Summary of model selection and results

Introduction:

More than 30000 tweets were extracted and used for training the models. The training was however done in two steps: 1) To find the sentiments using the nltk and tweepy API to find the sentiments of every tweets, and 2) To use these sentiments to train the model and then do an analysis of the model. This is the most basic approach for now as but in the future will be using some more complex model like Decision Tree and Neural Network.

In the following sections we have discussed about the experiment, setup and result for our prediction.

Experiment:

We used Tweepy API to stream the tweets from the twitter stored it as a csv file with different features or information about the tweets such as the date created, location, retweet count, favorite count etc. Our experiment objective is to develop different models that can be used to predict the upcoming tweets with the same information used to train the model. To achieve this objective, we first use the sentiment score from the TextBlob API/NLTK API to find the sentiments of the tweets. Then used a Linear model to train the model on it, predicted the outcome and accuracy for the testing data.

Setup:

TextBlob Library:

It is a python library for processing text data. It provides consistent solutions into common Natural Language Processing (NLP) tasks, here we used it for sentiment analysis. This sentiment module uses *PatternAnalyzer* and *NaiveBayesAnalyzer* of TextBlob to determine the polarity and subjectivity of the text. If the text is long, then it averages out the polarity of the whole text. It is fast and accurate as it uses statistical approaches and regular expressions.

If the polarity of the text is greater than zero, then it is considered to be positive, for polarity less than zero it is considered to be negative and for polarity equals to zero the text is considered to be neutral. Using this function, we created a new column in our dataset that stores the sentiments as ‘positive’, ‘negative’ and ‘neutral’.

NLTK Library:

NLTK is a python library that is used to work with the human language data. This library uses VADER (Valence Aware Dictionary and Sentiment Reasoner) Sentiment Analysis, which is a rule based sentiment analysis tool that is specifically attuned to sentiments expressed in a text. Calculation of these scores are based on the lexicon metrics which is further normalized between -1 and 1, -1 is considered to be extremely negative and 1 is considered to be extremely positive.

If the scores are above 0.5 then we have considered it to be positive, if the scores are below -0.5 then we have considered this to be negative and for the rest of the cases it is considered to be neutral. Using this function, we have created a new column in our dataset that stores the sentiments as ‘positive’, ‘negative, and ‘neutral’.

Note: *We have used two different libraries to find the sentiments, we will show the comparison between the two and will be using only one for training the model.*

Positive, negative and neutral words count:

Using the NLTK library while determining the sentiments we also stored the number of positive, negative and neutral counts for every tweet. The purpose for all this was to create additional data that could be used further while training any model, as the direct information of ‘positive’, ‘negative’ and ‘neutral’ were not enough for determining the models.

Model Training:

We used Multiple Linear Regression model to train our dataset. Our target was to predict the sentiments to be as 'positive', 'negative' and 'neutral'. The features used for predicting our target other than the positive, negative and neutral word counts were *'reply_count', 'retweet_count'* and *'favorite_count'*. The result of the model is summarized below.

```
Accuracy:  0.9008491508491508
R^2 of the Prediction:  0.7417011225475982
```

The predicted result was then converted into the 1 and 0 format. This transformation was conducted by choosing the maximum value or score (predicted by the model) among the three sentiments and assigning it a score of 1 and others as 0. This transformation was important as our training model was predicting a category and which was converted as dummy values. In order to have the accuracy the predicted values should also be in the same format, but instead it was a float value.

Results:

Figure 1: This figure shows us that the predicted sentiments by TextBlob were almost all Neutral. I have used almost as the result considered only 3 tweets to be positive and 3 tweets to be negative and rest of the 20012 tweets to be neutral. While plotting it was very difficult to show the positive and negative as it was comparatively very small.

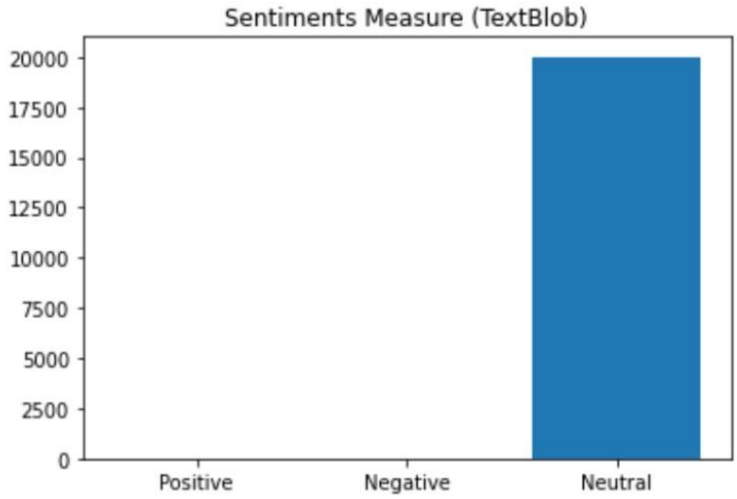

Fig 1. Sentiments measures using TextBlob Library

Figure 2: This figure has in my opinion a better result than what TextBlob. The way TextBlob worked it will always come up with a smaller score for any given tweets. There was no pre-processing done like removal of stopwords and tokenizing the text. NLTK on the other hand have a big repository of words and symbols and each of these words and symbols have their own scores.

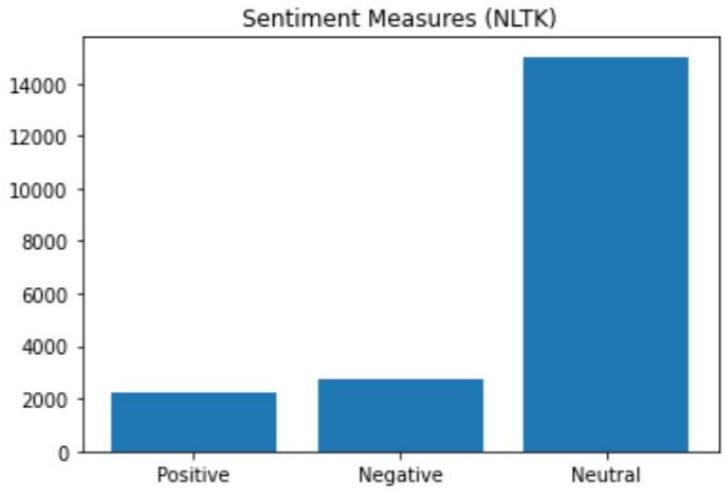

Fig 2. Sentiments analysis by NLTK library

Figure 3: This figure just shows the graph of actual and predicted sentiments of the test data. This graph shows that our model was nearly accurate to predict our data. It almost predicted negative sentiment correctly. From here we can say that the model was okay as it had the accuracy of 90 percent, and some changes like forward selection or backward selection can be done to further check the accuracy of the linear model.

```
Actual Positive:  413
Predicted Positive:  142
Actual Negative:  547
Predicted Negative:  549
Actual Neutral:  3044
Predicted Neutral:  3313
```

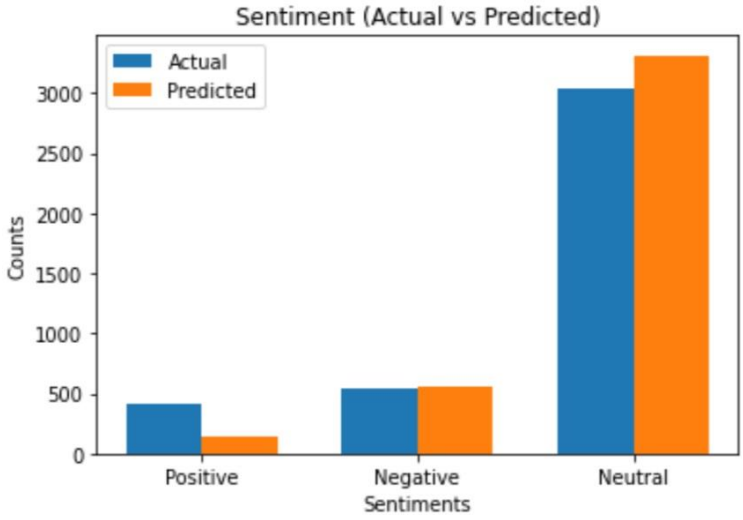

Fig 3. Comparison of the Actual and Predicted sentiments

Limitations:

There are however still few limitations with the tweet data, these limitations were not attended in this part of the project and would try to find a way to overcome at least few of them. The limitations are as follows:

1) *Sarcasm*: The libraries used in this part of project has no way of checking the sarcasm. In fact, identifying sarcasm is one of the most difficult task till date as it depends a lot on the tone of the speaker and could only be identified if a human is reading it.

2) *Short-cuts:* As the twitter allows only 140 characters, people have a tendency to use shortcuts like instead of writing 'I don't know' they will write 'IDK'. Words and characters like this are hard to be identified by the ongoing libraries and so there is this shortcoming where a lot of information is still not identified.

3) *Emoji's:* We haven't considered the emoji's for now, but will surely do it in the future work.

References:

1)https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f

2) https://textblob.readthedocs.io/en/dev/quickstart.html