1.INTRODUCTION

Cyberbullying is bullying that takes place over digital devices like cell phones, computers, and tablets. Cyberbullying can happen through SMS, Text, and apps, or online in social media, forums, or gaming where people can view, participate in, or share content. Cyberbullying includes sending, posting, or sharing negative, harmful or false content about someone else. It can include sharing personal or private information about someone else causing embarrassment or humiliation. Some cyberbullying crosses the line into unlawful or criminal behavior. Cyberbullying or bullying of any type is against the law. It can have horrible outcomes that will hurt others and sometimes it can lead to crimes, revenge, murders or even cause deaths to innocents that had nothing to do with it in the first place. As a result of the invention of social networks friendships, relationships and social communications have all gone to a new level with new definitions. One may have hundreds of friends without even seeing their faces. Meanwhile, alongside this transition there is increasing evidence that online social applications have been used by children and adolescents for bullying. State-of-the-art studies in cyberbullying detection have mainly focused on the content of the conversations while largely ignoring the users involved in cyberbullying. We propose that incorporation of the users' information, their characteristics, and post-harassing behaviour, for instance, posting a new status in another social network as a reaction to their bullying experience, will improve the accuracy of cyberbullying detection. Cross-system analyses of the users' behaviour -monitoring their reactions in different online environments -can facilitate this process and provide information that could lead to more accurate detection of cyberbullying.

1.1 MOTIVATION

Features of information acquired from text, user demography, and social network profile features are often used in cyberbullying detection. Since the text content is the most reliable, our work here focuses on text-based cyberbullying detection. Identification of bullying& non-bullying text.

After identification of bullying and non-bullying text we apply NaiveBayesclassifer on it and then we find accuracy.

1.2 PROBLEM DEFINITION

Cyber-bullying is defined as an aggressive, intentional act made by a group or an individual using electronic forms of contact repeatedly and over time against a victim who cannot easily defend him or herself. Cyber-bullying refers to bullying and harassment of others by means of new electronic technologies, primarily mobile phones and the Internet. Currently cyber-bullying has received significant media attention as certain cases have resulted in civil and criminal law suits filed against a perpetrator and/or a school.

1.3 OBJECTIVE OF PROJECT

The main aim of this study was to determine the prevalence of adolescents and adults engaged in cyberbullying, to recognize and respond early and effectively to behaviors that can lead to bullying and to learn about new, effective strategies for controlling bullying and cyberbullying. To be able to empathize with the targets of cyberbullying.

1.4 LIMITATIONS OF PROJECT

- This project is supported only for text messages.
- Multimedia content like audio , video and images cannot be supported.

1.5 ORGANIZATION OF DOCUMENTATION

In this documentation,we first gave a brief description on the existing system and proposed system. Later we analyzed the requirements of the system and in that both software and hardware requirements. We designed UML diagrams for project. Later we implement the modules in this project.

2.LITERATURE SURVEY

2.1 INTRODUCTION

The current scope of the project is limited to identification of bullying message detection . Cyber Bullying detection can be formulated as supervised learning problem. A classifier is first trained on a corpus labeled by humans, and the learned classifier[6] is used to recognize a bullying message. Since the text content is most reliable, our work here focuses on text-based Cyberbullying detection.

Cyberbullying can be defined as aggressive, intentional actions performed by an individual or a group of people via digital communication methods such as sending messages and posting comments against a victim. Different from traditional bullying that usually occurs at school during face-to-face communication, cyberbullying on social media can take place any where at any time. For bullies, they are free to hurt their peers feelings because they do not need to face someone and can hide behind the Internet. For victims, they are easily exposed to harassment since all of us, especially youth, are constantly connected to Internet or social media. As reported in, cyberbullying victimization rate changes from 10 to 40 percent. In the United States, approximately 43 percent of teenagers were ever bullied on social media[6]. The same as traditional bullying, cyberbullying has negative, insidious and sweeping impacts on children. The outcomes for victims under cyberbullying may even be tragic such as the occurrence of self-injurious behavior or suicides.

One way to address the cyberbullying problem is to automatically detect and promptly report bullying message so that proper measures can be taken to prevent possible tragedies. Previous works on computational studies of bullying have shown that natural language processing(NLP) is powerful tools to study bullying. Cyberbullying detection can de formulated as a supervised learning problem. A classifier is first trained on a cyberbullying corpus labeled by humans, and the learned classifier is then used to recognize a bullying message. Three kinds of information including text, user demography, and social network features are often used in cyberbullying detection. Since the text content is the most reliable, our work here focuses on text-based cyberbullying detection.

2.2 EXISTING SYSTEM

The following features are previously defined to detect bullying messages:

- Combination of BoW features, sentiment feature and contextual features .
- Scaling operation for social network features .
- NavieBayes classifier is used in our current project.

2.3 PROPOSED SYSTEM

Detection of bullying message is taken place with the help of NaiveBayes classifier as follows:

- Polarity value of messages is calculated with the help of algorithm.
- Thus, we can find positive and negative values of a message.
- Polarity value ranges between (-1.0, 1.0) per word.
- If message having more negative polarity value then we can say that, it is a bullying message.

Naive Bayes Classifier

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features. For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other

words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes[5] classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

Formula:

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$

Where:

- $P(A|B)$ is the posterior probability of class(c, target)given predictor(x,attributes).
- $P(A)$ is the prior probability of class.
- $P(B/A)$ is the likelihood which is the probability of predictor given class.
- $P(B)$ is the prior probability of predictor.

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular

feature does not affect the other. Hence it is called naive.

Sentiment Analysis

Sentiment analysis[4] is basically the process of determining the attitude or the emotion of the writer, i.e., whether it is positive or negative or neutral.

Polarity is float which lies in the range of [-1, 1] where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0, 1].

Lemmatization

Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization[2] is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors.

For example, lemmatization would correctly identify the base form of 'caring' to 'care', whereas, stemming would cut off the 'ing' part and convert it to car.

'Caring' -> Lemmatization -> 'Care'

'Caring' -> Stemming -> 'Car'

Also, sometimes, the same word can have multiple different 'lemma's. So, based on the context it's used, you should identify the 'part-of-speech' (POS) tag for the word in that specific context and extract the appropriate lemma.

Wordnet Lemmatizer with NLTK

Wordnet is a large, freely and publicly available lexical database for the English language aiming to establish structured semantic relationships between words. It offers lemmatization capabilities as well and is one of the earliest and most commonly used lemmatizers.

Wordnet Lemmatizer with appropriate POS tag

It may not be possible manually provide the corrent POS tag for every word for large texts. So, instead, we will find out the correct POS tag for each word, map it to the right input character that the WordnetLemmatizer accepts and pass it as the second argument to lemmatize().

So how to get the POS tag for a given word?

In nltk, it is available through the nltk.pos_tag() method. It accepts only a list (list of words), even if its a single word.

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meaning to one word.

Text preprocessing includes both Stemming as well as Lemmatization. Many times people find these two terms confusing. Some treat these two as same. Actually, lemmatization is preferred over Stemming because lemmatization does morphological analysis of the words.

Applications of lemmatization

- Used in comprehensive retrieval systems like search engines.
- Used in compact indexing

Examples of lemmatization:

->corpora : corpus

->better : good

2.4 CONCLUSION

As cyberbullying is a major issue in today's world we are focused on detecting bullying messages in our project. Cyberbullying in social networks is done in form of text messages, posts, comments, pictures. But in our project we focus on cyberbullying detection in text messages. Since the text is more reliable than other formats so we focus on text based form.

3. ANALYSIS

3.1 INTRODUCTION

- Cyberbullying takes place in various social media platforms like facebook, twitter, instagram.
- Here in our project we are focusing on twitter dataset which was taken from github.
- We have special characters like emojis in twitter dataset.
- As we are working on text messages only, we will be removing those special characters in our dataset.
- After removing those special characters we have used wordnet lemmatizer.
- Before applying wordnet lemmatizer we need tokenize the words.
- Lemmatization is the process of converting a word to its base form.
- After lemmatization we'll get one dataset.
- On that dataset we will identify bullying and non bullying messages.
- We will add the bullying messages in one separate file.
- Now Naive Bayes algorithm is applied.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

Requirements analysis is the process of analyzing the information needs of the end users, the organizational environment and any systems presently being used thereby developing the functional requirements of a system that can meet the needs of the users. Also, the requirements should be recorded in a document, email, user interface. The requirements documentation should be referred to throughout the rest of the system development process to ensure the developing project aligns with the needs and requirements. Here we are using Anaconda Navigator for our project.

3.2.1 User Requirements

The First step is to identify a need for the new system. This will include determine whether a business problem or opportunity exists, conducting a feasibility study to

determine if the proposed solution is cost effective, and developing a project plan. This process may involve end users who come up with an idea for improving their work.

3.2.2 Functional Requirements

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). How a system implements functional requirements is detailed in the system design. In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

FR1: Application will provide detection of bullying messages.

DESC: This application is applied on text data which will help in detecting message Bullying or Non-bullying.

FR2: This system will display all types of messages i.e. positive, negative, neutral.

POSITIVE:

DESC: After system is processed we get all positive messages.

NEGATIVE:

DESC: After system is processed we get all negative messages.

NETURAL:

DESC: After system is processed we get all neutral messages.

3.2.3 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions In general , functional requirements define what a system is supposed to do whereas non-functional requirements define how a system is supposed to be. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements," and "non-behavioral requirements.

NFR1: Data should be in text format.

DESC: Given input data have to be in text format for usage of this applicant.

3.2.4 Minimum Hardware Requirements

- RAM : 1 GB
- HDD : 80 GB

3.2.5 Minimum Software Requirements

- Language: Python 3
- Tools: Flask
- Libraries: TextBlob, NLTK
- Packages: NUMPY, PANDAS
- OS-Windows

3.3 Python Programming Language

Python is a general purpose programming language. Hence you can use python for developing both desktop and web applications. Also, you can use python for developing complex scientific and numeric applications, Python is designed features to facilitate data analysis and visualization. Pythonis a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is interpreted**: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it.

- **Python is Interactive**: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, UNIX shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Features of Python

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintain.
- A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more

efficient.

- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX.
- Scalable: Python provides a better structure and support for large programs than shell.

TextBlob

TextBlob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as parts-of-speech tagging, noun phrase extraction, sentiment analysis, tokenization etc.

TextBlob aims to provide access to these operations through a familiar interface. We can create TextBlob objects as if they were Python strings that learned how to do Natural Language Processing.

Text Blob stands on the shoulders of giants like NLTK and Pattern, provides text mining, text analysis and text processing modules for python developers. TextBlob reuses the NLTK corpora and if you installed the NLTK before, it seems simple. TextBlob will use your local version instead of the bundled version.

The simplest way to install TextBlob is by PyPI:

$ pip install –U textblob

$ python –m textblob.download_corpora

4.DESIGN

After the requirements have been determined, the necessary specification for the hardware, software, people, data resources, and the information products that will satisfy the functional requirements of the proposed system can be determined. The design will serve as a blue print for the systems and helps to detect these problems before these errors or problems are built into the final system.

4.1 INTRODUCTION

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is very important parts of developing object oriented software and the software development process. The UML uses most graphical notations to express the design of software projects.

The goals of UML are:

- To model systems using object-oriented concepts.
- To establish an explicit coupling between conceptual as well as executable.
- To address the issue of scale inherent in complex, mission critical system.
- To create a modeling language usable by both humans and machines.

Basic Building Blocks of UML

- **Use case Diagram**: shows a set of use cases, and how actors can use them.
- **Class Diagram**: describes the structure of the system, divided in classes with different connections and relationships.
- **Sequence Diagram**: shows the interaction between a set of objects, through the messages that may be dispatched between them.
- **State Chart Diagram**: state machines, consisting of states, transitions, events and activities.

- **Activity Diagram**: shows the flow through a program from an defined start point to an end point.
- **Object Diagram**: A set of objects and their relationships, this is an snapshot of instances of the things found in the class objects.

4.2 UML DIAGRAMS

4.2.1 Class Diagram

The classes in a Class diagram represent both the main objects, interactions in the application and the classes to be programmed.

In the diagram, classes are represented with boxes which contain three parts:

- The middle part contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom part contains the methods the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modeling, the classes of the conceptual design are often split into a number of sub classes.

- **Collaboration Diagram**: Collaboration diagram emphasize structural ordering of objects that send and receive messages.
- **Component Diagram**: shows organizations and dependencies among a set of components. These diagrams address the static implementation view of the system.
- **Deployment Diagram:** show the configuration of run-time processing nodes and components that live on them.

Relationships:

A relationship is a general term covering the specific types of logical connections found on class and object diagrams. UML shows the following relationships:

- **Links:** A Link is the basic relationship among objects.

- **Aggregation:** Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship. As a type of association, an aggregation can be named and have the same adornments that an association.

In UML, it is graphically represented as a hollow diamond shape on the containing class with a single line that connects it to the contained class. The aggregate is semantically an extended object that is treated as a unit in many operations, although physically it is made of several lesser objects.

- **Composition:** Composition is a stronger variant of the "has an" association relationship; composition is more specific than aggregation. Composition usually has a strong lifecycle dependency between instances of the container class and instances of the contained classes.

The UML graphical representation of a composition relationship is a filled diamond shape on the containing class end of the tree of lines that connect contained class (es) to the containing class.

- **Generalization:** The Generalization relationship ("is a") indicates that one of the two related classes (the subclass) is considered to be a specialized form of the other (the super type) and the super class is considered a 'Generalization' of the subclass.

The UML graphical representation of a composition relationship is a filled diamond shape on the containing class end of the tree of lines that connect contained classes to the containing class.

The generalization relationship is also known as the inheritance or "is a" relationship. Generalization can only be shown on class diagrams and on Use case diagrams.

- **Realization:** In UML modelling, a realization relationship is a relationship between two model elements, in which one model element realizes (implements or executes) the behavior that the other model element specifies.

The UML graphical representation of a Realization is a hollow triangle shape on the interface end of the dashed line (or tree of lines) that connects it to one or more

implementers. A plain arrow head is used on the interface end of the dashed line that connects it to its users. Realizations can only be shown on class or component diagrams.

- **Dependency:** Dependency is a weaker form of bond which indicates that one class depends on another because it uses it at some point in time. One class depends on other if the independent class is a parameter variable or local variable of a method of the dependent class.
- **Multiplicity:** This association relationship indicates that (at least) one of the two related classes make reference to the other.

The UML representation of an association is a line with an optional arrowhead indicating the role of the object(s) in the relationship, and an optional notation at each end indicating the multiplicity of instances of that entity (the number of objects that participate in the association).

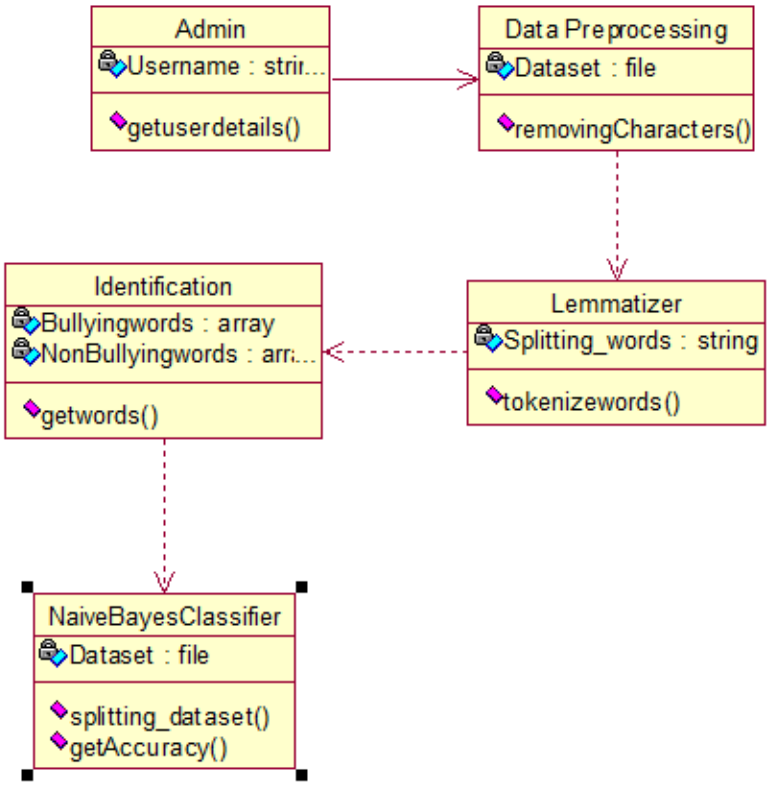


Fig 4.2.1 : Class Diagram

4.2.2 Use Case Diagram

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. Uses cases are the system functionalities written in an organized manner. The things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw an use case diagram we should have the following items identified.

Functionalities to be represented as a use case

Actor

Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
- Use note whenever required to clarify some important points.

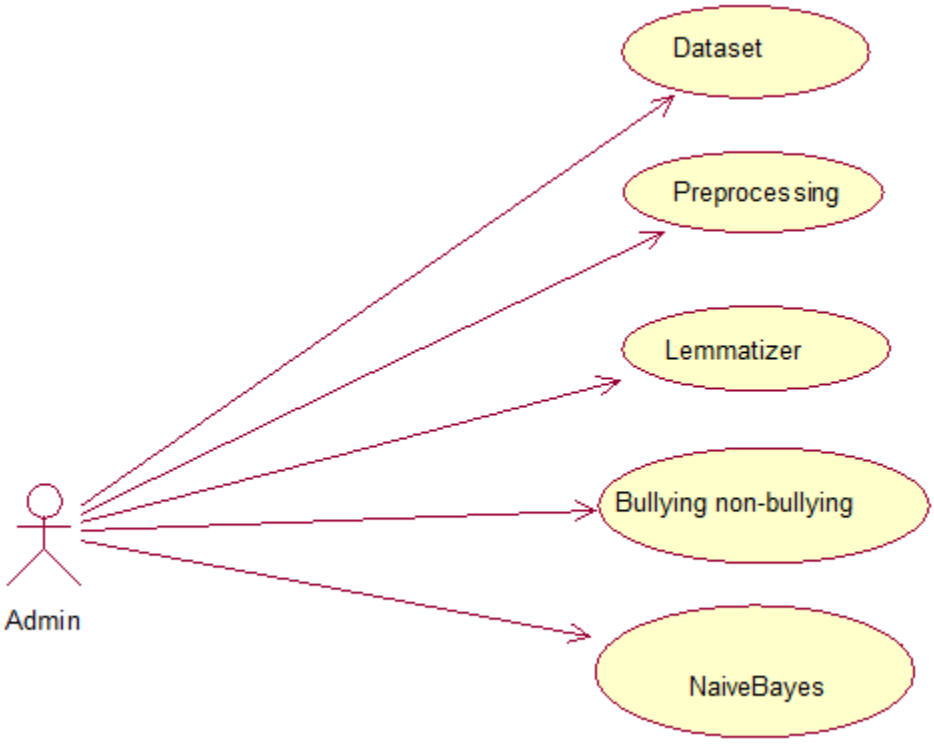


Fig 4.2.2 : Use case Diagram

4.2.3 Sequence Diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur.

The purpose of sequence diagram is to show the flow of functionality through a use case. In other words, we call it a mapping process in terms of data transfers from the actor through the corresponding objects.

The key points are:

- The main purpose is to represent the logical flow of data with respect to a process.
- A sequence diagram displays the objects and not the classes.

Messages are written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages.

When an object is destroyed (removed from memory), an X is drawn on top of the lifeline, and the dashed line ceases to be drawn below it (this is not the case in the first example though). It should be the result of a message, either from the object itself, or another.

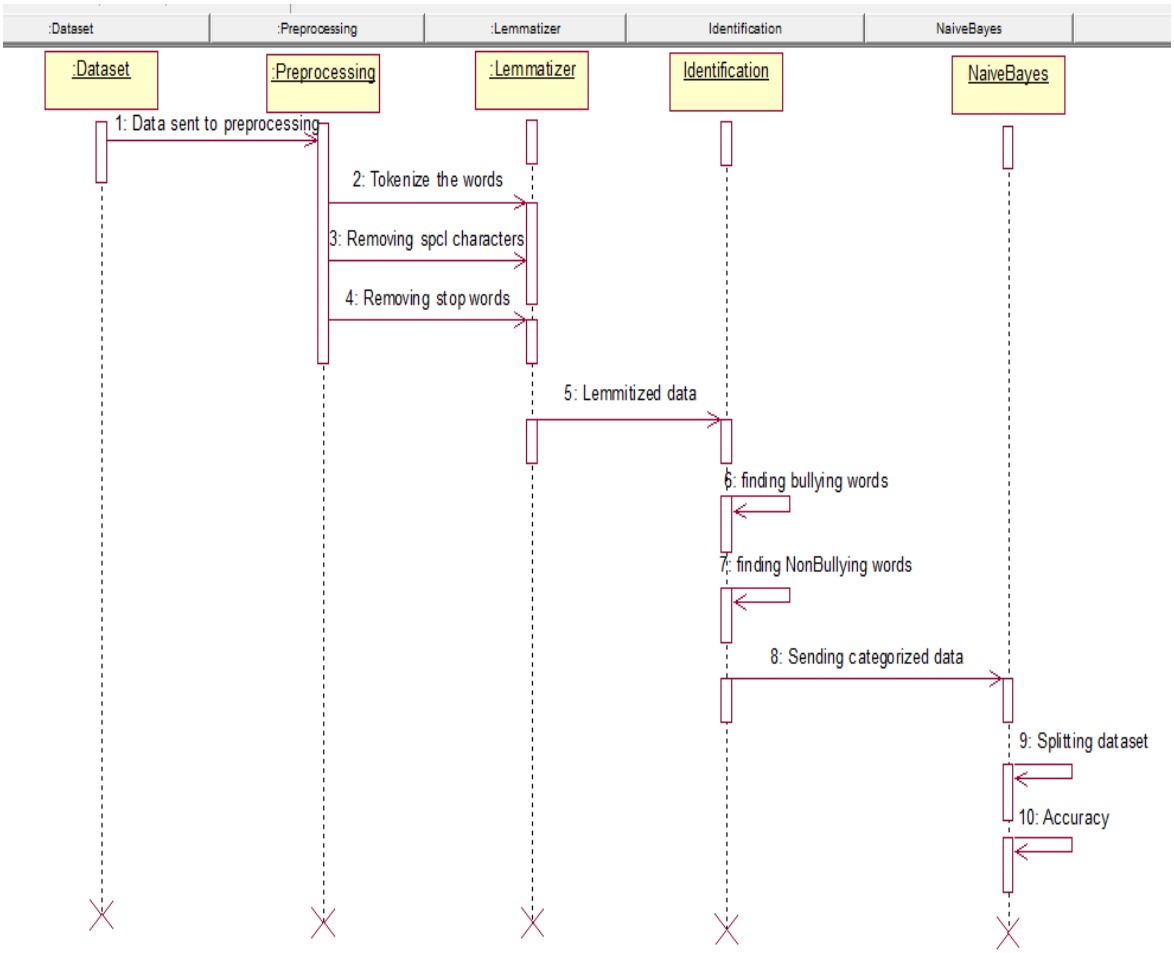


Fig 4.2.3 : Sequence Diagram

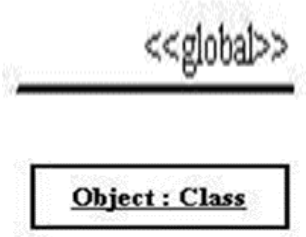
4.2.4 Collaboration Diagram

Collaboration diagram emphasize structural ordering of objects that send and receive messages.

Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system. Basic Collaboration diagrams symbols and notations.

Class roles:

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.



Association roles

Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.

Messages

Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution.

Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult to read. Several vendors offer software for creating and editing collaboration diagrams.

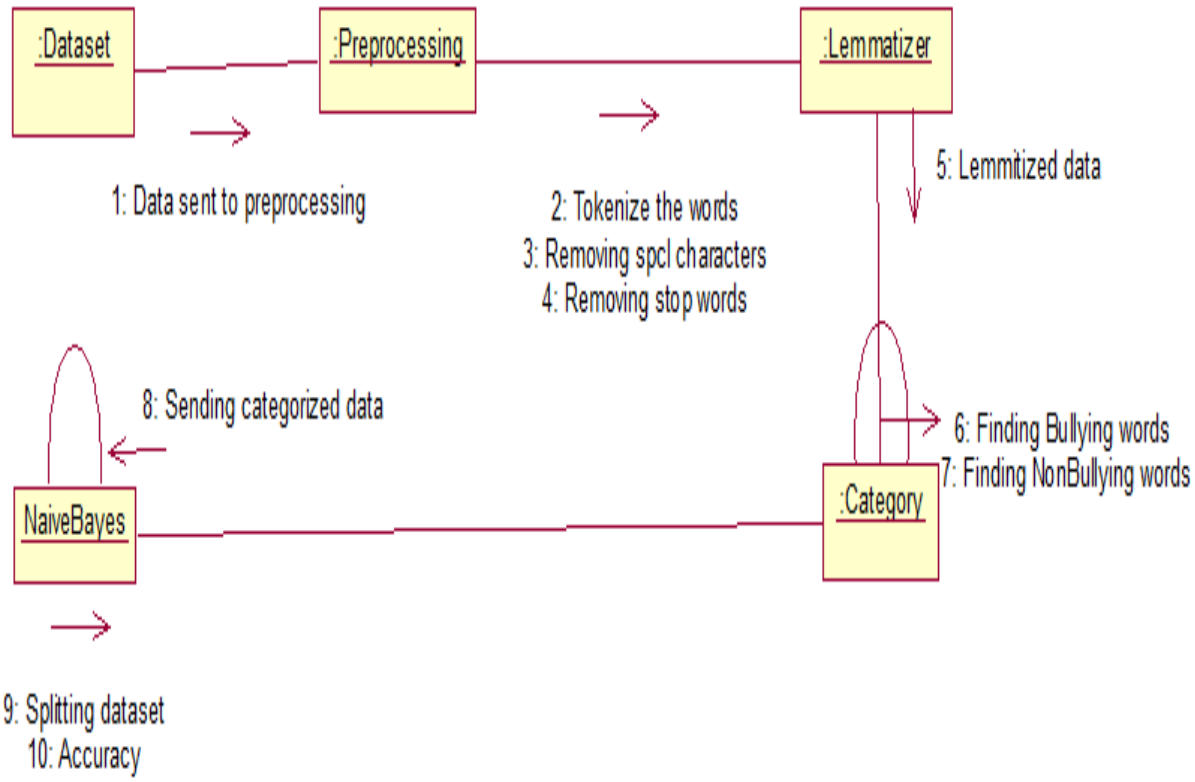


Fig 4.2.4 : Collaboration Diagram

4.2.5 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- Rounded rectangles represents actions.
- Diamonds represents decisions.
- Bars represents the start (split) or end (join) of concurrent activities.

- A black circle represents the start (initial state) of the workflow.
- An encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen.

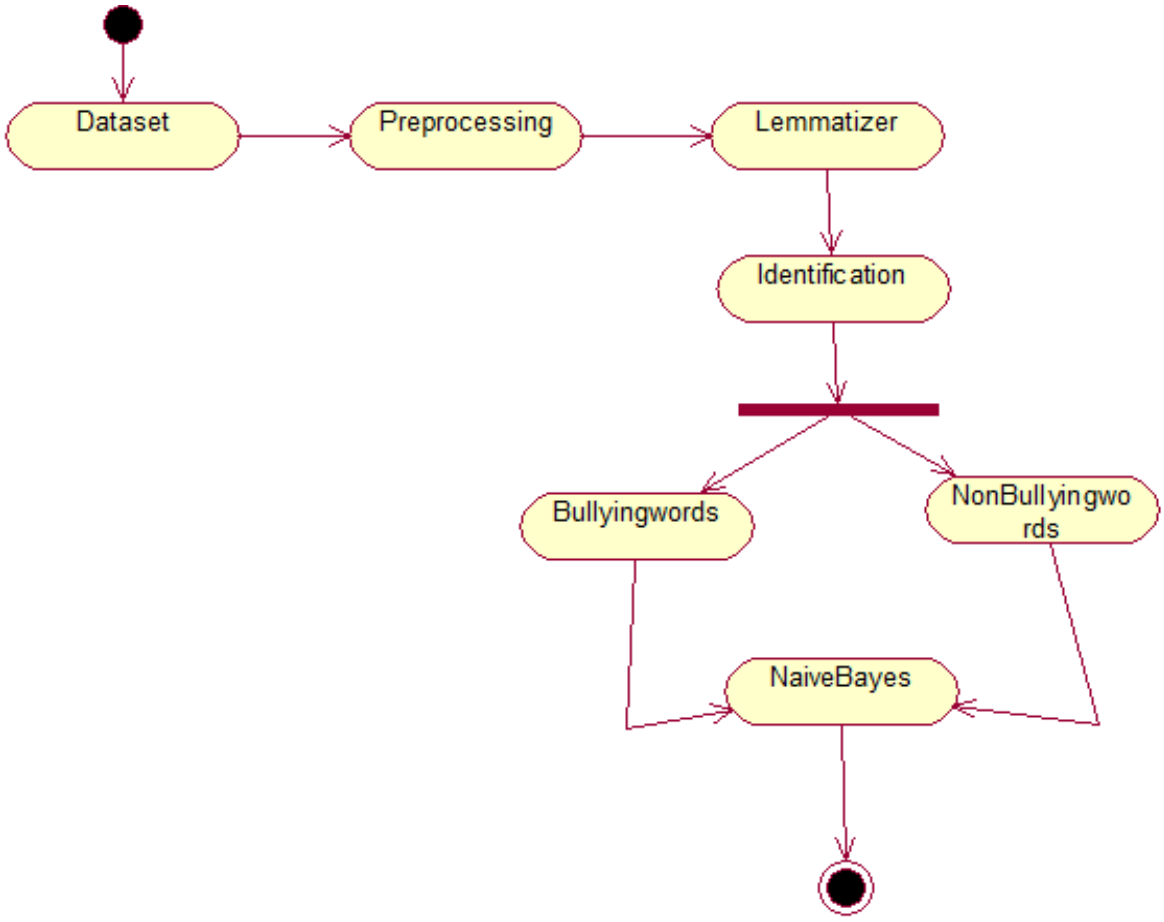


Fig 4.2.5 : Activity Diagram

5. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.1 INTRODUCTION

The modules in our project are**:**

- Loading the dataset
- Data Pre-processing
- Lemmatization
- Identifying bullying and non bullying words using polarity values.
- Applying the Naives Bayes Algorithm
- Predicting and Visualizing the results.

FLOW OF IMPLEMENTATION PROCESS:

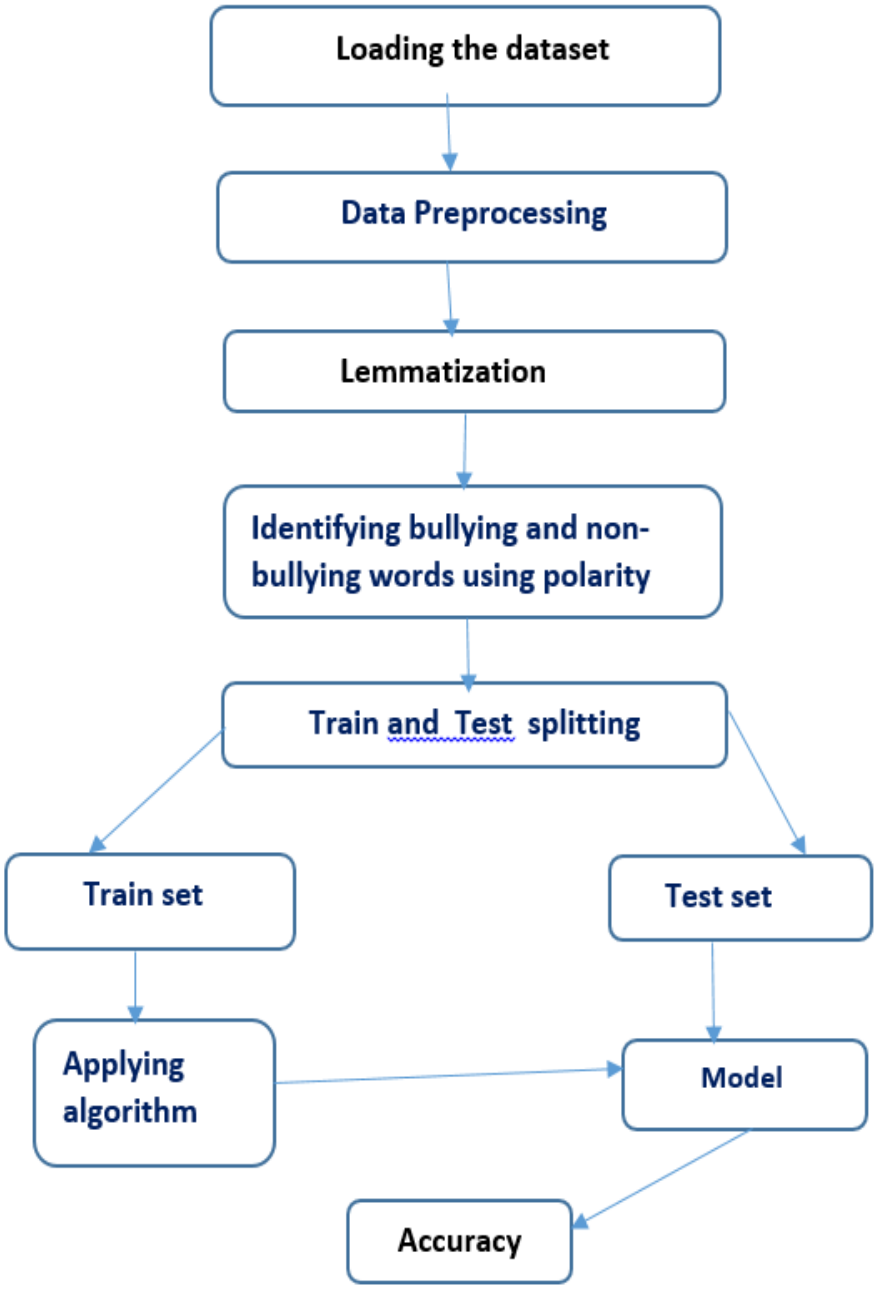


Fig 5.1 : Implementation Process

Loading the dataset

In this step we downloaded the existing Cyberbullying detection dataset from website[1].This dataset consists of Tweet and Text Label as attributes.

	Tweet	Text Label
2	.omg why are poc wearing fugly blue contacts so in, it is 2018. please make it stop.	Non-Bullying
3	.Sorry but most of the runners popular right now are just plain fugly.	Non-Bullying
4	.those jeans are hideous, and I?m afraid he?s bought the entire collection of those fugly sunglasses ??	Non-Bullying
5	.I had to dress up for a presentation in class today and I?m giving some serious Mr. Grey zaddy vibes if I do say so myself	Non-Bullying
6	.Am I the only one who thinks justin bieber is fugly af now?	Non-Bullying
7	We carry on? We as in fugly lookin unwanted people?	Non-Bullying
8	Don?t know what?s worse, the fact he?s hoying milk in before the water or his fat fugly face at the end of that vid.	Non-Bullying
9	Enjoy your New Smyrna Beach..full of seaweed, shallow fugly green water and sharks. Nothing comp.	Non-Bullying
10	Yeah honestly that?s fugly.	Non-Bullying
11	No one wants to see those fugly 3T bikes?	Non-Bullying
12	This luxury bldg they?re putting up across from .thepinhook is so fugly. I would still be mad even if it were aesthetically pleasing but ew.	Non-Bullying
13	Everyone else was in the line up except for her and they got to debut together while she gad to wear a fugly purple.	Non-Bullying
14	Fell for that again didnt ya XD #PhoneCallTrick.	Non-Bullying
15	fugly Am alright starting to get a headache myself.	Non-Bullying
16	Fugly?, Whom?, You two? Henny.....	Non-Bullying
17	o wow conq varus actually looks good unlike fucking horrifically fugly conq karma.	Non-Bullying
18	I'm fed up with this migraine.", "How's you	Non-Bullying
19	King Show off.	Non-Bullying
20	if anyone references talking about aliens when explaining aquarius they are fraudulent.', 'do not trust them, they are a fugly.	Non-Bullying
21	She's fugly.	Non-Bullying
22	.I am fugly crying!', 'I can?t handle this!', 'This was so beautiful!', 'Euphoria is such a bop too.', '????.	Non-Bullying
23	i literally only know one person who uses the word fugly.', 'lol fuck off anna..	Non-Bullying
24	.Fugly ass.	Non-Bullying
25	.today i am fatish and fugly.	Non-Bullying
26	The ?fugly? friend.', 'Funny &; ugly.	Non-Bullying
27	.Is fugly a curse word?.	Non-Bullying
28	.Pain is putting on the outfit that was so cute in your head only to find out it?s fugly on.	Non-Bullying
29	.And now theres like 20 people claimin one ov my fb statuses is bout them XD People round ear make me chuckle.	Non-Bullying
30	damn I?m fugly.	Non-Bullying
31	Fugly	Non-Bullying
32	Oh I know new titans unis are fugly!', 'Keep enjoying your mayo	Non-Bullying

Fig 5.1.1: Dataset

DATA PREPROCESSING

Data Preprocessing in Machine learning is a technique refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format.

Data pre-processing includes:

1. Analysing the dataset.

2. Identifying the categorical data
3. Encoding the categorical data

Lemmatization

Lemmatization considers the context and converts the word to its meaningful base form. For example, lemmatization would correctly identify the base form of ‘caring’ to ‘care’.

Test Train Splitting

In the dataset we are using usually divided into training set and testing set in the 8:2 ratio respectively.

Training set

The training data set in Machine Learning is the actual dataset used to train the model for performing various actions. This is the actual data the ongoing development process models learn with various API and algorithm to train the machine to work automatically.

Testing set

A test dataset is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. It is used to test the trained model and predict the results.

Identifying bullying and non bullying words

In dataset we are identifying bullying and non bullying words .In this if polarity above ‘one’ are positive and below ‘one’ are negative. If polarity is ‘zero’ which means the text is neutral.

Applying the Algorithm

There are several machine learning Algorithms for classifying and analyzing the data. We implemented the Naive Bayes Classifier on the preprocessed data to predict the results.

Prediction and visualizing the results

In this step the predicted results and accuracy will be displayed on the screen.

5.2 ALGORITHM

Step 1: The first task is to separate the training dataset instances by class value so that we can calculate statistics for each class.

We can do that by creating a map of each class value to a list of instances that belong to that class and sort the entire dataset of instances into the appropriate lists.

Step 2: Calculating Mean and Standard deviation

- Calculate the mean of each attribute for a class value. The mean is the central middle or central tendency of the data, and we will use it as the middle of our Gaussian distribution when calculating probabilities.
- We also need to calculate the standard deviation of each attribute for a class value. The standard deviation describes the variation of spread of the data, and we will use it to characterize the expected spread of each attribute in our Gaussian distribution when calculating probabilities.

Step 3: We can pull it all together by first separating our training dataset into instances grouped by class. Then calculate the summaries for each attribute.

Step 4: Calculate Gaussian Probability Density Function.

We used the Gaussian function to estimate the probability of a given attribute value, given the known mean and standard deviation for the attribute estimated from the training data**.**

$$P(x; \mu, \sigma) = \frac{1}{(\sigma\sqrt{2\pi})} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Where ɤ is the attribute value

μ is the Mean

σ is the standard deviation

Step 5: Calculate Class Probabilities

Now we can calculate the probability of an attribute belonging to a class, we can combine the probabilities of all of the attribute values for a data instance and come up with a probability of the entire data instance belonging to the class. We combine probabilities together by multiplying them.

Step 6: Now that we can calculate the probability of a data instance belonging to each class value, we can look for the largest probability and return the associated class. In this way we can predict the result.

5.3 Code

"""Importing the Libraries"""

```
from textblob import TextBlob

import csv

import re

import operator

import numpy as np

import pandas as pd
```

```
from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

import string

from nltk.stem import WordNetLemmatizer

import matplotlib.mlab as mlab

import matplotlib.pyplot as plt

import glob, os

import naiveb

from io import BytesIO

import base64

import io

from io import StringIO

from flask import Flask, make_response, request ,render_template, url_for

app = Flask(__name__)

APP_ROOT=os.path.dirname(os.path.abspath(__file__))

@app.route('/')

def form():

    return render_template('frontend.html')

@app.route('/transform', methods=["POST"])
```

```
def transform_view():
    file = request.files['data_file']
    filename = file.filename
    new_path = os.path.abspath(filename)
    print(new_path)
    if not file:
        return "No file"
    df1 = file.stream.read().decode("utf-8")
    target=os.path.join(APP_ROOT,'plots/')
    #target=os.path.dirname(APP_ROOT)
    if not os.path.isdir(target):
            os.mkdir(target)
    Tweet = []
    s=[]
    w=[]
    dw=[]


 """Dataset Reading"""
    df1 = pd.read_csv(new_path)
```

```
l1=""

l2=[]

tweets=[]

for row in df1["Tweet"]:

    """Reading the Dataset"""

    words = word_tokenize(row)

    clean_words = [word.lower() for word in words if word not in set(string.punctuation)]

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

    characters_to_remove = ["""",'``',"rt","https","’","“","”","\u200b","--","n't","'s","...","//t.c" ]

    clean_words = [word for word in clean_words if word not in english_stops]

    clean_words = [word for word in clean_words if word not in set(characters_to_remove)]

  """Applying wordnetlemmatizer"""

     wordnet_lemmatizer = WordNetLemmatizer()

    lemma_list = [wordnet_lemmatizer.lemmatize(word) for word in clean_words]

    Tweet.append(lemma_list)

    l1=" ".join(lemma_list)

    l2.append(l1)
```

```
out=open("C:/Users/Lavanya/ss.csv","w")
for column in l2:
    out.write("%s" % column)
    out.write('\n')
out.close()
```

"""Wrongwords file"""

```
 with open("C:/Users/Lavanya/Desktop/wrongwords2.txt") as f:
    lines=f.readlines()
#print(lines)
lines=[x.strip() for x in lines]
def does_contain_words(tweet, wordsToCheck):
    for word in wordsToCheck:
        if word in tweet:
            return True
    return False
```

”””Writing the bullying data into output.csv file”””

```
with open('C:/Users/Lavanya/ss.csv', 'r') as f,open('C:/Users/Lavanya/output.csv', 'w') as f_out:
    reader = csv.reader(f)
```

```
    writer = csv.writer(f_out)
    writer.writerow(("Tweet","Text label"))
    for row in reader:
        try:
            str1 = row[0]
        except IndexError:
            str1 = 'null'
        if does_contain_words(str1, lines):
            data='bullying'
            writer.writerow((row[0],data))
        else:
            data='Non-Bullying'
            #writer.writerow((row[0], data))
p=[]
df2 = pd.read_csv("C:/Users/Lavanya/outp.csv")
for row in df2['Tweet']:
    p.append(row)
for i in range(len(p)):
    q=dict()
```

```
    q['TextBlob']=TextBlob(p[i])
    tweets.append(q)
for q in tweets:
    q['polarity'] = float(q['TextBlob'].sentiment.polarity)
    q['subjectivity'] = float(q['TextBlob'].sentiment.subjectivity)
    if q['polarity'] >= 0.1:
        q['sentiment'] = 'positive'
    elif q['polarity'] <= -0.1:
        q['sentiment'] = 'negative'
    else:
        q['sentiment'] = 'neutral'
tweets_sorted = sorted(tweets, key=lambda k: k['polarity'])
```

"""Printing Negative Messages"""

```
 print("\n\nTOP NEGATIVE TWEETS")
negative_tweets = [d for d in tweets_sorted if d['sentiment'] == 'negative']
#print(negative_tweets)
for q in negative_tweets:
    print("msg=%s" % (q['TextBlob']))
```

```
    w.append(q['TextBlob'])
with open('C:/Users/Lavanya/csvf/lav1.csv', 'w') as f_out:
    writer1= csv.writer(f_out)
    writer1.writerow(("Tweet","Textlabel","sentiment","Category"))
    for q in negative_tweets:
        try:
            writer1.writerow((q['TextBlob'],q['sentiment'],q['polarity'],'bullying'))
        except IndexError:
            str1 = 'null'
#return render_template('csvfile.html')
```

"""Printing Positive Messages"""

```
print("\n\nTOP POSITIVE TWEETS")
positive_tweets = [d for d in tweets_sorted if d['sentiment'] == 'positive']
for q in positive_tweets:
    print("msg=%s" % (q['TextBlob']))
    dw.append(q['TextBlob'])
with open('C:/Users/Lavanya/csvf/lav2.csv', 'w') as f_out:
    writer1= csv.writer(f_out)
```

```
    writer1.writerow(("Tweet","Textlabel","sentiment","Category"))


    for q in positive_tweets:
        try:
            writer1.writerow((q['TextBlob'],q['sentiment'],q['polarity'],'non bullying'))
        except IndexError:
            str1 = 'null'
```

"""Printing Neutral Messages"""

```
print("\n\nTOP NEUTRAL TWEETS")
neutral_tweets = [d for d in tweets_sorted if d['sentiment'] == 'neutral']
for q in neutral_tweets:
   print("msg=%s" % (q['TextBlob']))
   s.append(q['TextBlob'])
with open('C:/Users/Lavanya/csvf/lav3.csv', 'w') as f_out:
    writer1= csv.writer(f_out)
    writer1.writerow(("Tweet","Textlabel","sentiment","Category"))
    for q in neutral_tweets:
        try:
```

```
            writer1.writerow((q['TextBlob'],q['sentiment'],q['polarity'],'non bullying'))
        except IndexError:
            str1 = 'null'
os.chdir("C:/Users/Lavanya/csvf")
results = pd.DataFrame([])
for counter, file in enumerate(glob.glob("lav*")):
namedf=pd.read_csv(file,skiprows=0,usecols=['Tweet','Textlabel','sentiment','Category'])
    results = results.append(namedf)
results.to_csv('C:/Users/Lavanya/combinedfile.csv')
naiveb.naiveb()
"""Histogram for CyberBullying"""
figfile = io.BytesIO()
x = [d['polarity'] for d in tweets_sorted]
num_bins = 21
n, bins, patches = plt.hist(x, num_bins, normed=1, facecolor='green', alpha=0.5)
plt.xlabel('Polarity')
plt.ylabel('Probability')
plt.title(r'Histogram of polarity')
plt.subplots_adjust(left=0.15)
```

```
#plt.show()

#plt.savefig('C:/Users/Lavanya/Flask/App/csvf/dig.png')

plt.savefig("C:/Users/Lavanya/Flask/App/csvf2/pic1.png")

plt.savefig(figfile, format='png')

figfile.seek(0)

figdata_png = base64.b64encode(figfile.getvalue()).decode()

plt.close()

result = '{}'.format(figdata_png)

 """Piechart for CyberBullying"""

figfile1 = io.BytesIO()

pos = len(positive_tweets)

neg = len(negative_tweets)

neu = len(neutral_tweets)

labels = 'Positive', 'Negative', 'Neutral'

sizes = [pos, neg, neu]

colors = ['yellowgreen', 'gold', 'lightcoral']

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

        autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
```

```
#plt.show()
    plt.savefig("C:/Users/Lavanya/Flask/App/csvf2/pic2.png")
    plt.savefig(figfile1, format='png')
    figfile1.seek(0)
    figdata_png1 = base64.b64encode(figfile1.getvalue()).decode()
    plt.close()
    result1 = '{}'.format(figdata_png1)
    if request.method == 'POST':
        if request.form.get('neg') == 'neg':
            return render_template("butt1.html",your_list=w)
        elif  request.form.get('pos') == 'pos':
            return render_template("butt2.html",your_list1=dw)
        elif  request.form.get('neu') == 'neu':
            return render_template("butt3.html",your_list2=s)
        elif request.form.get('imag') == 'imag':
            return render_template("graphs.html", name1=result)
        elif request.form.get('imag1') == 'imag1':
            return render_template("graphs1.html", name2=result1)
        else:
```

```
        return render_template('frontend.html')

if __name__ == '__main__':

app.run(port=4555, debug=True)
```

Algorithm Implementation:

```
“””Importing naive bayes as an module"""

def naive():

    “””Importing Librabies”””

    import numpy as np

    import pandas as pd

    import math

    “””Dataset Reading”””

    dataset1=pd.read_csv(r"C:/Users/Lavanya/combinedfile.csv")

    #df = pd.read_csv('1459966468_324.csv', encoding = 'utf8')

    x=dataset1.iloc[:, :].values

    dfx=pd.DataFrame(x)

“””Encoding Categorical Data”””

    from sklearn.preprocessing import LabelEncoder

    labelencoder=LabelEncoder()

    #dfx.values[:,0]=labelencoder.fit_transform(dfx.values[:,0])
```

```
dfx.values[:,2]=labelencoder.fit_transform(dfx.values[:,2])

dfx.values[:,4]=labelencoder.fit_transform(dfx.values[:,4])

dfx1=dfx.drop(0,axis=1)

dfx1=dfx.drop(1,axis=1)

dfx.to_csv("C:/Users/Lavanya/combinedfile1.csv")

path1='C:/Users/Lavanya/combinedfile1.csv'

dataset=np.genfromtxt(path1,delimiter=',',skip_header=1,dtype=float,usecols=[2,4])
```

”””Separating the training dataset instances by class value”””

```
def seperateByClass(dataset):
    seperated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if(vector[-1] not in seperated):
            seperated[vector[-1]] = []
        seperated[vector[-1]].append(vector)
    return seperated
```

”””Calculate mean”””

```
def mean(numbers):
        return sum(numbers)/float(len(numbers))
```

```
"""Calculate Standard Deviation"""
    def stdev(numbers):
        avg = mean(numbers)
        variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
        return math.sqrt(variance)
    def summarize(dataset):
        summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
        del(summaries[-1])
        return summaries
    """Summarizing the data by class value"""
    def summarizeByClass(dataset):
        separated = seperateByClass(dataset)
        summaries = {}
        for classValue, instances in separated.items():
            summaries[classValue] = summarize(instances)
        return summaries
    """Calculating the Gaussian Probability Density function"""
    def calculateProbability(x, mean, stdev):
        exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
```

```
        return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent

    """Calculating probability of an attribute belonging to class"""

    def calculateClassProbabilities(summaries, inputVector):

        probabilities = {}

        for classValue, classSummaries in summaries.items():

            probabilities[classValue] = 1

            for i in range(len(classSummaries)):

                mean, stdev = classSummaries[i]

                x = inputVector[i]

                probabilities[classValue] *= calculateProbability(x, mean, stdev)

        return probabilities

"""Making a prediction based on the best probability"""

    def predict(summaries, inputVector):

        probabilities = calculateClassProbabilities(summaries, inputVector)

        bestLabel, bestProb = None, -1

        for classValue, probability in probabilities.items():

            if bestLabel is None or probability > bestProb:

                bestProb = probability

                bestLabel = classValue
```

```
        return bestLabel
def getPredictions(summaries, testSet):
        predictions = []
        for i in range(len(testSet)):
                result = predict(summaries, testSet[i])
                predictions.append(result)
        return predictions
def getAccuracy(testSet, predictions):
        correct = 0
        for i in range(len(testSet)):
                if testSet[i][-1] == predictions[i]:
                        correct += 1
        return (correct/float(len(testSet))) * 100.0
def pred():
    from sklearn.model_selection import train_test_split
    trainingSet,testSet=train_test_split(dataset,test_size=0.2,random_state=0)
    #trainingSet,testSet=splitDataset(dataset,splitratio)
    #trainingSet,testSet=splitDataset(dataset, splitratio)
    summaries=summarizeByClass(trainingSet)
```

```
        #print(trainingSet)
        #print(testSet)
        predictions=getPredictions(summaries, testSet)
        #print("predictions=",predictions)
        #pr=np.array(predictions)
        #print(pr)
        accuracy=getAccuracy(testSet, predictions)
        print("accuarcy=",accuracy)
    pred()
```

Frontend.html

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <linkrel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

```
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min.js"></script
>

        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>

        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.bundle.min.js"></scr
ipt>

        <script
src="https://cdnjs.cloudflare.com/ajax/libs/typed.js/2.0.10/typed.min.js"></script>

        <script src="http://code.jquery.com/jquery-3.3.1.js"></script>

</head>

<style type=text/css>

 .leftdiv

       {

 background-color:red

  }

div

{

 background-color: 009900;

  border: solid black;
```

```
}

</style>

<center><b><h1 style="background-color:lightblue">

CYBERBULLYING DETECTION IN SOCIAL MEDIA</h1>

</b></center>

<body

<div class="leftdiv">

<p>Cyber bullying can be defined as an aggressive, intentional actions performed by an individual or a group of people via digital communication methods such as sending messages and posting comments against a victim. It can take place anywhere on social media at any time leading to occurrence of self-injuries or suicides.Our Project is intended to detect cyberbullying in social media with the help of lemmatizer. We are using Naive Bayes algorithm for predictions. The current scope of the project is limited to identification of bullying messages .Cyber Bullying detection can be formulated as supervised learning problem. We find the bullying messages with the help of polarity values. Later we apply Naive Bayes classifier for predictions. Since the text content is most reliable,our work here focuses on text-based Cyberbullying detection.

</p>

</div>

<div class="border container" style="border-radius: 25px;"><br>

 """Uploading a CSV File"""

<center><h2>Upload a CSV File</h2>
```

<form action="/transform" method="POST" enctype="multipart/form-data">

<input type="file" name="data_file" />

<button type="button" onclick="alert('CSV FILE UPLOADED SUCCESSFULLY')">Submit</button>

</br>

<button type="submit" name="neg" value="neg" style="background-color:pink" >Negative Messages</button>

</br>

<button type="submit" name="pos" value="pos" style="background-color:pink">Positive Messages</button>

</br>

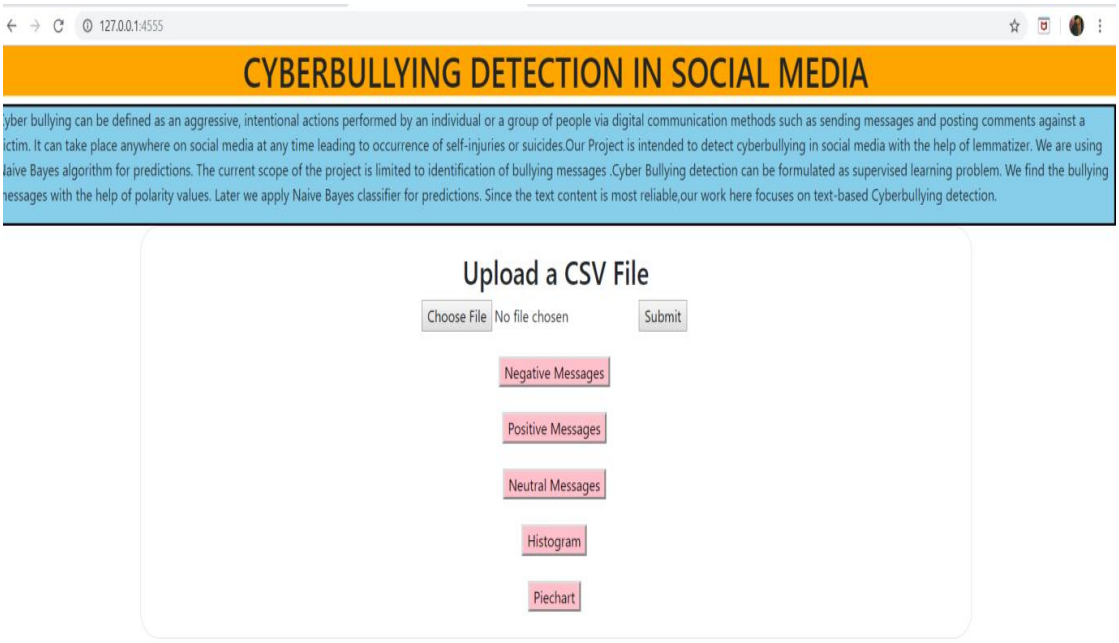
<button type="submit" name="neu" value="neu" style="background-color:pink" >Neutral Messages</button>

</br>

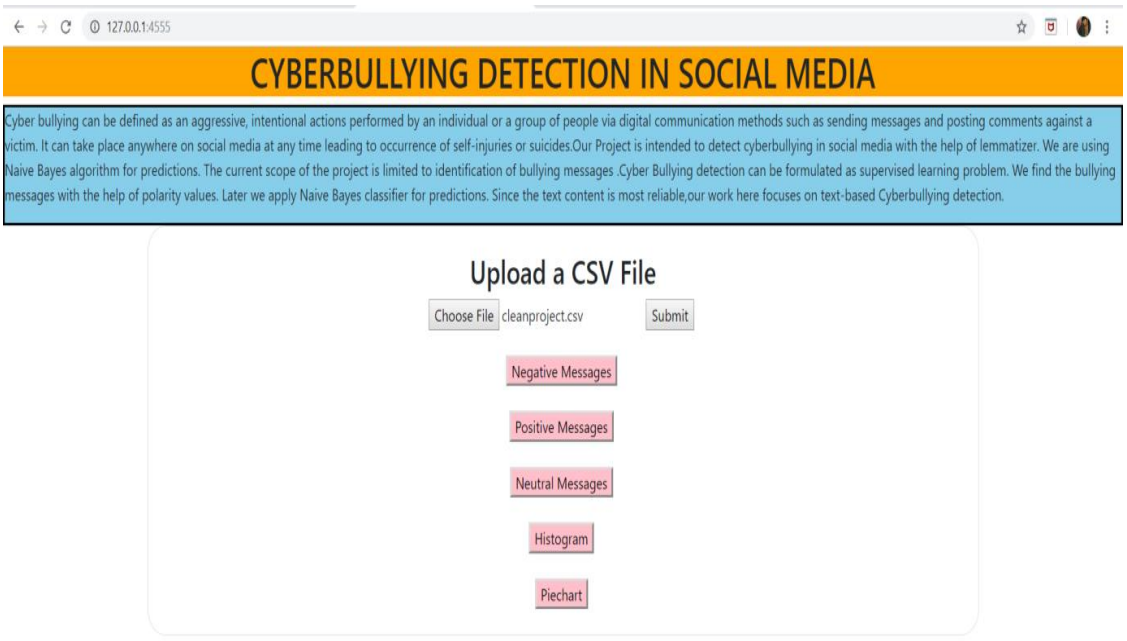
<button type="submit" name="imag" value="imag" style="background-color:pink"> Histogram </button>


```
</br>

<button type="submit" name="imag1" value="imag1" style="background-color:pink" >Piechart</button>

<br>

</br></div>

</form>

</center>

</body>

</html>
```

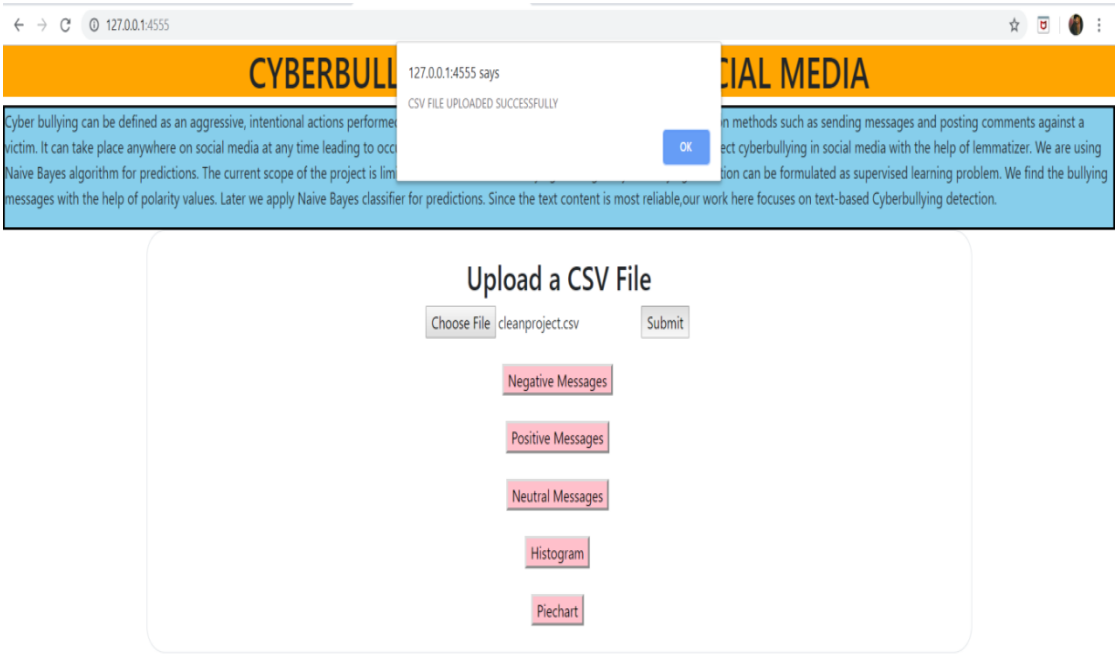
5.4 OUTPUT SCREENS



Screen 1: Home Page

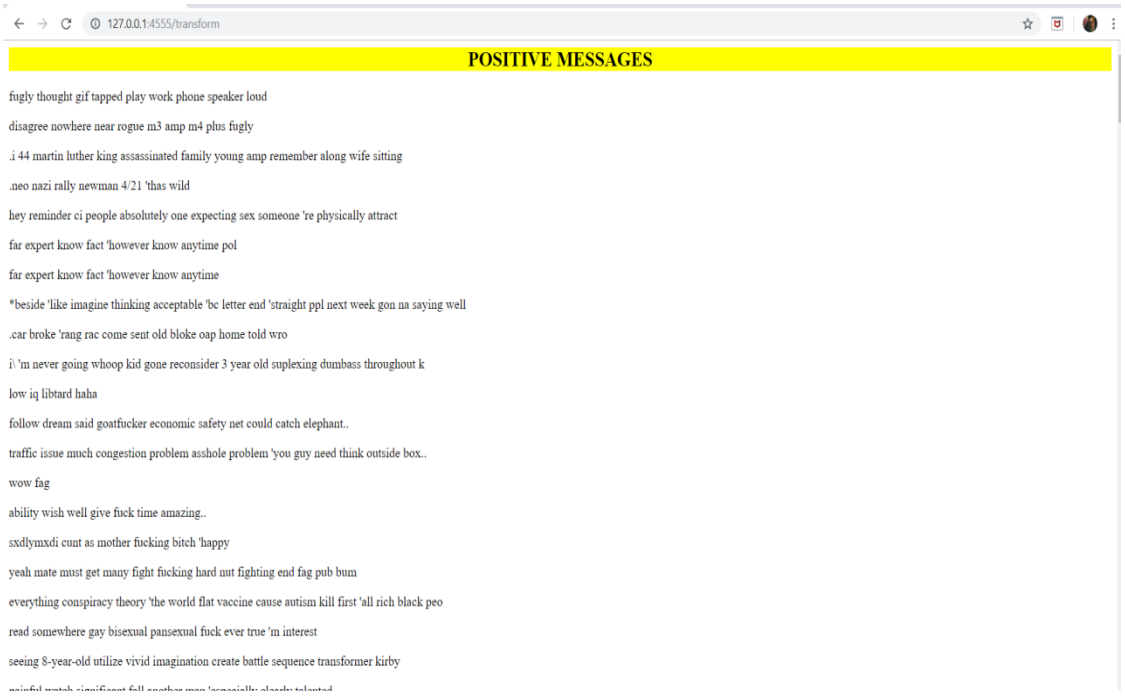


Screen 2: Choosing a CSV file

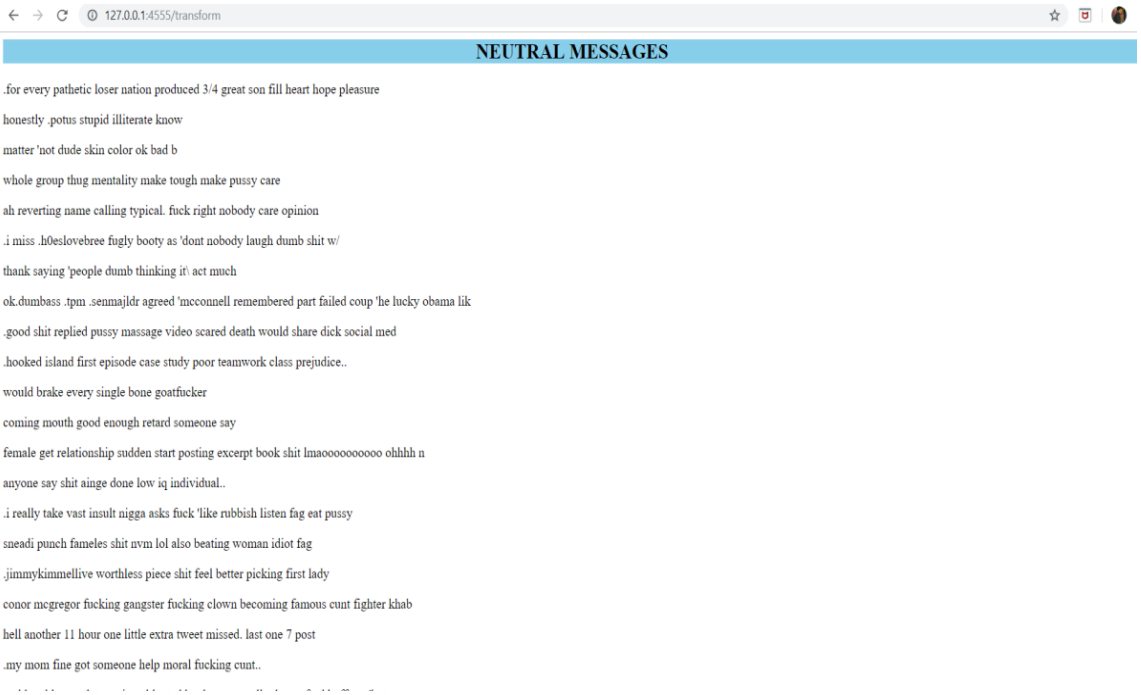


Screen 3: Succesful updation message

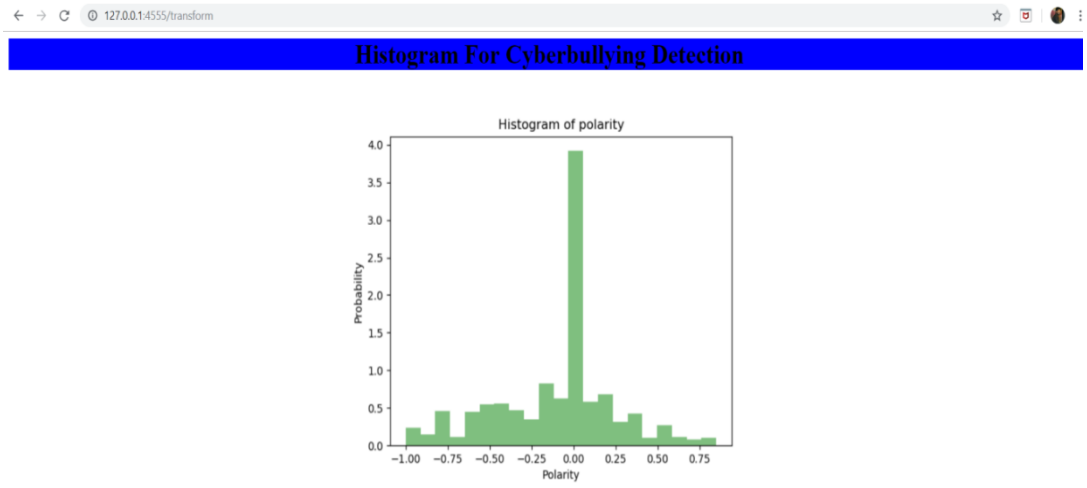
Screen 4: Negative Messages



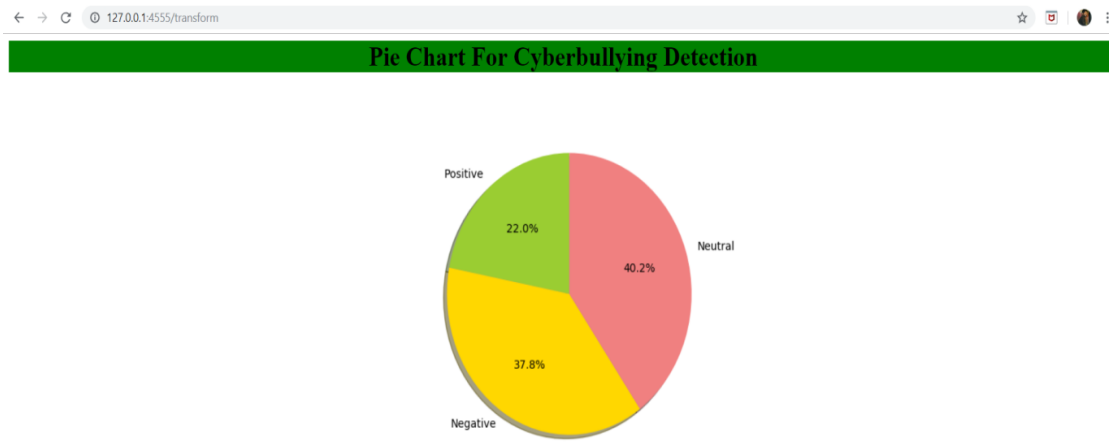
Screen 5: Positive Messages



Screen 6: Neutral Messages



Screen 7: Histogram For Message Identification



Screen 8: Pie Chart For Message Identification

6.CONCLUSION

The text based cyber bullying detection problem is being addressed, where robust and discriminative representation of messages are critical for an effective detection system. Our project can be implemented in social media platforms like FaceBook, Twitter. Since the text content is most reliable, our work here focussed on text-based Cyberbullying detection. We have used an algorithm which is defined in TextBlob to detect the Bullying text.

7.FUTURE ENHANCEMENT

Cyber Bullying detection is essential in today's world of internet and networking. As cyberbullying is a major issue in today's world. we are focused on detecting bullying messages in our project which are in textual format. If the same user tries to send many Bullying messages then we can block that user's account. Thus, we can reduce cyber crimes appearing on different social media platforms. This can be further implemented on other medias like audios, images and videos. This can be implemented on real time streaming data[6].

REFERENCES

[1]. www.kaggle.com

[2]. https://www.machinelearningplus.com/nlp/lemmatization-examples-python/

[3]. https://anaconda.org/anaconda/Jupyter

[4]. https://planspace.org/20150607-textblob_sentiment/

[5]. https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[6]. https://ieeexplore.ieee.org/document/7489220