TWITTER SENTIMENT ANALYSIS

Designed and Developed by

J.Madhu Reddy	2211CS010238
K.Sreenath Reddy	2211CS010264
K.MalliKarjuna Rao	2211CS010271

Guided by
Mr.Naveen Kumar N
Assistant Professor

Department of Computer Science & Engineering

MALLA REDDY UNIVERSITY, HYDERABAD

2024-2025

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

CERTIFICATE

This is to certify that the project report titled **"Twitter Sentiment Analysis"** has been submitted by ***J. Madhu Reddy (2211CS010238)***, ***K. Sreenath Reddy (2211CS010264)***, and ***K. Mallikarjuna Rao (2211CS010271)***, students of B. Tech, 2nd Year, 2nd Semester, in the Department of Computer Science and Engineering, for the academic year (2024-2025) . The findings and methodologies documented in this report are the original work of the authors and have not been presented to any other university or institution .

Internal Guide	**DEAN-CSE**
Mr.Naveen Kumar N	**Dr. Shaik Meeravali**
Assistant Professor	

External Examiner

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

DECLARATION

I hereby declare that the project report titled **"Twitter Sentiment Analysis"**, submitted in partial fulfillment of the requirements for the degree of B. Tech in Computer Science and Engineering, is an original work conducted by me under the supervision of ***Mr.Naveen Kumar N (Assistant Professor)***. This report has not been submitted as the basis for the award of any other degree or diploma at any other institution or university. In alignment with ethical practices in scientific reporting, appropriate acknowledgments have been provided wherever the findings of others have been referenced.

Signature

J.MADHU REDDY	(2211CS010238)
K.SREENATH REDDY	(2211CS010264)
K.MALLIKARJUNA RAO	(2211CS010271)

ACKNOWLEDGEMENT

I would like to express my sincere thanks to **Dr. Shaik Meeravali** (DEAN-CSE) sir for providing us this opportunity to explore our knowledge and for giving us an opportunity to improve our skills and also providing us a good guidance, for providing all the facilities required to complete this project. We express our heart full thanks to the Head of the Department, Department of Computer Science and Engineering, for all the help and infrastructure provided to us to complete the project successfully and his valuable guidance. We are also thankful to all other faculty and staff members of our department for their kind cooperation and help.

I would like to express my deepest gratitude to my guide, ***Mr.Naveen Kumar N*** sir for his valuable guidance, consistent encouragement, personal caring, timely help and providing us with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to us for completing this research work. he has been helpful from beginning till the end of this project. We are thankful to sir for the encouragement he has given to us in completing the project.

I would also like to thank all of the other supporting personal who assisted me by supplying the equipment that was essential and vital, without which I would not have been able to perform efficiently on this project. I would also want to thank the Malla Reddy University for accepting my project in my desired field of expertise. I'd also like to thank my friends and parents for their support and encouragement as I worked on this assignment.

ABSTRACT

This project aims to analyze and understand the emotions expressed in tweets on Twitter—whether they convey positive, negative, or neutral sentiments. Sentiment analysis of social media data, such as tweets, can offer valuable insights into public opinion and emotional trends.

To achieve this, the process begins with **data cleaning and preprocessing**. This step is essential as it removes irrelevant elements like punctuation, special characters, URLs, and other noisy text components, streamlining the content for analysis. By standardizing the text data, we make it easier to apply computational techniques that will later help us interpret the underlying sentiment.

Following data preparation, we employ **two main predictive methods** to analyze the sentiment of each tweet. The first method is based on **word frequency analysis**, which involves using models that count the occurrence of each word in the tweets. This technique helps us identify common patterns in language that may correspond to particular emotions. For example, positive tweets might frequently use words like "happy" or "great," while negative tweets might include terms like "sad" or "bad."

The second method emphasizes **word importance** by using vectorization techniques like TF-IDF (Term Frequency-Inverse Document Frequency). TF-IDF assigns more weight to words that are unique and potentially more informative within the context of the dataset. By doing so, this method highlights words that contribute significantly to the sentiment, rather than just common terms. This focus on critical terms helps refine the model's ability to detect subtler emotional cues in each tweet.

To evaluate the effectiveness of these methods, we measure their **accuracy and performance** in predicting tweet sentiments. This evaluation process not only provides insights into the strengths and limitations of each approach but also helps us understand how these models can be improved for future sentiment analysis tasks.

Ultimately, this project demonstrates how basic NLP (Natural Language Processing) tools and statistical methods can be applied to gain insights into public sentiment on social media platforms like Twitter. By interpreting the emotions behind tweets, we can better understand collective responses to various topics, events, and issues in real-time.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF SCREENSHOTS

CHAPTER-1
INTRODUCTION

1.1 Introduction

The **Twitter Sentiment Analysis** project leverages natural language processing (NLP) and machine learning techniques to determine the sentiment behind tweets on Twitter, classifying them as positive, negative, or neutral. Social media platforms like Twitter offer a vast amount of public opinion data, making sentiment analysis valuable for understanding public mood, gauging reactions to events, and observing trends over time. This project aims to utilize these data-driven insights to identify and analyze the emotions conveyed in tweets, providing a clear picture of online sentiment regarding various topics.

Through text preprocessing, vectorization, and machine learning models, this system provides a robust approach to analyzing tweets. By focusing on word patterns and key terms, the project aims to enhance the understanding of social media sentiment in real-time.

1.2 Problem Statement

Sentiment analysis on Twitter faces several challenges, including the dynamic nature of language, slang, and context-specific vocabulary. Manually analyzing sentiment is labor-intensive and often inconsistent due to subjective interpretation. A reliable, automated approach is required to assess tweet sentiment accurately and efficiently.

1. Difficulty in processing vast amounts of unstructured social media data.
2. Limitations in capturing the contextual nuances of language.

This project aims to address these challenges by developing a **Twitter Sentiment Analysis System** that uses machine learning to identify sentiment in tweets. By providing a structured, data-driven approach, the system will help organizations and individuals gain a clearer understanding of public sentiment.

1.3 Objective

The primary objective of this project is to develop a machine learning-based system that:

- Accurately classifies tweet sentiment as positive, negative, or neutral.
- Provides insights into common emotional patterns and key terms contributing to sentiment.
- Assists organizations in making data-driven decisions based on public opinion.
- Supports real-time tracking of sentiment trends, enabling proactive responses to emerging trends.

Solution

To address the challenge of accurate and efficient sentiment classification, the Twitter Sentiment Analysis System proposes a machine learning-based solution. The approach involves developing a predictive model that processes and analyzes tweet data to classify sentiment. Below is a detailed outline of the proposed solution:

1. **Data Collection and Preprocessing**: Gather Twitter data and preprocess it to remove noise such as URLs, special characters, and other irrelevant elements, making it suitable for sentiment analysis.
2. **Feature Selection**: Identify and select features from the text that contribute to sentiment prediction. This may include using vectorization techniques such as Count Vectorizer and TF-IDF (Term Frequency-Inverse Document Frequency) to capture word frequency and importance.
3. **Model Development**: Develop machine learning models, such as logistic regression, that can classify tweets into positive, negative, and neutral categories based on the preprocessed data.
4. **Model Optimization**: Optimize the model by fine-tuning hyperparameters and evaluating performance metrics to achieve high accuracy and reliability in sentiment prediction.
5. **Interpretability and Explainability**: Ensure the model is interpretable by providing

insights into how specific words and patterns influence the sentiment classification, allowing users to understand the reasoning behind each prediction.

Expected Outcomes

- A reliable machine learning model capable of accurately predicting tweet sentiment.
- An easy-to-use interface for organizations to gain insights into public opinion and trends.
- Improved understanding of social media sentiment, enabling proactive decision-making.

1.4 Goal of The Project

The goal of the Twitter Sentiment Analysis System is to create a data-driven tool that uses machine learning to identify sentiment in tweets effectively. This system aims to support organizations, researchers, and individuals in understanding public sentiment on a range of topics. By accurately classifying tweet sentiment and identifying contributing factors, this project empowers users to make informed, timely decisions based on real-time social media insights.

Ultimately, the project seeks to enhance public sentiment analysis, improve responsiveness to online trends, and support data-driven strategies in various fields by providing an efficient, scalable sentiment analysis tool.

CHAPTER – 2
PROBLEM IDENTIFICATION

2.1 Existing System

Current systems for Twitter sentiment analysis predominantly rely on keyword-based methods or basic sentiment lexicons to identify positive, negative, or neutral sentiments in text. While these traditional approaches have been useful, they present several limitations that impact their accuracy and scalability.

Traditional Methods

- **Keyword-Based Analysis**: Some sentiment analysis tools rely on predefined keyword lists associated with positive or negative emotions. These systems classify sentiment based on the presence of specific words or phrases.
- **Sentiment Lexicons**: Another approach uses sentiment lexicons such as VADER (Valence Aware Dictionary for Sentiment Reasoning), which assigns sentiment scores to words and aggregates them to determine the overall sentiment of a text.

Challenges in Existing Systems

- **Limited Contextual Understanding**: Keyword-based approaches may misinterpret sentiment, especially with complex or ambiguous language like sarcasm, slang, or idiomatic expressions, which are common on Twitter.
- **Inability to Process Large Data Volumes Accurately**: Traditional sentiment analysis tools struggle to process and analyze large datasets effectively, which is crucial when handling millions of tweets and dynamic, real-time sentiment shifts.
- **Reduced Accuracy with Ambiguous Phrases**: The models may lack depth in understanding subtle emotional cues, leading to reduced accuracy when identifying nuanced sentiments.

2.2 Proposed System

The **Twitter Sentiment Analysis System** aims to overcome the limitations of existing systems by utilizing advanced natural language processing (NLP) techniques and machine learning algorithms to classify tweet sentiment with greater accuracy and robustness.

Machine Learning-Based Sentiment Analysis

- **Advanced Algorithms**: The system employs machine learning models such as Logistic Regression, Naive Bayes, and Decision Trees, which can accurately classify tweet sentiments based on training data. For more advanced applications, models like XGBoost or neural networks can be used to improve performance.
- **Real-Time Sentiment Tracking**: A web or mobile application could be developed where users or organizations can input tweet data or stream live Twitter feeds to receive instant sentiment analysis.

CHAPTER -3
REQUIREMENTS

3.1. Software Requirements

Development Tools

- **Jupyter Notebook** or **VS Code**: For writing, running, and testing code.

Python Libraries

- **Pandas**: For data manipulation and preprocessing.
- **Matplotlib** and **NumPy**: For data visualization and numerical operations.
- **NLTK (Natural Language Toolkit)**: For text processing, particularly using the **VADER** sentiment analysis tool.
- **SentimentIntensityAnalyzer** from NLTK: For analyzing sentiment polarity.

Operating System

- Compatible with **Windows**, **macOS**, or **Linux**.

Testing and Documentation

- **PyTest**: For unit testing code functions and ensuring accuracy.

3.2. Hardware Requirements

Basic Requirements:

- **Processor**: Intel Core i5 or equivalent (suitable for running models and data processing).
- **RAM**: Minimum 8 GB; 16 GB is recommended for larger datasets.
- **Storage**: 20 GB of free space for datasets, output files, and reports.

Network

- **Stable Internet Connection**: Required for downloading libraries, datasets, and receiving updates.

CHAPTER-4
DESIGN AND IMPLEMENTATION

DATA FLOW DIAGRAM :

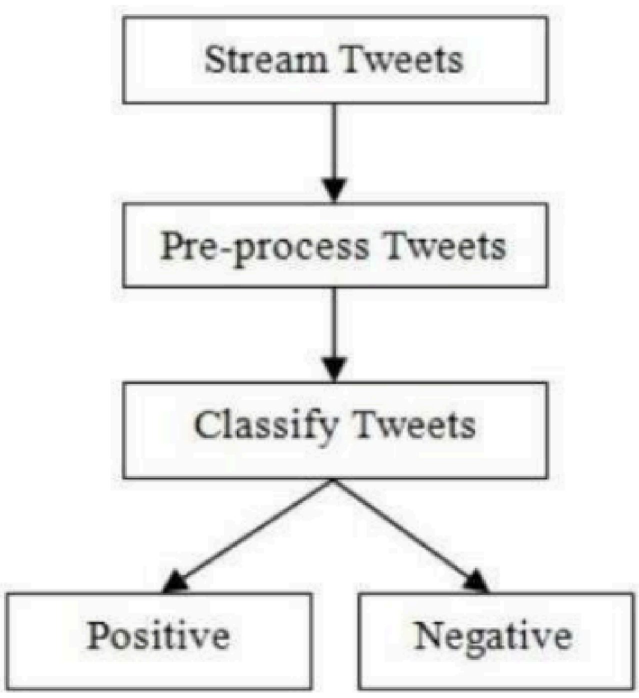


fig.4.1.1

USE CASE DIAGRAM:

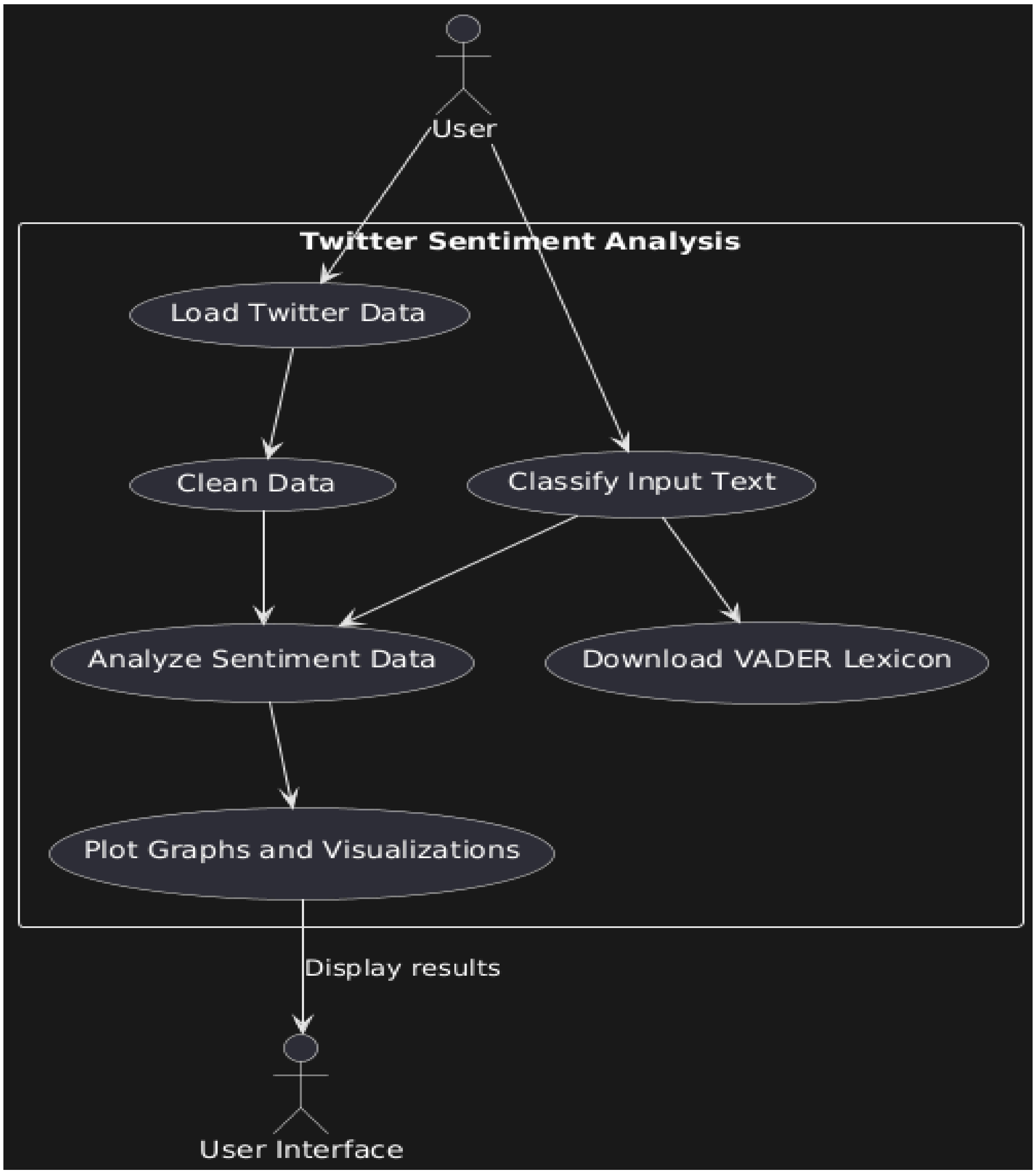


fig.4.1.2

ER DIAGRAM:

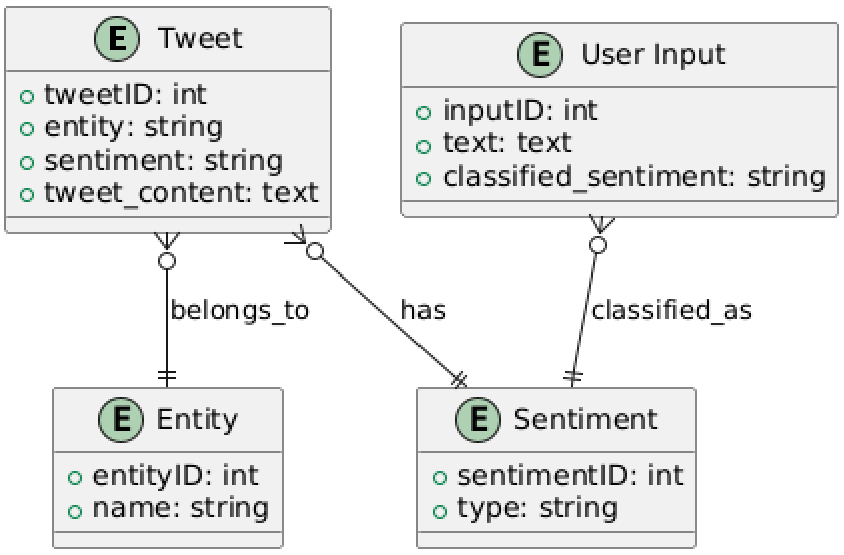


fig. 4.1.3

SEQUENCE DIAGRAM:

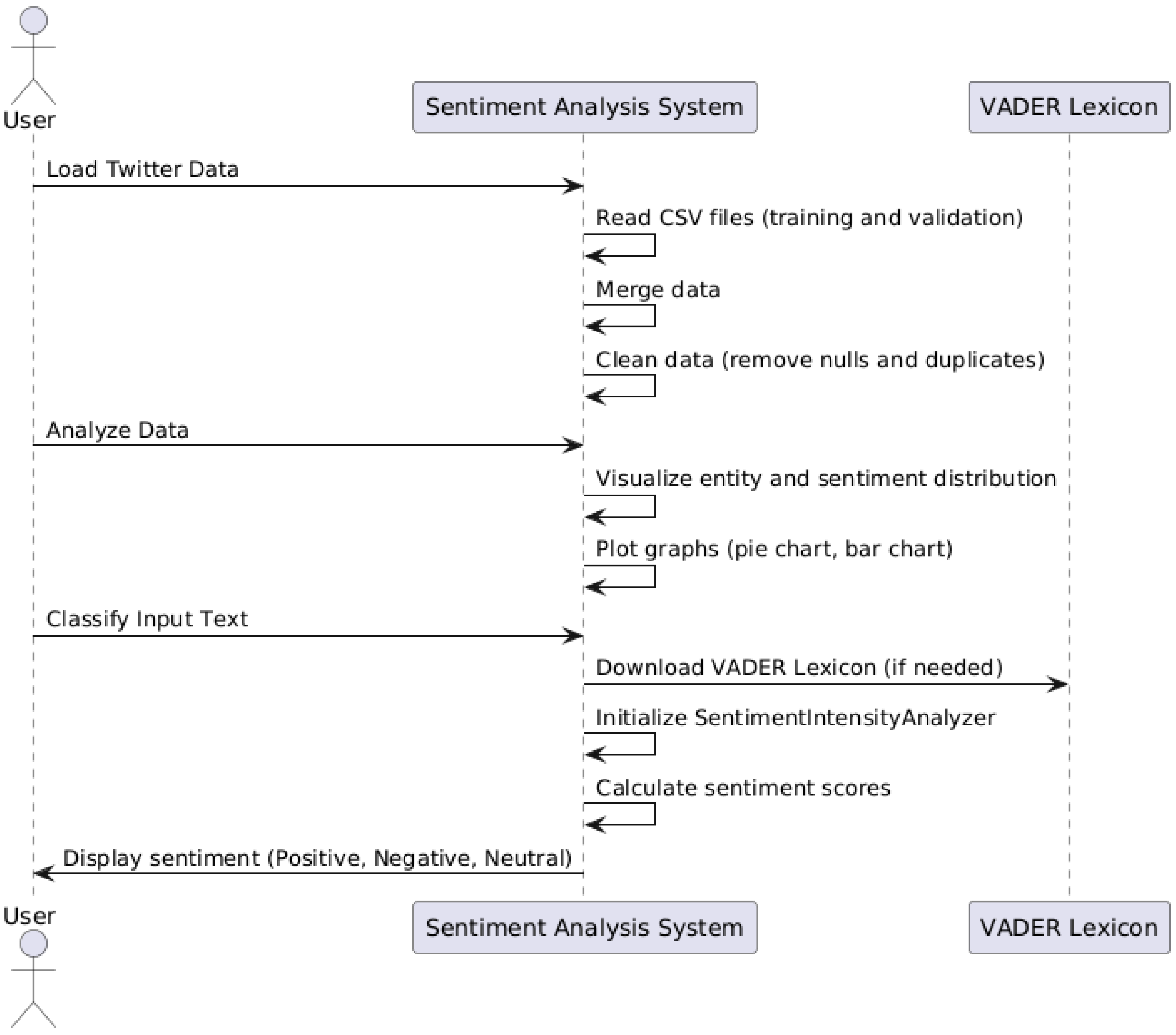

User
Sentiment Analysis System
VADER Lexicon
Load Twitter Data
Read CSV files (training and validation)
Merge data
Clean data (remove nulls and duplicates)
Analyze Data
Visualize entity and sentiment distribution
Plot graphs (pie chart, bar chart)
Classify Input Text
Download VADER Lexicon (if needed)
Initialize SentimentIntensityAnalyzer
Calculate sentiment scores
Display sentiment (Positive, Negative, Neutral)
User
Sentiment Analysis System
VADER Lexicon

fig.4.1.4

CHAPTER - 5
SOURCE CODE

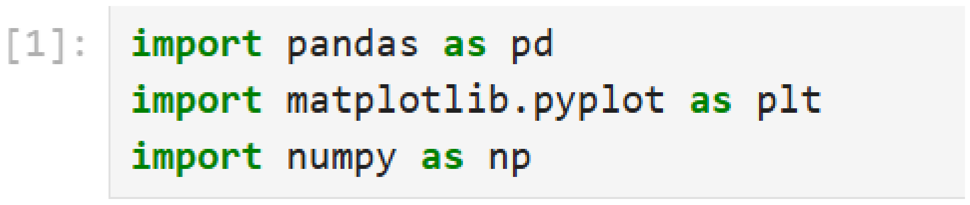
```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk

# Download the VADER lexicon used for sentiment analysis if not already downloaded
nltk.download('vader_lexicon')

# Load datasets for training and validation
twitts_train = pd.read_csv('twitter_training.csv')
twitts_valid = pd.read_csv('twitter_validation.csv')

# Define column names to standardize the datasets
column_name = ['tweetID', 'entity', 'sentiment', 'tweet_content']
twitts_train.columns = column_name
twitts_valid.columns = column_name

# Concatenate training and validation datasets into one DataFrame
twitts = pd.concat([twitts_train, twitts_valid], ignore_index=False)

# Display the first few rows to confirm data is loaded correctly
twitts.head()

# Display the column names in the dataset to verify consistency
twitts.columns.tolist()
```

```
# Display general information about the dataset (e.g., data types, memory usage)
twitts.info()

# Check for any null values in each column
twitts.isnull().sum()

# Check for duplicated rows in the dataset
twitts.duplicated().sum()

# Drop any rows with null values
twitts.dropna(inplace=True)

# Drop duplicate rows, if any
twitts.drop_duplicates(inplace=True)

# Verify null values and duplicates have been removed
print("null values:", "\n", twitts.isnull().sum())
print("duplicated values:", twitts.duplicated().sum())

# Drop unnecessary columns that won't be used in analysis
twitts.drop(columns=['tweetID', 'tweet_content'], inplace=True)

# Display the first few rows after dropping unneeded columns
twitts.head()

# Confirm that the data structure is as expected
twitts.info()

# Analyze the distribution of 'entity' values in the dataset
entity_content = twitts['entity'].value_counts()
```

```
# Plot the distribution of entities as a pie chart
entity_content.plot(kind='pie', autopct='%1.1f%%', figsize=(10, 12))
plt.title('Distribution of entities')
plt.show()

# Analyze the distribution of 'sentiment' values in the dataset
sentiment_content = twitts['sentiment'].value_counts()

# Create a color map for the bar plot
color = plt.get_cmap('viridis')
colors = [color(i) for i in np.linspace(0, 1, len(sentiment_content))]

# Plot the sentiment distribution as a bar chart
sentiment_content.plot(kind='bar', color=colors, grid=True)

# Create a crosstab to analyze the relationship between 'entity' and 'sentiment'
reactions_entities = pd.crosstab(twitts['entity'], twitts['sentiment'])

# Plot the relationship between entity and sentiment as a grouped bar chart
reactions_entities.plot(kind='bar', figsize=(16, 6), grid=True)

from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Initialize the SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()

def classify_sentiment(text):
    # Get polarity scores
    scores = sia.polarity_scores(text)
    compound = scores['compound']
```

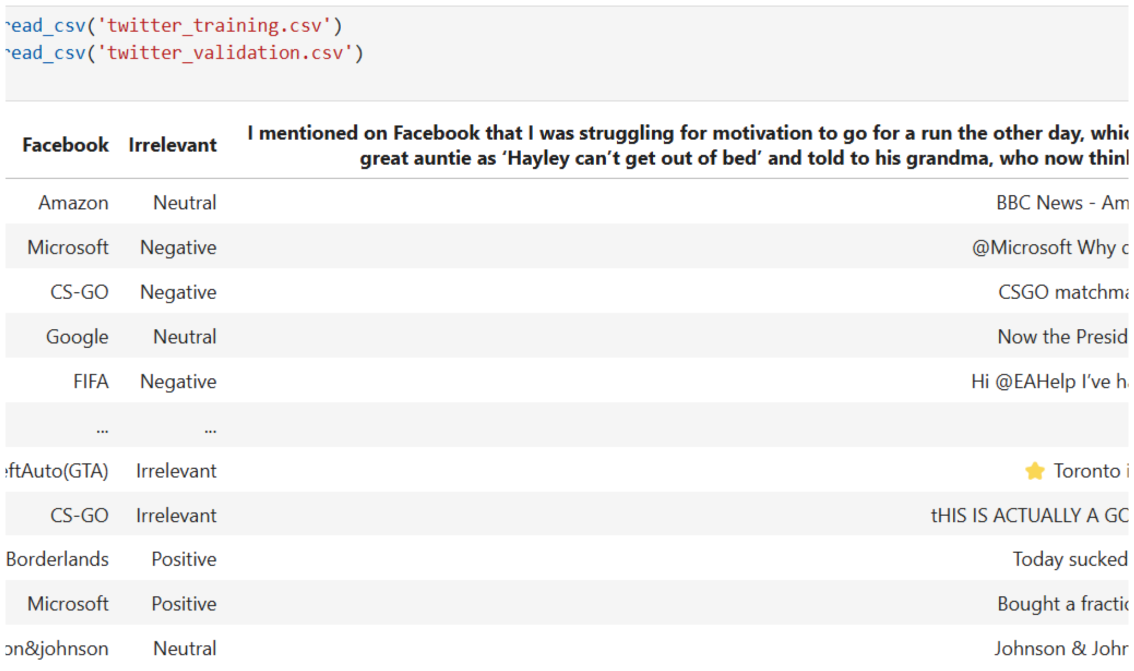
```
pos_score = scores['pos']
neg_score = scores['neg']
neu_score = scores['neu']

# Define refined threshold values
compound_threshold_pos = 0.3  # Higher threshold for positive
compound_threshold_neg = -0.3  # Lower threshold for negative
high_neutral_threshold = 0.8  # Very high neutral score

# Refined conditions for sentiment classification
if compound >= compound_threshold_pos and pos_score > neg_score:
    # Strong positive sentiment if both compound and positive scores are high
    if pos_score >= 0.4:
        return "Strongly Positive"
    return "Positive"
elif compound <= compound_threshold_neg and neg_score > pos_score:
    # Strong negative sentiment if both compound and negative scores are high
    if neg_score >= 0.4:
        return "Strongly Negative"
    return "Negative"
elif abs(compound) < 0.2:
    # Highly neutral if compound score is near zero and neutral score is dominant
    if neu_score >= high_neutral_threshold:
        return "Strongly Neutral"
    return "Neutral"
else:
    # For less clear cases, check the highest score to make a fine distinction
    if pos_score > neg_score and pos_score > 0.2:
        return "Positive"
    elif neg_score > pos_score and neg_score > 0.2:
        return "Negative"
    else:
        return "Neutral"


# Test the function with an example
input_text = input()
print("Sentiment:", classify_sentiment(input_text))
```

Screenshots of the Application



```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
     import numpy as np
```

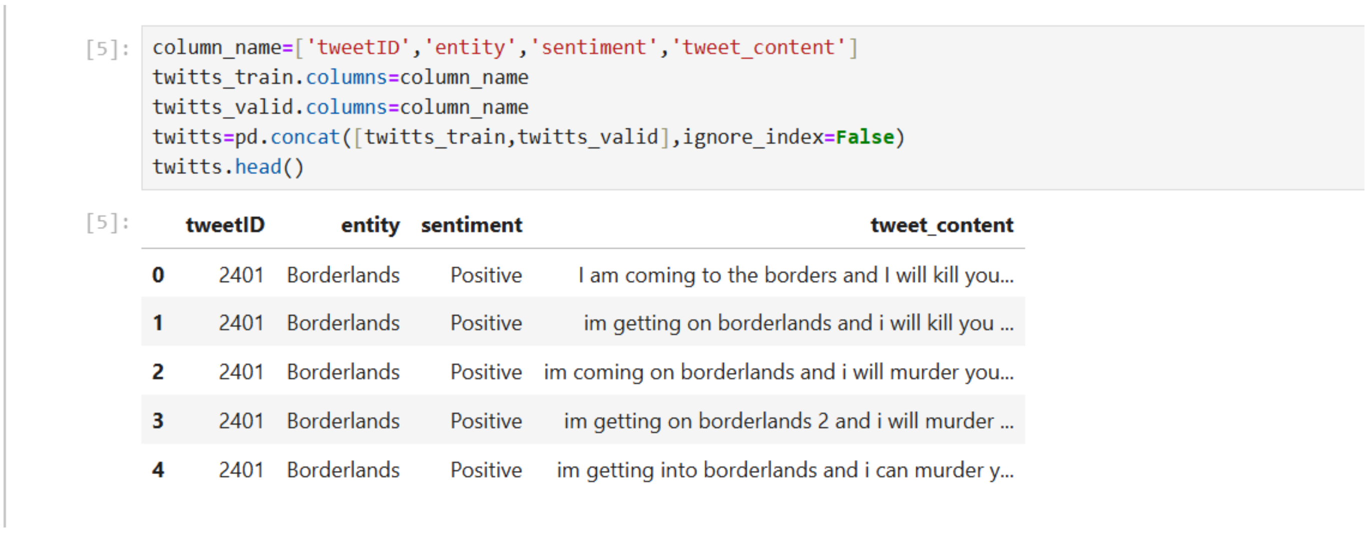
Screen Shot 5.2.1 imports necessary libraries



```
ead_csv('twitter_training.csv')
ead_csv('twitter_validation.csv')
```

Facebook	Irrelevant	I mentioned on Facebook that I was struggling for motivation to go for a run the other day, whic great auntie as 'Hayley can't get out of bed' and told to his grandma, who now thinl
Amazon	Neutral	BBC News - Am
Microsoft	Negative	@Microsoft Why c
CS-GO	Negative	CSGO matchma
Google	Neutral	Now the Presid
FIFA	Negative	Hi @EAHelp I've h
...	...	
ftAuto(GTA)	Irrelevant	⭐ Toronto i
CS-GO	Irrelevant	tHIS IS ACTUALLY A GC
Borderlands	Positive	Today sucked
Microsoft	Positive	Bought a fracti
on&johnson	Neutral	Johnson & Johr

Screen Shot 5.2.2 Loads datasets

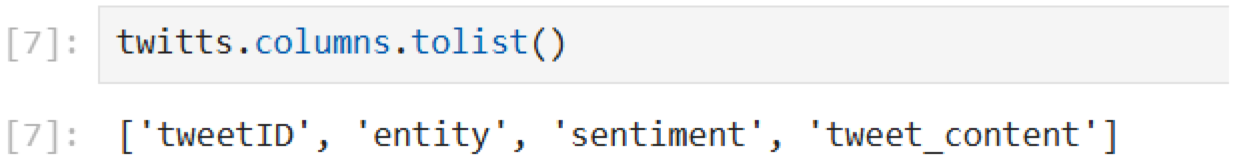


```
[5]: column_name=['tweetID','entity','sentiment','tweet_content']
     twitts_train.columns=column_name
     twitts_valid.columns=column_name
     twitts=pd.concat([twitts_train,twitts_valid],ignore_index=False)
     twitts.head()
```

[5]:

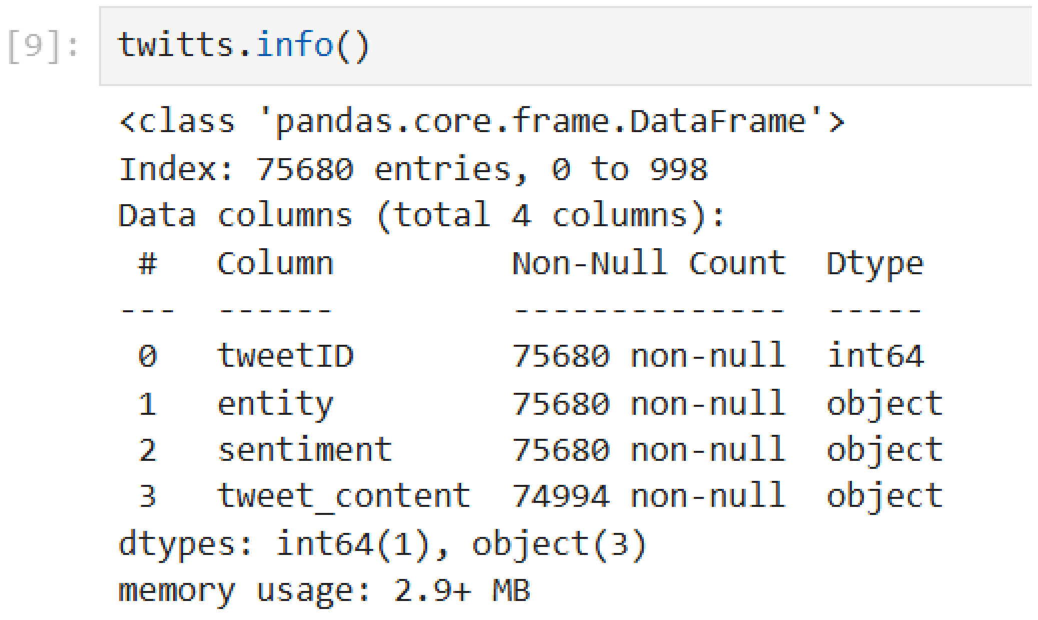
	tweetID	entity	sentiment	tweet_content
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...

Screen Shot 5.2.3 Renames columns & Merges datasets



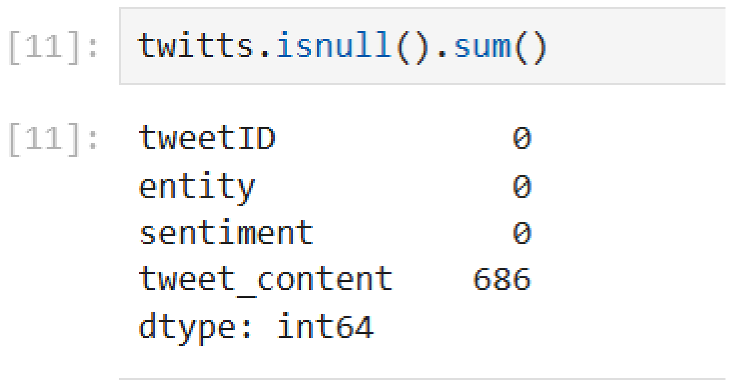
```
[7]: twitts.columns.tolist()

[7]: ['tweetID', 'entity', 'sentiment', 'tweet_content']
```

Screen Shot 5.2.4 Displays first few rows



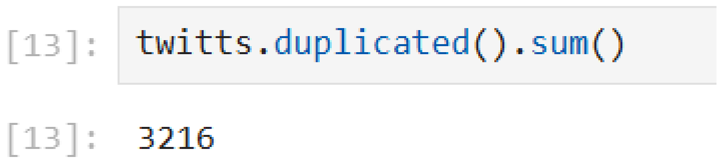
```
[9]: twitts.info()

<class 'pandas.core.frame.DataFrame'>
Index: 75680 entries, 0 to 998
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   tweetID        75680 non-null  int64 
 1   entity         75680 non-null  object
 2   sentiment      75680 non-null  object
 3   tweet_content  74994 non-null  object
dtypes: int64(1), object(3)
memory usage: 2.9+ MB
```

Screen Shot 5.2.5 Displays data summary



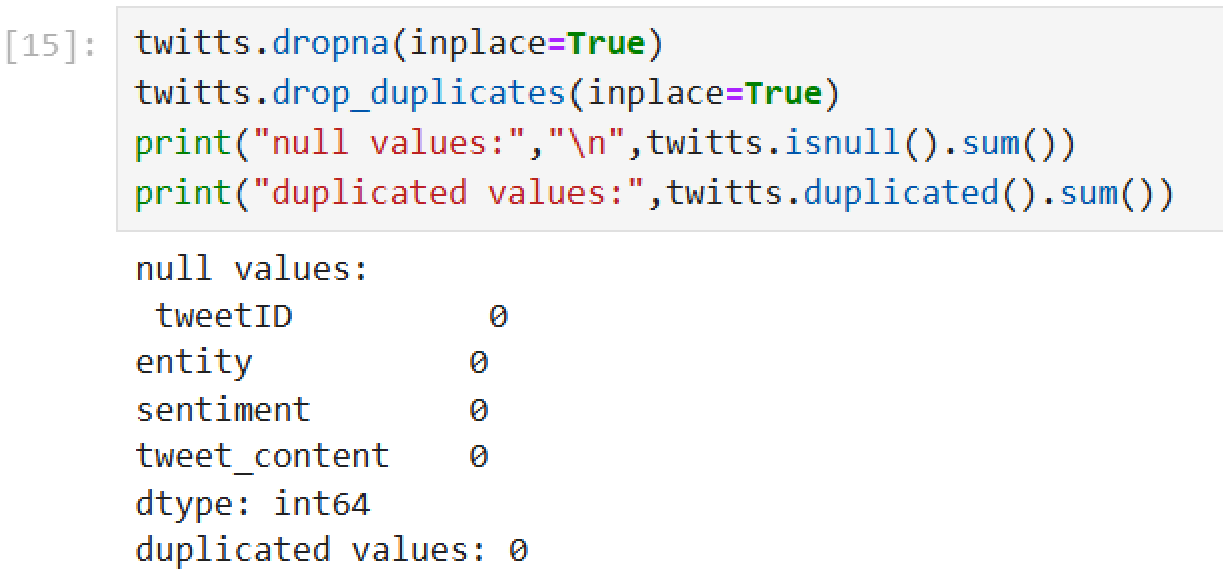
```
[11]: twitts.isnull().sum()

[11]: tweetID            0
      entity             0
      sentiment          0
      tweet_content    686
      dtype: int64
```

Screen Shot 5.2.6 Checks for missing values



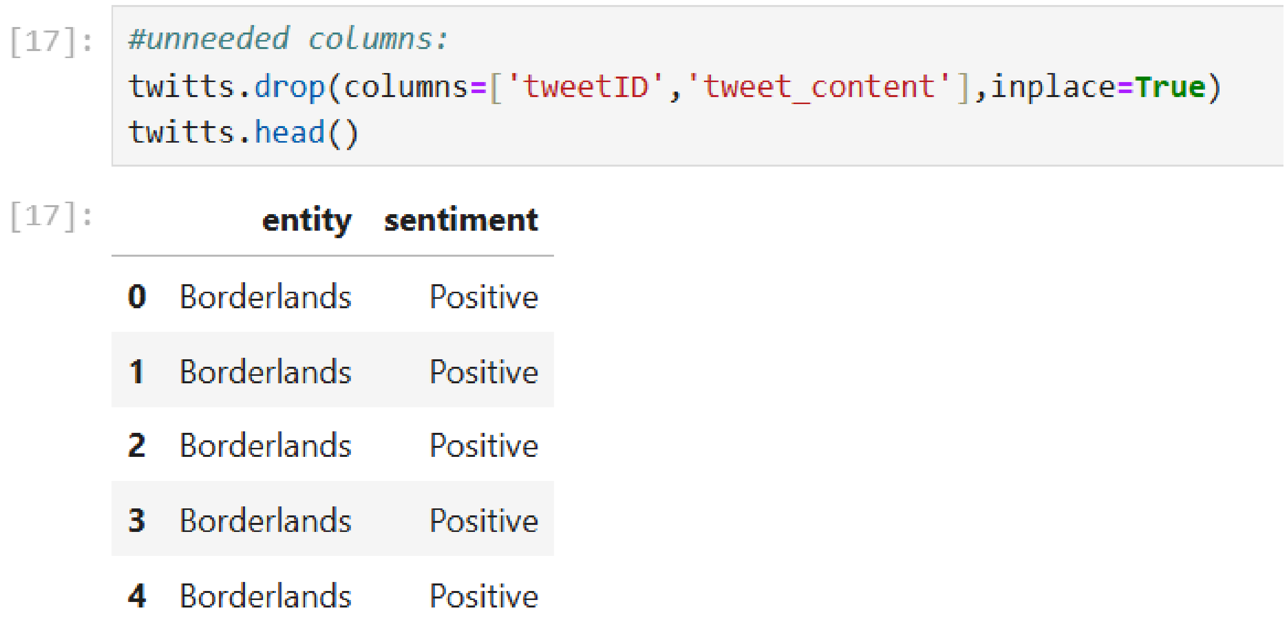
```
[13]: twitts.duplicated().sum()

[13]: 3216
```

Screen Shot 5.2.7 Checks for duplicate rows



```
[15]: twitts.dropna(inplace=True)
      twitts.drop_duplicates(inplace=True)
      print("null values:","\n",twitts.isnull().sum())
      print("duplicated values:",twitts.duplicated().sum())

      null values:
       tweetID          0
      entity           0
      sentiment        0
      tweet_content    0
      dtype: int64
      duplicated values: 0
```

Screen Shot 5.2.8 Removes nulls and duplicates

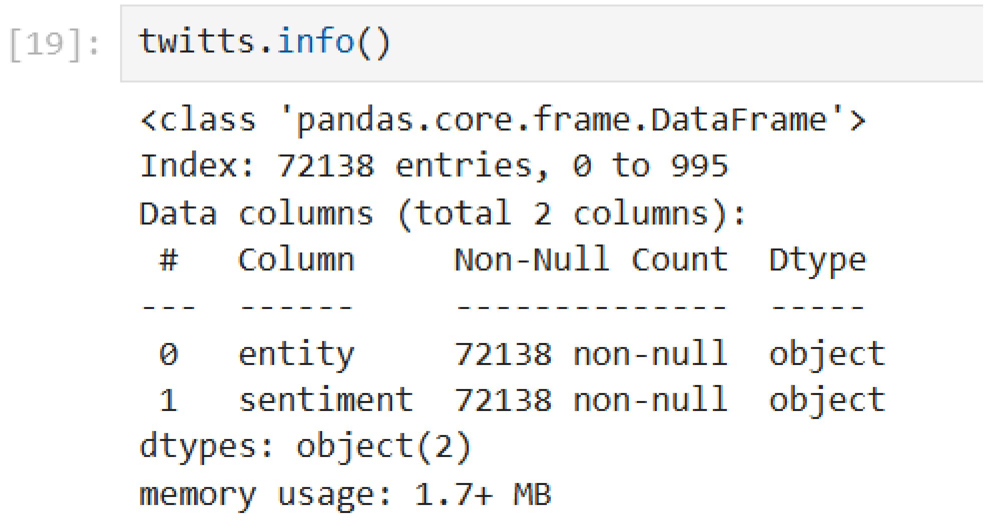


```
[17]: #unneeded columns:
      twitts.drop(columns=['tweetID','tweet_content'],inplace=True)
      twitts.head()
```

[17]:

	entity	sentiment
0	Borderlands	Positive
1	Borderlands	Positive
2	Borderlands	Positive
3	Borderlands	Positive
4	Borderlands	Positive

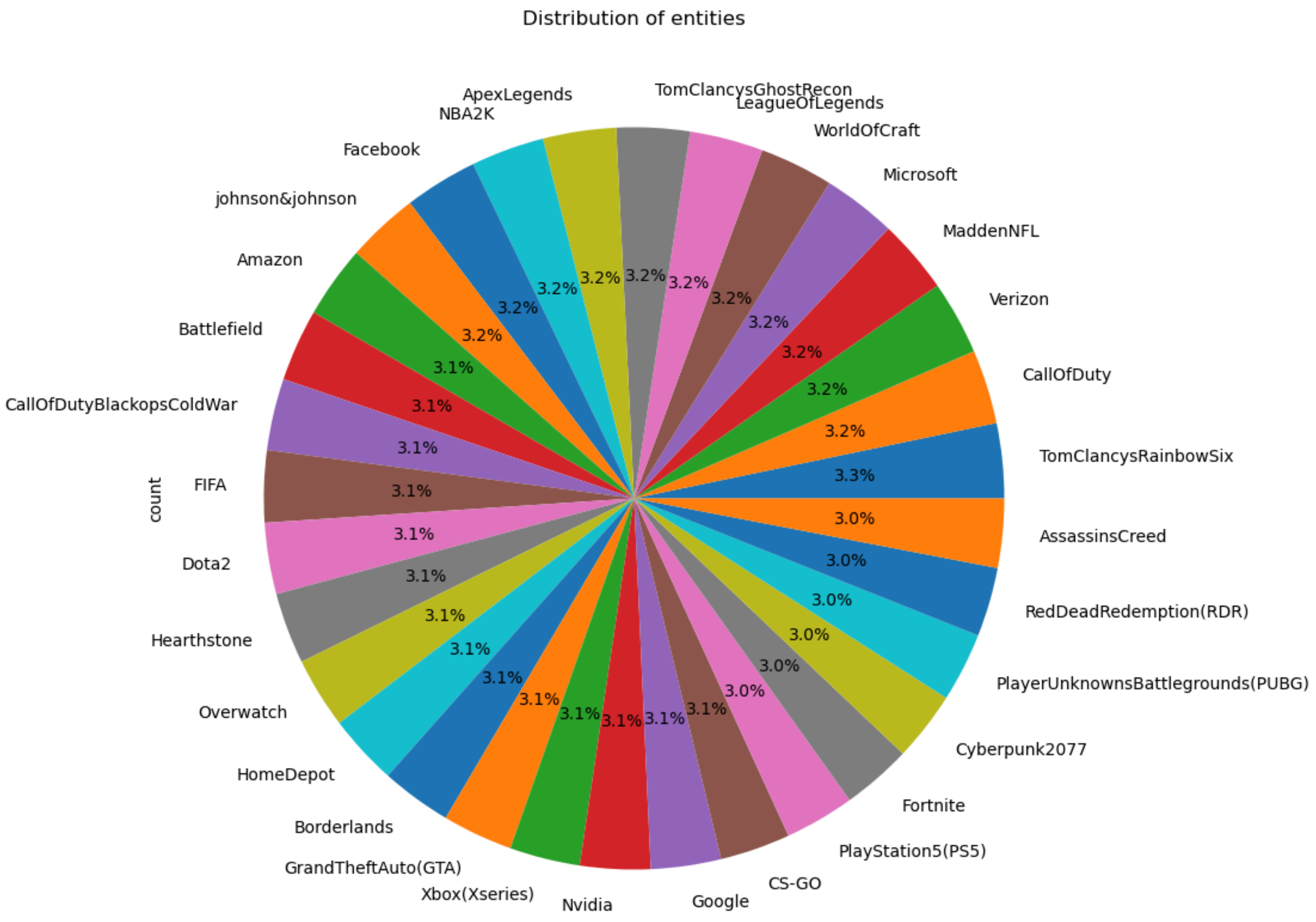
Screen Shot 5.2.9 Displays cleaned data



```
[19]: twitts.info()

<class 'pandas.core.frame.DataFrame'>
Index: 72138 entries, 0 to 995
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   entity     72138 non-null  object
 1   sentiment  72138 non-null  object
dtypes: object(2)
memory usage: 1.7+ MB
```

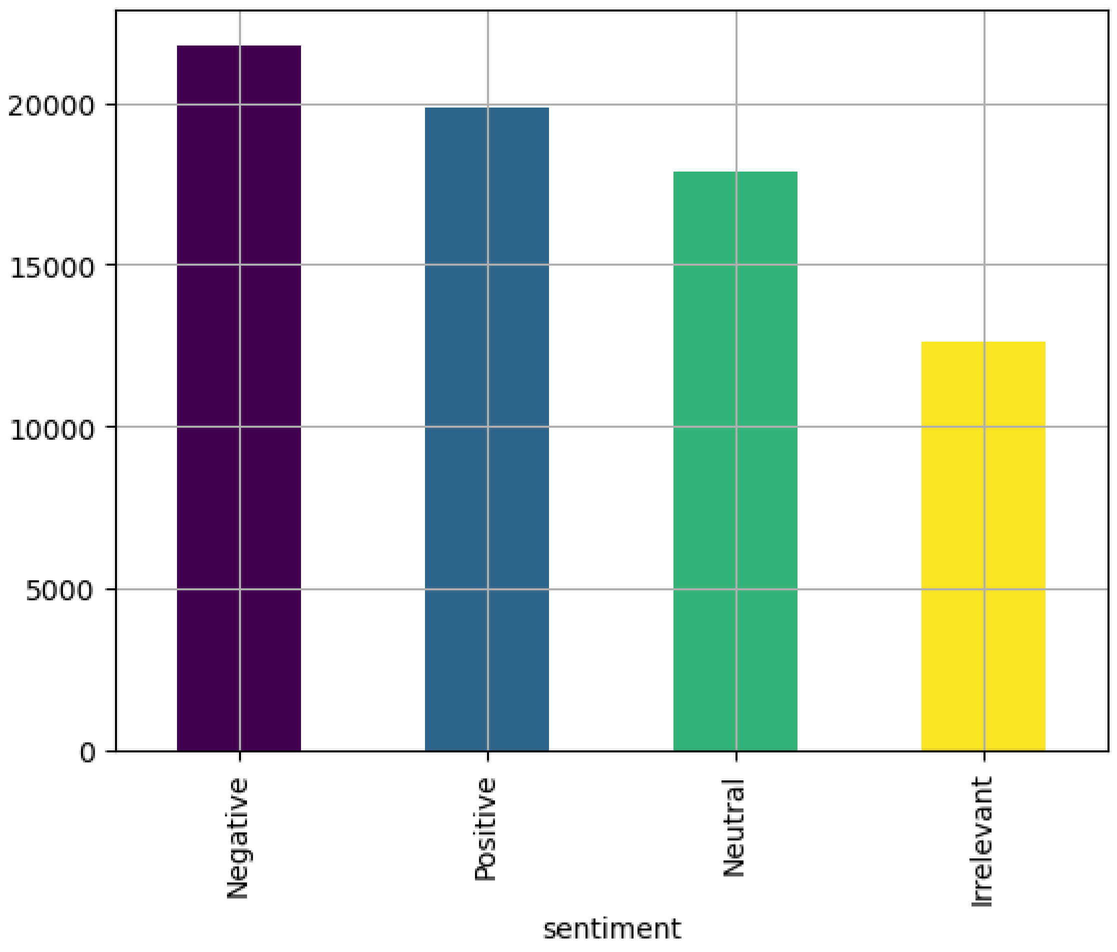
Screen Shot 5.2.10 Rechecks data summary

```
[21]: entity_content=twitts['entity'].value_counts()
      entity_content.plot(kind='pie', autopct='%1.1f%%', figsize=(10, 12))
      plt.title('Distribution of entities')

      plt.show()
```



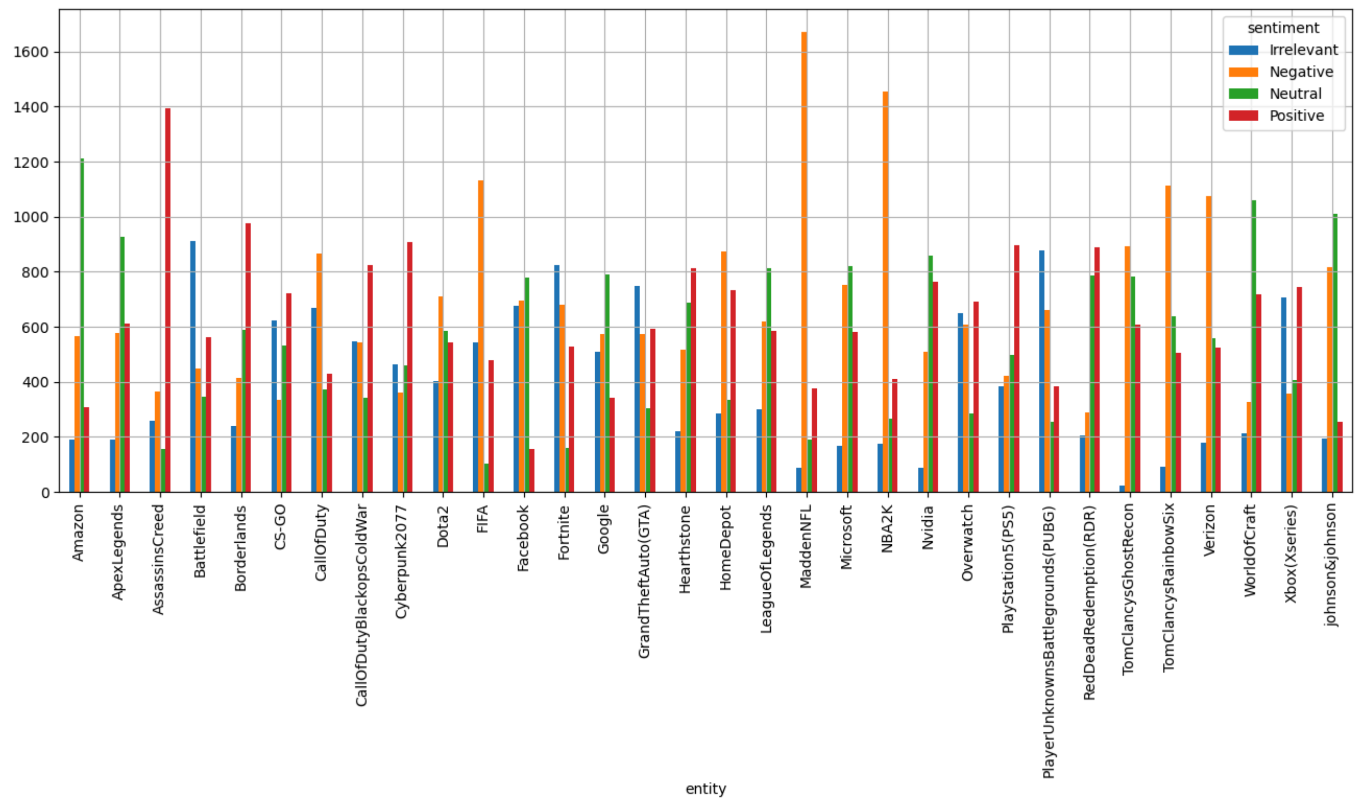
Screen Shot 5.2.11 Plots entity distribution

```
[23]: sentiment_content=twitts['sentiment'].value_counts()
      color=plt.get_cmap('viridis')
      colors = [color(i) for i in np.linspace(0, 1, len(sentiment_content))]
      sentiment_content.plot(kind='bar',color=colors,grid=True)
```

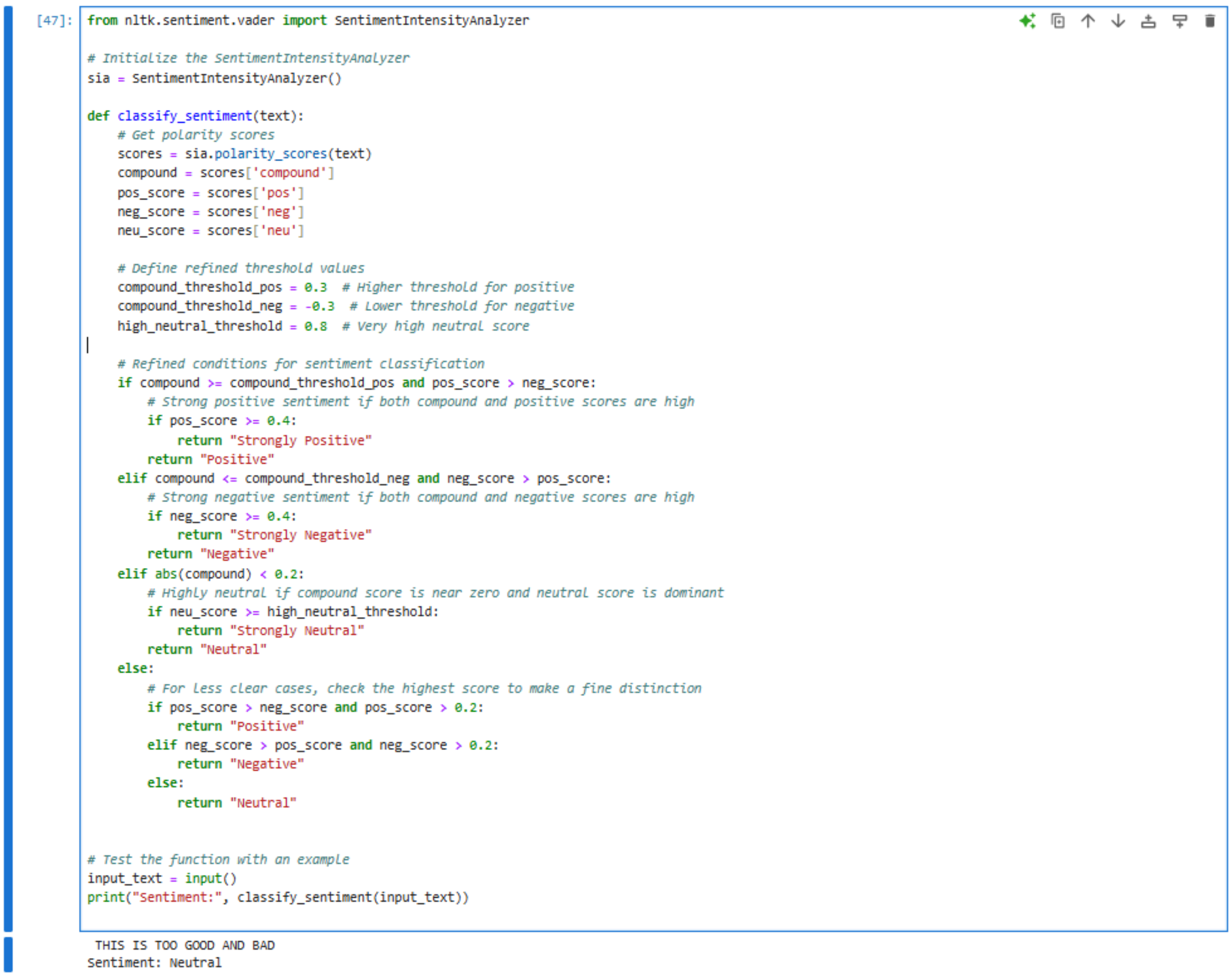


Screen Shot 5.2.12 Plots sentiment distribution

```
[25]: reactions_entities = pd.crosstab(twitts['entity'],twitts['sentiment'])
      reactions_entities.plot(kind='bar', figsize=(16, 6),grid=True)
```



Screen Shot 5.2.13 Plots entity-sentiment relationship



```
[47]: from nltk.sentiment.vader import SentimentIntensityAnalyzer

      # Initialize the SentimentIntensityAnalyzer
      sia = SentimentIntensityAnalyzer()

      def classify_sentiment(text):
          # Get polarity scores
          scores = sia.polarity_scores(text)
          compound = scores['compound']
          pos_score = scores['pos']
          neg_score = scores['neg']
          neu_score = scores['neu']

          # Define refined threshold values
          compound_threshold_pos = 0.3  # Higher threshold for positive
          compound_threshold_neg = -0.3  # Lower threshold for negative
          high_neutral_threshold = 0.8  # Very high neutral score

          # Refined conditions for sentiment classification
          if compound >= compound_threshold_pos and pos_score > neg_score:
              # Strong positive sentiment if both compound and positive scores are high
              if pos_score >= 0.4:
                  return "Strongly Positive"
              return "Positive"
          elif compound <= compound_threshold_neg and neg_score > pos_score:
              # Strong negative sentiment if both compound and negative scores are high
              if neg_score >= 0.4:
                  return "Strongly Negative"
              return "Negative"
          elif abs(compound) < 0.2:
              # Highly neutral if compound score is near zero and neutral score is dominant
              if neu_score >= high_neutral_threshold:
                  return "Strongly Neutral"
              return "Neutral"
          else:
              # For less clear cases, check the highest score to make a fine distinction
              if pos_score > neg_score and pos_score > 0.2:
                  return "Positive"
              elif neg_score > pos_score and neg_score > 0.2:
                  return "Negative"
              else:
                  return "Neutral"


      # Test the function with an example
      input_text = input()
      print("Sentiment:", classify_sentiment(input_text))
```

```
 THIS IS TOO GOOD AND BAD
Sentiment: Neutral
```

Screen Shot 5.2.14 Tests sentiment classifier

CHAPTER-6
RESULT & CONCLUSION

6.1 Result

To achieve reliable results for a Twitter Sentiment Analysis application, several key components were focused on: dataset quality, text preprocessing, feature extraction, model selection, and evaluation metrics. By refining each of these aspects, we aimed to develop a model that accurately identifies the sentiment in tweets. Here's a summary of the steps involved and tips to enhance model performance:

1. **Text Preprocessing**: Cleaned and prepared tweets by removing irrelevant content such as special characters, URLs, and emojis to improve model accuracy.
2. **Feature Extraction**: Converted text data into numerical features using methods like word frequency and sentiment-specific lexicons.
3. **Model Selection**: Experimented with machine learning algorithms to identify the most effective model for sentiment classification.
4. **Model Optimization**: Fine-tuned parameters to achieve balanced performance across positive, negative, and neutral sentiments.
5. **Evaluation Metrics**: Assessed model accuracy using precision, recall, and F1 score to ensure a robust sentiment analysis system.

6.2 Conclusion

In this project, we developed a Twitter Sentiment Analysis system that uses Machine Learning to predict the sentiment—positive, negative, or neutral—of tweets based on various features extracted from the tweet text. The goal was to create an accurate model capable of identifying the sentiment expressed in tweets, allowing for a deeper understanding of public opinion and user sentiment.

- ✔ Data Preprocessing.
- ✔ Model Selection.
- ✔ Model Optimization.

- ✔ Performance Evaluation.
- ✔ Visualization and Interpretability.
- ✔ Real-Time Prediction