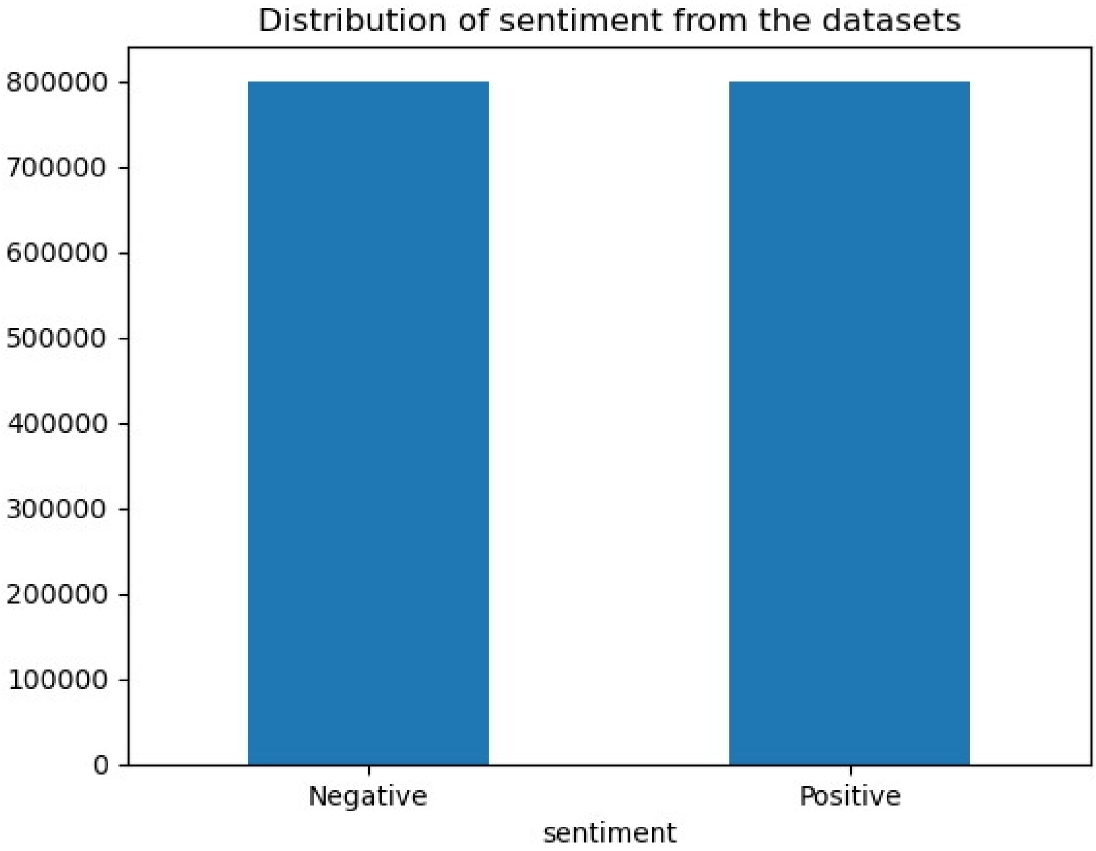
```
## importing the required libraries
import re
import pickle
import numpy as np
import pandas as pd
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from nltk.stem import WordNetLemmatizer
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, classification_report
from nltk.stem import WordNetLemmatizer

DATASET_COLUMNS  = ["sentiment", "ids", "date", "flag", "user",
"text"]
DATASET_ENCODING = "ISO-8859-1" # it includes almost all the western
languages for encoding
dataset = pd.read_csv('training.1600000.processed.noemoticon.csv',
                      encoding=DATASET_ENCODING ,
names=DATASET_COLUMNS)
dataset = dataset[['sentiment','text']]
# Replacing the values to ease understanding.
dataset['sentiment'] = dataset['sentiment'].replace(4,1)

# Plotting the distribution for dataset.
ax = dataset.groupby('sentiment').count().plot(kind='bar',
title='Distribution of sentiment from the datasets',
                                               legend=False)
ax.set_xticklabels(['Negative','Positive'], rotation=0)

# Storing data in lists.
text, sentiment = list(dataset['text']), list(dataset['sentiment'])
```



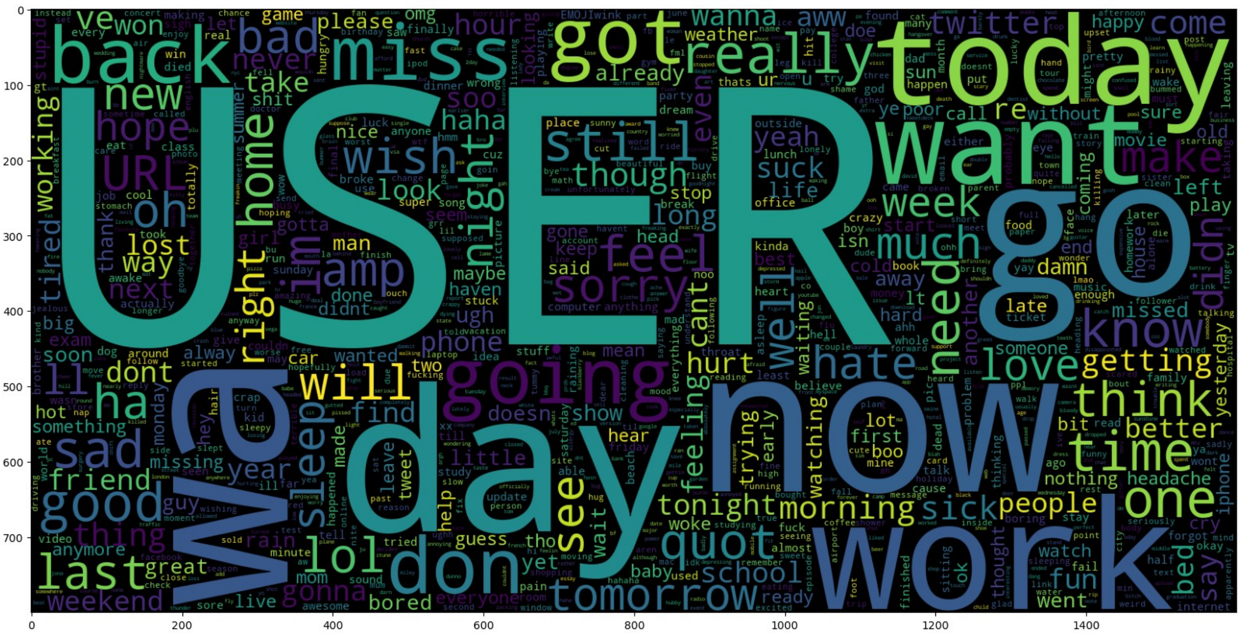
Preprocessing Text

```
# setting uo the the symbol emoji meaning to the actula emoji meaning
emojis = {':)': 'smile', ':-)': 'smile', ';d': 'wink', ':-E': 
'vampire', ':(': 'sad', 
          ':-(': 'sad', ':-<': 'sad', ':P': 'raspberry', ':O': 
'surprised',
          ':-@': 'shocked', ':@': 'shocked',':-$': 'confused', ':\\': 
'annoyed', 
          ':#': 'mute', ':X': 'mute', ':^)': 'smile', ':-&': 
'confused', '$_$': 'greedy',
          '@@': 'eyeroll', ':-!': 'confused', ':-D': 'smile', ':-0': 
'yell', 'O.o': 'confused',
          '<(-_-)>': 'robot', 'd[-_-]b': 'dj', ":'-)": 'sadsmile', 
';)': 'wink', 
          ';-)': 'wink', 'O:-)': 'angel','O*-)': 'angel','(:-D': 
'gossip', '=^.^=': 'cat'}

## Defining set containing all stopwords in english.
stopwordlist = ['a', 'about', 'above', 'after', 'again', 'ain', 'all', 
'am', 'an',
             'and','any','are', 'as', 'at', 'be', 'because', 'been',
'before',
             'being', 'below', 'between','both', 'by', 'can', 'd',
'did', 'do',
             'does', 'doing', 'down', 'during', 'each','few', 'for',
'from',
             'further', 'had', 'has', 'have', 'having', 'he', 'her',
'here',
             'hers', 'herself', 'him', 'himself', 'his', 'how', 'i',
'if', 'in',
             'into','is', 'it', 'its', 'itself', 'just', 'll', 'm',
'ma',
             'me', 'more', 'most','my', 'myself', 'now', 'o', 'of',
'on', 'once',
             'only', 'or', 'other', 'our', 'ours','ourselves', 'out',
'own', 're',
             's', 'same', 'she', "shes", 'should', "shouldve",'so',
'some', 'such',
             't', 'than', 'that', "thatll", 'the', 'their', 'theirs',
'them',
             'themselves', 'then', 'there', 'these', 'they', 'this',
'those',
             'through', 'to', 'too','under', 'until', 'up', 've',
'very', 'was',
             'we', 'were', 'what', 'when', 'where','which','while',
'who', 'whom',
             'why', 'will', 'with', 'won', 'y', 'you', "youd","youll",
"youre",
             "youve", 'your', 'yours', 'yourself', 'yourselves']

def preprocess(textdata):
    processedText = []

    # Create Lemmatizer and Stemmer.
    wordLemm = WordNetLemmatizer()

    # Defining regex patterns.
    urlPattern        = r"((http://)[^ ]*|(https://)[^ ]*|( www\.)
[^ ]*)"
    userPattern       = '@[^\s]+'
    alphaPattern      = "[^a-zA-Z0-9]"
    sequencePattern   = r"(.)\1\1+"
    seqReplacePattern = r"\1\1"

    for tweet in textdata:
        tweet = tweet.lower()

        # Replace all URls with 'URL'
        tweet = re.sub(urlPattern,' URL',tweet)
        # Replace all emojis.
        for emoji in emojis.keys():
            tweet = tweet.replace(emoji, "EMOJI" + emojis[emoji])

        # Replace @USERNAME to 'USER'.
        tweet = re.sub(userPattern,' USER', tweet)
        # Replace all non alphabets.
        tweet = re.sub(alphaPattern, " ", tweet)
        # Replace 3 or more consecutive letters by 2 letter.
        tweet = re.sub(sequencePattern, seqReplacePattern, tweet)

        tweetwords = ''
        for word in tweet.split():
            # Checking if the word is a stopword.
            #if word not in stopwordlist:
            if len(word)>1:
                # Lemmatizing the word.
                word = wordLemm.lemmatize(word)
                tweetwords += (word+' ')

        processedText.append(tweetwords)

    return processedText

processedtext = preprocess(text)
```

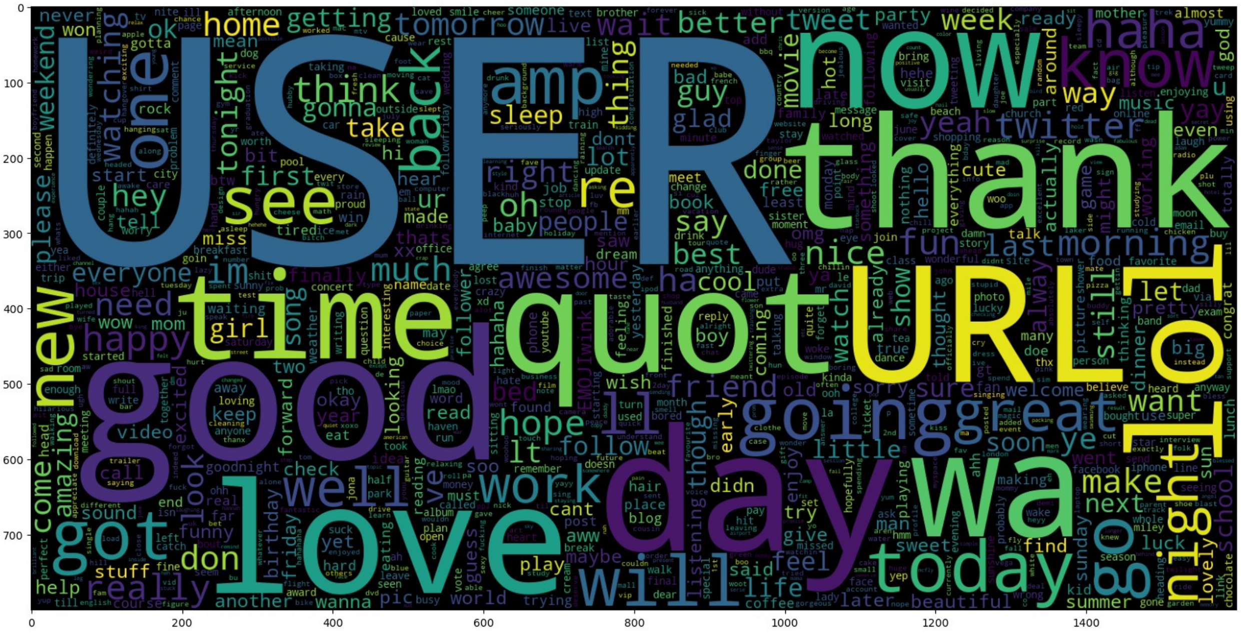
DATA ANALYSIS

```
data_neg = processedtext[:800000]
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(data_neg))
plt.imshow(wc)
```

```
<matplotlib.image.AxesImage at 0x1dede59c750>
```



```
data_pos = processedtext[800000:]
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
              collocations=False).generate(" ".join(data_pos))
plt.figure(figsize = (20,20))
plt.imshow(wc)

<matplotlib.image.AxesImage at 0x1de8a80f290>
```



Data splitting

95% of data will be used for training and 5% for testing

```
X_train, X_test, y_train, y_test = train_test_split(processedtext, 
sentiment,
                                                    test_size = 0.05, 
random_state = 0)
print(f'Dthas been sucessfully separated for testing and trainfge.')

Dthas been sucessfully separated for testing and trainfge.
```

TF-IDF Vectorization

```
vectoriser = TfidfVectorizer(ngram_range=(1,2), max_features=500000)
vectoriser.fit(X_train)
print(f'fitting a vectorizer')

fitting a vectorizer

X_train = vectoriser.transform(X_train)
X_test  = vectoriser.transform(X_test)
print(f'Data Transformed.')

Data Transformed.
```

Genearating and evaluation of the model

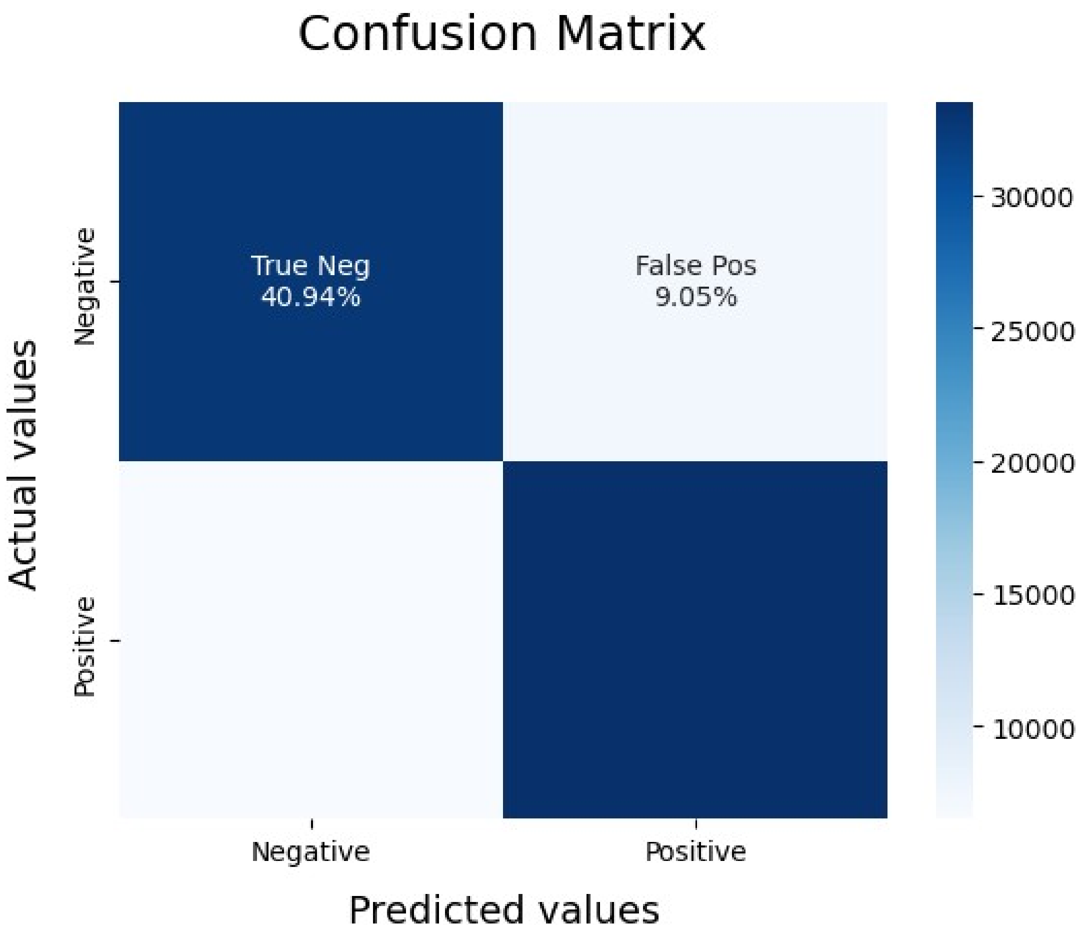
we will use logistic regression as the model

```
def model_Evaluate(model):
    
    # Predict values for Test dataset
    y_pred = model.predict(X_test)
    
    # Print the evaluation metrics for the dataset.
    print(classification_report(y_test, y_pred))
    
    # Compute and plot the Confusion matrix
    cf_matrix = confusion_matrix(y_test, y_pred)

    categories  = ['Negative','Positive']
    group_names = ['True Neg','False Pos', 'False Neg','True Pos']
    group_percentages = ['{0:.2%}'.format(value) for value in 
cf_matrix.flatten() / np.sum(cf_matrix)]

    labels = [f'{v1}\n{v2}' for v1, v2 in 
zip(group_names,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '',
                xticklabels = categories, yticklabels = categories)

    plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad =
10)
    plt.ylabel("Actual values"   , fontdict = {'size':14}, labelpad =
10)
    plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
```

Logistic Regression model

```
LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel)
```

```
              precision    recall  f1-score   support

           0       0.83      0.82      0.83     39989
           1       0.82      0.84      0.83     40011

    accuracy                           0.83     80000
   macro avg       0.83      0.83      0.83     80000
weighted avg       0.83      0.83      0.83     80000
```



Saving and using the model

```
file = open('Sentiment-LR.pickle','wb')
pickle.dump(LRmodel, file)
file.close()
```

Based on coustom input

```
def load_models():
    '''
    Replace '..path/' by the path of the saved models.
    '''

    # Load the vectoriser.
    file = open('..path/vectoriser-ngram-(1,2).pickle', 'rb')
    vectoriser = pickle.load(file)
    file.close()
    # Load the LR Model.
    file = open('..path/Sentiment-LRv1.pickle', 'rb')
    LRmodel = pickle.load(file)
    file.close()
```

```python
    return vectoriser, LRmodel

def predict(vectoriser, model, text):
    # Predict the sentiment
    textdata = vectoriser.transform(preprocess(text))
    sentiment = model.predict(textdata)

    # Make a list of text with sentiment.
    data = []
    for text, pred in zip(text, sentiment):
        data.append((text,pred))

    # Convert the list into a Pandas DataFrame.
    df = pd.DataFrame(data, columns = ['text','sentiment'])
    df = df.replace([0,1], ["Negative","Positive"])
    return df

if __name__=="__main__":
    # Loading the models.
    #vectoriser, LRmodel = load_models()

    # Text to classify should be in a list.
    text = ["hurray its a great deal",
            "i feel bad today",
            "finally the project works good"]

    df = predict(vectoriser, LRmodel, text)
    print(df.head())
```

```
                             text sentiment
0         hurray its a great deal  Positive
1                i feel bad today  Negative
2  finally the project works good  Positive
```