CYBERBULLYING DETECTION USING MACHINE LEARNING

A Project Progress Report

Submitted in partial fulfillment for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

N V S S AKHIL VARMA (S170166)

Under the Esteem Guidance of

Mrs. CH LAKSHMI BALA

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

RGUKT– SRIKAKULAM, ETCHERLA.

APRIL 2023

CERTIFICATE

This is to certify that the work entitled, **"Cyberbullying Detection using Machine Learning"** is the bonafide work of **N V S S AKHIL VARMA (ID NO: S170166)** carried out under my guidance and supervision for 4th year project of **Bachelor of Technology** in the department of Computer Science and Engineering under RGUKT IIIT SRIKAKULAM. This work is done during the academic session November 2022- March 2023, under our guidance.

Mrs. CH LAKSHMI BALA

Assistant professor,

Department of CSE

RGUKT-SRIKAKULAM

Mr. N SESHA KUMAR

Assistant Professor,

Department of CSE,

Head of the Department,

RGUKT-SRIKAKULAM.

DECLARATION

I **N V S S AKHIL VARMA (ID NO: S170166)** hereby declare that the project report entitled **"Cyberbullying Detection using Machine Learning"** done by me under the guidance of Mrs. CH LAKSHMI BALA, Assistant Professor is submitted for the partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering during the academic session November 2022- March 2023 at RGUKT-SRIKAKULAM.

I also declare that this project is a result of my own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Date: 29-03-2023

Place: NUZVID

N V S S AKHIL VARMA (S170166)

ACKNOWLEDGEMENT

I would like to express my profound gratitude and deep regards to our guide **Mrs. CH LAKSHMI BALA** for her exemplary guidance, monitoring and constant encouragement to me throughout the B. Tech course. I shall always cherish the time spent with her during the course of this work due to the invaluable knowledge gained in the field of reliability engineering.

I'm extremely grateful for the confidence bestowed in me and entrusting our project entitled **"Cyberbullying Detection using Machine Learning"**

I wish to extend my sincere thanks to **Mr. N SESHA KUMAR**, Head of the Computer Science and Engineering Department, for his constant encouragement throughout the project. I am also grateful to other members of the department without their support my work would not have been carried out so successfully. I thank one and all who have rendered help to me directly or indirectly in the completion of my thesis work.

ABSTRACT

The usage of social media has grown rapidly over time with the growth of the internet. Social media is an influential platform where we can find both good and bad things. This rapid increase in social connectivity results in online abuse, harassment, cyberbullying, cybercrime, and online trolling. We mainly focus on Cyberbullying, because it directly affects the mental health of a person, particularly this is a serious issue for women, in some cases, some women attempt suicide. Many incidents have recently occurred worldwide due to online harassment, such as sharing private chats, rumors, and sexual remarks. Therefore, the identification of bullying texts or messages on social media has gained a growing amount of attention among researchers. The main purpose of this project is to design and develop an effective technique to detect online abusive and bullying messages by merging natural language processing and machine learning. This project takes a text as input and checks whether it's cyberbullying comment or not.

TABLE OF CONTENTS

List of figures

CHAPTER 1
INTRODUCTION

1.1 Introduction:

Generally, Bullying is done intentionally, it is not an accidental incident, where the bully hurts a targeted person on a purpose. A bully repeats the same behavior, again and again, he/she gets happiness by bullying others. A bully always tries to bully weaker people. Cyberbullying means bullying others through social media, by sending abnormal messages, posting abnormal comments, etc. Social media has certain guidelines to follow because social media has to maintain a transparent platform for the people who use it for good purposes. Cyberbullying is of various types like bullying regarding color, body-shaming, religion, sexism, etc. Cyberbullying affects people directly or indirectly. In some cases, people attempt suicide due to cyberbullying. As per the NCRB (National Crime Records Bureau) report (2014, chapter 18), the number of cases reported under cyber crime was 9622 as compared to 5693 in 2013. This indicates a significant 69.0% increase over the previous year. Further, Cybercrime is a category of crime, which seems to be affecting a wide spread of ages including teenagers, which is alarming. Children get exposed to the Internet at a young age due to educational requirements, self-interest, or due to peer/social pressure. Mostly women are facing many problems in social media. Cyberbullying or cyber-harassment refers to an electronic method of bullying or harassment. **Bullying Text:** This type belongs to bully-type comments or harassment. For example, “go away bitch” is a bullying text or comment that we consider a negative comment. **Non-bullying Text:** These types of comments or posts are non-bullying or positive comments. For example, comment like “This photo is very beautiful” is positive and non-bullying comments.

1.2 Purpose of the project:

There is no system present in the real world to detect bullying texts. So , we wanted to build a model which detect those sort of texts.

This bullying has a physical and mental impact on the victim. The victims choose self-destructive acts like suicide because the trauma of cyberbullying which is hard to be endured. Thus, the identification and prevention of cyberbullying is important to protect teenagers.

Cyber bullying creates toxic environment in social media, detecting and preventing it can help users to provide a healthy environment.

1.3 Problem Statement:

The usage of social media has grown rapidly over time with the growth of the internet. Social media is an influential platform where we can find both good and bad things. This rapid increase in social connectivity results in online abuse, harassment, cyberbullying, cybercrime, and online trolling. We mainly focus on Cyberbullying, because it directly affects the mental health of a person, particularly this is a serious issue for women, in some cases, some women attempt suicide. Many incidents have recently occurred worldwide due to online harassment, such as sharing private chats, rumors, and sexual remarks. Therefore, the identification of bullying texts or messages on social media has gained a growing amount of attention among researchers. The main purpose of this project is to design and develop an effective technique to detect online abusive and bullying messages by merging natural language processing and machine learning. This project takes a text as input and checks whether it's cyberbullying comment or not.

1.4 Objective:

The objective of the project is to identify and prevent instances of cyberbullying on online platforms. Cyberbullying refers to the use of electronic communication technologies to harass, intimidate, or harm individuals or groups. It can take many forms, including posting derogatory comments or images, spreading rumours, or threatening others.

The development of a cyberbullying detection system involves using machine learning algorithms and natural language processing techniques to analyse online content and identify patterns of behaviour that are indicative of cyberbullying. By detecting and flagging instances of cyberbullying, such a system can help prevent harm to individuals and promote a safer and more respectful online environment.

Ultimately, the goal of a cyberbullying detection system is to reduce the incidence of cyberbullying and promote online safety, particularly for vulnerable populations such as children and teenagers who may be more susceptible to cyberbullying.

1.5 Applications:

Social media platforms:

These models can be used identify and block cyber bullying text in Social media platforms like Facebook , Instagram , twitter etc..

Email and Instant Messaging:

Companies and Organizations can use cyberbullying detection models to monitor employee email and instant messaging conversations to detect any instance of cyberbullying in the workplace

1.6 Limitations :

Apart from the well-established challenges that language-use poses (e.g., ambiguity, sarcasm), two factors in the event add further linguistic complexity, namely that of actor role and associated context. In contrast to tasks where adequate information is provided in the text of a single message alone, to completely map a cyberbullying event and pinpoint bully and victim implies some understanding of the dynamics between the involved actors and the concurrent textual interpretation.

We cannot determine whether it is bully or not by a single thread on social media, because there may also be a where two friends use abusing words and this cannot be considered as bullying

So, cyberbullying detection systems seem not applicable to real world situations

The model cannot detect bully words of the type which are posted each letter in a newline.

Example:(Nigga)

N

I

G

G

A

CHAPTER 2

LITERATURE SURVEY

2.1 Machine Learning:

Machine learning is a modern innovation that has enhanced many industrial and professional processes as well as our daily lives. It's a subset of artificial intelligence (AI), which focuses on using statistical techniques to build intelligent computer systems to learn from available data. With machine learning, computer systems can take all the customer data and utilize it. It operates on what's been programmed while also adjusting to new conditions or changes. Algorithms adapt to data, developing behaviors that were not programmed in advance. Learning to read and recognize context means a digital assistant could scan emails and extract essential information. Inherent in this learning is the ability to make predictions about future customer behaviors. This helps you understand your customers more intimately and not just be responsive, but proactive. Deep learning is a segment of machine learning. In essence, it's an artificial neural network with three or more layers. Neural networks with only one layer can make estimated predictions. The addition of more layers can assist with increasing optimization and accuracy. Machine learning is relevant in many fields, and industries, and has the capability to grow over time.

Real-time applications of Machine Learning:

Image recognition:

Image recognition is a well-known and widespread example of machine learning in the real world. It can identify an object as a digital image, based on the intensity of the pixels in black and white images or color images.

Speech recognition:

Machine learning can translate speech into text. Certain software applications can convert live voice and recorded speech into a text file. The speech can be segmented by intensities on time-frequency bands as well.

Real-world examples of speech recognition:

- Voice search
- Voice dialing
- Appliance control

Predictive analytics:

Machine learning can classify available data into groups, which are then defined by rules set by analysts. When the classification is complete, the analysts can calculate the probability of a fault. Real-world examples of predictive analytics:

- Predicting whether a transaction is fraudulent or legitimate
- Predictive analytics is one of the most promising examples of machine learning. It's applicable for everything; from product development to real estate pricing.

2.2 Random Forest:

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. **The reason for this wonderful effect is that the trees protect each other from their individual errors** (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

- There needs to be some actual signal in our features so that models built using those features do better than random guessing.
- The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

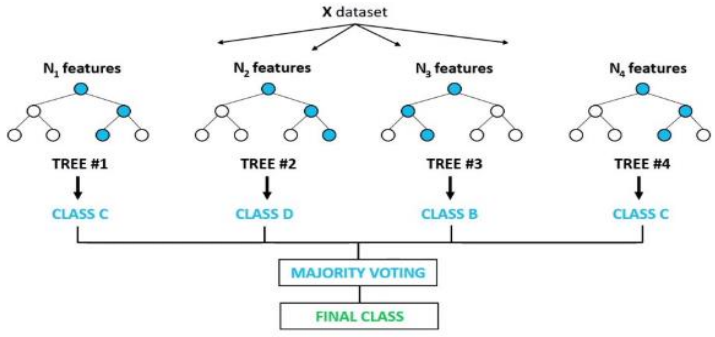


Fig-1: Structure and Working of Random Classifier

2.3 Naive Bayes Classifier:

The classification aims to assign fragments of text (i.e., documents) to classes by determining the probability that a document belongs to the class of other documents, having the same subject.

Each document consists of multiple words (i.e., terms), that contribute to an understanding of a document's contents. A class is a tag of one or multiple documents, referring to the same subject.

The labelling of documents with one of the existing classes is done by performing the statistical analysis, testing the hypothesis that a document's terms already occurred in other documents from a particular class. This increases the probability that a document is from the same class as the documents, already classified.

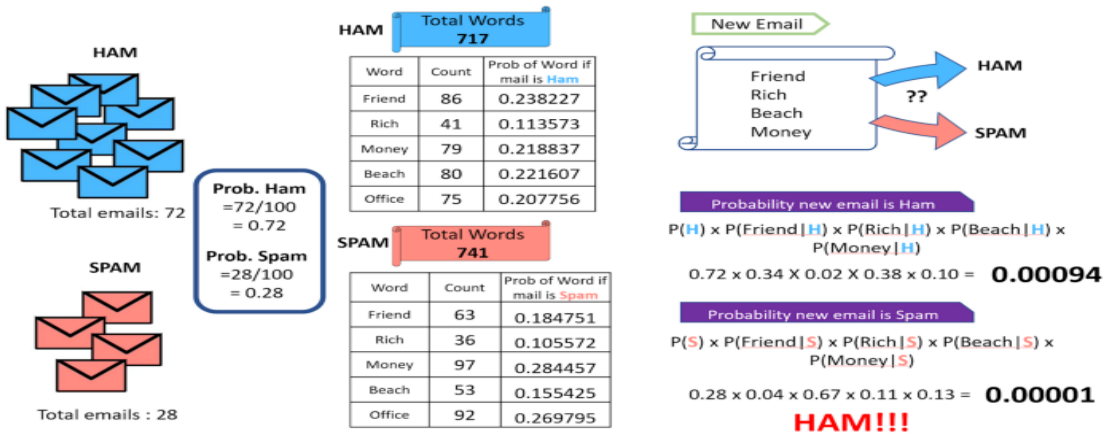


Fig-2: Naïve Bayes classifier Working

2.4 Support Vector Machine:

Support Vector machine, (SVM) can be used for both regression and classification tasks. But, it is widely used in classification objectives. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N – the number of features) that distinctly classifies the data points.

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Maximizing the margin distance provides some reinforcement so that future data points can be

classified with more confidence. A Support Vector Machine is a discriminative classifier formally defined by separating hyperplane. In other words given labelled training data, the algorithm outputs an optimal hyperplane which categorizes new examples. SVM can efficiently perform a non-linear classification, implicitly mapping their inputs into high dimensional feature spaces.

2.5 Decision Tree Classifier:

Decision Tree Classifier is a Supervised Machine Learning algorithm used for classification or regression. Decision Tree learns from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree the more complex the decision rules and fitter the model. It breaks down a dataset into smaller and smaller subsets while at the same time in associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has a two or more branches. Leaf node represents a classification or a decision. The top most decision node in the tree which corresponds to the best predicter called root node. Decision Trees can find both categorical and numerical data.

2.6 Multilayer Perceptron (MLP):

Multilayer Perceptron is the most fundamental type of neural network architecture when compared to other major types such as CNN, RNN, etc. In MLP perceptions (neurons) are stacked in multiple layers. Every node on each layer is connected to other nodes on the next layer. There is no connection between node within a single layer. An MLP is a fully connected neural network. In MLP data moves from input to the output through layers in one (forward) direction. The input neurons of the MLP do not perform any calculations and just output whatever they are given. We can say, they just hold the input data. The hidden layers are present between the input and output layers. As the number of hidden layers increases, neural network can model more complex relations in the data. However, it takes more time to train and also the model will tend to overfit the data. The output layer consists of output neurons that make the final predictions. We must determine the size of the output layer, that is the number of neurons in output layer. This value depends on the type of problem we want to solve.

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 Hardware components:

- Processor: 64-bit, quad-core, 2.5 GHz minimum per core
- RAM: 4 GB or more.
- HDD: 20 GB of available space or more.
- Keyboard: A standard keyboard.

3.2 Software components:

- **Python:** Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems.
- **NumPy:** NumPy is a Python library used for working with arrays.
- **SkLearn:** Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy, and Matplotlib.
- **NLTK:** NLTK(Natural Language Tool Kit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries
- **Pandas:** Pandas is mainly used for data analysis and associated manipulation of tabular data in Dataframes. Pandas allow importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel.

- **re:** A regular expression (re) is a set of characters with highly specialized syntax that we can use to find or match other characters or groups of characters. In short, regular expressions, or Regex, are widely used in the UNIX world. The re-module in Python gives full support for regular expressions of Pearl style. The re module raises the re.error exception whenever an error occurs while implementing or using a regular expression.

- **SeaBorn:** Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

- **StarUML:** StarUML is a software engineering tool for system modeling using the Unified Modeling Language, as well as Systems Modeling Language, and classical modeling notations. It is published by MKLabs and is available on Windows, Linux and MacOS.

- **Windows OS 64-bit.**

3.3 Functional requirements

- A Machine learning model that detects bullying text efficiently.

- A model to pre-process the dataset.

3.4 Non-Functional requirements

- Response time.

- Maintainability.

CHAPTER 4

PROPOSED MODEL AND FLOW OF THE PROJECT

4.1 Proposed model

Detection and classifying of bullying words involve analysing the text patterns. The analysis of text data is a popular research area with applications like text-classification. This project is focused on training the model to classify the input given by the user.

4.2 Flow of the project

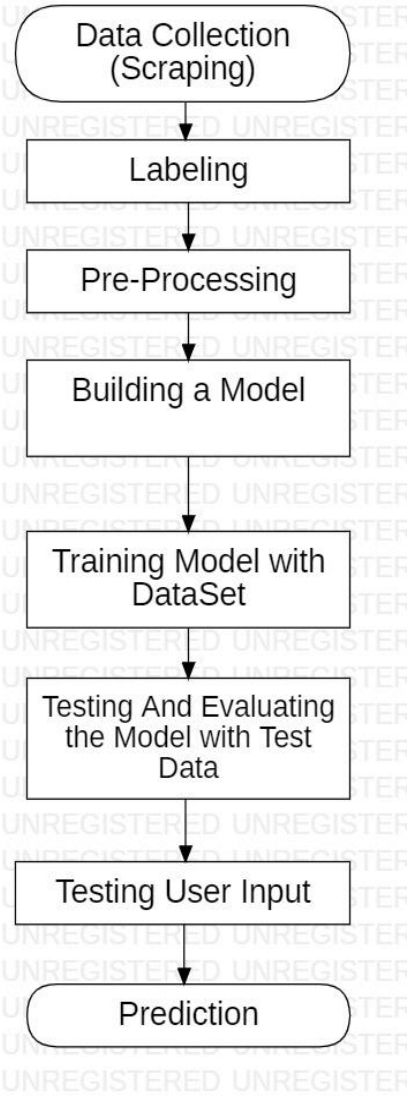


Fig-3: Flowchart for the Workflow of Project

4.2.1 What is a Confusion Matrix?

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:

Predicted \ Actual	Negative	Positive
Negative	True Negative	False Negative
Positive	False Positive	True Positive

Fig-4: Confusion Matrix

Understanding True Positive, True Negative, False Positive and False Negative in a Confusion Matrix

True Positive (TP): The predicted value matches the actual value. The actual value was positive and the model predicted a positive value

True Negative (TN): The predicted value matches the actual value. The actual value was negative and the model predicted a negative value

False Positive (FP): Type 1 error. The predicted value was falsely predicted. The actual value was negative but the model predicted a positive value Also known as the Type 1 error

False Negative (FN) : Type 2 error. The predicted value was falsely predicted. The actual value was positive but the model predicted a negative value. Also known as the Type 2 error

4.2.2 Data Collection (Scrapping):

Scrapping the data using python libraries (requests, scrappy etc) and API's. Data is collected from YouTube and Twitter. The scrapped data is saved into txt files then converted into csv files using Microsoft Excel.

4.2.3 Data Labelling:

Collected data is labelled using Label Studio. In Label Studio we create a new project and select text classification. Then we create the labels which we want to assign to the collected data. The we import our csv files into the data and assign label to them one by one. There are nearly 48,000 sentences in the collected data. After labelling all the collected data. We will download the final labelled file in csv format.

4.2.4 Data processing:

The data that we collected from different sources is noisy and have many unnecessary characters like URL's , '#' Tags , special characters, stopwords , numbers , punctuation's which will affect the model predictions. We will remove all this unnecessary data using regular expressions(re) module. Then we will convert this cleaned data into lower case and lemmatize it and then tokenize it. Then we will assign it into our data set. After that we will use TF-IDF Vectorizer to vectorize the data. After vectorizing the data we will split the data into training and testing sets. We took 70% of the data as training data and 30% data as testing data.

4.2.5 Building model:

In this we use Random forest Classifier and Naïve Bayes classifier methods. We use the Pre-defined models from the sklearn (Sci-Kit Learn) module in python. We initialize the two models into two variables and then we fit the text training data and label training data to the model for training.

4.3 Advantages and Disadvantages

4.3.1 Advantages

- Can identify the bullying words.
- Helps to maintain social media bullying free.
- Useful to avoid negativity among people.

4.3.2 Disadvantages

- By using this model we can predict whether the given text is bullying text or not. But predictions may not be accurate.
- The text may be viewed from different perspectives, so it is difficult to predict accurately.
- For example Two friends use those bullying words in a sarcastic way, but our model predicts those texts as bullying texts. This is a major limitation.

4.4 Applications

- This can be used in the Social media Platforms for filtering bullying comments.
- This system can be used social applications to identify users who are bullying others and block them or take actions on them.
- We can use this system on kids platforms to keep them safe from cyberbullying.

CHAPTER 5

IMPLEMENTATION

5.1 Random Forest Classifier

The random forests algorithm is a machine learning method that can be used for supervised learning tasks such as classification and regression. The algorithm works by constructing a set of decision trees trained on random subsets of features. In the case of classification, the output of a random forest model is the mode of the predicted classes across the decision trees.

Decision Trees: They are used for both regression and classification problems. They visually flow like trees, hence the name, and in the classification case, they start with the root of the tree and follow binary splits based on variable outcomes until a leaf node is reached and the final binary result is given.

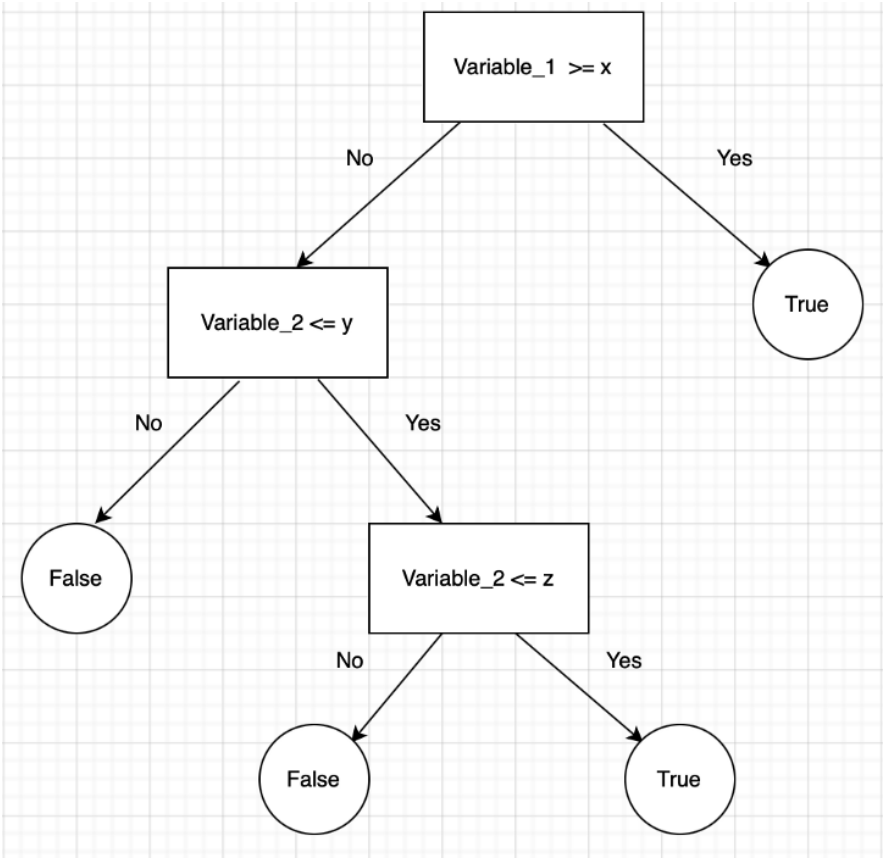


Fig-5: Decision Tree Structure

Ensemble learning: It is the process of using multiple models, trained over the same data, averaging the results of each model ultimately finding a more powerful prediction/classification result.

5.2 Multinominal Naïve Bayes:

A Naive Bayes classifier is a probabilistic machine learning model that's used for the classification task. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem: Using the Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is the presence of one particular feature does not affect the other. Hence it is called naive.

- When an assumption of independent predictors holds true, a Naive Bayes classifier performs better as compared to other models. It has a deeper network compared to the Inception V1 and V2 models, but its speed isn't compromised.
- Naive Bayes requires a small amount of training data to estimate the test data. So, the training period is less.
- Naive Bayes is also easy to implement.
- It doesn't require as much training data.
- It is highly scalable with the number of predictors and data points.

Performance Metrics:

Accuracy is measured as the total number of (TP + TN)/(All Cases), while a F1 score is calculated by 2*((precision*recall)/(precision + recall)), with precision = TP/(TP+FP), and recall = TP/(TP+FN).

5.3 Support Vector Machine (SVM):

Support Vector machine, (SVM) can be used for both regression and classification tasks. But, it is widely used in classification objectives. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N – the number of features) that distinctly classifies the data points.

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplane: A Hyperplane is a decision boundary that differentiates the two classes in SVM. A data point falling on either side of the hyperplane can be attributed to different classes. The dimension of the hyperplane depends on the number of input features in the dataset.

Support-Vectors: Support Vectors are the data points that are nearest to the hyperplane and affect the position and orientation of the hyperplane. We have to select a hyperplane, for which the margin, i.e., the distance between support vectors and hyperplane is maximum.

Maximum classification: The selected line must successfully segregate all the data points into respective classes.

Best Separation: We must choose a line that is perfectly able to separate data points.

5.4 Decision Tree Classifier:

Decision Tree Classifier is a Supervised Machine Learning algorithm used for classification or regression. Decision Tree learns from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree the more complex the decision rules and fitter the model.

It breaks down a dataset into smaller and smaller subsets while at the same time in associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has a two or more branches. Leaf node represents a classification or a decision. The top most decision node in the tree which corresponds to the best predicter called root node. Decision Trees can find both categorical and numerical data.

Decision Tree Classifier contains three steps:

- Grow the tree: Splitting the space by setting the rules.
- Prune the tree: Removing the unnecessary splits.
- Assign the class: Using the class with majority words as the prediction.

5.5 Multilayer Perceptron:

Multilayer Perceptron is the most fundamental type of neural network architecture when compared to other major types such as CNN, RNN, etc. In MLP perceptron (neurons) are stacked in multiple layers. Every node on each layer is connected to another nodes on the next layer. There is no connection

between nodes within a single layer. An MLP is a fully connected neural network. In MLP data moves from input to output through layers in one (forward) direction.

Starting with the input layer, MLP propagates data forward to the output layer. This step is forward propagation. Based on the output calculates the error. This error needs to be minimized. Back propagate the error. Find its derivative with respective each weight in the network and update the model. Repeat the same three steps over multiple epochs to learn ideal weights. Finally, the output is taken via a threshold function to obtain the predicted class labels.

5.6 Importing All Necessary Libraries:

```
# Importing all the necessary packages
import joblib
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
# from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import word_tokenize
import string  # used for preprocessing
from nltk.stem import WordNetLemmatizer
import re
import pandas as pd
import nltk
import numpy as np
from nltk.corpus import stopwords
lemmatizer = WordNetLemmatizer()
stop_list = set(stopwords.words('english'))
```

5.7 Reading and pre processing of data:

```
#Reading The CSV file and deleting empty rows if present
df = pd.read_csv("/content/cyberbullying_tweets.csv",engine="python")
# df.info()
# df.head()
df.dropna()
df.info()

def textLowerCase(text):
  return text.lower()
def removeUrls(text):
  newText = ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", text).split())
  return newText
def removeNumbers(text):
  result = re.sub(r'd+',",text)
  return result
def remove_punctuation(text):
  translator = str.maketrans(", ", string.punctuation)
  return text.translate(translator)
def tokenize(text):
  text = word_tokenize(text)
  return text
stop_words = set(stopwords.words('english'))
def removeStopWords(text):
  text = [value for value in text if not value in stop_words]
  return text
lemmatizer = WordNetLemmatizer()
def lemmatize(text):
  text = [lemmatizer.lemmatize(token) for token in text]
  return text
def preProcessingText(text):
  text = textLowerCase(text)
  text = removeUrls(text)
  text = removeNumbers(text)
  text = remove_punctuation(text)
  text = tokenize(text)
  text = removeStopWords(text)
  text = lemmatize(text)
  text = ' '.join(text)
  return text
preProcessedText = []
for text_data in df['tweet_text']:
  ProcessedText = preProcessingText(text_data)
  preProcessedText.append(ProcessedText)
df['tweet_text'] = preProcessedText
yComp = np.array(df['cyberbullying_type'])
```

5.7 Building Model:

```
# Models Initialization
#initialzing naïve bayes classifier
naiveBayesClassifier = MultinomialNB()

RandomClassifier = RandomForestClassifier(n_estimators=150)

dc_tree = tree.DecisionTreeClassifier(max_features=800)

mlp = MLPClassifier(alpha=1, max_iter=1000)


#Fitting models


# Training naïve Bayes  classifier with comments
naiveBayesClassifier.fit(X_train_comm ,y_train_comm)
y_pred_nb = naiveBayesClassifier.predict(X_test_comm)

# Training Random Forest classifier with comments
RandomClassifier.fit(X_train_comm, y_train_comm)
y_pred_comm = RandomClassifier.predict(X_test_comm)

# Training decision Tree classifier with comments
dc_tree.fit(X_train_comm, y_train_comm)
y_pred_dc = dc_tree.predict(X_test_comm)

# Training Multi Layer Perceptron  classifier with comments
mlp.fit(X_train_comm, y_train_comm)
y_pred_nnc = mlp.predict(X_test_comm)
```

CHAPTER 6

TESTING

6.1 Introduction:

Testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product:

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources.As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job oftesting is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones. Software testing can provide objective, independent information about the quality of software and risk of its failure.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs

6.2 Types of Testing:

Unit testing – Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code may require developing test driver modules or test harnesses.

As part of unit testing, we tested whether the individual Machine Learning models predicting correct results or not given an input.

Integration testing – Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules.

Once the unit testing on individual Machine Learning models is done, we combined all the models together to test whether they are producing correct results or not over an input given to those models. Here we combined the four models that we built as part our code.

6.3 Test Cases

```
def heatMaps(test , pred):
  array = confusion_matrix(test, pred)
  df_cm = pd.DataFrame(array, range(6), range(6))
  plt.figure(figsize=(10, 7))
  # sn.set(font_scale=1.4)  # for label size
  sn.heatmap(df_cm, annot=True, annot_kws={"size": 15})  # font size
  plt.show()
```

#Testing Random Forest Classifier

```
heatMaps(y_test_comm,yPred_randomClass)
```

#Testing Naïve Bayes Classifier

```
heatMaps(y_test_comm,y_pred_nb)
```

#Testing Decision Tree Classifier

```
heatMaps(y_test_comm,y_pred_dc)
```

#Testing Multi-layer Perceptron Classifier

```
heatMaps(y_test_comm,y_pred_nnc)
```

Testing the trained model to make predictions using the Input from the User:

Use the loaded model to make predictions

```
def testAllClassfiers(text):
    text = v.transform([text]).toarray()
    random_res = RandomClassifier.predict(text)
    naiveBayesRes =naiveBayesClassifier.predict(text)
    dcRes = dc_tree.predict(text)
    mlpRes = mlp.predict(text)
    print("Result of Random Forest model : ", random_res)
    print("Result of Naive Bayes Model : ", naiveBayesRes)
    print("Result of the Decision Tress MOdel : ", dcRes)
    print("Result of Neural Networks Model : ", mlpRes)

inputText = input("Enter text here :")
testAllClassfiers(inputText)
```

OUTPUT

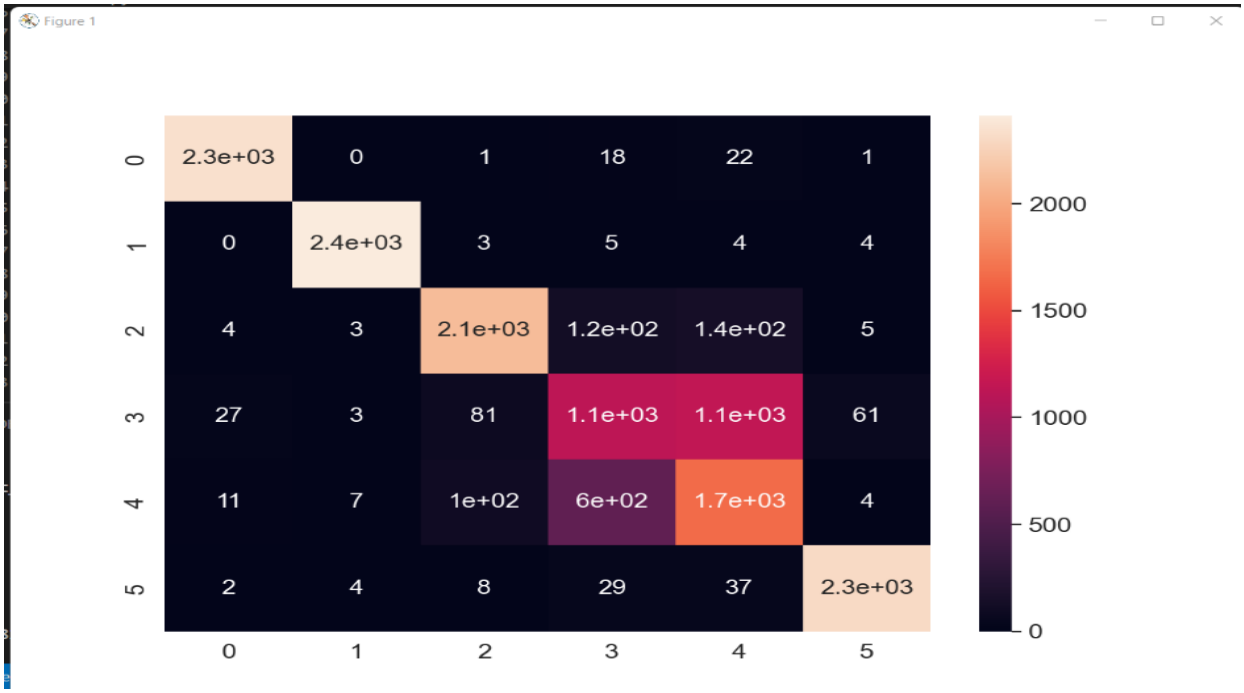


Fig-6: Random Forest Classifier Confusion Matrix Heat-Map

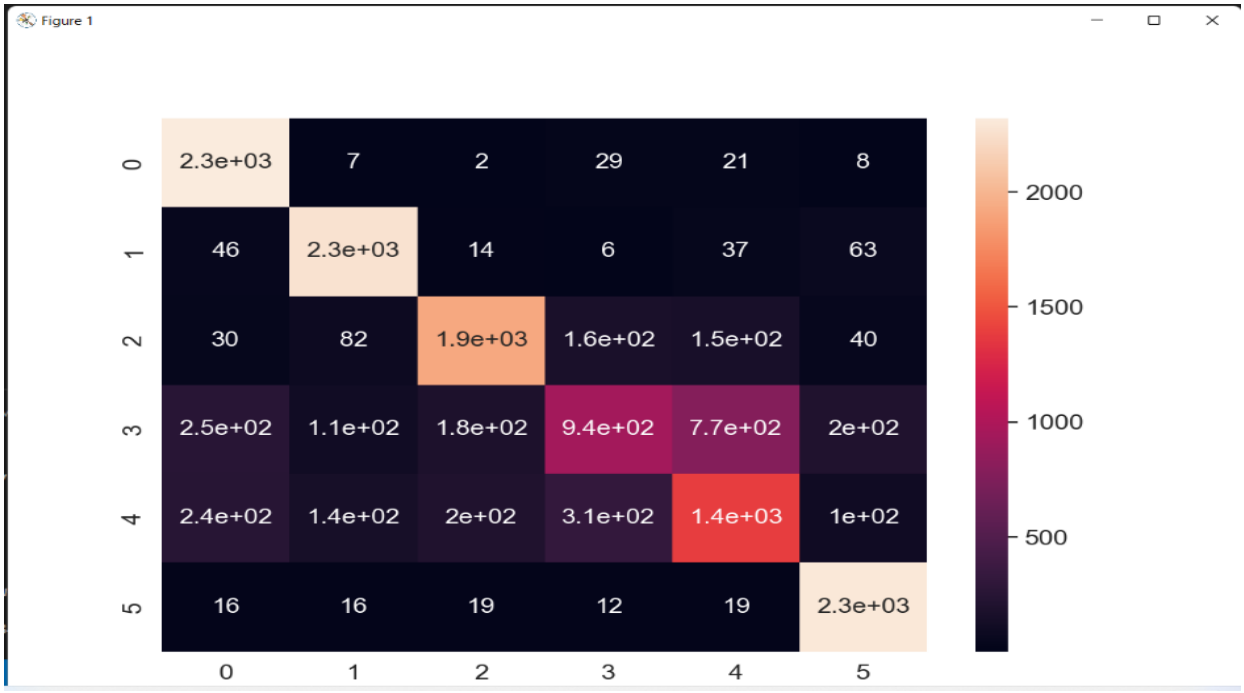


Fig-7: Naïve Bayes Classifier Confusion Matrix Heat-Map

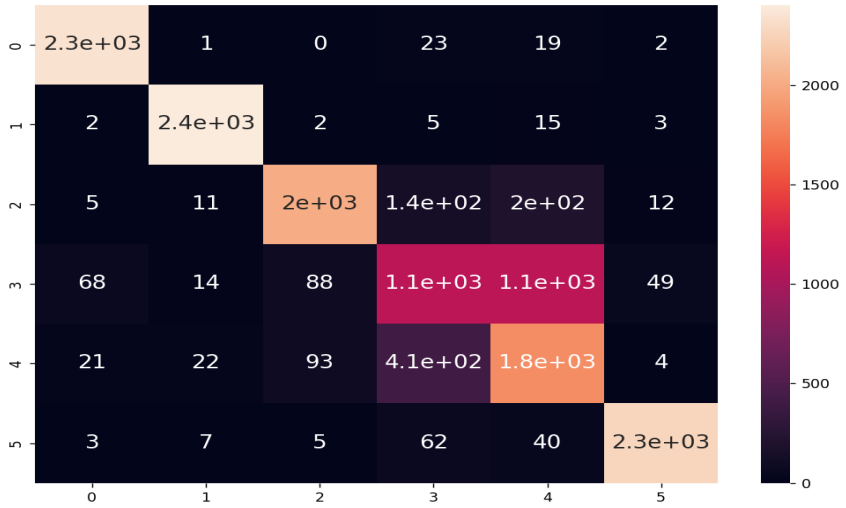


Fig-8: Support Vector Machine Confusion Matrix Heat-Map

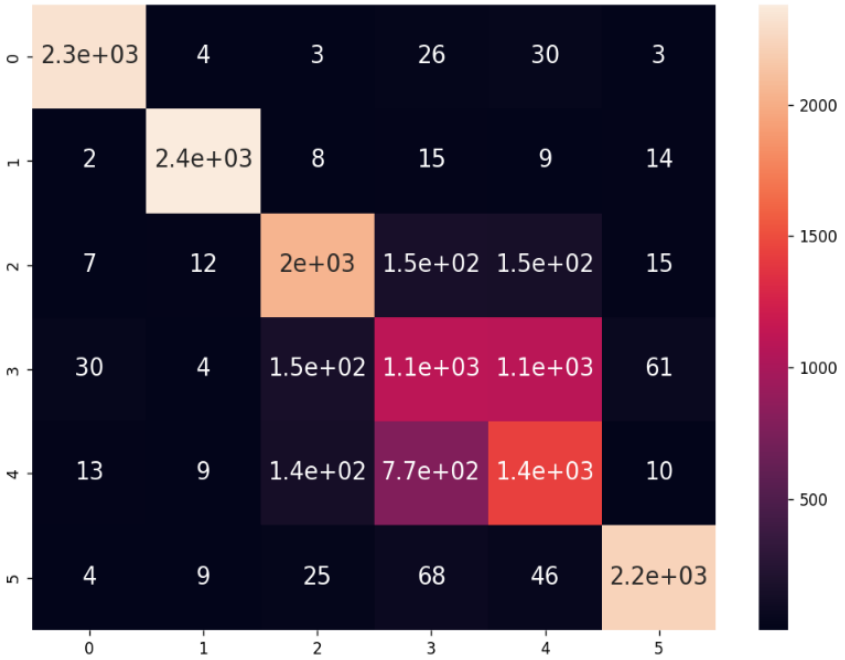


Fig-9: Decision Tree Classifier Confusion Matrix Heat-Map

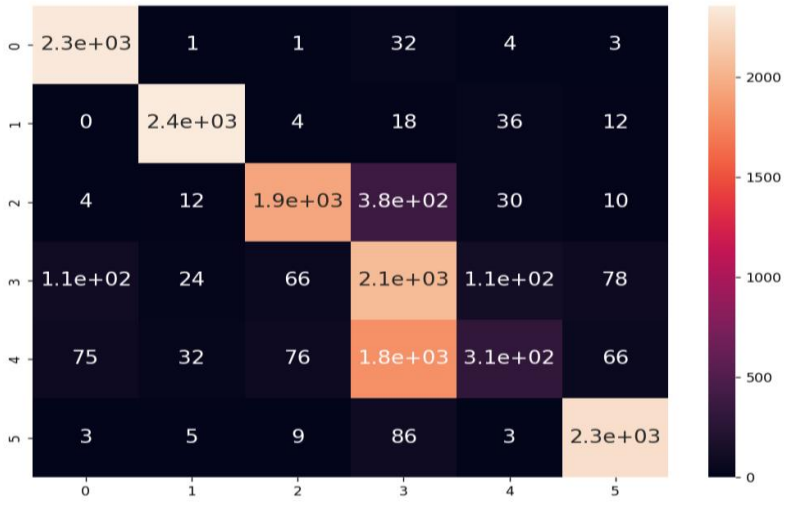


Fig-10: Multi-layer Perceptron Confusion Matrix Heat-Map

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

dtypes: object(2)
memory usage: 750.1+ KB
training , testing and evaluating performance measures of Random forest Classifier....

[[2345    0    1   18   22    1]
 [   0 2412    3    5    4    4]
 [   4    3 2113  118  136    5]
 [  27    3   81 1126 1145   61]
 [  11    7  100  595 1662    4]
 [   2    4    8   29   37 2303]]
                  precision    recall  f1-score   support

             age       0.98      0.98      0.98      2387
       ethnicity       0.99      0.99      0.99      2428
          gender       0.92      0.89      0.90      2379
not_cyberbullying       0.60      0.46      0.52      2443
           other       0.55      0.70      0.62      2379
        religion       0.97      0.97      0.97      2383

        accuracy                           0.83     14399
       macro avg       0.83      0.83      0.83     14399
    weighted avg       0.83      0.83      0.83     14399

0.8306826862976595
```

Fig-11: Random Forest Classifier Confusion Matrix and Performance Metrics

```
0.8306826862976595
Training , testing and evaluating performance metrics for Naive bayes....

[[2320    7    2   29   21    8]
 [  46 2262   14    6   37   63]
 [  30   82 1913  162  152   40]
 [ 247  107  179  941  774  195]
 [ 244  142  196  313 1384  100]
 [  16   16   19   12   19 2301]]
                   precision    recall  f1-score   support

              age       0.80      0.97      0.88      2387
        ethnicity       0.86      0.93      0.90      2428
           gender       0.82      0.80      0.81      2379
not_cyberbullying       0.64      0.39      0.48      2443
            other       0.58      0.58      0.58      2379
         religion       0.85      0.97      0.90      2383

         accuracy                           0.77     14399
        macro avg       0.76      0.77      0.76     14399
     weighted avg       0.76      0.77      0.76     14399

0.7723453017570665
```

Fig-12: Naïve Bayes Classifier Confusion Matrix and Performance Metrics

```
Training , testing and evaluating performance metrics for Support Vector Machine ....

[[2342    1    0   23   19    2]
 [   2 2401    2    5   15    3]
 [   5   11 2016  139  196   12]
 [  68   14   88 1111 1113   49]
 [  21   22   93  408 1831    4]
 [   3    7    5   62   40 2266]]
                   precision    recall  f1-score   support

              age       0.96      0.98      0.97      2387
        ethnicity       0.98      0.99      0.98      2428
           gender       0.91      0.85      0.88      2379
not_cyberbullying       0.64      0.45      0.53      2443
            other       0.57      0.77      0.65      2379
         religion       0.97      0.95      0.96      2383

         accuracy                           0.83     14399
        macro avg       0.84      0.83      0.83     14399
     weighted avg       0.84      0.83      0.83     14399

0.8310993819015209
```

Fig-13: Support Vector Machine Confusion Matrix and Performance Metrics

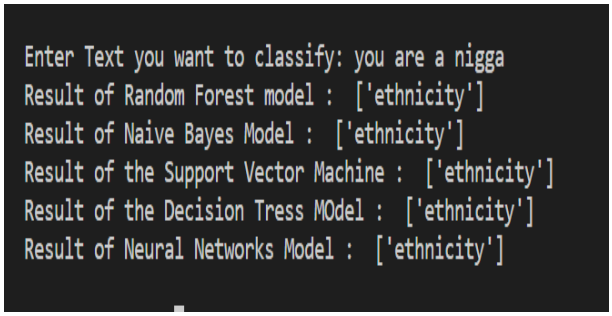
```
Training , testing and evaluating performance metrics for Decision Tree....

[[2321    4    3   26   30    3]
 [   2 2380    8   15    9   14]
 [   7   12 2042  153  150   15]
 [  30    4  154 1099 1095   61]
 [  13    9  136  773 1438   10]
 [   4    9   25   68   46 2231]]
                  precision    recall  f1-score   support

             age       0.98      0.97      0.97      2387
       ethnicity       0.98      0.98      0.98      2428
          gender       0.86      0.86      0.86      2379
not_cyberbullying       0.51      0.45      0.48      2443
           other       0.52      0.60      0.56      2379
        religion       0.96      0.94      0.95      2383

        accuracy                           0.80     14399
       macro avg       0.80      0.80      0.80     14399
    weighted avg       0.80      0.80      0.80     14399

0.7994305160080561
```

Fig-14: Decision Tree Classifier Confusion Matrix and Performance Metrics

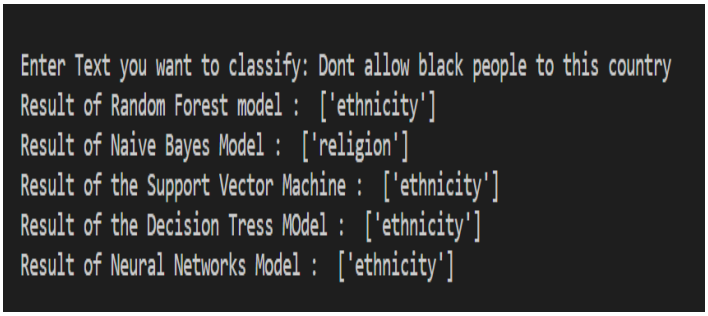
```
Training , testing and evaluating performance metrics for MLP Classifier....

[[2346    1    1   32    4    3]
 [   0 2358    4   18   36   12]
 [   4   12 1943  380   30   10]
 [ 106   24   66 2056  113   78]
 [  75   32   76 1822  308   66]
 [   3    5    9   86    3 2277]]
                  precision    recall  f1-score   support

             age       0.93      0.98      0.95      2387
       ethnicity       0.97      0.97      0.97      2428
          gender       0.93      0.82      0.87      2379
not_cyberbullying       0.47      0.84      0.60      2443
           other       0.62      0.13      0.21      2379
        religion       0.93      0.96      0.94      2383

        accuracy                           0.78     14399
       macro avg       0.81      0.78      0.76     14399
    weighted avg       0.81      0.78      0.76     14399

0.7839433293978748
```

Fig-15: Multi-layer Perceptron Classifier Corelation Matrix and Performance Metrics

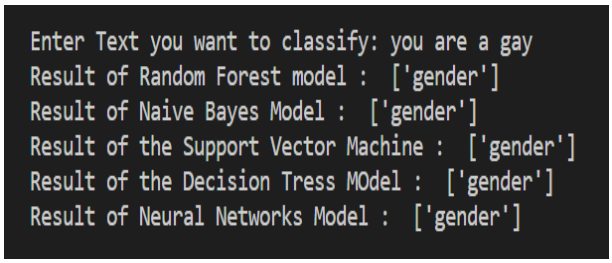
Prediction For User Input :



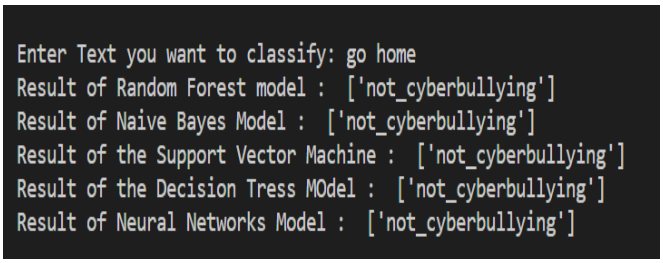
(a)



(b)



(c)



(d)

Figure 16 (a) Showing Output for the User Input as Gender
(b) Showing Output for User Input Text as Ethnicity
(c) Showing Output for User Input Text as gender
(d) Showing Output for User Input Text as not cyberbullying

CHAPTER 7

CONCLUSION

Data is collected by web scrapping. Collected Data is preprocessed using python functions. Model is built, trained and tested using the preprocessed data. The model takes input from the user and classifies it into different categories.

In particular, cyberbullying has become more common and has begun to raise significantly. Also, usage of social media is rapidly increasing. So, there is a need for automatic cyberbullying detection features on social media platforms. This model predicts whether the given input is a cyberbullying text or not. But it can be improved with advanced algorithms and methods to overcome the limitations and can be used on different platforms as a filter or a mechanism to detect and take measures to reduce cyberbullying on the internet.

REFERENCES

[1] Md Manowarul Islam; Md Ashraf Uddin; Linta Islam; Arnisha Akter; Selina Sharmin; Uzzal Kumar Acharjee , *"Cyberbullying Detection on Social Networks Using Machine Learning Approaches"*; 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), 16-18 December 2020.

[2] Rohith Gandhi, *"Naive Bayes Classifier"*; Towards Data Science, May 5, 2018.

[3] Joe Tran, *"Random Forest Classifier in Python"*; Towards Data Science, May 2, 2020.

[4] Nikolai Liubimov, *"Introducing Label Studio, a swiss army knife of data labeling"*; Towards Data Science, Jan 28, 2020.

[5] Priyanka Dave, *"How To Scrap YouTube Comments?"*; Analytics Vidhya, Mar 27, 2021.

[6] Manoj Nain, *"Scraping Tweets using Twitter APIs"*; Analytics Vidhya, Jul 4, 2020.