In [1]:
```
# importing necessary lobraries
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
nltk.download('stopwords')
nltk.download('punkt')

from tkinter import *
```

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

In [2]:
```
# Load the dataset into a pandas dataframe
df = pd.read_csv('Twitter_Data.csv')
```

Performing EDA to understand dataframe

In [3]:
```
df.head()
```

Out[3]:

	tweets	category
0	when modi promised “minimum government maximum...	-1.0
1	talk all the nonsense and continue all the dra...	0.0
2	what did just say vote for modi welcome bjp t...	1.0
3	asking his supporters prefix chowkidar their n...	1.0
4	answer who among these the most powerful world...	1.0

In [4]:
```
# Information of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162980 entries, 0 to 162979
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   tweets    162976 non-null  object 
 1   category  162973 non-null  float64
dtypes: float64(1), object(1)
memory usage: 2.5+ MB
```

In [5]:
```
# Count of each Category
df['category'].value_counts()
```

Out[5]:
```
 1.0    72250
 0.0    55213
-1.0    35510
Name: category, dtype: int64
```

Preprocess the data

In [15]:
```
stop_words = set(stopwords.words('english')) # set of stop words in English
def preprocess_tweet(tweet):
    if isinstance(tweet, str): # check if the tweet is a string
        tweet = re.sub(r"http\S+|www\S+|https\S+", '', tweet, flags=re.MULTILINE) # remove URLs
        tweet = re.sub(r'\@\w+|\#', '', tweet) # remove mentions and hashtags
        tweet = re.sub(r'\d+', '', tweet) # remove digits
        tweet = tweet.lower() # convert to lowercase
        tokens = word_tokenize(tweet) # tokenize the tweet into words
        tokens = [word for word in tokens if not word in stop_words] # remove stop words
        preprocessed_tweet = ' '.join(tokens) # join the remaining words back into a sentence
        return preprocessed_tweet
    else:
        return ''
```

In [16]:
```
# Applying the preprocessing function to each tweet
df['tweets'] = df['tweets'].apply(preprocess_tweet)
```

In [17]:
```
df['tweets'] = df['tweets'].replace('', np.nan) # replace empty strings with NaN
df = df.dropna() # drop rows with NaN values
```

In [18]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['tweets'], df['category'], test_size=0.2, random_state=42)
```

In [19]:
```
# Vectorize the tweets using TF-IDF technique
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

In [11]:
```
# Train a machine learning model on the training data
# Here, we will use Logistic Regression as our classifier
clf = LogisticRegression(random_state=42, max_iter=1000)
clf.fit(X_train, y_train)
```

Out[11]:
```
LogisticRegression(max_iter=1000, random_state=42)
```

In [12]:
```
# Evaluate the model's performance on the testing data
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
report = classification_report(y_test, y_pred)
print(f"Classification Report:\n{report}")
```

```
Accuracy: 0.8942295887047268
Classification Report:
              precision    recall  f1-score   support

        -1.0       0.89      0.78      0.83      7026
         0.0       0.87      0.97      0.91     10988
         1.0       0.92      0.90      0.91     14566

    accuracy                           0.89     32580
   macro avg       0.89      0.88      0.88     32580
weighted avg       0.90      0.89      0.89     32580
```

In [21]:
```
# Create a GUI
def predict_sentiment():
    tweet = tweet_input.get()
    preprocessed_tweet = preprocess_tweet(tweet)
    tweet_vector = vectorizer.transform([preprocessed_tweet])
    sentiment = clf.predict(tweet_vector)[0]
    if sentiment == -1:
        sentiment_text.set("Negative")
    elif sentiment == 0:
        sentiment_text.set("Neutral")
    else:
        sentiment_text.set("Positive")

root = Tk()
root.title("Twitter Sentiment Analysis")
root.geometry("400x200")

tweet_label = Label(root, text="Enter a tweet:")
tweet_label.pack()

tweet_input = Entry(root)
tweet_input.pack()

predict_button = Button(root, text="Predict", command=predict_sentiment)
predict_button.pack()

sentiment_label = Label(root, text="Sentiment:")
sentiment_label.pack()

sentiment_text = StringVar()
sentiment_output = Label(root, textvariable=sentiment_text)
sentiment_output.pack()

root.mainloop()
```

In [ ]:
```

```