

SENTIMENT ANALYSIS ON SOCIAL MEDIA USING NLP

INDEX

Chapter Name	Page n.o
1. Introduction	3
2. Objective of the Project	5
3. Software Requirements	6
4. Methodology for Implementation	9
5. Code Implementation Details	13
6. Results	19
7. CONCLUSION	24
Overall code	25

ABSTRACT:

The implements a sentiment analysis pipeline using machine learning techniques on Twitter data. It begins by preprocessing the data, including handling missing values and exploring the dataset's characteristics through visualizations. The text data is further processed using SpaCy for lemmatization and stop word removal. Two classifiers, Naive Bayes and Random Forest, are trained and evaluated for sentiment prediction, with accuracy scores and classification reports generated. Additionally, a Streamlit-based web application is developed for real-time sentiment analysis, leveraging a pre-trained Random Forest model and TF-IDF vectorization. This pipeline showcases a systematic approach to sentiment analysis, from data preprocessing to model deployment, demonstrating the efficacy of machine learning in extracting sentiment insights from textual data.

Problem Statement:

The problem statement revolves around conducting sentiment analysis on Twitter data. This task involves categorizing tweets into sentiment classes such as positive, negative, neutral, and irrelevant. The primary objectives include preprocessing the data by handling missing values, renaming columns, and preparing the text for machine learning models through techniques like stop word removal and lemmatization. Subsequently, the code aims to train and evaluate machine learning classifiers like Naive Bayes and Random Forest on the preprocessed data, assessing their accuracy and performance via classification reports. Additionally, a web application using Streamlit is developed to facilitate real-time sentiment analysis on user-entered text, leveraging a pre-trained Random Forest model and TF-IDF vectorizer for sentiment prediction. This project demonstrates the practical application of machine learning in sentiment analysis and provides a user-friendly solution for sentiment classification tasks.

1. INTRODUCTION

The emergence of web 2.0 is changing the world of social media. Not only online social media used to connect, share information and their personal opinion to others, but even business can also communicate, understand and improve their product and services through connecting in social media. The number of social media users increases every day and it is estimated in 2019 there will be up to 2.77 billion social media users worldwide. There is various type of information uploaded and shared on social media in the form of text, videos, photos and audio. Social media is rich with raw and unprocessed data and the improvement in technology, especially in machine learning and artificial intelligence, allow the data to be processed and converted it into a useful data that they can benefit most business organization.

Social media is becoming an increasingly more important source of information for an enterprise. Sentiment is an attitude, thought, or judgment prompted by feeling. Sentiment analysis, which is also known as opinion mining, studies people's sentiments towards certain entities. From a user's perspective, people are able to post their own content through various social media, such as forums, micro-blogs, or online social networking sites. A ground truth is more like a tag of a certain opinion, indicating whether the opinion is positive, negative, or neutral.

“It is a quite boring movie..... but the scenes were good enough. ”

The given line is a movie review that states that “it” (the movie) is quite boring but the scenes were good. Understanding such sentiments require multiple tasks.

Hence, SENTIMENTAL ANALYSIS is a kind of text classification based on Sentimental Orientation (SO) of opinion they contain. Sentiment analysis of product reviews has recently become very popular in text mining and computational linguistics research.

- Firstly, evaluative terms expressing opinions must be extracted from the review.
- Secondly, the SO, or the polarity, of the opinions must be determined.
- Thirdly, the opinion strength, or the intensity, of an opinion should also be determined.
- Finally, the review is classified with respect to sentiment classes, such as Positive and Negative, based on the SO of the opinions it contains.

Some of the common examples of Sentiment Analysis are

- Customer Feedback
- Product Analysis
- Social Media Monitoring
- Brand Perception
- Emotion Recognition
- Chatbot reactions
- Voice of Employee
- Threat Detection etc.

Different Types of Sentiment Analysis

There are different kinds of sentiment analysis and applications. This article will discuss 4 important types and popular use cases of Sentiment Analysis.

1. Fine-Grained Sentiment:

This type of analysis gives you an understanding of customer feedback. You can get precise results in terms of the polarity of the input. For example, you can label the reviews as Positive ,Negative, Neutral

2. Intent Analysis

This is a special type of analysis that goes deeper than basic sentiment analysis, and can determine whether the data is a complaint, suggestion, query. You are also able to capture the overall intention behind the specific data.This can help improve social engagement and customer experience.

3. Aspect Based

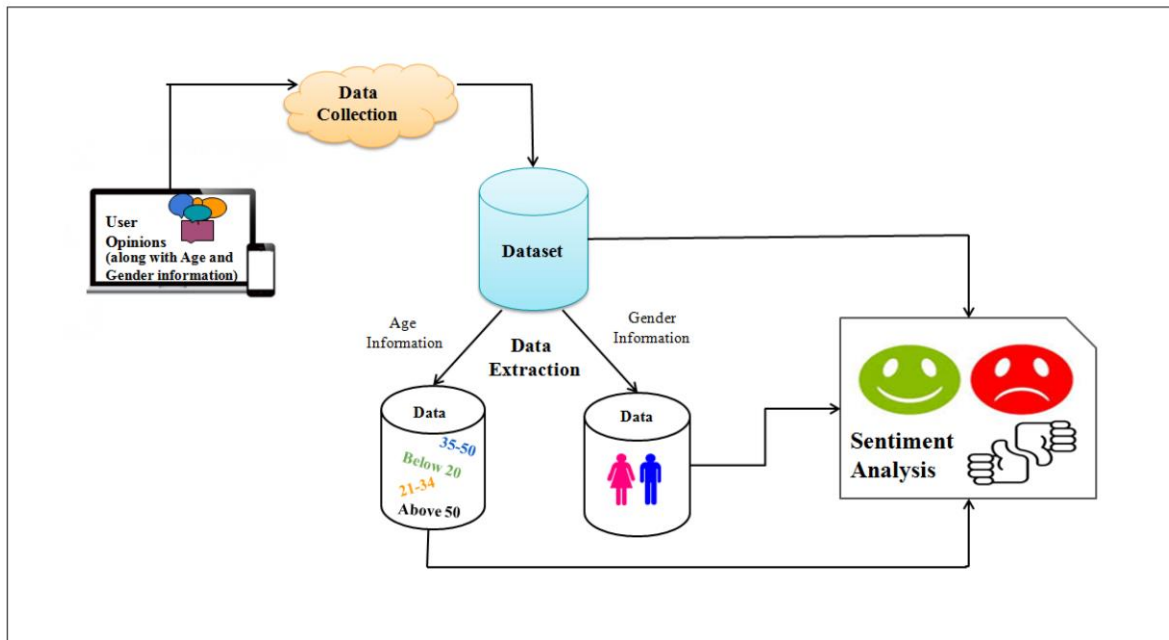
As this type of sentiment analysis allows you to analyze a specific aspect of the data. For example, let's say you have launched a new perfume, and you want to analyze the user's feedback to check which aspect of your product they liked the most. You can then find out whether they liked the packaging, or scent, or the price, etc.

4. Emotion Detection

This is one of the most commonly used sentiment analysis where we detect the emotion behind a sentence. We aim to detect whether the given sentence is happy, sad, frustrated, angry.

2. Objective of the Project

- Scrapping data set on various websites like kaggale featuring various social media specifically Twitter data analysis.
- Analyze and categorize data set.
- Analyze sentiment on dataset from document level.
- Categorization or classification of opinion sentiment into-
 1. Positive
 2. Negative
 3. Neutral
 4. Irrelevant



3. Software Requirements

Software Requirements:

- VS Code
- Google Collab
- Windows Operating System.

The required modules for the project are Pandas, nltk, matplotlib, seaborn, sklearn, joblib and streamlit.

PANDAS:The dataset is stored in a structured format such as CSV, Excel, or a database, pandas can be used to load this data into a DataFrame, which can then be manipulated and used for further processing. It provides powerful tools for data preprocessing, including handling missing values, encoding categorical variables, and scaling numerical features.

NLTK: NLTK (Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces and lexical resources, such as WordNet. Additionally, NLTK includes a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. In this project, nltk is used to remove stopwords from the news.

MATPLOTLIB: Matplotlib is a powerful plotting library for Python that enables you to create a wide variety of plots, charts, and visualizations. It is extensively used for data visualization tasks in fields such as data science, machine learning, finance, and scientific computing

SEABORN: Seaborn is designed to work well with Pandas DataFrames and NumPy arrays, making it an excellent tool for exploring and visualizing structured data. It provides a convenient and intuitive interface for exploring data and communicating insights effectively through visualizations. Whether you're analyzing data for exploratory purposes or preparing visualizations for presentations or reports, Seaborn offers the tools you need to create informative and visually appealing plots.

SKLEARN: Scikit-learn, also known as sklearn, is a popular machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It is built on top of other scientific computing libraries such as NumPy, SciPy, and Matplotlib, and it is designed to be userfriendly, modular, and easily

extensible. In this project, sklearn is used to get all the algorithms and even the accuracy metrics, classification reports. It is used for modelling purpose in this project. scikit-learn is a powerful and versatile library for machine learning in Python, suitable for both beginners and experienced practitioners. It provides a wide range of algorithms and tools for building, evaluating, and deploying machine learning models across various domains and applications.

JOB LIB: Joblib is a Python library that provides utilities for saving and loading Python objects, especially large NumPy arrays, efficiently to and from disk. It is particularly useful for saving and loading trained machine learning models, which often involve large parameter sets or complex data structures. In this project, after comparing accuracy the SVM got highest accuracy so that model is saved using joblib.

STREAMLIT: Streamlit is an open-source Python library that allows you to create interactive web applications for data science and machine learning projects with ease. It is designed to simplify the process of building and sharing data-centric web apps, enabling data scientists and machine learning engineers to quickly prototype and deploy their projects without requiring web development expertise. In this project, streamlit is used to design the web application.

MACHINE LEARNING:

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans. The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect. We have three types:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

Supervised Machine Learning:

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. In supervised learning, the training data provided to the

machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Unsupervised Machine Learning:

In the previous we got to know about supervised machine learning in which models are trained using labeled data under the supervision of training data. But there may be many cases in which we do not have labeled data and need to find the hidden patterns from the given dataset. Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.

ALGORITHMS USED:

The algorithms used in project are:

1. Random forest
2. Naive bayes

Multinomial Naive Bayes (nb_clf) classifier: Multinomial Naive Bayes is a probabilistic classifier based on Bayes' theorem with the assumption of feature independence. It is particularly effective for text classification tasks, where features represent the frequency of terms (words) in documents. Despite its simplicity, Multinomial Naive Bayes is computationally efficient and interpretable, making it suitable for scenarios where transparency and interpretability are crucial.

Random Forest (rf_clf) classifier: Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. Each decision tree is trained on a random subset of the data, and their predictions are aggregated to produce the final output. This technique makes Random Forest robust against overfitting and noise in the data. Additionally, Random Forest provides insights into feature importance, helping users understand which features contribute most to the model's predictive power.

4. Methodology for Implementation

Data Information:

The "twitter_training.csv" file is a dataset containing Twitter data that has been collected for sentiment analysis purposes. The dataset includes several columns that capture key information about each tweet. The columns typically found in the dataset may include:

Tweet ID: A unique identifier for each tweet, allowing for traceability and referencing.

Entity: Information about the entity or user associated with the tweet, such as the username or handle.

Sentiment: The sentiment label assigned to the tweet, indicating whether it is positive, negative, or neutral in sentiment.

Text: The actual text content of the tweet, which is the primary focus of sentiment analysis algorithms.

The data in the CSV file may also have undergone preprocessing steps, such as cleaning, tokenization, and possibly lemmatization, to prepare the text data for sentiment analysis. Additionally, the dataset may have been labelled or annotated with sentiment labels to enable supervised learning approaches for sentiment classification.

The CSV file "twitter_training.csv" contains a dataset with 74,681 rows and 4 columns. Each row represents a tweet from Twitter, while the columns provide specific information about each tweet.

A	B	C	D
2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
2401	Borderlands	Positive	I am coming to the borders and I will kill you all,
2401	Borderlands	Positive	im getting on borderlands and i will kill you all,
2401	Borderlands	Positive	im coming on borderlands and i will murder you all,
2401	Borderlands	Positive	im getting on borderlands 2 and i will murder you me all,
2401	Borderlands	Positive	im getting into borderlands and i can murder you all,
2402	Borderlands	Positive	So I spent a few hours making something for fun... If you don't know I am a HUGE @Borderlands fan and Maya is one of my favorite characters. So I decided to make myself a wallpaper for my PC. . Here is the original image
2402	Borderlands	Positive	So I spent a couple of hours doing something for fun... If you don't know that I'm a huge @ Borderlands fan and Maya is one of my favorite characters, I decided to make a wallpaper for my PC.. Here's the original picture con
2402	Borderlands	Positive	So I spent a few hours doing something for fun... If you don't know I'm a HUGE @ Borderlands fan and Maya is one of my favorite characters.
2402	Borderlands	Positive	So I spent a few hours making something for fun... If you don't know I am a HUGE RhandlerR fan and Maya is one of my favorite characters. So I decided to make myself a wallpaer for my PC. . Here is the original image ver

Format of Data set:

Tweet ID : 2401

Entity : Borderlands

Sentiment : Positive

Text : I am coming to the borders and I will kill you all

DATA COLLECTION:

Data which means produce about reviews on some company like (Nvidia, Borderlands, Amazon, Apex Legends, Player Unknowns Battlegrounds (PUBG)) etc., from the twitter data. Each review includes the following information 1. Tweet ID, 2. Entity, 3. Sentiment, 4. Text.

Methodology of Twitter sentiment analysis :

The methodology of the code you provided can be summarized as follows:

1. Data set download :

- Download the related data set from different websites like Kaggle etc.

2. Data set and Preparation:

- Load the Twitter training data from a CSV file into a Pandas DataFrame.
- Rename columns, drop unnecessary columns like "Tweet ID," and handle missing values by dropping rows with Null values.
- We find out the duplicate values.
- Check data types and ensure data cleanliness.

3. Exploratory Data Analysis (EDA):

- Perform EDA to understand the data's characteristics, such as checking for duplicates in the 'Text' column and exploring the distribution of sentiment labels.

4. Data Visualization :

- We perform the graphs Entity vs Sentiment and also how many sentiment data related bar graph like
 - o Entity vs Sentiment
 - o Sentiment bar graph
 - o Sentiment pie chart

- Word Cloud of Entities
- Word Cloud of Text

5. Text Preprocessing:

- Use the SpaCy library to preprocess text data by tokenizing, lemmatizing, and removing punctuation and stop words.
- Create a new column 'preprocessed_data' in the DataFrame containing the preprocessed text.

So the basic steps involved in cleaning the data are

- Tokenization
- Converting the text from upper case to lower case
- Correcting the spelling mistakes
- Punctuation removal
- Number removal
- Stop words removal
- Normalization via lemmatization or stemming

6. Label Encoding:

- Encode the 'Sentiment' column using LabelEncoder to convert sentiment labels into numerical values suitable for machine learning algorithms.

7. Splitting Data and Feature Extraction:

- Split the data into training and testing sets using train_test_split from sklearn.model_selection.
- Use TfidfVectorizer from sklearn.feature_extraction.text to convert text data into numerical features (TF-IDF representation).

8. Model Training and Evaluation:

- Train a Multinomial Naive Bayes (nb_clf) classifier and a Random Forest (rf_clf) classifier on the training data.
- Evaluate both classifiers' performance using accuracy_score and classification_report on the test data.

9. Prediction on Validation Data:

- Load validation data from a CSV file into another DataFrame.

- Preprocess text data in the validation set using the `preprocess_data` function.
- Transform preprocessed text into TF-IDF representation using the `TfidfVectorizer`.
- Use the trained Random Forest classifier (`rf_clf`) to predict sentiment labels for validation data.
- Compare predicted labels with true labels from the validation data and print the results.

WEB APPLICATION:

Your code defines a Streamlit web application for sentiment analysis using a trained `RandomForestClassifier` and `SpaCy` for text preprocessing. Here's a breakdown of what each part of your code does:

1. Imports necessary libraries/modules such as `joblib`, `pandas`, `spacy`, `sklearn`, and `streamlit`.
2. Loads the `SpaCy` English language model ('`en_core_web_sm`').
3. Loads a trained `RandomForestClassifier` model ('`random_forest_model.joblib`') and a `TfidfVectorizer` ('`tfidf_vectorizer.joblib`') used for text vectorization.
4. Defines a function '`preprocess_data`' to preprocess text data by tokenizing, filtering out punctuation and stop words, and lemmatizing the tokens.
5. Defines a function '`predict_sentiment`' to predict sentiment using the preprocessed text data.
6. Maps sentiment labels to their corresponding numeric values.
7. Creates a Streamlit app with a text input field for user input and a button to trigger sentiment analysis.
8. When the user clicks the "Analyze" button, the app preprocesses the input text, predicts the sentiment using the trained model, and displays the sentiment label.

5. Code Implementation Details

The following steps explain the code step by step:

1. Import some important Libraries:

- numpy for numerical operations.
- pandas for data manipulation and analysis.
- matplotlib.pyplot for data visualization.
- seaborn for enhanced data visualization.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
import spacy
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

2. Read Data: The huge dataset of reviews obtained from amazon.com comes in a .csv file format. A small python code has been implemented in order to read the dataset from those files and dump them in to a pickle file for easier.

```
df= pd.read_csv(r"/content/twitter_training.csv")
df
```

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...
...
74676	9200	Nvidia	Positive	Just realized that the Windows partition of my...
74677	9200	Nvidia	Positive	Just realized that my Mac window partition is ...
74678	9200	Nvidia	Positive	Just realized the windows partition of my Mac ...
74679	9200	Nvidia	Positive	Just realized between the windows partition of...
74680	9200	Nvidia	Positive	Just like the windows partition of my Mac is l...

74681 rows × 4 columns

3. Data Preprocessing:

- Renames columns to ['Tweet ID', 'Entity', 'Sentiment', 'Text'].
- Drops the 'Tweet ID' column as it's not needed.

```
df.columns = ['Tweet ID', 'Entity', 'Sentiment', 'Text']
df.head()
```

	Tweet ID	Entity	Sentiment	Text
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...

- Generate the data info, data types, data shape etc.

```
[162] print(df.shape)

(74681, 4)

df.dtypes
Entity      object
Sentiment   object
Text        object
dtype: object

[70] print(df.columns)

Index(['Entity', 'Sentiment', 'Text'], dtype='object')

[71] print(df.head()) # Display first few rows
print(df.tail())
```

- Checks for missing values (df.isna().any()) and drops rows with missing values (df.dropna()).

```
[72] df.isnull().any()

Entity      False
Sentiment   False
Text        True
dtype: bool

df.isnull().sum()
Entity      0
Sentiment   0
Text       686
dtype: int64

[74] df.dropna(inplace=True)

[75] df.isnull().sum()

Entity      0
Sentiment   0
Text        0
dtype: int64
```

- Checks for duplicated text in the 'Text' column (df.Text.duplicated().any()).

```
[76] df.Text.duplicated().any()
True

df[df.Text == "from"]
```

	Entity	Sentiment	Text
5230	Amazon	Neutral	from
6478	Amazon	Negative	from
34960	Microsoft	Negative	from
53512	RedDeadRedemption(RDR)	Positive	from
71818	TomClancysGhostRecon	Neutral	from
74596	Nvidia	Positive	from

```
[78] df.Sentiment.value_counts()
Negative    22358
Positive    20654
Neutral     18108
Irrelevant  12875
Name: Sentiment, dtype: int64
```

4. Exploratory Data Analysis (EDA):

- Visualizes the distribution of sentiment using a horizontal bar plot (df.Sentiment.value_counts().plot(kind='bar')).

```
df.groupby('Sentiment').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'), style='dashed', legend=True)
plt.gca().spines[['top', 'right']].set_visible(False)
```

- Visualizes sentiment distribution using a pie chart (plt.pie(sentiment_counts)).

```
encoder = LabelEncoder()
df['Sentiment'] = encoder.fit_transform(df['Sentiment'])
sentiment_counts = df['Sentiment'].value_counts()
sentiment_labels = encoder.inverse_transform(sentiment_counts.index)
plt.figure(figsize=(6, 6))
plt.pie(sentiment_counts, labels=sentiment_labels, autopct='%1.1f%%')
plt.title('Sentiment Distribution')
plt.show()
```

- Visualizes the count of entities vs. sentiment using a count plot (sns.countplot(data=df, x='Entity', hue='Sentiment')).

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Entity', hue='Sentiment')
plt.xticks(rotation=45, ha='right')
plt.title('Entity vs Sentiment')
plt.xlabel('Entity')
plt.ylabel('Count')
plt.legend(title='Sentiment')
plt.tight_layout()
plt.show()
```

- The code creates a word cloud visualization using the 'Entity' column of DataFrame df, importing libraries like WordCloud and matplotlib, and plots it using matplotlib, setting figure size, bilinear interpolation, and title.

```

plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Entity', hue='Sentiment')
plt.xticks(rotation=45, ha='right')
plt.title('Entity vs Sentiment')
plt.xlabel('Entity')
plt.ylabel('Count')
plt.legend(title='Sentiment')
plt.tight_layout()
plt.show()

```

5. Text Preprocessing: Uses SpaCy for text preprocessing (preprocess_data function):

- Checks if the text is a float and converts it to a string.
- Tokenizes the text using SpaCy, filters out punctuation and stop words, and lemmatizes the tokens.
- Joins the filtered tokens back into a string and stores it in a new column 'preprocessed_data' in the DataFrame.

```

[84] nlp = spacy.load("en_core_web_sm")

import spacy
nlp = spacy.load("en_core_web_sm")
def preprocess_data(text):
    if isinstance(text, float):
        return str(text)
    tokens = nlp(text)
    filtered_token = []
    for token in tokens:
        if token.is_punct or token.is_stop:
            continue
        filtered_token.append(token.lemma_)
    return " ".join(filtered_token)

[86] df['preprocessed_data'] = df['Text'].apply(preprocess_data)

```

6. Encodes the 'Sentiment' column using LabelEncoder from Scikit-learn (encoder.fit_transform(df.Sentiment)).

7. Splits the data into training and testing sets using train_test_split.

8. Uses TF-IDF Vectorizer (TfidfVectorizer) to convert text data into numerical features (v.fit_transform(x_train)).

9. Model Training and Evaluation:

- Trains a Naive Bayes classifier (MultinomialNB) and evaluates its performance using accuracy score and classification report.


```
[ ] from sklearn.naive_bayes import MultinomialNB

[94] nb_clf = MultinomialNB()
nb_clf.fit(X_train_normalized, y_train)

▼ MultinomialNB
MultinomialNB()

▶ y_pred = nb_clf.predict(X_test_normalized)

[116] from sklearn.metrics import accuracy_score, classification_report

[142] nb = accuracy_score(y_test, y_pred)
nb
```

- Trains a Random Forest classifier (RandomForestClassifier) and evaluates its performance using accuracy score and classification report

```
[99] from sklearn.ensemble import RandomForestClassifier

▶ rf_clf = RandomForestClassifier(n_estimators=60)
rf_clf.fit(X_train_normalized, y_train)

▼ RandomForestClassifier
RandomForestClassifier(n_estimators=60)

[135] y_pred = rf_clf.predict(X_test_normalized)

[138] ran = accuracy_score(y_test, y_pred)
ran
```

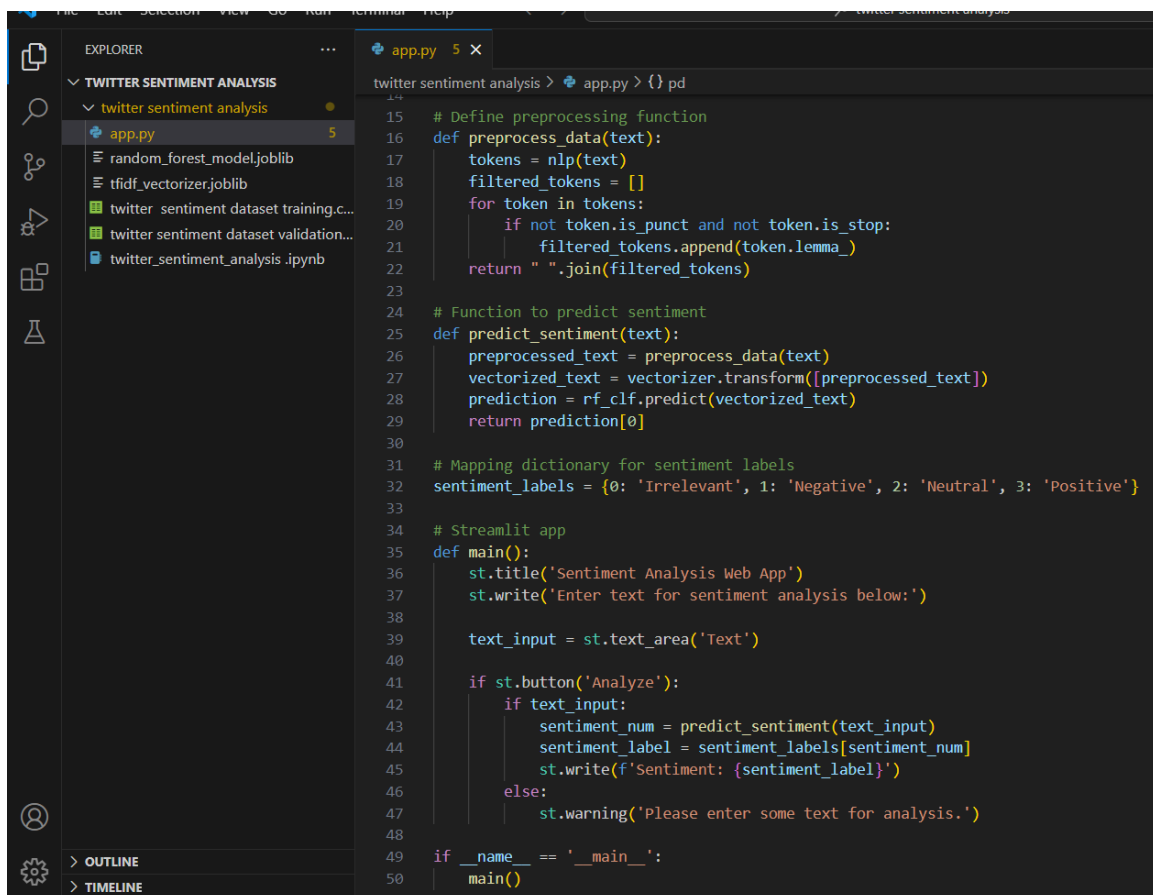
10. Splitting Data and Feature Extraction: Split the data into training and testing sets using `train_test_split` from `sklearn.model_selection`.

11. Prediction on Validation Data: Preprocess text data in the validation set using the `preprocess_data` function. Transform preprocessed text into TF-IDF representation using the `TfidfVectorizer`. Use the trained Random Forest classifier (`rf_clf`) to predict sentiment labels for validation data. Compare predicted labels with true labels from the validation data and print the results.

Web application:

- In VS code text preprocessing use the from the twitter data. Additionally, it loads a trained `RandomForestClassifier` model ('`random_forest_model.joblib`') for sentiment classification and a `TfidfVectorizer` ('`tfidf_vectorizer.joblib`') for text vectorization. This function tokenizes the text, filters out punctuation and stop words using `SpaCy`, and lemmatizes the tokens to normalize the text.

- This line creates a dictionary (sentiment_labels) that maps numeric sentiment labels to their corresponding sentiment categories. For example, 0 is mapped to 'Irrelevant', 1 to 'Negative', 2 to 'Neutral', and 3 to 'Positive'
- Streamlit App Main Function: This code defines the main function, which is the core of the Streamlit web application.
- It sets the title of the web app to 'Sentiment Analysis Web App' and displays a message prompting the user to enter text for sentiment analysis.
- It creates a text area using st.text_area where users can input their text for analysis.
- When the user clicks the 'Analyze' button (st.button('Analyze')), it checks if there is text input. If there is, it calls the predict_sentiment function to predict the sentiment of the input text, retrieves the corresponding sentiment label from sentiment_labels, and displays the sentiment label using st.write.
- If there is no text input when the 'Analyze' button is clicked, it displays a warning message asking the user to enter some text for analysis.



```

15 # Define preprocessing function
16 def preprocess_data(text):
17     tokens = nlp(text)
18     filtered_tokens = []
19     for token in tokens:
20         if not token.is_punct and not token.is_stop:
21             filtered_tokens.append(token.lemma_)
22     return " ".join(filtered_tokens)
23
24 # Function to predict sentiment
25 def predict_sentiment(text):
26     preprocessed_text = preprocess_data(text)
27     vectorized_text = vectorizer.transform([preprocessed_text])
28     prediction = rf_clf.predict(vectorized_text)
29     return prediction[0]
30
31 # Mapping dictionary for sentiment labels
32 sentiment_labels = {0: 'Irrelevant', 1: 'Negative', 2: 'Neutral', 3: 'Positive'}
33
34 # Streamlit app
35 def main():
36     st.title('Sentiment Analysis Web App')
37     st.write('Enter text for sentiment analysis below:')
38
39     text_input = st.text_area('Text')
40
41     if st.button('Analyze'):
42         if text_input:
43             sentiment_num = predict_sentiment(text_input)
44             sentiment_label = sentiment_labels[sentiment_num]
45             st.write(f'Sentiment: {sentiment_label}')
46         else:
47             st.warning('Please enter some text for analysis.')
48
49 if __name__ == '__main__':
50     main()

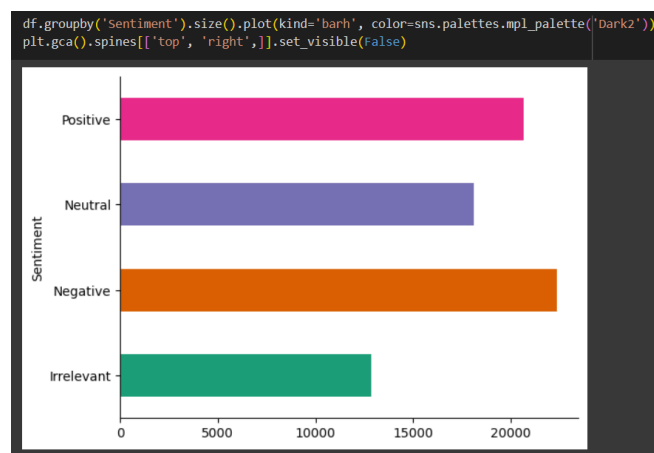
```

6. Results

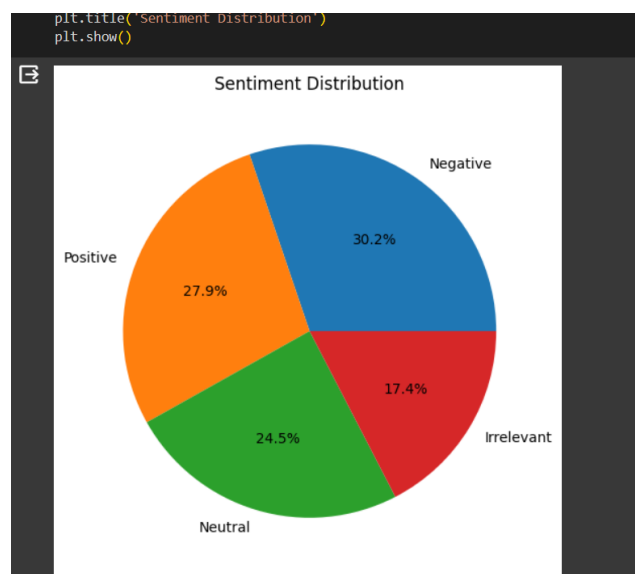
Results:

It begins by importing necessary libraries and loading the training dataset. The dataset is then cleaned, and the data types are checked. The data is visualized with some basic statistics, and the sentiment distribution is plotted. Next, a word cloud of entities and text is generated.. Multinomial Naive Bayes and Random Forest classifiers are trained and tested on the vectorized data. The classification results are evaluated using accuracy scores and classification reports. Additionally, the code reads a test data file, predicts the sentiment of a specific tweet, and compares the predicted result with the true value. Finally, a bar graph is plotted to compare the accuracy of the Naive Bayes and Random Forest classifiers.

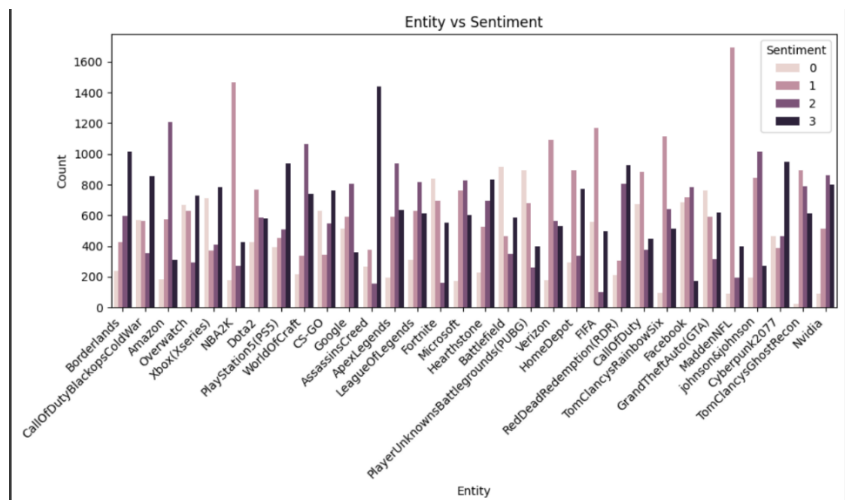
1. The graph between Sentiment



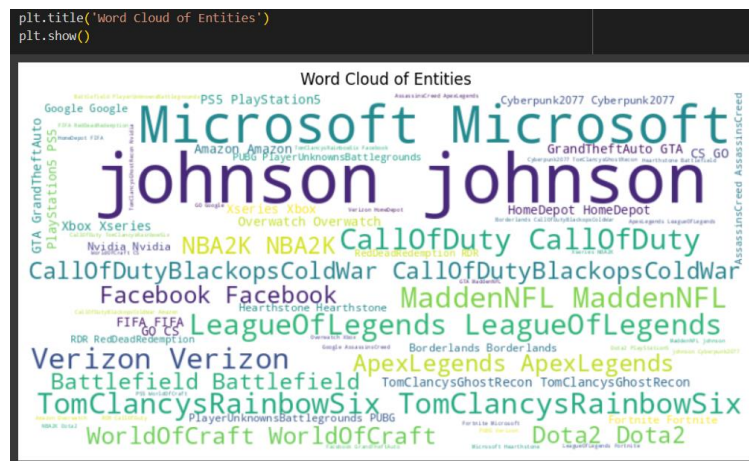
2. Pie chart on sentiment distribution



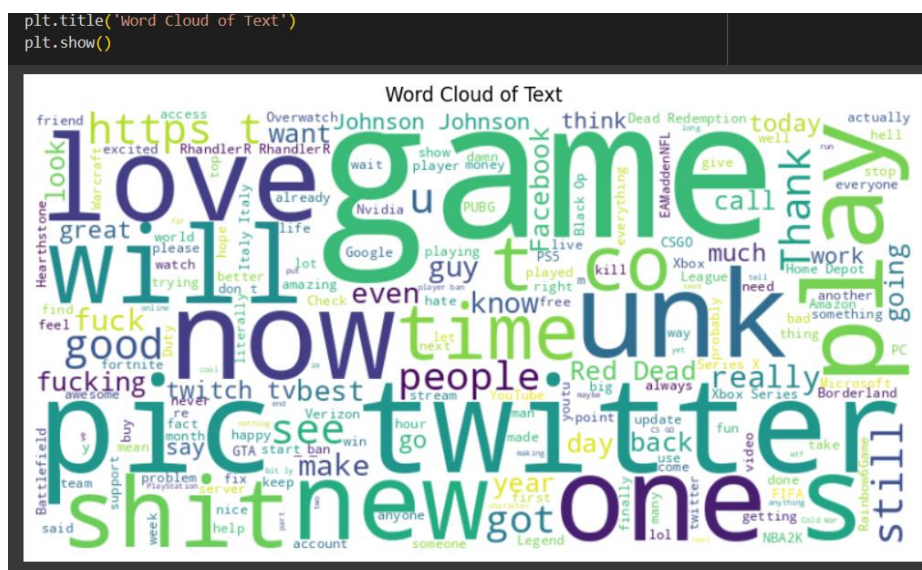
3. The graph between Entity vs Sentiment



4. Word Cloud of Entities



5. Word Cloud of Text



6. Accuracy and Classifications of Multinomial Naive Bayes (nb_clf) classifier.

```
[142] nb = accuracy_score(y_test, y_pred)
      nb
      0.7311980539225623
```

```
classification_report(y_test, y_pred)
print(classification)
```

	precision	recall	f1-score	support
0	0.95	0.46	0.62	2575
1	0.65	0.90	0.76	4472
2	0.84	0.63	0.72	3621
3	0.71	0.81	0.76	4131
accuracy			0.73	14799
macro avg	0.79	0.70	0.71	14799
weighted avg	0.77	0.73	0.72	14799

7. Accuracy and Classifications of a Random Forest (rf_clf) classifier.

```
[138] ran = accuracy_score(y_test, y_pred)
      ran
      0.9104669234407731
```

```
ran_class = classification_report(y_test, y_pred)
print(ran_class)
```

	precision	recall	f1-score	support
0	0.96	0.85	0.90	2575
1	0.93	0.93	0.93	4472
2	0.93	0.89	0.91	3621
3	0.85	0.94	0.89	4131
accuracy			0.91	14799
macro avg	0.92	0.90	0.91	14799
weighted avg	0.91	0.91	0.91	14799

9. Prediction of result.

```
[105] y_test = test_data.loc[998].Sentiment
      x = test_data.loc[998].Text
```

```
[106] preproc_x = preprocess_data(x)
```

```
[107] preproc_x
      'buy fraction Microsoft today small win'
```

```
x_testing = v.transform([preproc_x])
```

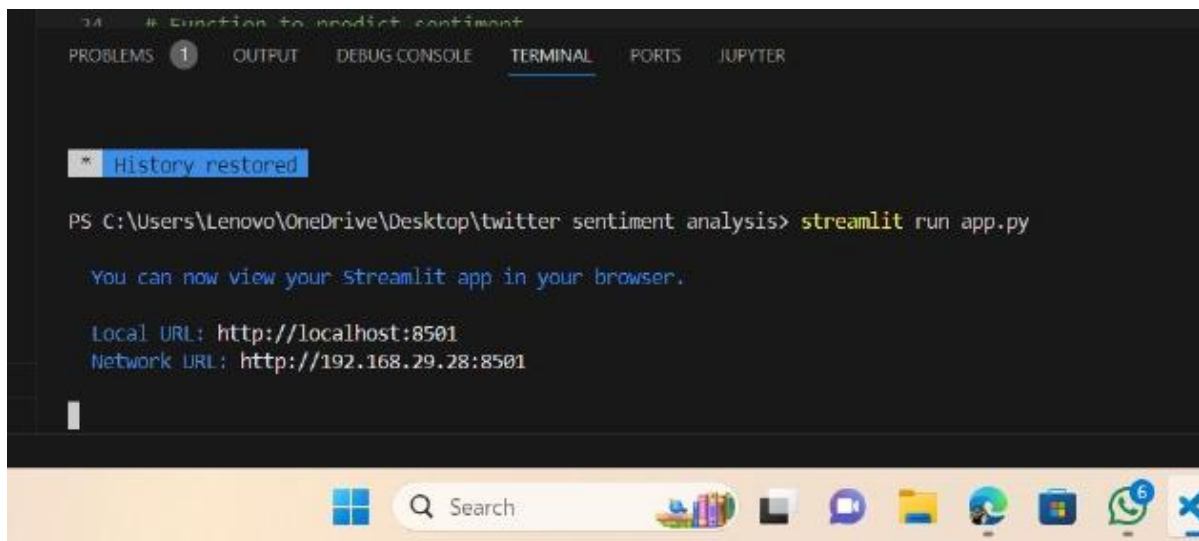
```
[109] x_testing.shape
      (1, 26746)
```

```
[110] y_pred = rf_clf.predict(x_testing)
```

```
[111] print(f"the predicted output is {y_pred} and it corresponds to {encoder.classes_[y_pred]} and\n the true value is {y_test}")
      the predicted output is [3] and it corresponds to [3] and
      the true value is Positive
```

```
[112] print(y_test)
      Positive
```

10. After run the code the link is provided output



```
74 # Function to predict sentiment
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

History restored

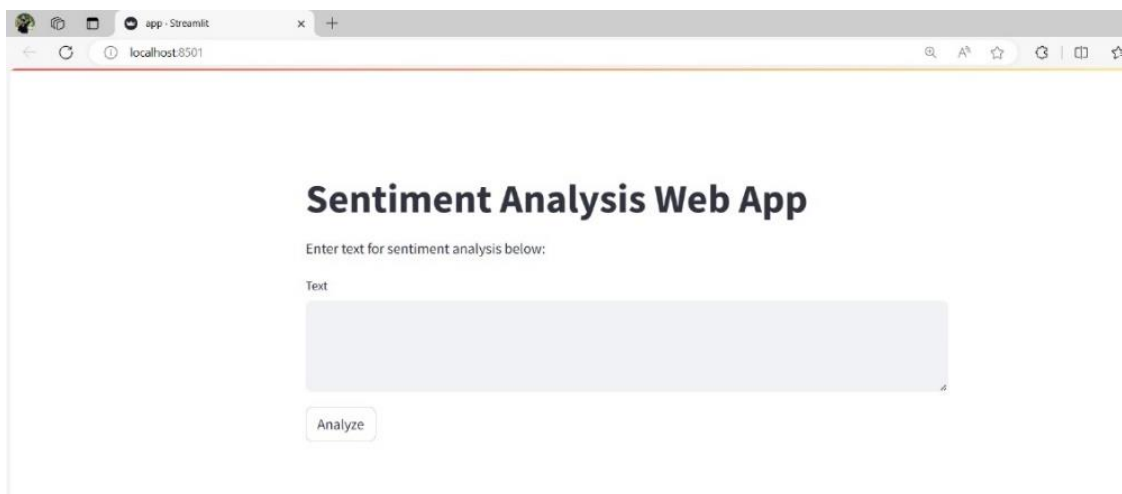
PS C:\Users\Lenovo\OneDrive\Desktop\twitter sentiment analysis> streamlit run app.py

You can now view your streamlit app in your browser.

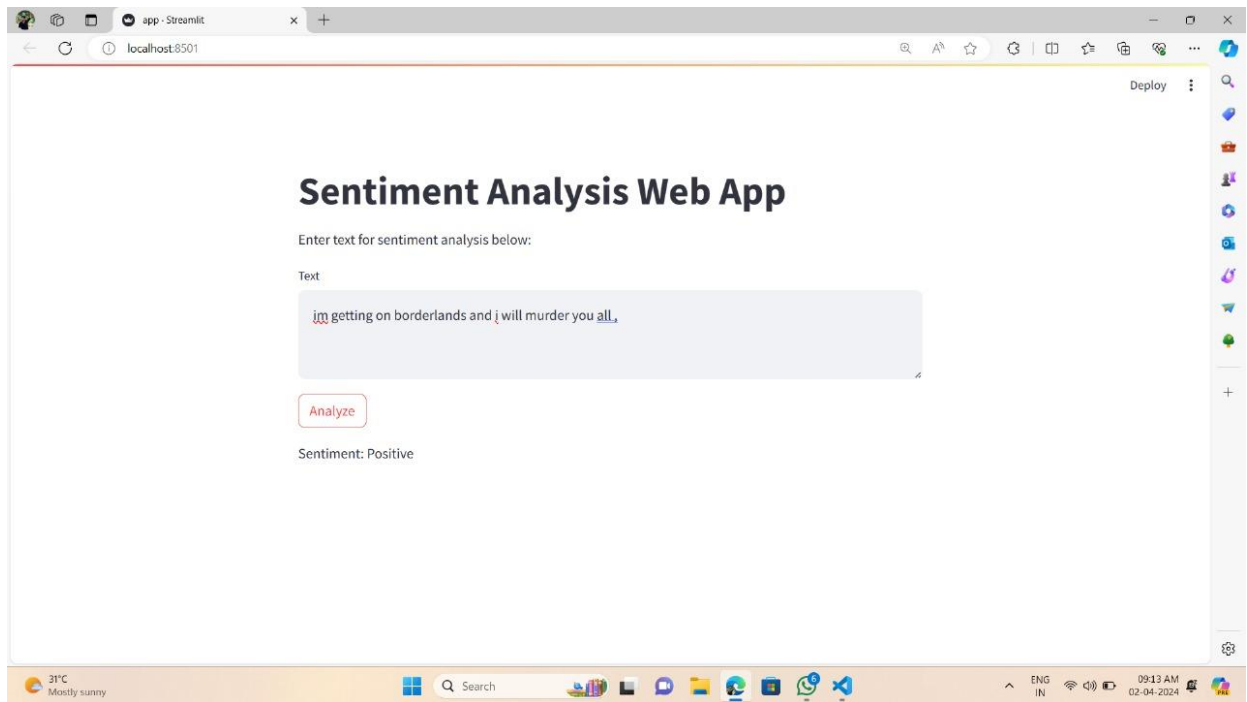
Local URL: http://localhost:8501
Network URL: http://192.168.29.28:8501
```

Link : <http://192.168.29.28:8507>

11. After click the link website link is open. In the website the sentiment analysis home page is open.



12. When the user clicks the 'Analyze' button (`st.button('Analyze')`), it checks if there is text input. If there is, it calls the `predict_sentiment` function to predict the sentiment of the input text, retrieves the corresponding sentiment label from `sentiment_labels`, and displays the sentiment label.



7. CONCLUSION

In conclusion, the code presented showcases a comprehensive approach to sentiment analysis on Twitter data using machine learning techniques. The process begins with data preparation, where the Twitter training data is loaded into a Pandas DataFrame and preprocessed to handle missing values and apply text preprocessing techniques using SpaCy. The sentiment labels are then encoded using LabelEncoder to prepare the data for model training.

Two machine learning classifiers, Multinomial Naive Bayes and Random Forest, are trained on the preprocessed data to predict sentiment. The code evaluates the performance of these classifiers using accuracy scores and classification reports on a test set, providing insights into the models' predictive capabilities.

Furthermore, the code includes testing with validation data, where a specific tweet from the validation set undergoes preprocessing and TF-IDF vectorization before being fed into the trained Random Forest classifier for sentiment prediction. The predicted sentiment is compared with the true sentiment value from the validation data, allowing for an assessment of the model's accuracy on new, unseen data. The accuracy score of , Multinomial Naive Bayes is 0.73 and Random Forest is 0.91.

So highest accuracy score are random forest, so we take random forest model will taken predict the sentiment text. Accuracy scores and classification reports used to assess model predictive capabilities. The highest accuracy model is random forest so we test the data random forest model. Preprocessed tweet from validation set preprocessed and TF-IDF vectorized before being fed into Random Forest classifier for sentiment prediction.

Overall, the demonstrates a structured approach to sentiment analysis, encompassing data preprocessing, model training, evaluation, and testing on real-world Twitter data. This process lays the groundwork for further refinement of the models and integration into larger applications for sentiment analysis and social media monitoring.

Overall Code

import all libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
import spacy
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

Call the CSV file

```
df= pd.read_csv(r"/content/twitter_training.csv")
df
```

Information of data set

```
df.columns = ['Tweet ID','Entity','Sentiment', 'Text']
df.head()
df.drop(['Tweet ID'], axis=1, inplace=True,)
print(df.shape)
df.dtypes
print(df.columns)
print(df.head()) # Display first few rows
print(df.tail())
```

#Null values

```
df.isnull().any()
```

```
df.isnull().sum()
```

```
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

Duplicated values

```
df.Text.duplicated().any()
```

```
df[df.Text == "from"]
```

```
df.Sentiment.value_counts()
```

Bar graph on sentiment data

```
df.groupby('Sentiment').size().plot(kind='barh',  
color=sns.palettes.mpl_palette('Dark2'))
```

```
plt.gca().spines[['top', 'right']].set_visible(False)
```

Pie chart on Sentiment

```
encoder = LabelEncoder()
```

```
df['Sentiment'] = encoder.fit_transform(df['Sentiment'])
```

```
sentiment_counts = df['Sentiment'].value_counts()
```

```
sentiment_labels = encoder.inverse_transform(sentiment_counts.index)
```

```
plt.figure(figsize=(6, 6))
```

```
plt.pie(sentiment_counts, labels=sentiment_labels, autopct='%1.1f%%')
```

```
plt.title('Sentiment Distribution')
```

```
plt.show()
```

Graph 'Entity vs Sentiment'

```
plt.figure(figsize=(10, 6))
```

```
sns.countplot(data=df, x='Entity', hue='Sentiment')
```

```
plt.xticks(rotation=45, ha='right')
```

```
plt.title('Entity vs Sentiment')
```

```

plt.xlabel('Entity')
plt.ylabel('Count')
plt.legend(title='Sentiment')
plt.tight_layout()
plt.show()

#'Word Cloud of Entities'

import sys

!{sys.executable} -m pip install wordcloud matplotlib

from wordcloud import WordCloud

import matplotlib.pyplot as plt

entities_text = ' '.join(df['Entity'].dropna())

wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(entities_text)

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title('Word Cloud of Entities')

plt.show()

#'Word Cloud of Text'

text = ' '.join(df['Text'].dropna())

wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title('Word Cloud of Text')

plt.show()

```

```

labels_counts = pd.DataFrame(df.Sentiment.value_counts())

# Preprocessing

import spacy

nlp = spacy.load("en_core_web_sm")

import spacy

nlp = spacy.load("en_core_web_sm")

def preprocess_data(text):
    if isinstance(text, float):
        return str(text)
    tokens = nlp(text)
    filtered_token = []
    for token in tokens:
        if token.is_punct or token.is_stop:
            continue
        filtered_token.append(token.lemma_)
    return " ".join(filtered_token)

df['preprocessed_data'] = df['Text'].apply(preprocess_data)

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

df.Sentiment = encoder.fit_transform(df.Sentiment)

encoder.classes_

X = df['preprocessed_data']

y = df['Sentiment']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y, random_state=42,
test_size=0.2, stratify=y)

```

```

from sklearn.feature_extraction.text import TfidfVectorizer
v = TfidfVectorizer()
X_train_normalized = v.fit_transform(x_train)
X_test_normalized = v.transform(x_test)

# MultinomialNB Random

from sklearn.naive_bayes import MultinomialNB
nb_clf = MultinomialNB()
nb_clf.fit(X_train_normalized, y_train)
y_pred = nb_clf.predict(X_test_normalized)

from sklearn.metrics import accuracy_score, classification_report
nb = accuracy_score(y_test, y_pred)
nb
classification = classification_report(y_test, y_pred)
print(classification)

# RandomForestClassifier

from sklearn.ensemble import RandomForestClassifier
rf_clf = RandomForestClassifier(n_estimators=60)
rf_clf.fit(X_train_normalized, y_train)
y_pred = rf_clf.predict(X_test_normalized)
ran = accuracy_score(y_test, y_pred)
ran
ran_class = classification_report(y_test, y_pred)
print(ran_class)

#split the dataset

test_data = pd.read_csv('/content/twitter_validation.csv', names=['Tweet
ID','Entity','Sentiment', 'Text'])

test_data

```

#Testing

```
y_test = test_data.loc[998].Sentiment
x = test_data.loc[998].Text
preproc_x = preprocess_data(x)
preproc_x
x_testing = v.transform([preproc_x])
x_testing.shape
y_pred = rf_clf.predict(x_testing)
print(f'the predicted output is {y_pred} and it corresponds to
{encoder.classes_[y_pred]} and\n the true value is {y_test}')
print(y_test)
```

WEB APPLICATION

```
import joblib
import pandas as pd
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
import streamlit as st
# Load SpaCy model
nlp = spacy.load("en_core_web_sm")
# Load trained model and vectorizer
rf_clf = joblib.load('random_forest_model.joblib')
vectorizer = joblib.load('tfidf_vectorizer.joblib')
# Define preprocessing function
def preprocess_data(text):
    tokens = nlp(text)
    filtered_tokens = []
```

```

    for token in tokens:
        if not token.is_punct and not token.is_stop:
            filtered_tokens.append(token.lemma_)
    return " ".join(filtered_tokens)

# Function to predict sentiment
def predict_sentiment(text):
    preprocessed_text = preprocess_data(text)
    vectorized_text = vectorizer.transform([preprocessed_text])
    prediction = rf_clf.predict(vectorized_text)
    return prediction[0]

# Mapping dictionary for sentiment labels
sentiment_labels = {0: 'Irrelevant', 1: 'Negative', 2: 'Neutral', 3: 'Positive'}

# Streamlit app
def main():
    st.title('Sentiment Analysis Web App')
    st.write('Enter text for sentiment analysis below:')
    text_input = st.text_area('Text')
    if st.button('Analyze'):
        if text_input:
            sentiment_num = predict_sentiment(text_input)
            sentiment_label = sentiment_labels[sentiment_num]
            st.write(f'Sentiment: {sentiment_label}')
        else:
            st.warning('Please enter some text for analysis.')

if __name__ == '__main__':
    main()

```

PROJECT LINKS

<https://github.com/hsahbuSM/Twitter-sentiment-analysis>

SUBMITTED BY :

M.Subhash

N. Gopala Krishna