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

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.16)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.2.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.7)
```

```
#Configuring path to json file
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

```
#Fetching API dataset from kaggle
!kaggle datasets download -d kazanova/sentiment140
```

```
Downloading sentiment140.zip to /content
 90% 73.0M/80.9M [00:00<00:00, 140MB/s]
100% 80.9M/80.9M [00:00<00:00, 129MB/s]
```

```
#extracting the file from compressed dataset
from zipfile import ZipFile
dataset = '/content/sentiment140.zip'

with ZipFile(dataset,'r') as zip: # opening in reading mode
  zip.extractall()
  print("dataset extracted successfully")
```

```
dataset extracted successfully
```

Adding some Dependencies

```
import numpy as np
import pandas as pd
import re  # regurlar expression
from nltk.corpus import stopwords #natural language toolkit (nltk)
from nltk.stem.porter import PorterStemmer #reduce the words to its root words
from sklearn.feature_extraction.text import TfidfVectorizer # converting textual data into visual
from sklearn.model_selection import train_test_split #spliting data into train and split data
from sklearn.linear_model import LogisticRegression #training with the data
from sklearn.metrics import accuracy_score #calcuate performance and accuracy of our machine leaning model
```

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

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

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

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
```

just checking in german

```
print(stopwords.words('german'))
```

```
['aber', 'alle', 'allem', 'allen', 'aller', 'alles', 'als', 'also', 'am', 'an', 'ander', 'andere', 'anderem', 'anderen', 'anderer', 'and
```

```
twitter_data = pd.read_csv('/content/training.1600000.processed.noemoticon.csv',encoding = 'ISO-8859-1')
```

```
twitter_data.shape
```

```
(1599999, 6)
```

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

here colums are not printed so adding the column names

```
column_names= ['target', 'id', 'date', 'flag','user', 'text']
twitter_data = pd.read_csv('/content/training.1600000.processed.noemoticon.csv',names= column_names,encoding = 'ISO-8859-1')
```

```
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 22:19:49	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by

```
#checking whether there are null values or not
twitter_data.isnull().sum()
```

```
target    0
id        0
date      0
flag      0
user      0
text      0
dtype: int64
```

Checking the distribution of target column here 0 means **negative** 2 means neutral and 4 means **positive**

```
twitter_data['target'].value_counts()
```

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

Here from the output it seems amoung 1.6 million i,e 16 lakhs tweets 8 lakhs are negative and remaining noes are positive it is distributed in **half**

here 4 and 0 seems odd so lets make **0** and **1** 0 for **negative** and 1 for *positive*

```
#converting 4 into 1
twitter_data.replace({'target': {4:1}}, inplace =True)
```

```
twitter_data['target'].value_counts()
```

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

Stemming

it is the proecss of reducing words

example *nepali*, *nepalese*, *gorkhali*, =**Nepal**

actor, *actress*, *acting* = **act**

```
port_strem = 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_strem.stem(word)for word in stemmed_content if not word in stopwords.words('english')]
  stemmed_content = ' '.join(stemmed_content)

  return stemmed_content
```

```
twitter_data['stemmed_content'] = twitter_data['text'].apply(stemming) # it took almost  4 minutes because its testing 1.6 million tweets
```

```
twitter_data.tail(10)
```

	target	id	date	flag	user	text
1599990	1	2193579249	Tue Jun 16 08:38:59 PDT 2009	NO_QUERY	razzberry5594	WOOOOO! Xbox is back
1599991	1	2193579284	Tue Jun 16 08:38:59 PDT 2009	NO_QUERY	AgustinaP	@rmedina @LaTati Mmmm That sounds absolutely ...
1599992	1	2193579434	Tue Jun 16 08:39:00 PDT 2009	NO_QUERY	sdancingsteph	ReCoVeRiNg FrOm ThE lOnG wEeKeNd

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

```
0          switchfoothttptwitpiccomyzlawwwthatsabummeryou...
1          isupsetthathecantupdatehisfacebookbytextingita...
2          kenichanidivedmanytimesfortheballmanagedtosave...
3                       mywholebodyfeelsitchyandlikeitsonfir
4          nationwideclassnoitsnotbehavingatallimmadwhyam...
```

```
                                  ...
1599995           justwokeuphavingnoschoolisthebestfeelingev
1599996    thewdbcomverycooltohearoldwaltinterviewshttpbl...
1599997         areyoureadyforyourmojomakeoveraskmefordetail
1599998     happythbirthdaytomybooofallltimetupacamarushakur
1599999    happycharitytuesdaythenspccsparkscharityspeaki...
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
```

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

```
print(X)
```

```
['switchfoothttptwitpiccomyzlawwwthatsabummeryoushouldagotdavidcarrofthirddaytodoitd'
 'isupsetthathecantupdatehisfacebookbytextingitandmightcryasaresultschooltodayalsoblah'
 'kenichanidivedmanytimesfortheballmanagedtosavetherestgooutofbound' ...
 'areyoureadyforyourmojomakeoveraskmefordetail'
 'happythbirthdaytomybooofallltimetupacamarushakur'
 'happycharitytuesdaythenspccsparkscharityspeakinguphh']
```

```
print(Y)
```

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

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2, stratify=Y, random_state=2) # 0.2 = 20%
```

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

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

```
print(X_train, X_test)
```

```
['thearmjustfellofmycomputerchairnowwhatamigoingtoleanon'
 'chrishillfanyougotmedianextlucki'
 'itsuckshavintowatchyurmotherbesickandcnthelph' ...
 'melosmoooopsididitagainsorryforthenonexistingmissingcal'
 'frowzledaccordingtojrgkachelmannitllstartrainingtomorrowoclockinthemorningletshopehesrightjakommeauchmit'
 'alcarltonlolnoideamateidgobuymoreclothesmuchsimpleropt'] ['ihatewhenthedogsbarkatsomethingwhennothinghappeneditsscarytimetotrytosleep'
 'williedaymeeitherbutiwillsupportitcuzilikeyouguy'
 'jordalynnyahitwasprettymessedup' ...
 'usedaspareusbflashdriveasvirtualramformypcnowitsgotgbofmemori'
 'sistatreensojealousimissthebeachesdownther'
 'sakurakurosakinoprobihopeyouhaveagoodoneltoohandwhatkindofcakeilovecak']
```

Now converting textual data into numerical data target is already 0 and 1 now we will do it to stemmed_content

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

```
print(X_train)
```

```
  (0, 4269)     1.0
  (1, 370478)   1.0
  (2, 259878)   1.0
  (3, 544662)   1.0
  (4, 1125692)  1.0
  (5, 183084)   1.0
```

```
  (6, 655943)   1.0
  (7, 427969)   1.0
  (8, 372293)   1.0
  (9, 414371)   1.0
  (10, 453206)  1.0
  (11, 946311)  1.0
  (12, 1165280) 1.0
  (13, 334893)  1.0
  (14, 1029544) 1.0
  (15, 1251389) 1.0
  (16, 26694)   1.0
  (17, 620642)  1.0
  (18, 371055)  1.0
  (19, 166384)  1.0
  (20, 939661)  1.0
  (21, 392755)  1.0
  (22, 288391)  1.0
  (23, 202689)  1.0
  (24, 1253472) 1.0
  :     :
  (1279975, 97244)      1.0
  (1279976, 1110776)    1.0
  (1279977, 938969)     1.0
  (1279978, 783480)     1.0
  (1279979, 800934)     1.0
  (1279980, 865418)     1.0
  (1279981, 587370)     1.0
  (1279982, 952975)     1.0
  (1279983, 412833)     1.0
  (1279984, 81684)      1.0
  (1279985, 59066)      1.0
  (1279986, 1015971)    1.0
  (1279987, 777243)     1.0
  (1279988, 1218709)    1.0
  (1279989, 450972)     1.0
  (1279990, 465151)     1.0
  (1279991, 249222)     1.0
  (1279992, 1169847)    1.0
  (1279993, 1170137)    1.0
  (1279994, 75555)      1.0
  (1279995, 1106989)    1.0
  (1279996, 561111)     1.0
  (1279997, 516692)     1.0
  (1279998, 400780)     1.0
  (1279999, 469313)     1.0
```

```
print(X_test)
```

```
  (22, 137709)  1.0
  (36, 790487)  1.0
  (55, 1232068) 1.0
  (104, 806213) 1.0
  (316, 675309) 1.0
  (317, 183110) 1.0
  (326, 530902) 1.0
  (343, 483231) 1.0
  (356, 1004786)        1.0
  (405, 294045) 1.0
  (411, 754846) 1.0
  (412, 671785) 1.0
  (503, 922733) 1.0
  (575, 484365) 1.0
  (576, 68315)  1.0
  (585, 553737) 1.0
  (597, 437993) 1.0
  (704, 437477) 1.0
  (730, 522522) 1.0
  (735, 1190472)        1.0
  (763, 201539) 1.0
  (851, 230654) 1.0
  (887, 945330) 1.0
  (905, 1080766)        1.0
  (965, 64114)  1.0
  :     :
  (319388, 469685)      1.0
  (319391, 363430)      1.0
  (319401, 69390)       1.0
  (319403, 1010168)     1.0
  (319419, 435823)      1.0
  (319442, 1154002)     1.0
  (319462, 331998)      1.0
  (319474, 363236)      1.0
  (319550, 1020181)     1.0
```

```
  (319573, 1087679)     1.0
  (319595, 377142)      1.0
  (319630, 416243)      1.0
  (319632, 324586)      1.0
  (319640, 333810)      1.0
  (319649, 564791)      1.0
  (319673, 473916)      1.0
  (319679, 363450)      1.0
  (319687, 874322)      1.0
  (319700, 937693)      1.0
  (319712, 230590)      1.0
  (319779, 553484)      1.0
  (319838, 1251413)     1.0
  (319891, 945330)      1.0
  (319931, 1137735)     1.0
  (319998, 836150)      1.0
```

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

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

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

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
print("accuracy score on training data is :", training_data_accuracy)
```

```
accuracy score on training data is : 0.99812265625
```

it shows 0.99 that means the accuracy is 99 %

```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
print("accuracy score on test data is :", test_data_accuracy)
```

```
accuracy score on test data is : 0.512653125
```

on test data its on 0.51 which is preety low its only 51 %

Model accuracy is 51 %

```
import pickle
```

Saving the data using pickle library

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

loading the saved data using future predictions

+ Code + Text

```
loaded_model = pickle.load(open('/content/trained_model.sav', 'rb'))
```

```
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")
```

```
1
[1]
positive tweet
```

Start coding or generate with AI.