CS-6375.003: Machine Learning – F18 Term Project

# Predictive Analysis of Trending Topics in Twitter

Sreyas Reddy Dadi (sxd180049), Akhil Bitra (axb170061), Aniket Pathak (adp170003)

---

## 1. Introduction:

The most common, traditional way to collect people's opinions is random sampling and asking survey questions by phone. If a News media is interested in knowing how popular Trump is, it would call up a thousand people and ask how they would rate Trump. The data collected in this manner is considered a gold standard. However, this approach bears shortcomings such as costs, limited targeted people, etc.

In this project, we attempt to predict people's opinions based on trends in social media such as twitter. There are millions of people twitting their opinion on various social, economic and political topics daily. We could use these opinions as the survey data and predict the trend. The very fact that many reactions collected on the social media regarding this topic has brought about the need to bifurcate the reactions based on their sentiments. Collecting polls by asking people in person or by phone is costly, but if we could get the same results from the freely available Web data, it could supplement or supplant the conventional way of collecting polls.

## 2. Problem Description and Algorithm:

To summarize the problem, we will attempt to conduct sentiment analysis on "tweets" using various different machine learning algorithms. We attempt to classify the polarity of the tweet where it is either positive or negative. If the tweet has both positive and negative elements, the more dominant sentiment should be picked as the final label.

We use the real time data which can be crawled from twitter using tweepy library and label them positive/negative. The data comes with emoticons, usernames and hashtags which are required to be processed and converted into a standard form. After pre-processing these

tweets we divide data into training data, testing data and validation data and train machine learning algorithms and test these trained models to obtain various evaluation metrics like accuracy, confusion matrix, precision, f-score, area-under-curve from roc curve.

# 3. Data:

## 3.1. Data Collection:

To access the data on the twitter we firstly start by creating a developer account with twitter and generate the consumer key, consumer secret, access token, access secret to authenticate and scrap data from twitter. We then use tweepy API to authenticate and start crawling tweets. We use various search topics with little variations to search tweets like "#Midterm", "#2018Midterm", "#MidtermElection" etc. While retrieving we remove duplicates tweets by storing them in dictionary and checking if tweet is already present before inserting into file. We retrieve tweets in English only for our purpose. Upon running the program, we got 1500 tweets and retweets. We store these tweets in a .txt file to further use it for data cleaning. Few sample tweets retrieved are shown below:

```
{'text': 'RT @acrossthemargin: A terse, but impactful, story about the
first free election in South Africa, 25 years ago next April. "A New
Dawn" by…'}
{'text': 'A terse, but impactful, story about the first free election in
South Africa, 25 years ago next April. "A New Dawn"… https://t.co/
CUuilrkxXp'}
{'text': 'The ballots have been counted! After our cutthroat election at
the #NCSS conference, the clear winner was chocolate… https://t.co/
7I7uFBCg3L'}
```

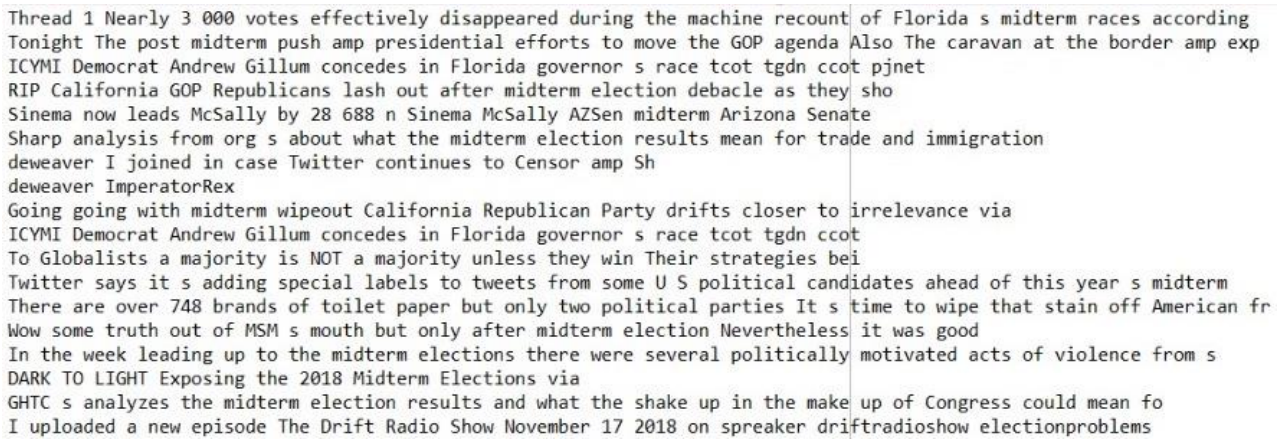
## 3.2. Data Cleaning and Pre-Processing:

Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as re- tweets, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. We apply the following pre-processing steps using a regular expression:

- Remove the header of each tweet.
- Strip spaces and quotes (" and ') from the ends of tweet.

- remove user names starting with @ symbols
- remove the hashtags starting with # symbols.
- Remove two or move spaces and replace with a single space.
- Remove emoticons, special characters and punctuation marks.

We the append each normalized tweet into a .txt file separated by a '\n' character for training various machine learning models. A sample cleaned data is shown below:




# 4. Feature Extraction and Labelling:

We label the retrieved tweets using TextBlob from textblob to find the polarity of the tweet and label the tweets into three classes 0,-1,+1 if the polarity is equal to 0, negative and positive respectively. We label each tweet as defined above and store the labels.

We use CountVectorizer from sklearn.feature_extraction.text to extract features from the read tweets which takes each word as a token with spaces as token separators and counts token occurrences to produce a feature . we cap the number of features generated to 1500.

We then divide the obtained data into training data, test data using train_test_split function from sklearn.model_selection.

# 5. Classifiers:

We use various different algorithms to evaluate their performance for our classification.

## 5.1. Naive Bayes:

Naive Bayes is a simple model which can be used for text classification. In this model, the class cˆ is assigned to a tweet t, where

$$\hat{c} = \underset{c}{argmax}\ \mathrm{P}(\mathrm{c}|\mathrm{t})$$

$$\mathrm{P}(\mathrm{c}|\mathrm{t}) \propto \mathrm{P}(\mathrm{c}) \prod_{i=1}^{n} \mathrm{P}(f_i|\mathrm{c})$$

In the formula above, fi represents the i-th feature of total n features. P(c) and P($f_i$|c) can be obtained through maximum likelihood estimates.

## 5.2. Decision Tree

Decision trees are a classifier model in which each node of the tree represents a test on the attribute of the data set, and its children represent the outcomes. The leaf nodes represents the final classes of the data points. It is a supervised classifier model which uses data with known labels to form the decision tree and then the model is applied on the test data. For each node in the tree the best test condition or decision has to be taken. We use the GINI factor to decide the best split. For a given node t, $GINI(t)=1-\Sigma_j[p(j|t)]^2$, where p(j|t) is the relative frequency of class j at node t, and $GINI_{split}=\Sigma_{i=1}^{k} n_i/n\ GINI(i)$ ($n_i$ = number of records at child i, n = number of records at p)indicates the quality of the split. We choose a split that minimizes the GINI factor.

## 5.3. SVM:

SVM, also known as support vector machines, is a non-probabilistic binary linear classifier. For a training set of points ($x_i$,$y_i$) where x is the feature vector and y is the class, we want to find the maximum-margin hyperplane that divides the points with $y_i$ = 1 and $y_i$=−1. The equation of the hyperplane is as follow

$$w \cdot x - b = 0$$

We want to maximize the margin, denoted by γ, as follows

$$\max_{w,\gamma} \gamma, s.t. \forall i, \gamma \leq y_i(w \cdot x_i + b)$$

in order to separate the points well.

## 5.4. Logistic Regression:

Logistic regression is a more performant algorithm used in classification problems. The most important feature is its ability to produce a sort of hard-limited hypothesis function:

$$0 \leq h_\theta(x) \leq 1$$

The hypothesis function is slightly different from the one used in linear regression. For logistic regression,

$$h_\theta(x) = g(\theta^T x)$$

which is the traditional hypothesis function processed by a new function g, defined as:

$$g(z) = 1/1+e^{-z}$$

## 5.5. Random Forest:

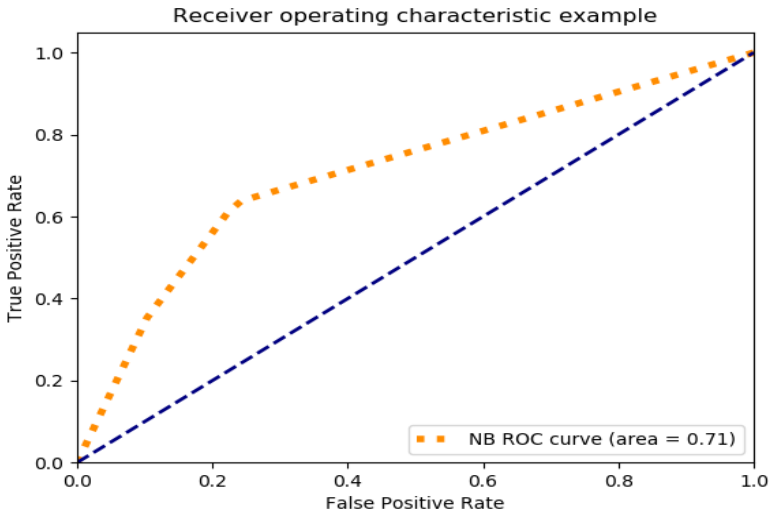
Random Forest is an ensemble learning algorithm for classification and regression. Random Forest generates a multitude of decision trees classifies based on the aggregated decision of those trees. For a set of tweets $x_1, x_2, \ldots x_n$ and their respective sentiment labels $y_1, y_2, \ldots y_n$ bagging repeatedly selects a random sample $(X_b, Y_b)$ with replacement. Each classification tree $f_b$ is trained using a different random sample $(X_b, Y_b)$ where $f_b$ ranges from 1 . . . B. Finally, a majority vote is taken of predictions of these B trees.

# 6. Results:

## 6.1. Naive Bayes

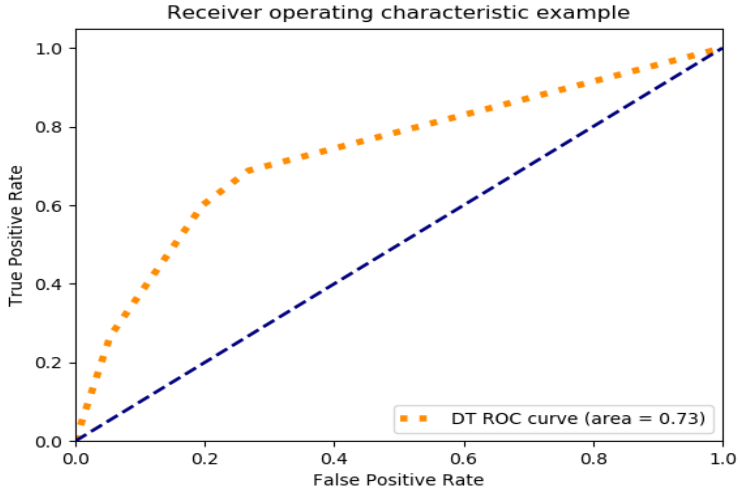
We used MultinomialNB from sklearn.naive_bayes package of scikit-learn for Naive Bayes classification. We used Laplace smoothed version of Naive Bayes with the smoothing parameter α set to its default value of 1. We obtain a best validation accuracy of

64.33 % using Naive Bayes with presence of unigrams and bigrams. The generated ROC curve is shown below with AUC 0.71. A comparison of accuracy, area under curve (AUC).



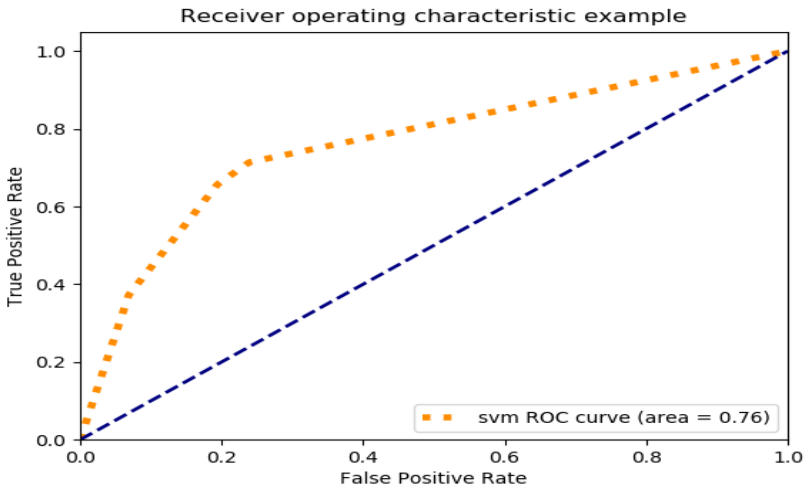

## 6.2. Decision Tree:

We use the DecisionTreeClassifier from sklearn.tree package provided by scikit-learn to build our model. GINI is used to evaluate the split at every node and the best split is chosen always. The best accuracy achieved using decision trees was 68.88%. The generated ROC curve is shown below with AUC 0.73. A comparison of accuracy, area under curve (AUC).



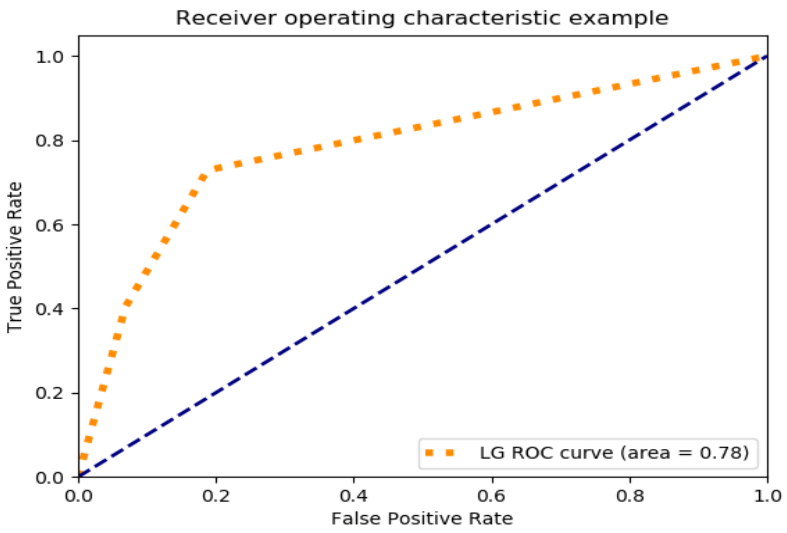
## 6.3. SVM:

We utilize the SVM classifier available in sklearn. We set the C term to be 0.1. C term is the penalty parameter of the error term. In other words, this influences the misclassification on the objective function. The best accuracy achieved using SVM was 69.3%. The generated ROC curve is shown below with AUC 0.76. A comparison of accuracy, area under curve (AUC).



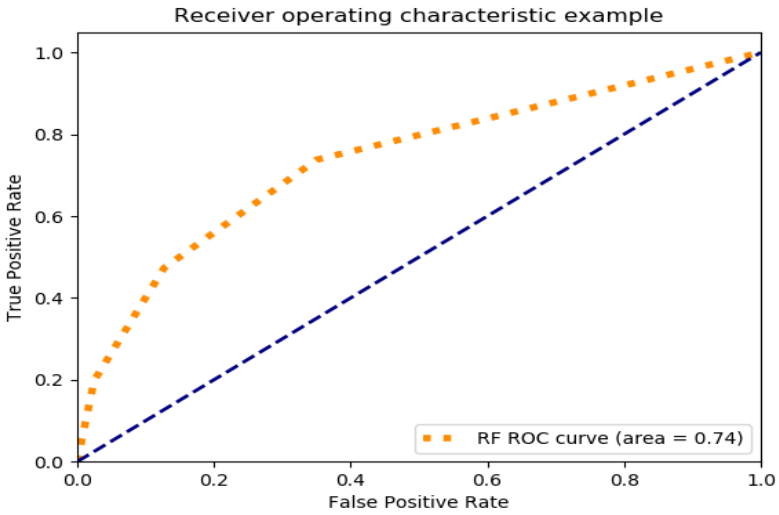

## 6.4. Logistic Regression:

We utilize the LogisticRegression classifier available in sklearn. The best accuracy achieved using this was 72.5%. The generated ROC curve is shown below with AUC 0.78. A comparison of accuracy, area under curve (AUC).




## 6.5. Random Forest:

We implemented random forest algorithm by using RandomForestClassifier from sklearn.ensemble provided by scikit-learn. The best accuracy achieved using this was 69.2%. The generated ROC curve is shown below with AUC 0.74. A comparison of accuracy, area under curve (AUC).

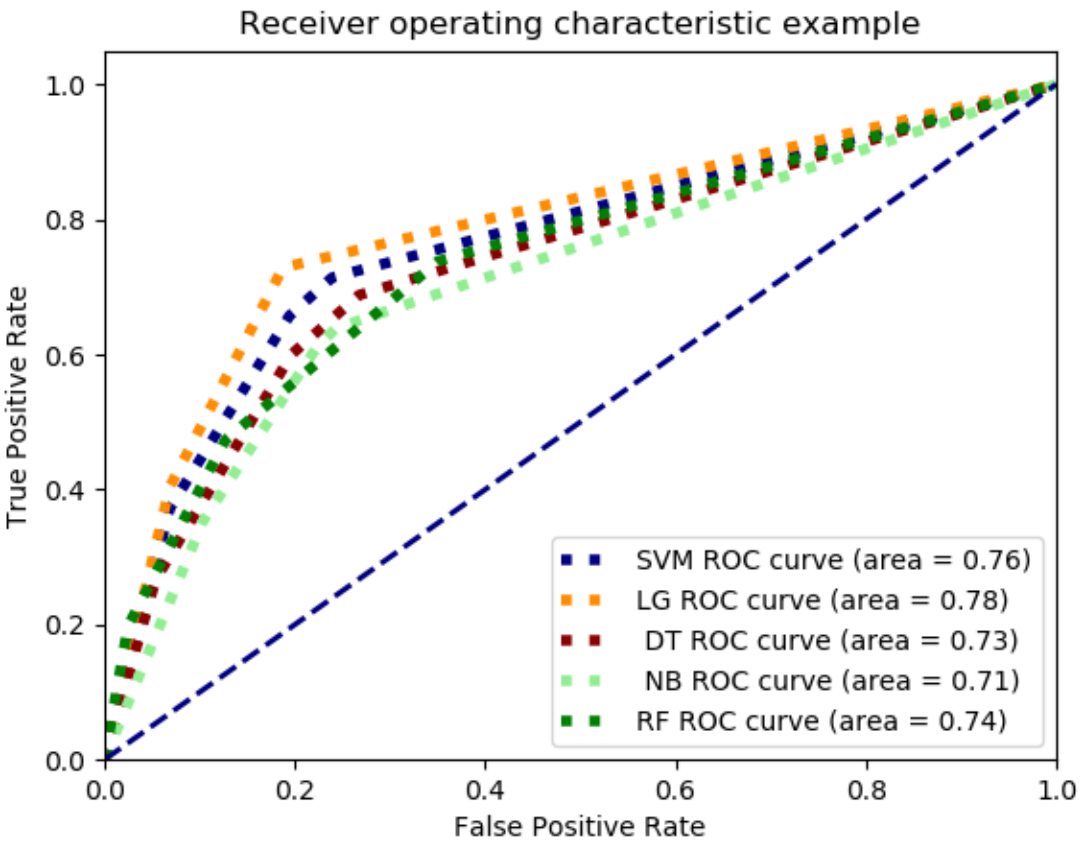



## 6.6. Performance of the Classifiers:

The comparison of the accuracies and area under the curve of various classifier is shown below:

| Classifier | Accuracy | Area Under Curve |
|---|---|---|
| **Naïve Bayes** | 64.33 | 0.71 |
| **Decision Tree** | 67.48 | 0.73 |
| **SVM** | 69.38 | 0.76 |
| **Logistic Regression** | 72.59 | 0.78 |
| **Random Forest** | 70.27 | 0.74 |

The comparison of ROC curves of all the algorithms plotted together is shown below:




# 7. Conclusion:

We have retrieved the data from twitter which contained various tweets regarding the selected topic “Mid Term Elections”. Before training different machine learning models we have preprocessed data to be suitable to train the models. We have used Decision Trees, Naïve Bayes, Logistic Regression, SVM, Random Forest to classify the polarity of the tweet. We used features extracted from Count Vectorizer and labels using Text Blob.

The comparison of the various algorithms is shown in the table above. We obtained the best accuracy of 72.5% and area under curve 0.78 both for logistic regression. The second-best accuracy was observed to be 70.27% for Random Forest and Area Under Curve was observed to be 0.76 for SVM.

# 8.References:


1. Alec Go, Lei Huang, and Richa Bhayani. 2009. Twitter sentiment analysis. Final Projects from CS224N for Spring 2008/2009 at The Stanford Natural Language Processing Group.

2. Kouloumpis, Efthymios, Theresa Wilson, and Johanna Moore. Twitter Sentiment Analysis: The Good the Bad and the OMG! in Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM-2011). 2011.

3. Jiang, Long, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target- dependent twitter sentiment classification. in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL- 2011). 2011.

4. H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 U.S. presidential election cycle. In Proceedings of the ACL 2012 System Demonstrations.

5. Scikit library documentation examples: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#sphx-glr-auto-examples-model-selection-plot-roc-py