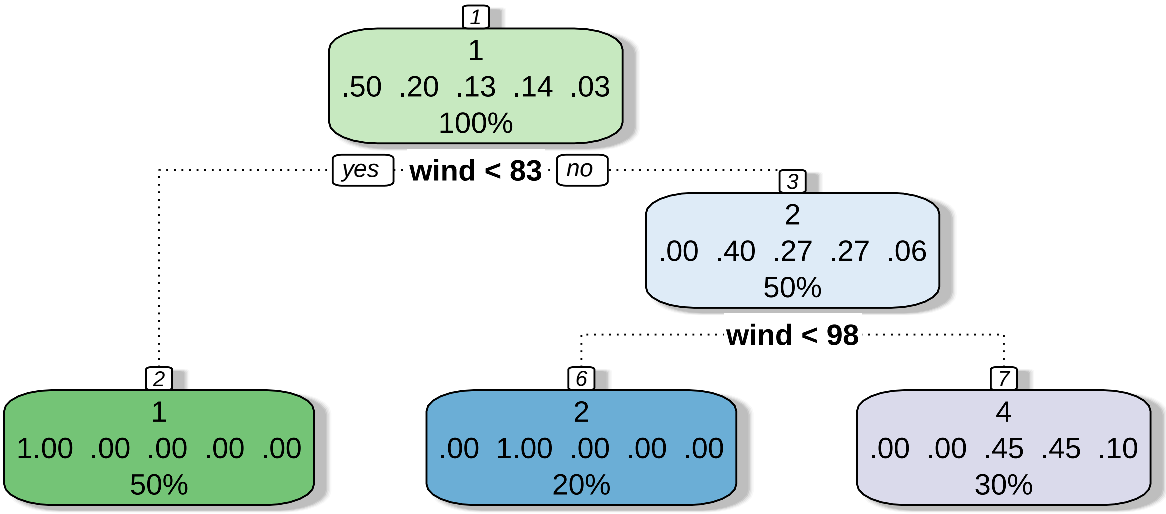
```
##
## Confusion Matrix for Test Set:
```

```r
testConfMatrix <- confusionMatrix(testSet$category, predictTestSet)
print(testConfMatrix)
```

```
## 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
# Plot the trained decision tree
fancyRpartPlot(
  treeForTrainSet$finalModel,
  caption = "Decision tree using hyperparameter values\n maxdepth=2, minsplit=5 and minbucket=3"
)
```



Decision tree using hyperparameter values
maxdepth=2, minsplit=5 and minbucket=3

The model performs well for classes 1 and 2 but struggles significantly with class 3, which it fails to classify entirely, and class 5. Class 4 also shows moderate misclassification. Despite these issues, the overall accuracy remains consistent between the training (83.56%) and test sets (83.67%), with similar Kappa values. Since there is no significant drop in performance, the model does not overfit and generalizes well, though improvements are needed to better classify the problematic classes.

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.

```
# Creating train and test data (80% train, 20% test)
set.seed(123)  # Set seed for reproducibility
trainIndex <- createDataPartition(y = stormData$category, p = 0.8, list = FALSE)

# Correct partitioning
trainSet <- stormData[trainIndex, ]  # 80% Train set
testSet <- stormData[-trainIndex, ]  # 20% Test set (complement of trainSet)

# Print data set sizes
cat("Number of rows in Train set:", nrow(trainSet), "\n")
```

```
## Number of rows in Train set: 1738
```

```r
cat("Number of rows in Test set:", nrow(testSet), "\n")
```

```
## Number of rows in Test set: 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.

```r
#Train Set 1
 treeForTrainSet <-
   train(
     category ~ .,
     data = trainSet,
     control = rpart.control(
       minsplit = 5,
       maxdepth = 2,
       minbucket = 3
     ),
     trControl = trainControl(method = "cv", number = 10),
     method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
 compare_table <-
   data.frame(
     "Nodes" = nrow(treeForTrainSet$finalModel$frame),
     "TrainAccuracy" = confusionMatrixForTrainSet$overall[1],
     "TestAccuracy" = confusionMatrixForTestSet$overall[1],
     "Minsplit" = 5,
     "MaxDepth" = 2,
     "Minbucket" = 3
   )

#Train Set 2
 treeForTrainSet <-
   train(
     category ~ .,
     data = trainSet,
     control = rpart.control(
 minsplit = 3,
      maxdepth = 2,
      minbucket = 3
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
```

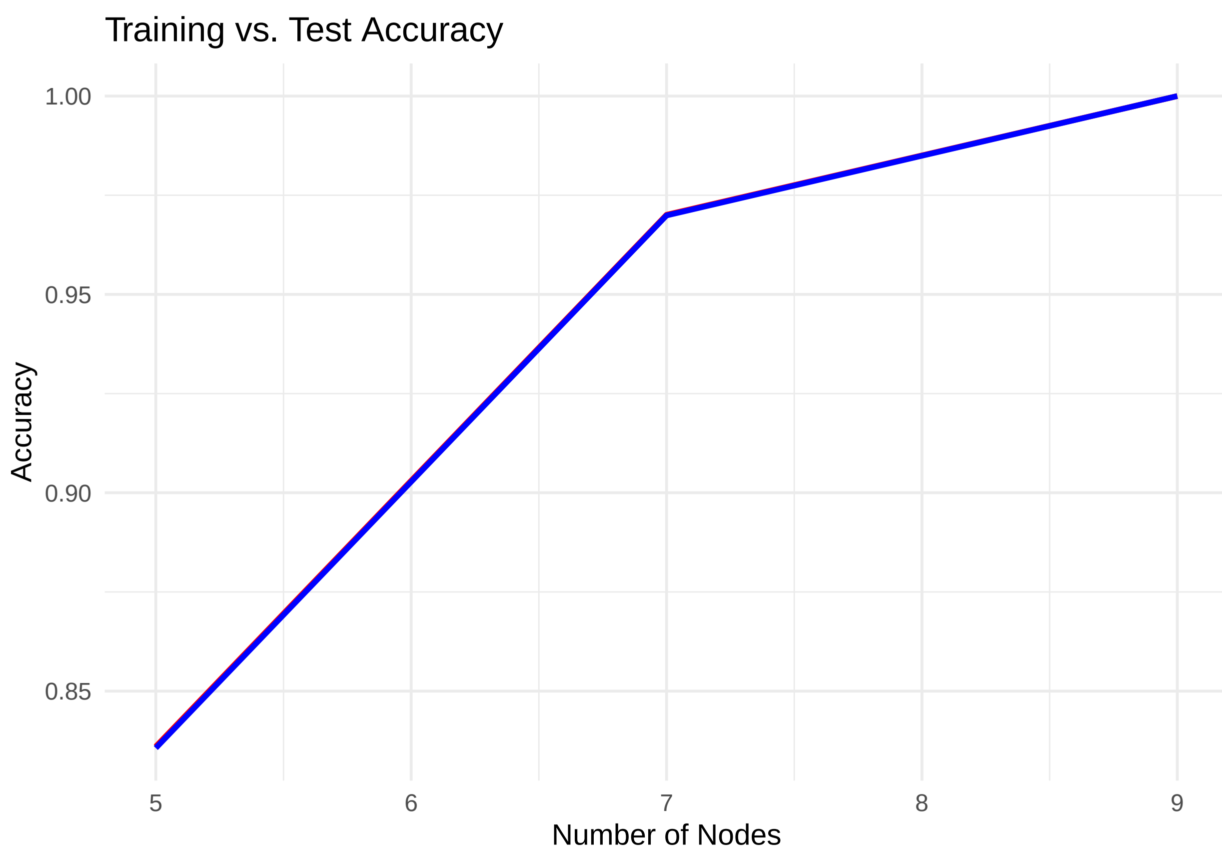
```
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
  compare_table %>% rbind(
    list(
      nrow(treeForTrainSet$finalModel$frame),
      confusionMatrixForTrainSet$overall[1],
      confusionMatrixForTestSet$overall[1],
      3,
      2,
      3
) )
#Train Set 3
treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 5,
      maxdepth = 3,
      minbucket = 5
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
  compare_table %>% rbind(
    list(
      nrow(treeForTrainSet$finalModel$frame),
      confusionMatrixForTrainSet$overall[1],
      confusionMatrixForTestSet$overall[1],
      5,
      3,
      5
) )
#Train Set 4
treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 6,
      maxdepth = 3,
      minbucket = 5
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
```

```
  compare_table %>% rbind(
    list(
      nrow(treeForTrainSet$finalModel$frame),
      confusionMatrixForTrainSet$overall[1],
      confusionMatrixForTestSet$overall[1],
      6,
      3,
      5
) )
#Train Set 5
 treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 10,
      maxdepth = 3,
      minbucket = 10
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
  compare_table %>% rbind(
    list(
      nrow(treeForTrainSet$finalModel$frame),
      confusionMatrixForTrainSet$overall[1],
      confusionMatrixForTestSet$overall[1],
      10,
      3,
      10
) )
#Train Set 6
treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 15,
      maxdepth = 10,
      minbucket = 19
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
  compare_table %>% rbind(
    list(
      nrow(treeForTrainSet$finalModel$frame),
```

```
      confusionMatrixForTrainSet$overall[1],
      confusionMatrixForTestSet$overall[1],
      15,
      10,
      19
) )
#Train Set 7
treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 25,
      maxdepth = 15,
      minbucket = 25
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
  compare_table %>% rbind(
    list(
      nrow(treeForTrainSet$finalModel$frame),
      confusionMatrixForTrainSet$overall[1],
      confusionMatrixForTestSet$overall[1],
      25,
      15,

25 )
)
#Train Set 8
treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 27,
      maxdepth = 17,
      minbucket = 25
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
  compare_table %>% rbind(
    list(
      nrow(treeForTrainSet$finalModel$frame),
      confusionMatrixForTrainSet$overall[1],
      confusionMatrixForTestSet$overall[1],
```

```
      27,
      17,
      25
) )
#Train Set 9
treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 21,
      maxdepth = 9,
      minbucket = 20
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
  compare_table %>% rbind(
    list(
      nrow(treeForTrainSet$finalModel$frame),
      confusionMatrixForTrainSet$overall[1],
      confusionMatrixForTestSet$overall[1],
      21,
      9,
      20
) )
#Train Set 10
treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 30,
      maxdepth = 17,
      minbucket = 30
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
)
confusionMatrixForTrainSet <- confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))
compare_table <-
  compare_table %>% rbind(
list(

nrow(treeForTrainSet$finalModel$frame),
       confusionMatrixForTrainSet$overall[1],
       confusionMatrixForTestSet$overall[1],
       30,
17,
```

```r
30 )
  )

library(ggplot2)

ggplot(compare_table, aes(x = Nodes)) +
  geom_line(aes(y = TrainAccuracy), color = "red", linewidth = 1) +
  geom_line(aes(y = TestAccuracy), color = "blue", linewidth = 1) +
  labs(y = "Accuracy", x = "Number of Nodes", title = "Training vs. Test Accuracy") +
  theme_minimal()
```



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.

```r
treeForTrainSet <-
  train(
    category ~ .,
    data = trainSet,
    control = rpart.control(
      minsplit = 15,
      maxdepth = 10,
      minbucket = 19
    ),
    trControl = trainControl(method = "cv", number = 10),
    method = "rpart1SE"
```

```r
  )
cat("Confusion Matrix for trainSet\n")
```

```
## Confusion Matrix for trainSet
```

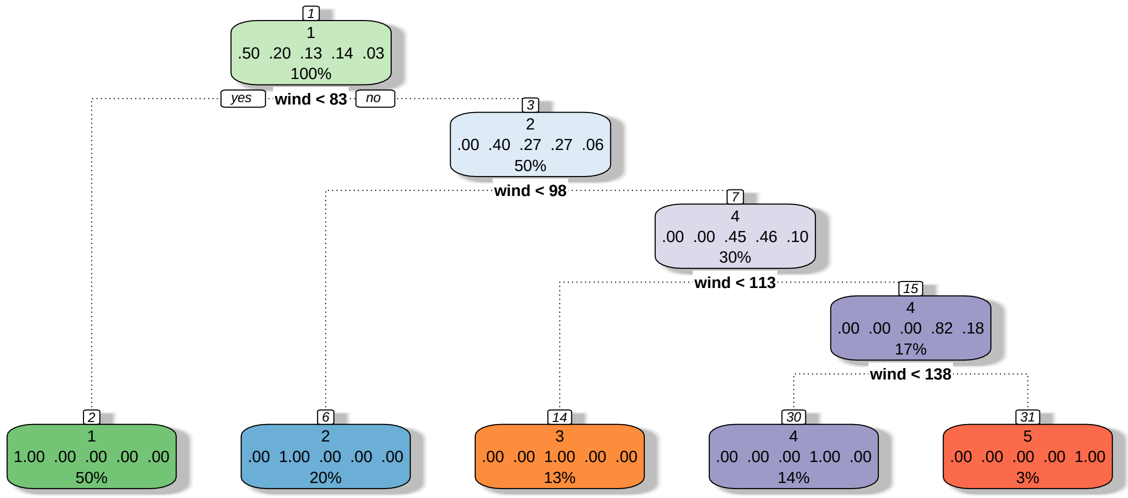
```r
## Confusion Matrix for trainSet
confusionMatrix(trainSet$category, predict(treeForTrainSet, trainSet))
```

```
## 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
 cat("Confusion Matrix for testSet\n")
```

```
## Confusion Matrix for testSet
```

```r
## Confusion Matrix for test Set
confusionMatrixForTestSet <- confusionMatrix(testSet$category, predict(treeForTrainSet, testSet))

#plotting the decision tree
fancyRpartPlot(treeForTrainSet$finalModel, caption = "Decision tree")
```



Decision tree

4. 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
# Load necessary libraries
library(rpart)
library(rpart.plot)

# Load the data
bankData <- read.csv("C:/Users/SDHURVE/Documents/Bank_Modified.csv")
head(bankData)
```

```
##   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
```

```r
names(bankData)
```

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

```r
# Remove ID column (assuming it's the first column)
bankData <- bankData[, -1]

# Convert categorical variables to factors
bankData$approval <- as.factor(bankData$approval)
bankData$bool1 <- as.factor(bankData$bool1)
bankData$bool2 <- as.factor(bankData$bool2)
bankData$bool3 <- as.factor(bankData$bool3)

# Summary before removing missing values
cat("Summary before excluding NA values:\n")
```

```
## Summary before excluding NA values:
```

```r
summary(bankData)
```

```
##      cont1           cont2            cont3         bool1    bool2
##  Min.   :13.75   Min.   : 0.000   Min.   : 0.000   f:329   f:395
##  1st Qu.:22.60   1st Qu.: 1.000   1st Qu.: 0.165   t:361   t:295
##  Median :28.46   Median : 2.750   Median : 1.000
##  Mean   :31.57   Mean   : 4.759   Mean   : 2.223
##  3rd Qu.:38.23   3rd Qu.: 7.207   3rd Qu.: 2.625
##  Max.   :80.25   Max.   :28.000   Max.   :28.500
##  NA's   :12
##      cont4      bool3       cont5           cont6           approval
##  Min.   : 0.0   f:374   Min.   :   0   Min.   :     0.0   -:383
##  1st Qu.: 0.0   t:316   1st Qu.:  75   1st Qu.:     0.0   +:307
##  Median : 0.0           Median : 160   Median :     5.0
##  Mean   : 2.4           Mean   : 184   Mean   :  1017.4
##  3rd Qu.: 3.0           3rd Qu.: 276   3rd Qu.:   395.5
##  Max.   :67.0           Max.   :2000   Max.   :100000.0
##                         NA's   :13
##   credit.score        ages
##  Min.   :583.7   Min.   :17.00
##  1st Qu.:666.7   1st Qu.:31.00
##  Median :697.3   Median :38.00
##  Mean   :696.4   Mean   :39.59
##  3rd Qu.:726.4   3rd Qu.:47.00
##  Max.   :806.0   Max.   :84.00
##
```

```r
# Remove rows with missing values
bankData <- bankData[complete.cases(bankData),]
```

```r
# Summary after removing missing values
cat("\nSummary after excluding NA values:\n")
```

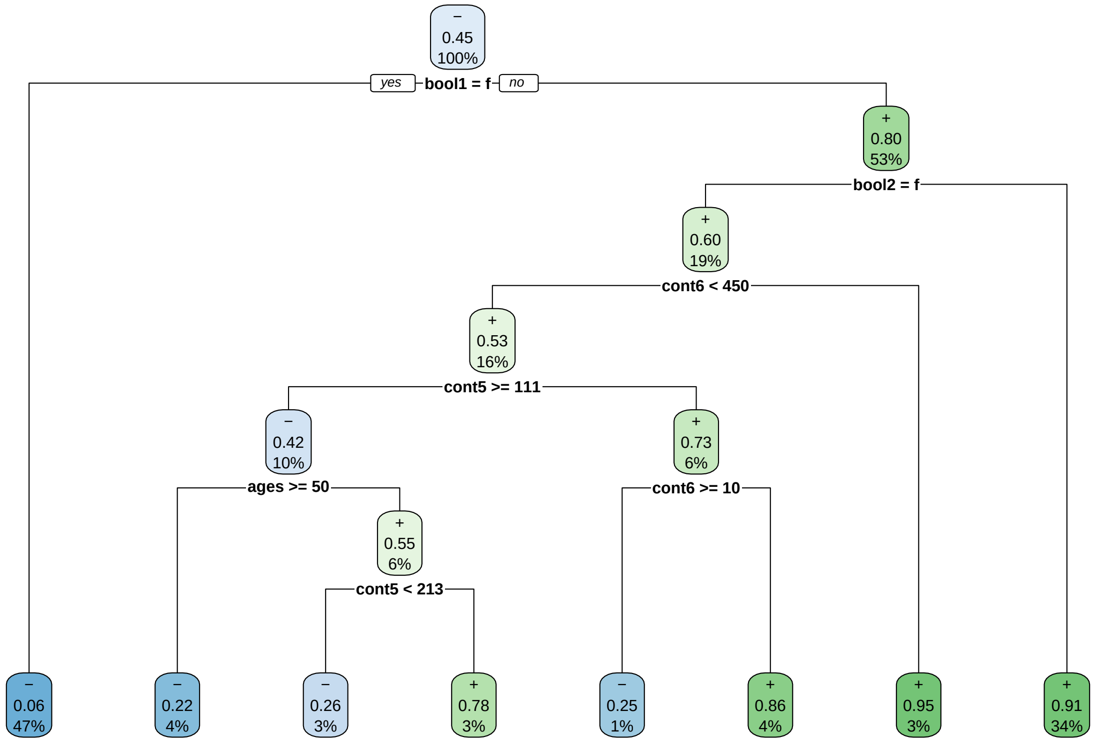
```
##
## Summary after excluding NA values:
```

```r
summary(bankData)
```

```
##      cont1           cont2            cont3        bool1    bool2
##  Min.   :13.75   Min.   : 0.000   Min.   : 0.000   f:314   f:376
##  1st Qu.:22.60   1st Qu.: 1.010   1st Qu.: 0.165   t:352   t:290
##  Median :28.50   Median : 2.750   Median : 1.000
##  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
##      cont4        bool3      cont5              cont6           approval
##  Min.   : 0.000   f:359   Min.   :   0.00   Min.   :     0.0   -:367
##  1st Qu.: 0.000   t:307   1st Qu.:  75.25   1st Qu.:     0.0   +:299
##  Median : 0.000           Median : 160.00   Median :     5.0
##  Mean   : 2.459           Mean   : 182.12   Mean   :   998.6
##  3rd Qu.: 3.000           3rd Qu.: 271.00   3rd Qu.:   399.0
##  Max.   :67.000           Max.   :2000.00   Max.   :100000.0
##   credit.score        ages
##  Min.   :585.1   Min.   :17.00
##  1st Qu.:666.4   1st Qu.:31.00
##  Median :697.1   Median :38.00
##  Mean   :696.3   Mean   :39.67
##  3rd Qu.:726.4   3rd Qu.:47.00
##  Max.   :806.0   Max.   :84.00
```

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

```r
# Load necessary libraries
library(rpart)
library(rpart.plot)

# Build the decision tree model (Basic rpart)
tree_model <- rpart(approval ~ ., data = bankData,
                    method = "class",
                    control = rpart.control(minsplit = 10, maxdepth = 20))

# Plot the decision tree
rpart.plot(tree_model)
```



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

```
# Extract variable importance directly from rpart model
cat("\nVariable Importance from rpart model:\n")
```

```
##
## Variable Importance from rpart model:
```

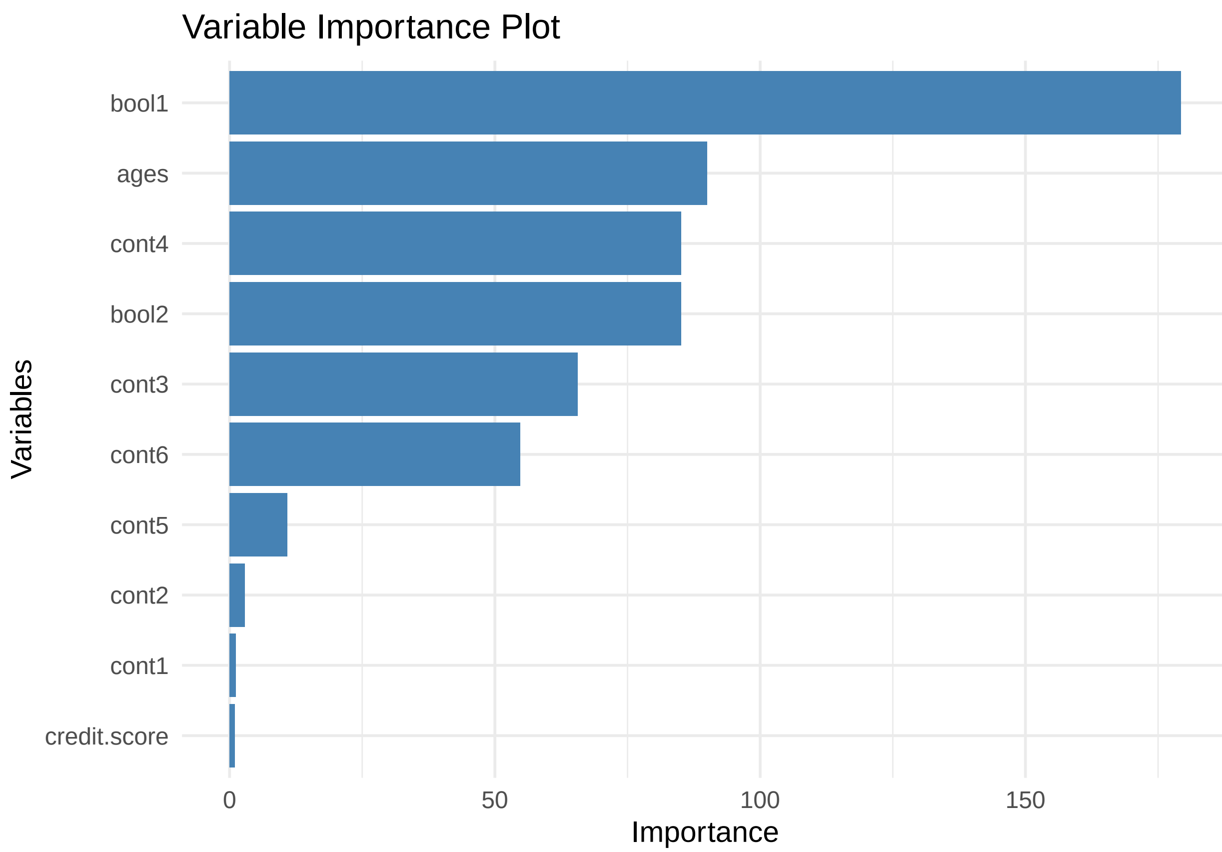
```
print(tree_model$variable.importance)
```

```
##        bool1         ages        bool2        cont4        cont3        cont6
##  179.2824370   89.9958659   85.0835919   85.0835919   65.5827631   54.8079946
##        cont5        cont2        cont1 credit.score
##   10.8719273    2.8939993    1.2143524    0.9735162
```

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

```
# Load necessary library
library(ggplot2)

# Extract variable importance from rpart model
var_importance <- data.frame(Variable = names(tree_model$variable.importance),
                             Importance = tree_model$variable.importance)
```

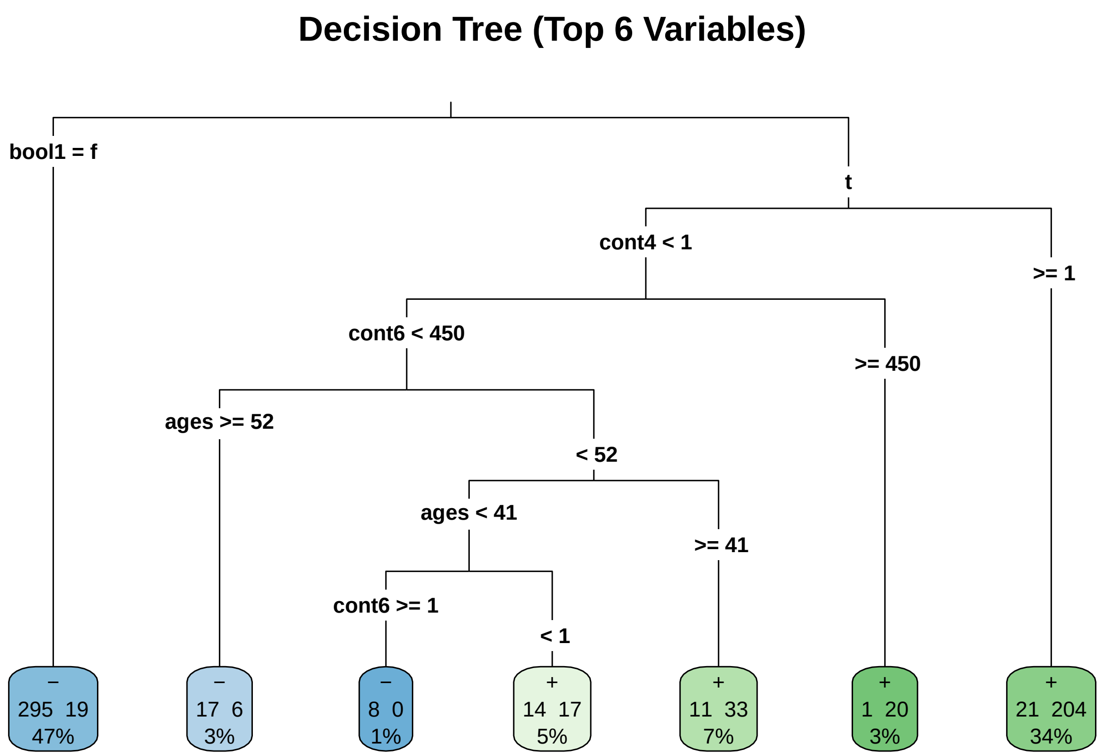
```r
# Plot variable importance using ggplot2
ggplot(var_importance, aes(x = reorder(Variable, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +  # Flip for better readability
  labs(title = "Variable Importance Plot", x = "Variables", y = "Importance") +
  theme_minimal()
```



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?

```r
# Load necessary libraries
library(rpart)
library(caret)

# Select the top six important variables from the previous analysis
top_vars <- c("bool1", "ages", "cont4", "bool2", "cont3", "cont6")

# Create a new dataset with only the top six variables + target variable
bankData_top6 <- bankData[, c(top_vars, "approval")]

# Build decision tree model with only top 6 variables
tree_model_top6 <- rpart(approval ~ ., data = bankData_top6,
                         method = "class",
                         control = rpart.control(minsplit = 10, maxdepth = 20))
```

```r
#Plot the decision tree for the reduced model
rpart.plot(tree_model_top6, main = "Decision Tree (Top 6 Variables)", type = 3, extra = 101)
```




```r
# Evaluate Accuracy Before and After
# Predict on the original dataset (full model)
pred_full <- predict(tree_model, bankData, type = "class")
accuracy_full <- sum(pred_full == bankData$approval) / nrow(bankData)

# Predict on the reduced dataset (top 6 model)
pred_top6 <- predict(tree_model_top6, bankData_top6, type = "class")
accuracy_top6 <- sum(pred_top6 == bankData_top6$approval) / nrow(bankData_top6)

# Print results
cat("\nModel Accuracy Comparison:\n")
```

```
##
## Model Accuracy Comparison:
```

```r
cat("Full Model Accuracy:", round(accuracy_full * 100, 2), "%\n")
```

```
## Full Model Accuracy: 90.54 %
```

```r
cat("Reduced Model (Top 6 Variables) Accuracy:", round(accuracy_top6 * 100, 2), "%\n")
```

```
## Reduced Model (Top 6 Variables) Accuracy: 89.19 %
```

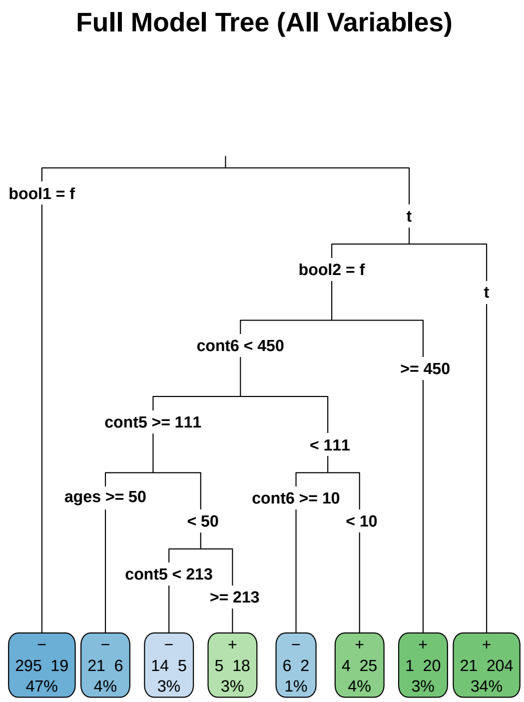
```r
# Compare the effect
if (accuracy_top6 > accuracy_full) {
  cat("Accuracy improved with fewer variables\n")
} else if (accuracy_top6 < accuracy_full) {
  cat("Accuracy decreased with fewer variables. More features were useful!\n")
} else {
  cat("Accuracy remained the same. Model is just as effective with fewer variables\n")
}
```

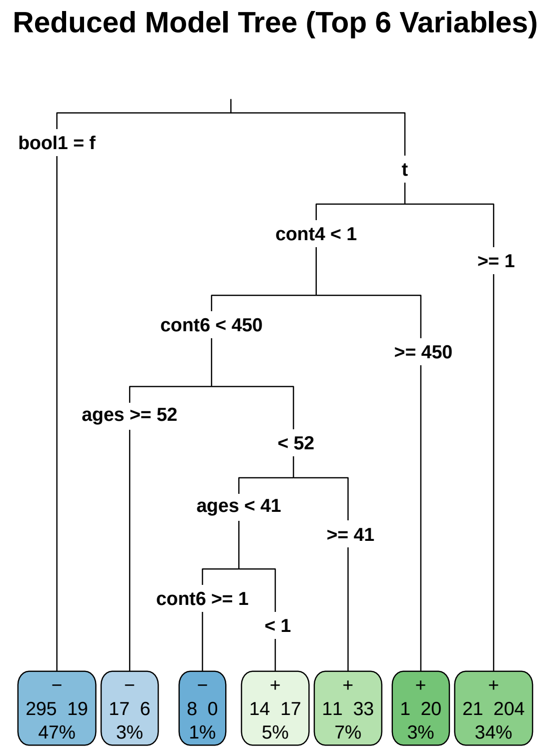
```
## Accuracy decreased with fewer variables. More features were useful!
```

Yes, the change did have an effect on the accuracy. After reducing the model to the top six variables, the accuracy decreased from 90.54% (full model) to 89.19% (reduced model). This indicates that removing some variables led to a slight drop in predictive performance, suggesting that the excluded variables contained useful information for classification. Thus, in this case, using all available features resulted in a more accurate model.

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?

```r
# Load necessary library
library(rpart.plot)

# Plot both trees side by side
par(mfrow = c(1, 2))  # Set up two side-by-side plots

# Plot the full model tree (4.a)
rpart.plot(tree_model, main = "Full Model Tree (All Variables)", type = 3, extra = 101)

# Plot the reduced model tree (4.d)
rpart.plot(tree_model_top6, main = "Reduced Model Tree (Top 6 Variables)", type = 3, extra = 101)
```






```r
# Reset layout
par(mfrow = c(1, 1))
```

Reducing the number of variables had an effect on the size of the decision tree, making it smaller and less complex. The full model tree had more decision nodes and deeper splits, while the reduced model tree, using only the top six variables, had fewer branches and a simpler structure. Although this resulted in a slight decrease in accuracy, it improved interpretability and reduced complexity, which can help with better generalization and prevent overfitting.