In [ ]:
```
#importing libraries
```

In [1]:
```
import pandas as pd
import numpy as np
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [2]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\konga\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[2]:
```
True
```

In [3]:
```
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "yo
u've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who',
'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'we
re', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did',
'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'wh
ile', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'thr
ough', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'he
re', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few',
'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'sam
e', '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', "doesn't", 'hadn', "had
n't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "might
n't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [ ]:
```
#DATA COLLECTION AND PREPROCESSING
```

In [4]:
```
#twitter data as td
td=pd.read_csv('training.1600000.processed.noemoticon.csv', encoding = 'ISO-8859-1'
```

In [5]:
```
td.shape
```

Out[5]:
```
(1048575, 6)
```

In [6]:
```
td.head(100)
```

Out[6]:

	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
...	...	...	...	...	...	...
95	0	1467836500	Mon Apr 06 22:26:28 PDT 2009	NO_QUERY	natalieantipas	so rylee,grace...wana go steve's party or not?...
96	0	1467836576	Mon Apr 06 22:26:29 PDT 2009	NO_QUERY	timdonnelly	hey, I actually won one of my bracket pools! T...
97	0	1467836583	Mon Apr 06 22:26:29 PDT 2009	NO_QUERY	homeworld	@stark YOU don't follow me, either and i work...
98	0	1467836859	Mon Apr 06 22:26:33 PDT 2009	NO_QUERY	willy_chaz	A bad nite for the favorite teams: Astros and ...
99	0	1467836873	Mon Apr 06 22:26:33 PDT 2009	NO_QUERY	LeakySpoon	Body Of Missing Northern Calif. Girl Found: P...

100 rows × 6 columns

In [7]:
```
column_names = ['target','id','date','flag','user','text']
td=pd.read_csv('training.1600000.processed.noemoticon.csv', names=column_names,enco
```

In [9]:
```
print(td.shape)
print(td.head())
```

```
print(td.isnull().sum())
```

```
(1600000, 6)
   target          id                          date      flag  \
0       0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY
1       0  1467810672  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY
2       0  1467810917  Mon Apr 06 22:19:53 PDT 2009  NO_QUERY
3       0  1467811184  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY
4       0  1467811193  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY

              user                                                text
0  _TheSpecialOne_  @switchfoot http://twitpic.com/2y1zl - Awww, t...
1    scotthamilton  is upset that he can't update his Facebook by ...
2         mattycus  @Kenichan I dived many times for the ball. Man...
3          ElleCTF    my whole body feels itchy and like its on fire
4           Karoli  @nationwideclass no, it's not behaving at all....
target    0
id        0
date      0
flag      0
user      0
text      0
dtype: int64
```

In [9]:
```
td1=td.head(100000)
```

In [10]:
```
td1.shape
```

Out[10]:
```
(100000, 6)
```

In [11]:
```
td1['target'].value_counts()
```

Out[11]:
```
target
1    50001
0    49999
Name: count, dtype: int64
```

In [12]:
```
td1.shape
```

Out[12]:
```
(100000, 6)
```

In [ ]:
```
#0--NEGATIVE TWEET
#1--POSITIVE TWEET
```

In [13]:
```
port_stem=PorterStemmer()
```

In [14]:
```
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]',' ',content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content
```

In [15]:
```
td1['stemmed_content'] = td1['text'].apply(stemming)
```

```
C:\Users\konga\AppData\Local\Temp\ipykernel_34948\3868112615.py:1: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  td1['stemmed_content'] = td1['text'].apply(stemming)
```

In [16]:
```
td1.head()
```

Out[16]:

	target	id	date	flag	user	text	stemmed_cont
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...	switchfoot h twitpic cor awww bum sh
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...	upset up facebook might cri re s
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...	kenichan d mani time manag save
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire	whole bodi itchi like
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....	nationwidec behav mad

In [17]:
```
print(td1['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
                               ...
99995          son develop new habit wake second pot coffe
99996                    look like router broke tweet fone
99997             realli dont want colleg right wish sunni
99998                                    flossa offer pepto
99999                          josiehobo would sooooo revis
Name: stemmed_content, Length: 100000, dtype: object
```

In [18]:
```
X = td1['stemmed_content'].values
Y = td1['target'].values
```

In [19]:
```
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' ...
 'realli dont want colleg right wish sunni' 'flossa offer pepto'
 'josiehobo would sooooo revis']
```

In [20]:
```
print(Y)
```

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

In [43]:
```
#SPLITTING THE DATA IN TO TRAINING AND TESTING
```

In [21]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,stratify=Y,random_state=2)
```

In [22]:
```
print(X.shape,X_train.shape,X_test.shape)
```

```
(100000,) (75000,) (25000,)
```

In [23]:
```
print(X_train)
```

```
['give alway give upp' 'tweetchild clean'
 'fuck gumbal ralli start la saturday want see crazi car' ...
 'sit front comput roleplay make photo edit fanfic bad case carpal tunnel syndrom'
 'omw eye r sooooo friggn sore agh afrikaan h w murder nd write english stori gt h
andwritten eep'
 'beetleginni littl fella mind nice laff']
```

In [24]:
```
print(X_test)
```

```
['khloekardashian wish love music googl autopsi photo gone'
 'whatdenni hate suck' 'food make sick' ...
 'mussomitchel hi mitchel pleas pleas pleas respond die'
 'great day red hill aunti except cousin toy poodl disloc shoulder trip'
 'darnit stop cough']
```

In [25]:
```
vectorizer=TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

In [26]:
```
print(X_train)
```

```
  (0, 48499)    0.5914785367531842
  (0, 1422)     0.3498929075857496
  (0, 17350)    0.7264489615805795
  (1, 8602)     0.5067925204843275
  (1, 47637)    0.8620680606432083
  (2, 6998)     0.2638523201766879
  (2, 9752)     0.2936658345469015
  (2, 40329)    0.20498206745716183
  (2, 49660)    0.17735372050156378
  (2, 39834)    0.2772629124565199
  (2, 25355)    0.31146858580770104
  (2, 43267)    0.22806908581046179
  (2, 37274)    0.46831071031870003
  (2, 18254)    0.5087323462798994
  (2, 16409)    0.25815481514101907
  (3, 17517)    0.1121765380684488
  (3, 49563)    0.1733580284283431
  (3, 6927)     0.1753883787303968
  (3, 4779)     0.18841117057765255
  (3, 43291)    0.2724021684048053
  (3, 29343)    0.314004012296871
  (3, 26448)    0.1339179016150534
  (3, 41353)    0.23887111866159058
  (3, 5948)     0.2796380193577574
  (3, 28039)    0.3645977088468043
  :     :
  (74997, 9140) 0.2127233603185336
  (74997, 41620)        0.2102464069164804
  (74997, 27967)        0.1553230738092831
  (74997, 3459) 0.15661277738864654
  (74998, 18653)        0.3390364916192929
  (74998, 639)  0.3390364916192929
  (74998, 16289)        0.3390364916192929
  (74998, 33539)        0.3067380075873912
  (74998, 31181)        0.2744395235554896
  (74998, 707)  0.2672330833232972
  (74998, 13330)        0.29481759812646896
  (74998, 42518)        0.23058584404028815
  (74998, 13884)        0.20379486020225687
  (74998, 43655)        0.21114116047187106
  (74998, 50993)        0.18454622082327407
  (74998, 14696)        0.189014252410759
  (74998, 31825)        0.21269910626057903
  (74998, 18179)        0.19537415216447873
  (74998, 42574)        0.18694423537845084
  (74999, 25464)        0.5322925217938426
  (74999, 4150) 0.5322925217938426
  (74999, 15188)        0.44247431322022146
  (74999, 29777)        0.3188156698474346
  (74999, 26726)        0.26003170423643196
  (74999, 32218)        0.26131572106673
```

In [27]:
```
print(X_test)
```

```
  (0, 50588)    0.2383741719247849
  (0, 35283)    0.3809268554775
  (0, 31205)    0.3522040242698775
  (0, 27260)    0.25312270150860755
  (0, 24505)    0.5790570424327512
  (0, 17677)    0.4173114171694695
  (0, 17626)    0.31701392857937455
  (1, 43946)    0.7394987573882745
  (1, 18860)    0.6731579219033212
  (2, 41329)    0.5314170749325476
  (2, 27967)    0.49685502635012946
  (2, 15850)    0.6860983714167721
  (3, 46113)    0.7096717702564034
  (3, 42574)    0.7045324538310089
  (4, 47217)    0.3406293148758098
  (4, 42098)    0.5509687513130553
  (4, 17350)    0.41758643545300117
  (4, 13130)    0.5252248950071067
  (4, 6927)     0.3607845389068098
  (5, 41205)    0.2679255688551221
  (5, 38456)    0.16386704126958204
  (5, 38018)    0.24898044698971505
  (5, 32069)    0.7536055592838043
  (5, 19011)    0.1810362993958573
  (5, 18261)    0.331506681245997
  :     :
  (24996, 32514)        0.4946421317624696
  (24996, 25687)        0.3248518971034589
  (24996, 21647)        0.5946476285941127
  (24996, 17517)        0.23796406034721793
  (24997, 38154)        0.32947037807336305
  (24997, 35674)        0.680291138141919
  (24997, 31250)        0.39133404012770934
  (24997, 30100)        0.3841435742871459
  (24997, 19464)        0.27378328605925706
  (24997, 11709)        0.23018977601879542
  (24998, 47286)        0.24287231149167876
  (24998, 46982)        0.31004490793931344
  (24998, 41205)        0.29323709166310263
  (24998, 37751)        0.25248974228331095
  (24998, 35918)        0.367600073595316
  (24998, 19540)        0.28278999079728323
  (24998, 17943)        0.19652236000040044
  (24998, 14552)        0.256829732148761
  (24998, 11937)        0.3796326690547401
  (24998, 10853)        0.1314431298288539
  (24998, 9599) 0.2735914348981612
  (24998, 3033) 0.3731126614652171
  (24999, 43642)        0.4115636897980651
  (24999, 10680)        0.7580290741639026
  (24999, 9528) 0.5059715920504024
```

In [44]:
```
#TRAING MACHINE MODEL
```

In [45]:
```
#LOGISTIC REGRESSION
```

In [28]:
```
model = LogisticRegression(max_iter=10000)
```

In [29]:
```
model.fit(X_train, Y_train)
```

Out[29]:
```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [46]:
```
#MODEL EVAULATION
```

In [30]:
```
X_train_prediction = model.predict(X_train)
training_data_accuracy=accuracy_score(Y_train, X_train_prediction)
```

In [31]:
```
print('Acuuracy score on training data:', training_data_accuracy)
```

```
Acuuracy score on training data: 0.7438533333333334
```

In [32]:
```
X_test_prediction = model.predict(X_test)
test_data_accuracy=accuracy_score(Y_test, X_test_prediction)
```

In [33]:
```
print('Acuuracy score on training data:', test_data_accuracy)
```

```
Acuuracy score on training data: 0.55524
```

In [35]:
```
#saving trainig data
```

In [34]:
```
import pickle
```

In [36]:
```
filename = 'trained_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

In [37]:
```
#using saved model for future prediction
```

In [39]:
```
loaded_model = pickle.load(open('trained_model.sav', 'rb'))
```

In [41]:
```
X_new = X_test[200]
print(Y_test[200])

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

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

```
0
[0]
Negative Tweet
```

In [42]:
```
X_new = X_test[3]
print(Y_test[3])

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

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

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

In [ ]: