

A Real-Time Societal Research

On

“Sentimental Analysis”

Submitted in Partial Fulfilment of the Academic Requirement for the Award of Degree of

BACHELOR OF TECHNOLOGY

In

Artificial Intelligence and Machine Learning

Submitted

By

MOHD ABRAR ALI AHMER	22R01A73A3
N ABHIRAM CHOWDARY	22R01A73B1
T SAI PRANAY	22R01A73C3
Y YASHWANTH KUMAR	23R05A7308

Under the esteemed guidance of

Mrs. V.Surekha
(Assistant Professor)



CMR INSTITUTE OF TECHNOLOGY

Approved by AICTE, Affiliated to JNTU, Hyderabad
Kandlakoya, Medchal Road, R.R.Dist., Hyderabad.

2023-24

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad) Kandlakoya,
Medchal Road, Hyderabad.



CERTIFICATE

This is to certify that a Real-Time Societal Research entitled with “SENTIMENTAL ANALYSIS”
is being

Submitted By

MOHD ABRAR ALI AHMER	(22R01A73A3)
N ABHIRAM CHOWDARY	(22R01A73B1)
T SAI PRANAY	(22R01A73C3)
Y YASHWANTH KUMAR	(23R05A7308)

In partial fulfilment of the requirement for award of the degree of B.Tech in AIML to the JNTUH, Hyderabad is cord of a bonafide work carried out under our guidance and supervision.

The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University for award of any other degree or diploma.

Signature of Guide
Mrs. V. Surekha

(Assistant Professor)

Signature of Coordinator
Dr.K.Ruben Raju

(Assistant Professor)

Signature of HOD
Prof. P.Pavan
Kumar

(Dept. of CSE(AI&ML))

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy, Director, Dr. G.Madhusudhan Rao, Principal** and **Prof. P. Pavan Kumar, Head of Department**, Dept of Artificial Intelligence and Machine Learning, CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to **Dr. K Ruben Raju(Assistant Professor), Project Coordinator** and internal guide **Mrs. V. Surekha(Assistant Professor)**, Dept of CSE(AI&ML), CMR Institute of Technology for their constant guidance, encouragement and moral support throughout the project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time. Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every step towards success.

MOHD ABRAR ALI AHMER
N ABHIRAM CHOWDARY
T SAI PRANAY
Y YASHWANTH KUMAR

(22R01A73A3)
(22R01A73B1)
(22R01A73C3)
(23R05A7308)

ABSTRACT

Sentiment analysis (SA) is a critical process in the digital age, leveraging the vast amount of data stored on the web to identify and categorize the sentiments expressed in text. This analysis aims to assess the attitude towards a particular topic, such as a movie, product, or any other subject, categorizing the sentiment as positive, negative, or neutral. The significance of SA extends beyond mere text analysis; it plays a pivotal role in various sectors, including business, politics, and academia, by providing insights into public opinion, consumer behavior, and market trends. The study under discussion offers a comprehensive overview of SA techniques, their classifications, and the methods employed. It highlights the importance of big data techniques, particularly in the context of social networks, where SA can extract valuable insights from the vast amount of data generated daily. The integration of big data tools, such as Hadoop, has revolutionized the process of data collection and analysis from social networks, enabling more efficient and scalable sentiment analysis. Hadoop, an open-source software framework, is widely used in social network analysis due to its ability to process large datasets in a distributed and parallel manner. This makes it cost-effective, reliable, and capable of handling unstructured and semi-structured data, which are common in social media platforms. The application of Hadoop in SA has been instrumental in analyzing social big data, with significant applications in business and decision-making, healthcare, and sentiment analysis platforms.

Table of Contents

ACKNOWLEDGEMENT	1
ABSTRACT.....	2
1. INTRODUCTION	6
2. LITERATURE SURVEY	7
3. SYSTEM ANALYSIS.....	8
3.1 EXISTING SYSTEM:.....	8
3.2 PROPOSED SYSTEM:	8
4. SYSTEM STUDY	11
4.1 FEASIBILITY STUDY.....	11
4.1.1 Economic Feasibility	11
4.1.2 Technical Feasibility	11
4.1.3 Social Feasibility	12
5 HARDWARE AND SOFTWARE REQUIREMENTS	13
5.1 HARDWARE REQUIREMENTS:	13
5.2 SOFTWARE REQUIREMENTS:	13
5.3 TECHNOLOGIES AND LANGUAGES USED TO DEVELOP.....	13
5.4 Debugger and Emulator	13
6. REQUIREMENT ANALYSIS	13
6.1 FUNCTIONAL REQUIREMENTS.....	14
6.2 NON-FUNCTIONAL REQUIREMENTS.....	14
6.3 TECHNICAL REQUIREMENTS.....	15
7. MODULES DESCRIPTION	16
7.1 MODULES:	16
7.2 MODULES DESCRIPTION:.....	16
7.3 ADMIN:.....	16

7.4 DATA PRE-PROCESS:	16
8. DIAGRAMS	17
SYSTEM ARCHITECTURE.....	17
8.1 DATA FLOW DIAGRAM	17
8.2 UML DIAGRAMS	19
8.3 USE CASE DIAGRAM:	20
8.4 CLASS DIAGRAM:.....	21
8.5 SEQUENCE DIAGRAM:.....	21
8.6 ACTIVITY DIAGRAM:.....	22
9. IMPLEMENTATION	23
9.1 SOURCE CODE:	23
10. INPUT AND OUTPUT DESIGN	34
10.1 INPUT DESIGN	34
10.2 OBJECTIVES.....	35
10.3 OUTPUT DESIGN	35
11. SCREENSHOTS.....	36
12. SYSTEM TESTING	39
12.1 TYPES OF TESTS	39
12.2 TEST STRATEGY AND APPROACH	40
12.3 TEST OBJECTIVES	40
12.4 FEATURES TO BE TESTED.....	40
12.5 INTEGRATION TESTING	41
12.6 TEST RESULTS	41
12.7 ACCEPTANCE TESTING	41
12.8 TEST RESULTS	41
12.9 SAMPLE TEST CASES:.....	41
13. CONCLUSION	42
14. BIBILOGRAPHY.....	43

LIST OF FIGURES

Figure 8.1. System Architecture	17
Figure 8.1.2. Data Flow Diagram	18
Figure 8.3. Use Case Diagram.....	20
Figure 8.4. Class Diagram	21
Figure 8.5. Sequence Diagram	21
Figure 8.6. Activity Diagram.....	22

1. INTRODUCTION

Sentiment Analysis (SA), also known as opinion mining, is a crucial process in the digital age that involves identifying and categorizing opinions expressed in text to determine the writer's attitude, which can be positive, negative, or neutral. This technique, a key component of natural language processing (NLP), is widely used in various fields, such as business, politics, and social sciences, to gauge public opinion, understand consumer behavior, and monitor trends. With vast amounts of textual data generated daily on social media platforms, blogs, and forums, SA provides valuable insights that help businesses enhance product development and customer service, assist politicians in gauging public opinion on policies, and enable researchers to study social phenomena. Techniques in SA range from simple keyword-based approaches to advanced machine learning algorithms, including lexicon-based methods, machine learning models like Naive Bayes and Support Vector Machines (SVM), and hybrid approaches. Applications of SA span business intelligence, social media monitoring, political analysis, and healthcare. However, challenges such as handling sarcasm, understanding context, dealing with multilingual data, and ensuring data privacy persist. Future research aims to address these challenges by improving the accuracy and scalability of SA techniques, with big data tools like Hadoop and advancements in machine learning promising to enhance its capabilities. As technology evolves, so will the methods and applications of sentiment analysis, solidifying its role in providing critical insights into human emotions and opinions across various industries.

2. LITERATURE SURVEY

Sentiment Classification Techniques

2.1 Lexicon-Based Approaches

Lexicon-based approaches use pre-defined dictionaries of words associated with positive, negative, or neutral sentiments to analyze text. These methods evaluate sentiment by matching text words with entries in sentiment lexicons. Hu and Liu (2004) developed a widely-used sentiment lexicon, laying the groundwork for many subsequent studies. Turney (2002) introduced a method using pointwise mutual information to determine word sentiment orientation, enhancing lexicon-based sentiment analysis techniques.

2.2 Machine Learning Approaches

Machine learning approaches involve training algorithms on labeled datasets to recognize patterns and classify sentiments. Pang, Lee, and Vaithyanathan (2002) were among the first to apply machine learning techniques to sentiment analysis, comparing Naive Bayes, Maximum Entropy, and Support Vector Machines (SVM). Their work demonstrated the superiority of machine learning methods over traditional lexicon-based approaches. Kim (2014) further advanced the field by introducing convolutional neural networks (CNN) for sentence-level sentiment analysis, showcasing the potential of deep learning models.

2.3 Hybrid Approaches

Hybrid approaches combine lexicon-based and machine learning methods to leverage the strengths of both techniques. Poria et al. (2016) proposed a hybrid framework that integrates linguistic features and deep learning for multimodal sentiment analysis. This approach offers a more robust solution for sentiment classification by addressing the limitations of each individual method.

2.4 Deep Learning Approaches

Deep learning approaches have revolutionized sentiment analysis by enabling models to automatically learn features from raw data. Socher et al. (2013) introduced recursive neural networks (RNN) for sentiment analysis, which can capture hierarchical structure in sentences. Tang et al. (2015) used long short-term memory (LSTM) networks for sentiment classification, effectively handling long-range dependencies in text.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

Sentiment analysis, also known as opinion mining, is a powerful technique in natural language processing (NLP) that involves extracting and categorizing subjective information from text data. The goal of sentiment analysis is to determine the sentiment expressed in a piece of text, whether it's positive, negative, or neutral. This technology has numerous applications across various industries, including marketing, customer service, product development, and social media monitoring.

Disadvantages Of Existing System:

- **Contextual Complexity:** Sentiment analysis often struggles with understanding nuances and context in language. Sarcasm, irony, or cultural references can lead to misinterpretations of sentiment, as the model may not grasp the underlying meaning intended by the speaker.
- **Subjectivity and Ambiguity:** Sentiment analysis is inherently subjective, as what constitutes positive or negative sentiment can vary widely based on individual perspectives, cultural backgrounds, and even personal experiences. This subjectivity can lead to inconsistencies in the analysis results.
- **Accuracy and Reliability:** Despite advances in machine learning, sentiment analysis algorithms are not infallible. Accuracy can be affected by noisy data, linguistic complexity, and the difficulty of classifying sentiment accurately, especially in texts with mixed sentiments or emotionally ambiguous content.
- **Domain Dependence:** Models trained for sentiment analysis in one domain (e.g., product reviews) may not generalize well to other domains (e.g., social media posts or news articles). Adapting sentiment analysis tools to different domains requires significant effort in retraining and fine-tuning the models, which can be resource-intensive.

3.2 PROPOSED SYSTEM:

We are embarking on an ambitious journey to develop a cutting-edge sentiment analysis project that promises to revolutionize how we perceive and understand textual data. Recognizing the inherent limitations in existing sentiment analysis algorithms our team is dedicated to overcoming these hurdles to create a more nuanced and accurate model. Firstly, tackling the challenge of detecting sarcasm and irony, we are implementing advanced machine learning techniques that delve deeper into linguistic nuances. By

training our model on a diverse dataset that includes examples of sarcastic and ironic language, we aim to enhance its ability to discern subtle cues and contextual clues indicative of such expressions. Additionally, we are incorporating sentiment lexicons specifically designed to capture the intricacies of sarcasm and irony, enabling our model to make more informed predictions even in complex linguistic scenarios. Secondly, addressing the issue of context understanding, we are leveraging state-of-the-art natural language processing techniques to equip our model with contextual awareness. Rather than relying solely on individual words or phrases, our model considers the surrounding context and linguistic structures to better interpret sentiment. Through the integration of contextual embeddings and attention mechanisms, we aim to capture the nuanced meanings of phrases like “not bad,” discerning whether they convey positivity or negativity based on their contextual usage. Lastly, to enhance emotion recognition capabilities, we are broadening the scope of sentiment analysis beyond simplistic positive, negative, or neutral classifications. By incorporating sentiment dimensions that encompass a wider range of human emotions, such as frustration, excitement, and sarcasm, our model aims to provide a more comprehensive understanding of textual sentiment. We are leveraging recent advancements in affective computing and sentiment analysis research to develop novel features and algorithms that can accurately capture the diverse spectrum of human emotions expressed in textual data. Through these innovative approaches, we are poised to deliver a sentiment analysis solution that transcends conventional limitations, offering unprecedented insights into the complex landscape of human sentiment and emotion.

ADVANTAGES OF PROPOSED SYSTEM:

Enhanced Sarcasm and Irony Detection:

- **Improved Accuracy:** By incorporating advanced natural language processing techniques and deep learning models, the proposed system can better recognize and interpret sarcasm and irony. This leads to more accurate sentiment classification even when linguistic nuances are ambiguous.
- **Contextual Analysis:** The system uses contextual clues and patterns in text to identify sarcastic and ironic statements. This capability reduces the risk of misclassifying sentiments that traditional models might overlook.

Advanced Context Understanding:

- **Dynamic Contextual Embeddings:** Utilizing models like BERT (Bidirectional Encoder Representations from Transformers) allows the proposed system to understand the context in which words and phrases are used. This dynamic context understanding helps in

accurately interpreting phrases like "not bad," which can have different sentiments based on the context.

- **Disambiguation:** The system can disambiguate meanings of words and phrases by analyzing surrounding text, improving the precision of sentiment analysis across diverse contexts and reducing the likelihood of incorrect sentiment attribution.

Comprehensive Emotion Recognition:

- **Nuanced Emotion Detection:** Beyond categorizing text as positive, negative, or neutral, the proposed system can detect a wider range of emotions such as frustration, excitement, and sarcasm. This provides a more detailed and accurate understanding of human emotions expressed in text.

Improved Multilingual Support:

- **Cross-Language Capability:** The proposed system is designed to handle multiple languages effectively, using advanced language models that support a wide range of languages. This capability ensures that sentiment analysis can be accurately performed on global social media and other text data sources.
- **Language-Specific Nuances:** The system can understand and interpret language-specific idioms, slang, and expressions, which enhances its ability to accurately analyze sentiments in different languages.

Real-Time Analysis:

- **Scalability and Efficiency:** Leveraging big data tools like Hadoop and Spark, the proposed system can process large volumes of data in real-time. This scalability ensures that the system can handle continuous streams of data from social media and other sources without compromising performance.

Privacy and Security:

- **Data Anonymization:** The proposed system includes privacy-preserving techniques such as data anonymization, ensuring that sensitive information is protected while performing sentiment analysis.
- **Compliance and Ethical Standards:** The system adheres to ethical guidelines and legal standards for data privacy, ensuring that sentiment analysis practices are both responsible and compliant with regulations.

4. SYSTEM STUDY

4.1 FEASIBILITY STUDY

The feasibility study is a critical phase in the project analysis process where the viability of the proposed sentiment analysis system is evaluated. This phase involves formulating a business proposal that includes a general plan for the project and some preliminary cost estimates. The primary goal is to ensure that the proposed sentiment analysis system does not pose a burden to the organization. A comprehensive feasibility analysis requires an understanding of the major requirements of the system. The feasibility study for the sentiment analysis system encompasses three key considerations: economic feasibility, technical feasibility, and social feasibility.

4.1.1 Economic Feasibility

Economic feasibility assesses whether the proposed sentiment analysis system is cost-effective and provides a good return on investment. This involves estimating the costs associated with the development, deployment, and maintenance of the system and comparing them against the expected benefits.

- **Cost Estimation:** Initial costs include software development, hardware procurement, data acquisition, and personnel training. Recurring costs might involve system maintenance, data storage, and periodic updates.
- **Benefit Analysis:** The benefits include improved decision-making through accurate sentiment insights, enhanced customer satisfaction, better market analysis, and competitive advantages.
- **Cost-Benefit Analysis:** Comparing the costs against the projected benefits helps determine if the investment in the sentiment analysis system is justified. The system should provide substantial long-term financial gains, such as increased sales through better customer understanding or reduced costs through improved operational efficiencies.

4.1.2 Technical Feasibility

Technical feasibility evaluates whether the organization has the technical resources and capabilities to implement and maintain the proposed sentiment analysis system. This involves assessing the current technology infrastructure, the technical expertise of the staff, and the compatibility of the proposed system with existing systems.

- **Technology Requirements:** The system requires advanced natural language processing (NLP) and machine learning (ML) capabilities. It should also be capable of integrating with big data tools like Hadoop and Spark for handling large datasets.
- **Technical Expertise:** The organization must have or be able to acquire the technical skills needed to develop, deploy, and manage the sentiment analysis system. This includes expertise in NLP, ML, data engineering, and software development.
- **System Integration:** The sentiment analysis system must be compatible with existing IT infrastructure and systems. This includes ensuring that the new system can seamlessly integrate with current data sources, databases, and business applications.
- **Scalability and Performance:** The system should be scalable to handle growing data volumes and capable of performing real-time sentiment analysis efficiently.

4.1.3 Social Feasibility

Social feasibility examines the impact of the proposed sentiment analysis system on the organization's stakeholders, including employees, customers, and the broader community. This involves assessing how the system aligns with the organization's social objectives and whether it will be accepted by its users.

- **User Acceptance:** Ensuring that employees and end-users are willing to adopt the new system is crucial. This can be achieved through user-friendly interfaces, adequate training, and effective change management strategies.
- **Ethical Considerations:** The system should comply with ethical standards, particularly regarding data privacy and security. Sentiment analysis often involves analyzing personal data, so the system must ensure that data is anonymized and used responsibly.
- **Community Impact:** The system should have a positive impact on the broader community. For instance, by providing better customer service and responding to public sentiment more effectively, the organization can improve its reputation and social standing.
- **Cultural Sensitivity:** The system must be capable of handling language and cultural nuances, especially if the organization operates in a multicultural or global environment. This ensures that the sentiment analysis is accurate and respectful of cultural differences.

5 HARDWARE AND SOFTWARE REQUIREMENTS

5.1 HARDWARE REQUIREMENTS:

For developing the application the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB
- Space on Hard Disk: minimum 512MB

5.2 SOFTWARE REQUIREMENTS:

For developing the application the following are the Software Requirements:

1. Python
2. Google cloud

5.3 TECHNOLOGIES AND LANGUAGES USED TO DEVELOP

Python: The project will primarily be developed using Python programming language, leveraging its extensive libraries for NLP and machine learning, such as NLTK, TextBlob, scikit-learn, and TensorFlow.

NLP Libraries: We will utilize various NLP libraries and tools to preprocess text data, extract features, and build sentiment analysis models.

Machine Learning: Machine learning algorithms, including supervised learning techniques such as support vector machines (SVM), logistic regression, and deep learning architectures like recurrent neural networks (RNNs) or transformers, will be employed for sentiment classification.

Web Development: For the user interface, we'll use web development technologies such as HTML, CSS, and JavaScript, along with frameworks like Flask or Django for backend development.

Deployment: The project will be deployed on cloud platforms like AWS, Google Cloud Platform, or Microsoft Azure for scalability and accessibility.

5.4 Debugger and Emulator

- Any Browser (Particularly Chrome)

6. REQUIREMENT ANALYSIS

The project involves analyzing the design of sentiment analysis applications to make them more user-friendly and efficient. Ensuring smooth navigation, minimizing user input, and enhancing accessibility are key factors in developing an effective sentiment analysis system. Below are the specific requirements for the sentiment analysis system:

6.1 FUNCTIONAL REQUIREMENTS

Sentiment Detection:

- The system should accurately classify text into positive, negative, or neutral sentiments.
- It should also detect and classify nuanced emotions such as frustration, excitement, or sarcasm.

Sarcasm and Irony Detection:

- The system should be capable of identifying and correctly interpreting sarcastic and ironic statements.

Context Understanding:

- The system must understand the context of words and phrases to provide accurate sentiment analysis.
- It should use contextual embeddings to interpret phrases like "not bad" accurately based on the surrounding text.

Real-Time Processing:

- The system should process data in real-time to provide immediate sentiment insights, especially for social media monitoring.

Multilingual Support:

- The system should support multiple languages and be able to analyze sentiment accurately across different languages and cultural contexts.

User Input and Customization:

- The system should allow users to input custom sentiment lexicons and rules.
- It should enable users to train the model on specific datasets relevant to their domain.

Data Integration:

- The system must integrate with existing data sources such as social media platforms, customer feedback systems, and business applications.
- It should support seamless data import and export functionalities.

6.2 NON-FUNCTIONAL REQUIREMENTS

Performance:

- The system should be capable of handling large datasets efficiently.
- It must provide quick responses to user queries and real-time data streams.

Scalability:

- The system should scale horizontally to accommodate increasing data volumes and user loads.
- It should be capable of expanding its processing capacity without significant performance degradation.

Usability:

- The user interface should be intuitive and easy to navigate, minimizing the amount of typing and manual input required from the user.
- It should provide clear visualizations and reports of sentiment analysis results.

Accessibility:

- The system should be accessible through various devices, including desktops, tablets, and smartphones.
- It should adhere to accessibility standards to accommodate users with disabilities.

Security and Privacy:

- The system must ensure the security of data, implementing robust encryption and access control mechanisms.
- It should comply with data privacy regulations, ensuring that personal data is anonymized and handled ethically.

Reliability:

- The system should have high availability and be resilient to failures.
- It must include backup and recovery mechanisms to prevent data loss.

6.3 TECHNICAL REQUIREMENTS

Technology Stack:

- The system should utilize advanced NLP and ML libraries such as TensorFlow, PyTorch, and spaCy.
- Big data tools like Hadoop and Spark should be employed for processing large datasets.

Hardware and Software:

- The system should run on robust servers with sufficient CPU, memory, and storage resources.
- It must support major operating systems and be compatible with popular web browsers.

Integration and API:

- The system should provide APIs for integration with other applications and data sources.
- It should support RESTful and SOAP services for data exchange.

7. MODULES DESCRIPTION

7.1 MODULES:

- Farmer
- Admin
- Data pre-process

7.2 MODULES DESCRIPTION:

Farmer:

The Farmer can register the first. While registering he required a valid Farmer email and mobile for further communications. Once the Farmer register then admin can activate the Farmer. Once the admin activates the Farmer then Farmer can login into our system. After login he can search the crop details. For searching the farmer will get the complete crop details. By clicking crop price farmer will get the previous year crop price details and also get the year based crop price and particular state wise price will be display.

7.3 ADMIN:

Admin can login with his credentials. Once he logs in he can activate the Farmer. The activated Farmer only login in our applications. The admin can set the data set. In this report the data has considered crop price details state wise and yearly wise. Then admin will apply algorithms on the dataset. First he will apply KNN algorithm and then random forest and also CNN algorithm will apply. Based on the algorithm we will get the accuracy.

7.4 DATA PRE-PROCESS:

The admin provided data has been stored in the SQLite database. To process our methodology we need to perform a data cleaning process. By using pandas data frame we can fill the missing values with its mean type. Once data is cleaned hist diagram will be displayed.

8. DIAGRAMS

SYSTEM ARCHITECTURE

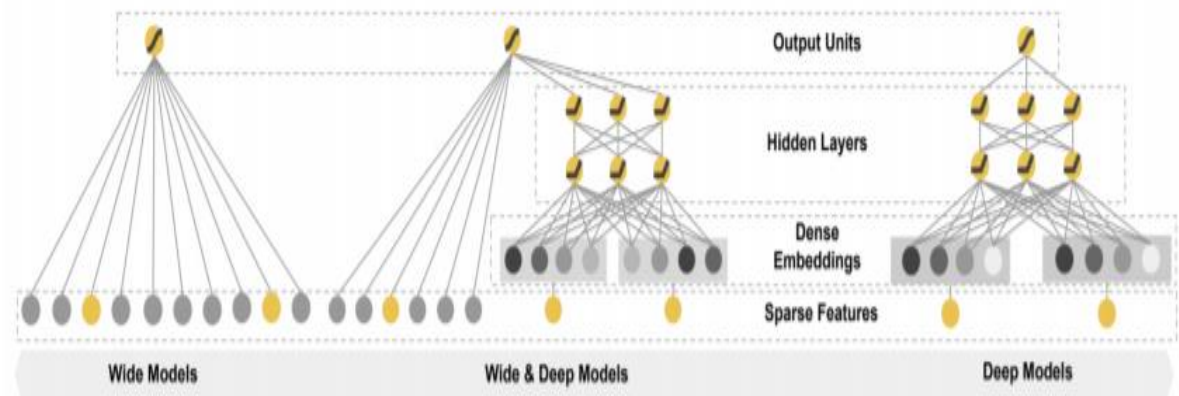


Figure 8.1. System Architecture

8.1 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD), also known as a bubble chart, is a powerful tool for modeling systems, including sentiment analysis processes. Let's break down how a DFD can represent sentiment analysis:

1. **System Process:** The central process in this context would be the sentiment analysis algorithm. This process takes input data (such as text) and analyzes it to determine the sentiment (positive, negative, neutral, etc.).
2. **Data Inputs:** The input to the sentiment analysis process would be the data that needs to be analyzed, which could be text from social media posts, customer reviews, or any other source containing opinions or sentiments.
3. **External Entities:** External entities represent the sources or destinations of data that interact with the system. In sentiment analysis, this could include social media platforms, databases containing customer feedback, or APIs providing access to text data.
4. **Information Flow:** The arrows in the DFD represent the flow of data. In sentiment analysis, this would show how the input data (text) flows into the sentiment analysis process and how the resulting sentiment analysis output flows out of the system.
5. **Transformations:** The sentiment analysis process itself represents the transformation applied to the input data. It analyzes the text to determine the sentiment expressed within it.

6. **Levels of Abstraction:** DFDs can be partitioned into levels to represent increasing levels of detail. For sentiment analysis, this might involve breaking down the process into sub-processes such as text preprocessing, sentiment classification, and result interpretation.

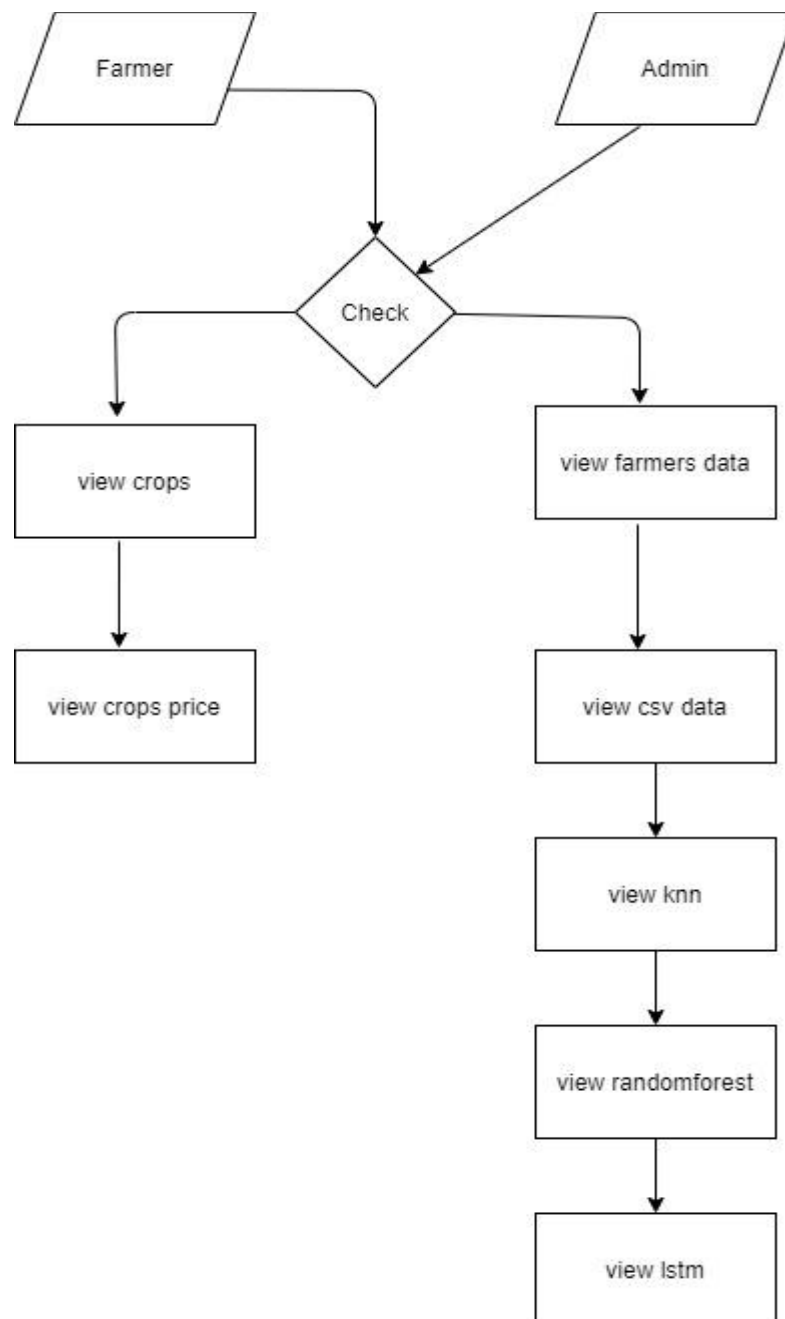


Figure 8.1.2. Data Flow Diagram

8.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

Provide users a ready-to-use, expressive visual modeling Language: In the context of sentiment analysis, this goal could translate to providing a standardized visual representation of the sentiment analysis process. This could involve creating UML diagrams to depict the flow of data and the sentiment analysis algorithm.

Provide extendibility and specialization mechanisms: Sentiment analysis systems may vary in complexity and requirements. UML's extendibility mechanisms could be utilized to tailor the sentiment analysis model to specific domains or to incorporate additional features such as sentiment intensity analysis or aspect-based sentiment analysis.

Be independent of particular programming languages and development process: UML's language-independent nature is beneficial for sentiment analysis, as it allows the modeling of sentiment analysis systems without being tied to a specific programming language or development environment. This facilitates communication and collaboration among stakeholders with diverse technical backgrounds.

Provide a formal basis for understanding the modeling language: UML's formal semantics can aid in precisely defining the components and interactions within a sentiment analysis system. This ensures clarity and consistency in the representation of the system's architecture and behavior.

GOALS:

The Primary goals in the design of the UML are as follows:

Facilitate Data Flow Visualization: UML can be used to visually represent the flow of data within a sentiment analysis system, including the input text data, intermediate processing steps, and the final sentiment analysis results. This goal aims to enhance understanding of how data moves through the system.

Enable Model Interpretability: UML can help in clarifying the structure and logic of sentiment analysis models, making them more interpretable. By representing the components and relationships within the model using standardized diagrams, stakeholders can better understand how sentiment analysis decisions are made.

Support Scalability and Modularity: UML can aid in designing sentiment analysis systems that are scalable and modular, allowing for easy integration of new features or enhancements. By

breaking down the system into smaller, interconnected modules, UML can help identify opportunities for parallel processing and optimization.

8.3 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its Purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

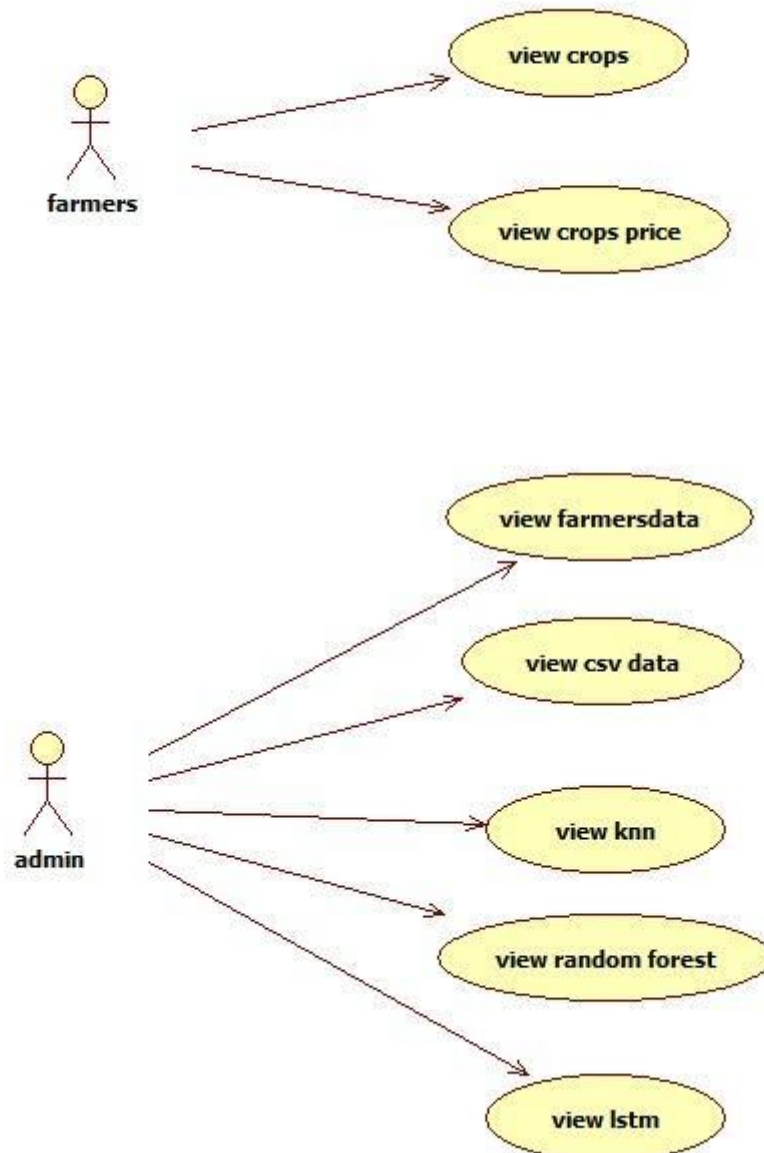


Figure 8.3. Use Case Diagram

8.4 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information

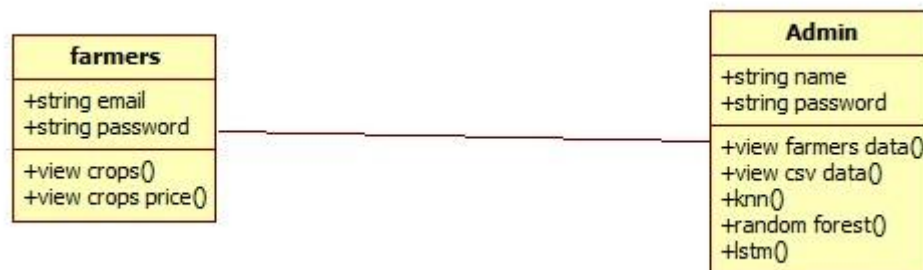


Figure 8.4. Class Diagram

8.5 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

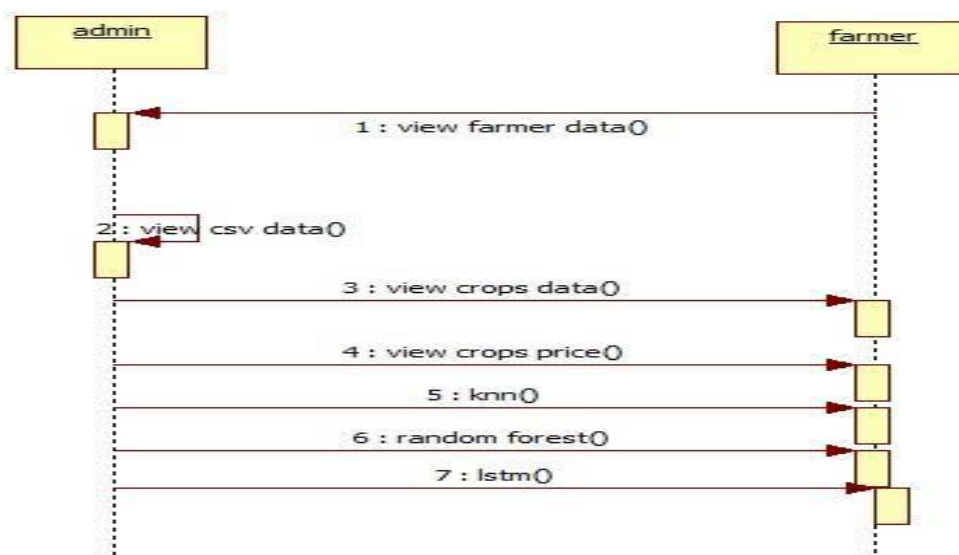


Figure 8.5. Sequence Diagram

8.6 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

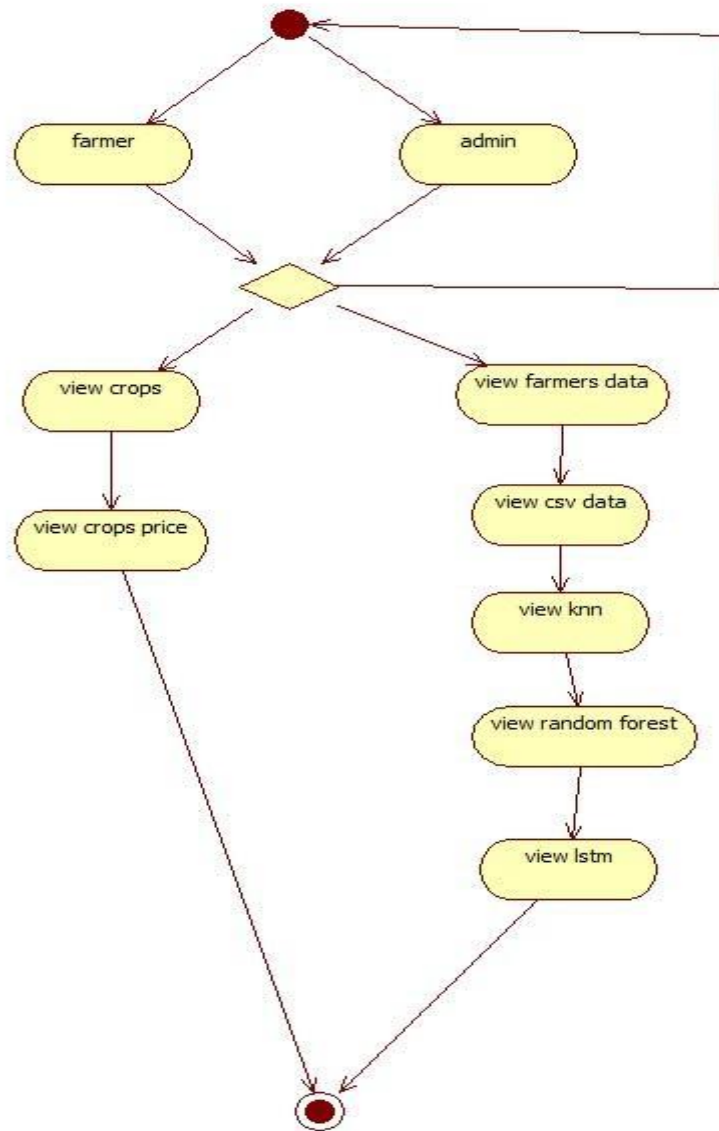


Figure 8.6. Activity Diagram

9. IMPLEMENTATION

9.1 SOURCE CODE:

```
# ignore warning
import warnings
warnings.filterwarnings('ignore')

# data manipulation
import pandas as pd

# data visulization
import seaborn as sns
import matplotlib.pyplot as plt

# text processing
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
import string
from nltk.stem import PorterStemmer

# remove emoji
import emoji

# regular expression
import re

# wordcloud
from wordcloud import WordCloud
from collections import Counter

# model building and evaluation
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
```

```

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score,
confusion_matrix

```

add Codeadd Markdown

Load Data

add Codeadd Markdown

[:]

```

# load data
cols=['tweetid', 'entity', 'target', 'content']

data0 = pd.read_csv("/kaggle/input/twitter-entity-sentiment-
analysis/twitter_training.csv",names=cols)
data1 = pd.read_csv("/kaggle/input/twitter-entity-sentiment-
analysis/twitter_validation.csv",names=cols)

final_df = pd.concat([data0,data1])

```

add Codeadd Markdown

[:]

```
final_df.head()
```

add Codeadd Markdown

[:]

```

# data dimension
final_df.shape

```

add Codeadd Markdown

[:]

```

# info about data
final_df.info()

```

add Codeadd Markdown

[:]

```

# check null values
final_df.isna().sum()

```

add Codeadd Markdown

[:]

```

# drop null values
final_df.dropna(inplace=True)

```

add Codeadd Markdown

[:

```
# check duplicates values
final_df.duplicated().sum()
```

add Codeadd Markdown

[:

```
# drop duplicates
final_df.drop_duplicates(inplace=True)
```

add Codeadd Markdown

[:

```
# value count of target col
final_df['target'].value_counts().plot(kind='pie', autopct='%.2f')
plt.title("per count of each target value")
plt.show()
```

add Codeadd Markdown

[:

```
target_count = final_df['target'].value_counts().reset_index()
target_count
```

add Codeadd Markdown

[:

```
plt.figure(figsize=(15,5))
ax
sns.barplot(data=target_count, x='target', y='count', palette='cubehelix')
for bars in ax.containers:
    ax.bar_label(bars)

plt.title("Count of each target value")
plt.show()
```

add Codeadd Markdown

[:

```
# tweet count of each user
tweet_count
final_df.groupby('tweetid')['target'].count().sort_values(ascending=False).
reset_index()
tweet_count = tweet_count.rename(columns={'target': 'count'})
tweet_count
```

add Codeadd Markdown

feature engineering

add Codeadd Markdown

[]:

```
# char count
final_df['char_count'] = final_df['content'].apply(len)
# word count
final_df['word_count'] = final_df['content'].apply(lambda
x:len(nltk.word_tokenize(x)))
# sentence count
final_df['sent_count'] = final_df['content'].apply(lambda
x:len(nltk.sent_tokenize(x)))
```

add Codeadd Markdown

[]:

```
# dist plot

fig, axes = plt.subplots(1,3,figsize=(18,5))
sns.distplot(ax=axes[0],x=final_df['char_count'],color='b')
axes[0].set_title('char distribution')

sns.distplot(ax=axes[1],x=final_df['word_count'],color='g')
axes[1].set_title('word distribution')

sns.distplot(ax=axes[2],x=final_df['sent_count'],color='r')
axes[2].set_title('sentence distribution')
plt.show()
```

add Codeadd Markdown

[]:

```
# drop unnecessary cols
final_df = final_df.drop(columns=['tweetid','entity'],axis=1)
```

add Codeadd Markdown

[]:

```
final_df.head()
```

add Codeadd Markdown

[]:

```
# remove emojis from tweets
final_df['content'] = final_df['content'].apply(lambda x:
emoji.replace_emoji(x,replace=''))
```

add Codeadd Markdown

Text Preprocessing

- lowercase
- remove punctuation
- remove stopwords
- stemming

add Codeadd Markdown

[:

```
# function for text preprocessing
```

```
ps = PorterStemmer()
```

```
def preprocessing(text):
```

```
    text = text.lower()
```

```
    text = nltk.word_tokenize(text)
```

```
    full_txt = []
```

```
    for i in text:
```

```
        if i not in string.punctuation and i not in stopwords.words('english'):
```

```
            full_txt.append(ps.stem(i))
```

```
    return ' '.join(full_txt)
```

add Codeadd Markdown

[:

```
final_df['content'] = final_df['content'].apply(preprocessing)
```

add Codeadd Markdown

[:

```
final_df.head()
```

add Codeadd Markdown

[:

```
final_df.duplicated().sum()
```

add Codeadd Markdown

[:

```
final_df = final_df.drop_duplicates()
```

add Codeadd Markdown

EDA

add Codeadd Markdown

[:

```
# word cloud for positive tweets
```

```

wc
WordCloud(width=1000,height=700,min_font_size=10,background_color='black')
positive
wc.generate(final_df[final_df['target']=='Positive']['content'].str.cat(sep
=" "))
plt.title('Wordcloud of positive tweet')
plt.axis('off')
plt.imshow(positive)
plt.show()

```

add Codeadd Markdown

[]:

```

# word cloud for negative tweets
wc
WordCloud(width=1000,height=700,min_font_size=10,background_color='black')
negative
wc.generate(final_df[final_df['target']=='Negative']['content'].str.cat(sep
=" "))
plt.title('Wordcloud of Negative tweet')
plt.axis('off')
plt.imshow(negative)
plt.show()

```

add Codeadd Markdown

[]:

```

# word cloud for neutral tweets
wc
WordCloud(width=1000,height=700,min_font_size=10,background_color='black')
neutral
wc.generate(final_df[final_df['target']=='Neutral']['content'].str.cat(sep
=" "))
plt.title('Wordcloud of Neutral tweet')
plt.axis('off')
plt.imshow(neutral)
plt.show()

```

add Codeadd Markdown

[]:

```

# word cloud for irrelevant tweets
wc

```

```

WordCloud(width=1000,height=700,min_font_size=10,background_color='black')
irrelevant =
wc.generate(final_df[final_df['target']=='Irrelevant']['content'].str.cat(sep=" "))
plt.title('Wordcloud of Irrelevant tweet')
plt.axis('off')
plt.imshow(irrelevant)
plt.show()

```

add Codeadd Markdown

[:

```

def pre(text):
    text = re.sub(r'^a-zA-Z\s', '', text).strip()
    return text

```

add Codeadd Markdown

[:

```

final_df['content'] = final_df['content'].apply(pre)

```

add Codeadd Markdown

[:

```

final_df.duplicated().sum()

```

add Codeadd Markdown

[:

```

final_df = final_df.drop_duplicates()

```

add Codeadd Markdown

[:

```

# most common words in positive tweets
positive = []
for txt in final_df[final_df['target']=='Positive']['content'].tolist():
    for word in txt.split():
        positive.append(word)

```

add Codeadd Markdown

[:

```

len(positive)

```

add Codeadd Markdown

[:

```

# plot most 50 common words from positive tweets
plt.figure(figsize=(15,5))
sns.barplot(x=pd.DataFrame(Counter(positive).most_common(50))[0],y=pd.DataF

```

```

rame(Counter(positive).most_common(50))[1],palette='rainbow')
plt.xlabel('word')
plt.ylabel('word count')
plt.title('Most common 50 words in positive tweet')
plt.xticks(rotation=90)
plt.show()

```

add Codeadd Markdown

[]:

```

# most common words in negative tweets
negative = []
for txt in final_df[final_df['target']=='Negative']['content'].tolist():
    for word in txt.split():
        negative.append(word)

```

add Codeadd Markdown

[]:

```

len(negative)

```

add Codeadd Markdown

[]:

```

# plot most 50 common words from negative tweets
plt.figure(figsize=(15,5))
sns.barplot(x=pd.DataFrame(Counter(negative).most_common(50))[0],y=pd.DataFrame(Counter(negative).most_common(50))[1],palette='rainbow')
plt.xlabel('word')
plt.ylabel('word count')
plt.title('Most common 50 words in negative tweet')
plt.xticks(rotation=90)
plt.show()

```

add Codeadd Markdown

[]:

```

# most common words in neutral tweets
neutral = []
for txt in final_df[final_df['target']=='Neutral']['content'].tolist():
    for word in txt.split():
        neutral.append(word)

```

add Codeadd Markdown

[]:

```

len(neutral)

```


add Codeadd Markdown

[:

```
# plot most 50 common words from neutral tweets
plt.figure(figsize=(15,5))
sns.barplot(x=pd.DataFrame(Counter(neutral).most_common(50))[0],y=pd.DataFrame(Counter(neutral).most_common(50))[1],palette='rainbow')
plt.xlabel('word')
plt.ylabel('word count')
plt.title('Most common 50 words in neutral tweet')
plt.xticks(rotation=90)
plt.show()
```

add Codeadd Markdown

[:

```
# most common words in irrelevant tweets
irrelevant = []
for txt in final_df[final_df['target']=='Irrelevant']['content'].tolist():
    for word in txt.split():
        irrelevant.append(word)
```

add Codeadd Markdown

[:

```
len(irrelevant)
```

add Codeadd Markdown

[:

```
# plot most 50 common words from irrelevant tweets
plt.figure(figsize=(15,5))
sns.barplot(x=pd.DataFrame(Counter(irrelevant).most_common(50))[0],y=pd.DataFrame(Counter(irrelevant).most_common(50))[1],palette='rainbow')
plt.xlabel('word')
plt.ylabel('word count')
plt.title('Most common 50 words in irrelevant tweet')
plt.xticks(rotation=90)
plt.show()
```

add Codeadd Markdown

Label Encoding

add Codeadd Markdown

[:

```
# Positive - 1
```

```
# Negative - 0
# Neutral - 2
# Irrelevant - 3

final_df['sentiment'] =
final_df['target'].replace({'Positive':1,'Negative':0,'Neutral':2,'Irrelevant':3})
```

add Codeadd Markdown

Extract Input and Target data

add Codeadd Markdown

[]:

```
X = final_df['content']
y = final_df['sentiment']
```

add Codeadd Markdown

Splitting data

add Codeadd Markdown

[]:

```
# split the data
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=42)
```

add Codeadd Markdown

Pipeline

add Codeadd Markdown

[]:

```
# step-1 convert text data into numeric
# step-2 apply RandomForestClassifier

sentiment_pipeline = Pipeline([
    ('tfidf',TfidfVectorizer()),
    ('rfc',RandomForestClassifier(random_state=42))
])
```

add Codeadd Markdown

[]:

```
# fit the data into pipeline
sentiment_pipeline.fit(X_train,y_train)
```

add Codeadd Markdown

Model Evaluation

add Codeadd Markdown

[:

```
y_pred = sentiment_pipeline.predict(X_test)
print(accuracy_score(y_test,y_pred))
```

add Codeadd Markdown

[:

```
# Positive - 1
# Negative - 0
# Neutral - 2
# Irrelevant - 3

label = ['Negative','Positive','Neutral','Irrelevant']
sns.heatmap(confusion_matrix(y_test,y_pred),xticklabels=label,yticklabels=label,annot=True,fmt='d',cmap='crest')
plt.title('Actual vs Predicted')
plt.show()
```

add Codeadd Markdown

[:

```
mnb_pipeline = Pipeline([
    ('tfidf',TfidfVectorizer()),
    ('mnb',MultinomialNB())
])
```

add Codeadd Markdown

[:

```
mnb_pipeline.fit(X_train,y_train)
```

add Codeadd Markdown

[:

```
mnb_pred = mnb_pipeline.predict(X_test)
print(accuracy_score(y_test,mnb_pred))
```

add Codeadd Markdown

[:

```
label = ['Negative','Positive','Neutral','Irrelevant']
sns.heatmap(confusion_matrix(y_test,mnb_pred),xticklabels=label,yticklabels=label,annot=True,fmt='d',cmap='crest')
plt.title('Actual vs Predicted')
plt.show()
```

add Codeadd Markdown

[]:

```
lr_pipeline = Pipeline([
    ('tfidf',TfidfVectorizer()),
    ('lr',LogisticRegression(penalty=None,solver='sag',max_iter=500))
])
```

add Codeadd Markdown

[]:

```
lr_pipeline.fit(X_train,y_train)
```

add Codeadd Markdown

[]:

```
lr_pred = lr_pipeline.predict(X_test)
print(accuracy_score(y_test,lr_pred))
```

add Codeadd Markdown

[]:

```
label = ['Negative','Positive','Neutral','Irrelevant']

sns.heatmap(confusion_matrix(y_test,lr_pred),xticklabels=label,yticklabels=
label,annot=True,fmt='d',cmap='crest')
plt.show()
```

add Codeadd Markdown

[]:

```
import pickle
pickle.dump(sentiment_pipeline,open('rfc_sentiment_model','wb'))
```

10. INPUT AND OUTPUT DESIGN

10.1 INPUT DESIGN

Input design is crucial in ensuring that the sentiment analysis system efficiently processes user input and provides accurate results. It involves specifying procedures for data preparation and determining the methods for inputting data into the system. The design focuses on controlling the amount of input required, minimizing errors, avoiding delays, and simplifying the process for users. Key considerations in input design for sentiment analysis include:

- **Data Specification:** Determining what data should be provided as input to the system, such as text data from social media posts, customer reviews, or other sources.
- **Data Arrangement and Coding:** Structuring the input data in a format suitable for

processing, which may involve tokenization, normalization, and encoding techniques.

- **User Interface Design:** Creating user-friendly screens and interfaces for data entry, ensuring ease of use and efficiency in inputting data.
- **Input Validation:** Implementing validation checks to ensure the accuracy and integrity of the input data, including checks for data format, range, and consistency.
- **Error Handling:** Developing procedures for handling errors during data input, including providing informative error messages and guiding users through error resolution steps.

10.2 OBJECTIVES

1. **Accuracy:** Input design aims to prevent errors in the data input process, ensuring that the system receives correct and reliable input from users.
2. **Efficiency:** Creating user-friendly input screens and interfaces helps to streamline the data entry process, making it easier for users to input large volumes of data quickly and accurately.
3. **Validation:** Implementing validation checks ensures that the input data meets specified criteria for accuracy, completeness, and consistency, improving the quality of the data processed by the system.

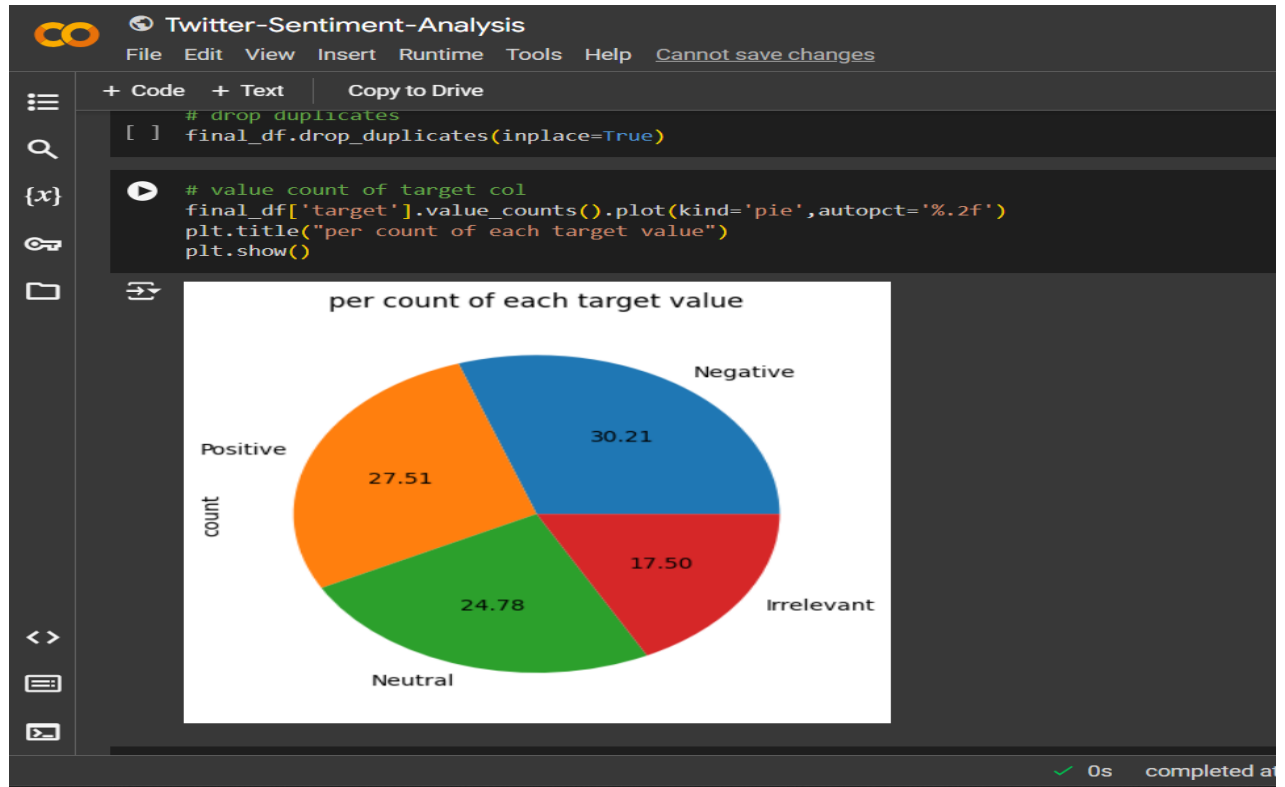
10.3 OUTPUT DESIGN

Output design is essential for presenting the results of sentiment analysis in a clear and meaningful manner to users and other systems. It involves determining how information is displayed for immediate use and in hard copy format. The objectives of output design for sentiment analysis include:

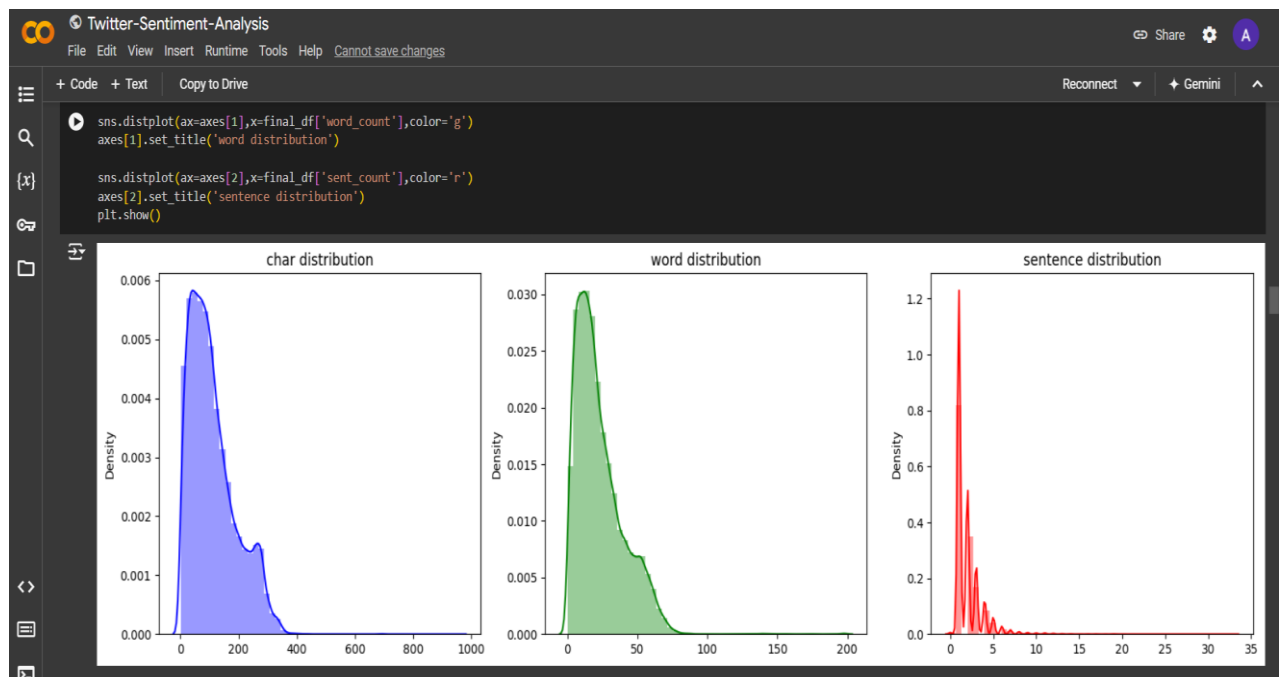
1. **Clarity:** Designing outputs that effectively communicate the results of sentiment analysis, ensuring that users can easily understand and interpret the information presented.
2. **Relevance:** Presenting information relevant to the user's needs and decision-making processes, such as sentiment trends, sentiment scores, and sentiment distributions.
3. **Actionability:** Providing outputs that signal important events, opportunities, problems, or warnings, enabling users to take appropriate actions based on the sentiment analysis results.
4. **Confirmation:** Generating outputs that confirm actions taken or decisions made based on sentiment analysis, providing users with feedback on the effectiveness of their actions.

11. SCREENSHOTS

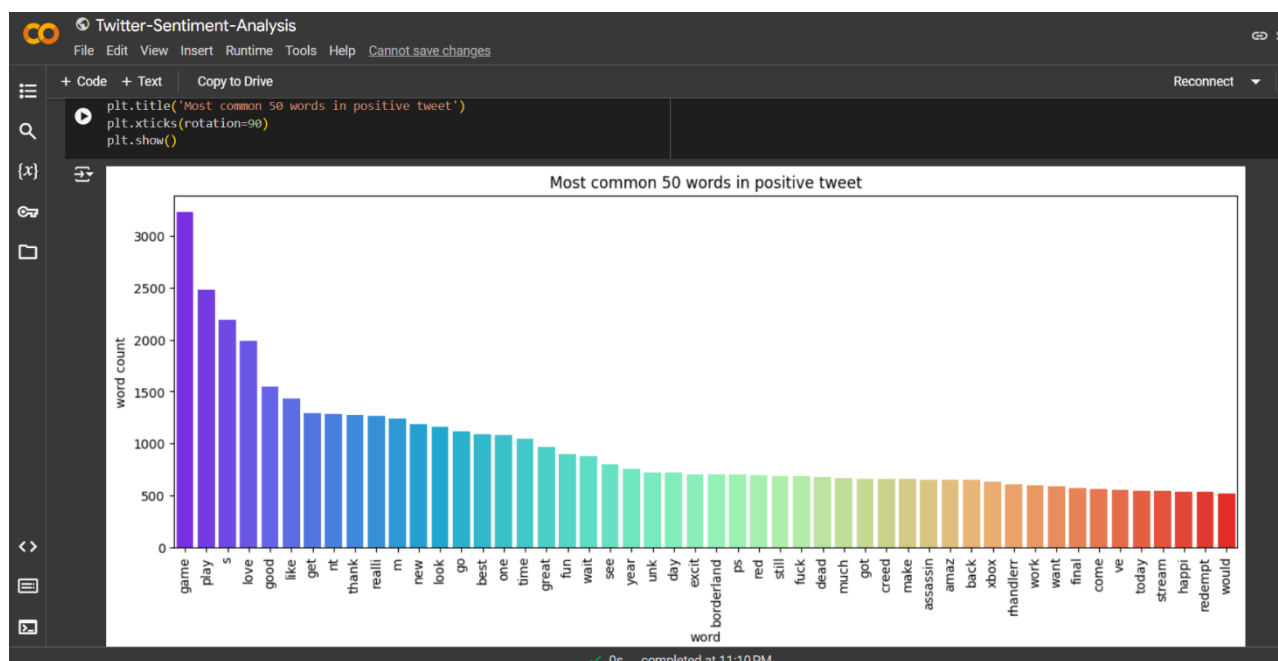
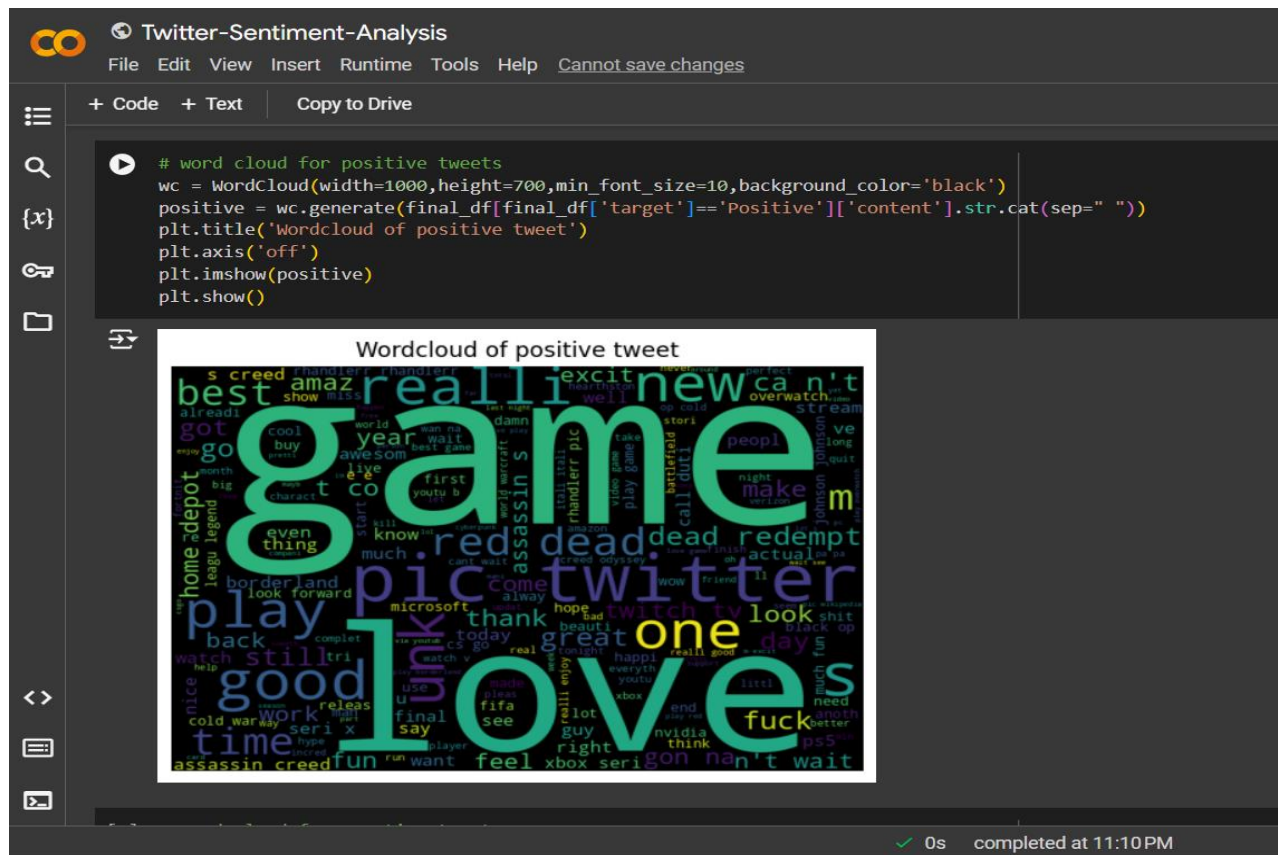
Screenshots:



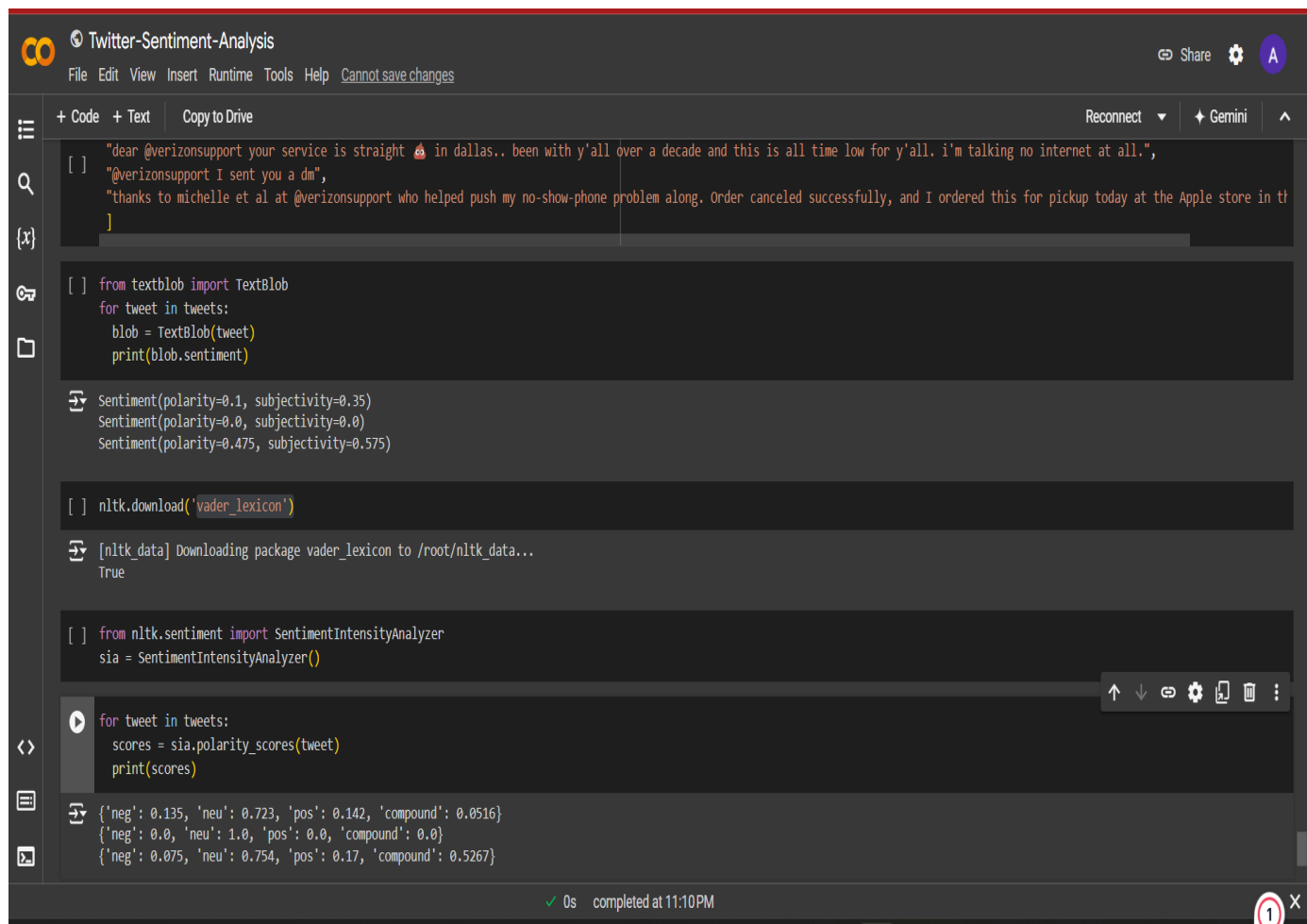
11.1 Distribution:



11.2 Tokenization:



Output:



The screenshot shows a Jupyter Notebook titled "Twitter-Sentiment-Analysis". The interface includes a top bar with a file menu (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar indicating "Cannot save changes". The notebook has three tabs: "+ Code", "+ Text", and "Copy to Drive". The left sidebar contains icons for file management, search, and execution. The main area displays the following code and output:

```
[ ] "dear @verizonsupport your service is straight 🤬 in dallas.. been with y'all over a decade and this is all time low for y'all. i'm talking no internet at all.",
    "@verizonsupport I sent you a dm",
    "thanks to michelle et al at @verizonsupport who helped push my no-show-phone problem along. Order canceled successfully, and I ordered this for pickup today at the Apple store in t"
]
```

```
[ ] from textblob import TextBlob
    for tweet in tweets:
        blob = TextBlob(tweet)
        print(blob.sentiment)
```

```
Sentiment(polarity=0.1, subjectivity=0.35)
Sentiment(polarity=0.0, subjectivity=0.0)
Sentiment(polarity=0.475, subjectivity=0.575)
```

```
[ ] nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

```
[ ] from nltk.sentiment import SentimentIntensityAnalyzer
    sia = SentimentIntensityAnalyzer()
```

```
for tweet in tweets:
    scores = sia.polarity_scores(tweet)
    print(scores)
```

```
{'neg': 0.135, 'neu': 0.723, 'pos': 0.142, 'compound': 0.0516}
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
{'neg': 0.075, 'neu': 0.754, 'pos': 0.17, 'compound': 0.5267}
```

The bottom status bar shows a green checkmark, "0s", and "completed at 11:10 PM". A red circle with the number "1" is visible in the bottom right corner.

12.SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

12.1 TYPES OF TESTS

12.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

12.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.

Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

12.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

12.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results.

An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

12.1.5 White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

12.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

12.1.7 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

12.2 TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

12.3 TEST OBJECTIVES

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

12.4 FEATURES TO BE TESTED

- Verify that the entries are of the correct format

- No duplicate entries should be allowed
- All links should take the user to the correct page.

12.5 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

12.6 TEST RESULTS

All the test cases mentioned above passed successfully. No defects encountered.

12.7 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

12.8 TEST RESULTS

All the test cases mentioned above passed successfully. No defects encountered.

12.9 SAMPLE TEST CASES:

S.No	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	Farmer Register	If User registration successfully.	Pass	If an already user email exists then it fails.
2.	Farmer Login	If the Username and password is correct then it will be a valid page.	Pass	Un Register Users will not log in.
3.	crop details	after login farmer will get the list of crop details	Pass	we can't get the crop details.
4.	crop price	we can get previous year crop prices for particular crops..	Pass	we can't get crop price details..
5.	Admin login	Admin can login with his login credential. If success he get his home page	Pass	Invalid login details will not allowed here

13.CONCLUSION

In conclusion, the implementation of sentiment analysis tools like PECAD (Predictive Economic Crop Analysis and Decision-making) presents significant challenges and opportunities in addressing the needs of non-profit agencies working with indebted farmers. While PECAD shows promise in predicting future crop prices accurately, there are several key challenges that need to be addressed for successful deployment and adoption.

Firstly, enhancing PECAD's predictive performance by incorporating historical weather patterns could improve its accuracy in determining future crop supply and prices. However, integrating physical weather prediction models with PECAD poses technical challenges that need to be addressed in future iterations.

Additionally, the adoption of sophisticated deep learning approaches like PECAD may face resistance among low-literate farmers due to concerns and suspicions. Public awareness campaigns and education initiatives within the agencies working with such programs can help alleviate these fears and encourage participation.

Moreover, the resource constraints faced by non-profit agencies, particularly in acquiring sophisticated computer hardware for training and running PECAD, necessitate a pragmatic approach. Deploying PECAD as a stand-alone web service that agencies can access without significant investment in hardware can facilitate its adoption and usage.

Furthermore, while PECAD represents a valuable tool in addressing farmer suicides, it is only one piece of the puzzle. Success depends on the availability of long-term crop pricing and volume data, which may not be readily accessible in all developing countries. Efforts to establish analogous data repositories and collaborations with relevant stakeholders are essential for the sustainability and scalability of PECAD.

In summary, PECAD offers a promising solution for predicting future produce prices based on past data patterns. Its innovative wide and deep learning architecture demonstrates superior performance compared to existing methods. Collaborations with non-profit agencies, ongoing reviews, and potential deployments underscore the significance of PECAD in addressing the challenges faced by indebted farmers and preventing farmer suicides.

14.BIBILOGRAPHY

- [1]S. ChandraKala, and C. Sindhu, “Opinion mining and sentiment classification: A survey”, ICTACT journal on soft computing, vol. 3, No. 1, pp. 420-425, October 2012.
- [2] B. Pang, and L. Lee, “Opinion mining and sentiment analysis”, Foundations and Trends® in Information Retrieval, Vol. 2, No. (1–2), pp. 1-135, July 2008, doi:10.1561/15000000011.
- [3] E. Aydoğan, and M.A. Akcayol, “A comprehensive survey for sentiment analysis tasks using machine learning techniques”, In Proceedings of 2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA) , Sinaia, Romania, pp. 1-7, August 2016.
- [4] B. Agarwal, N. Mittal, P. Bansal, and S. Garg, “Sentiment analysis using common-sense and context information”, Computational intelligence and neuroscience, Vol. 2015, March 2015, doi:10.1155/2015/715730.
- [5] <https://www.kaggle.com/code/gauravbosamiya/twitter-sentiment-analysis>