Dissertation on

Detection of Emotion from Textual Data using Machine Learning - An Approach

Thesis submitted towards partial fulfilment of the requirements for the degree of

Master of Technology in IT (Courseware Engineering)

Submitted by

Bikram Sarkar

EXAMINATION ROLL NO.: M4CWE24002
UNIVERSITY REGISTRATION NO.: 163770 0f 2022-2023

Under the guidance of

Mr. Joydeep Mukherjee

School of Education Technology
Jadavpur University

Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India
2024

M.Tech.IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **“Detection of Emotion from Textual Data using Machine Learning - An Approach”** is a Bonafide work carried out by **Bikram Sarkar** under our supervision and guidance for partial fulfilment of the requirements for the degree of **Master of Technology in IT (Courseware Engineering)** in **School of Education Technology**, during the academic session 2023-2024.

SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN - FISLM
Jadavpur University,
Kolkata-700 032

M.Tech.IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

Committee of final examination ---
--- for evaluation of the Thesis

** Only in case thesis is approved

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his **Master of Technology in IT (Courseware Engineering)** studies.

All Information in this document has been obtained and presented in accordance With academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME	:	BIKRAM SARKAR
EXAMINATION ROLL NUMBER	:	M4CWE24002
THESIS TITLE	:	Detection of Emotion from Textual Data using Machine Learning- An Approach

SIGNATURE

DATE

Acknowledgement

Firstly, I would like to express my sincere gratitude to my supervisor **Mr. Joydeep Mukherjee** for his continuous support in my thesis work and related research. His guidance and valuable suggestions always helped me at time of research and writing this thesis. I could not have imagined having a better advisor and mentor for my Master Degree study. I am very much thankful to him for the motivation and support given during the entire duration of research work.

Besides this, I would like to thank **Prof. Dr. Matangini Chattopadhyay** and **Dr. Saswati Mukherjee** for their continuous encouragement and support during my entire period of study in the School of Education Technology. I would like to express my gratitude to the entire staff, Lab Assistant and all those who were associated with my work for their co-operation and opinion. I am very grateful to all my classmates of M.Tech.IT (Courseware Engineering) and also Master in Multimedia Development courses for their suggestions and continuous support.

I would like to thank my parents for always supporting me in all ups and downs during my entire period of work.

I am also thankful to my friends and well- wishers who always have faith in me.

Bikram Sarkar
Examination Roll No.-M4CWE24002
Registration No-163770 of 2022-23
M.Tech.IT (Courseware Engineering)
School of Education Technology
Jadavpur University, Kolkata-700032

DEDICATED TO,
My Parents

CONTENT

List of Figures

List of Abbreviations

Abbreviations	Full Form
BNB	Bernoulli Naïve Bayes
GNB	Gaussian Naïve Bayes
MNB	Multinomial Naïve Bayes
SVC/SVM	Support Vector Machine
NB	Naïve Bayes
CV	Count Vectorizer
TF-IDF	Term Frequency- Inverse Document Frequency
NLTK	Natural Language Tool Kit
NLP	Natural Language Processing

List of Tables

List of Publications

1. Bikram Sarkar and Joydeep Mukherjee**, "EMOTION DETECTION FROM TEXT USING MACHINE LEARNING ALGORITHM"**, International Journal of Multidisciplinary Educational Research, Volume: 13, Issue:4(3), April 2024.

   DOI: http://ijmer.in.doi./2024/13.4.54

   Link:http://s3-ap-southeast-1.amazonaws.com/ijmer/pdf/volume13/volume13-issue4(3)/14.pdf

2. Bikram Sarkar and Joydeep Mukherjee, **"IMPROVEMENT OF FEATURE ENGINEERING ON EMOTION DETECTION FROM TEXTUAL DATA"** World Journal of Advanced Engineering Technology and Sciences, Volume 12 - Issue 1 (May - June 2024)

   DOI: https://doi.org/10.30574/wjaets.2024.12.1.0171

   Link: https://wjaets.com/sites/default/files/WJAETS-2024-0171.pdf

Executive Summary

Emotion detection from text has emerged as a critical area of research, given the vast amount of textual data generated daily on social media platforms, customer reviews, and other digital sources. Understanding emotions expressed in text has implications for various domains, including marketing, customer service, mental health, and sentiment analysis. This thesis aims to investigate the effectiveness of machine learning techniques in detecting and analyzing emotions from textual data. Specifically, the research seeks to develop a robust model capable of accurately identifying and categorizing emotions expressed in text. The methodology comprises data collection, data preprocessing, Feature Engineering, Model Development and Evaluation Process. In this Proposed Method Get () Sentiment Model is more accurate and effective to detect Sentiment from textual data as compare to previously used N-grams, BERT, Word Cloud etc. and solves word ambiguity problems as well. To check performance of system various machine learning algorithms like Logistic Regression, Naïve Bayes Classifier, Support Vector Machine, Random Forest, KNN and also unsupervised Machine Learning model like K-Means Clustering. Bernoulli and Gaussian Naïve Bayes algorithms never used for text classification but two Methods have been used to get better results. The performance of the proposed model with previously used model is compared. Experimental results demonstrate the effectiveness of machine learning models in detecting emotions from text.

ISEAR dataset has been used for this study and were tested all models using ISEAR standards criteria. The dataset contained different form of Text message, comments and reviews. For pre-cleaning process of dataset, all unrelated attributes have been ignored. Unrelated attributes things can make confused during analysis phase. But all necessary actions have been taken related to datasets during the time of data cleaning process. Every cycle of this research different actions, modifications have been taken as per need time to time. In the existing models have various kind of problems which have been found and solve some vital problems. Also Used new models that have been succeed to provide better results related to accuracy and other things. used to this study various machine learning techniques like Naïve bayes Classifier, Logistic Regression, Linear Regression, Random Forest Classifier, Gradient boost Classifier etc. Actually, various factor can affect our sentiment analysis accuracy, word ambiguity is one of them and a very serious problem. Previously various machine learning algorithms failed to provide very good results. This study one of its kind proposed systems are more accurate compare to previous and will help in the field of natural language processing in the context of emotion detection and sentiment analysis.

CHAPTER 1

1. INTRODUCTION

1.1 Overview

Emotion detection will play a promising role in the field of Human Computer Interaction and interface development. In today’s world, a majority of users have access to internet for exchange communication via text, Image, audio and video. Emotion can be expressed by a person’s speech, face expression and also written text known as Text based emotion. With the growing population in countries like India, it has led to tremendous growth in the number of users using Facebook, WhatsApp, Twitter, Instagram and also online shopping app like Amazon, Flipkart, Paytm Mall, Messo etc. The large scale of these users or people are providing feedback, asking any query related to online networks and related things via Text message now a days in the current Technological world. Researchers collected all text message from various online platforms and analysis accordingly to make them more efficient and more compact. But it has become a challenge in recent years to extract valuable information from these growing data in the form of posts, emails, blogs, tweets, revies, comments, surveys on the Web in the process of any decision making. It is very much difficult to detect emotion from Text because human mind is so complex and even dataset might not be ready for the proposed research. But to simply Two affect class has been categorized in three categories like positive, negative and neutral. Recently, researchers have proposed various methods for text emotion detection including keyword-based method, learning-based and hybrid models, lexical similarity models. At first, they introduced a rule-based approach such as lexical affinity based and keyword based. Then came a new approach, the learning-based approach. This method was more accurate and gave better results as expected. Many researchers begun to combine these approaches named hybrid model to get better accuracy they got more better accuracy but they failed to get very good accuracy as expected that can make this research more efficient.

Then came the name of new techniques such as machine learning based models to detect emotion and classify the all affect class one by one followed by getting very good accuracy as expected. Researches have used some machine learning models to classify the emotions and sentiments and also improve existing model’s accuracy. But also used machine learning models have some very crucial problems and some models also have been failed to detect emotion more accurately and to provide good accuracy results as per need.

In this study, the new methods have been proposed followed by solving the problems to the existing models, datasets and has been improved accuracy and got very good result.

1.2. Problem Statement

Detection of Emotion from Textual Data using Machine Learning

1.3. Objectives

The objectives of my research work are as follows:

- ✓ To find more efficient Machine Learning algorithms to detect Emotion from textual data for this study to get better results.
- ✓ To find out problems in dataset, data preprocessing phase and model Development phase.
- ✓ To find the problems in used models and how to solve the problems and improve accuracy of existing models.
- ✓ To solve the problem of word ambiguity in text messages.
- ✓ To Find best feature Extraction method for this study.
- ✓ To develop more accurate Sentiment detection algorithm.

1.4 Assumptions and Scopes

1.4.1 Assumptions

- Developers must have good network connection with laptops or PC.
- Developers must have a proper system setup like Google Colab, Jupyter Notebook already installed in PC.
- Developers must have good knowledge of Python, Data analysis and machine Learning for programmatic access.

1.4.2 Scopes

- To do emotion detection from text message such CSV. files
- To learn about Python and Machine Learning

1.5 Concept and Problem Analysis

Emotion detection involves identifying and categorizing emotions expressed by users through different modalities such as text, speech, and facial expressions. In the context of Human-Computer Interaction (HCI), this capability can enhance user experience by allowing systems to respond appropriately to users' emotional states. Detecting emotion from text is difficult due to the complexity of the human mind and the potential inadequacy of existing datasets for proposed research.

To simplify the task, emotions have been categorized into three classes: positive, negative, and neutral. Recently, researchers have proposed various methods for text emotion detection, including keyword-based methods, learning-based methods, hybrid models, and lexical similarity models. Initially, rule-based approaches such as lexical affinity-based and keyword-based methods were introduced. Subsequently, a learning-based approach was developed, which proved to be more accurate and yielded better results.

Researchers began combining these approaches into hybrid models to achieve higher accuracy. Although these models showed improved accuracy, they still fell short of the high expectations necessary for making the research more efficient. Machine learning-based models emerged as a new technique for detecting and classifying emotions, providing significantly improved accuracy. Researchers employed various machine learning models to classify emotions and sentiments, thereby enhancing the accuracy of existing models. However, these models also faced critical challenges and some failed to detect emotions accurately, resulting in suboptimal performance.

In this research, new methods are proposed to address the problems of existing models and datasets, ultimately improving accuracy and achieving impressive results. One notable issue at the document level is the expression of multiple emotions within the same document. Based on analysis and studies from previous research, several limitations have been identified:

- Handling the complexity of the human mind and the subtleties of text-based emotion.
- Ensuring datasets are comprehensive and representative of diverse emotional expressions.
- Achieving high accuracy in emotion detection despite the presence of multiple emotions in a single document.
- Developing models that can reliably and accurately classify emotions in various contexts and platforms.

By addressing these challenges, the research aims to advance the field of text-based emotion detection and enhance the capabilities of Human-Computer Interaction systems. Some problem of document-level occurs when multiple emotions are expressed in the same document. By analysis and study from previous research below are some limitations:

a) Word ambiguity.

b) Lack of linguistic Information.

c) In capability to recognize emotion in absence of emotion keyword.

d) Existing models not providing better accuracy.

e) Existing emotion detection- based Algorithm is not more efficient.

f) Some machine learning Technique’s overfitting Problem.

g) Different types of same algorithm (Naïve bayes) giving different results.

h) Multinomial Naïve Bayes Algorithm failed to perform well.

1.6 Organization of the Thesis

I. Chapter 1 – This chapter contains the introduction of the thesis which includes overview, problem statement, objectives, assumptions, scopes, concept and problem analysis.

II. Chapter 2 – It includes all the literature surveys done to carry out the research work.

III. Chapter 3 – It includes proposed approach that has been used to detect Emotion from text.

IV. Chapter 4 – This chapter contains the implementation and result.

V. Chapter 5 – This chapter contains the comparative analysis.

VI. Chapter 6 – This chapter describes the conclusion and scope of future scopes.

VII. References – All the references are given here.

VIII. Appendix Part A – This part contains system requirements, software requirements, programming requirements and download speed.

IX. Appendix Part B – Here are all the code snippets provided.

CHAPTER 2

2. LITERATURE SURVEY

Vishaka Singh et al. [1], proposed an Emotion Detection Model taking two different Feature extraction Model Term Frequency- Inverse Frequency (TFDF) and Count Vectorizer using Logistic Regression, Random Forest Classifier models etc. here researchers consider 20000 dataset for the study and applying models on preprocessed data the highest accuracy obtained in Count Vectorizer using Logistic Regression 88% in the case of Data set Split Ratio is 70:30 but more need to improve on preprocessing Phase on Textual Dataset and Consider more data as well.

Ms. Pinal Solanki [2], considered a small Data set for this research, the proposed study found that SVC and TFDF is more accurate and SVC gave highest accuracy, focuses on feature extraction method and word recognition for getting better results but word ambiguity problems were not considered here. During this study researchers faced some problems such text is commonly displayed unclear, some sentences may be sarcastic and sentiment is unclear due to the presence of multiple points of the view on the subjects

Firdaus et al. [3] focus on the application of text emotion detection for retweet prediction, a task crucial in social media analytics. The paper proposes a topic-specific approach to emotion detection, leveraging machine learning techniques. By associating emotions with specific topics, the authors aim to enhance the accuracy of retweet prediction models. This study highlights the practical applications of text emotion detection beyond sentiment analysis, demonstrating its utility in social media data analysis and prediction tasks.

Oliveira et al. [4], proposed Generalized Linear Model with taking 1000 features total of 2302 features sets were explored, where each features sets has 100-1000 features extracted from the Text. The results demonstrate Generalized Linear Model provides the best Accuracy score (0.92), Recall (0.902), Precision (0.902), F1 score (0.901) with standard deviation of accuracy of ±1,2%.

Shaikh Abdul et al. [5] proposed machine learning based sentiment detection model using Naïve Bayes Algorithm, Support Vector Machines, K-means Clustering. For this study the twitter data was required and converted from word to vector of Eight emotions and applied feature extraction techniques to get better classification accuracy. According to this research 13000 data used and experiment resulted that support vector machines accuracy was 80%, Naïve Bayes models Accuracy was 50%. It evident that SVC and Naïve bayes far better that K-means but the accuracy of every model needs to be improved as well. Hence Pre-processing data still remains one of the most crucial streps which needs to be improved to get more accurate results.

P Ancy et al. [6] approached a rule-based emotion detection model involves various NLP Process for classification. Automatic Classification Approach Supervised Machine Learning Models like Naïve Bayes Algorithm, SVC, Linear Regression and Unsupervised Machine Learning approach is used to explore data but got bad results due to various problems in Data set. In case of supervised Machine Learning Models highest accuracy obtained.

Amal Shameem et al. [8] proposed machine learning based sentiment detection model using Decision Tree Classifier Support Vector Machines. For this study the twitter data was required and converted from word to vector of Eight emotions and applied feature extraction techniques to get better classification accuracy. Decision Tree Classifier has the best average performance in terms of efficiency, sensitivity and f1score at 84.7%, 74.2%, and 94.1% respectively. Throughout this study Researchers work on the to identify emotions based on text., SVC, Nested Linear SVC methods can be used to identify emotions in multiclass based on the results of the discussion and evaluation conducted in the previous section. Random Forest Classifier has the best accuracy. The experimental findings demonstrated that machine learning-driven text emotion classification outperforms established learning methodologies, exhibiting notably superior accuracy rates.

Poonam Arya et al. [10] proposed a hybrid model that incorporates natural language processing technique, including keyword-based and machine learning-based emotion classification from textual data at sentence level. Supervised and unsupervised technique has been used. Limitations of this study are Word ambiguity, Incapability to recognize emotion in absence of emotion keyword, Emotion categories. Machine Learning Model provided 63% accuracy compared to Keyword Based Model.

Goru Swathi et al. [11] developed an emotion recognition system for text-based content. The proposed model is a combination of machine learning approaches. According to the observation Logistic regression gives highest accuracy of 84% as compared to KNN, SVM, Naïve Bayes, Decision Tree. But this research can be extended by making a real-time test-based emotion recognition system.

S. Arun Kumar S. et al. [12] The study delved into algorithms for identifying emotions from textual data and detecting emotional cues within the text. These approaches are combination of machine learning and CNN, here considered only Machine learning Algorithms. Besides NRC Lex, NLP method also considered for this study their accuracy 64.44, 83.36 respectively.

Garg and Saxena [16] proposed a machine learning-based approach for emotion detection and human behavior analysis. Garg and Saxena employed machine learning techniques for emotion detection from text data. They utilized computational intelligence methods for sentiment analysis, emphasizing the importance of accurate emotion classification for understanding human behavior.

Bhavya A.V. et al. [18] proposed an AI based machine Learning Model. Emotion detection from text has garnered significant attention due to its wide-ranging applications in various fields, including healthcare, customer service, and social media analysis. However, existing emotion detection models often overlook the personalized nature of emotions, leading to suboptimal performance in capturing individual nuances. In this paper, we propose a novel approach for personalized emotion detection from text using machine learning techniques. Present research worker’s approach leverages user-specific data to tailor emotion detection models to individual users, thereby enhancing the accuracy and effectiveness of emotion classification. Present research worker conducts experiment on a diverse dataset collected from social media platforms, demonstrating the superiority of our personalized approach over traditional methods. The results highlight the importance of considering individual differences in emotion expression for achieving more accurate emotion detection from text.

Nath, S., Shahi, et al [19] recognized an emotion Detection Model emotion recognition is a crucial task with applications in various domains such as human-computer interaction, affective computing, and mental health assessment. In this paper, they present a comparative study on SER utilizing machine learning techniques. Present research worker's study investigates the performance of different machine learning algorithms and feature extraction methods for recognizing emotions from speech signals. They conduct experiments on benchmark datasets, evaluating the accuracy, robustness, and computational efficiency of the proposed approaches. Through comprehensive analysis and comparison, they identify the strengths and weaknesses of each method, providing insights into the most effective strategies for SER tasks. Their findings contribute to advancing the state-of-the-art in speech emotion recognition and offer valuable guidance for researchers and practitioners in this field.

CHAPTER 3

3. PROPOSED APPROACH

In the proposed system, various supervised machine learning models have been used such as Naïve Bayes, Support Vector Machine, Random Forest Classifier, KNN and Logistic Regression. Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Gaussian Naïve Bayes, Compliment Naïve Bayes algorithms, this different type of Naïve Bayes Model will be used for this Study. For this study 34791 records of Text messages are collected. After the Classifiers are trained, Text Data can be fed into them to determine the emotion type. Other side for detecting Sentiment, get () sentiment Algorithm has been used and When model was ready, deployed on the Preprocessed clean dataset to detect sentiment. 'Neutral', 'Positive', 'Negative' three type of sentiments were detected by applying our proposed sentiment detection model. The documents were calculated on the basis of the confusion Matrix. Confusion Matrix has calculated four different table with true positive (TP), true negative (TN), false positive (FP) and false negative (FN). Two different more accurate feature extraction techniques namely TF-IDF, Count Vectorizer are used to get best classification accuracy as compared to previous research.

3.1 Dataset Description:

In this work, Textual data set collected from various resources as Natural language Toolkit (NLTK) corpus, Kaggle and ISEAR (International survey of Emotional Antecedents and Responses). The data set contain 34791 rows and 4 columns with about 34791 records of different tweet or text messages. These ISEAR dataset is sentiment label dataset where data is lightly cleaned and normalized. But as per research need dataset has been cleaned and more improved in preprocessing phase so that accuracy can be increased as well as model become more accurate for the research purpose. Dataset contained four different features such as Emotion, input, Text, Clean Text and after preprocessing preprocessed text included with dataset. After detection of sentiment this part also includes with Dataset and save the dataset again for final evaluation process. Eight emotion Classes has been Considered such as ‘Joy’, ‘Sadness’, ‘Fear’, ‘Anger’, ‘Neutral’, ‘Surprise’, ‘Shame’, ‘disgust’ and no of records and percentages are described in the following table.

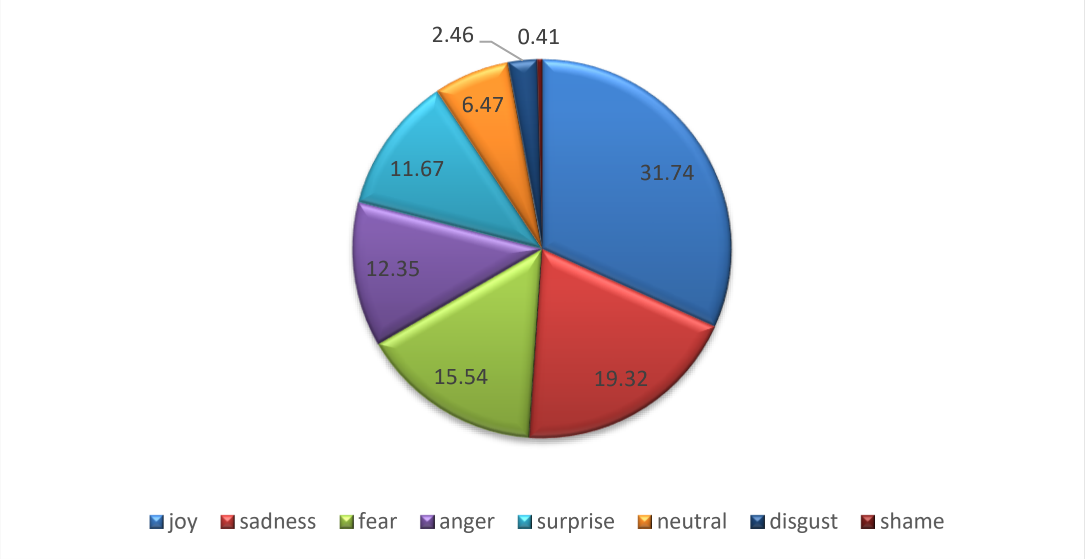
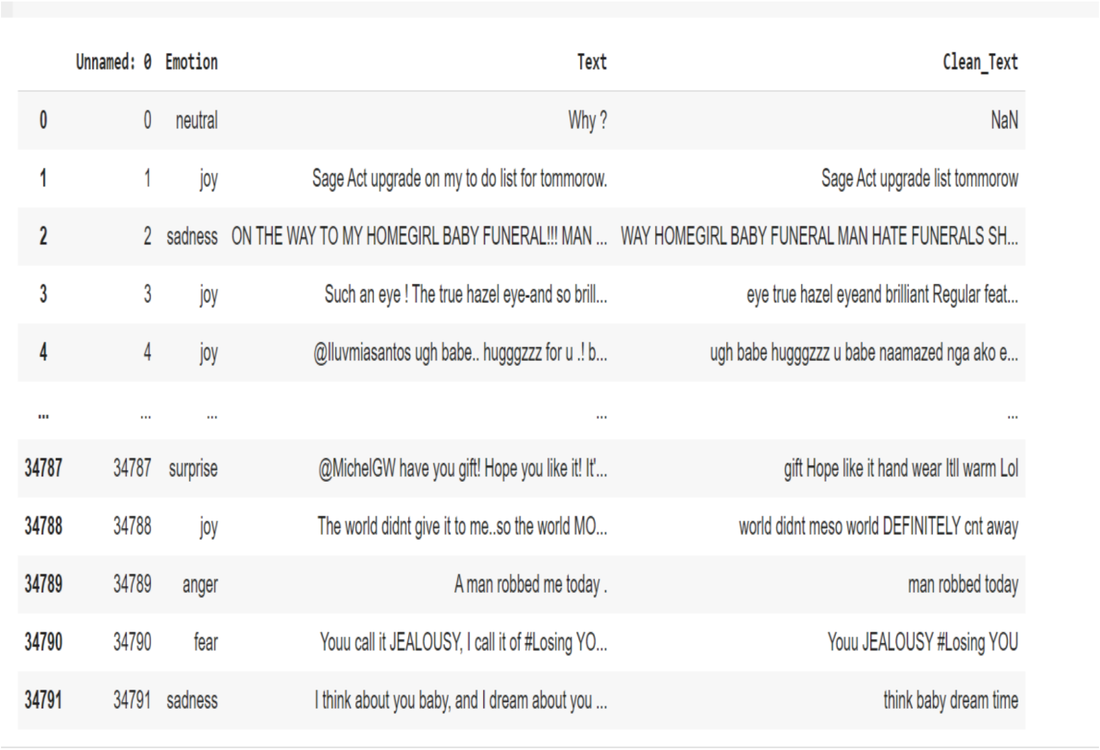


Fig-1: Percentage of Emotions

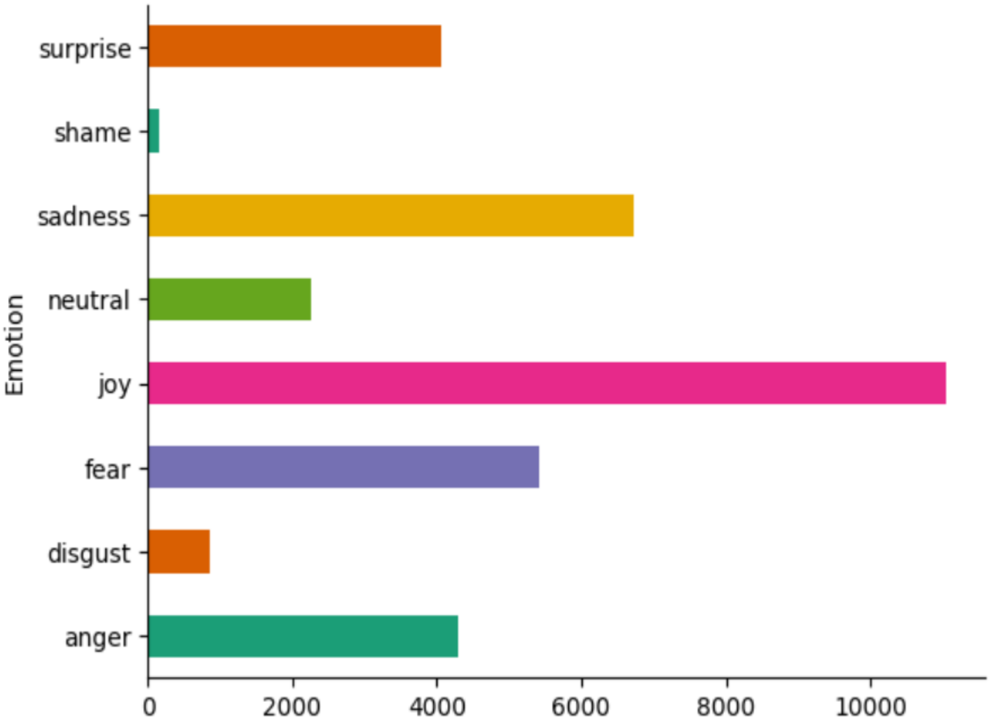
Table-1: No of Records and Percentage of Emotion

Emotion	No of Records	Percentage
Joy	11045	31.745804
Sadness	6722	19.320533
Fear	5410	15.549552
Anger	4297	12.350540
Neutral	2254	6.478501
Surprise	4062	11.675098
Shame	146	0.419637
disgust	856	2.460336
Total	34791	



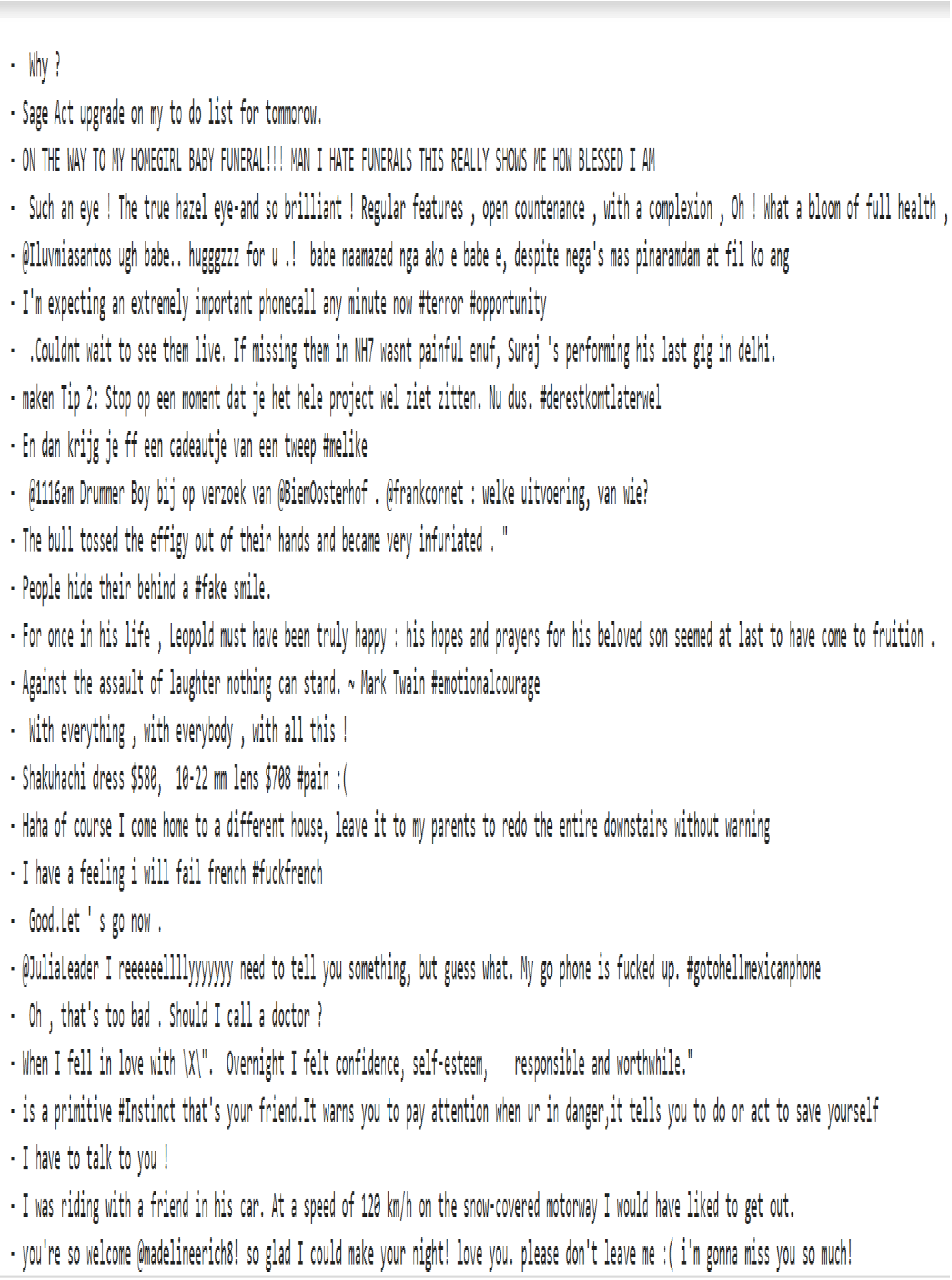
	Unnamed: 0	Emotion	Text	Clean_Text
0	0	neutral	Why ?	NaN
1	1	joy	Sage Act upgrade on my to do list for tommorow.	Sage Act upgrade list tommorow
2	2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL MAN HATE FUNERALS SH...
3	3	joy	Such an eye ! The true hazel eye-and so brill...	eye true hazel eyeand brilliant Regular feat...
4	4	joy	@lluvmiasantos ugh babe.. hugggzzz for u .! b...	ugh babe hugggzzz u babe naamazed nga ako e...
...	...	...	...	...
34787	34787	surprise	@MichelGW have you gift! Hope you like it! It'...	gift Hope like it hand wear Itll warm Lol
34788	34788	joy	The world didnt give it to me..so the world MO...	world didnt meso world DEFINITELY cnt away
34789	34789	anger	A man robbed me today .	man robbed today
34790	34790	fear	Youu call it JEALOUSY, I call it of #Losing YO...	Youu JEALOUSY #Losing YOU
34791	34791	sadness	I think about you baby, and I dream about you ...	think baby dream time

Fig-2: Dataset Description



No of Records

Fig-3: Distribution of Emotions



```
·  Why ?
· Sage Act upgrade on my to do list for tommorow.
· ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN I HATE FUNERALS THIS REALLY SHOWS ME HOW BLESSED I AM
·  Such an eye ! The true hazel eye-and so brilliant ! Regular features , open countenance , with a complexion , Oh ! What a bloom of full health ,
· @Iluvmiasantos ugh babe.. hugggzzz for u .!  babe naamazed nga ako e babe e, despite nega's mas pinaramdam at fil ko ang
· I'm expecting an extremely important phonecall any minute now #terror #opportunity
·  .Couldnt wait to see them live. If missing them in NH7 wasnt painful enuf, Suraj 's performing his last gig in delhi.
· maken Tip 2: Stop op een moment dat je het hele project wel ziet zitten. Nu dus. #derestkomtlaterwel
· En dan krijg je ff een cadeautje van een tweep #melike
·  @1116am Drummer Boy bij op verzoek van @BiemOosterhof . @frankcornet : welke uitvoering, van wie?
· The bull tossed the effigy out of their hands and became very infuriated . "
· People hide their behind a #fake smile.
· For once in his life , Leopold must have been truly happy : his hopes and prayers for his beloved son seemed at last to have come to fruition .
· Against the assault of laughter nothing can stand. ~ Mark Twain #emotionalcourage
·  With everything , with everybody , with all this !
· Shakuhachi dress $580,  10-22 mm lens $708 #pain :(
· Haha of course I come home to a different house, leave it to my parents to redo the entire downstairs without warning
· I have a feeling i will fail french #fuckfrench
·  Good.Let ' s go now .
· @JuliaLeader I reeeeeelllllyyyyyyy need to tell you something, but guess what. My go phone is fucked up. #gotohellmexicanphone
·  Oh , that's too bad . Should I call a doctor ?
· When I fell in love with \X\".  Overnight I felt confidence, self-esteem,   responsible and worthwhile."
· is a primitive #Instinct that's your friend.It warns you to pay attention when ur in danger,it tells you to do or act to save yourself
· I have to talk to you !
· I was riding with a friend in his car. At a speed of 120 km/h on the snow-covered motorway I would have liked to get out.
· you're so welcome @madelineerich8! so glad I could make your night! love you. please don't leave me :( i'm gonna miss you so much!
```

Fig-4: Sample of Text in Dataset

3.2 Data Pre-processing

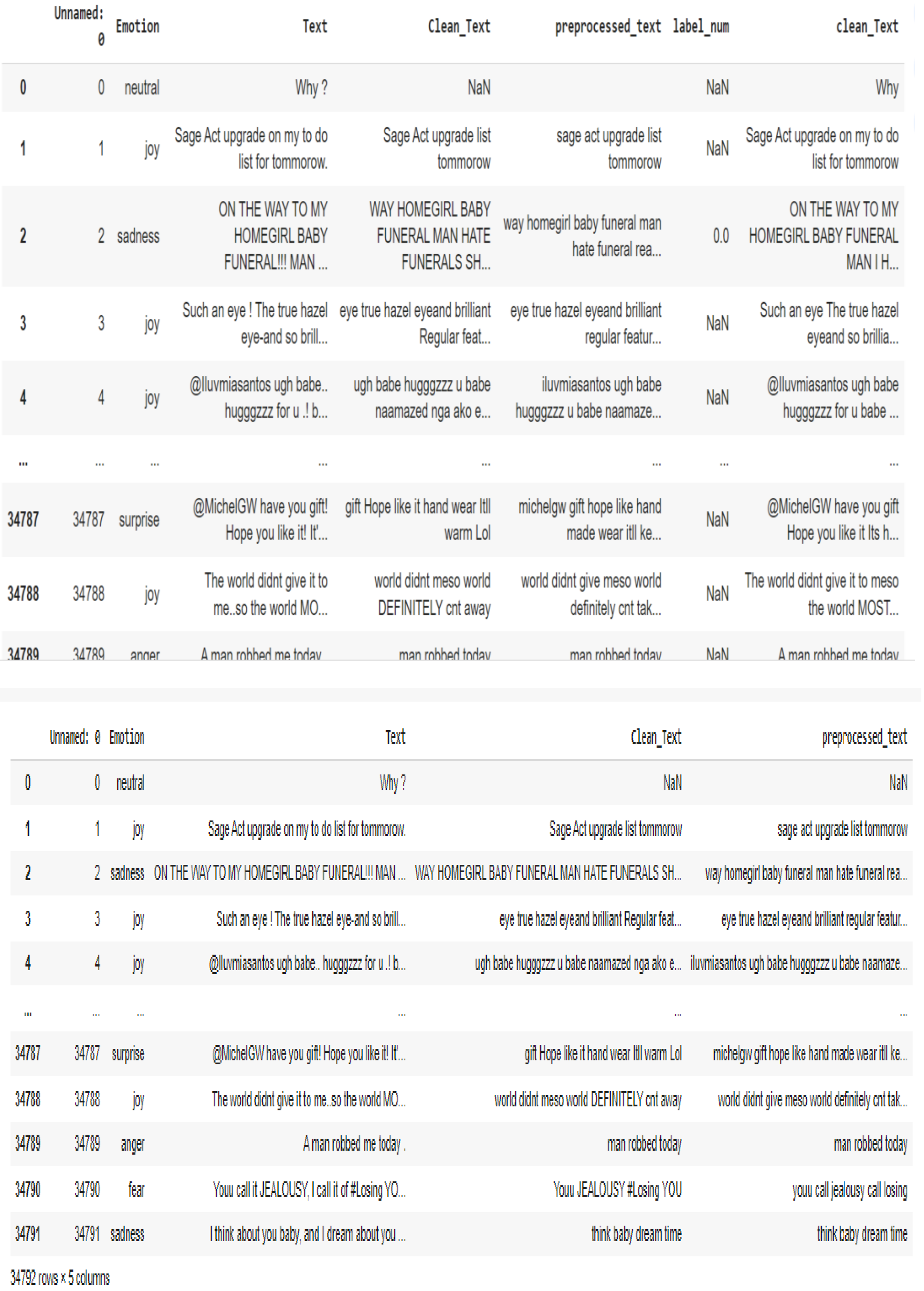
Preprocessing refers to the transformations applied to the data before providing the data to algorithms. This process is used to convert the raw data in to an understandable dataset. How ever dataset collected from Previous researcher is cleaned almost useable to apply using method but present research worker's research objectives are to gain more better results as compared to previous. In other Words, to get best accurate classification accuracy it is necessary to have dataset fully cleaned and normalized. Text processing is a technique to clean to the text data and make it ready to feed data to the model. Previous researcher taken 20000 data from 34791 data in dataset and in their preprocessing stage they cleaned data as their research need. However, there are many problems in the dataset that are found and solved in this phase and the data was adapted to run the algorithm to obtain more accurate results. The problems are as follows:

- Null Values: In the dataset There were 466 null values that could cause problems for the models, so these key values were dropped.
- Missing Values: There were some missing values in the dataset which were found and replaced with mean values to provide better accuracy.
- Duplicate Counts: There were no duplicate words, duplicate columns and rows.
- Removal of Stop Words.
- Removal of User_handels.
- Removal of non-English Words.
- Removal of special characters and digits.
- Removal of Punctuations.
- Tokenization: It is a process of splitting a string, text into lists of tokens.
- Find most common Keywords from every type of emotion so that machine can understand properly each keyword and their places.

- Also checked for data inconsistency, float value conversion timing, normalized
- Data Vectorization: Data vectorization refers to the process of converting raw data, such as text or images, into numerical vectors that can be understood and processed by machine learning algorithms. In the context of natural language processing (NLP), data vectorization specifically refers to converting text data into numerical representations.

  Here Count Vectorizer is used for data Vectorization as per need of this study in respective to the dataset.

  Various Python libraries that are based on natural language processing and these are Text Blob, NLTk, Neat Text, Nfx, WordNet Lemmatize,
  Tokenization, etc.



	Unnamed: 0	Emotion	Text	Clean_Text	preprocessed_text	label_num	clean_Text
0	0	neutral	Why ?	NaN		NaN	Why
1	1	joy	Sage Act upgrade on my to do list for tommorow.	Sage Act upgrade list tommorow	sage act upgrade list tommorow	NaN	Sage Act upgrade on my to do list for tommorow
2	2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL MAN HATE FUNERALS SH...	way homegirl baby funeral man hate funeral rea...	0.0	ON THE WAY TO MY HOMEGIRL BABY FUNERAL MAN I H...
3	3	joy	Such an eye ! The true hazel eye-and so brill...	eye true hazel eyeand brilliant Regular feat...	eye true hazel eyeand brilliant regular featur...	NaN	Such an eye The true hazel eyeand so brillia...
4	4	joy	@Iluvmiasantos ugh babe.. hugggzzz for u .! b...	ugh babe hugggzzz u babe naamazed nga ako e...	iluvmiasantos ugh babe hugggzzz u babe naamaze...	NaN	@Iluvmiasantos ugh babe hugggzzz for u babe ...
...	...	...	...	...	...	...	...
34787	34787	surprise	@MichelGW have you gift! Hope you like it! It'...	gift Hope like it hand wear Itl warm Lol	michelgw gift hope like hand made wear itl ke...	NaN	@MichelGW have you gift Hope you like it Its h...
34788	34788	joy	The world didnt give it to me..so the world MO...	world didnt meso world DEFINITELY cnt away	world didnt give meso world definitely cnt tak...	NaN	The world didnt give it to meso the world MOST...
34789	34789	anger	A man robbed me today	man robbed today	man robbed today	NaN	A man robbed me today

	Unnamed: 0	Emotion	Text	Clean_Text	preprocessed_text
0	0	neutral	Why ?	NaN	NaN
1	1	joy	Sage Act upgrade on my to do list for tommorow.	Sage Act upgrade list tommorow	sage act upgrade list tommorow
2	2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL MAN HATE FUNERALS SH...	way homegirl baby funeral man hate funeral rea...
3	3	joy	Such an eye ! The true hazel eye-and so brill...	eye true hazel eyeand brilliant Regular feat...	eye true hazel eyeand brilliant regular featur...
4	4	joy	@Iluvmiasantos ugh babe.. hugggzzz for u .! b...	ugh babe hugggzzz u babe naamazed nga ako e...	iluvmiasantos ugh babe hugggzzz u babe naamaze...
...	...	...	...	...	...
34787	34787	surprise	@MichelGW have you gift! Hope you like it! It'...	gift Hope like it hand wear Itl warm Lol	michelgw gift hope like hand made wear itl ke...
34788	34788	joy	The world didnt give it to me..so the world MO...	world didnt meso world DEFINITELY cnt away	world didnt give meso world definitely cnt tak...
34789	34789	anger	A man robbed me today .	man robbed today	man robbed today
34790	34790	fear	Youu call it JEALOUSY, I call it of #Losing YO...	Youu JEALOUSY #Losing YOU	youu call jealousy call losing
34791	34791	sadness	I think about you baby, and I dream about you ...	think baby dream time	think baby dream time

34792 rows × 5 columns

Fig-5: Preprocessed Dataset Description

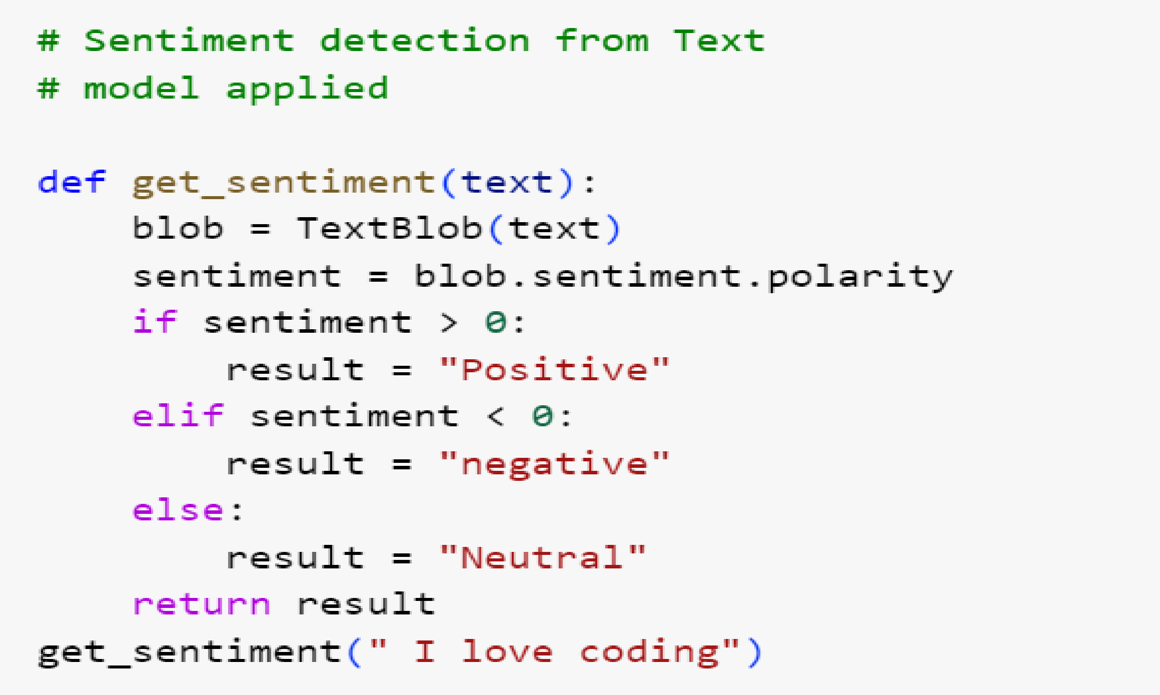
3.3 Sentiment extraction Model

Sentiment Detection is a very vital part of emotion recognition from textual data. Previously various algorithms used for detecting sentiment in past researches but they failed to give accurate sentiment in many cases.

Get () sentiment is a proposed algorithm by Python Environment to detect and analysis sentiment. Three different categories of sentiment have been detected by this algorithm. “Neutral”, “Positive”, “Negative” are three categories of detected sentiment. This algorithm is able to detect the sentiment by locating each sentence successfully and word to solve the problem of word ambiguity. The word ambiguity problem is hindering good security, which this algorithm largely solves. And the machine provides good accuracy. In other words, Machine and algorithms work properly and more accurately.

Table2: Number of Sentiment

Emotion	sentiment	
anger	Neutral	1386
	Positive	1124
	negative	1787
disgust	Neutral	251
	Positive	281
	negative	324
fear	Neutral	1844
	Positive	2032
	negative	1534
joy	Neutral	3649
	Positive	5714
	negative	1682
Neutral	Neutral	1523
	Positive	553
	negative	178
sadness	Neutral	2128
	Positive	1965
	negative	2629
shame	Neutral	50
	Positive	50
	negative	46
surprise	Neutral	1545
	Positive	1894
	negative	623

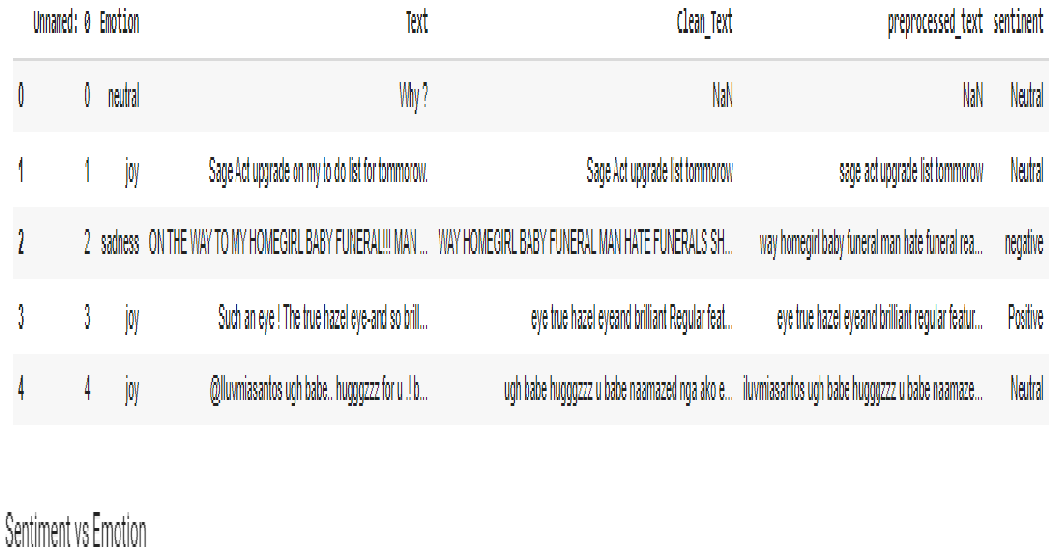


```python
# Sentiment detection from Text
# model applied

def get_sentiment(text):
    blob = TextBlob(text)
    sentiment = blob.sentiment.polarity
    if sentiment > 0:
        result = "Positive"
    elif sentiment < 0:
        result = "negative"
    else:
        result = "Neutral"
    return result
get_sentiment(" I love coding")
```

Fig6: Proposed Sentiment Detection Algorithm

3.3.1 <u>Sentiment Detection</u>: Python is one of the most powerful tools when it comes to performing data science tasks—it offers a multitude of ways to perform sentiment analysis in Python. The most popular ones are enlisted here

1. Using Text Blob
2. Using Bag of Words Vectorization-based Models
3. Using Transformer-based Models



	Unnamed: 0	Emotion	Text	Clean_Text	preprocessed_text	sentiment
0	0	neutral	Why ?	NaN	NaN	Neutral
1	1	joy	Sage Act upgrade on my to do list for tommorow.	Sage Act upgrade list tommorow	sage act upgrade list tommorow	Neutral
2	2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL MAN HATE FUNERALS SH...	way homegirl baby funeral man hate funeral rea...	negative
3	3	joy	Such an eye ! The true hazel eye-and so brill...	eye true hazel eyeand brilliant Regular feat...	eye true hazel eyeand brilliant regular featur...	Positive
4	4	joy	@Iluvmiasantos ugh babe.. hugggzzz for u .! b...	ugh babe hugggzzz u babe naamazed nga ako e...	iluvmiasantos ugh babe hugggzzz u babe naamaze...	Neutral

Sentiment vs Emotion

Fig7: Sentiment Detection

3.3.2 <u>Keyword Definition</u>: In the proposed model, Keyword Definition is a crucial component. While the model performs effectively when applied to the dataset, a notable challenge is that each keyword must be identified individually by the algorithm. Hence, Keyword Definition becomes essential for executing the proposed sentiment analysis model efficiently. Below, keywords are defined for various types of emotions.

a) **<u>Joy</u>:** Keywords: joy, X axis denotes no of words and Y axis denotes all keywords of every sentence used for ‘Joy’ emotion detection.

This List contain word count in Text Messages:

```
'the': 5299,
 'to': 4744,
 'I': 4471,
 'a': 3594,
 'and': 3221,
 'of': 2803,
 'my': 2793,
 'in': 2328,
 'for': 1997,
 'is': 1590,
 'with': 1375,
 'you': 1282,
 'that': 1234,
 'was': 1155,
 'me': 1144,
 'at': 1124,
 'on': 1120,
 'it': 1012,
 'have': 962,
 'be': 890,
 'this': 784,
 'day': 729,
 'amp': 678,
 'up': 658,
 'had': 639,
 'all': 636,
 'so': 603,
 'time': 601,
 'Im': 555,
 'when': 544,
 'your': 528,
 'When': 520,
 'The': 516,
 'work': 509,
 'from': 481,
 'not': 480,
 'get': 479,
 'like': 476,
 'an': 473,
 'today': 471,
 'tomorrow': 467,
 'Christmas': 453,
 'out': 453,
 'just': 452,
 'about': 450,
 'now': 442,
 'are': 438,
 'but': 402,
 'love': 393,
 'do': 386}
```

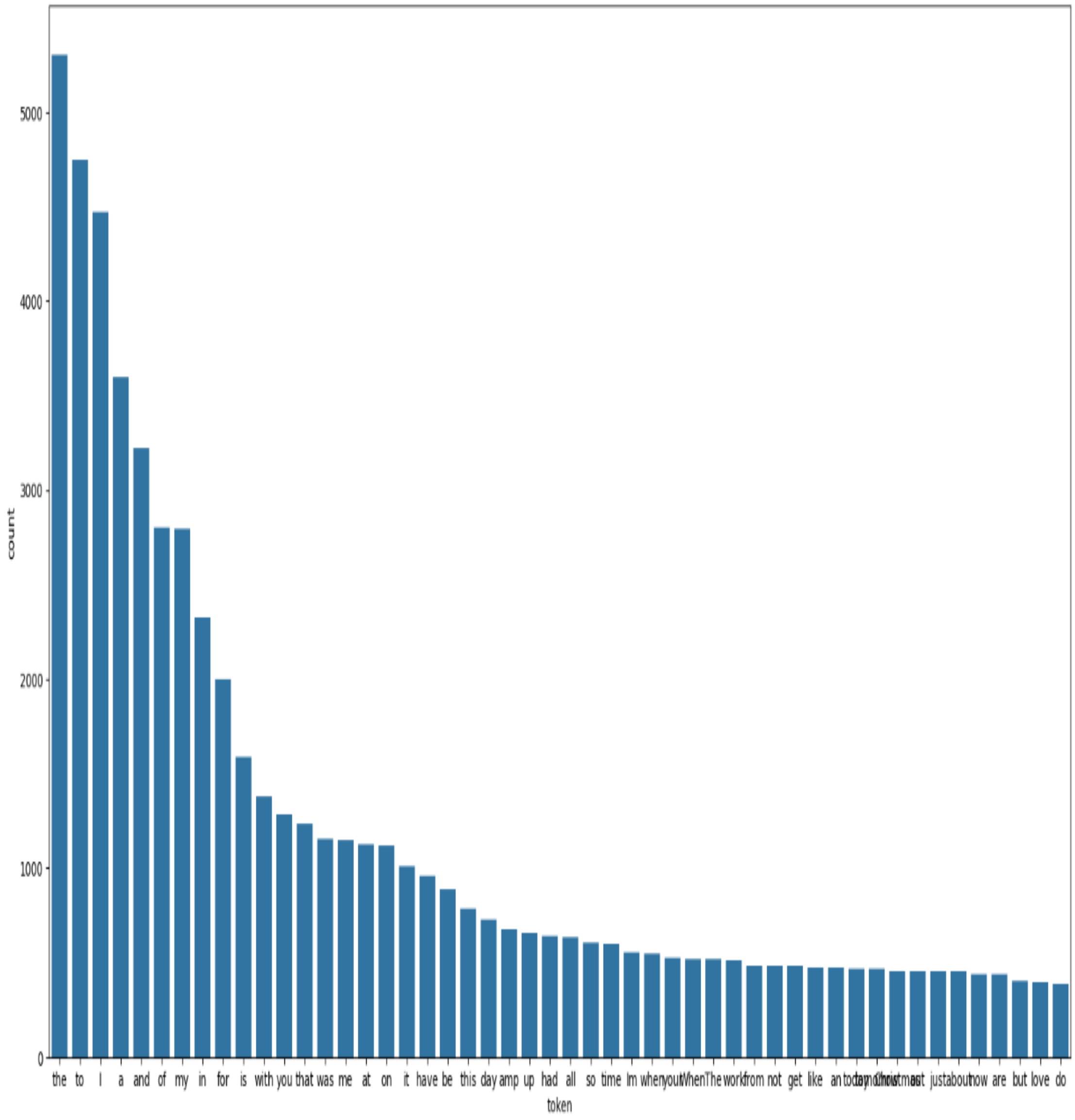


Fig-8: Keyword of Joy

b) **<u>Sadness</u>**: Keywords: sorrow, grief, melancholy, despair, anguish, heartbroken, despondent, miserable, gloomy, dejected. X axis denotes no of words and Y axis denotes all keywords of every sentence used for ‘Joy’ emotion detection.

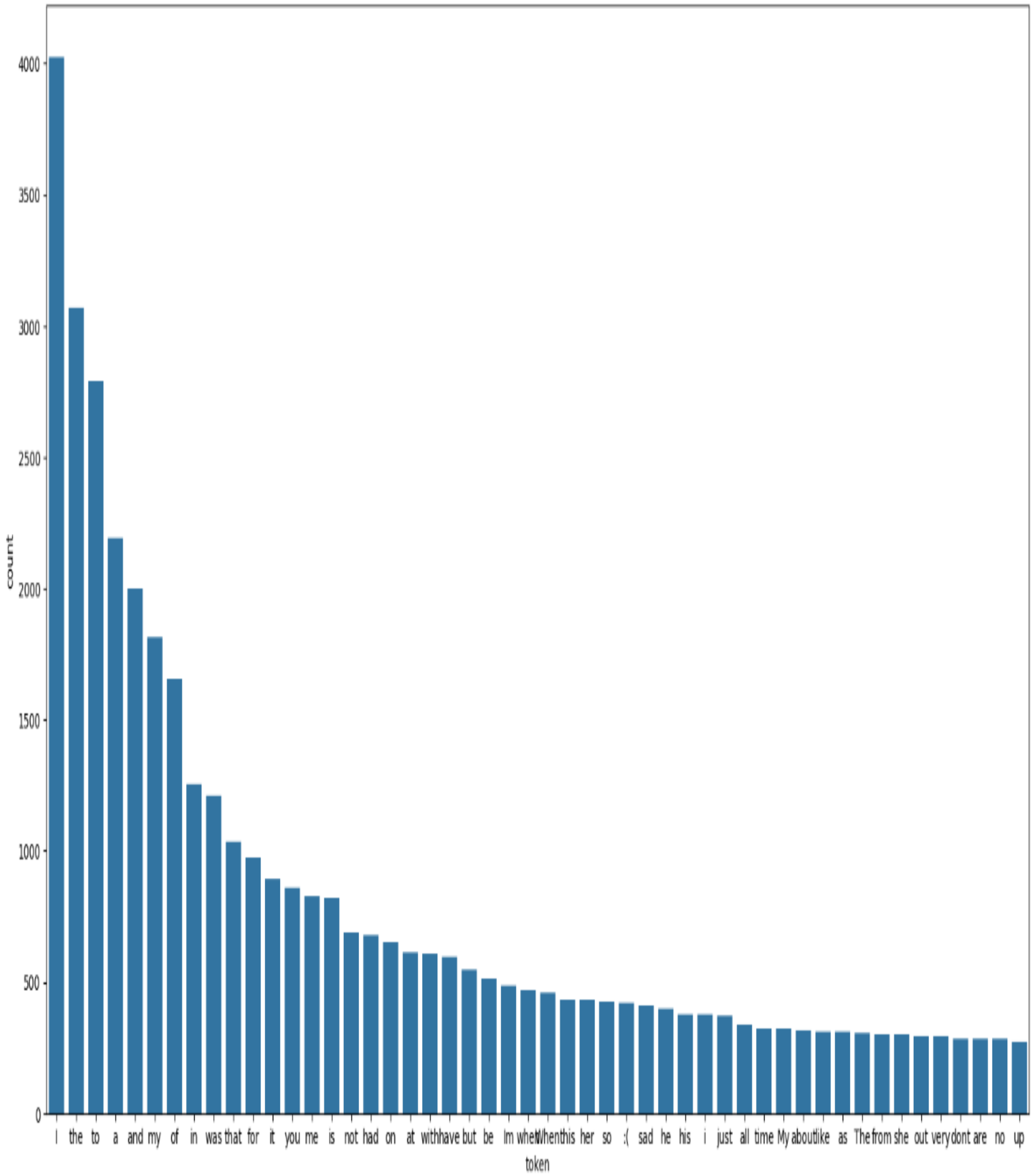


Fig-9: Keyword of Sadness

c) <u>Anger</u>: Keywords: rage, fury, wrath, indignation, annoyance, irritability, resentment, hostility, infuriated, enraged. X axis denotes no of words and Y axis denotes all keywords of every sentence used for 'Joy' emotion detection.

This List Contain Word Count in Text Messages:

```
'I': 2548,
'the': 2471,
'to': 2162,
'a': 1703,
'and': 1588,
'my': 1299,
'of': 1256,
'was': 1149,
'me': 973,
'in': 915,
'that': 855,
'it': 679,
'you': 677,
'for': 669,
'with': 632,
'had': 594,
'is': 588,
'at': 563,
'not': 558,
'he': 475,
'on': 474,
'when': 420,
'When': 390,
'her': 379,
'be': 338,
'his': 334,
'she': 316,
'have': 314,
'about': 308,
'angry': 305,
'so': 300,
'an': 298,
'up': 296,
'this': 294,
'out': 276,
'but': 275,
'’': 247,
'do': 238,
'were': 235,
'him': 234,
'as': 234,
's': 232,
'are': 219,
'The': 218,
'by': 218,
'they': 216,
'your': 210,
'who': 205,
'just': 202,
'all': 199}
```

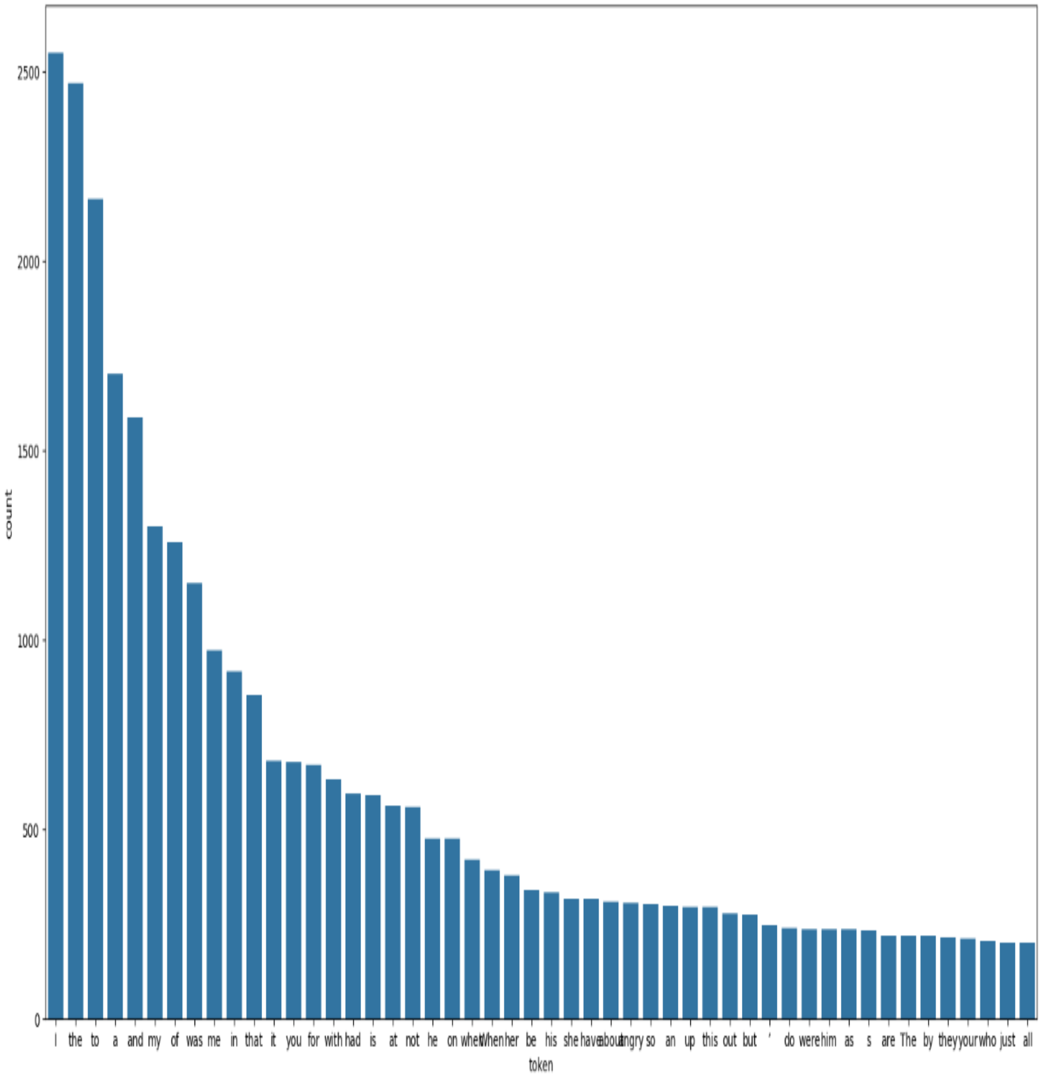


Fig-10: Keyword for anger

d) **Fear**: Keywords: terror, dread, anxiety, panic, apprehension, phobia, fright, nervousness, alarm, trepidation. X axis denotes no of words and Y axis denotes all keywords of every sentence used for ‘Joy’ emotion detection.

This List contain word count in Text Messages:

```
'I': 3623,
'the': 3503,
'of': 2252,
'and': 2252,
'to': 2229,
'a': 2120,
'in': 1454,
'was': 1427,
'my': 1028,
'for': 956,
'that': 955,
'not': 880,
'is': 865,
'me': 796,
'it': 686,
'have': 660,
'you': 624,
'at': 594,
'on': 590,
'afraid': 565,
'with': 485,
'had': 453,
'be': 449,
'love': 432,
'when': 405,
'today': 404,
'tomorrow': 399,
'fear': 396,
'seen': 364,
'When': 355,
'yesterday': 352,
'about': 330,
'The': 329,
'from': 326,
'we': 325,
'by': 322,
'but': 317,
'out': 315,
'he': 290,
'your': 284,
'as': 275,
'so': 272,
'are': 268,
'this': 267,
'all': 265,
'time': 256,
'an': 248,
'Im': 244,
'they': 244,
'night': 238}
```

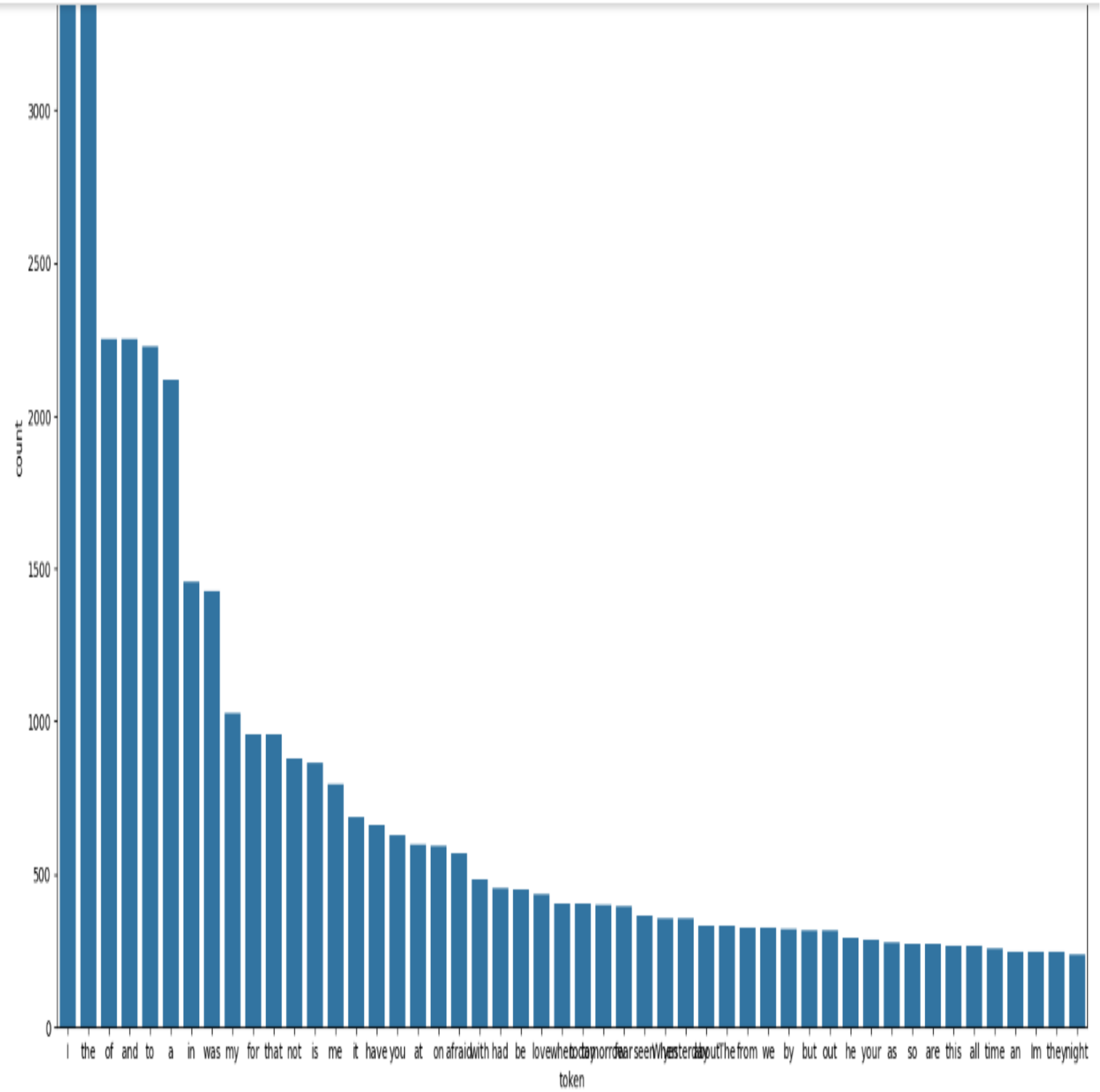


Fig-11: Keyword of fear

e) **Surprise**: Keywords: astonishment, amazement, wonder, shock, disbelief, awe, startle, astound, unexpected, startled. X axis denotes number of words and Y axis denotes all keywords of every sentence used for ‘Joy’ emotion detection.

This List contains word count in Text Messages:

```
{'the': 1383,
 'to': 1219,
 'a': 1207,
 'I': 1063,
 'my': 826,
 'and': 801,
 'in': 751,
 'for': 601,
 'of': 592,
 'you': 562,
 'is': 548,
 'it': 400,
 'on': 399,
 'me': 391,
 'that': 383,
 'was': 367,
 'en': 299,
 'een': 294,
 'at': 265,
 'be': 257,
 'with': 251,
 'de': 251,
 'when': 247,
 'i': 244,
 'up': 241,
 'Im': 227,
 'out': 223,
 'have': 223,
 'just': 211,
 'this': 209,
 'your': 204,
 ':)': 201,
 'her': 176,
 'so': 175,
 'are': 170,
 'know': 165,
 'not': 162,
 'from': 161,
 'home': 160,
 'he': 157,
 'but': 154,
 'one': 154,
 'van': 153,
 'today': 150,
'we': 149,
'day': 143,
'get': 142,
'she': 142,
'what': 139,
'ik': 137}
```

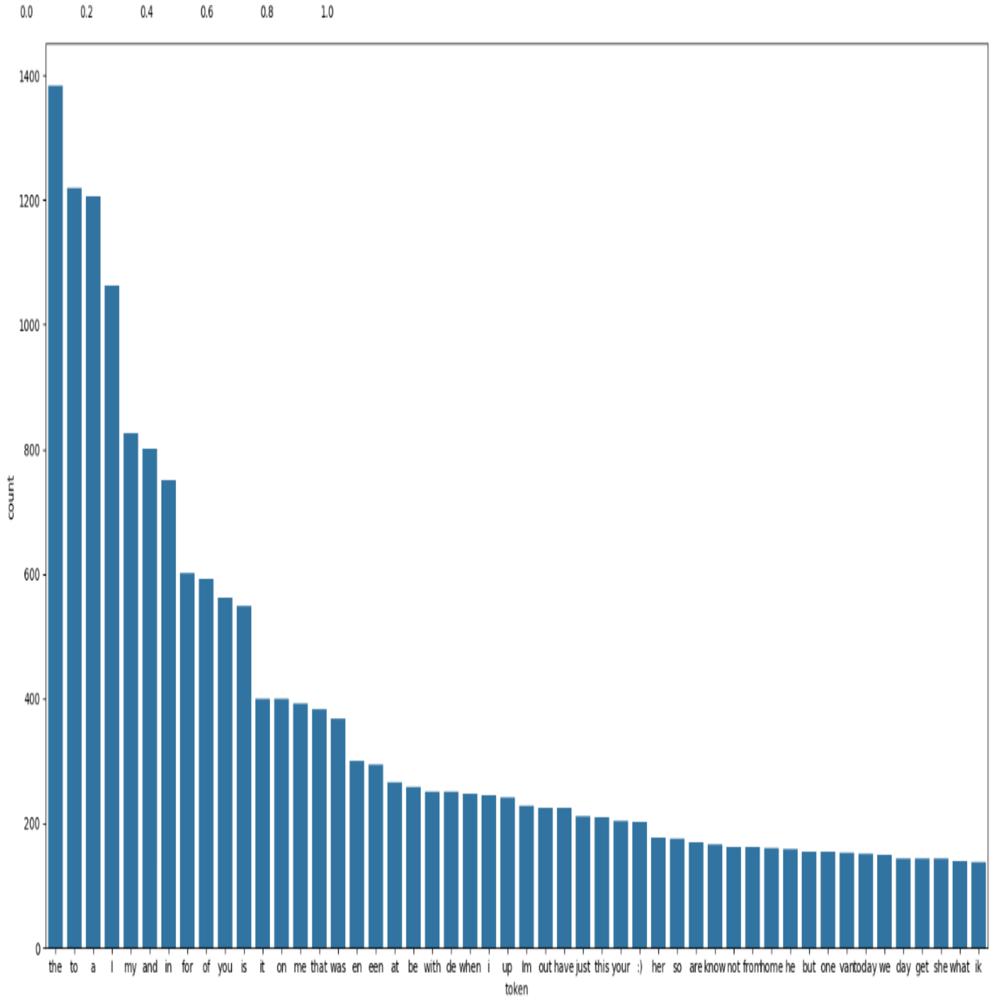
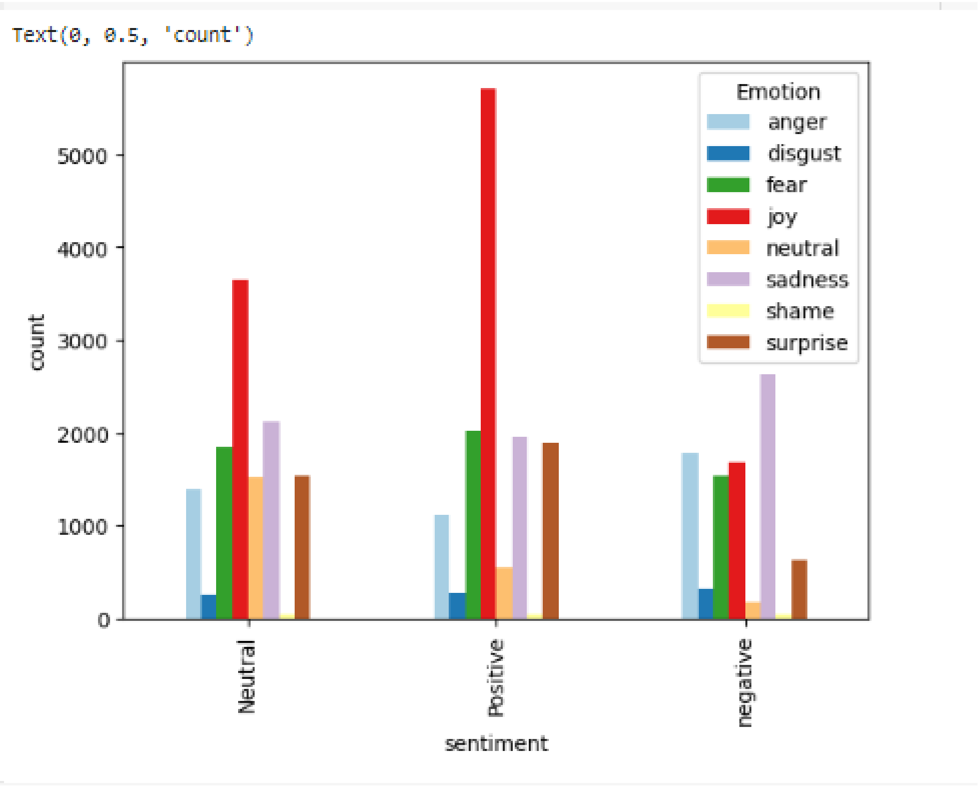


Fig-12: Keyword for Surprise

By defining these keywords for each emotion category, the sentiment analysis model can efficiently identify and analyze sentiments within the dataset.



```
Text(0, 0.5, 'count')
```

Fig-13: Distribution of Detecting Sentiment by Proposed system

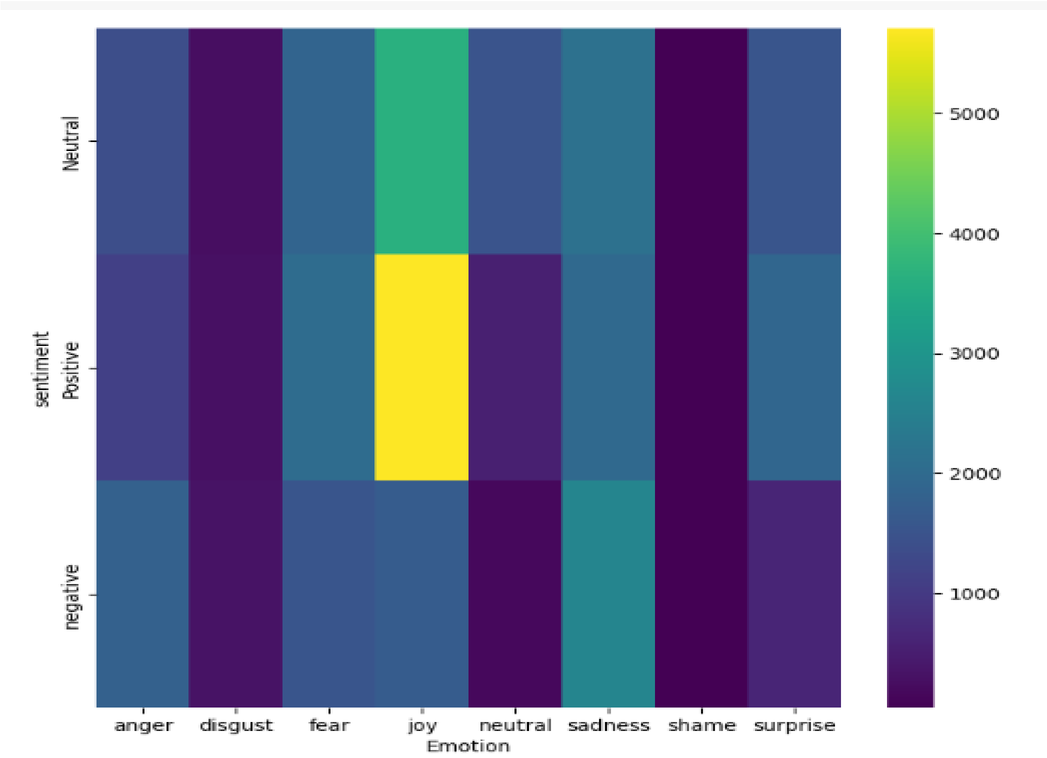
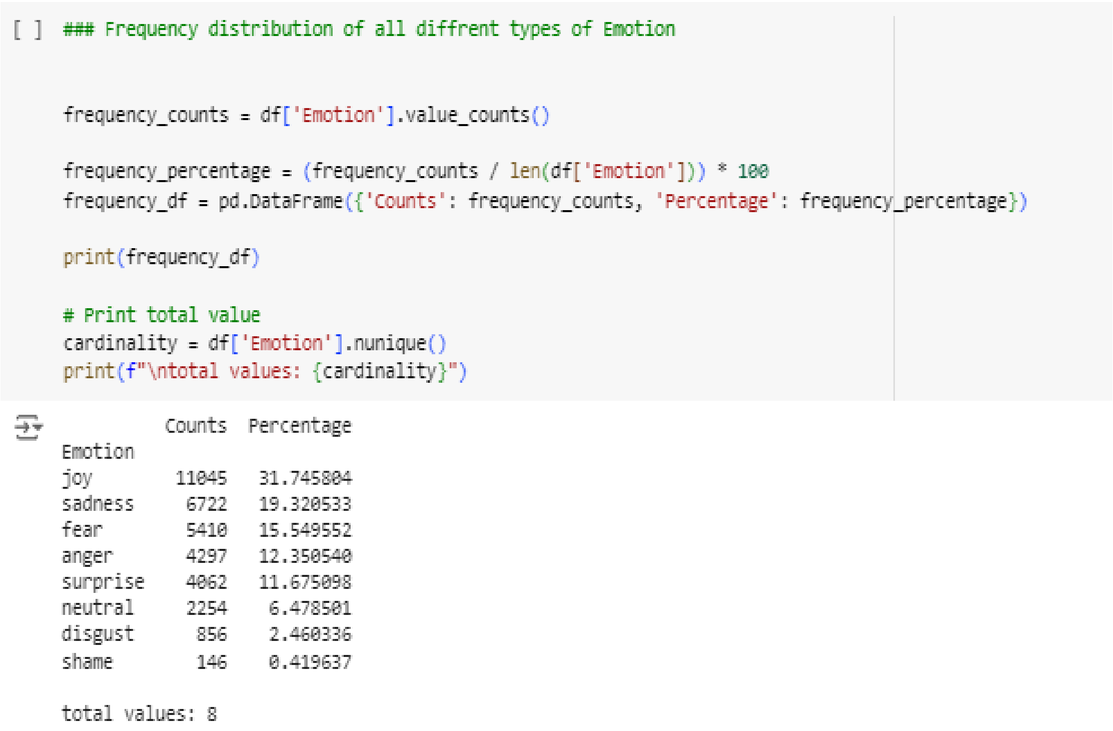


Fig-14: Heatmap of Sentiment



```
### Frequency distribution of all diffrent types of Emotion


frequency_counts = df['Emotion'].value_counts()

frequency_percentage = (frequency_counts / len(df['Emotion'])) * 100
frequency_df = pd.DataFrame({'Counts': frequency_counts, 'Percentage': frequency_percentage})

print(frequency_df)

# Print total value
cardinality = df['Emotion'].nunique()
print(f"\ntotal values: {cardinality}")
```

```
          Counts  Percentage
Emotion
joy        11045   31.745804
sadness     6722   19.320533
fear        5410   15.549552
anger       4297   12.350540
surprise    4062   11.675098
neutral     2254    6.478501
disgust      856    2.460336
shame        146    0.419637

total values: 8
```

Fig- 15: Number of Sentiments in terms of Emotions

3.3.3 Mapping Sentiment to Numerical Values

The code begins by creating a new column named "label Num" in the Data Frame `df `. It uses the `. map () ` function to transform the values in the "sentiment" column into numerical representations.

The mapping is defined as follows:

- 'negative' sentiment is mapped to 0
- 'positive' sentiment is mapped to 1
- 'neutral' sentiment is mapped to 2

Essentially, it assigns numerical labels to the different sentiment categories

a) <u>Data Frame Manipulation and Keyword Extraction Method</u>

After creating the "label Num" column, the code proceeds to modify the Data Frame `df`. It uses the `. drop () ` method to remove the original "sentiment" column from the Data Frame. This step is performed because the sentiment information has now been encoded into numerical values in the "label Num" column. The original textual representation of sentiment is no longer necessary.

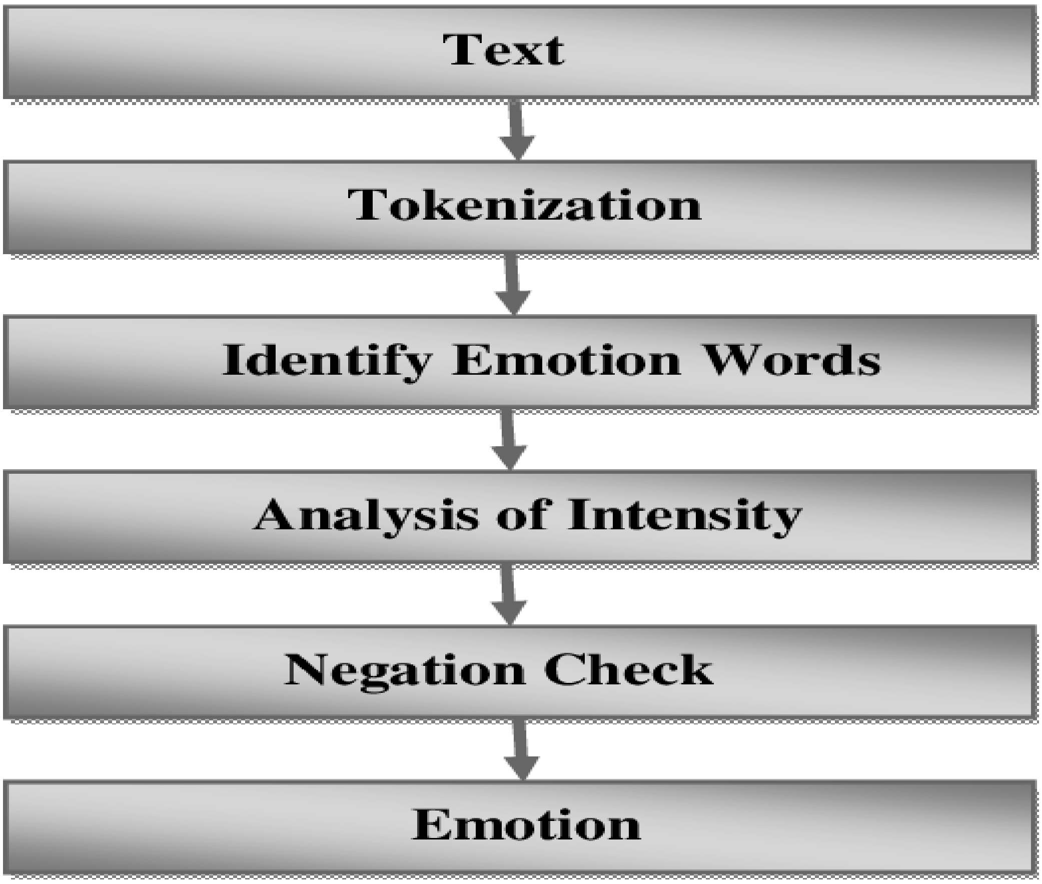


Fig-16: Keyword Extraction Method

Source: Semantic Scholar [25]

Displaying the Data Frame:

Lastly, the code displays the first five rows of the modified Data Frame `df` using the `. head (5) ` function. This provides a glimpse of the Data Frame after the changes have been applied, showcasing the newly created "label Num" column and the absence of the original "sentiment" column.

```
 # Mapping sentiment num / encode
df["label_num"] = df.sentiment.map({
   'negative': 0,
   'positive': 1,
   'neutral': 2
})
df = df.drop(columns=['sentiment'])
df.head(5)
```

In summary, this code snippet facilitates the transformation of textual sentiment labels into numerical representations, enhancing the data's suitability for machine learning tasks that require numerical input. It follows a systematic process of mapping, Data Frame manipulation, and display to achieve this transformation.

3.4 Feature Extraction Techniques

In the emotion recognition process through Machine Learning Models, feature extraction is the crucial part of emotion classification. The efficacy of feature extraction is intricately intertwined with the precision of emotion classification. In previous Researchers, various kinds of feature extraction methods are used such as N-grams, BERT, and Dict Vectorizer. Nowadays researchers are using TFDF to get good accuracy in the classification phase. In this research presently Count Vectorizer has been applied to get very good results. TFDF is also used for getting comparative analysis in terms of accuracy. So, these methods were carried out by considering two features namely Term Frequency- Inverse Document Frequency and Count Vectors. The data is error-free and clean. Applying both methods got very good results depending on the specific requirements of the task and the characteristics of the dataset.

- **TF-IDF:** This can be particularly useful in emotion detection because it helps to identify words that are unique or distinctive to certain emotions. TFDF stands for "Term Frequency - Document Frequency." It's a concept commonly used in information retrieval and text mining to evaluate the importance of a term within a document or a corpus of documents.

Here's how TFDF works:

- **Term Frequency (TF):** This component measures how often a term occurs in a document. It's calculated as the ratio of the number of times a term appears in a document to the total number of terms in that document. Essentially, it shows the relevance of a term within a specific document. A higher term frequency suggests that the term is important or central to the document.

$$\mathrm{TF}(t,d)=\frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

- **<u>Document Frequency (DF):</u>** This component measures how often a term occurs in the entire corpus of documents. It's calculated as the ratio of the number of documents that contain the term to the total number of documents in the corpus. Document frequency gives an idea of how common or rare a term is across all documents.

$$\mathrm{DF}(t)=\frac{\text{Number of documents containing term } t}{\text{Total number of documents in the corpus}}$$

- **<u>TF-IDF Score</u>**: The TFDF score combines both TF and DF to evaluate the importance of a term. It is calculated by multiplying the TF and IDF values.

$$\mathrm{TFDF}(t,d,D)=\mathrm{TF}(t,d)\times\log\left(\frac{1}{\mathrm{DF}(t)}\right)$$

  - \(t\) is the term.
  - \(d\) is the document.
  - \(D\) is the corpus of documents.

- **<u>Inverse Document Frequency (IDF)</u>**: This term accounts for the fact that certain terms might appear frequently across documents but are not necessarily important because they are common words (e.g., "the", "and"). IDF penalizes such terms. It's calculated as the logarithm of the inverse of DF.

$$\mathrm{IDF}(t,D)=\log\left(\frac{1}{\mathrm{DF}(t)}\right)$$

By combining TF and IDF, TFDF highlights terms that are both frequent within a document and rare across the entire corpus, thereby identifying terms that are significant to that specific document. TF-IDF is often used in information retrieval systems to rank documents based on their relevance to a query.

By using TF-IDF, the model can potentially give more weight to these important words and improve the accuracy of emotion detection.

Count Vectorizer: It can also be similarly used for emotion detection to TF-IDF. Count Vectorizer can be used to convert text data into a numerical format by counting the occurrences of words in each document. In the context of emotion detection.

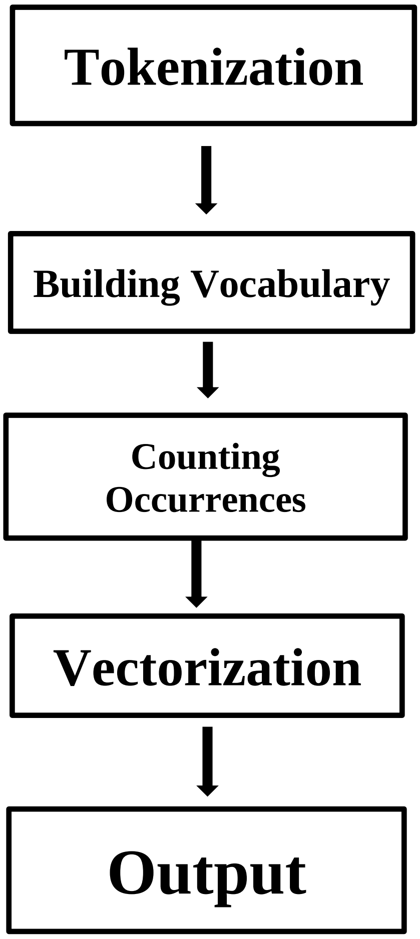


Fig -17: Working Principle of Count Vectorizer

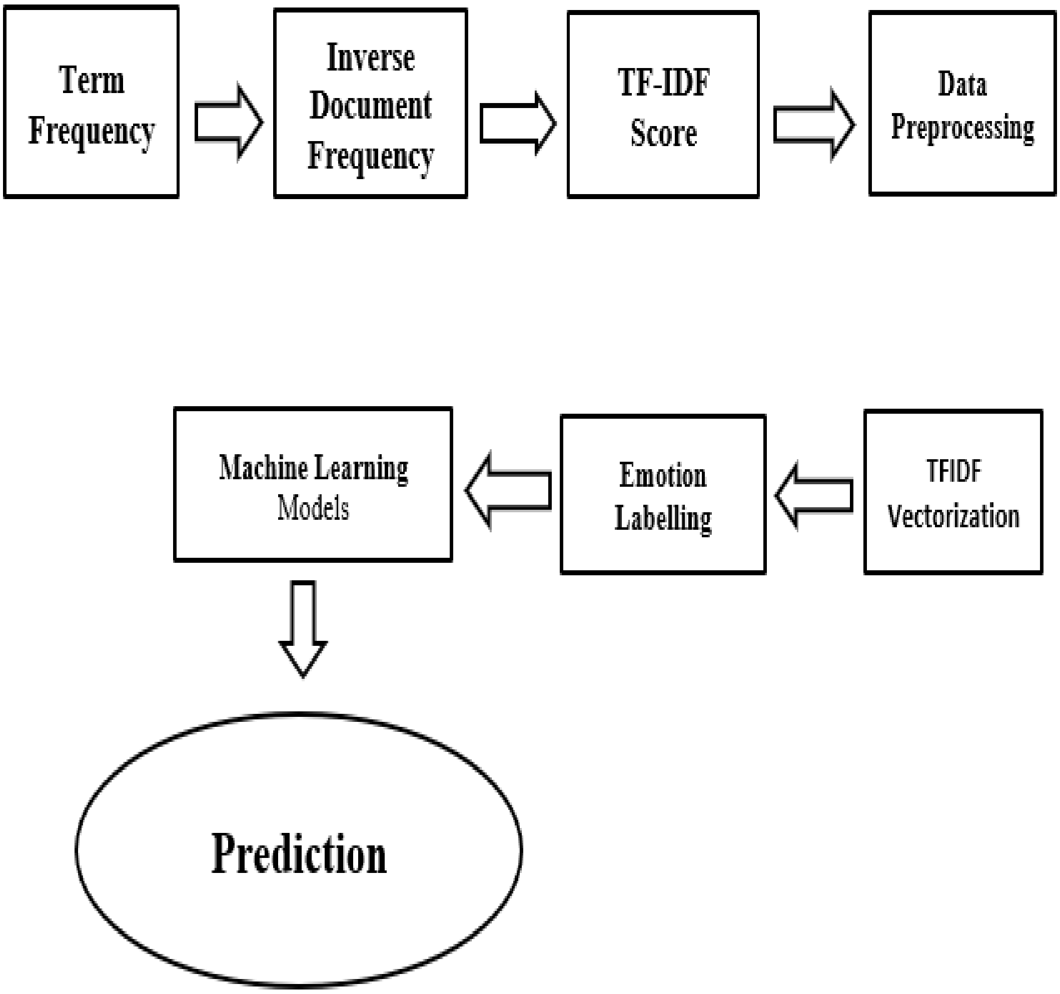


Fig -18: Working Principle of TF-IDF

3.5 Text Classification Techniques

Text classification is a common task in natural language processing (NLP) where the goal is to categorize text documents into predefined classes or categories. There are various machine learning models that can be used for text classification, each with its own advantages and disadvantages. Here are some proposed models as follows:

- Naïve bayes Algorithm

a) Multinomial Naïve Bayes

b) Bernoulli Naïve Bayes

c) Gassian Naïve Bayes

- Support Vector Machines (linear).
- Random Forest Classifier.
- Logistic regression.
- KNN algorithm.

The study explores the effectiveness of Naive Bayes Classifier in detecting emotions from textual data. It considers three types of Naive Bayes classifiers: Multinomial Naive Bayes, Bernoulli Naive Bayes, and Gaussian Naive Bayes. While Multinomial Naive Bayes has been traditionally used, the study introduces Bernoulli and Gaussian Naive Bayes classifiers as new approaches. The results indicate that both Bernoulli and Gaussian Naive Bayes classifiers outperform the Multinomial Naive Bayes algorithm, yielding high accuracy, F1 score, precision, and recall.

Additionally, the study explores other machine learning methods such as Random Forest, Logistic Regression, KNN, and linear Support Vector Machines (SVC) to achieve optimal results. By employing data preprocessing techniques and two proposed feature extraction methods, the study observes variations in performance. Specifically, the Count Vectorizer feature extraction method demonstrates superior accuracy compared to the TF-IDF feature extraction method.

In summary, the study underscores the effectiveness of Naive Bayes classifiers in emotion detection from textual data, showcasing improvements over traditional methods and highlighting the impact of feature extraction techniques on model performance.

3.5.1 Working Principle of Algorithms

- **Naïve Bayes Algorithm:**

Naive Bayes is a type of algorithm used for classification tasks. It's called "naive" because it makes a very simple assumption: that the presence of one feature doesn't affect the presence of another feature. This assumption simplifies the math behind the algorithm.

Here's how it works:

- **Bayes' Theorem**: It's a way to calculate probabilities. In the case of Naive Bayes, it helps to figure out the probability of a certain class (like "spam" or "not spam") given some data.
- **Naive Assumption**: This is the idea that features (like words in a text) are independent of each other when it comes to predicting the class. This assumption is often not true in real life, but it makes the math easier.

- **Training:** Naive Bayes looks at a bunch of examples where the class has been understood and it learns the probability of each feature belonging to each class.
- **Prediction:** When to classify something new, Naive Bayes calculates the probability of each class given the features and picks the class with the highest probability.

There are different types of Naive Bayes algorithms, each suited for different kinds of data. For example:

- ✓ Gaussian Naive Bayes: works well when features have a normal distribution.
- ✓ Multinomial Naive Bayes is good for things like word counts in text.
- ✓ Bernoulli Naive Bayes is useful when features are binary (like whether a word appears or not).

Naive Bayes classifiers, including Gaussian Naive Bayes and Bernoulli Naive Bayes, are simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. While they share some common principles, their architectures differ based on the distribution of the input features.

➢ Gaussian Naive Bayes:

- **Architecture:**

Assumes that continuous features follow a Gaussian (normal) distribution.

Each class is associated with a mean and a variance for each feature. The probability density function (PDF) of the Gaussian distribution is used to calculate the likelihood of observing a particular value given the class. During training, the mean and variance of each feature are estimated for each class. During prediction, the probability of a sample belonging to each class is calculated using Bayes' theorem, and the class with the highest probability is assigned to the sample.

- **Strengths:**

Effective for continuous or real-valued features.

Simple and efficient, especially for high-dimensional data.

Less affected by the curse of dimensionality compared to other classifiers.

➢ **<u>Bernoulli Naive Bayes</u>:**

- **Architecture:**

Assumes that features are binary-valued (e.g., presence or absence of a feature). Each feature is modeled as a binary random variable following a Bernoulli distribution. The probability of each feature being 1 or 0 is estimated for each class. During training, the probabilities of features being 1 or 0 are calculated for each class. During prediction, the likelihood of observing the feature values given the class is calculated using the Bernoulli distribution, and Bayes' theorem is applied to assign the class with the highest probability.

- **Strengths:**

Well-suited for binary or categorical features, such as text classification (presence or absence of words).

Handles sparse data efficiently. Robust to irrelevant features.

Often used in text mining and document classification tasks.

In summary, Gaussian Naive Bayes assumes that features follow a Gaussian distribution, making it suitable for continuous features, while Bernoulli Naive Bayes assumes binary features following a Bernoulli distribution, making it suitable for binary or categorical features. Both algorithms are simple, efficient, and effective for various classification tasks, depending on the nature of the input features.

Naive Bayes is popular because it's easy to understand, quick to train, and can work surprisingly well in many situations, especially with text data. However, it might not perform as well when the features are not actually independent or when there's not enough training data. Despite its simplicity, it's widely used and can be a powerful tool in the right situations.

➢ Support vector machine:

Here linear Support Vector Machines has been implemented. Support Vector Machines (SVM) are a type of algorithm designed to draw the best possible line or boundary in a multi-dimensional space, effectively separating different classes of data. The ultimate aim is to create a decision boundary, often referred to as a hyperplane, that can accurately categorize new data points in the future.

The SVM algorithm accomplishes this by identifying key data points known as support vectors. Support vectors are essential elements that determine the precise orientation and location of the hyperplane within the Support Vector Machine algorithm, crucial for effectively separating different classes of data. Essentially, SVM seeks to locate the most extreme points that help define the boundary between different classes. Hence, the term "Support Vector Machine" originates from this emphasis on identifying and utilizing these critical support vectors.

Imagine a scenario where data points belonging to two distinct categories are plotted on a graph. SVM works by strategically positioning a line or boundary, referred to as a hyperplane, to effectively separate these categories. This hyperplane is determined by identifying the most pivotal data points, known as support vectors, which play a significant role in defining the boundary. Through this process, SVM aims to create the most optimal decision boundary for accurate classification of new data points.

SVM aims to locate the most extreme points in the data set that are instrumental in determining the optimal hyperplane. By doing so, it creates a clear separation between different classes, making it easier to classify new data points accurately in the future. Therefore, SVM is not just about drawing any boundary; it's about finding the best possible boundary that maximizes the margin and minimizes the classification error, thereby enhancing the algorithm's ability to generalize well to unseen data. Below diagram consider for better understanding:

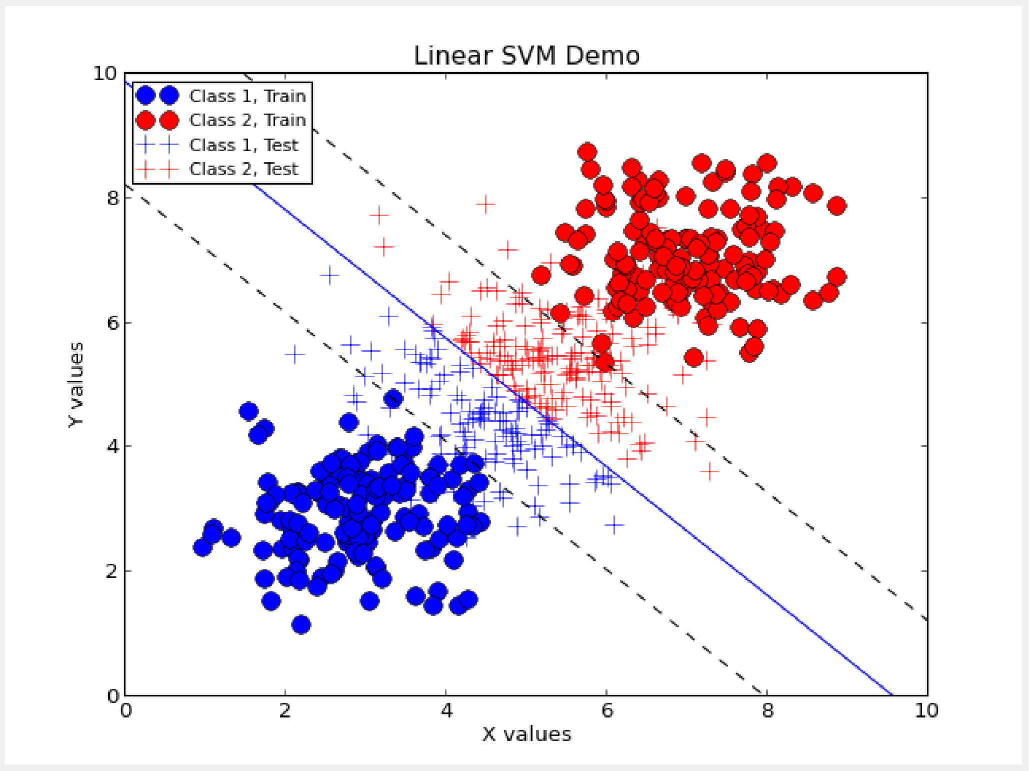


Fig-19: Support Vector Machine

Source: LinkedIn [21]

➢ <u>Logistic Regression</u>:

Logistic regression is a statistical method used for binary classification tasks, where the output variable is categorical and has only two possible outcomes, typically represented as 0 and 1. The working principle of logistic regression can be explained in a structured way, as follows:

- **Input Data:**

  Logistic regression takes input data features (X) and their corresponding labels (Y). X represents the independent variables or features, while Y represents the dependent variable or target variable with two classes (0 or 1).

- **Linear Combination:**

  Logistic regression begins by computing a linear combination of the input features and associated weights.

  It calculates the weighted sum of input features:

$$z = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

- **Logistic Function:**

  The linear combination is then transformed using the logistic function (also known as the sigmoid function) to produce the predicted probability. The logistic function maps any real-valued number to the range (0, 1), which is suitable for representing probabilities.

  The logistic function is defined as:

$$P(y = 1|x) = \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2+\ldots+w_nx_n)}}$$

- **Prediction:**

  After applying the logistic function, the output represents the predicted probability of the positive class (class 1). If the predicted probability is greater than a threshold (typically 0.5), the instance is classified as belonging to the positive class (1); otherwise, it is classified as belonging to the negative class (0).

- **Training:**

  During the training phase, the model learns the optimal values of the coefficients (weights) that minimize the difference between the predicted probabilities and the actual labels. This optimization process is typically performed using techniques like gradient descent or more advanced optimization algorithms.

- **Cost Function:**

  In logistic regression, the cost function (or loss function) is used to quantify the difference between the predicted probabilities and the actual labels. The most commonly used cost function for logistic regression is the cross-entropy loss function.

- **Gradient Descent:**

  Gradient descent is an iterative optimization algorithm used to minimize the cost function by adjusting the weights. It calculates the gradient of the cost function with respect to each weight and updates the weights in the opposite direction of the gradient to minimize the cost.

- **Model Evaluation:**

  Once trained, the logistic regression model can be evaluated using various performance metrics such as accuracy, precision, recall, F1-score, ROC curve, and AUC-ROC.

In summary, logistic regression works by fitting a logistic function to the input data to model the probability of a binary outcome and then making predictions based on this probability. It's a fundamental algorithm in the field of machine learning and is widely used for binary classification tasks.

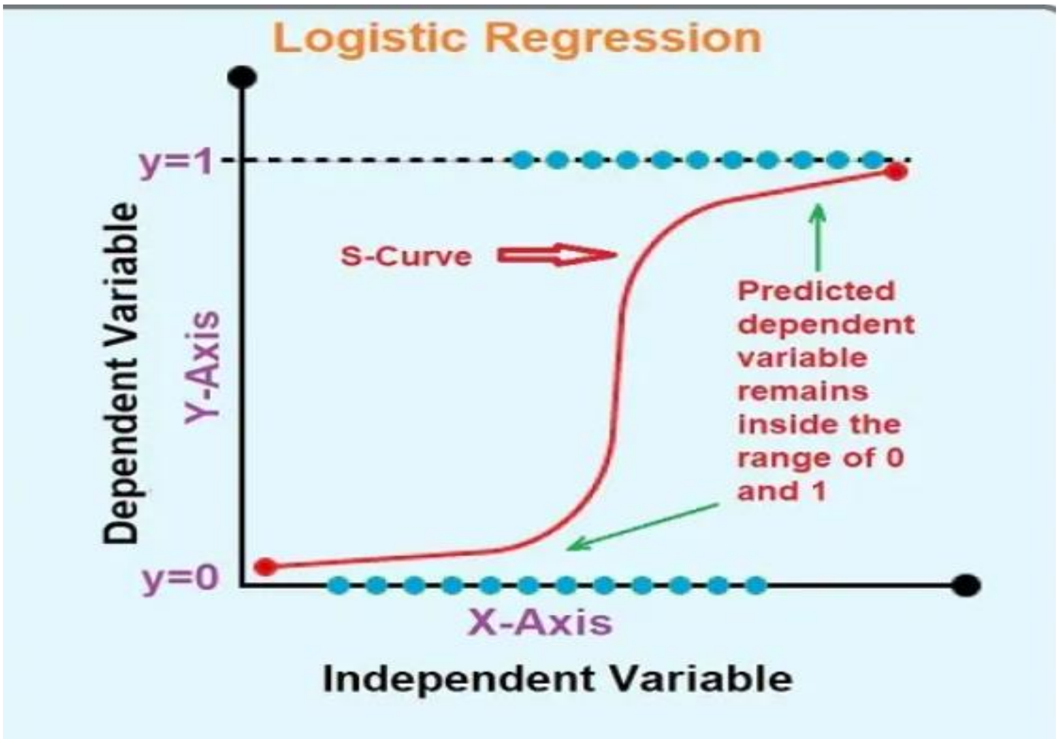


Fig-20: Logistic Regression

Source: E Jable [24]

- **K -Nearest Neighbors (KNN):**

Sure, here's a more advanced explanation of the K-Nearest Neighbors (KNN) algorithm: KNN is a non-parametric and lazy learning algorithm used for both classification and regression tasks. Non-parametric means it doesn't make assumptions about the underlying data distribution, and lazy learning means it doesn't learn a model during training; instead, it stores the entire training dataset and makes predictions based on the similarity between new data points and existing data points during inference.

In the context of classification:

- **Training**: KNN stores all the training data points and their corresponding class labels.
- **Prediction**: When a new data point is presented for prediction, KNN calculates the distances between the new point and all the points in the training set using a chosen distance metric, such as Euclidean distance.
- **Selection of K**: KNN selects the K-nearest neighbors to the new data point based on the calculated distances. The value of K is a hyperparameter that needs to be tuned and affects the algorithm's performance. A small K may lead to overfitting, while a large K may lead to underfitting.
- **Majority Voting**: For classification, KNN assigns the class label to the new data point based on the majority class among its K-nearest neighbors. It can handle ties in various ways, such as assigning equal weights to each neighbor or choosing the class with the smallest distance.
- **Prediction**: Finally, KNN assigns the class label of the majority class to new data point. In regression, instead of class labels, KNN predicts a continuous value by averaging (or weighted averaging) the target values of the K-nearest neighbors.

Key considerations for KNN include:

- **Choice of Distance Metric**: The distance metric used can significantly impact the algorithm's performance. Different distance metrics may be more suitable for different types of data.
- **Feature Scaling**: Since KNN relies on distance calculations, it's essential to scale the features to ensure that no single feature dominates the distance calculation.
- **Computational Complexity**: KNN's prediction time complexity grows

linearly with the size of the training dataset, making it computationally expensive for large datasets. Techniques like KD-trees or ball trees can be used to speed up the search process.

Despite its simplicity and ease of implementation, KNN may not perform well with high-dimensional data or imbalanced datasets. Additionally, it can be sensitive to noisy data and outliers. Nonetheless, KNN remains a versatile and widely used algorithm in machine learning, especially for smaller datasets or as a baseline model for comparison.

Euclidean Distance= $\sqrt{}(x_2-x_1)^2 + (y_2-y_1)^2$

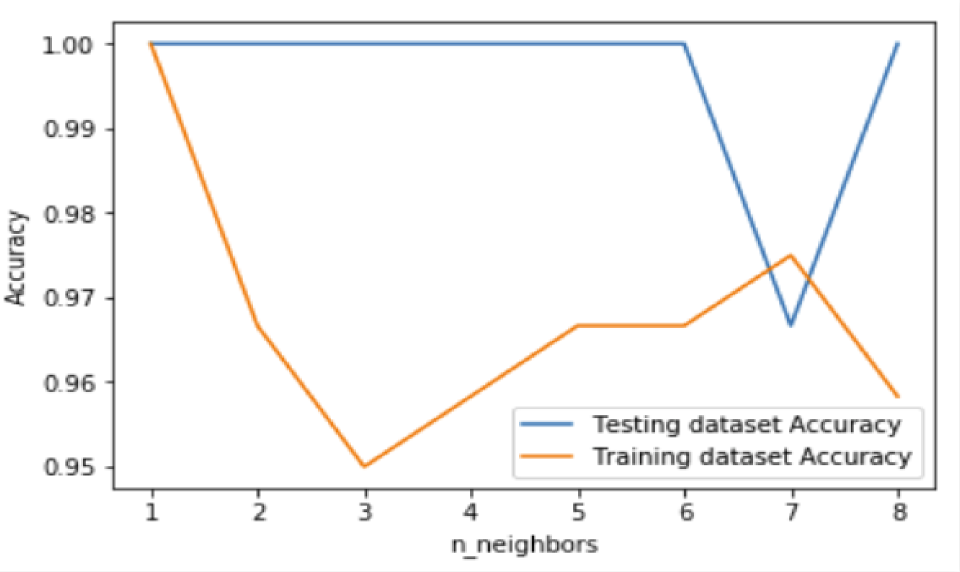


Fig-21: KNN algorithm

Source: Geeks for Geeks [23]

➢ Random Forest Classifier:

The Random Forest, or Random Decision Forest, stands as a supervised machine learning technique adept at tasks like classification and regression, leveraging decision trees. In essence, it constructs a collection of decision trees derived from randomly chosen subsets of the training data. Through this process, it gathers predictions from these diverse decision trees to formulate the ultimate prediction. What sets the Random Forest classifier apart is its versatility in handling both classification and regression tasks. Moreover, its capability to furnish feature importance scores adds considerable value by illuminating the relevance of various variables within the dataset.

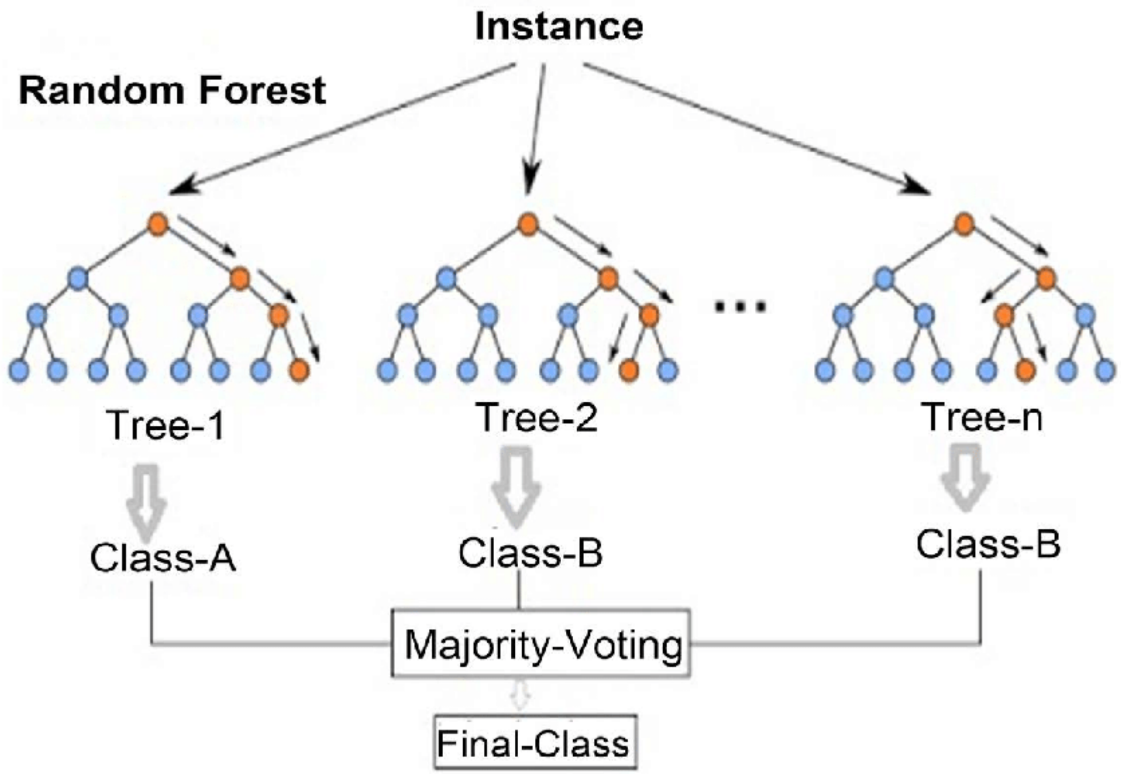


Fig-22: Random Forest Classifier

Source: Java point [26]

3.6 Training and Testing the Model:

The dataset has been split into 70:30 in majority cases for training and testing respectively. Training and testing a model typically refer to the process of building a machine learning model using a training dataset and evaluating its performance on a separate dataset called the testing or validation dataset. Training and Testing ratio has been considered different for different models as follows in a table:

Table 3: Training and Testing Data Distribution Percentage

Algorithms	**TFIDF**	**Count Vectorizer**
Logistic Regression	70:30	70:30
KNN	80:20	70:30
MNB	70:30	70:30
BNB	70:30	70:30
GNB	70:30	70:30
Random Forest	80:20	70:30

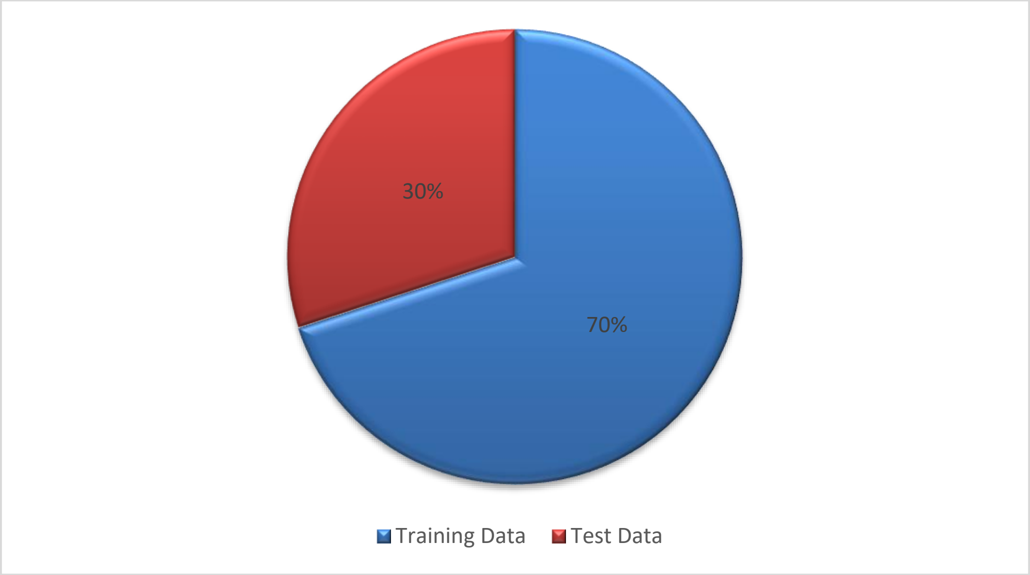


Fig-23: Training and Testing data distribution (70:30)

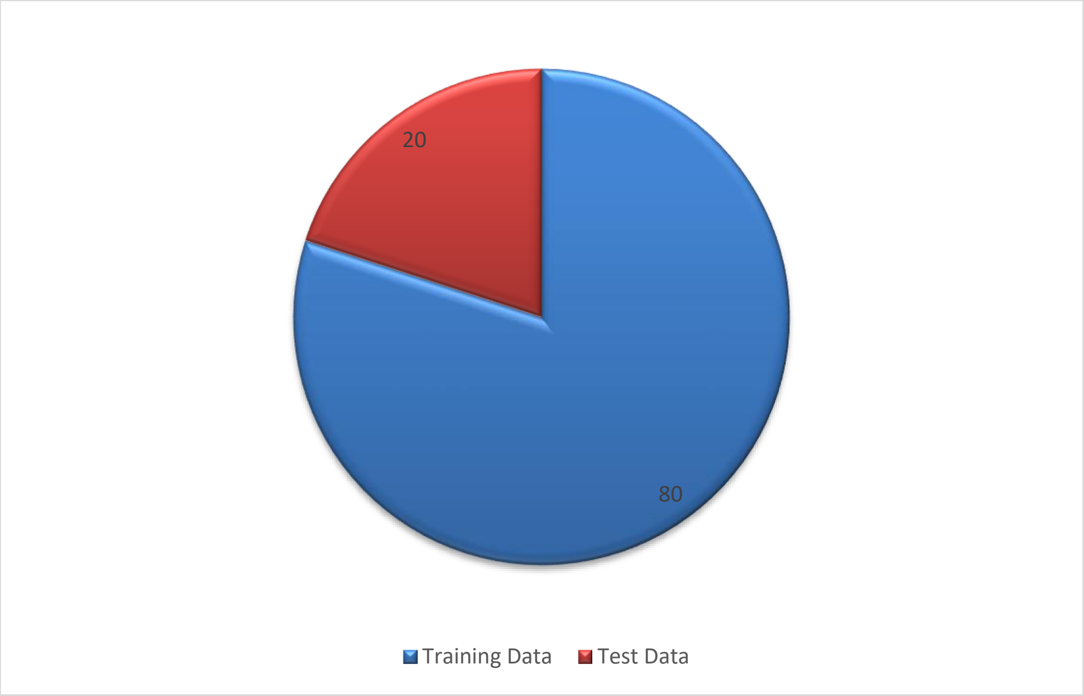


Fig-24: Training and Testing data distribution (80:20)

Here the Present research worker is conducting a comparative observational study where the researcher employed different splitting ratios for model evaluation, specifically utilizing different ratios for TF-IDF and count vectorizer techniques.

In this comparative observational study, Present research worker has implemented varying splitting ratios for model evaluation, distinguishing between TF-IDF and count vectorizer methodologies. This entails employing distinct ratios to partition the dataset for training and testing purposes, aiming to analyze and compare their respective performance outcomes.

3.7 Classification Accuracy:

This is the last stage of proposed approach. Introduce classification accuracy as a pivotal metric in assessing the performance of machine learning models, particularly in classification tasks. Highlight its significance in evaluating the model's ability to correctly predict class labels for given data instances. Classification accuracy represents the proportion of accurate predictions made by a model relative to the total number of input samples. It is calculated by dividing the count of correct predictions by the total number of input samples. Obtaining good assurance of this study was also a goal which was achieved. This has been discussed in detail in the experimentations and results section.

3.8 Architecture

Above mentioned machine learning algorithms have been applied on dataset through following steps:

I. Create Dataset/Collect Dataset
II. Perform Pre-Processing using NLP technique
III. Feature Extraction
IV. Data/Text classification using Supervised learning algorithms
V. Model Evaluation
VI. Classification (Accuracy)

Present research worker has taken 34791 data, a large dataset for this study. It is considered as large dataset for detecting emotion and sentiment in respective to the research. It refers to the transformations applied to the dataset before providing the data to algorithm. Present research worker has used Text Blob for data cleaning. Data cleaning process comprises various kind of activities as per demand for research. Present research worker has removed noise, stop words, special characters, punctuations-emojis from dataset to make the dataset clean. Duplicate counts, outliers checking, data balance or not have been checked by applying appropriate model on dataset time to time.

count vectorizer python library is being considered for feature extraction. For choosing a model we split the dataset in to train and test. Here data is split in to 3:1 ratio that means training data having 70% and testing data having 30%. This process performing TRAIN-TEST-SPLIT model. Observe the data and choose the type of algorithm. Prepare and clean the dataset and deploy the particular model. This step mainly of machine learning, here we will focus more on classification. Here the Present research worker predicts Text emotion and algorithm performance. A classification model tries to provide some output or conclusion from input values given from Training. Present research worker will get expected output or final conclusion from this step.

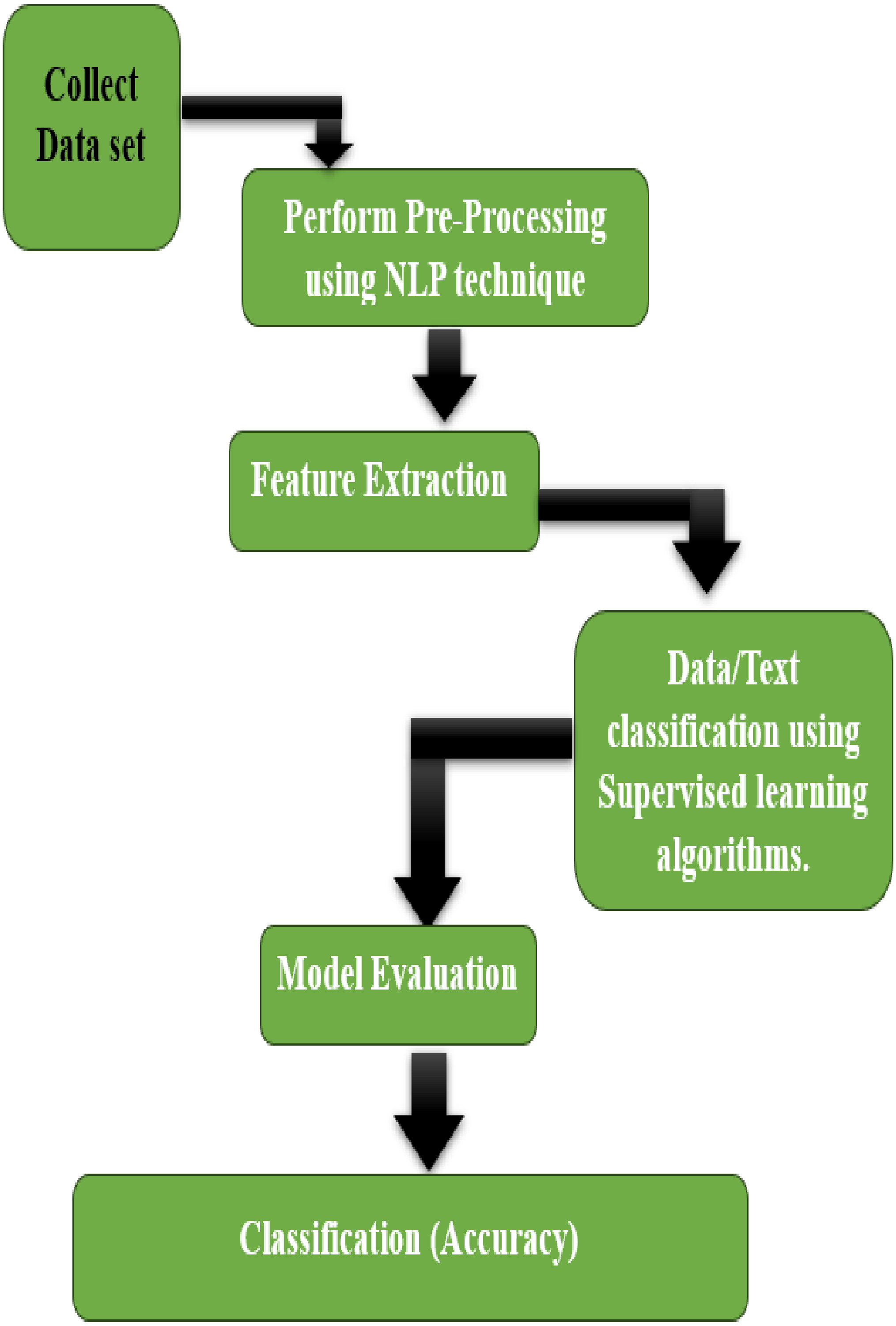


Fig-25: Architecture

CHAPTER 4

4. EXPERIMENTATIONS AND RESULTS

This Research utilizes Python version 3 along with essential built-in libraries. For implementing the machine learning model, Python 3 is chosen due to its flexibility and computational power. It leverages various Python libraries such as Scikit-learn, Matplotlib, Pandas, Seaborn, NumPy, among others.

Table 4: Confusion Matrix

		ACTUAL	
		YES	NO
PREDICTED	YES	TP	FP
	NO	FN	TN

- Accuracy, which represents the proportion of correctly classified cases out of all cases, is calculated using the following formula:

  Accuracy = (TP + TN) / (TP + TN + FP + FN)

- Precision is expressed as the proportion of positive cases that are correctly recognized as positive over all cases classified as positive and it is calculated according to the formula:

  Precision = TP / (TP + FP)

- Recall is expressed as the proportion of positive cases that are correctly recognized as positive over all actual positive cases and is calculated according to the formula:
- **Recall = TP / (TP + FN)**

- The F1 score is a metric used to evaluate the performance of a classification model. It considers both the precision and recall of the model to compute a single score. The formula for the F1 score is:

  F1 = 2 × (Precision × Recall) / (Precision + Recall)

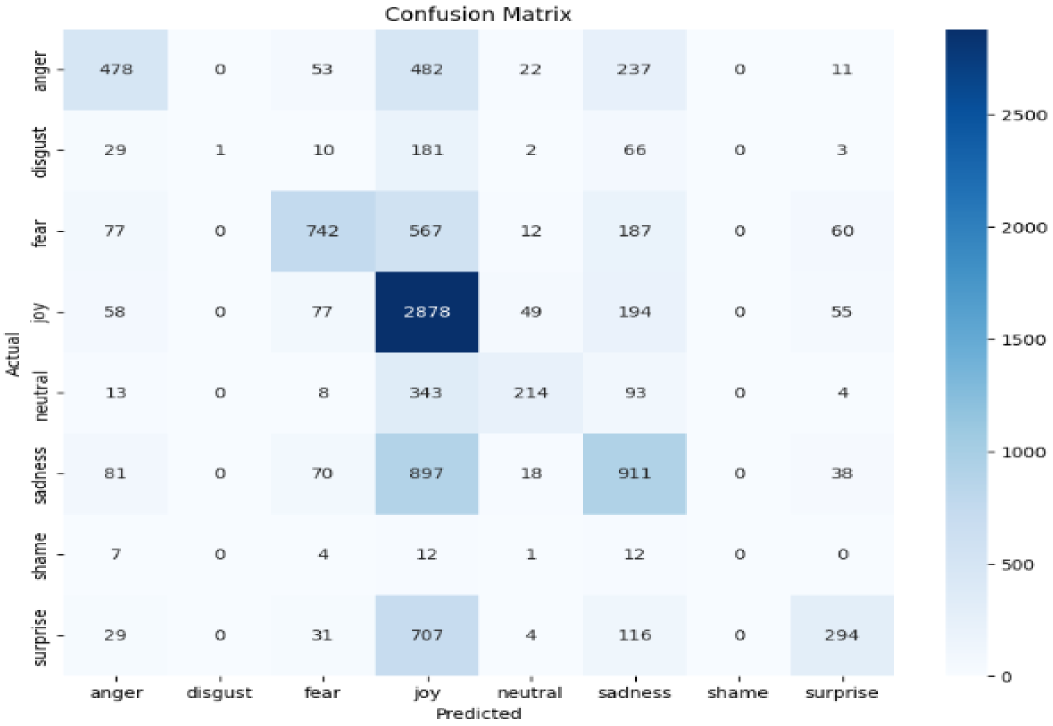
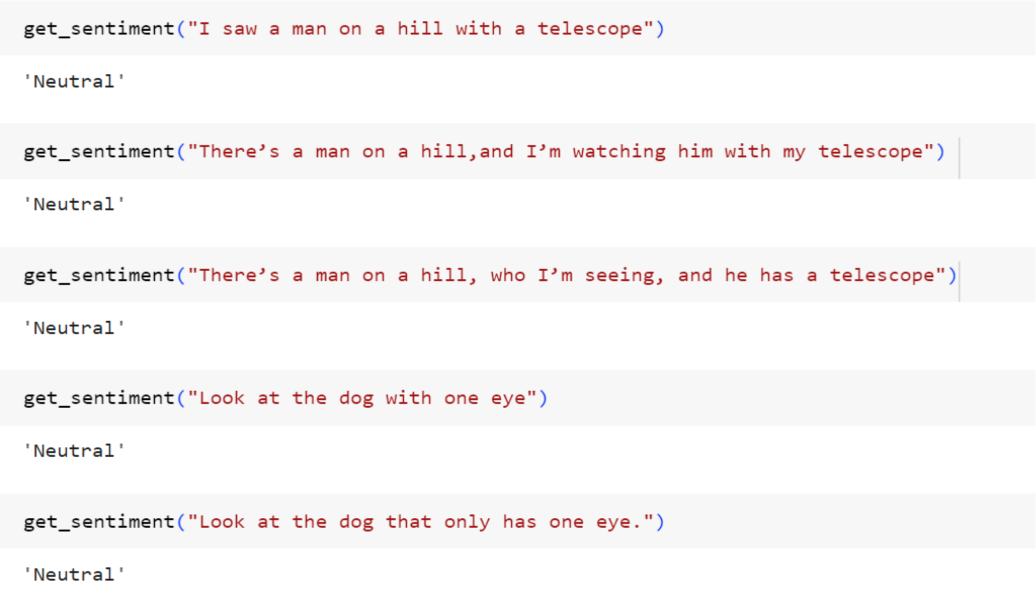


Fig-26: Heat Map of Confusion Matrix

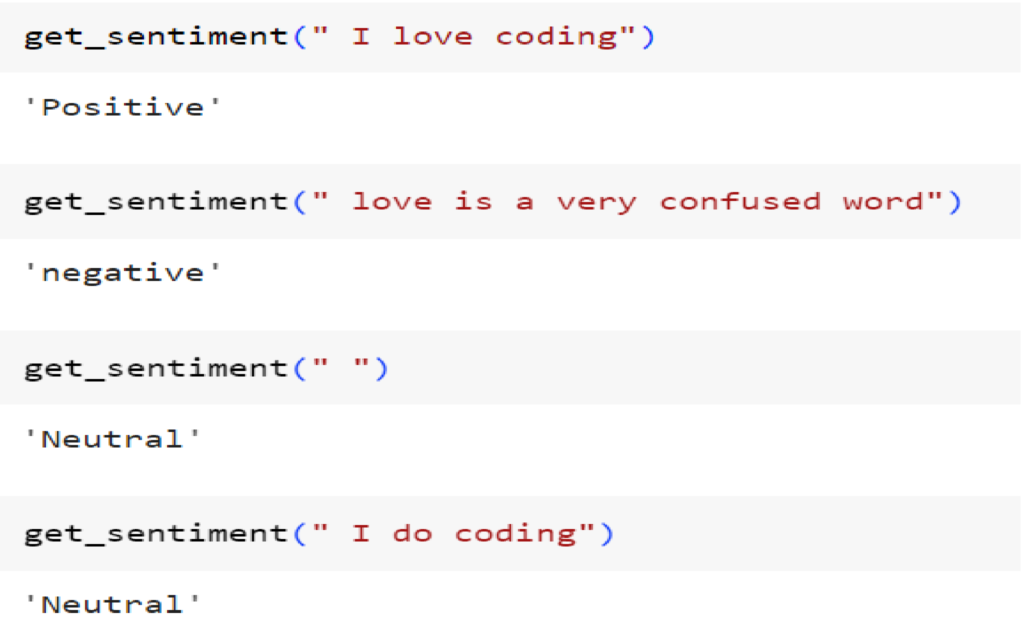
The sentiment detection model, named get (), is utilized for analyzing emotions in text data. It operates accurately on keywords and sentences, extracting emotions effectively. With a dataset comprising 34,791 entries and over 300,000 words, this model demonstrates superior performance compared to previous algorithms.

Notably, the algorithm addresses the issue of word ambiguity. Ambiguous sentences are examined to assess the algorithm's ability to detect emotions from ambiguous words. This robust approach ensures precise emotion detection even in complex linguistic contexts. Followings are some ambiguity sentences and getting result after detecting sentiment:



```
get_sentiment("I saw a man on a hill with a telescope")

'Neutral'

get_sentiment("There’s a man on a hill,and I’m watching him with my telescope")

'Neutral'

get_sentiment("There’s a man on a hill, who I’m seeing, and he has a telescope")

'Neutral'

get_sentiment("Look at the dog with one eye")

'Neutral'

get_sentiment("Look at the dog that only has one eye.")

'Neutral'
```

Fig-27: Results after detecting Sentiment from ambiguity word



```
get_sentiment(" I love coding")

'Positive'

get_sentiment(" love is a very confused word")

'negative'

get_sentiment(" ")

'Neutral'

get_sentiment(" I do coding")

'Neutral'
```

Fig-28: Results after detecting sentiment from any text.

Three types of sentiments are detected by applying proposed algorithm. These are:

a. Neutral.
b. Positive.
c. Negative.

The below figure shown as Sentiment and emotion distribution in Dataset:

The Classification Report for various algorithms as follows

Classification Performance Report

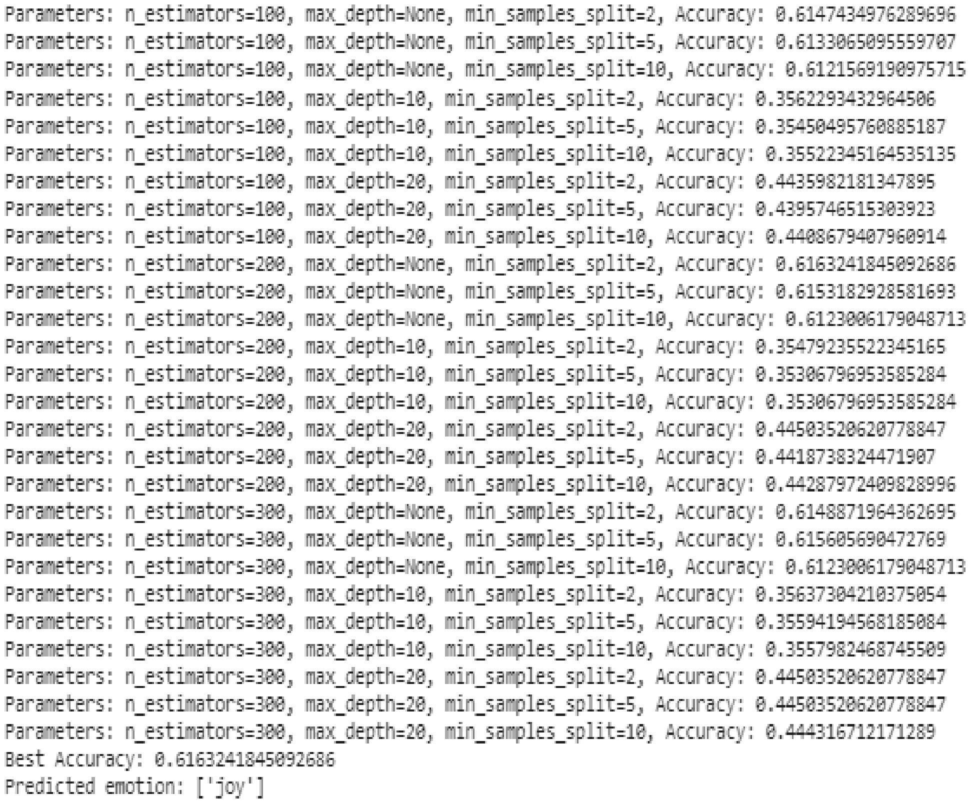
This report provides a comprehensive evaluation of the classification quality achieved by a machine learning model. It encompasses five main columns and (N+3) rows. The initial column lists the class labels, followed by Precision, Recall, F1-score, and Support metrics.

- **Class Label**: Identifies the specific class being evaluated.
- **Precision**: Indicates the accuracy of the model's predictions for a given class. Precision is calculated as the ratio of true positives to the total predicted positives, representing how many of the predicted instances of a class are actually relevant.
- **Recall**: Reflects the model's ability to correctly identify instances of a class within the dataset. Recall is calculated as the ratio of true positives to the total actual positives, illustrating the proportion of actual instances of a class that were correctly identified by the model.
- **F1-score**: Represents the harmonic mean of Precision and Recall. It provides a single metric that balances both Precision and Recall, offering a holistic measure of a model's performance for a specific class.

- **Support:** Denotes the total number of instances belonging to each class within the actual dataset. It is the sum of the rows corresponding to each class.
- The report comprises N rows, each corresponding to a unique class label, and three additional rows providing metrics for overall performance: Accuracy, Macro Average, and Weighted Average.
- **Accuracy:** Measures the overall correctness of the model across all classes, calculated as the ratio of correct predictions to the total number of predictions.
- **Macro Average**: Represents the unweighted mean of Precision, Recall, and F1-score across all classes. It gives equal importance to each class, irrespective of class frequency.
- **Weighted Average:** Computes the weighted average of Precision, Recall, and F1-score, considering the support (number of instances) for each class. It provides a performance measure that accounts for class imbalances in the dataset.
- This structured report offers a clear understanding of the classification performance of the constructed ML model, facilitating informed decision-making and model optimization efforts.

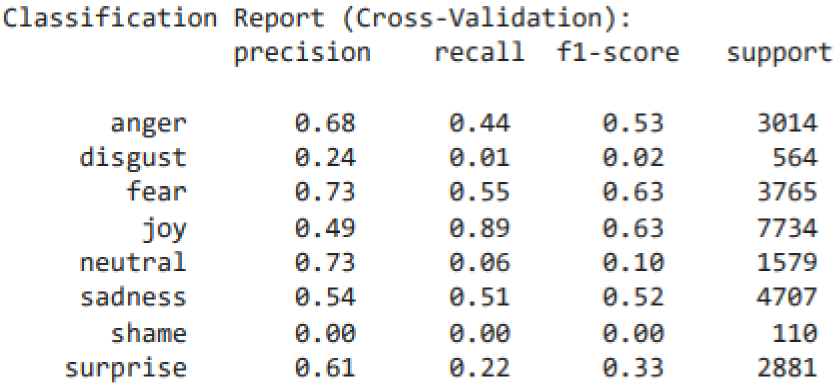
Classification Report

For TF-IDF



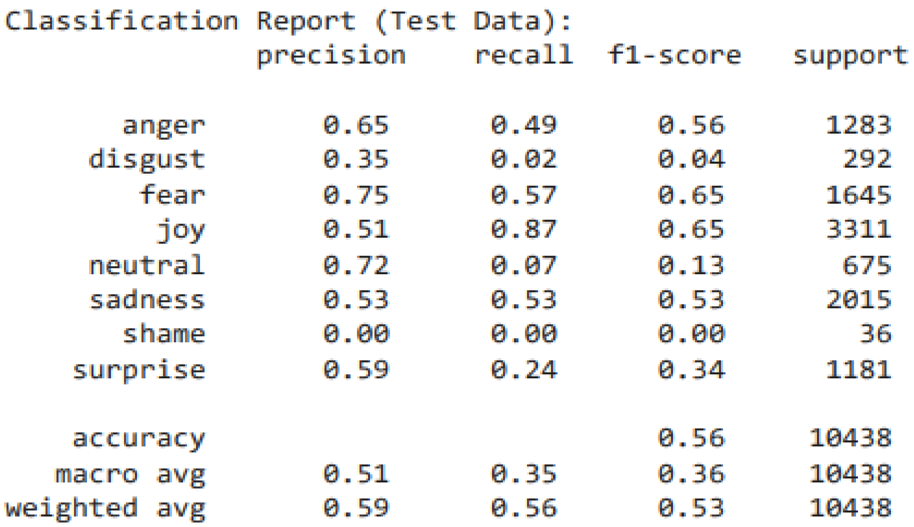
```
Parameters: n_estimators=100, max_depth=None, min_samples_split=2, Accuracy: 0.6147434976289696
Parameters: n_estimators=100, max_depth=None, min_samples_split=5, Accuracy: 0.6133065095559707
Parameters: n_estimators=100, max_depth=None, min_samples_split=10, Accuracy: 0.6121569190975715
Parameters: n_estimators=100, max_depth=10, min_samples_split=2, Accuracy: 0.3562293432964506
Parameters: n_estimators=100, max_depth=10, min_samples_split=5, Accuracy: 0.35450495760885187
Parameters: n_estimators=100, max_depth=10, min_samples_split=10, Accuracy: 0.35522345164535135
Parameters: n_estimators=100, max_depth=20, min_samples_split=2, Accuracy: 0.4435982181347895
Parameters: n_estimators=100, max_depth=20, min_samples_split=5, Accuracy: 0.4395746515303923
Parameters: n_estimators=100, max_depth=20, min_samples_split=10, Accuracy: 0.44086794079609 14
Parameters: n_estimators=200, max_depth=None, min_samples_split=2, Accuracy: 0.6163241845092686
Parameters: n_estimators=200, max_depth=None, min_samples_split=5, Accuracy: 0.6153182928581693
Parameters: n_estimators=200, max_depth=None, min_samples_split=10, Accuracy: 0.6123006179048713
Parameters: n_estimators=200, max_depth=10, min_samples_split=2, Accuracy: 0.35479235522345165
Parameters: n_estimators=200, max_depth=10, min_samples_split=5, Accuracy: 0.35306796953585284
Parameters: n_estimators=200, max_depth=10, min_samples_split=10, Accuracy: 0.35306796953585284
Parameters: n_estimators=200, max_depth=20, min_samples_split=2, Accuracy: 0.44503520620778847
Parameters: n_estimators=200, max_depth=20, min_samples_split=5, Accuracy: 0.4418738324471907
Parameters: n_estimators=200, max_depth=20, min_samples_split=10, Accuracy: 0.44287972409828996
Parameters: n_estimators=300, max_depth=None, min_samples_split=2, Accuracy: 0.6148871964362695
Parameters: n_estimators=300, max_depth=None, min_samples_split=5, Accuracy: 0.615605690472769
Parameters: n_estimators=300, max_depth=None, min_samples_split=10, Accuracy: 0.6123006179048713
Parameters: n_estimators=300, max_depth=10, min_samples_split=2, Accuracy: 0.35637304210375054
Parameters: n_estimators=300, max_depth=10, min_samples_split=5, Accuracy: 0.35594194568185084
Parameters: n_estimators=300, max_depth=10, min_samples_split=10, Accuracy: 0.3557982468745509
Parameters: n_estimators=300, max_depth=20, min_samples_split=2, Accuracy: 0.44503520620778847
Parameters: n_estimators=300, max_depth=20, min_samples_split=5, Accuracy: 0.44503520620778847
Parameters: n_estimators=300, max_depth=20, min_samples_split=10, Accuracy: 0.444316712171289
Best Accuracy: 0.6163241845092686
Predicted emotion: ['joy']
```

Fig-29: Classification Report for RFC



```
Classification Report (Cross-Validation):
              precision    recall  f1-score   support

       anger       0.68      0.44      0.53      3014
     disgust       0.24      0.01      0.02       564
        fear       0.73      0.55      0.63      3765
         joy       0.49      0.89      0.63      7734
     neutral       0.73      0.06      0.10      1579
     sadness       0.54      0.51      0.52      4707
       shame       0.00      0.00      0.00       110
    surprise       0.61      0.22      0.33      2881
```

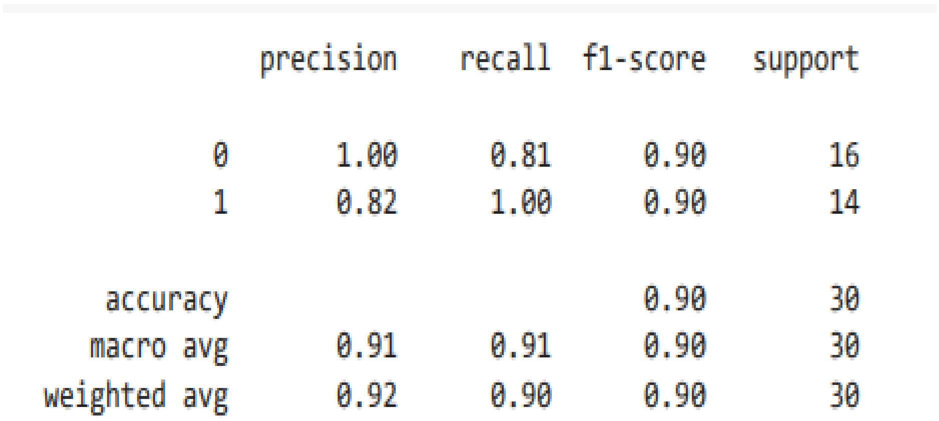
Fig- 30: Classification Report for KNN



```
Classification Report (Test Data):
              precision    recall  f1-score   support

       anger       0.65      0.49      0.56      1283
     disgust       0.35      0.02      0.04       292
        fear       0.75      0.57      0.65      1645
         joy       0.51      0.87      0.65      3311
     neutral       0.72      0.07      0.13       675
     sadness       0.53      0.53      0.53      2015
       shame       0.00      0.00      0.00        36
    surprise       0.59      0.24      0.34      1181

    accuracy                           0.56     10438
   macro avg       0.51      0.35      0.36     10438
weighted avg       0.59      0.56      0.53     10438
```

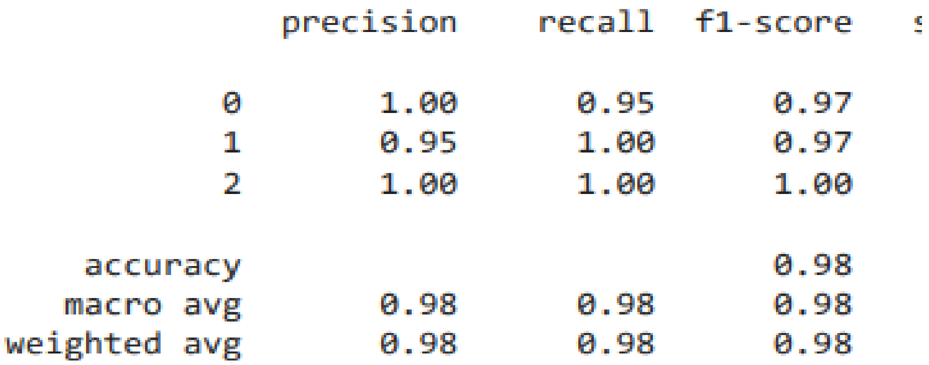
Fig-31: Classification Report for MNB

For Count Vectorizer



```
              precision    recall  f1-score   support

           0       1.00      0.81      0.90        16
           1       0.82      1.00      0.90        14

    accuracy                           0.90        30
   macro avg       0.91      0.91      0.90        30
weighted avg       0.92      0.90      0.90        30
```

Fig-32: Classification Report for NB



```
              precision    recall  f1-score

           0       1.00      0.95      0.97
           1       0.95      1.00      0.97
           2       1.00      1.00      1.00

    accuracy                           0.98
   macro avg       0.98      0.98      0.98
weighted avg       0.98      0.98      0.98
```

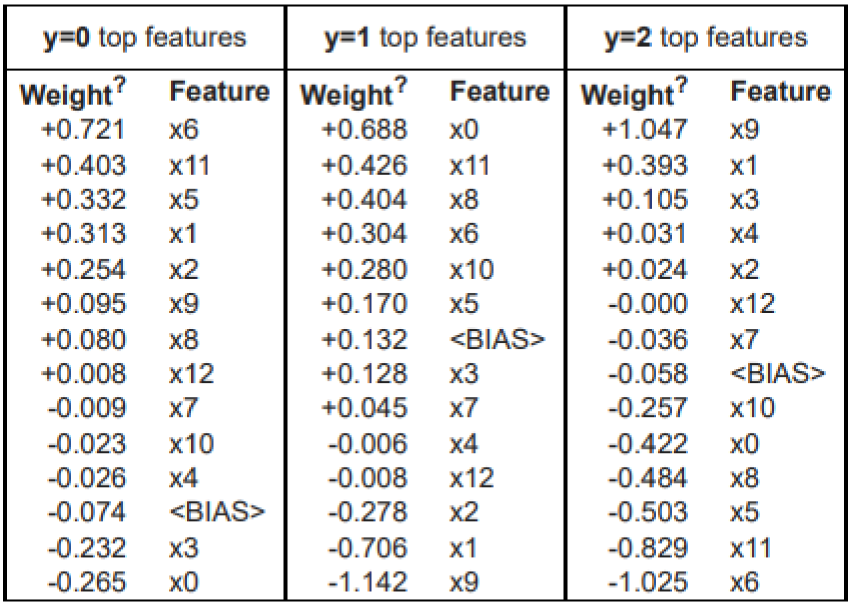
Fig-33: Classification Report for RFC (Case of CV)

Logistic Regression

Accuracy- 96%

Features Level eli5 interpretation

Table 5: Feature Level Eli5 Interpretation



y=0 top features		**y=1** top features		**y=2** top features	
Weight?	**Feature**	**Weight?**	**Feature**	**Weight?**	**Feature**
+0.721	x6	+0.688	x0	+1.047	x9
+0.403	x11	+0.426	x11	+0.393	x1
+0.332	x5	+0.404	x8	+0.105	x3
+0.313	x1	+0.304	x6	+0.031	x4
+0.254	x2	+0.280	x10	+0.024	x2
+0.095	x9	+0.170	x5	-0.000	x12
+0.080	x8	+0.132	<BIAS>	-0.036	x7
+0.008	x12	+0.128	x3	-0.058	<BIAS>
-0.009	x7	+0.045	x7	-0.257	x10
-0.023	x10	-0.006	x4	-0.422	x0
-0.026	x4	-0.008	x12	-0.484	x8
-0.074	<BIAS>	-0.278	x2	-0.503	x5
-0.232	x3	-0.706	x1	-0.829	x11
-0.265	x0	-1.142	x9	-1.025	x6

```
Text(0, 0.5, 'count')
```

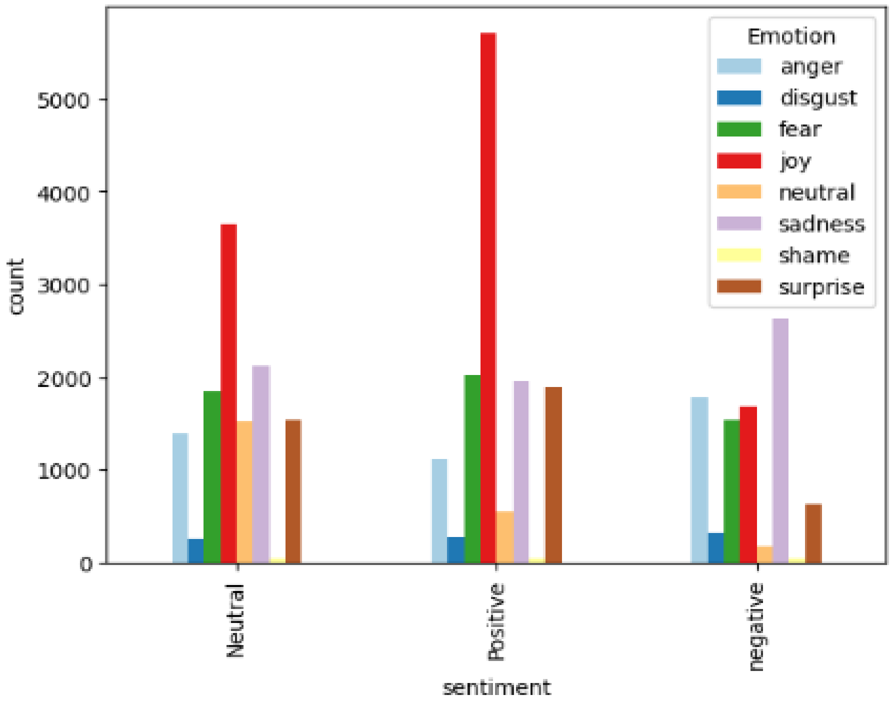


Fig-34: Sentiment distribution

```
Text(0, 0.5, 'count')
```

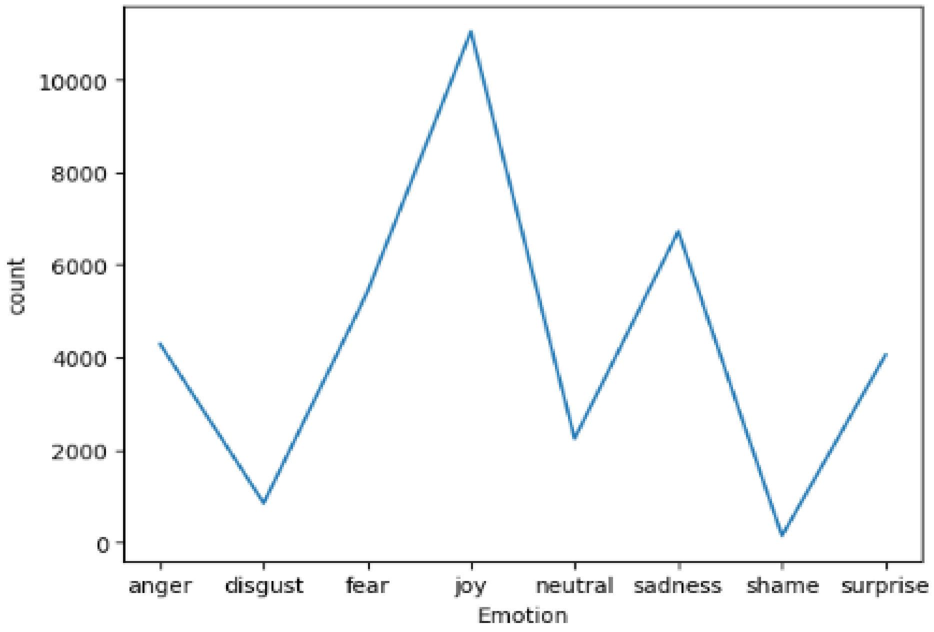


Fig-35: Frequency of Emotions over time

By discussing previous related- work, it is clear that existing systems are not 100% accurate. In previous, the existing system is based on multinomial naïve bayes algorithm, KNN algorithm and also logistic regression, Ad boost classifier etc. Among this four-algorithm provided accuracy as high as 68%, 64.8%,65% & 67.08% respectively. Multinomial Naïve bayes Algorithm resulted the best performance which an average accuracy of 68% but failed to perform well because a compact research project needs minimum above 80-85%. But proposed system like Bernoulli naïve bayes algorithm and gaussian algorithm has been solved the problem of multinomial naïve bayes algorithm and resulted very good result such as 89% accuracy for Bernoulli and 88% for gaussian naïve bayes algorithm.

This research also investigates the effectiveness support Vector classifier, linear regression, gradient boost classifier, Random Forest classifier, Naïve bayes algorithm and also

unsupervised learning algorithm like K-Means algorithm etc. The study was carried out on "Emotion dataset" with eight emotional groups. In machine learning the detection of textual emotions is the problem of content-based classification, it is the task of natural language processing. Detecting a person's emotion is a difficult task but we propose for emotion in English sentences where emotions are treated as generalized concepts extracted from sentences. Here present research worker considered ISEAR dataset with 34791 records where emotions are usually expressed as joy, sadness, neutral, anger, fear, surprise, shame, disgust etc. Existing sentiment detection model defined by previous researcher is not well capable to detect sentiment from emotional dataset, for reason of that some vital problems like ambiguity in keyword, Incapability to recognize emotion in absence of emotion keyword were there. Present research worker proposed a new sentiment detection model get () sentiment model which has solved all mentioned problems and worked more well and accurate.

Table-6: Accuracy of Previously Used Methods

Previous Method	**Accuracy**
Multinomial Naïve bayes	**68%**
KNN	**64.80%**
Logistic Regression	**65%**
Ada boost Classifier	**67.08%**

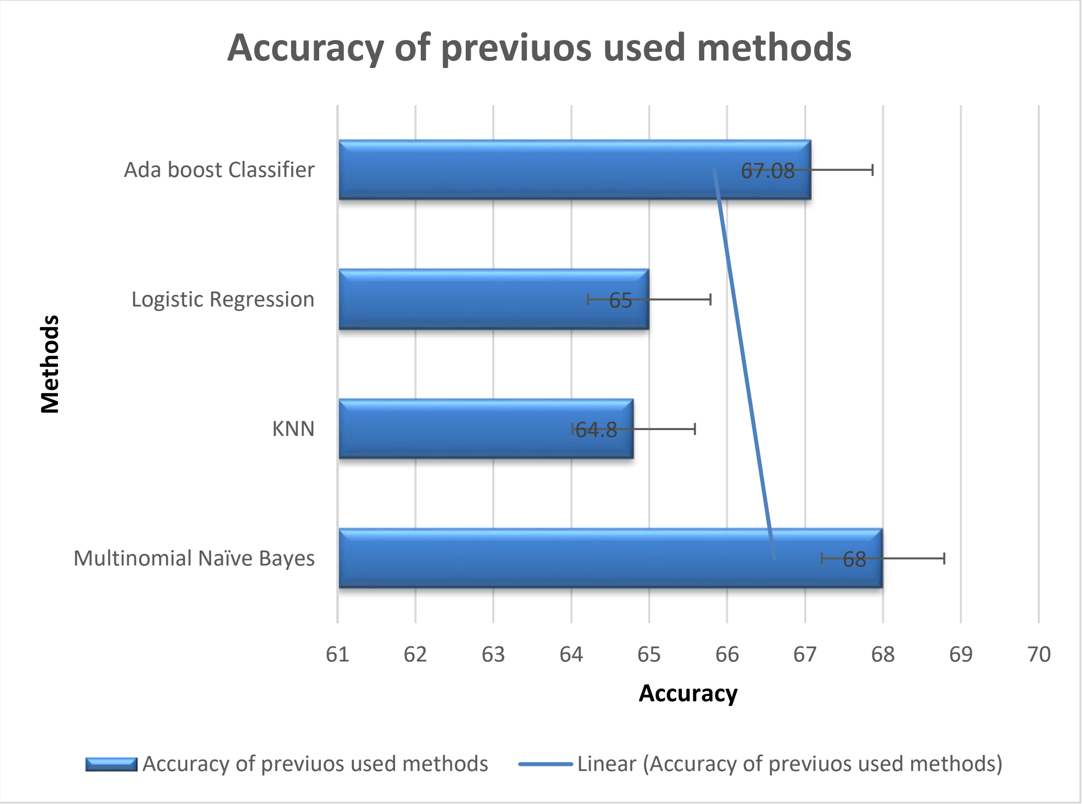


Fig-36: Accuracy of previously used methods

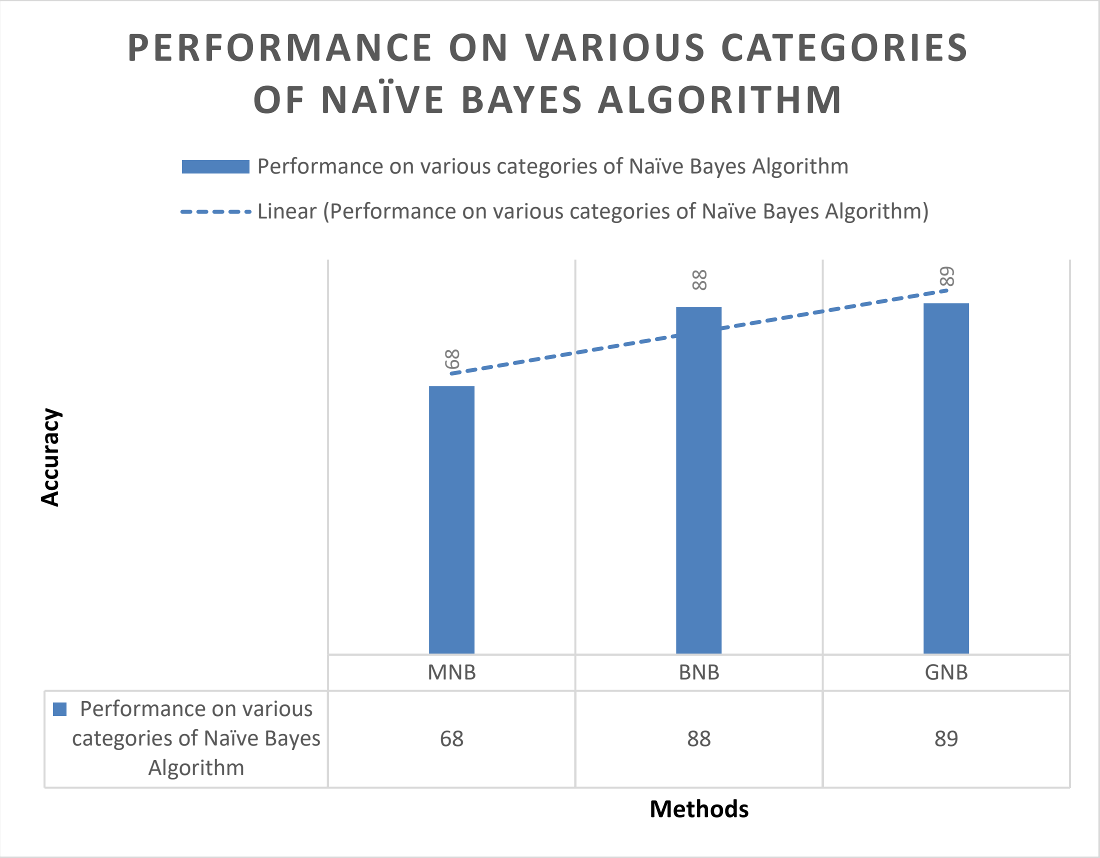


Fig-37: Accuracy Performance on various categories of Naïve Bayes Algorithm

In the feature extraction step of the machine learning algorithm, both TF-IDF and Count Vectorizer methods are employed to detect emotion and sentiment. Typically, TF-IDF is expected to yield higher accuracy during the classification step compared to Count Vectorizer. However, in this study, Count Vectorizer surprisingly outperformed TF-IDF.

Previous researchers found that TF-IDF struggled to achieve accuracy above 30% during the classification step. Interestingly, in this research, TF-IDF did not demonstrate superiority over Count Vectorizer either, but it did manage to achieve higher accuracy, surpassing 60%. On the other hand, Count Vectorizer exhibited significant improvement compared to previous studies, providing very good results.

Table 7: The results are summarized in the following

FEATURS	Dataset Split Ratio	Accuracy Obtained						
		SVM	Random forest	Logistic Regression	BNB	GNB	MNB	KNN
TF-IDF	70:30	63.7	70	65	63	65	58	55
	80:20	64	72	65.45	64	65	60	58
Count Vectorizer	70:30	95	98	96	93	93	70	74

Table-8: Classification Report of all proposed machine learning algorithm:

Proposed method	Precision	Recall	F-1 score	Accuracy
Gaussian Naïve Bayes (GNB)	0.87	0.94	0.90	0.89
Bernoulli Naïve Bayes (BNB)	0.86	0.92	0.89	0.90
KNN	0.72	0.76	0.68	0.70
Random Forest	0.93	1.0	0.91	0.98
Logistic Regression	0.93	0.92	0.92	0.94
Support Vector Machine (SVM)	0.75	0.86	0.83	0.84

CHAPTER 5

5. COMPARATIVE ANALYSIS

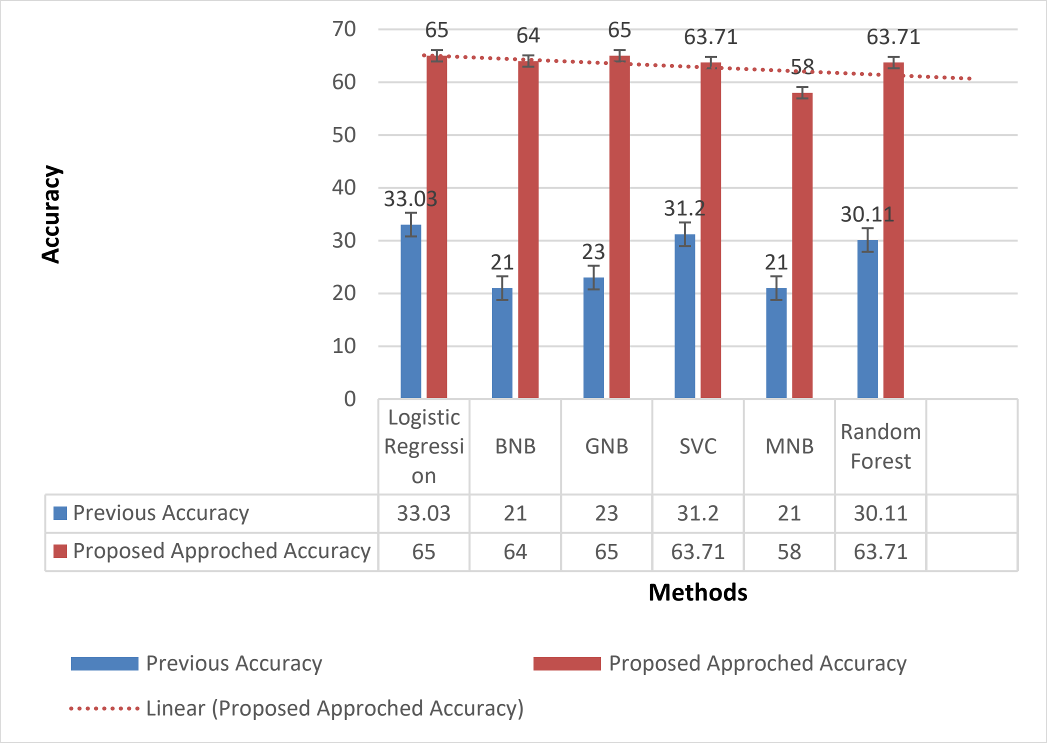
In the Comparative Accuracy Analysis employing the TF-IDF feature extraction method across various machine learning models such as Logistic Regression, Random Forest, BNB, GNB, SVC, and KNN, Random Forest emerged with the highest classification accuracy. However, it fell short in comparison to the results obtained using Count Vectorizer. Here's a juxtaposition between the findings of the previous and present studies regarding classification accuracy.

Accuracy Obtained by Existing Researchers:

For the Case of TF-IDF:

Table 9: Comparison between previous and presently obtained accuracy for TF-IDF

Algorithms	Previous Accuracy	Presently obtained Accuracy
Logistic Regression	33.03%	65%
BNB	21%	64%
GNB	23%	65%
SVC	31.2%	63.71%
MNB	21%	58%



	Logistic Regression	BNB	GNB	SVC	MNB	Random Forest
🟦 Previous Accuracy	33.03	21	23	31.2	21	30.11
🟥 Proposed Approched Accuracy	65	64	65	63.71	58	63.71

Fig-38: Comparative Analysis of Previous Model with Proposed method

For the Case of Count Vectorizer:

Table 10: Comparison between previous and presently obtained accuracy for Count Vectorizer

Algorithms	Previous Accuracy	Presently obtained Accuracy
Logistic Regression	89.01%	95%
BNB	67%	93%
GNB	68%	93%
SVC	88.35%	98%
MNB	56%	70%

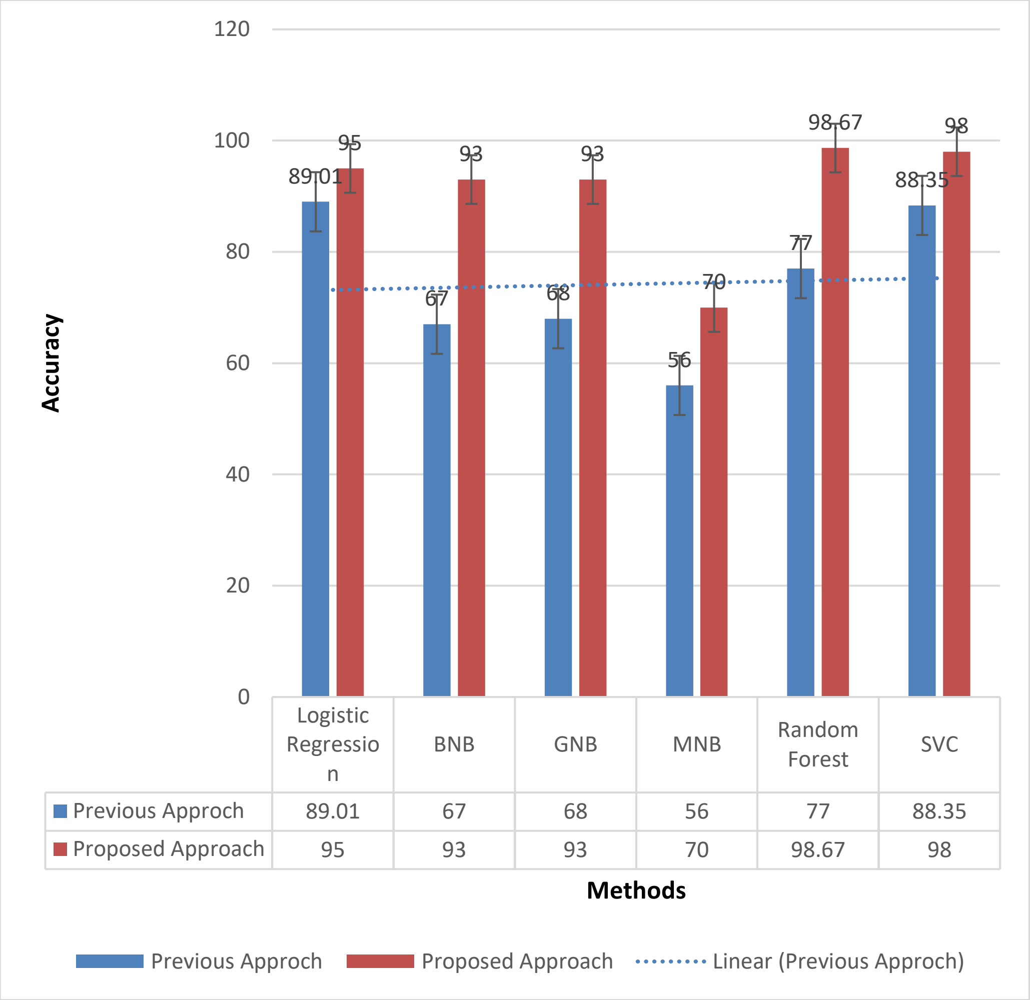


Fig-39: Comparative Analysis of Previous model with Proposed method for Count Vectorizer

So, for comparative analysis on both feature engineering, Random Forest gives best accuracy.

CHAPTER 6

6. CONCLUSION & FUTURE SCOPES

6.1 Conclusion

In this work, a text-based emotion detection algorithm has been proposed, that is capable of identifying each word in textual data one by one. Several supervised algorithms were used: K-Nearest Neighbors (KNN), Multinomial Naive Bayes (MNB), Bernoulli Naive Bayes (BNB), Gaussian Naive Bayes (GNB), Support Vector Machine (SVM), and Random Forest Classifier.

Two different feature extraction techniques have been employed: Count Vectorizer (CV) and Term Frequency-Inverse Document Frequency (TF-IDF). Initially, using the base model on a previous dataset, it was observed that the highest accuracy obtained with Count Vectorizer at 88%, and with TF-IDF at 33.03%.

After data augmentation is applied and the proposed algorithms have been used with Count Vectorizer, as a result a significant improvement in accuracy was achieved. The highest accuracy of 98% is achieved by the Random Forest Classifier, which represents the best result.

6.2 Future Scopes

Future work consists of experimenting like:

1. In Future, this work can be further extended for designing an emotion detection model with more improvement on feature extraction methods using other machine learning models like LSTM model, Linear Regression, Non-Linear SVC to get more classification accuracy.
2. Besides this research work, further enhancement can also be done using CNN, RNN and get a comparative study between Neural Networks and Machine

Learning Models.

3. Unsupervised Machine Learning Algorithms can be implemented for this work.
4. The accuracy of the model can also be increased by creating a customized database to the model and training the textual data on a larger database.

REFERENCES

[1] Vishaka Singh, Anushka Shirode, Manasi Sharma, Sanjay Mirchandani, "Text Emotion Detection using Machine Learning Algorithms", Proceedings of the 8th International Conference on Communication and Electronics Systems, IEEE Xplore part number: CFP23AWO-ART; 2023.

[2] Ms. Pinal Solanki, "A study on Emotion Detection & Classification from Text using Machine Learning", Journal of Artificial Intelligence, Machine Learning and Neural Network, Vol. 02, No. 02,2022.

[3] Syeda Nadia Firdaus, Chen Ding, Alireza Sadeghian, "Topic specific emotion detection for retweet prediction", International Journal of Machine Learning and Cybernetics, vol. 10, pp.2071-2083, 2019

[4] Andry Chowanda, Rhio Sutoyo, Meilliana, Sansari Tanachutiwat, "Exploring Text-based Emotions Recognition Machine Learning Techniques on Social Media Conversion", 5th International Conference on Computer Science and Computational Intelligence, ScienceDirect, Procedia Computer Science 179, 821-828, 2021.

[5] Shaikh Abdul Salam, Rajkumar Gupta, "Emotion Detection and Recognition from Text using Machine Learning", International Journal of Computer Science and Engineering, vol. 6, issue-6,2018.

[6] P Ancy Grana et al., “Sentiment Analysis of Text using Machine Learning Models”, International Research Journal of Modernization in Engineering Technology and Science, vol. 04,2022.

[7] Kristina Machova, Martina Szaboova, Jan Paralic, Jan Micko, “Detection of emotion by text analysis using machine learning”, Frontliers,2023.

[8] Amal Shameem, Rameshbabu G, Vigneshwaran L, Sundar K, Mrs. k. Veena, “Text Emotion Detection Using Machine Learning And NLP”, International Journal of Scientific Research in science, Engineering and Technology, vol. 9, Issue.,2022.

[9] V V Ramalingam, A Pandian, Abhijit Jaiswal, Nikhar Bhatia, “Emotion Detection from Text”, National Conference on Mathematical Techniques and its Application, Conf Series 1000, 2018.

[10] Poonam Arya, Shilpa Jain, “Text Based Emotion Detection”, International Journal of Computer Engineering and Technology, vol. 9, Issue. 3, pp. 95-104, 2018.

[11] Goru Swathi, Behara Meghna Pathak, Arjala Janani, Kanithi Karthik, Janni Divya, M. Jayanthi Rao, “Emotion Recognition from Text using Machine Learning”, International Journal of Food Science and Nutritional Sciences, vol.11, Issue. 12, 2022.

[12] Nabeela Altrabshesh, Mihaela Cocea, Sanaz Fallahkhair, “Predicting Learning-related Emotions from students’ textual classroom feedback via Twitter”, Processing’s of the 8th International Conference on Educational Data Mining, 2018.

[13] S. Arun Kumar s., A. Geetha, “Emotion Detection from Text using Natural Language Processing and Neural Networks”, International Journal of Intelligent Systems and Applications in Engineering, 2024.

[14] Chetan R. Chopade, “Text Based Emotion Recognition: A Survey”, International Journal of Science and Research, vol. 4, issue. 6, 2015.

[15] https://en.wikipedia.org/wiki/Sentiment_analysis

[16] Muskan Garg, Chandni Saxena, Emotion detection from text data using machine learning for human behavior analysis, Editor(s): D. Jude Hemanth, Computational Intelligence Methods for Sentiment Analysis in Natural Language Processing Applications, Morgan Kaufmann,2024.

[17] Younis, E.M.G., Mohsen, S., Houssein, E.H., “Machine learning for human emotion recognition: a comprehensive review”, Neural Compute & Applica ,2024.

[18] Bhavya, A.V., Dhanush, R.H., Sangeetha, J., Jose, A.C. Personalized Emotion Detection from Text Using Machine Learning. In: Marmolejo-Saucedo, J.A., Rodríguez-Aguilar, R., Vasant, P., Litvinchuk, I., Retana-Blanco, B.M. (eds) Computer Science and Engineering in Health Services. COMPSE 2022. EAI/Springer Innovations in Communication and Computing. Springer, Cham,2022.

[19] Nath, S., Shahi, A.K., Martin, T., Choudhury, N., Mandal, R. A Comparative Study on Speech Emotion Recognition Using Machine Learning. In: Tavares, J.M.R.S., Rodrigues, J.J.P.C., Misra, D., Bhattacharjee, D. (eds) Data Science and Communication. ICTDsC 2023. Studies in Autonomic, Data-driven and Industrial Computing. Springer, Singapore,2023.

[20] Amal Shameem, Rameshbabu G, Vigneshwaran L, Sundar K, Mrs. k. Veena, "Text Emotion Detection Using Machine Learning And NLP", International Journal of Scientific Research in science, Engineering and Technology, vol. 9, Issue. 3,2022.

[21] https://www.linkedin.com/pulse/machine-learning-basics-support-vector-machines-amsal-gilani/

[22] Jianhua Tao, "Context Based Emotion Detection from Text Input", National Laboratory of Pattern Recognition Institute of Automation, Chinese Academy of Sciences, Beijing, China,2018.

[23] http:/www.geeksforgeeks.adochub.com.

[24] https://www.ejable.com/tech-corner/ai-machine-learning-and-deep-learning/logistic-and-linear-regression/.

[25] https://www.semanticscholar.org/paper/Study-of-Various-Methods-forTokenization-Rai-Borah/8a8110ba524c6a58baeba442478a877fd6d2c261.

[26] https://www.javatpoint.com/machine-learning.

Appendix part A

➢ System Requirements

Personal Computer or laptop having the following features –

- Windows 10
- 132 GB RAM
- Internet Connection
- 15.6” HD display

➢ Software Requirements

- Browser: Windows internet explorer and Google Chrome
- Microsoft word 2019
- Google Colab
- MS Excel

➢ Programming Requirements

- Python

➢ Download Speeds

- Internet speed is measured in Mbps
- 3 -5 Mbps is recommended.

➢ Loading Testing Tool

- Google Browser

Appendix part B

```
###load pkages
import pandas as pd
import numpy as np
#from matplotlib import pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
#text cleaning
#python get-pip.py
!pip install neattext
#mpip.install.neattext
import neattext.functions as nfx
#import neattext as nt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
```

Load Dataset

```
# load dataset
df = pd.read_csv("Emotion Dataset.csv")
from google.colab import drive
drive.mount('/content/drive')
```

Dataset Description

```
df
# @title Emotion

df.head()

print(df.head())
df.head(10)
df.head(100)
print(df.head(50))
df.tail(50)
print(df.tail(50))
print(df.tail())
print('The train dataset contans {} rows and {}
columns'.format(df.shape[0], df.shape[1]))
print(df['Emotion'].unique())



from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Emotion').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Emotion').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
df.info()
```

Show Sample Tweets

```
# show sample tweets
for tweet in df["Text"][:5]:
  print(f"- {tweet}")

# show sample tweets
for tweet in df["Text"][:50]:
  print(f"- {tweet}")
# show sample of tweets with a specific emotion
for i,row in df[df["Emotion"] == "joy"].iterrows():
  print(f'- {row["Text"]}')
for i,row in df[df["Emotion"] == "sadness"].iterrows():
  print(f'- {row["Text"]}')

# show sample tweets
for tweet in df["Text"][:5000]:
  print(f"- {tweet}")
```

Plot Emotion Found in Tweets

```
# plot emotions found in tweets
plot_title = f"Emotions found in tweets about"
fig = px.histogram(df, x="Emotion", template="plotly_dark",
                   title=plot_title, color="Emotion")
fig.update_layout(showlegend=False)
fig.show()
```

Distribution of Emotions

```
# @title Distribution of Emotions

df['Emotion'].value_counts().plot(kind='bar')
```

Null Value Checking

```
print("null values",df.isnull().sum().sum())
```

Data Cleaning

```
duplicates_count = df.duplicated().sum()
print(f'Total duplicated rows: {duplicates_count}')
df[df['Emotion'].duplicated() == True]
# Unique values from 'sentiment'
unique_sentiments = df['Emotion'].unique()
print(unique_sentiments)
### Frequency distribution of all diffrent types of Emotion
```

```
frequency_counts = df['Emotion'].value_counts()

frequency_percentage = (frequency_counts / len(df['Emotion'])) * 100
frequency_df = pd.DataFrame({'Counts': frequency_counts, 'Percentage':
frequency_percentage})

print(frequency_df)

# Print total value
cardinality = df['Emotion'].nunique()
print(f"\ntotal values: {cardinality}")
### Covertion of float
import pandas as pd

# Create a DataFrame with strings containing commas
#df = pd.DataFrame({'values': ['1,234', '56,78', '9,100', '3.14']})

# Use the `replace()` function to remove commas
#df['Clean_Text'] = df['Clean_Text'].replace(any=True)

# Convert the column to floats
#df['Clean_Text'] = df['Clean_Text'].astype(float)

# Print the DataFrame
#print(df)
print(df.info())
print(df.columns)
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Drop Null Value and Plot Emotion

```
df=df.dropna(axis=0,how="any")
df

from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Emotion').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)

from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Emotion').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
from matplotlib import pyplot as plt
import seaborn as sns
```

```
df.groupby('Emotion').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
df.isnull().sum()
df.shape
df.dtypes

# value count
df['Emotion'].value_counts()


#df['Emotion'].value_counts('joy')
# Print the DataFrame
print(df)

# Print the DataFrame columns
print(df.columns)
# Check if the column name is misspelled
if 'Emotion' not in df.columns:
    # Find the closest match to the intended column name
    closest_match = df.columns[df.columns.str.contains('Emotion',
case=False, regex=True)].tolist()

    if closest_match:
        print(f"Did you mean '{closest_match[0]}'?")
    else:
        print("Column 'Emotion' not found in the DataFrame.")
        # value count
#df['Closest_match_to_Emotion'].value_counts()
#!pip install --upgrade matplotlib
import matplotlib.pyplot as plt
import numpy as np

df['Emotion'].value_counts('joy').plot(kind='pie')
plt.title("Emotion distribution Report")
plt.show()
```

Data Pre-Processing

```
#Install pywsd
!pip install pyws
!pip install textblob
from textblob import TextBlob
!pip install nltk
import nltk
nltk.download('punkt')
import nltk
nltk.download('stopwords')
import nltk
```

```
nltk.download('wordnet')
import pandas as pd

# Read the dataset
df = pd.read_csv('Emotion Dataset.csv')  # Replace 'your_dataset.csv'
with your actual dataset file path

# Apply preprocessing function to the 'text' column
df['Clean2_text'] = df['Text'].apply(preprocess_text)

# Save the preprocessed data
df.to_csv('preprocessed_dataset.csv', index=False)  # Replace
'preprocessed_dataset.csv' with your desired output file path
# Define the preprocess_text function here
def preprocess_text(text):
    # Implement the preprocessing logic here
    # ...
    return processed_text
!pip install my_module
import sys
sys.path.append("/path/to/module/directory")

# Import the module with the correct name
import my_correct_module

# Use the function from the imported module
df['Clean2_text'] = df['Text'].apply(my_correct_module.preprocess_text)
import sys
sys.path.append("/path/to/module/directory")
import sys
sys.path.append("/path/to/module/directory")

import my_module

# Use the function from the imported module
df['Clean2_text'] = df['Text'].apply(my_module.preprocess_text)
import sys
sys.path.append("/path/to/module/directory")

# Import the module with the correct name
import my_correct_module

# Use the function from the imported module
df['Clean2_text'] = df['Text'].apply(my_correct_module.preprocess_text)

dir(nfx)
df['clean_Text'] = df['Text'].apply(nfx.remove_stopwords)
df['clean_Text'] = df['Text'].apply(nfx.remove_userhandles)
df['clean_Text'] = df['Text'].apply(nfx.remove_punctuations)
```

```
df[['Text','clean_Text']]
df.shape
```

Keyword Extraction

```
from collections import Counter
ef extract_keywords(text,num=50):
    tokens = [ tok for tok in text.split()]
    most_common_tokens = Counter(tokens).most_common(num)
    return dict(most_common_tokens)
emotion_list = df['Emotion'].unique().tolist()
emotion_list
joy_list = df[df['Emotion'] == 'joy']['clean_Text'].tolist()
# create a document for keyword extraction
joy_docx= ' '.join(joy_list)
joy_docx
Keyword_joy= extract_keywords(joy_docx)
Keyword_joy

sadness_list = df[df['Emotion'] == 'sadness']['clean_Text'].tolist()
sadness_docx= ' '.join(sadness_list)
sadness_docx
Keyword_sadness= extract_keywords(sadness_docx)
Keyword_sadness

shame_list = df[df['Emotion'] == 'shame']['clean_Text'].tolist()
shame_docx= ' '.join(shame_list)
shame_docx
Keyword_shame= extract_keywords(shame_docx)
Keyword_shame

fear_list = df[df['Emotion'] == 'fear']['clean_Text'].tolist()
fear_docx= ' '.join(fear_list)
fear_docx
Keyword_fear= extract_keywords(fear_docx)
Keyword_fear

disgust_list = df[df['Emotion'] == 'disgust']['clean_Text'].tolist()
disgust_docx= ' '.join(disgust_list)
disgust_docx
Keyword_disgust= extract_keywords(disgust_docx)
Keyword_disgust

neutral_list = df[df['Emotion'] == 'neutral']['clean_Text'].tolist()
neutral_docx= ' '.join(neutral_list)
neutral_docx
Keyword_neutral= extract_keywords(neutral_docx)
Keyword_neutral
```

```
anger_list = df[df['Emotion'] == 'anger']['clean_Text'].tolist()
anger_docx= ' '.join(anger_list)
anger_docx
Keyword_anger= extract_keywords(anger_docx)
Keyword_anger

surprise_list = df[df['Emotion'] == 'surprise']['clean_Text'].tolist()
surprise_docx= ' '.join(surprise_list)
surprise_docx
Keyword_surprise= extract_keywords(surprise_docx)
Keyword_surprise


# plotting
def plot_most_common_words(mydict):
    df_02= pd.DataFrame(mydict.items(),columns=['token','count'])
    plt.title('plotting of joy keyword')
    plt.figure(figsize=(20,10))
    sns.barplot(x='token',y='count',data=df_02)
    #plt.xtricks(rotation=45)
    plt.show()

plot_most_common_words(Keyword_joy)
plot_most_common_words(Keyword_fear)
plot_most_common_words(Keyword_sadness)
plot_most_common_words(Keyword_neutral)
plot_most_common_words(Keyword_surprise)
plot_most_common_words(Keyword_anger)
plot_most_common_words(Keyword_shame)

Keyword Extraction for Anger
anger_list = df[df['Emotion'] == 'anger']['clean_Text'].tolist()
# create a document for keyword extraction
anger_docx= ' '.join(anger_list)
anger_docx
Keyword_anger= extract_keywords(anger_docx)
Keyword_anger

def plot_most_common_words(mydict):
    df_02= pd.DataFrame(mydict.items(),columns=['token','count'])
    plt.title('plotting of anger keyword')
    plt.figure(figsize=(20,10))
    sns.barplot(x='token',y='count',data=df_02)
    #plt.xtricks(rotation=45)
    plt.show()

plot_most_common_words(Keyword_anger)

joy_list = df[df['Emotion'] == 'joy']['clean_Text'].tolist()
```

```
# create a document for keyword extraction
anger_docx= ' '.join(joy_list)
anger_docx
Keyword_anger= extract_keywords(anger_docx)
Keyword_anger

sadness_list = df[df['Emotion'] == 'sadness']['clean_Text'].tolist()
# create a document for keyword extraction
sadness_docx= ' '.join(sadness_list)
sadness_docx
Keyword_sadness= extract_keywords(sadness_docx)
Keyword_sadness
```

Sentiment Detection Model

```
# Sentiment detection from Text
# model applied

def get_sentiment(text):
    blob = TextBlob(text)
    sentiment = blob.sentiment.polarity
    if sentiment > 0:
        result = "Positive"
    elif sentiment < 0:
        result = "negative"
    else:
        result = "Neutral"
    return result
get_sentiment(" I love coding")
get_sentiment(" I love coding")
get_sentiment(" love is a very confused word")
get_sentiment(" ")
get_sentiment(" I do coding"
get_sentiment("I donot like this")
get_sentiment("I must not like this")
get_sentiment("I saw a man on a hill with a telescope")
get_sentiment("There’s a man on a hill,and I’m watching him with my
telescope")
get_sentiment("There’s a man on a hill, who I’m seeing, and he has a
telescope")
get_sentiment("Look at the dog with one eye")
get_sentiment("Look at the dog that only has one eye.")
dir(nfx)
df['sentiment'] = df['Text'].apply(get_sentiment)
df = pd.read_csv('Emotion Dataset.csv')
df
df['sentiment'] = df['Text'].apply(get_sentiment)
```

Plot after Sentiment Detection

```
# @title Emotion vs sentiment

from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['sentiment'].value_counts()
    for x_label, grp in df.groupby('Emotion')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('Emotion')
_ = plt.ylabel('sentiment')
# @title Emotion

from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Emotion').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
df.groupby(['Emotion']).size().plot(kind='bar')
df.groupby(['Emotion']).size().plot(kind='pie')
sns.catplot(x='Emotion',data=df,kind='count',aspect= 1.5)
```

```
# Calculate the frequency distribution of 'sentiment'
frequency_counts = df['sentiment'].value_counts()

frequency_percentage = (frequency_counts / len(df['sentiment'])) * 100
frequency_df = pd.DataFrame({'Counts': frequency_counts, 'Percentage':
frequency_percentage})

print(frequency_df)

# Total value count
cardinality = df['sentiment'].nunique()
print(f"\ntotal values: {cardinality}")


# Mapping sentiment num / encode
df["label_num"] = df.sentiment.map({
    'negative': 0,
    'positive': 1,
    'neutral': 2
})

df = df.drop(columns=['sentiment'])
df.head(5)
```

Emotion Frequency

```
# @title Emotion Frequency over Time

df.groupby('Emotion')['Emotion'].count().plot(kind='line', x='Unnamed:
0')
df['Emotion'].value_counts().plot(kind='bar')
# @title Emotion Frequency Over Time

df.groupby('Unnamed: 0')['Emotion'].value_counts().unstack().plot()
cols = df.columns
```

Machine Learning Model

```
# Load ML Pkgs
# Estimators
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB

# Transformers
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix
#from sklearn.metrics import plot_confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import pandas as p

Xfeatures = df['Text']
ylabels = df['Emotion']
Xfeatures
ylabels
cv = CountVectorizer()
X = cv.fit_transform(Xfeatures
X_train,X_test,y_train,y_test =
train_test_split(X,ylabels,test_size=0.3,random_state=42)
#Logistic regressiobn
X = df['Clean_Text']
y = df['Emotion'] #labels

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state = 1)
```

```
# m1 = LogisticRegression() m1.fit(X_train, y_train) pred1 = m1.predict(X_test) print(classification_report(y_test, pred1))

Xfeatures = df['Text']

# Extract features and labels
X = df['Clean_Text']
y = df['Emotion']

# Split the data into training and testing sets

print(f"X_train data type: {type(X_train)}")
print(f"y_train data type: {type(y_train)}")

# Import necessary libraries
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Sample dataset (replace with your own data)
texts = ["I am happy", "I am sad", "I feel great", "I am angry", "I am neutral"]
labels = ["happy", "sad", "happy", "angry", "neutral"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.2, random_state=42)

# Convert text data into numerical features using CountVectorizer
vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

# Initialize and train the logistic regression model
model = LogisticRegression()
model.fit(X_train_counts, y_train)

# Predict on the test set
predictions = model.predict(X_test_counts)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Example of predicting emotions for new text
new_text = ["I am feeling happy today"]
new_text_counts = vectorizer.transform(new_text)
```

```
predicted_emotion = model.predict(new_text_counts)
print("Predicted emotion:", predicted_emotion)


nan_indices = [i for i, text in enumerate(X_train) if pd.isna(text)]
print(f"Number of NaN values in X_train: {len(nan_indices)}")
X_train = pd.DataFrame(X_train)
y_train = pd.DataFrame(y_train)
!pip install pandas
import pandas as pd
print(type(nan_indices))
if type(nan_indices) == list:
    print("nan_indices is a list.")
else:
    print("nan_indices is not a list.")
print(nan_indices[:5])
# Get the actual column names of X_train
actual_columns = X_train.columns

# Update nan_columns with the actual column names
nan_columns = [column for column in nan_columns if column in
actual_columns]

# Drop rows with NaN values in the specified columns
X_train_dropped = X_train.dropna(subset=nan_columns)
y_train_dropped = y_train.dropna(subset=nan_columns)
print(X_train_dropped.shape)
print(y_train_dropped.shape)
print(X_train_dropped.head())
print(y_train_dropped.head())
model.fit(X_train_counts, y_train)
predictions = model.predict(X_test_counts)
accuracy = accuracy_score(y_test, predictions)
print("Accuracy after handling missing values:", accuracy)
```

Multinomial NB for TF-IDF

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Example dataset
data = pd.read_csv("Emotion Dataset.csv")  # Load your dataset
X = data['Text']  # Text data
y = data['Emotion']  # Emotion labels
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Convert text data to numerical features using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=1000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Train a classifier (example: Naive Bayes)
classifier = MultinomialNB()
classifier.fit(X_train_tfidf, y_train)

# Predict the labels for the test set
y_pred = classifier.predict(X_test_tfidf)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap="Blues",
xticklabels=classifier.classes_, yticklabels=classifier.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Print classification report
print(classification_report(y_test, y_pred))
```

KNN for TF-IDF

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Example dataset
data = pd.read_csv("Emotion Dataset.csv")  # Load your dataset
X = data['Text']  # Text data
y = data['Emotion']  # Emotion labels

# Split the data into training and testing sets
```

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

# Convert text data to numerical features using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=1000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Train a KNN classifier
k = 5  # Number of neighbors
knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(X_train_tfidf, y_train)

# Predict the labels for the test set
y_pred = knn_classifier.predict(X_test_tfidf)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap="Blues", 
xticklabels=knn_classifier.classes_, 
yticklabels=knn_classifier.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Print classification report
print(classification_report(y_test, y_pred))

y_train = y_train.values.reshape(-1, 1)
```

```
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import LSTM, Dense, Embedding, SpatialDropout1D
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Example dataset
data = pd.read_csv("Emotion Dataset.csv")  # Load your dataset
X = data['Text']  # Text data
y = data['Emotion']  # Emotion labels
```

```
# Tokenization
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X)
X_seq = tokenizer.texts_to_sequences(X)

# Padding sequences
max_length = max([len(seq) for seq in X_seq])
X_pad = pad_sequences(X_seq, maxlen=max_length, padding='post')

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_pad, y,
test_size=0.2, random_state=42)

# LSTM Model
embedding_dim = 100
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index)+1,
output_dim=embedding_dim, input_length=max_length))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Train the model
batch_size = 64
epochs = 10
history = model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, validation_data=(X_test, y_test), verbose=2)

# Predictions
y_pred = model.predict_classes(X_test)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap="Blues")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Print classification report
print(classification_report(y_test, y_pred))
```

Random Forest for TF-IDF

```
om sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load dataset
df = pd.read_csv('Emotion Dataset.csv')

# Prepare data
texts = df['Text'].tolist()
labels = df['Emotion'].tolist()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(texts, labels,
test_size=0.2, random_state=42)

# Convert text data into numerical features using TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)  # Increase max_features
for better coverage
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Initialize and train a Random Forest classifier
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}
rf_model = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(rf_model, param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train_tfidf, y_train)

# Get the best model from the grid search
best_rf_model = grid_search.best_estimator_

# Predict on the test set
predictions = best_rf_model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
mport pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load dataset
df = pd.read_csv('Emotion Dataset.csv')
```

```
# Prepare data
texts = df['Text'].tolist()
labels = df['Emotion'].tolist()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(texts, labels,
test_size=0.2, random_state=42)

# Convert text data into numerical features using TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Initialize and train a Random Forest classifier
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

best_accuracy = 0
best_rf_model = None

for n_estimators in param_grid['n_estimators']:
    for max_depth in param_grid['max_depth']:
        for min_samples_split in param_grid['min_samples_split']:
            rf_model = RandomForestClassifier(n_estimators=n_estimators,
max_depth=max_depth, min_samples_split=min_samples_split,
random_state=42)
            rf_model.fit(X_train_tfidf, y_train)
            predictions = rf_model.predict(X_test_tfidf)
            accuracy = accuracy_score(y_test, predictions)
            print(f"Parameters: n_estimators={n_estimators},
max_depth={max_depth}, min_samples_split={min_samples_split}, Accuracy:
{accuracy}")
            if accuracy > best_accuracy:
                best_accuracy = accuracy
                best_rf_model = rf_model
```

BNB for TF-IDF

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score

# Load dataset
df = pd.read_csv('Emotion Dataset.csv')
```

```
# Prepare data
texts = df['Text'].tolist()
labels = df['Emotion'].tolist()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(texts, labels,
test_size=0.25, random_state=42)

# Convert text data into numerical features using TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000, binary=True)  # Using
binary=True for Bernoulli Naive Bayes
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Initialize and train the Bernoulli Naive Bayes classifier
bnb_model = BernoulliNB()
bnb_model.fit(X_train_tfidf, y_train)

# Predict on the test set
predictions = bnb_model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Example of predicting emotions for new text
new_text = ["I am feeling happy today"]
new_text_tfidf = vectorizer.transform(new_text)
predicted_emotion = bnb_model.predict(new_text_tfidf)
print("Predicted emotion:", predicted_emotion)
```

SVC for TF-IDF

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load dataset
df = pd.read_csv('Emotion Dataset.csv')

# Prepare data
texts = df['Text'].tolist()
labels = df['Emotion'].tolist()

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(texts, labels,
test_size=0.2, random_state=42)

# Convert text data into numerical features using TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Initialize and train the SVM classifier
svm_model = SVC(kernel='linear', C=1.0, random_state=42)
svm_model.fit(X_train_tfidf, y_train)

# Predict on the test set
predictions = svm_model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Example of predicting emotions for new text
new_text = ["I am feeling happy today"]
new_text_tfidf = vectorizer.transform(new_text)
predicted_emotion = svm_model.predict(new_text_tfidf)
print("Predicted emotion:", predicted_emotio
```

Feature Extraction Techniques

```
rom sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from sklearn.model_selection import cross_val_predict, StratifiedKFold
from sklearn.neural_network import MLPClassifier
#import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report
import numpy as np
# Define X_train as a numpy array
X_train = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(type(X_train))
#<class 'numpy.ndarray'>
#<class 'numpy.ndarray'>
#X_train = X_train.toarray()
import numpy as np

# Define X_train as a numpy array
```

```
X_train = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(type(X_train))
#<class 'numpy.ndarray'>

# Convert the numpy array to a list of lists
X_train = X_train.tolist()
print(type(X_train))
#<class 'list'>
```

```
model = MultinomialNB()

# Train
model.fit(X_train, y_train)

# Test
y_pred = model.predict(X_test)

# Evaluate
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

y_pred_cv = cross_val_predict(model, X_train, y_train, cv=cv)
classification_rep = classification_report(y_train, y_pred_cv)
print("Classification Report (Cross-Validation):\n", classification_rep)

accuracy = accuracy_score(y_train, y_pred_cv)
print("Accuracy (Cross-Validation):", accuracy)

model.fit(X_train, y_train)
y_pred_test = model.predict(X_test)

classification_rep_test = classification_report(y_test, y_pred_test)
print("Classification Report (Test Data):\n", classification_rep_test)
accuracy_test = accuracy_score(y_test, y_pred_test)
print("Accuracy (Test Data):", accuracy_test)

#import seaborn as sns
#import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
```

BNB for Count Vectorizer

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30)
bnb = BernoulliNB(binarize=0.0)
bnb.fit(X_train, y_train)
bnb.score(X_test, y_test)
#print(classification_report(Y_test, X_train))
print(classification_report(y_test, bnb.predict(X_test)))

import pandas as pd

# Convert X_train and X_test to Pandas DataFrames
X_train_df = pd.DataFrame(X_train)
X_test_df = pd.DataFrame(X_test)

# Print the number of missing values in each column of X_train and
X_test
print(X_train_df.isnull().sum().sum())
print(X_test_df.isnull().sum().sum())

# Print the number of missing values in y_train and y_test
print(y_train.isnull().sum())
print(y_test.isnull().sum())
print(X_train.duplicated().sum())
print(X_test.duplicated().sum())
print(y_train.duplicated().sum())
print(y_test.duplicated().sum())
```

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
print(X.shape)
print(y.shape)
# Ensure X and Y have the same number of samples
if X.shape[0] != Y.shape[0]:
    # Adjust the size of X or Y to match the other array
    if X.shape[0] > Y.shape[0]:
        X = X[:Y.shape[0]]
    else:
        Y = Y[:X.shape[0]]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.30)
gnb = GaussianNB()
gnb.fit(X_train, Y_train)
gnb.score(X_test, Y_test)
```

Logistic Regression for Count Vectorizer

```
lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.30)
lr_model.score(X_test,y_test)
```

```
from sklearn.metrics import classification_report
# Check if y_test and lr_model are valid inputs
print(f"y_test type: {type(y_test)}")
print(f"lr_model type: {type(lr_model)}")

from sklearn.metrics import classification_report
y_pred = lr_model.predict(X_test
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.30)
#lr_model.score(X_test,y_test)

print(classification_report(y_test, y_pred))
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.30)
lr_model.score(X_test,y_test)

import joblib
!pip install eli5
import eli5
eli5.show_weights(lr_model,top=240000)
```

Gradient Boost

```
grid = {
    'learning_rate': [0.3, 0.1, 0.5],
    'n_estimators': [100, 300],
    'max_depth': [1, 3, 9]
}

m3 = GridSearchCV(GradientBoostingClassifier(), grid, verbose = 2)
m3.fit(X_train, y_train)
print(m3.best_params_)
pred3 = m3.predict(X_test)
print(classification_report(y_test, pred3))
gbc = GradientBoostingClassifier(n_estimators=300,
                                 learning_rate=0.05,
                                 random_state=100,
                                 max_features=5 )
# Fit to training set
gbc.fit(X_train, y_train)

# Predict on test set
pred_y = gbc.predict(X_test)

# accuracy
acc = accuracy_score(y_test, y_pred)
print("Gradient Boosting Classifier accuracy is : {:.2f}".format(acc))
```

KNN

```
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# compute accuracy of the model
knn.score(X_test, y_test
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier(n_estimators= 24000, random_state= 10)
rfc.fit(X_train, y_train)
predictions = rfc.predict(X_test)
# Model evaluation

print (classification_report(y_test, predictions))
print (confusion_matrix(y_test, predictions))
```

Gaussian Naïve Bayes Algorithm

```
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split

#X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.30)
gnb = GaussianNB()
gnb.fit(X_train, y_train)
gnb.score(X_test, y_test)
```

Plot of Comparative Analysis

```
algorithms=("LR","KNN","NB","Random Forest")
scores = (lr_model.score,knn.score,nv_model.score,m2.score)
y_pos = np.arange(1,7)
colors = ("red","gray","purple","green","orange","blue")
plt.figure(figsize=(18,10))
plt.bar(y_pos,scores,color=colors)
plt.xticks(y_pos,algorithms,fontsize=18)
plt.yticks(np.arange(0.00, 1.01, step=0.05))
plt.grid()
plt.suptitle("Bar Chart Comparison of Models",fontsize=15)
plt.show()
```