Bot or Not: Building a Twitter Bot Classifier

Student ID: 0820812
Certificate in Business Intelligence: Data Mining

Business Problem

Twitter is a social networking, micro-blogging service that allow users to send and read short messages, up to 140 characters, called Tweets. Since its launch in 2006 Twitter's popularity has boomed. As of May 2015, they had over 500 million users, making it one of the world's largest virtual communities.

However, not every Twitter account is has a real person behind it. Twitter is infested with bots: computer programs that automatically send tweets. It is a mystery exactly how many bots are on Twitter, but it has been estimated that approximately 5-10% of Twitter's active accounts are bots. This is a real concern for advertisers who are paying to market their contents to people, not computer programs.

In order for Twitter to reassure its advertisers, who are its main source of revenue, it must be able to distinguish between bots and real users. With this data they can either delete bot accounts, to insure that humans are seeing the ads, or they can create a payment model where advertisers only pay for views by non-bot accounts. By analyzing account characteristics we have been able to create a model to determine whether or not an account is a bot.

Data

To perform this analysis we looked at about 6000 account, about half of which were confirmed bots, and the other half were confirmed humans. The characteristics we analyzed were numbers of followers, friends and favorites; whether or not they had default profiles, backgrounds and imagines or were verified or geo-enabled; the account age and days since they last tweeted (full list of variables in Appendix B).

Since this was a single, small data set, preprocessing was relatively simple. We converted our variable characteristics into factors and dropped the user ID.

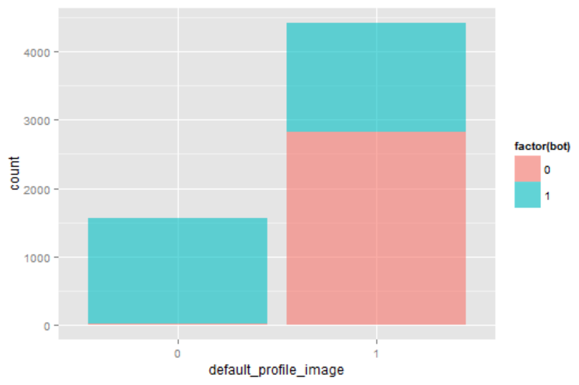
Methods

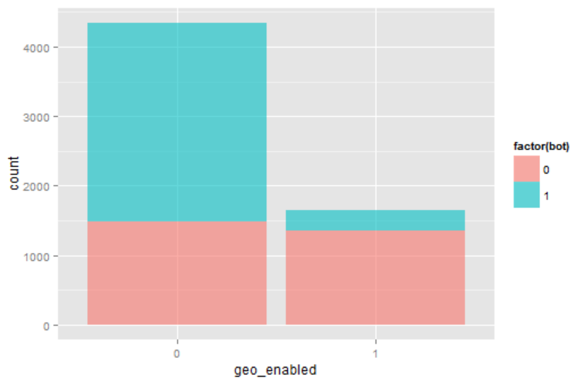
Exploratory Data Analysis

The first step was to do some exploratory data analysis. By using R's GGPairs and GGPlot packagse we were able to quickly spot which characteristics differed the most between bots and humans.

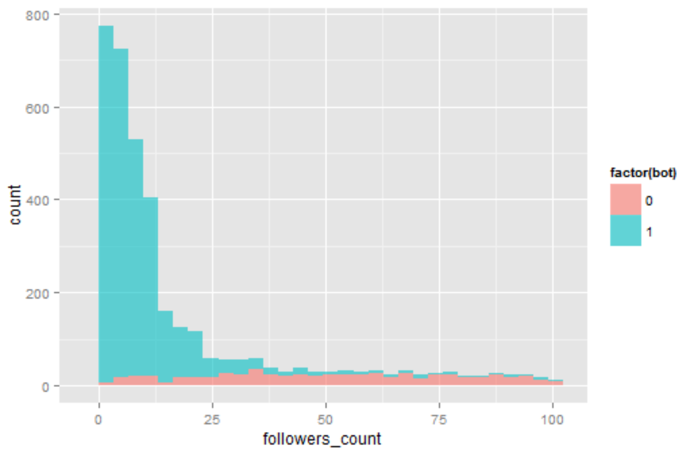
Here are some of our findings (see Appendix C for more details):

Almost no humans had a default profile picture (while almost half of bots did) and very view bots were geo-enabled.

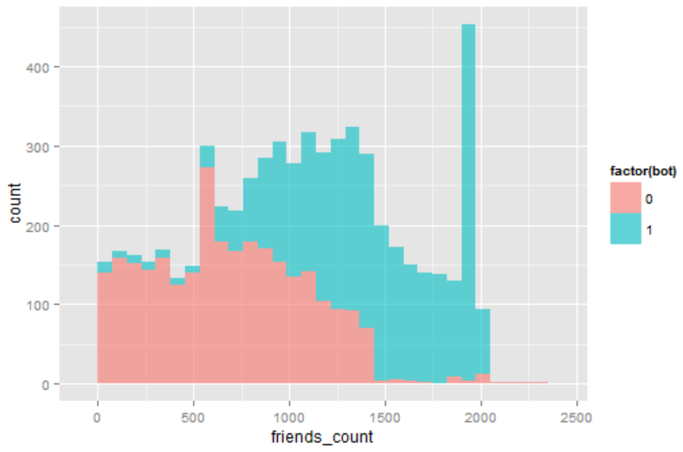




Bots were a lot more likely to have less than 25 followers than humans.



But the average amount of friends capped out around 1500 for most humans, while many bots had closer to 2000 friends.

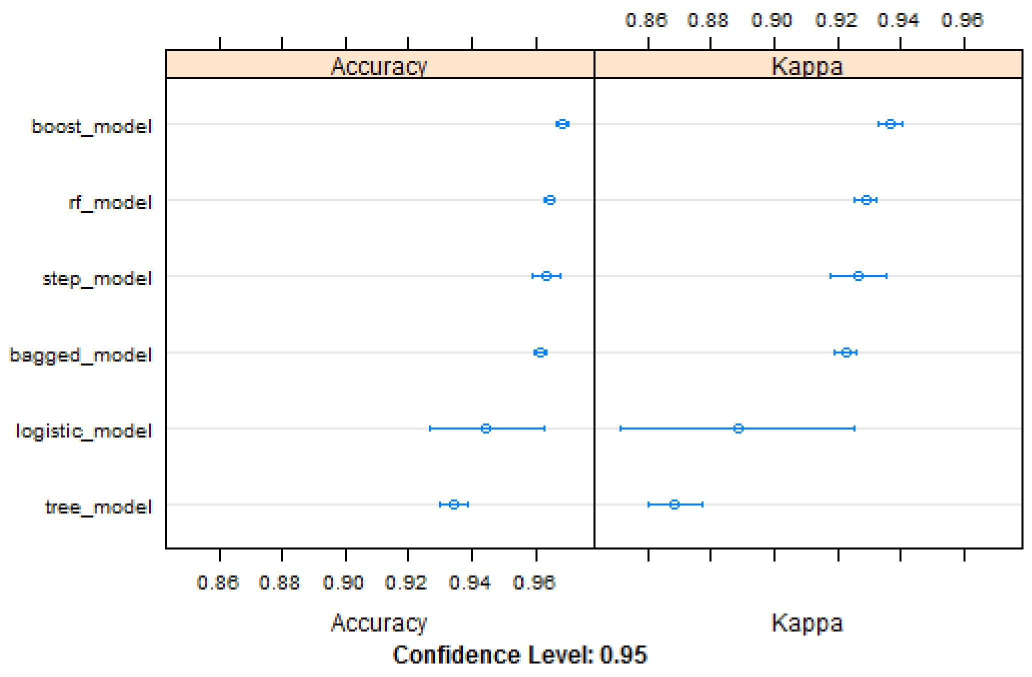


Modeling

After exploratory data analysis, the next step was using the Caret package to create and test models to find the best fit. First, we split our data into train (75%) and test (25%) datasets so that we could test the validity of our models after creating them. We tested the following types of models: Logistics Model, Step Model, Tree Model, Bagged Model, Boost Model and Random Forest Model. Tree models were more successful because they were not only more accurate, but also more intuitive and easy to understand for this type of analysis.

Results

The final model was selected based on which one had the best accuracy (see Appendix D for detail). The boost model had both the best accuracy and the best kappa. The boosting model is a type of tree model that amplifies weak predictors by putting more weight on misclassified observations at each split point (see Appendix E for model summary).



When we tested our model for data accuracy using the test data we found that 96.5% of our test accounts were categorized incorrectly. Of those accounts that were classified incorrectly, 2.2% were bots that got classified as humans and 1.3% were humans that were classified as bots.

Discussion

Based on these findings Twitter should be able to use this model relatively confidently to determine whether an account is tied to a real person or a computer program. However, if Twitter is using this information to go as far as eliminating accounts classified as botss, they might have some unhappy users. The 1.3% error of humans getting classified as bots would lead to the possible deletion of over 6 million accounts (1.3% of the current 500 million users) that belong to real people. In order to create a more accurate model Twitter could look at other variables including what type of platform messages were sent from or the trends in sending habits.

Additionally, although they do not contribute to Twitter's advertiser revenue bots are not inherently bad. Many bots are set up to automatically send out news headlines, public service messages and comedy. Even if Twitter had a model that could perfectly distinguish between bots and humans it would need to do some cost benefit analysis to see if deleting all bots is really the most sound business decision. Instead they may want to try to create a model that differentiates between spam bots and bots that are actually provide value to Twitter users. Another option would be using this data to create a payment model where their advertisers only pay for human views.

Appendix

A: Sources

Brustein, Joshua. (2014, August 12). Twitter’s Bot Census Didn’t Actually Happen. *Bloomberg Business.* Retrieved May 16, 2015 from http://www.bloomberg.com/bw/articles/2014-08-12/twitters-bot-population-remains-a-mystery-and-a-problem

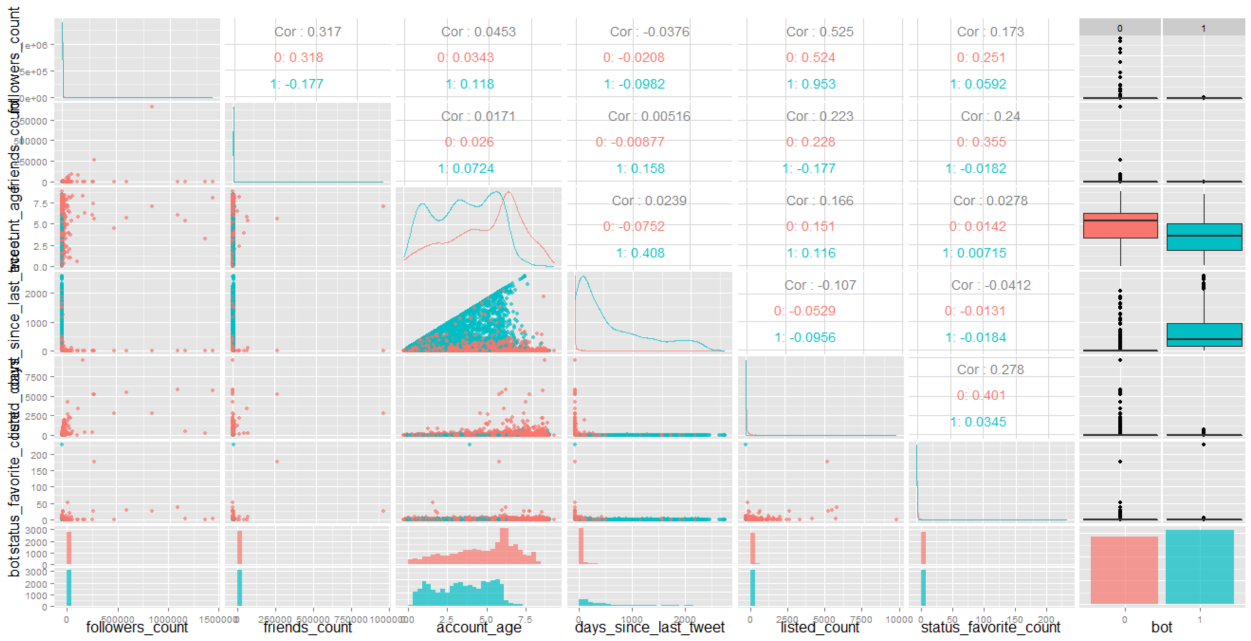
Seward, Zachary. (2014, August 12). Twitter admits that as many as 23 million of its active users are automated. *Quartz.* Retrieved May 16, 2015 from http://qz.com/248063/twitter-admits-that-as-many-as-23-million-of-its-active-users-are-actually-bots/

Twitter. (n.d.). Retrieved May 16, 2015, from http://en.wikipedia.org/wiki/Twitter

B: Variables

Variable	**Description**	**Factor?**
id	user id	N
bot	an indicator variable denoting if a user is a bot	Y
followers_count	how many followers do they have?	N
friends_count	how many people do they follow?	N
default_profile	is the profile the default?	Y
default_profile_image	is the profile photo the default?	Y
favourites_count	how many things has the user 'faved'	N
geo_enabled	have they enabled geolocation services?	Y
account_age	the age in years of the account	N
days_since_last_tweet	the number of days since the user lasted tweeted	N
listed_count	how many people have listed them?	N
profile_background_tile	is the background image the default?	Y
status_favorite_count	how many people faved their last tweet?	N
verified	is the user verified?	Y

C. Exploratory Data Analysis



D. Model Accuracy

```
Models: tree_model, bagged_model, boost_model, rf_model, step_model, logistic
_model
Number of resamples: 25

Accuracy
                 Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
tree_model     0.9136  0.9269 0.9338 0.9342  0.9418 0.9564    0
bagged_model   0.9510  0.9599 0.9618 0.9614  0.9639 0.9697    0
boost_model    0.9587  0.9652 0.9687 0.9683  0.9715 0.9756    0
rf_model       0.9571  0.9614 0.9648 0.9645  0.9675 0.9718    0
step_model     0.9240  0.9617 0.9648 0.9632  0.9693 0.9767    0
logistic_model 0.8375  0.9384 0.9671 0.9445  0.9698 0.9735    0

Kappa
                 Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
tree_model     0.8272  0.8543 0.8679 0.8686  0.8833 0.9126    0
bagged_model   0.9019  0.9198 0.9236 0.9226  0.9276 0.9393    0
boost_model    0.9172  0.9302 0.9373 0.9364  0.9429 0.9510    0
rf_model       0.9142  0.9227 0.9294 0.9288  0.9349 0.9433    0
step_model     0.8477  0.9232 0.9296 0.9263  0.9385 0.9534    0
logistic_model 0.6706  0.8762 0.9337 0.8883  0.9395 0.9471    0
```

E. Boost Model

```
Stochastic Gradient Boosting

4488 samples
  12 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 4488, 4488, 4488, 4488, 4488, 4488, ...

Resampling results across tuning parameters:

  interaction.depth  n.trees  Accuracy   Kappa      Accuracy SD  Kappa SD
  1                   50      0.9435739  0.8870312  0.005542214  0.011075218
  1                  100      0.9530039  0.9058586  0.004422359  0.008865097
  1                  150      0.9577610  0.9153639  0.004619407  0.009252482
  2                   50      0.9543499  0.9086190  0.005208973  0.010412701
  2                  100      0.9632762  0.9264087  0.004729666  0.009472176
  2                  150      0.9658579  0.9315585  0.004408098  0.008828271
  3                   50      0.9592894  0.9184510  0.004574525  0.009160945
  3                  100      0.9666738  0.9331921  0.004328819  0.008671938
  3                  150      0.9682930  0.9364268  0.004624441  0.009252496

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

F: R Code

```
library(dplyr)

library(ggplot2)

library(lubridate)

library(GGally)

library(scales)

library(caret)


summary(data)

data$bot = factor(data$bot)

data$default_profile = factor(data$default_profile)

data$default_profile_image = factor(data$default_profile_image)

data$geo_enabled = factor(data$geo_enabled)

data$profile_background_tile = factor(data$profile_background_tile)

data$verified = factor(data$verified)


ggplot(data, aes(x = followers_count, y = friends_count)) +

  geom_point(alpha = 0.10, aes(color = bot))


ggpairs(data[ , c('followers_count', 'friends_count', 'account_age',

                 'days_since_last_tweet', 'listed_count',

                 'status_favorite_count', 'bot')],

        lower = list(continuous = 'points', params = list(alpha = 0.70)),

        diag = list(continuous = 'density', params = list(alpha = 0.70)),

        upper = list(continuous = 'cor'),

        axisLabels = 'show', color = 'bot')


ggplot(data, aes(x = followers_count, fill = factor(bot))) +
  geom_histogram(alpha = 0.60)

ggplot(filter(data, followers_count < 100),
       aes(x = followers_count, fill = factor(bot))) +
  geom_histogram(alpha = 0.60)

ggplot(filter(data, friends_count < 2500),
       aes(x = friends_count, fill = factor(bot))) +
  geom_histogram(alpha = 0.60)

ggplot(data, aes(x = account_age, fill = factor(bot))) +
  geom_density(alpha = 0.60)

## xtabs gives you the crossection between 2 elements
xtabs(~bot + geo_enabled, data = data)
xtabs(~bot + default_profile_image, data = data)

ggplot(data, aes(x = default_profile_image, fill = factor(bot))) +
  geom_histogram(alpha = 0.60)

ggplot(data, aes(x = geo_enabled, fill = factor(bot))) +
  geom_histogram(alpha = 0.60)

xtabs(~bot + default_profile_image + geo_enabled, data = data)

## Modeling

set.seed(243)
data = na.omit(data)
```

```
# select the training observations
in_train = createDataPartition(y = data$bot,
                               p = 0.75, # 75% in train, 25% in test
                               list = FALSE)

train = data[in_train, ]
test = data[-in_train, ]

# drop the ids
train$id = NULL
test$id = NULL


## Logistics Model
logistic_model = train(bot ~ .,
                       data = na.omit(train),
                       method = 'glm',
                       family = binomial,
                       preProcess = c('center', 'scale'))

summary(logistic_model)
plot(varImp(logistic_model))

logistic_predictions = predict(logistic_model, newdata = test)
confusionMatrix(logistic_predictions, test$bot)


## Step Model
```

```
step_model = train(bot ~ .,
            data = na.omit(train),
            method = 'glmStepAIC',
            family = binomial,
            preProcess = c('center', 'scale'))
summary(step_model)
plot(varImp(step_model))

step_predictions = predict(step_model, newdata = test)
confusionMatrix(step_predictions, test$bot)

## Tree Model
tree_model = train(factor(bot) ~.,
            method = 'rpart',
            data = train)

print(tree_model)
print(tree_model$finalModel)
plot(varImp(tree_model))

plot(tree_model$finalModel)
text(tree_model$finalModel, use.n = TRUE, all = TRUE, cex = 0.60)

attributes(tree_model)

library(rattle)
fancyRpartPlot(tree_model$finalModel)

tree_predictions = predict(tree_model, newdata = test)
```

```
confusionMatrix(tree_predictions, test$bot)


## Larger tree
bigtree_model = train(bot ~.,
              tuneLength =5,
              method = 'rpart',
              data = train)
print(bigtree_model)
plot(bigtree_model)
print(bigtree_model$finalModel)
plot(varImp(bigtree_model))

fancyRpartPlot(bigtree_model$finalModel)

bigtree_predictions = predict(bigtree_model, newdata = test)
confusionMatrix(bigtree_predictions, test$bot)



## Bagged Model
bagged_model = train(bot ~.,
               method = 'treebag',
               data = train)

print(bagged_model)
print(bagged_model$finalModel)

bagged_predictions = predict(bagged_model, test)
confusionMatrix(bagged_predictions, test$bot)
```

```
## Boost Model
boost_model = train(bot ~.,
                  method = 'gbm',
                  data = na.omit(train),
                  verbose = FALSE)
print(boost_model)
plot(boost_model)
plot(varImp(boost_model))
summary(boost_model$finalModel)

boost_predictions = predict(boost_model, test)
confusionMatrix(boost_predictions, test$bot)

## Random Forest Model
rf_model = train(bot ~.,
               data = train,
               method = 'rf',
               prox = TRUE,
               verbose = TRUE)

print(rf_model)
summary(rf_model)
plot(rf_model)
plot(rf_model$finalModel)

# pull a tree out of the forest
getTree(rf_model$finalModel, k = 5)
```

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


## Results
results = resamples(list(tree_model = tree_model,
                    bagged_model = bagged_model,
                    boost_model = boost_model,
                    rf_model = rf_model,
                    step_model = step_model,
                    logistic_model = logistic_model))
# compare accuracy and kappa
summary(results)

# plot results
dotplot(results)
```