Classifiers To Identify Twitter Accounts As Bots OR Not Bots

Ashish Bansal
COMPUTER SCIENCE DEPARTMENT, NYU
ab6995@nyu.edu

John Martinez
COMPUTER SCIENCE DEPARTMENT, NYU
jzm218@nyu.edu

Zhongheng Li
COMPUTER SCIENCE DEPARTMENT, NYU
zl1761@nyu.edu

Abstract—In this project we explored how to classify twitter accounts as bots or not-bots accurately. To do this, we harvested several datasets and used them to train our model to distinguish bots from real users. The machine learning algorithms known as Naive bayes, logistic regression, and decision trees were tested. We will present here the results from the most successful of these and upon deciding which algorithm performs the best.

Keywords—Machine Learning, Naive bayes, Logistic regression, Decision trees, Random Forest

I. INTRODUCTION

In this project we aimed to recognize the behavior of account activities by bots and will compare it with behavior of tweets by humans by which we will train our model using certain algorithms. The algorithms we will be looking upon are Naive bayes, logistic regression, decision trees. Through these algorithms we will explore how accurately could the identification of twitter accounts as bots and not-bots can be done . In this we will harvest certain datasets, with the help of this training dataset, different attributes will be taken on which identification of bots and not bots will be done .In end, algorithms were attempted on different available datasets and identification of bots and not-bots would be done.There would be presentation of the results from the most successful algorithm we used and will distinguish which algorithm perform better.

II. MOTIVATION

On March 27th, 2017, a published research conducted by the University of Southern California and Indiana University, had discovered that 48 millions of Twitter accounts are non human bots. Which is 15% of Twitter's total active user population. Many of these social bots are being used for dissemination of news and publications. But not all bots are disseminating contents that are harmless. Malicious bots are emulating human behaviors to manufacture fake grassroots political support and promote terrorist propagandas and recruitments. On November 1st, 2016, the Atlantic had an article on how Twitter bots had shaped the 2016 US Presidential election. As we are in the age of information, where social media has great impacts on shaping the social structure and humanity, identifying the malicious socialbots is critical. It is critical for identifying the sources of malicious fake information, and distinguish the malicious ones from the useful bots that are providing helpful functions like alerting disasters and providing useful customer services. In order to achieve this goal, the first step is to distinguish all bots from real users.

III. RELATED WORK

1. Douglas Guilbeault and Samuel Woolley. "How Twitter Bots Are Shaping the Election." *The Atlantic*. Atlantic Media Company, 01 Nov. 2016. Web. 12 Mar. 2017.
2. ArXiv, Emerging Technology from the. "How to Spot a Social Bot on Twitter." *MIT Technology Review*. MIT Technology Review, 19 Sept. 2014. Web. 12 Mar. 2017
3. Harris, Derrick. "Meet the Algorithm That Can Spot and Kill Twitterbots before They Ever Start Spamming." *Derrick Harris*. Gigaom, 23 Aug. 2013. Web. 12 Mar. 2017.
4. Varol, Onur, Emilio Ferrara, Clayton A. Davis, Filipo Menczer, and Alessandro Flammini. "Online Human-Bot Interactions: Detection, Estimation, and Characterization." (2017): n. pag. 9 Mar. 2017. Web. 12 Mar. 2017.

IV. DATA

The data set used is an extract from Twitter API we used the module to query two endpoints: GET user/lookup.json and GET user/friends/list.json. originally intended to search as tweets by a certain user and tweets of friends of certain user account . The original data set has 130 attributes per user id . This dataset is a mixed set in which we can find many categorical and numerical variables with a lot of missing values. From all the attributes these were the ones that we considered:

1	id	
2	Id_str	
3	Screen_name	
4	Location	
5	Description	
6	Url	
7	Followers_count	
8	Friends_count	
9	Listed_count	
10	Created_at	
11	Favourites_count	
12	Verified	
13	Statuses_count	
14	Lang	
15	Status	
16	Default_profile	
17	Default_profile_image	
18	Has_extended_profile	
19	name	
20	Bot	

Many of the variables have many levels, including the status and entities variable .

To obtain a better result for our prediction model, we also did feature engineering by creating the following useful features.

- **screen_name_len** -   The length of the screen_name.
- **bot_is_substr -** The boolean indicator to show if the characters ‘bot’ is in the screen_name.
- **bot_in_de**s - The boolean indicator to show if the characters ‘bot’ is in the description.
- **age** - The calculated age of account in years from created_at
- **desc_pol** - The polarity in description by TextBlob
- **desc_subj** - The subjectivity in description by TextBlob
- **male_prob** - The gender analysis by Genderize
- **fem_prob** - The gender analysis by Genderize

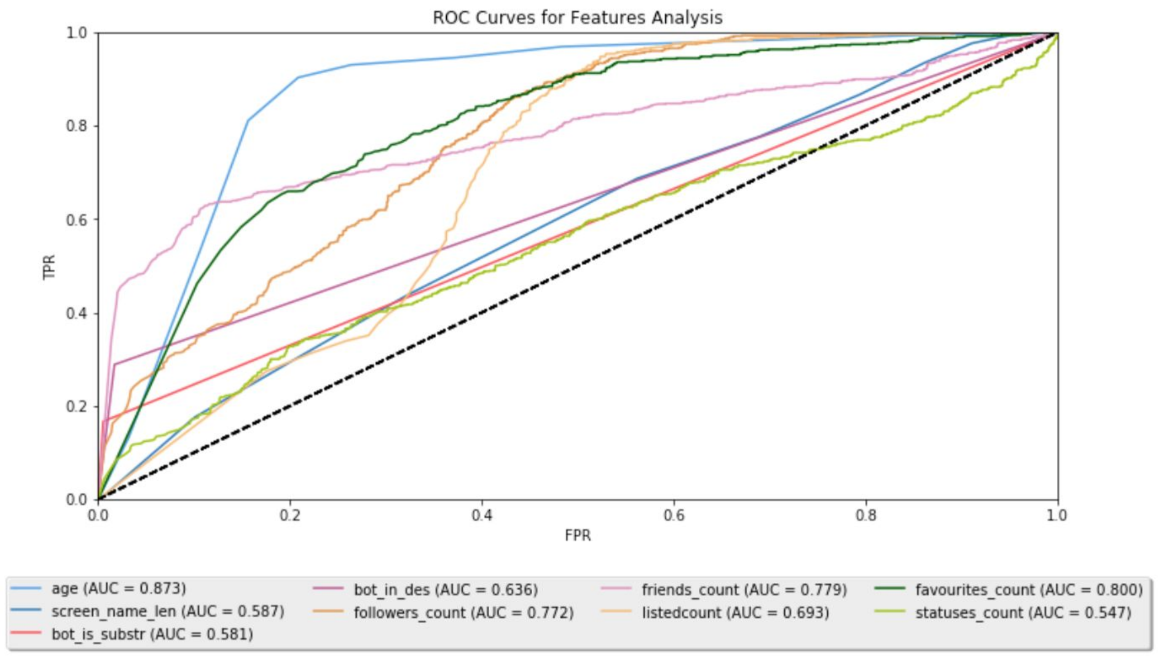
And turnout, from our feature importance with entropy in decision tree model, the created feature **age** has the highest importance among all features.

V. ALGORITHM(S) USED

A. Naive bayes :

First, MultinomialNB model is used to build a model to identify the twitter accounts as ‘bots’ .
Bag-of-words representation of the ‘Description’ is used for Naïve Bayes model. We have split the given data into two data sets which is training data (80%) and test data(20)%. For evaluating the performance of the classifier CountVectorizer and TfidfTransformer are used for training and test data. From this we get frequency of words in the description column which we get through

a) **Tokenizing** strings and giving an integer id for each possible token.
b) **Counting** the occurrences of tokens in each document and weighting with diminishing importance tokens that occur in the majority of samples / documents.



In this we Tokenize with importing CountVectorizer which will give us the occurrences of the data in description after which we will get frequencies of these words by importing TfidfTransformer from sklearn.We will call fit_transform on the transformers with training data. After that we can train a classifier to try to predict the category of a post. Let's start with a **Naive bayes** classifier, which provides a nice baseline for this task. There are several variants of this classifier; the one most suitable for word counts is the multinomial variant.
After training our model we will evaluate the performance on the test data to find Precision , Recall, F-1 score and Accuracy score and plot ROC_Curve.

B. Decision trees
A Decision tree classifier was also used to identify twitter bots. To setup the data for training, the original frame was first reindexed into a random permutation. A training set and test set were then created respectively from the first 80% and last 20% of accounts from the reindexed dataframe. After this, four new features were created from the dataframe: the age of the account, which was calculated from the information created_at, whether or not 'bot' was a substring of the screen name, whether it was a substring of the description, and the length of the screen name. For the age feature, an issue was encountered where not all dates in created_at were in the same format, and for many of the accounts, the month and day were ambiguous, so they were not included in the calculation. Thus, the age was a rounded number of years. Among the rest of the features, only the numerical and binary ones were used in the classifier. The binary 'default' features were ultimately excluded from the tree, as they had a feature_importance_ of essentially 0 when graphed. The classifier was then used to predict the values, with various branching and sample-split constraints, for the test set as well as 5-fold cross validated, and its accuracy, precision, recall, F-1, and ROC were calculated by calling upon the sklearn.metrics methods.

C. Logistic Regression
Since our problem is a classification problem to classify a twitter account is bot or not bot**,** logistic regression is an ideal model to use for our predictive model.

We use the normally randomized and reindexed data for this model. We used 80% of the randomized data as training set, and 20% of the data as test set. First of all, we flatten the target variable on both the training and test set for logistic regression computation with scikit-learn. Like what we did in decision tree, we started with the features that are only the numerical and binary features. Which are: **'age','followers_count','friends_count','listedcount', 'verified', 'favourites_count','statuses_count','default_profile','default_profile_image','screen_name_len','bot_is_substr' and 'bot_in_des'.** Once the features are prepared, we trained our logistic regression model with LogisticRegression function from scikit-learn. As we have have our initial model prepared, we examine the coefficients to evaluate the weights for each features above to see if there are any features that we can reduce.

	0	1
0	age	[-2.16156762954e-05]
1	followers_count	[-3.64084355884e-07]
2	friends_count	[6.28691485365e-06]
3	listedcount	[-0.000794617092691]
4	verified	[1.8178700417e-06]
5	favourites_count	[-0.00012767478018]
6	statuses_count	[2.45707146437e-05]
7	default_profile	[1.8178700417e-06]
8	default_profile_image	[1.8178700417e-06]
9	screen_name_len	[2.75445261514e-05]
10	bot_is_substr	[1.41731260178e-06]
11	bot_in_des	[2.5363467747e-06]

We reduce the features that have negative coefficients and high frustration. Additionally we manually compute the accuracies by remove the remaining features one at a time to compute the highest accuracy. Eventually, we picked 'followers_count','friends_count','favourites_count' and 'statuses_count' as the finalists since they gave us the highest accuracy with updated Training Accuracy jumped to **0.735722284434**, and updated Test Accuracy jumped **0.764573991031** compared to 0.733482642777 and 0.72869955157 previously.

Next, we evaluated our model using a validation set. We used the train_test_split function from scikit-learn to split and randomize the already randomized data set that we used earlier. We train our model again and name this newer version of model to be model2. We get the predicted values and class probabilities by running predict and predict_proba functions on the test data. And then we ran the classification_report function to get our precision, recall and f1-score for our model2.

```
Classifictiaon Report:
             precision    recall  f1-score   support

          0       0.74      0.75      0.74       222
          1       0.75      0.73      0.74       225

avg / total       0.74      0.74      0.74       447
```

Finally, we used Cross-Validation to evaluate our model again to check the accuracy by running cross_val_score function with 10 folds. And we see that the mean of resulting accuracies are around 74%. Which is pretty aligned with what we done at the first place.

Accuracy Score: 0.747627091377
Precision Score: 0.716093277091
Recall Score: 0.77358490566
F1 Score: 0.743140539575
ROC_AUC Score: 0.839856021166

D. Neural Network

We have also quickly tried Neural Network as a comparison to other models with the most moderate setting. However, it gave us the lowest accuracy score.

E. Random Forest

After using the classifiers such as Decision trees and logistic regression to calculate the accuracy on the test data, we proceeded to other classifier that is Random Forest to increase our model's accuracy to identify twitter accounts as bots. **Random forest** is an ensemble learning method for classification , regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. In this classifier we choose these attributes "age, followers_count, friends_count, listedcount, verified, favourites_count, statuses_count, default_profile, default_profile_image ,screen_name_len, bot_is_substr ,bot_in_des" and we used random forest classifier to fit these attributes into our model where we calculated the "entropy" by the classifier of the above attributes .We used three parameters in our classifier which are :

1.) **n_estimator** (which is no. of trees in the forest) to be 200 as it diversifies our tree and accepts all scope of the classifier .
2.) **oob_score** (Whether to use out-of-bag samples to estimate the generalization accuracy) and we had kept it as default which is boolean "False" as we are not using out of bag samples.
3.) **criterion:** this parameter is tree specific and we have kept that as "entropy" which means it is using the information gain of the attributes for the quality of splits.

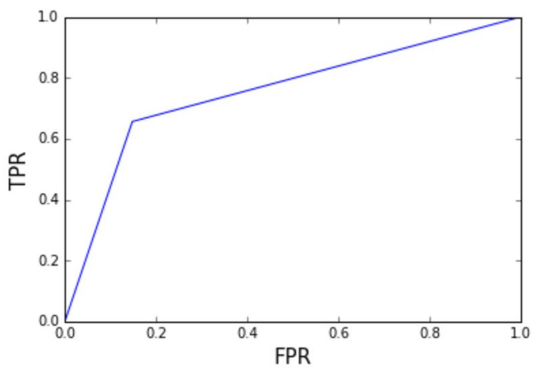
In this Random forests differ in only one way from this general scheme of decision trees : they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the trees, causing them to become correlated. Such as in this the importance of 'age' attribute is the strongest and others are very weak such as "screen_name_len, bot_is_substr ,bot_in_des" but together they become correlated and gives us the better accuracy for our model. By this classifier we get the highest accuracy among all that is **0.97213**.

VI. RESULT

A. Naive bayes : Result from Naive bayes classifier for different scores are

Accuracy Score: 0.769574944072
Precision Score: 0.769574944072
Recall Score: 0.769574944072
F1 Score: 0.769574944072
AUC Score: 0.77027027027027017

ROC Curve for the baseline classifier is below:



The accuracy from naive bayes is 77% . Our future algorithms show, we can greatly improve performance from Naïve Bayes.

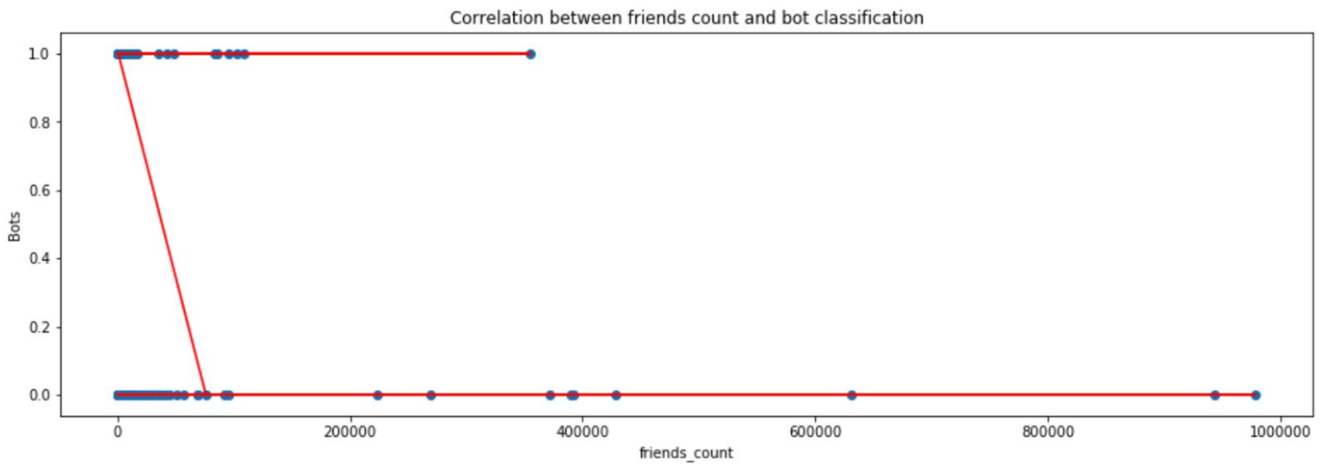
B. Decision Trees:
I. Results from 5-fold cross validation on the test set
Average accuracy: 0.864983576759
Average Precision: 0.863809581514
Average Recall: 0.835888501742
Average F-1: 0.850465115982
Average ROC: 0.866631452789
Accuracy of predictions on the test set: 0.883408071749

II. Results of evaluations on different branch and split constraints for predictions:
min_samples_split_values were in [20,25,30...65]
min_samples_leaf_values were in [20,21,22...29]
Maximum Accuracy: 0.887892376682
Minimum Accuracy: 0.881165919283
Average Accuracy: 0.884125560538

C. Logistic Regression:
I. Results from 10-fold cross validation on the test set

Accuracy: 0.747627091377
Precision: 0.716093277091
Recall: 0.77358490566
F1: 0.743140539575
ROC_AUC: 0.839856021166

II. Results from evaluation with validation set on selectively reduced features
Accuracy Score: 0.742729306488
ROC AUC Score: 0.865325325325



III. Results on visualizing the correlation between ‘friends_count’, the feature that has the highest score of coefficient, with our target variable ‘bot’.

When we plot the friends_count(x-axis) and bot (y-axis) into our graph, we clearly can see that real human user can have more friends_count than bots.

D. Neural Network
Accuracy: 0.71300448430493268
Precision: 0.527233379635
Recall: 0.912884097035
F1: 0.666424477771
ROC_AUC: 0.689800247988

E. Random Forest
Result by random forest is
Accuracy : **0.92152466367713**
Precision Score: 0.906103286385
Recall Score: 0.932367149758
F1 Score: 0.919047619048

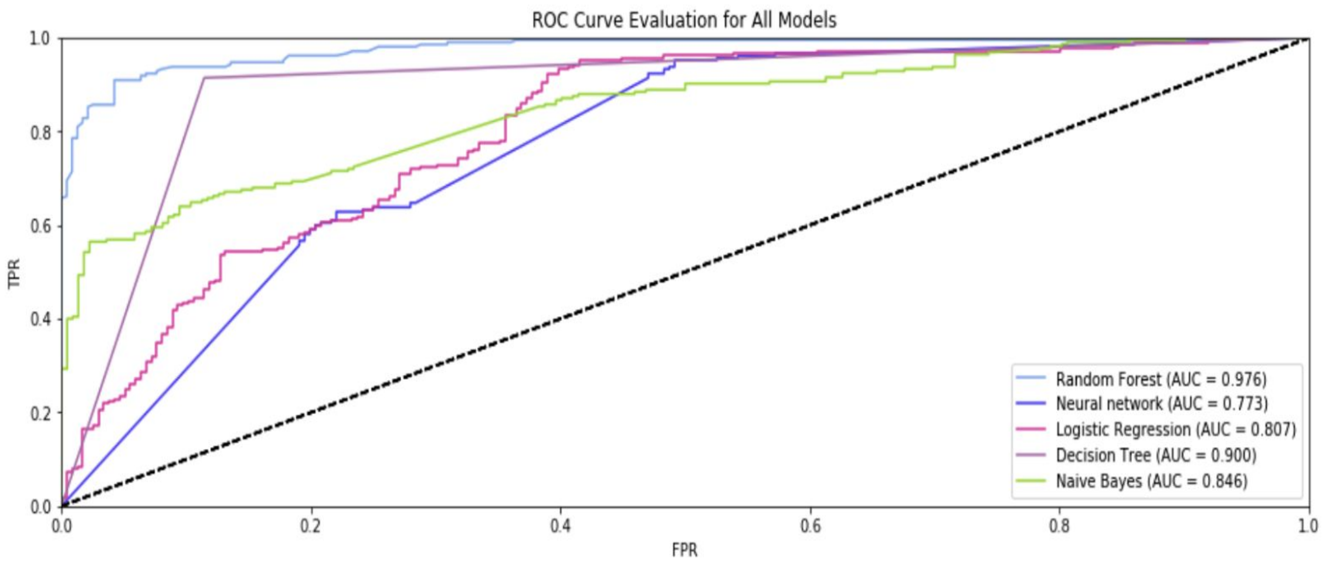
Final decision on our classification model:
According the ROC curve analysis and the results from each model, eventually we picked Random Forest Classification for our Kaggle competition since it gave us the highest accuracy.

In the initial attempt we already achieved the accuracy score **0.97213,** and was in the top 10 spot in the competition by the time we submitted. Later on, we made some modification on the parameters to train our random forest classifier, and with different combination of features. Even the highest accuracy score we had was **0.9327354260089686,** and we did see the changes in our predicts, however when we submitted our new result to Kaggle, we still received the same accuracy, **0.97213.**

As for the final result in the most recent stage of this report our highest accuracy received from our Kaggle’s competition is: **Accuracy** : **0.97213**

VII. Code
Github Link:
- Consolidated Team iPython NoteBook prepared by Zhongheng Li
- Multinomial Naive Bayes iPython Notebook prepared by Ashish Bansal
- Decision Trees iPython Notebook prepared by John Martinez
- Random Forest iPython Notebook prepared by Ashish Bansal
- Sentiment Analysis and Gender Detection iPython Notebook prepared by John Martinez

VIII. VIDEO LINK
Youtube Link:
- Classifiers To Identify Twitter Accounts As Bots OR Not Bots - NYU Tandon - CS6923 Machine Learning



IX. Evaluation

Cleaning up the data, extracting new features, and using the sklearn library were tasks that went mostly well. Cleaning up the data was simply a matter of utilizing the pandas and numpy libraries. New features, like bot_in_des and bot_in_substr were obtained by looping through the data's other features and using the 'in' operator, and using these features in the data and training the various classifiers in it took only a few function calls in sklearn. Although the age was shown by the decision tree classifier to have the highest feature importance, extracting the feature went only partially well. This is because the inconsistencies in the format of the 'created_at' feature rendered only the years of the age unambiguous, and the months and days had to be dismissed, even though a more precise measure of the age may have been helpful. We also attempted to increase the accuracy by measuring the polarity and subjectivity of the descriptions, but these measures did not affect the accuracy in any way. We also noticed that bots tend to have screen_names very similar to their names, so we added the edit distance between the two as a feature, but this also made no difference. Finally we attempted to increase the accuracy by detecting the gender from the names of each account, but adding the male and female probability as features actually decreased the accuracy. If we had more time, we would have placed more effort in using the 'status' feature. The format of this feature was far too inconsistent and difficult to parse within the time given, even though it may well have had bot-revealing information and text. In addition, with more time, we could have also experimented with different libraries for machine learning, like TensorFlow, and for sentiment analysis and gender classification, as using more robust libraries may have increased accuracy as well.

X. Conclusion

As the results show, the Random Forest has been demonstrably the best classifier for detecting and classifying twitter bots. This makes sense, since as aforementioned, Random Forest is an extension of Decision Trees, which was shown by experimentation to be the most accurate classifier before Random Forest was used. The fact that the Random Forest classifier achieved an accuracy as high as 97.2% shows that there are indeed measurable differences between human users and bot 'users'. With the prospect of more data, like the 'status' feature, and better libraries, we see clear progress toward the eventual true goal of experiments such as these, which is detecting and classifying malicious Twitter bots.