MACHINE LEARNING

Assignment-2B
Group No- 14
Roll numbers- 19CS10037,19CS30049
Names- K Sai Santhoshi Niharika, Surkanti Akshitha

Dataset info:

- The data used in training was provided in .csv format.
- The data set is related to twitter racist and nonracist comments. Label attribute helps us to differentiate them
- The data had the following attributes:
    - ID
    - Label
    - Tweet

The flow of events:

- Read the data from train.csv using the pandas library.
- We have pre-processed the data so that we can save the memory and reduce the computational process.
    - Remove hyperlinks, retweets, and tags because they don't contribute to the sentimental analysis of a tweet.
    - Removing stop words: The stopwords like 'the', 'is' don't contribute to sentiment. Therefore these words have to be removed.
    - Tokenization of a tweet.
    - Stemming: The words like 'took', 'taking' are treated as the same words and are converted to their base words, here it is 'take'. This saves a lot of memory and time.
- Matrix construction using sparse representation of a matrix(csr_matrix).
- The processed data was split into train and test datasets in a 70:30 ratio.
- Trained the data using the Naive Bayes classifier and computed the accuracy.
- Trained the data using the Naive Bayes classifier with the Laplace correction and computed the accuracy.

Matrix construction:

- **Indexing:** Indexing starts with 0 and gives contiguous numbers for unique words i.e., if the words are repeating we assign the same number as previous to that word. From this we will get to know how many words are present in the entire document.

- **Creating sparse matrix for data:** CSR matrix representation requires three arrays index, value and pointer.

```
<31962x29421 sparse matrix of type '<class 'numpy.int64'>'
        with 2 stored elements in Compressed Sparse Row format>
```

Functions:

- **remove_hyperlinks_marks_styles:** It takes tweet string as an input and uses re library to remove hyperlinks and retweet text from a tweet.
- **tokenize_tweet:** It takes a tweet string as input and tokenizes the input using the TweetTokenizer library.
- **remove_stopwords_punctuations:** It takes the tweet tokens list as an input and uses the re library to remove stop words and punctuations from tweet tokens. For this, the set of stop words is downloaded from the nltk library.
- **get_stem:** It takes a clean tweet list as input and uses PorterStemmer library for stemming.
    - Learnt → learn
    - Learning → learn
- **process_tweet:** This function combines all the preprocessed techniques i.e., remove_hyperlinks_marks_styles, tokenize_tweet, remove_stopwords_punctuations, get_stem into one, and outputs the processed data.
- **create_frequency:** This function creates a dictionary of these words and counts the occurrence of each word in racist and nonracist tweets.
- **train_naive_bayes:** This function calculates logprior and loglikelihood of unique words taking frequency dictionary, tweets, and labels as input. This uses the NumPy library for finding logarithms.
- **naive_bayes_predict:** To estimate the sentiment of the tweet this function takes input as a tweet, logprior, and likelihood of unique words and returns the estimated sentiment of a tweet. Log over naive Bayes equation gives the estimate of the sentiment of the given tweet as shown in the below equation.

$$log(\frac{P(pos)}{P(neg)}\prod_{i=1}^{n}\frac{P(w_i|pos)}{P(w_i|neg)}) \Rightarrow \underbrace{log\frac{P(pos)}{P(neg)}}_{\text{log prior}} + \underbrace{\sum_{i=1}^{n} log\frac{P(w_i|pos)}{P(w_i|neg)}}_{\text{log likelihood}}$$

- **train_naive_bayes_laplace:** This function calculates logprior and loglikelihood of unique words taking frequency dictionary, tweets, and labels as input. The probability of a word being in racist or nonracist class is calculated using Laplace smoothing as shown below.

$$P(w_i \mid class) = \frac{freq(wi,class)+1}{Nclass + V}$$

Nclass = Frequency of all words in class

V = number of unique words in the vocabulary.

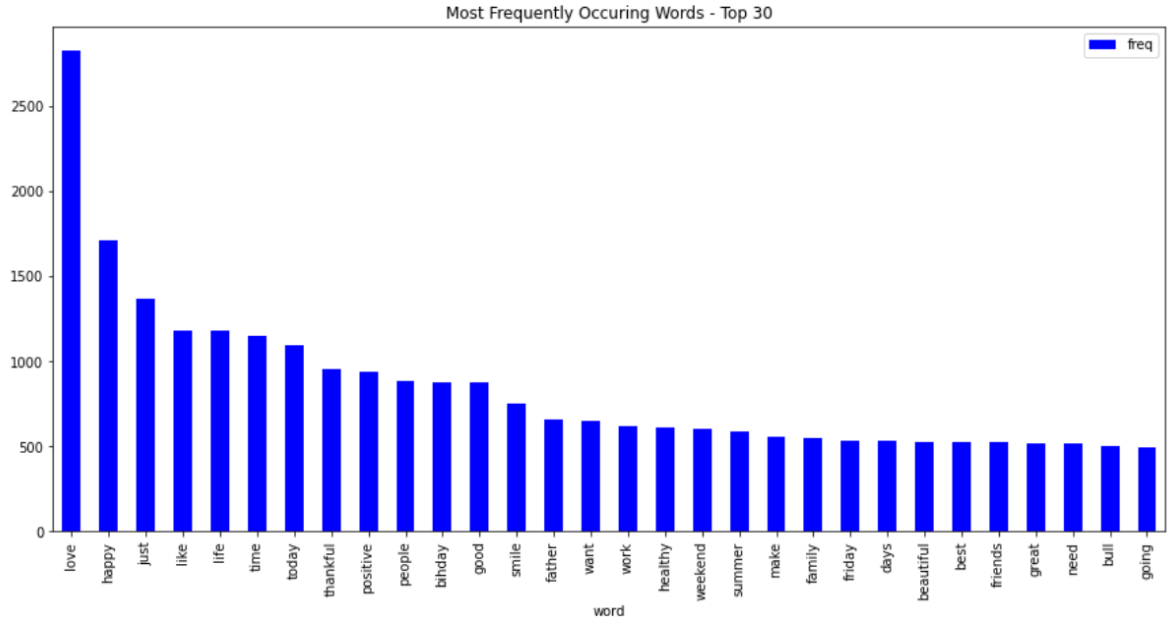
- **naive_bayes_predict_laplace**: To estimate the sentiment of the tweet this function takes input as a tweet, logprior, and likelihood of unique words and returns the estimated sentiment of a tweet. Log over naives Bayes equation gives the estimate of the sentiment of the given tweet. If the loglikelihood of the tweet is positive then it is a non-racist tweet and vice versa.

Help Function :

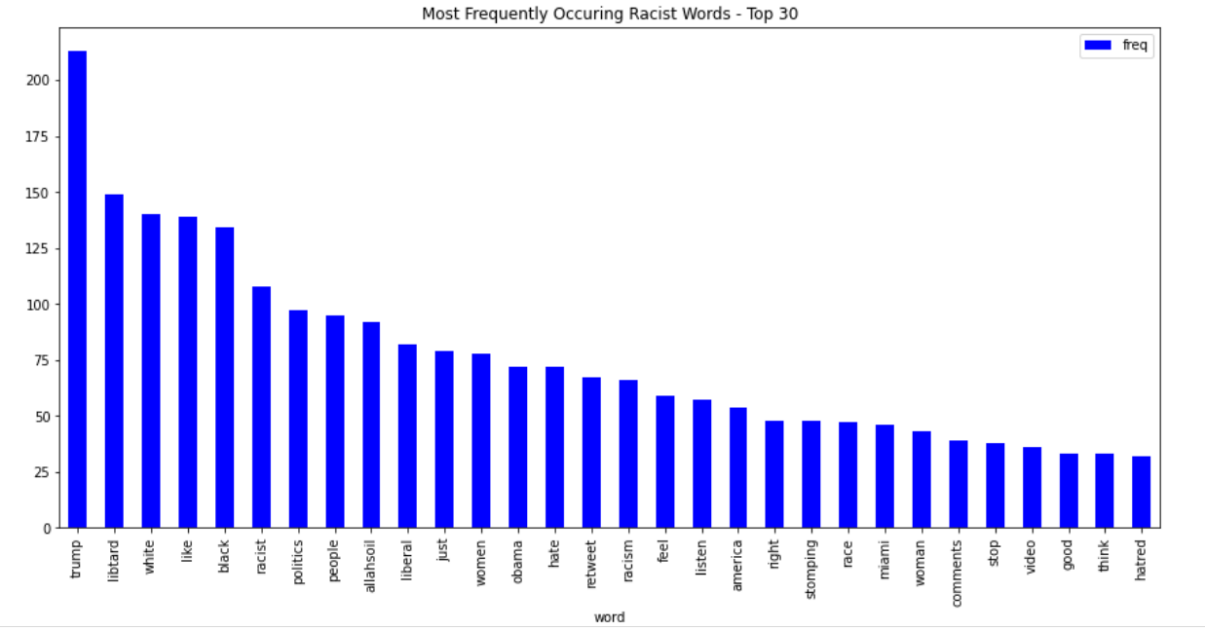
- **reverse:** This function changes the label of racist tweets to ‘0’ and non-racist tweets to ’1’.It is a help function for naive bayes and naive bayes with laplace smoothing classifiers.
  NOTE: run this function only once in the flow of code.
- **unique_pro :** This function helps to get the unique words in a list. It is a help function for creating Index, Val and pointer arrays in matrix construction

Plots :

- The plot of maximum occurring words:



- The plot of maximum words in racist tweets



Results:

95% confidence interval of the accuracy, precision, f-score, sensitivity and specificity for navies bayes with and without using laplace smoothing are as follows :

- Accuracy using naive Bayes

```
Accuracy using naive bayes:  96.93399  +/-  0.003

Precision using naive bayes:  97.74030  +/-  0.003

Sensitivity using naive bayes: 99.00625  +/-  0.002

F1_score using naive bayes: 98.36920  +/-  0.003

Specificity using naive bayes: 67.61453  +/-  0.009
```

- Accuracy using Laplace smoothing

```
Accuracy using naive bayes with laplace smoothing: 96.93399  +/-  0.003

Precision using naive bayes with laplace smoothing: 97.74030  +/-  0.003

Sensitivity using naive bayes with laplace smoothing: 99.00625  +/-  0.002

F1_score using naive bayes with laplace smoothing: 98.36920  +/-  0.003

Specificity using naive bayes with laplace smoothing: 67.61453  +/-  0.009
```

How to run the code :

- Download ipynb file and data set into your local machine
- Upload the above-downloaded files into the drive
- Open them in google collab from your drive
- Run every part in the notebook from the beginning to the end and the later parts will give you the desired results.
- After running the last parts you will observe the accuracy as output for both Naive Bayes and Naive Bayes with Laplace correction.