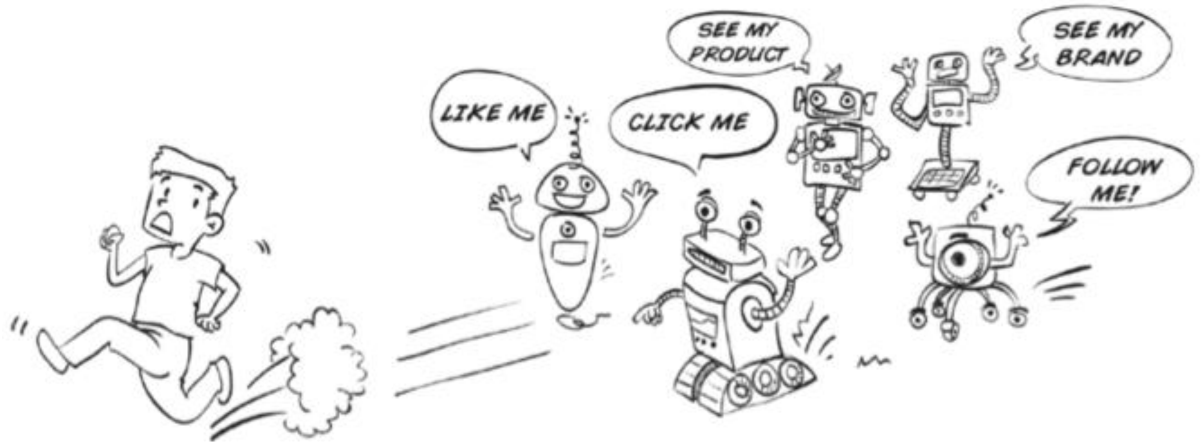
Apoorva Balasubramanian

PROJECT REPORT – TWITTER ANALYZER FOR BOT DETECTION

I. PROBLEM STATEMENT:

In social media I can observe the influence of bots increasing in recent times. What difference does it make if an account is a bot or a human?



(http://www.forbes.com/sites/lutzfinger/2015/02/17/do-evil-the-business-of-social-media-bots/2/#1fc573624e2b)

Bots can be notorious and can create some of the issues like:

1) Spamming
2) Influencing Opinion
3) Crony Advertising

The project focuses on finding if a particular account user is a human or a bot by using machine learning techniques and cloud infrastructure.

II. APPROACH:

We approach this project in two phases:

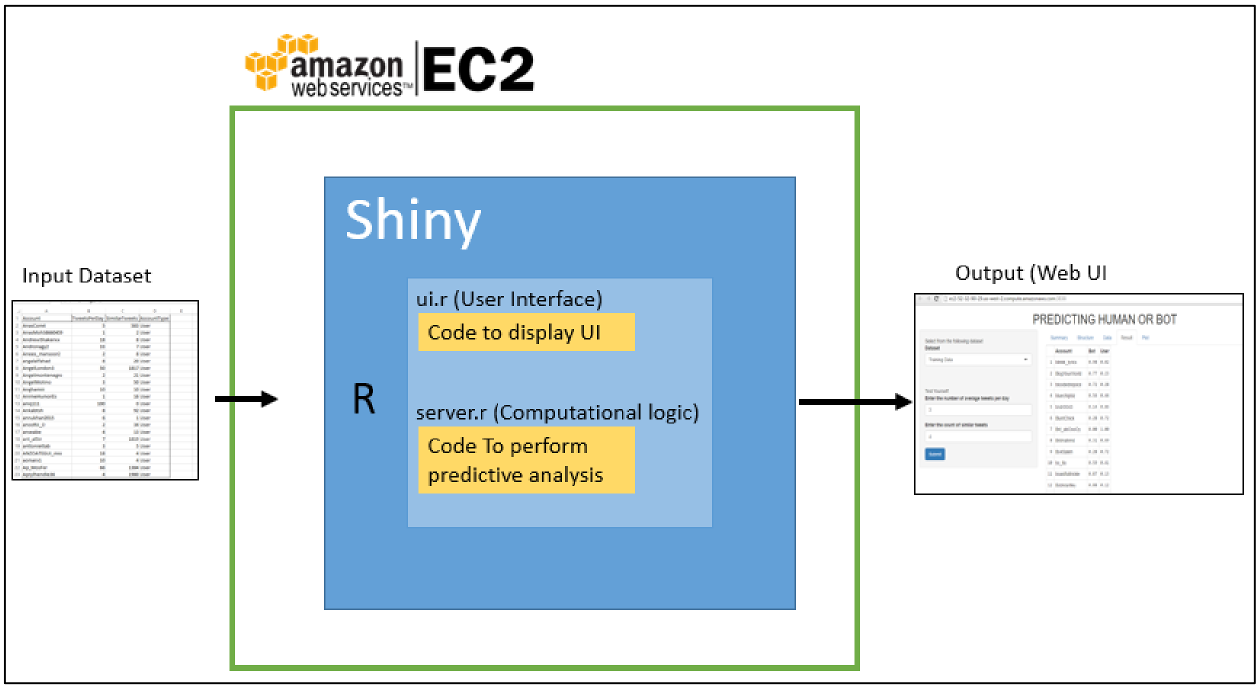
1. We use supervised learning techniques to train a given twitter dataset that has attributes like *average tweets per day* and *similar tweets count* of various bots and human accounts. I am using this trained model to predict the account type (human or bot) of any given tweet. This is implemented using R.

2. We then use an AWS EC2 instance, in which we install Shiny framework. Using this framework, we build a web application that will interact with the R code.

III. Architecture and Workflow:

1) Deploying R code in AWS:

The below block diagram displays the architecture of the project.



✓ Input dataset: Twitter dataset containing information about number of daily tweets and number of frequent tweets by each twitter account (user/ bot account).

✓ Output: Web application deployed on cloud to display the results of prediction algorithm and help user test the prediction in an interactive format.

- ✓ Programming language used: R
- ✓ Amazon AWS component used: Amazon EC2, Amazon S3.
- ✓ Web framework used: Shiny is used to develop interactive web application used analyze R scripts.

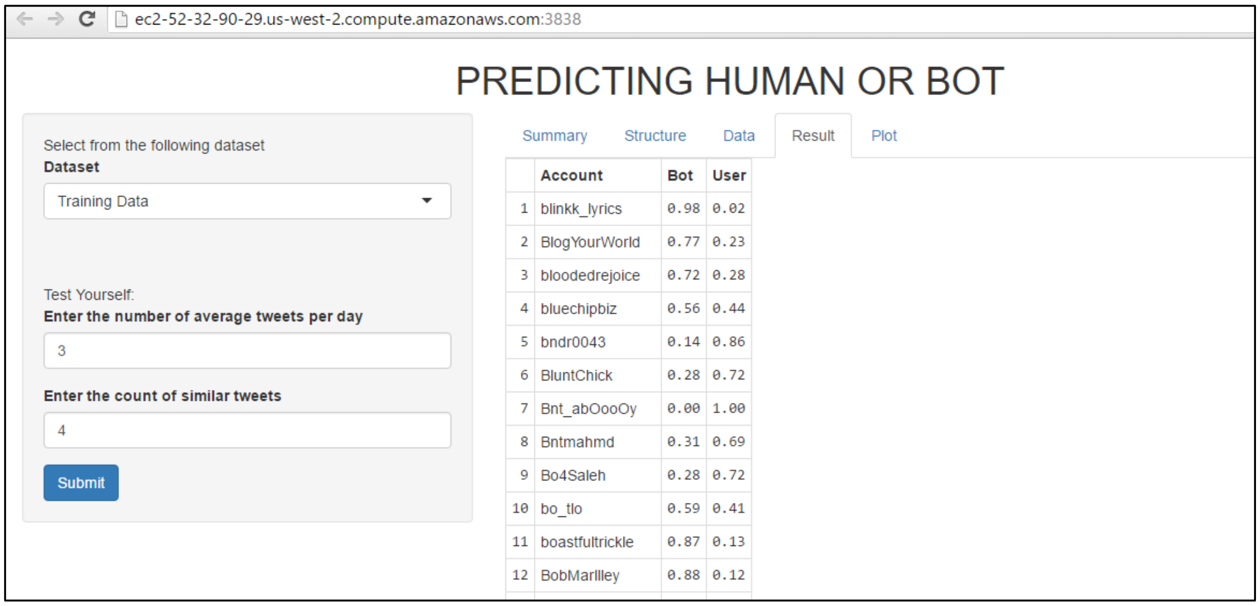
<u>ABOUT SHINY FRAMEWORK:</u>

Shiny server is a framework specifically design for R that is used to display interactive web applications Hosted Shiny server on AWS EC2 (t2.micro instance).

<u>COMPONENTS:</u>

The web application hosted on the cloud has 2 main components:

1. A USER-INTERFACE SCRIPT (ui.R):
   Controls the layout and appearance of application. In our project, the final application looks like this:



ec2-52-32-90-29.us-west-2.compute.amazonaws.com:3838

PREDICTING HUMAN OR BOT

Select from the following dataset
Dataset
Training Data

Test Yourself:
Enter the number of average tweets per day
3

Enter the count of similar tweets
4

Submit

	Account	Bot	User
1	blinkk_lyrics	0.98	0.02
2	BlogYourWorld	0.77	0.23
3	bloodedrejoice	0.72	0.28
4	bluechipbiz	0.56	0.44
5	bndr0043	0.14	0.86
6	BluntChick	0.28	0.72
7	Bnt_abOooOy	0.00	1.00
8	Bntmahmd	0.31	0.69
9	Bo4Saleh	0.28	0.72
10	bo_tlo	0.59	0.41
11	boastfultrickle	0.87	0.13
12	BobMarllley	0.88	0.12

The UI is divided into 2 main sections, the input sidebar at the left and the output main bar towards the right.

The left sidebar contains 2 sections:

a. Choosing the input dataset: We have subdivided the main dataset into 2 parts: Training dataset and Test data. The Random forest algorithm has been implemented on the training dataset and has been tested over the test dataset.

b. Testing the algorithm: This section presents the user with 2 inputs namely:
    i. Average tweets per day
    ii. Count of similar tweets

And predicts and displays the result in the form of a histogram that represents the probability of the account belonging to a user or a bot.

The right main bar contains 5 tabs that display results based on the input provided on the left sidebar section. Following are the 5 tabs:

a. Summary: Displays the R summary of the chosen input dataset.
b. Structure: Displays the table structure of the chosen input dataset.
c. Data: Displays the first 40 rows of the chosen input dataset.
d. Result: Displays the result of the Random Forest algorithm on the input dataset.
e. Plot: Displays result of the user input in the 2nd section (Testing the algorithm)

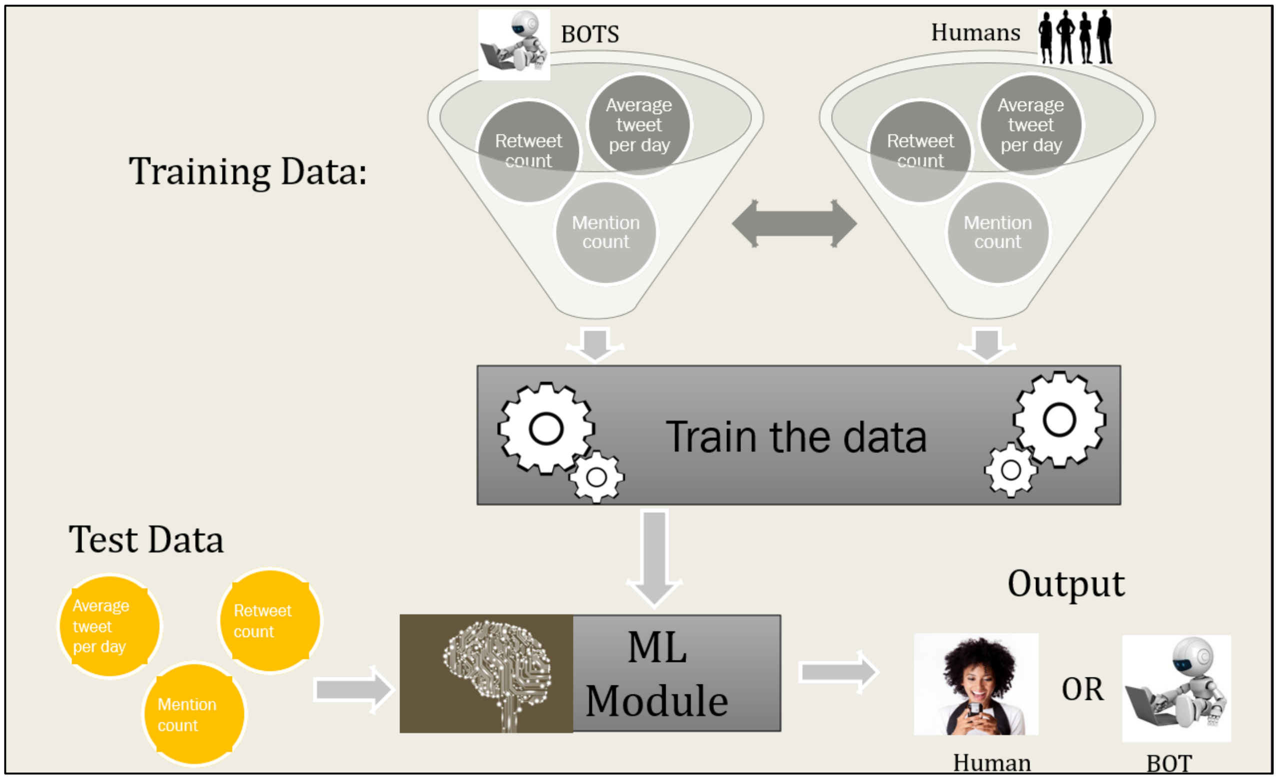
2. A SERVER SCRIPT (server.R):
   Controls the computations (instructions) that the server needs to build the application. In our project, this contains the script to perform Random Forest Machine learning algorithm on input dataset that predicts whether a twitter account belongs to a machine or a bot by analyzing the number of daily tweets and number of frequent tweets by the account.

   In order to deploy the R code, I uploaded the code over GitHub and created a local git repository in the ec2 instance and referenced the GitHub link.
   I uploaded the dataset on Amazon S3 and downloaded the dataset on the EC2 instance from the S3 location.

2) Machine Learning Module:

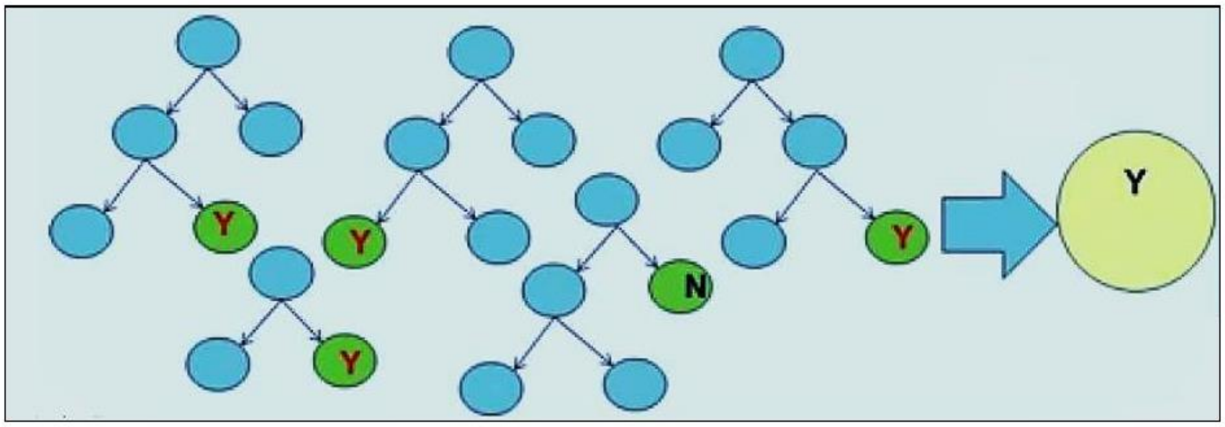
I train the data to identify the relationship between features provided and labels. As part of the training data I are using the features - Average tweet per day and similar tweet count for both humans and bots respectively. I then passed the test data to the classifiers which will return the labels (human or bot) based on the earlier learnings.

The following diagram represents the 1st part: ML MODULE:



The algorithm which I am using for predicting a human or a bot is RANDOMFOREST

Using Random Forest: A random forest is an ensemble of decision trees which will output a prediction value, in this case "account type". Each decision tree is constructed by using a random subset of the training data.



I then consolidate the outcome of each decision tree and choose the most popular outcome

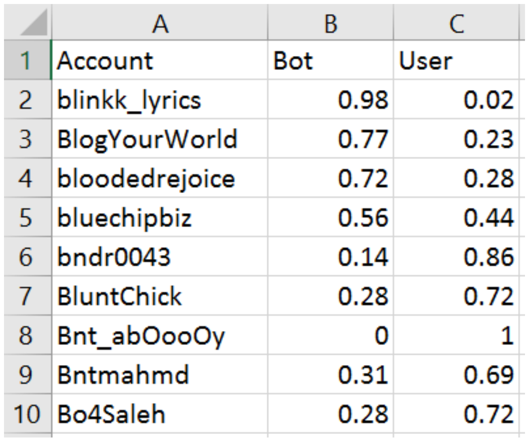
After training the data, I can then pass test data through it, in order to output a prediction.

The following is the R-Code snippet:

```
# Build the model
rf_mod <- randomForest(AccountType~TweetsPerDay+SimilarTweets,
                data = traindata,
                ntree = 100,
                na.action = na.omit)


# Predict using the test set
prediction <- predict(rf_mod, testdata, type = 'vote')
```

This Code outputs the prediction in the below format:



	A	B	C
1	Account	Bot	User
2	blinkk_lyrics	0.98	0.02
3	BlogYourWorld	0.77	0.23
4	bloodedrejoice	0.72	0.28
5	bluechipbiz	0.56	0.44
6	bndr0043	0.14	0.86
7	BluntChick	0.28	0.72
8	Bnt_abOooOy	0	1
9	Bntmahmd	0.31	0.69
10	Bo4Saleh	0.28	0.72

IV. LEARNING AND IMPROVEMENTS:

The focus of this project was more on the cloud aspects and relatively lesser on implementation of Machine Learning techniques. Hence I significantly allocated more time and energy on understanding the AWS (stack, services, deployment, frameworks).

➢ Learnings:

- Understand the AWS stack and services:
  The biggest learning in this project was hands-on exposure to AWS stack. I got to know the various services provided by Amazon. Eg. I created an EC2 instance with t2.micro configuration for this project.

- Creating a Web Application on Cloud using Shiny Framework:
  I had successfully created a R code using R Studio, however on cloud I had to build an interactive web application that could coordinate with R code. I learnt Shiny framework which helps to build easy HTML UIs and can communicate with R code that eventually trains and predicts data points.

- Installing Applications/software on cloud instance:
  I had to install Shiny framework on the EC2 instance and in that process of doing that I learnt the steps to install any software on cloud instance.
  I also got a chance to use Gitbash App that helped me recreate an Ubuntu CMD on my Windows OS.

- The importance of GitHub:
  I learnt yet another aspect of GitHub where-in I can do auto sync up of code from GitHub to a prescribed cloud instance.

Improvements:

- I have only used 2 features (Average Tweets and Similar Tweets) based on which I train and predict the data. Though prediction were surprisingly higher than initial estimate, I think I could get even better predictions if I use more features, like Twitter Account's Network, Retweet Counts, number of followers etc.

- The UI can be improved to include user input instead of only showing the test data results.

- In case of larger volumes (if training data were in millions) I may have to install Hadoop other big data tools to analyze and work with dataset.

- There was also scope to include other services of AWS like Cloud watch, for monitoring our instance, use multiple instances and have Elastic Load balancer to handle high traffic.

V. References:

[1] https://aws.amazon.com/codedeploy/
[2] http://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html
[3]https://www.rstudio.com/products/shiny/shiny-server/
[4] http://www.r-bloggers.com/installing-rstudio-shiny-server-on-aws/