

A Mini Project Report On

SENTIMENT SPECTRUM: ANALYZING EMOTIONS IN TEXT

Submitted in Partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

In

Department of Computer Science and Engineering

By

Kodipaka Akanksha (22245A0522)

Under the Esteemed guidance of

D PAVITHRA

Assistant Professor



Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Bachupalli, Kukatpally, Hyderabad, Telangana, India, 500090

2023-2024



GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

CERTIFICATE

This is to certify that the mini project entitled “**Sentiment Spectrum: Analyzing Emotion in Text**” is submitted by **Kodipaka Akanksha (22245A0522)**. Partial fulfillment of the requirements the award of a degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2023-2024**.

INTERNAL GUIDE

D PAVITHRA
Assistant Professor

HEAD OF THE DEPARTMENT

Dr. B. SANKARA BABU
Professor

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our internal guide **D Pavithra, Assistant Professor**, Department of CSE for his support in the completion of our project report. We wish to express our honest and sincere thanks to **Dr. B. Shankar Babu, HOD**, Department of CSE, and to our principal **Dr. J. Praveen** for providing the facilities to complete our mini project. We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Kodipaka Akanksha (22245A0522)

DECLARATION

I hereby declare that the Mini Project entitled “**Sentiment Spectrum: Analyzing Emotions in Text.**” is the work done during the period from 6th feb 2024 to 29th June 2024 and is submitted in the Partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology**. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Kodipaka Akanksha (22245A0522)

	Table of contents	
Chapter No	TITLE	Page No
	Abstract	1
1	Introduction	2
	1.1Importance of sentiment analysis	2
	1.2Algorithms Used	2
	1.3Application of sentiment analysis	2
2	System Requirements	3
	2.1 Software Requirements	3
	2.2 Hardware Requirements	3
	2.3 Data Set	3
3	Literature survey	4-6
4	Propose Approach , Modules Description, and UML Diagrams	7-14
	4.1 Modules	7-9
	4.2 UML Diagrams	10-14
5	Implementation, Experimental Results &Test Cases	15-23
6	Conclusion and Future Scope	24
7	References	25

	LIST OF FIGURES	
Fig No	Fig Title	Page No
1.0	Datasets	3
1.1	Usecase diagram	11
1.2	Class diagram	12
1.3	Sequence diagram	12
1.4	State diagram	13
1.5	System architecture	14
1.6	Importing libraries	16
1.7	Data cleaning	16
1.8	Preprocessing data	16
1.9	Lemmatization	17
1.10	Handling imbalanced data	17
1.11	Train test split	18
1.12	Vectorization	18
1.13	Logistic regression	19
1.14	Multinomial naïve bayes	20
1.15	Random forest classifier	21
1.16	Support vector machine	22
1.17	Deployment of front end and back end using flask	23
1.18	Input	24
1.19	output	24

ABSTRACT

In digital world, the ability to understand and interpret human emotions conveyed through text is becoming essential for various applications. This project, Sentient Spectrum, focuses on developing a comprehensive framework for analyzing and categorizing emotions expressed in textual data. It will serve as a project pushing into the more advanced techniques of NLP and machine learning, further away from simple sentiment detection into the spectrum of nuanced feelings such as joy, love, anger, sadness, surprise, and fear.

The study begins by reviewing existing methodologies in emotion detection, highlighting their strengths and limitations. Machine learning methods, both supervised and unsupervised, are discussed in detail, showcasing how they can be trained on large datasets to recognize complex emotional patterns. The project also explores the potential of deep learning models, particularly recurrent neural networks (RNNs) and transformers, in capturing the subtleties of human emotions.

One of the primary challenges addressed in this project is the accurate interpretation of context-dependent emotions, including sarcasm, irony, and mixed emotions. To tackle this, the Sentient Spectrum Analysis framework integrates context-aware models and employs techniques such as sentiment shift detection and emotion intensity scoring. These enhancements aim to improve the precision and reliability of emotion classification.

Future for Sentient Spectrum Analysis include expanding the emotion categories to encompass a wider range of human experiences and incorporating multimodal data to enhance the richness of emotion detection. By advancing the capabilities of emotion analysis in text, this project aims to contribute significantly to fields such as human-computer interaction, social science research, and beyond.

In conclusion, Sentient Spectrum Analysis offers a sophisticated approach to understanding human emotions in text, providing valuable insights that can transform how we interact with and interpret digital communications. This project not only addresses current limitations in emotion detection but also sets the stage for future innovations in the field.

CHAPTER 1

INTRODUCTION

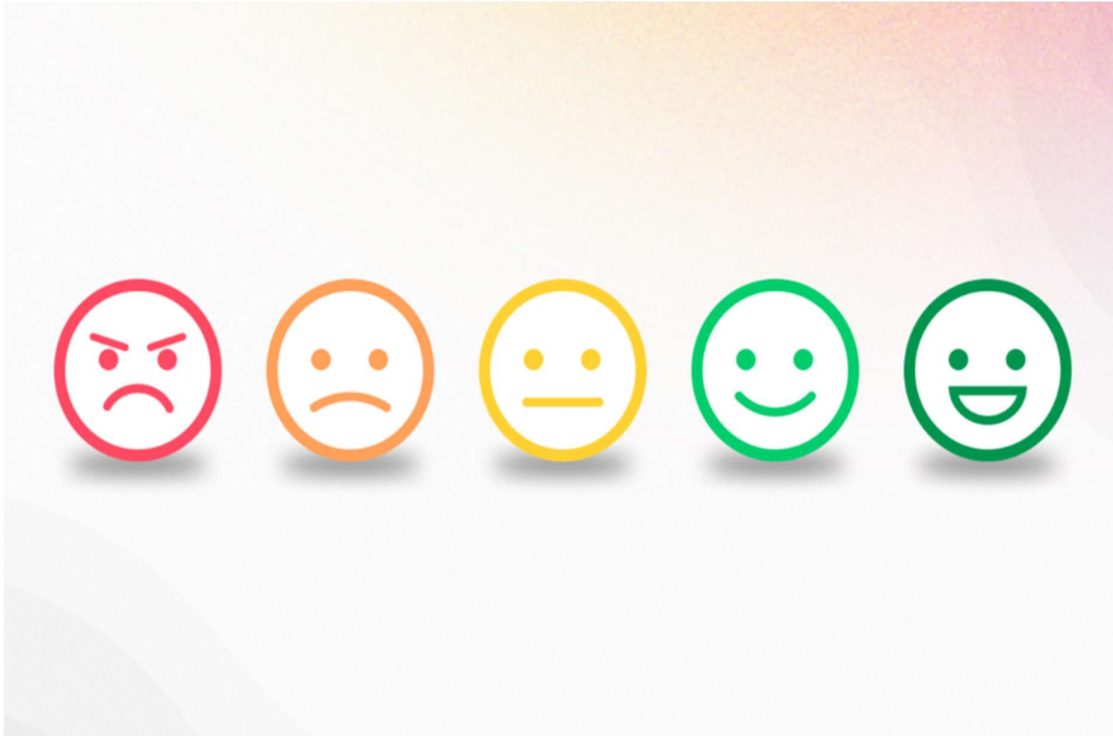
Sentiment analysis is the process for analyzing sentiments in a block of text. It can also be described as a simple procedure through which emotion in the given text can be classified. The main purpose of Sentiment analysis is to find out in what way the text has been gushing by the writer.

Now a days social media is developing a vast amount of data which consists of different kinds of emotions. Not everybody out there understands the text that the social media generates, sentiment analysis can work as a boon to such. Sentiment analysis predicts the emotions in the text and classifies it in to positive, negative or neutral. All this can be done by using different types of Machine learning algorithms which analyze the text in one go.

In today's generation many of them opt sentiment analysis in order to improve their marketing strategies. It has various kinds of advantages that are being used by the people. For example, talking of any running company, sentiment analysis can help them by letting them know how the customers or buyers feel about their service and their products. By this they can implement any changes basing up on the feedback they receive. Sentiment analysis also helps them know their strengths and weaknesses in the market.

1.1 Importance Of Sentiment Analysis:

- Sentiment analysis is becoming very important to study growing opinions faster and faster with in the social media and other sites.
- The data being generated now a days is very rich in emotions, has complex words. All these can not be understood in a traditional way.
- So, sentiment analysis comes in the picture. It makes the text easier to understand.
- The sentiment analysis requires 5 steps: collecting data, processing of data, feature extraction, sentiment classification, and output presentation.



1.2 Algorithms Used:

Sentiment analysis can be performed using different kinds of machine learning algorithms, few are listed below:

- Multinomial naïve bayes: this is an algorithm which classifies the text based on the bayes theorem.
- Logistic regression: This is usually used for simple and linearly separable data.
- Support vector machines (SVM): This algorithm finds a hyper plane which separates different classes from each other.
- Random forest: This algorithm creates decision trees by using a random subset of the data.

1.3 Application Of Sentiment Analysis:

Sentiment analysis is used in various kinds of industries, including:

- Feedback analysis: Companies can improve their work and the quality of their products by the customer feedback.
- Social media monitoring: Brands focus on social media to see where the public is more interested in and how they can approach them in an easier way.

- Market research: Sentiment analysis helps them to know their strengths and weaknesses in the market.
- Political analysis: This can be useful as the politicians find an easier way to convey their promises.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

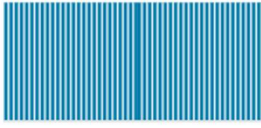

- Programming language: we use python
- Data collection library: libraries such as tweepy can be used
- Text processing library : Libraries like NLTK (Natural Language Toolkit), keras can be used for processing tasks such as tokenization, stemming, and lemmatization.
- Sentiment analysis library: Libraries such as NLTK, sklearn, tensor flow can be used for sentiment analysis. These libraries can assign polarity scores to texts.

2.2 Hardware Requirements

- Processor : 11th Gen Intel(R) Core(TM) i5
- System Type : 64-bit operating system, x64-based processor
- Memory : 8GB RAM

2.3 Data Set

- We have taken the dataset from the website Kaggle, called Emotions. The Entries in the dataset contains of text segment representing the emotion of the text. It has 3 fields ID, Text, Label. It has 6 labels sadness 0, joy 1, love 2, anger 3, fear 4, and surprise 5.
- Text: text representing the message.
- Label: class indicating the sentiment int the text.

# id	△ text Twitter message	# label Classification label
 0 417k	393822 unique values	 0 5
0	i just feel really helpless and heavy hearted	4
1	ive enjoyed being able to slouch about relax and unwind and frankly needed it after those last few w...	0
2	i gave up my internship with the dmrgr and am feeling distraught	4
3	i dont know i feel so lost	0

	Text	Label
0	i just feel really helpless and heavy hearted	4
1	ive enjoyed being able to slouch about relax a...	0
2	i gave up my internship with the dmrgr and am f...	4
3	i dont know i feel so lost	0
4	i am a kindergarten teacher and i am thoroughl...	4
5	i was beginning to feel quite disheartened	0
6	i would think that whomever would be lucky eno...	2

Fig.1.0 Datasets

CHAPTER 3

LITERATURE SURVEY

Sentiment analysis is the process of finding out the sentiments present in a block of text. It is the technique through which the emotion in the text can be classified. In this era, we are able to handle tremendously large volume of text data that includes miscellaneous information stated in the e-mails, comments in the social media platform, reviews, and so on in the other forms of text data. Using a very few number of sentiment analysis tools, the scan can be made to automatically determine the attitude of the author towards a topic.

The main approaches towards the sentimental analysis are by using techniques related to Machine Learning, sentiment classifying algorithms, including Logistic regression, Naive Bayes, Random Forest, and technique 4-Support Vector Machine. Using these above-specified algorithms can perform opinion mining among the various datasets and among the tools.

During the experiment, it was the best result that not even a single tool or technique was identified enough to be the best for sentiment identification. Although the second approach is rarely successful in finding improved performance far more frequently than the first. Many experiments have incorporated benchmark datasets along with a wide variety of term-weighting schemes to extract relevant features and then selected the features by employing a Chi-square method. For the classification, Support Vector Machine (SVM) was used for the process performed in the author's experiment. The evaluation of the results was done in terms of Precision, Recall, Accuracy, F measures, and AUC for analyzing the effectiveness of the proposed method.

Researches over Sentiment Analysis have been developed very much in the recent years. They have come up with many issues in domains of datasets, corpus types, size, and also multilingual contexts.

Some of the issues related to the domains are investigated, where sentiment analysis was applied to online movie reviews by combining

A number of studies are done to do sentiment classification in a multilingual context. An appropriate mechanism for feature selection is needed to look over the issues in these datasets to extract useful features before classification.

Datasets will go through a preprocessing task that involves text documents: tokenization, stop-word removal, lowercasing, and stemming.

Feature Selection methods

For selecting the features filter, wrapper and embedded are the feature selection methods.

In the experiments, SVM was selected as the classifier because it works quite well for text classification, as it can handle large features.

There are four effective measures that are used in this paper which are based on the confusion output matrix output, which are True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

- Precision(P) = $TP / (TP + FP)$
- Recall(R) = $TP / (TP + FN)$
- Accuracy(A) = $(TP + TN) / (TP + TN + FP + FN)$
- AUC (Area under the (ROC) curve) = $1/2 \cdot ((T - P / (TP + FN)) + (TN / (TN + FP)))$

The overall results have been to show that such a process of selecting the features based on their Chi-squared statistics value helps in reducing the dimensionality and the noisy data in the text, allowing high performance of a classifier that could be comparable to topic categorization. Also, it can be seen that the prediction accuracies by using SVM achieved higher when the number of features selected were fewer.

This other classification study uses an algorithm similar to the Naive Bayes method, in which sentiment for training data comes from the large data texts, public comments on social media, or movie reviews.

The main focus is on unigram features calculated based on term frequency-Laplace smoothing.

It started from the collection of data meant for measurement pertaining to classification and accuracy after data preprocessing and feature selection. The best classification prediction using the KNMIE tools is then compared with prediction using the Naive Bayes classifier, and finally, with the prediction using KNN and/or that of the decision tree classification.

Unigram feature extraction and frequency weighting are used for the Naive Bayes method; both are used as reference features in shaping the sentence-level analysis sentiment application. Accuracy values are compared using tools before determination of algorithm to be used for the application.

Using NLP techniques, that is, preprocessing data, stemming, and tokenization formation of the basic words wherein the training data is processed.

The keywords related to service are combined with sentiment words in filtering the training data.

The following previsions are used in selecting the training data among the large data:

- Determining the target/entity.
- Determine the positive/negative sentiment keywords that accompany a comment sentence.

There after, the statements are manually classified as positive, negative and neutral into separate files. The data will then be split the training data and the testing data. The training data is then manually labeled accordingly as either positive, negative or neutral. Sentiment classification is achieved using

This Naive Bayes algorithm is implemented in two stages: the learning process stage and the classification stage. By implementing the learning process of the Naive Bayes classifier in the training data, there is also the sentence classification stage in the test data by use of the word probability of the results of the training data.

Based on the research, the following finds out that it is possible to perform accurate sentiment classification analysis of the sentence level.

So, with the accuracy obtained, it made the sentiment analysis application classifiable under sentences of positive, negative, and neutral categories.

The other two algorithms that we have worked for the sentiment analysis are Random Forest classifier and Logistic Regression.

Random Forest classifies sentiments of user reviews with the advantages of Random Forest classifiers. It is an ensemble learning technique applied to augment the performance, in terms of correctness and speed, of sentiment classification. We tide away the promising approach to extract meaningful sentiment information from large sets of text data.

Sentiment analysis can also be done using the Logistic Regression model. Logistic Regression is a statistical method used for binary classification and will be perfect when applied to this task. There are some process steps for conducting Sentiment analysis by using Logistic Regression which are listed as follows:

- Data collection
- Preprocessing
 - Tokenization
 - Cleaning
 - Normalization
 - Removing stop words
 - Stemming/Lemmatization
- Feature extraction
- Model training with Logistic Regression
- Model testing and validation
- Application
- Challenges and considerations.

Through the above steps undertaken in the sentiment analysis, using a logistic regression model, the accuracy is obtained with which the classification can be done for the sentences under the categorization of positive, negative, or neutral.

CHAPTER 4

MODULES DESCRIPTION AND UML DIAGRAMS

4.1 Modules:

4.1.1 Python:

Python has been widely applied for sentiment analysis due to the fact that it offers many built-in tools and libraries that are oriented toward sentiment analysis based on text emotion understanding. Additionally, this ease is backed by a strong community that enables the easy development and production of state-of-the-art models.

4.1.2 NumPy:

NumPy provides support for huge multi-dimensional arrays and matrices, coupled with a collection of mathematical functions to execute several operations on them. This lends support to efficient numerical computations that lie at the heart of sentiment analysis.

4.1.3 Pandas:

Pandas is a library for data manipulation that enhances the process of cleaning, transformation, and analysis of data. More precisely, it can turn out to be very useful while dealing with structured data like text datasets, providing the capability of easy data manipulation and preprocessing before conducting sentiment analysis.

4.1.4 Matplotlib.pyplot:

Matplotlib.pyplot is an extremely flexible plotting library, used for drawing all types of graphs. In sentiment analysis, this can be advantageous in the processes geared toward visualizing data distributions, trends, and patterns, which will continually make sense out of an analysis through interpretation or communication of results

4.1.5 Seaborn:

Seaborn is based on Matplotlib and exposes a high level interface for creating beautiful and informative statistical graphics. It makes the process of creating complicate visualizations easy

and, therefore, very useful in exploring and finding data relationships and trends in sentiment analysis.

4.1.6 Word Cloud:

A specialized library for visual display of the text data, where the size of each word generally becomes the frequency or importance of the word. Word clouds emphasize the most important terms in one glance and allow an accessible overview of the main themes in a text corpus of sentiment analysis.

4.1.7 NLTK—Natural Language Toolkit:

NLTK is a general-purpose library for dealing with human language data. It includes interfaces to over 50 corpora and lexical resources, along with text processing libraries for doing things such as classification, tokenization, stemming, tagging, parsing, and more. NLTK will be used in a number of the preprocessing phases of sentiment analysis.

4.1.8 WordNet (from NLTK corpus):

WordNet combines lexical database of English with groupings of words into synsets, providing definitions and usage examples. This tool has much enhanced text understanding in sentiment analysis by detection of synonyms, word sense disambiguation, and semantic similarity measurement.

4.1.9 WordNet Lemmatizer (from NLTK.stem):

The WordNet lemmatizer is contained in NLTK's stem module. Lemmatization is the process of reducing words to a common base or dictionary form of words before they get actually inflected. It helps in normalizing the text by reducing an inflected word down to so that common base form to ensure more accurate sentiment analysis tasks.

4.1.10 Stop Words:

The library in NLP Stop words consists of certain specific words in a language that are mostly removed or filtered from text before processing. Getting rid of these stop words puts attention on resourceful and important words, ensuring effectiveness and accuracy in sentiment analysis.

4.1.11 word_tokenize:

The `word_tokenize` function has the effect of breaking down text into words or tokens; this stage is very important in the preparation of text data for analysis. This function is useful in tasks such as parsing, text mining, and sentiment analysis.

4.1.12 TfidfVectorizer (from `sklearn.feature_extraction.text`):

The class `TfidfVectorizer` from `sklearn.feature_extraction.text` permits the sentimental transformation of text documents into a matrix of TF-IDF features. TF-IDF levels the importance of a word in one document against a collection; it is a measure that puts more weight on the important words while reducing the weight of less informative words.

4.1.13 Train_test_split from `sklearn.model_selection`:

The `train_test_split` function from `sklearn.model_selection` does the splitting of a dataset into training and testing sets. This function also enables one to build a model of a machine learning algorithm on one dataset and test it on another unknown dataset.

4.1.14 TensorFlow:

Tesla— An open source framework developed by Google for creating and implementing machine learning models, designed to support a wide range of machine learning tasks—including neural network training or natural language processing—in a flexible and scalable fashion.

4.1.15 Tokenizer: from `tensorflow.keras.preprocessing.text`

The class `Tokenizer` from `tensorflow.keras.preprocessing.text` will turn text into sequences of integers where each integer represents a different word, or 'token'. This step is required since neural networks won't accept text data directly as input.

4.1.16 LogisticRegression (from `sklearn.linear_model`):

`LogisticRegression` from `sklearn.linear_model` is a popular machine learning algorithm used for binary and multiclass classification problems. The probability that a given input belongs to one of the classes is modeled.

4.1.17 MultinomialNB (from sklearn.naive_bayes):

MultinomialNB in sklearn.naive_bayes is a Naive Bayes algorithm for classification problems with discrete features, like counts of words in text classification.

4.1.18 SVM:

Support Vector Machine is a very strong technique in supervised machine learning for regression and classification. It finds the optimal hyperplane that separates the data into different classes.

4.1.19 RandomForestClassifier from sklearn.ensemble:

RandomForestClassifier is a scikit-learn class, a classification algorithm within its ensemble module. It trains many decision trees on the training data and in return, while classifying it returns the mode class of those classes predicted by clamping down through each of its trees from top to bottom.

4.1.20 Joblib:

Joblib is a set of tools to provide lightweight pipelining in Python. In particular, joblib offers transparent disk-caching of the output values and easy simple parallel computing of the input values. It efficiently saves and loads Python objects, especially huge ones so memoizing objects on disk were very easy and worked with most existing code.

4.1.21 Render_template:

The render_template function is purely a part of the Flask Python web framework that one could use to render HTML templates. This allows the creation of dynamic web pages comprising HTML templates and data passed into the template from the backend.

4.2 Proposed Approaches:

Proposed approaches for sentiment analysis often employ supervised machine learning algorithms for sentiment classification on text. Four such algorithms are in common use:

4.2.1 Logistic Regression:

Logistic Regression—This is a powerfully simple algorithm for binary and multi-class classification tasks. It models the relationship between the features extracted from the text and the probability of a particular sentiment class.

4.2.2 Random Forest:

Random Forest is another ensemble learning technique that involves the construction of many decision trees during the training phase. It generally improves predictive accuracy and makes the system more robust by reducing overfitting and handling high-dimensional feature spaces efficiently.

4.2.3 Multinomial Naive Bayes:

Multinomial Naive Bayes is a probabilistic classifier based on Bayes' theorem, particularly suited for text classification tasks with word counts or frequencies. Particularly speaking, it models a conditional probability of every class of sentiments given the word frequencies in the text.

4.2.4 Support Vector Machines:

SVM is a supervised learning method for regression and classification problems. Within the domain of sentiment analysis, SVMs find the optimal hyperplane that maximizes the margin between different classes based on the feature vectors derived from the text data.

Among the various machine learning algorithms available, each algorithm has different strengths, and each choice encompasses parameters such as dataset size, computational resources, and characteristics related to the sentiment analysis task. Experimenting with these algorithms on labeled datasets for sentiment and evaluating them goes a long way in picking the best approach for a specific application.

4.3 UML Diagrams

The unified modeling language (UML) is a conventional language for writing software layouts. It helps in defining, envisioning, assembling, and archiving the artifacts of software systems.

It is a Systematized graphical design framework used in the field of software to provide a Multi-purpose manner so as to present the design of the system.

4.3.1. Usecase diagram:

A use case diagram could be defined as a graphical representation of the interaction between a system's different users and the system itself to describe the system's functional requirements. It points out and explains all the main functionalities of the system and various types of users who interact with the system.

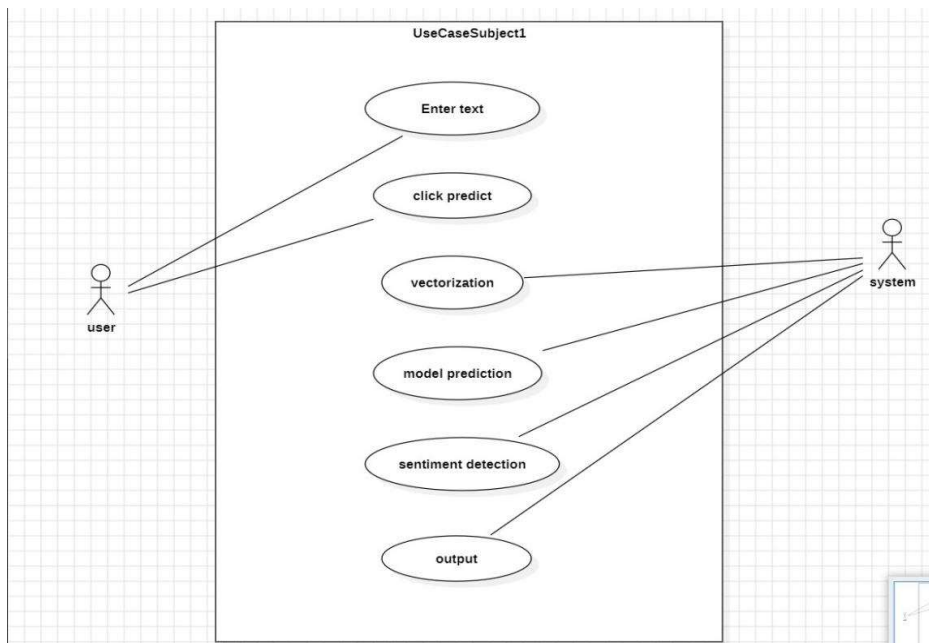


Fig1.1.Usecase diagram

4.3.2. Class Diagram:

A class diagram may be defined as a static structure UML diagram that describes the structure of a system by showing the classes involved, their attributes, operations, or methods, and the relationships between them.

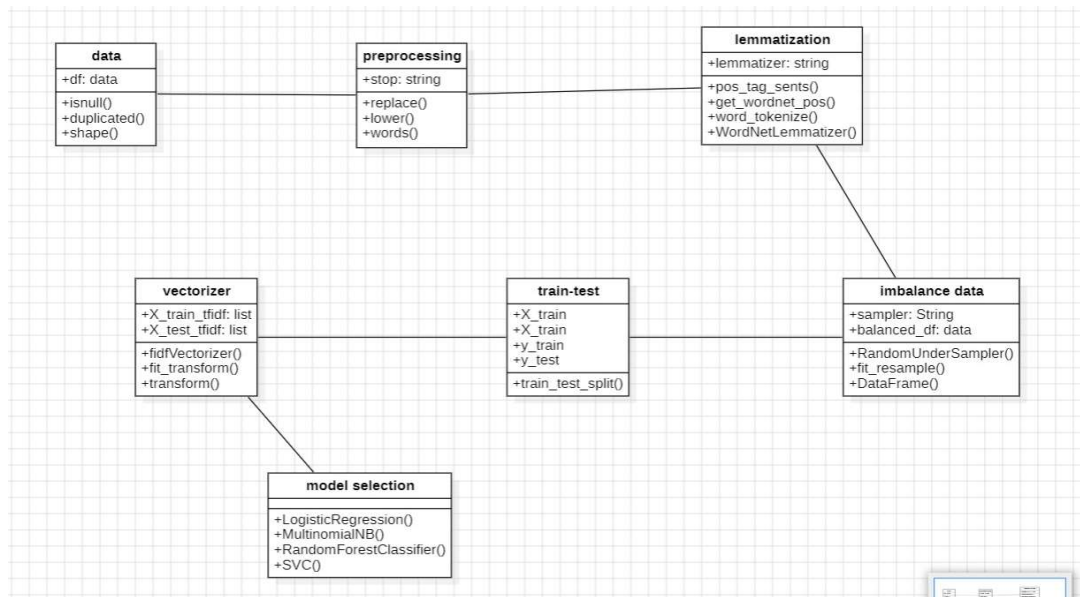


Fig1.2 Class Diagram

4.3.3 Sequence diagram:

An interaction diagram, a sequence diagram shows how objects work with each other and in what order. It is used to depict the sequence of messages exchanged between the objects in the system to, these are used to show the overall flow of control. To perform a particular function or process.

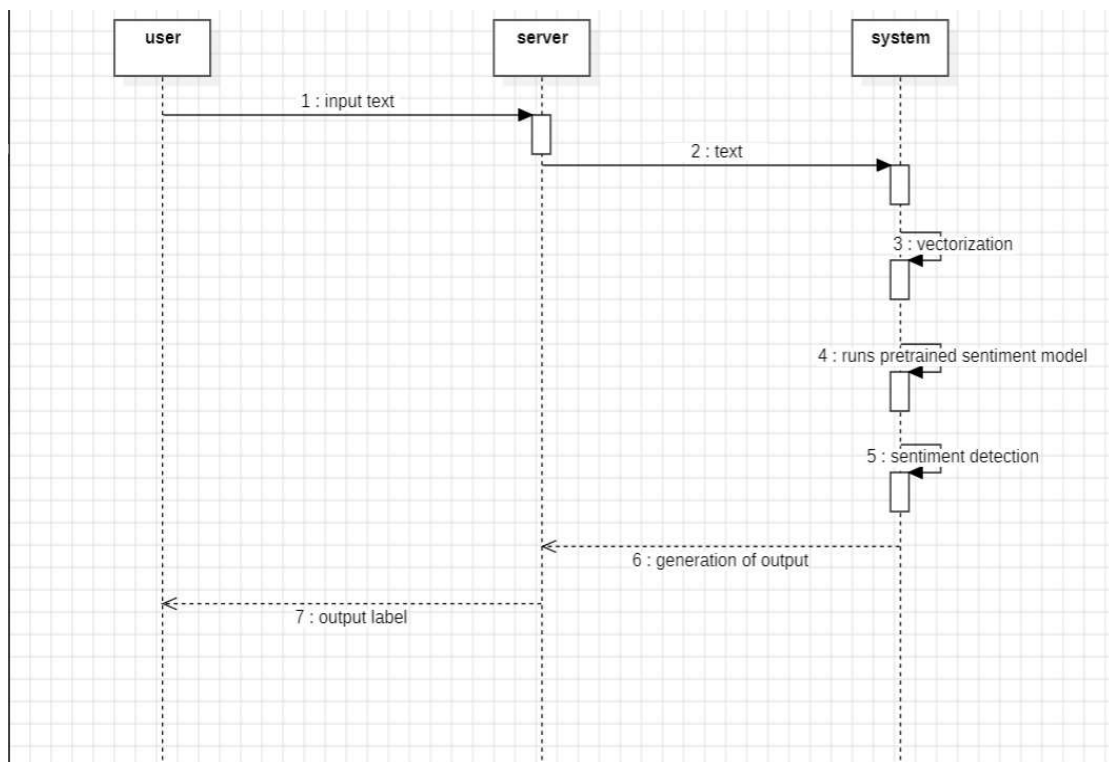


Fig 1.3. Sequence Diagram

4.3.4 State Diagram:

State diagrams, otherwise known as state machine diagrams, are among the behavioral diagrams in Unified Modeling Language. They are used to model the states an object can have and the transitions between these states due to the occurrence of events. State diagrams are used for modeling systems where dynamic behavior involves an object or system existing in different states, depending upon events and conditions.

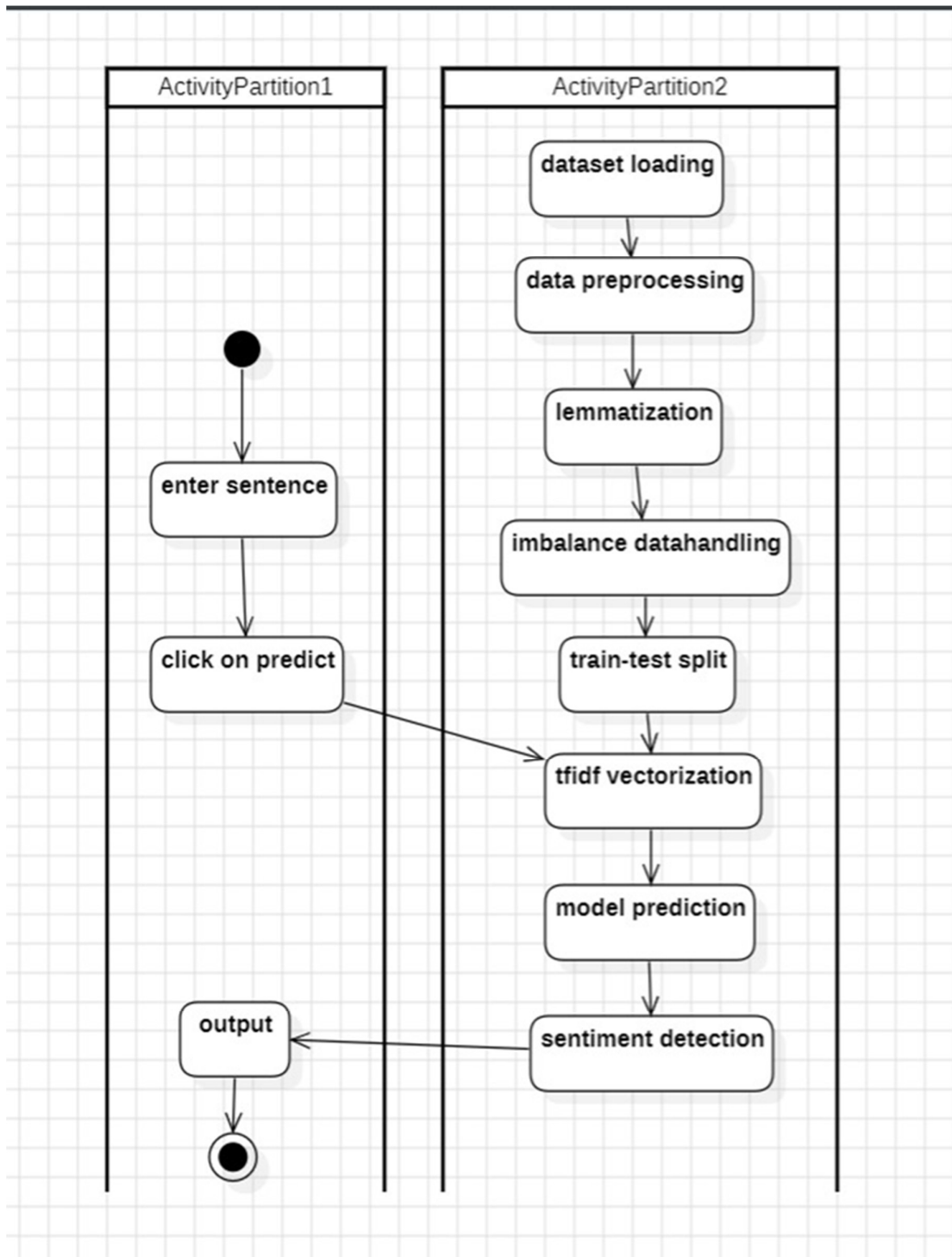


Fig 1.4. State Diagram

4.3.5. System architecture:

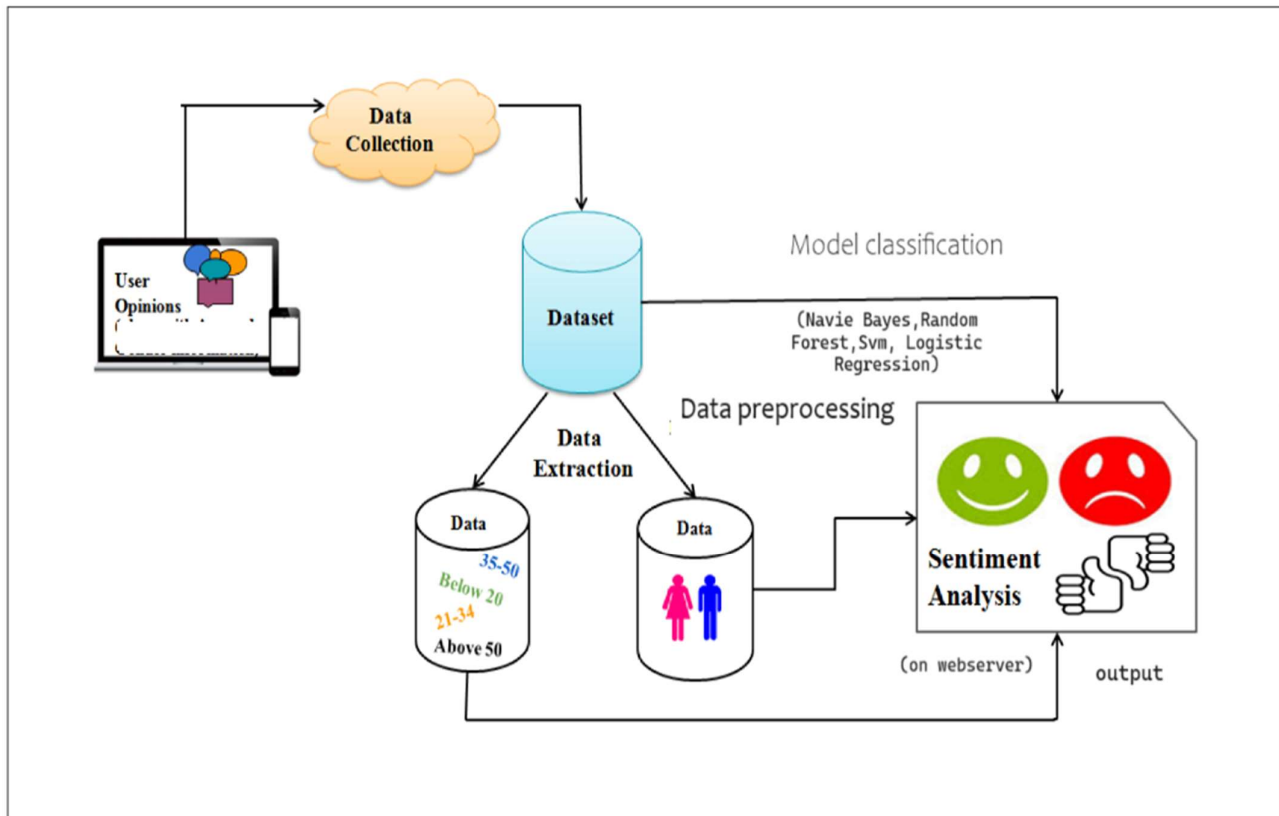


Fig 1.5. System Architecture

CHAPTER 5

IMPLEMENTATION, EXPERIMENTAL RESULTS

5.1 Implementation

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import nltk
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
#from keras.preprocessing import sequence
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing.text import Tokenizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn import svm as svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import re
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
df = pd.read_csv("archive.zip")
df.isnull().sum()
```

```

df.duplicated().sum()
#df.drop_duplicates(subset=['text'], keep='first', inplace=True)
df.rename(columns={'text': 'Text', 'label': 'Label'}, inplace=True)
df.drop('Unnamed: 0',axis=1,inplace=True)
print(f'The Shape Of Data Is : {df.shape}')
df.head()

# Removing URLs, punctuations, whitespaces, numbers
df['Text'] = df['Text'].str.replace(r'http\S+', '', regex=True)
df['Text'] = df['Text'].str.replace(r'^\w\s', '', regex=True)
df['Text'] = df['Text'].str.replace(r'\s+', ' ', regex=True)
df['Text'] = df['Text'].str.replace(r'd+', '', regex=True)
df['Text'] = df['Text'].str.lower()
df['Text'] = df['Text'].apply(lambda x: re.sub(r'^a-zA-Z\s', '', x))
stop = stopwords.words('english')
df["Text"] = df['Text'].apply(lambda x: ' '.join([word for word in x.split() if word not in
(stop)]))

# Display the first few rows of the DataFrame to verify the changes
print(df.head())

df['Text'] = [word_tokenize(sen) for sen in df['Text']]
tag = nltk.pos_tag_sents(df['Text'])
tag[0]

# Helper function for fetching wordnet pos labels
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('R'):
        return wordnet.ADV
    else:

```

```

        return wordnet.NOUN
for i, tokens in enumerate(df['Text']):
    lemmatizer = WordNetLemmatizer()
    processed = []
    for (token, pos) in tag[i]:
        # filter out stop words and non-alphabetic tokens
        lemmatized = lemmatizer.lemmatize(token, get_wordnet_pos(pos)) # PoS is used
    HERE for lemmatization
        processed.append(lemmatized)
    df.loc[i, 'Text'] = " ".join(processed)
# Lets Rename Label also {0: 'sadness', 1: 'joy', 2: 'love', 3: 'anger', 4: 'fear', 5: 'surprise'}
df['Label'] = df['Label'].replace(0, 'Sadness')
df['Label'] = df['Label'].replace(1, 'Joy')
df['Label'] = df['Label'].replace(2, 'Love')
df['Label'] = df['Label'].replace(3, 'Anger')
df['Label'] = df['Label'].replace(4, 'Fear')
df['Label'] = df['Label'].replace(5, 'Surprise')
# Make Seperate Data Set to Visualize text
# Sadness
df_sadness = df[df['Label']=='Sadness']
# Joy
df_joy = df[df['Label']=='Joy']
# Love
df_love = df[df['Label']=='Love']
# Anger
df_anger = df[df['Label']=='Anger']
# Fear
df_fear = df[df['Label']=='Fear']
# Surprise
df_surprise = df[df['Label']=='Surprise']

```

```

# Combine text from different categories
combined_sadness_text = ''.join(df_sadness['Text'])
combined_joy_text = ''.join(df_joy['Text'])
combined_love_text = ''.join(df_love['Text'])
combined_anger_text = ''.join(df_anger['Text'])
combined_fear_text = ''.join(df_fear['Text'])
combined_surprise_text = ''.join(df_surprise['Text'])

# Create word clouds
sadness_wordcloud = WordCloud(width=800, height=400,
background_color='yellow').generate(combined_sadness_text)
joy_wordcloud = WordCloud(width=800, height=400,
background_color='black').generate(combined_joy_text)
love_wordcloud = WordCloud(width=800, height=400,
background_color='purple').generate(combined_love_text)
anger_wordcloud = WordCloud(width=800, height=400,
background_color='red').generate(combined_anger_text)
fear_wordcloud = WordCloud(width=800, height=400,
background_color='green').generate(combined_fear_text)
surprise_wordcloud = WordCloud(width=800, height=400,
background_color='blue').generate(combined_surprise_text)

# Plot the word clouds
plt.figure(figsize=(16, 7))

plt.subplot(2, 3, 1)
plt.imshow(sadness_wordcloud, interpolation='bilinear')
plt.title('Sadness')
plt.axis('off')

```

```

plt.subplot(2, 3, 2)
plt.imshow(joy_wordcloud, interpolation='bilinear')
plt.title('Joy')
plt.axis('off')

plt.subplot(2, 3, 3)
plt.imshow(love_wordcloud, interpolation='bilinear')
plt.title('Love')
plt.axis('off')

plt.subplot(2, 3, 4)
plt.imshow(anger_wordcloud, interpolation='bilinear')
plt.title('Anger')
plt.axis('off')

plt.subplot(2, 3, 5)
plt.imshow(fear_wordcloud, interpolation='bilinear')
plt.title('Fear')
plt.axis('off')

plt.subplot(2, 3, 6)
plt.imshow(surprise_wordcloud, interpolation='bilinear')
plt.title('Surprise')
plt.axis('off')

plt.tight_layout()
plt.show()

# Count the occurrences of each unique label
label_counts = df['Label'].value_counts()

```

```

# Display the label counts
print("Label Counts:")
print(label_counts)

import matplotlib.pyplot as plt
import seaborn as sns

# Define the labels and corresponding counts
labels = ['Sadness', 'Joy', 'Love', 'Anger', 'Fear', 'Surprise']
counts = df['Label'].value_counts()

# Plot a pie chart
plt.figure(figsize=(12, 5))

# Count plot
plt.subplot(1, 3, 3)
sns.countplot(data=df, x='Label', palette='viridis')
plt.title('Count Plot - Class Distribution')
plt.xlabel('Emotion')
plt.ylabel('Count')

plt.tight_layout()
plt.show()

from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler

# Define the resampling strategy (undersampling or oversampling)
# Uncomment one of the following lines based on your choice

# Undersampling
sampler = RandomUnderSampler(sampling_strategy='auto')

```



```

# Oversampling
#sampler = RandomOverSampler(sampling_strategy='auto')

# Apply the resampling strategy to create a balanced dataset
X_resampled, y_resampled = sampler.fit_resample(df[['Text']], df['Label'])

# Create a new DataFrame with the balanced dataset
balanced_df = pd.DataFrame({'Text': X_resampled['Text'], 'Label': y_resampled})

# Print the class distribution in the balanced dataset
print(balanced_df['Label'].value_counts())

import matplotlib.pyplot as plt
import seaborn as sns

# Define the labels and corresponding counts
labels = ['Sadness', 'Joy', 'Love', 'Anger', 'Fear', 'Surprise']
counts = balanced_df['Label'].value_counts()

# Plot a pie chart
plt.figure(figsize=(12, 5))

# Count plot
plt.subplot(1, 3, 3)
sns.countplot(data=balanced_df, x='Label', palette='viridis')
plt.title('Count Plot - Class Distribution')
plt.xlabel('Emotion')
plt.ylabel('Count')

plt.tight_layout()

```

```

plt.show()

df['Label'] = df['Label'].replace('Sadness',0)
df['Label'] = df['Label'].replace('Joy',1)
df['Label'] = df['Label'].replace('Love',2)
df['Label'] = df['Label'].replace('Anger',3)
df['Label'] = df['Label'].replace('Fear',4)
df['Label'] = df['Label'].replace('Surprise',5)

X = balanced_df['Text']
y = balanced_df['Label']

# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Print the shapes of the training and testing sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)

# Initialize TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

# Fit and transform the training data
#tfidf_vectorizer.fit(balanced_df['Text'])
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

# Transform the testing data
X_test_tfidf = tfidf_vectorizer.transform(X_test)
print(X_test_tfidf)

lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train_tfidf, y_train)

```

```

y_pred_lr = lr_model.predict(X_test_tfidf)
accuracy_lr = accuracy_score(y_test, y_pred_lr)
print("Accuracy (Logistic Regression):", accuracy_lr)

# Classification report
print("\nClassification Report (Logistic Regression):")
print(classification_report(y_test, y_pred_lr))

# Initialize the Naive Bayes classifier
naive_bayes_classifier = MultinomialNB()

# Train the classifier on the TF-IDF transformed training set
naive_bayes_classifier.fit(X_train_tfidf, y_train)

# Predict on the test set
y_pred_nb = naive_bayes_classifier.predict(X_test_tfidf)
accuracy_lr = accuracy_score(y_test, y_pred_nb)
print("Accuracy (Navie Bayes):", accuracy_lr)

# Classification report
print("\nClassification Report (Navie Bayes):")
print(classification_report(y_test, y_pred_nb))

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train_tfidf, y_train)
#y_train_pred = rf_classifier.predict(X_train_tfidf)
y_pred_f = model.predict(X_test_tfidf)
accuracy_lr = accuracy_score(y_test, y_pred_f)
print("Accuracy ():", accuracy_lr)

# Classification report
print("\nClassification Report ():")
print(classification_report(y_test, y_pred_nb))

```

```

from sklearn.svm import SVC

svm_model = SVC(kernel='linear', random_state=42)

svm_model.fit(X_train_tfidf, y_train)

y_pred_f = svm_model.predict(X_test_tfidf)

accuracy_lr = accuracy_score(y_test, y_pred_f)

print("Accuracy (SVM):", accuracy_lr)

# Classification report

print("\nClassification Report (SVM):")

print(classification_report(y_test, y_pred_nb))

import joblib

joblib.dump(svm_model, 'model.joblib')

joblib.dump(tfidf_vectorizer, 'vectorizer.joblib')

```

Open.html:

```

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Flask ML Model Deployment</title>

  <style>

    body {

      background-image: url('/static/bg.jpg');

      background-size: cover;

      background-repeat: no-repeat;

      background-clip: border-box;

      color: black;

      display: flex;

      justify-content: center;

      align-items: center;
    }

```

```
height: 100vh;
margin: 0;
font-family: 'Arial', sans-serif;
}
.content {
background:#FFFF00(255, 255, 255, 0.8);
padding: 30px;
border-radius: 10px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
text-align: center;
}
h1 {
font-size: 50px;
font-family: "serif";
margin-bottom: 10px;
}
h2 {
font-size: 30px;
margin-bottom: 20px;
}
textarea {
width: 100%;
max-width: 500px;
height: 200px;
padding: 10px;
border: 2px solid #ccc;
border-radius: 5px;
font-size: 16px;
font-family: 'Arial', sans-serif;
resize: none;
```

```

    }
    input[type="submit"] {
        background-color:blue;
        color: white;
        border: none;
        padding: 15px 30px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 20px 0;
        cursor: pointer;
        border-radius: 5px;
        transition: background-color 0.3s;
    }
    input[type="submit"]:hover {
        background-color: #45a049;
    }
</style>
</head>
<body>
    <div class="content">
        <h1>SENTIMENT: SPECTRUM</h1>
        <h2>Analyzing Emotions in Text</h2>
        <form action="/predict" method="post">
            <textarea id="message" name="message" rows="18" cols="50" placeholder="Enter
your text here..."></textarea>
            <br>
            <input type="submit" value="Predict">
        </form>

```

```
</div>
</body>
</html>
```

Result.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Result</title>
  <script>
    window.onload = function() {
      var prediction = "{{ prediction }}";
      console.log("Prediction:", prediction); // Debug statement

      // Compare the predicted value and navigate to the corresponding page
      if (prediction.trim().toLowerCase() === "sadness") {
        // Navigate to the sadness image page
        window.location.href = "/static/sad.jpg";
      } else if (prediction.trim().toLowerCase() === "joy") {
        // Navigate to the joy image page
        window.location.href = "/static/joy.jpg";
      } else if (prediction.trim().toLowerCase() === "love") {
        // Navigate to the love image page
        window.location.href = "/static/love.jpg";
      } else if (prediction.trim().toLowerCase() === "surprise") {
        // Navigate to the joy image page
        window.location.href = "/static/surprise.jpg";
      } else if (prediction.trim().toLowerCase() === "anger") {
        // Navigate to the love image page
        window.location.href = "/static/anger.jpg";
      }
    }
  </script>
</head>
</html>
```

```

    }else if (prediction.trim().toLowerCase() === "fear") {
        // Navigate to the love image page
        window.location.href = "/static/fear.jpg";
    } else {
        // Handle any other cases or errors
        alert("Unknown prediction!");
    }
}
</script>
</head>
<body>
    <h2>The predicted emotion is: {{ prediction }}</h2>
</body>
</html>

```

Deployment of Front-end and Back-end using Flask.

App.py:

```

import numpy as np

from flask import Flask, render_template, request

from sklearn.linear_model import LogisticRegression

import joblib

app = Flask(__name__)

# Load the trained model

model = joblib.load('model.joblib')

# Load the vectorizer

vectorizer = joblib.load('vectorizer.joblib')

# Home route

@app.route('/')

def home():

```



```

    return render_template('open.html')

# Prediction route

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get the input message from the form
        data = request.form['message']

        data = data.lower() # Convert to lowercase

        # Transform the input text using the vectorizer
        data_vectorized = vectorizer.transform([data])

        # Predict using the logistic regression model
        prediction = model.predict(data_vectorized)[0]

        print("Received form data:", data)
        print("Prediction:", prediction)

        return render_template('result.html', prediction=prediction)
    except Exception as e:
        print("Error during prediction:", e) # Debugging statement

        return render_template('result.html', prediction=f'Error: {e}')

if __name__ == '__main__':
    app.run(debug=True)

```

5.2 Test Cases

Test Case ID	Test Case	Test Data	Expected Output	Actual Output	Status
1.	I feel so stupid that I realized it so late	Text	Sad	Sad	Pass
2.	I am feeling really out of place and irritated	Text	Angry	Angry	Pass
3.	I am feeling really out of place	Text	Sad	Sad	Pass
4.	I am feeling nostalgic about the place	Text	Love	Love	Pass
5.	I told him that maybe I just need time to think how I have been feeling indecisive about things lately	Text	Fear	Fear	Pass
6.	I was feeling an act of god at work in my life and it was an amazing feeling	Text	Surprise	Surprise	Pass
7.	I feel like im at the spa getting a wonderful facial when I use them	Text	Joy	Joy	Pass

5.3. Result

Input:



Fig.1.18. input

Output:



Fig.1.19.output

Input:

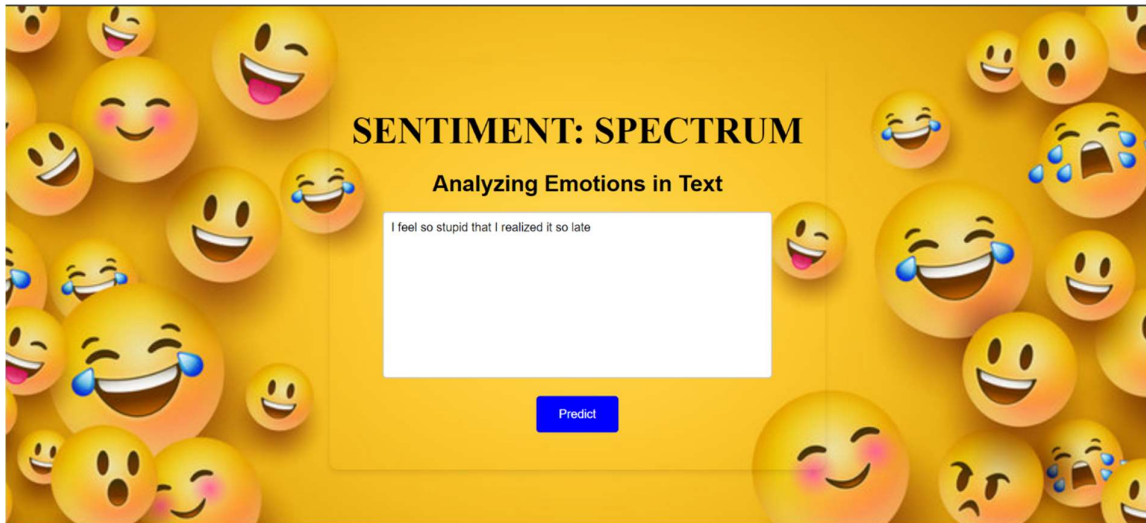


Fig.1.20. input

Output:

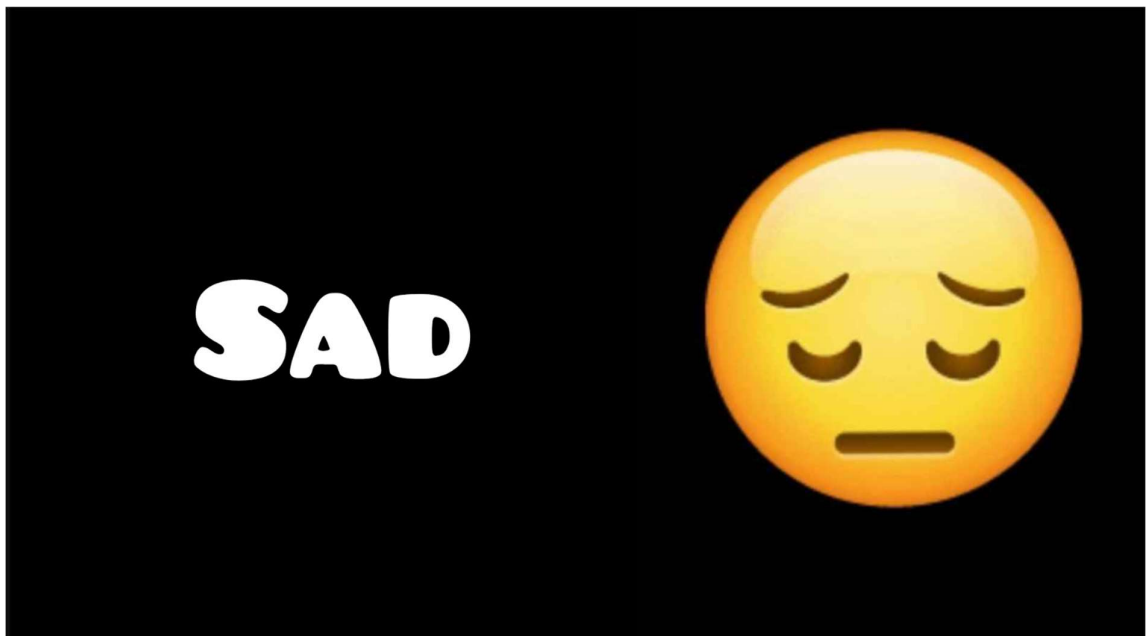


Fig.1.21.output

CHAPTER 6

CONCLUSION AND FURTHER SCOPE

In our project, we have used 4 machine Learning algorithms among which SVM has achieved high accuracy of 89.5%. In the code we also used preprocessing techniques like tokenization, lemmatization, stop word removal, and to handle imbalance data we used under sample method, TF-IDF vectorization is used for feature extraction. The SVM algorithm's capability to create optimal hyperplanes in multidimensional feature spaces proved effective in maximizing the margin between different classes, this will increase the model's ability to generalize to unseen data. We can also observe that it is efficient for multi class labels.

Additional enhancements: feature selection, tuning of hyperparameters, using grid search, random search. This will be a Bayesian optimization approach to find the best performing combination or approach of cross-validation evaluation for improving the model performance, to utilize advanced ML learning techniques like deep learning (neural networks) like RNN (recurrent neural network), LSTM(long short-term memory network),CNNs (convolutional neural networks) for better classification of the labels.

Overall, our project successfully Highlights the potential of sentiment analysis in extracting the emotions in the textual data. By leveraging these insights, organizations can make data-driven decisions to enhance their offerings and better meet customer expectations, Continuous improvement and adaptation of methodologies will further strengthen the reliability and applicability of sentiment analysis in various domains.

CHAPTER 7

REFERENCES

- [1] J. Li, S. Fong, Y. Zhuang, and R. Khoury, “ Hierarchical Classification in Text Mining for Sentiment Analysis,” in 2014 International Conference on Soft Computing and Machine Intelligence, september. 2014, p. 46-51,doi: 10.1109/ISCMI.2014.37.
- [2] H. Parveen and S. Pandey,“Sentiment analysis on Twitter Data-set using Naive Bayes algorithm,” in 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), july. 2016, p. 416-419, doi:10.1109/ICATCCT.2016.7912034.
- [3] S. S. and P. K.v., “ Sentiment analysis of malayalam tweets using machine learning techniques,”ICT Express, april. 2020, doi:10.1016/j.icte.2020.04.003.
- [4] R. A. Tuhin, B. K. Paul, F. Nawrine, M. Akter, and A. K. Das, “ An Automated System of Sentiment Analysis from Bangla Text using Super-vised Learning Techniques ,” in 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), february. 2019, p.360-364, doi: 10.1109/CCOMS.2019.8821658.
- [5] A. Poornima and K. S. Priya, “ A Comparative Sentiment Analy-sis Of Sentence Embedding Using Machine Learning Techniques ,”in 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), march 2020, p. 493-496, doi:10.1109/ICACCS48705.2020.9074312.
- [6] H. S and R. Ramathmika, “ Sentiment Analysis of Yelp Reviews by Machine Learning ,” in 2019 International Conference on Intelligent Computing and Control Systems (ICCS), may 2019, p. 700-704, doi:10.1109/ICCS45141.2019.9065812.
- [7] R. B. Shamantha, S. M. Shetty, and P. Rai, “Sentiment Analysis Using Machine Learning Classifiers: Evaluation of Performance ,” in 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), f`evr. 2019, p. 21-25, doi:10.1109/CCOMS.2019.8821650.

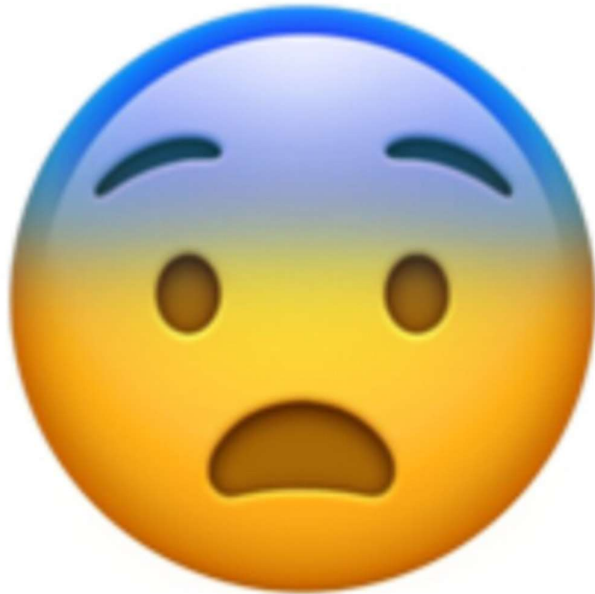
Snapshots of the result:

JOY



Prediction of joy

FEAR



Prediction of fear

SURPRISE



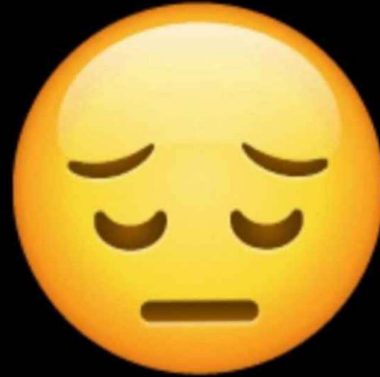
Prediction of Surprise

ANGER



Prediction of Anger

SAD



Prediction of Sadness

LOVE



Prediction of Love

PAPER NAME

d5.pdf

AUTHOR

D5 Griet

WORD COUNT

4748 Words

CHARACTER COUNT

26115 Characters

PAGE COUNT

45 Pages

FILE SIZE

4.5MB

SUBMISSION DATE

Jul 8, 2024 1:16 PM GMT+5:30

REPORT DATE

Jul 8, 2024 1:17 PM GMT+5:30**● 15% Overall Similarity**

The combined total of all matches, including overlapping sources, for each database.

- 13% Internet database
- 2% Publications database
- Crossref Posted Content database
- 9% Submitted Works database

● Excluded from Similarity Report

- Crossref database
- Bibliographic material
- Small Matches (Less than 8 words)
- Manually excluded text blocks

● 15% Overall Similarity

Top sources found in the following databases:

- 13% Internet database
- 2% Publications database
- Crossref Posted Content database
- 9% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	researchgate.net Internet	2%
2	ijarse.com Internet	1%
3	kluniversity.in Internet	1%
4	bokut.in Internet	<1%
5	researchsquare.com Internet	<1%
6	University of Queensland on 2023-06-04 Submitted works	<1%
7	coursehero.com Internet	<1%
8	cse.griet.ac.in Internet	<1%
9	javacodegeeks.com Internet	<1%

[Sources overview](#)

10	Indian Institute of Technology, Kharagpure on 2014-09-02 Submitted works	<1%
11	Alkhaldi, Amal Masoud Alnazawi, Amjad Fuad Alanazi, Ebtisam Mas... Publication	<1%
12	Indian School of Business on 2019-08-30 Submitted works	<1%
13	es.slideshare.net Internet	<1%
14	於2012-05-08提交至Pathfinder Enterprises Submitted works	<1%
15	dspace.daffodilvarsity.edu.bd:8080 Internet	<1%
16	carpentries-incubator.github.io Internet	<1%
17	export.arxiv.org Internet	<1%
18	repositorio-aberto.up.pt Internet	<1%
19	dokumen.pub Internet	<1%
20	frontiersin.org Internet	<1%
21	Shanghai Jiaotong University School of Medicine on 2024-01-12 Submitted works	<1%