

**SENTIMENT AND EMOTIONAL ANALYSIS OF USER
OPINIONS ON TWITTER DATA USING MACHINE
LEARNING TECHNIQUES**

A PROJECT REPORT

Submitted by

B.V.D.S.SAILENDRA	316126510124
K.KATYAYINI	316126510091
B.POOJA	316126510126
G.V.SAI PRASAD	316126510079

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING



Under esteemed guidance of

S.JOSHUA JOHNSON

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES
(AUTONOMOUS)**

(Affiliated to Andhra University)

SANGIVALASA:VISAKHAPATNAM -531162

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES

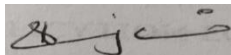
(AUTONOMOUS)

(Affiliated to Andhra University)

SANGIVALASA:VISAKHAPATNAM -531162

BONAFIDE CERTIFICATE

Certified that this report “**SENTIMENT AND EMOTIONAL ANALYSIS OF USER OPINIONS ON TWITTER DATA USING MACHINE LEARNING TECHNIQUES**” is the bonafide work of B.V.D.S.SAILENDRA(316126510124), K.KATYAYINI(316126510091), B.POOJA(316126510126), G.V.SAIPRASAD(31612651079)” who carried out the project under my supervision.



SIGNATURE

DR.R.SIVARANJINI

HEAD OF THE DEPARTMENT

COMPUTER SCIENCE ENGINEERING

ANIL NEERUKONDA INSTITUTE OF

TECHNOLOGY&SCIENCES

(AUTONOMOUS)



SIGNATURE

S.JOSHUA JOHNSON

ASSISTANT PROFESSOR

COMPUTER SCIENCE ENGINEERING

ANIL NEERUKONDA INSTITUTE OF

TECHNOLOGY&SCIENCES

(AUTONOMOUS)

DECLARATION

We, **B.V.D.S.Sailendra, K.Katyayini, B.Pooja, G.V.Sai Prasad**, of final semester B.Tech.,in the department of Computer Science Engineering from ANITS, Visakhapatnam,hereby declare that the project work entitled **“SENTIMENT AND EMOTIONAL ANALYSIS OF USER OPINIONS ON TWITTER DATA USING MACHINE LEARNING TECHNIQUES”** is carried out by us and submitted in partial fulfilment of the requirements for the award of **Bachelor of Technology in Computer Science Engineering** , under Anil Neerukonda Institute Of Technology & Sciences during the academic year 2016-2020 and has not been submitted to any other university for the award of any kind of degree.

B.V.D.S.Sailendra	316126510124
K.Katyayini	316126510091
B.Pooja	316126510126
G.V.Sai Prasad	316126510079

ACKNOWLEDGEMENT

An endeavor over a long period can be advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them. We owe our tributes to **Dr.R.Sivaranjani,Head of the Department, Computer Science Engineering** for her valuable support and guidance during the period of project implementation.

We wish to express our sincere thanks and gratitude to our project guide **S.Joshua Johnson,AssistantProfessor,DepartmentofComputer Science And Engineering,ANITS** for their simulating discussions, in analyzing problems associated with our project work and for guiding us throughout the project. Project meeting were highly informative. We express our sincere thanks for the encouragement, untiring guidance and the confidence they had shown in us. We are immensely indebted for their valuable guidance throughout our project. We also thank our B.Tech project coordinator **K.S.Deepthi**, for her support and encouragement.

We also thank all the staff members of CSE department for their valuable advice. We also thank Principal and supporting staff for providing resources as and when required.

B.V.D.S.Sailendra	316126510124
K.Katyayini	316126510091
B.Pooja	316126510126
G.V.Sai Prasad	316126510079

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 everyday. 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, Preprocessing , Tokenization, Stemming , Lemmatization , Twitter.

TABLE OF CONTENTS

INDEX	TITLE	PAGENUMBER
1.	INTRODUCTION	1
	1.1 Motivation for work	3
	1.2 Problem Statement	3
2.	LITERATURE SURVEY	5
	2.1 Introduction	5
	2.2 Existing methods	5
	2.2.1 Using a heterogeneous dataset for Emotional Analysis in text.	5
	2.2.2 Multiclass Emotional Analysis on Social media posts	6
	2.2.3 Classification of emotions from text using svm based opinion mining	7
	2.2.4 Emotion detection from text	8
	2.2.5 Emotion detection and analysis on social media	8
	2.2.6 A Novel approach of Sentiment classification using emoticons	9
	2.2.7 Analyzing Sentiment of twitter data suing machine learning algorithm	9

2.2.8	The impact of features extraction on the Sentiment Analysis	10
2.2.9	Methods for Sentiment analysis	11
3.	METHODOLOGY	12
3.1	Proposed System	13
3.1.1	System Architecture	17
4.	DESIGN	24
4.1	Structure Chart	25
4.2	UML Diagrams	26
4.2.1	Use case diagram	27
4.2.2	Class diagram	28
4.2.3	Sequence diagram	29
4.2.4	Activity diagram	31
4.2.5	Collaboration diagram	32
4.2.6	Flow chart diagram	33
4.2.7	Component diagram	34
5.	EXPERIMENT ANALYSIS	35
5.1	System Configuration	35
5.1.1	Software Requirements	35
5.1.2	Hardware Requirements	35
5.2	Sample Code	36

5.3	Input and Output	55
5.3.1	Input	55
5.3.2	Output	57
5.4	Experiment results and analysis	59
6.	CONCLUSION AND FUTURE WORK	60
6.1	Conclusion	62
6.2	Future Work	62
7.	REFERENCES	63

LIST OF FIGURES

Extraction of data	3.1.1
Architecture diagram for Sentiment analysis using Naive Bayes	3.1.1.1
Naive Bayes Classifier	3.1.1.2
Architecture diagram for Emotional analysis using SVM	3.1.1.3
SVM Classifier(I)	3.1.1.4
SVM Classifier(II)	3.1.1.5
Structure chart	4.1.1
Use case diagram	4.2.1.1
Class diagram	4.2.2.1
Sequence diagram for Sentiment Analysis(I)	4.2.3.1
Sequence diagram for Emotional Analysis(II)	4.2.3.2
Activity diagram	4.2.4.1
Collaboration diagram	4.2.5.1
Flow chart diagram	4.2.6.1
Component diagram	4.2.7.1
Train data for Sentiment Analysis	5.3.1.1
Train data for Emotional Analysis	5.3.1.2

Result for Sentiment Analysis of twitter data	5.3.2.1
Result for Emotional Analysis of twitter data	5.3.2.2
Accuracy comparison	5.4.1
Train data and Test data splits	5.4.2
Scores based on test_split	5.4.3

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 techniqueprovides labels to classifier to make it understand the insights among various features.Once the classifiergetsfamiliarized 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 users 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 analyze 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 in online 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 analyzed 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 hold out 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 v-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 clusterings. The clustering algorithm is trained to produce desirable clusters, given sets of items and complete clusterings 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 clusterings without additional information from the user. Similarity measure maps pairs of items to a real number indicating how similar the pair is; positive values indicate the pair is alike, negative values, unlike. 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 need 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 calculate 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 this 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 labeling of the tweets was performed to eliminate tweets that convey mixed emotions. Only those tweets which have a percentage of more than 70% for a particular emotion are labeled 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 Labeled- 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 given 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 Analyzing 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 an 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 analyzed, 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 labeled dataset using textblob and setting the sentiment fields accordingly. Thus our train data set without preprocessing is ready. Next we perform preprocessing 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

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. The details about the steps adopted for the transformation of information are described in next subsections.

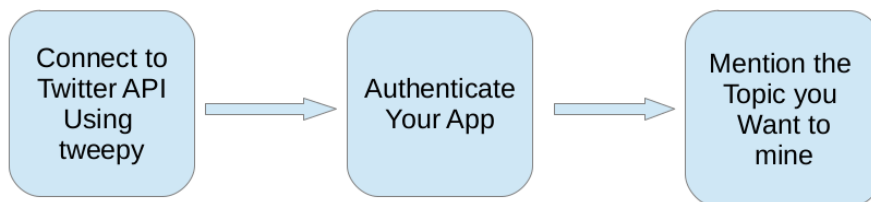


Fig 3.1.1 Extraction of data

Preprocessing Of Data:

Following are the Preprocessing steps that have been carried out:

Removing Html tags and urls:

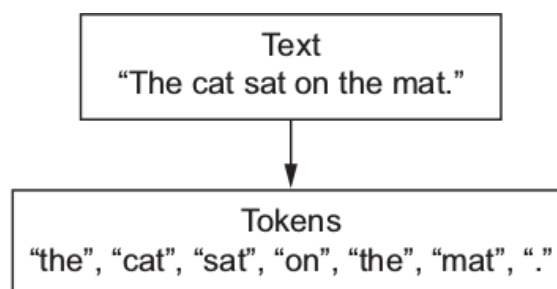
Html tags and urls often have minimum sentiments thus they are removed from tweets. Using regular expressions.

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.

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 tokenising the reviews into words.



Removing punctuations and special symbols:

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

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', '.']

After stop words removal:

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

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.

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

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.

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.

Result Analysis:

Here the accuracy of different classifiers are 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.

Visual Representation:

Our final results are plotted as pie charts which contains different fields such as positive, negative, neutral in case of sentiment analysis. where as 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.

3.1.1 SYSTEM ARCHITECTURE

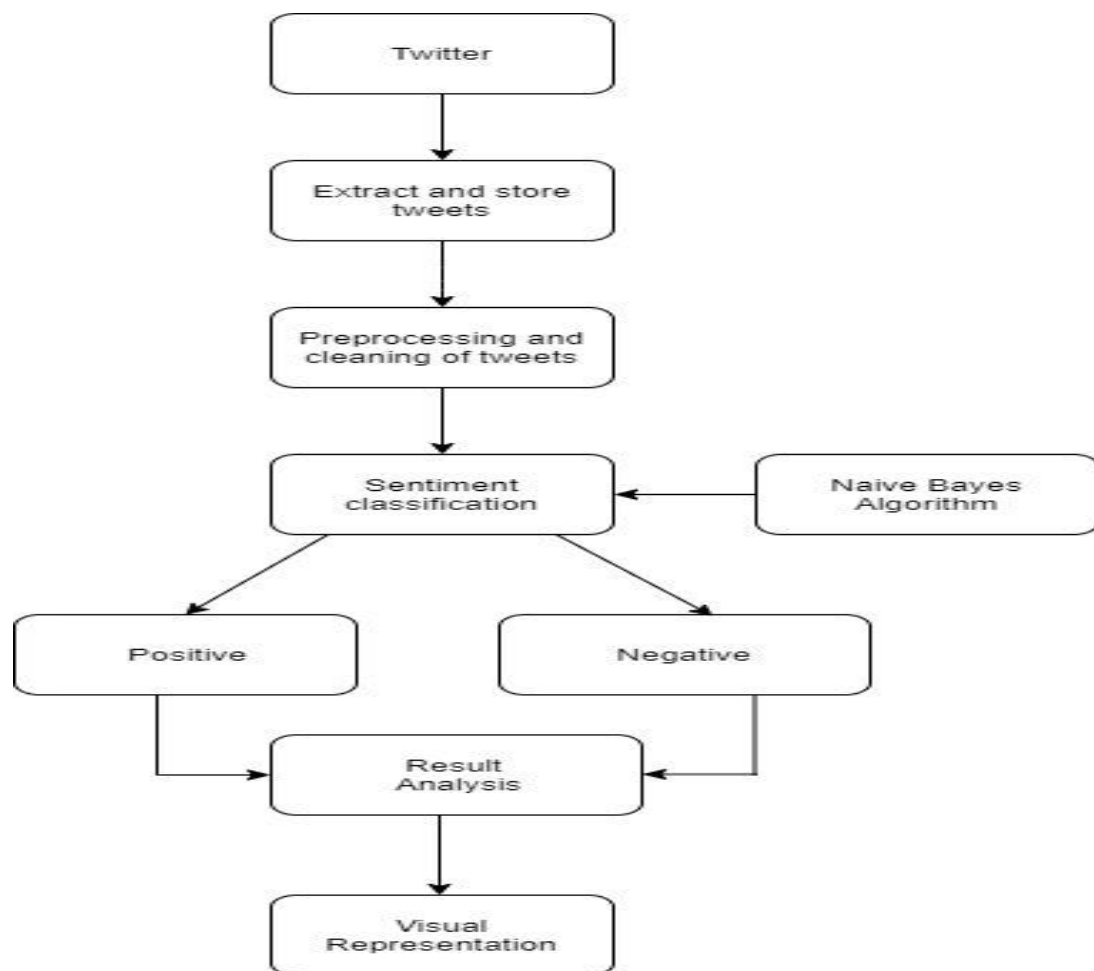


Fig 3.1.1.1 Architecture diagram for sentiment analysis using Naive Bayes

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

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

(Reference from Tom Mitchell)

$$P(X)$$

Where y is a class variable and X is feature vector.

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. 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 method of 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 $p(b)$.

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

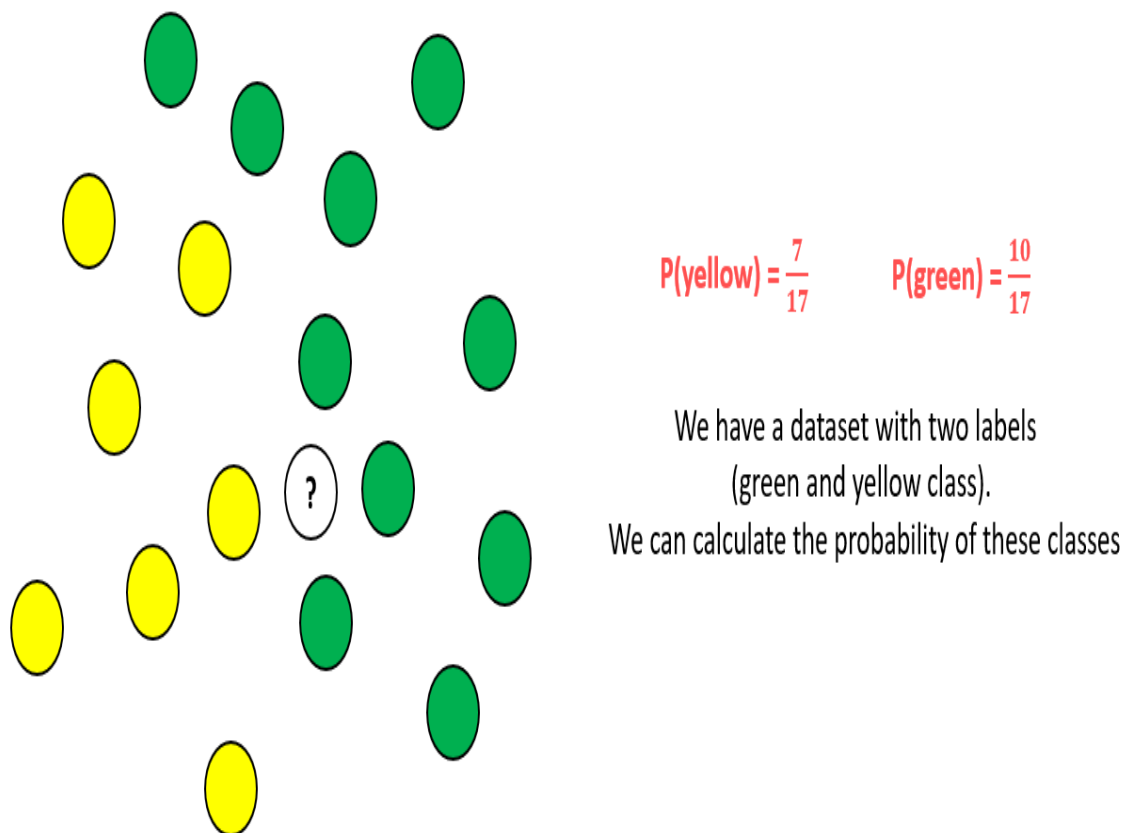


Fig3.1.1.2 Naive Bayes Classifier

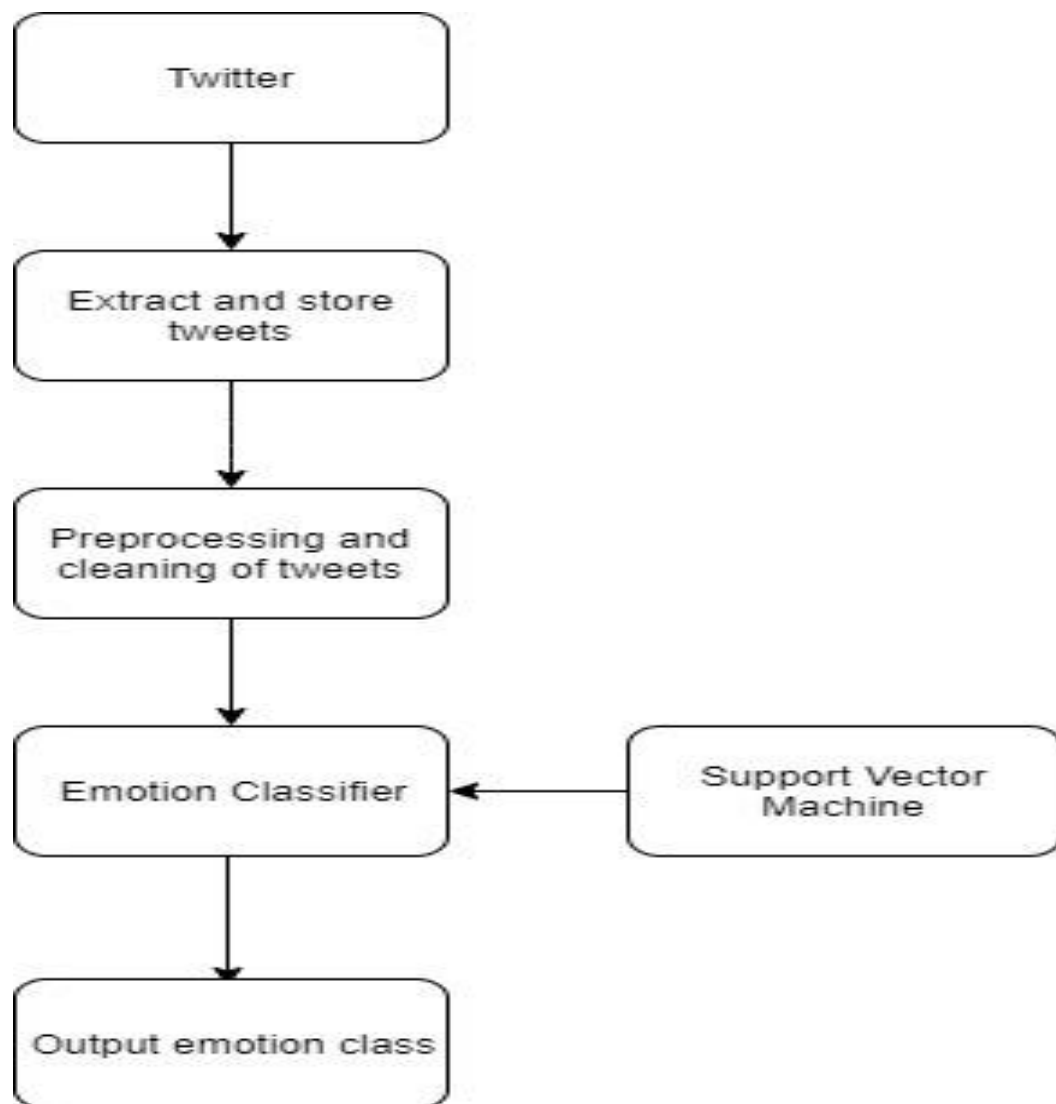


Fig 3.1.1.3 Architecture diagram for Emotion analysis using Support Vector Machines

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 labeled 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 (eg. 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 V. 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

$$f(x) = \text{sign}(w^T x + b) \quad (2)$$

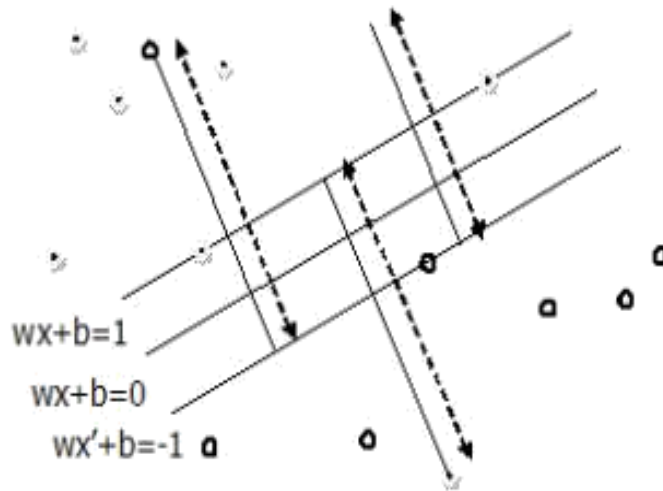


Fig 3.1.1.4 SVM Classifier(I)

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

$$Y_i(W^T X_i + b) \geq 1, \text{ for all } X_i \in P \cup N \quad (3)$$

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 (x_i, y_i) ,

$i = 1, 2, 3, \dots, l$ where $x_i \in R^n$, the class label of the i th pattern is denoted by

$y_i \in \{1, -1\}$. Nonlinearly separable problems are often solved by mapping the input data samples x_i to a higher dimensional feature space $\phi(x_i)$. The classical maximum margin SVM classifier aims to find a hyperplane of the form

$w^T \phi(x) + b = 0$, which separates patterns of the two classes. So far 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 are misclassified, we introduce a vector of slack variables $\Xi = (\xi_1, \dots, \xi_n)^T$

that measure the amount of violation of the constraints (3). The problem can then be written as

$$\text{Minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad (4)$$

w, b, ξ_i subject to the constraints

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \quad (5)$$

$$\xi_i \geq 0, \quad i = 1, 2, \dots, n$$

The solution to (4)-(5) yields the soft margin classifier, is termed because the distance or margin between the separating hyperplane

$w^T \phi(x) + b = 0$ is usually determined by considering the dual problem, which is given by

$$L(w, b, \alpha, \xi, \gamma) = \frac{1}{2} \|w\|^2 + \sum_{i=1}^n \alpha_i [y_i(w^T \phi(x_i) + b) - 1 + \xi_i] - \sum_{i=1}^n \gamma_i \xi_i + C \sum_{i=1}^n \xi_i$$

where $\Lambda = (\alpha_1, \dots, \alpha_n)^T$ and $\Gamma = (\gamma_1, \dots, \gamma_n)^T$, 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.

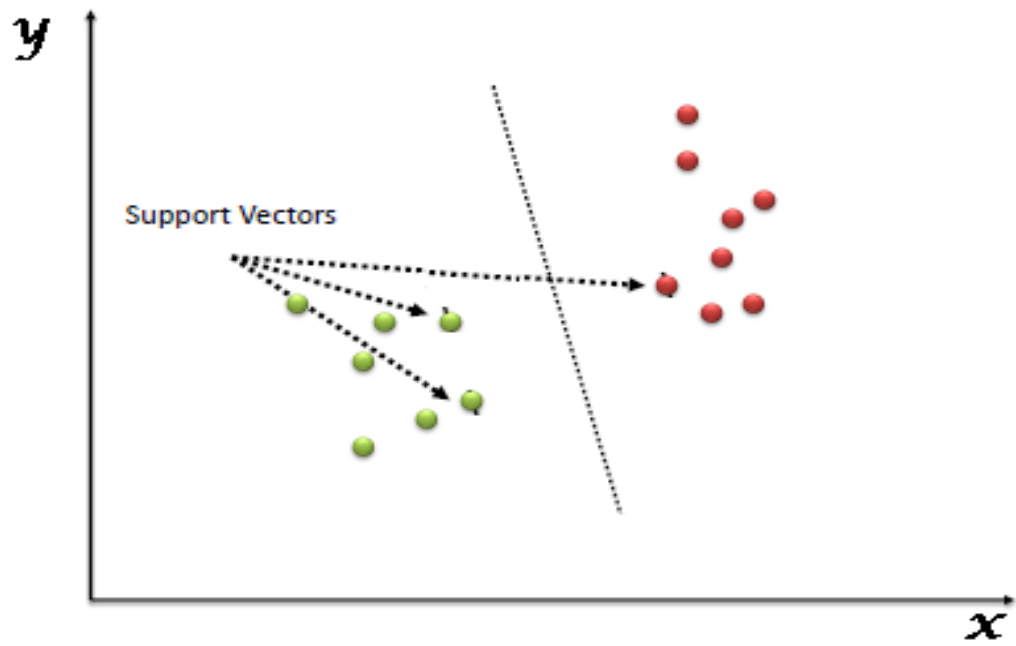


Fig 3.1.1.5 SVM Classifier(II)

4.1 STRUCTURE CHART

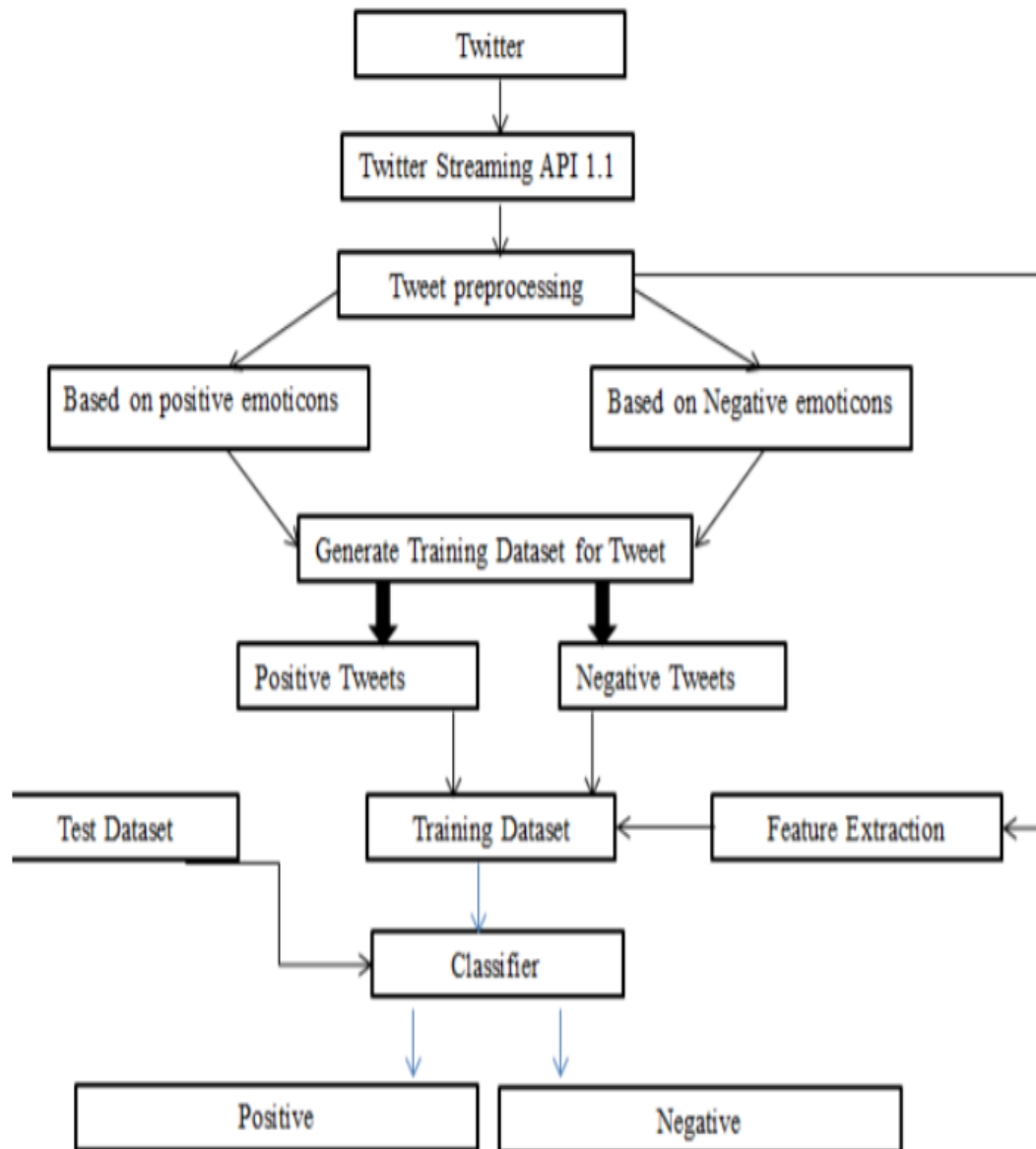


Fig 4.1.1 Structure Chart

4.2 UML DIAGRAMS

A **UML diagram** is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains **graphical elements** (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude **mixing** of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some **UML Tools** do restrict set of available graphical elements which could be used when working on specific type of diagram.

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

Structure diagrams show the **static structure** of the system and its parts on different abstraction and implementation **levels** and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**.

4.2.1 USE CASE DIAGRAM

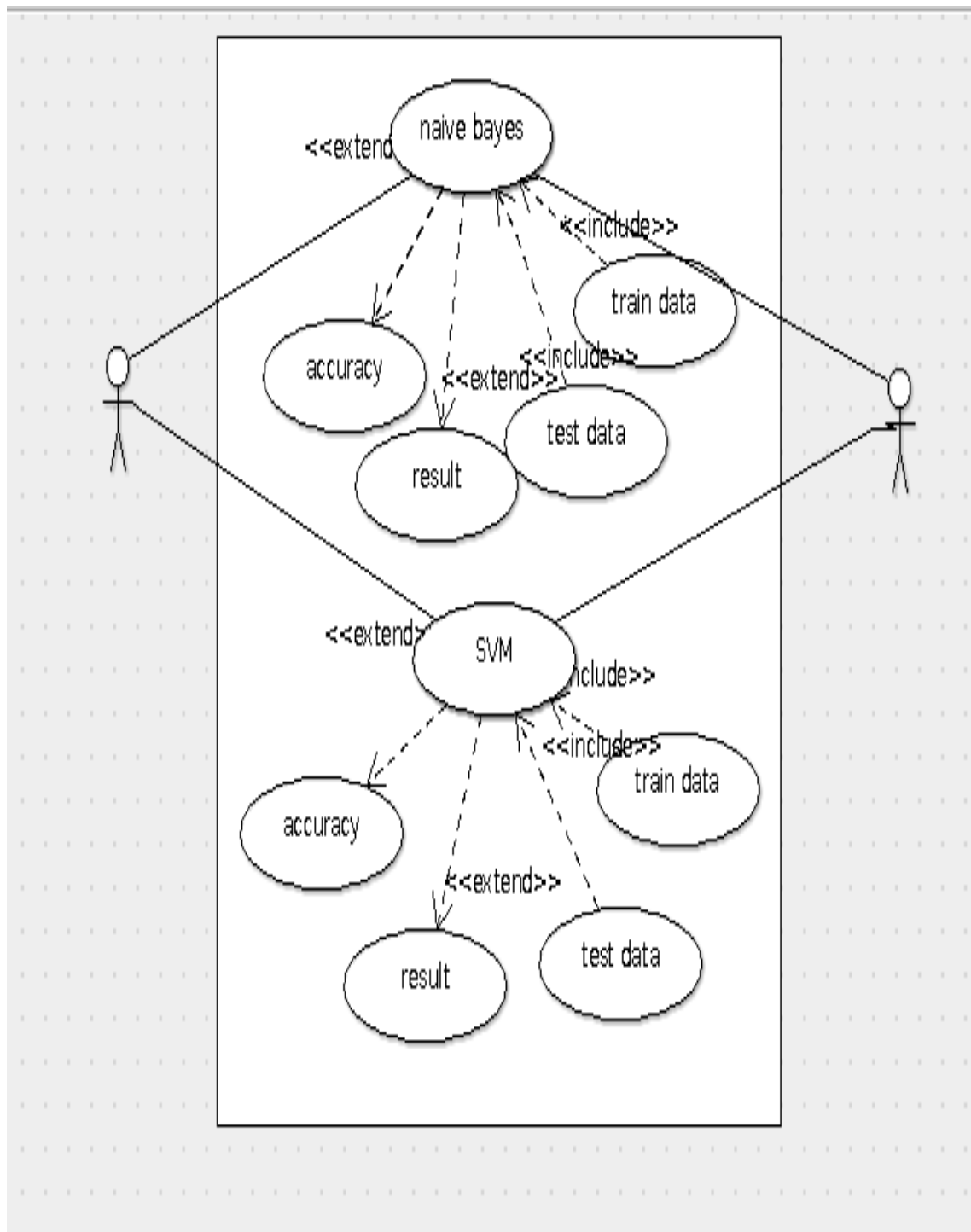


Fig 4.2.1.1 Use Case Diagram

4.2.2 CLASS DIAGRAM

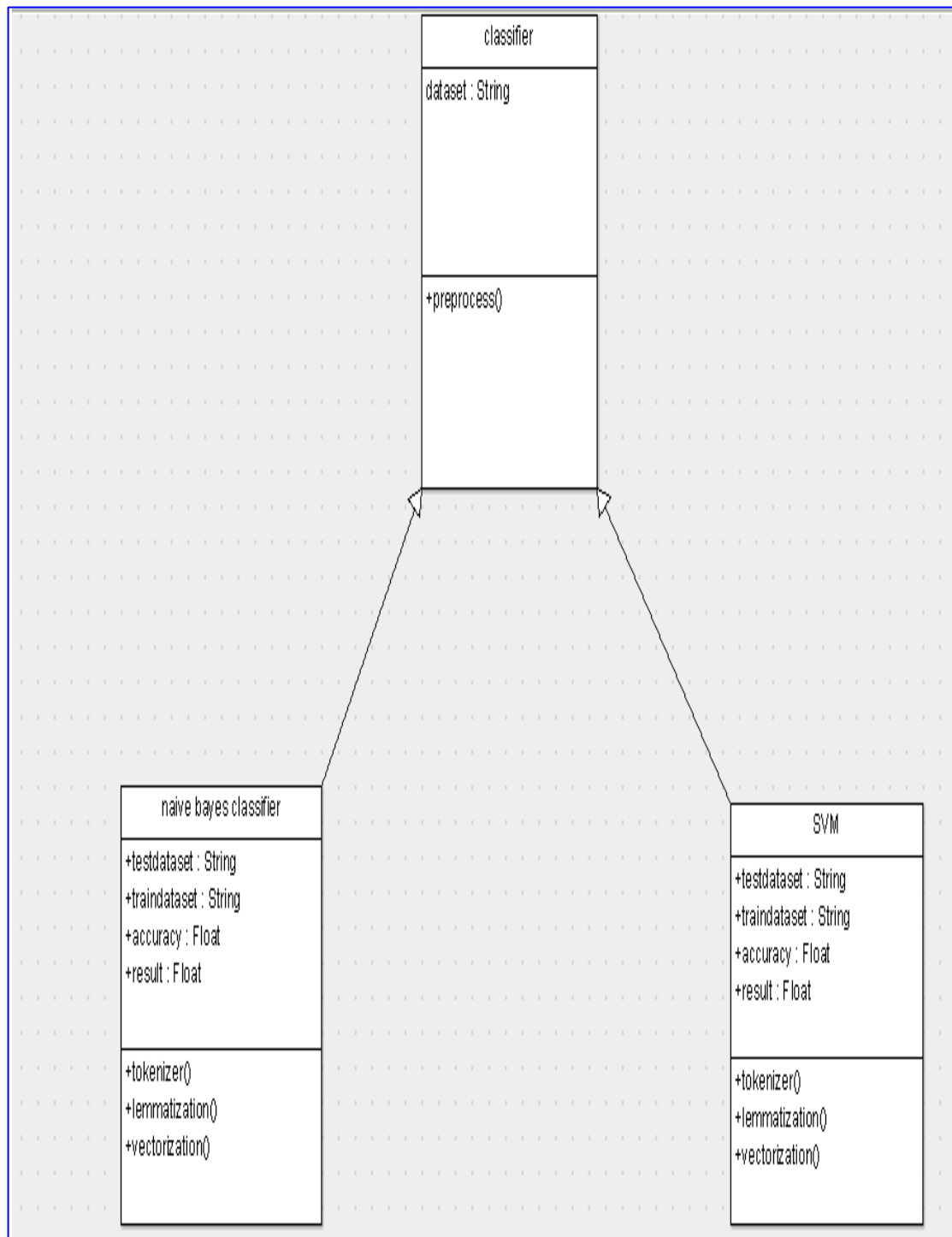


Fig 4.2.2.1 Class Diagram

4.2.3 SEQUENCE DIAGRAM

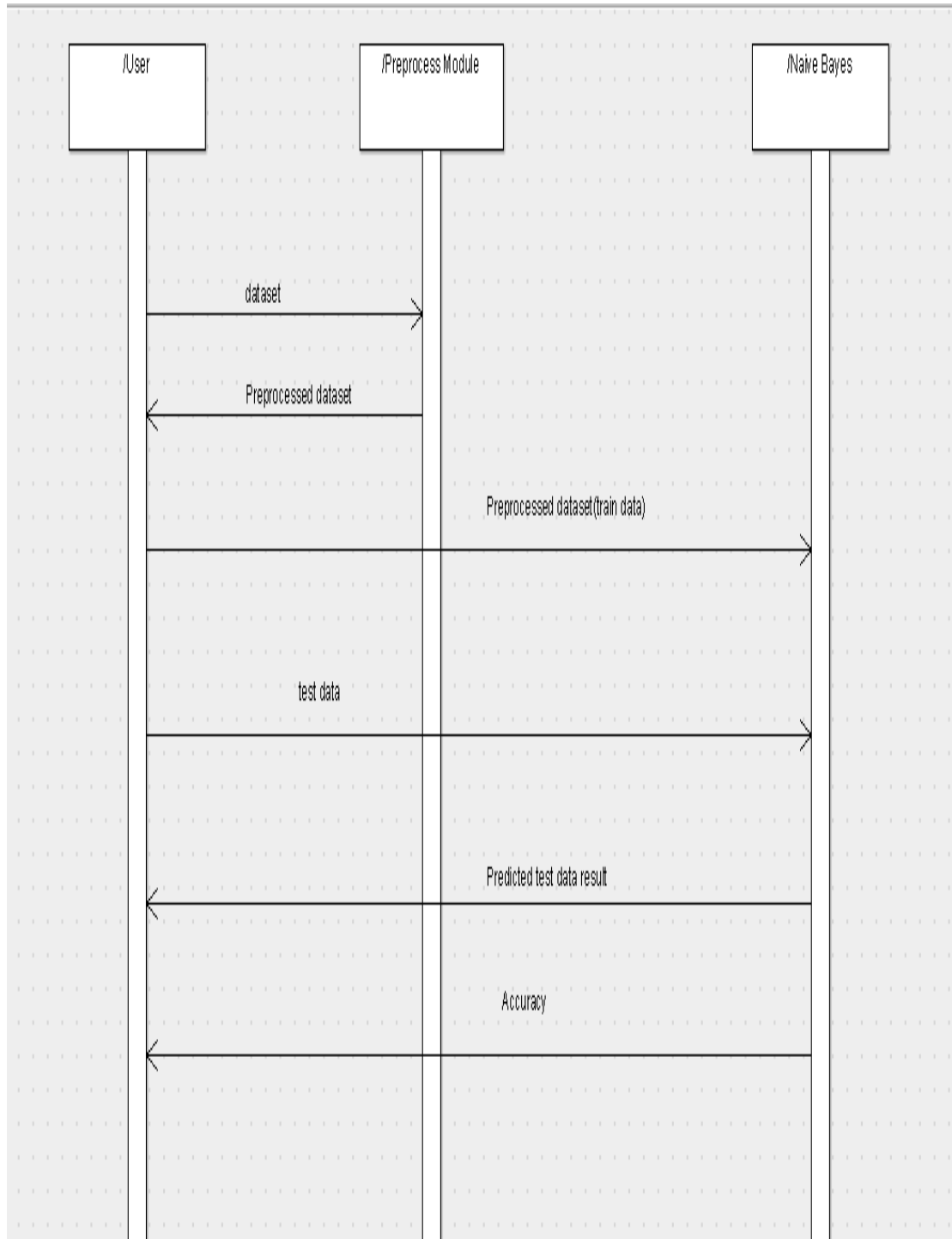


Fig 4.2.3.1 Sequence Diagram of Sentiment Analysis(I)

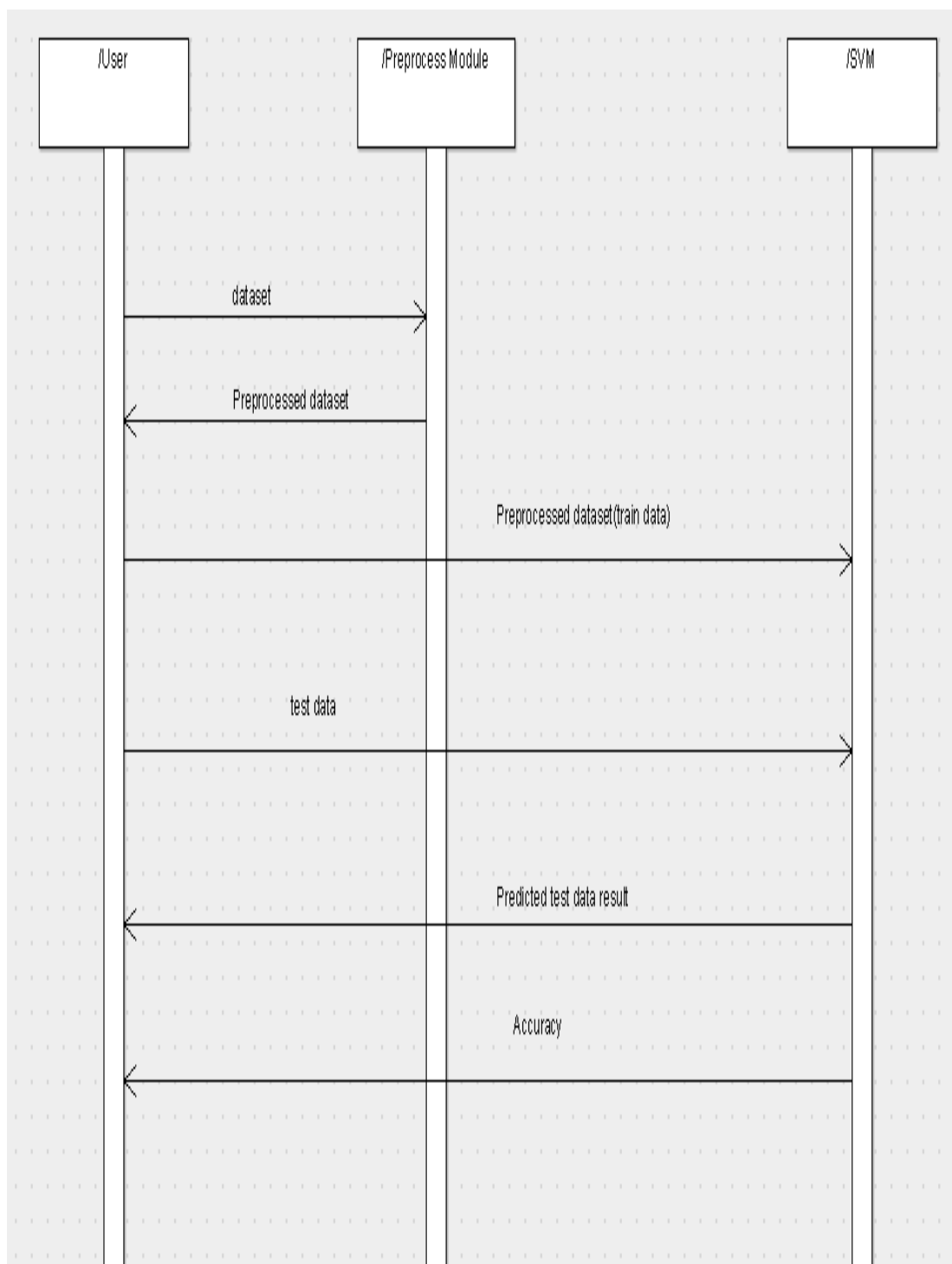


Fig 4.2.3.2 Sequence Diagram of Emotional Analysis(II)

4.2.4 ACTIVITY DIAGRAM

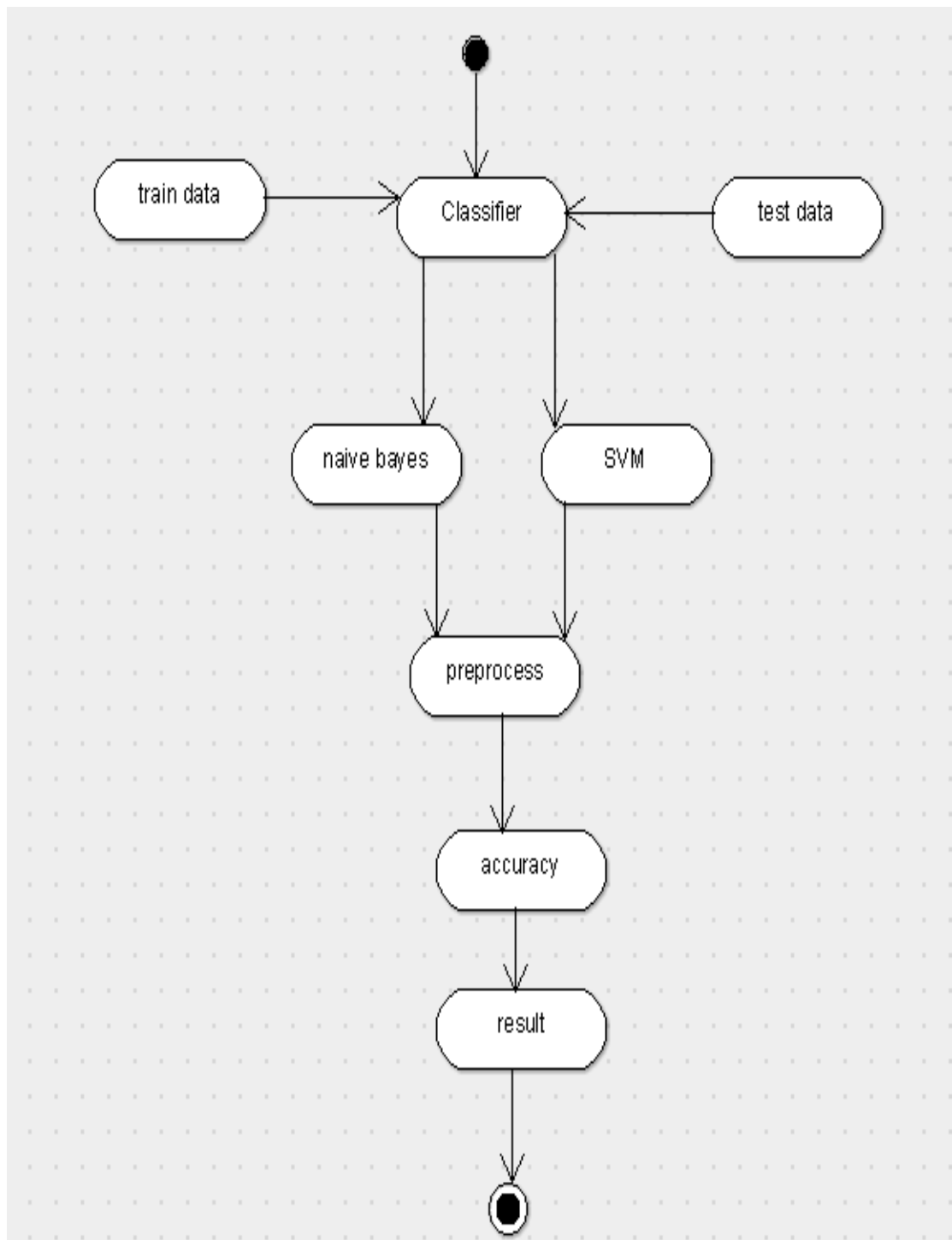


Fig 4.2.4.1 Activity Diagram

4.2.5 COLLABORATION DIAGRAM

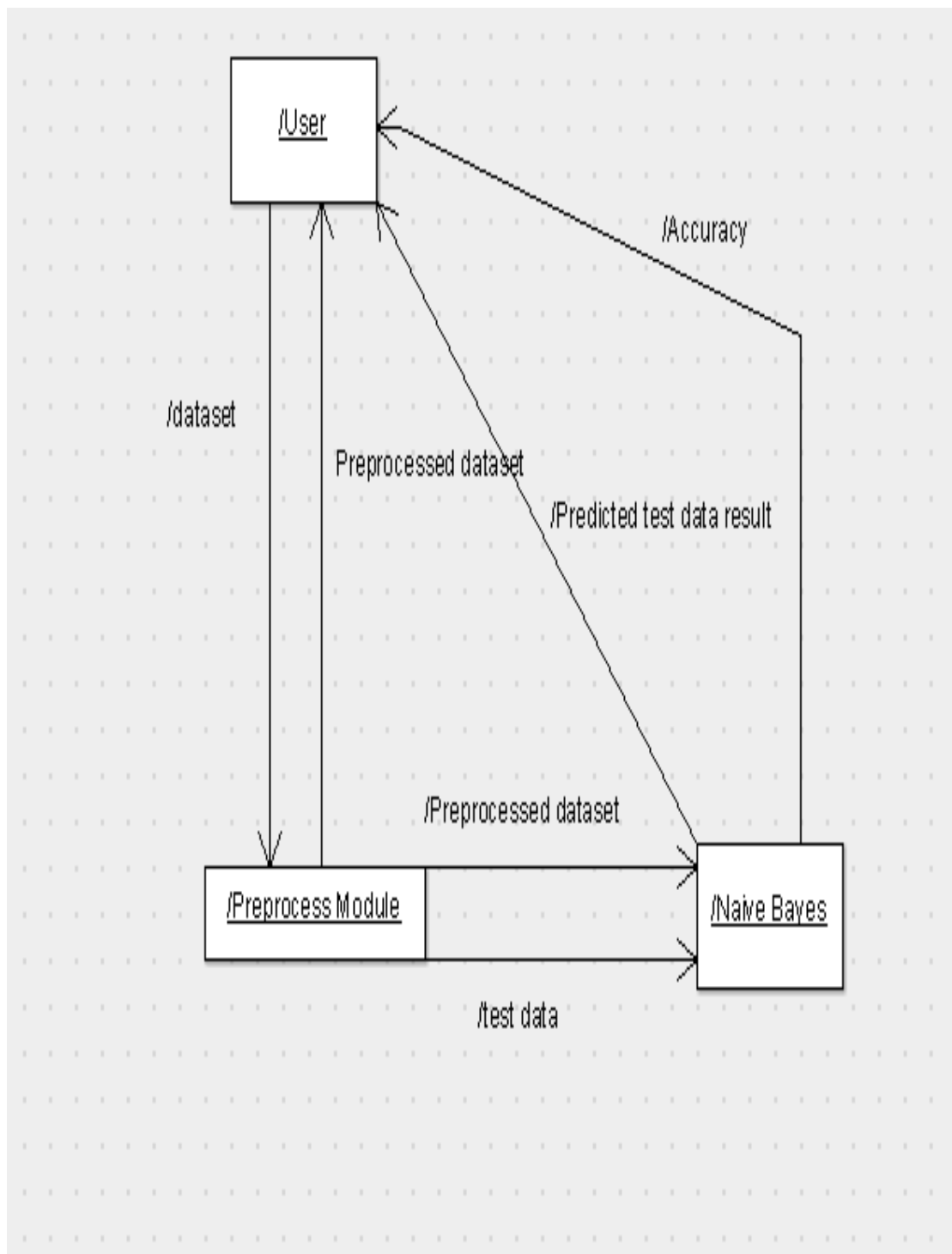


Fig 4.2.5.1 Collaboration Diagram

4.2.6 FLOW CHART DIAGRAM

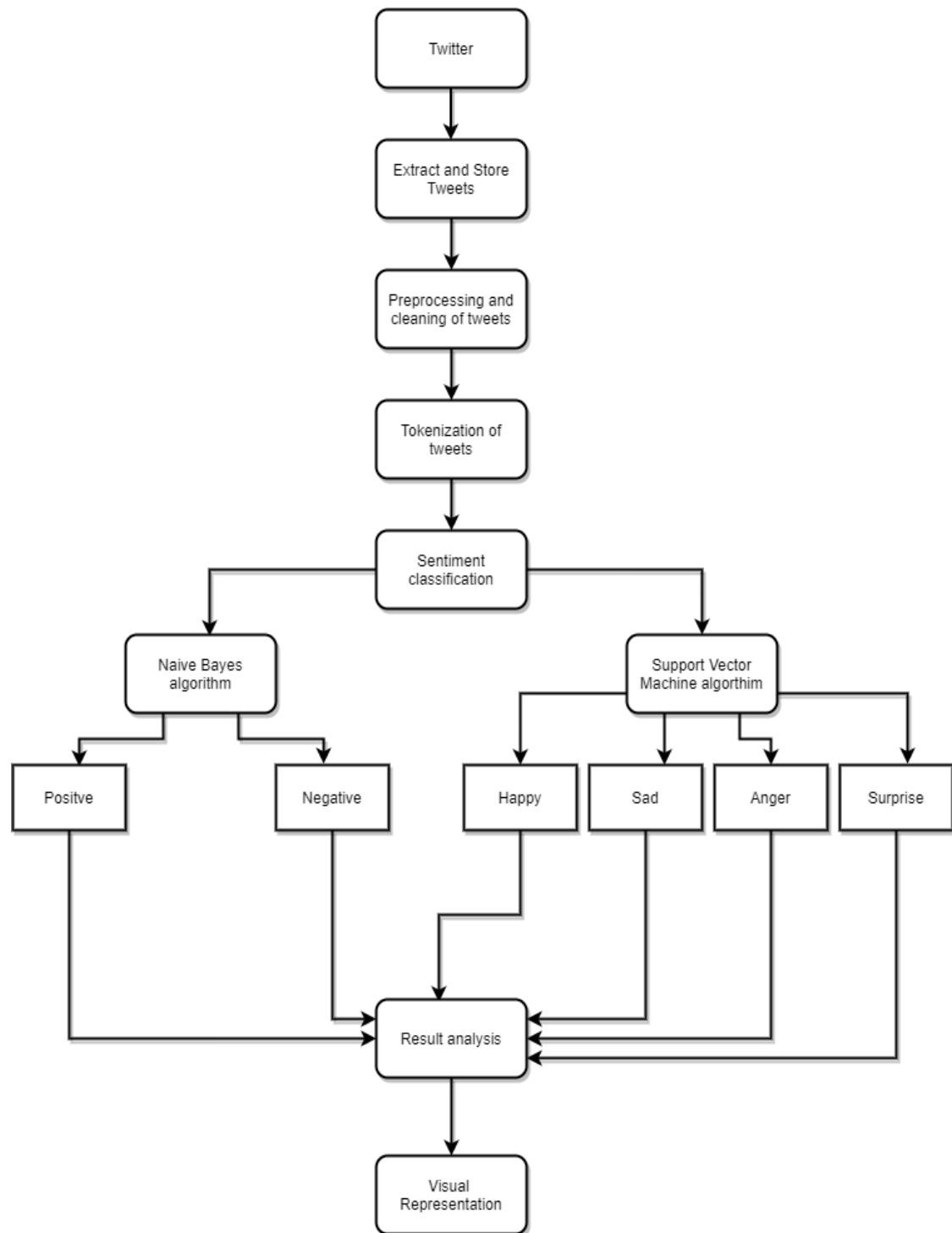


Fig 4.2.6.1 Flow Chart Diagram

4.2.7 COMPONENT DIAGRAM

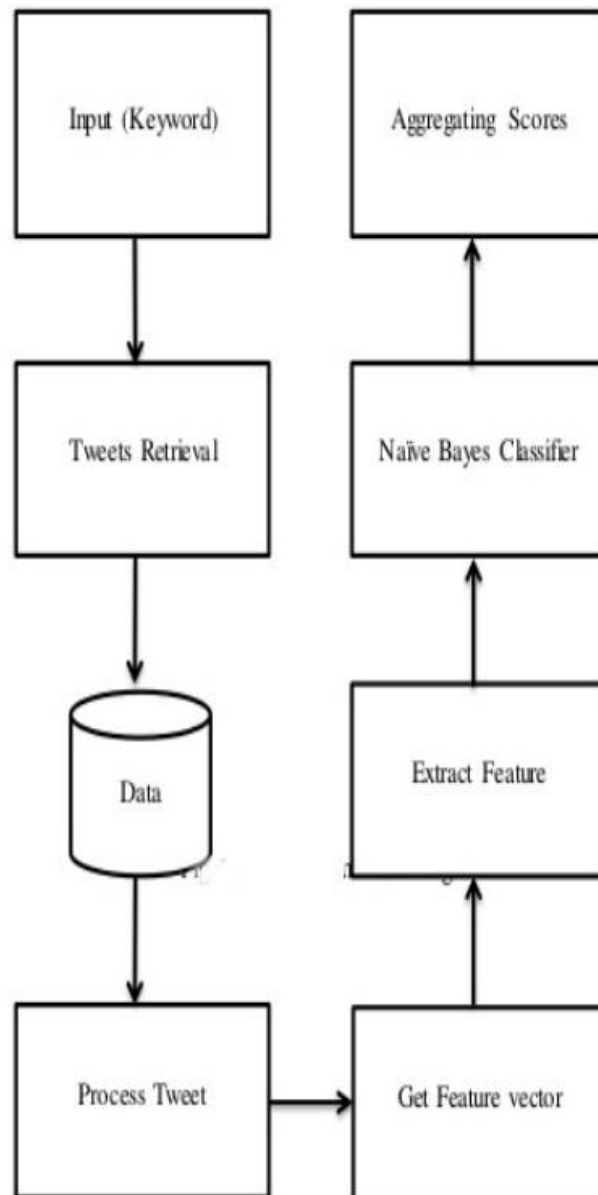


Fig 4.2.7.1 Component Diagram

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 REQUIRMENTS

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

1. Anaconda
2. Spyder
3. Jupiter Note Book
4. Nltk
5. Scikit-learn
6. Matplotlib
7. Tweepy
8. Pandas
9. Numpy
10. TextBlob
11. VaderSentiment
12. Csv

13.Regular Expressions)

14.Windows

5.5.2 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 atleast 2 cores of clock speeds > 1.5GHz

5.2 SAMPLE CODE

```
import tweepy

from tweepy import Stream

from tweepy import OAuthHandler

from tweepy.streaming import StreamListener

import json

import pandas as pd

import csv

import re #regular expression

#from textblob import TextBlob

import string

import preprocessor as p
```

```

import os

import nltk

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

ItemId=0

Sentiment=0

pos=0

neg=0

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

consumer_key = 'cksnY9Jp0jpry0WDtqT7kidyb'

consumer_secret =

'3R3UHvxHma9Q9DTQmZLrsCpGb9fgLuZZkqYK8BK2mKu8qo1iO9'

access_key= '1169646368903663617-VvzdXw2FtF2ZACfsc2Ww8bTDPpaeji'

access_secret = 'x1w3fVEb497b0Ftf2hyWOfngwB4YNsiFyg16nbYatlKPR'

auth=tweepy.OAuthHandler(consumer_key,consumer_secret)

auth.set_access_token(access_key,access_secret)

api=tweepy.API(auth)

analyzer=SentimentIntensityAnalyzer()

hacking_tweets=r"C:\Users\saille\OneDrive\Desktop\cyber.csv"

COLS = ['ItemId','Sentiment', 'SentimentText']

```

```

#set two date variables for date range

start_date = '2019-10-01'

end_date = '2019-10-31'

# Happy Emoticons

emoticons_happy = set([

':-)', ':)', ';)', ':o)', ':]', ':3', ':c)', ':>', '=]', '8)', '=)', ':}',

':^)', ':-D', ':D', '8-D', '8D', 'x-D', 'xD', 'X-D', 'XD', '=-D', '=D',

'=-3', '=3', ':-)))', ":'-)", ":')", ":'*", ":'^*", ">:P", ":-P", ':P', 'X-P',

'x-p', 'xp', 'XP', ":-p", ':p', '=p', ":-b", ':b', ">:)", ">:)", ">:-)",

'<3'

])

# Sad Emoticons

emoticons_sad = set([

':L', ":-/", ">:/", ':S', ">:[", ":'@", ":'-(, ":'[', ":'-||', '=L', '<',

':-[', ":'<', '=\\', '=/', ">:(, ":'(, ">.<", ":'-(, ":'(, ":'\\', ":'-c',

':c', ":'{, ">:\\, ":'(

])

#Emoji patterns

"""emoji_pattern = re.compile("[

u"\\U0001F600-\\U0001F64F" # emoticons

```



```

u"\U0001F300-\U0001F5FF" # symbols & pictographs

u"\U0001F680-\U0001F6FF" # transport & map symbols

u"\U0001F1E0-\U0001F1FF" # flags (iOS)

u"\U00002702-\U000027B0"

u"\U000024C2-\U0001F251"

"]+", flags=re.UNICODE)

'''

#combine sad and happy emoticons

emoticons = emoticons_happy.union(emoticons_sad)

#mrhod clean_tweets()

def clean_tweets(tweet):

    stop_words = set(stopwords.words('english'))

    word_tokens = word_tokenize(tweet)

    #after tweepy preprocessing the colon left remain after removing mentions

    #or RT sign in the beginning of the tweet

    tweet = re.sub(r':', '', tweet)

    #tweet = re.sub(r'Ä¶', '', tweet)

    #replace consecutive non-ASCII characters with a space

    tweet = re.sub(r'^[\x00-\x7F]+', '', tweet)

```

```
tweet=re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[*\(\), ]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', '', tweet)
```

```
tweet= re.sub(r'#([A-Za-z0-9_]+)', '', tweet)
```

```
#remove emojis from tweet
```

```
tweet = emoji_pattern.sub(r'', tweet)
```

```
#filter using NLTK library append it to a string
```

```
filtered_tweet = [w for w in word_tokens if not w in stop_words]
```

```
filtered_tweet = []
```

```
#looping through conditions
```

```
for w in word_tokens:
```

```
#check tokens against stop words , emoticons and punctuations
```

```
if w not in stop_words and w not in emoticons and w not in string.punctuation:
```

```
filtered_tweet.append(w)
```

```
return ' '.join(filtered_tweet)
```

```
#print(word_tokens)
```

```
#print(filtered_sentence)
```

```
def write_tweets(keyword, file,ItemId,pos,neg):
```

```
# If the file exists, then read the existing data from the CSV file.
```

```
if os.path.exists(file):
```

```

df = pd.read_csv(file, header=0)

else:

df = pd.DataFrame(columns=COLS)

#page attribute in tweepy.cursor and iteration

for page in tweepy.Cursor(api.search, q=keyword,

count=200, include_rts=False, since=start_date).pages(10):

for status in page:

new_entry = []

status = status._json

if status['lang'] != 'en':

continue

vs=analyzer.polarity_scores(status['text'])

if vs['compound']>=0.5:

Sentiment=1

pos=pos+1

else:

Sentiment=0

neg=neg+1

ItemId=ItemId+1

#new entry append

```

```

new_entry += [ItemId,Sentiment,status['text']]

#to append original author of the tweet

single_tweet_df = pd.DataFrame([new_entry], columns=COLS)

df = df.append(single_tweet_df, ignore_index=True)

csvFile = open(file, 'w' ,encoding='utf-8')

df.to_csv(csvFile, mode='w', columns=COLS, index=False, encoding="utf-8")

#declare keywords as a query for three categories

cyber_words="#cybertruck -filter:retweets"

#call main method passing keywords and file path

write_tweets(cyber_words,hacking_tweets,ItemId,pos,neg)

#C:\Users\saille\OneDrive\Desktop\cyber1.csv

import sys

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import re

import nltk

from nltk.tokenize import word_tokenize

from nltk.corpus import stopwords

train_data =pd.read_csv(r'C:\Users\saille\OneDrive\Desktop\cyber.csv')

```

```

rand_indexs = np.random.randint(1,len(train_data),50).tolist()

train_data["SentimentText"][rand_indexs]

tweets_text = train_data.SentimentText.str.cat()

emos = set(re.findall(r" ([xX;;][-']?.) ",tweets_text))

emos_count = []

for emo in emos:

    emos_count.append((tweets_text.count(emo), emo))

sorted(emos_count,reverse=True)

HAPPY_EMO = r" ([xX;;]-?[dD])|:-?[\)]|[:;][pP]) "

SAD_EMO = r" (:'?[/\(|) "

print("Happy emoticons:", set(re.findall(HAPPY_EMO, tweets_text)))

print("Sad emoticons:", set(re.findall(SAD_EMO, tweets_text)))

# Uncomment this line if you haven't downloaded punkt before

# or just run it as it is and uncomment it if you got an error.

#nltk.download('punkt')

def most_used_words(text):

    tokens = word_tokenize(text)

    frequency_dist = nltk.FreqDist(tokens)

    print("There are %d different words" % len(set(tokens)))

    return sorted(frequency_dist,key=frequency_dist.__getitem__, reverse=True)

```

```

most_used_words(train_data.SentimentText.str.cat()[:100])

# In[11]:

mw = most_used_words(train_data.SentimentText.str.cat())

most_words = []

for w in mw:

    if len(most_words) == 1000:

        break

    if w in stopwords.words("english"):

        continue

    else:

        most_words.append(w)

sorted(most_words)

# In[12]:

from nltk.stem.snowball import SnowballStemmer

from nltk.stem import WordNetLemmatizer

def stem_tokenize(text):

    stemmer = SnowballStemmer("english")

    stemmer = WordNetLemmatizer()

    return [stemmer.lemmatize(token) for token in word_tokenize(text)]

def lemmatize_tokenize(text):

```

```

lemmatizer = WordNetLemmatizer()

return [lemmatizer.lemmatize(token) for token in word_tokenize(text)]

# In[13]:

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.base import TransformerMixin, BaseEstimator

from sklearn.pipeline import Pipeline

class TextPreProc(BaseEstimator,TransformerMixin):

    def __init__(self, use_mention=False):

        self.use_mention = use_mention

    def fit(self, X, y=None):

        return self

    def transform(self, X, y=None):

        if self.use_mention:

            X = X.str.replace(r"@[a-zA-Z0-9_]* ", " @tags ")

        else:

            X = X.str.replace(r"@[a-zA-Z0-9_]* ", "")

            X = X.str.replace("#", "")

            X = X.str.replace(r"[-\.\n]", "")

            X = X.str.replace(r"&\w+;", "")

```

```

# Removing links

X = X.str.replace(r"https?://\S*", "")

# replace repeated letters with only two occurrences

# heeeelllloooo => heelloo

X = X.str.replace(r"(\1+)", r"\1")

# mark emoticons as happy or sad

X = X.str.replace(HAPPY_EMO, " happyemoticons ")

X = X.str.replace(SAD_EMO, " sademoticons ")

X = X.str.lower()

return X

# In[14]:

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import cross_val_score

from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import BernoulliNB, MultinomialNB

from sklearn import tree

sentiments = train_data['Sentiment']

tweets = train_data['SentimentText']

```



```

vectorizer = TfidfVectorizer(tokenizer=lemmatize_tokenize, ngram_range=(1,2))

pipeline = Pipeline([

('text_pre_processing', TextPreProc(use_mention=True)),

('vectorizer', vectorizer),

])

learn_data, test_data, sentiments_learning, sentiments_test = train_test_split(tweets,
sentiments, test_size=0.3)

learning_data = pipeline.fit_transform(learn_data)

lr = LogisticRegression()

bnb = BernoulliNB()

mnb = MultinomialNB()

clf = tree.DecisionTreeClassifier()

clf1 = RandomForestClassifier(n_estimators=10)

models = {

'logistic regression': lr,

'bernoulliNB': bnb,

'multinomialNB': mnb,

}

for model in models.keys():

```

```

scores = cross_val_score(models[model], learning_data, sentiments_learning,
                           scoring="f1", cv=10)

print("=== ", model, "===")

print("scores = ", scores)

print("mean = ", scores.mean())

print("variance = ", scores.var())

models[model].fit(learning_data, sentiments_learning)

print("score on the learning data (accuracy) = ",
      accuracy_score(models[model].predict(learning_data), sentiments_learning))

print("")

# In[9]:

from sklearn.model_selection import GridSearchCV

grid_search_pipeline = Pipeline([

('text_pre_processing', TextPreProc()),

('vectorizer', TfidfVectorizer()),

('model', MultinomialNB()),

])

params = [

{

'text_pre_processing__use_mention': [True, False],

```

```

'vectorizer__max_features': [1000, 2000, 5000, 10000, 20000, None],

'vectorizer__ngram_range': [(1,1), (1,2)],

},

]

grid_search = GridSearchCV(grid_search_pipeline, params, cv=5, scoring='f1')

grid_search.fit(learn_data, sentiments_learning)

print(grid_search.best_params_)

# In[9]:

mnb.fit(learning_data, sentiments_learning)

# In[15]:

testing_data = pipeline.transform(test_data)

mnb.score(testing_data, sentiments_test)

# In[12]:

# Data to plot

pos=124

neg=166

labels = 'Positive','Negative'

sizes = [pos,neg]

colors = ['gold', 'lightcoral']

explode = (0.1,0) # explode 1st slice

```

```

# Plot

plt.pie(sizes, explode=explode, labels=labels, colors=colors,

autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')

plt.show()

import pandas as pd

import numpy as np

import nltk

import re

import itertools

import time

import matplotlib.pyplot as plt

start_time = time.time()

import os

data = pd.read_csv(r'C:\Users\saille\OneDrive\Desktop\project\text_emotion.csv')

sizes1=[]

from nltk.stem.wordnet import WordNetLemmatizer

lem = WordNetLemmatizer()

def cleaning(text):

txt = str(text)

```

```
txt = re.sub(r"http\S+", "", txt)
```

```
if len(txt) == 0:
```

```
    return 'no text'
```

```
else:
```

```
    txt = txt.split()
```

```
    index = 0
```

```
    for j in range(len(txt)):
```

```
        if txt[j][0] == '@':
```

```
            index = j
```

```
    txt = np.delete(txt, index)
```

```
    if len(txt) == 0:
```

```
        return 'no text'
```

```
    else:
```

```
        words = txt[0]
```

```
        for k in range(len(txt)-1):
```

```
            words+= " " + txt[k+1]
```

```
        txt = words
```

```
        txt = re.sub(r'[\^\w]', ' ', txt)
```

```
    if len(txt) == 0:
```

```
        return 'no text'
```

```

else:

    txt = ".join(".join(s)[:2] for _, s in itertools.groupby(txt))

    txt = txt.replace("'", "")

    txt = nltk.tokenize.word_tokenize(txt)

    for j in range(len(txt)):

        txt[j] = lem.lemmatize(txt[j], "v")

    if len(txt) == 0:

        return 'no text'

    else:

        return txt

data['content'] = data['content'].map(lambda x: cleaning(x))

data = data.reset_index(drop=True)

for i in range(len(data)):

    words = data.content[i][0]

    for j in range(len(data.content[i])-1):

        words+= ' ' + data.content[i][j+1]

    data.content[i] = words

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics import classification_report

from sklearn import svm

```

```

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(data.content, data.sentiment,
test_size=0.25, random_state=0)

x_train = x_train.reset_index(drop = True)

x_test = x_test.reset_index(drop = True)

y_train = y_train.reset_index(drop = True)

y_test = y_test.reset_index(drop = True)

vectorizer = TfidfVectorizer(min_df=3, max_df=0.9)

train_vectors = vectorizer.fit_transform(x_train)

test_vectors = vectorizer.transform(x_test)

model = svm.SVC(kernel='linear')

model.fit(train_vectors, y_train)

predicted_sentiment = model.predict(test_vectors)

report=(classification_report(y_test, predicted_sentiment,output_dict=True))

df=pd.DataFrame(report).transpose()

df.to_csv('classification_report.csv',index=False)

sizes=df['support'].tolist()

for i in range(13):

    sizes1.append(int(i))

```

```

labels=['Anger','Boredom','Empty','Enthusiasm','Fun','Happiness','Hate','Love','Neutr
al','Relief','Sadness','Surprise','Worry']

colors=['Red','yellowgreen','lightcoral','orange','gold','purple','black','pink','brown','gr
een','blue','maroon','bluegreen']

plt.pie(sizes1, explode=None, labels=labels, colors=None,

autopct='%1.1f%%', shadow=True, startangle=None)

plt.axis('equal')

plt.show()

predicted_sentiments = []

for s in range(len(predicted_sentiment)):

predicted_sentiments.append(predicted_sentiment[s])

prediction_df=pd.DataFrame({'Content':x_test,

'Emotion_predicted':predicted_sentiment, 'Emotion_actual': y_test})

prediction_df.to_csv('emotion_recognizer_svm.csv', index = False)

elapsed_time = time.time() - start_time

print ("processing time:", elapsed_time, "seconds")

```


5.3 INPUT AND OUTPUT

5.3.1 INPUT

ItemId	Sentiment	SentimentText
1	0	@elonmusk The generation of graphics before #cybertruck
2	1	I received an awesome surprise #Cybertruck gift package today for joining the pre-order/reservation club!! Thanksâ€¦! https://t.co/dVo8YbFoQs
3	0	@Tesla did you forget about the most comfortable driving position that truck owners will want? Yup. #cybertruck https://t.co/creyXJ9EoQ
4	0	@elonmusk I'm very curious as to how the #cybertruck or any electric vehicles will do over a long term with daily mâ€™! https://t.co/7NEB2W8V57
5	0	The New Tesla Cybertruck is Dang Easy to Model in Blender https://t.co/jJoZSRQj7 @Tesla #Cybertruck #Tesla #futuretechnology

Fig 5.3.1.1 Train dataset for sentiment analysis

tweet_id	sentiment	author	content
1.957E+09	empty	xoshayzers	@tiffanylue i know i was listenin to bad habit earlier and i started freakin at his part =[
1.957E+09	sadness	wannamam	Layin n bed with a headache ughhhh...waitin on your call...
1.957E+09	sadness	coolfunky	Funeral ceremony...gloomy friday...
1.957E+09	enthusiasm	czareaquinc	wants to hang out with friends SOON!
1.957E+09	neutral	xkilljoyx	@dannycastillo We want to trade with someone who has Houston tickets, but no one will.
1.957E+09	worry	xxxPEACHE!	Re-pinging @ghostridah14: why didn't you go to prom? BC my bf didn't like my friends
1.957E+09	sadness	ShansBee	I should be sleep, but im not! thinking about an old friend who I want. but he's married now. damn, & he wants me 2! scandalous!

Fig 5.3.1.2 Train dataset for emotional analysis

5.3.2 OUTPUT

Below are the results for sentiment and emotional analysis represented as a pie-chart for users using matplotlib.

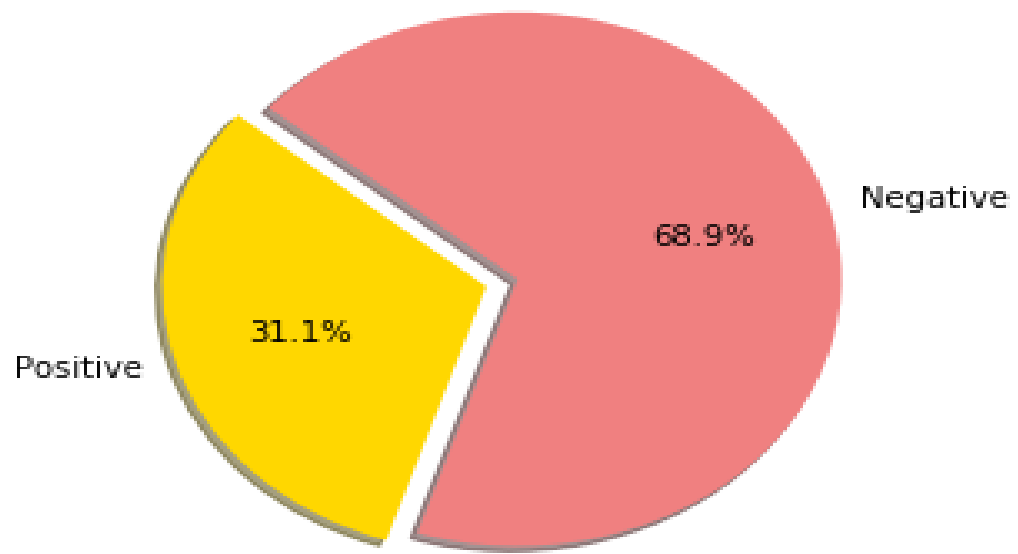


Fig 5.3.2.1 Result for Sentiment Analysis of Twitter Data

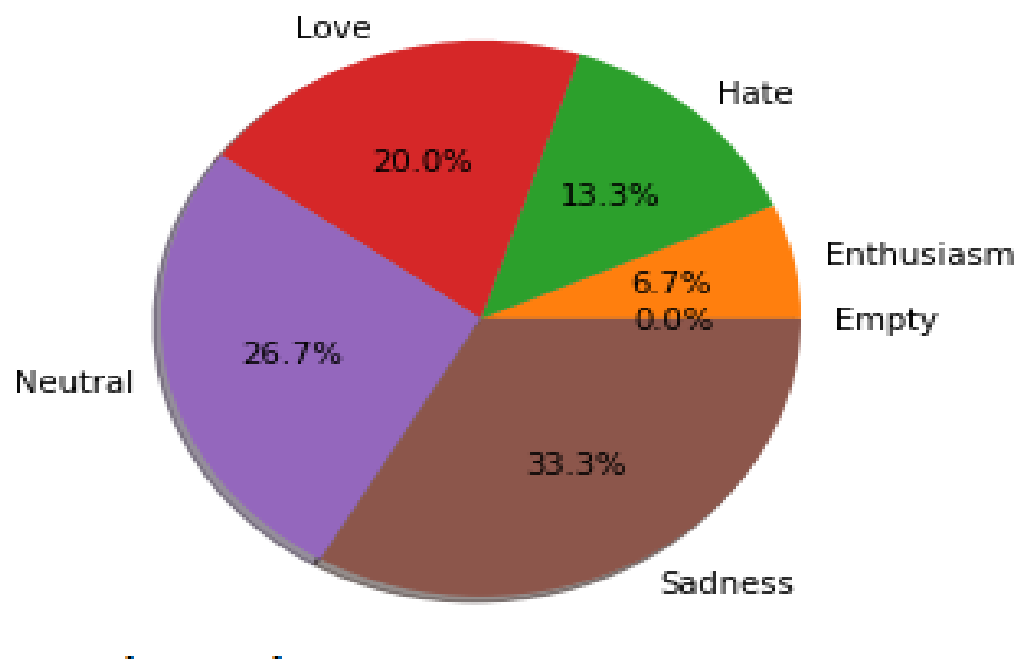


Fig5.3.2.2 Result for Emotional Analysis of Twitter Data

5.4 EXPERIMENT RESULTS AND ANALYSIS

ACCURACY COMPARISON:

```
=== logistic regression ===
scores = [0.80959922 0.81016909 0.80595209 0.81198447 0.80780817 0.81363307
0.81083696 0.81422158 0.81576402 0.81074569]
mean = 0.8110714362289213
variance = 7.933866749593147e-06
score on the learning data (accuracy) = 0.8718567836324151

=== bernoulliNB ===
scores = [0.79016702 0.78374591 0.78744018 0.78276063 0.7815876 0.78967136
0.79251899 0.79093912 0.7961597 0.79197936]
mean = 0.7886969879731057
variance = 2.0108782575180654e-05
score on the learning data (accuracy) = 0.9029031889358784

=== multinomialNB ===
scores = [0.80779944 0.80702918 0.80614631 0.80742148 0.80748963 0.80853918
0.80896121 0.81008703 0.8120753 0.81028724]
mean = 0.8085836007084021
variance = 2.901807591884587e-06
score on the learning data (accuracy) = 0.899074179906275
```

Fig 5.4.1 Scores generated using different algorithms at test-split=0.8

```
Out[71]: Text(0.5, 0, 'test-size percent')
```

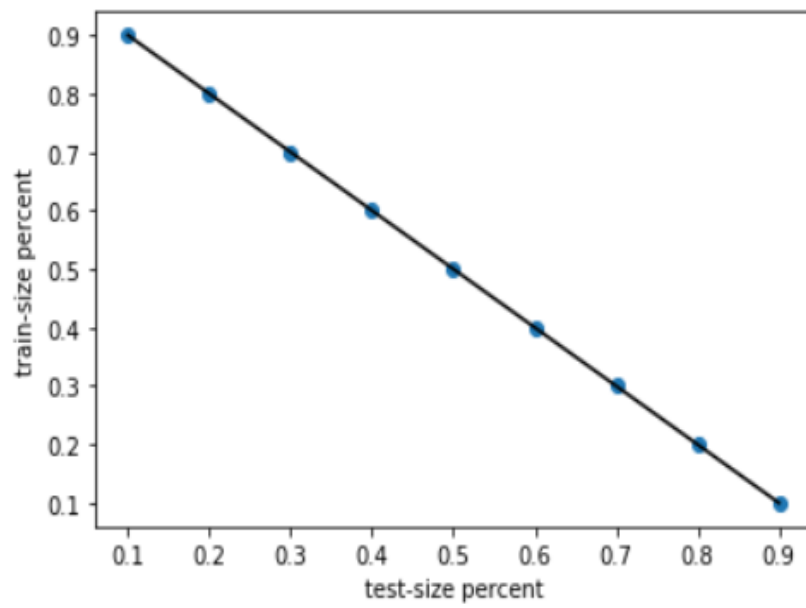


Fig 5.4.2 Train data and Test data(Their values affect final scores Fig 5.4.3)

```
Out[70]: Text(0.5, 0, 'test_split')
```

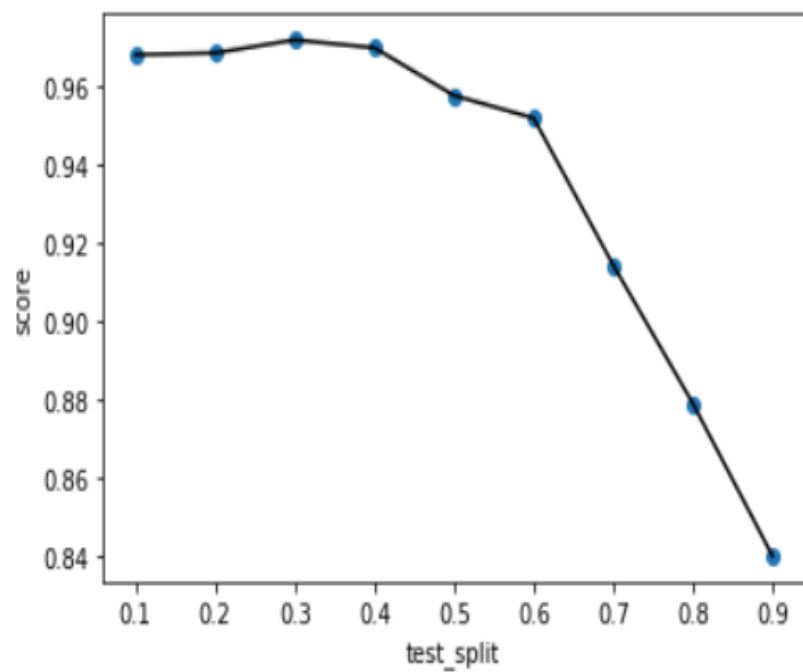


Fig 5.4.3 Scores based on test_split

Thus it can be inferred from these figures that test-split (which means selecting some percent as test data and the rest as train data) affects the final score of our classifier. So choosing an optimal value is a necessity in our case test-split=0.3 ended up with the maximum score.

Here in sentiment analysis we have compared accuracy of many other algorithms and it turns out to be that Naive Bayes outperformed all the other algorithms with an accuracy of 96% at test-split=0.3. Thence Naive Bayes classifier really predicts the sentiment out of tweets very well by mapping the insights instead. In case of emotional analysis we used Linear SVC to categorize the class of emotions to which a tweet belongs to. SVM's perform really well with multi-class classification problems by means of constructing hyper planes.

6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

We furnished results for Sentiment and Emotional Analysis on twitter data . On applying Logistic regression, Bernouille Naive Bayes and Multinomial Naive Bayes for sentiment analysis MultinomialNaive 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 .

6.2 FUTURE WORK

In future work , we aim to handle emoticons , 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. Mitchell

[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>