```
#install kaggle library
! pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-package
```

```
#configuring the path of kaggle.json file
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Importing Twitter Sentiment Analysis Dataset

```
#API to fetch the dataset from kggle
!kaggle datasets download -d kazanova/sentiment140
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (s
Dataset URL: https://www.kaggle.com/datasets/kazanova/sentiment140
License(s): other
Downloading sentiment140.zip to /content
 96% 78.0M/80.9M [00:00<00:00, 216MB/s]
100% 80.9M/80.9M [00:00<00:00, 201MB/s]
```

```
#Extracting the compressed dataset

from zipfile import ZipFile
dataset='/content/sentiment140.zip'

with ZipFile(dataset,'r')as zip:
  zip.extractall()
  print ("The Dataset is Extracted")
```

```
The Dataset is Extracted
```

Importing the Dependencies

```
import numpy as np
import pandas as pd
import re #regular expression
from nltk.corpus import stopwords #natural language tool kit
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer #convert text data to numeric
from sklearn.model_selection import train_test_split #splitting the data
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
import nltk
nltk.download('stopwords')
```

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

```
#printing stopwords in englis
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you'v
```

Data Preprocessing

```
#loading the data from csv file to pandas dataframe
twitter_data=pd.read_csv('/content/training.1600000.processed.noemoticon.csv',encoding='I
```

```
#checking the number of rows and columns
twitter_data.shape
```

```
(1599999, 6)
```

```
#print the first 5 rows of data frame
twitter_data.head()
```

	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	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...

```
#Naming the columns and reading the dataset again

column_names=['Target','Id','Date','Flag','User','Text']
twitter_data=pd.read_csv('/content/training.1600000.processed.noemoticon.csv',names=colum


twitter_data.shape
```

```
(1600000, 6)
```

```
twitter_data.head()
```

	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	NO_QUERY	scotthamilton	is upset that he can't

```
#Counting the number of missing values in the dataset
twitter_data.isnull().sum()
```

```
Target    0
Id        0
Date      0
Flag      0
User      0
Text      0
dtype: int64
```

```
#checking the distribution of target column
twitter_data['Target'].value_counts()
```

```
Target
0    800000
4    800000
Name: count, dtype: int64
```

Convert the target "4" to "1"

```
twitter_data.replace({'Target':{4:1}},inplace=True)
twitter_data['Target'].value_counts()
```

```
Target
0    800000
1    800000
Name: count, dtype: int64
```

0---> Negative Tweet

1---> Positive Tweet

Stemming

stemming is the process of reducing a word to its root word

example : actor, actress, acting (all are similar) = act

```
 port_stem = PorterStemmer()
```

```
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 in stopwo
  stemmed_content=' '.join(stemmed_content)

  return stemmed_content
```

```
twitter_data['stemmed_content']=twitter_data['Text'].apply(stemming)
```

```
twitter_data.head()
```

	Target	Id	Date	Flag	User	Text	stem
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...	a
1	0	1467810672	Mon Apr 06	NO_QUERY	scottthamilton	is upset that he can't	

```
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
```

```
print(twitter_data['Target'])
```

```
0          0
1          0
2          0
3          0
4          0
          ..
1599995    1
1599996    1
1599997    1
1599998    1
1599999    1
Name: Target, Length: 1600000, dtype: int64
```

```
#separating the data and label
X= twitter_data['stemmed_content'].values
Y= twitter_data['Target'].values
```

```
print(X)
print(Y)
```

```
['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']
```

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

Splitting the data to Training data and Test data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, rand
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(1600000,) (1280000,) (320000,)
```

```
print(X_train)
```

```
['watch saw iv drink lil wine' 'hatermagazin'
 'even though favourit drink think vodka coke wipe mind time think im gonna find new
 ... 'eager monday afternoon'
 'hope everyon mother great day wait hear guy store tomorrow'
 'love wake folger bad voic deeper']
```

```
print(X_test)
```

```
['mmangen fine much time chat twitter hubbi back summer amp tend domin free time'
 'ah may show w ruth kim amp geoffrey sanhueza'
 'ishatara mayb bay area thang dammit' ...
 'destini nevertheless hooray member wonder safe trip' 'feel well'
 'supersandro thank']
```

```
#converting te textual data to numerical data

vectorizer = TfidfVectorizer()
X_train1 = vectorizer.fit_transform(X_train)
X_test1 = vectorizer.transform(X_test)
```

```
print(X_train1)
```

```
  (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
```

```
print(X_test1)
```

```
  (0, 420984)   0.17915624523539803
  (0, 409143)   0.31430470598079707
  (0, 398906)   0.3491043873264267
  (0, 388348)   0.21985076072061738
  (0, 279082)   0.1782518010910344
  (0, 271016)   0.4535662391658828
  (0, 171378)   0.2805816206356073
  (0, 138164)   0.23688292264071403
  (0, 132364)   0.25525488955578596
  (0, 106069)   0.3655545001090455
  (0, 67828)    0.26800375270827315
  (0, 31168)    0.16247724180521766
  (0, 15110)    0.1719352837797837
  (1, 366203)   0.24595562404108307
  (1, 348135)   0.4739279595416274
  (1, 256777)   0.28751585696559306
  (1, 217562)   0.40288153995289894
  (1, 145393)   0.575262969264869
  (1, 15110)    0.211037449588008
  (1, 6463)     0.30733520460524466
  (2, 400621)   0.4317732461913093
  (2, 256834)   0.2564939661498776
  (2, 183312)   0.5892069252021465
```

```
(2, 89448)    0.36340369428387626
(2, 34401)    0.37916255084357414
:     :
(319994, 123278)      0.4530341382559843
(319995, 444934)      0.3211092817599261
(319995, 420984)      0.22631428606830145
(319995, 416257)      0.23816465111736276
(319995, 324496)      0.3613167933647574
(319995, 315813)      0.28482299145634127
(319995, 296662)      0.39924856793840147
(319995, 232891)      0.25741278545890767
(319995, 213324)      0.2683969144317078
(319995, 155493)      0.2770682832971668
(319995, 109379)      0.30208964848908326
(319995, 107868)      0.3339934973754696
(319996, 438709)      0.4143006291901984
(319996, 397506)      0.9101400928717545
(319997, 444770)      0.2668297951055569
(319997, 416695)      0.29458327588067873
(319997, 349904)      0.32484594100566083
(319997, 288421)      0.48498483387153407
(319997, 261286)      0.37323893626855326
(319997, 169411)      0.403381646999604
(319997, 98792)       0.4463892055808332
(319998, 438748)      0.719789181620468
(319998, 130192)      0.6941927210956169
(319999, 400636)      0.2874420848216212
(319999, 389755)      0.9577980203954275
```

Training the Machine Learning Model

LOGISTIC REGRESSION

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

```
model.fit(X_train1, Y_train)
```

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

Model Evaluation

Accuracy Score

```
# accuracy score on the training data
X_train_prediction = model.predict(X_train1)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
print("Accuracy score on the training data :", training_data_accuracy)
```

```
Accuracy score on the training data : 0.81018984375
```

```
# accuracy score on the testing data
X_test_prediction = model.predict(X_test1)
testing_data_accuracy = accuracy_score(Y_test,X_test_prediction )


print("Accuracy score on the testing data :", testing_data_accuracy)
```

```
Accuracy score on the testing data : 0.7780375
```

MODEL ACCURACY = 77.8%

Saving the trained model

```
import pickle


filename = 'trained_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

Using the saved model for future predictions

```
#loading the saved model
loaded_model = pickle.load(open('/content/trained_model.sav','rb'))


X_new = X_test1[3]
print(Y_test[3])

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

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

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

```
X_new = X_test1[200]
print(Y_test[200])

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

if (prediction[0]==0):
```