proj

Tsering Dolkar

12/1/2020

```
data <- read.csv("uselection.csv", sep = ";")
data <- data[,2:13]
party <- data.frame()

for(i in 1:nrow(data[1:500,])){
  if(data$PartyName[i] == "BothParty"){
    party <- rbind(party, 0)
  }
  if(data$PartyName[i] == "Republicans"){
    party <- rbind(party, 1)
  }
  if(data$PartyName[i] == "Democrats"){
    party <- rbind(party, 2)
  }
  if(data$PartyName[i] == "Neither"){
    party <- rbind(party, 3)
  }
}
#change all data2 to data in the code.
data2 <- cbind(data[1:500,], party)
#change "X0" to whatever default colname appears for party
names(data2)[names(data2) == "X0"] <- "Party"
data2$Party <- as.factor(data2$Party)
```

Programmatically split the data into 75% and 25% as training and validation sets respectively.

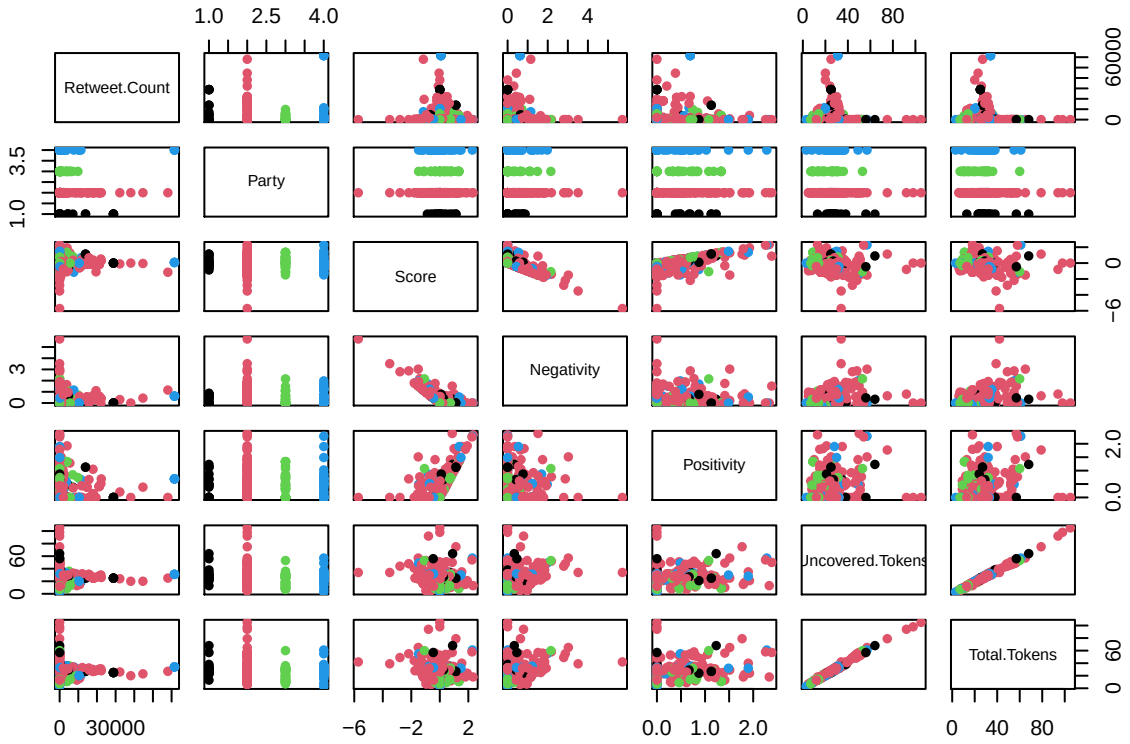
```
## 75% of the sample size
smp_size <- floor(0.75 * nrow(data2))

## set the seed to make your partition reproducible
set.seed(12345)

#change data2 to data later on
trn_idx <- sample(seq_len(nrow(data2)), size = smp_size)
training_data <- data2[trn_idx,]
rownames(training_data) <- NULL
validation_set <- data2[-trn_idx,]
rownames(validation_set) <- NULL

head(training_data)
```

```
##   From.User.Id    To.User.Id Language Retweet.Count   PartyName           Id
## 1 3.262551e+08 -1.000000e+00       en          1584   Democrats 1.278369e+18
## 2 6.139474e+07 -1.000000e+00       en          1008 Republicans 1.278369e+18
## 3 2.163314e+09 -1.000000e+00       en          3843   Democrats 1.278369e+18
## 4 8.199930e+17 -1.000000e+00       en           218     Neither 1.278369e+18
## 5 1.244346e+18  1.223065e+18       en             0 Republicans 1.278369e+18
## 6 7.561284e+17 -1.000000e+00       en           124 Republicans 1.278369e+18
##       Score
## 1 -1.128205
## 2  0.000000
## 3  1.333333
## 4  0.000000
## 5  2.333333
## 6  0.000000
##                                                  Scoring.String
## 1                                      lying (-0.62)  attacking (-0.51)
## 2
## 3                                       welcome (0.51)  paradise (0.82)
## 4
## 5 welcome (0.51)  happy (0.69)  share (0.31)  grateful (0.51)  shares (0.31)
## 6
##   Negativity Positivity Uncovered.Tokens Total.Tokens Party
## 1   1.128205   0.000000               27           29     2
## 2   0.000000   0.000000               29           29     1
## 3   0.000000   1.333333                9           11     2
## 4   0.000000   0.000000               37           37     3
## 5   0.000000   2.333333               13           18     1
## 6   0.000000   0.000000               17           17     1
```

2.

Produce pairwise scatter plots between relevant variables color coded by Party. This will help us see how effective classification using tree plot will be and which predictor might be the most relevant in making decisions in a metric.

```
pairs(cbind(training_data$Retweet.Count,training_data$Party,training_data$Score,training_data$Negativity
```



If we were to see clusters on the scatterplot, we will expect classification tree to be quite effective. However, this data doesn't seem very representative of all voters as we had initially hoped. However, we will proceed forward to see how well it works against other methods. Classification trees base splitting decisions on a metric, for example the trees presented in class use deviance to judge how to partition the tree. I used rpart tree classification package. Rpart uses gini impurity to select splits when performing classification.

3.

Unstandardized effect size statistic comparing the two groups for each variable in the set.

```
# unstandardized measure (e.g., the difference between group means or the unstandardized
# regression coefficients)
democrat <- data.frame()
republican <- data.frame()

for(i in 1:(nrow(training_data))){
  if(training_data$Party[i] == 2){
    democrat = rbind(democrat, training_data[i,])
  }
  if(training_data$Party[i] == 1){
    republican = rbind(republican, training_data[i,])
  }
}
effectsize_retweetcount <- mean(republican$Retweet.Count) - mean(democrat$Retweet.Count)
effectsize_score <- mean(republican$Score) - mean(democrat$Score)
```

```
effectsize_negativity <- mean(republican$Negativity) - mean(democrat$Negativity)
effectsize_positivity <- mean(republican$Positivity) - mean(democrat$Positivity)
effectsize_uncovered.token <- mean(republican$Uncovered.Tokens) -  mean(democrat$Uncovered.Tokens)
effectsize_total.token <- mean(republican$Total.Tokens) -  mean(democrat$Total.Tokens)
paste0("The unstandardized effect size for retweetcount of party 1 and 2 is: ", round(effectsize_retwee
```

```
## [1] "The unstandardized effect size for retweetcount of party 1 and 2 is: 1486.588"
```

```
paste0("The unstandardized effect size for score of party 1 and 2 is: ", round(effectsize_score, 3))
```

```
## [1] "The unstandardized effect size for score of party 1 and 2 is: -0.346"
```

```
paste0("The unstandardized effect size for negativity of party 1 and 2 is: ", round(effectsize_negativi
```

```
## [1] "The unstandardized effect size for negativity of party 1 and 2 is: 0.237"
```

```
paste0("The unstandardized effect size for positivity of party 1 and 2 is: ", round(effectsize_positivi
```

```
## [1] "The unstandardized effect size for positivity of party 1 and 2 is: -0.109"
```

```
paste0("The unstandardized effect size for uncovered.token of party 1 and 2 is: ", round(effectsize_unc
```

```
## [1] "The unstandardized effect size for uncovered.token of party 1 and 2 is: 6.131"
```

```
paste0("The unstandardized effect size for total.token of party 1 and 2 is: ", round(effectsize_total.t
```

```
## [1] "The unstandardized effect size for total.token of party 1 and 2 is: 6.476"
```

4.

```
# https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf

# xval: The number of cross-validations to be done. Usually set to zero during
# exploratory phases of the analysis. A value of 10, for instance, increases the
# compute time to 11-fold over a value of 0.

#xerror is cross validation error

#The splitting index can be "gini" or "information".

#minsplit: The minimum number of observations in a node for which the routine
#will even try to compute a split. The default is 20.

#minbucket: The minimum number of observations in a terminal node. This
#defaults to minsplit/3.

#maxcompete: It is often useful in the printout to see not only the variable that
```

```
# gave the best split at a node, but also the second, third, etc best. This parameter
# controls the number that will be printed. It has no effect on computational time,
# and a small effect on the amount of memory used. The default is 4.

# cp: The threshold complexity parameter
# Internally, rpart keeps track of something called the complexity of a tree. The complexity
# measure is a combination of the size of a tree and the ability of the tree to separate the
# classes of the target variable. If the next best split in growing a tree does not reduce the
# tree's overall complexity by a certain amount, rpart will terminate the growing process. This
# amount is specified by the complexity parameter, cp, in the call to rpart(). Setting cp to a
# negative amount ensures that the tree will be fully grown.

# Each row represents a different height of the tree. In general, more levels in the tree mean
# that it has lower classification error on the training. However, you run the risk of
# overfitting. Often, the cross-validation error will actually grow as the tree gets more   #levels (at
```

```
#cp_val <- c(0.10, 0.09, 0.08, 0.06, 0.05, 0.01, 0.0)
#cp_val[i]

minsplit_val <- seq(1,20,1)

minbucket_val <- seq(1,200,1)

misclassification_rate <- matrix(nrow = length(minsplit_val), ncol = length(minbucket_val))

minxerror <- data.frame()

for(i in 1:length(minsplit_val)){
  for(j in 1:length(minbucket_val)){
    # classification of the tree
    # set cp to 0, so we can see overfitting if it occurs, and then prune as
    # we see fit.
    party_tree <- rpart(Party~Retweet.Count+Score+Negativity+Positivity+
                          Uncovered.Tokens+Total.Tokens, training_data,
                        control = rpart.control(minsplit = minsplit_val[i],
                                                minbucket = minbucket_val[j],
                                                cp = 0))
    # predict the result target we will end up with
    t_pred <- predict(party_tree, newdata = training_data, method = "class")
    #mean of the tree being accurate
    misclassification_rate[i,j] <- mean(training_data$Party == t_pred)
    #find the cp of all the min xerror, i.e cross validation error.
    minxerror <- rbind(minxerror,
                       party_tree$cptable[which.min(party_tree$cptable[,"xerror"]),"CP"])
    if(is.unsorted(rev(party_tree$cptable[,'xerror'])) == FALSE && length(party_tree$cptable[,'xerror'])
       > 1){
      print(paste("(",i,",",j,")"))
      printcp(party_tree)
    }
  }
}
```

```
## [1] "( 1 , 12 )"
```

```
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Negativity       Positivity       Retweet.Count    Score
## [5] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0203252      0   1.00000 1.00000 0.073915
## 2 0.0104530      2   0.95935 0.97561 0.073441
## 3 0.0081301     10   0.86179 0.97561 0.073441
## 4 0.0000000     11   0.85366 0.96748 0.073278
## [1] "( 1 , 20 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 1 , 24 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
```

```
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.95122 0.072943
## [1] "( 1 , 25 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.96748 0.073278
## [1] "( 1 , 36 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.01355      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.95935 0.99187 0.073760
## [1] "( 1 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 1 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 1 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 1 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 2 , 24 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 2 , 26 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.97561 0.073441
## [1] "( 2 , 27 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
```

```
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 2 , 29 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 2 , 32 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.02168      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.93496 0.98374 0.073602
## [1] "( 2 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 2 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 2 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 3 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 3 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 4 , 19 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 4 , 26 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
```

```
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.94309 0.072771
## [1] "( 4 , 27 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 4 , 30 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.97561 0.073441
## [1] "( 4 , 33 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
```

```
## 1 0.01897      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.94309 0.99187 0.073760
## [1] "( 4 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 4 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 5 , 23 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
```

```
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.96748 0.073278
## [1] "( 5 , 33 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01897      0   1.00000      1 0.073915
## 2 0.00000      3   0.94309      1 0.073915
## [1] "( 5 , 37 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 5 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 5 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 5 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 5 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 6 , 22 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 6 , 28 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.97561 0.073441
## [1] "( 6 , 31 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.02439      0   1.00000      1 0.073915
## 2 0.00000      3   0.92683      1 0.073915
## [1] "( 6 , 37 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 6 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 6 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 6 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 7 , 22 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 7 , 23 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
```

```
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 7 , 26 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 7 , 29 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 7 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
```

```
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 8 , 12 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Negativity       Positivity       Retweet.Count    Score
## [5] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0203252      0   1.00000 1.00000 0.073915
## 2 0.0104530      2   0.95935 1.00000 0.073915
## 3 0.0081301     10   0.86179 1.00000 0.073915
## 4 0.0000000     11   0.85366 0.97561 0.073441
## [1] "( 8 , 13 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Score         Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0271003      0   1.00000 1.00000 0.073915
## 2 0.0243902      3   0.91870 0.99187 0.073760
## 3 0.0065041      4   0.89431 0.99187 0.073760
## 4 0.0000000      9   0.86179 0.99187 0.073760
## [1] "( 8 , 21 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
```

```
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 8 , 26 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 8 , 30 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.96748 0.073278
## [1] "( 8 , 32 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
```

```
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.02168      0   1.00000      1 0.073915
## 2 0.00000      3   0.93496      1 0.073915
## [1] "( 8 , 35 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.01897      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.94309 0.97561 0.073441
## [1] "( 8 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 8 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
```

```
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 9 , 12 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Negativity       Positivity       Retweet.Count    Score
## [5] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##          CP nsplit rel error xerror     xstd
## 1 0.0203252      0   1.00000      1 0.073915
## 2 0.0104530      2   0.95935      1 0.073915
## 3 0.0081301     10   0.86179      1 0.073915
## 4 0.0000000     11   0.85366      1 0.073915
## [1] "( 9 , 23 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 9 , 24 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.95935 0.073112
## [1] "( 9 , 26 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 9 , 27 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 9 , 28 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 9 , 29 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.96748 0.073278
## [1] "( 9 , 31 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.02439      0   1.00000      1 0.073915
## 2 0.00000      3   0.92683      1 0.073915
## [1] "( 9 , 35 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01897      0   1.00000      1 0.073915
## 2 0.00000      3   0.94309      1 0.073915
## [1] "( 9 , 37 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.01355      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.95935 0.98374 0.073602
## [1] "( 9 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 10 , 22 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 10 , 23 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 10 , 27 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.94309 0.072771
## [1] "( 10 , 37 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.01355      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.95935 0.98374 0.073602
## [1] "( 10 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 11 , 25 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 11 , 33 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.01897      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.94309 0.99187 0.073760
## [1] "( 11 , 35 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01897      0   1.00000      1 0.073915
## 2 0.00000      3   0.94309      1 0.073915
## [1] "( 11 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 11 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 12 , 29 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 12 , 34 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01897      0   1.00000      1 0.073915
## 2 0.00000      3   0.94309      1 0.073915
## [1] "( 12 , 36 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 12 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 12 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 12 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 13 , 22 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 13 , 24 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.96748 0.073278
## [1] "( 13 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 13 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 14 , 24 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.96748 0.073278
## [1] "( 14 , 25 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 14 , 26 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.96748 0.073278
## [1] "( 14 , 33 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01897      0   1.00000      1 0.073915
## 2 0.00000      3   0.94309      1 0.073915
## [1] "( 14 , 35 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01897      0   1.00000      1 0.073915
## 2 0.00000      3   0.94309      1 0.073915
## [1] "( 14 , 36 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.01355      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.95935 0.99187 0.073760
## [1] "( 14 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 14 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 14 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 14 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 15 , 22 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.97561 0.073441
## [1] "( 15 , 27 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000 1.0000 0.073915
## 2 0.0000      3    0.9187 0.9187 0.072239
## [1] "( 15 , 32 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.02168      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.93496 0.98374 0.073602
## [1] "( 15 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
```

```
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 16 , 16 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Score         Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0271003      0   1.00000 1.00000 0.073915
## 2 0.0243902      3   0.91870 0.99187 0.073760
## 3 0.0065041      4   0.89431 0.94309 0.072771
## 4 0.0000000      9   0.86179 0.91870 0.072239
## [1] "( 16 , 20 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 16 , 25 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 16 , 28 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 16 , 30 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.95935 0.073112
## [1] "( 16 , 35 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01897      0   1.00000      1 0.073915
## 2 0.00000      3   0.94309      1 0.073915
## [1] "( 16 , 37 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.01355      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.95935 0.99187 0.073760
## [1] "( 16 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 16 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 16 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 17 , 23 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 17 , 28 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.97561 0.073441
## [1] "( 17 , 31 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.02439      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.92683 0.96748 0.073278
## [1] "( 17 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 17 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 17 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 18 , 19 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 18 , 21 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 18 , 25 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.95935 0.073112
## [1] "( 18 , 29 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error xerror     xstd
## 1 0.0271      0    1.0000      1 0.073915
## 2 0.0000      3    0.9187      1 0.073915
## [1] "( 18 , 31 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.02439      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.92683 0.95935 0.073112
## [1] "( 18 , 33 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01897      0   1.00000      1 0.073915
## 2 0.00000      3   0.94309      1 0.073915
## [1] "( 18 , 36 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 18 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 18 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 18 , 41 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.00271      0   1.00000      1 0.073915
## 2 0.00000      3   0.99187      1 0.073915
## [1] "( 19 , 23 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.98374 0.073602
## [1] "( 19 , 27 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##       CP nsplit rel error  xerror     xstd
## 1 0.0271      0    1.0000 1.00000 0.073915
## 2 0.0000      3    0.9187 0.99187 0.073760
## [1] "( 19 , 32 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.02168      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.93496 0.97561 0.073441
## [1] "( 19 , 39 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 19 , 40 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 20 , 32 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.02168      0   1.00000      1 0.073915
## 2 0.00000      3   0.93496      1 0.073915
## [1] "( 20 , 35 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
```

```
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error  xerror     xstd
## 1 0.01897      0   1.00000 1.00000 0.073915
## 2 0.00000      3   0.94309 0.97561 0.073441
## [1] "( 20 , 36 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
## [1] "( 20 , 38 )"
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = minsplit_val[i], minbucket = minbucket_val[j],
##         cp = 0))
##
## Variables actually used in tree construction:
## [1] Total.Tokens     Uncovered.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##        CP nsplit rel error xerror     xstd
## 1 0.01355      0   1.00000      1 0.073915
## 2 0.00000      3   0.95935      1 0.073915
```

```
colnames(minxerror) <- "min_cp"
#find the min of all the min xerrors, and we will have the optimal cp.
cp_val <- min(minxerror)
# we look at the misclassification rate and choose the row and column index that gives us the
#best classification and the largest observation acceptance for the parameter minbucket and
```

```
#minsplit.
misclassification_rate
```

```
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
##  [1,] 0.20066667 0.14066667 0.07600000 0.07066667 0.05600000 0.05733333
##  [2,] 0.20066667 0.14066667 0.07600000 0.07066667 0.05600000 0.05733333
##  [3,] 0.17333333 0.14066667 0.07600000 0.07066667 0.05600000 0.05733333
##  [4,] 0.15466667 0.14066667 0.07600000 0.07066667 0.05600000 0.05733333
##  [5,] 0.12666667 0.11466667 0.07600000 0.07066667 0.05600000 0.05733333
##  [6,] 0.10866667 0.10933333 0.07600000 0.07066667 0.05600000 0.05733333
##  [7,] 0.09600000 0.08600000 0.07600000 0.07066667 0.05600000 0.05733333
##  [8,] 0.09000000 0.08600000 0.07600000 0.07066667 0.05600000 0.05733333
##  [9,] 0.07933333 0.07800000 0.06800000 0.06000000 0.05600000 0.05733333
## [10,] 0.07933333 0.07200000 0.06800000 0.05333333 0.05600000 0.05733333
## [11,] 0.07733333 0.07066667 0.06800000 0.05133333 0.05600000 0.05733333
## [12,] 0.07466667 0.06800000 0.06200000 0.05133333 0.05600000 0.05733333
## [13,] 0.07466667 0.06800000 0.06200000 0.05066667 0.05600000 0.05466667
## [14,] 0.07266667 0.06733333 0.06133333 0.05066667 0.05600000 0.05466667
## [15,] 0.06000000 0.05466667 0.05466667 0.04333333 0.04866667 0.04600000
## [16,] 0.06000000 0.05466667 0.05466667 0.04333333 0.04866667 0.04600000
## [17,] 0.06000000 0.05466667 0.05466667 0.02733333 0.03266667 0.03000000
## [18,] 0.05533333 0.05000000 0.05000000 0.01933333 0.02466667 0.02200000
## [19,] 0.05466667 0.05000000 0.05000000 0.01933333 0.02466667 0.02200000
## [20,] 0.05000000 0.05000000 0.05000000 0.01933333 0.02466667 0.02200000
##             [,7]       [,8]       [,9]      [,10] [,11] [,12]       [,13]
##  [1,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
##  [2,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
##  [3,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
##  [4,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
##  [5,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
##  [6,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
##  [7,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
##  [8,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
##  [9,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [10,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [11,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [12,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [13,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [14,] 0.04933333 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [15,] 0.04066667 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [16,] 0.04066667 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [17,] 0.02466667 0.02000000 0.01666667 0.01666667 0.016 0.016 0.004666667
## [18,] 0.01666667 0.01666667 0.01666667 0.01666667 0.016 0.016 0.004666667
## [19,] 0.01666667 0.01666667 0.01666667 0.01666667 0.016 0.016 0.004666667
## [20,] 0.01666667 0.01666667 0.01666667 0.01666667 0.016 0.016 0.004666667
##             [,14]       [,15]       [,16]        [,17]        [,18] [,19] [,20]
##  [1,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
##  [2,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
##  [3,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
##  [4,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
##  [5,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
##  [6,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
##  [7,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
```

```
##  [8,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
##  [9,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [10,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [11,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [12,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [13,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [14,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [15,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [16,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [17,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [18,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [19,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
## [20,] 0.004666667 0.004666667 0.004666667 0.0006666667 0.0006666667     0     0
##       [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
##  [1,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [2,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [3,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [4,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [5,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [7,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [8,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [9,]     0     0     0     0     0     0     0     0     0     0     0     0
## [10,]     0     0     0     0     0     0     0     0     0     0     0     0
## [11,]     0     0     0     0     0     0     0     0     0     0     0     0
## [12,]     0     0     0     0     0     0     0     0     0     0     0     0
## [13,]     0     0     0     0     0     0     0     0     0     0     0     0
## [14,]     0     0     0     0     0     0     0     0     0     0     0     0
## [15,]     0     0     0     0     0     0     0     0     0     0     0     0
## [16,]     0     0     0     0     0     0     0     0     0     0     0     0
## [17,]     0     0     0     0     0     0     0     0     0     0     0     0
## [18,]     0     0     0     0     0     0     0     0     0     0     0     0
## [19,]     0     0     0     0     0     0     0     0     0     0     0     0
## [20,]     0     0     0     0     0     0     0     0     0     0     0     0
##       [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44]
##  [1,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [2,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [3,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [4,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [5,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [7,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [8,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [9,]     0     0     0     0     0     0     0     0     0     0     0     0
## [10,]     0     0     0     0     0     0     0     0     0     0     0     0
## [11,]     0     0     0     0     0     0     0     0     0     0     0     0
## [12,]     0     0     0     0     0     0     0     0     0     0     0     0
## [13,]     0     0     0     0     0     0     0     0     0     0     0     0
## [14,]     0     0     0     0     0     0     0     0     0     0     0     0
## [15,]     0     0     0     0     0     0     0     0     0     0     0     0
## [16,]     0     0     0     0     0     0     0     0     0     0     0     0
## [17,]     0     0     0     0     0     0     0     0     0     0     0     0
## [18,]     0     0     0     0     0     0     0     0     0     0     0     0
## [19,]     0     0     0     0     0     0     0     0     0     0     0     0
```

```
## [20,]     0     0     0     0     0     0     0     0     0     0     0     0
##       [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56]
##  [1,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [2,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [3,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [4,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [5,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [7,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [8,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [9,]     0     0     0     0     0     0     0     0     0     0     0     0
## [10,]     0     0     0     0     0     0     0     0     0     0     0     0
## [11,]     0     0     0     0     0     0     0     0     0     0     0     0
## [12,]     0     0     0     0     0     0     0     0     0     0     0     0
## [13,]     0     0     0     0     0     0     0     0     0     0     0     0
## [14,]     0     0     0     0     0     0     0     0     0     0     0     0
## [15,]     0     0     0     0     0     0     0     0     0     0     0     0
## [16,]     0     0     0     0     0     0     0     0     0     0     0     0
## [17,]     0     0     0     0     0     0     0     0     0     0     0     0
## [18,]     0     0     0     0     0     0     0     0     0     0     0     0
## [19,]     0     0     0     0     0     0     0     0     0     0     0     0
## [20,]     0     0     0     0     0     0     0     0     0     0     0     0
##       [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68]
##  [1,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [2,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [3,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [4,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [5,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [7,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [8,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [9,]     0     0     0     0     0     0     0     0     0     0     0     0
## [10,]     0     0     0     0     0     0     0     0     0     0     0     0
## [11,]     0     0     0     0     0     0     0     0     0     0     0     0
## [12,]     0     0     0     0     0     0     0     0     0     0     0     0
## [13,]     0     0     0     0     0     0     0     0     0     0     0     0
## [14,]     0     0     0     0     0     0     0     0     0     0     0     0
## [15,]     0     0     0     0     0     0     0     0     0     0     0     0
## [16,]     0     0     0     0     0     0     0     0     0     0     0     0
## [17,]     0     0     0     0     0     0     0     0     0     0     0     0
## [18,]     0     0     0     0     0     0     0     0     0     0     0     0
## [19,]     0     0     0     0     0     0     0     0     0     0     0     0
## [20,]     0     0     0     0     0     0     0     0     0     0     0     0
##       [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80]
##  [1,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [2,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [3,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [4,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [5,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [7,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [8,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [9,]     0     0     0     0     0     0     0     0     0     0     0     0
## [10,]     0     0     0     0     0     0     0     0     0     0     0     0
```

```
## [11,]     0     0     0     0     0     0     0     0     0     0     0     0
## [12,]     0     0     0     0     0     0     0     0     0     0     0     0
## [13,]     0     0     0     0     0     0     0     0     0     0     0     0
## [14,]     0     0     0     0     0     0     0     0     0     0     0     0
## [15,]     0     0     0     0     0     0     0     0     0     0     0     0
## [16,]     0     0     0     0     0     0     0     0     0     0     0     0
## [17,]     0     0     0     0     0     0     0     0     0     0     0     0
## [18,]     0     0     0     0     0     0     0     0     0     0     0     0
## [19,]     0     0     0     0     0     0     0     0     0     0     0     0
## [20,]     0     0     0     0     0     0     0     0     0     0     0     0
##       [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92]
##  [1,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [2,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [3,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [4,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [5,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [7,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [8,]     0     0     0     0     0     0     0     0     0     0     0     0
##  [9,]     0     0     0     0     0     0     0     0     0     0     0     0
## [10,]     0     0     0     0     0     0     0     0     0     0     0     0
## [11,]     0     0     0     0     0     0     0     0     0     0     0     0
## [12,]     0     0     0     0     0     0     0     0     0     0     0     0
## [13,]     0     0     0     0     0     0     0     0     0     0     0     0
## [14,]     0     0     0     0     0     0     0     0     0     0     0     0
## [15,]     0     0     0     0     0     0     0     0     0     0     0     0
## [16,]     0     0     0     0     0     0     0     0     0     0     0     0
## [17,]     0     0     0     0     0     0     0     0     0     0     0     0
## [18,]     0     0     0     0     0     0     0     0     0     0     0     0
## [19,]     0     0     0     0     0     0     0     0     0     0     0     0
## [20,]     0     0     0     0     0     0     0     0     0     0     0     0
##       [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100] [,101] [,102] [,103]
##  [1,]     0     0     0     0     0     0     0      0      0      0      0
##  [2,]     0     0     0     0     0     0     0      0      0      0      0
##  [3,]     0     0     0     0     0     0     0      0      0      0      0
##  [4,]     0     0     0     0     0     0     0      0      0      0      0
##  [5,]     0     0     0     0     0     0     0      0      0      0      0
##  [6,]     0     0     0     0     0     0     0      0      0      0      0
##  [7,]     0     0     0     0     0     0     0      0      0      0      0
##  [8,]     0     0     0     0     0     0     0      0      0      0      0
##  [9,]     0     0     0     0     0     0     0      0      0      0      0
## [10,]     0     0     0     0     0     0     0      0      0      0      0
## [11,]     0     0     0     0     0     0     0      0      0      0      0
## [12,]     0     0     0     0     0     0     0      0      0      0      0
## [13,]     0     0     0     0     0     0     0      0      0      0      0
## [14,]     0     0     0     0     0     0     0      0      0      0      0
## [15,]     0     0     0     0     0     0     0      0      0      0      0
## [16,]     0     0     0     0     0     0     0      0      0      0      0
## [17,]     0     0     0     0     0     0     0      0      0      0      0
## [18,]     0     0     0     0     0     0     0      0      0      0      0
## [19,]     0     0     0     0     0     0     0      0      0      0      0
## [20,]     0     0     0     0     0     0     0      0      0      0      0
##       [,104] [,105] [,106] [,107] [,108] [,109] [,110] [,111] [,112] [,113]
##  [1,]      0      0      0      0      0      0      0      0      0      0
```

```
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,114] [,115] [,116] [,117] [,118] [,119] [,120] [,121] [,122] [,123]
##  [1,]      0      0      0      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,124] [,125] [,126] [,127] [,128] [,129] [,130] [,131] [,132] [,133]
##  [1,]      0      0      0      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
```

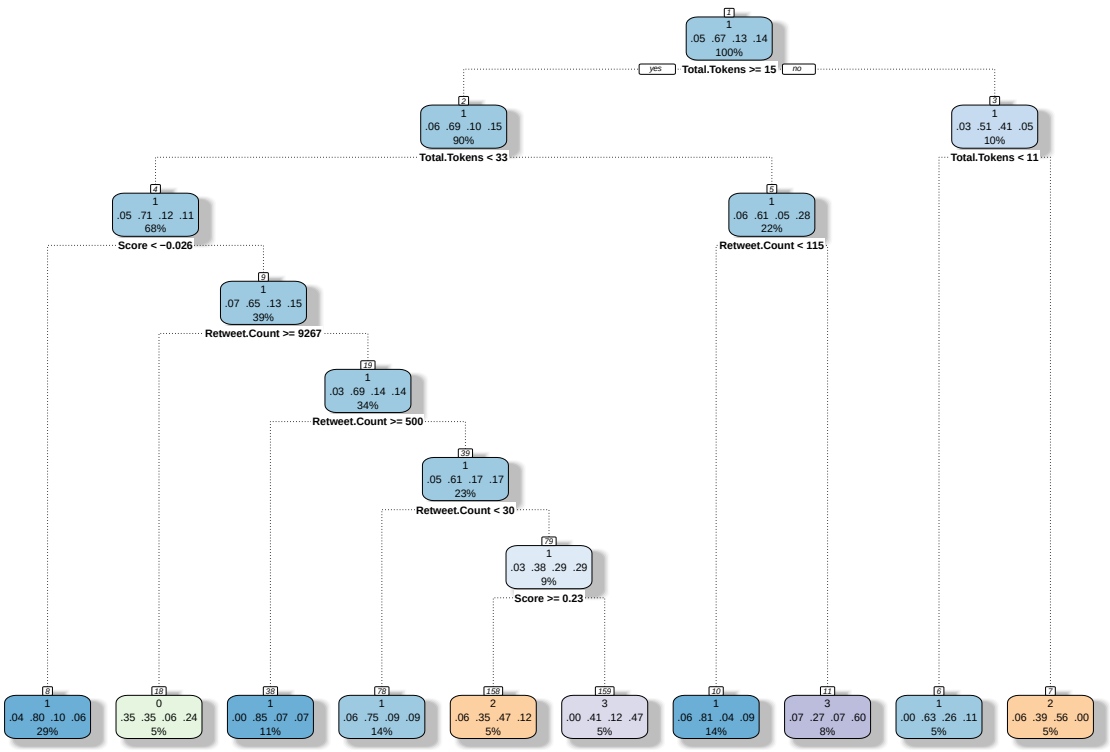
```
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,134] [,135] [,136] [,137] [,138] [,139] [,140] [,141] [,142] [,143]
##  [1,]      0      0      0      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,144] [,145] [,146] [,147] [,148] [,149] [,150] [,151] [,152] [,153]
##  [1,]      0      0      0      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,154] [,155] [,156] [,157] [,158] [,159] [,160] [,161] [,162] [,163]
##  [1,]      0      0      0      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
```

```
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,164] [,165] [,166] [,167] [,168] [,169] [,170] [,171] [,172] [,173]
##  [1,]      0      0      0      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,174] [,175] [,176] [,177] [,178] [,179] [,180] [,181] [,182] [,183]
##  [1,]      0      0      0      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
```

```
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,184] [,185] [,186] [,187] [,188] [,189] [,190] [,191] [,192] [,193]
##  [1,]      0      0      0      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0      0      0      0
##       [,194] [,195] [,196] [,197] [,198] [,199] [,200]
##  [1,]      0      0      0      0      0      0      0
##  [2,]      0      0      0      0      0      0      0
##  [3,]      0      0      0      0      0      0      0
##  [4,]      0      0      0      0      0      0      0
##  [5,]      0      0      0      0      0      0      0
##  [6,]      0      0      0      0      0      0      0
##  [7,]      0      0      0      0      0      0      0
##  [8,]      0      0      0      0      0      0      0
##  [9,]      0      0      0      0      0      0      0
## [10,]      0      0      0      0      0      0      0
## [11,]      0      0      0      0      0      0      0
## [12,]      0      0      0      0      0      0      0
## [13,]      0      0      0      0      0      0      0
## [14,]      0      0      0      0      0      0      0
## [15,]      0      0      0      0      0      0      0
## [16,]      0      0      0      0      0      0      0
## [17,]      0      0      0      0      0      0      0
## [18,]      0      0      0      0      0      0      0
## [19,]      0      0      0      0      0      0      0
## [20,]      0      0      0      0      0      0      0
```

We see from the matrix that the best classification while choosing the minimum observation to split on and minimum observation at a terminal node required is at a varying range of minbucket and minsplit values. I read the printcp(training_set_tree), and looked at all the xerror, which is the cross validation error. Since none of the xerror for these trees seem to be increasing when we look down the column, we know from this that none of these parameters give us overfitted tree. Keeping in mind that minsplit is the minimum number of observations in a node for which the routine will even try to compute a split, we will choose a minsplit = 1.

```
#minsplit = 5
party_tree.chosen1 <- rpart(Party~Retweet.Count+Score+Negativity+Positivity+
                          Uncovered.Tokens+Total.Tokens,
                                   training_data, control =
                            rpart.control(minsplit = 1,
                                   minbucket = 17, cp = 0))
fancyRpartPlot(party_tree.chosen1, caption = NULL)
```

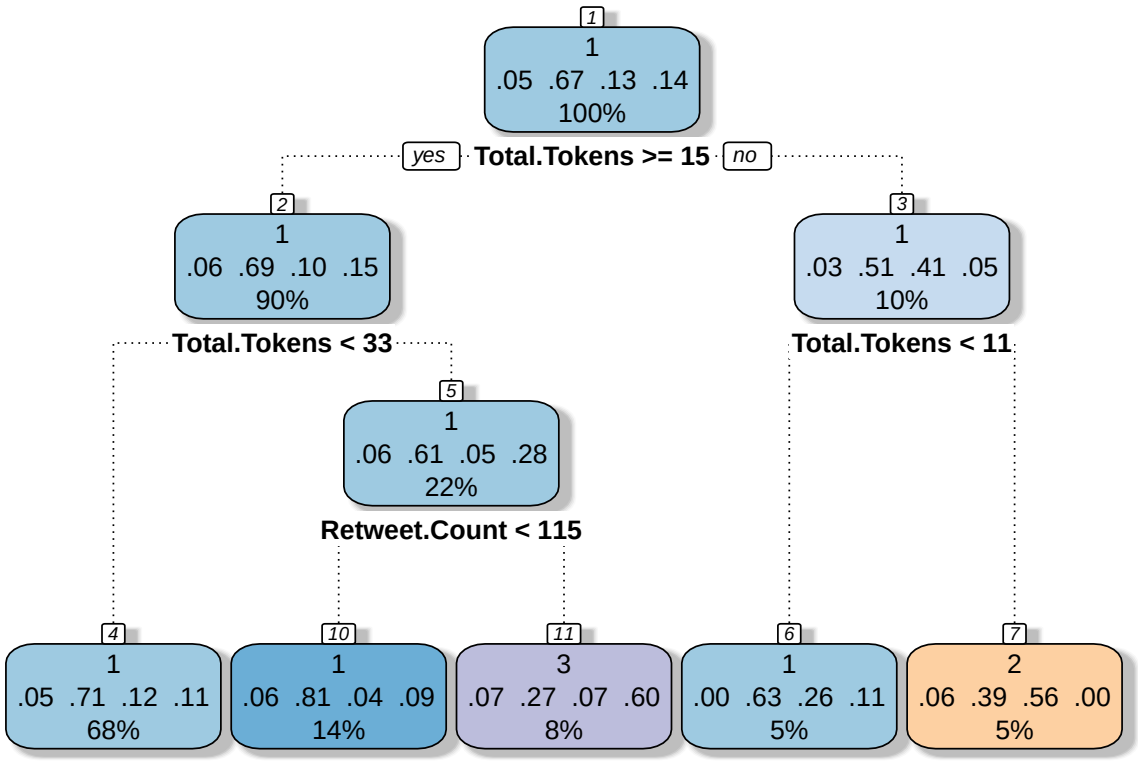



```
printcp(party_tree.chosen1)
```

```
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = 1, minbucket = 17, cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Score         Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
##
##         CP nsplit rel error xerror     xstd
```

```
## 1 0.027100      0   1.00000 1.0000 0.073915
## 2 0.024390      3   0.91870 1.0569 0.074926
## 3 0.004878      4   0.89431 1.0244 0.074364
## 4 0.000000      9   0.86992 1.0813 0.075320
```

```
#minsplit = 6
party_tree.chosen2 <- rpart(Party~Retweet.Count+Score+Negativity+Positivity+
                          Uncovered.Tokens+Total.Tokens, training_data,
                          control = rpart.control(minsplit = 1, minbucket = 18, cp = 0))
fancyRpartPlot(party_tree.chosen2, caption = NULL)
```




```
printcp(party_tree.chosen2)
```

```
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = training_data,
##     control = rpart.control(minsplit = 1, minbucket = 18, cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Total.Tokens
##
## Root node error: 123/375 = 0.328
##
## n= 375
```

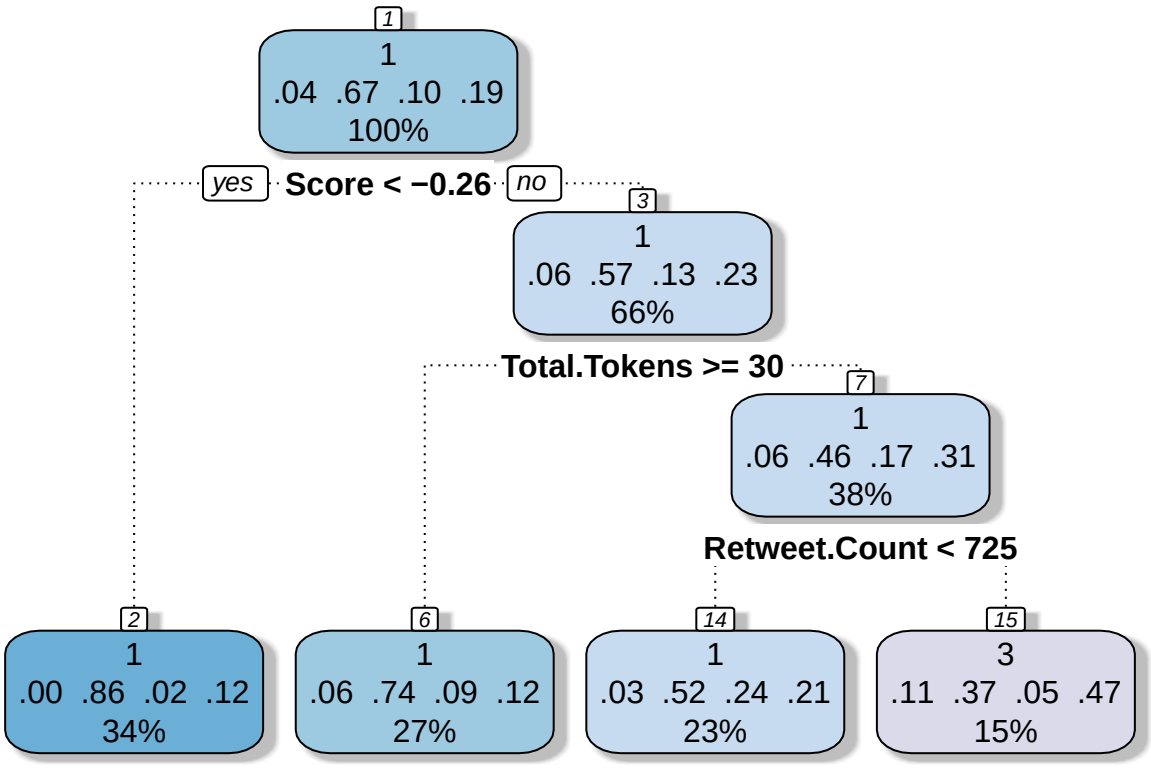
```
##
##        CP nsplit rel error xerror     xstd
## 1 0.02710      0   1.00000 1.0000 0.073915
## 2 0.02439      3   0.91870 1.0081 0.074067
## 3 0.00000      4   0.89431 1.0244 0.074364
```

Although we have a number of possible minsplit and minbucket values, we will choose two classification trees with smallest minsplit

```
# At minsplit = 5

party_tree.chosen1 <- rpart(Party~Retweet.Count+Score+Negativity+Positivity+
                          Uncovered.Tokens+Total.Tokens, validation_set,
                          control = rpart.control(minsplit = 1, minbucket = 17,
                                                  cp = 0))
fancyRpartPlot(party_tree.chosen1, caption = NULL)
printcp(party_tree.chosen1)
```

```
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = validation_set,
##     control = rpart.control(minsplit = 1, minbucket = 17, cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Score         Total.Tokens
##
## Root node error: 41/125 = 0.328
##
## n= 125
##
##        CP nsplit rel error xerror    xstd
## 1 0.01626      0   1.00000 1.0000 0.12802
## 2 0.00000      3   0.95122 1.2439 0.13402
```

```
# At minsplit = 6

party_tree.chosen2 <- rpart(Party~Retweet.Count+Score+Negativity+Positivity+
                          Uncovered.Tokens+Total.Tokens,validation_set,
                          control = rpart.control(minsplit = 1, minbucket = 18,
                                                  cp = 0))
fancyRpartPlot(party_tree.chosen2, caption = NULL)
```




```
printcp(party_tree.chosen2)
```

```
##
## Classification tree:
## rpart(formula = Party ~ Retweet.Count + Score + Negativity +
##     Positivity + Uncovered.Tokens + Total.Tokens, data = validation_set,
##     control = rpart.control(minsplit = 1, minbucket = 18, cp = 0))
##
## Variables actually used in tree construction:
## [1] Retweet.Count Score         Total.Tokens
##
## Root node error: 41/125 = 0.328
##
## n= 125
##
##         CP nsplit rel error xerror    xstd
## 1 0.01626      0   1.00000 1.0000 0.12802
## 2 0.00000      3   0.95122 1.0976 0.13089
```

From this, it correctly classifies the data into a Republican majority twitter users. This could be because we don't have a data representative of the population on twitter, or that there was some kind of bias while collecting this data. If it is the first reason, it could be that since the key role of Republicans, i.e. Trump, is especially active on twitter, more republicans are politically active on twitter than the other parties and therefore, in effect caused bias in this data from twitter.