```
! pip install kaggle # Installing Kaggle Library
```

```
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) (2023.11.17)
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) (4.66.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.1)
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) (3.6)
```

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

Importing Twitter Sentiment Dataset

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

```
Downloading sentiment140.zip to /content
 87% 70.0M/80.9M [00:00<00:00, 128MB/s]
100% 80.9M/80.9M [00:00<00:00, 129MB/s]
```

```
# Extracting the compressed dataset

from zipfile import ZipFile
dataset = '/content/sentiment140.zip'
with ZipFile(dataset, 'r') as zip:
  zip.extractall()
  print("The dataset has been extracted!")
```

```
The dataset has been 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
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
```

```
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')) # Machine Learning Model doesn't require these word and also not required for processing
```

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

Data Processing

```
# Loading the data from csv file to pandas dataframe
twitter_data = pd.read_csv('/content/training.1600000.processed.noemoticon.csv', encoding = 'ISO-8859-1')
```

```
# Checking the Number of rows and columns
twitter_data.shape
```

```
(1599999, 6)
```

```
# Printing the first 10 rows of the dataframe
twitter_data.head(10)
```

	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

```
# 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 = column_names, encoding = 'ISO-8859-1')
```

```
# Checking the Number of rows and columns
twitter_data.shape
```

```
(1600000, 6)
```

```
# Counting the number of missing values in the dataset
twitter_data.isnull().sum() # Tells how many missing values are there in each columns
```

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

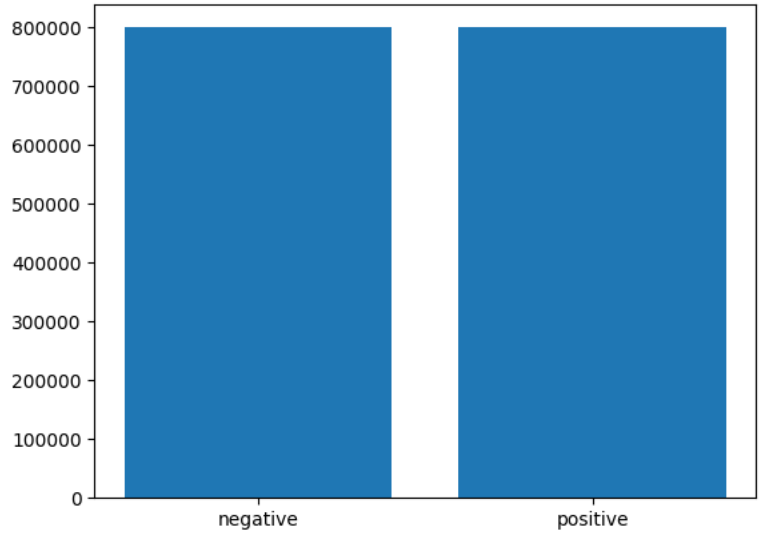
```
# Checking the distribution of target column
twitter_data['target'].value_counts()
```

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

CONVERT THE TARGET '4' TO '1'

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

```
# Checking the distribution of target column
plt.bar(['negative', 'positive'],twitter_data['target'].value_counts())
plt.show()
```



0 ---> Negative Tweet

1 ---> Positive Tweet

STEMMING

Process of reducing a word to its root word

```
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 stopwords.words('english')]
  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
			Mon Apr 06			is upset that he can't	

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

Splitnig the data into training and test data

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

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

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

```
# Converting the textual data to numerical data
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

Training the Machine Learning Model

Logistic Regression

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

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

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

Model Evaluation

Accuracy Score

```
# Accuracy score on the Training data
X_train_prediction = model.predict(X_train)
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 test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
print('Accuracy score on the testing data :', test_data_accuracy)
```

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

Model Accuracy = 77.8 %

Confusion Matrix

```
Y_pred = model.predict(X_test)
cm = confusion_matrix(Y_test, Y_pred, labels = twitter_data['target'].unique())
twitter_data_cm = pd.DataFrame(cm, index = twitter_data['target'].unique(), columns = twitter_data.target.unique())
twitter_data_cm
```

	0	1
0	121445	38555
1	32473	127527

Confusion Matrix in percentage expression

```
twitter_data_cm_percentage = twitter_data_cm.copy()
for val in twitter_data_cm_percentage:
  twitter_data_cm_percentage[val] /= twitter_data_cm_percentage[val].sum()
twitter_data_cm_percentage
```

	0	1
0	0.789024	0.232144
1	0.210976	0.767856

Saving the trained model

```
import pickle


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

Using the saved model for the future predictions

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


new_tweets = 'hey, congrats mr mukul for finally joining twitter'
vectTweet = vectorizer.transform(np.array([new_tweets]))
prediction = loaded_model.predict(vectTweet)
print(prediction[0])
if prediction[0] > 0:
  print('Tweet is Positive')
elif prediction[0] == 0:
  print('Tweet is Negative')
else:
  print('Tweet is Neutral')
```

```
1
Tweet is Positive
```

```
tweetList = ['Best Game Ever!', 'Working days are worst....']
vectTweet = vectorizer.transform(np.array(tweetList))
prediction = loaded_model.predict(vectTweet)
for val, i in enumerate(tweetList):
  print(i, ': This tweet is', 'positive' if prediction[val] == 1 else 'negative')
```

```
Best Game Ever! : This tweet is positive
Working days are worst.... : This tweet is negative
```

Start coding or generate with AI.