CYBER BULLYING DETECTION USING MACHINE LEARNING

A Major Project Report Submitted
In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Information Technology

by

Mergoju Usha Rani	**18N31A1298**
Sathvika Kurella	**18N31A1282**
Moyya Gnan Akshith	**18N31A12A6**

Under the esteemed guidance of

D. Chandhra Sekhar Reddy
Associate Professor

Department of Information Technology

Malla Reddy College of Engineering & Technology
(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

2018-2022

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled "**Cyber Bullying Detection using Machine Learning**", submitted by **Mergoju Usha Rani(18N31A1298), Sathvika Kurella (18N31A1282)** and **Moyya Gnan Akshith (18N31A12A6)** of B. Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Information Technology, Department of IT during the year 2021-2022. The results embodied in this project report have not been submitted to any other university orinstitute for the award of any degree or diploma.

Internal Guide

D. Chandhra Sekhar Reddy
Assoc. Professor

Head of the Department

Dr. G.Sharada
Professor

External Examiner

DECLARATION

We hereby declare that the project titled "**Cyber Bullying Detection using Machine Learning**" submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Information Technology is a result of original research carried-out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Mergoju Usha Rani - 18N31A1298
Sathvika Kurella - 18N31A1282
Moyya Gnan Akshith - 18N31A12A6

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous), our Director Dr. V.S.K. Reddy and Principal Dr S Srinivasa Rao who gave us the opportunity to have experience in engineering and profound technical knowledge.

We express our heartiest thanks to our Head of the Department Dr. G. Sharada for encouraging us in every aspect of our project and helping us realize our full potential.

We would like to thank our internal guide D. Chandhra Sekhar Reddy for the regular guidance and constant encouragement. We are extremely grateful to the valuable suggestions and unflinching co-operation throughout project work.

We would like to thank our class in charge B. Aruna Kumari who in spite of being busy with his duties took time to guide and keep us on the correct path.

We would also like to thank all the supporting staff of the Department of IT and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

With regards and gratitude,

Mergoju Usha Rani - 18N31A1298

Sathvika Kurella - 18N31A1282

Moyya Gnan Akshith - 18N31A12A6

Abstract

Cyberbullying is an activity of sending threatening messages to insult person. To prevent cyber victimization from the activity is challenging. Cyberbully is a misuse of technology advantage to bully a person. Cyberbully and its impact have occurred around the world and now the number of cases are increasing. Cyberbullying detection is very important because the online information is too large, so it is not possible to be tracked by humans. This project enhanced the Naïve Bayes classifier for extracting the words and examining loaded pattern clustering. The purpose of this research is to construct a classification model with optimal accuracy in identifying cyberbully conversation using Naive Bayes.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 INTRODUCTION:

Due to the advances of internet and information technology, Online Social Network (OSN) services, such as Facebook, Twitter, and MySpace are gaining in popularity as a main source of spreading messages to other people. Messaging is widely used and very useful in variouspurposes, for example, business, education, and socialization. However, it also provides opportunity to create harmful activities. There are numerous evidences showing that messaging can introduce the very concerned problem, namely cyberbullying.

Cyberbullying involves the offensive information such as harassment, insult, and hate in the messages which are sent or post using OSN services for the purpose of intentionally hurting people emotionally, mentally, or physically. It can cause low self-esteem, anxiety, depression, a variety of other emotional problems, and even suicide. Its tragic consequences have continuouslyreported typically among the school-age children. Since the number of cyberbullying experiences has recently been increasing, an intensive study of how to effectively detect and prevent it from happening in real time manner is urgently needed. To prevent victims from the incidents,blocking the message is not an effective way. Instead, texts in the messages should be monitored,processed and analyzed as quickly as possible in order to support real time decisions.

As the problems mentioned, a number of studies are dedicated to explore various techniques to detect cyberbullying efficiently. Manual detection is considered the most accurate detection, but it is hardly employed because it takes too much time and lots of resources. Automatic cyberbullying detection system is therefore emphasized. Text mining technology and technique are mostly used. Karthik applied multi classifiers such as Naïve Bayes, JRip, J48 and SMO with YouTube comments. Vinita used LDA to extract features and employ weighted term frequency-inverse document frequency (TF-IDF) function to improve the classification with datasets from

Kongregate, Slashdot, and MySpace. Homa used Support Vector Machines classifier with datasets from Instagram and Romsaiyud applied the ExpectationMaximization (EM) for clustering documents from data streams for threat cyberbullying detection.

Even though cyberbullying detection system has extensively been exploring, cyberbullying remains a growing concern and the existing approaches are still inadequate especially when dealing with a large volume of data. Various kinds of OSN services can represent different formsor patterns of data. Additionally, reduction in computation time becomes very crucial. The detection of cyberbullying is therefore still challenging.

1.2 PROBLEM STATEMENT:

- According to research from The University of British Columbia, cyberbullying is a bigger problem than traditional bullying. There were surveys of 733 adolescents, stating that 25- 30% of them had been involved in cyberbullying, while only 12% of them had been involved in traditional bullying. 95% of them declared using mocks in internet only as a joke, and the rest is meant to insult or hurt someone. It states that teenagers greatly underestimate the danger of cyberbullying.

1.3 PROJECT OBJECTIVE:

- The detection of cyberbullying in the use of online platforms becomes very important. Due to too much information that is not possible to track by humans, automatic detection is needed that can identify threatening situations and hazardous content. This enables large-scale monitoring of social media.

1.4 EXISTING SYSTEM:

Several studies related to cyberbullying analysis and detection using text mining by classifying conversations or posting. Using supervised learning, labeling using N grams and weighting using TF-IDF.

- supervised machine learning approach, they collected youtube comments, labeled them manually and implemented various binary and multiclass classifications.

PROPOSED SYSTEM:

We developed an automatic cyberbullying detection system to detect, identify, and classify cyberbullying activities from the large volume of streaming texts from Live chatting. Texts are fed into cluster and discriminant analysis stage which is able to identify abusive texts. The abusive texts are then clustered by using Naïve Bayes is used as classification algorithms to build a classifier from our training datasets and build a predictive model . The first method aims to clean and pre-process our datasets by removing non-printable and special characters, reducing the duplicate words and clustering the datasets.The second one concerns classification model to predict the text messages for preventing cyberbullying and changing bullying word to hide state using start marks.

1.5 SCOPE OF THE PROJECT:

Cyberbullying is the use of electronic communication to bully a person by sending harmful messages using social media, instant messaging or through digital messages. Cyberbullying can be very damaging to adolescents and teens. It can lead to anxiety, depression, and even suicide. Also, once things are circulated on the Internet, they may never disappear, resurfacing at later times to renew the pain of cyberbullying. Cyberbullying can be very damaging to adolescents and teens. It can lead to anxiety, depression, and even suicide. Also, once things are circulated on the Internet, they may never disappear, resurfacing at later times to renew the pain of cyberbullying. So overcome these issues detecting the cyberbullying is very important in now a days which will help to stop cyberbullying on social media networks

1.6 LITERATURE SURVEY:

AUTHORS:- Yin, Xue & Hong

PAPER:- Psychological, Physical, and Academic Correlates of Cyberbullying and Traditional

bullying, J. Adolescent Health, 2013, vol. 53, no. 1, pp.513-520.

In recent years there are several studies related to cyberbullying analysis and detection using text mining by classifying conversations or posting. Yin, Xue and Hong use supervised learning, labeling using N-grams and weighting using TF-IDF.

AUTHORS:- Dinakar, Reichart and Lieberman

PAPER:- Improving cyberbullying detection with user context, Advances in Information Retrieval, Springer, 17(2),124-129.

Dinakar, Reichart and Lieberman conducted a supervised machine learning approach, they collected youtube comments, labeled them manually and implemented various binary and multiclass classifications. Kelly Reynolds used the decision tree (J48) and k-nearest neighbor (k = 1 and k = 3), labeling using Amazon Mechanical Turk. Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes and Bart Desmet use Support Vector Machines (SVM) as classification algorithms because they are well proven in classification. In their research, when the preprocessing step happens, they apply tokenization, PoS-tagging and lemmatization to the data by using LeTs Preprocess Toolkit..

AUTHOR:- Kelly Reynolds

PAPER:- An Effective Approach for Cyberbullying Detection,Communications in Information Science and Management Engineering, 2013, vol. 3.

Based on those facts, this research will be done to classify cyberbullying on text conversations using text mining method by developing previous research from Kelly Reynolds (2012). The research was conducted by identifying the characteristics of cyberbullying on the conversation as well as classification using SVM and Naive Bayes method as comparison with Kelly Reynolds method of decision tree (J48) and k-NN (k = 1 and k = 3). Support Vector Machines (SVM) as the classification algorithm, since they have been proven to work well for highskew text classification tasks. Naïve Bayes requires little data for training, can produce maximum results. Besides classification of 2 classes conducted Kelly Reynolds, in this research will be classified 4 classes and 11 classes for using recommendations according to results of classification.

2. SYSTEM ANALYSIS

2.1.1 HARDWARE REQUIREMENTS:

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- System : Pentium IV 2.4 GHz.
- Hard Disk : 100 GB.
- Monitor : 15 VGA Color.
- Mouse : Logitech.
- RAM : 1 GB.

2.1.2 SOFTWARE REQUIREMENTS:

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

- Operating system : Windows XP/7/10
- Coding Language : python
- Development Kit : pyqt
- Programming language : Python
- IDE : Anaconda prompt

2.2 SOFTWARE REQUIREMENT SPECIFICATION

2.2.1 Requirements Specification:

Requirement Specification provides a high secure storage to the web server efficiently. Software

requirements deal with software and hardware resources that need to be installed on a serve which provides optimal functioning for the application. These software and hardware requirements need to be installed before the packages are installed. These are the most common set of requirements defined by any operation system. These software and hardware requirements provide a compatible support to the operation system in developing an application.

2.2.2 FUNCTIONAL REQUIREMENTS:

The functional requirement refers to the system needs in an exceedingly computer code engineering method.

The key goal of determinant "functional requirements" in an exceedingly product style and implementation is to capture the desired behavior of a software package in terms of practicality and also the technology implementation of the business processes.

2.2.3 NON FUNCTIONAL REQUIREMENTS

All the other requirements which do not form a part of the above specification are categorized as Non-Functional needs. A system perhaps needed to gift the user with a show of the quantity of records during info. If the quantity must be updated in real time, the system architects should make sure that the system is capable of change the displayed record count at intervals associate tolerably short interval of the quantity of records dynamic. Comfortable network information measure may additionally be a non-functional requirement of a system.

The following are the features:

- Accessibility
- Availability
- Backup
- Certification
- Compliance

- Configuration Management
- Documentation
- Disaster Recovery
- Efficiency(resource consumption for given load)
- Interoperability

2.2.4 PERFORMANCE REQUIREMENTS:

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

3. TECHNOLOGIES USED

3.1. INTRODUCTION TO PYTHON:

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable:

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable:

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor:

Just about any text editor will suffice for creating Python script files.

You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word, or just about any word processor if you want to.

Difference between a script and a program:

Script: Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, where as the applications they control are traditionally compiled to native machine code.

Program: The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived(e.g., compiled)

Python

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages.Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-ish to interface with C++ (via SWIG)

• Great interactive environment

Python is 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. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

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, and Unix shell and other scripting languages.

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

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

3.2 Python Features

Python's features include -

- **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 portedto 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 scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.

- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

StandardDataTypes

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

PythonNumbers

Number data types store numeric values. Number objects are created when you assign a value to them

PythonStrings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

PythonLists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

PythonTuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

PythonDictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python

type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

3.3 Python libraries:

1. Requests. The most famous http library written by kenneth reitz. It's a must have for every python developer.

2. Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won't use any other.

3. wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

6. BeautifulSoup. I know it's slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

8. NumPy. How can we leave this very important library ? It provides some advance math functionalities to python.

9. SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.

12. Pyglet. A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made

13. pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.

14. pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.

15. Scapy. A packet sniffer and analyzer for python made in python.

16. pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. nltk. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But it's capacity is beyond that. Do check it out.

18. nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. IPython. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

• High quality plotting library.

Python class and objects

These are the building blocks of OOP class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently "the big thing" in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early '90s, I realized functions were just like subroutines so there wasn't much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you'll realize that it's actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it's not mandatory to use objects in your code in a way that works best; maybe you don't need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you've already seen, Python can do just fine with functions. Unlike languages such as Java, you aren't tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here's a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by subclassing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we've seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python's built-in operation methods.

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program's flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a KeyError exception.

• Searching a list for a non−existent value will raise a ValueError exception

. • Calling a non−existent method will raise an AttributeError exception.

• Referencing a non−existent variable will raise a NameError exception.

• Mixing datatypes without coercion will raise a TypeError exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing.

Because exceptions aren't supposed to happen very often, they aren't processed until they occur.

Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won't have to do this very often but it's nice to have the option when necessary.

However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

3.4 Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

Testing code

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

he entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function**.**

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carred out, the function can or can not return one or more values.There are three types offunctions in python: help(),min(),print().

Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and familiy name (surname).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.

Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names.

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data withoutrequiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX : Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

3.5 Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, templating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
- what is a web framework? is an in-depth explanation of what web frameworks are and their relation to web servers.

- Django vs Flash vs Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python’s web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but theoutput is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? is a language agnostic Reddit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

4. SYSTEM DESIGN & UML DIAGRAMS

4.1 SYSTEM DESIGN:

System design is transitioning from a user oriented document to programmers or database personnel. The design is a solution, how to approach the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, preparing input and output specification, details of implementation plan and preparing a logical design walkthrough.

In designing the software following principles are followed:

- **Modularity and partitioning**: Software is designed such that each system should consist of a hierarchy of modules and serve to partition into separate functions.
- **Coupling:** Modules should have little dependence on other modules of a system.
- **Cohesion:** Modules should be carried out in a single processing function.
- **Shared use:** Avoid duplication by allowing a single module to be called by others that need the function it provides.

4.2 SYSTEM ARCHITECTURE

The purpose of the design phase is to arrange an answer of the matter such as by the necessity document. This part is that the opening moves in moving the matter domain to the answer domain. The design phase satisfies the requirements of the system. The design of a system is probably the foremost crucial issue warm heartedness the standard of the software package. It's a serious impact on the later part, notably testing and maintenance.

The output of this part is that the style of the document. This document is analogous to a blueprint of answer and is employed later throughout implementation, testing and maintenance. The design activity is commonly divided into 2 separate phases System Design and Detailed Design.

System Design conjointly referred to as top-ranking style aims to spot the modules that ought to

be within the system, the specifications of those modules, and the way them move with one another to supply the specified results.

At the top of the system style all the main knowledge structures, file formats, output formats, and also the major modules within the system and their specifications square measure set. System design is that the method or art of process the design, components, modules, interfaces, and knowledge for a system to satisfy such as needs. Users will read it because the applicationof systems theory to development.

Detailed Design, the inner logic of every of the modules laid out in system design is determined. Throughout this part, the small print of the info of a module square measure sometimes laid out in a high-level style description language that is freelance of the target language within which the software package can eventually be enforced.

In system design the main target is on distinguishing the modules, whereas throughout careful style the main target is on planning the logic for every of the modules.

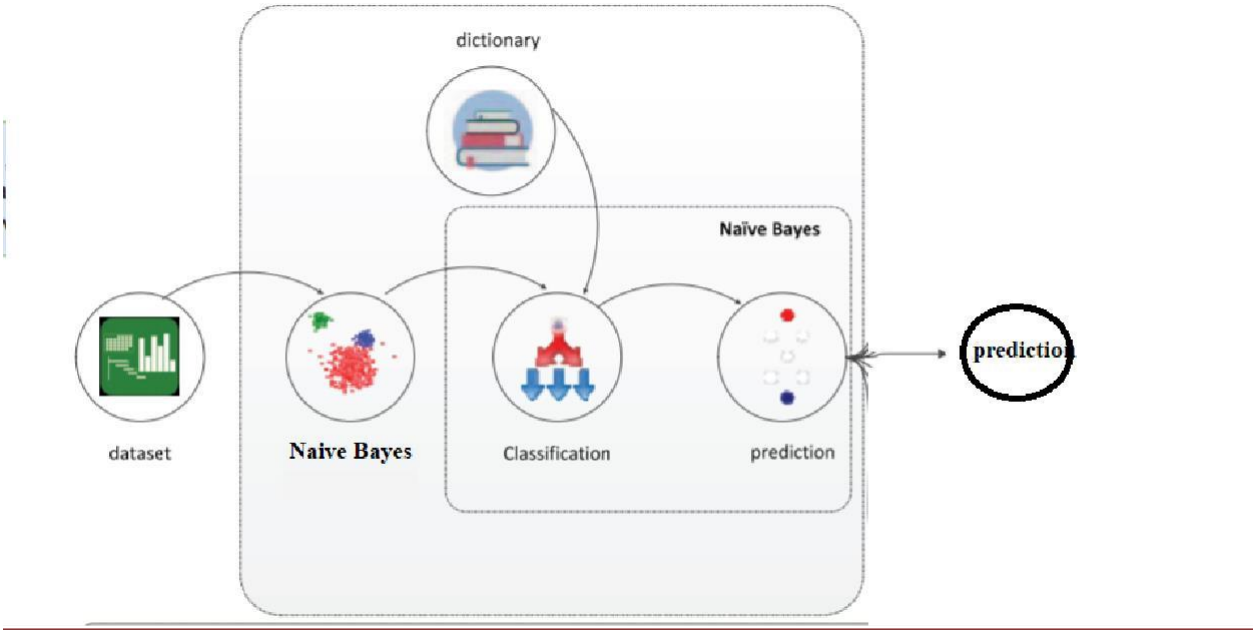


Figure 4.1: Architecture diagram

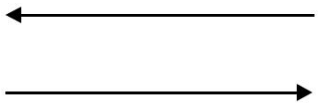
Here first we collect the data sets and process the data and we remove if there are any impurities in the data sets. Next the data is normalized if needed like it can be converted to smaller volume of data. Next the data is converted to supporting format. And then it is stored in the databases. Next the required method is applied. Now we get the final results.

4.3 DATA FLOW DIAGRAMS:

Data Flow Diagram can also be termed as bubble chart. It is a pictorial or graphical form, which can be applied to represent the input data to a system and multiple functions carried out on the data and the generated output by the system.

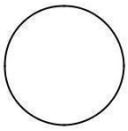
A graphical tool accustomed describe and analyze the instant of knowledge through a system manual or automatic together with the method, stores of knowledge, and delays withinthe system. The transformation of knowledge from input to output, through processes, is also delineate logically and severally of the physical elements related to the system. The DFD is also known as a data flow graph or a bubble chart.The BasicNotation used to create a DFD's are as follows:

- **Dataflow:**



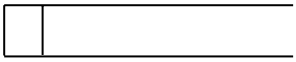
- **Process:**

.

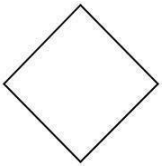


- **Source:**

- **Data Store:**



- **Rhombus**:



4.4 UML DIAGRAMS

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

User Model View

This view represents the system from the users perspective. The analysis representation describes a usage scenario from the end-users perspective.

Structural Model view

In this model the data and functionality are arrived from inside the system. This model view models the static structures.

Behavioral Model View

It represents the dynamic parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.

4.3.1 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

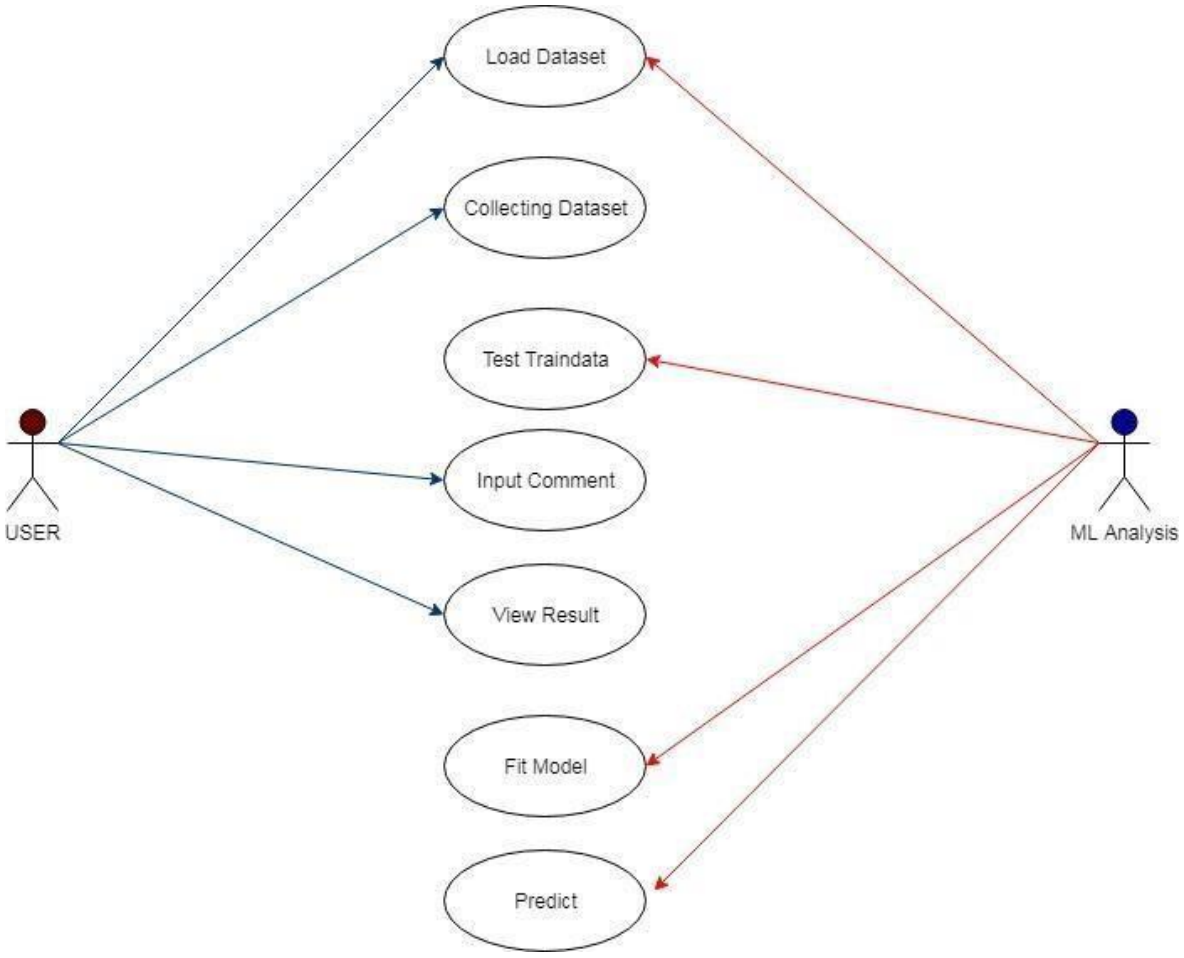


Figure 4.3.1 Use Case Diagram

4.3.2 CLASS DIAGRAM

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A class with three sections, in the diagram, classesis represented with boxes which contain three parts:

The upper part holds the name of the class

The middle part contains the attributes of the class

The bottom part gives the methods or operations the class can take or undertake.

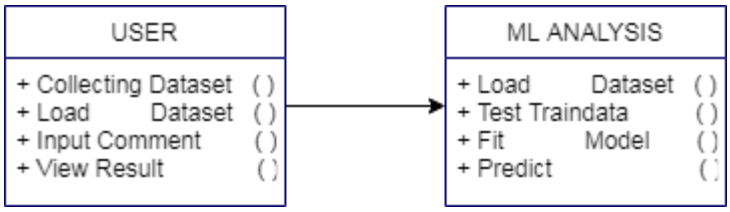


Figure 4.3.2: Class Diagram.

4.3.3 SEQUENCEDIAGRAM

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. 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. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

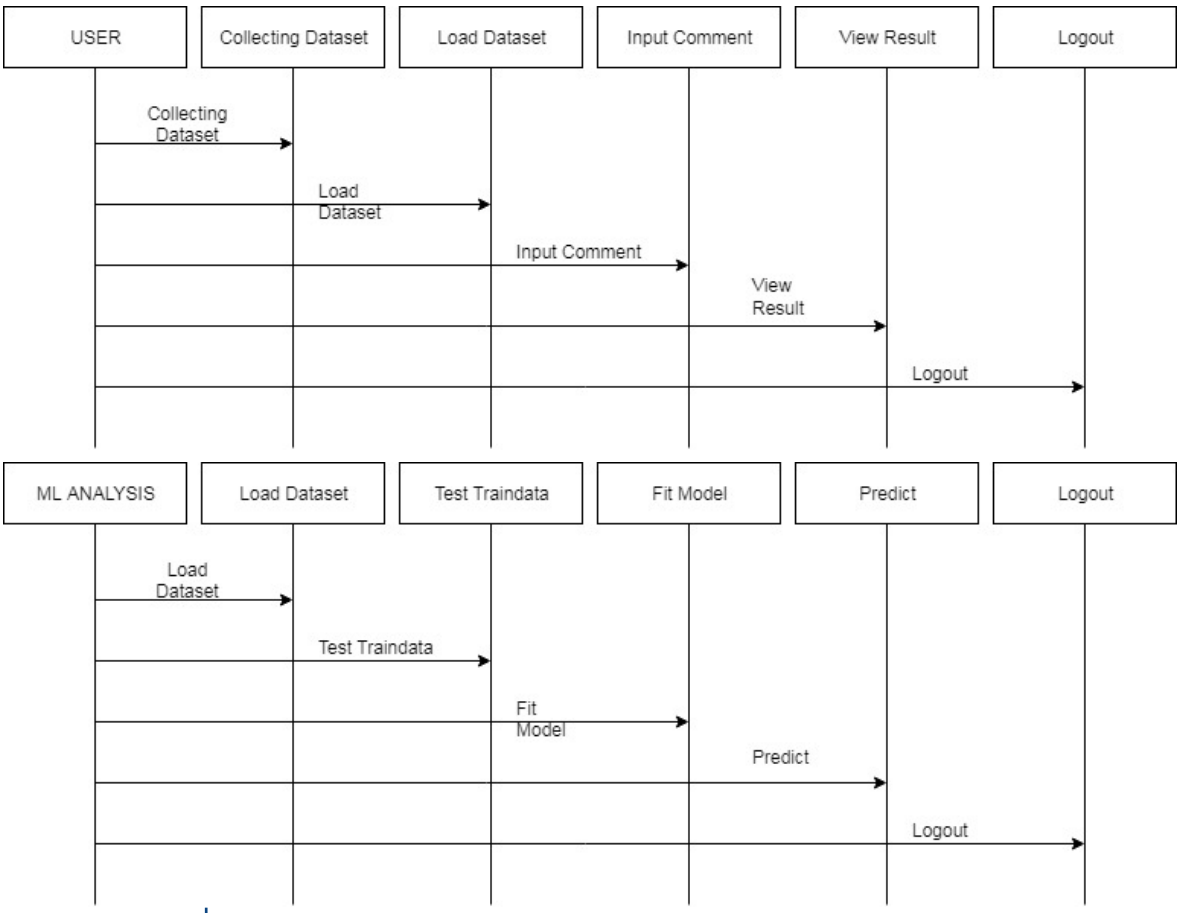


Figure 4.3.3: Sequence diagram

4.3.4 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

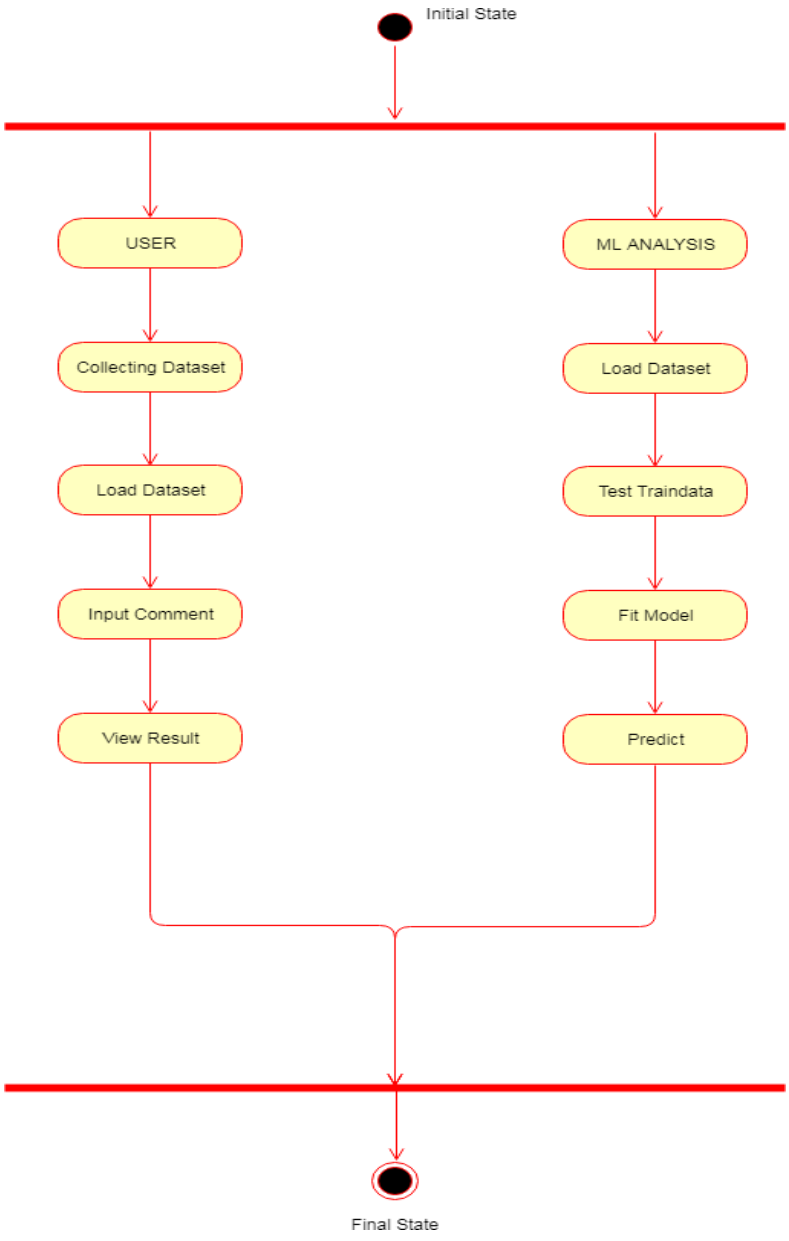


Figure 4.3.4: Activity Diagram

4.3.4 STATECHART DIAGRAM

Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

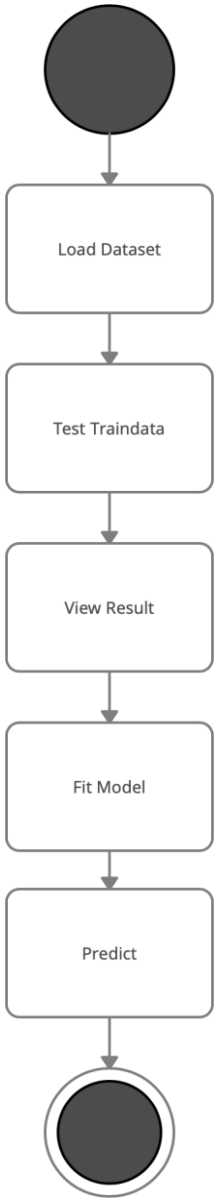


Figure 4.3.5: Statechart Diagram

5.INPUT AND OUTPUT DESIGN

5.1. INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

5.1.1 INPUT STAGES:

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

5.1.2 INPUT TYPES:

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.

- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

INPUTMEDIA:

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

5.2 OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization

- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

OUTPUT DEFINITION:

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

6.IMPLEMENTATION

Module 1:

```
import pandas as pd
import numpy as np
# ML Packages For Vectorization of Text For Feature Extraction
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
# Visualization Packages
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model


def prediction_result(tweet):
    check = tweet
    df = pd.read_csv("new_data.csv")
    df_data    =    df[["tweet","label"]]
    df_data.columns
    df_x = df_data['tweet']
    df_y = df_data['label']


    # #### Feature Extraction From Text
    # + CountVectorizer
```

```
    # + TfidfVectorizer


    # Extract Feature With CountVectorizer

    corpus = df_x

    #print(corpus)

    cv = CountVectorizer()

    X = cv.fit_transform(corpus) # Fit the Data

    #print(X)


    X.toarray()


    # get the feature names

    cv.get_feature_names()


    from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(X , df_y, test_size=0.33,
random_state=42)

    X_train

    # Naive Bayes Classifier

    from sklearn.naive_bayes import MultinomialNB

    #clf = MultinomialNB()

    clf=linear_model.LogisticRegression(fit_intercept=False)

    clf.fit(X_train,y_train)
```

```
clf.score(X_test,y_test)

# Accuracy of our Model
#print("Accuracy of Model",clf.score(X_test,y_test)*100,"%")

### Save the Model

import pickle

naivebayesML = open("Cyberbull.pkl","wb")

pickle.dump(clf,naivebayesML)

naivebayesML.close()

# Load the model

ytb_model = open("Cyberbull.pkl","rb")

new_model = pickle.load(ytb_model)

new_model
```

```
    # Sample Prediciton 3
    comment2 = [check]
    vect = cv.transform(comment2).toarray()
    #new_model.predict(vect)



    if new_model.predict(vect) == 1:
        result = True
        #print("not bullying")
    else:
        result = False
       # print("bullying")
    return result
```

Client Module:

```
#!/usr/bin/env python3
"""Script for Tkinter GUI chat client."""
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread
import tkinter
```

```
def receive():
    """Handles receiving of messages."""
    while True:
        try:
            msg = client_socket.recv(BUFSIZ).decode("utf8")
            msg_list.insert(tkinter.END, msg)
        except OSError:  # Possibly client has left the chat.
            break


def send(event=None): # event is passed by binders.
    """Handles sending of messages."""
    msg = my_msg.get()
    my_msg.set("")  # Clears input field.
    client_socket.send(bytes(msg, "utf8"))
    if msg == "{quit}":
        client_socket.close()
        top.quit()


def on_closing(event=None):
    """This function is to be called when the window is closed."""
    my_msg.set("{quit}")
```

```
    send()


top = tkinter.Tk()

top.title("cyber Bullying messenger")


messages_frame = tkinter.Frame(top)

my_msg = tkinter.StringVar()  # For the messages to be sent.

my_msg.set("")

scrollbar = tkinter.Scrollbar(messages_frame)  # To navigate through past messages.

# Following will contain the messages.

msg_list = tkinter.Listbox(messages_frame, height=35, width=80,
yscrollcommand=scrollbar.set)

scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)

msg_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)

msg_list.pack()

messages_frame.pack()


entry_field = tkinter.Entry(top, textvariable=my_msg)

entry_field.bind("<Return>", send)

entry_field.pack()

send_button = tkinter.Button(top, text="Send", command=send)

send_button.pack()
```

```
top.protocol("WM_DELETE_WINDOW", on_closing)


#----Now comes the sockets part----
HOST = input('Enter host: ')
PORT = input('Enter port: ')
if not PORT:
    PORT = 33000
else:
    PORT = int(PORT)

BUFSIZ = 1024
ADDR = (HOST, PORT)

client_socket = socket(AF_INET, SOCK_STREAM)
client_socket.connect(ADDR)

receive_thread = Thread(target=receive)
receive_thread.start()
tkinter.mainloop()  # Starts GUI execution.
```

Server Module:

```
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread
```

```
import string
import pickle
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
import numpy as np
from clas_new import prediction_result

# from pyserve import *

# print(msg)

def accept_incoming_connections():
    """Sets up handling for incoming clients."""
    while True:
        client, client_address = SERVER.accept()
        print("%s:%s has connected." % client_address)
        client.send(bytes("Type your name and press enter!", "utf8"))
        addresses[client] = client_address
        Thread(target=handle_client, args=(client,)).start()


def handle_client(client):  # Takes client socket as argument.
    """Handles a single client connection."""
```

```
name = client.recv(BUFSIZ).decode("utf8")
welcome = 'Welcome %s! If you ever want to quit, type {quit} to exit.' % name
client.send(bytes(welcome, "utf8"))
msg = "%s has joined the chat!" % name
# print(msg)
broadcast(bytes(msg, "utf8"))
clients[client] = name

while True:
    msg = client.recv(BUFSIZ)
    msgString = msg.decode('utf-8')
    print (msgString)
    print(str(msgString.split(" ")))
    print("clients",clients[client])
    bullying = False

    #for word in msgString.split(" "):
        #if word in badwords:

    # Load the model
    Y_predict=prediction_result(msg)
            #print(Y_predict)
```

```
        print ("bullying")
        if Y_predict == True:
                # broadcast(bytes(msg, "utf8"))
            broadcast(bytes("%s, Stop bullying people and behave decently. If you do this again we will block you." % name, "utf8"))
            bullying = True
            break
        if bullying == False:
            # api.update_status("\n Good job, you're not a bully! (I am a bot in testing, don't take this too seriously)", status.id)
            # broadcast(bytes(msg, "utf8"))
            print ("not bullying")
        if msg != bytes("{quit}", "utf8"):
            broadcast(msg, name+": ")
        else:
            client.send(bytes("{quit}", "utf8"))
            client.close()
            del clients[client]
            broadcast(bytes("%s has left the chat." % name, "utf8"))
            break



def handle_client(client):  # Takes client socket as argument.
    """Handles a single client connection."""
```

```
name = client.recv(BUFSIZ).decode("utf8")
welcome = 'Welcome %s! If you ever want to quit, type {quit} to exit.' % name
client.send(bytes(welcome, "utf8"))
msg = "%s has joined the chat!" % name
# print(msg)
broadcast(bytes(msg, "utf8"))
clients[client] = name

while True:
    msg = client.recv(BUFSIZ)
    msgString = msg.decode('utf-8')
    print (msgString)
    print(str(msgString.split(" ")))
    print("clients",clients[client])
    bullying = False

    for word in msgString.split(" "):
        if word:

            # Load the model
            Y_predict=prediction_result(msg)
            #print(Y_predict)
```

```
            print ("bullying")
            if Y_predict == True:
                # broadcast(bytes(msg, "utf8"))
                 broadcast(bytes("%s, Stop bullying people and behave decently. If you do this again we will block you." % name, "utf8"))
            bullying = True
            break
        if bullying == False:
            # api.update_status("\n Good job, you're not a bully! (I am a bot in testing, don't take this too seriously)", status.id)
            # broadcast(bytes(msg, "utf8"))
            print ("not bullying")
        if msg != bytes("{quit}", "utf8"):
            broadcast(msg, name+": ")
        else:
            client.send(bytes("{quit}", "utf8"))
            client.close()
            del clients[client]
            broadcast(bytes("%s has left the chat." % name, "utf8"))
            break


def broadcast(msg, prefix=""):  # prefix is for name identification.
    """Broadcasts a message to all the clients."""
```

```
    for sock in clients:
        sock.send(bytes(prefix, "utf8")+msg)



clients = {}
addresses = {}

HOST = ''
PORT = 33000
BUFSIZ = 1024
ADDR = (HOST, PORT)

SERVER = socket(AF_INET, SOCK_STREAM)
# SERVER.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)
SERVER.bind(ADDR)

if __name__  == "_main_":
    SERVER.listen(5)
    print("Waiting for connection...")
    ACCEPT_THREAD = Thread(target=accept_incoming_connections)
    ACCEPT_THREAD.start()
    ACCEPT_THREAD.join()
    SERVER.close()
```

7.TESTING

7.1 INTRODUCTION

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition.

7.2 STRATEGIC APPROACH OF SOFTWARE TESTING

The following is the description of the testing strategies, which were carried out during the testing period.

- **SYSTEM TESTING**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

- **MODULE TESTING**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the requiredfunction, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module

is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

- **INTEGRATION TESTING**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system

- **ACCEPTANCE TESTING**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

7.3 TEST CASES:

Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Upload the tasks dataset	Verify either file is loaded or not	If dataset is not uploaded	It cannot display the file loaded message	File is loaded which displays task waiting time	High	High

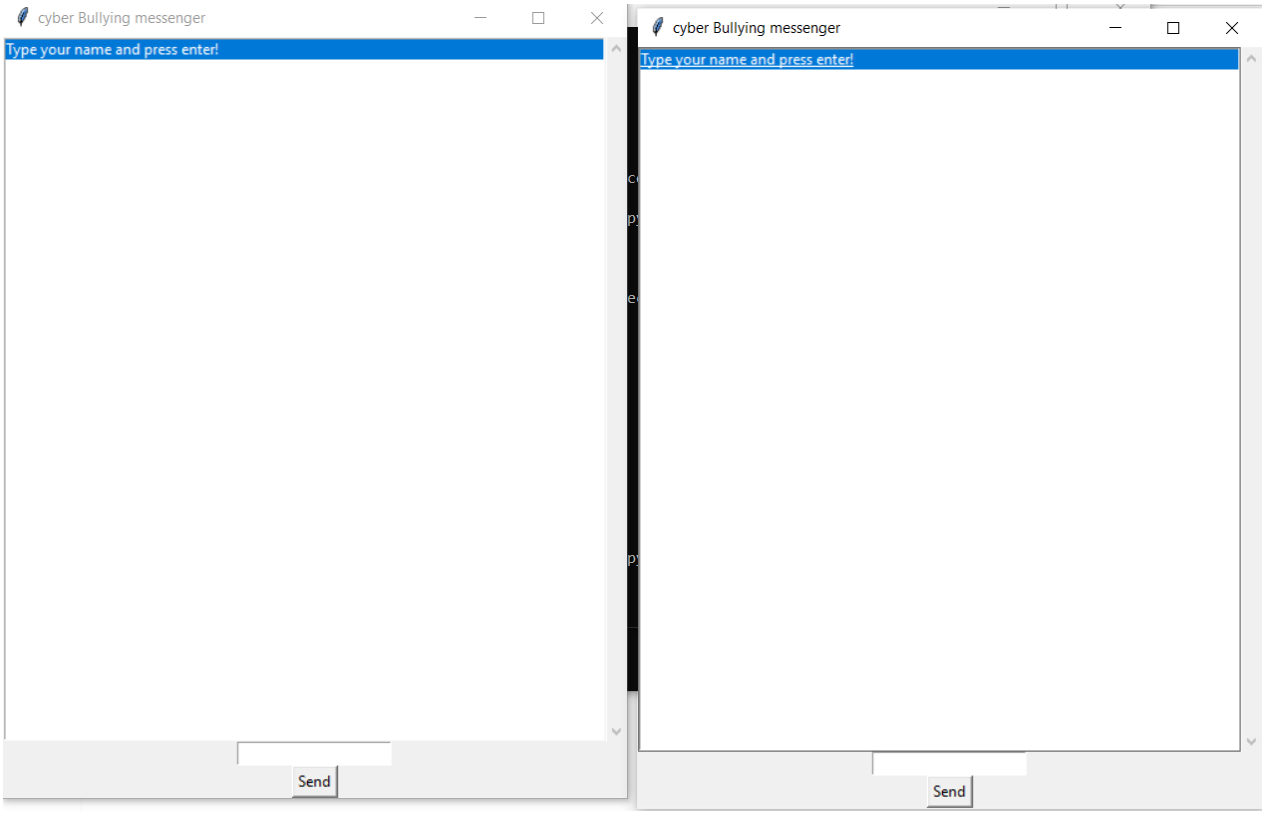
02	Send chat messages	Verify either chat messages are sending and received	If chat messages are note received	It cannot display chat messages	It can display chat messages	low	High
03	Preprocessing	Whether preprocessing on the dataset applied or not	If not applied	It cannot display the necessary data for further process	It can display the necessary data for further process	Medium	High
04	Prediction Navie bayes	Whether Prediction algorithm applied on the data or not	If not applied	Navie bayes model is created	Navie bayes is created	High	High
05	Prediction cyberbullying chats	Whether predicted data is displayed or not	If not displayed	It cannot view prediction containing bullying chats	It can view prediction containing bullying chats or not	High	High

06	Noisy Records Chart	Whether the graph is displayed or not	If graph is not displayed	It does not show the variations in between clean and noisy records	It shows the variations in between clean and noisy records	Low	Medium

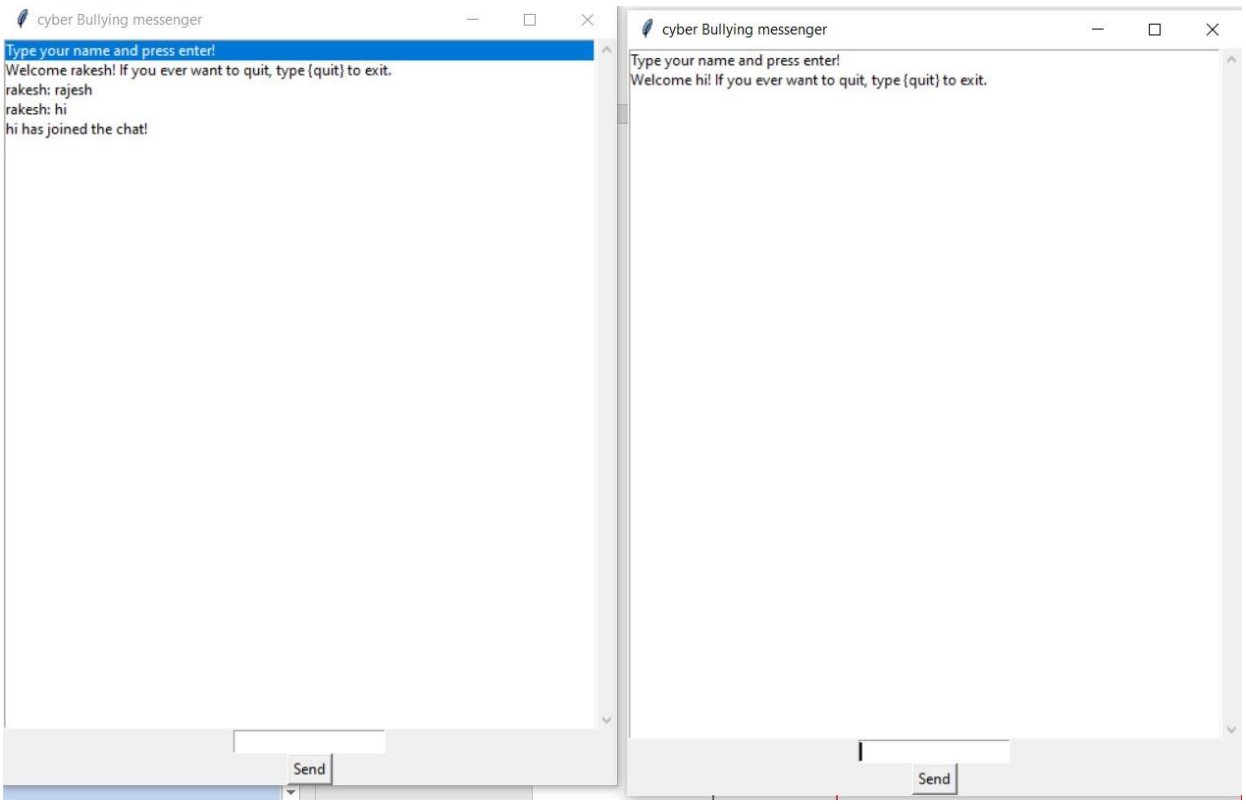
TABLE 7.1 TESTCASES

8.OUTPUT SCREENS

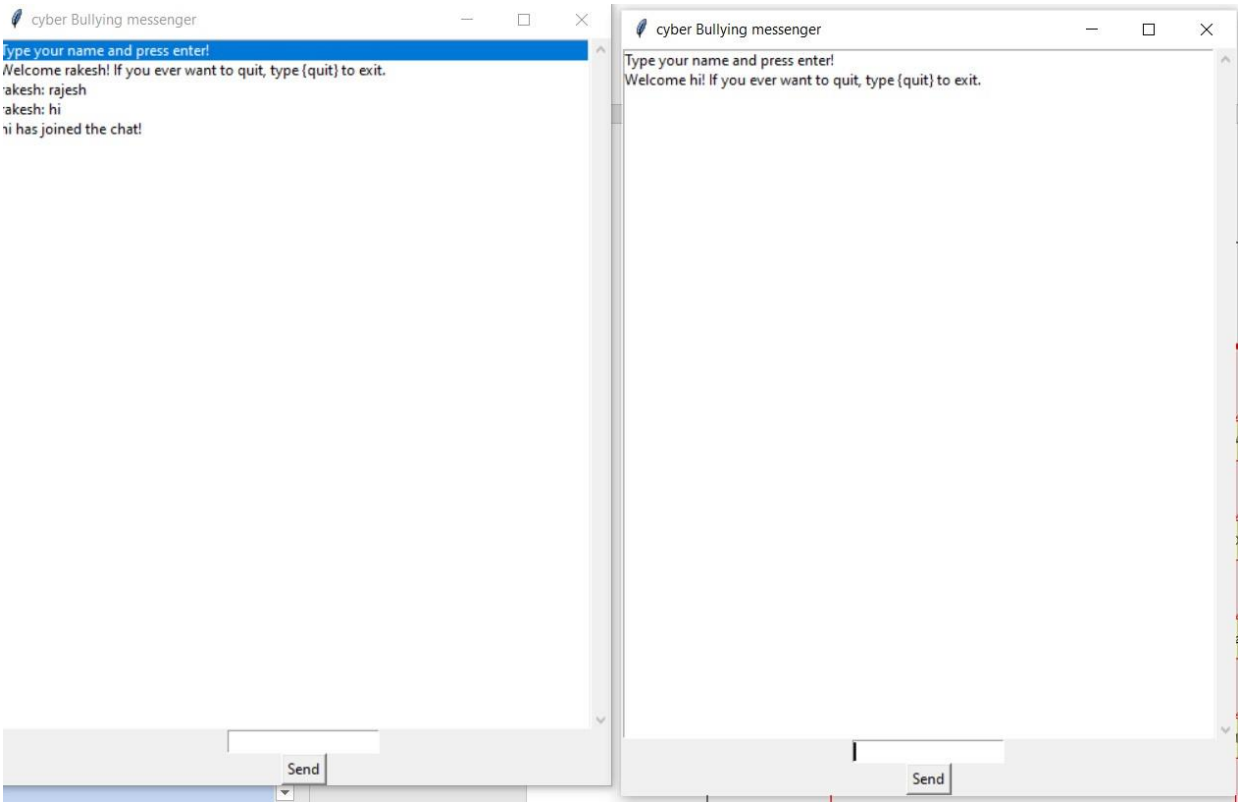
Chat application with 2 clients



Chat between 2 clients



When cyber bullying words are detected


cyber Bullying messenger
Type your name and press enter!
Welcome rakesh! If you ever want to quit, type {quit} to exit.
rakesh: rajesh
rakesh: hi
hi has joined the chat!
Send
cyber Bullying messenger
Type your name and press enter!
Welcome hi! If you ever want to quit, type {quit} to exit.
Send

9.CONCLUSION AND FUTURE SCOPE

The primarily study is to enhance the features of a Naïve Bayes classifier for extracting the words and generating model on text streaming. Moreover, a local optimum is guaranteed by our proposed method. The method was executed on CyberCrime Data, which is a manually labeled dataset, for 170,019 posts and Twitter web site for 467 million Twitter posts. .

The most optimal Naive Bayes kernel in classifying cyberbullying is the Poly kernel with a high average accuracy, because of the data used in this study are non-linear separable. Therefore, the optimal function for separating the sample into different classes is Naive Bayes with poly kernel.The application of n-gram may increase the accuracy level in cyberbullying classification, due tothe highest accuracy level at n-gram 5, the lowest accuracy set at n-gram 1.

Future Enhancements:

- However, in future we will study more on the streaming K-mean clustering using Apache Spark for increasing a performance of computation time and cost on the different data types from many data sets

10.BIBLIOGRAPHY

- R.M. Kowalski and S.P. Limber, “Psychological, Physical, and Academic Correlates of Cyberbullying and Traditional bullying,” J. Adolescent Health, 2013, vol. 53, no. 1, pp.513-520.
- Cyberbullying Research Center, 'Summary of Our Cyberbullying Research (2004-2016)', 2016. Available: http://cyberbullying.org/summary-of-our-cyberbullying-research.
- M. Dadvar, D. Trieschnigg, R. Ordelman, and F. De Jong, “Improving cyberbullying detection with user context,” Advances in Information Retrieval, Springer, 2013.
- D. Karthik, R. Roi, and L. Henry, “Modeling the detection of textual cyberbullying,” International Conference on Weblog and Social Media - Social Mobile Web Workshop, 2011.
- N. Vinita, L. Xue, and P. Chaoyi, “An Effective Approach for Cyberbullying Detection,” Communications in Information Science and Management Engineering, 2013, vol. 3.

  H. Homa, A. M. Sabrina, I. R. Rahat, H. Richard, L. Qin, and M. Shivakant, “Detection of Cyberbullying Incidents on the Instagram Social Network,” 2015.

    - python Complete Reference
    - python Script Programming by Yehuda Shiran
    - python server pages by Larne Pekowsley
    - python Server pages by Nick Todd
    - HTML Black Book by Holzner