A PROJECT REPORT

ON

Social Media Sentiment Analysis Using Twitter Dataset

Submitted by

SHUBHANGI BHARDWAJ (1/20/FET/BCS/191)

JANHAVI GUPTA (1/20/FET/BCS/190)

MOHIT AHUJA (1/20/FET/BCS/126)

TALHA KHAN (1/20/FET/BCS/193)

Under the Guidance of

Ms Neha Batra

Assistant Professor

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

School of Engineering & Technology

Manav Rachna International Institute of Research and Studies, Faridabad

November, 2023

DECLARATION

We hereby declare that this project report entitled "**Social Media Sentiment Analysis Using Twitter Dataset**" *by* **SHUBHANGI BHARDWAJ (1/20/FET/BCS/191), JANHAVI GUPTA (1/20/FET/BCS/190), MOHIT AHUJA (1/20/FET/BCS/126), TALHA KHAN (1/20/FET/BCS/193),** being submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in **Computer Science & Engineering** under School of Engineering & Technology of Manav Rachna International Institute of Research and Studies, Faridabad, during the academic year 2020-24, is a bonafide record of our original work carried out under the guidance of **Ms. Neha Batra, Assistant Professor, Faculty of Engineering(FET).**

We further declare that we have not submitted the matter presented in this Project for the award of any other Degree/Diploma of this University or any other University/Institute.

1. Shubhangi Bhardwaj**(**1/20/FET/BCS/191)
2. Janhavi Gupta**(**1/20/FET/BCS/190)
3. Mohit Ahuja**(**1/20/FET/BCS/126)
4. Talha khan**(**1/20/FET/BCS/193)

Manav Rachna International Institute of Research and Studies, Faridabad

School of Engineering & Technology

Department of Computer Science and Engineering

November,2024

Certificate

This is to certify that this project report entitled "**Social Media Sentiment Analysis Using Twitter Dataset**" *by* **SHUBHANGI BHARDWAJ (1/20/FET/BCS/191), JANHAVI GUPTA (1/20/FET/BCS/190), MOHIT AHUJA (1/20/FET/BCS/126), TALHA (1/20/FET/BCS/193),** submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in **Computer Science and Engineering** under Faculty of Engineering & Technology of Manav Rachna International Institute of Research and Studies, Faridabad, during the academic year 2020-24, is a bonafide record of work carried out under my guidance and supervision. I hereby declare that the work has been carried out under my supervision and has not been submitted elsewhere for any other purpose.

(Signature of Project Guide)
Ms Neha Batra
Assistant Professor
Department of Computer Science and Engineering
SET, MRIIRS, Faridabad

(Signature of HoD)
Dr. Mamta Dahiya
Head of Department
Department of Computer Engineering
SET, MRIIRS, Faridabad

ACKNOWLEDGEMENT

The successful realization of project is an outgrowth of a consolidated effort of people from desperate fronts. We are thankful to Ms **Neha Batra** (Assistant Professor) for his/her variable advice and support extended to us without which we could not be able to complete our project for a success.

We are thankful to **Dr. Supriya Panda,** Project Coordinator, Professor, CSE department for her guidance and support.

We express our deep gratitude to **Dr. Mamta Dahiya,** Head of Department (CSE) for his endless support and affection towards us. His constant encouragement has helped to widen the horizon of our knowledge and inculcate the spirit of dedication to the purpose.

We would like to express our sincere gratitude to **Dr. Geeta Nijhawan**, Associate Dean SET, MRIIRS for providing us the facilities in the Institute for completion of our work.

Words cannot express our gratitude for all those people who helped us directly or indirectly in our Endeavour. We take this opportunity to express our sincere thanks to all staff members of CSE department for the valuable suggestion and also to our family and friends for their support.

Mohit Ahuja
(1/20/FET/BCS/126)

Janhavi Gupta
(1/20/FET/BCS/190)

Shubhangi Bhardwaj
(1/20/FET/BCS/191)

Talha Khan
(1/20/FET/BCS/193)

Table Of Contents

ABSTRACT

Technology today has become a momentous driving vehicle for communication world-wide Social media platforms like twitter, Facebook, Instagram are the most important arenas for expressing views on transformations happening in and around the world every day. Twitter is a rich origin of info for mining of user opinions. This paper reflects the idea of taking user opinions into consideration performing sentiment, emotion analysis and establishing conclusions on interested topics using Machine Learning algorithms. Naive Bayes and Support Vectors Machines in Machine Learning are tuned-up using supervised learning to obtain outputs for sentiment emotion analysis respectively. Sentiment analysis desires to obtain sentiment polarity (positive or negative) and Emotion analysis intent to obtain emotion (eg.., empty , sadness , anger etc..,) from user data. Such analysis essentially serves a gateway for consumer needs and generates growth opportunities in businesses.

Keywords: Sentiment Analysis, Emotion analysis, Naive Bayes, Support Vector Machines, Logistic Regression, Pre-processing, Tokenization, Stemming, Lemmatization, Twitter.

LIST OF FIGURES

Figure **Page No.**

LIST OF TABLES

1. INTRODUCTION

Social media sentiment analysis has turn out to be a distinguished area of study and experimentation in current years. Twitter a micro-blogging site, has lion's share in social media info. Most research has been confined to classify tweets into positive, negative categories ignoring sarcasm. Human emotions are extremely diverse and cannot be restricted to certain metrics alone. Polarity analysis gives limited information on the actual intent of message delivered by author and just positive or negative classes are not sufficient to understand nuances of underlying tone of a sentence. This brings the need to take one step above sentiment analysis leading to emotion analysis. In this paper we throw light on methods we have used to derive sentiment analysis considering sarcasm and how we have accomplished emotion analysis of user opinions.

A supervised learning technique provides labels to classifier to make it understand the insights among various features. Once the classifier gets familiarized with train data it can perform classification on unseen test data. We have chosen Naive Bayes and Support Vector Machine classification algorithms to carry out sentiment and emotional analysis respectively.

Performing SA (sentiment analysis) and EA (emotion analysis) will help organizations or companies to improve services, track products and obtain customer feedback in a normalized form. Gaining insights from large volumes of data is a mountain of a task for humans hence using an automated process will easily drill down into different customer feedback segments mentioned on social media or elsewhere. Effective business strategies can be built from results of sentiment and emotion analysis. Identifying clear emotions will establish a transparent meaning of text which potentially develops customer relationships, motivation and extends consumer expectations towards a brand or service.

Emotion detection involves a wide platter of emotions classified into states like joy, fear, anger, surprise and many more. We here examine sentiments and emotions of short texts coined as tweets from the famous social media, twitter.

Generally, people discuss a lot of things daily but it is difficult to get insights just by reading through each of their opinions so there should be a way that helps us to get insights of user's opinions in an unbiased manner, so this model helps in drawing out Sentiment, Emotions

of users, classify them and finally present them to us. Sentiment analysis is the prediction of emotions in a word, sentence or corpus of documents.

It is intended to serve as an application to understand the attitudes, opinions and emotions expressed within an online mention. The intention is to gain an overview of the wider public opinion behind certain topics.

What is Sentiment Analysis?

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. However, analysis of social media streams is usually restricted to just basic sentiment analysis and count based metrics. This is akin to just scratching the surface and missing out on those high value insights that are waiting to be discovered.

What is Emotional Analysis?

It is the process of identifying human emotions, most typically from facial expressions as well as from verbal expressions. It relies on a deeper analysis of human emotions and sensitivities.

Emotions analytics (EA) software collects data on how a person communicates verbally and nonverbally to understand the person's mood or attitude. The technology, also referred to as emotional analytics, provides insights into how a customer perceives a product, the presentation of a product or their interactions with a customer service representative.

Just as with other data related to customer experience, emotions data is used to create strategies that will improve the business's customer relationship management (CRM). EA software programs can be used with companies' data collection, data classification, data analytics and data visualization initiatives.

1.1. MOTIVATION FOR WORK

Businesses primarily run over customers satisfaction, customer reviews about their products. Shifts in sentiment on social media have been shown to correlate with shifts in stock markets. Identifying customer grievances thereby resolving them leads to customer satisfaction as well as trustworthiness of an organization. Hence there is a necessity of an unbiased automated system to classify customer reviews regarding any problem.

In today's environment where we're justifiably suffering from data overload (although this does not mean better or deeper insights), companies might have mountains of customer feedback collected; but for mere humans, it's still impossible to analyse it manually without any sort of error or bias.

Oftentimes, companies with the best intentions find themselves in an insights vacuum. You know you need insights to inform your decision making and you know that you're lacking them, but don't know how best to get them.

Sentiment analysis provides some answers into what the most important issues are, from the perspective of customers, at least. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition that isn't always right.

1.2. PROBLEM STATEMENT

Generating statistical information regarding emotions, sentiments out of analysis of user's opinions from tweets, which can be used as an inference to understand how users feel thereby improving users' experiences regarding. Despite the availability of software to extract data regarding a person's sentiment on a specific product or service, organizations and other data workers still face issues regarding the data extraction. With the rapid growth of the World Wide Web, people are using social media such as Twitter which generates big volumes of opinion texts in the form of tweets which is available for the sentiment analysis. This translates to a huge volume of information from a human viewpoint which make it difficult to extract a sentence, read them, analyse tweet by tweet, summarize them and organize them into an understandable format in a timely manner.

2. LITERATURE SURVEY

2.1. INTRODUCTION

"What other people think" has always been an important piece of information for most of us during the decision-making process. The Internet and the Web have now (among other things) made it possible to find out about the opinions and experiences of those in the vast pool of people that are neither our personal acquaintances nor well-known professional critics — that is, people we have never heard of. And conversely, more and more people are making their opinions available to strangers via the Internet. The interest that individual users show inonline opinions about products and services, and the potential influence such opinions wield, is something that is driving force for this area of interest. And there are many challenges involved in this process which needs to be walked all over in order to attain proper outcomes out of them.

In this survey we analysed basic methodology that usually happens in this process and measures that are to be taken to overcome the challenges being faced.

2.2. EXISTING METHODS

2.2.1. Using a Heterogeneous Dataset for Emotion Analysis in Text:

A supervised machine learning approach was adopted to recognize six basic emotions (anger, disgust, fear, happiness, sadness and surprise) using a heterogeneous emotion-annotated dataset which combines news headlines, fairy tales and blogs. For this purpose, different features sets, such as bags of words, and N-grams, were used. The Support Vector Machines classifier (SVM) performed significantly better than other classifiers, and it generalized well on unseen examples. Five data sets were considered to compare among various approaches. In bag of words Each sentence in the dataset was represented by a feature vector composed of Boolean attributes for each word that occurs in the sentence. If a word occurs in a given sentence, its corresponding attribute is set to 1; otherwise it is set to 0. In N grams approach they are defined as sequences of words of length n. N-grams can be used for catching syntactic patterns in text and may include important text features such as negations, e.g., "not happy". Negation is an important feature for the analysis of emotion in text because

it can totally change the expressed emotion of a sentence. The author concludes some research studies in sentiment analysis claimed that N-grams features improve performance beyond the BOW approach

2.2.2. Multiclass Emotional Analysis on Social Media Posts:

The author conveys that among the models they have built SVM has outperformed with greatest accuracy. After considering around 13,000 examples per emotion they had split 63%for training set a holdout cross validation set (27%), and a final test set (10%). The first model trained and optimized for the task was Multinomial Naive Bayes. The model gave very good results, even as a baseline, whereas a random classifier would have performed with 6.6% accuracy on 15 classes, this Naive Bayes model was performing at 28.01% accuracy. After Multinomial Naive Bayes, they trained and optimized SoftMax Regression models. While optimizing the SoftMax Regression model, several checks on pre- processing techniques were conducted. One of the conclusions drawn was that stemming was actually decreasing the accuracy of the models. The next model trained was a linear SVM. This model was trained with both tf-idf vectors and count vectors. To reduce the number of features to prevent over-fitting, PCA was given a shot on document vectors.

Later, they had tried training a kernel SVM with RBF, however the size of the data made it impossible for a regular computer to train the model in reasonable time. Thus, they trained a ν-SVM instead with the kernel trick. Results proved that SVM (linear kernel) was maintaining greatest accuracy.

This paper presents an SVM algorithm for supervised clustering. This algorithm learns an item-pair similarity measure to optimise performance of correlation clustering on a variety of performance measures. This method holds the clustering algorithm constant and modifies the similarity measure so that the clustering algorithm produces desirable clustering. The clustering algorithm is trained to produce desirable clusters, given sets of items and complete clustering over these sets, how to cluster future sets of items depending on information provided by the user (through manual adjustment of similarity measure). A clustering algorithm may not produce desirable clustering without additional information from the user. Similarity measure maps pairs of items to a real number indicating how similar the pair is; positive valuesindicate the pair is alike, negative values, unalike. Each pair of different items has a feature vector to describe the pair. Correlation clustering is used. The author concludes that SVM

cluster's ability to optimize to a custom loss function and exploit transitive dependencies in data does improve performance compared to a naive classification approach.

2.2.3. Classification of Emotions from text using SVM based Opinion Mining:

SVM classification using Quadratic programming was used. Steps included preparing the data set, annotating the dataset with predefined emotions, using NLP prepare the database matrix of test emotions and training emotions, classify the training set with a support vector machine using quadratic programming algorithm. Compute the prediction of support vector machine using kernel function and its parameter for classification and finally compute the accuracy of the classification. The basic idea of SVM is to find the optimal hyperplane to separate two classes with largest margin of pre-classified data. After this hyperplane is determined it is used for classifying data into two classes based on which side they are located. By applying appropriate transformations to the data space after computing the separating hyperplane,

SVM can be extended to cases where the margin between two classes is non- linear. Finally, on classifying the data set, superior results have been obtained.

2.2.4. Emotion Detection from Text:

The author proposes a new framework which is divided into two components: Emotion ontology and emotion detector. The emotion detector algorithm calculates weight for a particular emotion by adding weights at each level of hierarchy and also calculates same for its counter emotion, then compares the both scores and greater one is taken as the detected emotion.

The first step is calculation of parameters. Different parameters include Parent- child relationship if a text document belongs to a child it also indirectly refers to the parent of it. Hence if a certain value is added to the child's score, parent score also needs to be modified. Another parameter depth in ontology gives an idea about how specific is the term in relation to its corresponding ontology structure.

The more specific it is the more weight age should be given to it. The parameter frequency in text document stresses on giving importance to a frequently occurring term in the

document and computes value by parsing the text document and searches for occurrences of the terms. The proposed algorithm calculates the score for each emotion word with the help of parameters from previous steps. This score will be directly proportional to the frequency of the term and inversely proportional to its depth in the ontology. For every primary level emotion class, a respective score will be calculated. Finally, Emotion class having highest score will win the race and declared as Emotion state of the corresponding text document.

2.2.5. Emotion Detection and Analysis on Social Media:

In these two approaches were followed NLP and Machine Learning. In machine learning approach the author reveals that for generation of the training set a set of seed words were chosen, comprising of commonly used Emotion words and emoticons from the EWS, evenly distributed over all the Emotion- Categories. Then Tweepy was queried using these seed words to develop a huge database of around 13,000 tweets. The seed words are used to ensure that we get tweets that express at least one of the six emotions. Later filtering and labelling of the tweets were performed to eliminate tweets that convey mixed emotions. Only those tweets which have a percentage of more than 70% for a particular emotion are labelled and fed to the classifier for training.

For training the classifier an open source library weka was used. Before classification the pre-processing steps like stop-word filtering, lower casing all words and Stemming each word using Weka's Snowball Stemmer were applied. The results of NLP approach and ML approach are combined. The scores which are generated by the first approach are modified, according to the Labelled- Category of the classifier. The Emotion-Category with the maximum final score is decided as the final Emotion-Category of the tweet (or the piece of text).

2.2.6. A Novel Approach of Sentiment Classification using Emoticons

Author here signifies the importance of Emoticons in the process of identifying sentiment of a hive sentence. For example, consider the following statement School starts in 6 days, school starts in 6 days :(have quite different meaning. Hence Emoticons can be very useful in identifying sentiment of a sentence.so in this approach first he chooses some random tweets for many number of times and chooses the emoticons that appears for most number of

times and then assigns EmScore algo. Which gives a sentiment score for each emoticon in the list of emoticons. Next calculates sentiment classification using SentE algo. Then after calculates overall sentiment value for a given tweet.

2.2.7. Analysing Sentiment of Twitter Data using Machine Learning Algorithm:

Author here classifies the process of Sentiment analysis as follows: Tweets posted on twitter are freely available through a set of APIs of twitter. At first, we collected a corpus of positive, negative, neutral and irrelevant tweets from twitter API. Then pre-processing done by removing stop words, negations, URL, full stop, commas etc. to reduce noise from tweets and to prepare our data for sentiment classification. After that, we apply machine learning algorithms to our dataset and compare their results. Results helps to identify which machine learning algorithm is best suited for classification of SA. The stages involved in this process are:

1. Data Collection: obtain training data of twitter
2. Pre-processing Setup: removing unrelated contents
3. Sentiment Classifier: various machine learning algorithms are used
4. Evaluation: produces result.

And each step is further classified like in the pre-processing step some sub- steps like stemming, stop word extractor are also included. The efficiency of a classifier usually depends on the pre-processing step.

2.2.8. The Impact of Features Extraction on the Sentiment Analysis:

Author in this paper analysed the impact and importance of extracting the features and how they play a crucial role in the performance of the classifier and its outcome. Here six pre-processing techniques are used and then features are extracted from the pre-processed data. There are many feature extraction techniques such as Bag of words, NLP, TF-IDF etc.., Here he analysed the impact of two features TF-IDF word level and N-Gram on a Twitter data set and found that performance of classifier is 3-4% in high when TF-IDF feature is chosen than

N-Gram and analysis is done using many classification algorithms like Naïve Bayes, Support Vector Machines etc.., Eventually at the end this paper author emphasizes on the importance of feature selection process which affects the Sentiment Evaluation result.

2.2.9. Methods for Sentiment Analysis:

In this paper, various approaches to sentiment analysis have been examined and analysed, Techniques such as Streaming API SVM etc., discussed. These techniques all have different strengths and weaknesses.

1. Sentiment Analysis on Twitter using streaming API:

It uses NLP where it helps in tokenization, stemming, classification, tagging, parsing and sentiment reasoning Its basic feature is to convert unstructured data into structured data. It uses Naive Bayes for classification which requires number of linear parameters.

To find out the sentiment an automated system must be developed "Support Vector Machine" can be used for this method. SVM is machine that takes the input and store them in a vector then using SentiWordNet it scores it decides the sentiment. It also classifies the opinion in overall way by positive, negative or Neutral.

There are many more techniques but these are the most familiar ones, performs more efficiently. And selection of both features and techniques affect the final outcome. So proper analysis must be done to get intended, as accurate results as possible.

3. METHODOLOGY

The sentiment analysis of Twitter data is an emerging field that needs much more attention. We use Tweepy an API to stream live tweets from Twitter. User based on his interest chooses a keyword and tweets containing that keyword are collected and stored into a csv file. Then we make it a labelled dataset using textblob and setting the sentiment fields accordingly. Thus, our train data set without pre-processing is ready. Next, we perform pre-processing to clean, remove unwanted text, characters out of the tweets. Then we train our classifier by fitting the train data to the classifier, there after prediction of results over unseen test data set is made which there after provides us with the accuracy with which the classifier had predicted the outcomes. There after we present our results in a pictorial manner which is the best way to showcase results because of its easiness to understand information out of it.

3.1. PROPOSED SYSTEM

3.1.1 Extraction of Data

Tweets based on a keyword of user's choice of interest have been collected using a famous twitter API known as Tweepy and stored into a csv file. This data set collected for sentiment analysis have tweets based on a keyword e.g., cybertruck. Tweets mimicking various emotions as a dataset downloaded from Kaggle is used for emotional analysis. Since both the machines are trained using supervised learning and work on different parameters different data sets have been considered.

In order to extract the opinion first of all data is selected and extracted from twitter in the form of tweets. After selecting the data set of the tweets, these tweets were cleaned from emoticons, unnecessary punctuation marks and a database was created to store this data in a specific transformed structure. In this structure, all the transformed tweets are in lowercase alphabets and are divided into different parts of tweets in the specific field.

Fig 3.1.1 Extraction of data

3.1.2 Pre-processing of Data:

Following are the Pre-processing steps that have been carried out:

3.1.3 Removing Html tags and URLs:

Html tags and URLs often have minimum sentiments thus they are removed from tweets. Using regular expressions.

3.1.4 Conversion to lowercase:

To maintain uniformity all the tweets are converted to lowercase. This will benefit to avert inconsistency in data. Python provides a function called lower () to convert sentences to lower case.

3.1.5 Tokenization:

Tokenization is the process of converting text into tokens before transforming it into vectors. It is also easier to filter out unnecessary tokens. For example, a document into paragraphs or sentences into words. In this case we are tokenizing the reviews into words.

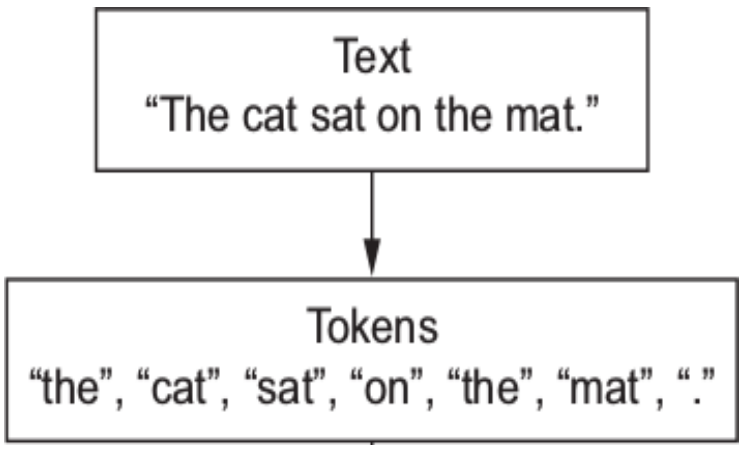

Fig 3.1.5.1 Tokenization

Purposes of Tokenization

- Normalizing Text: Tokenization helps to normalize text by removing whitespaces, converting text to lowercase, and handling punctuation and special characters consistently.

- Identifying Language Units: Tokenization identifies meaningful units of language, such as words, phrases, or idioms, which can be used for further analysis.
- Feature Extraction: Tokens can be used as features for machine learning algorithms, allowing them to identify patterns and relationships in the text.
- Linguistic Analysis: Tokenization is essential for various linguistic analysis tasks, such as part-of-speech tagging, dependency parsing, and sentiment analysis.

Types of Tokenization

- Word Tokenization: This is the most common type of tokenization, where the text is split into individual words.
- Sentence Segmentation: This type of tokenization breaks the text into individual sentences, which may be useful for tasks like sentiment analysis or summarization.
- N-gram Tokenization: This type of tokenization creates n-grams, which are sequences of n consecutive words. N-grams are often used in language modeling and machine translation.
- Character Tokenization: This type of tokenization breaks the text into individual characters, which may be useful for languages with complex word structures or for tasks like machine transliteration.

Implementation of Tokenization

Tokenization can be implemented using various programming languages and NLP toolkits. Popular libraries for tokenization include NLTK (Natural Language Toolkit) in Python and spaCy in Python.

Benefits of Tokenization

- Improves Accuracy: Tokenization can improve the accuracy of NLP tasks by providing meaningful units for analysis.
- Reduces Complexity: Tokenization simplifies the text representation, making it easier for NLP algorithms to process.
- Enhances Efficiency: Tokenization can improve the efficiency of NLP algorithms by reducing the amount of data they need to process.

Overall, tokenization is a crucial step in many NLP applications, enabling the analysis and processing of text data for a wide range of tasks.

3.1.6 Removing punctuations and special symbols:

Apart from the considered set of emoticons punctuations and symbols like &,\,; are removed.

3.1.7 Stop words removal:

Stop words are the most commonly occurring words which are not relevant in the context of the data and do not contribute any deeper meaning to the phrase. In this case contain no sentiment. NLTK provide a library used for this.

"This is a sample sentence, showing off the stop words filtration."

['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']

3.1.8 After stop words removal:

['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']

3.1. 9 Stemming and Lemmatization:

Sentences are always narrated in tenses, singular and plural forms making most words accompany with -ing,-ed,es and ies. Therefore, extracting the root word will suffice to identify sentiment behind the text.

Base forms are the skeleton for grammar stemming and lemmatization reduces inflectional forms and derivational forms to common base forms.

Example: Cats is reduced to cat, ponies is reduced to poni.

Stemming is a crude way of reducing terms to their root, by just defining rules of chopping off some characters at the end of the word, and hopefully, gets good results most of the time. *The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.* With that being said, stemming/lemmatizing helps us reduce the number of overall terms to certain “root” terms.

Stemming and lemmatization are two text normalization techniques used in natural language processing (NLP) to reduce words to their base or root forms. This is done to improve the accuracy of information retrieval, text mining, and machine translation. Stemming is a more aggressive approach

to text normalization. It simply removes the affixes (prefixes and suffixes) from a word to get its stem, which is the part of the word that remains after removing the affixes. For example, the stem of the word "running" is "run."

Stemming is a fast and efficient process, but it can sometimes lead to incorrect results. This is because stemming does not consider the context of the word, and it can sometimes remove too much of the word, resulting in an invalid or meaningless stem. For example, stemming the word "ran" would result in "r," which is not a valid word.

Lemmatization is a more sophisticated approach to text normalization. It uses a dictionary or morphological analyzer to identify the lemma of a word, which is the dictionary form of the word. This means that lemmatization takes into account the context of the word, and it can produce more accurate results than stemming. For example, lemmatizing the word "ran" would result in "run," which is the correct lemma.

Lemmatization is a more computationally expensive process than stemming, but it is more accurate. It is therefore the preferred method for text normalization in many NLP applications.

Stemming:
playing -> play
dancing -> dance
coding -> code

Lemmatization:
running -> run
dancing -> dance
coding -> code

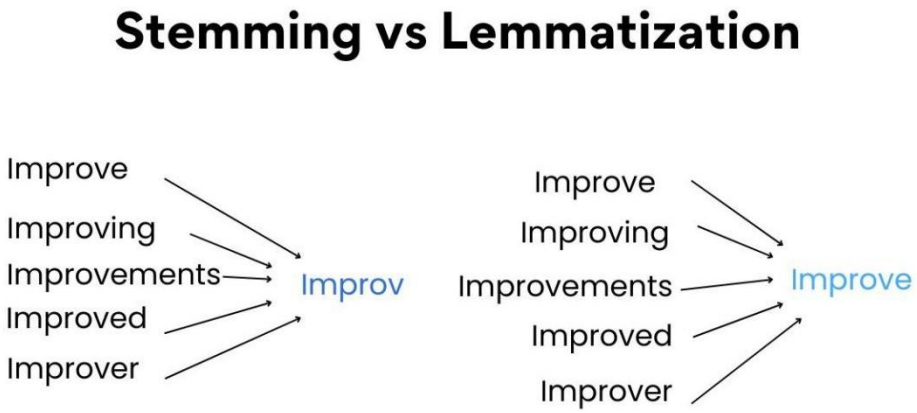

Fig 3.1.9.1 Stemming vs Lemmatization

Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

Fig 3.1.9.2 Stemming, Lemmatization Rule and Example

3.1.10 Feature Extraction:

Text data demands a special measure before you train the model. Words after tokenization are encoded as integers or floating-point values for feeding input to machine learning algorithm. This practice is described as vectorization or feature extraction. Scikit-learn library offers TF-IDF vectorizer to convert text to word frequency vectors.

3.1.11 Fitting Data to Classifier and predicting test data:

Train data is fitted to a suitable classifier upon feature extraction, then once the classifier is trained enough then we predict the results of the test data using the classifier, then compare the original value to the value returned by the classifier.

3.1.12 Result Analysis:

Here the accuracy of different classifiers is shown among which the best classifier with highest accuracy percent is the chosen. Some factors such as f- score, mean, variance etc., also accounts for consideration of the classifiers.

3.1.13 Visual Representation:

Our final results are plotted as pie charts which contains different fields such as positive, negative, neutral in case of sentiment analysis. Whereas happy, sad, joy etc., in case of emotional analysis. Pictorial representation is the best way to convey information without much efforts. Thus, it is chosen.

3.1.14 System architecture

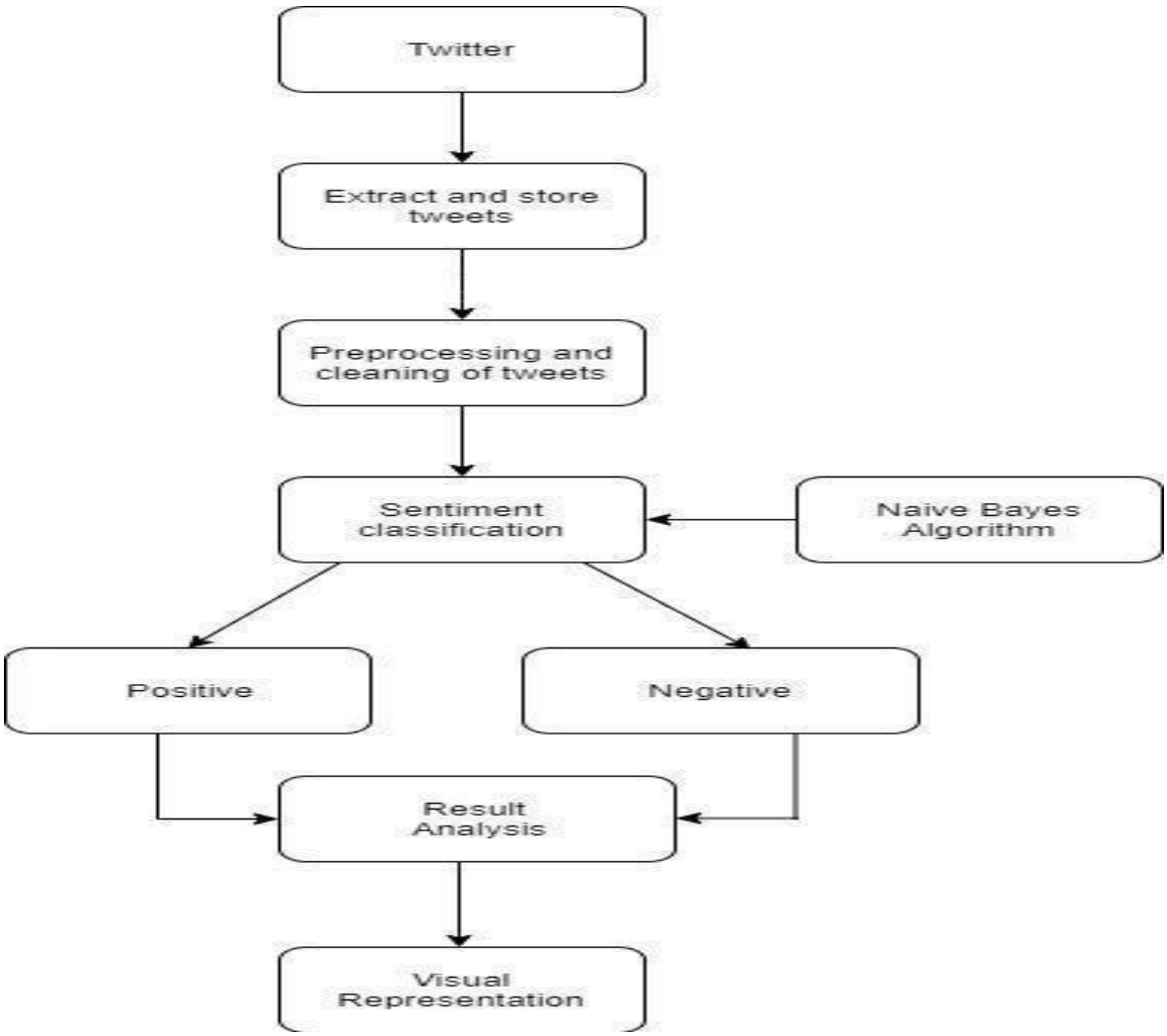

Fig 3.1.14.1 Architecture diagram for sentiment analysis using Naive Bayes

Naive Bayes Algorithm:

Naive Bayes algorithm which is based on well-known Bayes theorem which is mathematically represented as

$$P(A/B) = [P(B/A)\ P(A)] / P(B)$$

Where,

A and B are events

P(A/B) is the likeliness of happening of event A given that event B is true and has happened, Which is known to be as posterior probability.

P(A) is the likeliness of happening of an event A being true, Which is known to be as prior probability.

P(B/A) is the likeliness of happening of an event B given A was true ,Which is known to be as Likelihood.

P(B) is the likeliness of happening of an event B, Which is known to be as Evidence . Bayes theorem can now be applied on data sets in following way

$P(y/X) = [P(X/y) P(y)] / P(X)$

This is a classification method that relies on Bayes' Theorem with strong (naive) independence assumptions between the features. A Naive Bayes classifier expects that the closeness of a specific feature (element) in a class is disconnected to the closeness of some other elements. For instance, an organic fruit might be considered to be an apple if its colour is red, its shape is round and it measures approximately three inches in breadth. Regardless of whether these features are dependent upon one another or upon the presence of other features, a Naïve Bayes classifier would consider these properties independent due to the likelihood that this natural fruit is an apple. Alongside effortlessness, the Naive Bayes is known to out- perform even exceedingly modern order strategies. The Bayes hypothesis is a methodof computing for distinguishing likelihood P(a|b) from P(a), P(b) and P(b|a) as follows:

$p(a|b) = [p(b|a) * p(a)] / p(b)$

Where p(a|b) is the posterior probability of class a given predictor b and p(b|a) is the likelihood that is the probability of predictor b given class a.

The prior probability of class a is denoted as p(a), and the prior probability of predictor p is denoted as p(b).

Bayes' theorem is a mathematical formula that describes the probability of an event occurring given prior knowledge of conditions that might be related to the event. In the context of Naive Bayes, it is used to calculate the probability of a particular class (e.g., spam or not spam) given a set of features (e.g., words in an email).

The formula for Bayes' theorem is:

$P(A|B) = (P(B|A) * P(A)) / P(B)$

Where:

P(A|B) is the probability of event A occurring given event B
P(B|A) is the probability of event B occurring given event A
P(A) is the prior probability of event A
P(B) is the prior probability of event B

The naive assumption in Naive Bayes is that all features are independent of each other. This means that the presence or absence of one feature does not affect the presence or absence of any other feature. This assumption is often violated in real-world data, but it can still lead to accurate predictions in many cases.

The Naive Bayes algorithm uses Bayes' theorem to calculate the probability of each class given a set of features. The class with the highest probability is then predicted as the label for the data point.

For each class C: a. Calculate the prior probability P(C) b. For each feature F: i. Calculate the conditional probability P(F|C)
For each data point: a. For each class C: i. Calculate the posterior probability P(C|F1, F2, ..., Fn) using Bayes' theorem b. Predict the class with the highest posterior probability.

Advantages of Naive Bayes

- Simple and easy to implement
- Fast training and prediction times
- Effective for text classification
- Disadvantages of Naive Bayes

Assumes independence of features, which may not always be true Sensitive to outliers.

Applications of Naive Bayes

- Text classification (e.g., spam filtering, sentiment analysis)
- Medical diagnosis
- Customer segmentation
- Fraud detection

The Naive Bayes is widely used in the task of classifying texts into multiple classes and was recently utilized for sentiment analysis classification.

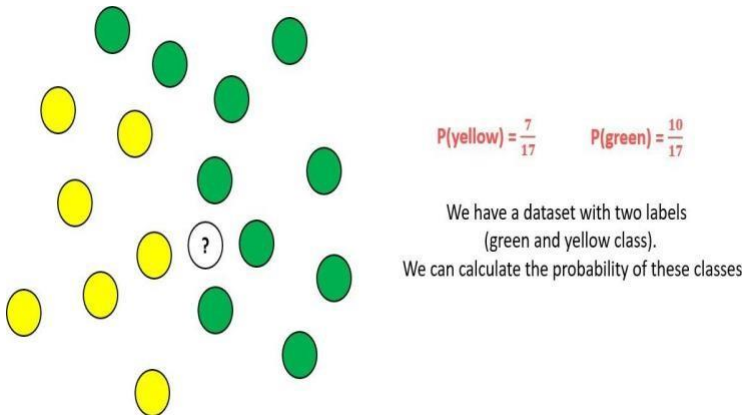

Fig3.1.14.2 Naive Bayes Classifier

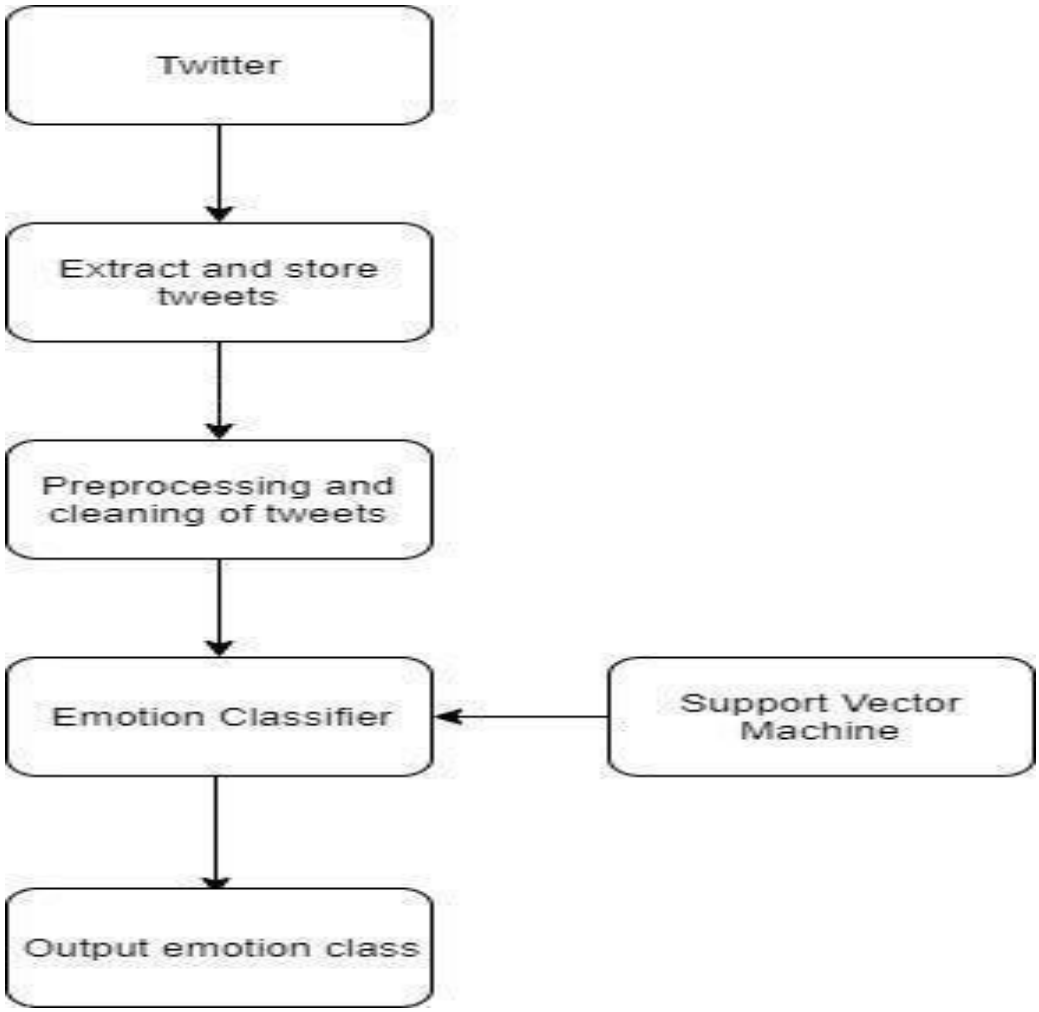

Fig 3.1.14.3 Architecture diagram for Emotion analysis using SVM

Support Vector Machine:

Support Vector Machines is a supervised machine learning algorithm, adopted conventionally for classification as well as regression problems. SVMs for classification, work by figuring out the right hyperplane among the classes. After being trained by a labelled data set, SVM outputs an optimal hyperplane that categorizes new examples. Classification by SVMs for different data sets is governed by tuning parameters namely kernel, regularization, gamma and margin. When data is 2-dimensional Support vector classifier is a line, if it is 3D SVC forms a plane instead of a line. When data is more than 4D then classifier is a hyperplane. For highly distributed data Maximal margin and support vector classifier fail and hence SVMs are used. For linearly separable patterns optimal hyperplane is formed and for non-linearly separable patterns transformation of original data into a new space is performed determined by kernel function. The trouble of discovering an optimal hyperplane is an optimization problem and can be worked out using optimization techniques (e.g. Lagrange). To classify tweets into different emotion classes a linear kernel has been utilized. Linear kernel is preferable for text classification problems because text has lot of features, linear kernel is faster and less parameters are optimized. When SVM is trained with a linear kernel only C regularization parameter need to be optimized whereas for other kernels you need to optimize gamma parameter also.

Support Vector Machines (SVM) is a machine learning model proposed by. N. Vapnik. The basic idea of SVM is to find an optimal hyperplane to separate two classes with the largest margin from pre-classified data.

After this hyperplane is determined, it is used for classifying data into two classes based on which side they are located. By applying appropriate transformations to the data space before computing the separating hyperplane, SVM can be extended to cases where the margin between two classes is non-linear.

Linearly Separable Case

If the training data are linearly separable, then there exists a pair (w,b) such that $W^T X_i + b \geq 1$, for all $X_i \in P$ (1)$W^T X_i + b \leq -1$, for all $X_i \in N$

The decision function is of the form $f(x) = \text{sign}(wtx + b)$ (2)

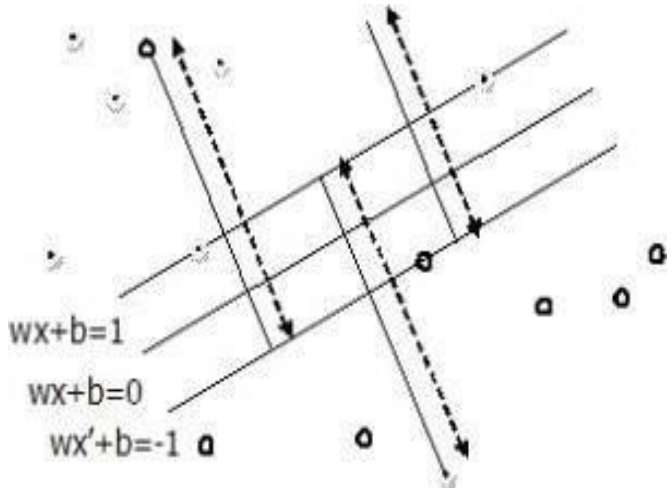

Fig 3.1.14.4 SVM Classifier(I)

Optimal separating hyperplane for Binary classification problem.

W is termed the weight vector and b the bias (or b− is termed the threshold). The inequality constraints (1) can be combined to give

$Yi(WTXi + b) \geq 1$, for all $Xi \in P \cup N$ (3)

W is termed the weight vector and b the bias (or b− is termed the threshold). The inequality constraints (1) can be combined to give

Given a training set of instance-label pairs (xi, yi),

i =1, 2, 3, ...l where $xi \in Rn$, the class label of the ith pattern is denoted by $yi \in \{1, - 1\}t$. Nonlinearly separable problem are often solved by mapping the input data samples xi to a higher dimensional feature space ø(xi). The classical maximum margin SVM classifier aimsto find a hyperplane of the form wt ø(x) +b =0, which separates patterns of the two classes. Sofar we have restricted ourselves to the case where the two classes are noise-free. In the case of noisy data, forcing zero training error will lead to poor generalization. To take account of the fact that some data points misclassified, we introduce a vector of slack variables $\Xi = (\xi 1, , \xi 1)^T$ that measure the amount of violation of the constraints (3). The problem can then be written as l

Minimize $W^t w + C \Sigma \xi i$ (4)

W,b,ξ i=1 subject to the constraints $yi(wt\phi(xi) + b) \geq 1 - \xi i$ (5)

$\xi i = 0$, i =1, 2, 3, l

The solution to (4)-(5) yields the soft margin classifier, is termed because the distance or margin between the separating hyperplane $wt\phi(x) + b = 0$ is usually determined by considering the dual problem, which is given by $l\ l\ l\ L(w,b,\alpha i, \Xi, \Gamma) = \|w\|^2 + \sum \alpha i[yi(wT\phi(xi) + b) - 1 + \xi i] - \sum \gamma i\ \xi i + C\sum \xi i$ i=1 i=1 i=1 where $\Lambda = (\alpha 1, \ldots, \alpha 2)^T$ and $\Gamma = (\gamma 1, , \gamma 1)^T$, are the Lagrange multipliers corresponding to the positivity of the slack variables. The solution of this problem is the saddle point of the Lagrangian given by minimizing L with respect to w, Ξ and b, and maximizing with respect to $\Lambda \geq 0$ and $\Gamma \geq 0$. Differentiating with respect to w, b and Ξ and setting the results equal to zero

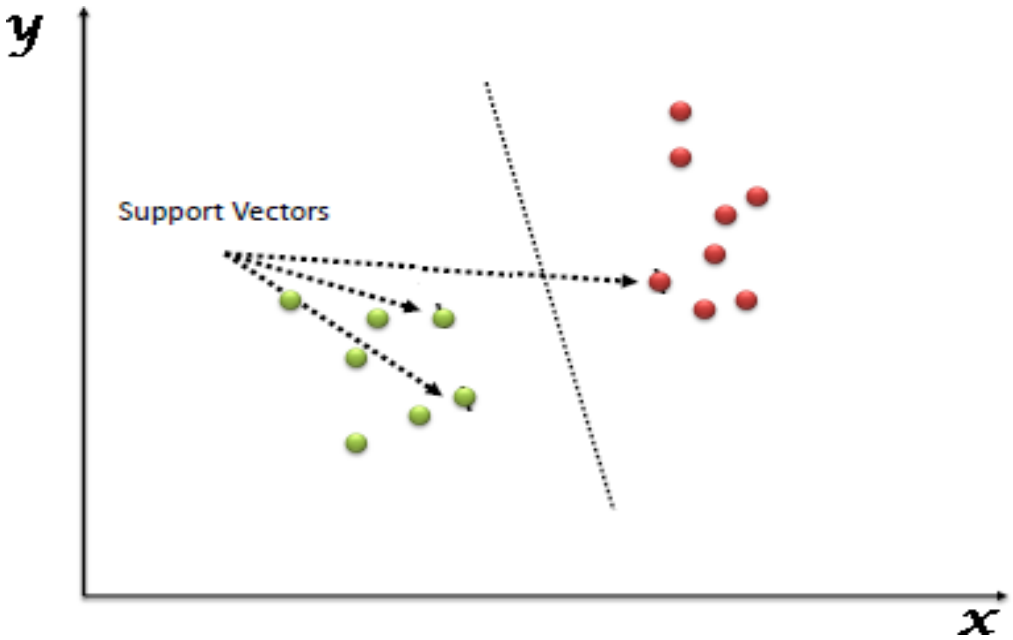

Fig 3.1.14.5 SVM Classifier (II)

Concept of Hyperplane

At the core of SVMs lies the concept of a hyperplane, which is a decision boundary that separates data points belonging to different classes. In two-dimensional space, a hyperplane is a straight line, while in higher-dimensional spaces, it becomes a more complex geometric structure.

Finding the Optimal Hyperplane

The goal of an SVM is to find the optimal hyperplane that maximizes the margin, which is the distance between the hyperplane and the closest data points from each class, also known as support vectors. By maximizing the margin, the SVM ensures that the decision boundary is as clear as possible, minimizing the risk of misclassification.

Kernel Function

In cases where data points cannot be linearly separated, SVMs employ a technique called the kernel function. The kernel function maps the data points into a higher-dimensional space, where a linear hyperplane can effectively separate them. This process is known as the kernel trick.

Advantages of SVMs

- High-Dimensional Data Handling: SVMs can effectively handle high-dimensional data, making them suitable for complex classification problems.
- Robustness to Outliers: SVMs are relatively insensitive to outliers, allowing them to perform well in noisy datasets.
- Non-Parametric Nature: SVMs are non-parametric, meaning they do not make assumptions about the underlying distribution of the data.

Applications of SVMs

- Image Classification: SVMs are used to classify images into different categories, such as animals, objects, or scenes.
- Text Classification: SVMs are used to classify text documents.

For highly distributed data Maximal margin and support vector classifier fail and hence SVMs are used. For linearly separable patterns optimal hyperplane is formed and for non-linearly separable patterns transformation of original data into a new space is performed determined by kernel function.

3.2 Logistic Regression:

Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class. It is used for classification algorithms its name is logistic regression. it's referred to as regression because it takes the output of the linear regression function as input and uses a sigmoid function to estimate the probability for the given class. The difference between linear regression and logistic regression is that linear regression output is the continuous value that can be anything while logistic regression predicts the probability that an instance belongs to a given class or not.

3.2.1 Applications of Logistic Regression:

It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value.

It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.

3.2.2 Logistic Function (Sigmoid Function):

The sigmoid function is a mathematical function used to map the predicted values to probabilities.

It maps any real value into another value within a range of 0 and 1. o The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the “S” form.

The S-form curve is called the Sigmoid function or the logistic function.

In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

3.2.3 Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

3.2.3.1 Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

3.2.3.2 Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as “cat”, “dogs”, or “sheep”

3.2.3.3 Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as “low”, “Medium”, or “High”.

3.2.3 How does Logistic Regression work?

We use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

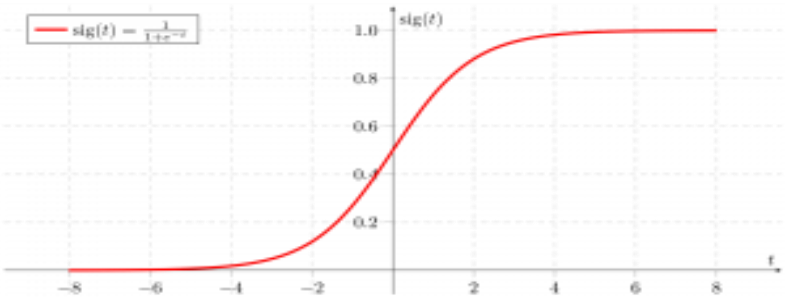

Fig 3.1.14.6 Logistic Regression

As shown above, the figure sigmoid function converts the continuous variable data into the probability i.e. between 0 and 1.

The logistic regression model is represented by the logistic function, also known as the sigmoid function, which transforms the linear combination of input variables into a probability value. The logistic function is given by:

$f(x) = 1 / (1 + e^{(-bx)})$

where:
f(x) is the probability of the positive outcome
x is the linear combination of input variables
b is the vector of model parameters

The goal of logistic regression is to estimate the model parameters (coefficients in the logistic function) that best fit the training data. This involves minimizing the loss function, which measures the difference between the predicted probabilities and the actual labels of the training data. A common loss function for logistic regression is cross-entropy.

The coefficients in the logistic function represent the relative importance of each input variable in predicting the outcome. A positive coefficient indicates that the variable increases the probability of the positive outcome, while a negative coefficient indicates the opposite. The magnitude of the coefficient reflects the strength of the relationship between the variable and the outcome.

Applications

- Medical Diagnosis: Predicting the likelihood of a patient having a particular disease based on symptoms and medical history
- Spam Filtering: Identifying spam emails based on content and sender information
- Customer Churn Prediction: Predicting the likelihood of customers leaving a service or subscription
- Click Prediction: Predicting the likelihood of users clicking on an advertisement or link
- Fraud Detection: Detecting fraudulent transactions or activities based on financial data and behavior.

Advantages of Logistic Regression

- Easy to interpret and understand
- Computationally efficient
- Effective for binary classification tasks
- Robust to outliers

Disadvantages of Logistic Regression

- Assumes independence of input variables, which may not always be true
- Sensitive to the choice of features
- Not suitable for multi-class classification problems

Overall, logistic regression is a powerful and versatile statistical method for binary classification tasks. Its simplicity, interpretability, and effectiveness make it a popular choice for various applications in machine learning and data analysis.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value.

It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.

The difference between linear regression and logistic regression is that linear regression output is the continuous value that can be anything while logistic regression predicts the probability that an instance belongs to a given class or not.

4. DESIGN

4.1. STRUCTURE CHART

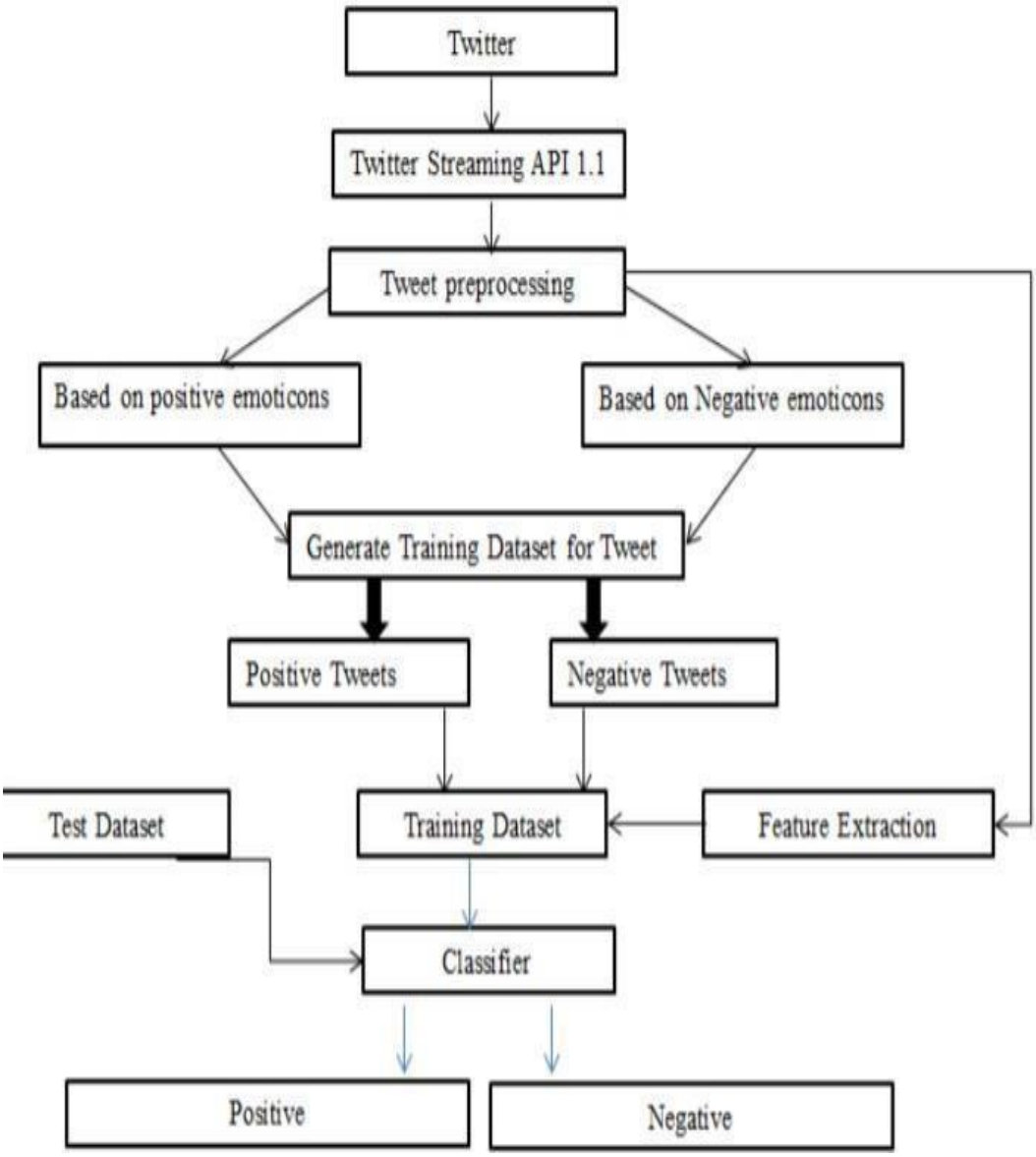

Fig 4.1.1 Structure Chart

4.2. FLOW CHART DIAGRAM

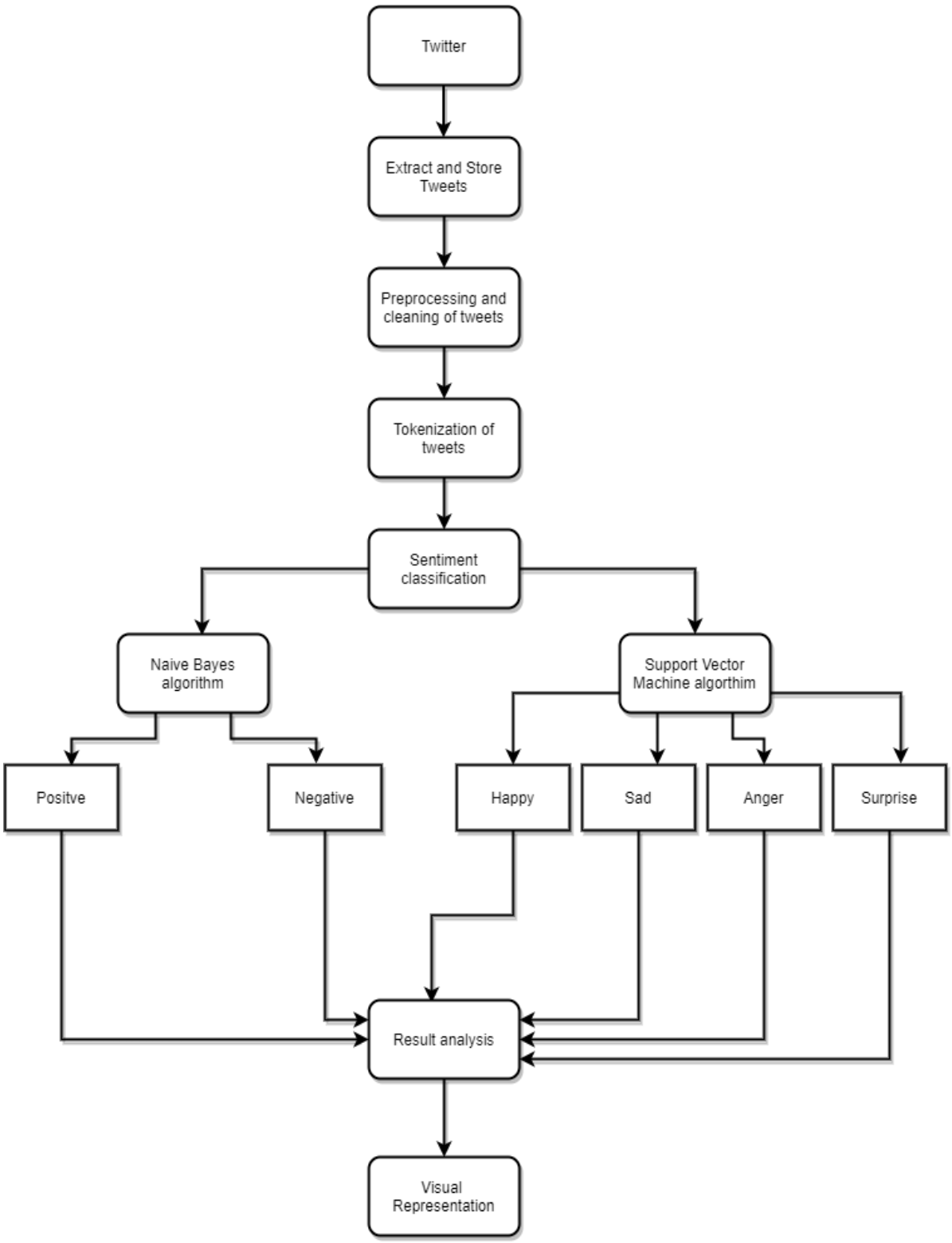

Fig 4.2.1 Flow Chart Diagram

4.3. COMPONENT DIAGRAM

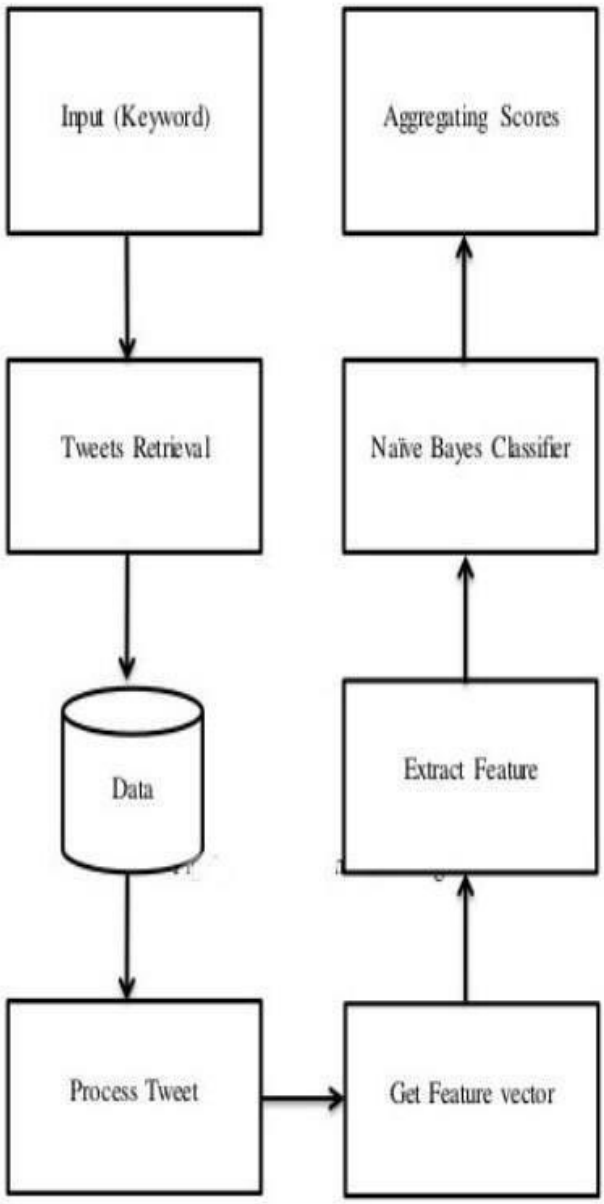

Fig 4.3.1 Component Diagram

4.4. Activity Diagram

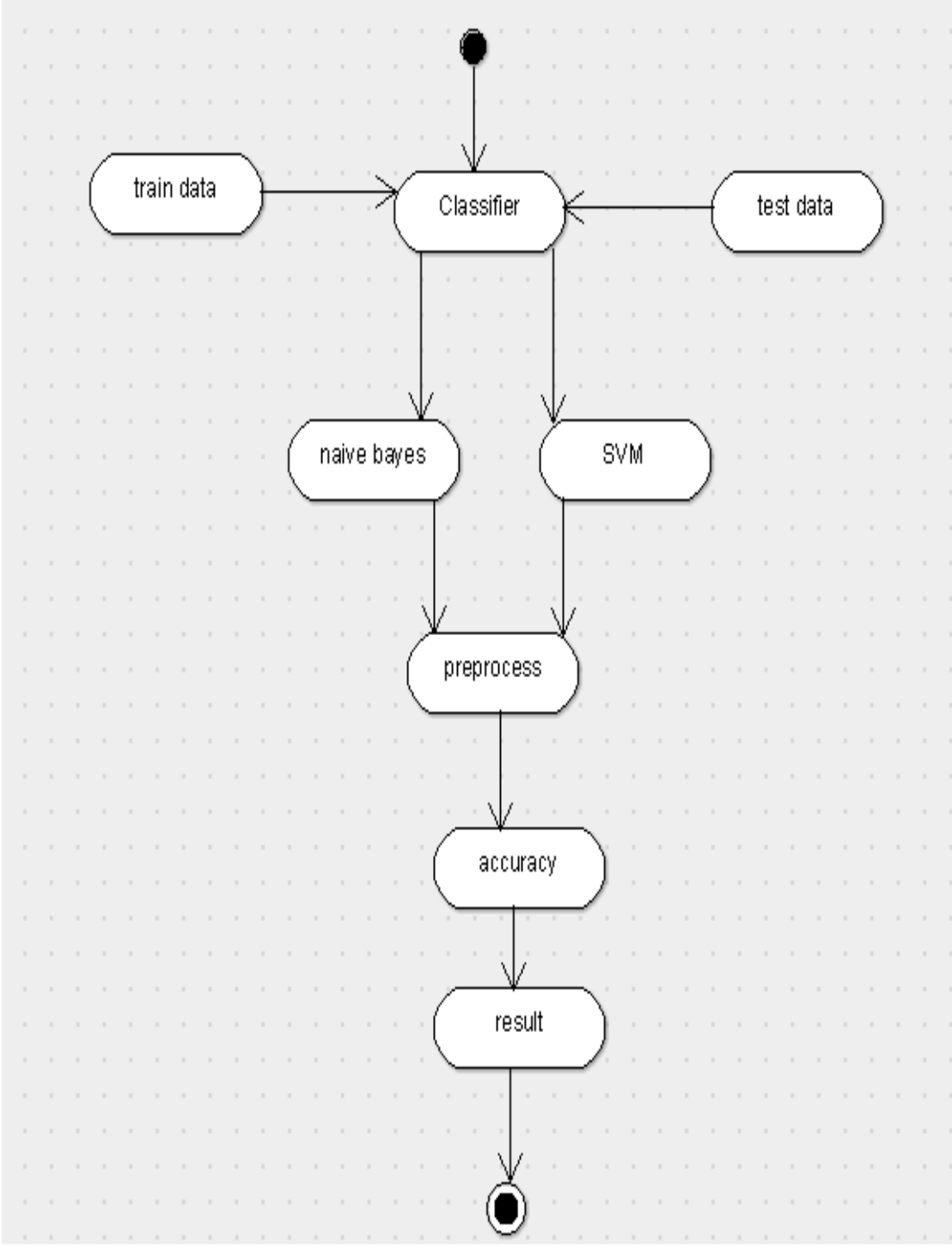

Fig 4.4.1 Activity Diagram

4.5. COLLABORATION DIAGRAM

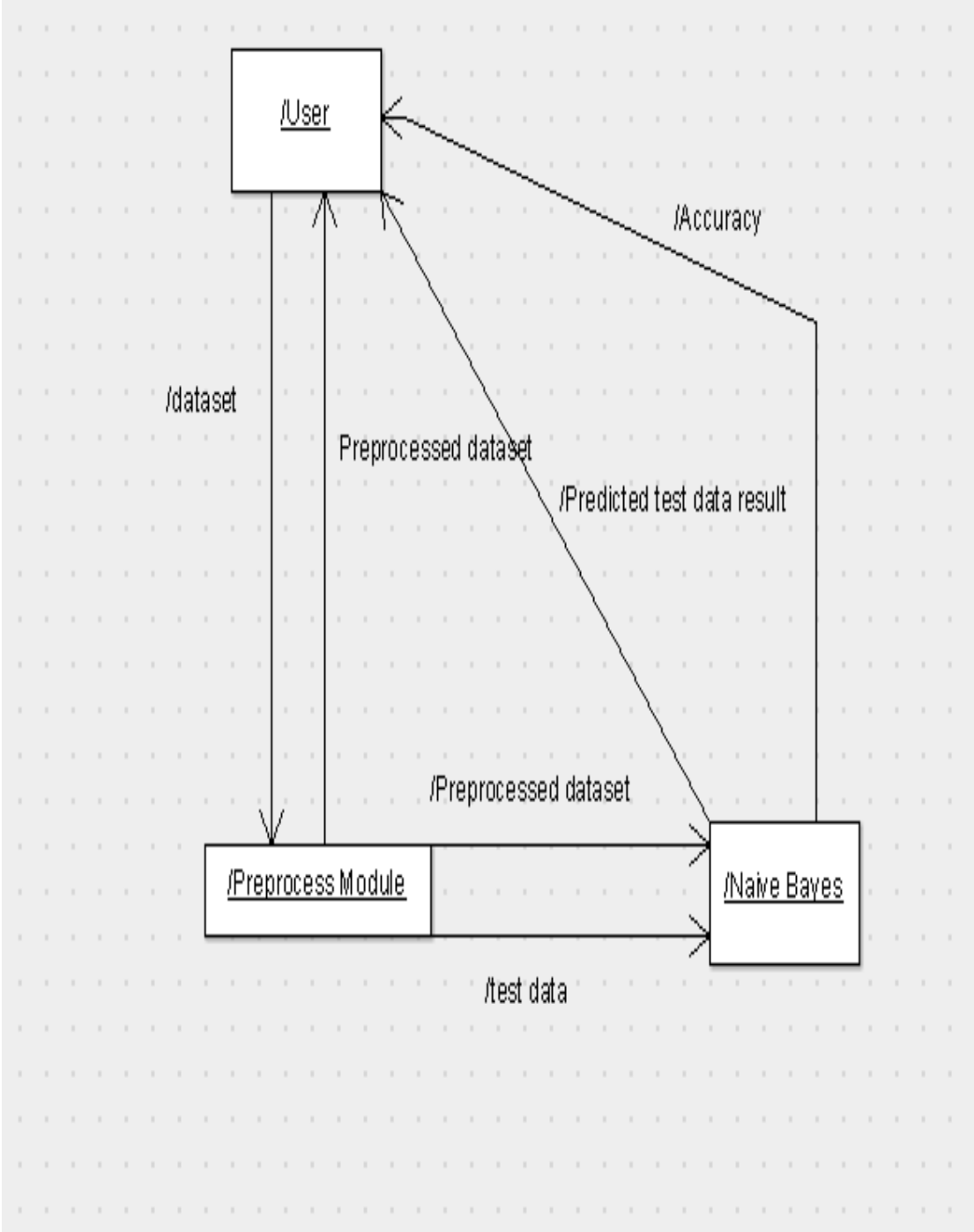

Fig 4.5.1 Collaboration Diagram

5. EXPERIMENTAL ANALYSIS AND RESULTS

5.1. SYSTEM CONFIGURATION

This project can run on commodity hardware. We ran entire project on an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, it also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the project just takes very little amount of time that depends on the size of data set upon which classifier is working upon. Second part Emotional analysis takes some time around 5-10 mins to produce results because of its large volume data set.

5.1.1. Software Requirements

Following are the software and modules that needs to be installed for successful execution of the project. They are:

1. **Anaconda**

Anaconda is an open-source distribution of Python and R programming languages for data science and machine learning. It includes over 250 popular packages for data science, including NumPy, pandas, SciPy, Matplotlib, Seaborn, TensorFlow, and PyTorch. Anaconda also provides a comprehensive environment management system that simplifies the installation, management, and updating of packages.

Key features of Anaconda:

- Distribution of Python and R: Anaconda provides pre-compiled versions of Python and R, along with a wide range of popular packages for data science and machine learning.
- Environment management: Anaconda's environment management system allows users to create and manage isolated virtual environments for different projects. This helps to avoid conflicts between packages and ensures that each project has the correct dependencies.
- Package management: Anaconda's package manager makes it easy to install, update, and remove packages from the Anaconda repository. It also allows users to create custom packages and repositories.

- Version control integration: Anaconda integrates with Git and other version control systems, making it easy to track changes to projects and share them with others.

Benefits of using Anaconda:

- Reduced setup time: Anaconda simplifies the setup process for data science and machine learning projects by pre-installing common packages and providing environment management tools.
- Improved reproducibility: Anaconda's environment management system ensures that projects have the correct dependencies and can be reproduced consistently on different systems.
- Simplified package management: Anaconda's package manager makes it easy to install, update, and remove packages, reducing the risk of conflicts and dependencies issues.

Who should use Anaconda:

Anaconda is a valuable tool for anyone who works with data science or machine learning. It is particularly well-suited for:

- Data scientists: Anaconda provides a comprehensive set of tools for data analysis, data cleaning, exploratory data analysis, and machine learning.
- Machine learning engineers: Anaconda is a popular choice for machine learning engineers who need to quickly set up and deploy machine learning models.
- Data analysts and researchers: Anaconda is a helpful tool for data analysts and researchers who need to work with large datasets and perform complex data analysis tasks.

Overall, Anaconda is a powerful and versatile tool that can be used for a wide range of data science and machine learning tasks. Its ease of use, extensive package library, and robust environment management system make it a valuable choice for anyone who works with data.

2. Spyder

Spyder is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. It is written in Python and designed by scientists and is exclusively for scientists, data analysts, and engineers.

Key features of Spyder:

- Integrated development environment (IDE): Spyder provides a comprehensive set of tools for developing, debugging, and testing Python code, including a code editor, a debugger, a console, and a variable explorer.
- Scientific computing: Spyder is designed for scientific computing and includes support for NumPy, SciPy, Matplotlib, and other popular scientific computing libraries.
- Interactive exploration: Spyder provides interactive tools for exploring and visualizing data, such as an IPython console, a plotting widget, and a data viewer.
- Extendability: Spyder is extensible with first-party and third-party plugins, allowing users to customize the IDE to their specific needs.

Benefits of using Spyder:

- Ease of use: Spyder is easy to use and has a user-friendly interface that is familiar to scientists and engineers.
- Scientific computing support: Spyder provides comprehensive support for scientific computing libraries, making it an ideal choice for scientific programming.
- Interactive exploration: Spyder's interactive tools make it easy to explore and visualize data, facilitating data analysis and insight generation.
- Extendability: Spyder's extensibility allows users to customize the IDE to their specific needs and workflows.

Who should use Spyder:

Spyder is a valuable tool for anyone who works with scientific computing in Python, including:

- Scientists: Spyder is well-suited for scientists who need to develop, test, and analyze data using Python.
- Data analysts: Data analysts can utilize Spyder for data cleaning, exploration, and visualization tasks.
- Engineers: Engineers can employ Spyder to develop and test numerical algorithms and models.
- Students and hobbyists: Students and hobbyists can learn Spyder to gain practical experience in scientific computing with Python.

Overall, Spyder is a powerful and versatile IDE that is specifically designed for scientific programming in Python. Its ease of use, scientific computing support, interactive tools, and extensibility make it a valuable choice for scientists, data analysts, engineers, and anyone who works with scientific data in Python.

3. Jupyter Note Book

Jupyter Notebook is a web-based interactive computing environment that combines code, narrative text, and equations. It is widely used for data science, machine learning, and scientific computing. Jupyter Notebooks are written in Python, but they can also be used with other programming languages, such as R and Julia.

Key Features of Jupiter Notebook:

- Interactive coding: Jupyter Notebook allows users to write and execute code in an interactive environment, making it easy to experiment with code and see the results immediately.
- Narrative text: Jupyter Notebook allows users to interweave code with narrative text, equations, and images, creating a rich and informative document.

- Open-source: Jupyter Notebook is an open-source project, making it freely available and customizable.
- Extensive community: Jupyter Notebook has a large and active community, providing support and resources for users.

Benefits of Using Jupyter Notebook:

- Improved learning: Jupyter Notebook's interactive nature facilitates learning and experimentation, making it a valuable tool for students and beginners.
- Enhanced collaboration: Jupyter Notebooks can be easily shared and collaborated on, enabling teams to work together on projects.
- Reproducible research: Jupyter Notebooks promote reproducible research by providing a clear and organized record of code and analysis.
- Versatility: Jupyter Notebook can be used for a wide range of tasks, including data cleaning, analysis, visualization, and machine learning.

Who Should Use Jupyter Notebook:

Jupyter Notebook is a valuable tool for anyone who works with data science, machine learning, or scientific computing, including:

- Data scientists: Jupyter Notebook is a popular choice for data scientists who need to explore, clean, and analyze data.
- Machine learning engineers: Jupyter Notebook is often used by machine learning engineers to develop, train, and evaluate machine learning models.
- Researchers: Researchers can use Jupyter Notebook to perform analysis, record results, and share their findings with others.
- Students and hobbyists: Jupyter Notebook is a great option for students and hobbyists who want to learn about data science and machine learning.

Examples of Jupyter Notebook Use:

- Exploring datasets: Jupyter Notebook can be used to explore and analyze datasets by loading data, cleaning it, and performing statistical analysis.
- Building machine learning models: Jupyter Notebook can be used to build machine learning models by training, evaluating, and optimizing models.
- Visualizing data: Jupyter Notebook can be used to create interactive visualizations of data using libraries like Matplotlib and Seaborn.
- Communicating results: Jupyter Notebooks can be easily shared and published, allowing researchers and others to communicate their findings.

Overall, Jupyter Notebook is a powerful and versatile tool that has become an essential part of the data science and machine learning toolkit. Its interactive nature, rich documentation capabilities, and extensive community make it a valuable tool for anyone who works with data.

4. Nltk

NLTK (Natural Language Toolkit) is a free and open-source Python library for natural language processing (NLP). It provides a wide range of features for text processing, including tokenization, stemming, lemmatization, part-of-speech tagging, named entity recognition, and sentiment analysis.

Key features of NLTK:

- Tokenization: NLTK provides a variety of tokenization algorithms for splitting text into words, sentences, or other units.
- Stemming and lemmatization: NLTK provides tools for reducing words to their root forms, either by stemming or lemmatization.
- Part-of-speech tagging: NLTK provides tools for assigning part-of-speech tags to words in a text.
- Named entity recognition: NLTK provides tools for identifying named entities in a

text, such as people, places, and organizations.

- Sentiment analysis: NLTK provides tools for determining the sentiment of a text, such as whether it is positive, negative, or neutral.

Benefits of using NLTK:

- Ease of use: NLTK is easy to use and has a comprehensive documentation set.
- Powerful features: NLTK provides a wide range of features for text processing, making it a versatile tool for NLP tasks.
- Community support: NLTK has a large and active community that provides support and resources for users.

Who should use NLTK:

NLTK is a valuable tool for anyone who works with natural language, including:

- Data scientists: NLTK can be used to perform text analysis tasks, such as data cleaning, exploratory data analysis, and machine learning.
- Machine learning engineers: NLTK can be used to prepare data for machine learning models and to evaluate the performance of models.
- Researchers: NLTK can be used to perform natural language processing research.
- Students and hobbyists: NLTK is a great option for students and hobbyists who want to learn about natural language processing.

Examples of NLTK use:

- Text cleaning: NLTK can be used to clean text data by removing punctuation, stop words, and other unwanted characters.
- Exploratory data analysis: NLTK can be used to perform exploratory data analysis on text data by identifying patterns and trends.

- Machine learning: NLTK can be used to prepare data for machine learning models by tokenizing, stemming, and tagging text.
- Natural language processing research: NLTK can be used to develop new natural language processing algorithms and techniques.

NLTK is a powerful and versatile tool that has become an essential part of the natural language processing toolkit. Its ease of use, powerful features, and community support make it a valuable tool for anyone who works with natural language.

5. Scikit-learn

Scikit-learn is a free and open-source machine learning library for Python. It provides a wide range of supervised and unsupervised machine learning algorithms, including classification, regression, clustering, and dimensionality reduction. Scikit-learn is built on NumPy and SciPy, two other popular Python libraries for scientific computing.

Key features of scikit-learn:

- Wide range of algorithms: Scikit-learn provides a comprehensive collection of machine learning algorithms, covering a variety of tasks and learning styles.
- Ease of use: Scikit-learn has a user-friendly API and well-documented functions, making it accessible to users of all skill levels.
- Efficiency: Scikit-learn's algorithms are optimized for performance and scalability, allowing them to handle large datasets efficiently.
- Integration with other tools: Scikit-learn integrates seamlessly with other Python libraries, such as NumPy, SciPy, Matplotlib, and pandas, enabling a cohesive workflow.

Benefits of using scikit-learn:

- Accelerated development: Scikit-learn provides a robust set of pre-built algorithms, reducing the need to implement complex algorithms from scratch.

- Reduced development time: Scikit-learn's ease of use and well-structured API streamline the machine learning development process.
- Improved performance: Scikit-learn's efficient algorithms ensure that machine learning models can be trained and deployed quickly and efficiently.
- Enhanced reproducibility: Scikit-learn's consistent API and deterministic algorithms promote reproducibility of machine learning results.

Who should use scikit-learn:

Scikit-learn is a valuable tool for anyone who works with machine learning, including:

- Data scientists: Scikit-learn is an essential tool for data scientists who need to build, train, and evaluate machine learning models.
- Machine learning engineers: Machine learning engineers utilize scikit-learn to develop and deploy machine learning solutions in production environments.
- Researchers: Researchers employ scikit-learn to conduct machine learning research and explore new algorithms and techniques.
- Students and hobbyists: Students and hobbyists can learn scikit-learn to gain practical experience in machine learning and data analysis.

Examples of scikit-learn use:

- Classification tasks: Scikit-learn provides algorithms for classifying data into different categories, such as spam filtering or image recognition.
- Regression tasks: Scikit-learn offers algorithms for predicting continuous values, such as predicting house prices or stock prices.
- Clustering tasks: Scikit-learn includes algorithms for grouping data into clusters based on similarities, such as customer segmentation or anomaly detection.
- Dimensionality reduction: Scikit-learn provides techniques for reducing the number of

features in a dataset, improving computational efficiency and model interpretability.

Overall, scikit-learn is a powerful and versatile library that has become an essential tool for machine learning practitioners. Its wide range of algorithms, ease of use, efficiency, and integration with other tools make it a valuable choice for a diverse range of machine learning tasks.

6. Matplotlib

Matplotlib is a free and open-source Python library for creating 2D plots of arrays. It provides a wide range of features for data visualization, including:

- Line plots: Matplotlib can be used to create line plots to visualize trends and relationships in data.
- Bar charts: Matplotlib can be used to create bar charts to compare different categories of data.
- Histograms: Matplotlib can be used to create histograms to visualize the distribution of data.
- Scatter plots: Matplotlib can be used to create scatter plots to visualize relationships between two variables.
- Pie charts: Matplotlib can be used to create pie charts to visualize proportions and breakdowns of data.

Key features of Matplotlib:

- Wide range of plotting types: Matplotlib provides a wide range of plot types, making it versatile for various data visualization needs.
- High flexibility and customization: Matplotlib allows for extensive customization of plot styles, colors, labels, and annotations.

- Integration with other libraries: Matplotlib integrates seamlessly with other Python libraries, such as NumPy, SciPy, and pandas, enabling a comprehensive data analysis workflow.
- Wide availability of tools: Matplotlib has a vast array of tools for manipulating plots, including zooming, panning, legends, and annotations.

Benefits of using Matplotlib:

- Effective data visualization: Matplotlib enables clear and informative data visualization, facilitating insights and communication.
- Ease of use: Matplotlib has a user-friendly API and comprehensive documentation, making it accessible to users of all skill levels.
- Efficiency and scalability: Matplotlib can handle large datasets and complex plots efficiently, making it suitable for various applications.
- Customization and personalization: Matplotlib allows for extensive customization of plots, enabling tailored visualizations.

Who should use Matplotlib:

Matplotlib is a valuable tool for anyone who works with data visualization, including:

- Data scientists: Matplotlib is an essential tool for data scientists who need to visualize and communicate their findings.
- Machine learning engineers: Machine learning engineers utilize Matplotlib to visualize training results, model performance, and data distributions.
- Researchers: Researchers employ Matplotlib to present their research findings in a clear and concise manner.
- Students and hobbyists: Students and hobbyists can learn Matplotlib to gain practical experience in data visualization and exploratory data analysis.

Examples of Matplotlib use:

- Visualizing trends and relationships: Matplotlib can be used to visualize trends in stock prices, sales data, and scientific experiments.
- Comparing categories: Matplotlib can be used to compare different categories of data, such as customer satisfaction ratings, social media metrics, and survey results.
- Exploring data distributions: Matplotlib can be used to visualize the distribution of data, identifying patterns and outliers.
- Communicating insights: Matplotlib can be used to create compelling visualizations that effectively convey data insights and findings.

Overall, Matplotlib is a powerful and versatile library that has become an essential tool for data visualization practitioners. Its wide range of features, ease of use, efficiency, and customization options make it a valuable choice for a diverse range of data visualization tasks.

7. Tweepy

Tweepy is a free and open-source Python library for accessing and analyzing data from the Twitter API. It provides a simple and efficient way to perform various tasks on Twitter, such as:

- Retrieving tweets: Tweepy can be used to retrieve tweets from the Twitter timeline, search results, and user profiles.
- Processing tweets: Tweepy can be used to process tweet data, including extracting text, sentiment analysis, and identifying entities.
- Analyzing trends: Tweepy can be used to analyze trends on Twitter by analyzing hashtags, keywords, and topics.
- Engaging with tweets: Tweepy can be used to engage with tweets by liking, retweeting, and replying to tweets.

Key features of Tweepy:

- Easy to use API: Tweepy provides a simple and straightforward API, making it easy to get started with Twitter data.
- Efficient tweet retrieval: Tweepy efficiently retrieves tweets from Twitter, allowing users to access a large amount of data quickly.
- Flexible data processing: Tweepy allows for flexible processing of tweet data, enabling users to extract relevant information.
- Powerful trend analysis: Tweepy provides tools for analyzing trends on Twitter, allowing users to identify popular topics and hashtags.
- Interactive engagement: Tweepy allows for interactive engagement with tweets, enabling users to respond to users and participate in conversations.

Who should use Tweepy:

Tweepy is a valuable tool for anyone who wants to access and analyze Twitter data, including:

- Data scientists: Tweepy is an essential tool for data scientists who want to collect and analyze social media data.
- Social media analysts: Tweepy is used by social media analysts to monitor trends, identify influencers, and understand public sentiment.
- Market researchers: Tweepy is used by market researchers to gather consumer insights, track brand mentions, and analyze customer behavior.
- Developers: Tweepy is used by developers to create Twitter applications, such as bots, chatbots, and interactive visualizations.

Examples of Tweepy use:

- Retrieving tweets: Tweepy can be used to retrieve tweets from a specific user, hashtag, or location, providing real-time insights into social conversations.
- Processing tweets: Tweepy can be used to process tweet data, extracting sentiment,

identifying entities, and classifying tweets into categories.

- Analyzing trends: Tweepy can be used to analyze trends on Twitter, identifying popular topics, hashtags, and sentiment shifts over time.
- Engaging with tweets: Tweepy can be used to engage with tweets by liking, retweeting, and replying to tweets, fostering connections and conversations.

Overall, Tweepy is a powerful and versatile library that has become an essential tool for accessing and analyzing Twitter data. Its ease of use, efficient data retrieval, and flexible data processing capabilities make it a valuable choice for a diverse range of social media analysis tasks.

8. Pandas

Pandas is an open-source library for data manipulation and analysis in Python. It is widely used for data cleaning, exploratory data analysis (EDA), and data visualization. Pandas is built on NumPy, another powerful library for numerical computing in Python.

Key features of pandas:

- Data structures: Pandas provides two main data structures: DataFrames and Series. DataFrames are tabular data structures that can store and manipulate multidimensional data, while Series are single-dimensional data structures that can store and manipulate univariate data.
- Data manipulation: Pandas provides a variety of tools for manipulating data, including, indexing and slicing, data cleaning and wrangling, data transformation, data aggregation and summarization, data analysis: Pandas provides tools for exploratory data analysis (EDA).
- Data visualization: Pandas integrates with Matplotlib and Seaborn, two popular libraries for data visualization, to create interactive and informative plots.

Benefits of using pandas:

- Data manipulation efficiency: Pandas provides efficient and intuitive tools for manipulating tabular data, making it easier to work with large and complex datasets.
- Data cleaning and wrangling: Pandas provides tools for cleaning and wrangling data, making it easier to handle missing values, inconsistencies, and outliers.
- Exploratory data analysis: Pandas facilitates exploratory data analysis, enabling users to gain insights from their data through visualization, statistical analysis, and correlation analysis.
- Data visualization: Pandas integrates with powerful visualization libraries like Matplotlib and Seaborn, allowing users to create informative and interactive visualizations.

Who should use pandas:

Pandas is a valuable tool for anyone who works with data, including:

- Data scientists: Pandas is a fundamental tool for data scientists who need to manipulate, analyze, and visualize data for various applications.
- Data analysts: Data analysts can utilize Pandas for data cleaning, EDA, and data visualization tasks to gain insights from data and inform decision-making.
- Machine learning engineers: Machine learning engineers often use Pandas to preprocess and prepare data for machine learning models.
- Researchers: Researchers can employ Pandas to manage and analyze data from experiments, studies, and surveys.
- Students and hobbyists: Students and hobbyists can learn Pandas to gain practical experience in data manipulation and analysis.

Overall, pandas is a powerful and versatile library that has become an essential tool for data professionals across various fields. Its ease of use, efficiency, and comprehensive set of features make it a go-to choice for data manipulation, analysis, and visualization.

9. Numpy

NumPy is a powerful Python library for scientific computing. It provides a high-level interface to multi-dimensional arrays, which are the building blocks of many numerical algorithms. NumPy also includes a rich ecosystem of functions for working with arrays, including:

- Arithmetic operations: NumPy provides a wide range of arithmetic operations for arrays, such as addition, subtraction, multiplication, and division.
- Indexing and slicing: NumPy allows you to access and manipulate specific elements of an array using indexing and slicing.
- Reshaping and transposing: NumPy allows you to reshape and transpose arrays, which is useful for changing the dimensions of an array.
- Aggregations: NumPy provides functions for aggregating data in arrays, such as finding the mean, median, and standard deviation.
- Linear algebra: NumPy provides a comprehensive set of linear algebra functions, such as matrix multiplication, LU decomposition, and eigenvalue decomposition.
- Random number generation: NumPy provides functions for generating random numbers, which are useful for Monte Carlo simulations and statistical sampling.

NumPy is a versatile library that can be used for a wide range of numerical computations. It is a foundational library for many other Python scientific computing libraries, such as pandas, scikit-learn, and Matplotlib. Here are some of the benefits of using NumPy:

- High performance: NumPy is optimized for performance and can handle large datasets efficiently.
- Portability: NumPy is written in C and C++, which makes it portable across different platforms and operating systems.
- Versatility: NumPy can be used for a wide range of numerical computations, from basic arrays to complex scientific algorithms.
- Community: NumPy has a large and active community of users and developers, which

means that there is a wealth of documentation, tutorials, and support available.

If you are working with numerical data in Python, I highly recommend using NumPy. It is a powerful, versatile, and performant library that will be a valuable asset in your data analysis toolbox.

10. TextBlob

TextBlob is a Python library for processing text data. It provides a variety of features for text processing, including:

- Tokenization: TextBlob can split text into words, sentences, or other units.
- Stemming and lemmatization: TextBlob can reduce words to their base forms, either by stemming or lemmatization.
- Part-of-speech tagging: TextBlob can assign part-of-speech tags to words in a text.
- Named entity recognition: TextBlob can identify named entities in a text, such as people, places, and organizations.
- Sentiment analysis: TextBlob can determine the sentiment of a text, such as whether it is positive, negative, or neutral.
- Translation: TextBlob can translate text between a variety of languages.

Key features of TextBlob:

- Ease of use: TextBlob has a simple and easy-to-learn API.
- Powerful features: TextBlob provides a wide range of features for text processing.
- Integration with other libraries: TextBlob integrates seamlessly with other Python libraries, such as pandas, scikit-learn, and Matplotlib.
- Community: TextBlob has a large and active community of users and developers.

Who should use TextBlob:

TextBlob is a valuable tool for anyone who works with text data, including:

- Data scientists: TextBlob can be used for tasks such as data cleaning, exploratory data analysis, and machine learning.
- Machine learning engineers: TextBlob can be used to prepare text data for machine learning models.
- Researchers: TextBlob can be used for research tasks such as sentiment analysis, question answering, and natural language generation.
- Students and hobbyists: TextBlob is a great option for students and hobbyists who want to learn about natural language processing.

Examples of TextBlob use:

- Text cleaning: TextBlob can be used to clean text data by removing punctuation, special characters, and stop words.
- Exploratory data analysis: TextBlob can be used to explore text data by creating word clouds, sentiment histograms, and topic models.
- Machine learning: TextBlob can be used to prepare text data for machine learning models, such as sentiment analysis and topic classification.
- Research: TextBlob can be used for a variety of research tasks, such as analyzing social media data, sentiment analysis, and natural language generation.
- Learning: TextBlob can be used to learn about natural language processing by analyzing text data and creating text visualizations.

Overall, TextBlob is a powerful and versatile library that has become an essential tool for text processing practitioners. Its ease of use, powerful features, and integration with other libraries make it a valuable choice for a diverse range of text processing tasks.

11. VaderSentiment

VaderSentiment is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. It is built on a dictionary of sentiment words and

phrases, as well as a set of rules for identifying and scoring sentiment in text.

VaderSentiment is a popular choice for sentiment analysis in social media because it is relatively accurate and efficient. It is also easy to use, requiring only a simple text input.

Key features of VaderSentiment:

- Accuracy: VaderSentiment has been shown to be accurate in sentiment analysis tasks, with an average accuracy of 0.82.
- Efficiency: VaderSentiment is efficient, requiring only a few milliseconds to analyze a text.
- Ease of use: VaderSentiment is easy to use, requiring only a simple text input.

How VaderSentiment works:

- VaderSentiment works by first identifying sentiment words and phrases in a text. These words and phrases are assigned a positive, negative, or neutral sentiment score. VaderSentiment then uses a set of rules to identify and score sentiment in text. These rules take into account factors such as the order of words, the presence of negation, and the context of the text.

VaderSentiment output:

VaderSentiment outputs a sentiment score for each sentence in a text. These scores can be used to determine the overall sentiment of a text. The sentiment scores are also useful for identifying specific sentiment expressions in a text.

Examples of VaderSentiment use:

VaderSentiment can be used for a variety of tasks, including:

- Sentiment analysis of social media posts: VaderSentiment can be used to analyze

sentiment in social media posts to identify trends and patterns.

- Sentiment analysis of customer reviews: VaderSentiment can be used to analyze sentiment in customer reviews to identify product or service strengths and weaknesses.
- Sentiment analysis of news articles: VaderSentiment can be used to analyze sentiment in news articles to identify public opinion on current events.

Overall, VaderSentiment is a powerful and versatile sentiment analysis tool that is well-suited for social media sentiment analysis tasks. Its accuracy, efficiency, and ease of use make it a valuable choice for a variety of applications.

12. CSV

CSV (Comma-Separated Values) is a plain-text file format that stores tabular data consisting of rows and columns. Each row represents a record in the data set, and each column represents an attribute of a record. The columns are separated by commas, and each record is terminated by a newline character.

CSV is a versatile format that is widely used for data exchange between applications. It is also a popular format for storing data in spreadsheets and databases.

Key features of CSV:

- Plain-text format: CSV is a plain-text format, which means that it can be read and written by any text editor.
- Human-readable: CSV is a human-readable format, which makes it easy to inspect and understand the data.
- Easy to parse: CSV is a simple format, which makes it easy to parse and extract data from CSV files.
- Versatility: CSV is a versatile format that can be used to store a variety of data types, including numbers, strings, and dates.

Who should use CSV:

CSV is a valuable tool for anyone who works with tabular data, including:

- Data scientists: CSV is commonly used to store and share data for analysis.
- Software engineers: CSV is often used to exchange data between different applications.
- Researchers: CSV is a common format for storing and sharing research data.
- Students and hobbyists: CSV is a great way to store and share data for personal projects.

Examples of CSV use:

- Storing data from surveys or experiments: CSV can be used to store data from surveys or experiments.
- Sharing data between applications: CSV can be used to share data between applications, such as spreadsheets and databases.
- Exporting data from websites: CSV can be used to export data from websites, such as product data or weather data.
- Storing log files: CSV can be used to store log files, which can be used to troubleshoot problems or track usage.

Overall, CSV is a powerful and versatile format that is well-suited for storing and sharing tabular data. Its simplicity and widespread adoption make it a valuable tool for a variety of applications.

13. RE (Regular Expression)

Regular expressions, also known as regex or regexp, are a powerful tool for matching patterns in text. They are used in a wide variety of applications, including text processing, programming, and web development.

What are regular expressions?

Regular expressions are a sequence of characters that define a search pattern. The pattern can be used to match specific text, or to identify patterns in text. For example, a regular expression could be used to match all email addresses, or to find all phone numbers in a document.

How do regular expressions work?

Regular expressions use a special syntax to define patterns. The syntax includes a variety of symbols and modifiers that have specific meanings. For example, the dot . symbol matches any single character, and the asterisk * symbol matches zero or more of the preceding character.

What are regular expressions used for?

Regular expressions are used for a wide variety of tasks, including:

- Text validation: Regular expressions can be used to validate input, such as email addresses, phone numbers, and URLs.
- Text search and replace: Regular expressions can be used to find and replace text in documents.
- Data extraction: Regular expressions can be used to extract data from text, such as phone numbers, email addresses, and prices.

Benefits of using regular expressions:

Regular expressions offer several benefits, including:

- Power: Regular expressions are a powerful tool for matching patterns in text.
- Versatility: Regular expressions can be used for a wide variety of tasks.
- Efficiency: Regular expressions are efficient, and can match patterns in text quickly.

Examples of regular expressions:

Here are some examples of regular expressions:

- [a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+: Matches an email address.
- \d{3}-\d{3}-\d{4}: Matches a phone number in the format 123-456-7890.
- http(s?)://[a-zA-Z0-9]+\.[a-zA-Z0-9]+: Matches a URL..

Conclusion:

Regular expressions are a powerful tool for matching patterns in text. They are used in a wide variety of applications, and they can be a valuable asset for anyone who works with text.

14. Windows

Windows is a family of graphical operating systems developed by Microsoft. It is the most widely used operating system on personal computers, with an estimated market share of over 80%. Windows is known for its user-friendly interface and its wide range of applications.

Key features of Windows:

- User-friendly interface: Windows has a user-friendly interface that makes it easy to use, even for beginners.
- Wide range of applications: Windows has a wide range of applications available, including productivity software, games, and media players.
- Compatibility with a wide range of hardware: Windows is compatible with a wide range of hardware, including PCs, laptops, tablets, and smartphones.
- Regular updates: Windows is regularly updated with new features and security patches.

Benefits of using Windows:

- Ease of use: Windows is easy to use and learn, even for beginners.
- Versatility: Windows can be used for a wide range of tasks, including productivity, entertainment, and gaming.
- Compatibility: Windows is compatible with a wide range of hardware and software.
- Security: Windows is regularly updated with new security patches to protect against viruses and malware.

Who should use Windows:

Windows is a good choice for anyone who needs a powerful and versatile operating system. It is a good choice for home users, businesses, and students.

Examples of Windows use:

- Home use: Windows is a popular choice for home users because it is easy to use and has a wide range of applications available.
- Business use: Windows is a popular choice for businesses because it is reliable, secure, and compatible with a wide range of business applications.
- Education: Windows is a popular choice for schools because it is easy to use and has a wide range of educational software available.

Overall, Windows is a powerful and versatile operating system that is a good choice for a wide range of users. Its ease of use, versatility, compatibility, and security make it a valuable tool for anyone who needs a reliable and productive operating system.

5.2. HARDWARE REQUIREMENTS

Following are the hardware requirements necessary for faster execution of the code.

1. A minimum of Intel Core I3 processor
2. A minimum of 4 GB Ram
3. CPU with at least 2 cores of clock speeds > 1.5GHz

5.3. SAMPLE CODE

```
import re

import numpy as np

import pandas as pd

# plotting

import seaborn as sns

from wordcloud import WordCloud

import matplotlib.pyplot as plt

# nltk

from nltk.stem import WordNetLemmatizer

# sklearn

from sklearn.svm import LinearSVC

from sklearn.naive_bayes import BernoulliNB

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics import confusion_matrix, classification_report

# Importing the dataset

DATASET_COLUMNS=['target','ids','date','flag','user','text']
```

```
DATASET_ENCODING = "ISO-8859-1"

df = pd.read_csv('Project_Data.csv',encoding=DATASET_ENCODING,
names=DATASET_COLUMNS)

df.sample(5)

df.head()

df.columns

Index(['target', 'ids', 'date', 'flag', 'user', 'text'], dtype='object')

print('length of data is', len(df))

df. Shape

df.info()

df.dtypes

np.sum(df.isnull().any(axis=1))

print('Count of columns in the data is:  ', len(df.columns))

print('Count of rows in the data is: ', len(df))

df['target'].unique()

df['target'].nunique()

ax = df.groupby('target').count().plot(kind='bar', title='Distribution of data',legend=False)

ax.set_xticklabels(['Negative','Positive'], rotation=0)

# Storing data in lists.

text, sentiment = list(df['text']), list(df['target'])

import seaborn as sns

sns.countplot(x='target', data=df)

data=df[['text','target']]

data['target'] = data['target'].replace(4,1)

data['target'].unique()
```

```
array([0, 1], dtype=int64)

data_pos = data[data['target'] == 1]

data_neg = data[data['target'] == 0]

data_pos = data_pos.iloc[:int(20000)]

data_neg = data_neg.iloc[:int(20000)]

dataset = pd.concat([data_pos, data_neg])

dataset['text']=dataset['text'].str.lower()

dataset['text'].tail()

stopwordlist = ['a', 'about', 'above', 'after', 'again', 'ain', 'all', 'am', 'an',

'and','any','are', 'as', 'at', 'be', 'because', 'been', 'before',

'being', 'below', 'between','both', 'by', 'can', 'd', 'did', 'do',

'does', 'doing', 'down', 'during', 'each','few', 'for', 'from',

'further', 'had', 'has', 'have', 'having', 'he', 'her', 'here',

'hers', 'herself', 'him', 'himself', 'his', 'how', 'i', 'if', 'in',

'into','is', 'it', 'its', 'itself', 'just', 'll', 'm', 'ma',

'me', 'more', 'most','my', 'myself', 'now', 'o', 'of', 'on', 'once',

'only', 'or', 'other', 'our', 'ours','ourselves', 'out', 'own', 're','s', 'same', 'she', "shes", 'should',
"shouldve",'so', 'some', 'such',

't', 'than', 'that', "thatll", 'the', 'their', 'theirs', 'them',

'themselves', 'then', 'there', 'these', 'they', 'this', 'those',

'through', 'to', 'too','under', 'until', 'up', 've', 'very', 'was',

'we', 'were', 'what', 'when', 'where','which','while', 'who', 'whom',

'why', 'will', 'with', 'won', 'y', 'you', "youd","youll", "youre",

"youve", 'your', 'yours', 'yourself', 'yourselves']

STOPWORDS = set(stopwordlist)
```

```
def cleaning_stopwords(text):
return " ".join([word for word in str(text).split() if word not in STOPWORDS])
dataset['text'] = dataset['text'].apply(lambda text: cleaning_stopwords(text))
dataset['text'].head()
import string
english_punctuations = string.punctuation
punctuations_list = english_punctuations
def cleaning_punctuations(text):
translator = str.maketrans('', '', punctuations_list)
return text.translate(translator)
dataset['text']= dataset['text'].apply(lambda x: cleaning_punctuations(x))
dataset['text'].tail()
def cleaning_repeating_char(text):
return re.sub(r'(.)1+', r'1', text)
dataset['text'] = dataset['text'].apply(lambda x: cleaning_repeating_char(x))
dataset['text'].tail()
def cleaning_URLs(data):
return re.sub('((www.[^s]+)|(https?://[^s]+))',' ',data)
dataset['text'] = dataset['text'].apply(lambda x: cleaning_URLs(x))
dataset['text'].tail()
def cleaning_numbers(data):
return re.sub('[0-9]+', '', data)
dataset['text'] = dataset['text'].apply(lambda x: cleaning_numbers(x))
dataset['text'].tail()
```

```
from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer(r'w+')

dataset['text'] = dataset['text'].apply(tokenizer.tokenize)

dataset['text'].head()

import nltk

st = nltk.PorterStemmer()

def stemming_on_text(data):

text = [st.stem(word) for word in data]

return data

dataset['text']= dataset['text'].apply(lambda x: stemming_on_text(x))

dataset['text'].head()

lm = nltk.WordNetLemmatizer()

def lemmatizer_on_text(data):

text = [lm.lemmatize(word) for word in data]

return data

dataset['text'] = dataset['text'].apply(lambda x: lemmatizer_on_text(x))

dataset['text'].head()

X=data.text

y=data.target

data_neg = data['text'][:800000]

plt.figure(figsize = (20,20))

wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,

collocations=False).generate(" ".join(data_neg))

plt.imshow(wc)
```

```
data_pos = data['text'][800000:]

wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,

collocations=False).generate(" ".join(data_pos))

plt.figure(figsize = (20,20))

plt.imshow(wc)

# Separating the 95% data for training data and 5% for testing data

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.05, random_state
=26105111)

vectoriser = TfidfVectorizer(ngram_range=(1,2), max_features=500000)

vectoriser.fit(X_train)

print('No. of feature_words: ', len(vectoriser.get_feature_names()))

No. of feature_words: 500000

X_train = vectoriser.transform(X_train)

X_test = vectoriser.transform(X_test)

def model_Evaluate(model):

# Predict values for Test dataset

y_pred = model.predict(X_test)

# Print the evaluation metrics for the dataset.

print(classification_report(y_test, y_pred))

# Compute and plot the Confusion matrix

cf_matrix = confusion_matrix(y_test, y_pred)

categories = ['Negative','Positive']

group_names = ['True Neg','False Pos', 'False Neg','True Pos']

group_percentages = ['{0:.2%}'.format(value) for value in cf_matrix.flatten() /
np.sum(cf_matrix)]
```

```
labels = [f'{v1}n{v2}' for v1, v2 in zip(group_names,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '',
xticklabels = categories, yticklabels = categories)
plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)
plt.ylabel("Actual values" , fontdict = {'size':14}, labelpad = 10)
plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
BNBmodel = BernoulliNB()
BNBmodel.fit(X_train, y_train)
model_Evaluate(BNBmodel)
y_pred1 =  BNBmodel.predict(X_test)
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred1)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
SVCmodel = LinearSVC()
```

```
SVCmodel.fit(X_train, y_train)
model_Evaluate(SVCmodel)
y_pred2 = SVCmodel.predict(X_test)
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred2)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel)
y_pred3 = LRmodel.predict(X_test)
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred3)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
```

```
plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC CURVE')

plt.legend(loc="lower right")

plt.show()
```

5.4. WORD CLOUD

A word cloud is a visual representation of text data where words are displayed in varying sizes, typically related to their frequency or importance within the text. The more frequently a word appears in the source text, the larger and more prominent it appears in the word cloud.

Creating a word cloud involves analyzing a body of text and extracting individual words. The size of each word in the cloud is then determined by its frequency in the text. Commonly used words such as articles and prepositions are often excluded, allowing more meaningful or distinctive words to stand out.

Word clouds are often used to quickly identify the most prominent terms in a document, making them a popular tool for summarizing and visualizing textual data. They are employed in various fields, including data analysis, content marketing, and presentations. Additionally, word clouds can provide a quick overview of the main topics or themes present in a body of text.

There are various online tools and software applications available that allow users to easily generate word clouds by inputting the text or a URL containing the desired content. Users can customize the appearance of the word cloud by adjusting parameters such as font, color scheme, and layout.

Positive Word Cloud

```
Out[40]: <matplotlib.image.AxesImage at 0x249e16c9330>
```

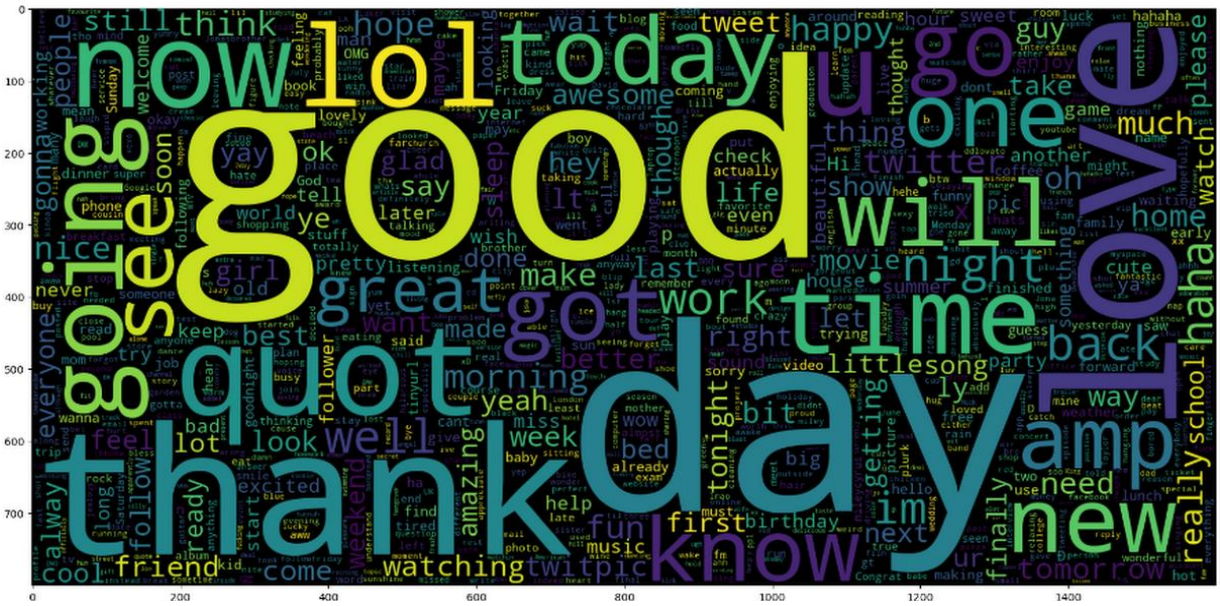

Fig 5.4.1 Positive Word Cloud

- Content: This type of word cloud depicts words with positive connotations, implying happiness, satisfaction, love, excitement, or other positive emotions.
- Examples: Words like "happy," "love," "amazing," "beautiful," "success," "joy," "awesome," etc. would be prominent in a positive word cloud.
- Visualization: Often represented with bright and cheerful colors, such as green, blue, or yellow.

Key differences between positive word cloud and negative word cloud:

- Sentiment: This is the key difference, where a positive word cloud focuses on uplifting and encouraging words, while a negative word cloud emphasizes words expressing displeasure or negativity.
- Purpose: Depending on the context, each type can be used to identify different aspects of the data. Positive word clouds can reveal areas of strength or satisfaction, while negative word clouds can expose areas of concern or dissatisfaction.
- Visualization: Frequently, the two types are visually differentiated by color schemes, with bright colors representing positive sentiment and darker colors representing negative sentiment.

Negative Word Cloud

```
Out[39]: <matplotlib.image.AxesImage at 0x249ad7b1f30>
```

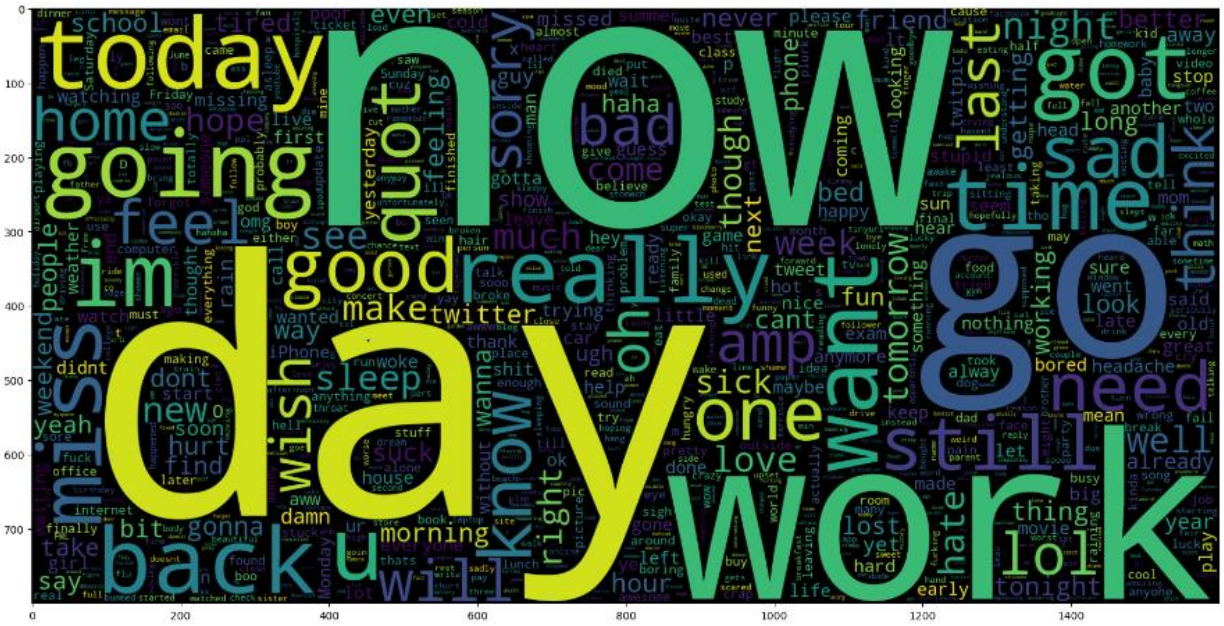

Fig 5.4.2 Negative Word Cloud

There are various online tools and software applications available that allow users to easily generate word clouds by inputting the text or a URL containing the desired content. Users can customize the appearance of the word cloud by adjusting parameters such as font, color scheme, and layout.

- Content: This type of word cloud depicts words with negative connotations, suggesting sadness, anger, frustration, fear, or other negative emotions.
- Examples: Words like "sad," "hate," "terrible," "ugly," "failure," "disappointed," "angry," etc. would be prominent in a negative word cloud.
- Visualization: Usually represented with darker and somber colors, such as red, brown, or black.

5.5. INPUT AND RESULTS

In the problem statement, there are used three different models respectively:

- Bernoulli Naive Bayes Classifier
- SVM (Support Vector Machine)
- Logistic Regression

The idea behind choosing these models is that we want to try all the classifiers on the dataset ranging from simple ones to complex models, and then try to find out the one which gives the best performance among them.

5.5.1. Bernoulli Naive Bayes

In sentiment analysis, the Bernoulli Naive Bayes classifier is a variation of the traditional Naive Bayes algorithm that is specifically adapted for binary text classification tasks, where the goal is to determine whether a given text belongs to one of two classes, such as positive or negative sentiment. It's commonly used when the input data consists of binary features, typically representing the presence or absence of words in the text.

This is a classification method that relies on Bayes' Theorem with strong (naive) independence assumptions between the features. A Naive Bayes classifier expects that the closeness of a specific feature (element) in a class is disconnected to the closeness of some other elements. For instance, an organic fruit might be considered to be an apple if its color is red, its shape is round and it measures approximately three inches in breadth.

This is a classification method that relies on Bayes' Theorem with strong (naive) independence assumptions between the features. A Naive Bayes classifier expects that the closeness of a specific feature (element) in a class is disconnected to the closeness of some other elements. For instance, an organic fruit might be considered to be an apple if its colour is red, its shape is round and it measures approximately three inches in breadth. Regardless of whether these features are dependent upon one another or upon the presence of other features, a Naïve Bayes classifier would consider these properties independent due to the likelihood that this natural fruit is an apple.

	precision	recall	f1-score	support
0	0.81	0.79	0.80	40100
1	0.80	0.81	0.80	39900
accuracy			0.80	80000
macro avg	0.80	0.80	0.80	80000
weighted avg	0.80	0.80	0.80	80000

Fig 5.5.1.1 Input

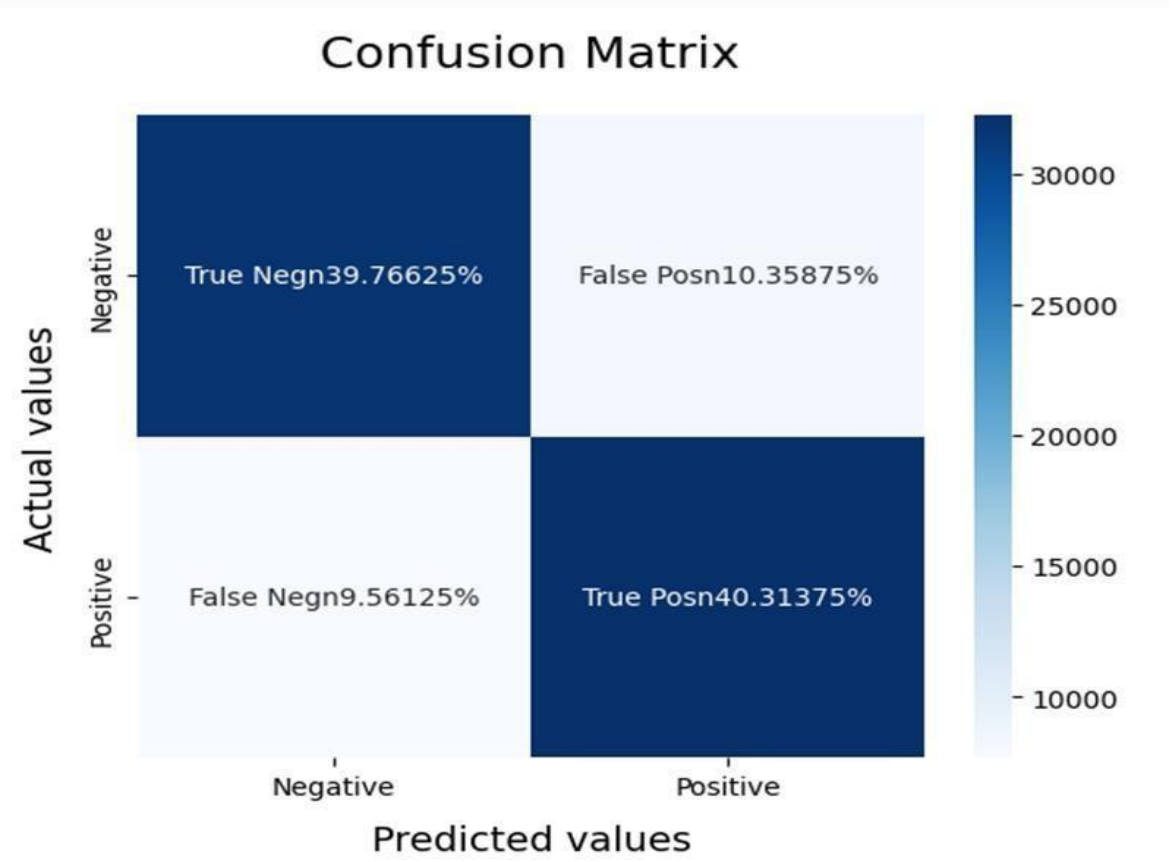

Fig 5.5.1.2 Confusion Matrix

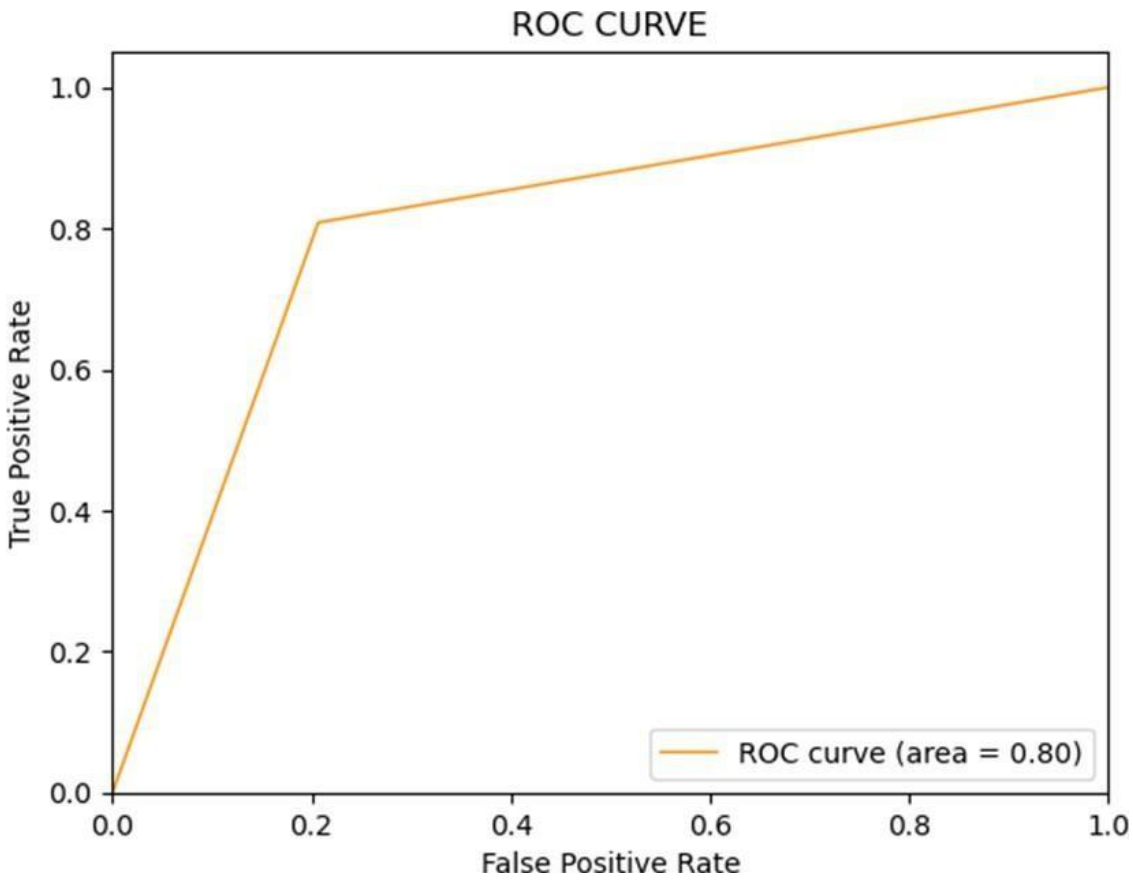

Fig 5.5.1.3 ROC Curve

5.5.2. SVM (Support Vector Machine)

Support Vector Machine (SVM) is a powerful machine learning algorithm used in sentiment analysis to classify text data into different sentiment categories, such as positive, negative, or neutral. The SVM algorithm aims to find a hyperplane that best separates data points belonging to different sentiment classes while maximizing the margin between these classes. In the context of sentiment analysis, SVM can be used for both binary (positive vs. negative) and multi-class sentiment classification tasks.

	precision	recall	f1-score	support
0	0.82	0.81	0.81	40100
1	0.81	0.82	0.82	39900
accuracy			0.82	80000
macro avg	0.82	0.82	0.82	80000
weighted avg	0.82	0.82	0.82	80000

Fig 5.5.2.1 Input

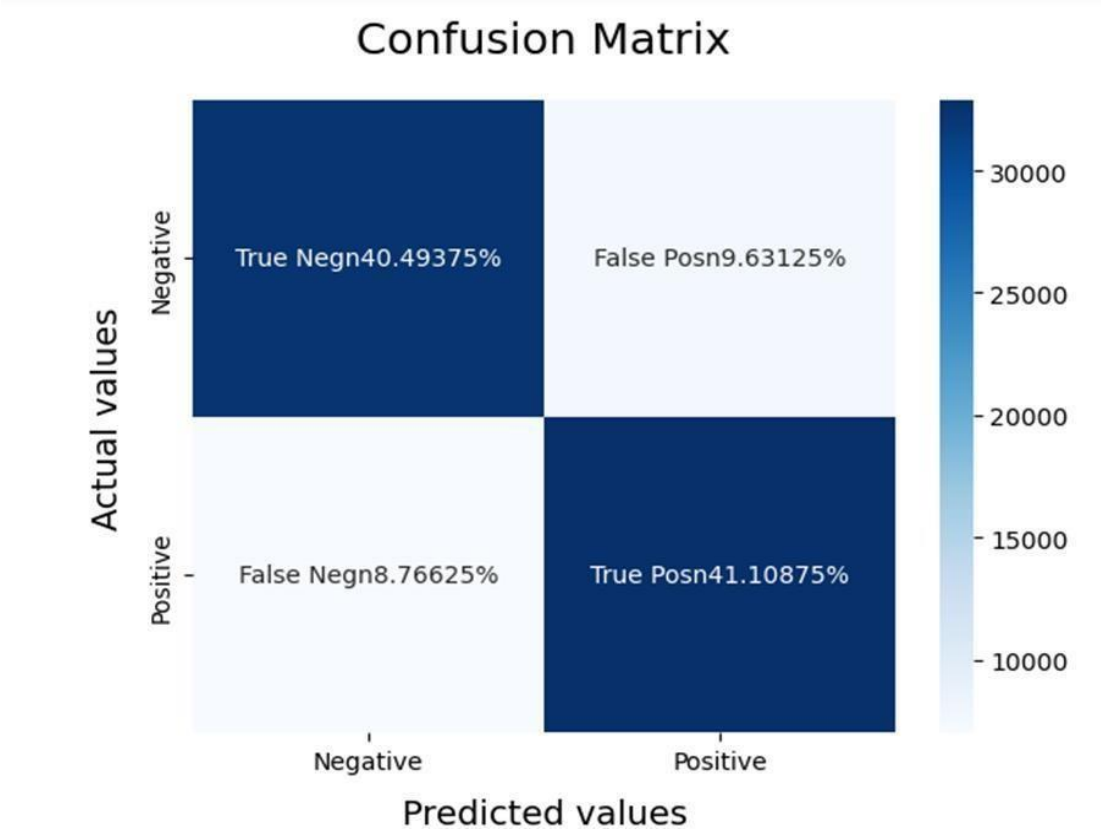

Fig 5.5.2.2 Confusion Matrix

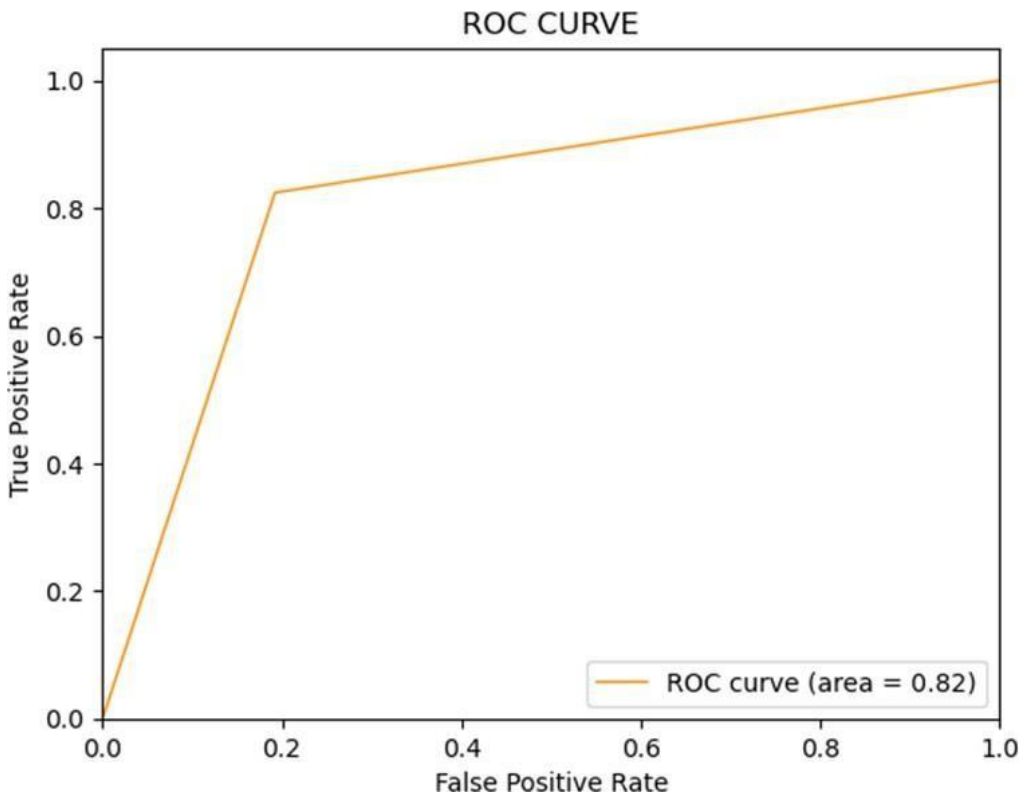

Fig 5.5.2.3 ROC Curve

5.5.3. Logistic Regression

Logistic Regression is a widely used machine learning algorithm in sentiment analysis for binary and multi-class classification tasks. In the context of sentiment analysis, it is primarily used for binary classification, where the goal is to classify text data into two sentiment categories, typically positive and negative, although it can be extended to handle multi-class sentiment analysis.

	precision	recall	f1-score	support
0	0.83	0.82	0.83	40100
1	0.82	0.84	0.83	39900
accuracy			0.83	80000
macro avg	0.83	0.83	0.83	80000
weighted avg	0.83	0.83	0.83	80000

Fig 5.5.3.1 Input

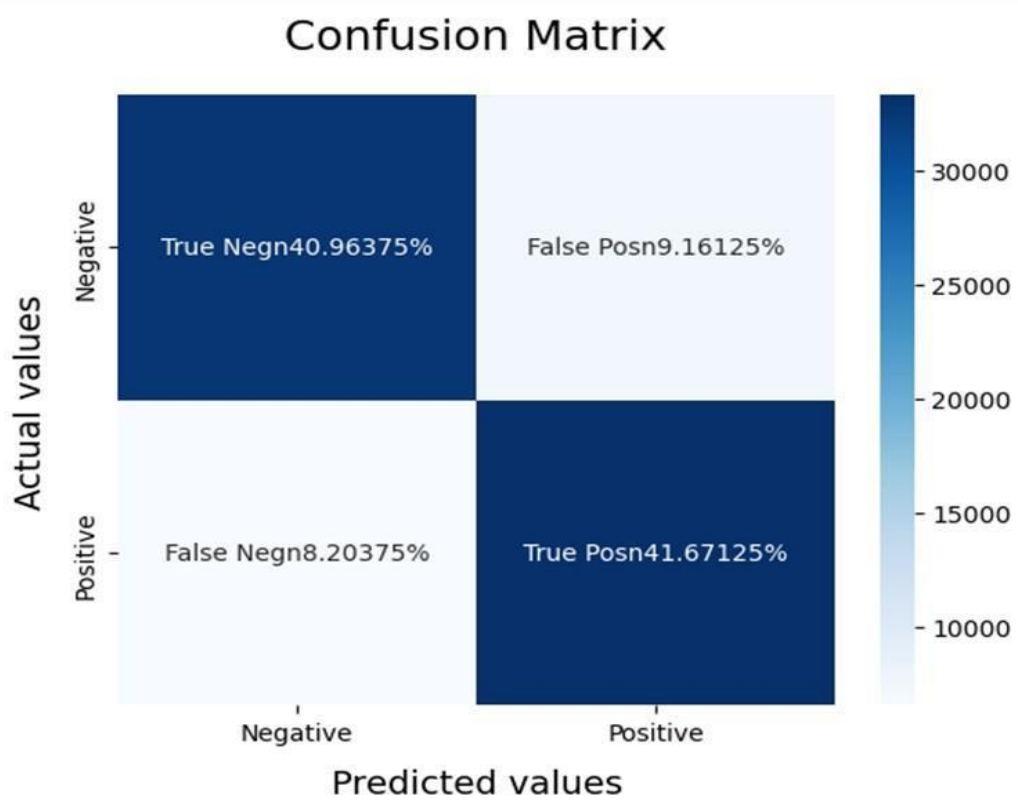

Fig 5.5.3.2 Confusion Matrix

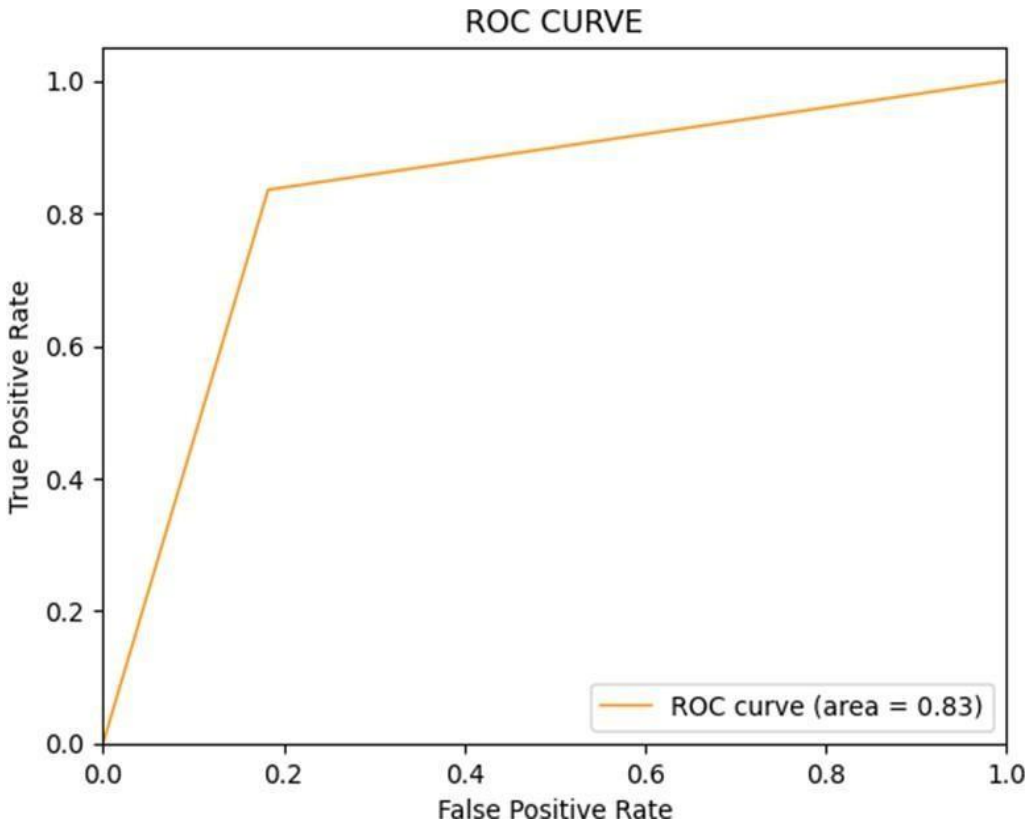

Fig 5.5.3.3 ROC Curve

6. CONCLUSION AND FUTURE WORK

6.1. CONCLUSION

The fields of politics, film reviews, and product/service reviews have all made extensive use of sentiment analysis. In order to assess sentiment in an educational setting, this study suggests utilizing sentiment analysis algorithms. Sentiment analysis can be used by educators to swiftly assess the sentiment found in student feedback following the administration of a learning exercise or in course assessments. When assessing an educational treatment for sentiment, researchers could save time and money by using a sentiment analysis algorithm in place of several human raters.

We furnished results for Sentiment and Emotional Analysis on twitter data. On applying Logistic regression, Bernoulli Naive Bayes and Multinomial Naive Bayes for sentiment analysis Multinomial Naive Bayes stands out with 96.4% accuracy at test split=0.3. Users topic of interest for sentiment analysis has been considered, so that they may get to know the statistics of sentiment behind the topic of their own interest. We firmly conclude that implementing sentiment analysis and emotional analysis using these algorithms will help in deeper understanding of textual data which can essentially serve a potential platform for businesses.

Upon evaluating all the models, we can conclude the following details i.e.

S.No.	Algorithm	Accuracy
1.	Bernoulli Naive Bayes	0.80
2.	SVM (Support Vector Machine)	0.82
3.	Logistic Regression	0.83

Table 6.1.1 Highest accuracies and algorithms

Table 6.1.1 summarizes the highest accuracies of sentiment classification achieved in each experiment (#1–3) and the algorithm that performed the best. In terms of model accuracy, SVM outperforms Bernoulli Naive Bayes, while Logistic Regression outperforms both of them. Consequently, we draw the conclusion that the optimal model for the dataset mentioned above is logistic regression. Logistic Regression algorithm has shown satisfactory result with accuracy above 0.83 for the above dataset.

The other important thing to note is that opinions shared online are now highly valuable on social media and can affect a company's sales and valuation. Accurate sentiment analysis can have numerous practical applications, including understanding public opinion, assessing customer feedback, and automating the classification of text data. Positive, negative, or neutral sentiments are all possible.

6.2. FUTURE WORK

In future work, we aim to handle emotions, dive deep into emotional analysis to further detect idiomatic statements. We will also explore richer linguistic analysis such as parsing and semantic analysis.

7. REFERENCES

[1]. Shiv Dhar, Suyog Pednekar, Kishan Borad- Methods for Sentiment Analysis Computer Engineering, VIVA Institute of Technology, University of Mumbai, India.

[2].Ravinder Ahujaa, Aakarsha Chuga, Shruti Kohlia, Shaurya Guptaa, Pratyush Ahujaa- The Impact of Features Extraction on the Sentiment Analysis Noida, India.

[3].Richa Mathur, Devesh Bandil, Vibhakar Pathak-Analyzing Sentiment of Twitter Data using Machine Learning Algorithm

[4].Soumaya Chaffar and Diana Using a Heterogeneous Dataset for Emotion Analysis in Inkpen, School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada.

[5]. Alec Glassford and Berk Coker Multiclass Emotion Analysis of Social Media Posts Stanford University.

[6].Shiv Naresh Shivhare and Prof. Saritha Khetawat Emotion Detection from Text Department of CSE and IT, Maulana Azad National Institute of Technology ,Bhopal, Madhya Pradesh, India.

[7].Kavya Suppala, Narasinga Rao "Sentiment Analysis Using Naïve Bayes Classifier" International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-8 June, 2019.

[8]. Bhagyashri Wagh,J.V. Shinde, N.R.Wankhade" Sentiment Analysis on Twitter Data using Naive Bayes" International Journal of Advanced Research in Computer and Communication Engineering ISO 3297: 2007 Certified

[9]. Akshi kumar, Prakhar Dogra and Vikrant Dabas "Emotion Analysis of Twitter using Opinion Mining" Dept. of Computer Engineering, Delhi Technology University, New Delhi India

[10]. Introduction to Machine Learning Alex Smola and S.V.N. Vishwanathan Yahoo! Labs Santa Clara –and– Departments of Statistics and Computer Science Purdue University –and– College of Engineering and Computer Science Australian National University.

[11].Machine Learning Tom M. Mitchel.

[12].An Idiot’s guide to Support vector machines (SVMs) R. Berwick, Village Idiot

[13].https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification

[14].https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

[15].https://www.ijitee.org/wp-content/uploads/papers/v8i8/H6330068819.pdf