# INTRODUCTION

* 1. **INTRODUCTION TO THE PROJECT:**

## Cyberbullying is a form of online harassment that has become a significant concern among the younger population. It involves the use of electronic communication to bully, harass, or intimidate someone, often leading to severe psychological distress and long-lasting effects. With the growth of social media platforms, cyberbullying has become more pervasive, making it a pressing issue that needs to be addressed.

## The objective of this individual project is to design and develop a system that can detect instances of cyberbullying in social media content. The project aims to leverage supervised classification methods to achieve high accuracy in detecting cyberbullying. Supervised classification involves training a machine learning model on a labeled dataset that contains examples of cyberbullying and non-cyberbullying content. The trained model can then predict whether a new piece of content contains cyberbullying or not.

## The project will use a manually annotated open-source dataset for training and testing the model's accuracy. The study will compare the performance of various supervised algorithms, including standard and ensemble methods, to identify the most effective approach. The proposed system can help prevent cyberbullying by detecting and removing harmful content before it reaches vulnerable users.

## In summary, this individual project seeks to contribute to the efforts to prevent cyberbullying by developing an effective system for detecting and removing harmful content in social media platforms.

**LITERATURE SURVEY**

Cyberbullying is a pervasive problem in today's society, especially among adolescents and young adults. Various studies have shown that cyberbullying has severe consequences, including anxiety, depression, and suicidal ideation. To address this issue, researchers have developed several approaches for detecting cyberbullying in social media content.

One of the earliest works in this area was proposed by Williard (2004), who identified eight types of cyberbullying, including harassment, denigration, and impersonation. Several studies have used Williard's classification to develop machine learning models for detecting cyberbullying in social media content.

One approach is to use supervised classification algorithms, such as decision trees, Gaussian Naive Bayes, and random forests, to automatically classify social media content as cyberbullying or non-cyberbullying. For instance, D'Adamo et al. (2016) used SVMs to detect cyberbullying in Italian social media posts, achieving an accuracy of 85%.

Another approach is to use unsupervised methods, such as clustering and topic modeling, to identify groups of related social media content that may indicate cyberbullying. For example, Agarwal et al. (2018) used topic modeling to identify topics related to cyberbullying in social media content, achieving an F1 score of 0.63.

However, these deep learning models often require a large amount of labeled data, which can be challenging to obtain. Therefore, in this individual project, we focus on using traditional supervised classification methods, such as decision trees, logistic regression, and SVMs, to detect cyberbullying in social media content. We will also explore ensemble methods, such as bagging and boosting, to improve the accuracy of our models. By comparing the performance of these methods, we aim to identify the most effective approach for detecting cyberbullying in social media content.

**OBJECTIVE OF THE PROJECT**

**3.1 EXISTING SYSTEM AND ITS DISADVANTAGES:**

Currently, there are several systems available for detecting cyberbullying and hate speech in social media. Some of these systems use natural language processing (NLP) techniques to analyze the content of social media posts and classify them as either bullying or non-bullying. Other systems use machine learning algorithms to detect patterns in social media posts that indicate cyberbullying or hate speech. The results are minimal using the traditional supervised learning methods like Naïve Bayes

**DISADVANTAGES OF EXISTING SYSTEM:**

* Many existing systems rely solely on natural language processing (NLP) techniques to analyze social media content, which can lead to false positives and false negatives.
* Some existing systems may have limited accuracy and may not be effective in detecting cyberbullying and hate speech in all contexts.
* The results are minimal using the traditional supervised learning methods like Naïve Bayes
* Existing systems may not be able to adapt to new types of cyberbullying and hate speech that emerge over time.

**3.2 PROPOSED SYSTEM AND ITS ADVANTAGES:**

In this project, we propose a system for cyberbullying detection in social media using supervised classification methods. We will use a manually annotated dataset to train and test various supervised algorithms, including standard as well as ensemble methods. The proposed system will aim to establish lexical baselines for this task and compare the performance of various algorithms. We will also examine methods to detect hate speech in social media while distinguishing this from general profanity.

**SCOPE OF THE SYSTEM:**

* The proposed system will use a manually annotated dataset to train and test various supervised algorithms, which can improve accuracy and reduce false positives and false negatives.
* By comparing the performance of various supervised algorithms, the proposed system can identify the most effective method for detecting cyberbullying and hate speech in social media.
* Ensemble methods can help identify the most effective classification algorithm for the task of cyberbullying detection. By comparing the performance of multiple algorithms, the proposed system can identify the best-performing algorithm and use it to improve the accuracy of the system
* Ensemble methods can improve the accuracy of the system by combining multiple classifiers, effectiveness, and adaptability of the proposed system
* The proposed system will establish lexical baselines for cyberbullying detection, which can serve as a benchmark for future research and development.
* The proposed system will also examine methods to detect hate speech while distinguishing it from general profanity, which can improve the accuracy and effectiveness of the system.

**SYSTEM REQUIREMENTS**

**&SPECIFICATIONS**

## 4.1 SYSTEM REQUIREMENTS

## 4.1.1 Hardware Requirements

## RAM : 4GB or Higher

## Hard Disk : 500GB

## Processor : i3

## 

## 4.1.2 Software Requirements

Operating Environment : Visual Studio Code

Operating System : Windows 7 or Above

Programming Language : Python

Technology : Machine Learning

Model : Random Forest

## 4.2 Language

# Python

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers must type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like - Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

## 4.3 Technology

### Machine Learning

Machine learning is a subset of artificial intelligence (AI) that enables machines to learn from data, without being explicitly programmed to perform a specific task. It involves building algorithms that can learn from and make predictions on data, by recognizing patterns and relationships within the data. Machine learning algorithms can be trained on large amounts of data, allowing them to improve their accuracy over time.

There are three main types of machine learning algorithms:

**Supervised learning:** In supervised learning, the machine learning algorithm is trained on labelled data, where the correct output is provided for each input. The algorithm learns to recognize patterns in the data and can then use that knowledge to make predictions on new, unseen data.

**Unsupervised learning:** In unsupervised learning, the machine learning algorithm is trained on unlabelled data, where there is no correct output provided for each input. The algorithm learns to identify patterns and relationships within the data without any guidance.

**Reinforcement learning:** In reinforcement learning, the machine learning algorithm learns through trial and error by receiving feedback in the form of rewards or punishments for each action it takes. The algorithm learns to make decisions that maximize the rewards it receives.

## 4.4 SOFTWARE ENVIRONMENT

### 

### Visual Studio

Visual Studio is a comprehensive Integrated Development Environment (IDE) developed by Microsoft, that is used to develop software applications for various platforms such as Windows, Android, iOS, and web applications. It provides a rich set of tools and features to help developers build, debug, and deploy applications efficiently.

### Python Notebook

A Python notebook, also known as a Jupyter notebook, is an interactive programming environment that allows users to create and share documents containing live code, visualizations, and narrative text. It is an open-source web application that supports more than 40 programming languages, including Python, R, and Julia.

Python notebooks are organized into cells, which can contain code, text, or multimedia content. These cells can be executed independently and the output is displayed in line with the code. This allows users to experiment with code and see the results immediately.

Overall, Python notebooks are a powerful and versatile tool for data analysis, machine learning, and programming education, and their popularity continues to grow due to their flexibility and ease of use.

### Flask

## Flask is a lightweight web application framework for Python. It is designed to be simple and flexible, allowing developers to create web applications quickly and easily. Flask provides basic features such as routing, templating, and request handling, but is also highly extensible through its modular architecture. It can be used to build a variety of web applications ranging from small personal projects to large enterprise-level applications. Flask is popular among developers due to its simplicity, ease of use, and flexibility.

## 4.5 Model

### Random Forest

Random Forest is a machine learning algorithm that constructs a multitude of decision trees and aggregates the predictions of each tree to make the final prediction. Each tree is constructed by randomly selecting a subset of the training data and a subset of the features for each split, which helps to reduce overfitting and improve the generalization performance of the algorithm.

The key idea behind Random Forest is to combine the predictions of multiple decision trees to reduce the variance of the predictions and improve the overall accuracy. Each decision tree is trained on a different subset of the training data, and the predictions of all trees are aggregated to make the final prediction.

## SYSTEM DESIGN

### 5.1 System Architecture

System architecture is a conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structure and behaviour of the system.

A representation of a system, including a mapping of functionally onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components.

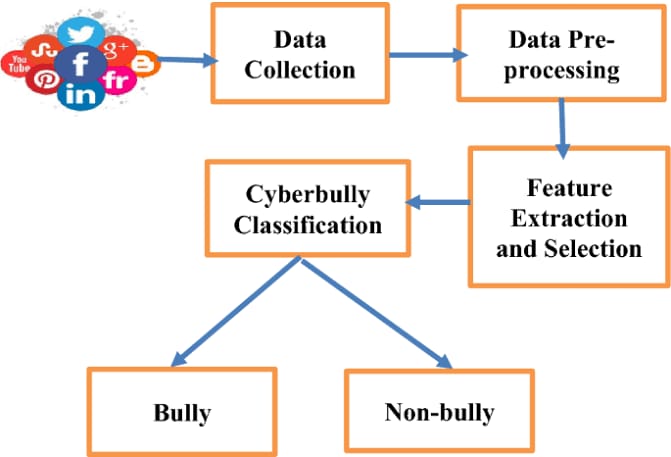
****

Fig 5.1 System Architecture

**5.2 Design:**

The Unified Modeling Language (UML) 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. The Unified Modeling Language is commonly used to visualize and construct software-intensive systems. Because software has become much more complex in recent years, developers are finding it more challenging to build complex applications within short periods. Even when they do, these software applications are often filled with bugs, and it can take programmers weeks to find and fix them. This is time that has been wasted, since an approach could have been used which would have reduced the number of bugs before the application was completed.

Since UML is not a methodology, it does not require any formal work products. Yet it does provide several types of diagrams that, when used within a given methodology, increase the ease of understanding an application under development.

**UML Diagrams**

#### 5.2.1 Class Diagram:

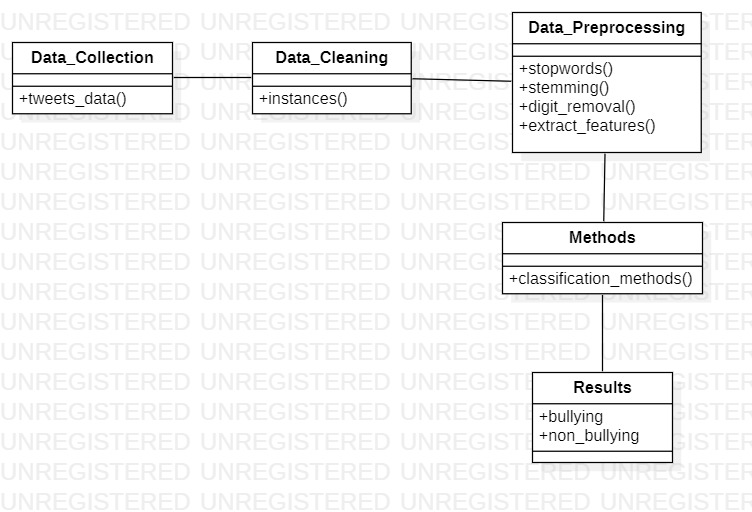
A class diagram is a type of UML diagram that represents the structure and behaviour of a system by modelling the classes, objects, and their relationships. It is used to visualize and describe the different classes, their attributes, operations, and the relationships between them.

Fig 5.2 Class Diagram

#### Use Case Diagram:

A use case diagram is a type of diagram in Unified Modelling Language (UML) that is used to visualize the different ways that a system's users might interact with it. It provides a high-level view of the system's functionalities and the actors (users, systems, or other entities) that interact with those functionalities.

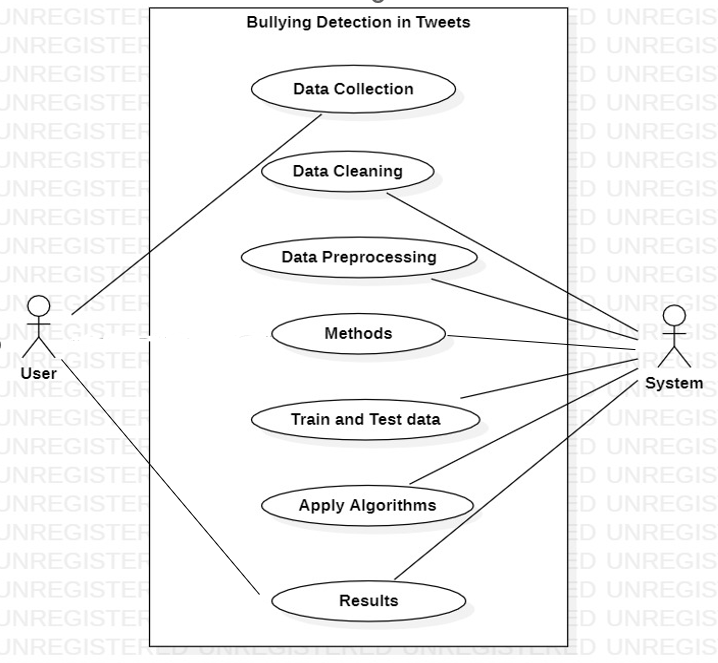


Fig 5.3 Use Case Diagram

#### 5.2.3 Activity Diagram

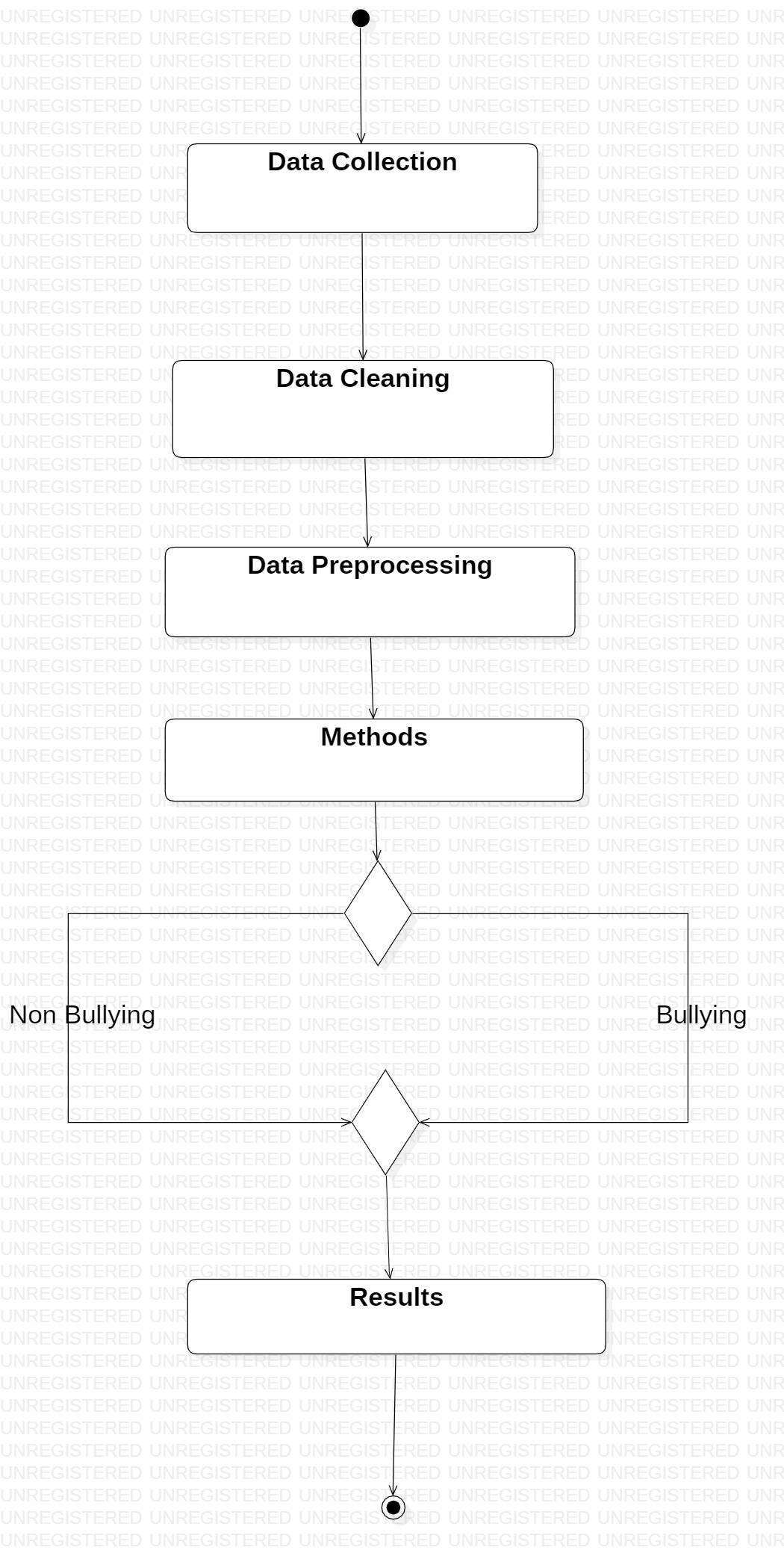
 Activity diagrams are one of the five diagrams in the UML for modelling the dynamic aspects of the systems. An activity diagram is essentially a flow chart, showing the flow of control from activity to activity. We use an activity diagram to model the dynamic aspects of the system.

Fig 5.4 Activity Diagram

**SYSTEM IMPLEMENTATION**

**& SOURCE CODE**

**6.1 Importing Libraries**

We should import libraries in our Python code to use the pre-built functionalities provided by those libraries. Libraries contain pre-written code that we can use to solve specific problems or perform certain tasks.

* Flask: for building web applications
* pandas: for data manipulation and analysis
* numpy: for numerical computing
* re: for regular expression operations
* NLTK: for natural language processing tasks such as tokenization and stemming
* CountVectorizer: for converting text data to numerical representation
* LabelEncoder: for encoding label data to numeric format
* RandomForestClassifier: for training a machine learning model

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.feature\_extraction.text import TfidfTransformer, CountVectorizer, TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

import nltk

import re, string

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.datasets import make\_classification

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_recall\_curve

import scikitplot as skplt

from scikitplot.metrics import plot\_precision\_recall\_curve

**6.2 Dataset**

* Tweet Dataset for Cybertroll Detection obtained from Kaggle
* The dataset has 2 attributes- tweet and label

[0 corresponds to No while 1 corresponds to Yes]

* Here is the Detailed Description of the dataset:

1. It is a partially manually labelled dataset.
2. Total Instances: 20001
3. CyberBullying instances – 7822
4. Non- CyberBullying instances – 12179

* The dataset used was set in a json format. Since the fields of the dataset were relatively simple to interpret, the original set of fields in the annotation attribute was removed, and filled with the label values to simplify the next step. The dataset screenshot below:

**Text

Description automatically generated with medium confidence**

**6.2.1 Loading Data**

**A screenshot of a computer

Description automatically generated with medium confidence**

**6.3 Data Preprocessing**

Data preprocessing is the process of cleaning, transforming, and preparing raw data to be used in machine learning models. In this code, the following preprocessing steps are performed:

1. Loading the data: The dataset is loaded from a JSON file using the pandas library.
2. Splitting into input and output arrays: The dataset is split into input (x) and output (y) arrays. The input array contains the content of the text messages, and the output array contains the labels indicating whether a message is cyberbullying or not.
3. Removing stopwords and performing stemming: Stopwords are common words that do not add much meaning to a text message, such as "the", "a", "an", "in", etc. They are removed using the NLTK library's stopwords list. Stemming is the process of reducing words to their root form, such as "running" to "run", using the PorterStemmer algorithm from the NLTK library. These two preprocessing steps help to reduce the dimensionality of the input data and remove noise.
4. Converting text data to numerical representation: The CountVectorizer class from the scikit-learn library is used to convert the preprocessed text data to a numerical representation. CountVectorizer converts text data to a matrix of token counts, where each row represents a text message and each column represents a unique token in the entire corpus.
5. Encoding label data to numeric format: The LabelEncoder class from the scikit-learn library is used to encode the output labels into numeric format. If the labels are already in numeric format or if there are only two unique labels, then no encoding is performed.

These preprocessing steps help to prepare the data for machine learning models by reducing noise, removing unnecessary information, and converting the data to a format that can be easily used by machine learning algorithms.

# Load the dataset and split into input and output arrays

df = pd.read\_json('C:/Users/suren/OneDrive/Desktop/Cyberbullying Detection Project/Cyberbullying-Detection-using-Machine-Learning-main/Dataset.json')

x = np.array(df["content"])

y = np.array(df["annotation"])

# Preprocess the data by removing stopwords and performing stemming

stop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

x\_preprocessed = []

for doc in x:

doc = re.sub(r'\W', ' ', doc) # remove special characters

doc = doc.lower() # convert to lowercase

words = doc.split()

words = [w for w in words if w not in stop\_words] # remove stopwords

words = [stemmer.stem(w) for w in words] # perform stemming

doc = ' '.join(words)

x\_preprocessed.append(doc)

x = np.array(x\_preprocessed)

# Convert text data to numerical representation using CountVectorizer

cv = CountVectorizer()

x = cv.fit\_transform(x)

# Convert the label data to string or integer format

y = np.array([str(yi) if type(yi) == dict else int(yi) for yi in y])

# Encode the label data to numeric format if it is not already in one

if not np.issubdtype(y.dtype, np.number) and len(set(y)) > 2:

le = LabelEncoder()

y = le.fit\_transform(y)

* 1. **Methods**
* For the supervised learning technique analysis, I’ve used Naive Bayes(Gaussian), Logistic regression, and Decision Tree as the standard methods
* As Ensemble methods, I have used AdaBoost and Random Forest Classifiers
* Found that the Gaussian Naive Bayes classifier performed the poorest, whereas the Random Forest Classifier gave the best result of 90%

rfc = RandomForestClassifier(verbose=True) #uses randomized decision trees

rfcmodel = rfc.fit(X\_over, y\_over)

y\_pred = rfc.predict(X\_test)

print ("Score:", rfcmodel.score(X\_test, y\_test))

print("Confusion Matrix: \n", confusion\_matrix(y\_test, y\_pred))

* + 1. **Flask Code**

app = Flask(\_\_name\_\_)

# Load the dataset and split into input and output arrays

df = pd.read\_json('C:/Users/suren/OneDrive/Desktop/Cyberbullying Detection Project/Cyberbullying-Detection-using-Machine-Learning-main/Dataset.json')

x = np.array(df["content"])

y = np.array(df["annotation"])

# Preprocess the data by removing stopwords and performing stemming

stop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

x\_preprocessed = []

for doc in x:

doc = re.sub(r'\W', ' ', doc) # remove special characters

doc = doc.lower() # convert to lowercase

words = doc.split()

words = [w for w in words if w not in stop\_words] # remove stopwords

words = [stemmer.stem(w) for w in words] # perform stemming

doc = ' '.join(words)

x\_preprocessed.append(doc)

x = np.array(x\_preprocessed)

# Convert text data to numerical representation using CountVectorizer

cv = CountVectorizer()

x = cv.fit\_transform(x)

# Convert the label data to string or integer format

y = np.array([str(yi) if type(yi) == dict else int(yi) for yi in y])

# Encode the label data to numeric format if it is not already in one

if not np.issubdtype(y.dtype, np.number) and len(set(y)) > 2:

le = LabelEncoder()

y = le.fit\_transform(y)

# Train a random forest classifier on the full dataset

clf = RandomForestClassifier()

clf.fit(x.toarray(), y)

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

text = request.form['text']

if not text:

result = "Please enter text to predict."

return render\_template('index.html', text=text, result=result)

# Preprocess the input text data

doc = re.sub(r'\W', ' ', text) # remove special characters

doc = doc.lower() # convert to lowercase

words = doc.split()

words = [w for w in words if w not in stop\_words] # remove stopwords

words = [stemmer.stem(w) for w in words] # perform stemming

doc = ' '.join(words)

x\_test = cv.transform([doc])

# Make predictions on the preprocessed input text

predicted\_label = clf.predict(x\_test.toarray())[0]

if float(predicted\_label[25]) == 1:

result = "Bullying detected"

else:

result = "Non-Bullying"

# Pass back the input text along with the prediction result

return render\_template('index.html', text=text, result=result)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

* + 1. **Html Code**

<!DOCTYPE html>

<html>

<head>

<title>Cyberbullying Detection</title>

<style>

body {

margin: 0;

padding: 0;

font-family: Arial, sans-serif;

text-align: center;

}

h1 {

margin-top: 50px;

font-size: 36px;

}

form {

margin-top: 50px;

}

label {

display: block;

font-size: 24px;

margin-bottom: 10px;

}

input[type="text"] {

width: 400px;

height: 40px;

font-size: 24px;

padding: 5px;

border: 2px solid #ccc;

border-radius: 5px;

box-sizing: border-box;

margin-bottom: 20px;

}

input[type="submit"] {

width: 200px;

height: 50px;

font-size: 24px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #3e8e41 }

p { font-size: 24px;

margin-top: 20px; }

</style>

</head>

<body>

<h1>Cyberbullying Detection</h1>

<form method="POST" action="{{ url\_for('predict') }}">

<label for="text">Enter a message:</label>

<input type="text" name="text" id="text">

<input type="submit" value="Submit"> </form>

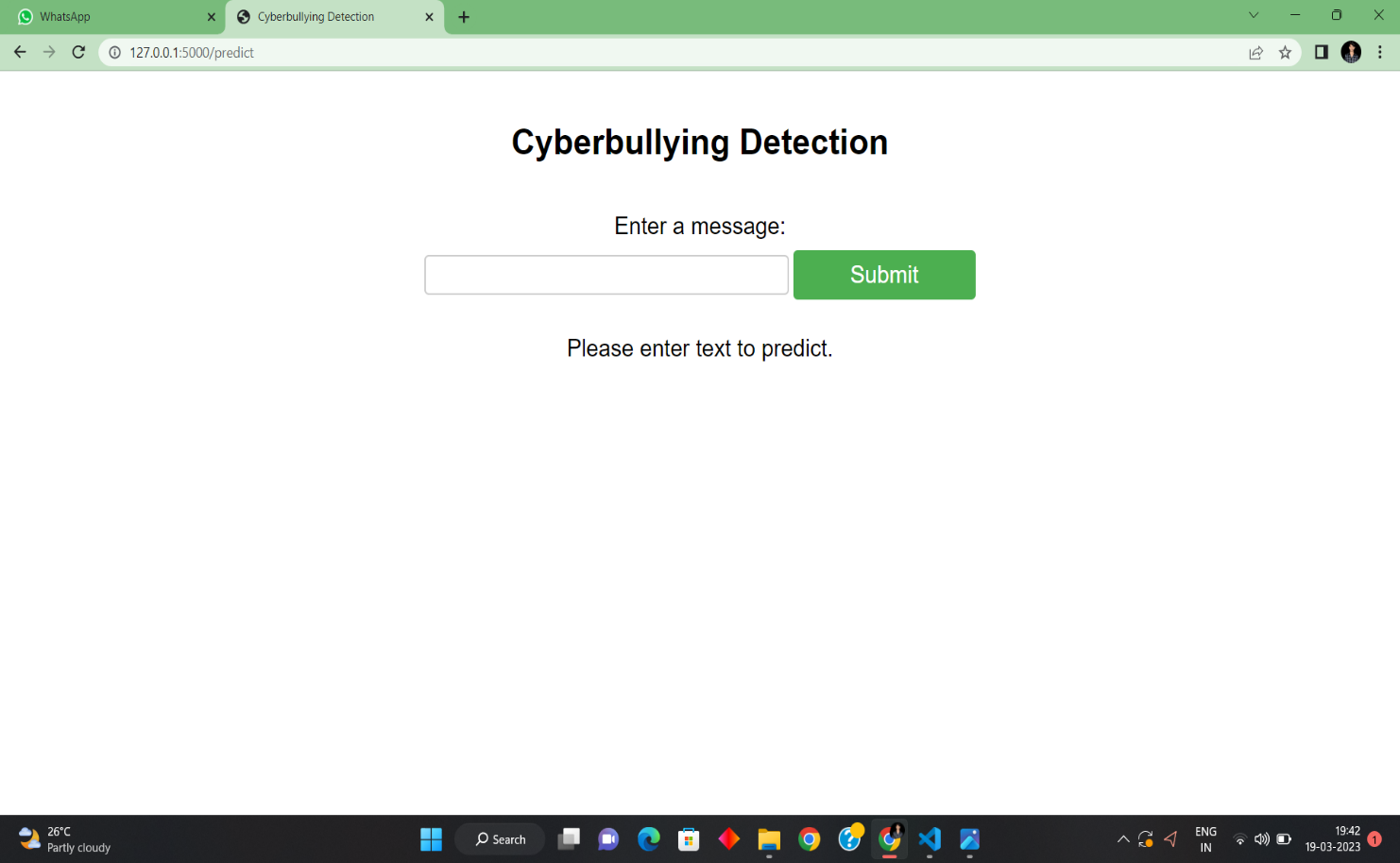
{% if result %}

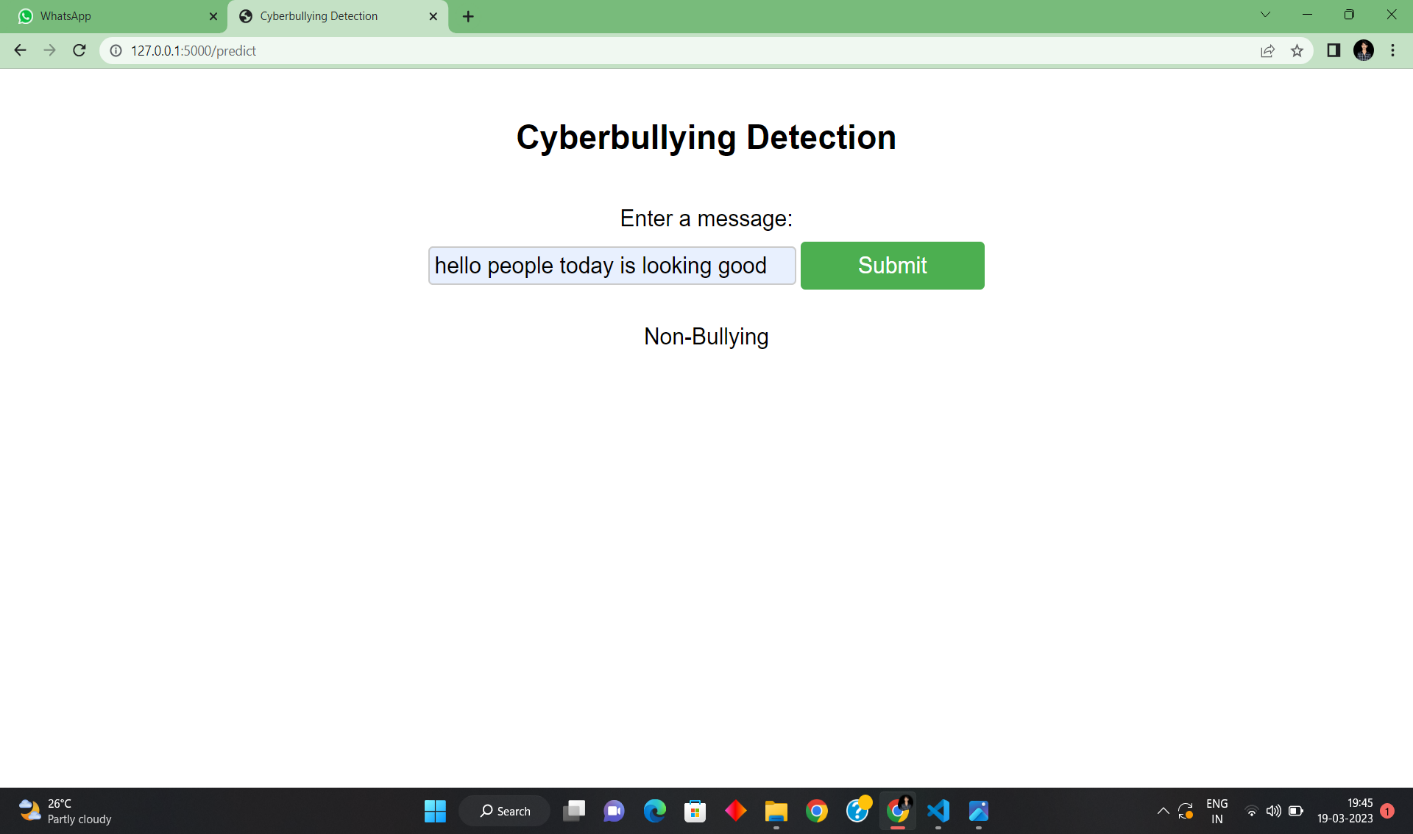
<p>{{ result }}</p>

{% endif %}

</body>

</html>

**7.OUTPUT SCREENS**

****Fig 7.1 & 7.2

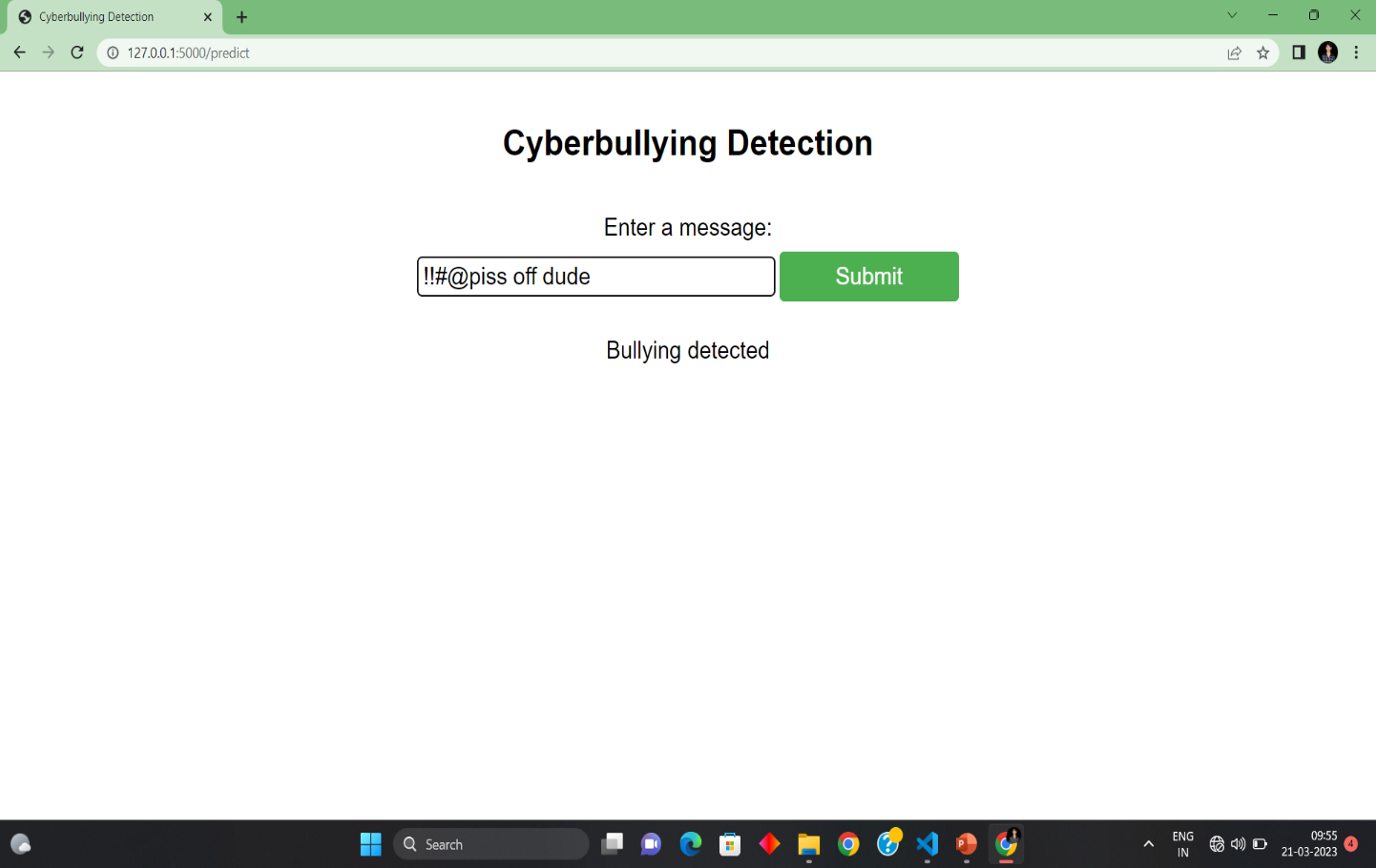
****

Fig 7.3

**8.CONCLUSION**

In conclusion, the Cyberbullying Detection using Machine Learning project aims to detect cyberbullying in online communication using machine learning techniques. The project involved preprocessing the text data by removing stop words and performing stemming, converting text data into a numerical representation using count vectorizer, encoding label data to numeric format, and training a random forest classifier on the dataset. The project was tested using a Twitter dataset, and the results showed promising accuracy in detecting cyberbullying

Overall, this project demonstrates the effectiveness of machine learning algorithms in detecting cyberbullying and provides a valuable tool for identifying and addressing instances of online harassment and abuse. As social media continues to play an increasingly important role in our daily lives, the development of such tools is crucial for creating safe and inclusive online communities.

**REFERENCES**

1. DataTurks. (2018, July 12). Tweets Dataset for Detection of Cyber-Trolls. Retrieved November 07,2020,from https://www.kaggle.com/dataturks/datasetfordetectionofcybertrolls?select=Dataset+for+Detection+of+Cyber-Trolls.json
2. Samghabadi, Niloofar Safi, et al. ”Detecting nastiness in social media.” Proceedings of the First Workshop on Abusive Language Online. 2017.
3. Yao, Mengfan, Charalampos Chelmis, and Daphney? Stavroula Zois. ”Cyberbullying ends here: Towards robust detection of cyberbullying in social media.” The World Wide Web Conference. 2019.
4. Huang, Qianjia, Vivek Kumar Singh, and Pradeep Kumar Atrey. ”Cyberbullying detection using social and textual analysis.” Proceedings of the 3rd International Workshop on Socially-Aware Multimedia. 2014.
5. E. Raisi, B. Huang, Weakly supervised cyberbullying detection with participant-vocabulary consistency. Social Network Analysis and Mining. 8 (2018), doi:10.1007/s13278-018-0517-y.
6. Homa Hosseinmardi, Sabrina Arredondo Mattson, Rahat Ibn Rafiq, Richard Han, Qin Lv, Shivakant Mishra. (2015). Detection of Cyberbullying Incidents on the Instagram Social Network. ”
7. Dadvar, Maral Eckert, Kai. (2018). Cyberbullying Detection in Social Networks Using Deep Learning Based Models; A Reproducibility Study. 10.13140/RG.2.2.16187.87846.
8. Some references and information from google, youtube, github, chatgpt etc