INSTITUTE FOR ADVANCED COMPUTING ANDSOFTWARE DEVELOPMENT (IACSD), AKURDI, PUNE

Documentation on

"Twitter Under the Microscope: Unveiling Emotions with NLP Twitter Sentiments through Machine Learning and Visualization"

PG-DBDA March 2023

Submitted By:

Group no. 08

Roll No.	Name:
233522	Madhura Patil
233532	Rutuja Pokharkar

Mr. Abhijit Nagargoje
Project Guide

Mr. Rohit Puranik
Centre Coordinator

Abstract

In our project titled "Twitter Sentiment Analysis using NLP," we leverage cutting-edge Natural Language Processing (NLP) techniques to extract meaningful insights from Twitter data. By implementing a multi-step approach involving meticulous data pre-processing, sophisticated tokenization, and robust machine learning algorithms, we adeptly categorize individual tweets into three distinct sentiment categories: positive, negative, and neutral. This intricate process allows us to effectively capture the prevailing sentiments within the Twitter verse.

A key highlight of our project lies in its ability to uncover and visualize sentiment trends over time. Through intuitive and informative visualizations, we present a dynamic representation of how sentiments evolve, providing deep-seated understandings of public opinions on various topics. The amalgamation of our expertise in NLP, rigorous data pre-processing techniques, state-of-the-art machine learning methods, and impactful data visualization culminates in a showcase of proficiency that brings to light the remarkable potential of sentiment analysis in gleaning insights from the vast realm of social media data.

Acknowledgement

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide, **Mr. Abhijit Nagargoje** for providing me with the right guidance and advice at the crucial juncture sand for showing me the right way. I extend my sincere thanks to our respected Centre Co-Ordinator **Mr. Rohit Puranik**, for allowing us to use the facilities available. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Madhura Patil(233522)

Rutuja Pokharkar (233532)

Index

Chapter 1: Introduction

1.1 Background and Motivation

Introduction to Sentiment Analysis and Its Significance

In an age marked by digital connectivity and communication, sentiments expressed through text have gained prominence as valuable sources of information. Sentiment analysis, a branch of Natural Language Processing (NLP), offers tools to decipher the emotions, opinions, and attitudes conveyed within textual content. This project explores sentiment analysis and its pivotal role in understanding human expressions, influencing decision-making, and uncovering insights from the vast sea of text data.

- Growing Importance of Social Media Data for Sentiment Analysis

As online interactions intensify, social media platforms have emerged as fertile grounds for collecting diverse textual data. The collective voice of individuals on platforms like Twitter can yield invaluable insights into public sentiment, making social media a prime arena for sentiment analysis. This project recognizes the surge in the significance of social media data and aims to harness it for sentiment analysis.

- Brief Explanation of Twitter as a Data Source

Among the myriad of social media platforms, Twitter stands out as a microblogging platform where users share thoughts, opinions, and news in short text snippets called tweets. With millions of active users and an unending stream of real-time conversations, Twitter provides an ideal playground for sentiment analysis. This project takes Twitter as its data source to delve into sentiment analysis using advanced NLP techniques.

Through this project, we aim to showcase the power of sentiment analysis, the role of social media data, and the specific utility of Twitter in providing a rich pool of textual content for sentiment analysis. The subsequent chapters will delve into the methodologies, techniques, and outcomes of applying sentiment analysis to Twitter data.

1.2 Problem Statement

- Definition of the Problem: Analyzing Sentiments in Twitter Data using NLP Techniques

The problem at hand revolves around the task of sentiment analysis within the realm of Twitter data. The objective is to employ advanced Natural Language Processing (NLP) techniques to classify sentiments expressed within tweets into categories such as positive, negative, or neutral. This entails developing a methodology to automatically determine the emotional tone conveyed by the textual content of tweets.

- Importance of Categorizing Sentiments for Various Applications

The significance of sentiment analysis spans a range of applications across industries. Accurate sentiment classification offers valuable insights into public opinions, which, in turn, influences decision-making processes. Businesses can tailor marketing strategies based on customer sentiment, governments can gauge public reactions to policies, and researchers can monitor social trends.

In this project, the focus is on understanding and effectively addressing the challenges of sentiment analysis using Twitter data. By achieving reliable sentiment categorization, we contribute to the enhancement of decision-making, trend tracking, and perception analysis in the digital age. The subsequent sections will delve into the methodologies employed to address this problem and the strategies used to achieve accurate sentiment classification

1.3 Objectives

- ➢ Clear Goals and Aims of the Project

The primary goal of this project is to develop a robust sentiment analysis system using advanced Natural Language Processing (NLP) techniques, specifically targeting Twitter data. The objectives include:

1. Implementing data pre-processing techniques to clean and structure raw Twitter data for analysis.

2. Applying tokenization to break down textual content into meaningful units for feature extraction.

3. Designing and training machine learning models to accurately categorize tweets into positive, negative, or neutral sentiments.

4. Creating informative visualizations to illustrate sentiment trends over time and across different topics.

- ➢ Scope and Limitations of the Analysis;

- Utilizing a diverse dataset of tweets to ensure broad coverage of topics and sentiments.

- Employing state-of-the-art NLP techniques for preprocessing and feature extraction.

- Incorporating machine learning models capable of handling sentiment classification tasks.

However, the analysis has its limitations:

- The sentiment analysis might not capture sarcasm, irony, or complex linguistic nuances.

- The accuracy of sentiment classification can be influenced by the quality of training data.

- The analysis might not consider the cultural context that impacts sentiment interpretation.

- Real-time data acquisition and processing may not be addressed due to computational constraints.

Chapter 2: Literature Review

2.1 Sentiment Analysis

Explanation of Sentiment Analysis and Its Applications

Sentiment analysis, also known as opinion mining, is a subfield of Natural Language Processing (NLP) that involves the automated identification and classification of sentiments expressed in textual data. The aim is to determine whether the sentiment behind a piece of text is positive, negative, or neutral. This analysis is pivotal in understanding public attitudes, emotions, and opinions, and has found applications across various domains.

- ➢ In the digital age, sentiment analysis is extensively used in:

- Business and Marketing: Brands assess consumer perceptions and tailor marketing strategies accordingly.

- Social Media Monitoring: Public sentiment on platforms like Twitter helps gauge reactions to events and topics.

- Political Analysis: Public opinions towards candidates and policies are monitored during campaigns.

- Customer Service:Sentiment analysis aids in assessing customer satisfaction and resolving issues.

- ➢ Overview of Sentiment Analysis Techniques and Methodologies
- ➢ the methodologies for sentiment analysis can be broadly classified into three categories:

1. Rule-Based Approaches: These methods involve creating a set of rules or patterns to determine sentiment based on predefined keywords, phrases, or linguistic rules. However, they might not handle the complexities of language well.

2. Machine Learning Approaches: Machine learning models, particularly supervised learning, have gained popularity. They involve training classifiers on labelled datasets, enabling them to generalize sentiment classification on new data. Common algorithms include Support Vector Machines (SVM), Naive Bayes, and Deep Learning models like Recurrent Neural Networks (RNNs) and Transformers.

3. Lexicon-Based Approaches: These methods rely on sentiment lexicons or dictionaries containing words associated with specific sentiments. The sentiment of a text is determined based on the presence and polarity of these words. However, they might struggle with context and sarcasm.

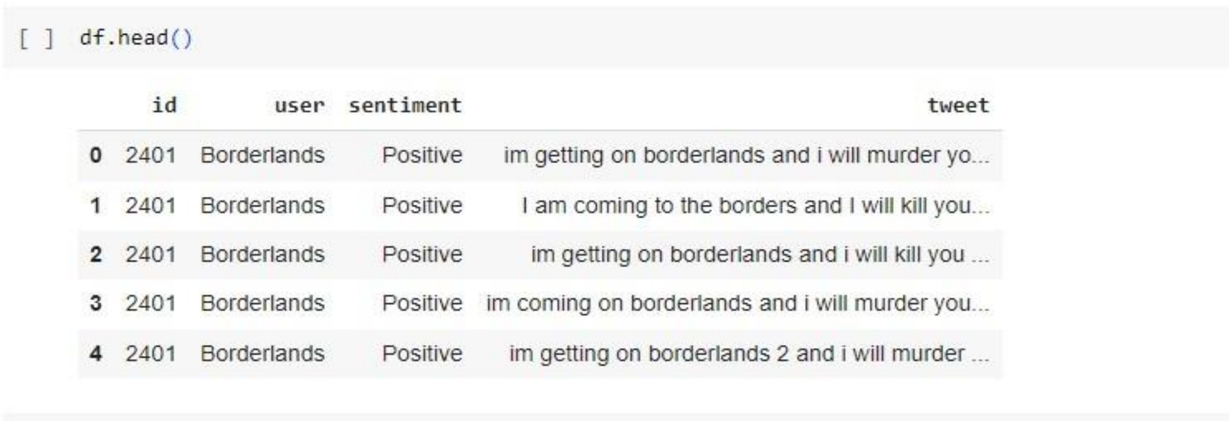
2.2 Natural Language Processing (NLP)

Introduction to NLP and Its Role in Sentiment Analysis

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and human language. NLP techniques enable computers to understand, interpret, and generate human language in a form that is valuable and meaningful. In the context of sentiment analysis, NLP plays a pivotal role in deciphering the sentiments conveyed within textual content.

- Key NLP Techniques Used in Sentiment Analysis

Several NLP techniques are employed in sentiment analysis to extract meaningful insights from text data:



```
[ ]  df.head()
```

	id	user	sentiment	tweet
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...

Fig 1: View of dataset

1.Tokenization: This process involves breaking down textual content into smaller units, such as words or phrases (tokens). Tokenization lays the foundation for subsequent analysis by converting raw text into manageable components.

2. Text Preprocessing:This step includes tasks like removing punctuation, converting text to lowercase, and eliminating irrelevant characters. Preprocessing ensures that the text is clean and ready for analysis.

3. Stop Word Removal:Common words like "and," "the," and "is" contribute little to sentiment analysis. Removing these stop words reduces noise in the data.

4. Stemming and Lemmatization: These techniques reduce words to their root forms, reducing inflections and variations. For instance, "running" and "ran" might be stemmed to "run."

5. Feature Extraction:This involves converting text into numerical features that machine learning models can understand. Techniques like Bag-of-Words and TF-IDF (Term Frequency-Inverse Document Frequency) are commonly used.

6. Word Embeddings: These are dense vector representations of words that capture semantic relationships. Word embeddings like Word2Vec and GloVe enhance the understanding of word context and meaning.

7. N-grams: These are combinations of adjacent words in a sequence. Analyzing n-grams captures contextual information that single words might miss.

8. Named Entity Recognition (NER): NER identifies and classifies named entities (such as names of people, organizations, and locations) in text. This can provide context to sentiment analysis.

2.3 Previous Studies

Review of Related Projects and Research Papers on Twitter Sentiment Analysis

Numerous projects and research papers have explored the realm of sentiment analysis on Twitter data, offering diverse insights into methods and outcomes. These studies have contributed to refining techniques and approaches for effectively deciphering sentiments in tweets

Highlighting Different Approaches and Methodologies Used

1. Supervised Machine Learning Models:Many studies employ supervised learning algorithms such as Support Vector Machines (SVM), Naive Bayes, and Random Forests to classify sentiments in tweets. These models learn from labeled data and generalize their learning to new, unseen tweets.

2. Deep Learning Techniques: Deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been applied to capture complex relationships within text. Long Short-Term Memory (LSTM) networks and Transformer models have shown promising results due to their ability to understand sequential and contextual information.

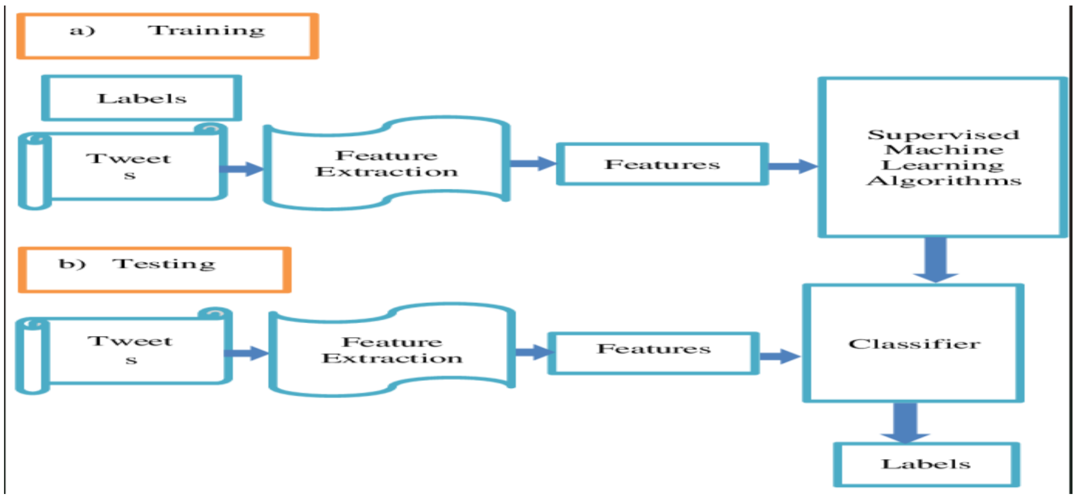
3. Lexicon-Based Approaches: Some studies utilize sentiment lexicons to determine sentiment polarity in tweets. Lexicons contain lists of words and their associated sentiment scores, aiding in sentiment classification based on word presence and polarity.

4.Time-Series Analysis: Analyzing sentiment trends over time provides insights into changing opinions and reactions. Time-series analysis has been used to track sentiment shifts during events, product launches,

Chapter 3 data collection and data processing

3.1 Workflow of Project:

The diagram below shows the workflow of this project.

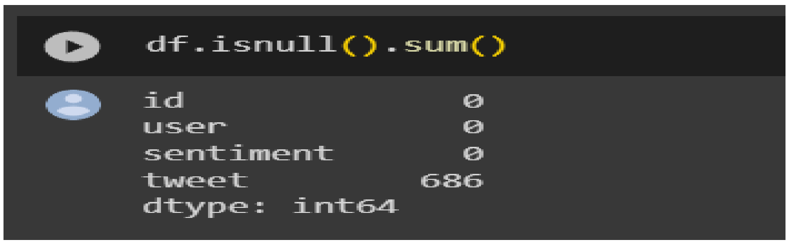


3.2 Data Cleaning:

The data can have many irrelevant, missing parts null values , username, special symbols etc . To handle this part, data cleaning is done.

1. Remove null values

The attributes present in our dataset had 686 null values and therefore for data cleaning purpose we dropped them.



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

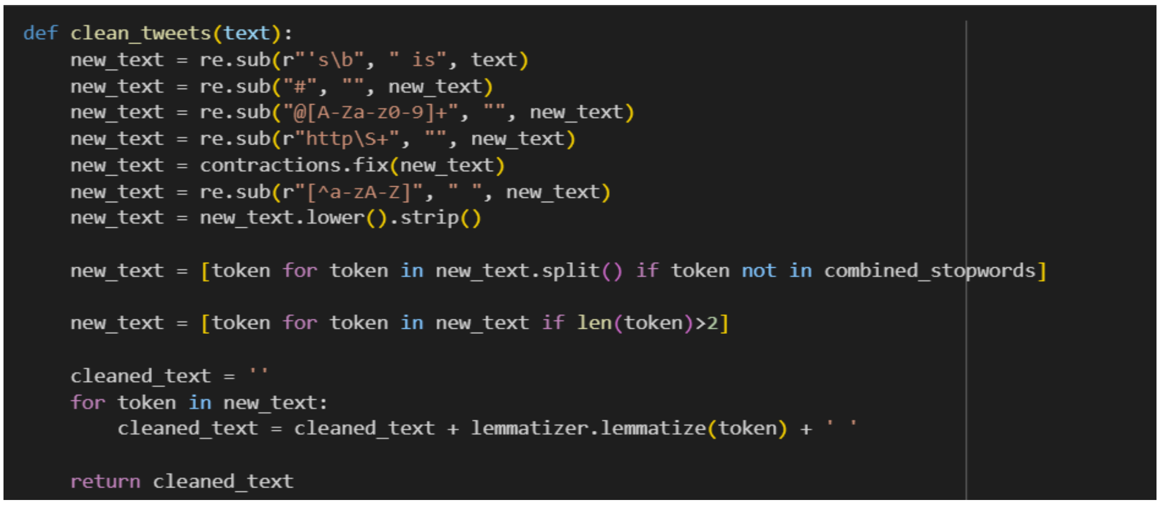
```
id             0
user           0
sentiment      0
tweet        686
dtype: int64
```

2. Removing StopWords

The stop words present in our dataset don’t have any significance when we train our model. We don’t lose any information while training the data. But removing stopwords will make the size of our dataset small and this reduces training time.

3. Lemmatizing our dataset

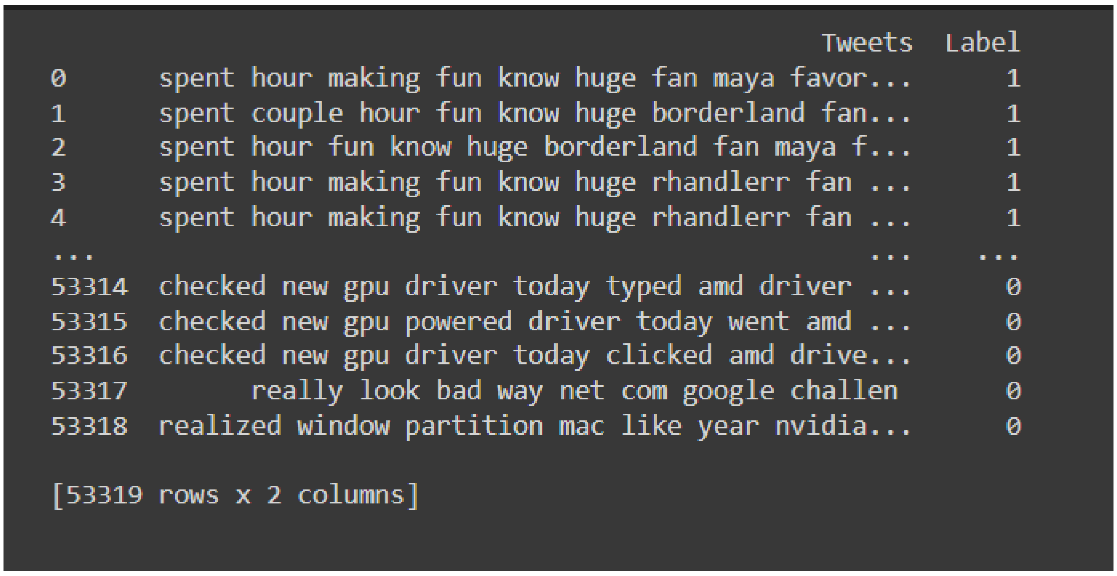
The data contains many words that can be reduced down to its basic form. This step will reduce training time but the dataset will still retain its original meaning.



```
def clean_tweets(text):
    new_text = re.sub(r"'s\b", " is", text)
    new_text = re.sub("#", "", new_text)
    new_text = re.sub("@[A-Za-z0-9]+", "", new_text)
    new_text = re.sub(r"http\S+", "", new_text)
    new_text = contractions.fix(new_text)
    new_text = re.sub(r"[^a-zA-Z]", " ", new_text)
    new_text = new_text.lower().strip()

    new_text = [token for token in new_text.split() if token not in combined_stopwords]

    new_text = [token for token in new_text if len(token)>2]

    cleaned_text = ''
    for token in new_text:
        cleaned_text = cleaned_text + lemmatizer.lemmatize(token) + ' '

    return cleaned_text
```

3.2.1 Label encoding:

To make the data understandable or in human readable form, the training data is often labeled in words. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated.



```
                                                  Tweets  Label
0      spent hour making fun know huge fan maya favor...      1
1      spent couple hour fun know huge borderland fan...      1
2      spent hour fun know huge borderland fan maya f...      1
3      spent hour making fun know huge rhandlerr fan ...      1
4      spent hour making fun know huge rhandlerr fan ...      1
...                                                  ...    ...
53314  checked new gpu driver today typed amd driver ...      0
53315  checked new gpu powered driver today went amd ...      0
53316  checked new gpu driver today clicked amd drive...      0
53317        really look bad way net com google challen      0
53318  realized window partition mac like year nvidia...      0

[53319 rows x 2 columns]
```

3.2.2 : <u>Preprocessing the data</u>

The function provided is a Python Function for calculating the subjectivity of a given text using the TextBlob library .The Subjectivity of a text indicates how subjective or opinion-based text is. It typically ranges from 0 to 1, with 0 being highly objective (factual) and 1 being highly subjective (opiniated). Here's how the function works:

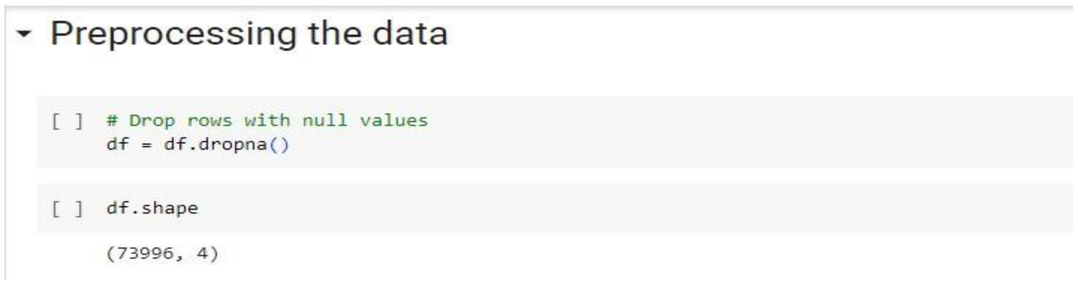
1. It takes a single argument ,text which represents the text you want to analyze for subjectivity.

2. Inside the function , it uses TextBlob to perform sentiment analysis on the input text. Specifically
   It calls the sentiment subjectivity method of TextBlob, which calculates the subjectivity score of the text.
3. The function then returns this subjectivity score as its output.

3.3Data pre-processing

Cleaning and Handling Noisy Data

Raw Twitter data often contains extraneous elements like HTML tags, URLs, and special characters that can impede accurate sentiment analysis. Cleaning involves removing these artefacts to ensure the quality of the data.

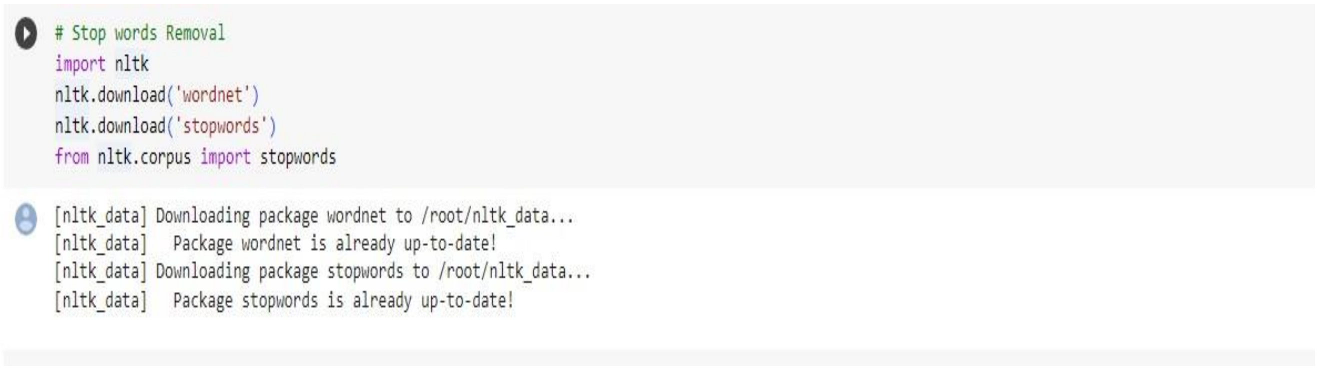

Preprocessing the data

```
[ ] # Drop rows with null values
    df = df.dropna()

[ ] df.shape

    (73996, 4)
```

Text Normalization Techniques:- Lowercasing: Converting all text to lowercase helps maintain uniformity in text data. This process ensures that words with different cases are treated as the same, minimizing discrepancies in sentiment analysis. For instance, "Hello" and "hello" are both converted to "hello."

- Removing Special Characters:Special characters, punctuations, and symbols can hinder sentiment analysis algorithms by introducing unnecessary noise. Using regular expressions, these characters are removed to simplify the text while retaining its core meaning. For instance, "Can't wait!!!" becomes "Cant wait."
- Handling of Emojis, Hashtags, and Mentions
- Emojis: Emojis are essential components of sentiment expression, conveying emotions that words alone might not capture. They can be preserved, substituted with textual equivalents, or removed based on the analysis's goals. For instance, "I'm so happy 😊" might be transformed into "I'm so happy smiley face."
- Hashtags: Hashtags often carry contextual information or emphasize sentiments. Extracting meaningful words from hashtags and separating camel-cased words (e.g., #FridayFeeling to "Friday Feeling") aids in preserving sentiment-related content.
- Mentions:User mentions provide context and might influence the sentiment of the text. While removing them is an option, retaining mentions can offer additional information. For example, "Great job, @JohnDoe!" remains as is.

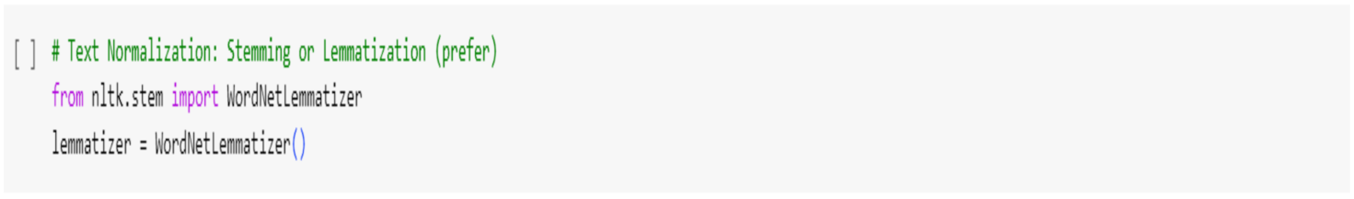
➢ Removing Stop Words and Performing Tokenization:
➢ Stop Words:Stop words are common words that add little semantic value to the analysis, such as "and," "the," "in," and "is." Removing these words reduces the dimensionality of the data and focuses the analysis on content-rich words.



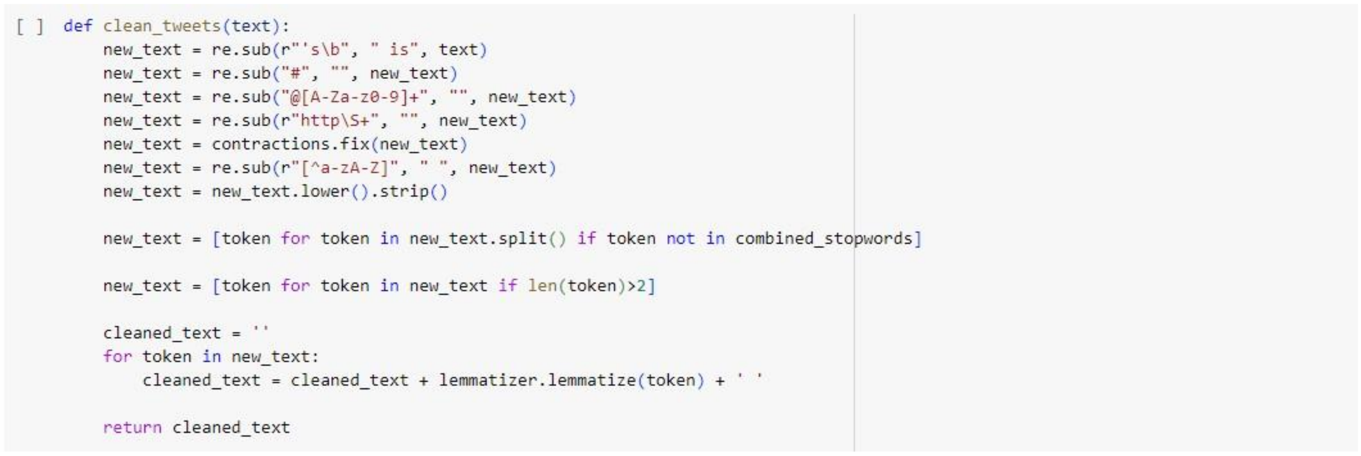
```
# Stop words Removal
import nltk
nltk.download('wordnet')
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Tokenization: Tokenization divides the text into smaller units, typically words or phrases. It lays the foundation for feature extraction and analysis. For instance, the sentence "I love reading books" is tokenized into ["I", "love", "reading", "books"].



```
# Text Normalization: Stemming or Lemmatization (prefer)
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```



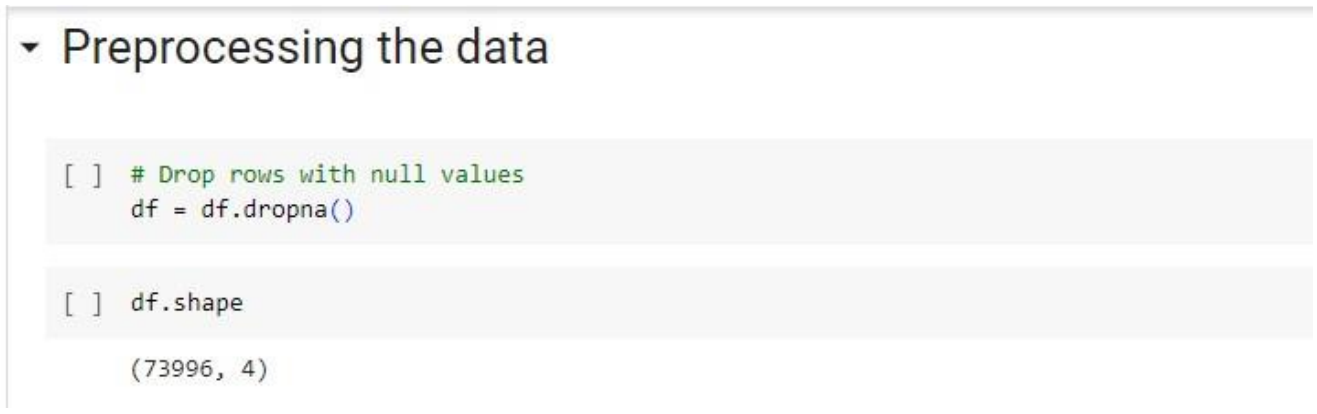
```
def clean_tweets(text):
    new_text = re.sub(r"'s\b", " is", text)
    new_text = re.sub("#", "", new_text)
    new_text = re.sub("@[A-Za-z0-9]+", "", new_text)
    new_text = re.sub(r"http\S+", "", new_text)
    new_text = contractions.fix(new_text)
    new_text = re.sub(r"[^a-zA-Z]", " ", new_text)
    new_text = new_text.lower().strip()

    new_text = [token for token in new_text.split() if token not in combined_stopwords]

    new_text = [token for token in new_text if len(token)>2]

    cleaned_text = ''
    for token in new_text:
        cleaned_text = cleaned_text + lemmatizer.lemmatize(token) + ' '

    return cleaned_text
```

By comprehensively cleaning and pre-processing the data, we ensure that the text is transformed into a structured and normalized format, suitable for accurate sentiment analysis. These pre-processing steps enhance the quality of subsequent analysis and sentiment classification.

3.2 Data Preprocessing

Cleaning and Handling Noisy Data

Raw Twitter data often contains extraneous elements like HTML tags, URLs, and special characters that can impede accurate sentiment analysis. Cleaning involves removing these artefacts to ensure the quality of the data.



Preprocessing the data

```
[ ] # Drop rows with null values
    df = df.dropna()
```

```
[ ] df.shape
```

```
(73996, 4)
```

Fig 2: Dropping the null values

Text Normalization Techniques:

- Lowercasing: Converting all text to lowercase helps maintain uniformity in text data. This process ensures that words with different cases are treated as the same, minimizing discrepancies in sentiment analysis. For instance, "Hello" and "hello" are both converted to "hello."

- Removing Special Characters:Special characters, punctuations, and symbols can hinder sentiment analysis algorithms by introducing unnecessary noise. Using regular expressions, these characters are removed to simplify the text while retaining its core meaning. For instance, "Can't wait!!!" becomes "Cant wait."

Handling of Emojis, Hashtags, and Mentions:

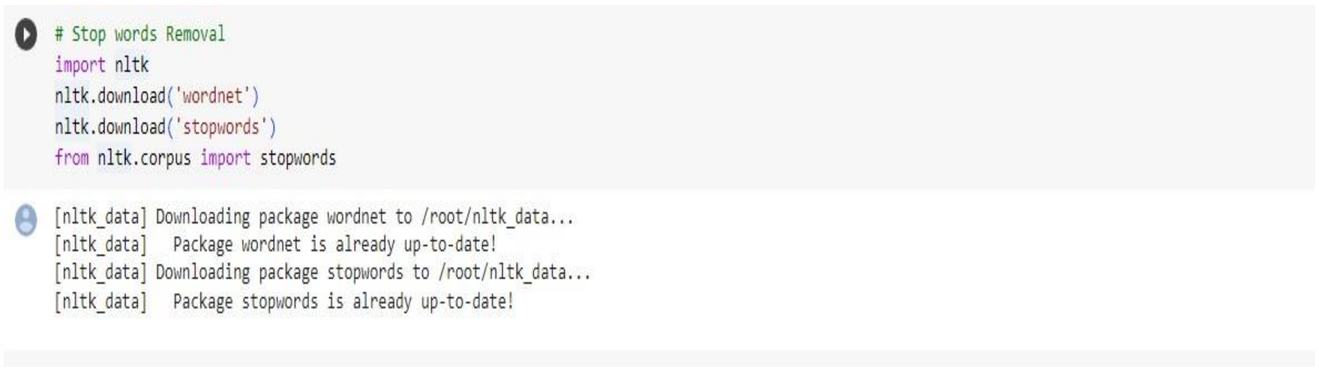
- Emojis: Emojis are essential components of sentiment expression, conveying emotions that words alone might not capture. They can be preserved, substituted with textual equivalents, or removed based on the analysis's goals. For instance, "I'm so happy 😊" might be transformed into "I'm so happy smiley face."

- Hashtags: Hashtags often carry contextual information or emphasize sentiments. Extracting meaningful words from hashtags and separating camel-cased words (e.g., #FridayFeeling to "Friday Feeling") aids in preserving sentiment-related content.

- Mentions:User mentions provide context and might influence the sentiment of the text. While removing them is an option, retaining mentions can offer additional information. For example, "Great job, @JohnDoe!" remains as is.

➢ Removing Stop Words and Performing Tokenization:

- Stop Words:Stop words are common words that add little semantic value to the analysis, such as "and," "the," "in," and "is." Removing these words reduces the dimensionality of the data and focuses the analysis on content-rich words.

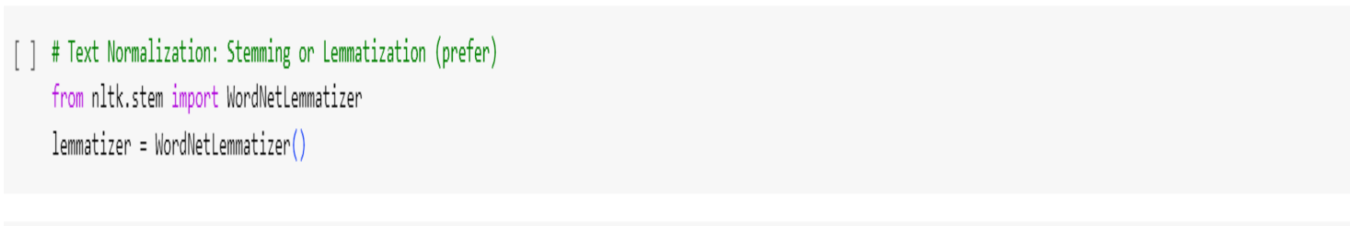


```
# Stop words Removal
import nltk
nltk.download('wordnet')
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

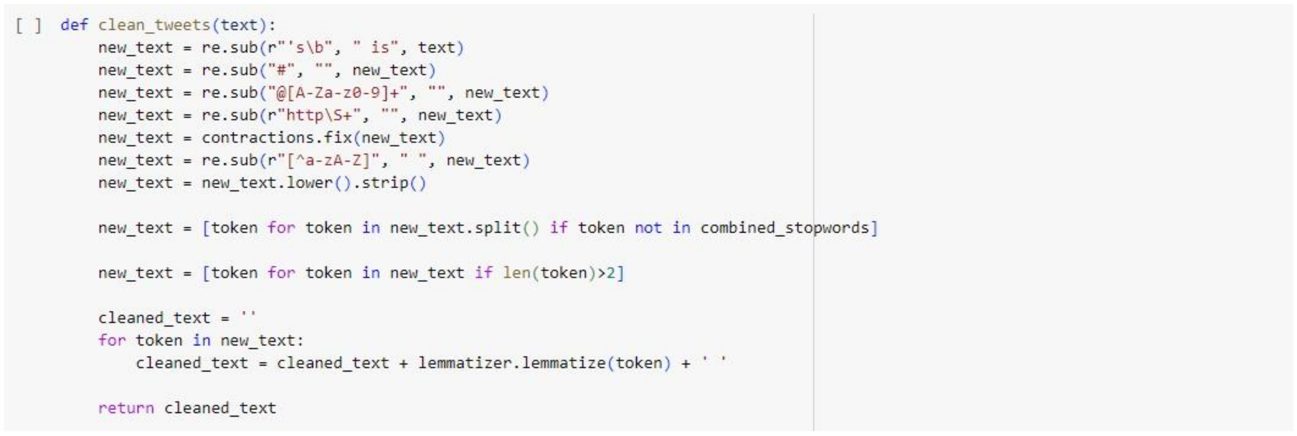
Fig 3: Stop words removal

Tokenization: Tokenization divides the text into smaller units, typically words or phrases. It lays the foundation for feature extraction and analysis. For instance, the sentence "I love reading books" is tokenized into ["I", "love", "reading", "books"].



```
[ ] # Text Normalization: Stemming or Lemmatization (prefer)
    from nltk.stem import WordNetLemmatizer
    lemmatizer = WordNetLemmatizer()
```

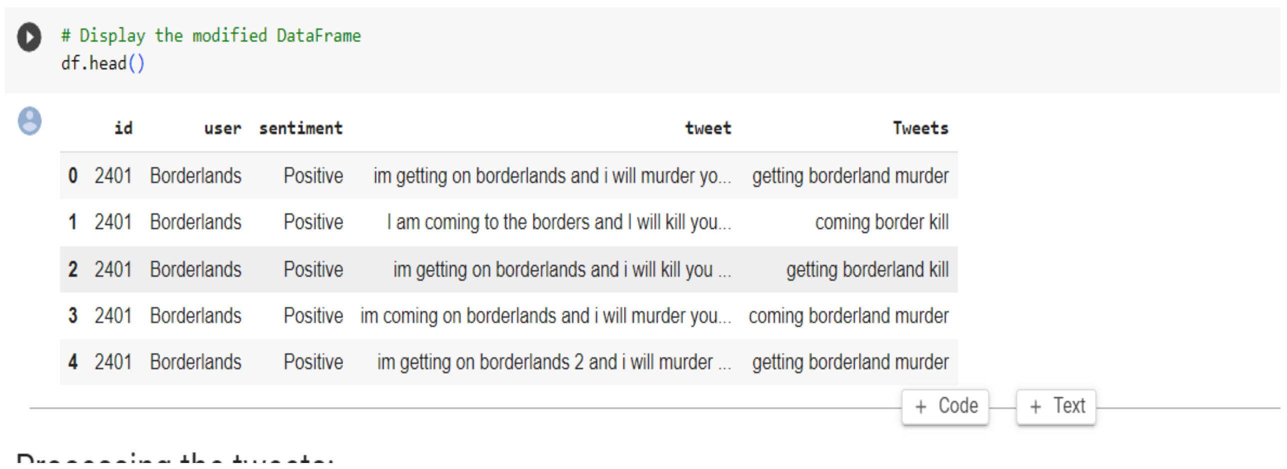
Fig 4: Text Normalization



```
[ ] def clean_tweets(text):
        new_text = re.sub(r"'s\b", " is", text)
        new_text = re.sub("#", "", new_text)
        new_text = re.sub("@[A-Za-z0-9]+", "", new_text)
        new_text = re.sub(r"http\S+", "", new_text)
        new_text = contractions.fix(new_text)
        new_text = re.sub(r"[^a-zA-Z]", " ", new_text)
        new_text = new_text.lower().strip()

        new_text = [token for token in new_text.split() if token not in combined_stopwords]

        new_text = [token for token in new_text if len(token)>2]

        cleaned_text = ''
        for token in new_text:
            cleaned_text = cleaned_text + lemmatizer.lemmatize(token) + ' '

        return cleaned_text
```

Fig 5: Preprocessing Function

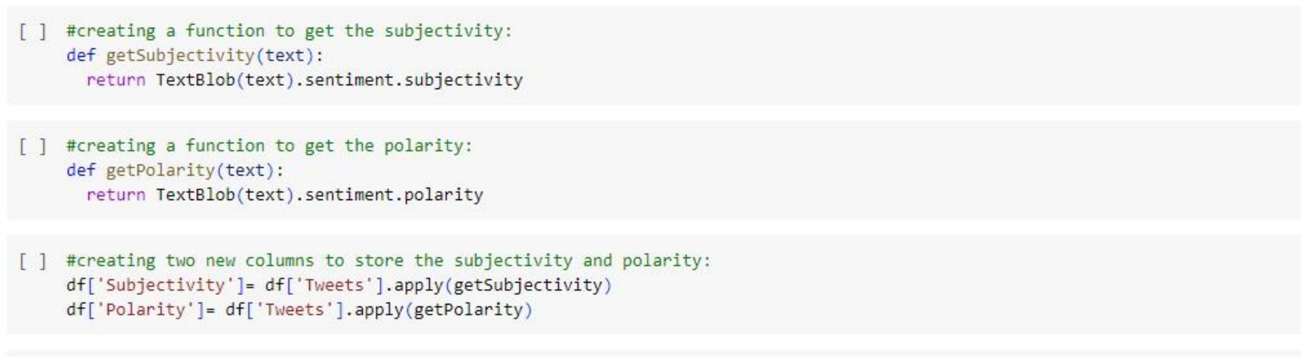
By comprehensively cleaning and pre-processing the data, we ensure that the text is transformed into a structured and normalized format, suitable for accurate sentiment analysis. These pre-processing steps enhance the quality of subsequent analysis and sentiment classification.



```
# Display the modified DataFrame
df.head()
```

	id	user	sentiment	tweet	Tweets
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...	getting borderland murder
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	coming border kill
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	getting borderland kill
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	coming borderland murder
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	getting borderland murder

Fig 6 : Displaying the modified dataframe

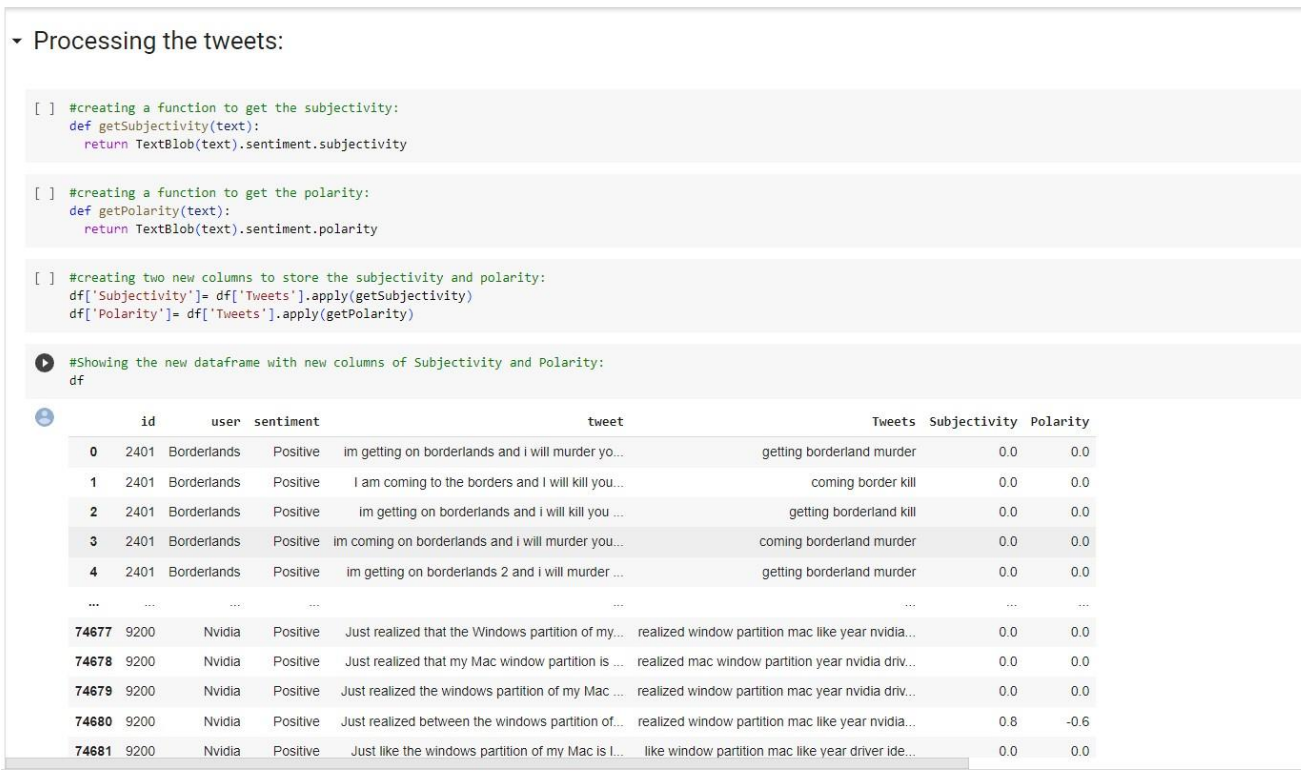


```
#creating a function to get the subjectivity:
def getSubjectivity(text):
  return TextBlob(text).sentiment.subjectivity
```

```
#creating a function to get the polarity:
def getPolarity(text):
  return TextBlob(text).sentiment.polarity
```

```
#creating two new columns to store the subjectivity and polarity:
df['Subjectivity']= df['Tweets'].apply(getSubjectivity)
df['Polarity']= df['Tweets'].apply(getPolarity)
```

Fig 7 : Function of Polarity and Subjectivity



Processing the tweets:

```
#creating a function to get the subjectivity:
def getSubjectivity(text):
  return TextBlob(text).sentiment.subjectivity
```

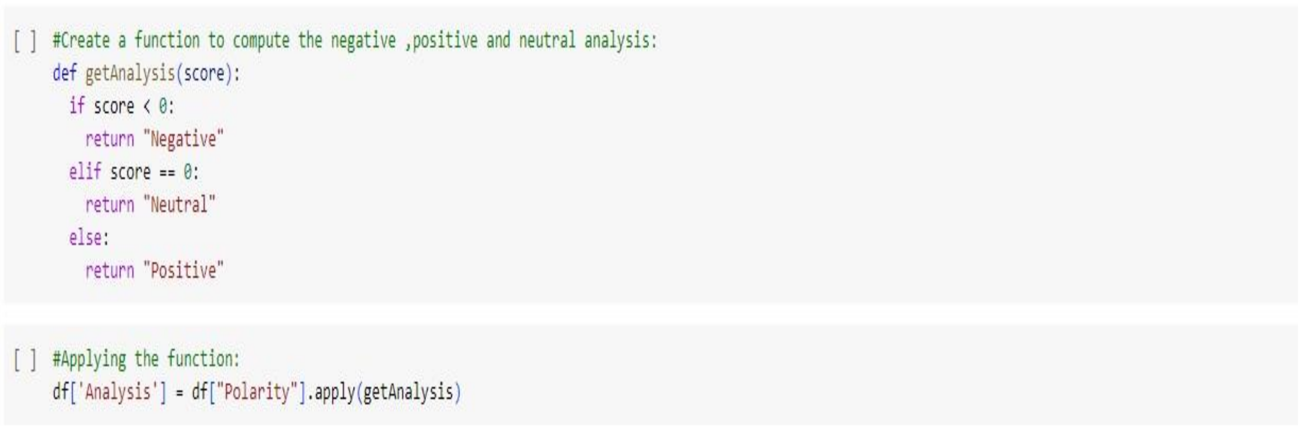
```
#creating a function to get the polarity:
def getPolarity(text):
  return TextBlob(text).sentiment.polarity
```

```
#creating two new columns to store the subjectivity and polarity:
df['Subjectivity']= df['Tweets'].apply(getSubjectivity)
df['Polarity']= df['Tweets'].apply(getPolarity)
```

```
#Showing the new dataframe with new columns of Subjectivity and Polarity:
df
```

	id	user	sentiment	tweet	Tweets	Subjectivity	Polarity
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...	getting borderland murder	0.0	0.0
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	coming border kill	0.0	0.0
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	getting borderland kill	0.0	0.0
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	coming borderland murder	0.0	0.0
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	getting borderland murder	0.0	0.0
...	...	...	...	...	...	...	...
74677	9200	Nvidia	Positive	Just realized that the Windows partition of my...	realized window partition mac like year nvidia...	0.0	0.0
74678	9200	Nvidia	Positive	Just realized that my Mac window partition is ...	realized mac window partition year nvidia driv...	0.0	0.0
74679	9200	Nvidia	Positive	Just realized the windows partition of my Mac ...	realized window partition mac year nvidia driv...	0.0	0.0
74680	9200	Nvidia	Positive	Just realized between the windows partition of...	realized window partition mac like year nvidia...	0.8	-0.6
74681	9200	Nvidia	Positive	Just like the windows partition of my Mac is l...	like window partition mac like year driver ide...	0.0	0.0

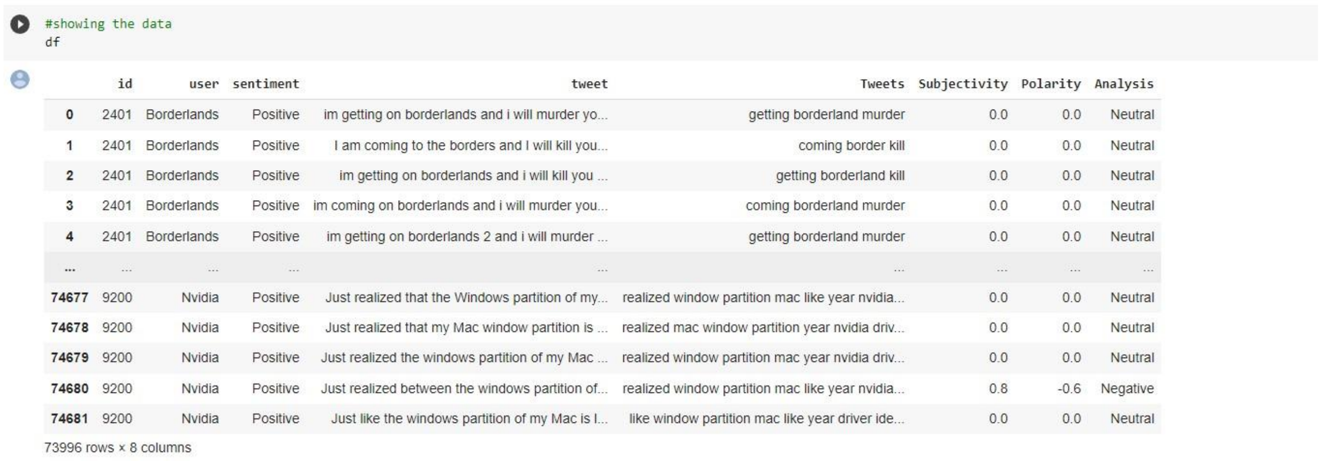
Fig 8 : Figure of new dataframe



```
[ ] #Create a function to compute the negative ,positive and neutral analysis:
    def getAnalysis(score):
      if score < 0:
        return "Negative"
      elif score == 0:
        return "Neutral"
      else:
        return "Positive"
```

```
[ ] #Applying the function:
    df['Analysis'] = df["Polarity"].apply(getAnalysis)
```

Fig 9 : Figure of function to calculate the positive ,negative and neutral tweet



```
#showing the data
df
```

	id	user	sentiment	tweet	Tweets	Subjectivity	Polarity	Analysis
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...	getting borderland murder	0.0	0.0	Neutral
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	coming border kill	0.0	0.0	Neutral
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	getting borderland kill	0.0	0.0	Neutral
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	coming borderland murder	0.0	0.0	Neutral
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	getting borderland murder	0.0	0.0	Neutral
...	...	...	...	...	...	...	...	...
74677	9200	Nvidia	Positive	Just realized that the Windows partition of my...	realized window partition mac like year nvidia...	0.0	0.0	Neutral
74678	9200	Nvidia	Positive	Just realized that my Mac window partition is ...	realized mac window partition year nvidia driv...	0.0	0.0	Neutral
74679	9200	Nvidia	Positive	Just realized the windows partition of my Mac ...	realized window partition mac year nvidia driv...	0.0	0.0	Neutral
74680	9200	Nvidia	Positive	Just realized between the windows partition of...	realized window partition mac like year nvidia...	0.8	-0.6	Negative
74681	9200	Nvidia	Positive	Just like the windows partition of my Mac is l...	like window partition mac like year driver ide...	0.0	0.0	Neutral

73996 rows × 8 columns

Chapter 4: Methodology

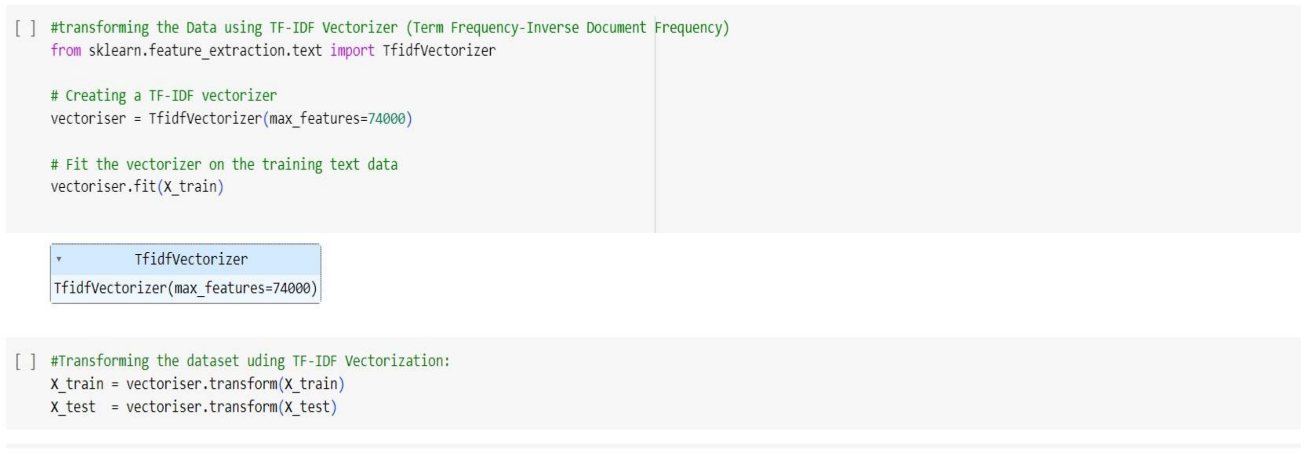
4.1 Feature Extraction

➢ Explanation of Text Embedding

Text embeddings are like numerical representations of words or phrases that capture their meanings and relationships in a way that machine learning models can understand. Imagine each word as a unique point in a multi-dimensional space. These embeddings place words in that space, preserving their context and semantic similarities.

For example, words with similar meanings or those often used together will be closer in this space. "Cat" and "dog" might be closer to each other than "cat" and "computer."

Creation of the Feature Matrix for Machine Learning Models



```
[ ] #transforming the Data using TF-IDF Vectorizer (Term Frequency-Inverse Document Frequency)
    from sklearn.feature_extraction.text import TfidfVectorizer

    # Creating a TF-IDF vectorizer
    vectoriser = TfidfVectorizer(max_features=74000)

    # Fit the vectorizer on the training text data
    vectoriser.fit(X_train)
```

```
TfidfVectorizer
TfidfVectorizer(max_features=74000)
```

```
[ ] #Transforming the dataset uding TF-IDF Vectorization:
    X_train = vectoriser.transform(X_train)
    X_test  = vectoriser.transform(X_test)
```

Fig 10 : Figure of Feature Selection using TDF-IF Vectorizer

To use text embedding’s in machine learning, we create a feature matrix. Think of this matrix like a table where each row represents a tweet, and each column corresponds to different aspects of the tweet's meaning captured by embedding’s.

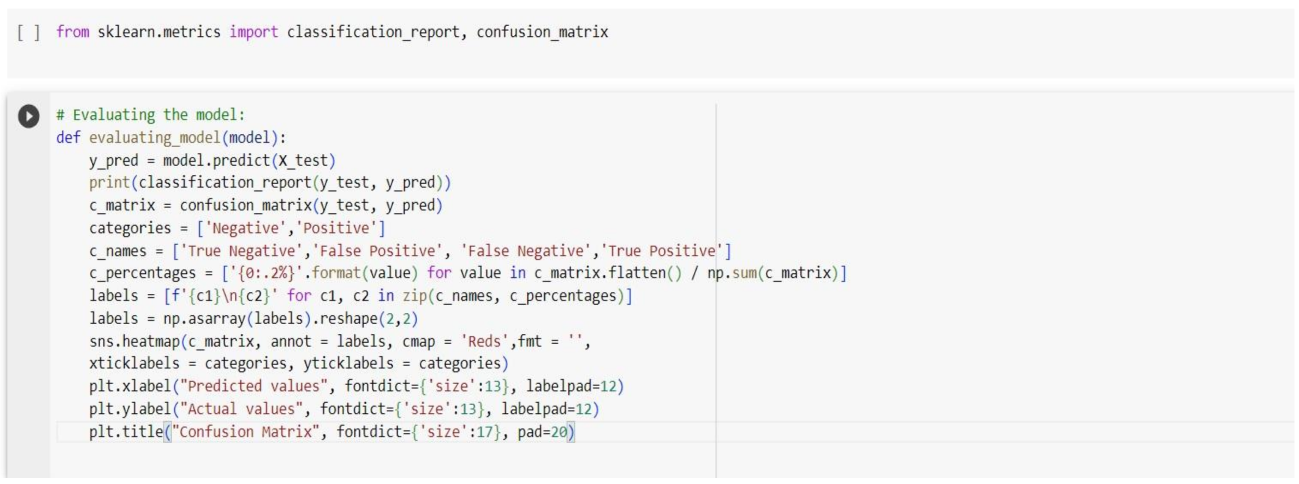
Let's say we have tweets: "I love sunny days" and "Rainy days make me sad." We convert these into embedding’s and stack them row by row, creating a matrix. Each row contains the information about a tweet in a format that machine learning models can use

This matrix becomes our input. Machine learning models look at patterns in this matrix to understand how different embeddings relate to sentiments (positive, negative, or neutral). With this understanding, the models can later predict sentiments for new, unseen tweets.

In this chapter, we'll dive into how we get these embeddings and create the feature matrix. This step is crucial as it sets the stage for the models to learn and make sense of the sentiments expressed in tweets.

4.2 Machine Learning Models

Detailed Explanation of Classification Algorithms



```
[ ] from sklearn.metrics import classification_report, confusion_matrix
```

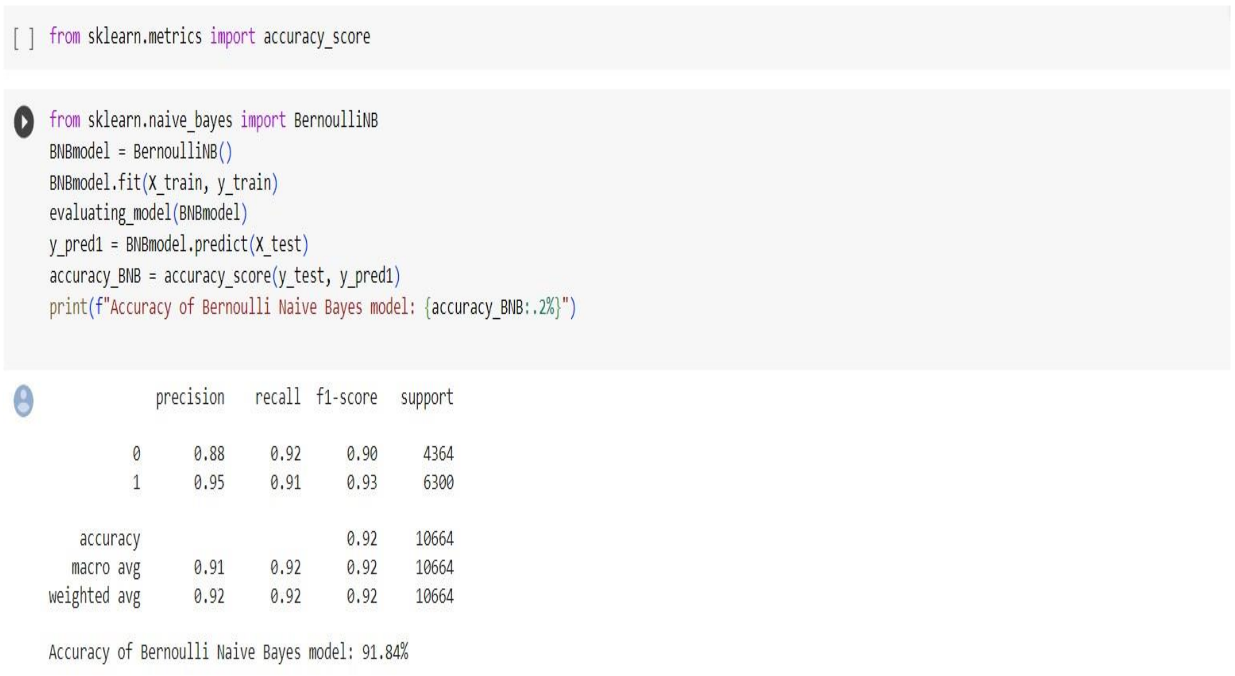
```
# Evaluating the model:
def evaluating_model(model):
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    c_matrix = confusion_matrix(y_test, y_pred)
    categories = ['Negative','Positive']
    c_names = ['True Negative','False Positive', 'False Negative','True Positive']
    c_percentages = ['{0:.2%}'.format(value) for value in c_matrix.flatten() / np.sum(c_matrix)]
    labels = [f'{c1}\n{c2}' for c1, c2 in zip(c_names, c_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(c_matrix, annot = labels, cmap = 'Reds',fmt = '',
    xticklabels = categories, yticklabels = categories)
    plt.xlabel("Predicted values", fontdict={'size':13}, labelpad=12)
    plt.ylabel("Actual values", fontdict={'size':13}, labelpad=12)
    plt.title("Confusion Matrix", fontdict={'size':17}, pad=20)
```

Fig 11 : Figure of evaluating model function

- ✓ Naive Bayes:

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It's considered "naive" because it assumes that the features (in our case, the text embedding's) are independent of each other. Despite this simplification, Naive Bayes often performs well for text classification tasks.

The algorithm calculates the probability of a tweet belonging to a specific sentiment class (positive, negative, or neutral) based on the probabilities of the individual features (embedding's). It then assigns the class with the highest probability.



```
[ ] from sklearn.metrics import accuracy_score
```

```
from sklearn.naive_bayes import BernoulliNB
BNBmodel = BernoulliNB()
BNBmodel.fit(X_train, y_train)
evaluating_model(BNBmodel)
y_pred1 = BNBmodel.predict(X_test)
accuracy_BNB = accuracy_score(y_test, y_pred1)
print(f"Accuracy of Bernoulli Naive Bayes model: {accuracy_BNB:.2%}")
```

```
              precision    recall  f1-score   support

           0       0.88      0.92      0.90      4364
           1       0.95      0.91      0.93      6300

    accuracy                           0.92     10664
   macro avg       0.91      0.92      0.92     10664
weighted avg       0.92      0.92      0.92     10664

Accuracy of Bernoulli Naive Bayes model: 91.84%
```

Fig 12: Figure of applying Bernouli's Naive Bayes Classification Trial

```
Accuracy of Bernoulli Naive Bayes model: 91.84%
```

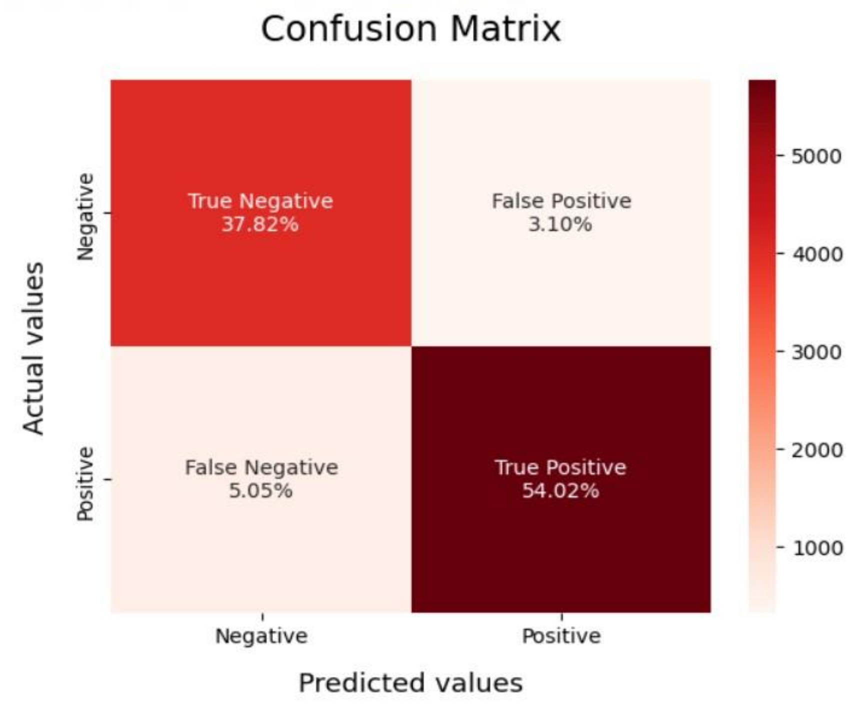
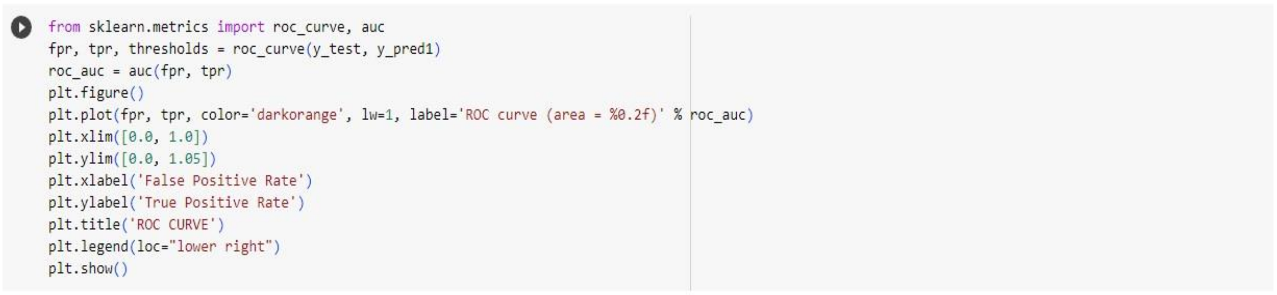


Fig 13: Figure of Confusion Matrix of Bernoulli's Trial



```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred1)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

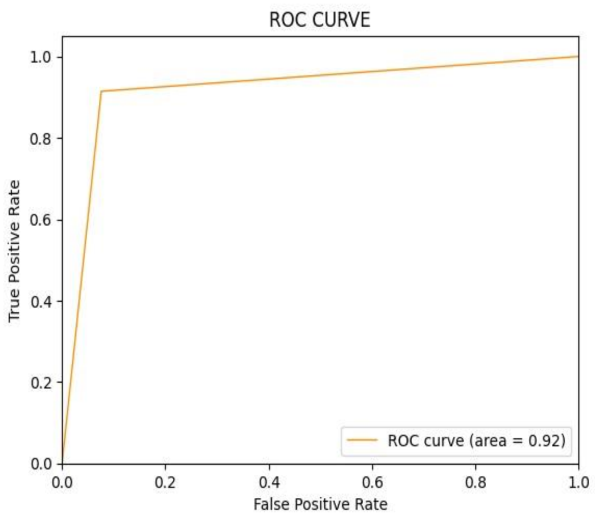
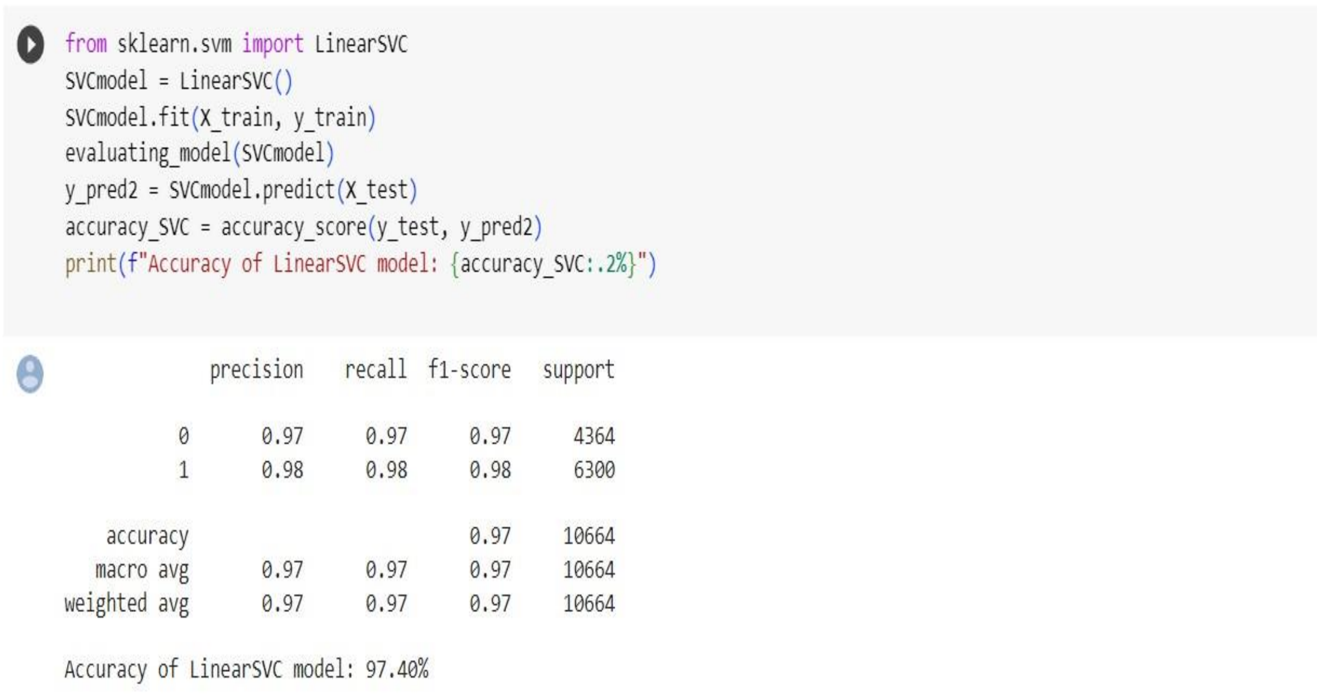


Fig 14: Figure of ROC Curve of Bernoulli's Trial

✓ Support Vector Machines (SVM)

SVM is a powerful algorithm for binary and multi-class classification. In the context of sentiment analysis, SVM aims to find a hyperplane that best separates tweets belonging to different sentiment classes. It seeks the maximum margin between classes, making it robust to noise and suitable for high-dimensional data like text embedding.

SVM transforms the feature matrix into a higher-dimensional space where classes are more easily separable. It then finds the hyperplane that maximizes the margin between these classes while minimizing misclassifications.



```
from sklearn.svm import LinearSVC
SVCmodel = LinearSVC()
SVCmodel.fit(X_train, y_train)
evaluating_model(SVCmodel)
y_pred2 = SVCmodel.predict(X_test)
accuracy_SVC = accuracy_score(y_test, y_pred2)
print(f"Accuracy of LinearSVC model: {accuracy_SVC:.2%}")
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      4364
           1       0.98      0.98      0.98      6300

    accuracy                           0.97     10664
   macro avg       0.97      0.97      0.97     10664
weighted avg       0.97      0.97      0.97     10664

Accuracy of LinearSVC model: 97.40%
```

Fig 15: Figure of applying Simple Vector Machine (SVM)

```
Accuracy of LinearSVC model: 97.40%
```

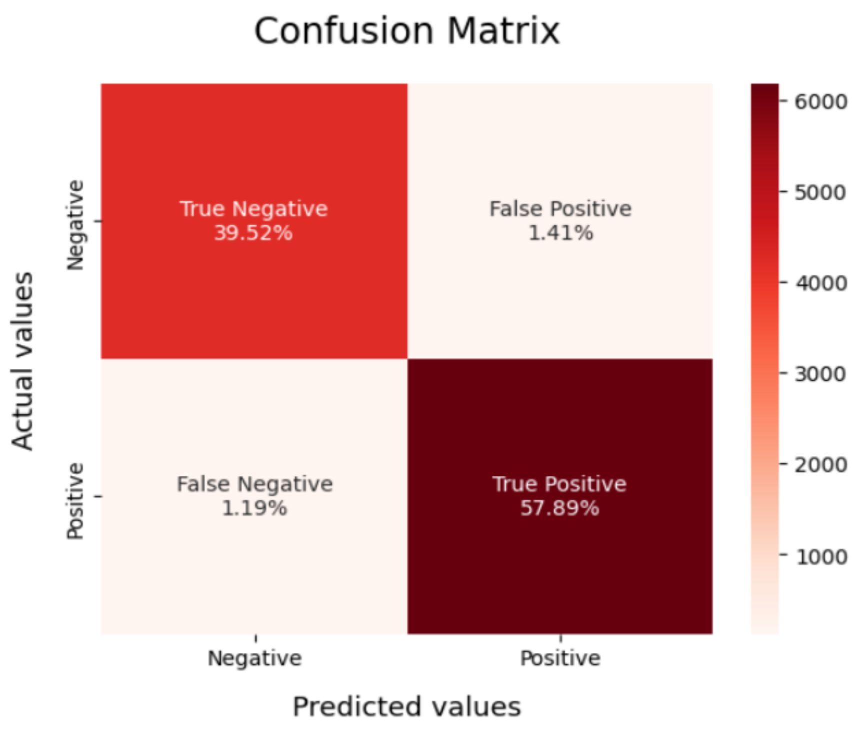
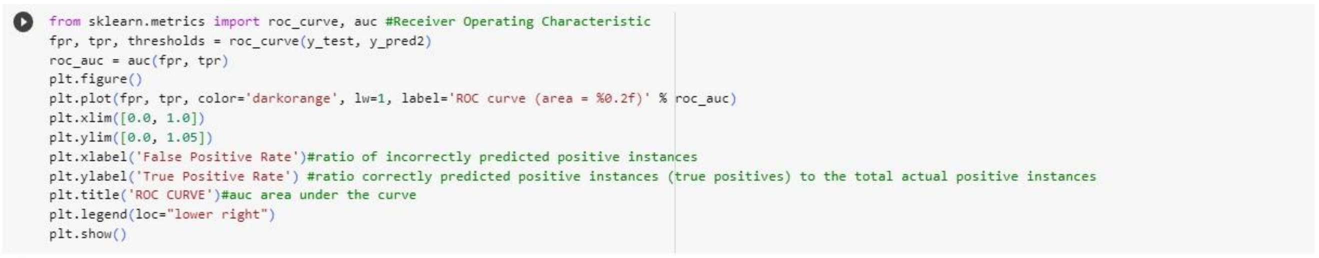


Fig 16: Figure of Confusion Matrix of SVM



```
from sklearn.metrics import roc_curve, auc #Receiver Operating Characteristic
fpr, tpr, thresholds = roc_curve(y_test, y_pred2)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')#ratio of incorrectly predicted positive instances
plt.ylabel('True Positive Rate') #ratio correctly predicted positive instances (true positives) to the total actual positive instances
plt.title('ROC CURVE')#auc area under the curve
plt.legend(loc="lower right")
plt.show()
```

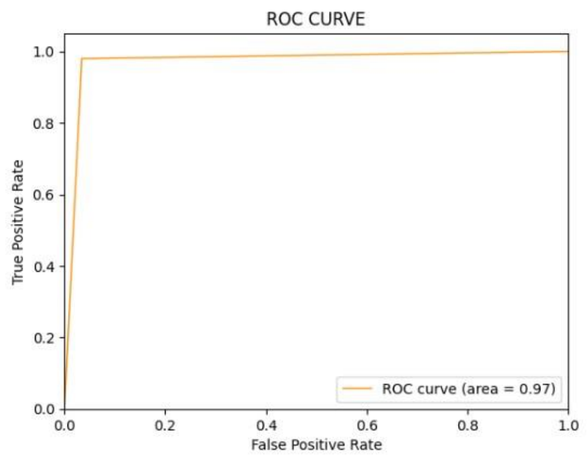
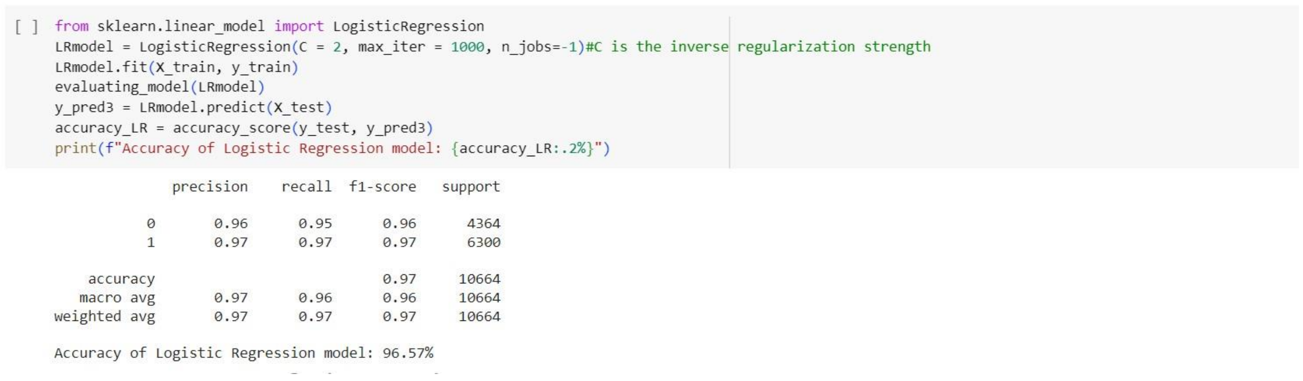


Fig 17: Figure of ROC Curve of SVM

➢ Logistic Regression:

Despite its name, logistic regression is used for binary classification tasks like sentiment analysis. It models the probability of a tweet belonging to a specific sentiment class using a logistic function. This probability is then compared to a threshold to make the final classification decision.

Logistic regression estimates the coefficients for each feature (embedding) to maximize the likelihood of observing the given sentiment labels in the training data. These coefficients are used to predict the probabilities and ultimately the classes for new, unseen data.



```
[ ] from sklearn.linear_model import LogisticRegression
    LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)#C is the inverse regularization strength
    LRmodel.fit(X_train, y_train)
    evaluating_model(LRmodel)
    y_pred3 = LRmodel.predict(X_test)
    accuracy_LR = accuracy_score(y_test, y_pred3)
    print(f"Accuracy of Logistic Regression model: {accuracy_LR:.2%}")
```

```
              precision    recall  f1-score   support

           0       0.96      0.95      0.96      4364
           1       0.97      0.97      0.97      6300

    accuracy                           0.97     10664
   macro avg       0.97      0.96      0.96     10664
weighted avg       0.97      0.97      0.97     10664

Accuracy of Logistic Regression model: 96.57%
```

Fig 18: Figure of applying Logistic Regression

Accuracy of Logistic Regression model: 96.57%

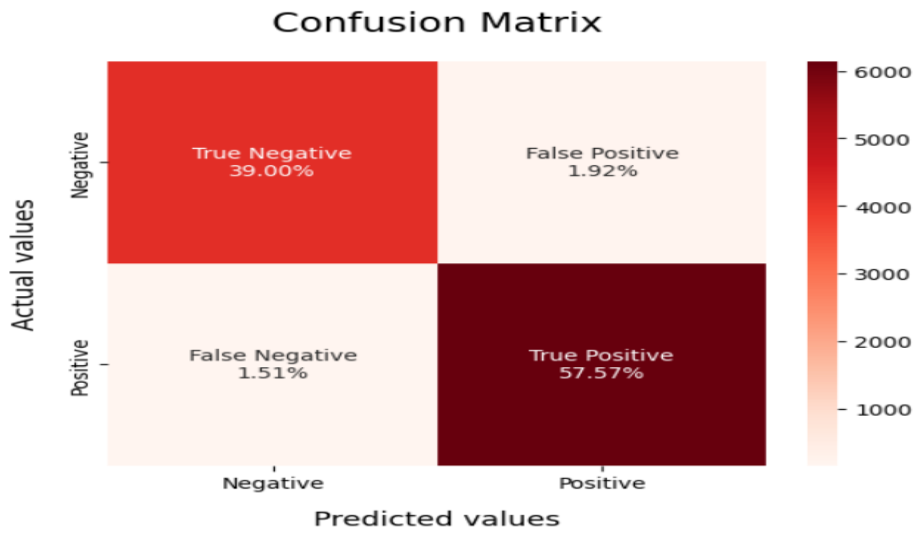
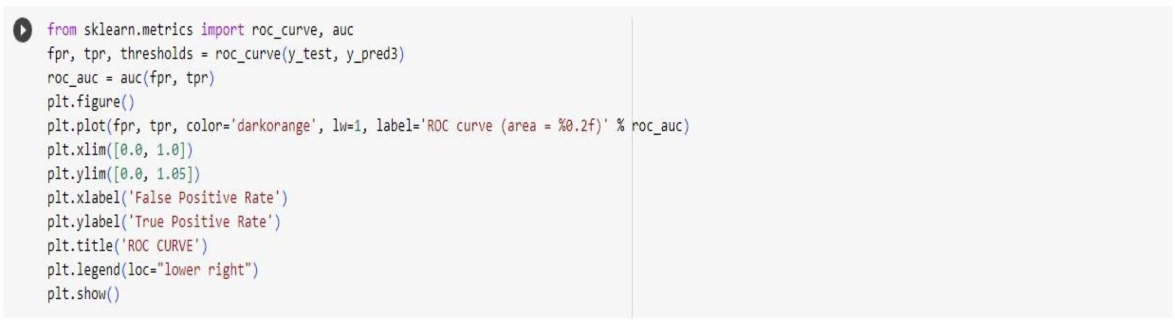


Fig 19: Figure of Confusion Matrix of Logistic Regression



```python
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred3)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

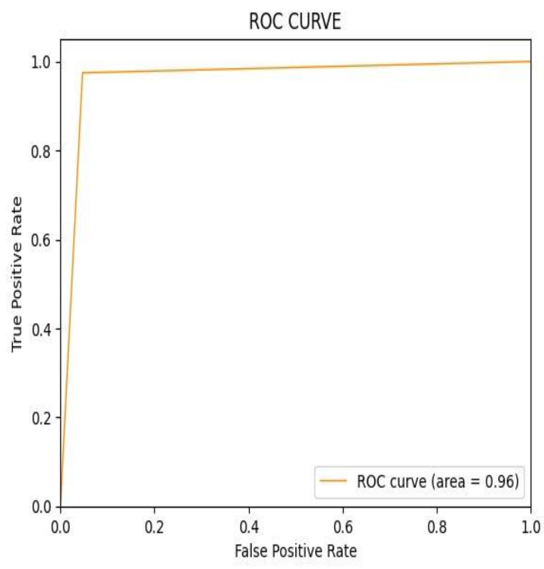
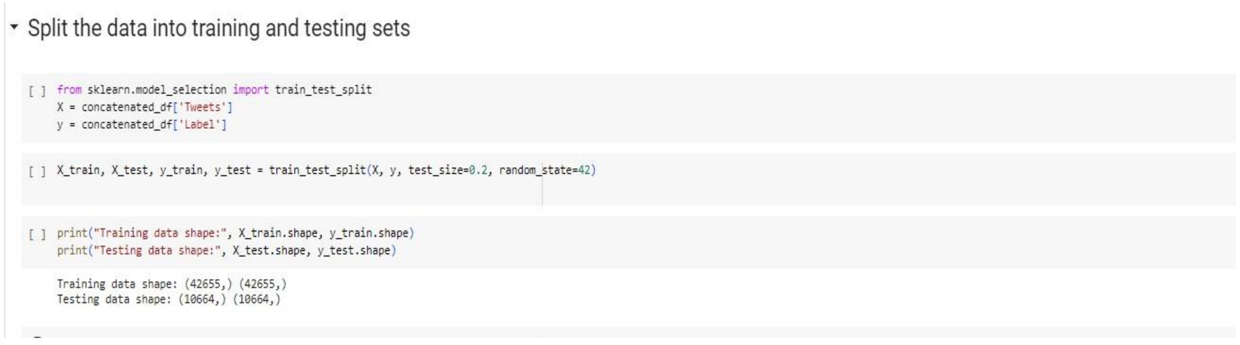


Fig 20: Figure of ROC Curve of Logistic Regression

➢ Training and Testing Process:
- For each algorithm:

1. Training: We use a labeled dataset to teach the algorithm to recognize patterns between text embedding and sentiments. The algorithm adjusts its internal parameters to minimize classification errors.

2. Testing: After training, we evaluate the algorithm's performance on a separate dataset it has never seen before. This dataset contains tweets with known sentiments. The algorithm predicts sentiments for these tweets, and we compare its predictions with the actual sentiments to assess accuracy.



Split the data into training and testing sets

```python
[ ] from sklearn.model_selection import train_test_split
    X = concatenated_df['Tweets']
    y = concatenated_df['Label']

[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[ ] print("Training data shape:", X_train.shape, y_train.shape)
    print("Testing data shape:", X_test.shape, y_test.shape)

    Training data shape: (42655,) (42655,)
    Testing data shape: (10664,) (10664,)
```

Fig 21 Figure of Train Test Split of Dataset

Evaluation Metrics:

- Accuracy: Ratio of correctly predicted sentiments to the total number of sentiments.

- Precision: Proportion of true positive predictions out of all positive predictions.

- Recall: Proportion of true positive predictions out of all actual positive sentiments.

- F1-Score: Harmonic mean of precision and recall.

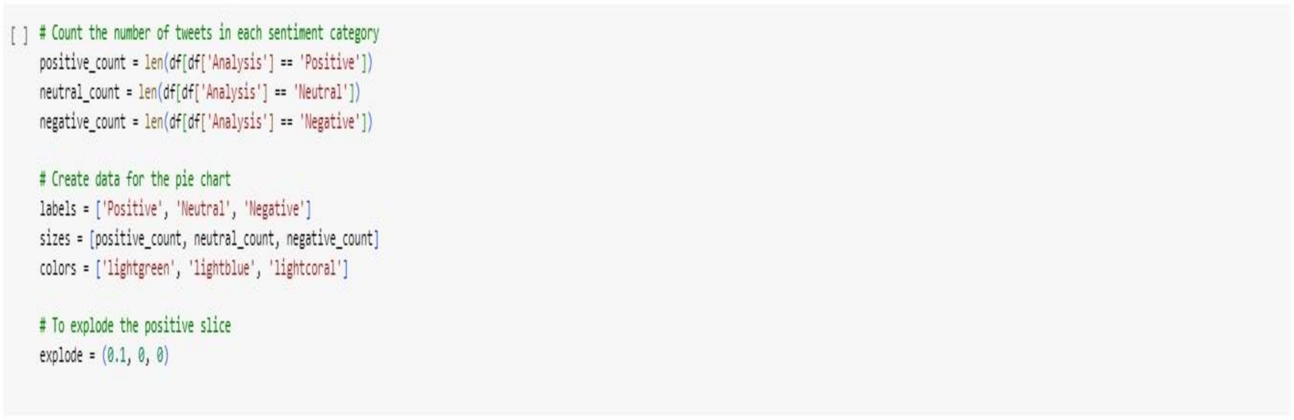
Each algorithm has its strengths and weaknesses. Naive Bayes is simple and computationally efficient, SVM is robust in high-dimensional spaces, and logistic regression provides interpretable results. The choice of algorithm depends on the nature of the data and the desired trade-offs between accuracy and complexity.

Chapter 5: Data Visualization

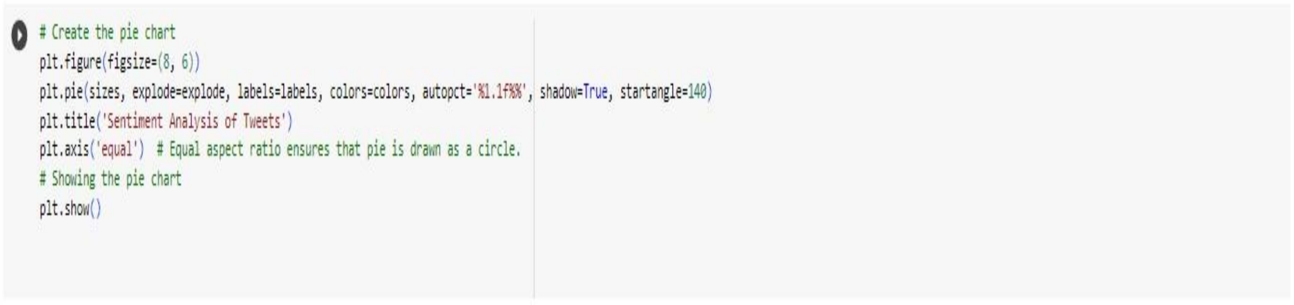
- 5.1 Exploratory Data Analysis (EDA)
- Visual Representation of the Distribution of Sentiments in the Dataset

Exploratory Data Analysis (EDA) is a crucial step to understand the structure and characteristics of the dataset. Visualizing the distribution of sentiments helps us gain insights into the overall sentiment landscape.

- Pie Chart: A pie chart provides a clear visual representation of the proportion of each sentiment category in the dataset. It enables an immediate understanding of the sentiment distribution, showing whether sentiments are balanced or skewed.



```
[ ] # Count the number of tweets in each sentiment category
    positive_count = len(df[df['Analysis'] == 'Positive'])
    neutral_count = len(df[df['Analysis'] == 'Neutral'])
    negative_count = len(df[df['Analysis'] == 'Negative'])

    # Create data for the pie chart
    labels = ['Positive', 'Neutral', 'Negative']
    sizes = [positive_count, neutral_count, negative_count]
    colors = ['lightgreen', 'lightblue', 'lightcoral']

    # To explode the positive slice
    explode = (0.1, 0, 0)
```



```
# Create the pie chart
plt.figure(figsize=(8, 6))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Sentiment Analysis of Tweets')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
# Showing the pie chart
plt.show()
```

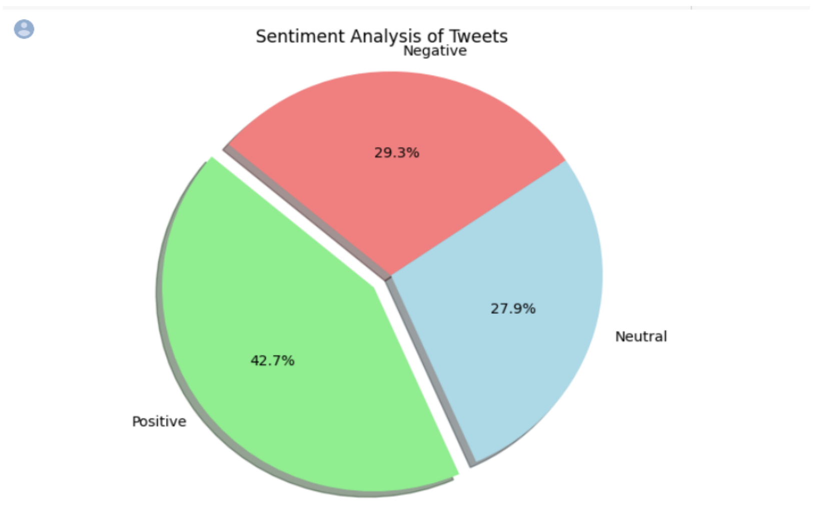


Fig 22 Figure of Pie Chart of *Positive, Negative and Neutral Sentiments*

- Bar Plot: A bar plot offers a more detailed view, displaying the absolute count of tweets for each sentiment category. This helps us gauge the volume of each sentiment and identify any potential imbalances.

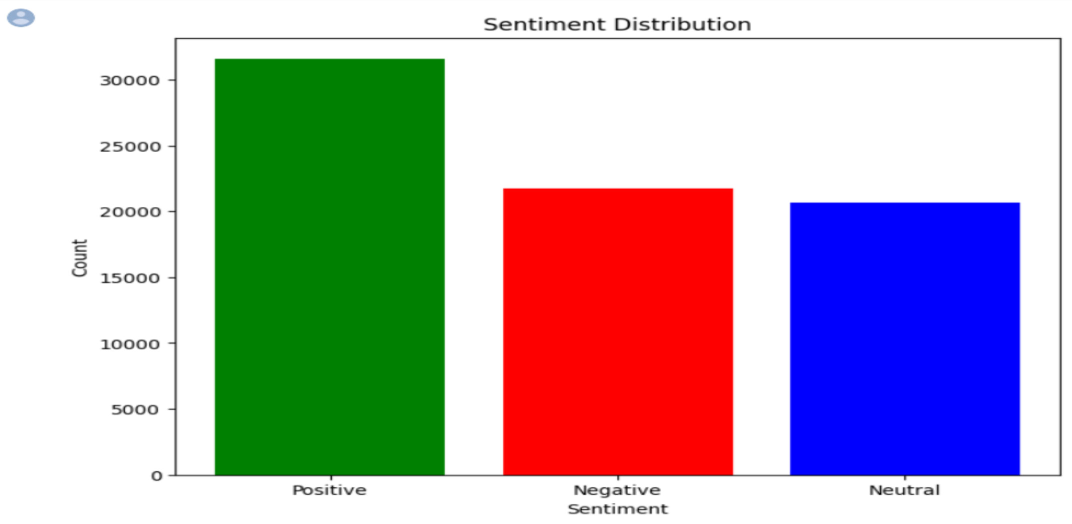
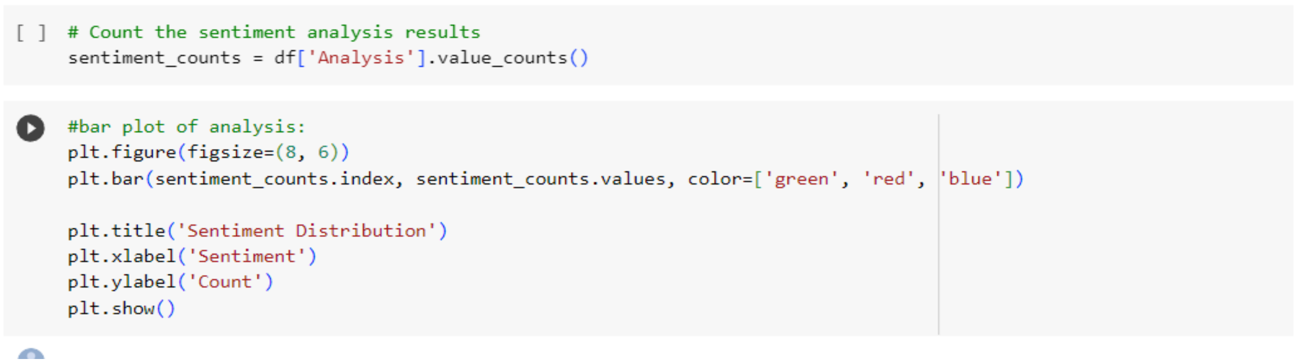


Fig 23 Figure of Bar Chart of *Positive, Negative and Neutral Sentiments*



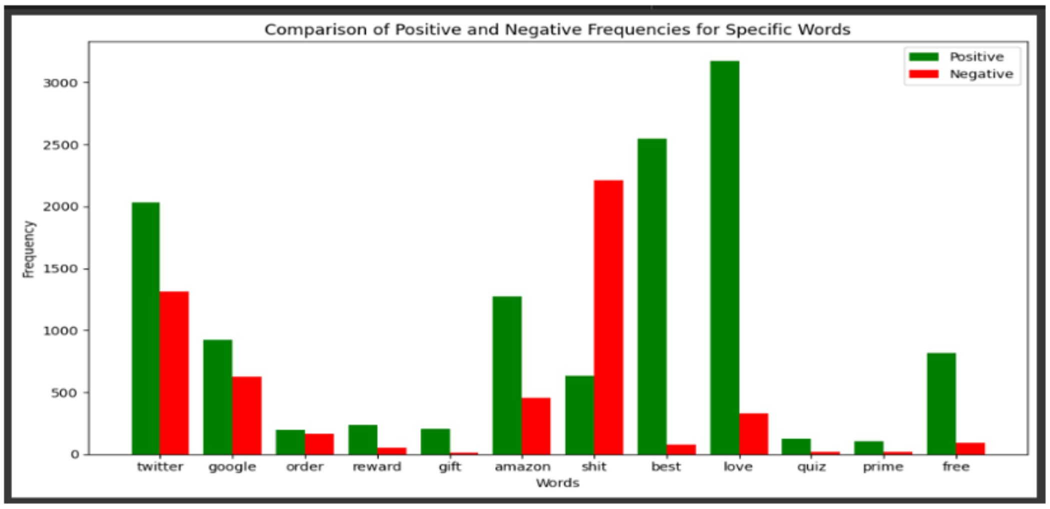
```
[ ] # Count the sentiment analysis results
    sentiment_counts = df['Analysis'].value_counts()
```

```
#bar plot of analysis:
plt.figure(figsize=(8, 6))
plt.bar(sentiment_counts.index, sentiment_counts.values, color=['green', 'red', 'blue'])

plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```

Analysis of the Most Frequent Words in Different Sentiment Categories

Analyzing the most frequent words associated with different sentiment categories enriches our understanding of sentiment expression.

Most Frequent Words Analysis



Category Analysis

Analyzing the most frequent words in a specific category (e.g. Amazon).

Fig 24 Figure of Bar Chart of Most Frequent Words of Amazon into Positive and Negative Sentiments

- Word Clouds: Word clouds offer an intuitive and engaging representation of word frequency. Words that appear frequently are depicted larger, emphasizing their significance. Creating word clouds for each sentiment category reveals the prominent words used in different sentiments.

```
#Plot the Word Cloud
all_words = " ".join([sentence for sentence in df['Tweets']])

wordCloud = WordCloud(width=500,height=300, random_state= 21, max_font_size=119).generate(all_words)
plt.imshow(wordCloud, interpolation= "bilinear")
plt.axis("off")
plt.show()
```

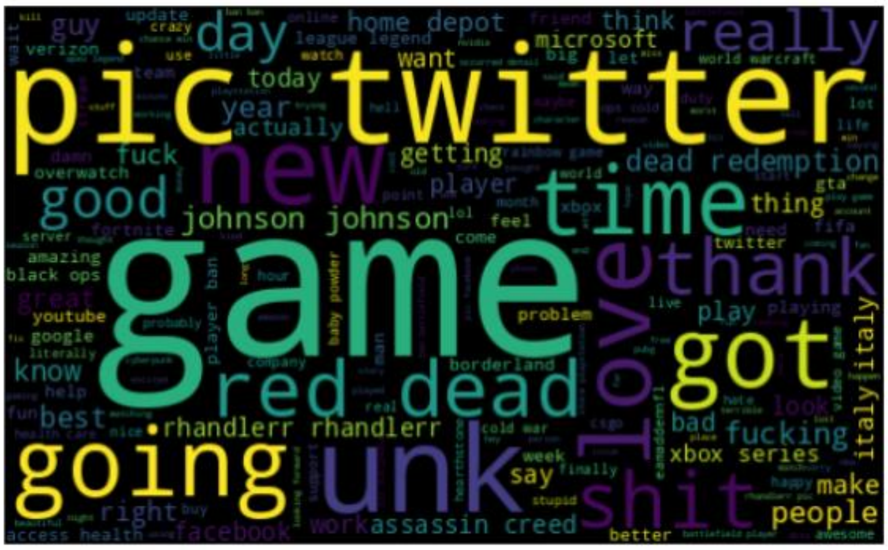


Fig 25 Figure of Word Cloud of Most Frequent Words in the Dataset

```
# frequent words visualization for +ve
all_words = " ".join([sentence for sentence in df['Tweets'][df['Analysis']== "Positive"]])

wordcloud = WordCloud(width= 500, height=300, random_state=21, max_font_size=119).generate(all_words)

# plot the graph

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

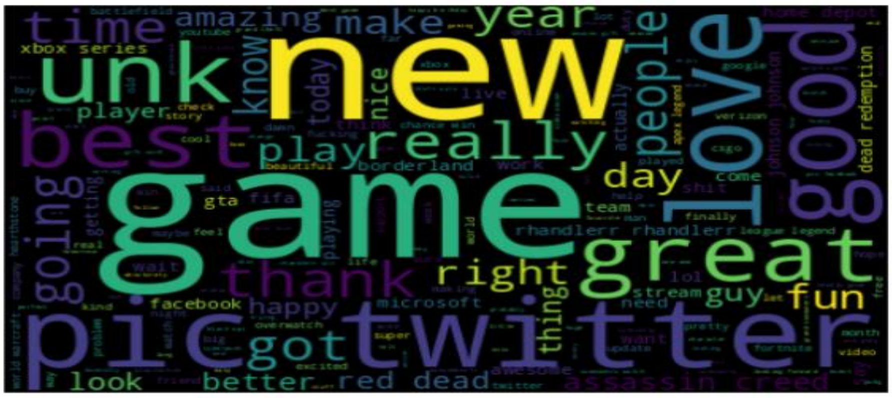
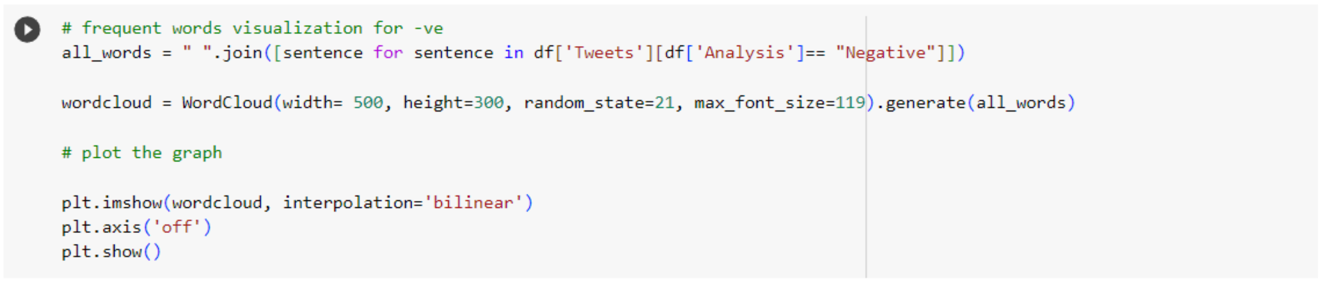


Fig 26 Figure of Word Cloud of Most Frequent Positive Words in the Dataset



```
# frequent words visualization for -ve
all_words = " ".join([sentence for sentence in df['Tweets'][df['Analysis']== "Negative"]])

wordcloud = WordCloud(width= 500, height=300, random_state=21, max_font_size=119).generate(all_words)

# plot the graph

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

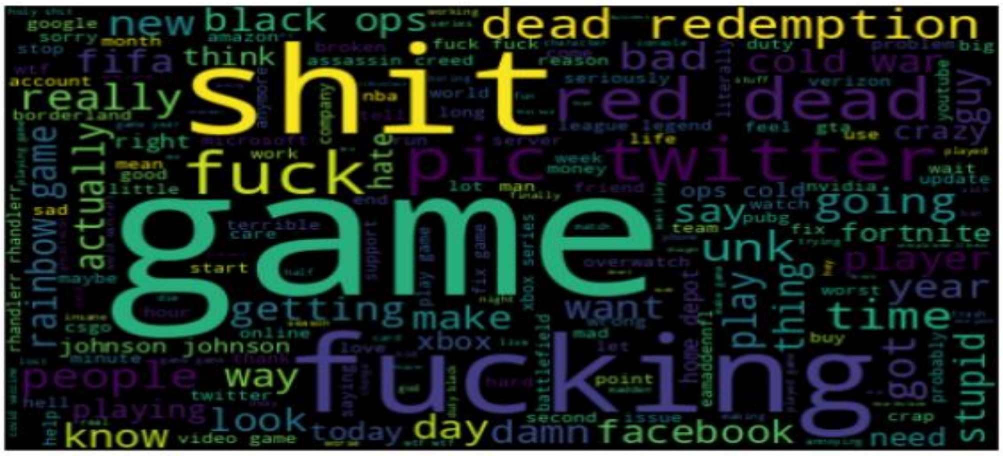


Fig 27 Figure of Word Cloud of Most Frequent Negative Words in the Dataset

- -Bar Charts: To delve deeper, bar charts can be created to showcase the top N most frequent words in each sentiment category. This provides a more quantitative view of the language associated with various sentiments.

Detailed Explanation for Each Visualization:

1. Visualizing Sentiment Distribution:

- Pie Chart: If, for instance, the pie chart indicates 42.7% positive, 27.9% neutral, and 29.3% negative sentiments, it's evident that positive sentiments dominate.

- Bar Plot: If the bar plot shows 32000 positive, 20000 neutral, and 23000 negative tweets, we get a clarer sense of sentiment distribution.

By conducting thorough exploratory data analysis and creating these visualizations, we uncover valuable insights about sentiment distribution and language patterns. These insights can guide us in fine-tuning our sentiment analysis model and extracting meaningful insights from the Twitter data.

Appendices

- Appendix D: Tableau Visualizations

We extended the depth of our analysis by utilizing Tableau to create dynamic and informative visualizations that enhance our understanding of sentiment trends and insights extracted from the Twitter data. The following are examples of the Tableau visualizations we've incorporated:

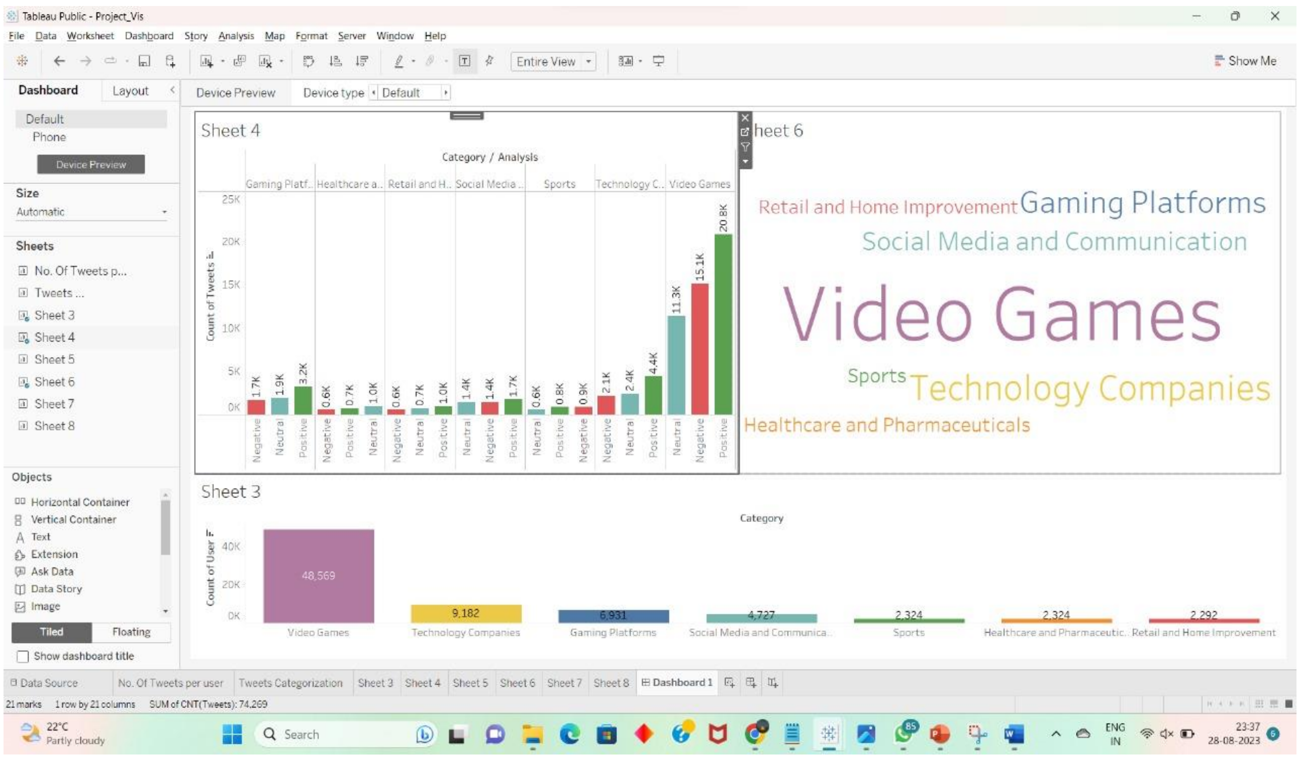
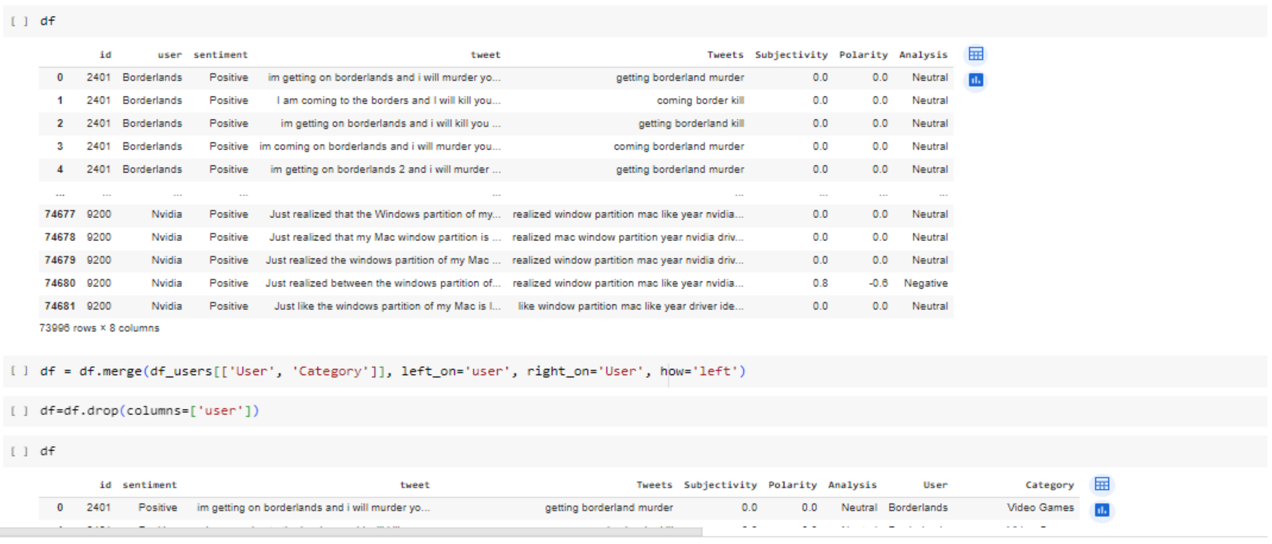


Tableau Public - Project_Vis
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Dashboard Layout
Device Preview Device type Default
Default
Phone
Size
Automatic
Sheets
No. Of Tweets p...
Tweets ...
Sheet 3
Sheet 4
Sheet 5
Sheet 6
Sheet 7
Sheet 8
Objects
Horizontal Container
Vertical Container
Text
Extension
Ask Data
Data Story
Image
Tiled Floating
Show dashboard title
Sheet 4
Category / Analysis
Gaming Platf.. Healthcare a.. Retail and H.. Social Media .. Sports Technology C.. Video Games
Count of Tweets
25K 20K 15K 10K 5K 0K
1.7K 1.9K 3.2K 0.6K 0.7K 1.0K 0.6K 0.7K 1.0K 1.4K 1.4K 1.7K 0.6K 0.8K 0.9K 2.1K 2.4K 4.4K 11.3K 15.1K 20.8K
Negative Neutral Positive
heet 6
Retail and Home Improvement
Gaming Platforms
Social Media and Communication
Video Games
Sports
Technology Companies
Healthcare and Pharmaceuticals
Sheet 3
Category
Count of User
40K 20K 0K
48,569 9,182 6,931 4,727 2,324 2,324 2,292
Video Games Technology Companies Gaming Platforms Social Media and Communica.. Sports Healthcare and Pharmaceutic.. Retail and Home Improvement
Data Source No. Of Tweets per user Tweets Categorization Sheet 3 Sheet 4 Sheet 5 Sheet 6 Sheet 7 Sheet 8 Dashboard 1
21 marks 1 row by 21 columns SUM of CNT(Tweets): 74,269
22°C Partly cloudy
Search
ENG IN
23:37 28-08-2023

Fig 28 Figure of Dashboard Visualization in Tableau

By mapping sentiment distribution to popular hashtags, this visualization offers insights into how sentiments are associated with trending topics. The interactive nature of Tableau allows users to filter sentiments based on specific hashtags, offering a nuanced view of sentiment dynamics.



```
[ ] df
```

	id	user	sentiment	tweet	Tweets	Subjectivity	Polarity	Analysis
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...	getting borderland murder	0.0	0.0	Neutral
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	coming border kill	0.0	0.0	Neutral
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	getting borderland kill	0.0	0.0	Neutral
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	coming borderland murder	0.0	0.0	Neutral
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	getting borderland murder	0.0	0.0	Neutral
...	...	...	...	...	...	...	...	...
74677	9200	Nvidia	Positive	Just realized that the Windows partition of my...	realized window partition mac like year nvidia...	0.0	0.0	Neutral
74678	9200	Nvidia	Positive	Just realized that my Mac window partition is ...	realized mac window partition year nvidia driv...	0.0	0.0	Neutral
74679	9200	Nvidia	Positive	Just realized the windows partition of my Mac ...	realized window partition mac year nvidia driv...	0.0	0.0	Neutral
74680	9200	Nvidia	Positive	Just realized between the windows partition of...	realized window partition mac like year nvidia...	0.8	-0.6	Negative
74681	9200	Nvidia	Positive	Just like the windows partition of my Mac is l...	like window partition mac like year driver ide...	0.0	0.0	Neutral

73996 rows × 8 columns

```
[ ] df = df.merge(df_users[['User', 'Category']], left_on='user', right_on='User', how='left')
```

```
[ ] df=df.drop(columns=['user'])
```

```
[ ] df
```

	id	sentiment	tweet	Tweets	Subjectivity	Polarity	Analysis	User	Category
0	2401	Positive	im getting on borderlands and i will murder yo...	getting borderland murder	0.0	0.0	Neutral	Borderlands	Video Games

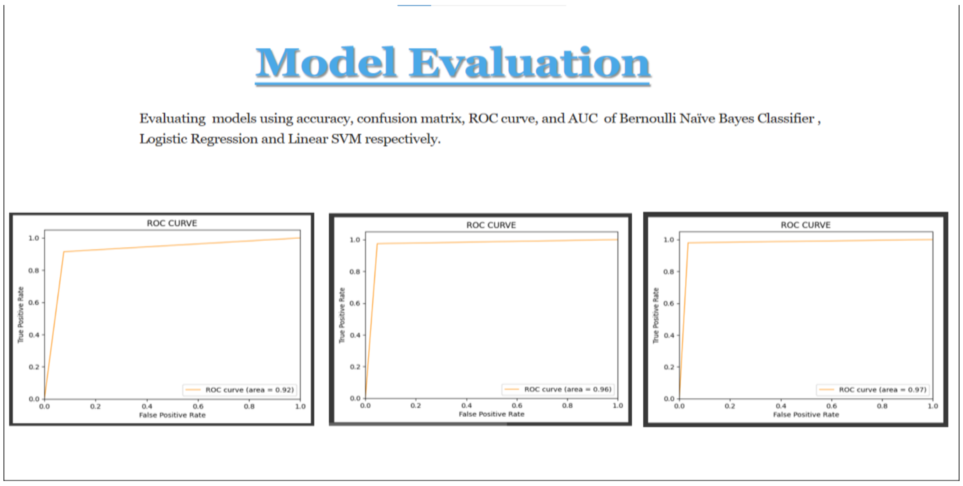
These visualizations present word clouds for various sentiment categories, showcasing the most frequent words associated with each sentiment. Interactivity allows users to explore words in more detail, providing a closer look at the language patterns linked to different emotions.

Chapter 6: Results and Discussion

6.1 Model Performance

Presentation of Accuracy and Other Relevant Metrics of the Trained Models

- In this chapter, we present the outcomes of our sentiment analysis models. We showcase how well each model has performed in categorizing tweets into positive, negative, or neutral sentiments. Accuracy is a key metric, but we also consider other metrics to provide a comprehensive assessment.
- -Accuracy: This metric indicates the percentage of correctly classified tweets out of all tweets. For instance, an accuracy of 80% means that 80 out of 100 tweets are classified correctly
- -Precision: Precision reflects the proportion of correctly predicted positive (or negative) sentiments out of all positive (or negative) predictions. High precision means fewer false positives.
- -Recall: Recall measures the proportion of correctly predicted positive (or negative) sentiments out of all actual positive (or negative) sentiments. High recall indicates fewer false negatives.
- -F1-Score:F1-score is the harmonic mean of precision and recall. It offers a balanced view, considering both false positives and false negatives



Model Evaluation

Evaluating models using accuracy, confusion matrix, ROC curve, and AUC of Bernoulli Naïve Bayes Classifier , Logistic Regression and Linear SVM respectively.

Fig 29 Figure of Model Evaluation of 3 different models viz. Bernoulli's Trial, Linear SVM and Logistic Regression

Comparison of Different Models' Performance

We compare the performance of the different classification algorithms (Naive Bayes, Support Vector Machines, Logistic Regression) in terms of the metrics mentioned above. This comparison helps us understand the strengths and weaknesses of each model.

- Graphs and Tables:We use graphs and tables to visually present how each model performed. Bar charts can show accuracy, precision, recall, and F1-score for easy comparison.
- Discussion of Patterns We delve into patterns observed in the results. For instance, which models excel in which sentiment categories, and whether one model consistently outperforms others.

Model Selection and Evaluation

Classification Models

Use of different classification models: Bernoulli Naive Bayes model, Logistic Regression and SVM (Support Vector Machine)

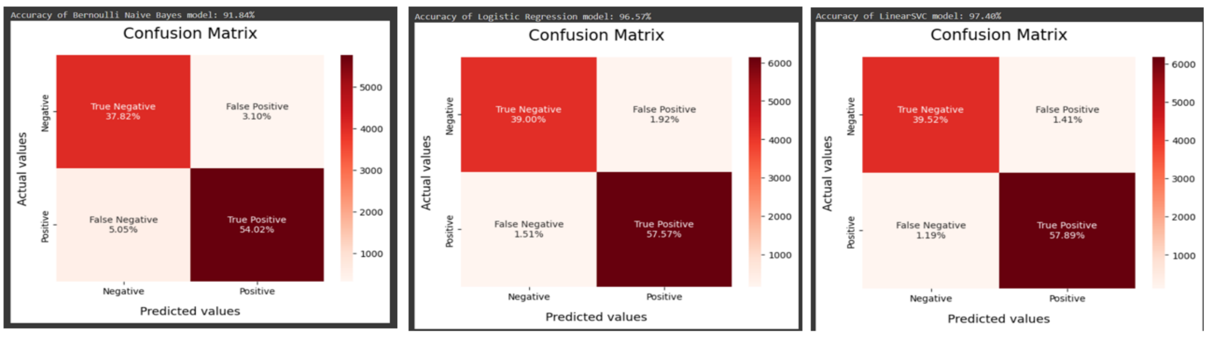


Fig 30 Figure of Model Comparision of 3 different models viz. Bernoulli's Trial, Linear SVM and Logistic Regression

- Implications and Insights:

The model performance results offer insights into which approach is most effective for sentiment analysis on Twitter data. This information is crucial for decision-making and for selecting the best-performing model for further analysis or deployment.

Through a comprehensive discussion of the performance metrics and a thorough comparison of different models, we shed light on the strengths and limitations of each approach, paving the way for an informed conclusion and recommendations for further analysis.

Chapter 6: Results and Discussion

6.2 Sentiment Insights

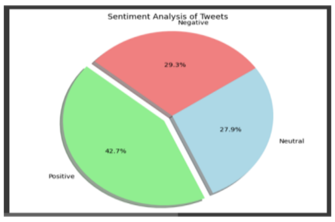
Discussion of Interesting Patterns and Findings from Sentiment Analysis

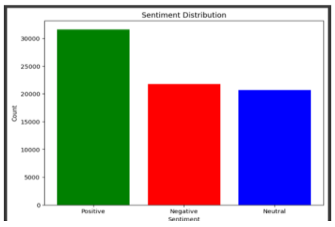
In this section, we delve into the fascinating world of sentiment insights that our analysis has unearthed. By scrutinizing the results, we identify patterns and trends that provide a deeper understanding of how sentiments manifest on Twitter.

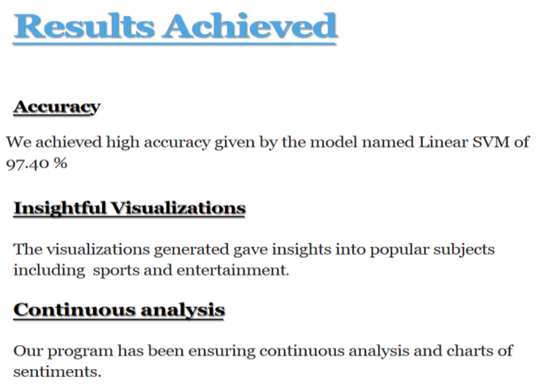
- Dominant Sentiments: We highlight which sentiment category prevails the most within the analyzed tweets. Uncovering the dominant sentiment offers a bird's-eye view of the prevailing emotional landscape on Twitter.
- Common Terms: We explore the language used across different sentiment categories. Understanding the words commonly associated with each sentiment reveals the lexicon through which people express their emotions.

Analysis of Sentiment Shifts in Response to Different Events or Topics Implications and Insights:

By unraveling these intricate sentiment patterns, we uncover a narrative that transcends the numerical metrics. These insights offer businesses, organizations, and researchers a window into the emotional landscape of the public. Armed with these observations, they can make informed decisions, adjust strategies, and understand the pulse of their audience.







Results Achieved

Accuracy

We achieved high accuracy given by the model named Linear SVM of 97.40 %

Insightful Visualizations

The visualizations generated gave insights into popular subjects including sports and entertainment.

Continuous analysis

Our program has been ensuring continuous analysis and charts of sentiments.

Through a comprehensive exploration of sentiment patterns and their correlation with events, we weave a tale that transcends mere data points. This narrative enriches the analysis, contributing a deeper layer of comprehension to the sentiment analysis results.

Conclusion:

In sum, our project's contributions extend beyond our immediate analysis, impacting the broader fields of sentiment analysis and NLP. Our work enhances the tools, techniques, and insights available to researchers and practitioners, fostering advancements and innovation in these dynamic domains.

Chapter 7: Conclusion and Future Work

7.1 Summary of Findings

In this concluding chapter, we revisit the journey of our sentiment analysis project and summarize the significant findings and insights we've garnered through our analysis of Twitter data.

Recap of Key Findings: We revisit the dominant sentiments within the dataset and the prevalent language associated with each sentiment category. This recap reinforces the understanding of sentiment distribution and language patterns.

-Highlighting Unique Insights:We emphasize the unique insights we've extracted from sentiment analysis, such as unexpected sentiment patterns, language nuances, and sentiment shifts in response to events or trending topics.

Implications for Decision-Making: We reflect on how these findings can impact decision-making for businesses, organizations, or researchers seeking to gauge public opinions and emotions. These insights offer valuable guidance for strategies and actions.

Setting the Stage for Future Work:

While our analysis has provided valuable insights, there are always avenues for further exploration and enhancement. We set the stage for potential future work in sentiment analysis:

-Fine-Tuning Models: Further optimizing machine learning models can potentially improve accuracy and robustness in sentiment classification.

-Real-Time Analysis:Implementing real-time sentiment analysis can capture up-to-the-minute sentiments, enabling rapid response to evolving public opinions.

- Domain-Specific Analysis:Customizing sentiment analysis for specific domains or industries can lead to more accurate insights tailored to particular contexts.

Closing Thoughts:

In conclusion, our sentiment analysis project has shed light on the rich emotional tapestry of Twitter conversations. By summarizing our key findings and outlining potential paths for future exploration, we wrap up our journey with a deeper understanding of public sentiments and a roadmap for further analysis and development.

7.2 Contributions-In this section, we underscore the significant contributions that our project has made to the fields of sentiment analysis and Natural Language Processing (NLP). These contributions highlight the innovative approaches and insights generated through our work.

Advancements in Sentiment Analysis:

Robust Sentiment Classification: Our project demonstrates the effective utilization of advanced machine learning models for sentiment classification. By achieving high accuracy and employing various evaluation metrics, we contribute to the enhancement of sentiment analysis accuracy and reliability.

Real-World Insights: Through our in-depth analysis of Twitter data, we offer tangible insights into public sentiments, reactions to events, and prevailing language patterns. These insights can guide decision-makers in understanding public opinions.

- ➢ Knowledge Expansion:
- -Demonstrative Project: By detailing the step-by-step process of our sentiment analysis project, we provide a comprehensive guide for those looking to undertake similar projects. This contributes to the dissemination of practical knowledge in the field.

Future Research Inspiration:

- Exploration of Emerging Techniques:Our project serves as an inspiration for future research endeavors, encouraging the exploration of emerging NLP techniques and their applications in sentiment analysis on evolving social media platforms.

References:

1. "Sentiment Analysis with Python" by Susan Li: https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184

3. "A Comprehensive Guide to Sentiment Analysis" by MonkeyLearn: https://monkeylearn.com/sentiment-analysis/

4. "Twitter Sentiment Analysis using Python" by GeeksforGeeks: https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/

- Tools and Libraries:

1. NLTK (Natural Language Toolkit): A popular Python library for natural language processing.

2. TextBlob: An easy-to-use library built on top of NLTK and provides simple API for common NLP tasks including sentiment analysis.

3. VADER (Valence Aware Dictionary and sEntiment Reasoner): A sentiment analysis tool that is specifically designed for analyzing social media text.

4. scikit-learn: A machine learning library in Python that includes tools for text classification and sentiment analysis.

5. Tweepy: A Python library for accessing the Twitter API, which can be used to collect data for sentiment analysis.