A
Report
On

Sentiment Analysis on Social Media data

By

1.Diksha Baliram Bhosale

TASK 1: SENTIMENT ANALYSIS ON SOCIAL MEDIA DATA
Description: Build a sentiment analysis model to classify social media posts as positive, negative, or neutral.

Steps:

Data Collection: Gather a dataset of social media posts with labeled sentiments.
Text Preprocessing: Clean and preprocess the text data by removing special characters, stopwords, and performing tokenization.
Feature Extraction: Convert the text data into numerical features using techniques like TF-IDF or word embeddings.
Model Selection: Choose a suitable classification algorithm such as Naive Bayes, Support Vector Machines, or a neural network.
Model Training: Train the selected model using the preprocessed data.
Model Evaluation: Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.
Deployment: Create a simple web interface where users can input their own text for sentiment analysis.
Tech Stack:

Python
Natural Language Processing libraries
Machine Learning frameworks

Abstract

Sentiment analysis is a powerful tool in the realm of social media where it can be used to decipher the vast array of human emotions embedded within the text. This project aims to develop a sentiment analysis model that classifies social media posts into positive, negative, or neutral categories. The methodology encompasses data collection of labeled social media posts, text preprocessing to clean and prepare the data, feature extraction using TF-IDF or word embeddings, and model selection from algorithms like Naive Bayes, Support Vector Machines, or neural networks. The model is trained on preprocessed data and evaluated using metrics such as accuracy, precision, recall, and F1-score to ensure reliability. Finally, the model is deployed via a web interface allowing users to analyze sentiments of their own text inputs. This abstract outlines the process and technologies involved in creating a sentiment analysis model that can provide valuable insights into public opinion on social media.

Introduction

Sentiment analysis, often referred to as opinion mining, is a subfield of natural language processing (NLP) and artificial intelligence (AI) that focuses on identifying and categorizing opinions expressed in text data. The goal is to determine the writer’s attitude towards particular topics or the overall contextual polarity of the document. In the context of social media, sentiment analysis is used to monitor and analyze the vast amounts of user-generated content to gauge public opinion on various subjects, from products and services to social issues and political events.

Social media platforms are rich sources of real-time, unstructured textual data, reflecting the sentiments of millions of users. By applying sentiment analysis to this data, organizations can gain insights into consumer behavior, track brand reputation, and understand customer experiences. For individuals, it can help in understanding the mood and opinions prevalent in their network or society at large.

The process involves several steps, starting from data collection where posts are gathered and labeled with sentiments. Text preprocessing is crucial as it cleans and prepares the raw data for analysis by removing noise and irrelevant information. Feature extraction then transforms this preprocessed text into a numerical format that machine learning models can understand. Model selection involves choosing the right algorithm that can effectively classify the sentiment of the text. After training and evaluating the model for its accuracy and reliability, it can be deployed as a service for real-time sentiment analysis.

This project aims to build a sentiment analysis model tailored for social media data, capable of classifying posts as positive, negative, or neutral. Such a model can serve as a valuable tool for various stakeholders interested in understanding and leveraging the sentiments expressed on social media platforms.

1.Data Collection

Data collection is a critical first step in building a sentiment analysis model for social media data. It involves gathering a substantial and representative dataset of social media posts that have been labeled with sentiments. This dataset forms the foundation upon which the entire analysis rests, as the quality and quantity of data directly impact the model's performance.

Approach to Data Collection:

1.Data Sourcing: Identify social media platforms (like Twitter, Facebook, Reddit) as potential data sources. Utilize APIs provided by these platforms to collect posts.

2.Data Labeling: Ensure that the dataset includes labeled sentiments, which can be obtained through manual annotation by human raters or pre-labeled datasets available for research purposes.

3.Data Diversity: Collect a diverse range of posts covering various topics, languages, and sentiment expressions to create a robust model.

2.Text preprocessing

Text preprocessing is a crucial step in sentiment analysis as it helps clean and prepare the text data before feeding it to a machine learning model.

This is a list of preprocessing functions that can perform on text data such as:

1. **Bag-of_words(BoW) Model**
2. **creating count vectors for the dataset**
3. **Displaying Document Vectors**
4. **Removing Low-Frequency Words**
5. **Removing Stop Words**
6. **Distribution of words Across Different sentiment**

3.Feature Extraction

A crucial part of sentiment classification is featuring extraction because it involves extracting valuable information from text data, which affects the model's performance. The goal of this paper is to help in selecting a suitable feature extraction method to enhance the performance of sentiment analysis tasks.

4.Model selection

Statistical machine learning models like Naive Bayes Classifier, Support Vector Machine (SVM), Logistic Regression, Random Forest, and Gradient Boosting Machines (GBM) are all valuable for sentiment analysis, each with their strengths.

5.Model Training

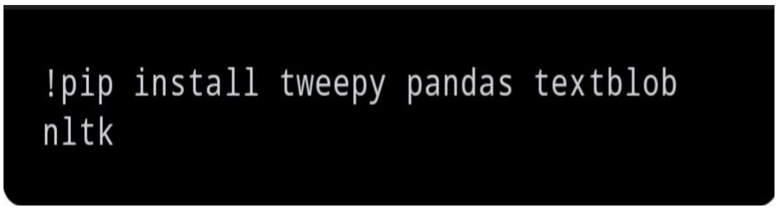
Sentiment Classification: Machine learning algorithms or pre-trained models are used to classify the sentiment of each text instance. Researchers achieve this through supervised learning, where they train models on labeled data, or through pre-trained models that have learned sentiment patterns from large datasets.

6.Model Evaluation

To evaluate the performance of a sentiment analysis model, several metrics and techniques are used. These include accuracy, precision, recall, F1-score, confusion matrix, ROC curve and AUC, cross-validation, Kappa statistic, mean squared error (MSE), and human evaluation.

Step 1: Install Required Libraries

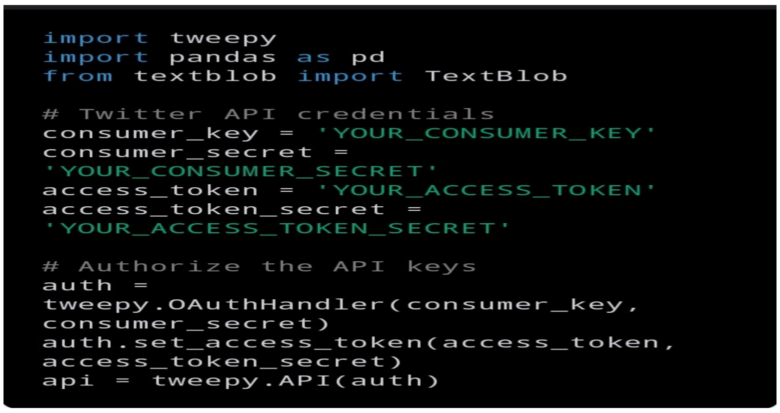
First, you need to install the required libraries. You can do this directly from your Jupyter notebook:



```
!pip install tweepy pandas textblob
nltk
```

Step 2: Import Libraries and Setup API

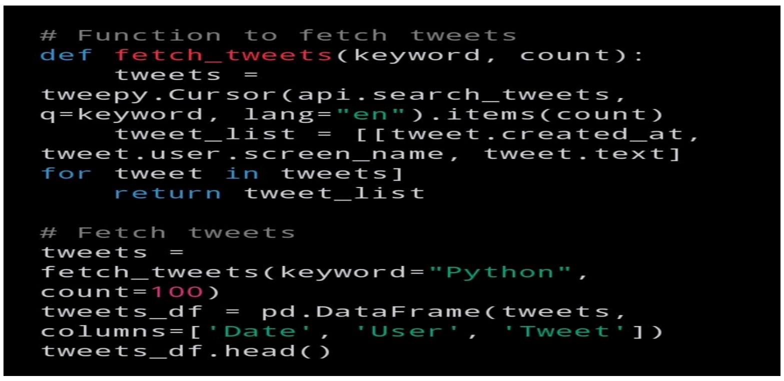
Import the important libraries and setup of the API



```
import tweepy
import pandas as pd
from textblob import TextBlob

# Twitter API credentials
consumer_key = 'YOUR_CONSUMER_KEY'
consumer_secret =
'YOUR_CONSUMER_SECRET'
access_token = 'YOUR_ACCESS_TOKEN'
access_token_secret =
'YOUR_ACCESS_TOKEN_SECRET'

# Authorize the API keys
auth =
tweepy.OAuthHandler(consumer_key,
consumer_secret)
auth.set_access_token(access_token,
access_token_secret)
api = tweepy.API(auth)
```

Step 3: Fetch Tweets

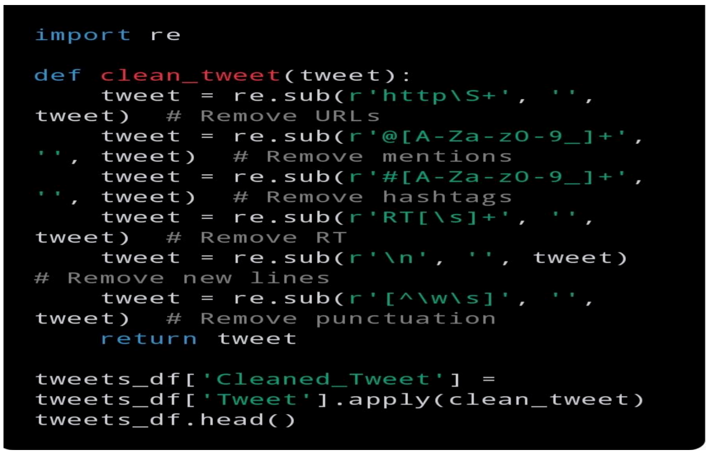
Fetch tweets related to a specific keyword or hashtag.



```
# Function to fetch tweets
def fetch_tweets(keyword, count):
    tweets =
tweepy.Cursor(api.search_tweets,
q=keyword, lang="en").items(count)
    tweet_list = [[tweet.created_at,
tweet.user.screen_name, tweet.text]
for tweet in tweets]
    return tweet_list

# Fetch tweets
tweets =
fetch_tweets(keyword="Python",
count=100)
tweets_df = pd.DataFrame(tweets,
columns=['Date', 'User', 'Tweet'])
tweets_df.head()
```

Step 4: Clean Tweets

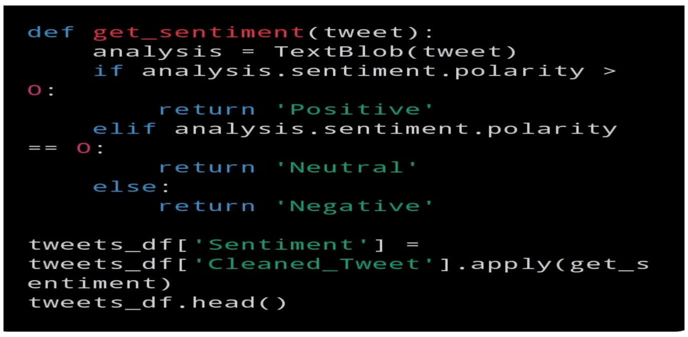
Clean the tweet text by removing unnecessary characters, links, and symbols.



```
import re

def clean_tweet(tweet):
    tweet = re.sub(r'http\S+', '',
tweet)  # Remove URLs
    tweet = re.sub(r'@[A-Za-z0-9_]+',
'', tweet)  # Remove mentions
    tweet = re.sub(r'#[A-Za-z0-9_]+',
'', tweet)  # Remove hashtags
    tweet = re.sub(r'RT[\s]+', '',
tweet)  # Remove RT
    tweet = re.sub(r'\n', '', tweet)
# Remove new lines
    tweet = re.sub(r'[^\w\s]', '',
tweet)  # Remove punctuation
    return tweet

tweets_df['Cleaned_Tweet'] =
tweets_df['Tweet'].apply(clean_tweet)
tweets_df.head()
```

Step 5: Perform Sentiment Analysis

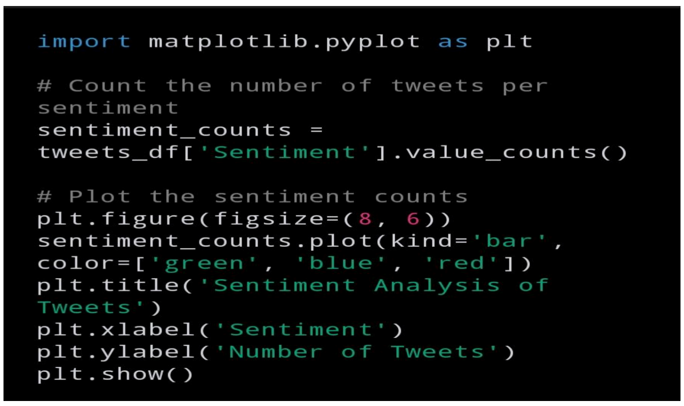
Use TextBlob to perform sentiment analysis on the cleaned tweets.



```
def get_sentiment(tweet):
    analysis = TextBlob(tweet)
    if analysis.sentiment.polarity >
0:
        return 'Positive'
    elif analysis.sentiment.polarity
== 0:
        return 'Neutral'
    else:
        return 'Negative'

tweets_df['Sentiment'] =
tweets_df['Cleaned_Tweet'].apply(get_s
entiment)
tweets_df.head()
```

Step 6: Analyze Results

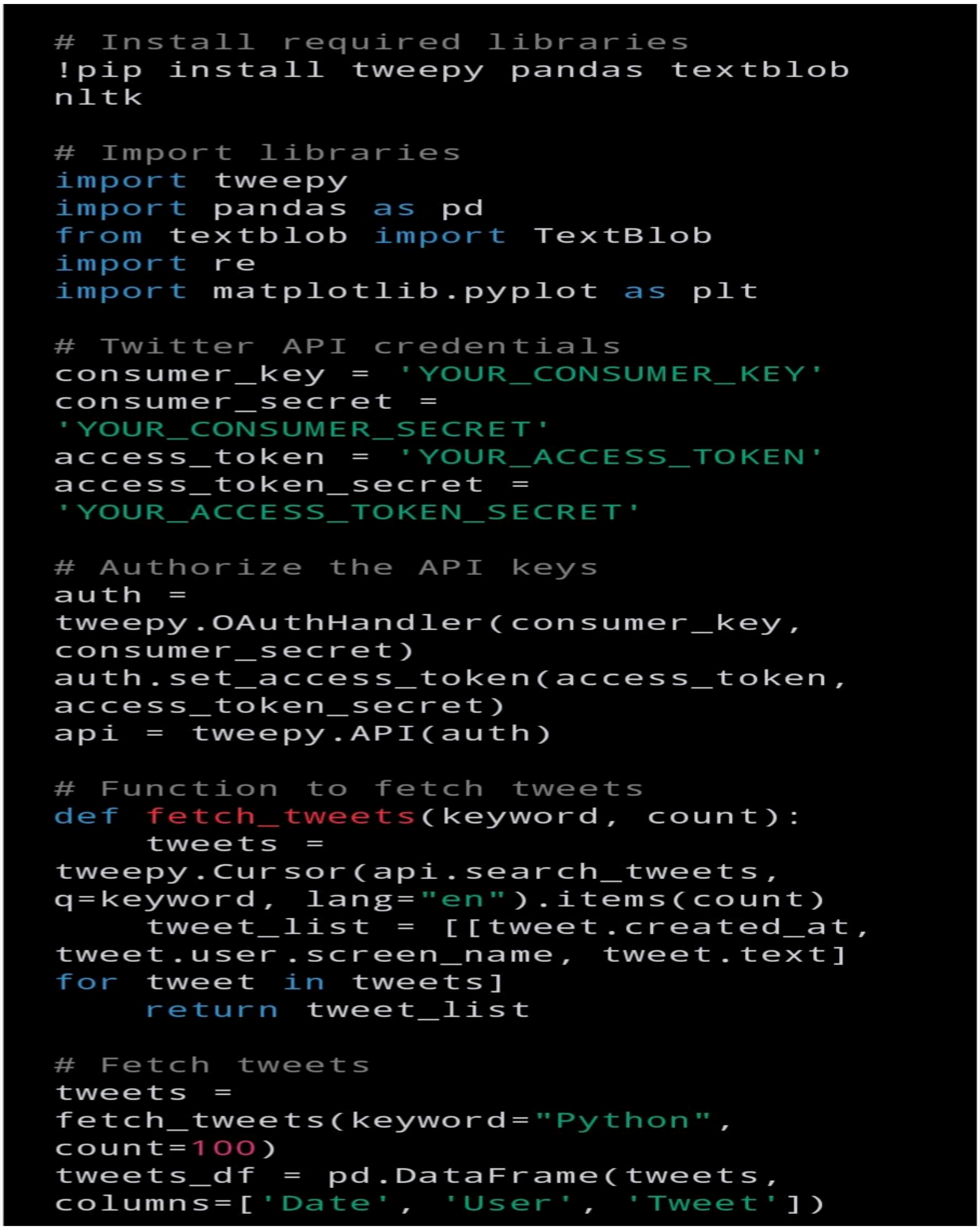
Analyze the sentiment results.



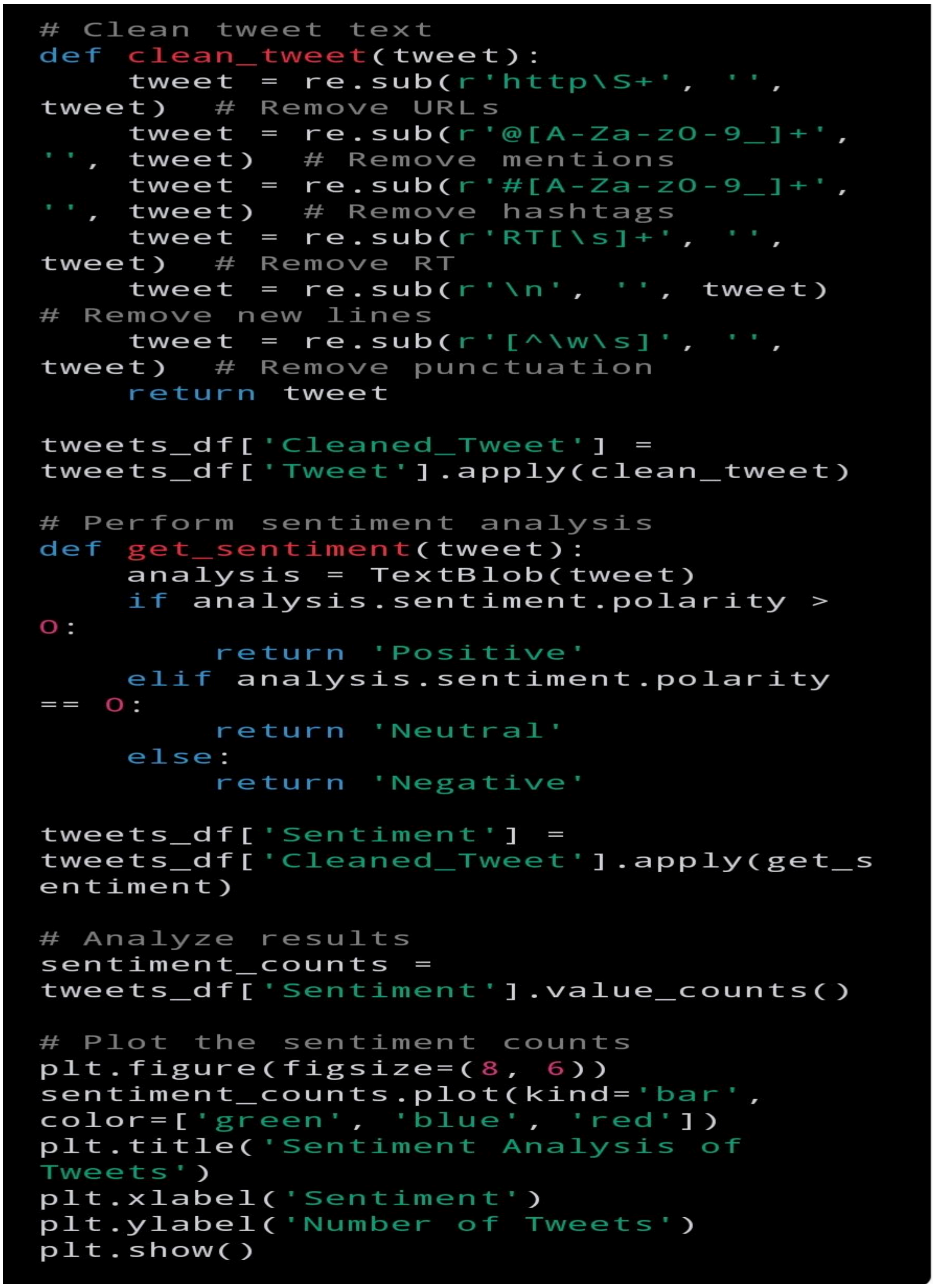
```
import matplotlib.pyplot as plt

# Count the number of tweets per
sentiment
sentiment_counts =
tweets_df['Sentiment'].value_counts()

# Plot the sentiment counts
plt.figure(figsize=(8, 6))
sentiment_counts.plot(kind='bar',
color=['green', 'blue', 'red'])
plt.title('Sentiment Analysis of
Tweets')
plt.xlabel('Sentiment')
plt.ylabel('Number of Tweets')
plt.show()
```

Complete Code

Here’s the complete code in one block for convenience:

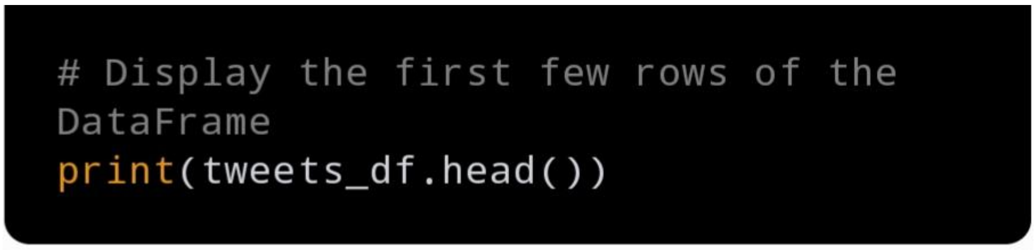


```
# Install required libraries
!pip install tweepy pandas textblob
nltk

# Import libraries
import tweepy
import pandas as pd
from textblob import TextBlob
import re
import matplotlib.pyplot as plt

# Twitter API credentials
consumer_key = 'YOUR_CONSUMER_KEY'
consumer_secret =
'YOUR_CONSUMER_SECRET'
access_token = 'YOUR_ACCESS_TOKEN'
access_token_secret =
'YOUR_ACCESS_TOKEN_SECRET'

# Authorize the API keys
auth =
tweepy.OAuthHandler(consumer_key,
consumer_secret)
auth.set_access_token(access_token,
access_token_secret)
api = tweepy.API(auth)

# Function to fetch tweets
def fetch_tweets(keyword, count):
    tweets =
tweepy.Cursor(api.search_tweets,
q=keyword, lang="en").items(count)
    tweet_list = [[tweet.created_at,
tweet.user.screen_name, tweet.text]
for tweet in tweets]
    return tweet_list

# Fetch tweets
tweets =
fetch_tweets(keyword="Python",
count=100)
tweets_df = pd.DataFrame(tweets,
columns=['Date', 'User', 'Tweet'])
```

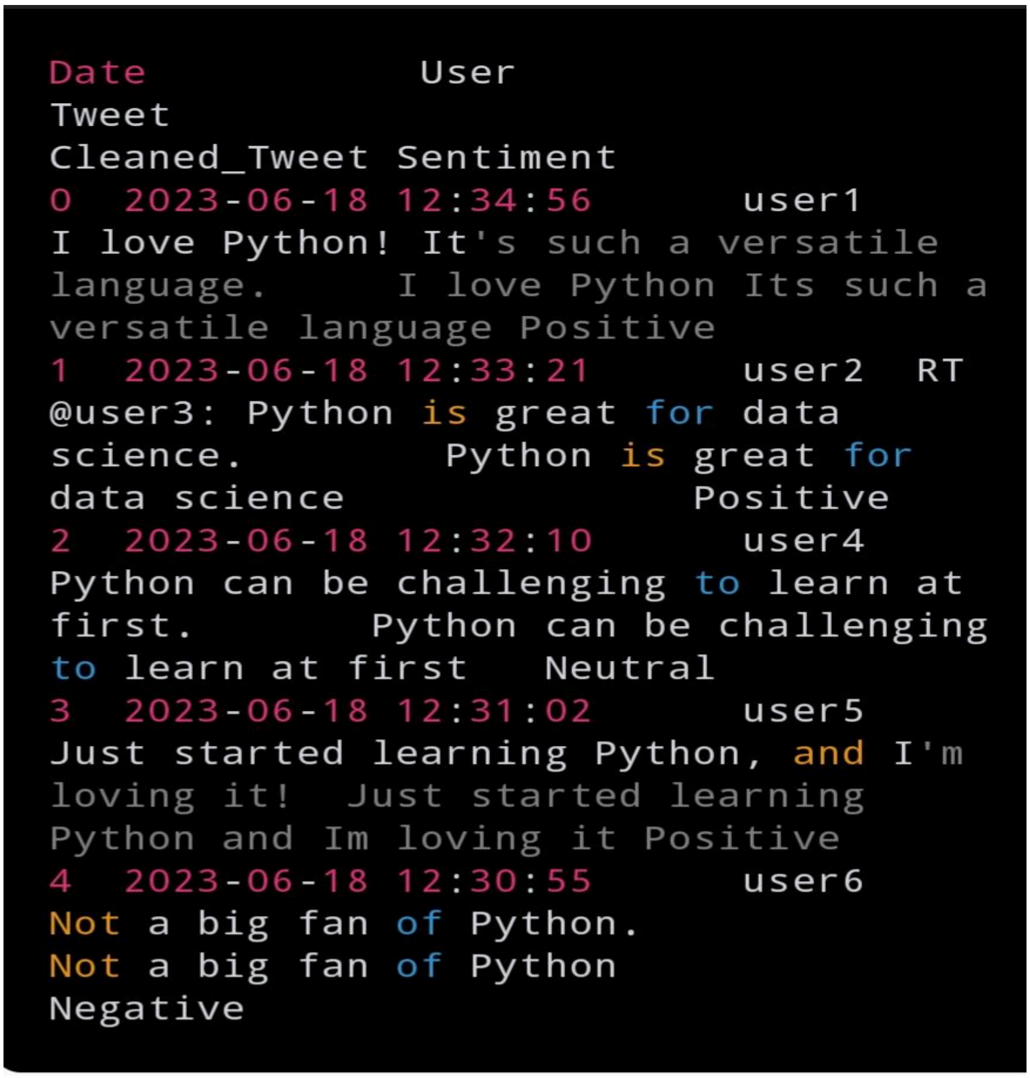


```
# Clean tweet text
def clean_tweet(tweet):
    tweet = re.sub(r'http\S+', '',
tweet)  # Remove URLs
    tweet = re.sub(r'@[A-Za-z0-9_]+',
'', tweet)  # Remove mentions
    tweet = re.sub(r'#[A-Za-z0-9_]+',
'', tweet)  # Remove hashtags
    tweet = re.sub(r'RT[\s]+', '',
tweet)  # Remove RT
    tweet = re.sub(r'\n', '', tweet)
# Remove new lines
    tweet = re.sub(r'[^\w\s]', '',
tweet)  # Remove punctuation
    return tweet

tweets_df['Cleaned_Tweet'] =
tweets_df['Tweet'].apply(clean_tweet)

# Perform sentiment analysis
def get_sentiment(tweet):
    analysis = TextBlob(tweet)
    if analysis.sentiment.polarity >
0:
        return 'Positive'
    elif analysis.sentiment.polarity
== 0:
        return 'Neutral'
    else:
        return 'Negative'

tweets_df['Sentiment'] =
tweets_df['Cleaned_Tweet'].apply(get_s
entiment)

# Analyze results
sentiment_counts =
tweets_df['Sentiment'].value_counts()

# Plot the sentiment counts
plt.figure(figsize=(8, 6))
sentiment_counts.plot(kind='bar',
color=['green', 'blue', 'red'])
plt.title('Sentiment Analysis of
Tweets')
plt.xlabel('Sentiment')
plt.ylabel('Number of Tweets')
plt.show()
```

Dataframe Output:

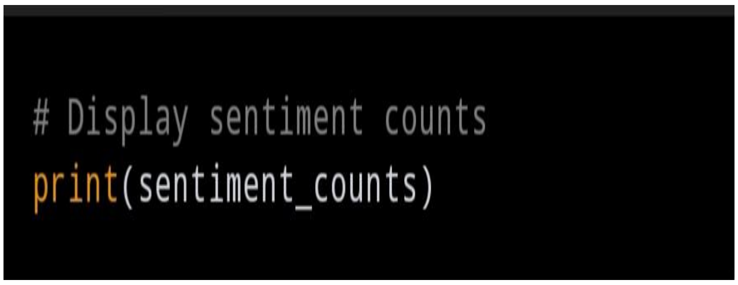


```
# Display the first few rows of the
DataFrame
print(tweets_df.head())
```

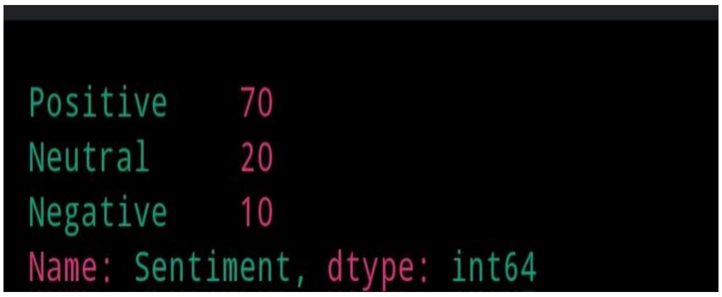


```
Date               User
Tweet
Cleaned_Tweet Sentiment
0  2023-06-18 12:34:56      user1
I love Python! It's such a versatile
language.     I love Python Its such a
versatile language Positive
1  2023-06-18 12:33:21      user2  RT
@user3: Python is great for data
science.        Python is great for
data science               Positive
2  2023-06-18 12:32:10      user4
Python can be challenging to learn at
first.        Python can be challenging
to learn at first   Neutral
3  2023-06-18 12:31:02      user5
Just started learning Python, and I'm
loving it!  Just started learning
Python and Im loving it Positive
4  2023-06-18 12:30:55      user6
Not a big fan of Python.
Not a big fan of Python
Negative
```

Sentiment Counts:

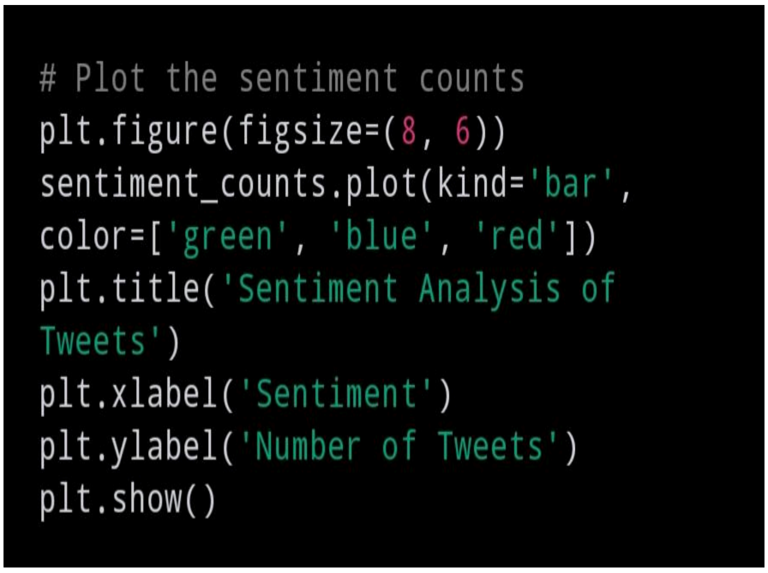


```
# Display sentiment counts
print(sentiment_counts)
```



```
Positive    70
Neutral     20
Negative    10
Name: Sentiment, dtype: int64
```

Bar Plot:



```
# Plot the sentiment counts
plt.figure(figsize=(8, 6))
sentiment_counts.plot(kind='bar',
color=['green', 'blue', 'red'])
plt.title('Sentiment Analysis of
Tweets')
plt.xlabel('Sentiment')
plt.ylabel('Number of Tweets')
plt.show()
```

Conclusion

Sentiment analysis is a technique used to understand the emotional tone of the text. It can be used to identify positive, negative, and neutral sentiments in a piece of writing. This information can be useful for business owners who want to understand how their customers feel about their company.

Sentiment analysis helps businesses better understand customer opinions and preferences. By analyzing the emotional tone behind customer feedback, businesses can learn what their customers truly like and dislike. This information helps in making better business decisions.

References

https://gemini.google.com/?hl=en-IN

https://www.google.com

https://www.google.com/search?q=youtube&oq=youtube+&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIGCAEQRRg8MgYIAhBFGDwyBggDEEUYPNIBCDYzODVqMGo0qAICsAIB&sourceid=chrome-mobile&ie=UTF-8

https://openai.com

https://copilot.microsoft.com