HW3

2024-10-14

HW 3 - DSC 441

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 cancer. As a preprocessing step, *remove the IDNumber column* and *exclude rows with NA from the dataset*.

```
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 ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
cancer <- read_csv("/Users/danielkim/Downloads/breast_cancer_updated.csv")
```

```
## Rows: 699 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): Class
## dbl (10): IDNumber, ClumpThickness, UniformCellSize, UniformCellShape, Margi...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(cancer)
```

```
## # A tibble: 6 x 11
##   IDNumber ClumpThickness UniformCellSize UniformCellShape MarginalAdhesion
##      <dbl>          <dbl>           <dbl>            <dbl>            <dbl>
## 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
## # i 6 more variables: EpithelialCellSize <dbl>, BareNuclei <dbl>,
## #   BlandChromatin <dbl>, NormalNucleoli <dbl>, Mitoses <dbl>, Class <chr>
```

```r
dim(cancer)
```

```
## [1] 699  11
```

```r
cancer <- cancer %>% select(-c("IDNumber")) %>% drop_na()
summary(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
```

```r
dim(cancer)
```

```
## [1] 683  10
```

- 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.

```r
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(lattice)

train_control <- trainControl(method = 'cv', number = 10)

model_tree <- train(Class ~., data = cancer, method = 'rpart', trControl = train_control)

model_tree
```

```
## 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, 615, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02510460  0.9428815  0.8755201
##   0.05439331  0.9282822  0.8458280
##   0.79079498  0.8276641  0.5519574
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0251046.
```

The accuracy for the optimal model was 93.84%, which had a corresponding complexity parameter value of 0.0251.

```r
tree_predictions <- predict(model_tree, cancer)

confusionMatrix(as.factor(cancer$Class), tree_predictions)
```

```
## 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
##
```

The predictions made on the testing set using the trained model earned an accuracy of 94.58% with a statistically significant p-value, rejecting the null hypothesis which stated there was no relationship between the independent and dependent variables. The no information rate, for which it describes the proportion of the majority class, was 64.57%.

- b: Generate a visualization of the decision tree.

```r
library(rattle)
```

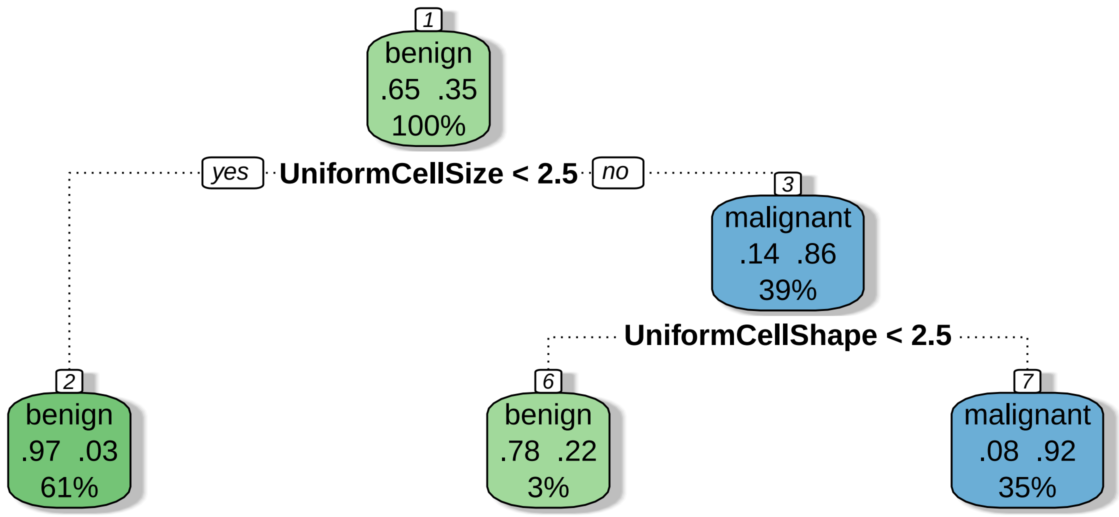
```
## Loading required package: bitops
```

```
## 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
attributes(model_tree)
```

```
## $names
##  [1] "method"       "modelInfo"    "modelType"    "results"      "pred"
##  [6] "bestTune"     "call"         "dots"         "metric"       "control"
## [11] "finalModel"   "preProcess"   "trainingData" "ptype"        "resample"
## [16] "resampledCM"  "perfNames"    "maximize"     "yLimits"      "times"
## [21] "levels"       "terms"        "coefnames"    "xlevels"
##
## $class
## [1] "train"         "train.formula"
```

```r
fancyRpartPlot(model_tree$finalModel, caption = "Decision Tree for Breast Tissue Classification")
```



Decision Tree for Breast Tissue Classification

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

```r
model_tree$finalModel
```

```
## n= 683
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 683 239 benign (0.65007321 0.34992679)
##   2) UniformCellSize< 2.5 418  12 benign (0.97129187 0.02870813) *
##   3) UniformCellSize>=2.5 265  38 malignant (0.14339623 0.85660377)
##     6) UniformCellShape< 2.5 23   5 benign (0.78260870 0.21739130) *
##     7) UniformCellShape>=2.5 242  20 malignant (0.08264463 0.91735537) *
```

- If the breast tissue has a uniform cell size that is less than 2.5 mm,
  - then the breast tissue cell is a benign tumor.
- If the breast tissue has a uniform cell size that is greater than or equal to 2.5 mm
  - and has a uniform cell shape of less than 2.5 mm,
    - then the breast tissue cell is a benign tumor.
  - and has a uniform cell shape of greater than or equal to 2.5 mm,
    - then the breast tissue cell is a malignant tumor.

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 database, 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).

```r
library(dplyr)

data(storms)
dim(storms)
```

```
## [1] 19537    13
```

```r
storms$category <- as.factor(storms$category)

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>        <fct>    <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
# drop rows without classification
storms <- storms %>% drop_na(category)
dim(storms)
```

```
## [1] 4803   13
```

```r
# hurricane_force_diameter and tropicalstorm_force_diameter have null values
dim(storms)
```

```
## [1] 4803   13
```

```r
summary(storms)
```

```
##      name                year          month             day
##  Length:4803        Min.   :1975   Min.   : 1.000   Min.   : 1.00
##  Class :character   1st Qu.:1992   1st Qu.: 8.000   1st Qu.: 8.00
##  Mode  :character   Median :2001   Median : 9.000   Median :16.00
##                     Mean   :2001   Mean   : 8.952   Mean   :15.88
##                     3rd Qu.:2012   3rd Qu.: 9.000   3rd Qu.:24.00
##                     Max.   :2022   Max.   :12.000   Max.   :31.00
```

```
##
##       hour             lat             long
##  Min.   : 0.000   Min.   : 9.50   Min.   :-119.3
##  1st Qu.: 5.000   1st Qu.:19.70   1st Qu.: -76.2
##  Median :12.000   Median :26.40   Median : -63.2
##  Mean   : 9.156   Mean   :26.49   Mean   : -63.9
##  3rd Qu.:18.000   3rd Qu.:32.50   3rd Qu.: -51.8
##  Max.   :23.000   Max.   :50.80   Max.   : -14.1
##
##                    status      category      wind            pressure
##  hurricane             :4803   1:2548   Min.   : 65.00   Min.   : 882.0
##  disturbance           :   0   2: 993   1st Qu.: 70.00   1st Qu.: 958.0
##  extratropical         :   0   3: 593   Median : 80.00   Median : 973.0
##  other low             :   0   4: 553   Mean   : 86.59   Mean   : 968.8
##  subtropical depression:   0   5: 116   3rd Qu.:100.00   3rd Qu.: 983.5
##  subtropical storm     :   0            Max.   :165.00   Max.   :1005.0
##  (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
##  NA's   :2633                 NA's   :2633
```

```r
# first drop rows with null values in hurricane_force_diametere
storms <- storms %>% drop_na(hurricane_force_diameter)
dim(storms)
```

```
## [1] 2170   13
```

```r
# confirmed no more null values
sum(is.na(storms))
```

```
## [1] 0
```

```r
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>     <fct>    <int>    <int>
## 1 Alex   2004     8     3     6  33   -77.4 hurricane 1           70      983
## 2 Alex   2004     8     3    12  34.2 -76.4 hurricane 2           85      974
## 3 Alex   2004     8     3    18  35.3 -75.2 hurricane 2           85      972
## 4 Alex   2004     8     4     0  36   -73.7 hurricane 1           80      974
## 5 Alex   2004     8     4     6  36.8 -72.1 hurricane 1           80      973
## 6 Alex   2004     8     4    12  37.3 -70.2 hurricane 2           85      973
## # i 2 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

```r
# drop columns with NA values

# with a sample size of 4803 remaining, dropping 2633 samples with missing values would reduce the samp
# rather keep the sample size and drop columns with missing values especially as many force diameter da

# name was dropped as it's purpose is the ID the specific storm

# status was dropped as all instances were categorized under hurricane once rows without categories wer
summary(storms$status)
```

```
##            disturbance          extratropical              hurricane
##                      0                      0                   2170
##              other low subtropical depression      subtropical storm
##                      0                      0                      0
##    tropical depression         tropical storm          tropical wave
##                      0                      0                      0
```

```r
storms <- storms %>% select(-c("name", "status"))
head(storms)
```

```
## # A tibble: 6 x 11
##    year month   day  hour   lat  long category  wind pressure
##   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct>    <int>    <int>
## 1  2004     8     3     6  33   -77.4 1           70      983
## 2  2004     8     3    12  34.2 -76.4 2           85      974
## 3  2004     8     3    18  35.3 -75.2 2           85      972
## 4  2004     8     4     0  36   -73.7 1           80      974
## 5  2004     8     4     6  36.8 -72.1 1           80      973
## 6  2004     8     4    12  37.3 -70.2 2           85      973
## # i 2 more variables: tropicalstorm_force_diameter <int>,
## #   hurricane_force_diameter <int>
```

```r
summary(storms)
```

```
##       year          month             day             hour
##  Min.   :2004   Min.   : 1.000   Min.   : 1.00   Min.   : 0.000
##  1st Qu.:2008   1st Qu.: 8.000   1st Qu.: 7.00   1st Qu.: 5.000
##  Median :2012   Median : 9.000   Median :15.00   Median :10.500
##  Mean   :2013   Mean   : 8.951   Mean   :15.13   Mean   : 9.112
##  3rd Qu.:2018   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:16.750
##  Max.   :2022   Max.   :12.000   Max.   :31.00   Max.   :23.000
##       lat             long          category      wind           pressure
##  Min.   : 9.50   Min.   :-119.30   1:1083   Min.   : 65.00   Min.   : 882.0
##  1st Qu.:19.10   1st Qu.: -78.20   2: 434   1st Qu.: 70.00   1st Qu.: 954.0
##  Median :25.40   Median : -65.20   3: 291   Median : 85.00   Median : 969.0
##  Mean   :25.59   Mean   : -65.13   4: 297   Mean   : 88.53   Mean   : 965.6
##  3rd Qu.:31.20   3rd Qu.: -53.35   5:  65   3rd Qu.:100.00   3rd Qu.: 981.0
##  Max.   :48.80   Max.   : -14.10            Max.   :160.00   Max.   :1001.0
##  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.

```
library(rpart)
# sample method
train_control = trainControl(method = "cv", number = 10)

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

# train the model
tree <- train(category ~., data = storms, control = hypers, trControl = train_control, method = "rpart1S

tree
```

```
## CART
##
## 2170 samples
##   10 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1954, 1953, 1953, 1953, 1953, 1952, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8359504  0.7550553
```

The accuracy of the decision tree model using cross validation was 83.59%.

- 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.
  1. Create a train/test partition.
  2. Train on the training set.
  3. 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).
  4. 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?

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

```r
# Everything except the generated indices
test_set = storms[-index,]


# Train the model with training set
tree1 <- train(category ~., data = train_set, method = "rpart1SE", trControl = train_control)


# Evaluate the fit on training set predictions
pred_tree1 <- predict(tree1, train_set)
# Evaluate the fit on testing set predictions
pred_tree1_2 <- predict(tree1, test_set)


# Confusion Matrix with training set predictions
confusionMatrix(pred_tree1, train_set$category)
```

```
## 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   0 204   0   0
##          4   0   0   0 208   0
##          5   0   0   0   0  46
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9976, 1)
##     No Information Rate : 0.499
##     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             1.000   1.0000   1.0000   1.0000  1.00000
## Specificity             1.000   1.0000   1.0000   1.0000  1.00000
## Pos Pred Value          1.000   1.0000   1.0000   1.0000  1.00000
## Neg Pred Value          1.000   1.0000   1.0000   1.0000  1.00000
## Prevalence              0.499   0.1999   0.1341   0.1368  0.03024
## Detection Rate          0.499   0.1999   0.1341   0.1368  0.03024
## Detection Prevalence    0.499   0.1999   0.1341   0.1368  0.03024
## Balanced Accuracy       1.000   1.0000   1.0000   1.0000  1.00000
```

```r
# Confusion Matrix with testing set predictions
confusionMatrix(pred_tree1_2, test_set$category)
```

```
## 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  87   0   0
##          4   0   0   0  89   0
##          5   0   0   0   0  19
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9943, 1)
##     No Information Rate : 0.4992
##     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             0.4992   0.2003   0.1341   0.1371  0.02928
## Detection Rate         0.4992   0.2003   0.1341   0.1371  0.02928
## Detection Prevalence   0.4992   0.2003   0.1341   0.1371  0.02928
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000  1.00000
```

The decision tree did not have problems reporting any category classes. The model that was trained on the training set had the same performance on the training and testing sets, indicating that the model was not over fittting. There would be an issue with over fitting if the model had a better performance with the training set than it did with the testing set, however, as the performance of the model on both sets were the same, there was no over fitting issue.

Problem 3

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

```r
# Partition the data
part = createDataPartition(y=storms$category, p=0.8, list=FALSE)

# Everything in the generated index list
train_part = storms[index,]
# Everything except the generated indices
test_part = storms[-index,]
```

- 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 over fitting 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
# Initialize cross validation
train_control = trainControl(method = "cv", number = 10)

# Tree 1
hyper1 = rpart.control(minsplit =  2, maxdepth = 1, minbucket = 2)
tree1 <- train(category ~., data = train_part, control = hyper1, trControl = train_control, method = "r

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree1 <- predict(tree1, train_part)
# Confusion Matrix
cfm_train1 <- confusionMatrix(train_set$category, pred_tree1)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree1 <- predict(tree1, test_part)
# Confusion Matrix
cfm_test1 <- confusionMatrix(test_part$category, pred_tree1)

# Get training accuracy
acc_train <- cfm_train1$overall[1]
# Get testing accuracy
acc_test <- cfm_test1$overall[1]
# Get number of nodes
nodes <- nrow(tree1$finalModel$frame)

# Form the table
comp_tbl <- data.frame("Nodes" = nodes, "TrainAccuracy" = acc_train, "TestAccuracy" = acc_test,
                       "MaxDepth" = 1, "Minsplit" = 2, "Minbucket" = 2)


# Tree 2
hyper2 = rpart.control(minsplit =  5, maxdepth = 2, minbucket = 5)
tree2 <- train(category ~., data = train_part, control = hyper2, trControl = train_control, method = "r

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree2 <- predict(tree2, train_part)
# Confusion Matrix
cfm_train2 <- confusionMatrix(train_part$category, pred_tree2)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree2 <- predict(tree2, test_part)
# Confusion Matrix
cfm_test2 <- confusionMatrix(test_part$category, pred_tree2)
```

```r
# Get training accuracy
a_train <- cfm_train2$overall[1]
# Get testing accuracy
a_test <- cfm_test2$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
hyper3 = rpart.control(minsplit = 50, maxdepth = 3, minbucket = 50)
tree3 <- train(category ~., data = train_part, control = hyper3, trControl = train_control, method = "r

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree3 <- predict(tree3, train_part)
# Confusion Matrix
cfm_train3 <- confusionMatrix(train_part$category, pred_tree3)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree3 <- predict(tree3, test_part)
# Confusion Matrix
cfm_test3 <- confusionMatrix(test_part$category, pred_tree3)

# Get training accuracy
a_train <- cfm_train3$overall[1]
# Get testing accuracy
a_test <- cfm_test3$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))


# Tree 9
hyper9 = rpart.control(minsplit = 100, maxdepth = 3, minbucket = 100)
tree9 <- train(category ~., data=train_part, control = hyper9, trControl = train_control, method = 'rpa

# Training Set
# Evaluate the fit with confusion matrix
pred_tree9 <- predict(tree9, train_part)
# Confusion Matrix
cfm_train9 <- confusionMatrix(train_part$category, pred_tree9)

# Test Set
# Evaluate fit with a confusion matrix
pred_tree9 <- predict(tree9, test_part)
# Confusion Matrix
cfm_test9 <- confusionMatrix(test_part$category, pred_tree9)
```

```r
#Get training accuracy
a_train <- cfm_train9$overall[1]
#Get testing accuracy
a_test <- cfm_test9$overall[1]
#Get number of nodes
nodes <- nrow(tree9$finalModel$frame)

# Add row to table
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 3, 100, 100))


# Tree 11
hyper11 = rpart.control(minsplit = 1000, maxdepth = 3, minbucket = 1000)
tree11 <- train(category ~., data=train_part, control = hyper11, trControl = train_control, method = 'r

# Training Set
# Evaluate the fit with confusion matrix
pred_tree11 <- predict(tree11, train_part)
# Confusion Matrix
cfm_train11 <- confusionMatrix(train_part$category, pred_tree11)

# Test Set
# Evaluate fit with a confusion matrix
pred_tree11 <- predict(tree11, test_part)
# Confusion Matrix
cfm_test11 <- confusionMatrix(test_part$category, pred_tree11)

#Get training accuracy
a_train <- cfm_train11$overall[1]
#Get testing accuracy
a_test <- cfm_test11$overall[1]
#Get number of nodes
nodes <- nrow(tree11$finalModel$frame)

# Add row to table
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 3, 1000, 1000))


# Tree 12
hyper12 = rpart.control(minsplit = 25, maxdepth = 4, minbucket = 25)
tree12 <- train(category ~., data=train_part, control = hyper12, trControl = train_control, method = 'r

# Training Set
# Evaluate the fit with confusion matrix
pred_tree12 <- predict(tree12, train_part)
# Confusion Matrix
cfm_train12 <- confusionMatrix(train_part$category, pred_tree12)

# Test Set
# Evaluate fit with a confusion matrix
pred_tree12 <- predict(tree12, test_part)
# Confusion Matrix
cfm_test12 <- confusionMatrix(test_part$category, pred_tree12)
```

```
#Get training accuracy
a_train <- cfm_train12$overall[1]
#Get testing accuracy
a_test <- cfm_test12$overall[1]
#Get number of nodes
nodes <- nrow(tree12$finalModel$frame)

# Add row to table
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 4, 25, 25))


# Tree 10
hyper10 = rpart.control(minsplit = 50, maxdepth = 4, minbucket = 50)
tree10 <- train(category ~., data=train_part, control = hyper10, trControl = train_control, method = 'r

# Training Set
# Evaluate the fit with confusion matrix
pred_tree10 <- predict(tree10, train_part)
# Confusion Matrix
cfm_train10 <- confusionMatrix(train_part$category, pred_tree10)

# Test Set
# Evaluate fit with a confusion matrix
pred_tree10 <- predict(tree10, test_part)
# Confusion Matrix
cfm_test10 <- confusionMatrix(test_part$category, pred_tree10)

#Get training accuracy
a_train <- cfm_train10$overall[1]
#Get testing accuracy
a_test <- cfm_test10$overall[1]
#Get number of nodes
nodes <- nrow(tree10$finalModel$frame)

# Add row to table
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 4, 50, 50))


# Tree 4
hyper4 = rpart.control(minsplit = 100, maxdepth = 4, minbucket = 100)
tree4 <- train(category ~., data = train_part, control = hyper4, trControl = train_control, method = "r

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree4 <- predict(tree4, train_part)
# Confusion Matrix
cfm_train4 <- confusionMatrix(train_part$category, pred_tree4)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree4 <- predict(tree4, test_part)
# Confusion Matrix
cfm_test4 <- confusionMatrix(test_part$category, pred_tree4)
```

```
# Get training accuracy
a_train <- cfm_train4$overall[1]
# Get testing accuracy
a_test <- cfm_test4$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, 100, 100))

# Tree 5
hyper5 = rpart.control(minsplit = 1000, maxdepth = 4, minbucket = 1000)
tree5 <- train(category ~., data = train_part, control = hyper5, trControl = train_control, method = "rp

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree5 <- predict(tree5, train_part)
# Confusion Matrix
cfm_train5 <- confusionMatrix(train_part$category, pred_tree5)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree5 <- predict(tree5, test_part)
# Confusion Matrix
cfm_test5 <- confusionMatrix(test_part$category, pred_tree5)

# Get training accuracy
a_train <- cfm_train5$overall[1]
# Get testing accuracy
a_test <- cfm_test5$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, 1000, 1000))


# Tree 13
hyper13 = rpart.control(minsplit = 25, maxdepth = 5, minbucket = 25)
tree13 <- train(category ~., data = train_part, control = hyper13, trControl = train_control, method =

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree13 <- predict(tree13, train_part)
# Confusion Matrix
cfm_train13 <- confusionMatrix(train_part$category, pred_tree13)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree13 <- predict(tree13, test_part)
# Confusion Matrix
cfm_test13 <- confusionMatrix(test_part$category, pred_tree13)
```

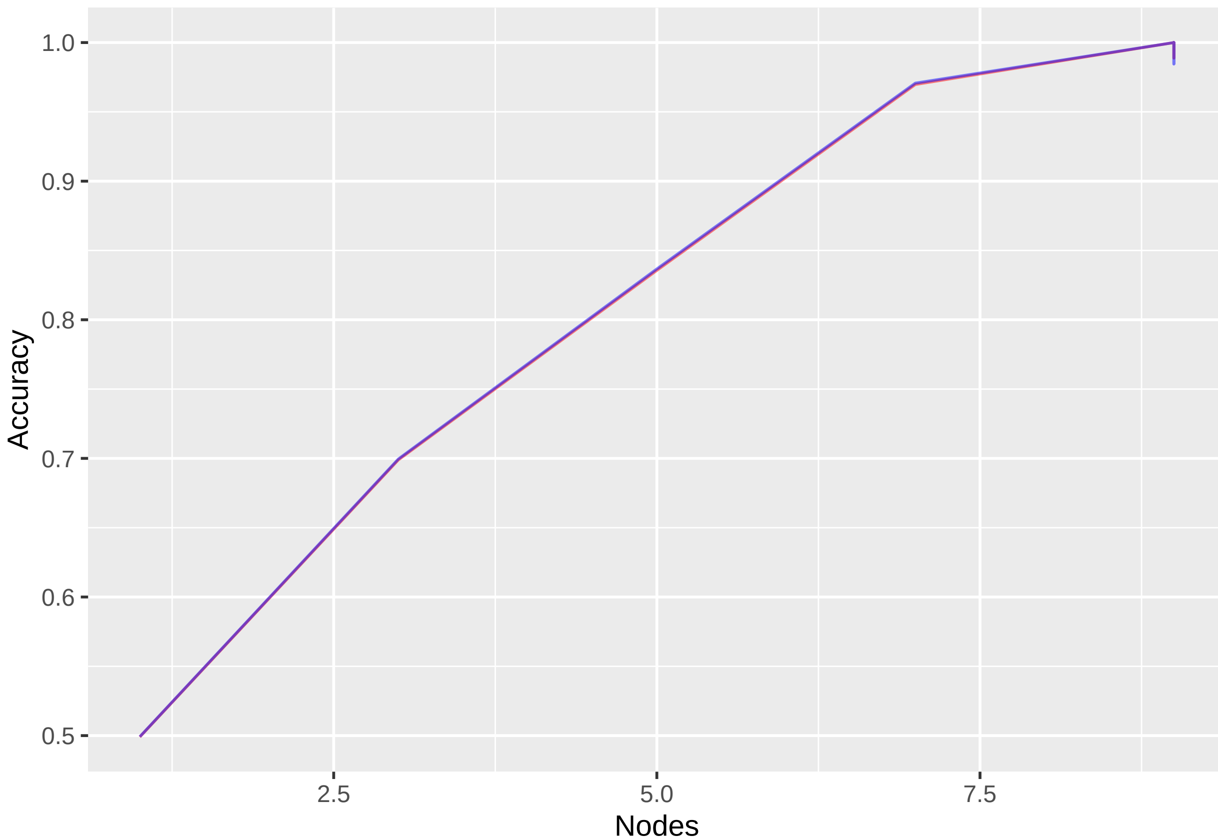
```r
# Get training accuracy
a_train <- cfm_train13$overall[1]
# Get testing accuracy
a_test <- cfm_test13$overall[1]
# Get number of nodes
nodes <- nrow(tree13$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 5, 25, 25))

# Tree 7
hyper7 = rpart.control(minsplit = 50, maxdepth = 5, minbucket = 50)
tree7 <- train(category ~., data = train_part, control = hyper7, trControl = train_control, method = "r

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree7 <- predict(tree7, train_part)
# Confusion Matrix
cfm_train7 <- confusionMatrix(train_part$category, pred_tree7)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree7 <- predict(tree7, test_part)
# Confusion Matrix
cfm_test7 <- confusionMatrix(test_part$category, pred_tree7)

# Get training accuracy
a_train <- cfm_train7$overall[1]
# Get testing accuracy
a_test <- cfm_test7$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, 5, 50, 50))


# Tree 8
hyper8 = rpart.control(minsplit = 100, maxdepth = 5, minbucket = 100)
tree8 <- train(category ~., data=train_part, control = hyper8, trControl = train_control, method = 'rpa

# Training Set
# Evaluate the fit with confusion matrix
pred_tree8 <- predict(tree8, train_part)
# Confusion Matrix
cfm_train8 <- confusionMatrix(train_part$category, pred_tree8)

# Test Set
# Evaluate fit with a confusion matrix
pred_tree8 <- predict(tree8, test_part)
# Confusion Matrix
cfm_test8 <- confusionMatrix(test_part$category, pred_tree8)
```

```
#Get training accuracy
a_train <- cfm_train8$overall[1]
#Get testing accuracy
a_test <- cfm_test8$overall[1]
#Get number of nodes
nodes <- nrow(tree8$finalModel$frame)

# Add row to table
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 5, 100, 100))


# Tree 6
hyper6 = rpart.control(minsplit = 1000, maxdepth = 5, minbucket = 1000)
tree6 <- train(category ~., data = train_part, control = hyper6, trControl = train_control, method = "r

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree6 <- predict(tree6, train_part)
# Confusion Matrix
cfm_train6 <- confusionMatrix(train_part$category, pred_tree6)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree6 <- predict(tree6, test_part)
# Confusion Matrix
cfm_test6 <- confusionMatrix(test_part$category, pred_tree6)

# Get training accuracy
a_train <- cfm_train6$overall[1]
# Get testing accuracy
a_test <- cfm_test6$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, 5, 1000, 1000))


comp_tbl
```

```
##          Nodes TrainAccuracy TestAccuracy MaxDepth Minsplit Minbucket
## Accuracy     3     0.6988823    0.6995378        1        2         2
## 1            5     0.8356345    0.8366718        2        5         5
## 11           7     0.9697567    0.9707242        3       50        50
## 12           7     0.9697567    0.9707242        3      100       100
## 13           1     0.4990138    0.4992296        3     1000      1000
## 14           9     1.0000000    1.0000000        4       25        25
## 15           9     0.9888231    0.9845917        4       50        50
## 16           7     0.9697567    0.9707242        4      100       100
## 17           1     0.4990138    0.4992296        4     1000      1000
## 18           9     1.0000000    1.0000000        5       25        25
## 19           9     0.9888231    0.9845917        5       50        50
## 110          7     0.9697567    0.9707242        5      100       100
```

```
## 111          1     0.4990138    0.4992296        5      1000       1000
```

```r
# Visualize with line plot
ggplot(comp_tbl, aes(x=Nodes)) + geom_line(aes(y = TrainAccuracy), color = "red", alpha = 0.5) + geom_li
```



- 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
# train_control = trainControl(method = 'cv', numbers = 10)
final_hypers = rpart.control(maxdepth = 4, minsplit = 25, minbucket = 25)
final_tree <- train(category ~., data = train_part, trControl = train_control, control = final_hypers, m

train_predict <- predict(final_tree, train_part)
cfm_final_train <- confusionMatrix(train_part$category, train_predict)
cfm_final_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   0 204   0   0
##          4   0   0   0 208   0
```

```
##          5   0   0   0   0  46
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9976, 1)
##     No Information Rate : 0.499
##     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             1.000   1.0000   1.0000   1.0000  1.00000
## Specificity             1.000   1.0000   1.0000   1.0000  1.00000
## Pos Pred Value          1.000   1.0000   1.0000   1.0000  1.00000
## Neg Pred Value          1.000   1.0000   1.0000   1.0000  1.00000
## Prevalence              0.499   0.1999   0.1341   0.1368  0.03024
## Detection Rate          0.499   0.1999   0.1341   0.1368  0.03024
## Detection Prevalence    0.499   0.1999   0.1341   0.1368  0.03024
## Balanced Accuracy       1.000   1.0000   1.0000   1.0000  1.00000
```

```r
test_predict <- predict(final_tree, test_part)
cfm_final_test <- confusionMatrix(test_part$category, test_predict)
cfm_final_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  87   0   0
##          4   0   0   0  89   0
##          5   0   0   0   0  19
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9943, 1)
##     No Information Rate : 0.4992
##     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             0.4992   0.2003   0.1341   0.1371  0.02928
## Detection Rate         0.4992   0.2003   0.1341   0.1371  0.02928
## Detection Prevalence   0.4992   0.2003   0.1341   0.1371  0.02928
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000  1.00000
```

With the max depth parameter of 4, minimum bucket and minimum split of 25, the prediction accuracy on the training and testing sets for this decision tree model was 100%. The similarity in performance on both sets of data indicate the model does not have an issue with over fitting. With these parameters, the decision tree had a total of 9 nodes.

Problem 4

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

```r
#change hyper parameters
hyper_prob4 <- rpart.control(minsplit =  10, maxdepth = 20)

# train the model
tree_prob4 <- train(category ~., data = storms, control = hyper_prob4, trControl = train_control, metho

tree_prob4
```

```
## CART
##
## 2170 samples
##   10 predictor
##    5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1953, 1953, 1954, 1953, 1951, 1954, ...
## Resampling results:
##
##   Accuracy  Kappa
##   1         1
```

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

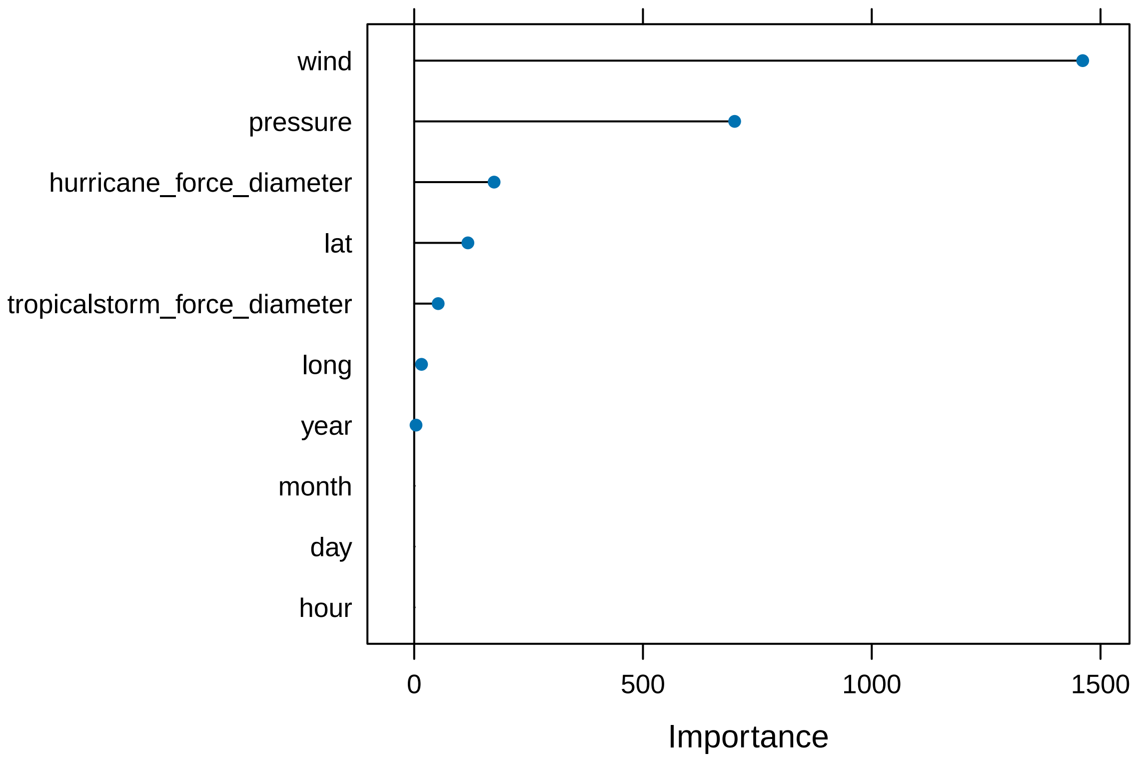
```r
# View the variable importance scores
var_imp <- varImp(tree_prob4, scale = FALSE)
print(var_imp)
```

```
## rpart1SE variable importance
##
##                               Overall
## wind                         1461.078
## pressure                      700.375
## hurricane_force_diameter      174.656
```

```
## lat                              117.377
## tropicalstorm_force_diameter      52.391
## long                              16.022
## year                               3.996
## month                              0.000
## day                                0.000
## hour                               0.000
```

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

```r
# Summarize 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?

```r
storms_6vars <- storms %>% select(c("category", "wind","pressure","hurricane_force_diameter", "lat", "t

# train the model w/ top 6 vars
tree_prob4_b <- train(category ~., data = storms_6vars, control = hyper_prob4, trControl = train_contro

tree_prob4_b
```

```
## CART
##
## 2170 samples
##    6 predictor
##    5 classes: '1', '2', '3', '4', '5'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1952, 1954, 1953, 1954, 1953, 1954, ...
## Resampling results:
##
##   Accuracy  Kappa
##   1         1
```

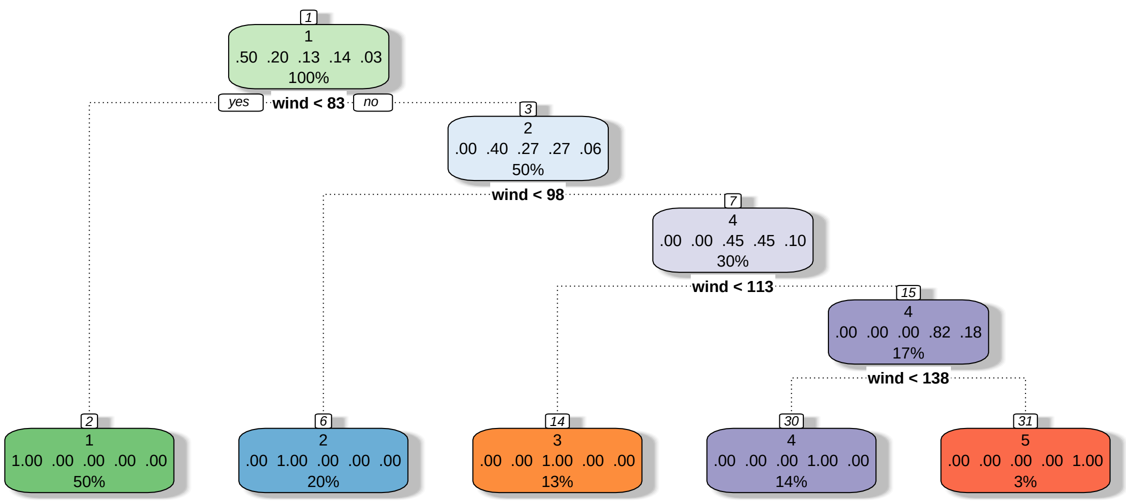
```
nodes_prob4 <- nrow(tree_prob4_b$finalModel$frame)
nodes_prob4
```

```
## [1] 9
```

With the use of only the top 6 variables, the model did not show any change on accuracy as it continued to have an accuracy rate of 100%.

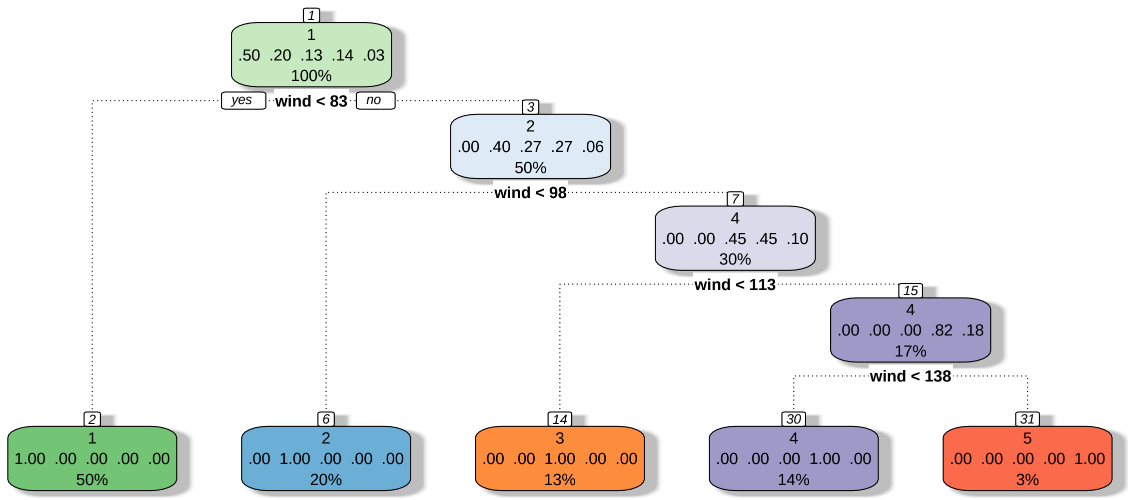
- e: Visualize the trees from (a) and (d) and report if reducing the number of variables had an effect on the size of the tree?

```
fancyRpartPlot(tree_prob4$finalModel, caption = "Decision Tree for Storm Category")
```



Decision Tree for Storm Category

```
fancyRpartPlot(tree_prob4_b$finalModel, caption = "Decision Tree for Storm Category with 6 Variables")
```



Decision Tree for Storm Category with 6 Variables

According to the decision trees that was trained from the entire data set and the data set with the top 6 important variables, there was no difference in performance, including regarding the size of the trees with the maximum depth parameter set to 20 and minimium split value of 10. These trees had a total of 9 nodes, including root, internal, and leaf nodes.