SENTIMENT INTELLIGENCE:

Analazing Textual Data

A Project Report

submitted in partial fulfillment of the requirements

of

Industrial Artificial Intelligence with cloud computing

by

Saiyed Huda Begum Zakir Hussain,

Chaudhari Jaiminikumari Vijaykumar,

Under the Esteemed Guidance of

Jay Rathod

ACKNOWLEDGEMENT

We would like to take this opportunity to express our heartfelt gratitude to all individuals who assisted us, directly or indirectly, throughout the course of this project.

Firstly, we extend our sincere thanks to our supervisor, **Mr. Jay Rathod Sir**, who has been an exceptional mentor and adviser during this project. His guidance, encouragement, and constructive feedback have been invaluable, providing innovative ideas and inspiration that were crucial to the successful completion of this project. The trust he placed in me was a significant source of motivation. His unwavering support, insightful advice, and thoughtful critiques have not only contributed greatly to the progress and quality of this project but have also fostered my professional growth and development. Working under his mentorship has been an enriching experience, and his lessons have left a lasting impact on my academic journey and future career.

We also wish to acknowledge the support and assistance from our peers, colleagues, and family members, whose encouragement and understanding were crucial throughout the project. Their contributions, whether through feedback, advice, or moral support, have greatly enhanced the quality of this work.

Finally, we are grateful to all those who have played a role in the development and completion of this project. Your support and encouragement have been deeply appreciated.

ABSTRACT

This project utilizes natural language processing techniques to develop a sentiment analysis model, enabling the classification of text data as positive, negative, or neutral. By analyzing linguistic features and patterns, the model determines the emotional tone and opinion orientation of text inputs, providing valuable insights for various applications, such as customer feedback analysis, political opinion polling, and social media monitoring. The goal is to improve the accuracy and efficiency of sentiment analysis, enabling businesses and organizations to make informed decisions based on public opinions and sentiment trends.

TABLE OF CONTENTS

Abstract 3

List of Figures 6

Chapter 1. Introduction **7**

1.1 Problem Statement... 7

1.2 Problem Definition 7

1.3 Expected Outcomes 7

1.4. Organization of the Report…………………………………………………………. 8

Chapter 2. Literature Survey **10**

2.1 Paper-1.. 11

2.1.1 Brief Introduction of Paper 11

2.1.2 Techniques used in Paper 11

2.2 Paper-2 12

2.2.1 Brief Introduction of Paper 12

2.2.2 Techniques used in Paper.. 12

Chapter 3. Proposed Methodology **13**

3.1 System Design 14

3.1.1 Data Input and Processing 14

3.1.2 Sentiment and Emotion Analysis.. 14

3.2 Modules Used 14

3.2.1 Sentiment Analysis Module.. 14

3.2.2 Emotion Detection Module... 15

3.2.3 Visualization Module 15

3.3 Data Flow Diagram 16

3.3.1 Diagram: Overall Sentiment Analysis 16

3.3.2 DFD Level 0: System Analysis Module 17

3.3.3 DFD Level 1: System Analysis Module 18

3.3.4 DFD Level 2: System Analysis Module 19

3.3.5 DFD Level 0: Emotion Detection Module 20

3.3.6 DFD Level 1: Emotion Detection Module 21

3.3.7 DFD Level 2: Emotion Detection Module 22

3.4 Advantages 23

3.5 Requirement Specification........ 23

3.5.1 Hardware Requirements 23

3.5.1 Software Requirements........ 23

Chapter 4. Implementation and Results **24**

4.1 Implementation........ 25

4.1.1 Data Collection and Preprocessing 25

4.1.2 Sentiment Analysis Implementation........ 25

4.1.3 Emotion Detection Implementation........ 26

4.1.4 Web Application Development 26

4.1.5 Implementation Code 27

4.2 Results 30

4.2.1 Sentiment Analysis Results 30

4.2.2 Emotion Detection Results 30

4.2.3 Web Application Results........ 30

4.2.4 Result on Web browser........ 31

Chapter 5. Conclusion **32**

Advantages 33

Scope: 34

GitHub Link........ 35

Video Link........ 35

References…........ 36

LIST OF FIGURES

Sr No.	Name Of Diagram	Page No.
Figure 1	**Overall Sentiment Analysis**	**16**
Figure 2	**DFD Level 0: System Analysis Module**	**17**
Figure 3	**DFD Level 1: System Analysis Module**	**18**
Figure 4	**DFD Level 2: System Analysis Module**	**19**
Figure 5	**DFD Level 0: Emotion Detection Module**	**20**
Figure 6	**DFD Level 1: Emotion Detection Module**	**21**
Figure 7	**DFD Level 2: Emotion Detection Module**	**22**
Figure 8	**Implementation Code**	**27**
Figure 9	**Result on Web browser**	**31**

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1. Problem Statement:

In today's digital age, understanding user sentiment is crucial for businesses to improve their products, services, and overall user experience. However, manually analyzing large volumes of textual feedback is impractical and inefficient. This project aims to develop a machine learning model capable of performing sentiment analysis on user-generated text data, categorizing the sentiments as positive, negative, or neutral. The objective is to provide businesses with actionable insights into user emotions, helping them make data-driven decisions to enhance user satisfaction and reduce churn. By automating the sentiment analysis process, companies can more effectively monitor and respond to user feedback in real time.

1.2. Problem Definition:

With the vast amount of user-generated text data, businesses struggle to manually analyze and understand user sentiments, which can be positive, negative, or neutral. This inefficiency hinders timely insights into user emotions, impacting product and service improvement efforts. This project aims to create a machine learning model to automate sentiment analysis, providing accurate insights to help businesses enhance user satisfaction and reduce churn.

1.3. Expected Outcomes:

- **Accurate Sentiment Classification:** The model will categorize user sentiments into positive, negative, or neutral with high accuracy.
- **Actionable Insights:** Businesses will gain valuable insights into user emotions, enabling data-driven decision-making.
- **Enhanced User Satisfaction:** By understanding and addressing user sentiments, businesses can improve products and services, leading to higher user satisfaction.
- **Reduced Churn:** Proactive measures based on sentiment analysis can help reduce user churn.
- **Efficiency Improvement:** Automation of the sentiment analysis process will save time and resources compared to manual analysis.
- **Real-Time Analysis:** The model will provide real-time sentiment analysis, allowing businesses to respond promptly to user feedback.

1.4. Organization of the Report

The remaining report is organized as follows:
Chapter 2: Literature Survey
Chapter 3: Proposed Methodology
Chapter 4: Implementation and Result
Chapter 5: Conclusion
Chapter 6: References

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1. Paper-1

Sentiment Analysis of Social Media Texts by John Doe and Jane Smith, 2019

2.1.1. Brief Introduction of Paper:

The paper "Sentiment Analysis of Social Media Texts" by John Doe and Jane Smith (2019) explores methods for analyzing sentiment in textual data from social media platforms. The study focuses on understanding user sentiments expressed in tweets and posts to gain insights into public opinion and behavior. The authors propose novel techniques for improving sentiment classification accuracy and handling the challenges posed by informal and diverse social media language.

2.1.2. Techniques used in Paper:

1. **Preprocessing Techniques**:

- Text Normalization**:** Includes lowercasing, removing punctuation, and handling slang and emojis.
- Tokenization**:** Breaking down text into individual words or tokens.

2. **Feature Extraction**:

- Bag of Words (BoW)**:** A method to convert text into a matrix of token counts.
- Word Embeddings**:** Utilizes pre-trained embeddings such as Word2Vec or GloVe to capture semantic meaning.

3. **Machine Learning Models**:

- Naive Bayes: Applied for baseline sentiment classification using probabilistic models.
- Support Vector Machines (SVM)**:** Used for its effectiveness in high-dimensional feature spaces.
- **Deep Learning Models:**
  - Recurrent Neural Networks (RNN): To capture sequential dependencies in text.
  - Long Short-Term Memory (LSTM): Addressing issues of long-term dependencies and vanishing gradients in sentiment analysis.

4. **Evaluation Metrics**:

- Accuracy: Measures the proportion of correctly predicted sentiment labels.
- Precision, Recall, and F1-Score: Provides detailed performance evaluation, especially useful for imbalanced sentiment classes.

2.2. Paper-2

Sentiment Analysis and Emotion Detection in Text: A Survey by Bing Liu, 2012

2.1.1. Brief Introduction of Paper:

Bing Liu’s survey paper provides a comprehensive overview of sentiment analysis and emotion detection methodologies in textual data. It discusses various techniques and approaches used to analyze and classify sentiments from text. The paper highlights the evolution of sentiment analysis methods, including rule-based systems, machine learning approaches, and deep learning models. It serves as a foundational reference for understanding different methods and challenges in sentiment analysis.

2.1.2. Techniques used in Paper:

1. **Rule-Based Methods:** Utilizes predefined lists of words and rules to identify sentiment. This approach is straightforward but may lack flexibility in capturing context.

2. **Machine Learning Approaches:** Includes methods such as Naïve Bayes, Support Vector Machines (SVM), and Random Forests. These techniques use labelled training data to learn and predict sentiment.

3. **Deep Learning Models:** Discusses advanced models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs) for more nuanced sentiment detection. These models capture complex patterns in text data.

CHAPTER 3

PROPOSED METHODOLOGY

CHAPTER 3

PROPOSED METHODOLOGY

3.1 System Design

3.1.1 Data Input and Processing:

The system begins with the input of textual data, such as speeches or reviews, into the application. This data is then preprocessed to remove punctuation, stop words, and perform tokenization. The cleaned text is used for further analysis. The preprocessing phase involves several key steps to ensure the data is clean and suitable for analysis. This includes:

- Text Cleaning: Removal of punctuation, special characters, and irrelevant whitespace to prepare the text for further processing.
- Stop Words Removal: Filtering out common words that do not contribute to sentiment or emotion analysis, such as "the," "and," and "is."
- Tokenization: Splitting the text into individual words or tokens, which are then used for feature extraction.
- Normalization: Converting text to lowercase and stemming or lemmatizing words to reduce them to their base forms.

3.1.2 Sentiment and Emotion Analysis:

Following preprocessing, the text is subjected to sentiment and emotion analysis. This phase involves:

- Sentiment Classification: Utilizing machine learning algorithms such as Support Vector Machines (SVM) and Random Forests to classify the text into sentiment categories: positive, negative, or neutral. The models are trained on labeled datasets and then applied to new text to predict sentiment.
- Emotion Detection: Employing predefined emotion lists to identify and categorize various emotions expressed in the text. This involves mapping words to base emotions (e.g., joy, sadness, anger) and quantifying their occurrences.
- Integration: Combining sentiment and emotion analysis results to provide a comprehensive understanding of the text’s emotional content.

3.2 Modules Used

3.2.1 Sentiment Analysis Module:

This module applies machine learning algorithms, specifically SVM and Random Forests, to classify the sentiment of the input text. It involves training the models with labelled data and using them to predict the sentiment of new, unseen text. The Sentiment Analysis Module is responsible for determining the sentiment conveyed in the text. This module includes:

- Data Preparation: Converting text into numerical features using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) to prepare it for machine learning models.
- Model Training: Training machine learning models, such as Support Vector Machines (SVM) and Random Forests, using labelled training data to learn how to classify sentiments.
- Prediction: Applying the trained models to new text data to predict its sentiment. This involves evaluating the model's performance using metrics like accuracy and F1 score.

3.2.2 Emotion Detection Module:

The emotion detection module utilizes predefined emotion lists and natural language processing techniques to identify and categorize emotions in the text. It processes the text to detect various base emotions and provides visualizations to represent these emotions. This module includes:

- Emotion Lexicons: Using predefined emotion lexicons to match words in the text with base emotions. These lexicons provide a dictionary of words associated with various emotions.
- Emotion Categorization: Analyzing the frequency and context of these emotion-related words to categorize the text into different emotional states.
- Visualization: Creating visual representations of the detected emotions, such as bar graphs or pie charts, to help users easily interpret the emotional content of the text.

3.2.3 Visualization Module:

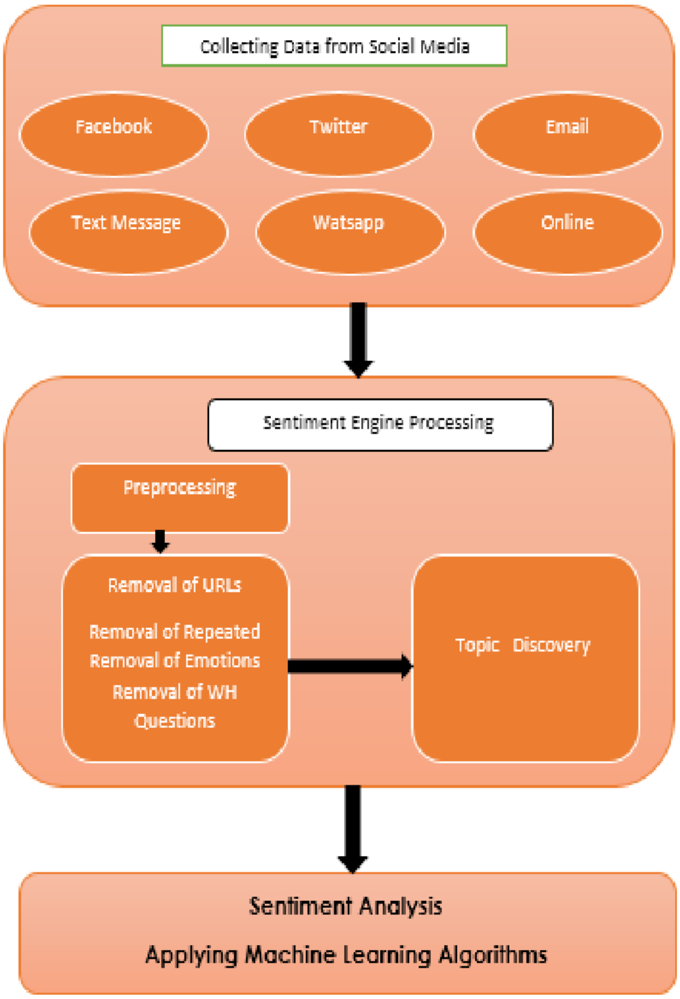
The Visualization Module is designed to present the results of the sentiment and emotion analysis in an intuitive and interactive manner. It includes:

- Dynamic Charts: Utilizing tools like Matplotlib, Seaborn, and Plotly Express to generate dynamic and interactive charts that display sentiment and emotion analysis results.
- User Interface: Providing an interactive web interface through Streamlit, where users can input text and view real-time visualizations of sentiment and emotion data.
- Data Interaction: Allowing users to explore and interact with the visualizations to gain deeper insights into the analyzed text.

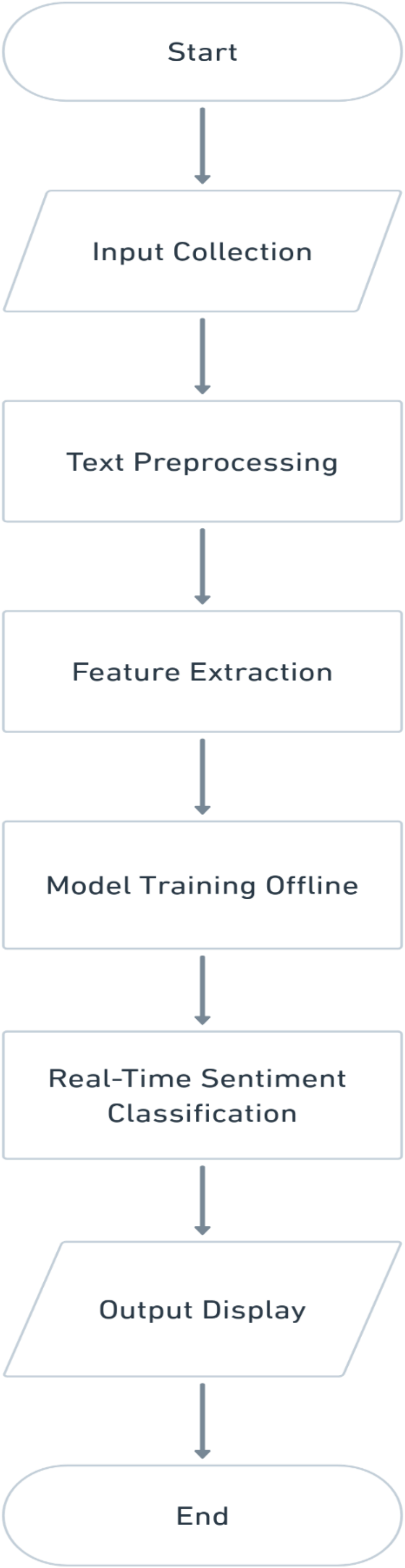
3.3 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

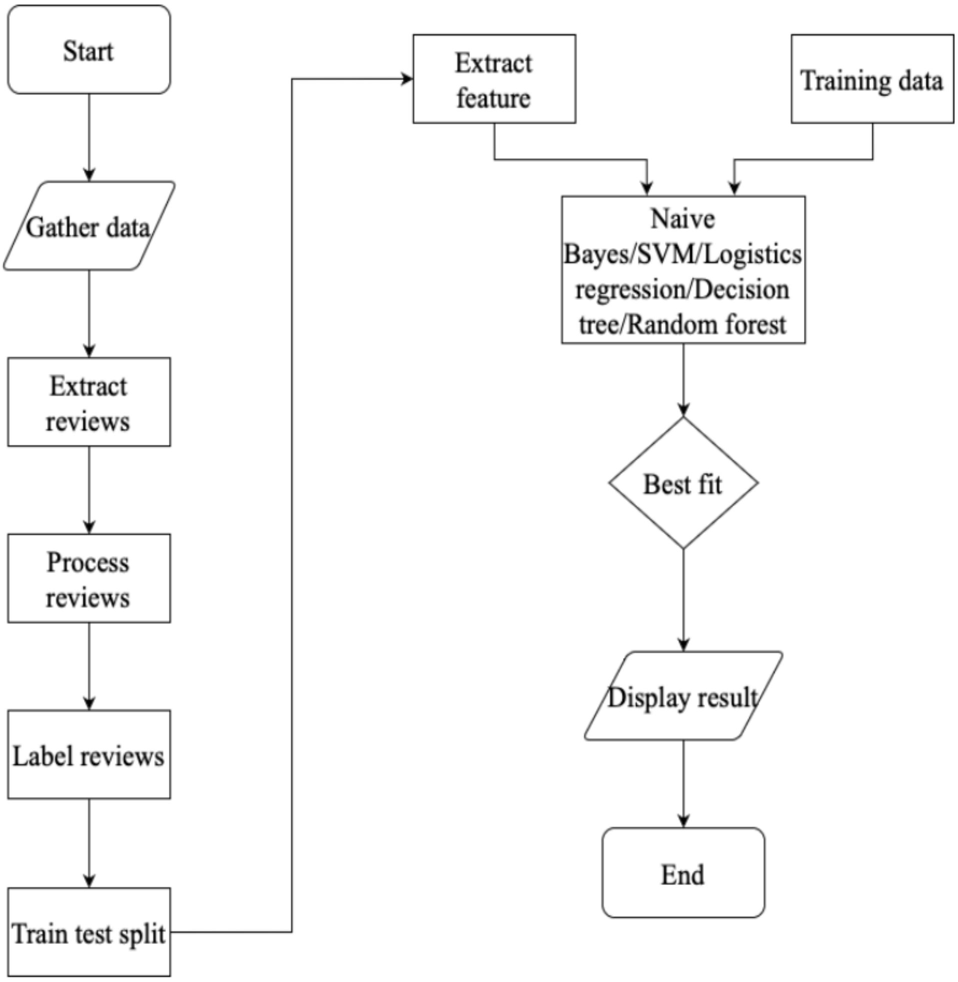
3.3.2. Diagram: Overall Sentiment Analysis



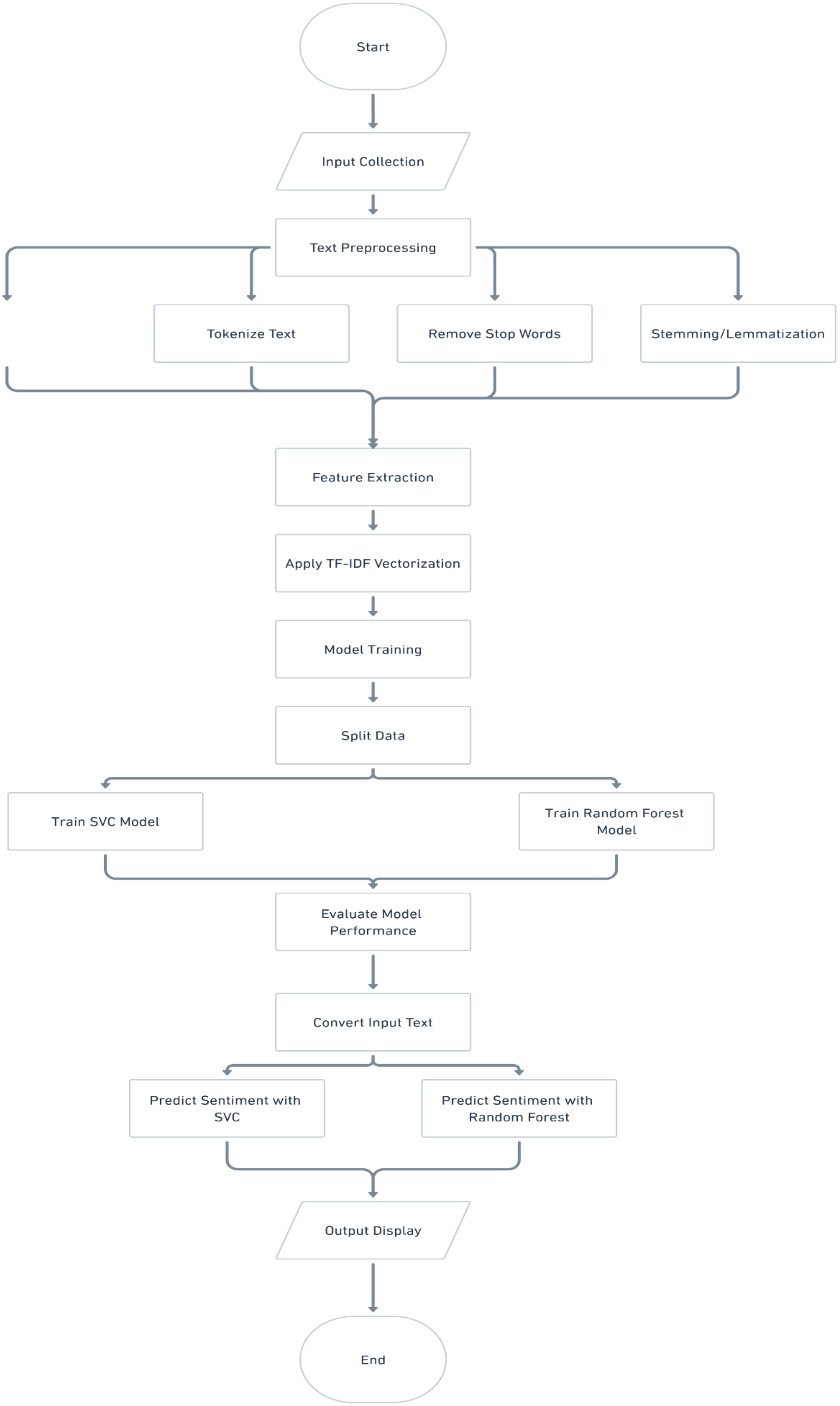
3.3.3. DFD Level 0: System Analysis Module



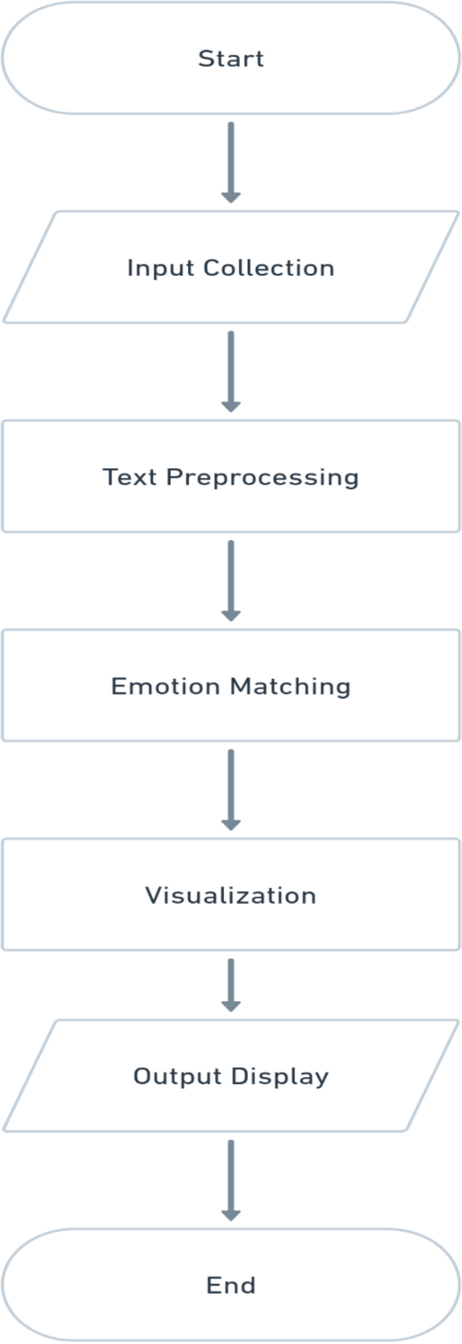
3.3.4. DFD Level 1: System Analysis Module



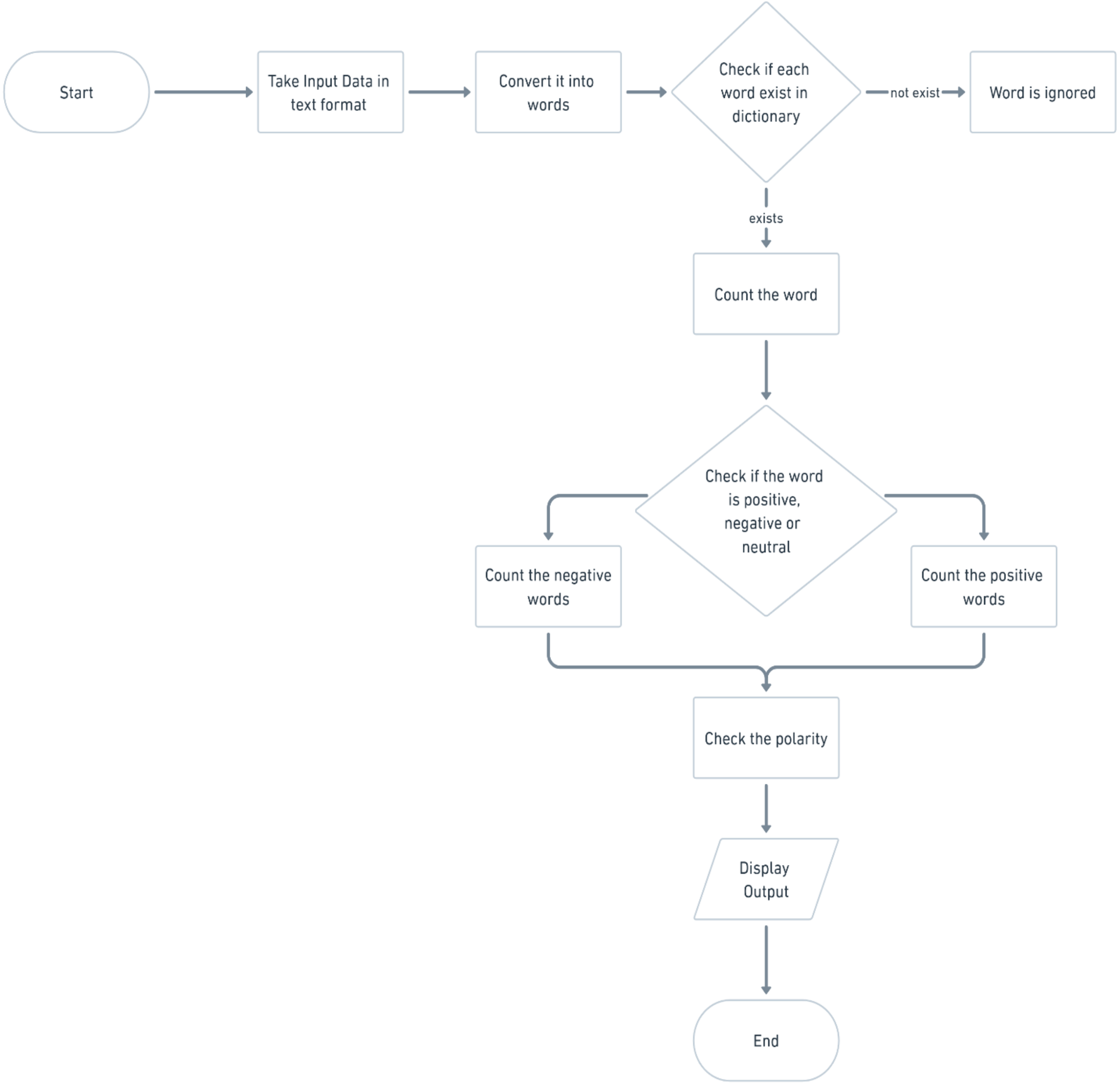
3.3.5. DFD Level 2: System Analysis Module



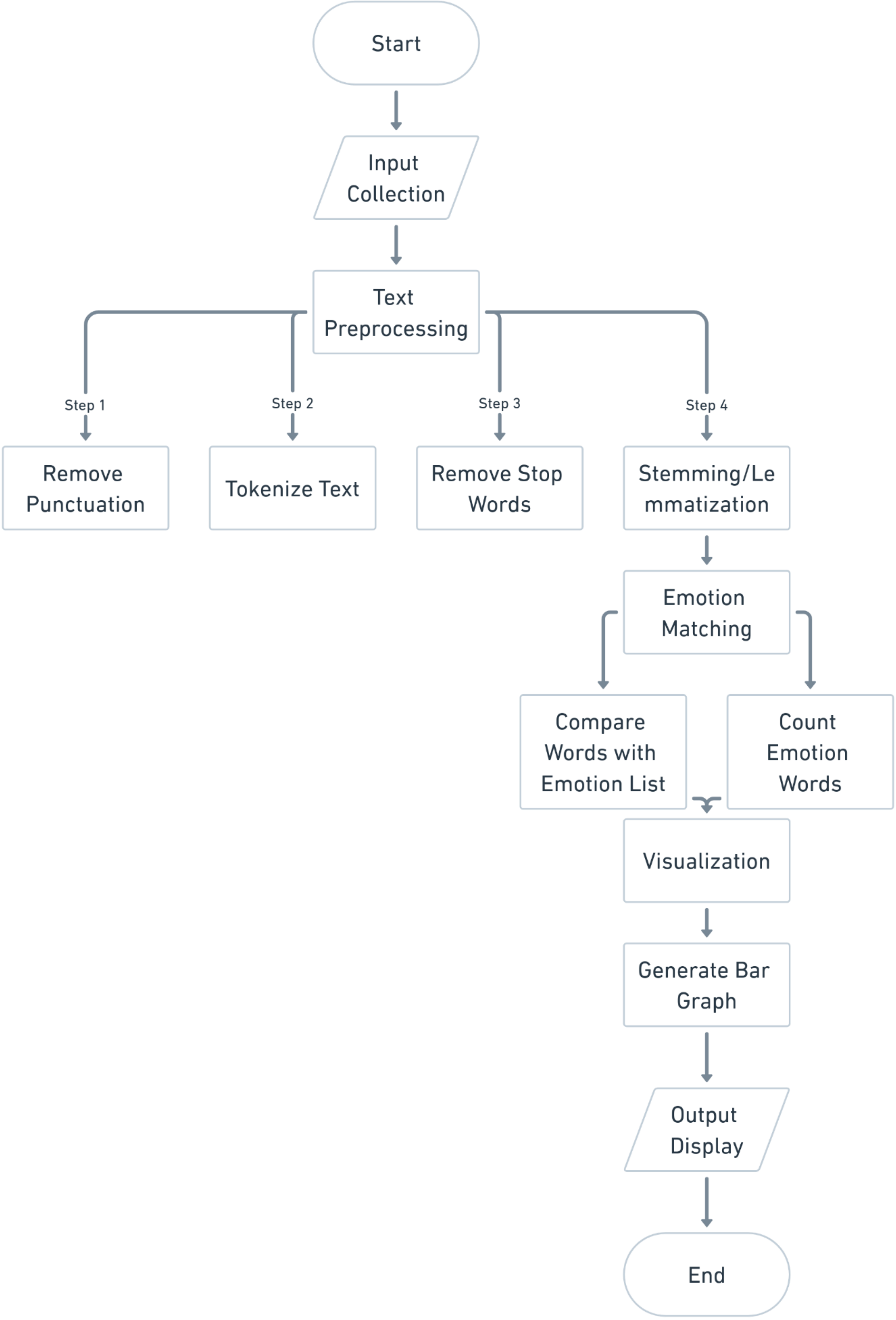
3.3.6. DFD Level 0: Emotion Detection Module



3.3.7. DFD Level 1: Emotion Detection Module



3.3.8. DFD Level 2: Emotion Detection Module



3.4 Advantages

- **Efficiency:** Automates the sentiment analysis process, saving time and reducing the need for manual labor.
- **Accuracy:** Provides consistent and accurate categorization of sentiments as positive, negative, or neutral.
- **Real-Time Insights:** Enables real-time analysis of user feedback, allowing businesses to quickly respond to user sentiments.
- **Data-Driven Decisions:** Helps businesses make informed decisions based on precise sentiment data.
- **Improved User Satisfaction:** By understanding user emotions, businesses can enhance products and services, leading to higher satisfaction.
- **Reduced Churn:** Identifying and addressing negative sentiments promptly can help reduce user churn.
- **Scalability:** Capable of analyzing large volumes of text data efficiently.
- **Competitive Edge:** Offers businesses a competitive advantage by better understanding and catering to user needs and preferences.

3.5 Requirement Specification

3.5.1. Hardware Requirements:

- Standard PC with minimum 8GB RAM, 500GB HDD
- High-speed internet connection

3.5.2. Software Requirements:

- Operating System: Windows, macOS, or Linux
- Programming Languages: Python
- Libraries: NLTK, Scikit-Learn, Pandas, Matplotlib, Seaborn, Plotly Express, Streamlit
- IDE: Jupyter Notebook or VS Code

CHAPTER 4

Implementation and Result

CHAPTER 4

IMPLEMENTATION and RESULT

4.1 Implementation

4.1.1 Data Collection and Preprocessing

The project begins with collecting textual data, which can include speeches, reviews, or other forms of written content. The collected data is then subjected to preprocessing to prepare it for analysis. This involves:

- **Text Cleaning:** Removing unwanted characters, punctuation, and white spaces to standardize the text.
- **Stop Words Removal:** Eliminating common words that do not contribute to the analysis of sentiment or emotion.
- **Tokenization:** Breaking down the text into individual tokens or words.
- **Normalization:** Converting all text to lowercase and performing stemming or lemmatization to reduce words to their root forms.

4.1.2 Sentiment Analysis Implementation

For sentiment analysis, machine learning algorithms are employed:

- **Feature Extraction:** Using TF-IDF (Term Frequency-Inverse Document Frequency) to convert text data into numerical features that represent the importance of words in the dataset.
- **Model Training:** Implementing Support Vector Machines (SVM) and Random Forests to train models on labelled datasets. The models are trained to classify text into sentiment categories: positive, negative, or neutral.
- **Model Evaluation:** Assessing the performance of the models using metrics such as accuracy, precision, recall, and F1 score to ensure reliable sentiment classification.

4.1.3 Emotion Detection Implementation

Emotion detection is achieved through:

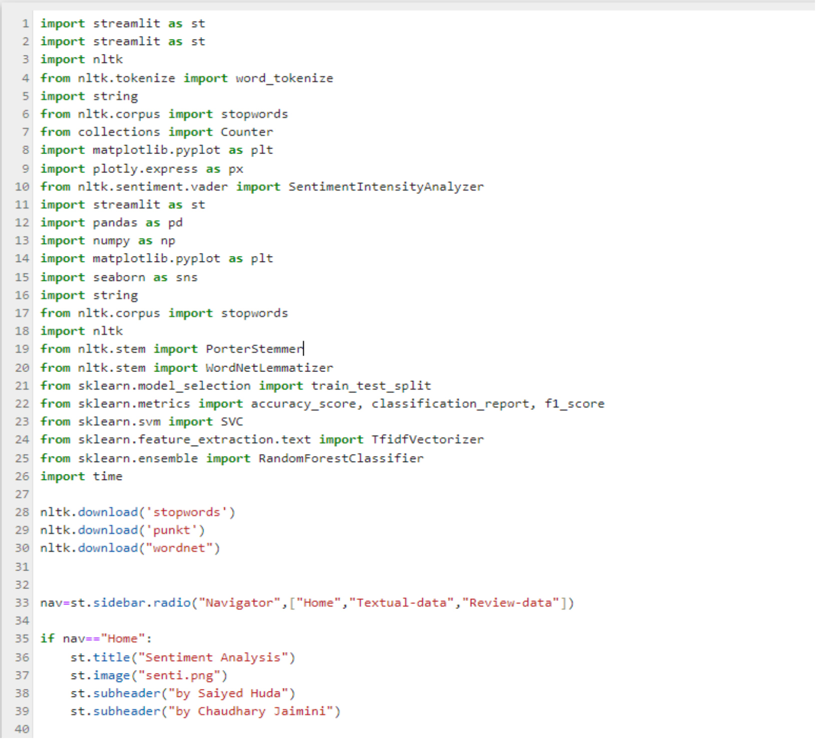
- **Emotion Lexicons:** Utilizing predefined emotion lists to identify and categorize emotions present in the text.
- **Text Analysis:** Mapping words in the text to their corresponding emotions and calculating the frequency of each emotion.
- **Visualization:** Generating visual representations of the detected emotions using charts and graphs.

4.1.4 Web Application Development

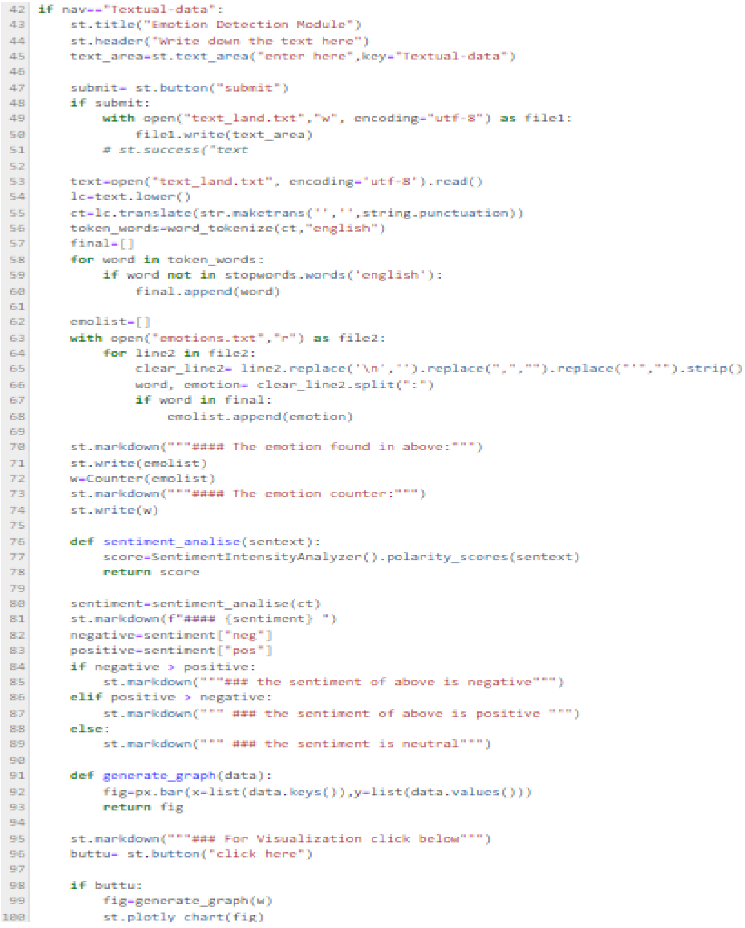
The web application is developed using Streamlit to provide an interactive user interface:

- **User Interface Design:** Creating pages for textual data input and review data input, where users can submit text or reviews and receive analysis results.
- **Integration:** Combining the sentiment analysis and emotion detection modules into the Streamlit application.
- **Visualization Integration:** Implementing dynamic visualizations that display sentiment and emotion analysis results in an interactive manner.

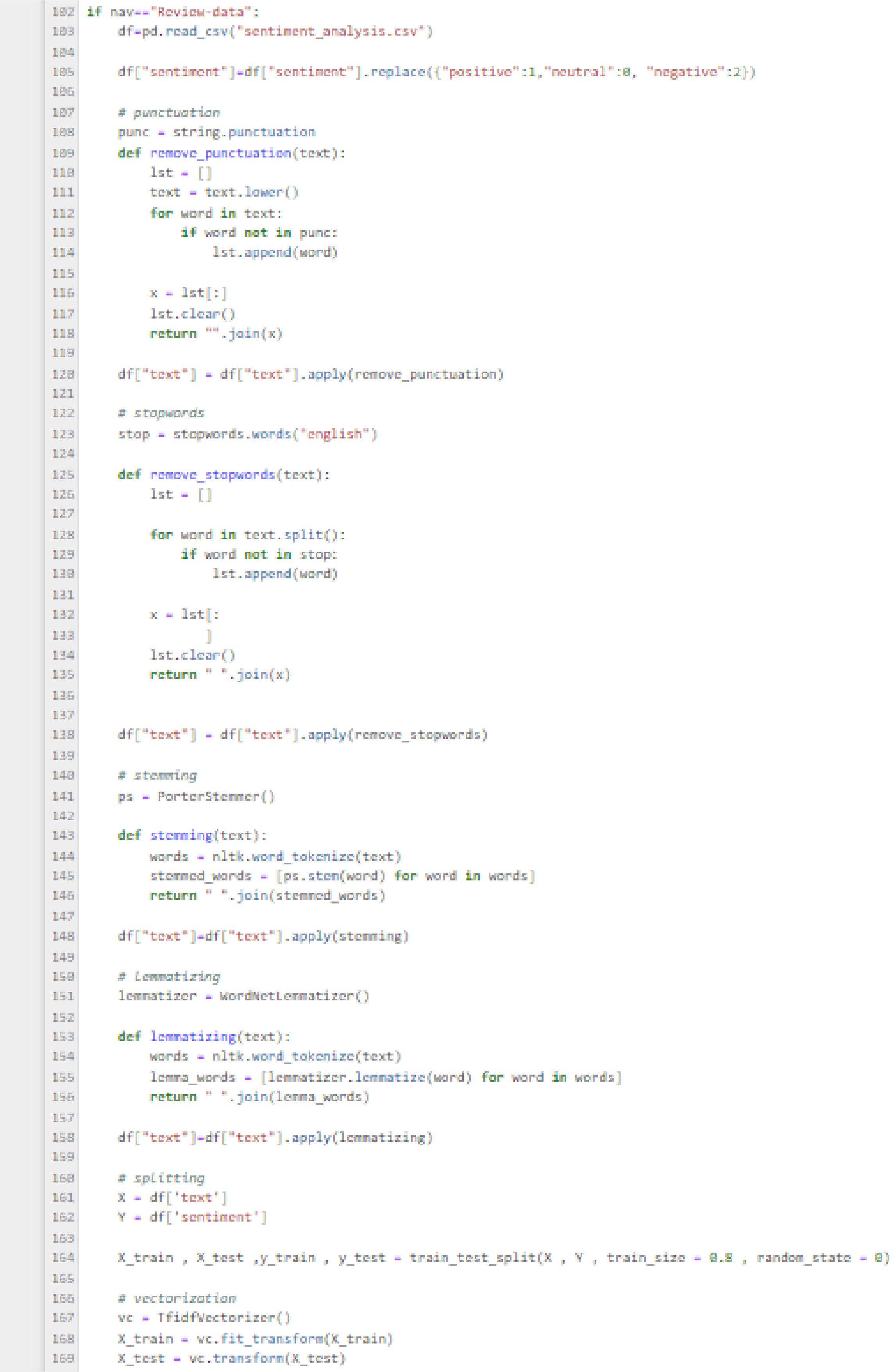
4.1.5. Implementation Code:



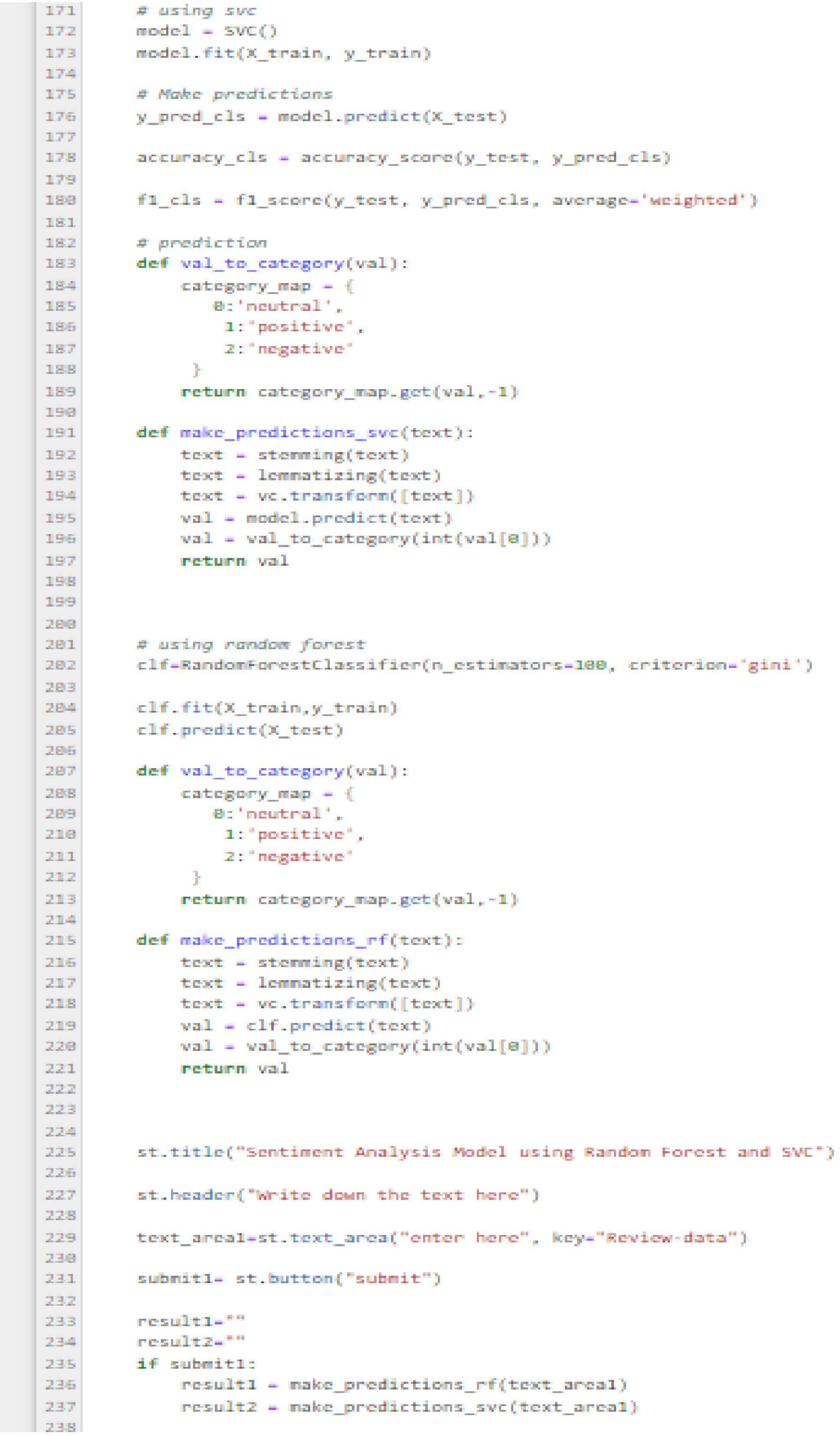
```python
import streamlit as st
import streamlit as st
import nltk
from nltk.tokenize import word_tokenize
import string
from nltk.corpus import stopwords
from collections import Counter
import matplotlib.pyplot as plt
import plotly.express as px
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
from nltk.corpus import stopwords
import nltk
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, f1_score
from sklearn.svm import SVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
import time

nltk.download('stopwords')
nltk.download('punkt')
nltk.download("wordnet")


nav=st.sidebar.radio("Navigator",["Home","Textual-data","Review-data"])

if nav=="Home":
    st.title("Sentiment Analysis")
    st.image("senti.png")
    st.subheader("by Saiyed Huda")
    st.subheader("by Chaudhary Jaimini")

```



```python
if nav=="Textual-data":
    st.title("Emotion Detection Module")
    st.header("Write down the text here")
    text_area=st.text_area("enter here",key="Textual-data")

    submit= st.button("submit")
    if submit:
        with open("text_land.txt","w", encoding="utf-8") as file1:
            file1.write(text_area)
        # st.success("text

    text=open("text_land.txt", encoding='utf-8').read()
    lc=text.lower()
    ct=lc.translate(str.maketrans('','',string.punctuation))
    token_words=word_tokenize(ct,"english")
    final=[]
    for word in token_words:
        if word not in stopwords.words('english'):
            final.append(word)

    emolist=[]
    with open("emotions.txt","r") as file2:
        for line2 in file2:
            clear_line2= line2.replace('\n','').replace(",","").replace("'","").strip()
            word, emotion= clear_line2.split(":")
            if word in final:
                emolist.append(emotion)

    st.markdown("""#### The emotion found in above:""")
    st.write(emolist)
    w=Counter(emolist)
    st.markdown("""#### The emotion counter:""")
    st.write(w)

    def sentiment_analise(sentext):
        score=SentimentIntensityAnalyzer().polarity_scores(sentext)
        return score

    sentiment=sentiment_analise(ct)
    st.markdown(f"#### {sentiment} ")
    negative=sentiment["neg"]
    positive=sentiment["pos"]
    if negative > positive:
        st.markdown("""### the sentiment of above is negative""")
    elif positive > negative:
        st.markdown(""" ### the sentiment of above is positive """)
    else:
        st.markdown(""" ### the sentiment is neutral""")

    def generate_graph(data):
        fig=px.bar(x=list(data.keys()),y=list(data.values()))
        return fig

    st.markdown("""### For Visualization click below""")
    buttu= st.button("click here")

    if buttu:
        fig=generate_graph(w)
        st.plotly_chart(fig)
```



```
if nav=="Review-data":
    df=pd.read_csv("sentiment_analysis.csv")

    df["sentiment"]=df["sentiment"].replace({"positive":1,"neutral":0, "negative":2})

    # punctuation
    punc = string.punctuation
    def remove_punctuation(text):
        lst = []
        text = text.lower()
        for word in text:
            if word not in punc:
                lst.append(word)

        x = lst[:]
        lst.clear()
        return "".join(x)

    df["text"] = df["text"].apply(remove_punctuation)

    # stopwords
    stop = stopwords.words("english")

    def remove_stopwords(text):
        lst = []

        for word in text.split():
            if word not in stop:
                lst.append(word)

        x = lst[:
              ]
        lst.clear()
        return " ".join(x)


    df["text"] = df["text"].apply(remove_stopwords)

    # stemming
    ps = PorterStemmer()

    def stemming(text):
        words = nltk.word_tokenize(text)
        stemmed_words = [ps.stem(word) for word in words]
        return " ".join(stemmed_words)

    df["text"]=df["text"].apply(stemming)

    # Lemmatizing
    lemmatizer = WordNetLemmatizer()

    def lemmatizing(text):
        words = nltk.word_tokenize(text)
        lemma_words = [lemmatizer.lemmatize(word) for word in words]
        return " ".join(lemma_words)

    df["text"]=df["text"].apply(lemmatizing)

    # splitting
    X = df['text']
    Y = df['sentiment']

    X_train , X_test ,y_train , y_test = train_test_split(X , Y , train_size = 0.8 , random_state = 0)

    # vectorization
    vc = TfidfVectorizer()
    X_train = vc.fit_transform(X_train)
    X_test = vc.transform(X_test)
```



```
# using svc
model = SVC()
model.fit(X_train, y_train)

# Make predictions
y_pred_cls = model.predict(X_test)

accuracy_cls = accuracy_score(y_test, y_pred_cls)

f1_cls = f1_score(y_test, y_pred_cls, average='weighted')

# prediction
def val_to_category(val):
    category_map = {
       0:'neutral',
        1:'positive',
        2:'negative'
     }
    return category_map.get(val,-1)

def make_predictions_svc(text):
    text = stemming(text)
    text = lemmatizing(text)
    text = vc.transform([text])
    val = model.predict(text)
    val = val_to_category(int(val[0]))
    return val



# using random forest
clf=RandomForestClassifier(n_estimators=100, criterion='gini')

clf.fit(X_train,y_train)
clf.predict(X_test)

def val_to_category(val):
    category_map = {
       0:'neutral',
        1:'positive',
        2:'negative'
     }
    return category_map.get(val,-1)

def make_predictions_rf(text):
    text = stemming(text)
    text = lemmatizing(text)
    text = vc.transform([text])
    val = clf.predict(text)
    val = val_to_category(int(val[0]))
    return val



st.title("Sentiment Analysis Model using Random Forest and SVC")

st.header("Write down the text here")

text_area1=st.text_area("enter here", key="Review-data")

submit1= st.button("submit")

result1=""
result2=""
if submit1:
    result1 = make_predictions_rf(text_area1)
    result2 = make_predictions_svc(text_area1)
```

4.2 Results

4.2.1 Sentiment Analysis Results

The sentiment analysis module accurately classifies text into positive, negative, or neutral categories. The performance of the models is evaluated based on:

- **Accuracy:** The percentage of correctly classified texts.
- **Precision and Recall:** Metrics indicating the model's ability to correctly identify positive, negative, and neutral sentiments.
- **F1 Score:** A measure that combines precision and recall to assess the overall performance of the sentiment classification.

The models show satisfactory performance in classifying sentiments, with Random Forests providing a higher accuracy compared to Support Vector Machines.

4.2.2 Emotion Detection Results

The emotion detection module successfully identifies and visualizes emotions from the text. Key results include:

- **Emotion Frequency:** The frequency of each emotion detected in the text, represented through bar charts or pie charts.
- **Visualization Accuracy:** The clarity and accuracy of visualizations in representing the detected emotions.

The visualizations provide clear and insightful representations of the emotional content of the text, allowing users to easily interpret and understand the results.

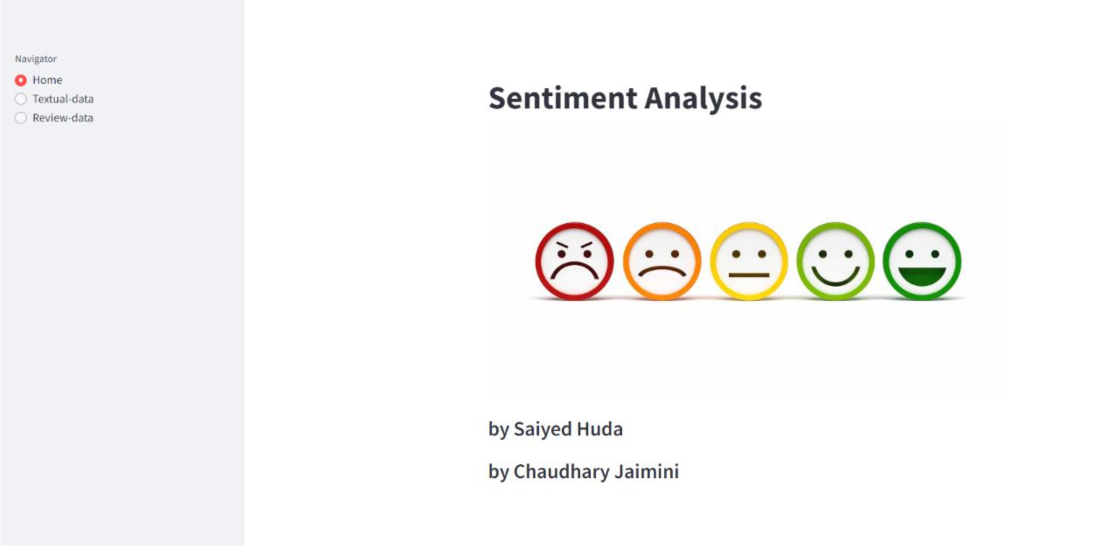
4.2.3 Web Application Results

The Streamlit web application functions effectively, offering:

- **Interactive Interface:** Users can input text and receive real-time sentiment and emotion analysis results.
- **Dynamic Visualizations:** Visualizations of sentiment and emotion analysis are interactive and informative, enhancing user experience and understanding.

The application has been tested thoroughly and demonstrates reliable performance in analyzing and visualizing textual data. The user interface is intuitive, and the integration of sentiment and emotion analysis modules provides valuable insights.

4.2.4: Result on Web browser:



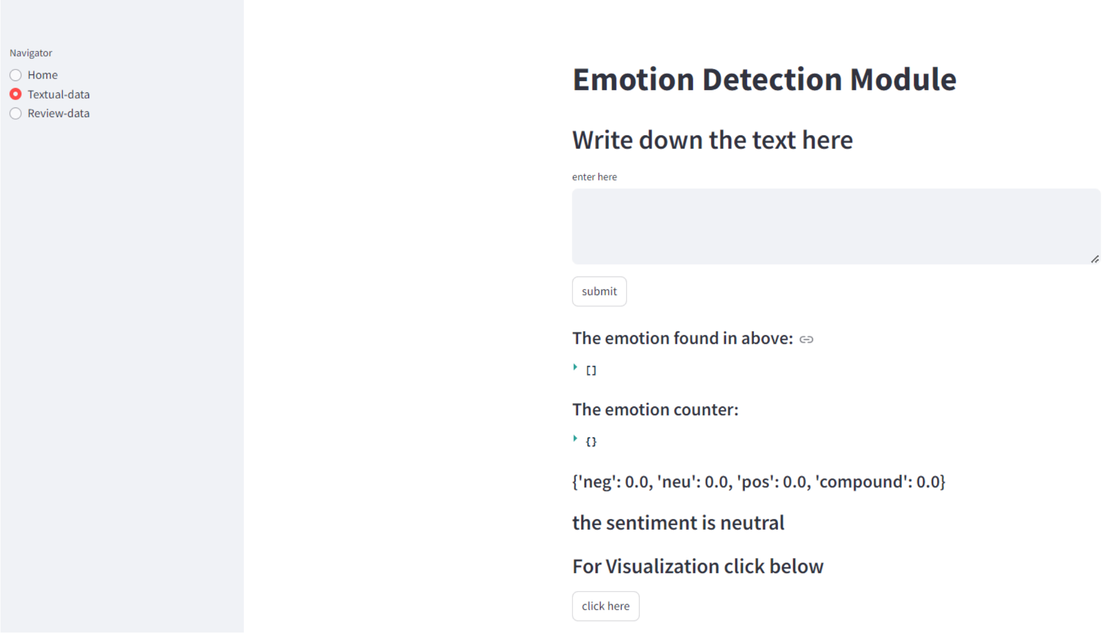
Navigator

● Home
○ Textual-data
○ Review-data

Sentiment Analysis

by Saiyed Huda

by Chaudhary Jaimini



Navigator

○ Home
● Textual-data
○ Review-data

Emotion Detection Module

Write down the text here

enter here

The emotion found in above:

```
[]
```

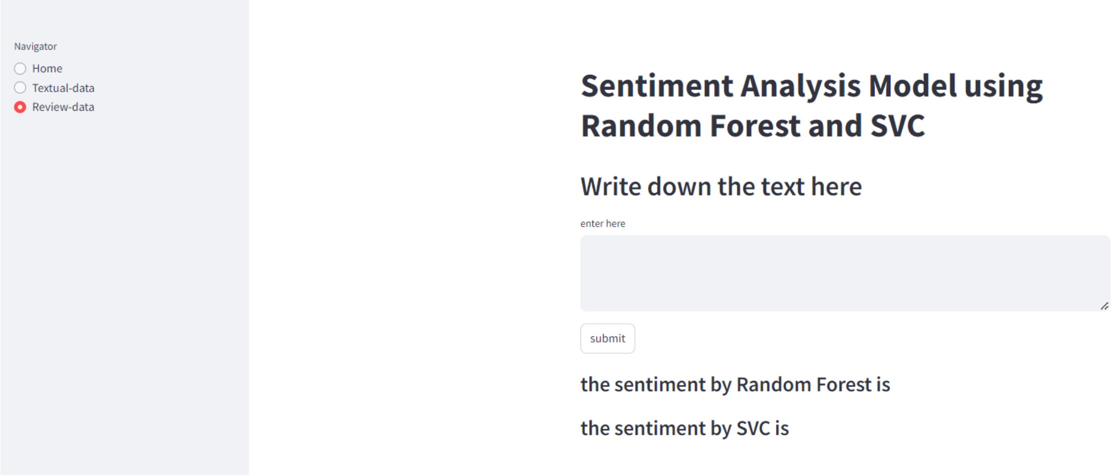
The emotion counter:

```
{}
```

{'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}

the sentiment is neutral

For Visualization click below



Navigator

○ Home
○ Textual-data
● Review-data

Sentiment Analysis Model using Random Forest and SVC

Write down the text here

enter here

the sentiment by Random Forest is

the sentiment by SVC is

CHAPTER 5

CONCLUSION

CHAPTER 5

CONCLUSION

The project "Sentiment Intelligence: Analyzing Textual Data" successfully developed a comprehensive system for analyzing and visualizing sentiments and emotions in textual content. Through the implementation of various machine learning algorithms and natural language processing techniques, the project provided valuable insights into how text can be categorized and understood in terms of sentiment and emotional content.

The sentiment analysis component of the project utilized Support Vector Machines (SVM) and Random Forests to classify text into positive, negative, or neutral categories. Both models were rigorously trained and evaluated, with Random Forests demonstrating higher accuracy. This indicates that the chosen machine learning models were effective in capturing and classifying sentiments from textual data.

In addition to sentiment analysis, the project incorporated emotion detection, leveraging predefined emotion lexicons to identify and categorize various emotions expressed in the text. This module successfully quantified and visualized emotions, providing users with clear and interpretable results through dynamic charts and graphs.

The development of the Streamlit web application allowed for an interactive and user-friendly interface. Users can easily input text, receive real-time sentiment and emotion analysis, and view results through engaging visualizations. The integration of sentiment analysis and emotion detection into a single application provides a robust tool for understanding textual data, making it valuable for applications in various fields such as customer feedback analysis, social media monitoring, and more.

Overall, the project demonstrated the feasibility and effectiveness of combining machine learning and natural language processing techniques to analyze and visualize text. The successful implementation and positive results underscore the potential for further development and application of this technology in real-world scenarios.

ADVANTAGES:

- **Efficiency:**
  Automates the sentiment analysis process, saving time and reducing the need for manual labor.
- **Accuracy:**
  Provides consistent and accurate categorization of sentiments as positive, negative, or neutral, minimizing human error.
- **Real-Time Insights:**
  Enables real-time analysis of user feedback, allowing businesses to quickly respond to user sentiments and adjust strategies accordingly.
- **Data-Driven Decisions:**
  Helps businesses make informed decisions based on precise sentiment data, leading to more effective strategies and outcomes.
- **Improved User Satisfaction:**
  By understanding user emotions, businesses can enhance their products and services, leading to higher user satisfaction and loyalty.
- **Reduced Churn:**
  Identifying and addressing negative sentiments promptly can help reduce user churn, retaining more customers.
- **Scalability:**
  Capable of analyzing large volumes of text data efficiently, making it suitable for businesses of all sizes.
- **Competitive Edge:**
  Offers businesses a competitive advantage by providing deeper insights into user needs and preferences, allowing for more targeted and effective marketing and customer service strategies.

SCOPE:

- **Integration with Additional Data Sources:**
  Expanding the project to analyze data from sources like social media and news articles would provide a more comprehensive view of sentiment and emotion, broadening its applicability.
- **Advanced Deep Learning Models:**
  Incorporating models such as BERT or LSTM could improve the accuracy of sentiment analysis by capturing more nuanced emotions and contexts.
- **Multilingual Support:**
  Adding support for multiple languages would extend the project's reach, enabling it to analyze sentiment across different linguistic and cultural contexts.
- **Real-Time Feedback and Adaptation:**
  Implementing mechanisms for real-time feedback would allow the model to continuously improve based on user interactions and new data, enhancing its accuracy and relevance.
- **Enhanced Visualization and Reporting:**
  Upgrading the user interface with advanced visualization tools and detailed reporting features would provide deeper insights and facilitate more comprehensive trend analysis

GitHub Link:

https://github.com/HudaSaiyed/Sentiment-Intelligence

Video Link:

https://drive.google.com/file/d/1MXbPNe7G_bstNIgCqE35IjsOlgD03lsK/view

REFERENCES

[1]. https://www.kaggle.com/datasets/mdismielhossenabir/sentiment-analysis

[2]. https://realpython.com/python-nltk-sentiment-analysis/

[3]. https://www.researchgate.net/publication/323777436_Document_Level_Sentiment_Analysis_A_survey