Project name: Comparison between sentiment analysis of Twitter data using Naive Bayes and TextBlob

Sentiment Analysis can help us decipher the mood and emotions of the general public and gather insightful information regarding the context. Sentiment Analysis is a process of analyzing data and classifying it based on the need of the research. These sentiments can be used for a better understanding of various events and impact caused by it.
Tweets can usually be put into two categories: **positive tweets** are hopeful and confident and think of good aspects of a situation rather than the bad ones. On the other hand, **negative tweets** are seen as expressing negative connotations like refusal or denial. By analysing these sentiments, we may find what people like, what they want and what their major concerns are.

In this project I have done a **comparison** of sentiment analysis of twitter data using
i) Naive Bayes Classifier
ii) TextBlob NLP library

Naïve Bayes Classifier

The Naïve Bayes algorithm is a **supervised learning algorithm**, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. The Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.
Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B and,

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

Sentiment Analysis using TextBlob

TextBlob is a python library for Natural Language Processing (NLP). For lexicon-based approaches, a sentiment is defined by its semantic orientation of each word in the sentence. This requires a pre-defined dictionary classifying negative and positive words.

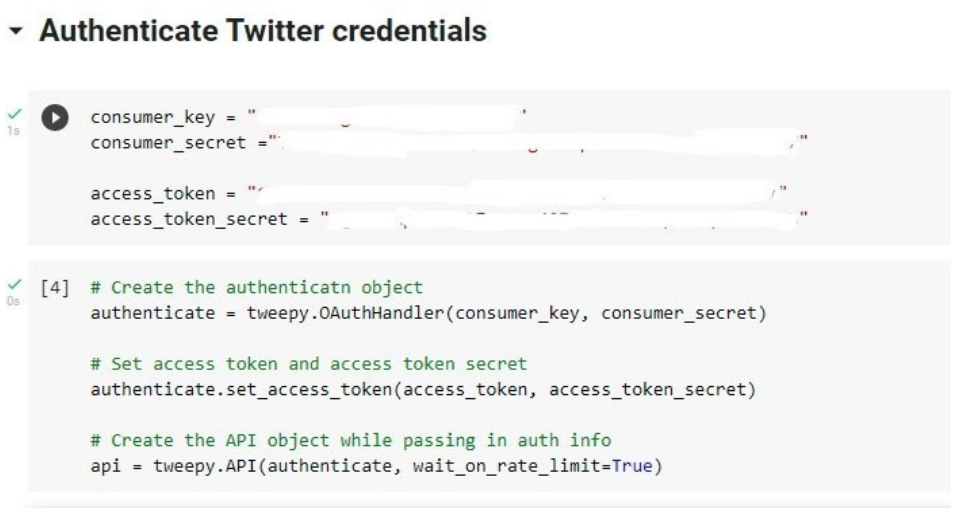
TextBlob returns polarity and subjectivity of a sentence. Polarity lies between [-1,1], -1 defines a negative sentiment and 1 defines a positive sentiment. Negation words reverse the polarity. TextBlob has semantic labels that help with fine-grained analysis. Subjectivity lies between [0,1]. It quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains personal opinion rather than factual information.

For example: We calculated polarity and subjectivity for “I do not like this example at all, it is too boring”. For this particular example, polarity = -1 and subjectivity is 1, which is fair. However, for the sentence “This was a helpful example but I would prefer another one”. It returns 0.0 for both subjectivity and polarity, which is not very helpful. However, it works fine for simple sentences.

Explaining the process

Steps:

1) **Authenticate Twitter credentials**: The twitter API is verified



2) **Download twitter sentiment training data for Naive Bayes**: NLTK's Twitter corpus currently contains a sample of 20k Tweets (named 'twitter_samples') retrieved from the Twitter Streaming API, together with another 10k which are divided according to sentiment into negative and positive.

```
nltk.download('twitter_samples')
```

```
[nltk_data] Downloading package twitter_samples to /root/nltk_data...
[nltk_data]   Unzipping corpora/twitter_samples.zip.
True
```

Assign variables for training data

```
# load text fields of positive and negative tweets
train_pos = twitter_samples.strings('positive_tweets.json')
train_neg = twitter_samples.strings('negative_tweets.json')

train_x = train_pos + train_neg
train_y = np.append(np.ones(len(train_pos)), np.zeros(len(train_neg)))  # 1 = positive, 0 = negative
```

```
print('No. of positive tweets = ', len(train_pos))    # no. of all positive tweets
print('No. of negative tweets = ', len(train_neg))    # no. of all negative tweets
```

```
No. of positive tweets =  5000
No. of negative tweets =  5000
```

3) **Get Test Tweets**: Tweets to be analysed are gathered using keyword/hashtag/particular id. Here the trending hashtag #climatechange has been used to gather 100 tweets. A dataframe containing tweets is created.

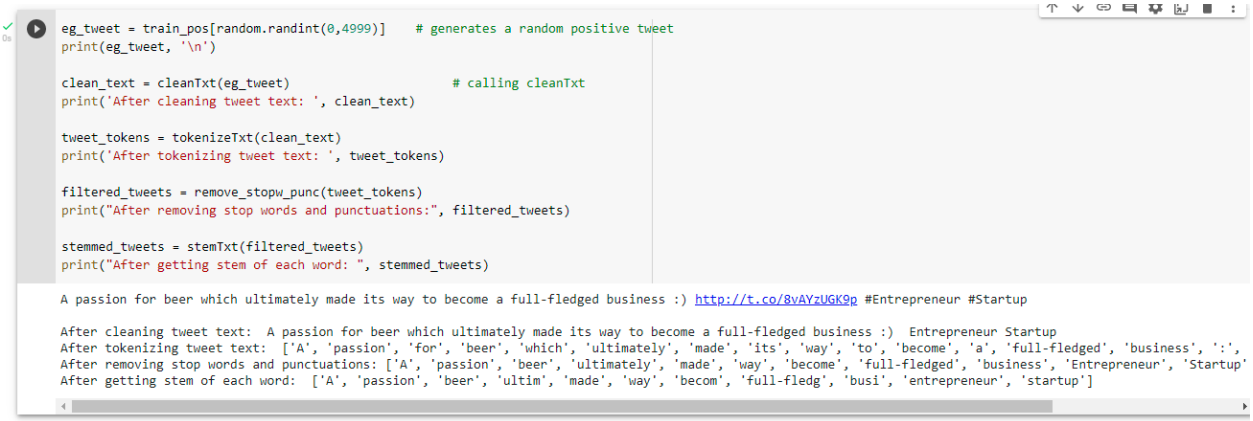
```
# Extract 'n' number of tweets from twitter user
n = 100
keyword = 'climatechange'
posts = tweepy.Cursor(api.search, q=keyword, tweet_mode="extended", lang='en').items(n)
```

Create df for tweets

```
# Create a df with a col called 'Tweet'
df = pd.DataFrame([tweet.full_text for tweet in posts], columns=['Tweets'])
print(df)
```

```
                                               Tweets
0   RT @EpochOpinion: Opinion💭by @DennisPrager\n\n...
1   RT @chriscartw83: Depressing #Europe  emitting...
2   RT @docsforclimate: 🚨Webinar Alert🚨\n\nJoin @C...
3   @edwasp Without HS2 we simply won't have the r...
4   RT @DineshDSouza: Today: A wide-ranging, enter...
..                                                ...
95  The last official search was Saturday, with 12...
96  police said the body was found on a ridge in a...
97  It was a run that would've taken him an hour… ...
98  Missing Berkeley student, Philip Kreycik, has ...
99  RT @GeraldKutney: My MP is @PierrePoilievre in...
```

4) **Data preprocessing**:
   a) cleanTxt() : removes '@' mentions, '#' symbol, old style retweets and hyperlinks from tweets
   b) tokenizeTxt() : reduces a sentence into individual word, ie, token list
   c) remove_stopw_punc() : removes words irrelevant to analysis, eg. 'they', 'it', 'the', 'are' using nltk stopwords package.
   d) stemTxt() : reduces each token into its stem using PorterStemmer(), eg. 'swimmer' and 'swimming' both are reduced to 'swim'
   e) PreprocessTxt() : calls above functions to combine them into a single call
   f) Create Frequency Dictionary for NB: We create a dictionary consisting of a pair of word and its polarity as keys, and the frequency as values, eg. {(happy,1), 32…}
   g) Get subjectivity and polarity of data for NLP: Subjectivity quantifies the amount of personal opinion and factual information contained in the text, while Polarity tells us how positive or negative it is.



```
eg_tweet = train_pos[random.randint(0,4999)]    # generates a random positive tweet
print(eg_tweet, '\n')

clean_text = cleanTxt(eg_tweet)                      # calling cleanTxt
print('After cleaning tweet text: ', clean_text)

tweet_tokens = tokenizeTxt(clean_text)
print('After tokenizing tweet text: ', tweet_tokens)

filtered_tweets = remove_stopw_punc(tweet_tokens)
print("After removing stop words and punctuations:", filtered_tweets)

stemmed_tweets = stemTxt(filtered_tweets)
print("After getting stem of each word: ", stemmed_tweets)
```

```
A passion for beer which ultimately made its way to become a full-fledged business :) http://t.co/8vAYzUGK9p #Entrepreneur #Startup

After cleaning tweet text:  A passion for beer which ultimately made its way to become a full-fledged business :)  Entrepreneur Startup
After tokenizing tweet text:  ['A', 'passion', 'for', 'beer', 'which', 'ultimately', 'made', 'its', 'way', 'to', 'become', 'a', 'full-fledged', 'business', ':',
After removing stop words and punctuations: ['A', 'passion', 'beer', 'ultimately', 'made', 'way', 'become', 'full-fledged', 'business', 'Entrepreneur', 'Startup'
After getting stem of each word:  ['A', 'passion', 'beer', 'ultim', 'made', 'way', 'becom', 'full-fledg', 'busi', 'entrepreneur', 'startup']
```

5) **Training model using Naive Bayes**: we find out 2 probabilities: data based (log_prob) and word based (log_dict). Log_prob is the log of ratios of the number of positive and negative training tweets(D_pos and D_neg). Log_dict stores as values, for every word as key, the log of probability of it being positive or negative. Eg how many times is the word 'company' used in positive or negative annotations. Then we can calculate the probability of the word being positive or negative using bayes theorem.

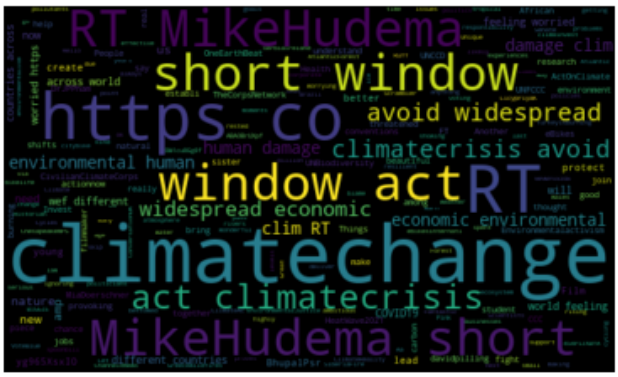
```
[71] # Create cols to store subjectivity and polarity
     df['Subjectivity'] = df['Tweets'].apply(getSubjectivity)
     df['Polarity'] = df['Tweets'].apply(getPolarity)
```

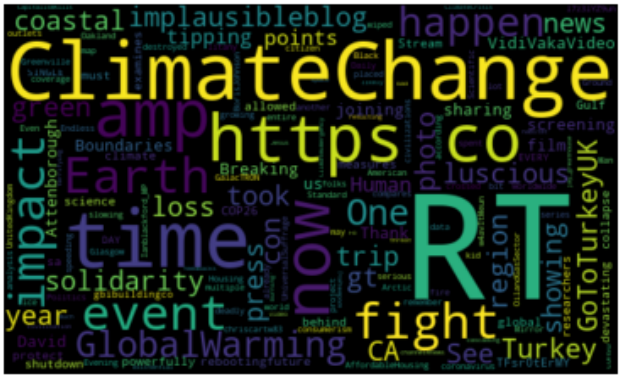
Training model using Naive Bayes

```
# Build frequency dictionary for traning data
freq_dict = create_freqdict(train_x, train_y)
print(freq_dict)
```

```
, ('mare', 0.0): 1, ('ivan', 0.0): 1, ('happi', 0.0): 25, ('trip', 0.0): 11, ('keep', 0.0): 34, ('safe', 0.0): 5, ('see', 0.0): 125, ('soon', 0.0): 46, ('tire',
```

6) **Analysing tweets**
   a) using Bayes theorem: After preprocessing tweet, we calculate the log_dict of each token in the tweet to calculate its net sentiment.
   b) using TextBlob: we analyse the polarity score of the tweet to identify its sentiment
   c) We then plot the positive and negative tweet word clouds.



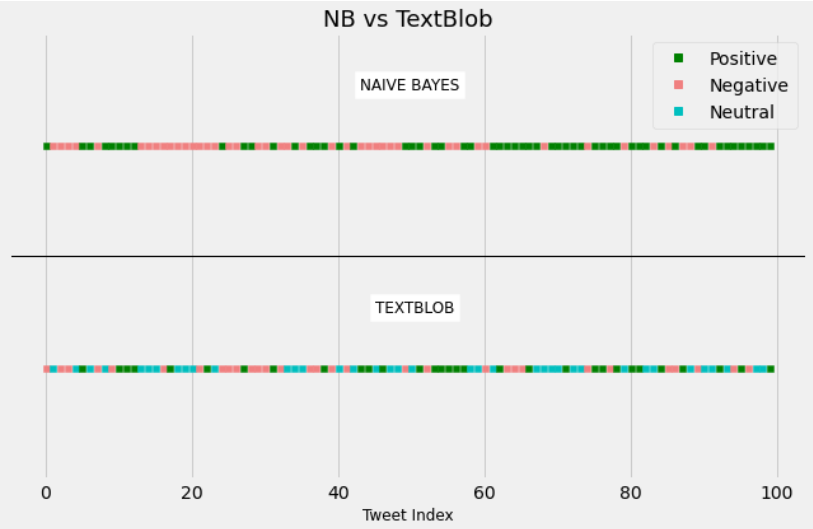


7) **Plotting comparison graphs**
   a) Creating ‘nb’ and ‘nlp’ arrays that store sentiments corresponding to tweet index.

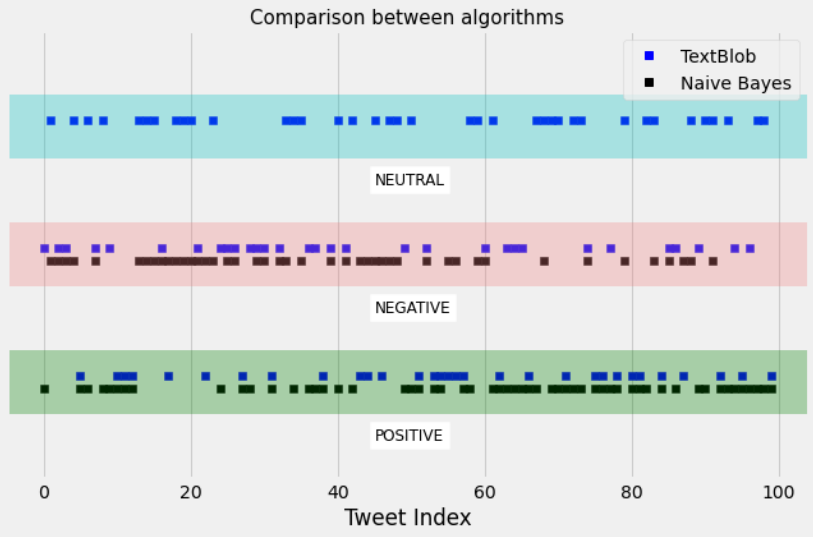
```
nb array is:  ['Positive' 'Negative' 'Negative' 'Negative' 'Negative' 'Positive'
 'Positive' 'Negative' 'Positive' 'Positive' 'Positive' 'Positive'
 'Positive' 'Negative' 'Negative' 'Negative' 'Negative' 'Negative'
 'Negative' 'Negative' 'Negative' 'Negative' 'Negative' 'Negative'
 'Positive' 'Negative' 'Negative' 'Positive' 'Positive' 'Negative'
 'Negative' 'Positive' 'Negative' 'Negative' 'Positive' 'Negative'
 'Positive' 'Positive' 'Positive' 'Negative' 'Positive' 'Negative'
 'Positive' 'Negative' 'Negative' 'Negative' 'Negative' 'Negative'
 'Negative' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
 'Positive' 'Negative' 'Negative' 'Positive' 'Positive' 'Negative'
 'Negative' 'Positive' 'Positive' 'Positive' 'Positive' 'Positive'
 'Positive' 'Negative' 'Positive' 'Positive' 'Positive' 'Positive'
 'Positive' 'Negative' 'Positive' 'Positive' 'Positive' 'Negative'
 'Positive' 'Negative' 'Positive' 'Negative' 'Negative' 'Positive'
 'Positive' 'Negative' 'Positive' 'Positive' 'Positive' 'Positive'
 'Positive' 'Positive' 'Positive' 'Positive']

nlp array is:  ['Negative' 'Neutral' 'Negative' 'Negative' 'Neutral' 'Positive' 'Neutral'
 'Negative' 'Neutral' 'Negative' 'Positive' 'Positive' 'Positive'
 'Neutral' 'Neutral' 'Neutral' 'Negative' 'Positive' 'Neutral' 'Neutral'
 'Neutral' 'Negative' 'Positive' 'Neutral' 'Negative' 'Negative'
 'Negative' 'Positive' 'Negative' 'Negative' 'Negative' 'Positive'
 'Negative' 'Neutral' 'Neutral' 'Neutral' 'Negative' 'Negative' 'Positive'
 'Negative' 'Neutral' 'Negative' 'Neutral' 'Positive' 'Positive' 'Neutral'
 'Positive' 'Neutral' 'Neutral' 'Negative' 'Neutral' 'Positive' 'Negative'
 'Positive' 'Positive' 'Positive' 'Positive' 'Positive' 'Neutral'
 'Neutral' 'Negative' 'Neutral' 'Positive' 'Negative' 'Negative'
 'Negative' 'Positive' 'Neutral' 'Neutral' 'Neutral' 'Neutral' 'Positive'
```

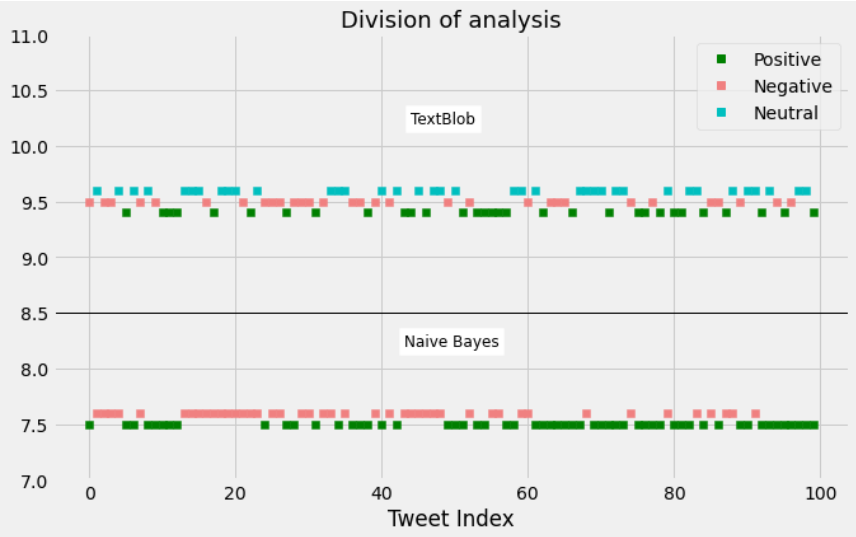
   b) NB vs NLP graph is drawn to ensure all tweets have been analysed



c) Comparison between algorithms has been plotted to compare the number of tweets classified as positive, negative or neutral using different algorithms.



d) Division of analysis has been plotted to compare the number of tweets classified as positive, negative or neutral within the same algo.


Division of analysis
Positive
Negative
Neutral
TextBlob
Naive Bayes
11.0
10.5
10.0
9.5
9.0
8.5
8.0
7.5
7.0
0
20
40
60
80
100
Tweet Index