Twitter Sentiment Analysis

Pratyaksh Tyagi , S. Chandradeep
Pattern Recognition & Machine Learning

***Abstract*— In this report, we address the problem of sentiment classification on twitter dataset. We use a number of machine learning methods to perform sentiment analysis. In the end, we use a majority vote ensemble method with 4 of our best models to achieve the classification accuracy of 75 %.**

I. Introduction

In this report, 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.

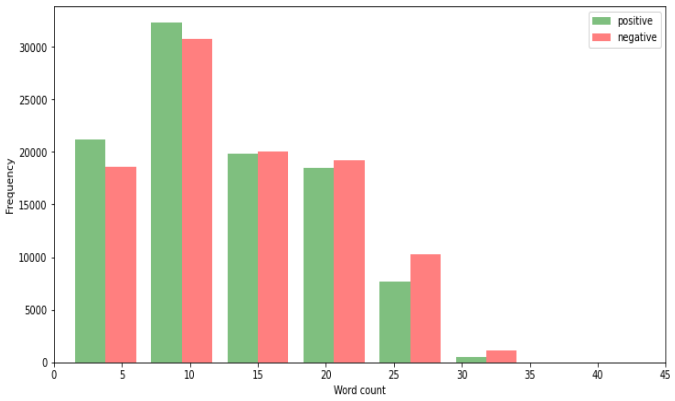
II. Data description

The data given is in the form of comma-separated values files with tweets and their corresponding sentiments. The dataset is a mixture of words, emoticons, symbols, URLs and references to people. Words and emoticons contribute to predicting the sentiment, but URLs and references to people don't. Therefore, URLs and references can be ignored. The words are also a mixture of misspelled words, extra punctuations, and words with many repeated letters. The tweets, therefore, have to be preprocessed to standardize the dataset
Details of the dataset

- date
- polarity : the polarity of the tweet (0 = negative 4 = positive)
- user : the user that tweeted
- text : the text of the tweet

Target: the polarity of the tweet (0 = negative, 4 = positive)

Word count distribution for both positive and negative



III. Pre - processing

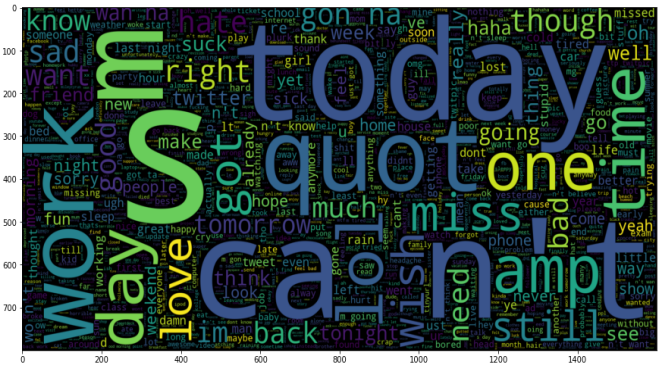
Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as retweets, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, raw twitter data has to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

- Lower Casing: Each text is converted to lowercase.
- Removing URLs: Links starting with "http" or "https" or "www" are replaced by "".
- Removing Usernames: Replace @Usernames with word "". (eg: "@XYZ" to "")
- Removing Short Words: Words with length less than 2 are removed.
- Removing Stopwords: Stopwords are the English words which do not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. (eg: "the", "he", "have")
- Lemmatizing: Lemmatization is the process of converting a word to its base form. (e.g: "wolves" to "wolf")

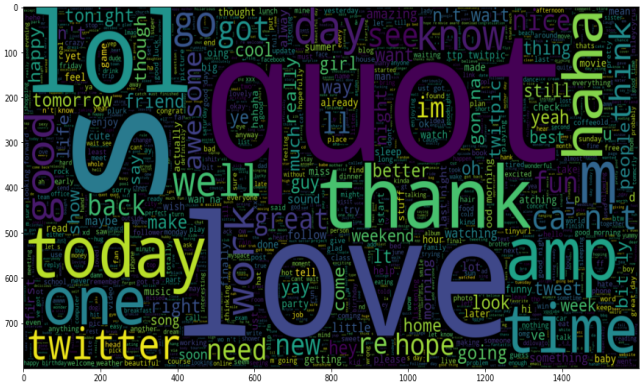
Polarity	Text	Processed tweets
1	@ddlovato Hey, uh, you wanna have a quick jam	ddlovato hey uh wan na quick jam session ahaha...
0	"I don't listen to you much." Thanks..	quot n't listen much. quot thanks sam b*tch .....
1	CD.net is in a babysteps slow process.... but ...	d.net babysteps slow process ... 's going rock

Analyzing the data

Now we're going to analyse the preprocessed data to get an understanding of it. We'll plot word clouds for positive and negative tweets from our dataset and see which words occur the most.



WORD CLOUD FOR NEGATIVE TWEETS



WORD CLOUD FOR NEGATIVE TWEETS

Random Forest

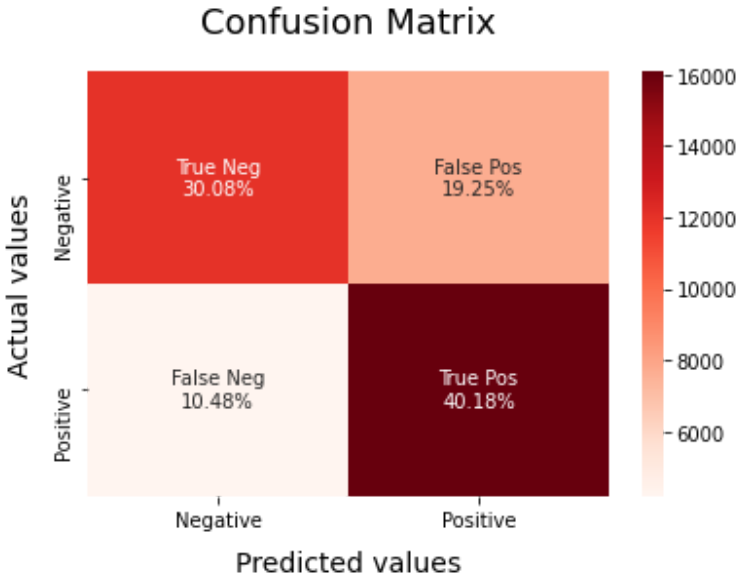
Random Forest is an ensemble learning algorithm for classification and regression. Random Forest generates a multitude of decision trees classified based on the aggregated decision of those trees. For a set of tweets x1, x2, . . . xn and their respective sentiment labels y1, y2, . . .n bagging repeatedly selects a random sample (Xb, Yb) with replacement. Each classification tree fb is trained using a different random sample (Xb, Yb) where b ranges from 1 . . . B. Finally, a majority vote is taken of predictions of these B trees.

Using random forest we achieved the following metrics -

Accuracy of model on training data : **75.125625 %**

Mean Accuracy on testing data using 5 fold cross validation: **67.4375 %**

Standard Deviation: **0.0325**



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

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

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

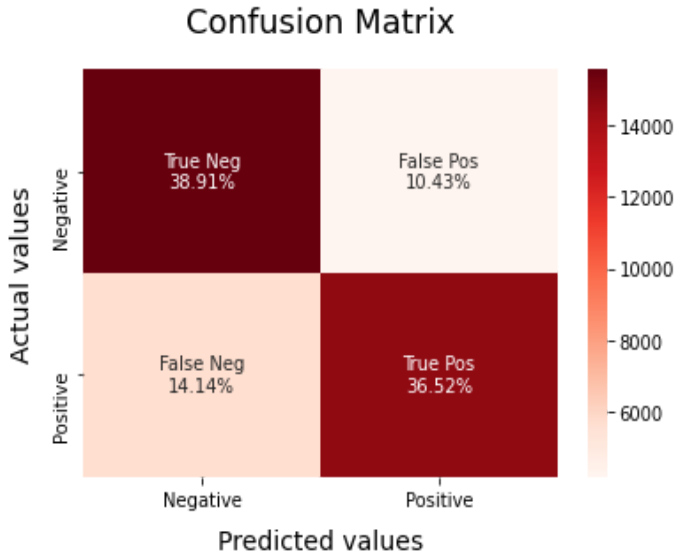
In the formula above, fi represents the i-th feature of total n features. P(c) and P(fi |c) can be obtained through maximum likelihood estimates.

Using Naive bayes classifier we achieved the following metrics -

Accuracy of model on training data: **86.105625 %**

Mean Accuracy on testing data using 5 fold cross validation: **72.7325 %**

Standard Deviation - **0.0675**



SVM

SVM, also known as support vector machines, is a non-probabilistic binary linear classifier. For a training set of points (xi , yi) where x is the feature vector and y is the class, we want to find the 7maximum-margin hyperplane that divides the points with yi = 1 and yi = −1. The equation of the hyperplane is as follows -

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

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

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

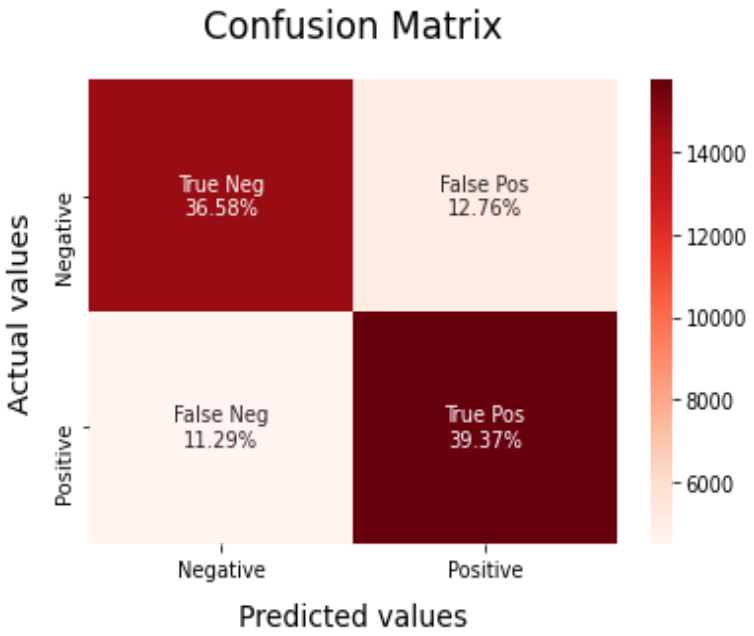
in order to separate the points well.

Using the Linear SVM classifier we were able to achieve the following metrics stated as below -

Accuracy of model on training data : **92.441875%**

Mean Accuracy on testing data using 5 fold cross validation: **73.2725 %**

Standard Deviation: **0.04250**



Logistic regression :

Logistic regression is one of the most popular ways to fit models for categorical data, especially for binary response data. It is the most important (and probably most used) member of a class of models called generalized linear models. Unlike linear

regression, logistic regression can directly predict probabilities (values that are restricted to the (0,1) interval); furthermore, those probabilities are well-calibrated when compared to the probabilities predicted by some other classifiers, such as Naive Bayes. Logistic regression preserves the marginal probabilities of the training data. The coefficients of the model also provide some hint of the relative importance of each input variable.

$$\log \frac{P(\mathbf{x})}{1 - P(\mathbf{x})} = \sum_{j=0}^{K} b_j x_j$$

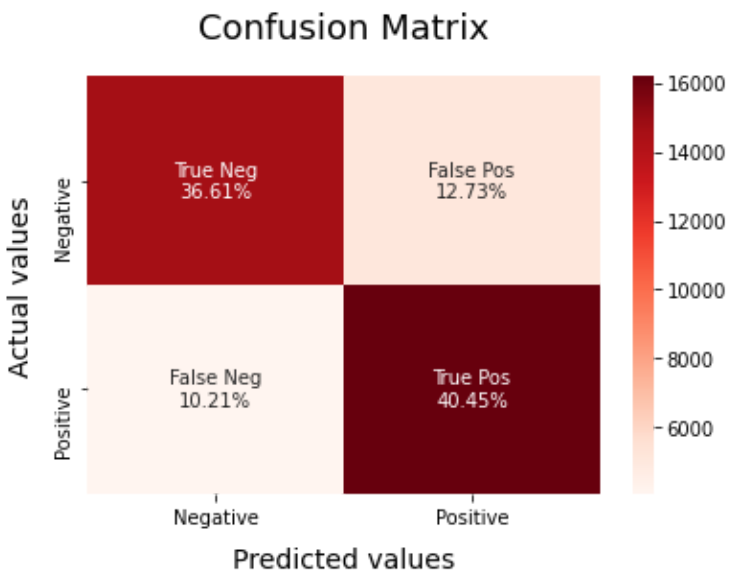
$$P(\mathbf{x}) = \frac{\exp z}{1 + \exp z},$$
$$z = \sum_{j=0}^{K} b_j x_j$$

$$\mathrm{L}(X|P) = \prod_{i=1, y_i=1}^{N} P(\mathbf{x}_i) \prod_{i=1, y_i=0}^{N} (1 - P(\mathbf{x}_i))$$

Using the LogisticRegression classifier we were able to achieve the following metrics stated as below -

Accuracy of model on training data: **83.153125%**

Mean Accuracy on testing data using 5 fold cross validation: **74.4625 %**
Standard Deviation: **0.2375**



Conclusions

From the above analysis we can observe that Logistic Regression gave the best accuracy at 74.4 % while Random Forest gave 67.4 %. Logistic regression performs better when the number of noise variables is less than or equal to the number of explanatory variables and the random forest has a higher true and false positive rate as the number of explanatory variables increases in a dataset.

Contribution

Pratyaksh Tyagi (B19EE067) - Wrote the Code, Implementation of Models.

S Chandradeep (B19CSE089) - Wrote Analysis and did implementation of models.

REFERENCE

Randomforest - randomForest function - RDocumentation
Naive Bayes - Naïve Bayes Classifier — H2O 3.32.1.2 documentation
SVM - Support Vector Machine (SVM) — H2O 3.32.1.2 documentation
Logisticregression- sklearn.linear_model.LogisticRegression — scikit-learn 0.24.2 documentation