NLP Assignment

Assignment Submitted by: Group 6

Member Details:

1. Sumanta Kumar Patel - 2022OG04032

2. Rahul Khandpur - 2022OG04037

Context

This is the sentiment140 dataset. It contains 1,600,000 tweets extracted using the twitter api . The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiment .

Using the data set we need to create a ML model using NLP for text processing to predict Tweeter Sentiments

As the dataset is very huge, we are using Kaggle API to download the dataset directly into the worspaces directory

In [ ]:
```
! pip install kaggle
```

In [ ]:
```
import shutil
import os

# Source path of downloaded kaggle.json file
source_path = 'kaggle.json'

# Destination directory for Kaggle API file
destination_directory = os.path.join(os.path.expanduser('~'), '.kaggle')

# Create the directory if it doesn't exist
os.makedirs(destination_directory, exist_ok=True)

# Move the kaggle.json file to the destination directory
shutil.move(source_path, os.path.join(destination_directory, 'kaggle.json'))

# Set permissions for the file (if necessary)
os.chmod(os.path.join(destination_directory, 'kaggle.json'), 0o600)

print('kaggle.json configuration completed')
```

In [ ]:
```
! kaggle datasets download -d kazanova/sentiment140
```

In [ ]:
```
# Extract the compressed dataset
from zipfile import ZipFile
dataset = 'sentiment140.zip'

with ZipFile(dataset, 'r') as compfile:

```

```
  compfile.extractall()
  print('The dataset is extracted')
```

```
The dataset is extracted
```

In [10]:
```
# Importing important libraries

import numpy as np
import pandas as pd
import re # library for using regular expressions
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer # from stemming a word - reducting wo
from sklearn.feature_extraction.text import TfidfVectorizer # converting text da
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
print('All important libraries are imported successfully')
```

```
All important libraries are imported successfully
```

In [ ]:
```
# Importing all the stop words for NLP
import nltk
nltk.download('stopwords')
```

What are stop words?

Stopwords are the words in any language which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For some search engines, these are some of the most common, short function words, such as the, is, at, which, and on. In this case, stop words can cause problems when searching for phrases that include them, particularly in names such as “The Who” or “Take That”.

In [12]:
```
# Print stop words in english
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "y
ou've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'hi
m', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
s', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which',
'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'w
as', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'unt
il', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'in
to', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'u
p', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'the
n', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'onl
y', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just',
'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've',
'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
oesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "sha
n't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "wo
n't", 'wouldn', "wouldn't"]
```

Data Processing

In [58]:
```
# Loading the data from csv to pandas dataframe
twitter_data = pd.read_csv('training.1600000.processed.noemoticon.csv', encoding
```

In [59]:
```
twitter_data.shape
```

Out[59]:
```
(1599999, 6)
```

In [60]:
```
twitter_data.head(5)
```

Out[60]:

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

As we see there is no header in the data, the first row is considers as header, so we need to add correct header for each column of the data.

In [61]:
```
# Naming the columns and reading the datasset again
column_names = ['target', 'id', 'date', 'flag', 'user', 'text']
twitter_data = pd.read_csv('training.1600000.processed.noemoticon.csv', names =
```

In [62]:
```
twitter_data.shape
```

Out[62]:
```
(1600000, 6)
```

In [63]:
```
twitter_data.head(5)
```

Out[63]:

	target	id	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

In [64]:
```
# Checking the missing values in dataset

twitter_data.isnull().sum()
```

Out[64]:
```
target    0
id        0
date      0
flag      0
user      0
text      0
dtype: int64
```

In [65]:
```
# Checking the disctribution of target column, 0 = negative, 4 = positive)
twitter_data['target'].value_counts()
```

Out[65]:
```
target
0    800000
4    800000
Name: count, dtype: int64
```

Converting the Target "4" to "1"

In [66]:
```
twitter_data.replace({'target':{4:1}}, inplace = True)
```

In [67]:
```
# Checking the disctribution of target column
twitter_data['target'].value_counts()
```

Out[67]:
```
target
0    800000
1    800000
Name: count, dtype: int64
```

Now

- 0 --> negative Tweet
- 1 --> positive Tweet

Stemming :

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words **“chocolates”, “chocolatey”, “choco” to the root word, “chocolate”** and **“retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”**. Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words. How do we get these tokenized words? Well, tokenization involves breaking down the document into different words.

Stemming is a natural language processing technique that is used to reduce words to their base form, also known as the root form. The process of stemming is used to normalize text and make it easier to process. It is an important step in text pre-processing, and it is commonly used in information retrieval and text mining applications.

Preprocessing, before Stemming the Text

In [68]:
```
#Preprocessing, before Stemming the Text
portStem = PorterStemmer()

def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]',' ', content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [portStem.stem(word) for word in stemmed_content if not wo
    stemmed_content = ' '.join(stemmed_content)

    return stemmed_content
```

In [69]:
```
# Testing the Stemming function before applying to whole data
i = 4
print("Original Text: ", twitter_data['text'][i])
print("Stemmed Text: ", stemming(twitter_data['text'][i]))
```

```
Original Text:  @nationwideclass no, it's not behaving at all. i'm mad. why am i
here? because I can't see you all over there.
Stemmed Text:  nationwideclass behav mad see
```

In [71]:
```
twitter_data['stemmed_content'] = twitter_data['text'].apply(stemming)
```

In [72]:
```
#stemmed_data = twitter_data.to_csv('stemmed_data.csv', index=False)
```

In [73]:
```
print(twitter_data['stemmed_content'])
```

```
0          switchfoot http twitpic com zl awww bummer sho...
1          upset updat facebook text might cri result sch...
2          kenichan dive mani time ball manag save rest g...
3                           whole bodi feel itchi like fire
4                            nationwideclass behav mad see
                                ...
1599995                       woke school best feel ever
1599996    thewdb com cool hear old walt interview http b...
1599997                      readi mojo makeov ask detail
1599998    happi th birthday boo alll time tupac amaru sh...
1599999    happi charitytuesday thenspcc sparkschar speak...
Name: stemmed_content, Length: 1600000, dtype: object
```

Separating data and label

In [89]:
```
X = twitter_data['stemmed_content'].values
Y  = twitter_data['target'].values
```

In [90]:
```
print(X)
```

```
['switchfoot http twitpic com zl awww bummer shoulda got david carr third day'
 'upset updat facebook text might cri result school today also blah'
 'kenichan dive mani time ball manag save rest go bound' ...
 'readi mojo makeov ask detail'
 'happi th birthday boo alll time tupac amaru shakur'
 'happi charitytuesday thenspcc sparkschar speakinguph h']
```

In [91]:
```
print(Y)
```

```
[0 0 0 ... 1 1 1]
```

Splitting the data into training and test data

In [103…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratif
```

In [104…
```
print("Shape of X, X_train and X_test: ", X.shape, X_train.shape, X_test.shape)
print("Shape of Y, Y_train and Y_test: ", Y.shape, Y_train.shape, Y_test.shape)
```

```
Shape of X, X_train and X_test:  (1600000,) (1280000,) (320000,)
Shape of Y, Y_train and Y_test:  (1600000,) (1280000,) (320000,)
```

Converting textual data into numeric data with Vectorization Technique

In [105…
```
vectorizer = TfidfVectorizer()

X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

In [126…
```
import pickle
```

In [127…
```
X_train_vectorized = X_train
```

In [130…
```
# Saving the trained vectorizer to be used in the application.
with open('trained_vectorizer.pkl', 'wb') as f:
```

```
    pickle.dump(vectorizer, f)
```

In [106…

```
print(X_train)
```

```
(0, 443066)   0.4484755317023172
(0, 235045)   0.41996827700291095
(0, 109306)   0.3753708587402299
(0, 185193)   0.5277679060576009
(0, 354543)   0.3588091611460021
(0, 436713)   0.27259876264838384
(1, 160636)   1.0
(2, 288470)   0.16786949597862733
(2, 132311)   0.2028971570399794
(2, 150715)   0.18803850583207948
(2, 178061)   0.1619010109445149
(2, 409143)   0.15169282335109835
(2, 266729)   0.24123230668976975
(2, 443430)   0.3348599670252845
(2, 77929)    0.31284080750346344
(2, 433560)   0.3296595898028565
(2, 406399)   0.32105459490875526
(2, 129411)   0.29074192727957143
(2, 407301)   0.18709338684973031
(2, 124484)   0.1892155960801415
(2, 109306)   0.4591176413728317
(3, 172421)   0.37464146922154384
(3, 411528)   0.27089772444087873
(3, 388626)   0.3940776331458846
(3, 56476)    0.5200465453608686
:     :
(1279996, 390130)     0.22064742191076112
(1279996, 434014)     0.2718945052332447
(1279996, 318303)     0.21254698865277746
(1279996, 237899)     0.2236567560099234
(1279996, 291078)     0.17981734369155505
(1279996, 412553)     0.18967045002348676
(1279997, 112591)     0.7574829183045267
(1279997, 273084)     0.4353549002982409
(1279997, 5685)       0.48650358607431304
(1279998, 385313)     0.4103285865588191
(1279998, 275288)     0.38703346602729577
(1279998, 162047)     0.34691726958159064
(1279998, 156297)     0.3137096161546449
(1279998, 153281)     0.28378968751027456
(1279998, 435463)     0.2851807874350361
(1279998, 124765)     0.32241752985927996
(1279998, 169461)     0.2659980990397061
(1279998, 93795)      0.21717768937055476
(1279998, 412553)     0.2816582375021589
(1279999, 96224)      0.5416162421321443
(1279999, 135384)     0.6130934129868719
(1279999, 433612)     0.3607341026233411
(1279999, 435572)     0.31691096877786484
(1279999, 31410)      0.248792678366695
(1279999, 242268)     0.19572649660865402
```

Training the Machine Learning Model with Logistic Regression

In [108…

```
model = LogisticRegression(max_iter=1000)
```

In [109…
```
model.fit(X_train, Y_train)
```

Out[109…
```
LogisticRegression(max_iter=1000)
```

Model Evaluation - Accuracy Score

In [110…
```
# Accuracy Score on the Training Data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

In [111…
```
print("Accuracy score on the Training Data: ", training_data_accuracy)
```

```
Accuracy score on the Training Data:  0.8102109375
```

In [112…
```
# Accuracy Score on the Test Data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

In [113…
```
print("Accuracy score on the Test Data: ", test_data_accuracy)
```

```
Accuracy score on the Test Data:  0.77800625
```

Model Accuracy is 77.8% which is good.

Saving the Trained Model

In [114…
```
import pickle
```

In [115…
```
filename = 'twitter_sentiment_analysis_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

Using Trained Model for Further Prediction

In [121…
```
# loading the saved model
loaded_model = pickle.load(open('twitter_sentiment_analysis_model.sav', 'rb'))
```

In [122…
```
X_new = X_test[200]
print(Y_test[200])

prediction = loaded_model.predict(X_new)
print(prediction)

if (prediction[0] == 0):
    print("Negative Tweet")
else:
    print("Positive Tweet")
```

```
1
[1]
Positive Tweet
```