R Notebook

```r
library(rpart)
library(caret)
suppressMessages(library("tidyverse"))
library(rpart.plot)


shredCVL <- readRDS("capstone dataset/shredCVL")
```

```r
table(shredCVL$bot)
```

```
##
##    0    1
## 2107  511
```

```r
in_train <- createDataPartition(y = shredCVL$bot, p = .75, list = FALSE)
testing_set <- shredCVL[-in_train,]
training_set <- shredCVL[in_train,]

str(training_set)
```

```
## 'data.frame':    1964 obs. of  15 variables:
##  $ screenName          : chr  "15_margiecastro" "2001shaira" "78fb1da7564c40c" "AaaRDieeee" ...
##  $ bot                 : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ statusesCount       : num  3642 2646 4450 NA NA ...
##  $ friendsCount        : num  620 392 113 NA NA 104 929 395 146 432 ...
##  $ followersCount      : num  136 209 15 NA NA ...
##  $ listedCount         : num  1 0 2 NA NA 0 11 13 6 3 ...
##  $ acct_created        : POSIXct, format: "2013-10-20 02:13:13" "2016-01-23 11:10:22" ...
##  $ julianCreated       :Class 'difftime'  atomic [1:1964] 15998 16823 16701 16031 16031 ...
##   .. ..- attr(*, "units")= chr "days"
##  $ acct_age            : num  1687547 499010 675345 1640758 1640758 ...
##  $ langDiv             : num  0.814 0.872 NA 0.866 0.866 ...
##  $ mean_time_betwn_tweets: num  463 189 152 NA NA ...
##  $ App                 : chr  "Twitter for Android" "UnFollowSpy" NA "Twitter for Android" ...
##  $ Count               : int  6238 21 NA 6238 44 5447 NA 1543 NA 5447 ...
##  $ App.BoN             : num  0 0 NA 0 0 0 NA 0 NA 0 ...
##  $ mCount              : int  1 1 1 3 3 1 1 2 1 1 ...
```

```r
table(testing_set$bot)
```

```
##
##   0   1
## 519 135
```

```r
table(training_set$bot)
```

```
##
##    0    1
## 1588  376
```

```r
121/654
```

```
## [1] 0.185
```

```r
390/1964
```

```
## [1] 0.1986
```

```r
# both ratios hold to an 20:80 proportion.
```

```r
model1 <- rpart(formula = bot ~., data = shredCVL, method = "class")

str(model1)
```

```
## List of 15
##  $ frame              :'data.frame': 3 obs. of  9 variables:
##   ..$ var       : Factor w/ 2 levels "<leaf>","screenName": 2 1 1
##   ..$ n         : int [1:3] 2618 2107 511
##   ..$ wt        : num [1:3] 2618 2107 511
##   ..$ dev       : num [1:3] 511 0 0
##   ..$ yval      : num [1:3] 1 1 2
##   ..$ complexity: num [1:3] 1 0.01 0.01
##   ..$ ncompete  : int [1:3] 4 0 0
##   ..$ nsurrogate: int [1:3] 5 0 0
##   ..$ yval2     : num [1:3, 1:6] 1 1 2 2107 2107 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:6] "" "" "" "" ...
##  $ where              : Named int [1:2618] 3 3 3 3 3 3 3 3 3 3 ...
##   ..- attr(*, "names")= chr [1:2618] "1" "2" "3" "4" ...
##  $ call               : language rpart(formula = bot ~ ., data = shredCVL, method = "class")
##  $ terms              :Classes 'terms', 'formula'  language bot ~ screenName + statusesCount + friendsCount +
##   .. ..- attr(*, "variables")= language list(bot, screenName, statusesCount, friendsCount, followersCount,
##   .. ..- attr(*, "factors")= int [1:15, 1:14] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:15] "bot" "screenName" "statusesCount" "friendsCount" ...
##   .. .. .. ..$ : chr [1:14] "screenName" "statusesCount" "friendsCount" "followersCount" ...
##   .. ..- attr(*, "term.labels")= chr [1:14] "screenName" "statusesCount" "friendsCount" "followersCount" ...
##   .. ..- attr(*, "order")= int [1:14] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(bot, screenName, statusesCount, friendsCount, followersCount,
##   .. ..- attr(*, "dataClasses")= Named chr [1:15] "numeric" "character" "numeric" "numeric" ...
##   .. .. ..- attr(*, "names")= chr [1:15] "bot" "screenName" "statusesCount" "friendsCount" ...
##  $ cptable            : num [1:2, 1:5] 1 0.01 0 1 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:5] "CP" "nsplit" "rel error" "xerror" ...
##  $ method             : chr "class"
##  $ parms              :List of 3
##   ..$ prior: num [1:2(1d)] 0.805 0.195
##   .. ..- attr(*, "dimnames")=List of 1
##   .. .. ..$ : chr [1:2] "1" "2"
##   ..$ loss : num [1:2, 1:2] 0 1 1 0
##   ..$ split: num 1
##  $ control            :List of 9
##   ..$ minsplit      : int 20
##   ..$ minbucket     : num 7
##   ..$ cp            : num 0.01
##   ..$ maxcompete    : int 4
##   ..$ maxsurrogate  : int 5
##   ..$ usesurrogate  : int 2
##   ..$ surrogatestyle: int 0
##   ..$ maxdepth      : int 30
##   ..$ xval          : int 10
##  $ functions          :List of 3
##   ..$ summary:function (yval, dev, wt, ylevel, digits)
```

```
##   ..$ print  :function (yval, ylevel, digits)
##   ..$ text   :function (yval, dev, wt, ylevel, digits, n, use.n)
##  $ numresp            : int 4
##  $ splits             : num [1:10, 1:5] 2618 2425 2454 2618 2589 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:10] "screenName" "langDiv" "Count" "mCount" ...
##   .. ..$ : chr [1:5] "count" "ncat" "improve" "index" ...
##  $ csplit             : int [1:2, 1:1630] 1 1 3 3 1 1 1 1 3 1 ...
##  $ variable.importance: Named num [1:6] 822.5 505.4 407.2 67.6 45.1 ...
##   ..- attr(*, "names")= chr [1:6] "screenName" "langDiv" "Count" "App" ...
##  $ y                  : int [1:2618] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ordered            : Named logi [1:14] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..- attr(*, "names")= chr [1:14] "screenName" "statusesCount" "friendsCount" "followersCount" ...
##  - attr(*, "xlevels")=List of 2
##   ..$ screenName: chr [1:1630] "_AlexisWasHere" "_alyssaeguia" "_American_Loser" "_EderMoura" ...
##   ..$ App       : chr [1:87] "2016TWGtwit" "Ask.fm" "AT-Falcon.Cab" "autoposta16" ...
##  - attr(*, "ylevels")= chr [1:2] "0" "1"
##  - attr(*, "class")= chr "rpart"
```

```r
print(model1)
```

```
## n= 2618
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 2618 511 0 (0.8048 0.1952)
##   2) screenName=_AlexisWasHere,_American_Loser,_EderMoura,_Hiraku_,_meghanmckenna_,_sirtainly,11AliveNews,123
##   3) screenName=_alyssaeguia,_FeelingsPosts,_kitkatx,_PayoNiJuan_,15_margiecastro,2001shaira,78fb1da7564c40c,
```

```r
formula <- bot ~ statusesCount + friendsCount + followersCount + listedCount + acct_age + langDiv + mean_time_be

model2 <- rpart(formula, data = shredCVL, method = "class")

str(model2)
```

```
## List of 14
##  $ frame              :'data.frame': 13 obs. of  9 variables:
##   ..$ var       : Factor w/ 6 levels "<leaf>","acct_age",..: 4 5 1 6 6 1 1 2 1 3 ...
##   ..$ n         : int [1:13] 2618 2292 1661 631 525 516 9 106 28 78 ...
##   ..$ wt        : num [1:13] 2618 2292 1661 631 525 ...
##   ..$ dev       : num [1:13] 511 189 66 123 70 61 0 53 1 26 ...
##   ..$ yval      : num [1:13] 1 1 1 1 1 1 2 1 1 2 ...
##   ..$ complexity: num [1:13] 0.622 0.017 0 0.017 0.017 ...
##   ..$ ncompete  : int [1:13] 4 4 0 4 4 0 0 4 0 4 ...
##   ..$ nsurrogate: int [1:13] 1 5 0 3 1 0 0 2 0 2 ...
##   ..$ yval2     : num [1:13, 1:6] 1 1 1 1 1 1 2 1 1 2 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:6] "" "" "" "" ...
##  $ where              : Named int [1:2618] 13 13 6 13 13 13 13 13 6 13 ...
##   ..- attr(*, "names")= chr [1:2618] "1" "2" "3" "4" ...
##  $ call               : language rpart(formula = formula, data = shredCVL, method = "class")
##  $ terms              :Classes 'terms', 'formula'  language bot ~ statusesCount + friendsCount + followersCou
##   .. ..- attr(*, "variables")= language list(bot, statusesCount, friendsCount, followersCount, listedCount,
##   .. ..- attr(*, "factors")= int [1:9, 1:8] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:9] "bot" "statusesCount" "friendsCount" "followersCount" ...
##   .. .. .. ..$ : chr [1:8] "statusesCount" "friendsCount" "followersCount" "listedCount" ...
##   .. ..- attr(*, "term.labels")= chr [1:8] "statusesCount" "friendsCount" "followersCount" "listedCount" ...
```

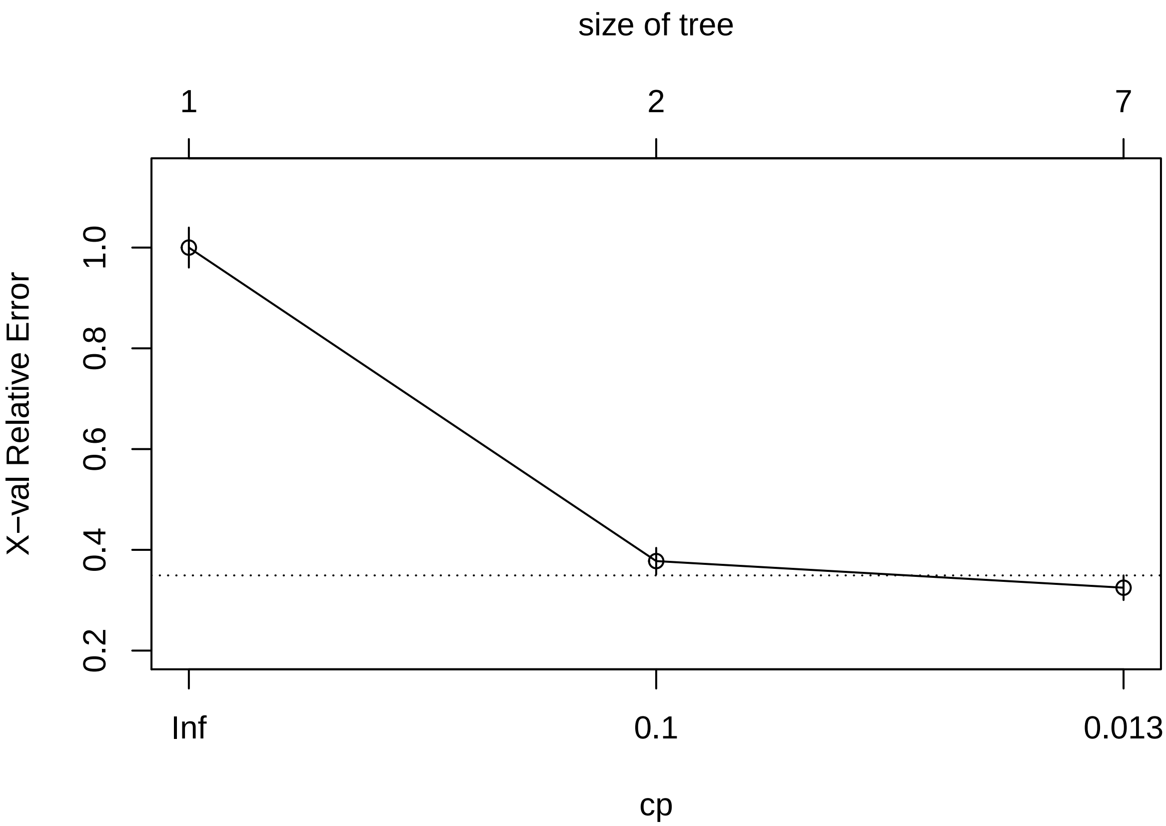
```
##   .. ..- attr(*, "order")= int [1:8] 1 1 1 1 1 1 1 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(bot, statusesCount, friendsCount, followersCount, listedCount,
##   .. ..- attr(*, "dataClasses")= Named chr [1:9] "numeric" "numeric" "numeric" "numeric" ...
##   .. .. ..- attr(*, "names")= chr [1:9] "bot" "statusesCount" "friendsCount" "followersCount" ...
##  $ cptable            : num [1:3, 1:5] 0.622 0.017 0.01 0 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "1" "2" "3"
##   .. ..$ : chr [1:5] "CP" "nsplit" "rel error" "xerror" ...
##  $ method             : chr "class"
##  $ parms              :List of 3
##   ..$ prior: num [1:2(1d)] 0.805 0.195
##   .. ..- attr(*, "dimnames")=List of 1
##   .. .. ..$ : chr [1:2] "1" "2"
##   ..$ loss : num [1:2, 1:2] 0 1 1 0
##   ..$ split: num 1
##  $ control            :List of 9
##   ..$ minsplit      : int 20
##   ..$ minbucket     : num 7
##   ..$ cp            : num 0.01
##   ..$ maxcompete    : int 4
##   ..$ maxsurrogate  : int 5
##   ..$ usesurrogate  : int 2
##   ..$ surrogatestyle: int 0
##   ..$ maxdepth      : int 30
##   ..$ xval          : int 10
##  $ functions          :List of 3
##   ..$ summary:function (yval, dev, wt, ylevel, digits)
##   ..$ print  :function (yval, ylevel, digits)
##   ..$ text   :function (yval, dev, wt, ylevel, digits, n, use.n)
##  $ numresp            : int 4
##  $ splits             : num [1:44, 1:5] 2425 2618 2589 2580 2589 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:44] "langDiv" "mCount" "listedCount" "mean_time_betwn_tweets" ...
##   .. ..$ : chr [1:5] "count" "ncat" "improve" "index" ...
##  $ variable.importance: Named num [1:8] 544.7 42.2 41.3 25.1 22.7 ...
##   ..- attr(*, "names")= chr [1:8] "langDiv" "statusesCount" "mCount" "followersCount" ...
##  $ y                  : int [1:2618] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ordered            : Named logi [1:8] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..- attr(*, "names")= chr [1:8] "statusesCount" "friendsCount" "followersCount" "listedCount" ...
##  - attr(*, "xlevels")= Named list()
##  - attr(*, "ylevels")= chr [1:2] "0" "1"
##  - attr(*, "class")= chr "rpart"
```

```r
print(model2)
```

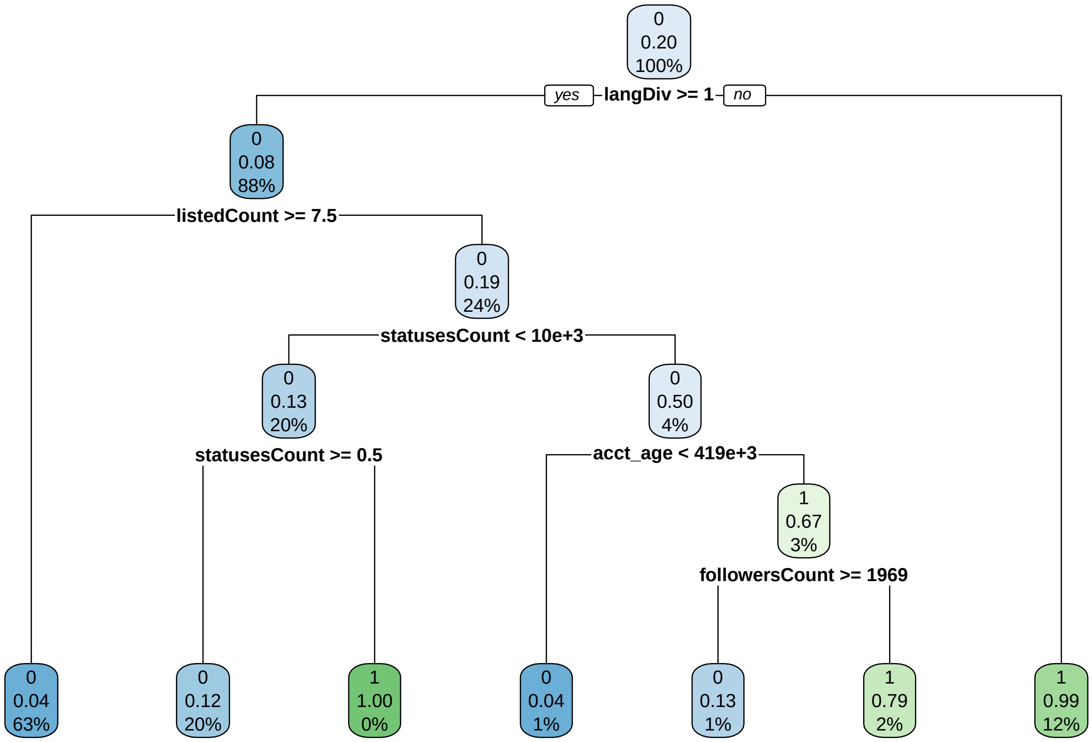
```
## n= 2618
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2618 511 0 (0.80481 0.19519)
##    2) langDiv>=1.035 2292 189 0 (0.91754 0.08246)
##      4) listedCount>=7.5 1661  66 0 (0.96026 0.03974) *
##      5) listedCount< 7.5 631 123 0 (0.80507 0.19493)
##       10) statusesCount< 1.007e+04 525  70 0 (0.86667 0.13333)
##         20) statusesCount>=0.5 516  61 0 (0.88178 0.11822) *
##         21) statusesCount< 0.5 9   0 1 (0.00000 1.00000) *
```

```
##       11) statusesCount>=1.007e+04 106  53 0 (0.50000 0.50000)
##         22) acct_age< 4.188e+05 28   1 0 (0.96429 0.03571) *
##         23) acct_age>=4.188e+05 78  26 1 (0.33333 0.66667)
##           46) followersCount>=1969 15   2 0 (0.86667 0.13333) *
##           47) followersCount< 1969 63  13 1 (0.20635 0.79365) *
##    3) langDiv< 1.035 326   4 1 (0.01227 0.98773) *
```

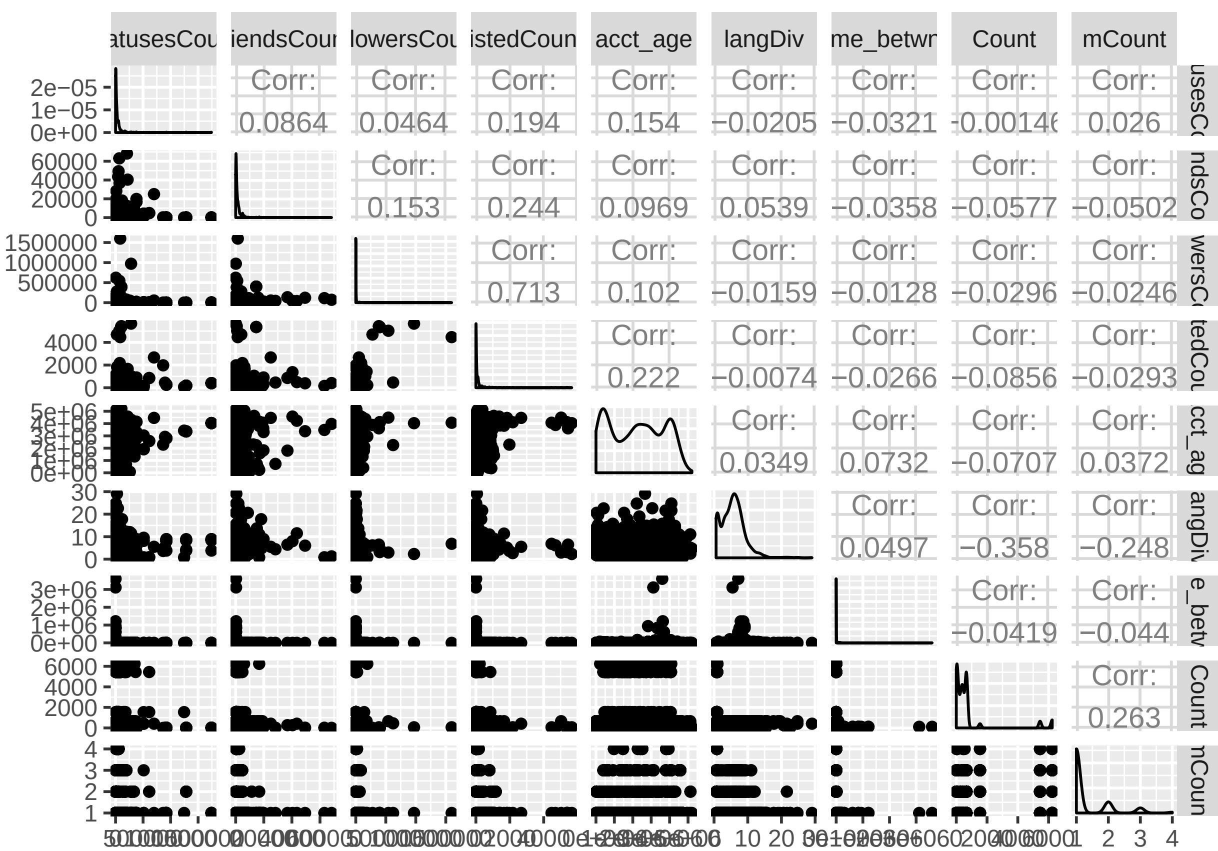
```r
plotcp(model2)
```



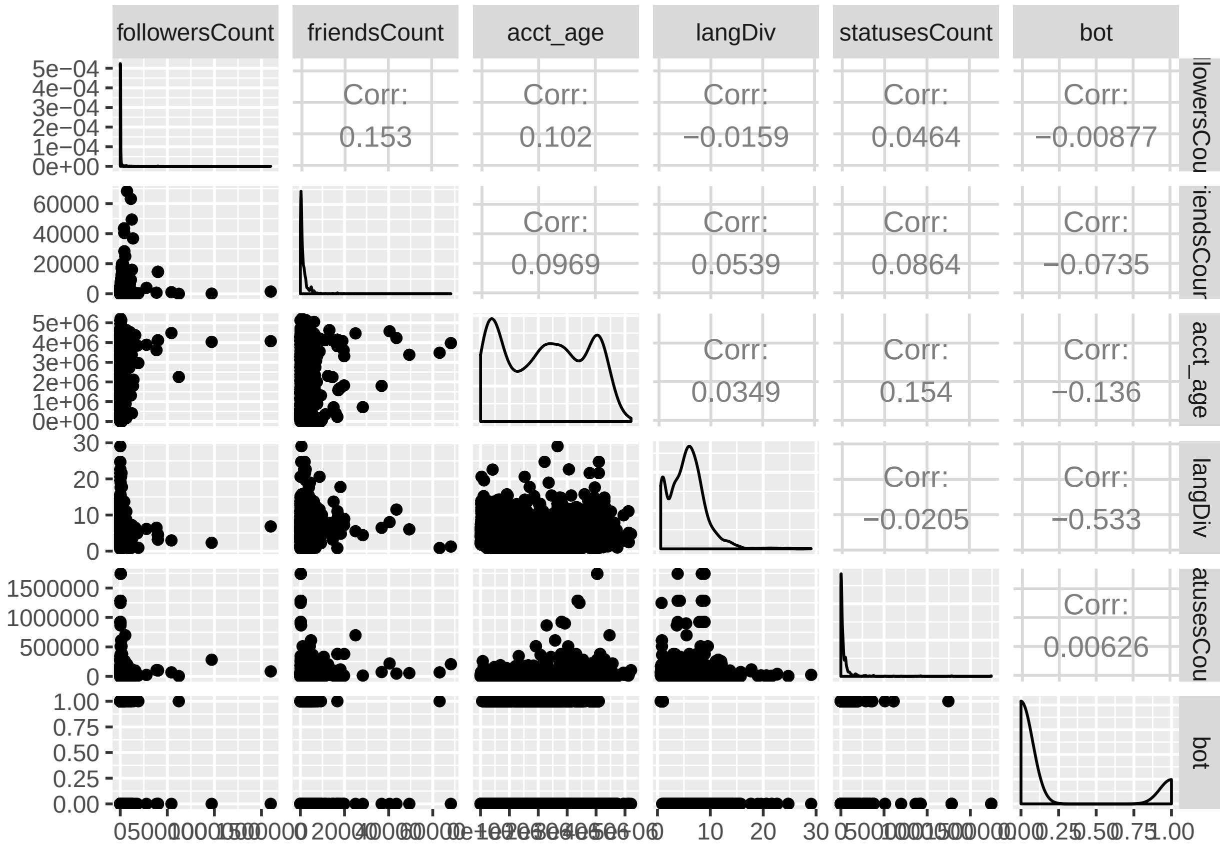

```r
rpart.plot(model2)
```



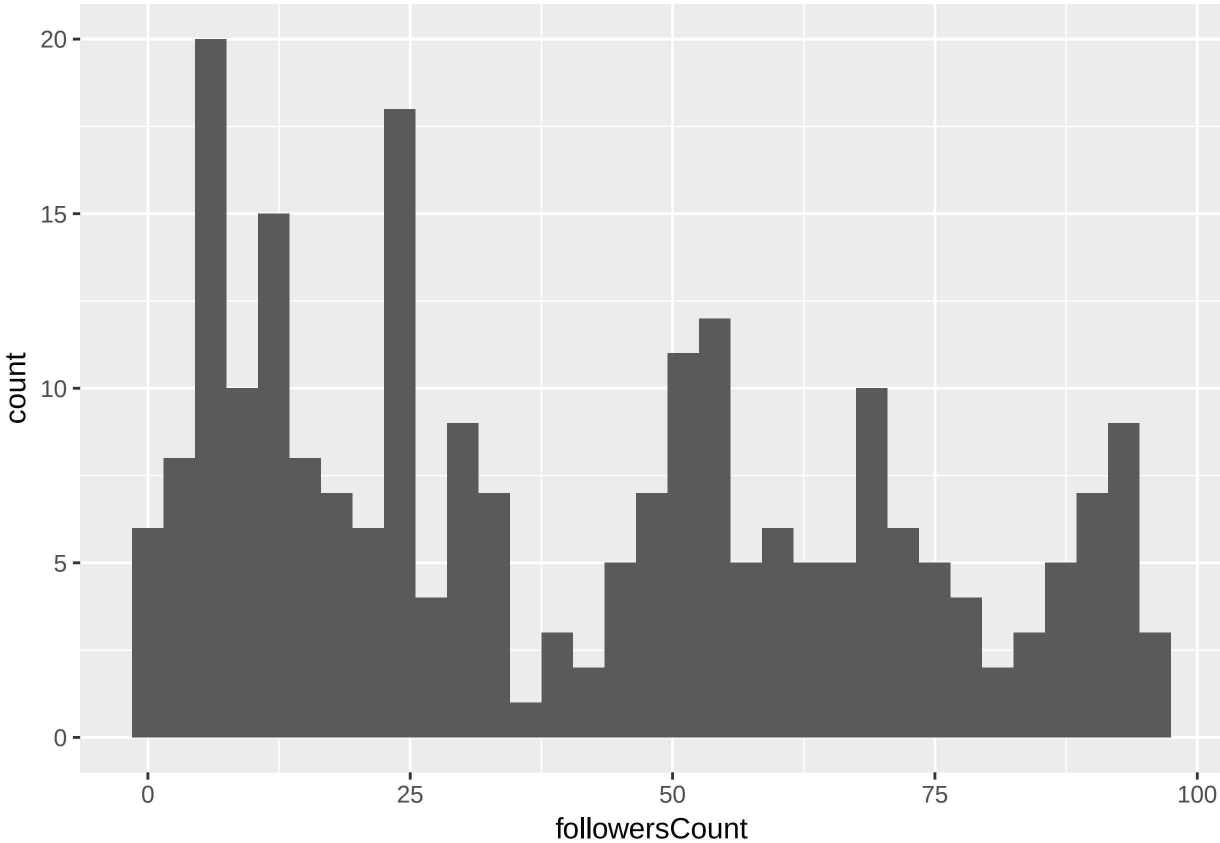

```r
library(GGally)

ggpairs(training_set[,c(-1,-2,-7,-8,-12,-14)])
```



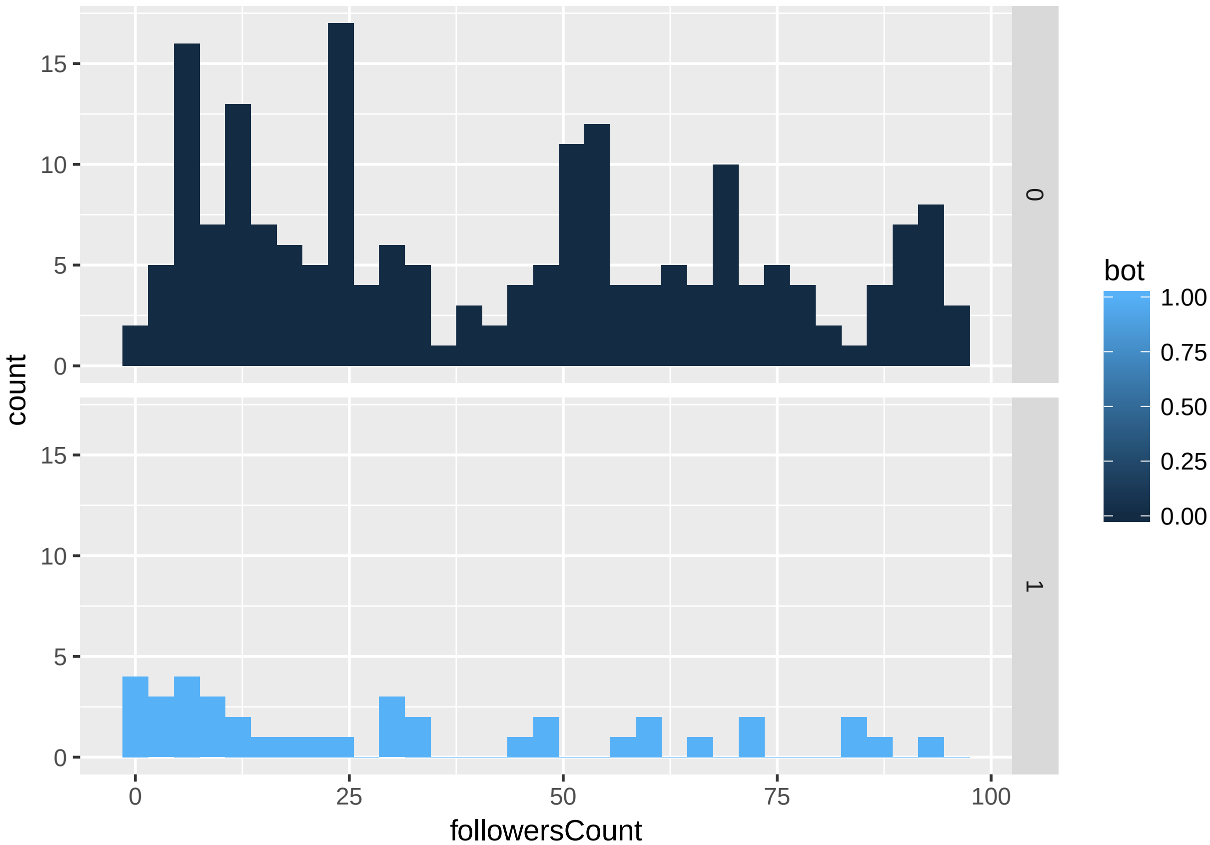
```
ggpairs(training_set[ , c('followersCount', 'friendsCount', 'acct_age',
                    'langDiv', 'statusesCount', 'bot')])
```



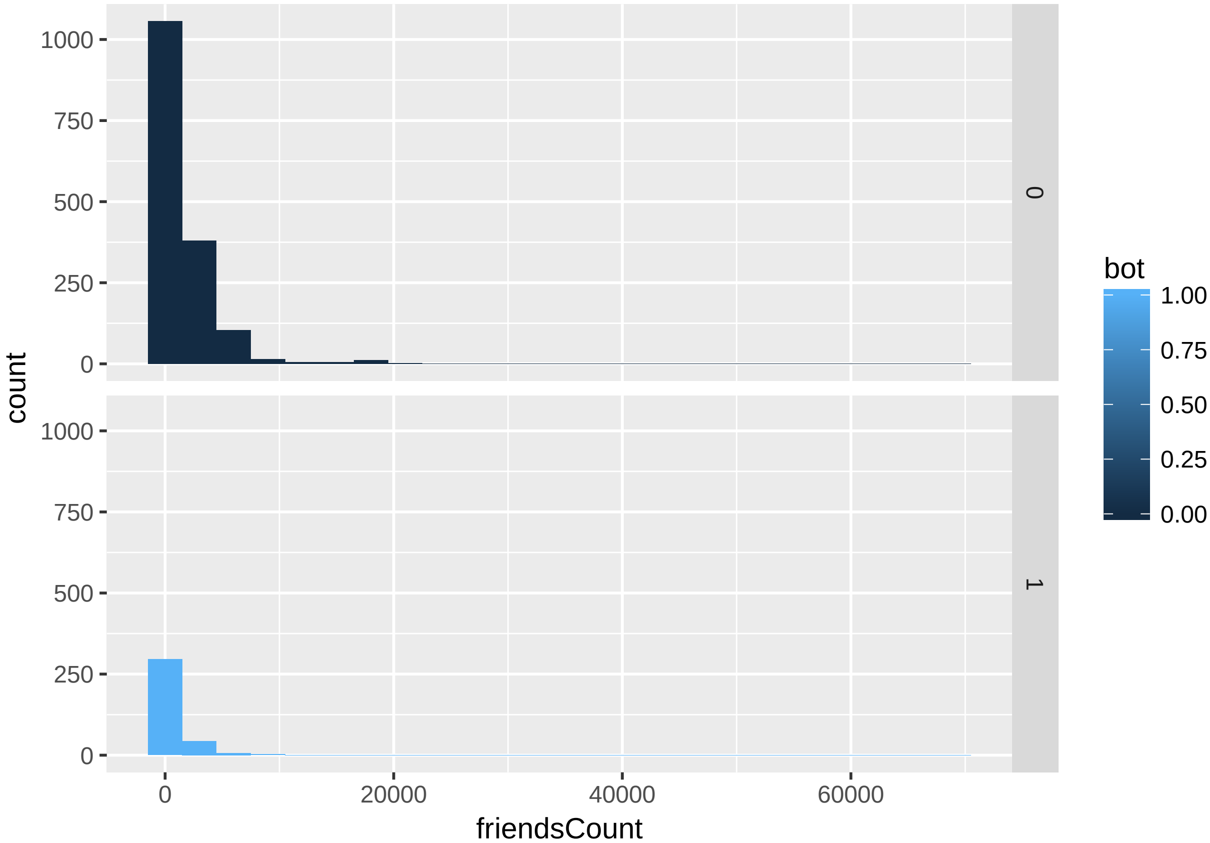

```
ggplot(filter(training_set, followersCount < 100),
       aes(x = followersCount, fill = bot)) +
  geom_histogram(binwidth = 3)
```



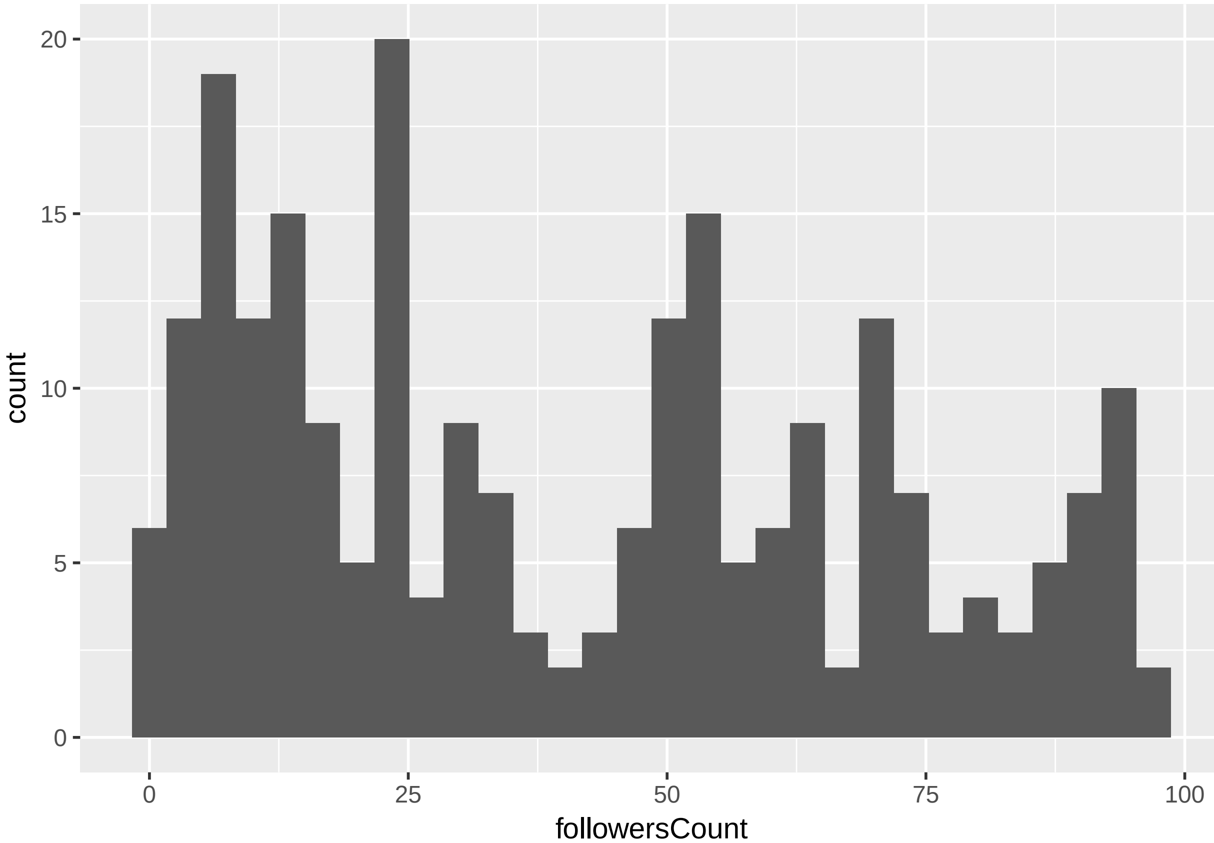

```r
ggplot(filter(training_set, followersCount < 100),
       aes(x = followersCount, fill = bot)) +
  geom_histogram(binwidth = 3) +
  facet_grid(bot ~.)
```



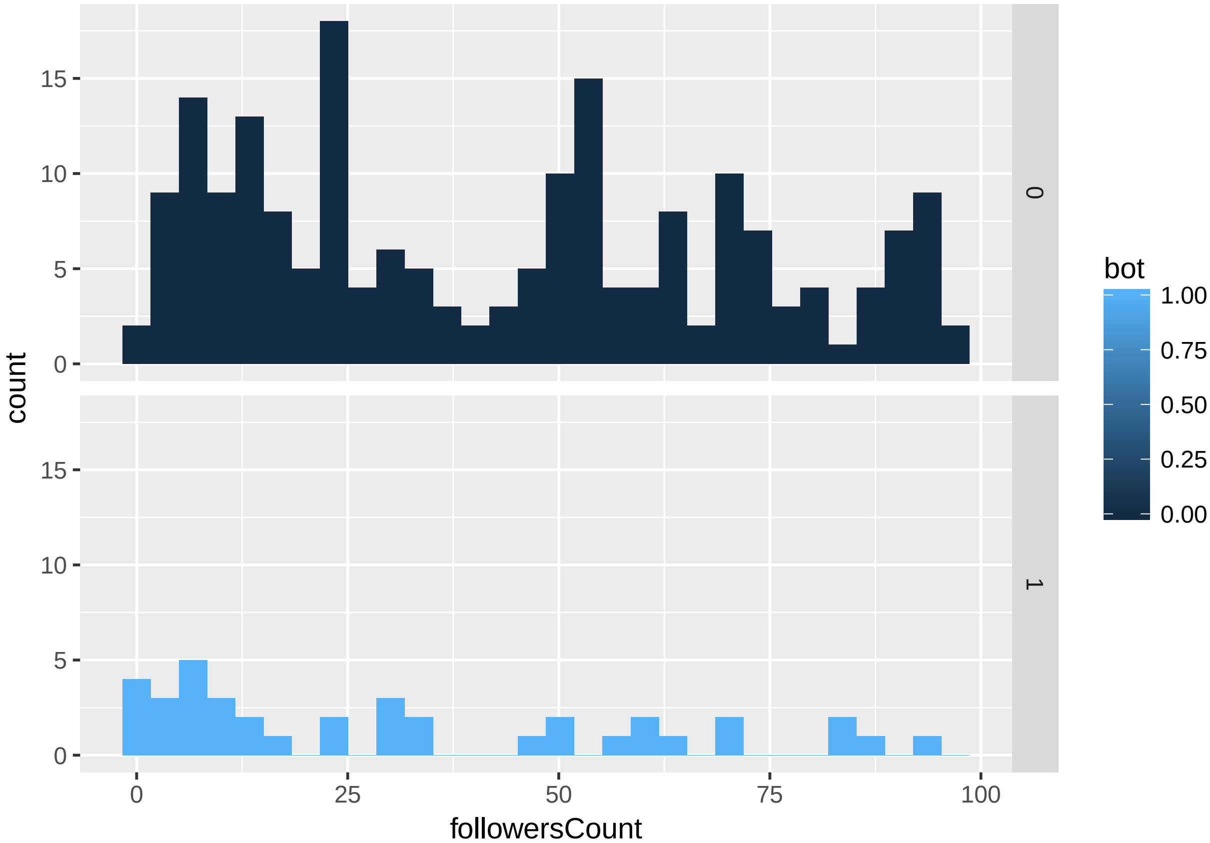

```r
# how about the number of people they follow?
ggplot(training_set, aes(x = friendsCount, fill = bot)) +
  geom_histogram(binwidth = 3000) +
  facet_grid(bot ~.)
```



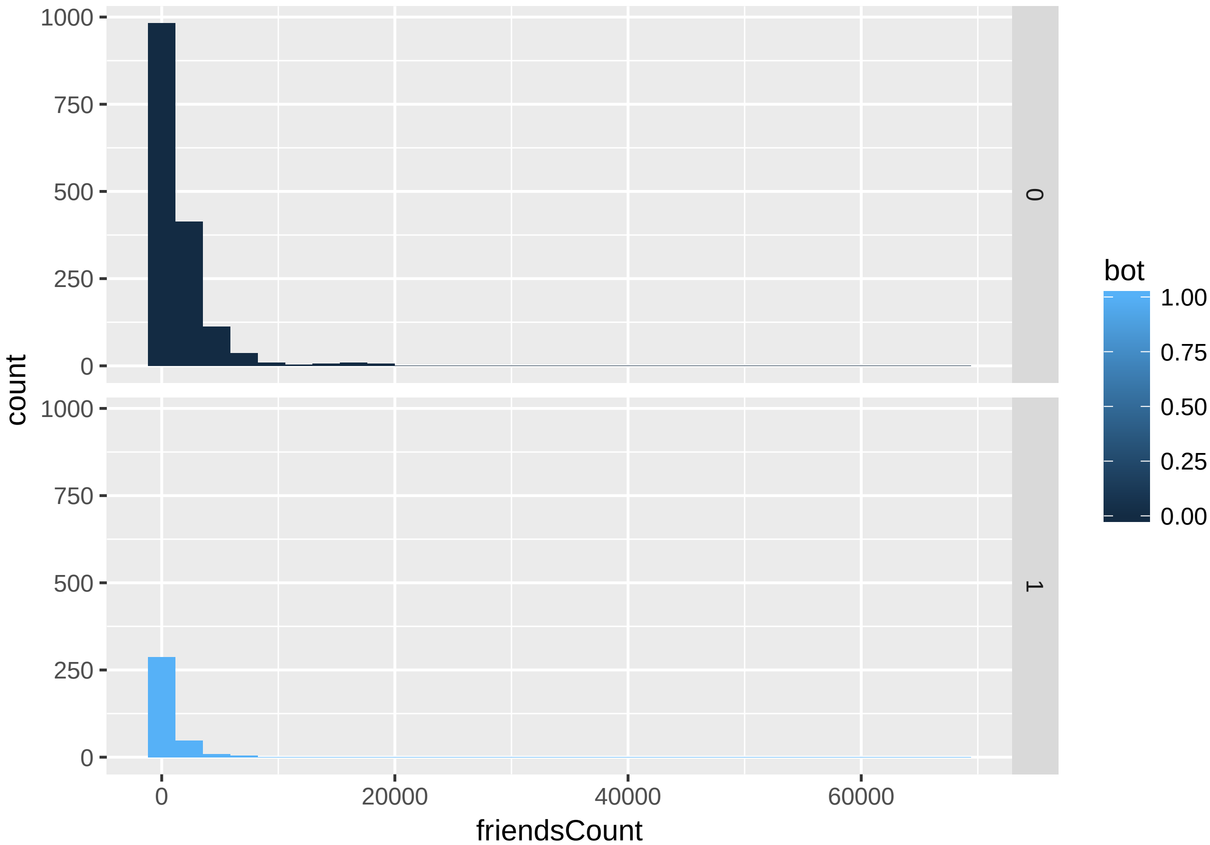

```r
# Some people have a lot of followers, but most don't. we need to lob off
# the long tail so we can see the distribution better
ggplot(filter(training_set, followersCount < 100),
       aes(x = followersCount, fill = bot)) +
  geom_histogram()
```



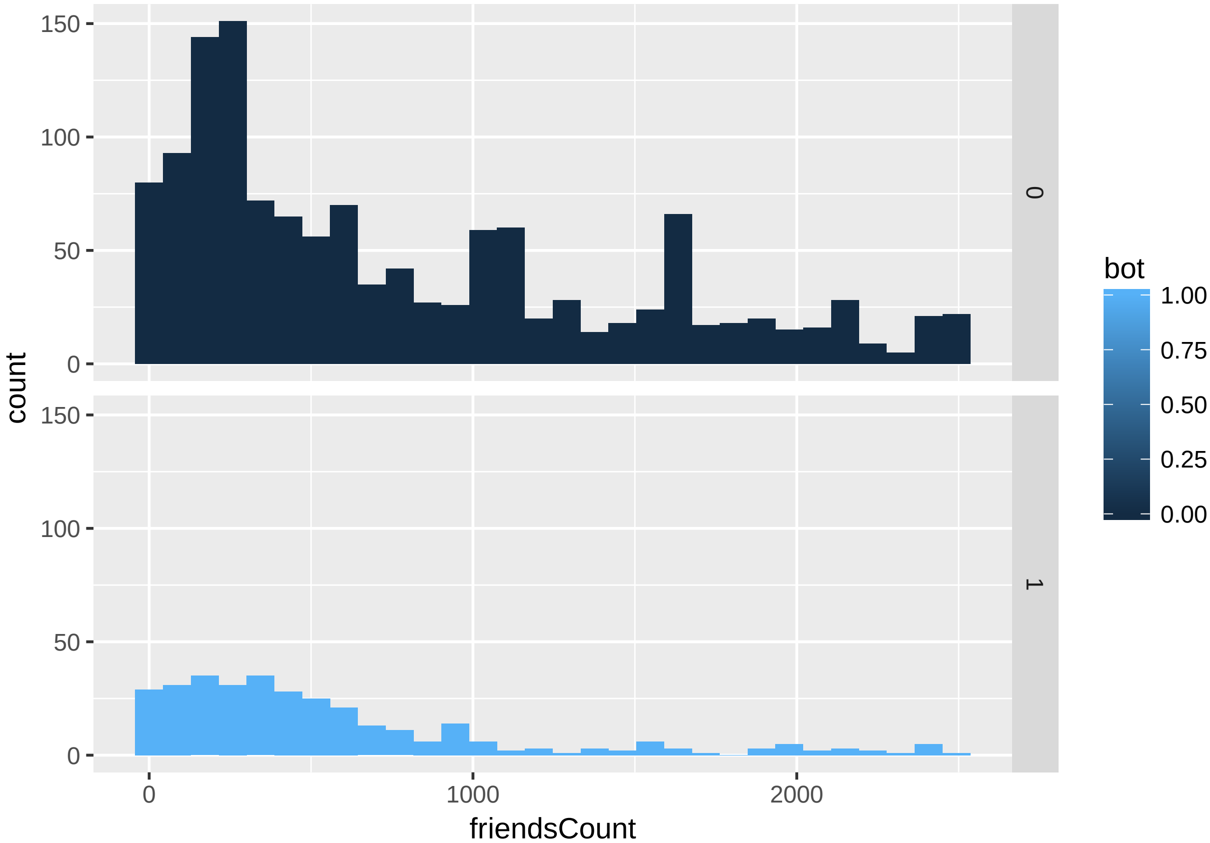

```
ggplot(filter(training_set, followersCount < 100),
       aes(x = followersCount, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



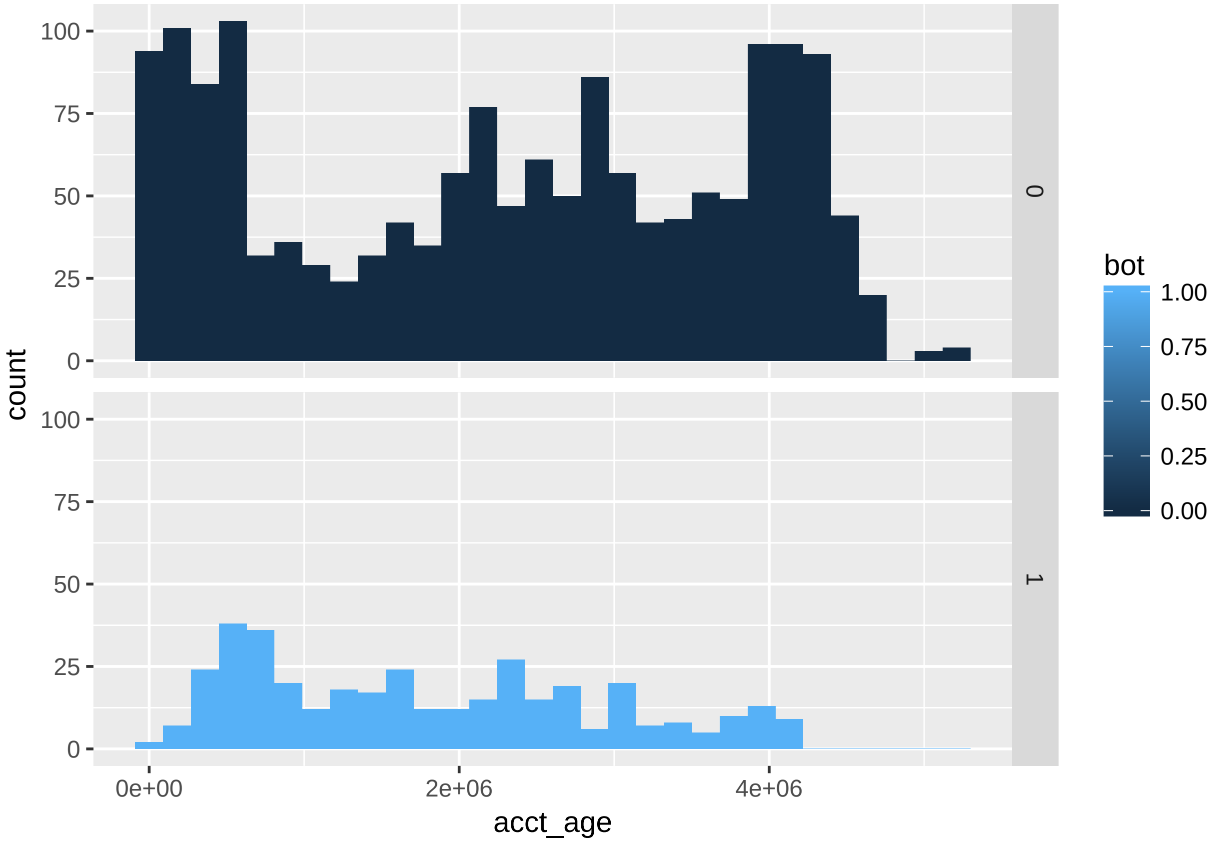

```r
# how about the number of people they follow?
ggplot(training_set, aes(x = friendsCount, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



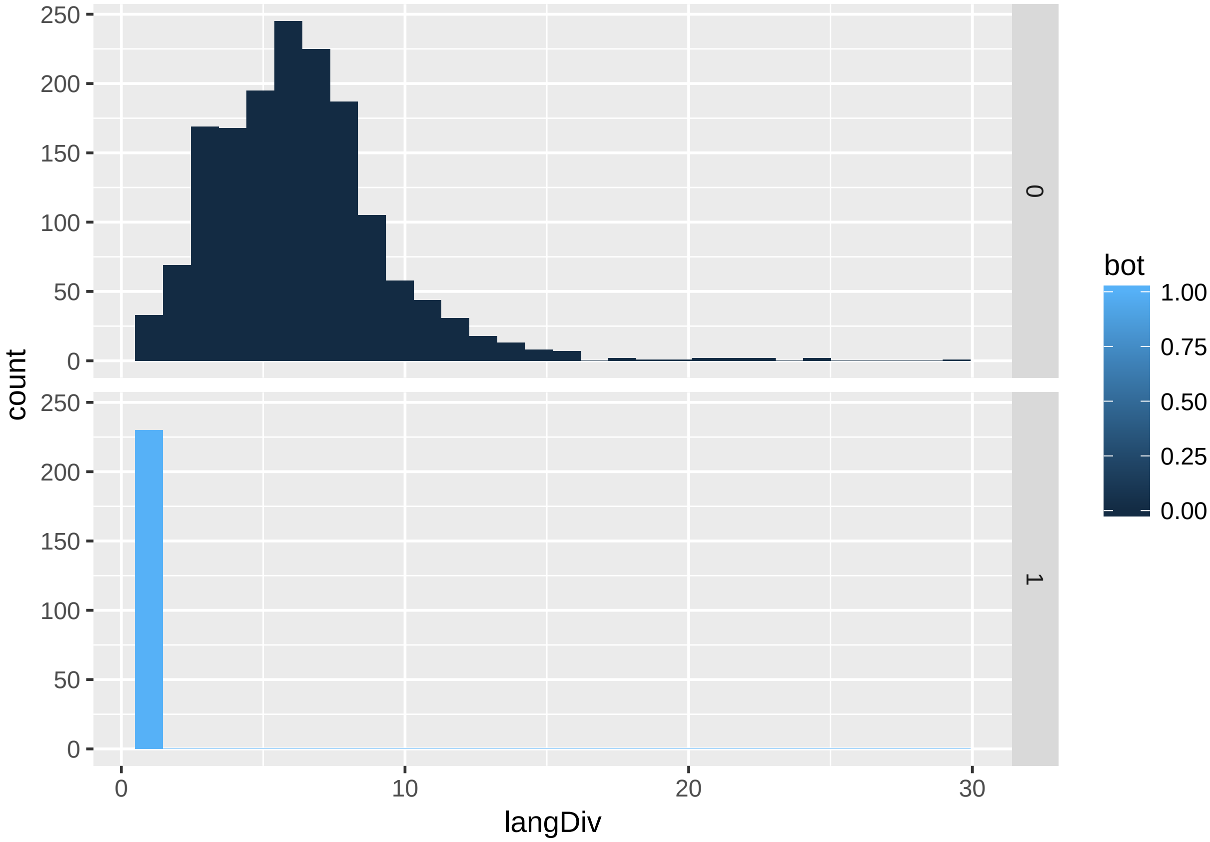

```r
# it's a little hard to see
ggplot(filter(training_set, friendsCount < 2500),
       aes(x = friendsCount, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



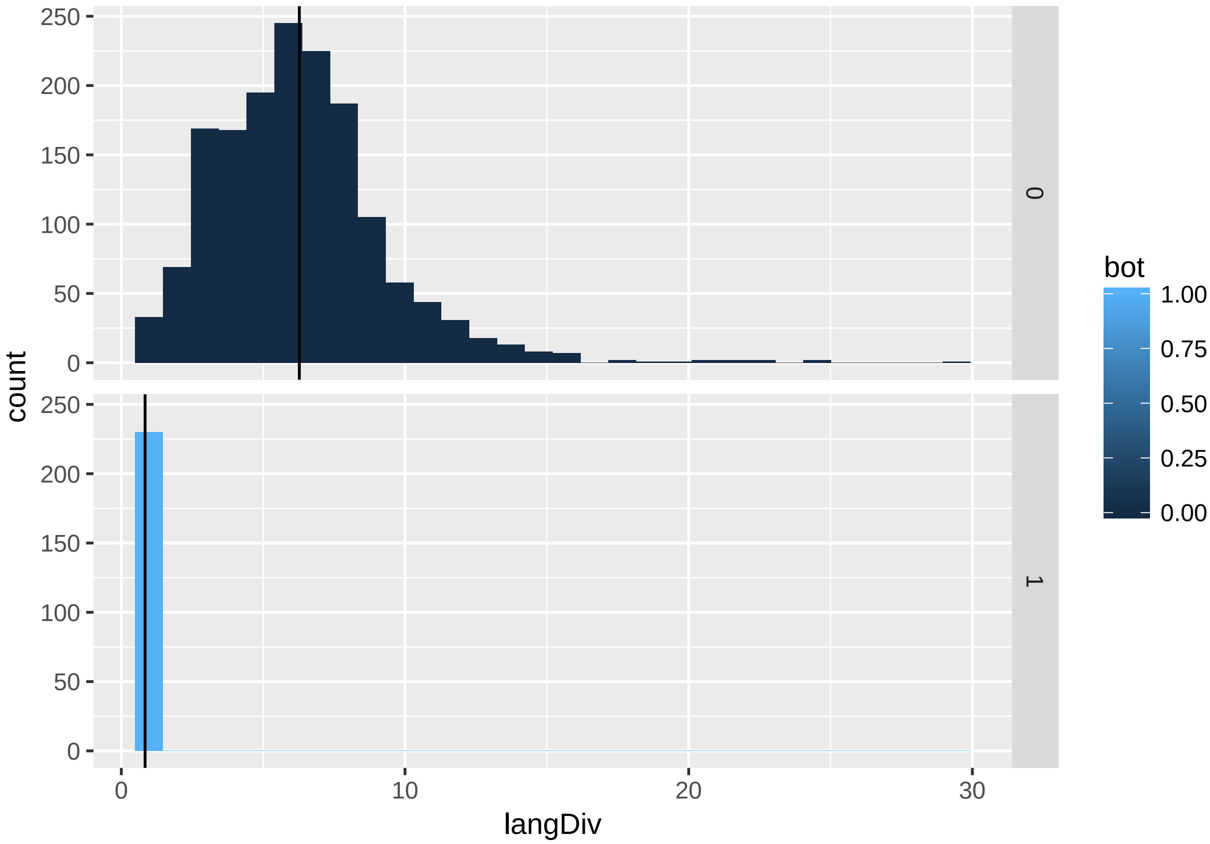

```r
# what about account age?
ggplot(training_set, aes(x = acct_age, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```




```r
# lexical diversity
ggplot(training_set, aes(x = langDiv, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



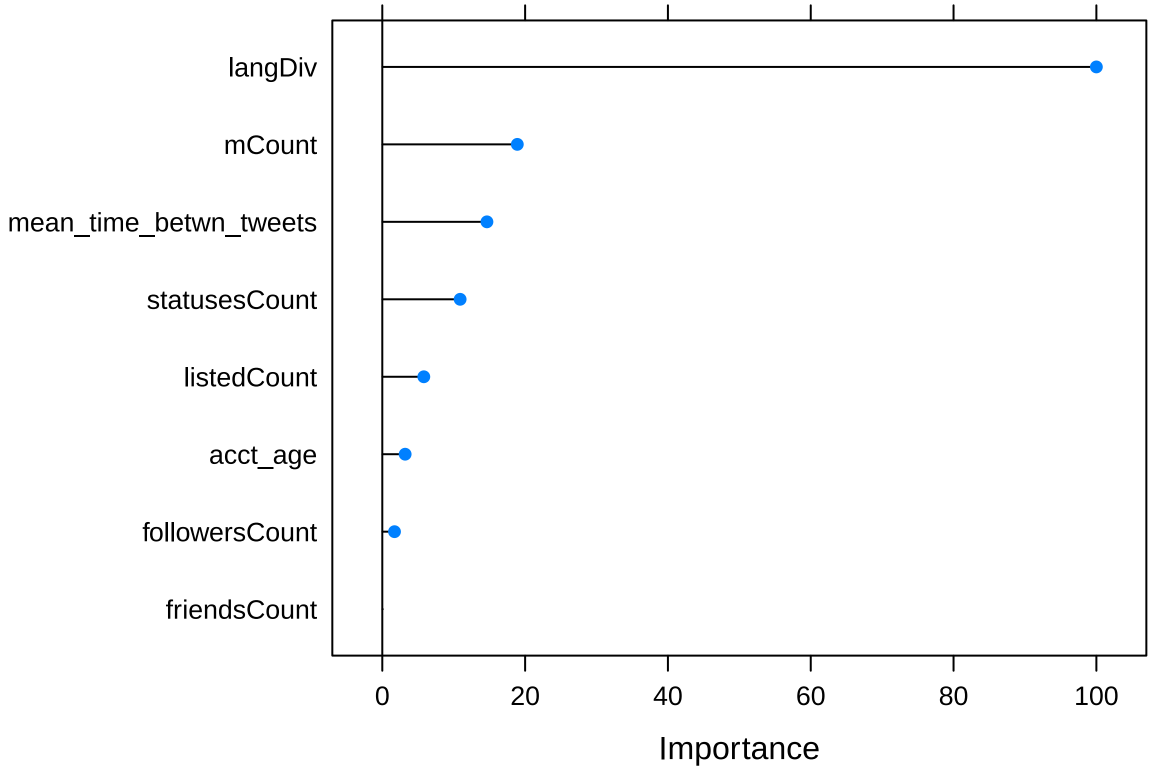

```
# what are the average values?
meanDiv =
  training_set %>%
    group_by(bot) %>%
    summarize(meanDiv = mean(langDiv, na.rm = TRUE))

# add it to the plot
ggplot(training_set, aes(x = langDiv, fill = bot)) +
  geom_histogram() +
  geom_vline(data = meanDiv, aes(xintercept = meanDiv)) +
  facet_grid(bot ~.)
```




```
bagged_model = train(formula,
                    method = 'treebag',
                    data = training_set, na.action = na.omit)

print(bagged_model)
```

```
## Bagged CART
##
## 1964 samples
##    8 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1810, 1810, 1810, 1810, 1810, 1810, ...
## Resampling results:
##
##   RMSE     Rsquared  MAE
##   0.04077  0.9827    0.003533
```

```
plot(varImp(bagged_model))
```



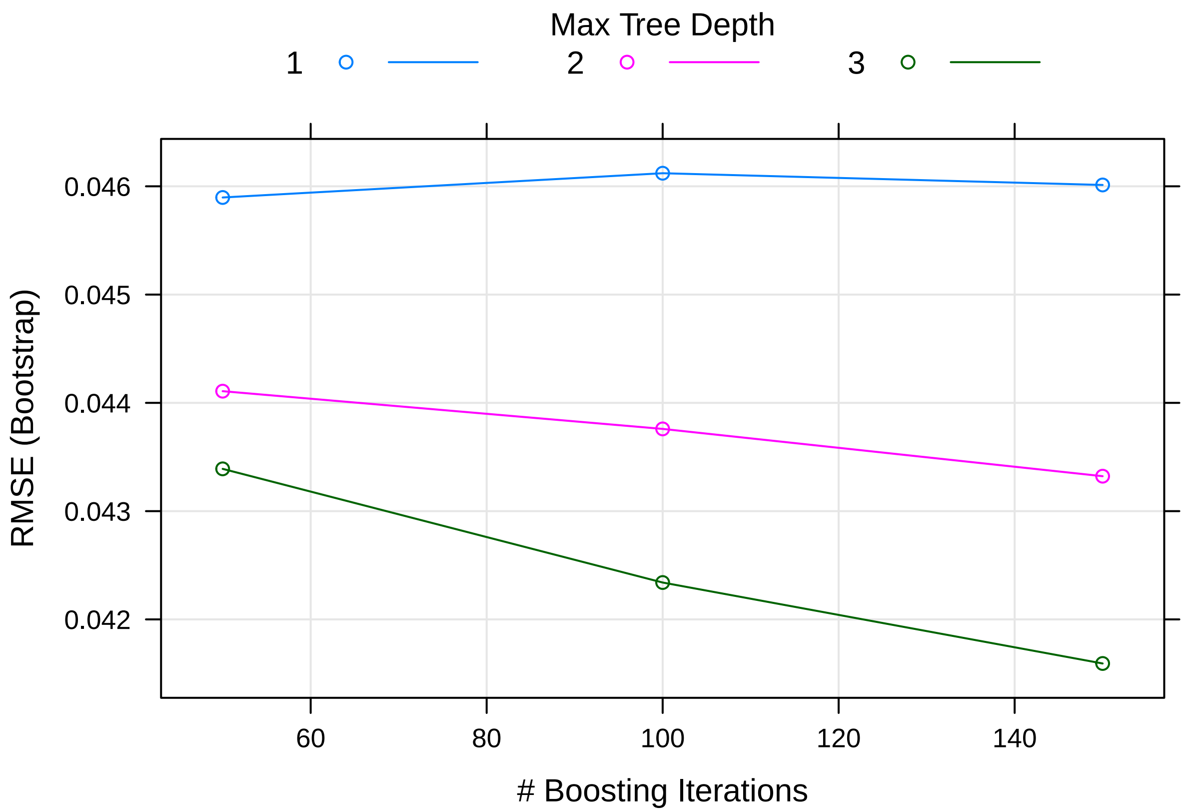

```r
bagged_predictions = predict(bagged_model, testing_set)
#confusionMatrix(bagged_predictions, testing_set$bot)
```

```r
boost_model = train(formula,
                    method = 'gbm',
                    data = training_set,
                    verbose = FALSE, na.action = na.omit)

print(boost_model)
```

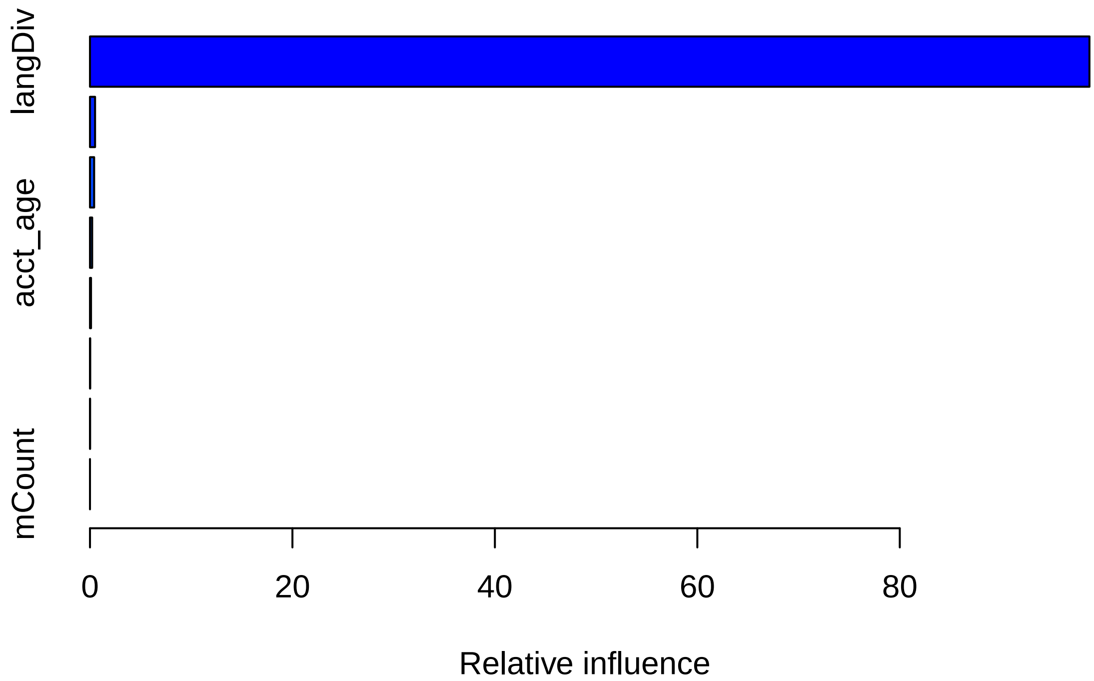
```
## Stochastic Gradient Boosting
##
## 1964 samples
##    8 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1810, 1810, 1810, 1810, 1810, 1810, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE     Rsquared  MAE
##   1                   50      0.04590  0.9787    0.006206
##   1                  100      0.04612  0.9786    0.006280
##   1                  150      0.04601  0.9787    0.006168
##   2                   50      0.04411  0.9801    0.006100
##   2                  100      0.04376  0.9804    0.006351
##   2                  150      0.04332  0.9809    0.006603
##   3                   50      0.04339  0.9806    0.005581
##   3                  100      0.04234  0.9814    0.005501
##   3                  150      0.04159  0.9819    0.005503
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
```

```
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
plot(boost_model)
```




```r
#plot(varImp(boost_model))

summary(boost_model$finalModel)
```




```
##                                            var    rel.inf
## langDiv                                 langDiv 98.736687
## mean_time_betwn_tweets mean_time_betwn_tweets  0.502002
## statusesCount                     statusesCount  0.411862
## acct_age                               acct_age  0.216090
## followersCount                   followersCount  0.102470
## friendsCount                       friendsCount  0.021207
## listedCount                         listedCount  0.009681
## mCount                                   mCount  0.000000
```

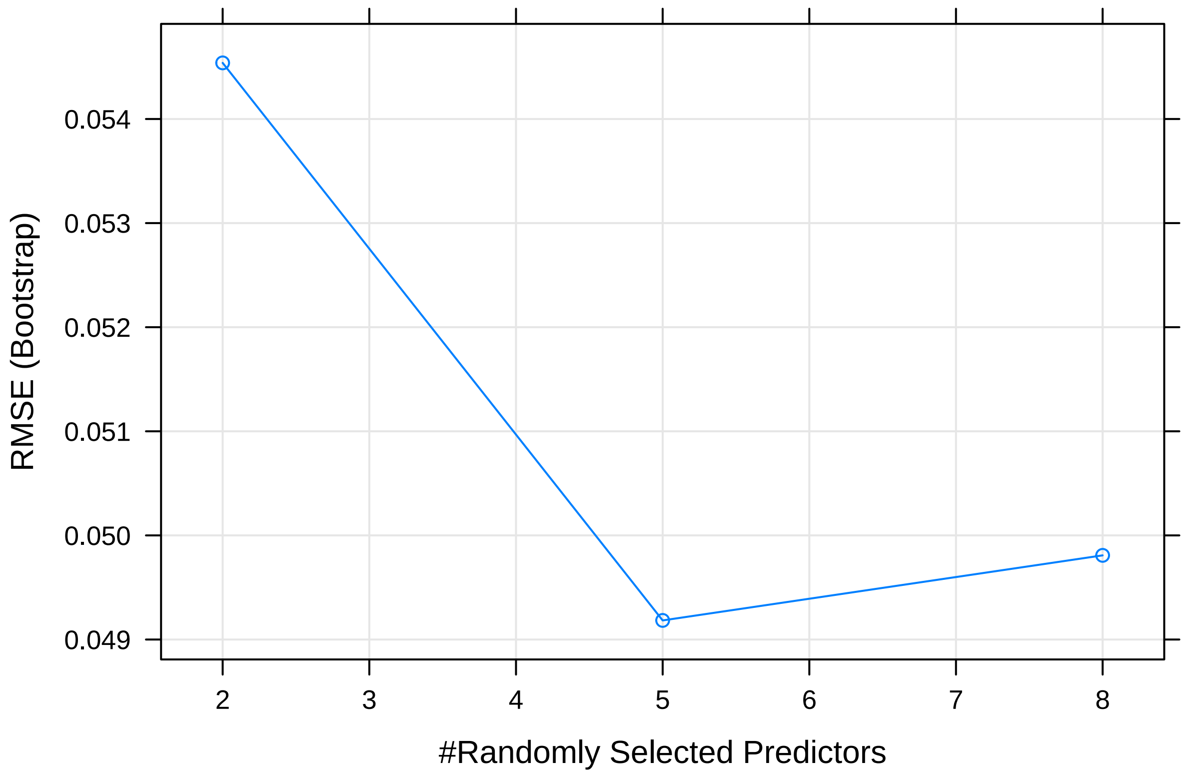
```
# predict
boost_predictions = predict(boost_model, testing_set)
#confusionMatrix(boost_predictions, testing_set$bot)
```

```
rf_model = train(formula,
                 data = training_set,
                 method = 'rf',
                 prox = TRUE,
                 verbose = TRUE, na.action = na.omit)


print(rf_model)
```

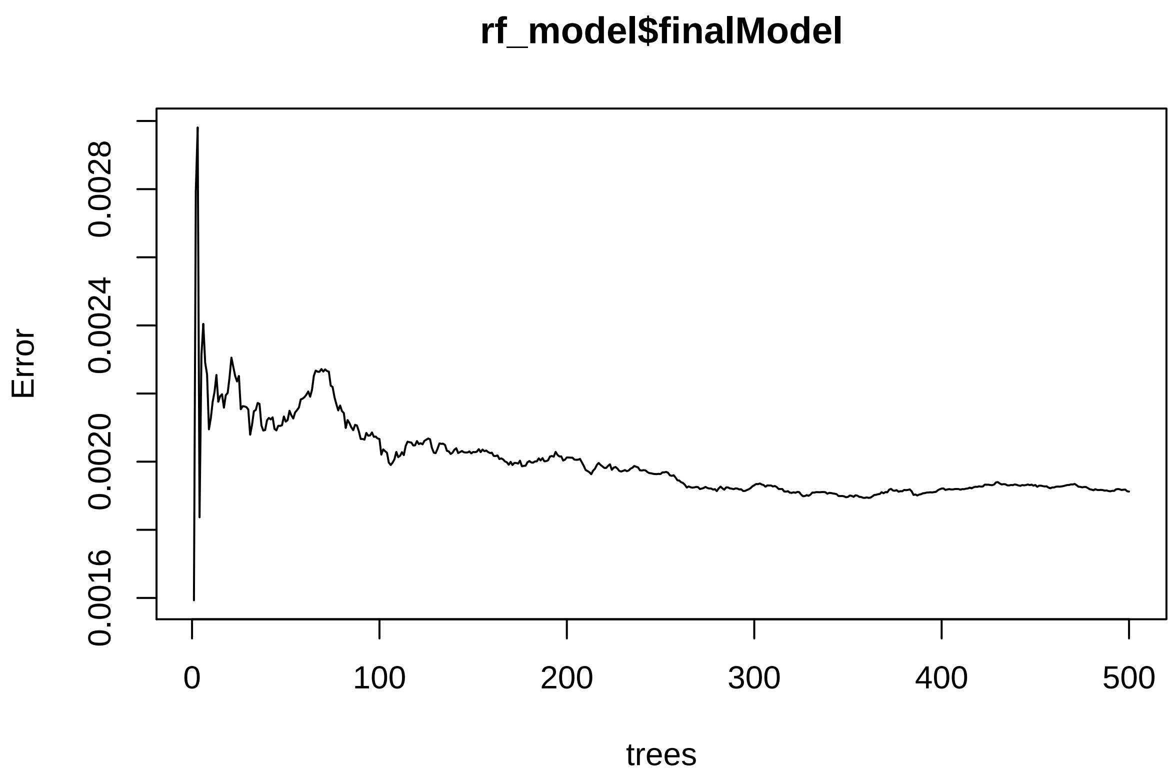
```
## Random Forest
##
## 1964 samples
##    8 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1810, 1810, 1810, 1810, 1810, 1810, ...
## Resampling results across tuning parameters:
```

```
##
##   mtry  RMSE     Rsquared  MAE
##   2     0.05454  0.9749    0.014923
##   5     0.04918  0.9773    0.004544
##   8     0.04981  0.9764    0.003623
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 5.
```

```r
plot(rf_model)
```




```r
plot(rf_model$finalModel)
```

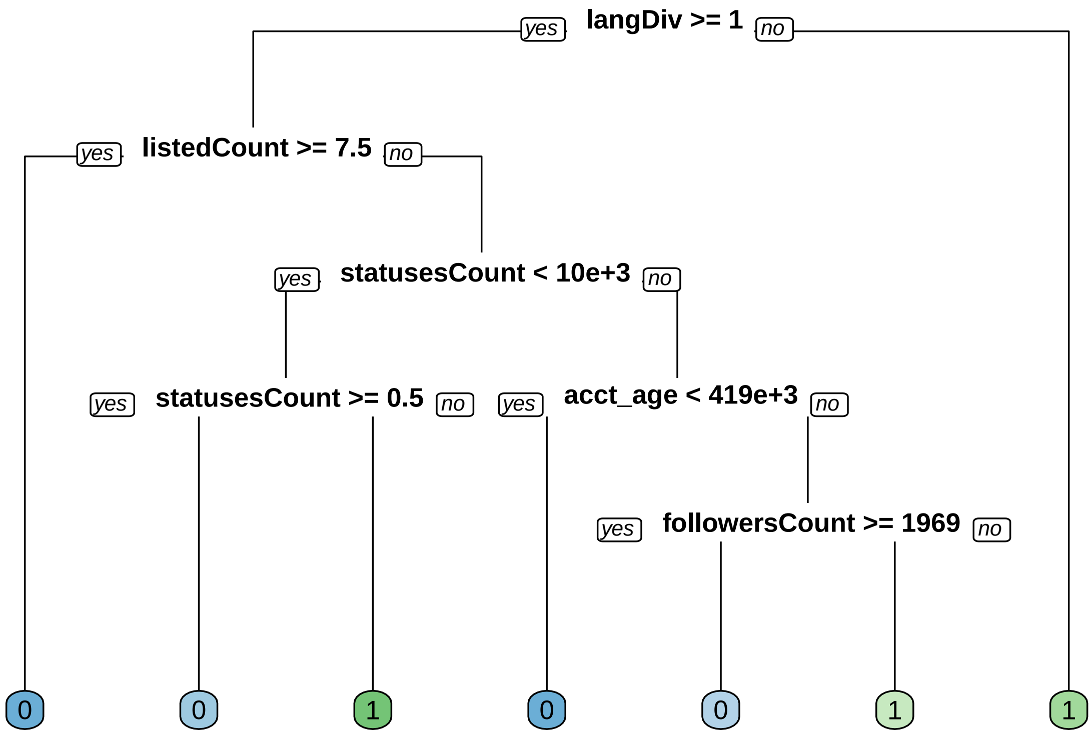



```r
# pull a tree out of the forest
head(randomForest::getTree(rf_model$finalModel, k = 5, labelVar = TRUE))
```

```
##   left daughter right daughter      split var split point status
## 1             2              3        langDiv       1.035     -3
## 2             4              5 followersCount      31.500     -3
## 3             0              0           <NA>       0.000     -1
## 4             0              0           <NA>       0.000     -1
## 5             6              7    listedCount      12.000     -3
## 6             8              9  statusesCount   23703.500     -3
##   prediction
## 1  1.276e-01
## 2  9.957e-01
## 3 -1.943e-15
## 4  0.000e+00
## 5  1.000e+00
## 6  1.000e+00
```

```r
# predict
rf_predictions = predict(rf_model, testing_set)
#confusionMatrix(rf_predictions, testing_set$bot)
```

```r
# Display the results
rpart.plot(x = model2, yesno = 2, type = 0, extra = 0)
```



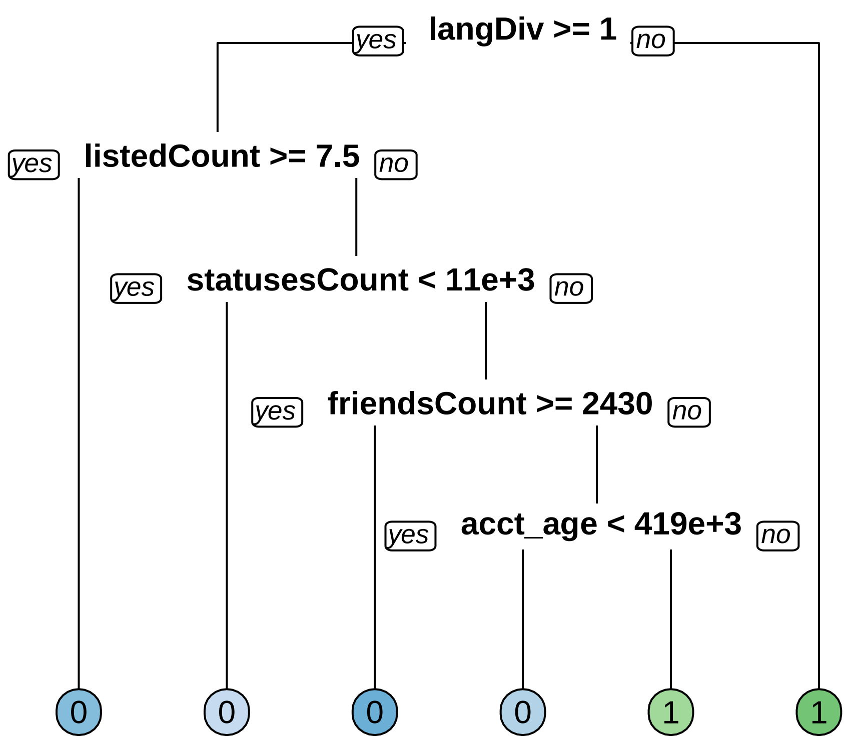

```
# Train the model (to predict 'bot')
model3 <- rpart(formula,
                     data = training_set,
                     method = "class", na.action = na.omit)

# Look at the model output
print(model3)
```

```
## n=1810 (154 observations deleted due to missingness)
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 1810 222 0 (0.87735 0.12265)
##   2) langDiv>=1.035 1585   0 0 (1.00000 0.00000) *
##   3) langDiv< 1.035 225   3 1 (0.01333 0.98667) *
```

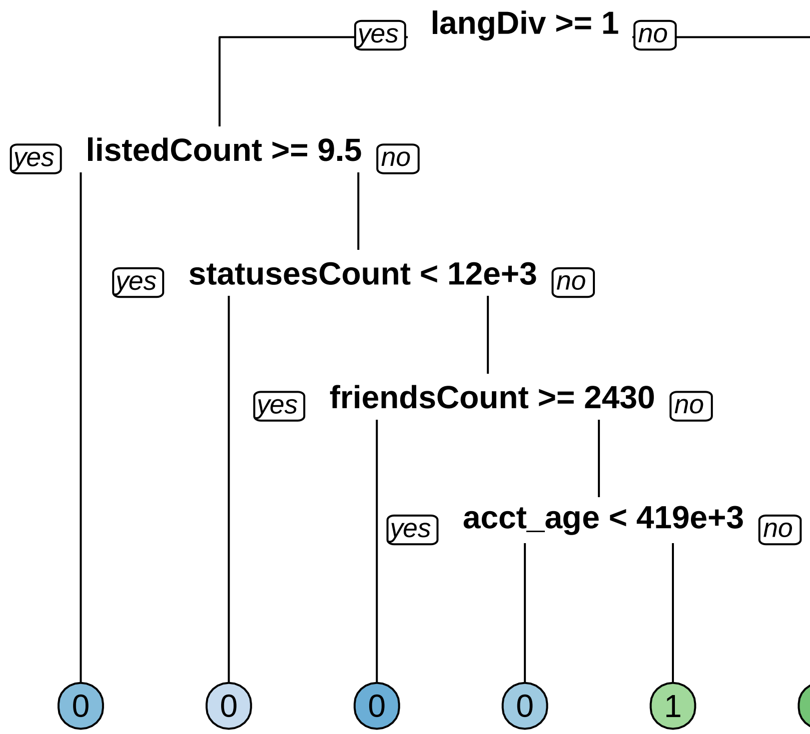
```
#Evaluate performance
#library(caret)
# Generate predicted classes using the model object
class_prediction <- predict(object = model3,
                            newdata = testing_set,
                            type = "class")

# Calculate the confusion matrix for the test set
#confusionMatrix(data = class_prediction,
 #               reference = testing_set$default)

#Minimize impurity measure - Gini Index
#Lower the Gini Index, higher the purity of the split
# Train a gini-based model
model31 <- rpart(formula,
```

```
                       data = training_set,
                       method = "class",
                       parms = list(split = 'gini'))

# Train an information-based model
model32 <- rpart(formula,
                       data = training_set,
                       method = "class",
                       parms = list(split = 'information'))

# Generate predictions on the validation set using the gini model
pred31 <- predict(object = model31,
                 newdata = testing_set,
                 type = 'class')

# Generate predictions on the validation set using the information model
pred32 <- predict(object = model32,
                 newdata = testing_set,
                 type = "class")
```

```
# Display the results

rpart.plot(x = model31, yesno = 2, type = 0, extra = 0)
```




```
rpart.plot(x = model32, yesno = 2, type = 0, extra = 0)
```




```r
######################

# Train the model (to predict 'bot')

modelClass <- rpart(formula,
                    data = training_set,
                    method = "class", na.action = na.omit)

# Look at the model output

print(modelClass)
```

```
## n=1810 (154 observations deleted due to missingness)
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 1810 222 0 (0.87735 0.12265)
##   2) langDiv>=1.035 1585   0 0 (1.00000 0.00000) *
##   3) langDiv< 1.035 225   3 1 (0.01333 0.98667) *
```

```r
#Evaluate performance
#library(caret)
# Generate predicted classes using the model object

class_prediction <- predict(object = modelClass,
                            newdata = testing_set,
                            type = "class")

# Calculate the confusion matrix for the test set
confusionMatrix(data = class_prediction,
                reference = testing_set$bot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 518  46
##          1   1  89
##
##                Accuracy : 0.928
##                  95% CI : (0.906, 0.947)
##     No Information Rate : 0.794
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.75
##  Mcnemar's Test P-Value : 1.38e-10
##
##             Sensitivity : 0.998
##             Specificity : 0.659
##          Pos Pred Value : 0.918
##          Neg Pred Value : 0.989
##              Prevalence : 0.794
##          Detection Rate : 0.792
##    Detection Prevalence : 0.862
##       Balanced Accuracy : 0.829
##
##        'Positive' Class : 0
##
```

```r
#Minimize impurity measure - Gini Index
#Lower the Gini Index, higher the purity of the split
# Train a gini-based model

model41 <- rpart(formula,
                      data = training_set,
                      method = "class",
                      parms = list(split = 'gini'))

# Train an information-based model
model42 <- rpart(formula,
                      data = training_set,
                      method = "class",
                      parms = list(split = 'information'))

# Generate predictions on the validation set using the gini model
pred41 <- predict(object = model41,
                 newdata = testing_set,
                 type = 'class')

# Generate predictions on the validation set using the information model
pred42 <- predict(object = model42,
                 newdata = testing_set,
                 type = "class")
```

```r
# Compare classification error
library(Metrics)


ce(actual = testing_set$bot,
   predicted = pred41)
```

```
## [1] 0.05657
```

```r
ce(actual = testing_set$bot,
   predicted = pred42)
```

```
## [1] 0.05352
```

```r
dt_preds <- pred42
```