UNIVERSITY OF WATERLOO

MSCI 446 Project Report
Trump Twitter Mining

Aditya Kalia - 20560357
Chris Bombino - 20583450
Xinrui Shao - 20550721
Zhewen Xu - 20565907

Table of Contents

Abstract

On November 8, 2016 history was changed forever. Donald Trump, one of the most controversial political figures ever, beat Hillary Clinton in the United States presidential race. Throughout his campaign Trump had used Twitter to post various tweets that sparked outrage. This report attempted to find a relationship between Donald Trump's tweets and his approval rating. After the models were built, run, and tested, they still had fairly low accuracy. This is due to the complex nature of Trump's approval ratings, and that it is influenced by far more factors than just his tweets. While we originally thought that most of the things that happen with Trump will be tweeted about, we didn't take into account any delay between Trump's bad behavior and his tweets about it.

We also hypothesized that his tweets often contain similar clusters of words, that are also common media headlines. For this hypothesis, k means clustering was used, and the 20 clusters that were gathered from his tweets were indeed all common news headlines such as "Trump Urges Russia to find Clinton's Missing Emails" from CNN.

Introduction

During his campaign for the US presidency, Donald Trump surprised the world by mastering Twitter and turning his tweets and other social media platforms into a political apparatus more effective than those of any other candidates. His success triggered the realization of social media as a powerful tool for politicians to express their political views and interact with the public. After his inauguration, Trump continued to post tweets on a daily basis about his personal views on global issues and his agendas. With his identity now as the 45th President of the United States, all his speeches and announcements receive global attention, among which, a large portion were made publically through his Twitter account. Therefore, one can say that the whole world reacts based on what Trump tweets.

To keep track of the public support for Trump's decisions and views, a large number of pollsters are collecting data through public media like newspaper. The major factor that could affect a president's approval rate is the people's response to his actions and achievements. Being the primary channel of communication between Trump and the public, his tweets receive millions of responses every day, either positive or negative. Therefore, it is possible to predict the impact on his public support whenever he posts a tweet.

One may then ask themselves if any insights can be drawn from mining Trump's Twitter data. For example, maybe a certain word or phrase he tweets causes a drop or spike in his approval rating among the American people. The vision of the project is to provide a model that analyzes all the tweets posted by Trump since his inauguration to find keywords or sets of keywords that have generally resulted in a decline of his approval rate. Using the model, it is possible to predict the change in the approval rate whenever Trump posts a tweet in the future. With this feature, Trump and his advisory team are able to understand public opinions on certain issues and policies, and thus become aware of policies and ideas that are more likely to be accepted by the people. Similarly, the Republican Party and their opponent parties could also make benefits out of the model as it helps them with their strategic decision-making in general.

Related Scholarly Work

Since Donald Trump has only been a political figure for a relatively short period of time, there have not been many published scholarly articles about him, and none that specifically match this project. However, the team has explored other twitter mining studies, some of whose techniques and results can be taken into consideration when developing the models.

One other project in particular tested the theory that every non-hyperbolic tweet came from an iPhone and was tweeted by his staff, and every hyperbolic tweet came from an Android and was tweeted by Donald Trump himself. This analysis, performed by Stack Overflow Data Scientist David Robinson, concluded that there is indeed a difference in the tweets that come from an iPhone versus an Android, and that the negative tweets coming from an iPhone could be a staff member attempting to sound like Donald Trump.

A scholarly work that the team looked into is “Will Sanders Supporters Jump Ship for Trump? Fine-grained Analysis of Twitter Followers”, This article focused on analyzing how many Bernie Sanders supporters would shift their vote towards Donald Trump in the election, based on twitter followers. In this scholarly article the decision variable is the number of followers that are following Bernie Sanders and then decide to follow Donald Trump. With this Key relationship it can be mapped the percentage of voters that shifted towards Donald Trump.

Another inspiring article is “Harnessing Twitter, Donald Trump Style” by Andy Pemberton. This article focuses on how the 45th President uses twitter to listen to his audience and test the waters. He was able to poll himself on twitter and see how his base reacted. The more reactions he got on Twitter, the crazier he got with his tweets. He attacked climate change, Obamacare, and even talked about throwing Clinton in jail. Therefore, this article provided a general expectation of the outcome of the analytical model and the possible business insights that can be explored.

Variables and Data

Variables

The dependent variable that the model predicts is a binary variable that describes whether the public approval rate for Donald Trump increases or decreases as a result of his tweets. It is assumed that there are only two outcomes, either the rate decreases or not (increasing or unchanged).

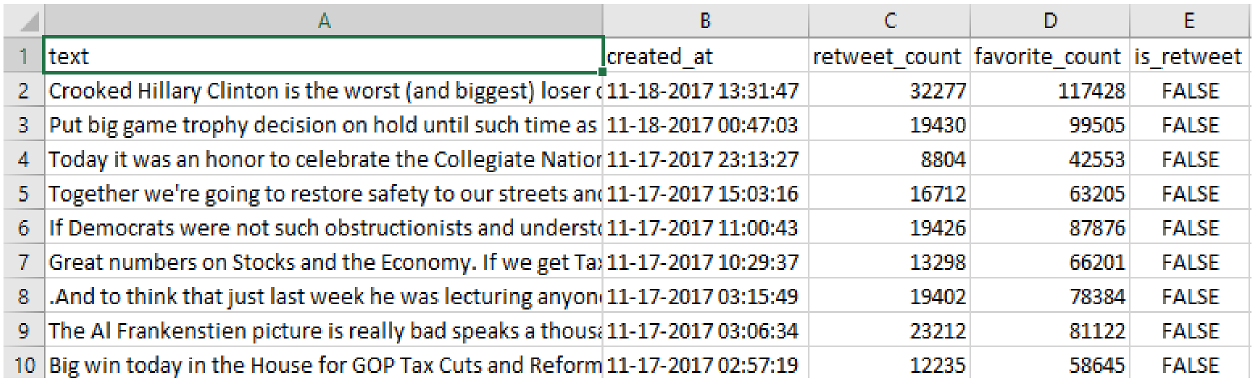
One of the key explanatory variables in the tweet-analysis model would be text contents of all tweets posted by Donald Trump on a single day. It will be collected in the format of pure text. Since the primary function of the system is to perform predictive analysis based on his tweets, it is extremely crucial to capture all the words he posted to find a correlation between the change in his approval rate and specific vocabularies from his tweets.

The public feedback for each tweet is also considered. This allows the system to understand the public attention on the topic discussed in the tweet and the people's opinions about it. This information is collected as three numeric variables: the average numbers of likes, replies, and retweets for each tweet on each day. A large value in any of these three variables indicates that the topic has aroused mass social attention. In other cases where the numbers of replies and retweets are high while the number of likes is relatively low, the tweet might be about a heated topic but the public does not agree with what Trump says in the tweet.

Another important variable to consider is the number of posts on each day, which is also collected as a numeric variable. The importance of this information can be understood intuitively. If the data shows there is a large number of tweets on a specific day, it could indicate that a major event has happened on that day, which could result in similar contents in the tweets. For example, if a severe earthquake hits california, and Trump posts more tweets than regular on the same day, it is very like that many of these tweets share a common theme about the earthquake.

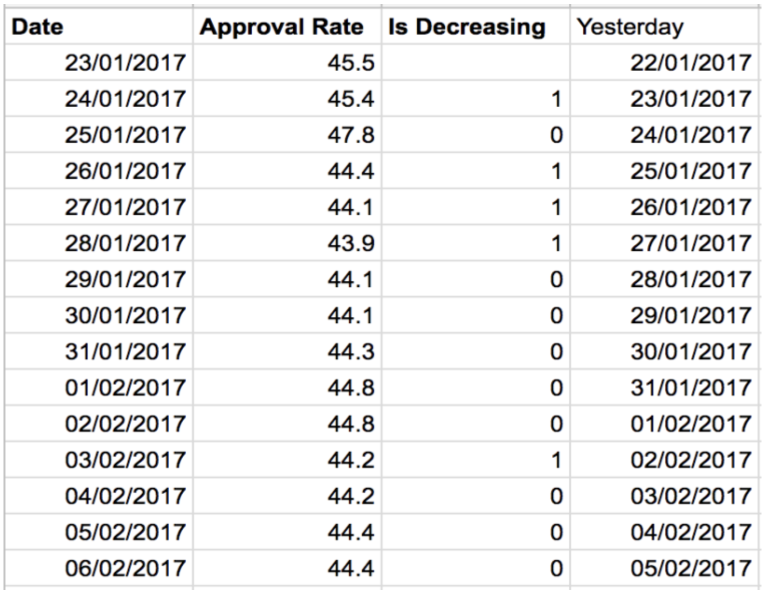
Collection

For this project, the first step in collecting data was to use a crawler to crawl through Trump's twitter and gather information about his tweets. The tweet data that was gathered was the content of the tweet (text), the date and timestamp of the tweet (numeric), the number of retweets (numeric), favorites (numeric), and if the tweet was retweeted from another account (boolean). The below figure shows the beginning of the mined twitter data.

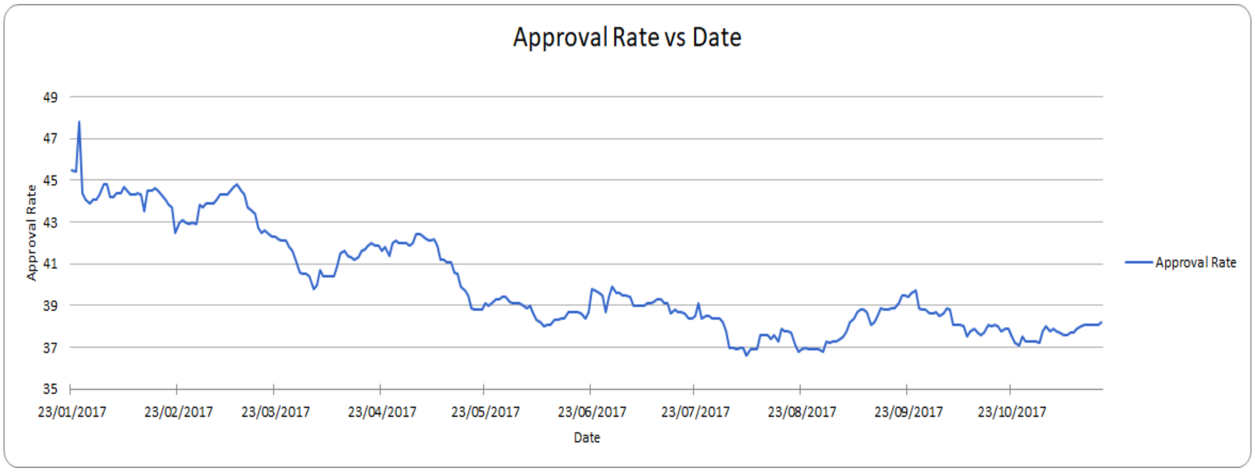


	A	B	C	D	E
1	text	created_at	retweet_count	favorite_count	is_retweet
2	Crooked Hillary Clinton is the worst (and biggest) loser c	11-18-2017 13:31:47	32277	117428	FALSE
3	Put big game trophy decision on hold until such time as	11-18-2017 00:47:03	19430	99505	FALSE
4	Today it was an honor to celebrate the Collegiate Natior	11-17-2017 23:13:27	8804	42553	FALSE
5	Together we're going to restore safety to our streets anc	11-17-2017 15:03:16	16712	63205	FALSE
6	If Democrats were not such obstructionists and underst	11-17-2017 11:00:43	19426	87876	FALSE
7	Great numbers on Stocks and the Economy. If we get Ta:	11-17-2017 10:29:37	13298	66201	FALSE
8	.And to think that just last week he was lecturing anyon	11-17-2017 03:15:49	19402	78384	FALSE
9	The Al Frankenstien picture is really bad speaks a thous:	11-17-2017 03:06:34	23212	81122	FALSE
10	Big win today in the House for GOP Tax Cuts and Reform	11-17-2017 02:57:19	12235	58645	FALSE

In addition to this, we also collected information about Trump's approval rating from his Inauguration date (January 20, 2017.) This Inauguration data was collected manually from https://projects.fivethirtyeight.com/trump-approval-ratings by recording the date and approval rating from that date. After this data was collected, another column was added showing if the approval rating was decreasing from the previous day. All of this data was numeric, with the isDecreasing value being binary. The figure below is a graph of Trump's approval rating from his Inauguration up to late October.



Date	Approval Rate	Is Decreasing	Yesterday
23/01/2017	45.5		22/01/2017
24/01/2017	45.4	1	23/01/2017
25/01/2017	47.8	0	24/01/2017
26/01/2017	44.4	1	25/01/2017
27/01/2017	44.1	1	26/01/2017
28/01/2017	43.9	1	27/01/2017
29/01/2017	44.1	0	28/01/2017
30/01/2017	44.1	0	29/01/2017
31/01/2017	44.3	0	30/01/2017
01/02/2017	44.8	0	31/01/2017
02/02/2017	44.8	0	01/02/2017
03/02/2017	44.2	1	02/02/2017
04/02/2017	44.2	0	03/02/2017
05/02/2017	44.4	0	04/02/2017
06/02/2017	44.4	0	05/02/2017



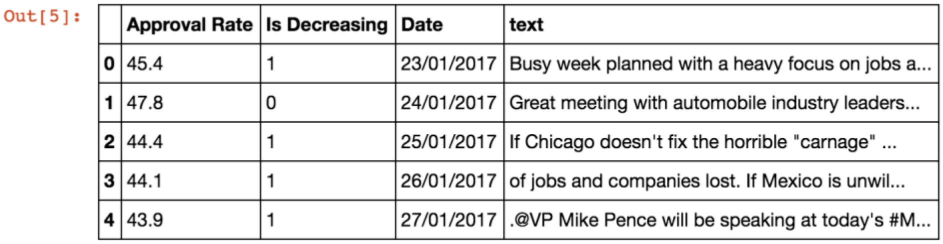
Feature Transformation-merging

Before feeding cleaned tweets into data mining algorithms for predicting if Trump's daily approval rating decreases or not, all the tweets in one date are merged into one text feature.

For K-mean clustering, this step is unnecessary, because the purpose of K-mean clustering is to find topics among each individual tweets.

After merging all the tweets in one date into a single row, the text data is joined with the approval rate dataset based on "date" and "yesterday" respectively, so that predictions on approval rate for tomorrow can be made based on today's tweets.

Table below is the result after merging.



Out[5]:

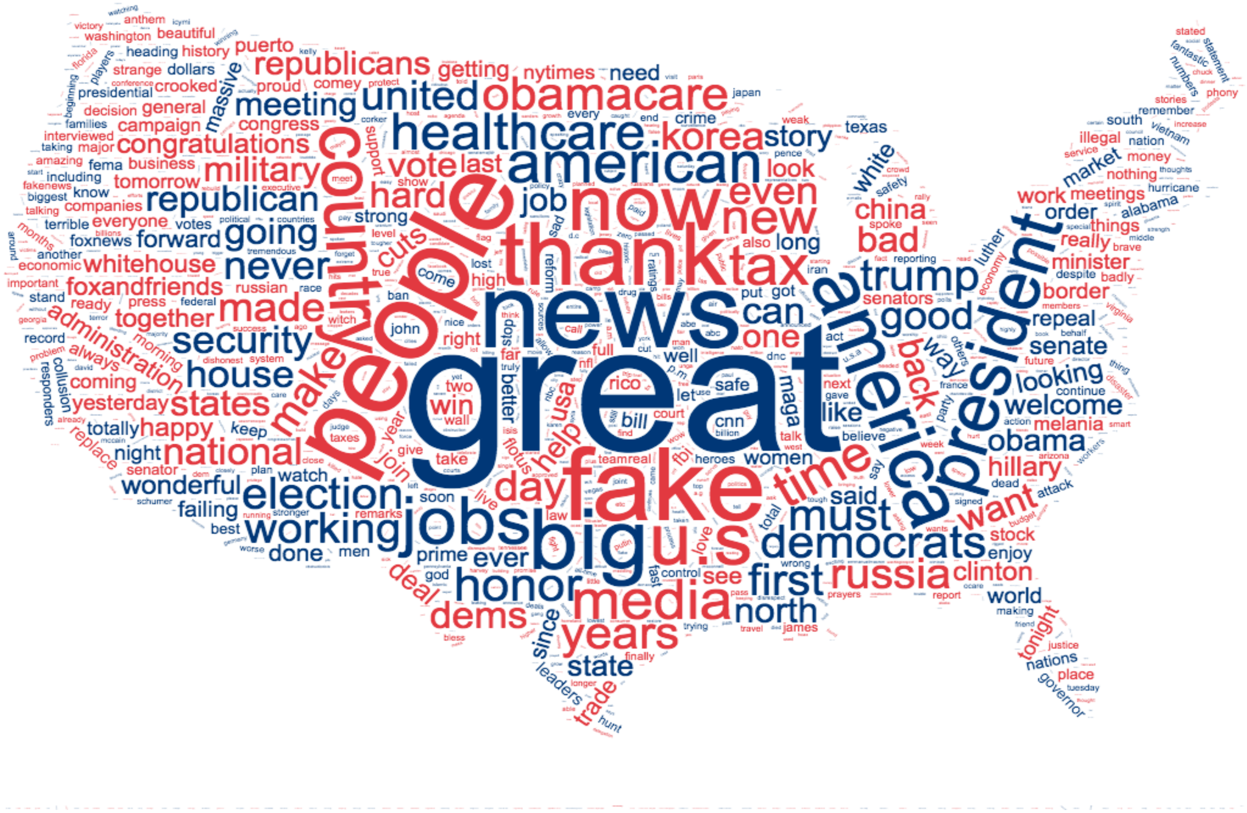
	Approval Rate	Is Decreasing	Date	text
0	45.4	1	23/01/2017	Busy week planned with a heavy focus on jobs a...
1	47.8	0	24/01/2017	Great meeting with automobile industry leaders...
2	44.4	1	25/01/2017	If Chicago doesn't fix the horrible "carnage" ...
3	44.1	1	26/01/2017	of jobs and companies lost. If Mexico is unwil...
4	43.9	1	27/01/2017	.@VP Mike Pence will be speaking at today's #M...

Cleaning

From here the tweet data was cleaned to remove common words of the English language that give little meaning to an overall tweet (words such as "a", "and", "the"), as well as URLs, N/A, special characters, and usernames that the President may have tweeted at. The reason these were removed from the data is because they provide little to no value to the actual content of the tweet, and end up just cluttering the tweet with useless information. To make sure that the tweets were cleaned as expected, all clean tweets were printed out in the notebook console for the team to check. After cleaning, all the tweets were tokenized into words and all these words were stemmed in order to reduce the total number of unique words for visualization as demonstrated in the codes in the Appendix.

Visualization

To give a general representation of what words Trump uses the most, the tweets were then parsed into individual words, and counted. This word cloud allowed us to visualize all words that Trump uses, and which words are most popular.



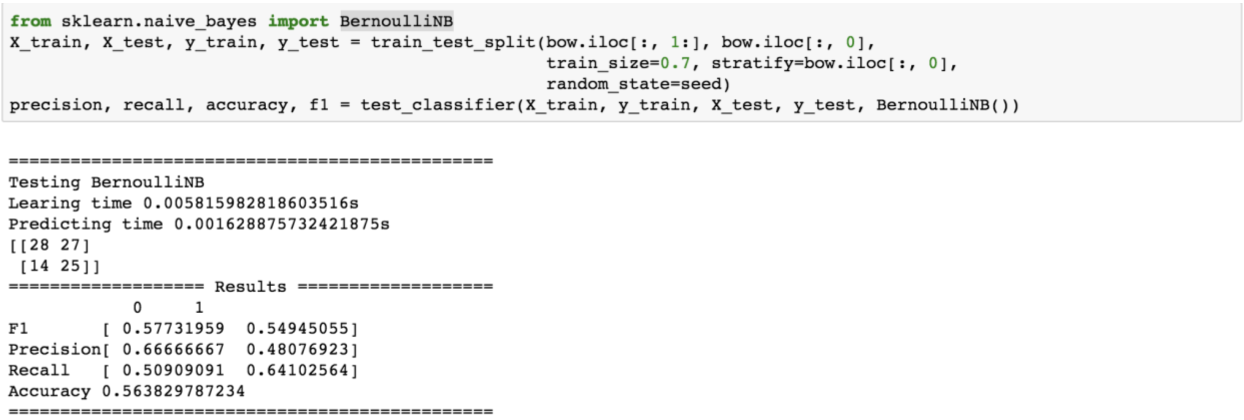
Feature Transformation-bag-of-words

The next step is to transform tokenized tweets into bag of words. For each text feature/tweets, the length of bag of words is equal to the size of vocabulary (1071). Each element in the bag of words is a binary variable 1 or 0, where 1 stands for a particular word is in this text feature/tweets, 0 otherwise.

Based on the word cloud, some interesting patterns can be explored regarding his usage of words in his tweets. Clearly, the words great, people, news, America, and fake are the most popular. This makes sense because the official campaign slogan of the Trump administration was “Make America Great Again.” Also, certain topics that he frequently discusses are reflected, such as “jobs”, “election”, “tax”, etc. Moreover, judging from the similar sizes of certain words like “fake” and “news”, “healthcare” and “obamacare”, “russia” and “korea”, etc., one can guess that Trump tends to use those words together. Therefore, it is likely that the other word also appears in the same tweet if one of the keywords is in his tweet.

Models Development

Naive Bayes



```
from sklearn.naive_bayes import BernoulliNB
X_train, X_test, y_train, y_test = train_test_split(bow.iloc[:, 1:], bow.iloc[:, 0],
                                                    train_size=0.7, stratify=bow.iloc[:, 0],
                                                    random_state=seed)
precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, BernoulliNB())
```

```
===============================================
Testing BernoulliNB
Learing time 0.005815982818603516s
Predicting time 0.001628875732421875s
[[28 27]
 [14 25]]
=================== Results ===================
            0         1
F1       [ 0.57731959  0.54945055]
Precision[ 0.66666667  0.48076923]
Recall   [ 0.50909091  0.64102564]
Accuracy 0.563829787234
===============================================
```

One of the main goals of this project was to determine if the approval rating would be increasing or decreasing the day after Trump tweets. The Naive Bayes classifier was the first method we used to check if the approval rating would be increasing or decreasing the day after Trump sent out a series of tweets. This was the first model we used because it can be trained very quickly, doesn’t require as much training data as other models, and is not very computationally intensive.

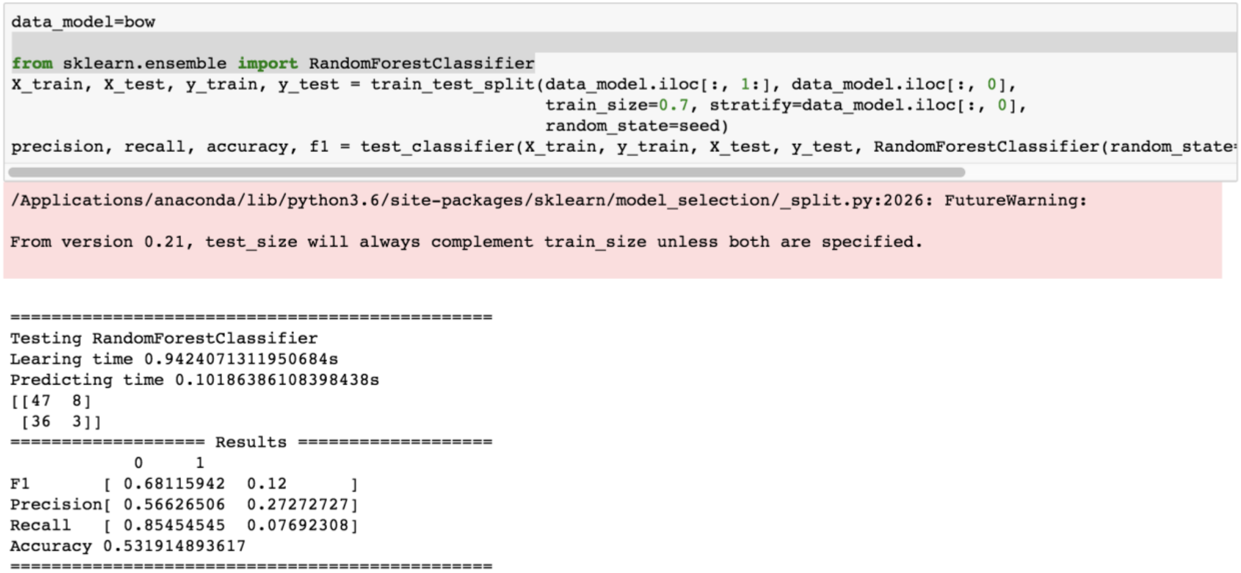
The confusion matrix was the following:

28	27
14	25

The Naive Bayes model was better at correctly predicting an increase in approval rating than a decrease in approval rating. This gives an overall accuracy of just over 53%. The one thing of particular interest with the confusion matrix is the number of false positives, and specifically the fact that there are more false positives than true positives. This gives a precision that is lower than 50% (48.07% to be exact) which is not a very good score.

Random Forest

The random forest is an ensemble learning method that works by constructing many decision trees on different subsets of features and taking the average results of all the decision trees. In this case, the feature it is creating the subsets on are the bag of words.



```
data_model=bow

from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(data_model.iloc[:, 1:], data_model.iloc[:, 0],
                                                    train_size=0.7, stratify=data_model.iloc[:, 0],
                                                    random_state=seed)
precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, RandomForestClassifier(random_state
```

```
/Applications/anaconda/lib/python3.6/site-packages/sklearn/model_selection/_split.py:2026: FutureWarning:

From version 0.21, test_size will always complement train_size unless both are specified.
```

```
================================================
Testing RandomForestClassifier
Learing time 0.9424071311950684s
Predicting time 0.10186386108398438s
[[47  8]
 [36  3]]
=================== Results ===================
            0     1
F1       [ 0.68115942  0.12      ]
Precision[ 0.56626506  0.27272727]
Recall   [ 0.85454545  0.07692308]
Accuracy 0.531914893617
================================================
```

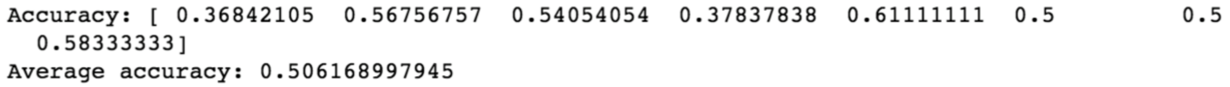
The random forest takes input of a “Bag of words”, a bag of words are all the words Trump has used in his tweets. The bags are separated by day, and since it has been 311 days since Trump’s inauguration, there are 311 unique bags of words with all of the words Trump has used on that day. Another input that is taken is the approval rate of Trump. The bag of words are mapped to the approval rate. The model outputs a binary variable, with 1 meaning a decreasing approval rate, and 0 meaning the approval rate stays the same or increases.

From the model the following confusion matrix is as constructed:

47	8
36	3

It is interesting to note that the true negative score is very high for this model, indicating that it is able to associate well when the approval rate is going to stay the same or increase (binary value = 0), however the model has a low true positive score, indicating it is not able to effectively predicate when the approval rating is going go to go down based on a bag of words.

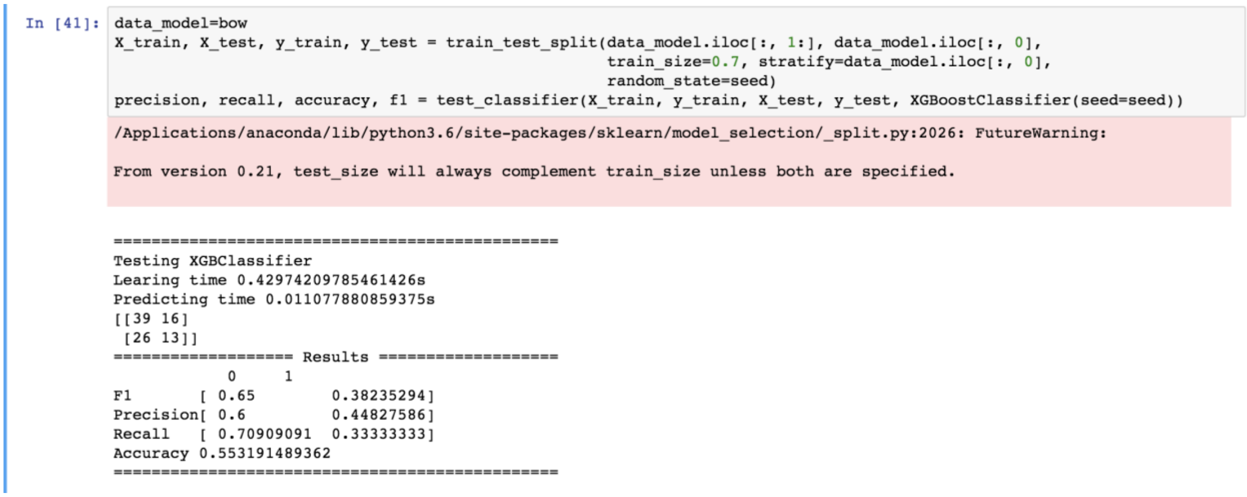
The model has a cross validation average accuracy of 0.506168997. This average accuracy was determined by testing the model with 8 fold cross validation. The following is the accuracy scores of each of the 8 fold cross validation :



```
Accuracy: [ 0.36842105  0.56756757  0.54054054  0.37837838  0.61111111  0.5         0.5
  0.58333333]
Average accuracy: 0.506168997945
```

XG Boost

The third model for supervised learning used is XGBoost. XGBoost is an optimized distributed gradient boosting system designed to be highly efficient, flexible.

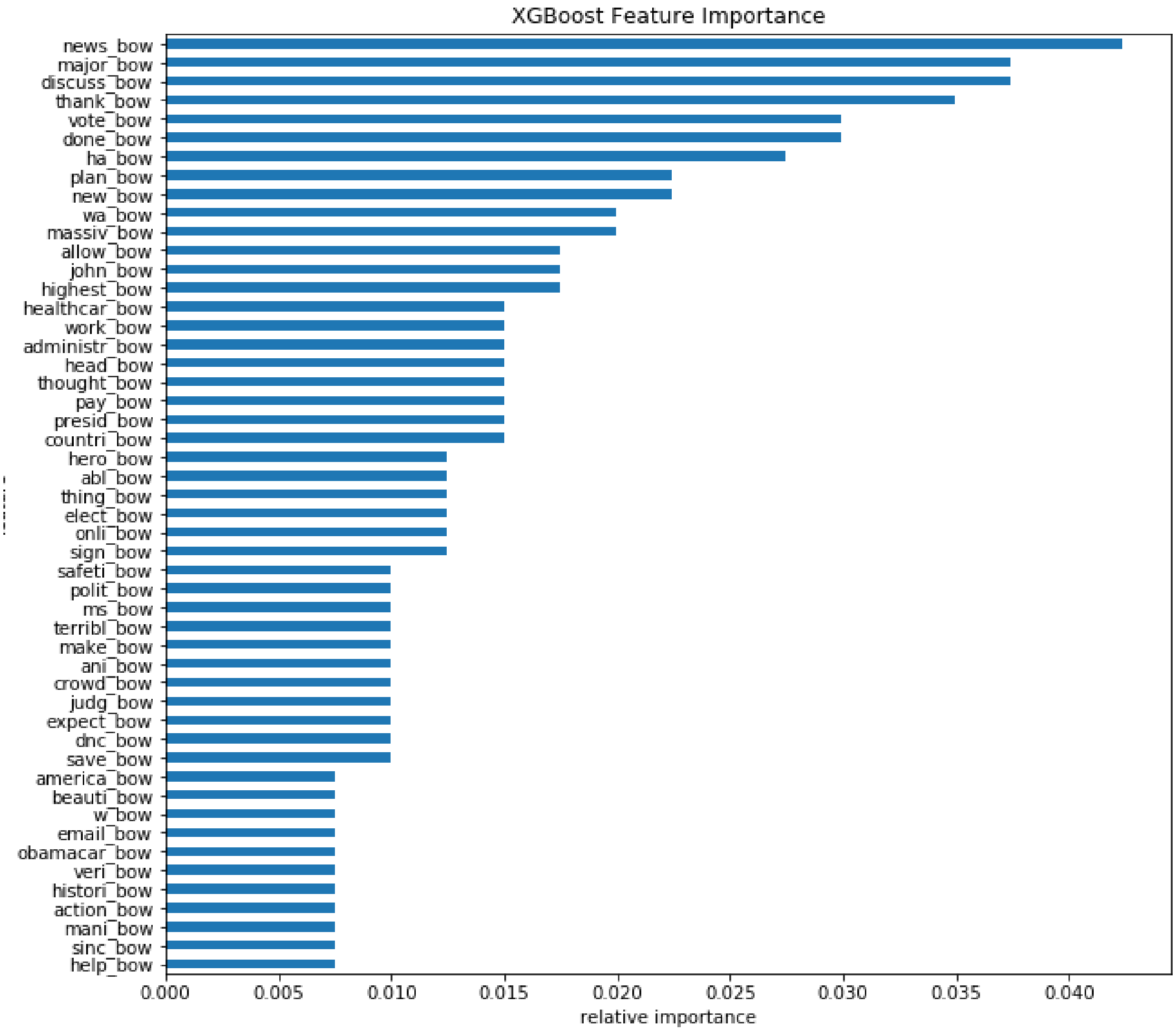


```
In [41]: data_model=bow
         X_train, X_test, y_train, y_test = train_test_split(data_model.iloc[:, 1:], data_model.iloc[:, 0],
                                                             train_size=0.7, stratify=data_model.iloc[:, 0],
                                                             random_state=seed)
         precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, XGBoostClassifier(seed=seed))

         /Applications/anaconda/lib/python3.6/site-packages/sklearn/model_selection/_split.py:2026: FutureWarning:

         From version 0.21, test_size will always complement train_size unless both are specified.


         ==============================================
         Testing XGBClassifier
         Learing time 0.42974209785461426s
         Predicting time 0.011077880859375s
         [[39 16]
          [26 13]]
         =================== Results ===================
                     0      1
         F1        [ 0.65        0.38235294]
         Precision[ 0.6         0.44827586]
         Recall    [ 0.70909091  0.33333333]
         Accuracy 0.553191489362
         ==============================================
```

Confusion matrix:

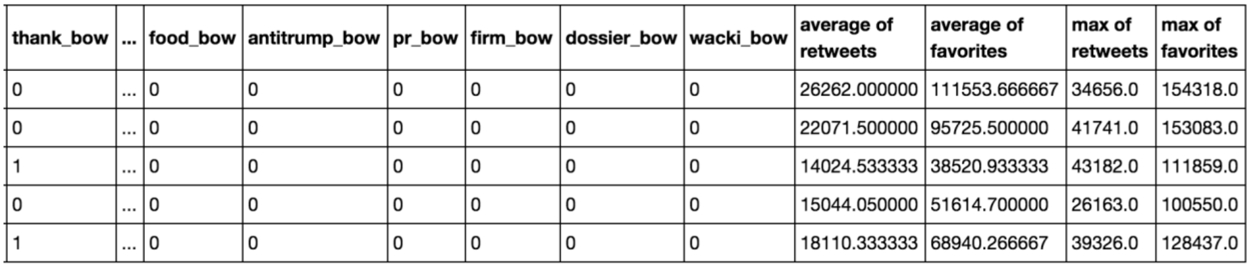
39	16
26	13

Compared to the first two model, XGBoost performs better than randomforest but worse than naive bayes in terms of precision for class 1(0.448) and total accuracy(0.55). One nice feature that XGBoost has is to output feature importance as shown below where the most important features are displayed given their average impact on the approval rate. Some insights can be drawn from the chart below. For example, one can say that if a tweet contains words like “news”, “major” and “discuss”, it is likely to have a strong impact on Trump’s approval rating, either positive or negative. Furthermore, the frequency a word is used does not have a decisive effect on the importance of that word in changing the approval rating since the relative importance is calculated by taking the average of the absolute value of the change in the rating associated with tweets that contain this word. Correspondingly, it is shown in the chart below that most of the important words do not have a noticeably high frequency of appearance in Trump’s tweets.



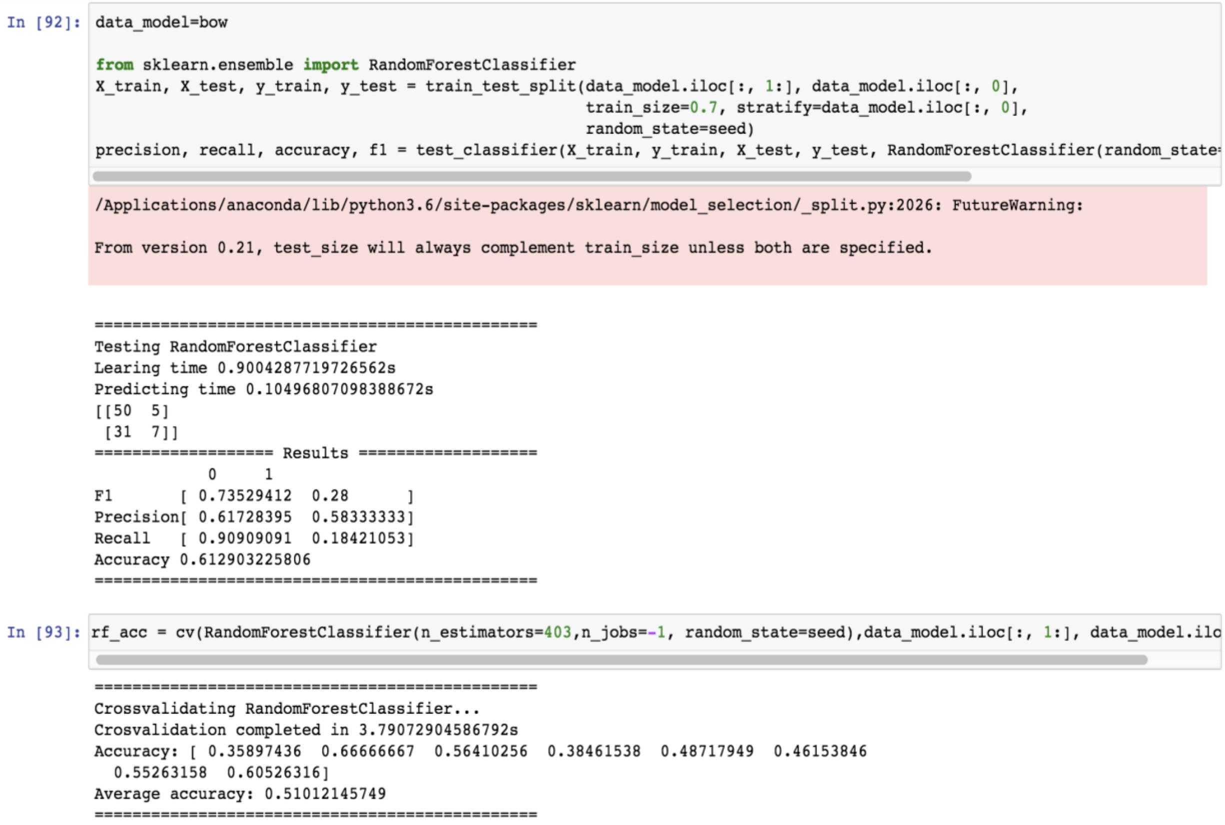
Additional Features for Supervised Models

To get better prediction accuracy, the first thing tried was adding other features including average number of retweets, favorites for each date and maximum number of retweets, favorites in addition to bag of words as shown below.



thank_bow	...	food_bow	antitrump_bow	pr_bow	firm_bow	dossier_bow	wacki_bow	average of retweets	average of favorites	max of retweets	max of favorites
0	...	0	0	0	0	0	0	26262.000000	111553.666667	34656.0	154318.0
0	...	0	0	0	0	0	0	22071.500000	95725.500000	41741.0	153083.0
1	...	0	0	0	0	0	0	14024.533333	38520.933333	43182.0	111859.0
0	...	0	0	0	0	0	0	15044.050000	51614.700000	26163.0	100550.0
1	...	0	0	0	0	0	0	18110.333333	68940.266667	39326.0	128437.0

After feeding the new features into the same random forest model used before , a major improvement in the model performance was achieved.



```
In [92]: data_model=bow

         from sklearn.ensemble import RandomForestClassifier
         X_train, X_test, y_train, y_test = train_test_split(data_model.iloc[:, 1:], data_model.iloc[:, 0],
                                                             train_size=0.7, stratify=data_model.iloc[:, 0],
                                                             random_state=seed)
         precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, RandomForestClassifier(random_state=
```

```
/Applications/anaconda/lib/python3.6/site-packages/sklearn/model_selection/_split.py:2026: FutureWarning:

From version 0.21, test_size will always complement train_size unless both are specified.
```

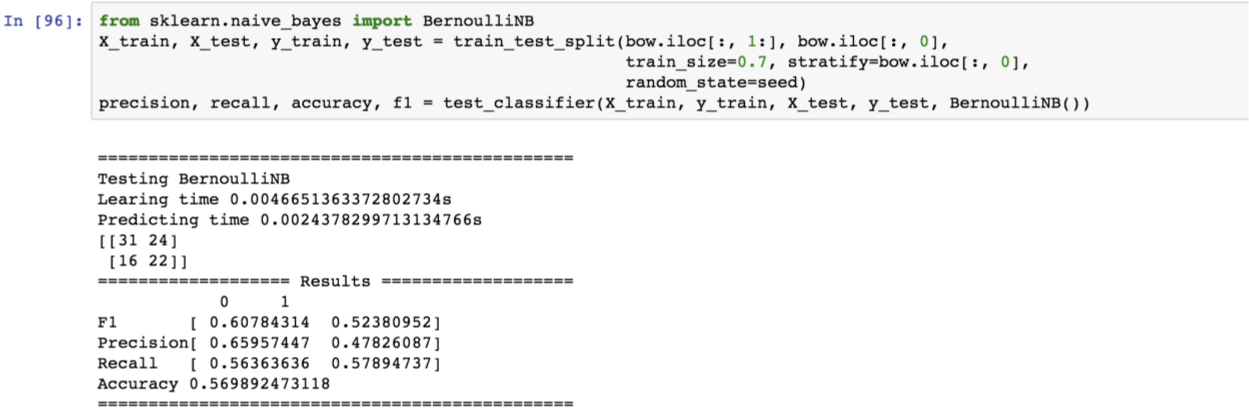
```
===============================================
Testing RandomForestClassifier
Learing time 0.9004287719726562s
Predicting time 0.10496807098388672s
[[50  5]
 [31  7]]
=================== Results ===================
            0     1
F1       [ 0.73529412  0.28      ]
Precision[ 0.61728395  0.58333333]
Recall   [ 0.90909091  0.18421053]
Accuracy 0.612903225806
===============================================
```

```
In [93]: rf_acc = cv(RandomForestClassifier(n_estimators=403,n_jobs=-1, random_state=seed),data_model.iloc[:, 1:], data_model.ilo
```

```
===============================================
Crossvalidating RandomForestClassifier...
Crosvalidation completed in 3.79072904586792s
Accuracy: [ 0.35897436  0.66666667  0.56410256  0.38461538  0.48717949  0.46153846
  0.55263158  0.60526316]
Average accuracy: 0.51012145749
===============================================
```

Precision for class 1 increased from 0.27 to 0.58, accuracy increased from 0.53 to 0.61, and average accuracy for 8-fold cross validation went from 0.506 to 0.51.

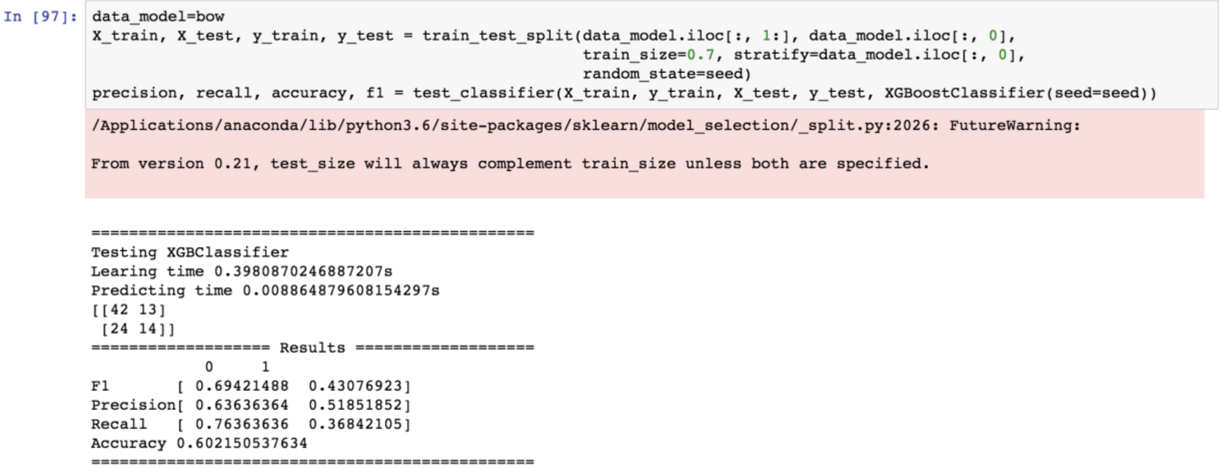
For Naive Bayes, the improvement of performance is less significant. Although overall accuracy increased from 0.5319 to 0.5698, the precision for class decreased from 0.4807 to 0.478.



```
In [96]: from sklearn.naive_bayes import BernoulliNB
         X_train, X_test, y_train, y_test = train_test_split(bow.iloc[:, 1:], bow.iloc[:, 0],
                                                             train_size=0.7, stratify=bow.iloc[:, 0],
                                                             random_state=seed)
         precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, BernoulliNB())
```

```
=============================================
Testing BernoulliNB
Learing time 0.004665136337280273 4s
Predicting time 0.0024378299713134766s
[[31 24]
 [16 22]]
================== Results ==================
            0         1
F1       [ 0.60784314  0.52380952]
Precision[ 0.65957447  0.47826087]
Recall   [ 0.56363636  0.57894737]
Accuracy 0.569892473118
=============================================
```

For XGBoost, the improvement of performance is more significant than naive bayes.



```
In [97]: data_model=bow
         X_train, X_test, y_train, y_test = train_test_split(data_model.iloc[:, 1:], data_model.iloc[:, 0],
                                                             train_size=0.7, stratify=data_model.iloc[:, 0],
                                                             random_state=seed)
         precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, XGBoostClassifier(seed=seed))
```

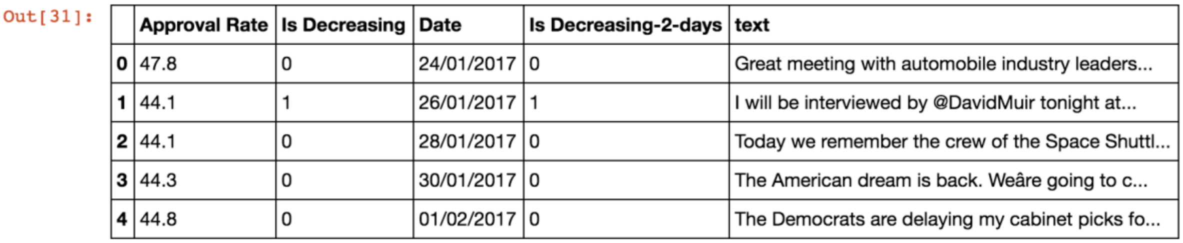
```
/Applications/anaconda/lib/python3.6/site-packages/sklearn/model_selection/_split.py:2026: FutureWarning:

From version 0.21, test_size will always complement train_size unless both are specified.
```

```
=============================================
Testing XGBClassifier
Learing time 0.3980870246887207s
Predicting time 0.008864879608154297s
[[42 13]
 [24 14]]
================== Results ==================
            0         1
F1       [ 0.69421488  0.43076923]
Precision[ 0.63636364  0.51851852]
Recall   [ 0.76363636  0.36842105]
Accuracy 0.602150537634
=============================================
```

Precision for class 1 increased from 0.448 to 0.518 and the accuracy went from 0.553 to 0.6.

Merging Tweets in Two Consecutive Dates

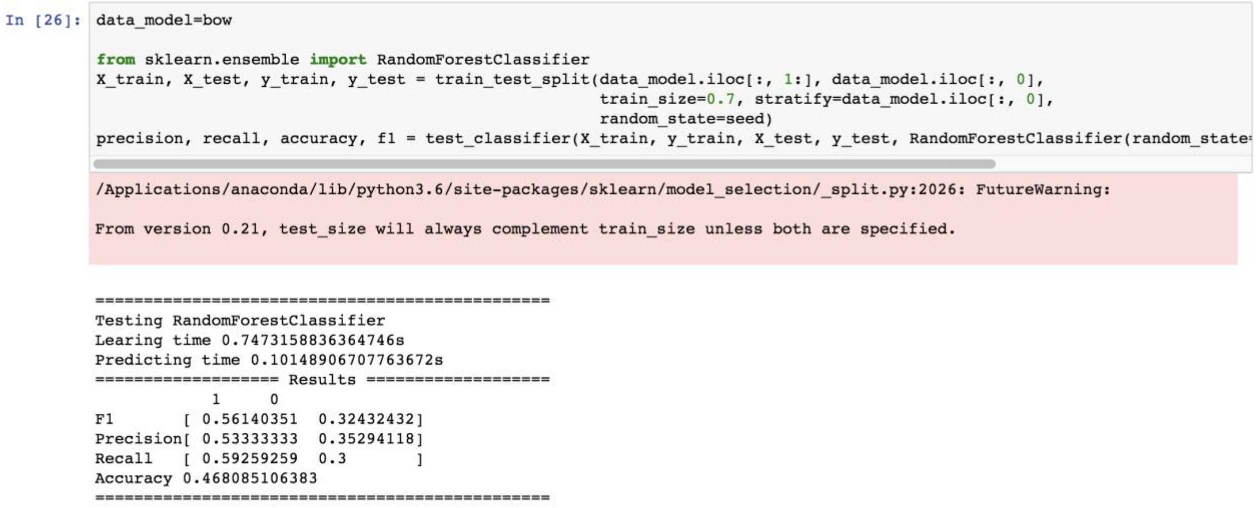
Another aspect tried to improve the performance of prediction models is to merge tweets in two consecutive dates instead of tweets in the same date and to predict its effects on the approval rate for the third date. The logic behind this is that Trump's daily approval rate on a particular date may be affected by his tweets which are more than one day ago. The merged dataset is shown below.



Out[31]:

	Approval Rate	Is Decreasing	Date	Is Decreasing-2-days	text
0	47.8	0	24/01/2017	0	Great meeting with automobile industry leaders...
1	44.1	1	26/01/2017	1	I will be interviewed by @DavidMuir tonight at...
2	44.1	0	28/01/2017	0	Today we remember the crew of the Space Shuttl...
3	44.3	0	30/01/2017	0	The American dream is back. Weâre going to c...
4	44.8	0	01/02/2017	0	The Democrats are delaying my cabinet picks fo...

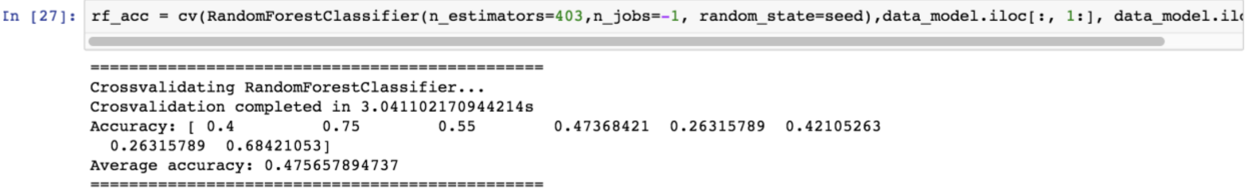
By performing the same cleaning and applying the same sets of models, it turned out that this transformation hindered the overall performance for all the models applied.



```
In [26]: data_model=bow

         from sklearn.ensemble import RandomForestClassifier
         X_train, X_test, y_train, y_test = train_test_split(data_model.iloc[:, 1:], data_model.iloc[:, 0],
                                                             train_size=0.7, stratify=data_model.iloc[:, 0],
                                                             random_state=seed)
         precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, RandomForestClassifier(random_state=
```

```
/Applications/anaconda/lib/python3.6/site-packages/sklearn/model_selection/_split.py:2026: FutureWarning:

From version 0.21, test_size will always complement train_size unless both are specified.
```

```
===============================================
Testing RandomForestClassifier
Learing time 0.7473158836364746s
Predicting time 0.10148906707763672s
=================== Results ===================
            1     0
F1       [ 0.56140351  0.32432432]
Precision[ 0.53333333  0.35294118]
Recall   [ 0.59259259  0.3       ]
Accuracy 0.468085106383
===============================================
```



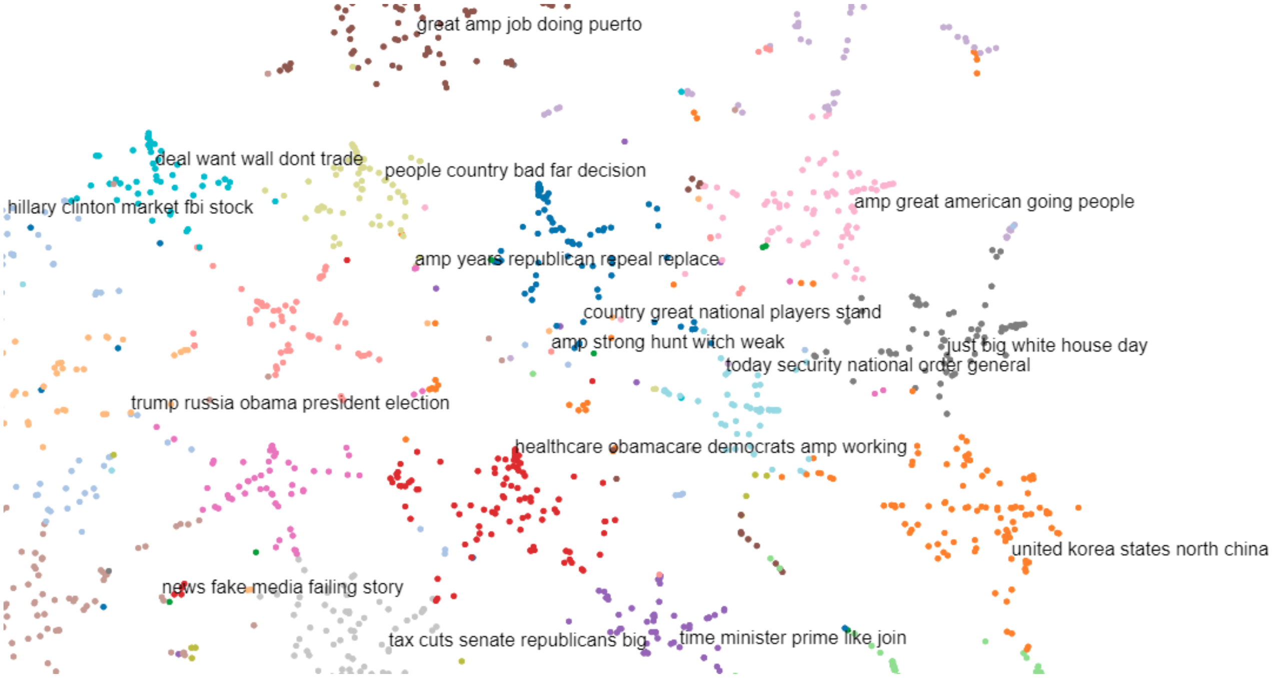
```
In [27]: rf_acc = cv(RandomForestClassifier(n_estimators=403,n_jobs=-1, random_state=seed),data_model.iloc[:, 1:], data_model.il
```

```
===============================================
Crossvalidating RandomForestClassifier...
Crosvalidation completed in 3.041102170944214s
Accuracy: [ 0.4         0.75        0.55        0.47368421  0.26315789  0.42105263
  0.26315789  0.68421053]
Average accuracy: 0.475657894737
===============================================
```

K-Mean Clustering

One of the primary objectives of this project is to explore topic clusters that Trump tends to discuss based on his tweets. By doing this, it is possible to classify a new tweet into an existing cluster based on its “distance” (in terms of the words in the tweet) from the cluster centers. It is the most reasonable and efficient to adopt the unsupervised K-mean clustering model to find the cluster centers given a certain number of clusters (k). What makes the result of this model meritorious is that the cluster centers are constantly being redefined whenever a new tweet is entered into the dataset. If in the future, Trump starts to tweet a lot about a new topic to a quantity that leverages any of the existing cluster center, a new topic center will be identified about that topic. Therefore, important business insights can be drawn from this model to not only discover his current topic centers, but also discover the shift of Trump’s attention onto new topics by comparing the topic clusters of different time.

To perform K-mean clustering on the tweets, all the tweets are transformed into bag of words as discussed in feature transformation. The distances between each tweet is computed using Jaccard distance in which the all words with a value of 0 in the bag of words is disregarded because matching zeros do not contribute to the similarity of two tweets. Based on the distance matrix, the cluster centers can be identified by going through iterations for K-mean clustering. The results can be visualized on a scatter plot. In addition, in order to clearly show the topic clusters, the top 5 frequently used words in each cluster is displayed to represent the cluster.



As shown in the scatterplot above, all the tweets of focus are represented as single dots. The result of K-mean clustering with 20 clusters generated some interesting findings. For example, one cluster that involves the frequent use of words “united”, “states”, “north”, “korea”, and “china” represents a topic or an issue that concerns United States, China, and North Korea, potentially indicating the nuclearization issue of North Korea. Another interesting topic includes “trump”, “obama”, “russia”, “president”, and “election” as its top 5 words. This reflects that even 10 months after the election, the legitimacy problem of Trump being the president is still being tweeted about, with Russia very much involved in the discussion. Unsurprisingly, the cluster with top words “fake”, “news”, “media”, “failing”, and “story” is identified among one of the largest clusters, suggesting that Trump has mentioned that the failing domestic media is delivering fake news and stories to the public in a large portion of his tweets.

Comparing with the results of the test run of this model in October, there has been an arising topic cluster containing “great”, “job”, “doing”, and “puerto”. This has been noted as a significant change because it indicates that the amount of tweets about the rescue and recovery of Puerto Rico after the storm in September has significantly increased to such a level that all top 5 words have replaced the previous top words.

Conclusion

Clearly the models that were tested did not achieve the desired accuracy when attempting to predict Trump's approval rating based on his tweets from the previous day. To improve the models, additional features are considered, such as number of likes and number of retweets for each tweet. A slight increase in the accuracy of each (supervised) prediction models has been noticed, indicating that the public attention on a certain tweet, as represented by the number of likes and retweets, has a slight impact on Trump's public approval rating.

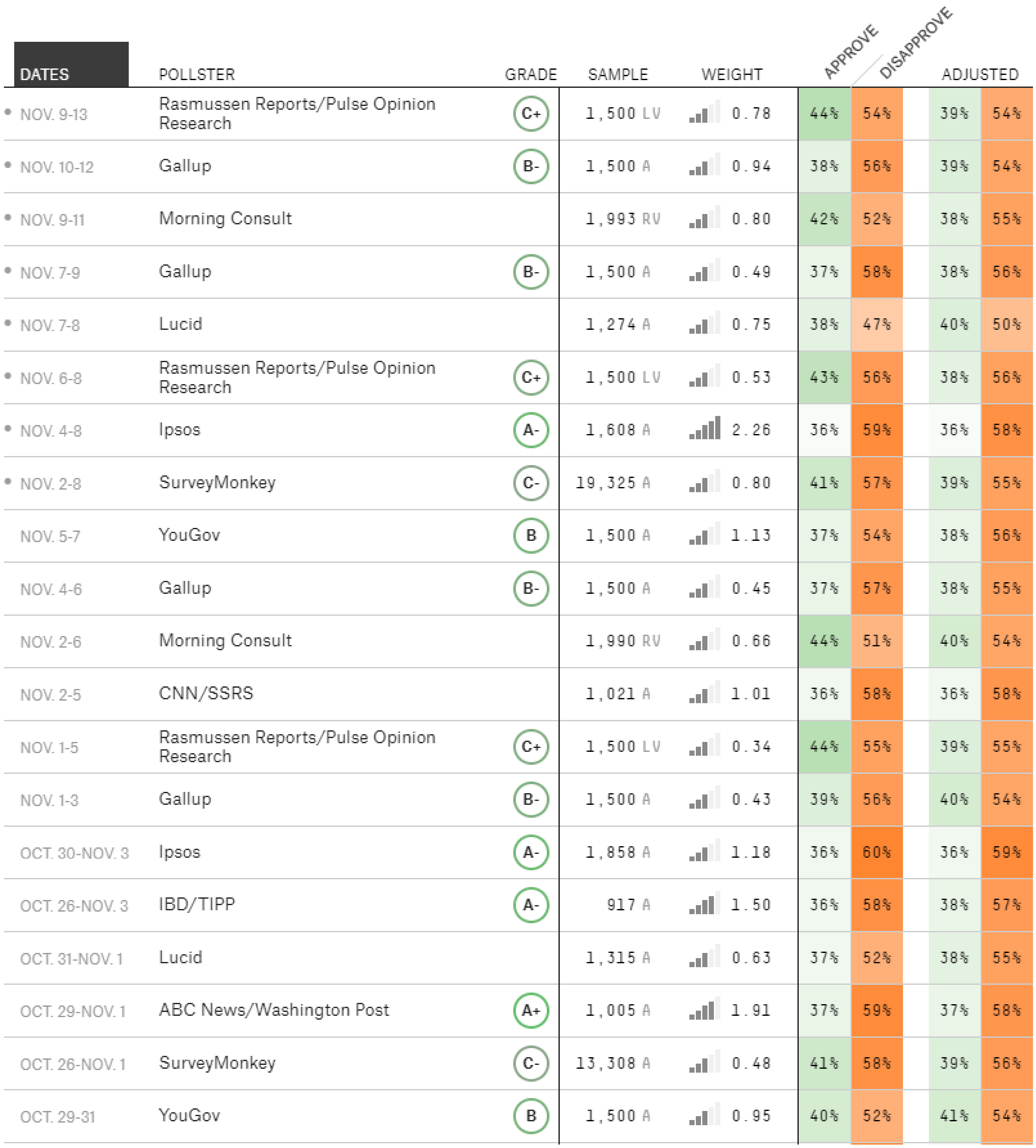
Although Donald Trump is extremely active on Twitter, there are many other factors that need to be considered when discussing his approval rating. Some of these factors include his public behavior, his responses to criticism and things he has done as president. Another thing to consider is that Trump's behavior one day may cause his approval rate to drop the following day, however it may be a few days until Trump decides to tweet about the issue. One final external factor that most likely accounted for the inaccuracy of the model is the behavior of other political figures. If other presidential candidates have a positive light shone on them in the media, this could cause the approval rating to drop for Trump. In addition, if someone on Trump's team is caught in a scandal, or some other negative behavior, this would also likely cause the approval rating to drop.

Despite the relatively low accuracy of the models when predicting the approval rating, the k-means clusters gave much more insight. One hypothesis based on the word cloud was that words with similar frequencies were often tweeted together in the same tweets. For example, "news" and "fake", as well as "obamacare" and "healthcare". The k-mean clusters confirmed this hypothesis, because the aforementioned word groups were clustered together. These clusters also gave other insights, because the various clusters contain keywords that are often seen in media headlines such as "Trump Urges Russia to find Clinton's Missing Emails" from CNN.

A much more accurate model could be created, however it would require a more experienced team to dedicate lots of time to developing a model with more features, which is outside the scope of this course.

Appendices

Appendix 1: Sample Pollster Approval Rates



DATES	POLLSTER	GRADE	SAMPLE	WEIGHT	APPROVE	DISAPPROVE	ADJUSTED	
• NOV. 9-13	Rasmussen Reports/Pulse Opinion Research	C+	1,500 LV	0.78	44%	54%	39%	54%
• NOV. 10-12	Gallup	B-	1,500 A	0.94	38%	56%	39%	54%
• NOV. 9-11	Morning Consult		1,993 RV	0.80	42%	52%	38%	55%
• NOV. 7-9	Gallup	B-	1,500 A	0.49	37%	58%	38%	56%
• NOV. 7-8	Lucid		1,274 A	0.75	38%	47%	40%	50%
• NOV. 6-8	Rasmussen Reports/Pulse Opinion Research	C+	1,500 LV	0.53	43%	56%	38%	56%
• NOV. 4-8	Ipsos	A-	1,608 A	2.26	36%	59%	36%	58%
• NOV. 2-8	SurveyMonkey	C-	19,325 A	0.80	41%	57%	39%	55%
NOV. 5-7	YouGov	B	1,500 A	1.13	37%	54%	38%	56%
NOV. 4-6	Gallup	B-	1,500 A	0.45	37%	57%	38%	55%
NOV. 2-6	Morning Consult		1,990 RV	0.66	44%	51%	40%	54%
NOV. 2-5	CNN/SSRS		1,021 A	1.01	36%	58%	36%	58%
NOV. 1-5	Rasmussen Reports/Pulse Opinion Research	C+	1,500 LV	0.34	44%	55%	39%	55%
NOV. 1-3	Gallup	B-	1,500 A	0.43	39%	56%	40%	54%
OCT. 30-NOV. 3	Ipsos	A-	1,858 A	1.18	36%	60%	36%	59%
OCT. 26-NOV. 3	IBD/TIPP	A-	917 A	1.50	36%	58%	38%	57%
OCT. 31-NOV. 1	Lucid		1,315 A	0.63	37%	52%	38%	55%
OCT. 29-NOV. 1	ABC News/Washington Post	A+	1,005 A	1.91	37%	59%	37%	58%
OCT. 26-NOV. 1	SurveyMonkey	C-	13,308 A	0.48	41%	58%	39%	56%
OCT. 29-31	YouGov	B	1,500 A	0.95	40%	52%	41%	54%

Appendix 2: Code to Crawl Trump's Tweets

Twitter.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 29 15:56:44 2017

@author: xinruishao
"""

def isIn(tweets,tweet):
  for i in reversed(tweets):
    if i == tweet:
      return True
    elif len(i)>7 and len(tweet) >8:
      if ":" in tweet[6] and tweet[5]==i[5]:
       return True
  return False
def transformRetweet(tweet):
  start=tweet.index("More")
  print(start)
  temp=tweet[0:start+1]
  end=tweet.index("Reply")
  print(end)
  middle=" ".join(tweet[start+1:end])
  temp.append(middle)
  temp=temp+tweet[end:]
  return temp
import time
import csv
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
browser=webdriver.Chrome('/Users/xinruishao/Desktop/chromedriver')
url='https://twitter.com/realDonaldTrump?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Ea
uthor'
browser.get(url)
time.sleep(1)
body=browser.find_element_by_tag_name('body')
myData=[]
for _ in range(100000):
```

```
body.send_keys(Keys.PAGE_DOWN)
tweets=browser.find_elements_by_xpath("//div[@class='content']")
for tweet in tweets:
  temp=[]
  #time=tweet.find_element_by_xpath("//small[@class='time']/span")
  gg=tweet.text.splitlines()
  if 'Jan 21' in gg:
   with open('tweets.csv', 'w', newline='') as csvfile:
    spamwriter = csv.writer(csvfile, delimiter=';')
    for tweet in myData:
      if len(tweet) > 12 :
        tweet=transformRetweet(tweet)
      #print(tweet)
      if len(tweet)==12:
        spamwriter.writerow(tweet)

    exit()
  if  len(gg) > 9:
    if not isIn(myData,gg):
     myData.append(gg)
  #temp= tweet.text.splitlines()

time.sleep(0.2)
```

Appendix 3: Raw Tweets

```
0    We had a wonderful visit to Vietnam thank you ...    11-12-2017  11:20:20
1      Just landed in the Philippines after a great d...  11-12-2017  10:21:06
2      Just landed in the Philippines after a great d...  11-12-2017  10:06:58
3    Why would Kim Jong-un insult me by calling me ...    11-12-2017  00:48:01
4    Does the Fake News Media remember when Crooked...    11-12-2017  00:43:36
5     When will all the haters and fools out there r...   11-12-2017  00:18:32
6    Met with President Putin of Russia who was at ...    11-12-2017  00:16:05
7     Will be doing a joint press conference in Hano...   11-11-2017  23:35:39
8     President Xi of China has stated that he is up...   11-11-2017  23:32:25
9       “Presidential Proclamation Commemorating the 5... 11-11-2017  12:35:46
10   On this wonderful Veterans Day I want to expre...    11-11-2017  12:11:53
11   On behalf of an entire nation Happy 242nd Birt...    11-10-2017  12:57:23
12   The United States has been reminded time and a...    11-10-2017  10:43:48
13   The United States has been reminded time and a...    11-10-2017  10:31:32
```

```
14      The United States is prepared to work with eac...        11-10-2017  10:26:22
15      Today I am here to offer a renewed partnership...        11-10-2017  10:06:37
16      In more and more places throughout this region...        11-10-2017  09:50:25
17      Throughout my travels I've had the pleasure of...        11-10-2017  09:32:07
18      Just landed in Da Nang Vietnam to deliver a sp...        11-10-2017  05:45:36
19      I am leaving China for #APEC2017 in Vietnam. @...        11-10-2017  01:17:15
20      My meetings with President Xi Jinping were ver...        11-09-2017  23:44:17
21      I don’t blame China I blame the incompetence o...        11-09-2017  23:39:56
22      In the coming months and years ahead I look fo...        11-09-2017  13:58:32
23      President Xi thank you for such an incredible ...        11-09-2017  07:08:33
24      Congratulations to all of the ”DEPLORABLES” an...        11-08-2017  18:17:41
25      Looking forward to a full day of meetings with...        11-08-2017  15:40:40
26      NoKo has interpreted America's past restraint ...        11-08-2017  15:15:08
27      On behalf of @FLOTUS Melania and I THANK YOU f...        11-08-2017  14:27:18
28      Leaving South Korea now heading to China. Look...        11-08-2017  04:57:40
29      Together we dream of a Korea that is free a pe...        11-08-2017  03:43:59
...                                                   ...                     ...
1811    The #MarchForLife is so important. To all of y...        01-27-2017  16:27:02
1812    Mexico has taken advantage of the U.S. for lon...        01-27-2017  13:19:10
1813    Look forward to seeing final results of VoteSt...        01-27-2017  13:12:52
1814    Miami-Dade Mayor drops sanctuary policy. Right...        01-26-2017  23:53:37
1815    Will be interviewed by @SeanHannity on @FoxNew...        01-26-2017  23:45:28
1816    Spoke at the Congressional @GOP Retreat in Phi...        01-26-2017  19:21:17
1817    Spoke at the Congressional @GOP Retreat in Phi...        01-26-2017  19:15:44
1818    of jobs and companies lost. If Mexico is unwil...        01-26-2017  13:55:03
1819    The U.S. has a 60 billion dollar trade deficit...        01-26-2017  13:51:46
1820    Ungrateful TRAITOR Chelsea Manning who should ...        01-26-2017  11:04:24
1821    Interview with David Muir of @ABC News in 10 m...        01-26-2017  02:48:25
1822    As your President I have no higher duty than t...        01-26-2017  02:14:56
1823    Beginning today the United States of America g...        01-26-2017  00:03:33
1824    I will be interviewed by @DavidMuir tonight at...        01-25-2017  22:05:59
1825    I will be making my Supreme Court pick on Thur...        01-25-2017  12:17:01
1826    even those registered to vote who are dead (an...        01-25-2017  12:13:46
1827    I will be asking for a major investigation int...        01-25-2017  12:10:01
1828    Big day planned on NATIONAL SECURITY tomorrow....        01-25-2017  02:37:48
1829    If Chicago doesn't fix the horrible "carnage" ...        01-25-2017  02:25:40
1830    Congratulations to @FoxNews for being number o...        01-25-2017  02:16:19
1831    Great meeting with Ford CEO Mark Fields and Ge...        01-25-2017  00:46:57
1832    Signing orders to move forward with the constr...        01-24-2017  17:49:17
1833    Great meeting with automobile industry leaders...        01-24-2017  17:04:01
1834    A photo delivered yesterday that will be displ...        01-24-2017  16:58:06
```

1835	Will be meeting at 9:00 with top automobile ex...	01-24-2017	11:11:47
1836	Busy week planned with a heavy focus on jobs a...	01-23-2017	11:38:16
1837	Peaceful protests are a hallmark of our democr...	01-22-2017	14:23:17
1838	Wow television ratings just out: 31 million pe...	01-22-2017	12:51:36
1839	Watched protests yesterday but was under the i...	01-22-2017	12:47:21
1840	Had a great meeting at CIA Headquarters yester...	01-22-2017	12:35:09

Appendix 4: Count of Likes, Retweets, and Retweeted Contents

	9976	55423	false
0	10019	51862	False
1	10245	66298	False
2	869	3751	False
3	256702	570776	False
4	27263	111078	False
5	38073	150707	False
6	14756	70386	False
7	9227	54925	False
8	15649	73286	False
9	11968	54332	False
10	30538	114821	False
11	24590	99014	False
12	13542	55845	False
13	497	1700	False
14	10938	47280	False
15	9231	41687	False
16	8483	38055	False
17	11147	50495	False
18	7699	50225	False
19	13421	70697	False
20	14760	80875	False
21	28977	132588	False
22	20125	92351	False
23	19215	90689	False
24	35089	132327	False
25	13384	62972	False
26	23422	87511	False
27	12938	60171	False
28	11359	71775	False
29	12809	52820	False

```
...                         ...                 ...                 ...
1811                        45457        184954                  False
1812                        30604        160660                  False
1813                     17319              88640                False
1814                        24835        112620                  False
1815                     10658              73559                False
1816                     11365              72438                False
1817                            9                         9          False
1818                        28106        115631                  False
1819                        25881        106893                  False
1820                        28033        128497                  False
1821                          6517              53811            False
1822                        29219        152335                  False
1823                        23095        109885                  False
1824                     12454              75988                False
1825                        21700        132199                  False
1826                        18914        107227                  False
1827                        26155        129979                  False
1828                        53949        193872                  False
1829                        56038        209363                  False
1830                        32107        162721                  False
1831                        18697        101012                  False
1832                        25694        131943                  False
1833                     16331              99607                False
1834                        20939        109406                  False
1835                        23743        154857                  False
1836                        26750        177839                  False
1837                        81527        390826                  False
1838                        40085        217610                  False
1839                        45718        213295                  False
1840                        16906        127677                  False

[1841 rows x 5 columns]>
```

Appendix 5: Code to Clean the Data

Trump.ipynb

```
import nltk
import pandas as pd
import re as regex
from collections import Counter
nltk.download()


class TwitterData_Initialize():
  data = []
  processed_data = []
  wordlist = []

  data_model = None
  data_labels = None
  is_testing = False

  def initialize(self, csv_file, is_testing_set=False, from_cached=None):
    if from_cached is not None:
      self.data_model = pd.read_csv(from_cached)
      return

    self.is_testing = is_testing_set

    if not is_testing_set:
      self.data = pd.read_csv(csv_file, header=0,encoding = "ISO-8859-1")

    else:
      self.data = pd.read_csv(csv_file, header=0, names=["id", "text"],dtype={"id":"int64","text":"str"},nrows=4000)
      not_null_text = 1 ^ pd.isnull(self.data["text"])
      not_null_id = 1 ^ pd.isnull(self.data["id"])
      self.data = self.data.loc[not_null_id & not_null_text, :]

    self.processed_data = self.data
    self.wordlist = []
    self.data_model = None
    self.data_labels = None

data = TwitterData_Initialize()
data.initialize("trump.csv")
data.processed_data.head(5)
```

	text	created_at	retweet_count	favorite_count	is_retweet
0	Thank you! https://t.co/TD0rYcWN8C	11-12-2017 14:29:22	9976	55423	False
1	We had a wonderful visit to Vietnam thank you ...	11-12-2017 11:20:20	10019	51862	False
2	Just landed in the Philippines after a great d...	11-12-2017 10:21:06	10245	66298	False
3	Just landed in the Philippines after a great d...	11-12-2017 10:06:58	869	3751	False
4	Why would Kim Jong-un insult me by calling me ...	11-12-2017 00:48:01	256702	570776	False

```python
class TwitterCleanuper:
  def iterate(self):
    for cleanup_method in [self.remove_urls,
                  self.remove_usernames,
                  self.remove_na,
                  self.remove_special_chars,
                  self.remove_numbers]:
      yield cleanup_method

  @staticmethod
  def remove_by_regex(tweets, regexp):
    tweets.loc[:, "text"].replace(regexp, "", inplace=True)
    return tweets

  def remove_urls(self, tweets):
    return TwitterCleanuper.remove_by_regex(tweets, regex.compile(r"http.?://[^\s]+[\s]?"))

  def remove_na(self, tweets):
    return tweets[tweets["text"] != "Not Available"]

  def remove_special_chars(self, tweets): # it unrolls the hashtags to normal words
    for remove in map(lambda r: regex.compile(regex.escape(r)), [",", ":", "\"", "=", "&", ";", "%", "$",
                                  "@", "%", "^", "*", "(", ")", "{", "}",
                                  "[", "]", "|", "/", "\\", ">", "<", "-",
                                  "!", "?", ".", "'",
                                  "--", "---", "#"]):
```

```
            tweets.loc[:, "text"].replace(remove, "", inplace=True)
        return tweets

    def remove_usernames(self, tweets):
        return TwitterCleanuper.remove_by_regex(tweets, regex.compile(r"@[^\s]+[\s]?"))

    def remove_numbers(self, tweets):
        return TwitterCleanuper.remove_by_regex(tweets, regex.compile(r"\s?[0-9]+\.?[0-9]*"))

class TwitterData_Cleansing(TwitterData_Initialize):
    def __init__(self, previous):
        self.processed_data = previous.processed_data

    def cleanup(self, cleanuper):
        t = self.processed_data
        for cleanup_method in cleanuper.iterate():
            if not self.is_testing:
                t = cleanup_method(t)
            else:
                if cleanup_method.__name__ != "remove_na":
                    t = cleanup_method(t)

        self.processed_data = t

data = TwitterData_Cleansing(data)
data.cleanup(TwitterCleanuper())
data.processed_data.head(5)
```

	text	created_at	retweet_count	favorite_count	is_retweet
0	Thank you	11-12-2017 14:29:22	9976	55423	False
1	We had a wonderful visit to Vietnam thank you ...	11-12-2017 11:20:20	10019	51862	False
2	Just landed in the Philippines after a great d...	11-12-2017 10:21:06	10245	66298	False
3	Just landed in the Philippines after a great d...	11-12-2017 10:06:58	869	3751	False

4	Why would Kim Jongun insult me by calling me o...	11-12-2017 00:48:01	256702	570776	False

```
class TwitterData_TokenStem(TwitterData_Cleansing):
  def __init__(self, previous):
    self.processed_data = previous.processed_data

  def stem(self, stemmer=nltk.PorterStemmer()):
    def stem_and_join(row):
      #row["text"] = list(map(lambda str: stemmer.stem(str.lower()), row["text"]))
      row["text"] = list(map(lambda str: str.lower(), row["text"]))
      return row

    self.processed_data = self.processed_data.apply(stem_and_join, axis=1)

  def tokenize(self, tokenizer=nltk.word_tokenize):
    def tokenize_row(row):
      row["text"] = tokenizer(row["text"])
      row["tokenized_text"] = [] + row["text"]
      return row

    self.processed_data = self.processed_data.apply(tokenize_row, axis=1)

data = TwitterData_TokenStem(data)
data.tokenize()
data.stem()
data.processed_data.head(5)
```

	text	**created_at**	**retweet_count**	**favorite_count**	**is_retweet**	**tokenized_text**
0	[thank, you]	11-12-2017 14:29:22	9976	55423	False	[Thank, you]
1	[we, had, a, wonderful, visit, to, vietnam, th...	11-12-2017 11:20:20	10019	51862	False	[We, had, a, wonderful, visit, to, Vietnam, th...
2	[just, landed, in, the, philippines, after, a,...	11-12-2017 10:21:06	10245	66298	False	[Just, landed, in, the, Philippines, after, a,...

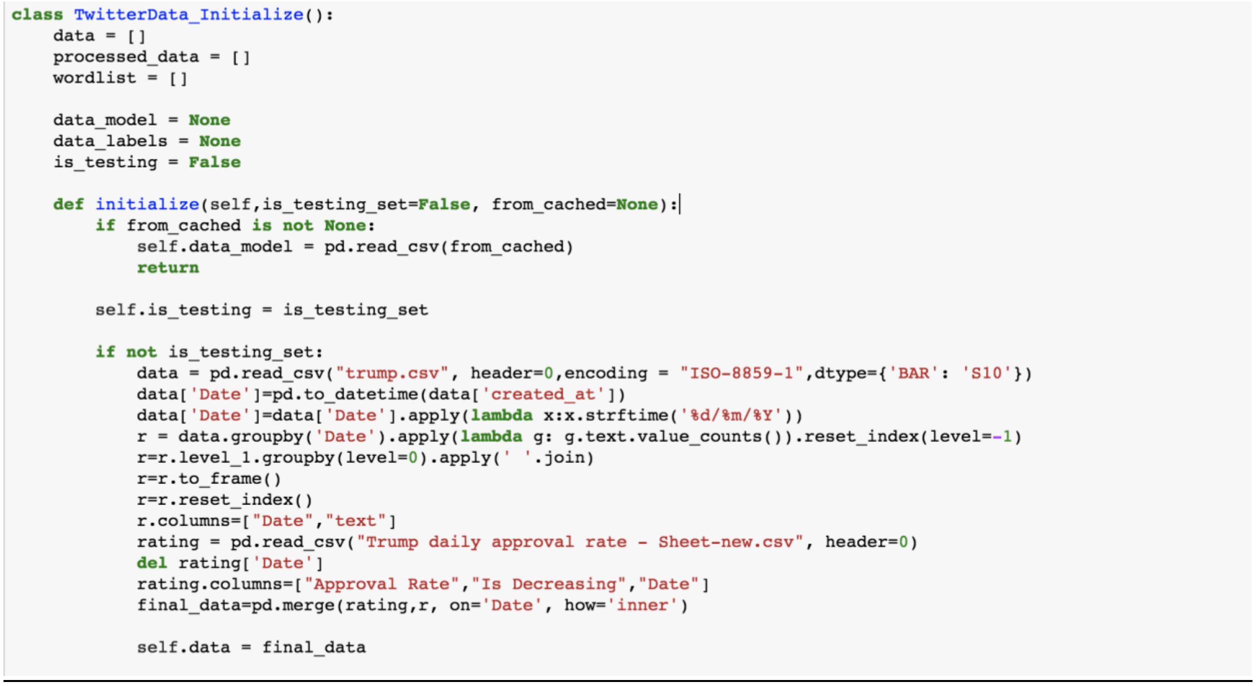
3	[just, landed, in, the, philippines, after, a,...	11-12-2017 10:06:58	869	3751	False	[Just, landed, in, the, Philippines, after, a,...
4	[why, would, kim, jongun, insult, me, by, call...	11-12-2017 00:48:01	256702	570776	False	[Why, would, Kim, Jongun, insult, me, by, call...

Appendix 6: Word Frequency Counter

```
words = Counter()
for idx in data.processed_data.index:
    words.update(data.processed_data.loc[idx, "text"])

stopwords=nltk.corpus.stopwords.words("english")
stopwords = stopwords +['should','would','get','do','will','today','amp']
whitelist = []
for idx, stop_word in enumerate(stopwords):
    if stop_word not in whitelist:
        del words[stop_word]
words.most_common(30)
```

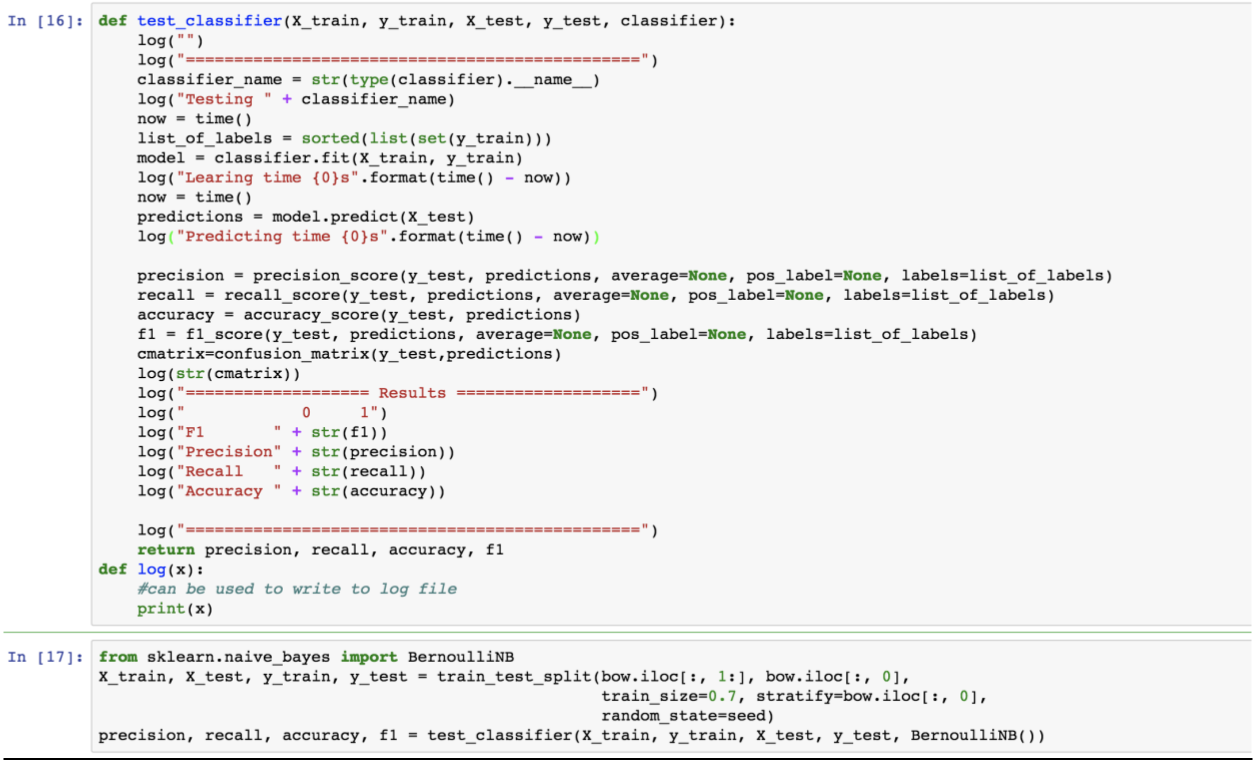
```
[('great', 380),
 ('people', 152),
 ('news', 147),
 ('fake', 140),
 ('us', 136),
 ('thank', 121),
 ('big', 117),
 ('america', 105),
 ('president', 96),
 ('country', 96),
 ('jobs', 93),
 ('american', 81),
 ('many', 81),
 ('media', 78),
 ('tax', 78),
 ('healthcare', 71),
 ('time', 70),
 ('much', 69),
 ('new',                 67),
 ('honor',               65),
 ('years',               64),
 ('obamacare',           64),
 ('democrats',           63),
 ('must',                62),
 ('election',            61),
 ('first',               60),
 ('trump',               60),
 ('russia',              58),
 ('make',                58),
 ('working',             58)]
```

Appendix 7: Code for feature transformation



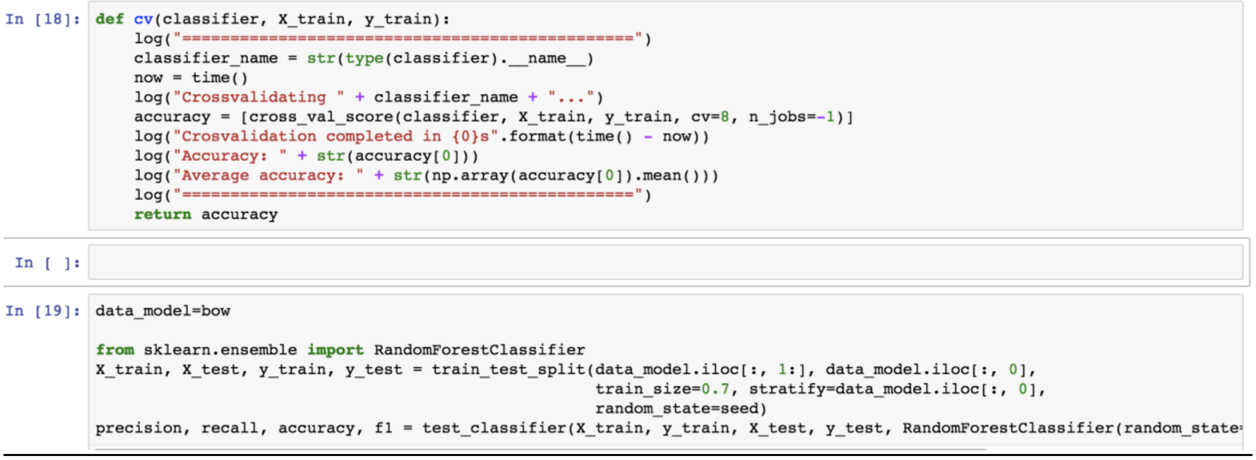
```python
class TwitterData_Initialize():
    data = []
    processed_data = []
    wordlist = []

    data_model = None
    data_labels = None
    is_testing = False

    def initialize(self,is_testing_set=False, from_cached=None):
        if from_cached is not None:
            self.data_model = pd.read_csv(from_cached)
            return

        self.is_testing = is_testing_set

        if not is_testing_set:
            data = pd.read_csv("trump.csv", header=0,encoding = "ISO-8859-1",dtype={'BAR': 'S10'})
            data['Date']=pd.to_datetime(data['created_at'])
            data['Date']=data['Date'].apply(lambda x:x.strftime('%d/%m/%Y'))
            r = data.groupby('Date').apply(lambda g: g.text.value_counts()).reset_index(level=-1)
            r=r.level_1.groupby(level=0).apply(' '.join)
            r=r.to_frame()
            r=r.reset_index()
            r.columns=["Date","text"]
            rating = pd.read_csv("Trump daily approval rate - Sheet-new.csv", header=0)
            del rating['Date']
            rating.columns=["Approval Rate","Is Decreasing","Date"]
            final_data=pd.merge(rating,r, on='Date', how='inner')

            self.data = final_data
```

Appendix 8: Code for Naive Bayes Model



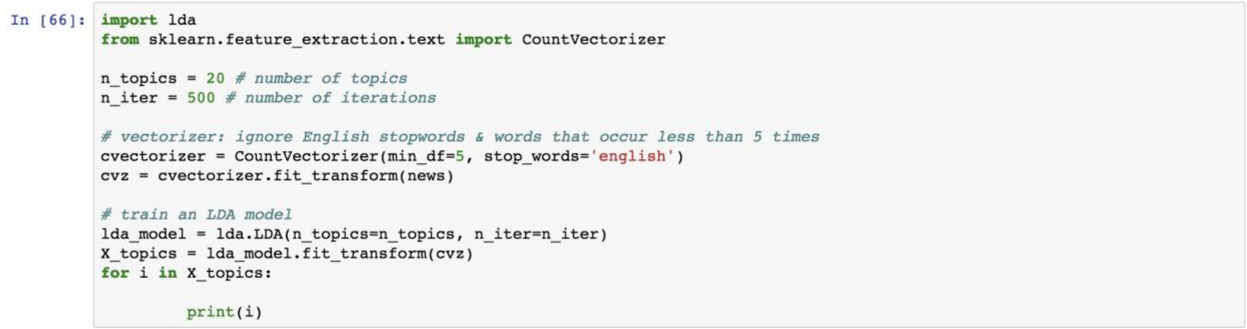
```
In [16]: def test_classifier(X_train, y_train, X_test, y_test, classifier):
             log("")
             log("===============================================")
             classifier_name = str(type(classifier).__name__)
             log("Testing " + classifier_name)
             now = time()
             list_of_labels = sorted(list(set(y_train)))
             model = classifier.fit(X_train, y_train)
             log("Learing time {0}s".format(time() - now))
             now = time()
             predictions = model.predict(X_test)
             log("Predicting time {0}s".format(time() - now))

             precision = precision_score(y_test, predictions, average=None, pos_label=None, labels=list_of_labels)
             recall = recall_score(y_test, predictions, average=None, pos_label=None, labels=list_of_labels)
             accuracy = accuracy_score(y_test, predictions)
             f1 = f1_score(y_test, predictions, average=None, pos_label=None, labels=list_of_labels)
             cmatrix=confusion_matrix(y_test,predictions)
             log(str(cmatrix))
             log("=================== Results ===================")
             log("            0     1")
             log("F1       " + str(f1))
             log("Precision" + str(precision))
             log("Recall   " + str(recall))
             log("Accuracy " + str(accuracy))

             log("===============================================")
             return precision, recall, accuracy, f1
         def log(x):
             #can be used to write to log file
             print(x)

In [17]: from sklearn.naive_bayes import BernoulliNB
         X_train, X_test, y_train, y_test = train_test_split(bow.iloc[:, 1:], bow.iloc[:, 0],
                                                             train_size=0.7, stratify=bow.iloc[:, 0],
                                                             random_state=seed)
         precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, BernoulliNB())
```

Appendix 9: Code for Random Forest Model



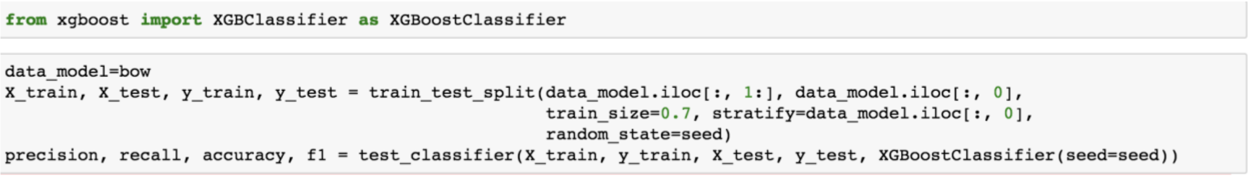
```
In [18]: def cv(classifier, X_train, y_train):
             log("===============================================")
             classifier_name = str(type(classifier).__name__)
             now = time()
             log("Crossvalidating " + classifier_name + "...")
             accuracy = [cross_val_score(classifier, X_train, y_train, cv=8, n_jobs=-1)]
             log("Crosvalidation completed in {0}s".format(time() - now))
             log("Accuracy: " + str(accuracy[0]))
             log("Average accuracy: " + str(np.array(accuracy[0]).mean()))
             log("===============================================")
             return accuracy

In [ ]: 

In [19]: data_model=bow

         from sklearn.ensemble import RandomForestClassifier
         X_train, X_test, y_train, y_test = train_test_split(data_model.iloc[:, 1:], data_model.iloc[:, 0],
                                                             train_size=0.7, stratify=data_model.iloc[:, 0],
                                                             random_state=seed)
         precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, RandomForestClassifier(random_state
```

Appendix 10: Code for K-Mean Clustering Model



```
In [66]: import lda
         from sklearn.feature_extraction.text import CountVectorizer

         n_topics = 20 # number of topics
         n_iter = 500 # number of iterations

         # vectorizer: ignore English stopwords & words that occur less than 5 times
         cvectorizer = CountVectorizer(min_df=5, stop_words='english')
         cvz = cvectorizer.fit_transform(news)

         # train an LDA model
         lda_model = lda.LDA(n_topics=n_topics, n_iter=n_iter)
         X_topics = lda_model.fit_transform(cvz)
         for i in X_topics:

                 print(i)
```

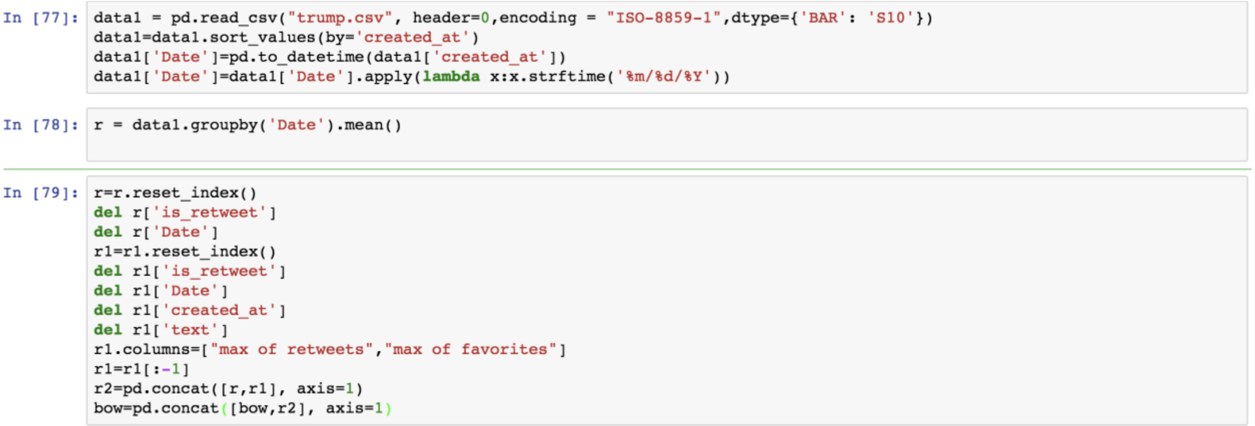
Appendix 11: Code for XG Boost Model



```
from xgboost import XGBClassifier as XGBoostClassifier
```

```
data_model=bow
X_train, X_test, y_train, y_test = train_test_split(data_model.iloc[:, 1:], data_model.iloc[:, 0],
                                                    train_size=0.7, stratify=data_model.iloc[:, 0],
                                                    random_state=seed)
precision, recall, accuracy, f1 = test_classifier(X_train, y_train, X_test, y_test, XGBoostClassifier(seed=seed))
```

Appendix 12: Code for adding additional features

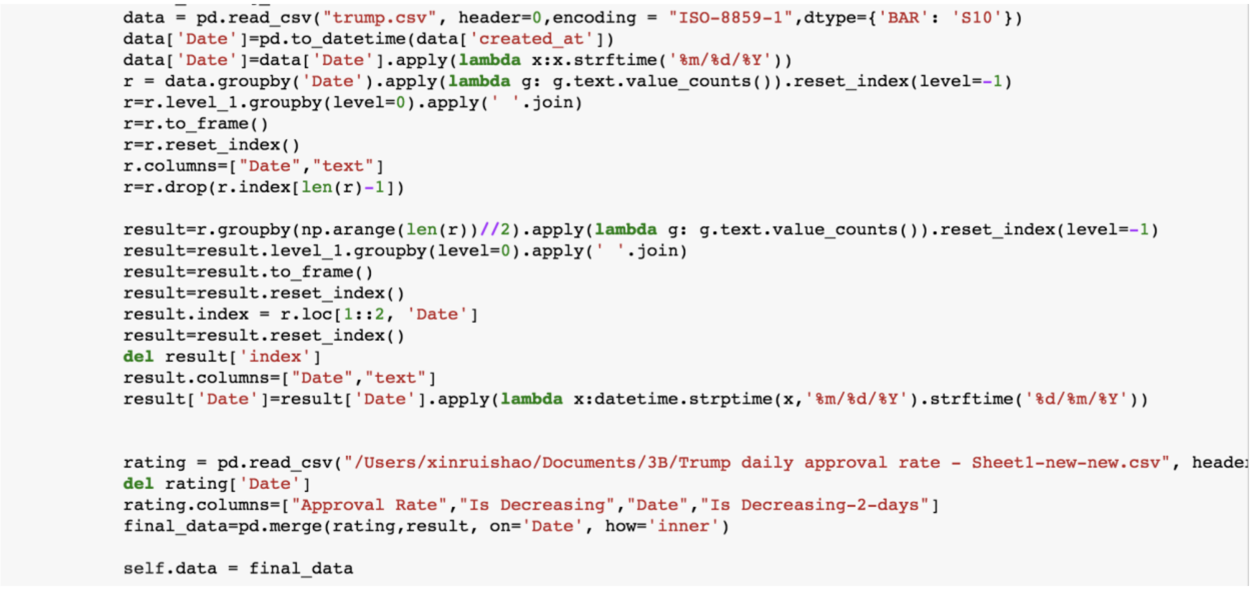


```
In [77]: data1 = pd.read_csv("trump.csv", header=0,encoding = "ISO-8859-1",dtype={'BAR': 'S10'})
         data1=data1.sort_values(by='created_at')
         data1['Date']=pd.to_datetime(data1['created_at'])
         data1['Date']=data1['Date'].apply(lambda x:x.strftime('%m/%d/%Y'))
```

```
In [78]: r = data1.groupby('Date').mean()
```

```
In [79]: r=r.reset_index()
         del r['is_retweet']
         del r['Date']
         r1=r1.reset_index()
         del r1['is_retweet']
         del r1['Date']
         del r1['created_at']
         del r1['text']
         r1.columns=["max of retweets","max of favorites"]
         r1=r1[:-1]
         r2=pd.concat([r,r1], axis=1)
         bow=pd.concat([bow,r2], axis=1)
```

Appendix 13: Code for merging tweets in consecutive dates



```
data = pd.read_csv("trump.csv", header=0,encoding = "ISO-8859-1",dtype={'BAR': 'S10'})
data['Date']=pd.to_datetime(data['created_at'])
data['Date']=data['Date'].apply(lambda x:x.strftime('%m/%d/%Y'))
r = data.groupby('Date').apply(lambda g: g.text.value_counts()).reset_index(level=-1)
r=r.level_1.groupby(level=0).apply(' '.join)
r=r.to_frame()
r=r.reset_index()
r.columns=["Date","text"]
r=r.drop(r.index[len(r)-1])

result=r.groupby(np.arange(len(r))//2).apply(lambda g: g.text.value_counts()).reset_index(level=-1)
result=result.level_1.groupby(level=0).apply(' '.join)
result=result.to_frame()
result=result.reset_index()
result.index = r.loc[1::2, 'Date']
result=result.reset_index()
del result['index']
result.columns=["Date","text"]
result['Date']=result['Date'].apply(lambda x:datetime.strptime(x,'%m/%d/%Y').strftime('%d/%m/%Y'))


rating = pd.read_csv("/Users/xinruishao/Documents/3B/Trump daily approval rate - Sheet1-new-new.csv", heade
del rating['Date']
rating.columns=["Approval Rate","Is Decreasing","Date","Is Decreasing-2-days"]
final_data=pd.merge(rating,result, on='Date', how='inner')

self.data = final_data
```

References

Polls. (n.d.). Retrieved September 28, 2017, from https://www.realclearpolitics.com/epolls/other/president_trump_job_approval-6179.html

Silver, N. (2017, March 02). How We're Tracking Donald Trump's Approval Ratings. Retrieved September 27, 2017, from http://fivethirtyeight.com/features/how-were-tracking-donald-trumps-approval-ratings/

Wang, Yu, et al. "Will Sanders Supporters Jump Ship for Trump? Fine-Grained Analysis of Twitter Followers." Will Sanders Supporters Jump Ship for Trump? Fine-Grained Analysis of Twitter Followers, 31 May 2016, doi:arXiv:1605.09473.