Twitter Sentiment Analysis using Python

The Twitter Sentiment Analysis Project focuses on analyzing sentiments expressed in tweets about a specific topic or keyword to gain insights into public opinion.Sentiment analysis falls under the domain of natural language processing and is crucial for monitoring user sentiments across social media platforms. Twitter, in particular, often showcases a prevalence of negative opinions, especially during political discussions. It is essential for these platforms to regularly conduct sentiment analysis to identify users who contribute to the spread of hate and negativity within their communities.

importing necessary modules

[1]:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import re
import nltk
import nltk
```

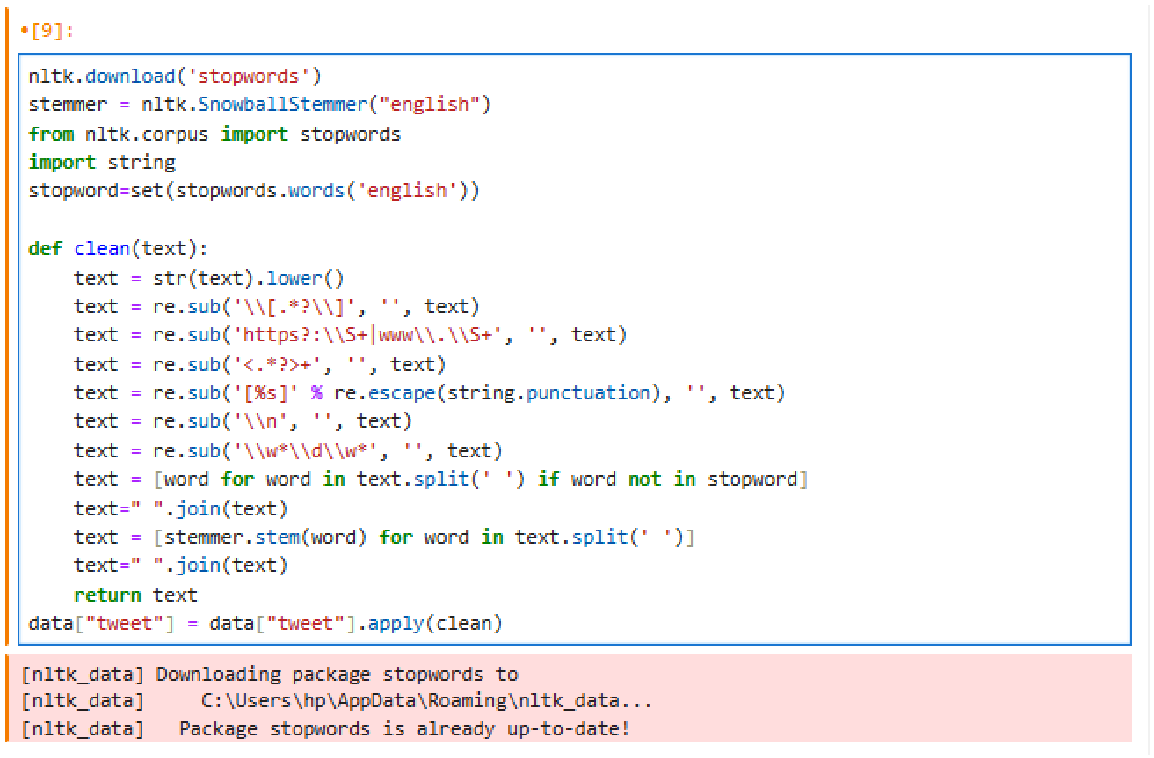
[3]:

```
#load the dataset
data = pd.read_csv("C:\\Users\\hp\\Downloads\\twitter.csv")
data.head()
```

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

The tweet column in the dataset holds the tweets we need for analyzing user sentiments in the discussion.

Cleaning the tweet column



```
•[9]:
nltk.download('stopwords')
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))

def clean(text):
    text = str(text).lower()
    text = re.sub('\\[.*?\\]', '', text)
    text = re.sub('https?:\\S+|www\\.\\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\\n', '', text)
    text = re.sub('\\w*\\d\\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["tweet"] = data["tweet"].apply(clean)
```

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

Explaination:

nltk: The Natural Language Toolkit (NLTK) is a library for working with human language data.

re: This module provides support for regular expressions, which are used for searching and manipulating strings.

string: This module contains common string operations, such as punctuation handling.

nltk.download('stopwords') This line downloads the list of common stopwords (e.g., "the," "is," "in") from the NLTK library, which are typically removed from text during preprocessing as they carry little meaning.

stemmer = nltk.SnowballStemmer("english") This initializes a stemmer for the English language using the Snowball algorithm. Stemming reduces words to their base or root form (e.g., "running" becomes "run").

def clean(text) Defines a function named clean that takes a string text (each tweet) as input.

Inside the clean Function:

text = str(text).lower(): Converts the text to lowercase for uniformity.

text = re.sub('[.*?]', '', text): Removes any text within square brackets.

text = re.sub('https?://\S+|www.\S+', '', text): Removes any URLs from the text.

text = re.sub('<.*?>+', '', text): Removes HTML tags from the text.

text = re.sub('[%s]' % re.escape(string.punctuation), '', text): Removes all punctuation from the text.

text = re.sub('\n', '', text): Removes newline characters.

text = re.sub('\w*\d*\w', '', text): Removes words containing digits.

text = [word for word in text.split(' ') if word not in stopword]: Filters out stopwords from the text.

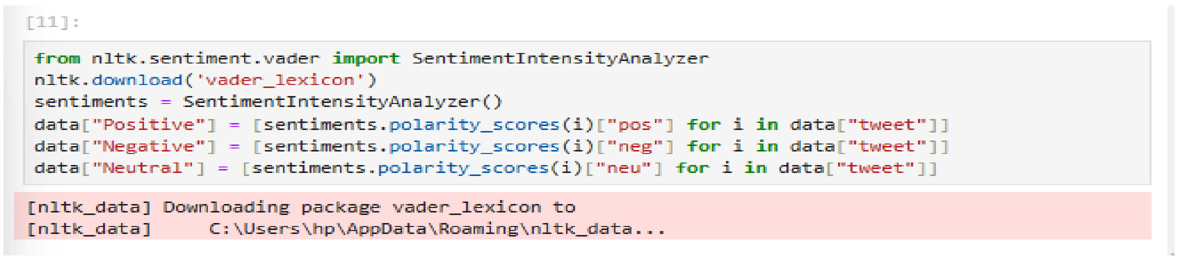
text = " ".join(text): Joins the remaining words back into a single string.

text = [stemmer.stem(word) for word in text.split(' ')]: Applies stemming to each remaining word.

text = " ".join(text): Joins the stemmed words back into a single string.

Calculate the sentiment scores:

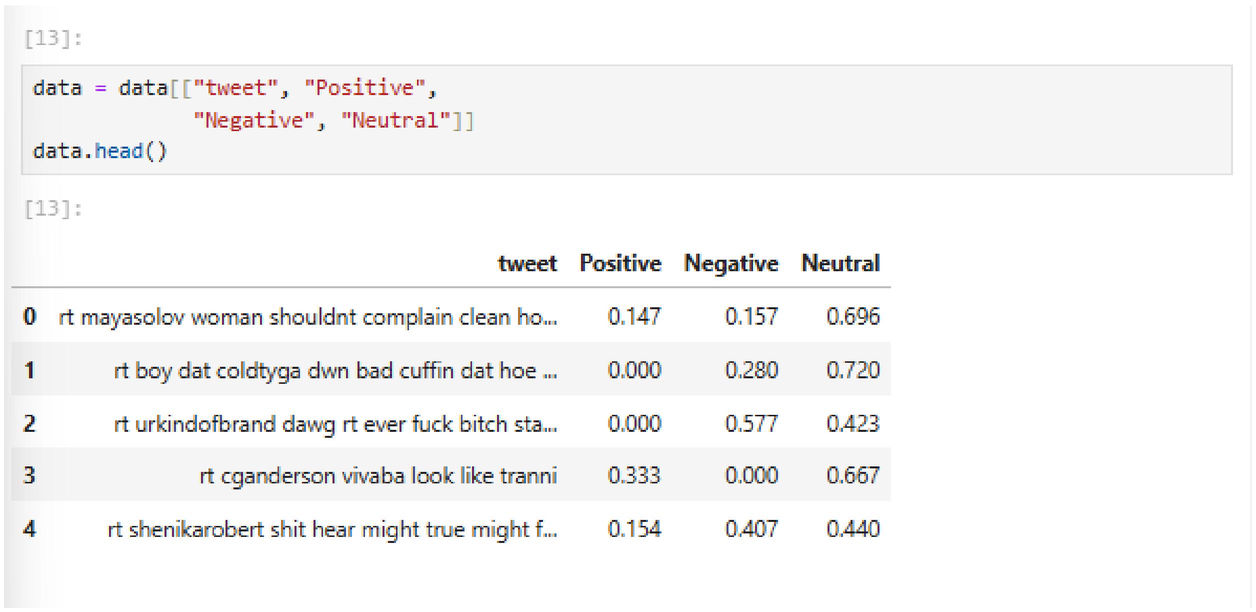
Now, the next step is to calculate the sentiment scores of these tweets and assign a label to the tweets as positive, negative, or neutral. Here is how you can calculate the sentiment scores of the tweets:



```
[11]:
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
data["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in data["tweet"]]
data["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in data["tweet"]]
data["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in data["tweet"]]

[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\hp\AppData\Roaming\nltk_data...
```

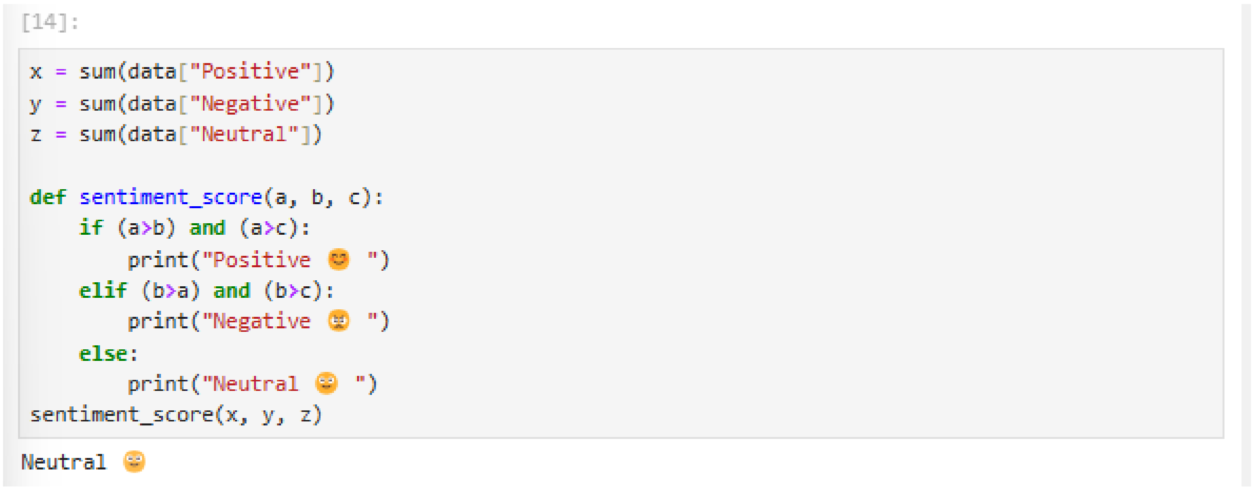
Select the columns from this data that we need for the rest of the task of Twitter sentiment analysis:



```
[13]:
data = data[["tweet", "Positive",
             "Negative", "Neutral"]]
data.head()

[13]:
```

	tweet	Positive	Negative	Neutral
0	rt mayasolov woman shouldnt complain clean ho...	0.147	0.157	0.696
1	rt boy dat coldtyga dwn bad cuffin dat hoe ...	0.000	0.280	0.720
2	rt urkindofbrand dawg rt ever fuck bitch sta...	0.000	0.577	0.423
3	rt cganderson vivaba look like tranni	0.333	0.000	0.667
4	rt shenikarobert shit hear might true might f...	0.154	0.407	0.440



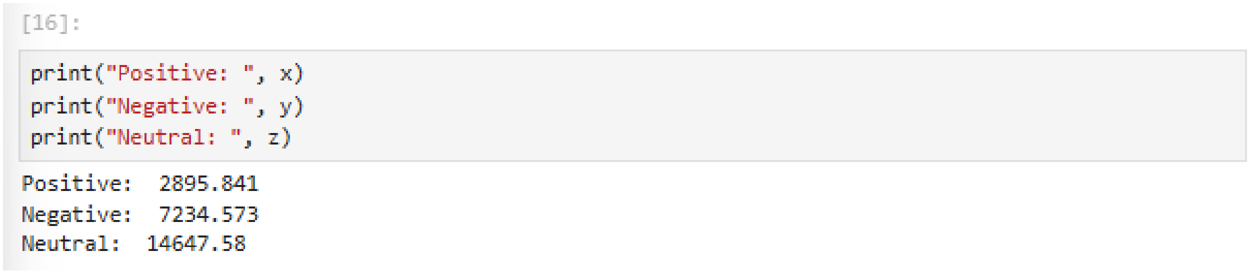
```
[14]:
x = sum(data["Positive"])
y = sum(data["Negative"])
z = sum(data["Neutral"])

def sentiment_score(a, b, c):
    if (a>b) and (a>c):
        print("Positive 😊 ")
    elif (b>a) and (b>c):
        print("Negative 😠 ")
    else:
        print("Neutral 😑 ")
sentiment_score(x, y, z)

Neutral 😑
```

The code calculates the total counts of positive, negative, and neutral sentiments from a dataset. It defines a function to evaluate which sentiment is the most prominent and prints an appropriate message with an emoji indicating the result: Positive if positive sentiments are the highest, Negative if negative sentiments are the highest, Neutral otherwise.

So the most of the tweets are neutral, which means they are neither positive nor negative. Now let's have a look at the total of the sentiment scores:



```
[16]:
print("Positive: ", x)
print("Negative: ", y)
print("Neutral: ", z)

Positive:  2895.841
Negative:  7234.573
Neutral:  14647.58
```

The total of neutral is way higher than negative and positive, but out of all the tweets, the negative tweets are more than the positive tweets, so we can say that most of the opinions are negative.