CS S-109A
Protopapas, Rader

MILESTONE #3: PRELIMINARY FINDINGS AND UPDATED PLAN

July 27, 2018

Prepared by Project Team #6
Eumar Assis
Andrew Caide
Mark Carlebach
Jiang Yusheng

EXPLORATORY DATA ANALYSIS

We have collected tweet and tweet author information from two sources:

- We have **X** users and **Y** tweets collected directly from Twitter using the tweepy API. This data is for X verified users and Y unverified users. (We treat verified and unverified users as the same in our K=2 classification models below. If time permits, we have the option to consider K=3 classes subsequently.)
- We have **X** users and **Y** tweets for known bots provided by this website: https://www.nbcnews.com/tech/social-media/now-available-more-200-000-deleted-russian-troll-tweets-n844731.

We have significantly more bot data than non-bot data. This unequal ratio will prove relevant when evaluating whether our models score well relative to the most simplistic baseline accuracy measure (e.g., all tweets come from bots). We will be aware of this and intend to get more non-bot data before our final modeling is complete.

The following tables represent the attributes and relationships within this data. The data is stored as two .json files at the moment, though we are considering moving the data to a mySQL data with time permitting before the end of the project.

Status/Tweet Data	**User/Author Data**
● User_id	● Id
● User_screen_name **(key to user)**	● Name
● Created_at	● Screen_name
● Status.created_at.isoformat()	● Location
● Id	● Url

● Text ● Source ● Truncated ● Retweet count ● Favorite count ● Language ● Is_tweet	● Description ● Verified ● Followers_count ● Listed_count ● Favourites_count ● Statuses_count ● Created_at ● Lang

Data Cleaning and Reconciliation

In our preliminary review of our data, we encountered the following issues which we have resolved as described:

- Tweepy's API provides "tweets" that are both tweets and retweets. Accordingly, Tweepy's documentation indicates Twitter provides a retweet flag for tweet objects that are actually retweets. **We noticed tweepy was not providing this retweet flag** in our test data. We investigated and concluded this was due to our using the standard, free API (in contrast to a paid for service level). We "corrected" for this missing data by setting the retweet flag ourselves when the tweet text includes "RT", which seems to be inserted into the text field by Twitter for all retweets.

- **Other fields we expected to receive from Tweepy are also missing**, such as user favorites count and followers count. This could be due to coding errors, but we think it is probably related to the free service level from Twitter. We aim to resolve or confirm the source of the problem. If the latter, our options with which to model will be more limited.

- As described above, we are getting data from two sources: Twitter directly and the NBC news website. Our intention was to use the userid in the tweet data as our foreign key to data in the user table. When we analyzed data provided for bots by NBC, we noticed **the userid field was largely empty**. We explored the NBC data and inferred they used an alternative field ('screen name') to link its tweets to its user table. We went back and adopted this same strategy for linking tweets to users for data from Twitter as well.

- Based on research into similar projects, we expect the timing of tweets throughout the day to play a role in classification. Our attempts to explore this relationship were originally limited by our discovering that moving data in and out of .json files with certain techniques resulted in the **datetime stamp becoming corrupt for a majority of our tweets**. We tried many alternative methods of reading/writing .json files until we were able to retain the accuracy of the datetime stamp (writing to .json via *df.to_json(orient='records')* and reading via *pd.read_json()*).

<u>NLP Feature Engineering</u>

We have incorporated natural language processing (NLP) features into our classification modeling. We achieved this by using an API from Microsoft's AI Text Analytics service on Azure's cloud computing platform. The API allowed us to evaluate the text component of each tweet to create two, new NLP features:

- *Sentiment Score:* a real number between 0 and 1 that corresponds to the negative to positive sentiment of the tweet.
- *Topic Count:* an integer between 0 and infinity (where 10 appears to be close to an actual upper bound) of the number of topics mentioned in the tweet.

We expect these features will be prominent in our analysis--especially since a) Tweepy does not deliver every user attribute via its free, standard service level and b) we lack time to analyze fully deeper aspects of each tweet such as embedded URLs.

<u>Data Exploration and Analysis</u>

We are using Python charting abilities to explore the data from many perspectives. Below are descriptions and displays that represent our main analyses so far:

<u>Figure 1. Correlation Matrix</u>

We felt looking at correlations among the features was standard practice and might add insight. The chart below is a very early version of this with a very limited number of attributes and does not give us any new perspective. Once we have the the better features described below in our Final Project Statement, we will repeat this analysis and add a scatter matrix to see if we can learn more from it.

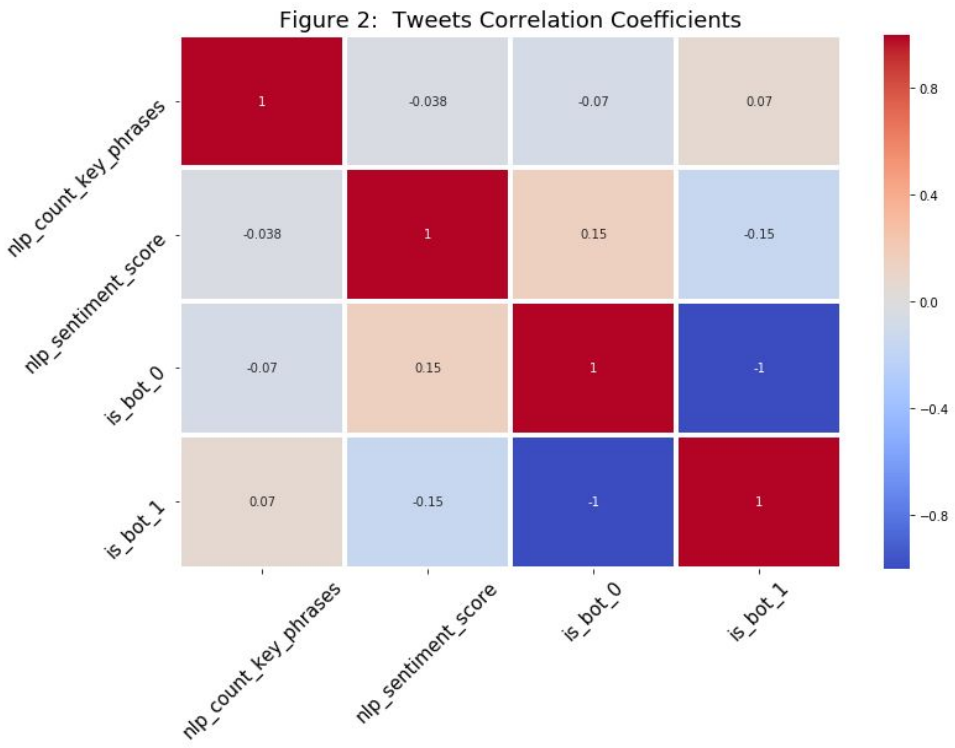
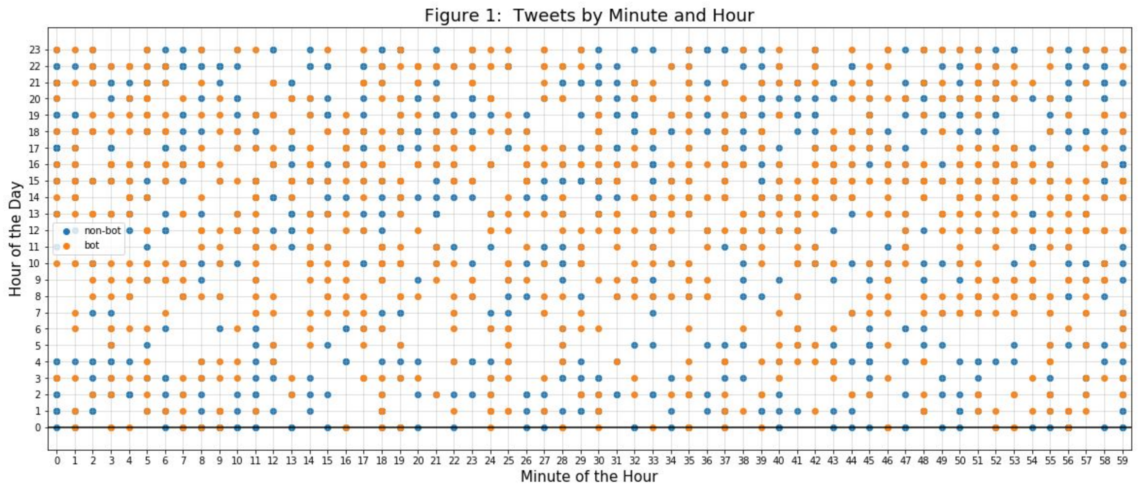


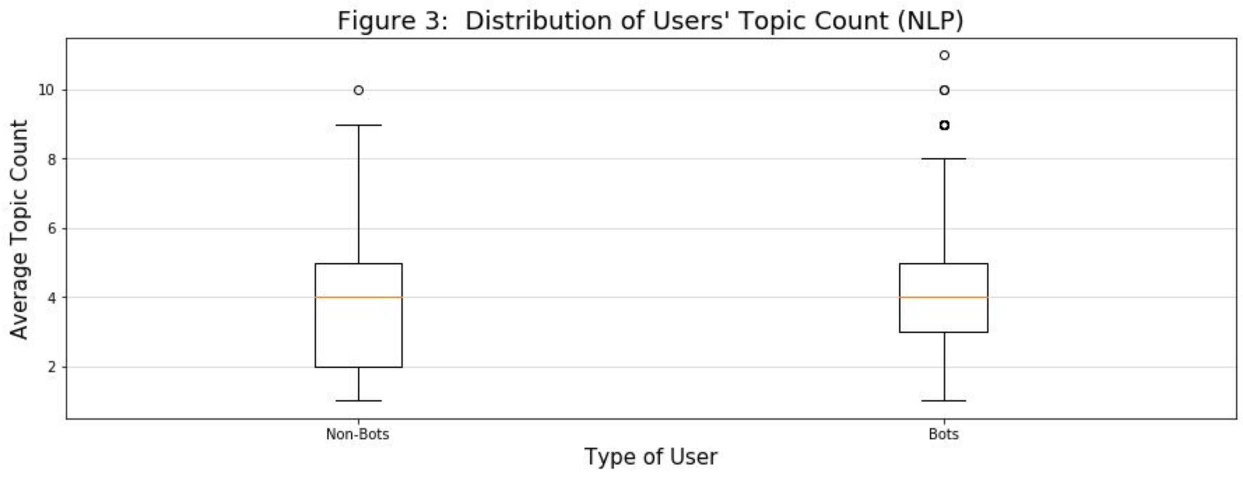
Figure 2. Time of Tweet Analysis

Our reading of others' research suggests that patterns of tweeting throughout the day are indicative of bot vs. non-bot behavior. The following chart was one attempt to gain intuition of potentially different behavior between the classes. The chart shows tweets for each of class of user in separate colors and plots the occurrence of a tweet by minute of the hour vs. hour of the day. We do not discern any useful pattern with this type of analysis yet.



<u>Figure 3. NLP Topic Count Analysis</u>

In this diagram, we look at the distribution by class of each user's average tweet topic count (i.e., for each user, we calculate the average topic count across all of that user's tweets and then plot the distribution). It appears bots tweet within a slightly narrower IQR of this measure.



<u>Figure 4. NLP Sentiment Analysis</u>

In this diagram, we look at the distribution by class of each user's average tweet sentiment score (i.e., for each user, we calculate the average sentiment score across all of that user's tweets and then plot the distribution). It appears bots tweet within a slightly narrower IQR of this measure, with a median and many outliers that are much more negative.

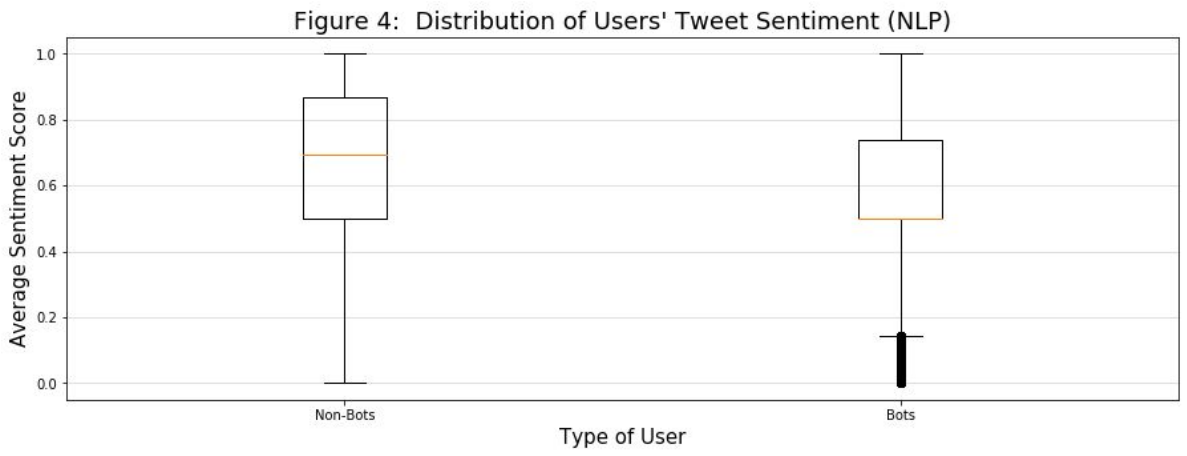


Figure 5. NLP Topic and Sentiment Analysis

As in **Figure 2**, this chart was an attempt to gain intuition of potentially different behavior of the two classes. The chart shows users for each of class in separate colors and plots the user by its average sentiment score and topic count. While this chart does ensure an equal number of tweets are included, we need to make sure we are comparing equal numbers of users. Using jitter will improve the readability of this type of chart.

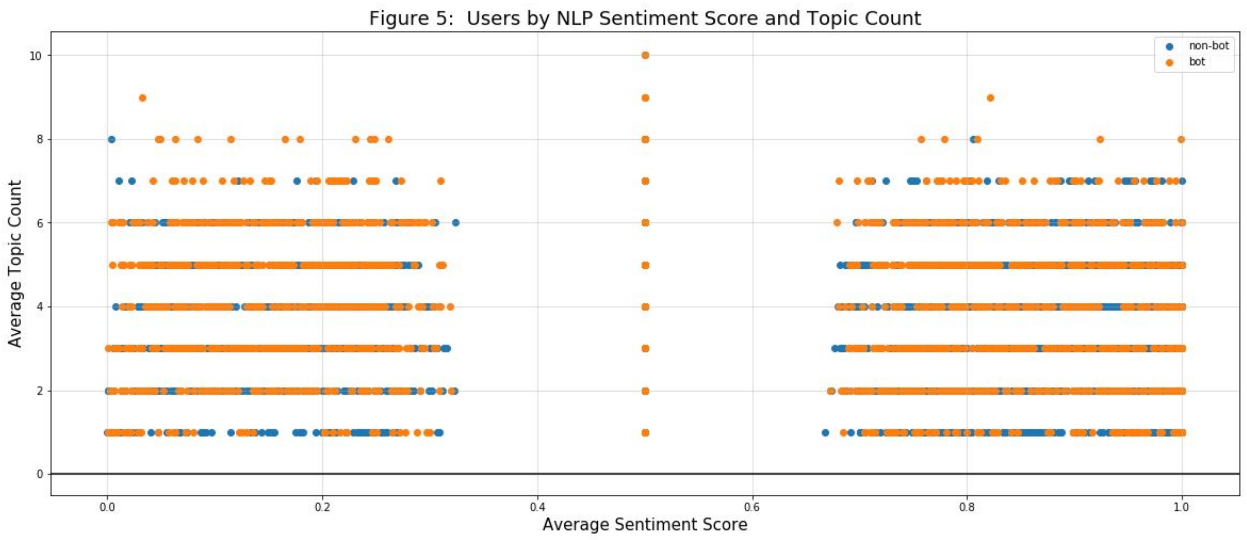


Figure 6. NLP Tweet Length Analysis

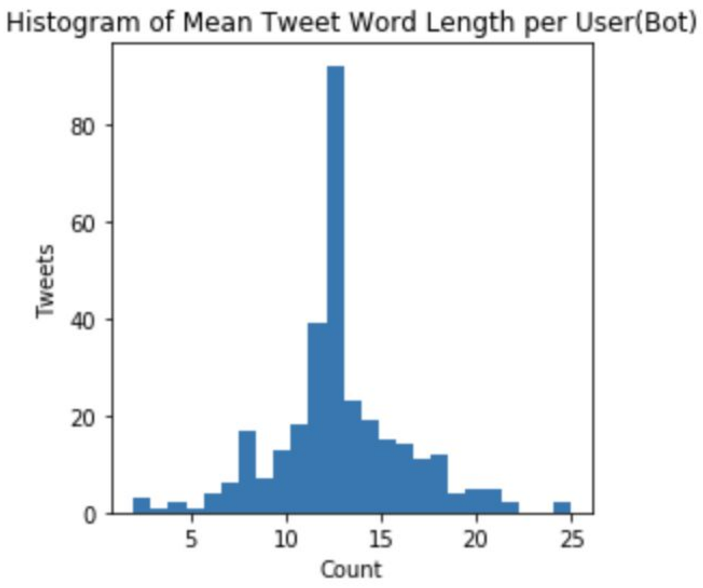


Figure 7. Additional NLP Tweet Length Analysis

```
It looks like we have a average tweet length of 12.87.
Let's select only tweets from bots who tweet messaages with more than 6 words.
```

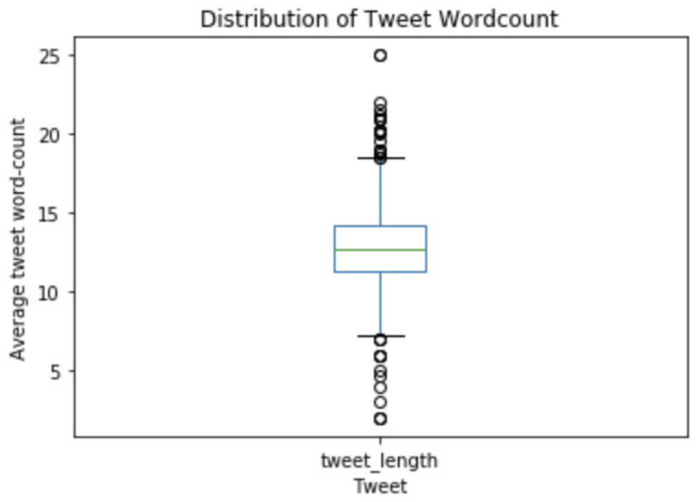
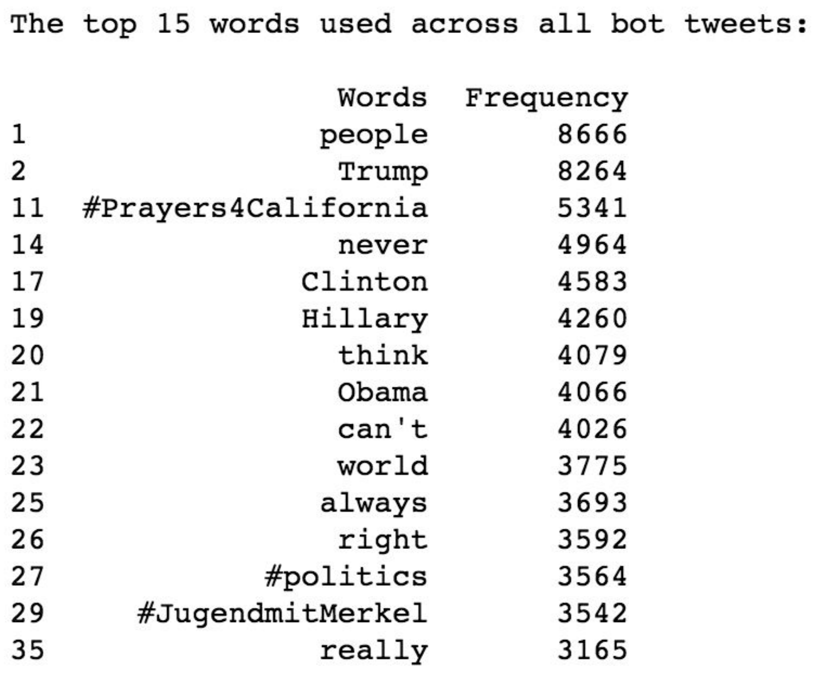


Figure 8. NLP Topic Analysis

```
          Words  Frequency
0         Trump        223
1       Clinton        201
2       Hillary        197
3         Obama        185
4        Donald        181
5        people        170
6       America        163
8     President        161
9         would        160
12    president        158
16        first        155
19        think        154
26        could        148
30        going        146
31        black        145
```

Figure 9. Additional NLP Topic Analysis



```
The top 15 words used across all bot tweets:

                     Words  Frequency
1                   people       8666
2                    Trump       8264
11   #Prayers4California       5341
14                   never       4964
17                 Clinton       4583
19                 Hillary       4260
20                   think       4079
21                   Obama       4066
22                   can't       4026
23                   world       3775
25                  always       3693
26                   right       3592
27               #politics       3564
29        #JugendmitMerkel       3542
35                  really       3165
```

REVISED PROJECT STATEMENT

Our current plan is to identify whether a **Logistic Regression** model with multiple predictors, an **LDA** model with multiple predictors or a **QDA** model with multiple predictors is best at identifying whether a tweet came from a bot or a non-bot.

Feature Engineering

To perform a meaningful bake-off, we need to improve the features we have for each user. The following are the features we intend to develop/refine for continued analysis and modeling:

- User's retweet/tweet ratio
- User's average length of tweet
- User's average NLP-sentiment score
- User's average NLP-topic count
- Average inter-tweet time period

The above attributes do not come directly from our sources but will require a small amount of work to engineer. Specifically, we anticipate grouping tweet information by user and calculating appropriate measures for aggregated fields that will allow us to create these new user features.

Since the above features are all averages or ratios, we do not anticipate the need to standardize or normalize our data further.

It is also worth re-stating that we had hoped to gain more insight from Twitter provided user attributes such as number of followers and number of favorites for each user. At this point, we do not believe we can obtain this data reliably but will explore further if time permits.

We do not intend to introduce interaction variables or polynomials into this list of features. Our intuition does not suggest any obvious combinations or exponentiations, and we wish to avoid over-fitting.

Cross Validation, Feature Selection and Model Selection[1]

We will use forward selection with cross validation to determine which set of features performs best within each of the 3 classes of models. Scoring will be based on classification accuracy only.

The following is more detail on how we see this process working for all three classes of models:

- We will fit and score a base case model with no predictors (y = mode of response variable).
- We will then conduct rounds of feature selection where the number of rounds is equal to the number of predictors.
- In the first round, we will fit and score a model with each of the one predictors alone.
    - Scoring each model in the round will be performed using 5-fold cross validation with the training data.
    - We will select the winner of the round as the model with the best average CV score.

[1] We believe this framework for analysis uses some of the tools we have learned in class correctly. We will seek confirmation/correction from our adviser before finalizing our work.

    - If the CV score for the winner of the round is better than the base, then we will consider the winner of the round as the "current champion model", pending results from subsequent rounds.
    - The new predictor added in the latest, just-completed round will not be considered in subsequent rounds.
- We will continue the above process for as many rounds as there are predictors. In each subsequent round, we will fit and score a model using the features that won in the previous round plus each of the one remaining predictors alone.
    - Scoring each model in the round will be performed using 5-fold cross validation with the training data.
    - We will select the winner of the round as the model with the best average CV score.
    - If the CV score for the winner of the round is better than the "current champion", then we will consider the winner of the round as the "current champion model", pending results from subsequent rounds.
    - The new predictor added in the latest, just-completed round will not be considered in subsequent rounds.
- The winner-of-a-round with the best score across all rounds will be the best-in-class model.
- We will score each best-in-class model on the full training data.

We will repeat the above for the three classes of models. To determine our overall winning model, we will score the best logistics model against the best LDA and the best QDA models on the testing data.

We will attempt to provide a description and interpretation of the overall winning model.

Visualization

We anticipate producing the following charts to demonstrate and explain out work:

- We will create a scatter matrix for all features being evaluated for each class (e.g., bot and non-bot).

- For many pairs of ‘A and B features’, we will plot each of the two classes in different colors on a Feature A/Feature B coordinate system to gain intuition into how well our features separate the classes.
- For each class of model, we will plot cross validation scores for the winner of each round of forward selection.
- For each class of model, we will plot the scores on full training data and full testing data

INITIAL MODEL

We have created one classification model at this point which we have begun to train on our training data. This one model uses sklearn’s **LogisticRegression()** model (with minimum regularization) and has been fitted with several explanatory variables currently in our testing data set.

By simply refitting and scoring on a few combinations of features, we have achieved a classification accuracy of **72.72%** on the training data with the following predictors:

- User average NLP topic count
- User average NLP sentiment score
- User total tweet count

The classification on the testing data, however, was only **.4**

We have included sample code for the above model building in the **Appendix**.

Caveat 1: Since our ratio of bot users to non-bot users is well over 50%, this training classification rate is not as impressive as it first appears and the testing rate is particularly bad.

Caveat 2: We realize the testing data should be in a vault at this point. We promise not to use anything we just learned by looking at this result and intend to add to our data sets as well.

APPENDIX

Initial Model .ipynb Code

```
def misclassification_rate(y_true, y_hat):

    """
    Simple function to calculate error rate.
    Will find and replace with sklearn equivalent.

    y_true:  actual 1/0 for class
    y_hat:  probabilty predicted by model
    """
    # Number of observations
    n_obs = len(y_hat)

    # Initialize error count
    error_count = 0

    # For each observation, count as error
    # if true value is 0 and predicted is >.5
    # of if true is 1 and predicted < .5
    for i in range(n_obs):
        if (y_hat[i] < .5 and y_true[i] == 1):
            error_count = error_count + 1
        elif (y_hat[i] > .5 and y_true[i] == 0):
            error_count = error_count +1
        else:
            pass

    # Return error rate
    return (error_count/n_obs)

# Split the data
train_data, test_data = train_test_split(tweet_df_grouped_user, test_size=.2,
                                         random_state=42,
                                         stratify=tweet_df_grouped_user['is_bot'])

# Create y-vectors
```

```
y_train = train_data['is_bot'].values
y_test = test_data['is_bot'].values

# Create feature set
X_train = train_data[['nlp_count_key_phrases', 'nlp_sentiment_score', 'is_tweet']]
X_test = test_data[['nlp_count_key_phrases', 'nlp_sentiment_score', 'is_tweet']]

# Fit logistics model
multi_logistic_fit = LogisticRegression(C=100000).fit(X_train,y_train)

# Predict y-hat for training data
y_train_hat = multi_logistic_fit.predict(X_train)

# Calculate misclassification and classification rates on training data
missclassification_rate = misclassification_rate(y_train, y_train_hat)
classification_accuracy_train = 1 - missclassification_rate

# Predict y_hat on for X_test
y_test_hat = multi_logistic_fit.predict(X_test)

# Calculate misclassification and classification rates on testing data
missclassification_rate = misclassification_rate(y_train, y_test_hat)
classification_accuracy_test = 1 - missclassification_rate

# Show results
print("Classification Rate - Training data: {0}".format(classification_accuracy_train))
print("Classification Rate - Test data: {0}".format(classification_accuracy_test))
print(multi_logistic_fit.coef_)
```

Initial Model .ipynb Output

```
Classification Rate - Training data: 0.7272727272727273
Classification Rate - Test data: 0.4
[[ 0.27927176 -9.85399974 -0.52977845]]
```