****

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**DEPARTMENT OF COMPUTING TECHNOLOGIES**

**VIII SEMESTER – JANUARY 2023**

**18CSP109L - PROJECT**

**Guide name: Dr. Vegesna S M Srinivasavarma**

**Designation: Research Assistant Professor  
Department: CTECH**

**Batch ID: 415**

Student 1 Reg. No: RA1911033010109

Name: Burra Sai Krishna Bharadwaj

Student 2 Reg. No: RA1911003010621

Name: Gutta Koteswararao

**PROJECT TITLE**

**A Hybrid Approach for Detecting Cyberbullying on Social Media Platform using Machine Learning Algorithms**

# ABSTRACT

Cyberbullying is on the rise on social media sites. With the popularity and pervasiveness of social media among individuals of all ages, it is critical to safeguard social media platforms against cyberbullying. Cyberbullying is spread across various social media platforms. It is a wrong act in which the victim is harassed by receiving derogatory, provocative, or sensitive images or text messages from the bully. Detection of such a message or post on such large platforms is very difficult and may sometimes lead to false detection.

In this paper, we present a systematic review of some published research on cyberbullying detection approaches and examine methods to detect hate speech in social media while distinguishing this from general profanity. We aim to establish lexical baselines for this task by applying supervised classification methods to a manually annotated and sorted open source datasets for this purpose.This paper does a comparative study of various supervised algorithms, including standard as well as ensemble methods. The evaluation of the result shows the algorithm that executed better results among the existing other implemented algorithms.

**CHAPTER 1**

# INTRODUCTION

# 1.1 Project Description

Social media applications such as Instagram, twitter, facebook, whatsapp etc have become most used and preferred platforms for interaction between people across the globe. These platforms enable people to communicate in different ways, it also leads to vicious ways.

This paper provides a comprehensive and structured overview of automatic hate speech detection, and compares few of its current approaches in a systematic manner Our proposed work provides a way in detecting cyberbullying using algorithms such as logistic regression, SVM, Decision tree classifier, ADA boost classifier, Gaussian NB, random forest and MLP classifier and evaluate the best possible way in detection. To develop an effective system for automatically detecting instances of cyberbullying on WhatsApp using a hybrid approach that combines rule-based and machine learning techniques. And improve the accuracy and efficiency of cyberbullying detection compared to existing state-of-the-art methods. Developing a system that is capable of handling the complexities of language and behavior used in cyberbullying and can detect new and emerging forms of such behavior. We ensure that the system protects user privacy and security while detecting instances of cyberbullying. By explore the potential applications of the system in social media platforms, educational institutions, law enforcement agencies, and mental health professionals. We are going to contribute the growing body of research on the use of machine learning for cyberbullying detection and prevention.

## 

## 1.2 Motivation

The major objective for performing this research is to develop a model for detecting the cyberbullying. Algorithms are applied at different levels of evaluations in a comparative study and analysis to corroborate this work. Though these machine learning algorithms are widely utilized. Consequently, a range of levels and evaluation strategy types are used to analyze the eleven algorithms namely logistic regression, SVM, Decision Tree, Ada Boost, naive bayes, random

forest, MLP classifier, SGD Classifier, K-Neighbour Classifier, MultinomialNB, Bagging Classifier. This will produce a better way possible in identifying the bullying across the media platform.

**1.3 Software Requirements Specifications**

Operating System : Windows 11

Technology : Python 3.10

IDE : Jupyter Notebook

# CHAPTER 2

# LITERATURE REVIEW

This section gives an overview the study of previously proposed models in cyberbullying detection which are related to our current work.

Numerous methods have been developed for the purpose of detecting cyberbullying, with the majority of these methods relying on Natural Language Processing and Information Retrieval. These methods then classify textual data by extracting its features using TF-IDF, Sentiment Analysis, Dimensionality Reduction, etc., and they have achieved commendable accuracy.

Sambhagadi et al.[1] uses NLP approaches to try and find solutions to identify and eventually stop cyberbullying on social media. NLP techniques are applied in such a way that they can even determine whether profanities are used in data in an offensive or neutral manner. Crowdsourcing and in-lab annotations are utilized to iteratively modify the annotations used in the paper. Data was gathered from English posts on social media platforms, even semi-anonymous ones like ask.fm. A ranked list of expletives and NLP made it possible to crawl efficiently. Modified linear SVM was used to categorize the data, and in addition to inappropriate words that could have been missed, features like question-and-answer posts and emoticons were also taken into account. Finally, the F1-Score was determined to be 0.59 (which, while smaller than the Kaggle winner, nevertheless appears promising given that this study did not employ personalized data and a fresh and superior dataset).

Challenges faced in this study were –

* With ask.fm comments, questions and answers are paired together in a way that they are not in other datasets, and both questions and answers may just comprise a single word, making it challenging for the algorithm to identify them without knowing the whole context.
* On social media, people utilize slang and casual language, which is full of misspellings and abbreviations, making it exceedingly challenging to process them.

Using supervised classification Machine Learning techniques, R. R. Dalvi et al.[2] proposes a technique to identify and stop Internet exploitation on Twitter. The live Twitter API is used in this study to gather tweets and create datasets. Using the gathered datasets, the suggested model compares Support Vector Machine with Naive Bayes. They have utilized the TFIDF vectorizer to extract the feature. The findings indicate that the accuracy of the Support Vector Machine-based cyberbullying model is close to 71.25%, which is higher than the accuracy of the Naive Bayes model, which was close to 52.75%.

Silva et al.[3] suggested a strategy for identifying cyberbullying based on psychological research; it details the design of an app called BullyBlocker, which seeks to alert the user's parents if cyberbullying is discovered. It employs conventional techniques to evaluate the user's social media data by looking through their messages and comments and classifying them as bullying indicators or warning indications. It is designed primarily for teenagers and uses antiquated Facebook detection techniques, but it has the potential to develop by serving as a data-gathering app over which machine learning categorization may be applied.

A dictionary-based method was used by Peter Burnap et al. [4] to find cyberhatred on Twitter. In this study, the predetermined vocabulary of hostile words was used to build the numerical vectors using an N-gram feature engineering technique. The authors received a maximum F-score of 67% after feeding the produced numeric vector to the SVM ML classifier. A dictionary-based strategy was also employed by Stéphan Tulkens et al. [6] for the automatic detection of racism in Dutch Social Media. The authors of this study employed the word distribution across three dictionaries as a characteristic. They supplied the SVM classifier with the generated characteristics. Their experimental findings yielded an F-Score of 0.46. Hate speech in online forums and blogs was classified using an ML-based classifier by Njagi Dennis et al. [5].

In 2021, Kumar et al. [7] released their research on how users behave online when interacting or submitting something publicly. This research aims to identify cyberbullying in tweets from Twitter. They have analyzed people's behaviours using machine learning methods.

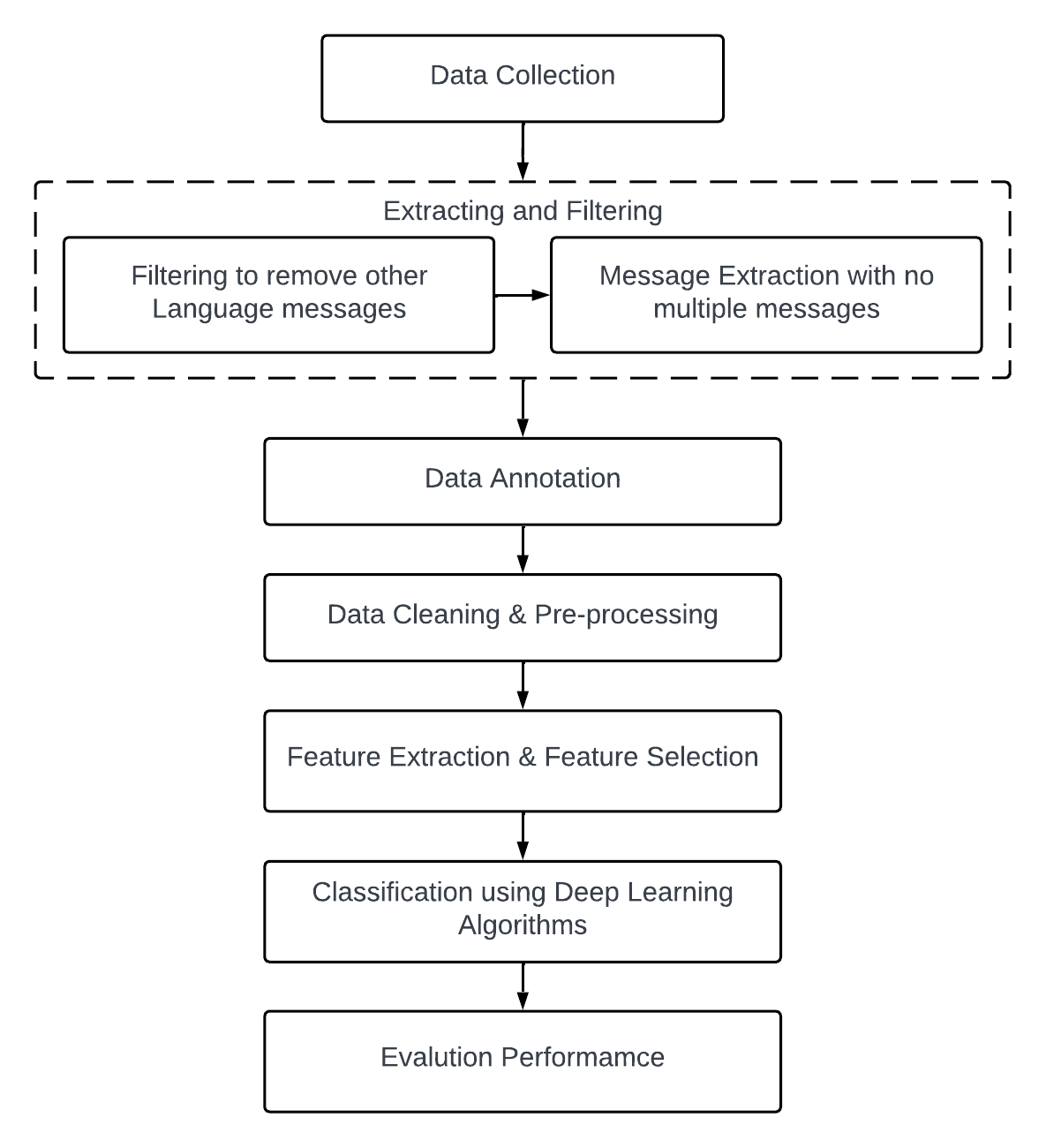
A review paper to identify cyberbullying in social media using machine learning techniques was published in 2021 by Singh et al. [8]. In this publication, they have examined the research on cyberbullying from 30 different researchers. They conducted research on methods for identifying and stopping cyberbullying, and they also provided users with some instructions for using online platforms.

Few researchers have used machine learning techniques to automatically identify hate speech in recent years. For instance, Karthik Dinakar et al.[9] categorized delicate subjects in comments or postings on social media. To create the numerical feature vectors for their study, they used the unigram with the TFIDF feature representation technique. The obtained features were fed to Naive Bayes, rule-based, J48, and SVM classifiers by the authors. According to their experimental findings, the rule-based classifier performed better than NB, J48, and SVM classifiers by achieving 73% accuracy. Web content pages were categorized into categories of hatred or violence by Shuhua Liu et al. [10]. They used trigram characteristics, which are represented by TFIDF, in their investigation. The Nave Bayes classifier was employed by the authors. The Naive Bayes classifier attained the maximum accuracy of 68% in their experimental circumstances.

**CHAPTER 3**

**SYSTEM ARCHITECTURE DIAGRAM**

The architecture of the system will provide a glance of working of the entire system.



### Fig 1: Architecture Diagram

The following is an explanation of how the system works:

We collected data and sorted it by combining various datasets that are openly available from internet sources. The data set consists of have both the normal and abusive tweets that helps us in supervising the data model. Attribute selection process selects the important attributes for identifying . So, once we have that dataset, we need to process this data set because we cannot beat this raw data directly into machine learning algorithm therefore, we should process our data. Before splitting our data into two sets we will perform exploratory data analysis to have a more thorough insight in the data and have a clearer understanding of it. After processing it we will have to split our data into two sets training data and testing data.

Once we do that we will feed our training data to our machine learning model in this case we are going to use a logistic regression model because this particular use case is a binary. We are going to perform this with other algorithms also like SVM, Decision Tree, Ada Boost, naive bayes, random forest, MLP classifier, SGD Classifier, K-Nearest Neighbour Classifier, MultinomialNB, Bagging Classifier.. Finally, we will measure the accuracy of all our models that are being used and we will go the most accurate algorithm.

## 

## 3.2 Algorithms

### 3.2.1 Logistic Regression Algorithm

A commonly used machine learning approach is logistic regression, algorithms, an element of the supervised learning process. Categorical dependent quantity is anticipated utilizing a combination of factors. The outcome of dependent categorical variable is estimated via logistic regression. The result must thus be a discrete or categorical value. Rather of giving the exact values of 0 and 1, it presents the probability numbers that fall between 0 and 1. It could be either Yes or No, 0 or 1, false or true etc. The major difference in logistic regression and linear regression is it is used. Problems involving regression are handled using linear regression, however, to overcome categorization issues, logistic regression is used. In logistic regression, we design a "S"-shaped curve instead of a regression line. By using the logistic function, two highest values are predicted (0 or 1). The logistic function's curve indicates the probability of such an event. Dependent on its weight, a mouse is either obese or not, depends on whether the cell is malignant and so on. Because it delivers the following benefits over all other machine learning algorithms. The capacity to evaluate fresh data using discrete and continuous methods and to calculate probability datasets.

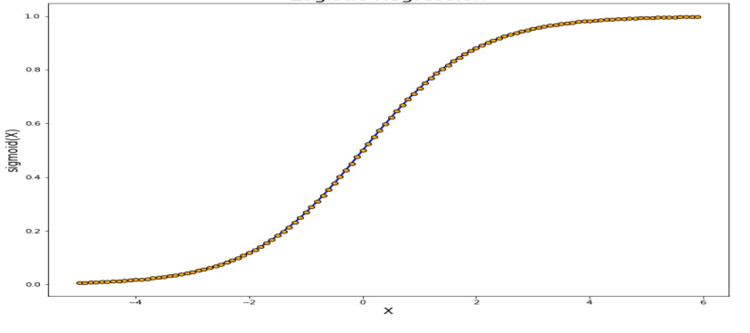


Fig 2 : graph of logistic regression

### 

### 3.2.2 Naive Bayes Algorithm

It is classification algorithm founded on Bayes Theorem and founded on the theory of predictor independence. A Naive Bayes classifier, to put it more simply, considers that the possession of one characteristic in a class has no connection to the appearance of any additional attribute. The Naive Bayes system is easy to construct and is exceptionally advantageous for the really big data sets. In addition to being uncomplicated, Naive Bayes is known to work better compared to even the most complex classification methods.

The naive bayes algorithm mainly comprises of two terms those are Naive and Bayes, which can be elaborated as:

* Naive : As it presumes that the existence of one aspect is irrelevant to the prevalence of many other attributes, it is known as naive. For example, if a fruit is identified as an apple depending on its red, delicious and spherical fruit., flavour and form So, without dependent on one another, each attribute allows us to identify it as an apple.
* Bayes : It is called bayes because it is mainly dependent on the principle of the bayes theorem.

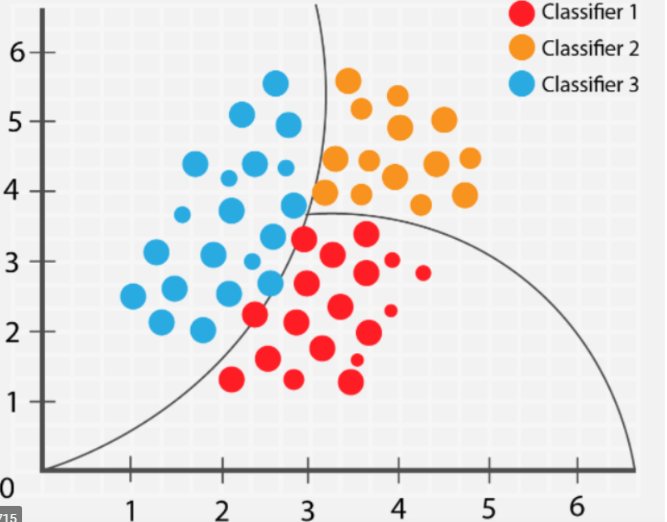


Fig 3: Naive bayes Classifier

### 

### 3.2.3 Support Vector Machine (SVM)

One of the most popular supervised learning algorithms is known as Support Vector Machine, or SVM, is utilized to solve problems involving regression and classification. But, it is employed widely in Machine Learning Classification problems. The SVM algorithm's goal is to determine the best line or decision boundary that can separate n-dimensional space into classes, enabling us to quickly categorize fresh data points in the near future. A hyperplane is the label given to this optimal decision boundary. SVM chooses the extreme vectors and points that assist in the development of the hyperplane. Support vectors are the terminology for these exceptional circumstances, and as a result it is termed as, Support Vector Machine.

Important terminologies in SVM are as follows:

Support vectors - those data points that are very near to the hyperplane is known as support vectors.

Hyperplane - Hyperplane is nothing but a decision boundary which separates the set of objects belonging to different classes.

Margin - it is defined as the gap between two lines that is drawn on the nearest data points in different classes.

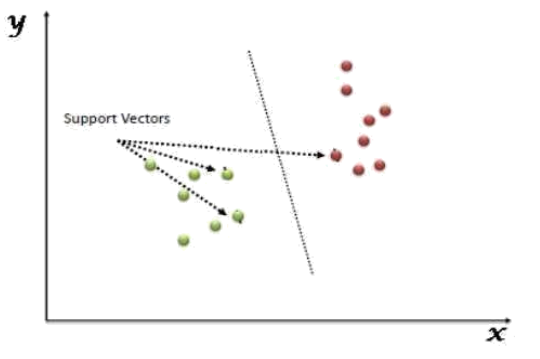


Fig 4 Support Vector Machine

### 3.2.4 K Nearest Neigbour Algorithm

The k-nearest neighbours (KNN) algorithm is a data classification strategy that evaluates the likelihood that a data point will join one group or another group based upon which group the data points that really are nearest to it are a member of. A supervised machine learning technique called as the k-nearest neighbour algorithm is utilized to handle regression and classification issues. Because it does not provide any training whenever you provide the training data, this technique is known as a lazy learner. However, it doesn't make any calculations during training; it just stores

the information. A query on the dataset needs to be run before such a model can be built. KNN is therefore perfectly suited for data mining.

Advantages and disadvantages of KNN algorithm are as follows:

* It is straightforward to put into practise.
* It can tolerate chaotic training data.
* When there is a lot of training data, it could work much better.
* K's value must be constantly determined, and sometimes this can be difficult.
* The powerful computational cost is caused due to the requirement to determine the difference between every data point for every training sample.

Below are some of the points that needs to be taken into consideration while we are calculating the K value for our KNN algorithm:

* The best value for "K" cannot be identified in a particular manner, thus we must try with different numbers to discover the number that works best. K is best depicted by the number 5.
* A relatively small amount of K, such K=1 or K=2, could be noisy and cause outliers impacts in the model.
* Although K should really have large values, there could be some issues.

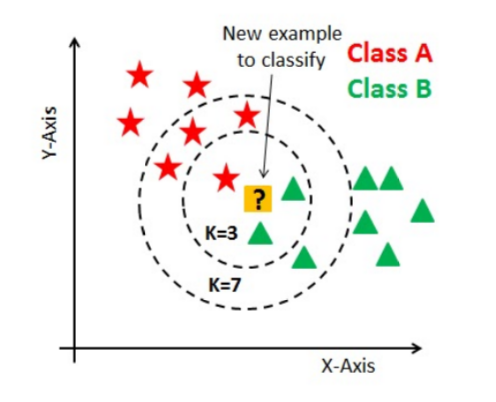


Fig 5 KNN Algorithm

**3.2.5 Decision Tree Algorithm**

Decision tree classifiers are a type of machine learning algorithm that works by constructing a tree-like model of decisions and their potential consequences in order to classify data. The algorithm uses a set of input features to recursively split the data into subsets, with the aim of minimizing the impurity or uncertainty of the resulting subsets. Each internal node of the tree represents a decision based on the value of a specific feature, and the tree branches out into two or more subtrees that correspond to the different possible values of that feature.

Advantages of decision tree classifiers:

* Easy to understand and interpret: Decision trees are simple and intuitive to understand, making them a popular choice for non-technical stakeholders who need to make decisions based on the results of a machine learning model.
* Can handle both categorical and numerical data: Unlike some other machine learning algorithms, decision trees can handle both categorical and numerical input data.
* Provide feature importance ranking: Decision trees can provide insight into the relative importance of different input features for making accurate predictions.

Disadvantages of decision tree classifiers:

* Prone to overfitting: Decision trees can be prone to overfitting, where the algorithm becomes too complex and fits the training data too closely, resulting in poor generalization to new, unseen data.
* Instability: Decision trees can be unstable, meaning that small changes in the training data can lead to significant changes in the resulting tree and its predictions.
* Biased towards features with many levels: Decision trees can be biased towards features with many levels, as they can create many splits and potentially overfit the training data.

**3.3.6 Ada Boost Classifier:**

Adaptive Boosting (AdaBoost) is a prominent ensemble learning strategy for classification tasks in machine learning. It creates a powerful classifier by merging numerous weak or base learners.

Here is how it works:

Initialize sample weights: Assign equal weight to each sample in the training dataset and Train a weak classifier on the training dataset. The weak classifier can be any algorithm that performs only slightly better than random guessing, such as a decision tree with a single split.

Evaluate the first base learner: Evaluate the performance of the weak classifier on the training dataset. The samples that were incorrectly classified by the weak classifier will be assigned a higher weight than the correctly classified samples.

Adjust the sample weights: Increase the weight of the misclassified samples and decrease the weight of the correctly classified samples.

Train the next base learner: Train a new weak classifier on the same dataset, but with the updated sample weights.

Evaluate the performance of the combined model: Combine the two weak classifiers using a weighted sum, where the weight of each classifier is proportional to its performance on the training dataset. Evaluate the performance of the combined model on the training dataset.

Iterate: Repeat steps 4-6, adjusting the sample weights and training new weak classifiers, until a specified number of weak classifiers have been trained or a specified performance criterion has been met then To make predictions on new data, apply the weak classifiers in sequence to the input features, using the weights assigned during training to combine their predictions into a final output.

**3.2.7 Random Forest Classifier**

Random Forest Classifier is a popular ensemble learning method used for classification tasks in machine learning. It builds multiple decision trees on different subsets of the input features and samples, and then combines their

predictions to make a final classification. Steps involved

1. Randomly sample the dataset: Select a random subset of samples from the training dataset.
2. Randomly select features: Select a random subset of features from the input features.
3. Build a decision tree: Build a decision tree on the selected features and samples, using a splitting criterion such as information gain or Gini impurity.
4. Repeat steps 1-3: Repeat steps 1-3 to build multiple decision trees, with each tree trained on a different subset of the features and samples.
5. Combine the predictions: To make a classification, apply each decision tree in the forest to the input features, and then combine their predictions using a majority voting scheme.

**3.2.8 MLP Classifier**

Multi-Layer Perceptron (MLP) Classifier is a type of neural network that is widely used for classification tasks in machine learning. It is a feedforward neural network, which means that the information flows in one direction, from input to output, without any feedback connections. MLP Classifier is composed of multiple layers of interconnected nodes (perceptrons) that process input features and generate predictions.

Each layer of MLP Classifier consists of multiple nodes, also known as neurons. The input features are fed into the first layer, and then processed through a series of weighted connections and activation functions. The output of each node is multiplied by a weight, and then summed with the outputs from other nodes in the same layer. The sum is then passed through an activation function, such as the sigmoid function or the ReLU (Rectified Linear Unit) function, to generate an output. The output of each layer is fed as input to the next layer, until the final output layer generates a prediction for each class.

The weights and bias terms in each layer of MLP Classifier are learned through backpropagation, which is an iterative optimization algorithm that adjusts the weights and bias terms in response to the error between the predicted outputs and the true labels. Backpropagation calculates the gradient of the error with respect to the weights and bias terms, and then updates the weights and bias terms in the opposite direction of the gradient, using a learning rate parameter that controls the size of the updates.

MLP Classifier can model complex input-output mappings, making it suitable for a wide range of classification tasks. It can handle both categorical and numerical input data, and can learn non-linear relationships between the input features and the target variable. MLP Classifier can generalize well to new, unseen data, if properly trained and optimized. However, it requires careful training and optimization to avoid overfitting and achieve high performance, and can be slow and difficult to interpret.

**3.2.9 SGD Classifier**

The main idea behind Stochastic Gradient Descent (SGD) Classifier is to iteratively update the weights and bias terms using a small subset of the training data, called a minibatch. This allows SGD Classifier to process large datasets efficiently, since it only needs to access a small fraction of the data at a time. The minibatch is randomly selected from the training data at each iteration, which introduces stochasticity into the optimization process.

SGD Classifier updates the weights and bias terms using the gradient of the loss function with respect to the weights and bias terms, which measures the difference between the predicted outputs and the true labels. The loss function is typically a convex function, such as the hinge loss for linear SVM (Support Vector Machine) or the log loss for logistic regression. The gradient of the loss function is estimated using the minibatch, and then used to update the weights and bias terms using a learning rate parameter that controls the size of the updates.

**3.2.10 MultinomialNB**

Multinomial Naive Bayes (NB) is a probabilistic classifier that is widely used for text classification tasks, such as document classification and sentiment analysis. It is based on Bayes' theorem, which describes the relationship between the probability of a hypothesis given some evidence, and the probability of the evidence given the hypothesis.In Multinomial NB, the input features are typically the frequencies of the words or tokens in a text document. The classifier assumes that the frequency distribution of each word in each class follows a multinomial distribution, which models the probability of observing a particular word given the class. The probability of a document belonging to a particular class is then calculated by combining the probabilities of observing each word in the document given the class, using Bayes' theorem.

Multinomial NB has several advantages for text classification tasks:

Efficiency: Multinomial NB is computationally efficient and can process large datasets quickly.

Robustness to irrelevant features: can handle irrelevant features and noisy data, since it estimates the probabilities of each feature independently.

Interpretability: Multinomial NB provides a straightforward interpretation of the classification decision, since it calculates the probability of each class given the input features.

Limitations and Challenges:

Independence assumption: It assumes that the input features are conditionally independent given the class, which may not always be true in practice. This can affect the accuracy of the classifier.

Lack of modeling of word interactions: It does not model the interactions between words, which can be important for some text classification tasks.

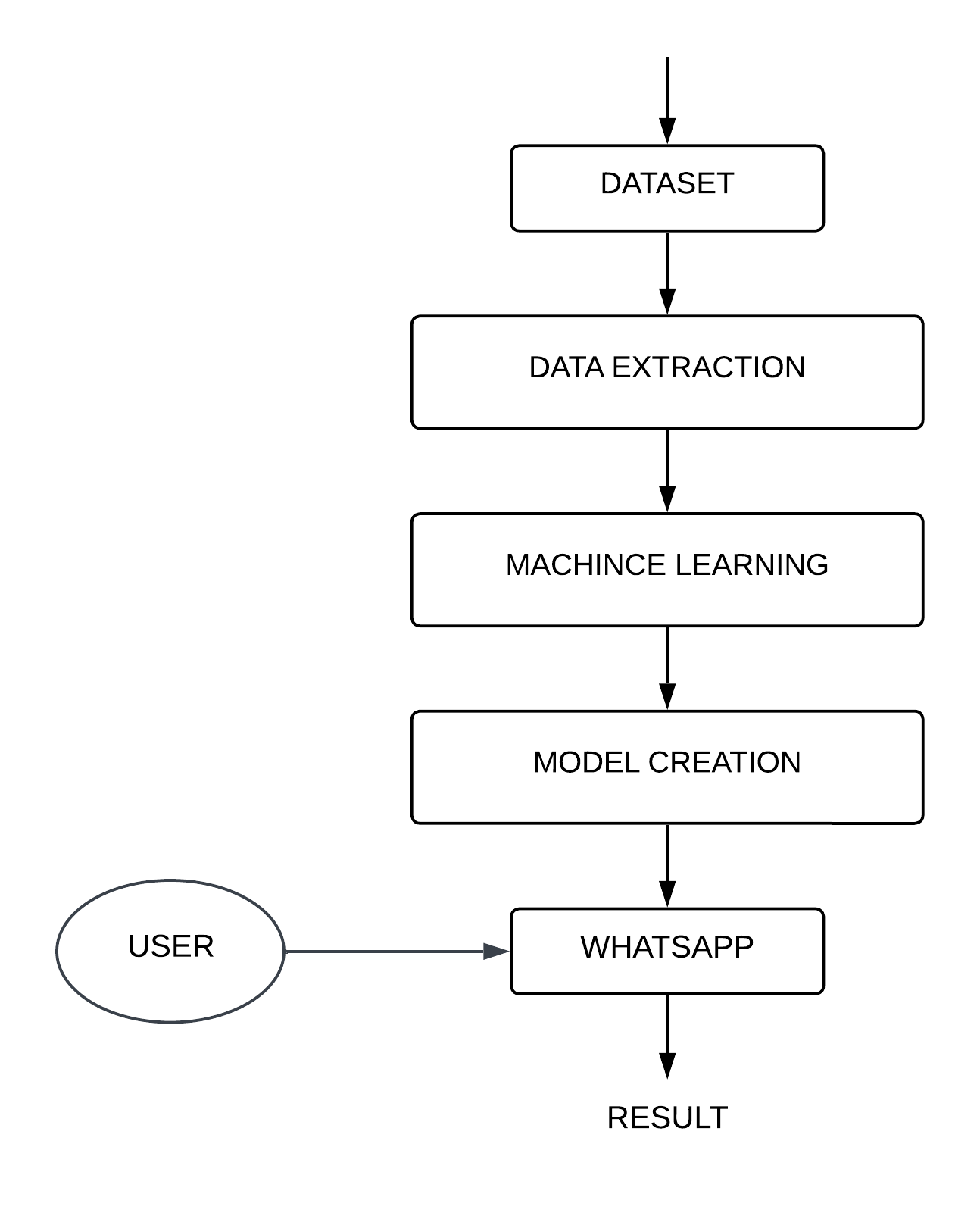
Sensitivity to feature scaling: This is sensitive to the scaling of the input features, since it uses the frequency of each word as the input feature.

**3.2.11 Bagging Classifier**

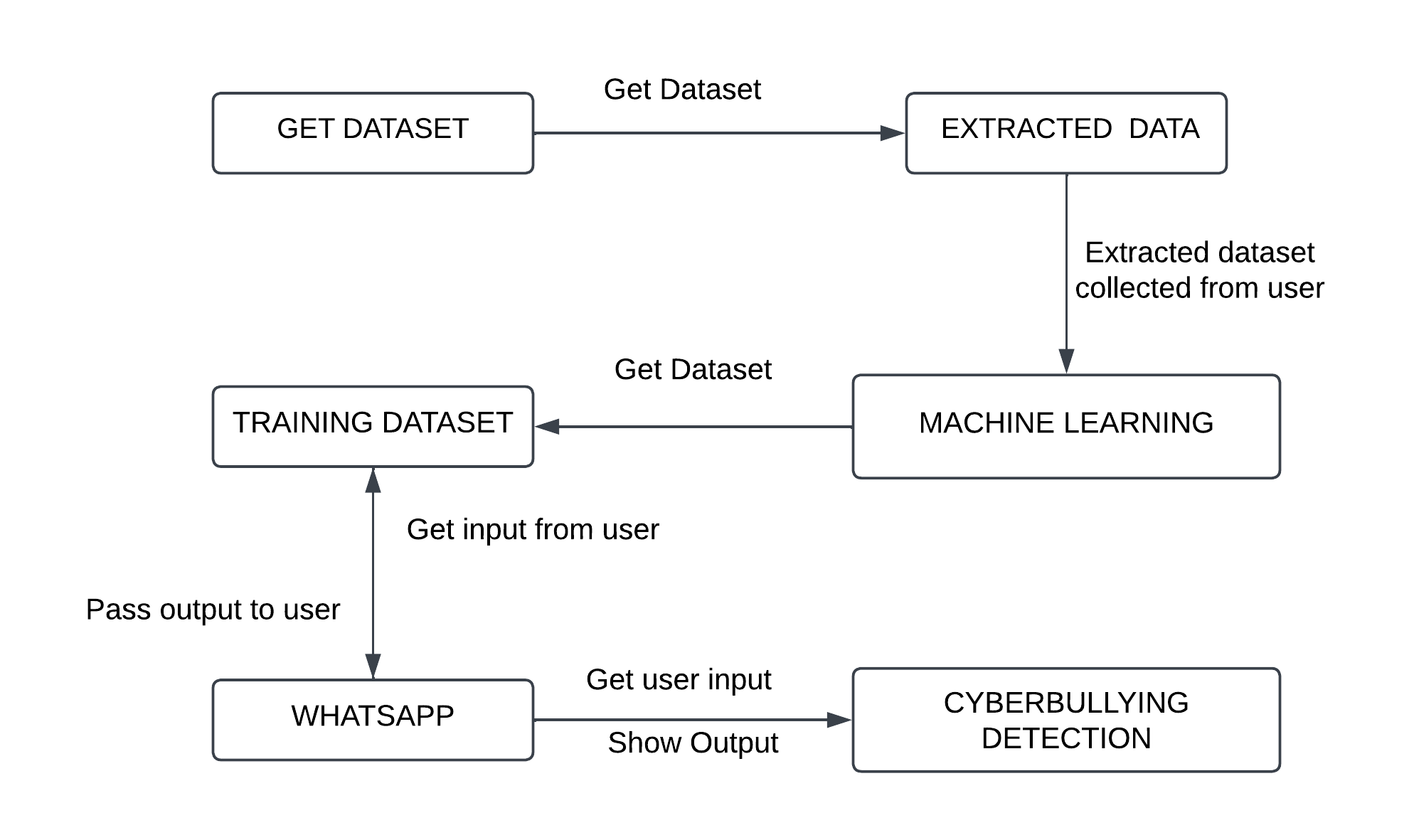
Bagging (Bootstrap Aggregating) is an ensemble learning method that combines many basic classifiers to increase the classification model's accuracy and stability. Bagging Classifier is a classification task-specific implementation of the Bagging technique.

The training dataset is randomly picked with replacement in Bagging Classifier to generate numerous bootstrap samples. On each bootstrap sample, a base classifier, such as Decision Tree, is trained to generate many weak classifiers. To make the final classification decision, the weak classifiers are combined using a voting system such as majority voting.

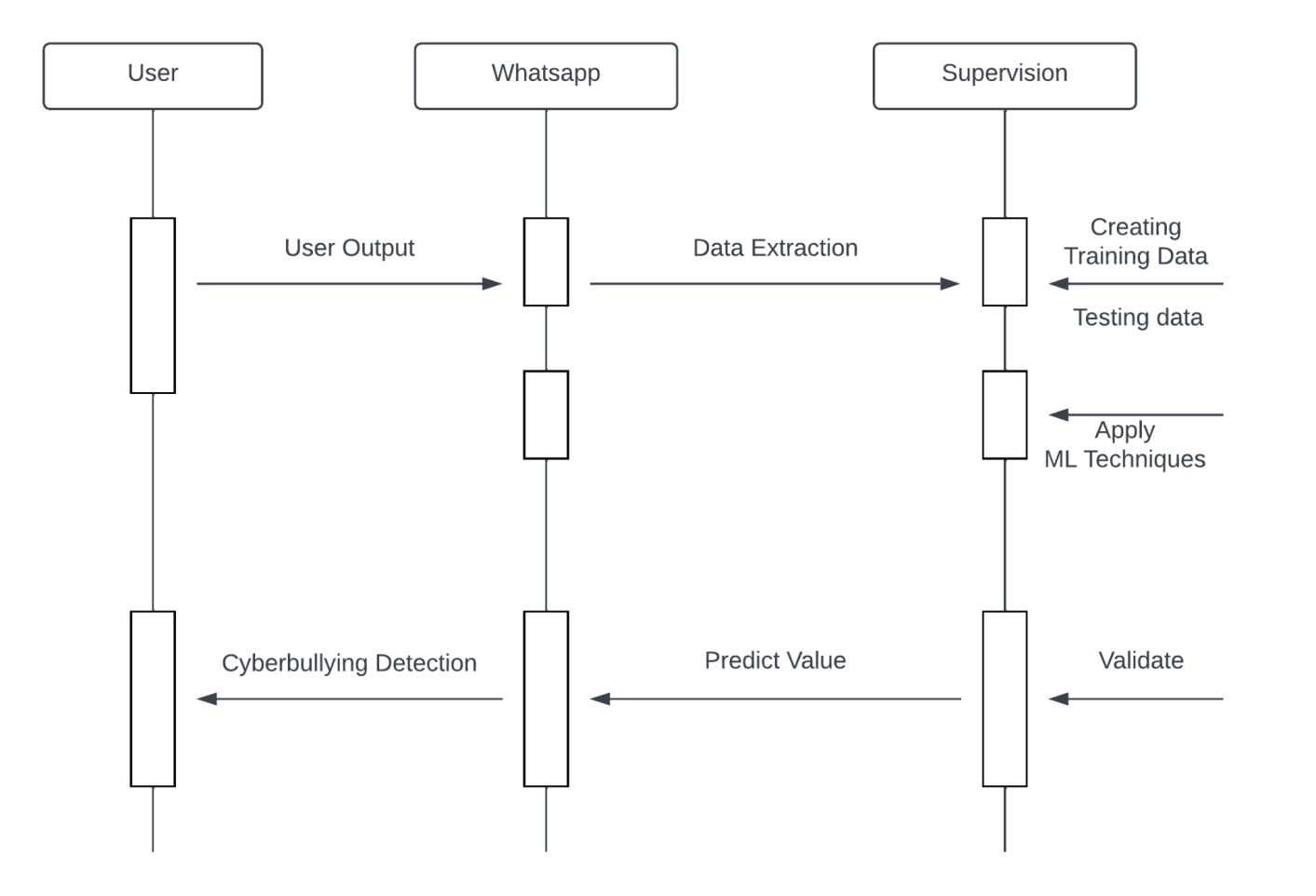
**4 .UML Diagrams**

****

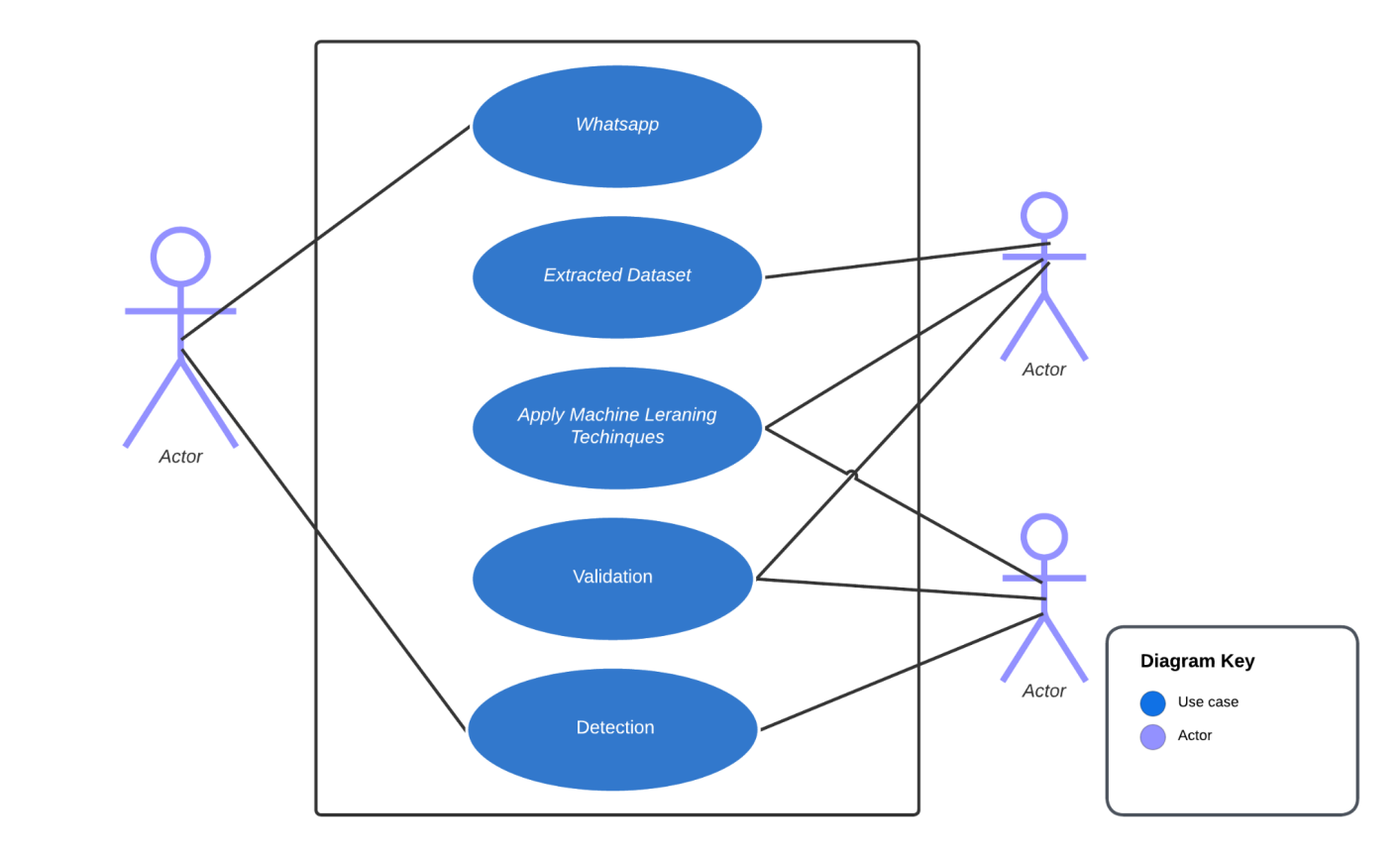
**Fig 4.1: Activity Diagram**



**Fig 4.2 : Collaboration Diagram**

****

**Fig 4.3 : Sequence Diagram**



**Fig 4.4 : Use Case Diagram**

**CHAPTER 5**

**METHODOLOGY**

**5.1. Data Collection**

We have gathered data from Twitter's 24000+ tweets for the project and organized that data to feed it to several machine learning algorithms. Because of the seriousness of the issue we aim to resolve, it was crucial to choose a dataset that was complete, reliable, relevant, and to the point. While we considered many other datasets as well, many of them either had missing attributes, were too low in quality, or were found to have irrelevant data after manual inspection. Thus we sorted the existing datasets available and made full fledged dataset.

1. It is partially named as labelled data
2. Total number of instances:24783

**5.2. Data Pre-Processing**

The preprocessing steps were done as follows using the nltk library along with regex:

1) Word Tokenization: A Token is a single entity that is building blocks for sentence or paragraph. Word Tokenization converts our text to separate words in a list.

2) Stop words filtering is done using nltk.corpus.stopwords.words(‘english’) to fetch a list of stopwords in the English dictionary, after which they are removed. Stop words are words such as “the”, “a”, “an”, “in”, which are not significant and do not affect the meaning of the data to be interpreted.

3) To remove punctuation, we save only the characters that are not punctuation, which can be checked by using string.punctuation .

4) Stemming: Stemming is a process of linguistic normalization, which reduces words to their word root word. We stem the tokens using nltk.stem.porter.PorterStemmer to get the stemmed tokens. For example, connection, connected, connecting word reduce to a common word ”connect”.

5) Digit removal: We also filtered out any numeric content as it doesn’t contribute to cyberbullying.

**5.3.Extracting and filtering:**

Extracting and filtering of data are two important data processing tasks that are often used in data analysis and machine learning.

Extracting data refers to the process of selecting and retrieving specific data from a larger dataset. This is typically done when we want to focus on a particular subset of the data that is relevant to our analysis. Extraction can involve selecting specific columns or rows of data based on some criteria, such as a date range, a particular category or tag, or a specific data value. Extraction can be performed using tools such as database queries, scripting languages like Python, or data visualization software.

**5.4. Feature Extraction and Feature Selection:**

Feature Extraction and Feature Selection are both techniques used to reduce the dimensionality of data in machine learning. Feature Extraction involves transforming raw data into a set of meaningful features that can be used to train a model, while Feature Selection involves selecting a subset of the most relevant features from the original dataset.

**5.5. Classification and Detection**

Random Forest is a commonly used machine learning algorithm that combines the outputs of multiple decision trees to produce a single result. Its ease of use and flexibility are driving its adoption as it can handle both classification and regression problems.

SVM is one of the most popular supervised learning algorithms used for both classification and regression problems. It is mainly used for machine learning classification problems. It creates the decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category.

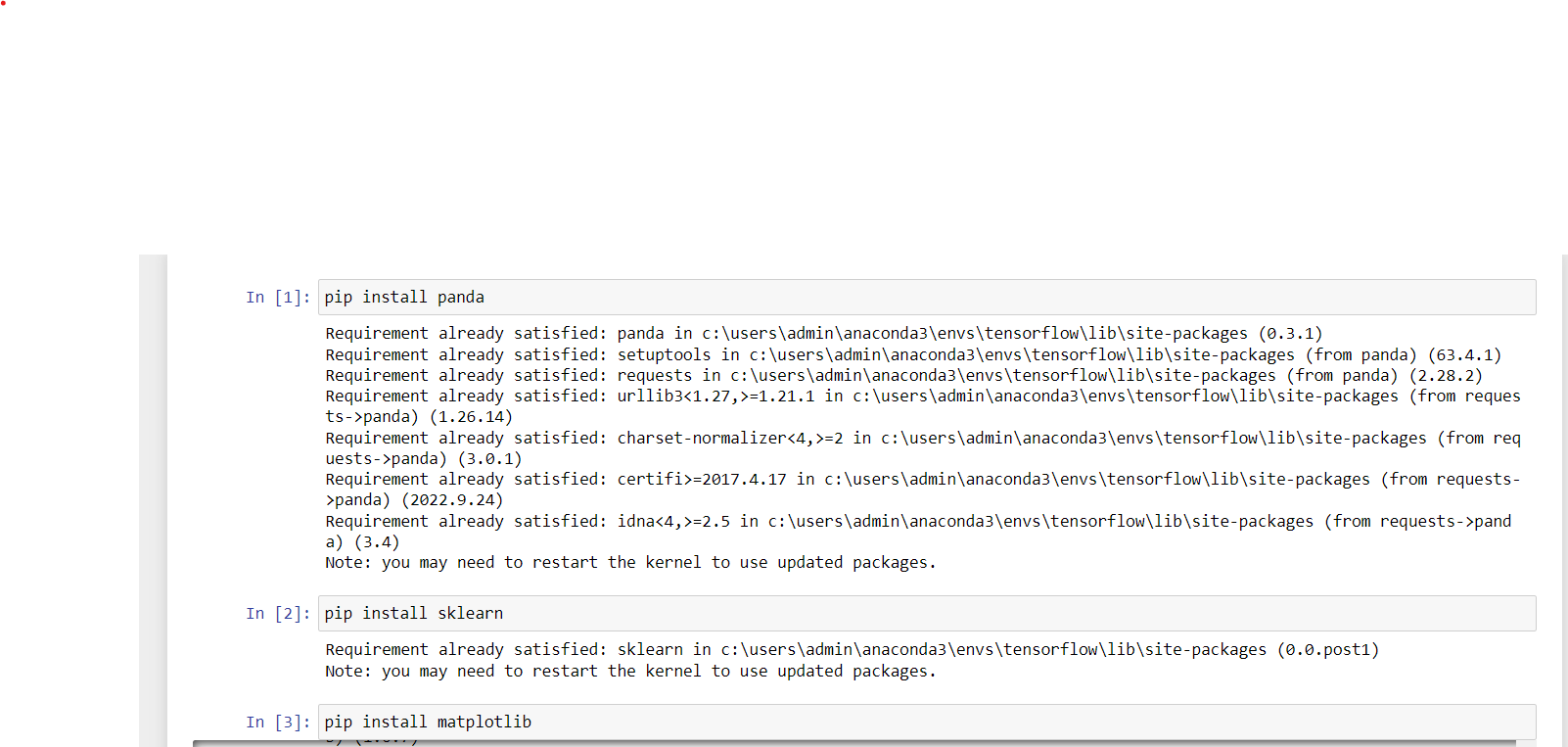
Recurrent neural network (RNN) is a type of artificial neural network which uses used for ordinal or temporal problems, such as language translation, Natural Language Processing (NLP), speech recognition.

**CHAPTER 6**

**CODING AND TESTING**

**6.1 Installing Required Libraries**

Here we are installing the libraries which are necessarily needed for the implementation of the models so here we installing libraries like pandas, sklearn, matplotlib, seaborn and the other algorithm related libraries.



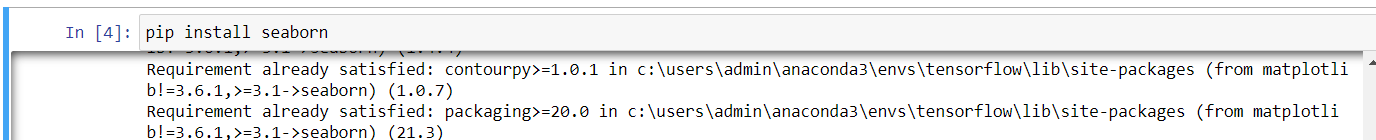


Fig 6.1.1 Installing Libraries

**6.2 Importing Libraries**

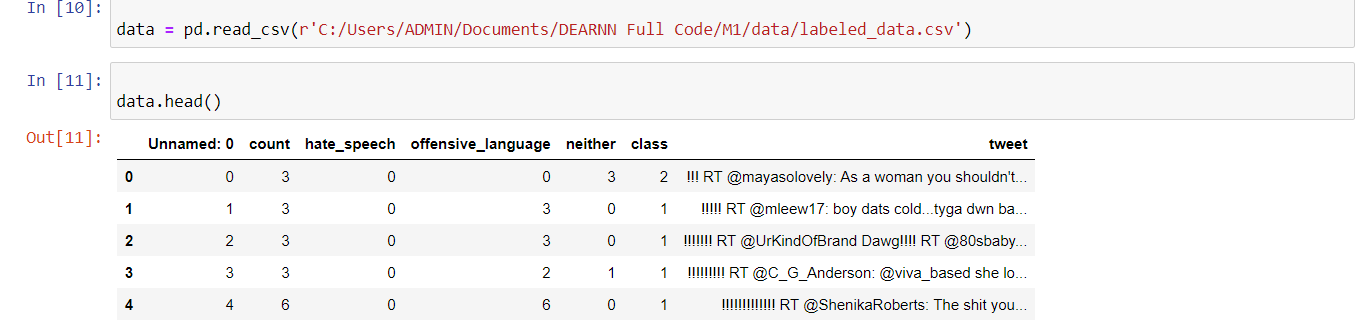
Here we are importing the libraries which are necessarily needed for the implementation of the models so here we importing libraries like pandas, sklearn, matplotlib, seaborn and the other algorithm related libraries.



Fig 6.2.1 Importing libraries

**6.3 IMPORTING DATASET**

Now we will import the dataset using the below code using pandas through read function and we check the head of the dataset that is the first 5 columns of the dataset to check whether we got the dataset imported properly or not. Now we will print the information of the dataset using info function which gives us the complete details of the dataset that is the tota number of rows the total number of columns, the attributes that are present in dataset, the datatypes of each of the attribute values and the null values and the non null values.

****

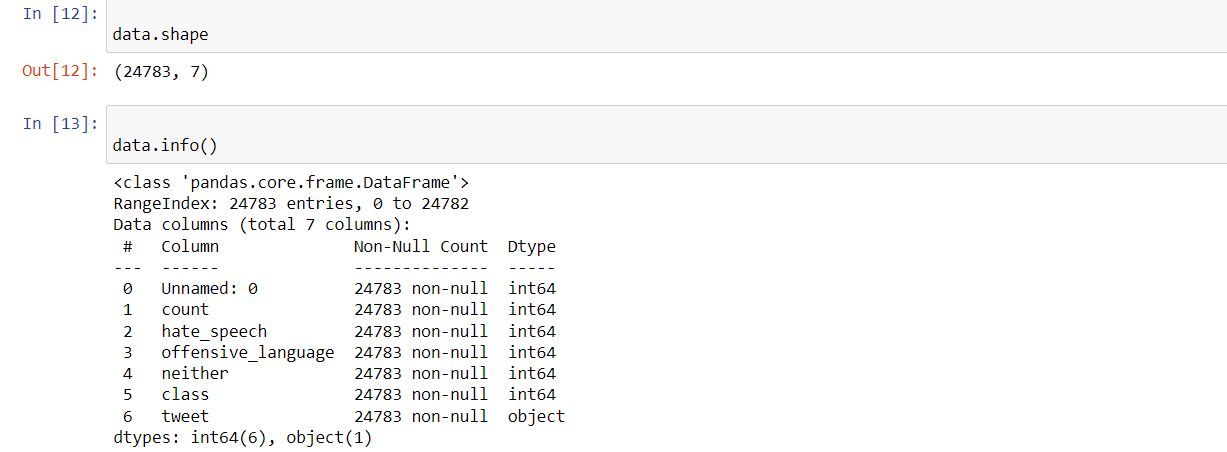
****

Fig 6.3.1 Dataset Importing

**6.4 Algorithm Implementation**

Support Vector Machine (SVM) is a type of supervised learning algorithm that aims to find the best separating boundary (hyperplane) between different classes of data. The algorithm uses a kernel function to map the input data into a high-dimensional feature space, where the boundary can be found more easily.

The algorithm was implemented using from sklearn.svm import SVC.



6.4.1 SVM Implementation

Gaussian Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem of mathematics. In simple words, the Bayes’ theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. It is called naive Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution. This extension of Naive Bayes is called Gaussian Naive Bayes. Beside the Gaussian Naive Bayes there are also existing the Multinomial naive Bayes and the Bernoulli naive Bayes. We picked the Gaussian Naive Bayes because it is the most popular one and one of the simplest to implement because we only need to estimate the mean and the standard deviation from the training data. The classifier was implemented using sklearn.naive bayes package.

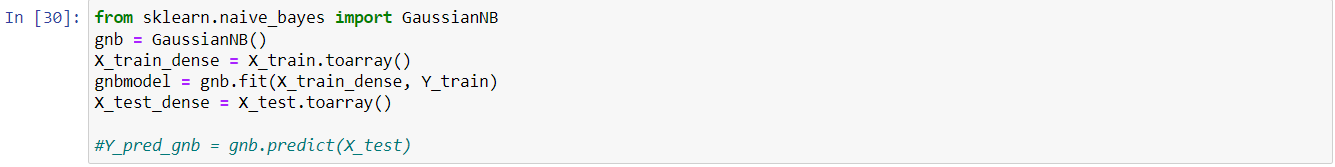


Fig 6.4.2 Gaussian Implementation

Logistic Regression Regression analysis is a predictive modelling technique that analyzes the relation between the target or dependent variable and independent variable in a dataset. Regression analysis techniques get used when the target and independent variables show a linear or non-linear relationship between each other, and the target variable contains continuous values. Regression analysis involves determining the best fit line, which is a line that passes through all the data points in such a way that distance of the line from each data point is minimized. Logistic regression is one of the types of regression analysis technique, which gets used when the dependent variable is discrete. Example: 0 or 1, true or false, etc. This means the target variable can have only two values, and a sigmoid curve denotes the relation between the target variable and the independent variable, by mapping any real value to a value between 0 and 1. We chose Logistic Regression as the size of our data set was large, and it had almost equal occurrence of values to come in target variables. Moreover, there was no correlation between independent variables in the dataset.

The classifier was implemented using sklearn.linear model package.



Fig 6.4.3 Logistic Regression Implementation

Decision Tree Classifier A Decision Tree is constructed by asking a series of questions with respect to the dataset. Each time an answer is received, a follow-up question is asked until a conclusion about the class label of the record. The series of questions and their possible answers can be organised in the form of a decision tree, which is a hierarchical structure consisting of nodes and directed edges. It has 3 types of nodes: Root, Internal, and Leaf nodes. In a decision tree, each leaf node is assigned a class label. The non-terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics. Using the decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG) (reduction in uncertainty towards the final decision). In an iterative process, we can then repeat this splitting procedure at each child node until the leaves are pure. This means that the samples at each leaf node all belong to the same class. The classifier was implemented using sklearn.tree package.

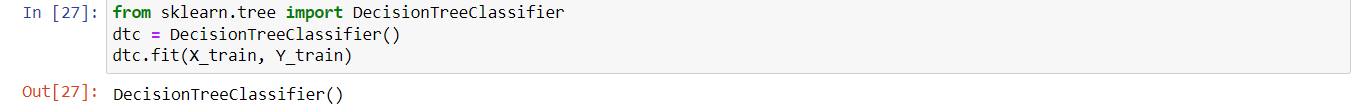


Fig 6.4.4 Decision Tree Implementation

Adaboost Classifier AdaBoost is an iterative ensemble method. The general idea behind boosting methods is to train predictors sequentially, each trying to correct its predecessor. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. At a high level, AdaBoost is similar to Random Forest as they both tally up the predictions made by each decision trees within the forest to decide on the final classification. There however, lie some subtle differences. In AdaBoost, the decision trees have a depth of 1 (i.e. 2 leaves). In addition, the predictions made by each decision tree have varying impact on the final prediction made by the model. Rather than taking the average of the predictions made by each decision tree in the forest (or majority in the case of classification), in the AdaBoost algorithm, every decision tree contributes a varying amount to the final prediction. The classifier was implemented using sklearn.ensemble package.



Fig 6.4.5 Ada Boost Implementation

Random Forest Classifier As its name implies, Random Forest Classifier 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 as they can produce ensemble predictions that are more accurate than any of the individual predictions, as the trees protect each other from their individual errors. The process of Bagging is used to diversify models as each individual tree is allowed to randomly sample from the dataset with replacement. The classifier was implemented using sklearn.ensemble package.

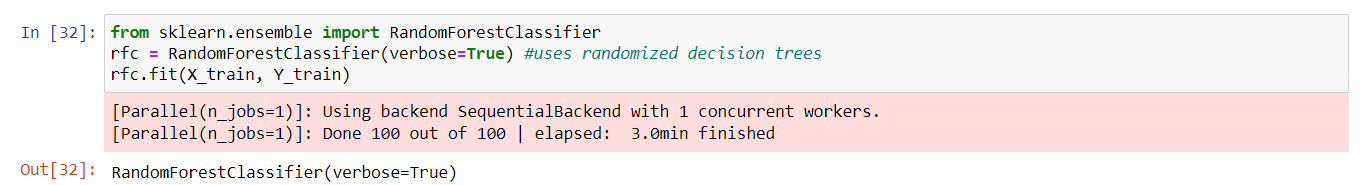
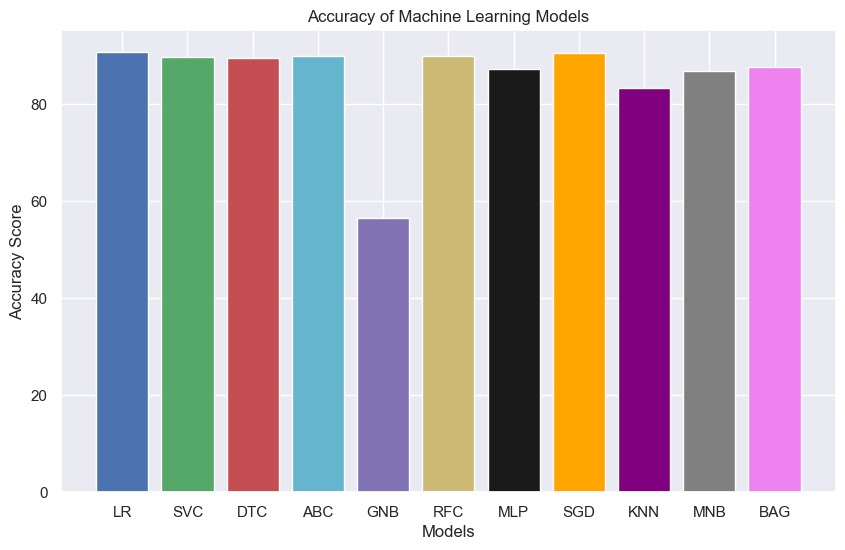
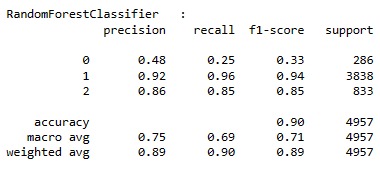
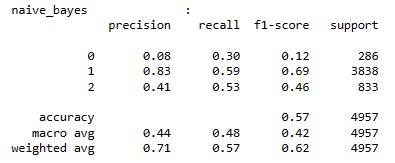
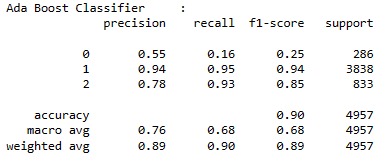
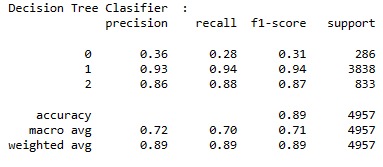
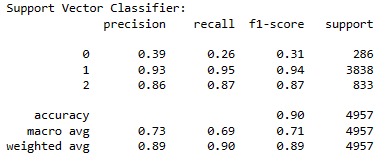
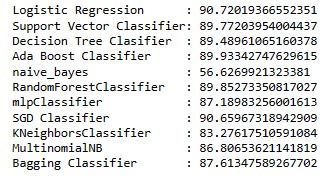
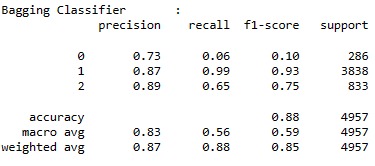
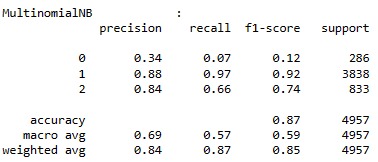
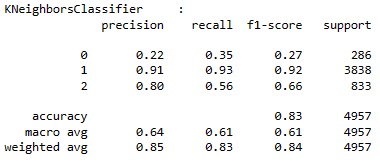
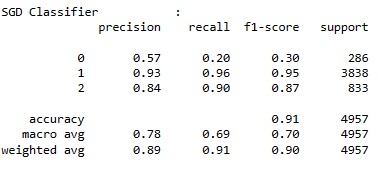
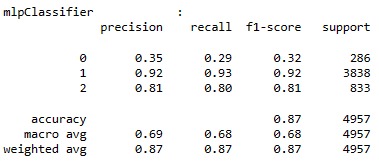


Fig 6.4.6 Random Forest Implementation

**RESULTS**







References

1. Lida Ketsbaia, Biju Issac and Xiaomin Chen, “Detection of Hate Tweets using Machine Learning and Deep Learning,” in Proc. of the 19th IEEE International conference on Trust, Security and Privacy in Computing and Communications, Guangzhou, China, pp. 751-758, 2020.
2. Vaibhav Jain, Ashendra Kumar Saxena, Dr. Athithan Senthil, Abhishek Jain, and Arpit Jain, “Cyber-Bullying Detection in Social Media Platform using Machine Learning” in Proc. of the 10th International Conference on System Modeling & Advancement in Research Trends (SMART), Morabad, India, pp. 401-405, 2021.
3. Monirah A. Al-Ajlan, and Mourad Ykhlef, “Optimized Twitter Cyberbullying Detection based on Deep Learning,” in Proc. of the 21st Saudi Computer Society National Computer Conference (NCC), Riyadh, Saudi Arabia, pp. 1-5, 2018.
4. Jaideep Yadav, Devesh Kumar, and Dheeraj Chauhan, “Cyberbullying Detection using Pre-Trained BERT Model,” in Proc. of the International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, pp. 1096-1100, 2020.
5. Nidhi Chandra, Sunil Kumar Khatri, and Subhranil Som, “Cyberbullying Detection using Recursive Neral Network through offline Repository,” in Proc. of the 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, pp. 748-754, 2018.
6. Samghabadi, Niloofar Safi, et al. ”Detecting nastiness in social media.” Proceedings of the First Workshop on Abusive Language Online. 2017.
7. R. R. Dalvi, S. Baliram Chavan and A. Halbe, Detecting A Twitter Cyberbullying Using Machine Learning, ICICCS, pp. 297-301, doi: 10.1109/ICICCS48265.2020.9120893. (2020)
8. N. Silva, D. L. Hall, C. Rich, BullyBlocker: toward an interdisciplinary approach to identify cyberbullying. Social Network Analysis and Mining. 8 (2018), doi:10.1007/s13278-018-0496-z
9. Burnap, P. and M.L. Williams, Us and them: identifying cyber hate on Twitter across multiple protected characteristics. EPJ Data Science, 2016. 5(1): p. 11.
10. Gitari, N.D., et al., A lexicon-based approach for hate speech detection. International Journal of Multimedia and Ubiquitous Engineering, 2015. 10(4): p. 215-230.
11. Tulkens, S., et al., A dictionary-based approach to racism detection in dutch social media. arXiv preprint arXiv:1608.08738, 2016.
12. Jain, A., Gahlot, A. K., Dwivedi, R., Kumar, A., & Sharma, S. K. (2018). Fat Tree NoC Design and Synthesis. In Intelligent Communication, Control and Devices (pp. 1749-1756). Springer, Singapore.
13. Kumar, Sandeep, E. G. Rajan, and Shilpa Rani. “A Study on Vehicle Detection through Aerial Images: Various Challenges, Issues and Applications.” In 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp. 504-509. IEEE, 2021
14. Dinakar, K., R. Reichart, and H. Lieberman. Modeling the detection of textual cyberbullying. In fifth international AAAI conference on weblogs and social media. 2011.
15. Liu, S. and T. Forss. Combining N-gram based Similarity Analysis with Sentiment Analysis in Web Content Classification. in KDIR. 2014.