

ABSTRACT

In today's digital age, social media platforms have become an integral part of our lives, connecting people from all over the world. While these platforms have brought numerous benefits, they have also exposed vulnerable individuals, particularly children, to various online risks. Among these risks are online child predators and cyber harassers who exploit the anonymity and reach of social media to harm others. In the past, combating these threats relied heavily on manual reporting and human moderators. Users would report suspicious activities, and then human moderators would review the content to determine if it violated platform guidelines. However, this reactive approach often resulted in delays, enabling harmful content to spread before action could be taken. Recognizing the need for more proactive and efficient measures, researchers have turned to machine learning, a branch of artificial intelligence that allows computers to learn from data and make predictions. By harnessing the power of machine learning algorithms, the goal is to develop an automated system capable of identifying potential online child predators and cyber harassers swiftly and accurately. The proposed machine learning-based approach offers several advantages over traditional methods. First and foremost, it significantly reduces the response time, enabling platforms to take immediate action against harmful content and individuals. Moreover, machine learning algorithms can identify patterns and connections in vast amounts of data that might not be evident to human moderators, thereby increasing the accuracy of detection. By integrating machine learning into the social media moderation process, human moderators can focus on more complex tasks that require human judgment and intervention. This not only improves the overall efficiency of content moderation but also lightens the burden on human moderators. Further, the integration of machine learning in identifying and combating online child predators and cyber harassers marks a significant step forward in enhancing online safety. The proactive nature and enhanced accuracy of machine learning-based approaches offer promising solutions to mitigate the threats posed by harmful individuals on social media platforms, making them a crucial addition to the fight against online abuse.

Keywords: Online social networks, Child predators, Cyber harassers, Natural language processing, Machine learning

CHAPTER 1

INTRODUCTION

In today's digital age, social media platforms have transformed the way we communicate and connect with others. These platforms have undoubtedly brought about numerous opportunities for global interaction, but they have also given rise to significant challenges [1]. Among these challenges are the presence of online child predators and cyber harassers, individuals who exploit the anonymity and reach of the internet for harmful purposes. Safeguarding individuals, particularly children, from these online threats has become a pressing concern [2]. The history of addressing online threats such as child predators and cyber harasser's dates back to the early days of the internet when these issues first emerged. Over the years, various efforts have been made to combat these threats. These efforts have included legal measures, educational campaigns aimed at users, and the development of technology-based solutions [3]. As technology advanced, particularly in the fields of machine learning and data analysis, new possibilities emerged for more effective identification and mitigation of these online dangers.

The need for an automated system for identifying online child predators and cyber harassers arises from practical considerations:

- **Scale:** The sheer volume of online content makes manual monitoring unfeasible. Automated solutions are essential for processing and analysing vast datasets effectively.
- **Speed:** Threats in the online world can escalate rapidly. Rapid detection and response are critical to preventing harm.
- **Complexity:** Identifying predatory or harassing behaviour often involves analysing text, images, and user behaviour patterns. Machine learning and data analysis techniques can significantly enhance this process.
- **Accuracy:** Reducing false positives and false negatives is paramount. Striking the right balance ensures that innocent users are not wrongly targeted, and potential threats are not overlooked.

1.1 Significance

The importance of identifying and addressing online child predators and cyber harassers cannot be overstated:

- **Child Safety:** Ensuring the safety of children in the online world is of paramount importance. Early identification of potential predators can prevent harm to minors.
- **Cyberbullying Prevention:** Cyberbullying can have severe psychological and emotional effects on individuals. Timely detection and intervention can significantly mitigate the impact of cyberbullying.
- **Legal Compliance:** Many countries have enacted laws that require online platforms to take measures against online harassment and child exploitation. Compliance with these laws is essential to avoid legal consequences.
- **Protecting Online Communities:** Maintaining a safe and respectful online environment is crucial for the reputation and trustworthiness of social media platforms and online communities.

1.2 Objective

Therefore, the proposed "Identification of Online Child Predators and Cyber Harassers" application is a sophisticated web-based tool developed using the Django framework. It seamlessly integrates several essential components, including user registration, content monitoring, machine learning algorithms, and an admin panel, to tackle the complex challenges posed by online threats. Users can report suspicious activity they encounter, while administrators have the capability to monitor, assess, and act against harmful content and users. In summary, the development and implementation of tools and systems like the Django-based application described here are instrumental in addressing the persistent challenges posed by online child predators and cyber harassers in our contemporary digital landscape. By combining technology, user involvement, and regulatory compliance, these tools strive to create safer online environments, particularly for children and other vulnerable individuals, while upholding the principles of privacy and security.

CHAPTER 2

LITERATURE SURVEY

Online harassment has been a pervasive issue since the early days of social media, and it still exists today. These studies began with an attempt to develop an automated system to detect and report this type of misconduct. Studies have been started to reduce or detect cases of sexual harassment and protect children from bullying to create a safe environment using two approaches: machine learning and deep learning. In this study [4], the authors tracked the occurrence of cyberbullying on social media, using fuzzy logic and genetic algorithms. They identified and classified cyberbullying words and activities on social media, including inflammatory, harassing, racist, and terroristic comments. The resulting F-measure was 0.91. A genetic algorithm is used to optimize parameters and achieve correct performance.

In Facebook message filtering, the authors in ref [5] used three weighting schemes, namely entropy, term frequency-inverse document frequency (TFIDF), and modified TF-IDF was used as feature selection. A Support Vector Machine (SVM) was used to measure the accuracy, precision, and recall of a support vector. The test results indicated that the modified TF-IDF, with an accuracy of 96.50%, outperforms the other schemes. This work in [6] was carried out to compare supervised machine learning (ML) algorithms for natural language learning and assessment of online harassment in Twitter messages as part of social media competition and harassment (a feature). The feature extraction was done by TF-IDF and Word2Vec embeddings. The results accurately included over (80%) of all types of harassment considered within the data. This study [7] combines feeling analysis with a modern approach to sentencing vectors. The language pattern of Long-Short-Term-Memory, Recurrent Neural Network (LSTM_RNN) is used as a new means of predators Sexual Identification to produce word vectors. A record-breaking accuracy rate has been achieved in the last phase of determining the feeling value from the SoftMax layer outputs, with a recall of 81.10%.

The authors in ref. [8] use convolution neural networks (CNN) for extracting features from tags and creating a Twitter post classification model for malicious intent. They analysed a four-month Twitter dataset to examine the narrative contexts that expressed malicious intent. They talked about the significance of such cases in developing gender-based violence regulations.

Sweta Karlekar in [9] presents the work of the Safe City Web Community to categorize and analyse different types of sexual harassment. Safe City Web uses this information to help victims compile web directories, provide more detailed safety advice services by sharing their accounts, and help individuals uncover relevant cases to prevent further sexual violence. The single-label CNN-RNN model achieves 86.5% accuracy in processing, linking, and annotating tags. Espinoza [10] uses Twitter for a new data set with four classes of harassment detection. They applied two various deep learning architectures (CNN and LSTM) to classify the tweets. When training the data, the measurement for F1 was equal to 55 percent, while the results for the test set alone hit 46 percent for F1.

Arijit Josh Chowdhury [11] proposes a language model for disclosure. The ULMFiT fine-tuning architecture consists of a language model, a specific mediator (Twitter), and a task-specific classifier. The complete comparison shows the benefits of using particular and lightweight LSTM-based mean language models and an improved vocabulary reflecting linguistic nuances in the deep text that challenges sexual harassment. The neural network models with high results demonstrated about 10,000 personal sexual harassment stories annotated to extract core elements and automatically categorize the stories. Additional categorization improvements were accomplished through the details of key features with an accuracy of 92.9%. This author in [12] provides an automatic method for analysing the text of online communication and determining if a cyber-predator's prey is another user or one of the participants. Classification tasks provide a RNN (recurrent neural network) in each stage, which achieved an F0.5 score of 0.9058. In [13], the authors provide a novel way to classify sexual harassment and sexual predators in English using BiLSTM with Gated Recurrent Unit (GRU) algorithms and word embedding. The pre-trained Global Vectors (GloVe) words achieved 97.27%, compared with Extreme Gradient Boosting (XGBoost), which reached 90.10%. In [14] The study presents a two-stage method for identifying sexual predators in online chats using machine learning classifiers, with the best method, soft voting and Naive Bayes, achieving an F 0.5 score of 0.9348. [15] It is the most popular dataset for online chat predatory detection related tasks and there are many recent related works use this dataset. [16] The study proposes a method to detect predatory online conversations, aiming to protect children from cyber-security issues like child grooming, using Natural Language Processing and linear SVM. [17] The study explores early detection of sexual predators in online chat conversations using message-based, author-based, and conversation-based approaches, with Neural Network classifiers showing promising results. [18] A pilot study demonstrates the effectiveness of automatic

text categorization techniques in identifying online sexual predators, with a k-NN classifier achieving an f-measure of 0.943. [19] The paper introduces a decision-making method for document classification in grooming attack recognition, comparing seven algorithms to address the complexity of identifying potential hazards for minor users.[20] Pedophilia is defined as the sexual interest of adults in minors, and obviously, age plays an important role in this aspect.

CHAPTER 3

EXISTING MODEL

3.1 Naive Bayes algorithm

Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature. The presence or absence of a feature does not affect the presence or absence of the other feature.

How Naive Bayes works?

Naive Bayes is a powerful algorithm that is used for text data analysis and with problems with multiple classes. To understand Naive Bayes theorem's working, it is important to understand the Bayes theorem concept first as it is based on the latter.

Bayes theorem, formulated by Thomas Bayes, calculates the probability of an event occurring based on the prior knowledge of conditions related to an event. It is based on the following formula:

$$P(A|B) = P(A) * P(B|A)/P(B)$$

Where we are calculating the probability of class A when predictor B is already provided.

$P(B)$ = prior probability of B

$P(A)$ = prior probability of class A

$P(B|A)$ = occurrence of predictor B given class A probability

3.2 Drawbacks of Existing system

The Naive Bayes algorithm has the following disadvantages:

- The prediction accuracy of this algorithm is lower than the other probability algorithms.
- It is not suitable for regression. Naive Bayes algorithm is only used for textual data classification and cannot be used to predict numeric values.

CHAPTER 4

PROPOSED METHODOLOGY

4.1 Overview

This project implements a web application using the Django framework. The application appears to be related to the identification of online child predators and cyber harassers in social media environments. In addition, this project represents the backend logic of a web application designed for identifying and monitoring online child predators and cyber harassers in social media. Users can register, log in, send posts, and run machine learning algorithms to classify text messages as potentially harmful or not. The results are presented on web pages for users and administrators to monitor.

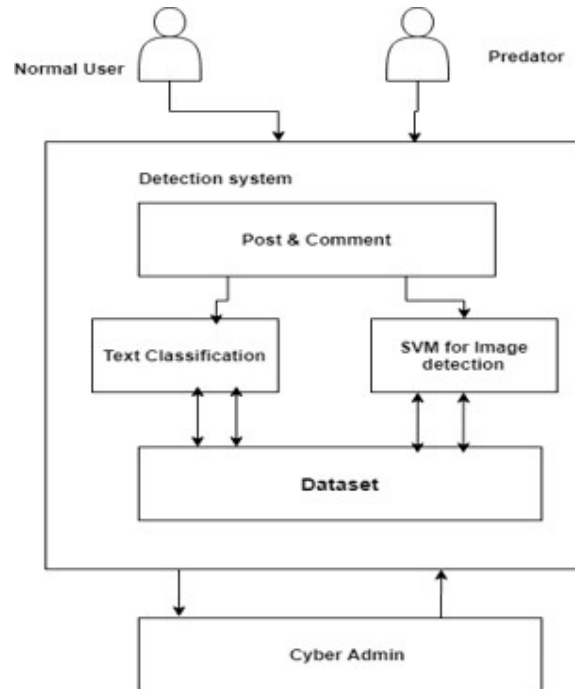


Figure 4.1: Proposed system architecture.

The main components and functionalities are as follows:

- **Import Statements:** It starts by importing various Python libraries and Django modules that will be used throughout the application. These libraries include tools for database access, data processing, machine learning, and web development
- **Global Variables:** Several global variables are declared at the beginning. These include classifier, label-count, X, Y, and corpus, which will be used for machine learning and data processing purposes.
- **Django Views:** It defines several Django views, each associated with a specific URL endpoint. These views include functions like index, Send Post, Register, Admin, Login, Add Cyber Messages, Run Algorithms, Monitor Post, Add Bullying Words, Signup, User Login, Admin Login, View Users, View User Post, word-count, prediction, cal-accuracy, and classify Post. These views handle HTTP requests and render HTML templates for different pages of the web application.
- **Database Access:** This establishes a connection to a MySQL database using the pymysql library. It interacts with the database to retrieve and insert data, such as user information and posts.
- **Machine Learning:** This contains machine learning algorithms. It loads and preprocesses a dataset from a file named 'dataset.txt,' performs text preprocessing, and trains machine learning models (e.g., SVM, Decision Tree, K-Nearest Neighbors, Random Forest, Naive Bayes) to classify text data. The selected model is stored in the classifier variable for later use.
- **Web Forms Handling:** Functions like Add Bullying Words, Signup, User Login, and Admin Login handle form submissions from users and perform actions such as adding data to the database or verifying user credentials.
- **File Upload:** This handles file uploads, such as user profile pictures and text files containing messages to be classified.
- **HTML Templates:** The web application uses HTML templates (e.g., 'index.html', 'SendPost.html', 'Register.html', 'Admin.html') to render the user interface.
- **Data Processing:** This will preprocess text data by removing special characters, converting text to lowercase, and tokenizing words.
- **Classification:** The machine learning models are used to classify text messages, and the results are displayed to the user.
- **Session Management:** The script manages user sessions, storing the username in a file named 'session.txt' after successful login.

- **Result Presentation:** The classification results and other relevant information are presented to the user in HTML templates.

4.2 Data Preprocessing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task. A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set

Importing Libraries: To perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

Numpy: Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

```
import numpy as nm
```

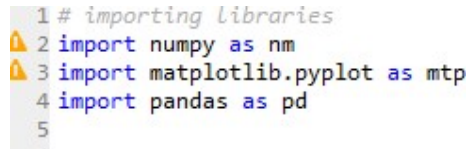
Here we have used nm, which is a short name for Numpy, and it will be used in the whole program.

Matplotlib: The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub-library pyplot. This library is used to plot any type of charts in Python for the code. It will be imported as below:

```
import matplotlib.pyplot as mpt
```

Here we have used mpt as a short name for this library.

Pandas: The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. Here, we have used pd as a short name for this library. Consider the below image:



```
1 # importing libraries
2 import numpy as nm
3 import matplotlib.pyplot as mtp
4 import pandas as pd
5
```

Handling Missing data: The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset. There are mainly two ways to handle missing data, which are:

- By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.
- By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.

Encoding Categorical data: Categorical data is data which has some categories such as, in our dataset; there are two categorical variables, Country, and Purchased. Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So, it is necessary to encode these categorical variables into numbers.

Feature Scaling: Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no variable dominates the other variable. A machine learning model is based on Euclidean distance, and if we do not scale the variable, then it will cause some issue in our machine learning model. Euclidean distance is given as:

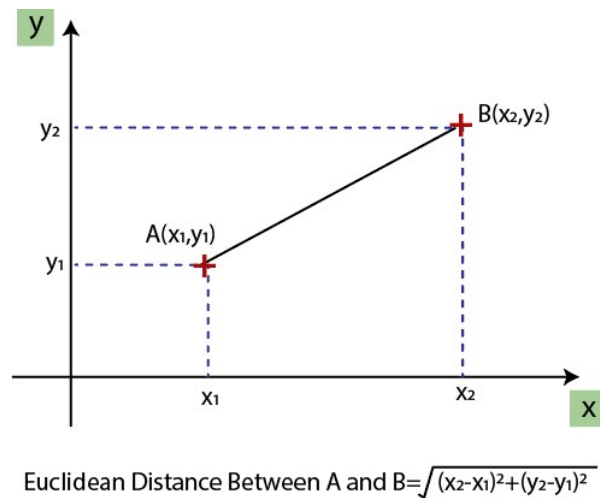


Figure 4.2: Feature scaling

If we compute any two values from age and salary, then salary values will dominate the age values, and it will produce an incorrect result. So, to remove this issue, we need to perform feature scaling for machine learning.

4.3 Splitting the Dataset

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

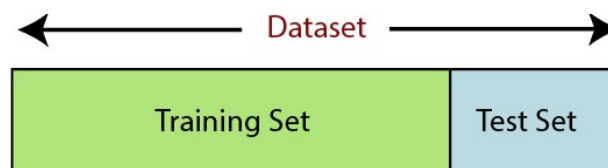


Figure 4.3: Splitting the dataset.

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```

Explanation

- In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
- In the second line, we have used four variables for our output that are
 - x_train: features for the training data
 - x_test: features for testing data
 - y_train: Dependent variables for training data
 - y_test: Independent variable for testing data
- In train_test_split () function, we have passed four parameters in which first two are for arrays of data, and test_size is for specifying the size of the test set. The test_size maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.
- The last parameter random_state is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

4.4 TF-IDF Feature Extraction

TF-IDF, short for Term Frequency-Inverse Document Frequency, is a commonly used technique in NLP to determine the significance of words in a document or corpus. To give some background context, a survey conducted in 2015 showed that 83% of text-based recommender systems in digital libraries use TF-IDF for extracting textual features. That's how popular the technique is. Essentially, it measures the importance of a word by comparing its frequency within a specific document with the frequency to its frequency in the entire corpus. The underlying assumption is that a word that occurs more frequently within a document but rarely in the corpus is particularly important in that document.

Mathematical formula for calculating TF-IDF

TF (Term Frequency) is determined by calculating the frequency of a word in a document and dividing it by the total number of words in the document.

- $TF = (\text{Number of times the word appears in the document}) / (\text{Total number of words in the document})$
- IDF (Inverse Document Frequency), on the other hand, measures the importance of a word within the corpus. It is calculated as:
- $IDF = \log ((\text{Total number of documents in the corpus}) / (\text{Number of documents containing the word}))$

The tf-idf value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general. TF-IDF use two statistical methods, first is Term Frequency and the other is Inverse Document Frequency. Term frequency refers to the total number of times a given term t appears in the document doc against (per) the total number of all words in the document and the inverse document frequency measure of how much information the word provides. It measures the weight of a given word in the entire document. IDF show how common or rare a given word is across all documents. TF-IDF can be computed as $tf * idf$

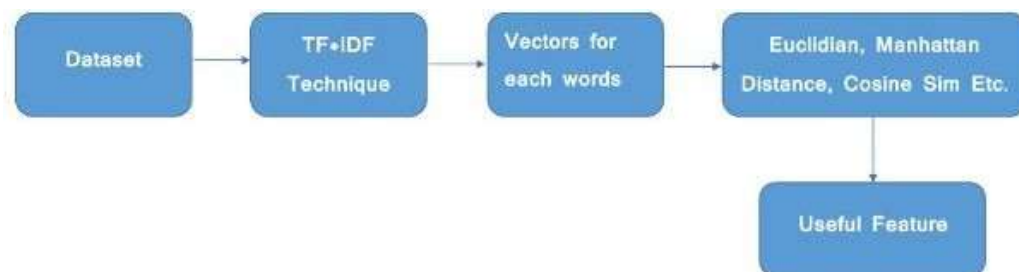


Fig. 4.4: TF-IDF block diagram.

TF-IDF do not convert directly raw data into useful features. Firstly, it converts raw strings or dataset into vectors and each word has its own vector. Then we'll use a particular technique for retrieving the feature like Cosine Similarity which works on vectors, etc.

Terminology

t — term (word)

d — document (set of words)

N — count of corpus

corpus — the total document set

Step 1: Term Frequency (TF): Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, “Data Science is awesome!” A simple way to start out is by eliminating documents that do not contain all three words “Data” is, “Science”, and “awesome”, but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency. The weight of a term that occurs in a document is simply proportional to the term frequency.

$$tf(t, d) = \text{count of } t \text{ in } d / \text{number of words in } d$$

Step 2: Document Frequency: This measures the importance of document in whole set of corpora, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d , whereas DF is the count of occurrences of term t in the document set N . In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

$$df(t) = \text{occurrence of } t \text{ in documents}$$

Step 3: Inverse Document Frequency (IDF): While computing TF, all terms are considered equally important. However, it is known that certain terms, such as “is”, “of”, and “that”, may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. The IDF is the inverse of the document frequency which measures the informativeness of term t . When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as “is” is present in almost all of the documents, and N/df

will give a very low value to that word). This finally gives what we want, a relative weightage.

$$idf(t) = N/df$$

Now there are few other problems with the IDF, in case of a large corpus, say 100,000,000, the IDF value explodes, to avoid the effect we take the log of idf. During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

$$idf(t) = \log(N/(df + 1))$$

The TF-IDF now is at the right measure to evaluate how important a word is to a document in a collection or corpus. Here are many different variations of TF-IDF but for now let us concentrate on this basic version.

$$tf - idf(t, d) = tf(t, d) * \log(N/(df + 1))$$

Step 4: Implementing TF-IDF: To make TF-IDF from scratch in python, let's imagine those two sentences from different document:

first sentence: "Data Science is the sexiest job of the 21st century".

second sentence: "machine learning is the key for data science".

4.5 SVM Classifier

SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as SVM. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

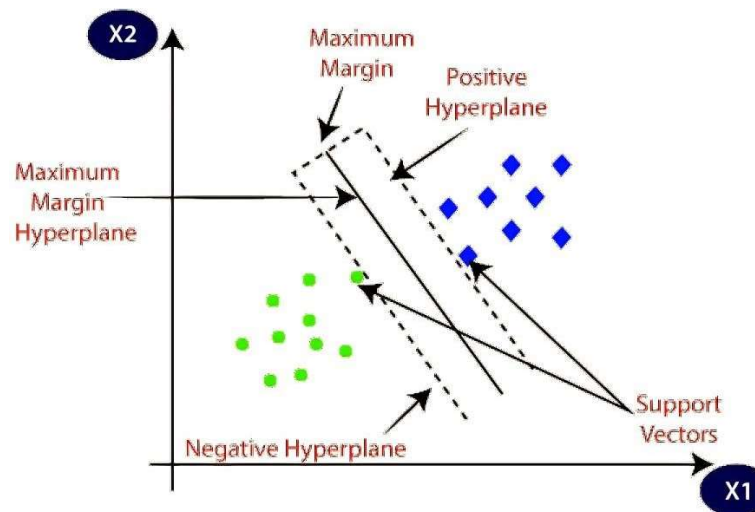


Figure 4.5: SVM working.

SVM plays a pivotal role in the detection of spam comments on YouTube when combined with TF-IDF (Term Frequency-Inverse Document Frequency) extracted features. This collaborative approach begins with the transformation of comment text into a numerical format through TF-IDF vectorization. TF-IDF assigns numerical values to individual terms (words) within a comment, signifying their significance within that comment relative to a broader dataset of comments. Consequently, each comment is represented as a high-dimensional vector, with each dimension corresponding to a unique term and the value in each dimension indicating the TF-IDF score of that term in the comment. The subsequent step involves data preparation, where a labelled dataset comprising TF-IDF feature vectors, and their respective labels (harassment or non-harassment) is readied for SVM model training. This dataset is typically divided into two subsets: a training set for model training and a testing set for evaluation purposes.

The SVM model is then trained using the training set. SVM is a binary classifier that seeks to identify a hyperplane within the high-dimensional TF-IDF feature space that effectively separates harassment comments from non-harassment comments. The objective is to maximize the margin between the hyperplane and the nearest data points of both classes, which are referred to as support vectors. The SVM utilizes the TF-IDF feature vectors of the training set comments, along with their labels, to determine the optimal hyperplane. The choice of the SVM kernel function is crucial, as it maps the data into a higher-dimensional space where a linear separator can be found.

For classification of new, unseen comments, the TF-IDF vectorization process is applied to generate TF-IDF feature vectors. These vectors are then fed into the trained SVM, which predicts whether each comment is spam or not based on the learned hyperplane. The decision boundary provided by the hyperplane dictates the classification of each comment as either spam or non-spam. To gauge the effectiveness of the SVM model, its predictions on the testing set are compared against the true labels. Various evaluation metrics, such as accuracy, precision, recall, F1-score, and ROC curves, are employed to measure the model's accuracy in correctly identifying spam comments and minimizing false positives.

4.6 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

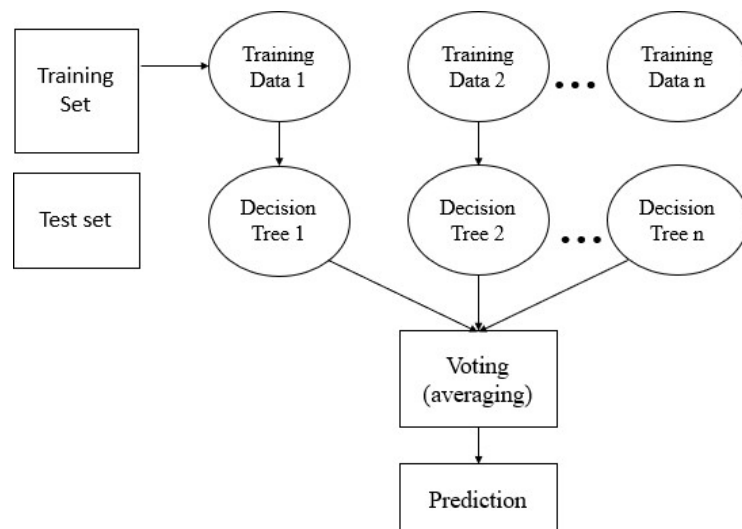


Fig. 4.6: Random Forest algorithm.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Operation

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

Important Features of Random Forest

- **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
- **Immune to the curse of dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
- **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
- **Train-Test split**- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
- **Stability**- Stability arises because the result is based on majority voting/ averaging.

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.
- Below are some points that explain why we should use the Random Forest algorithm
- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

Types of Ensembles

Before understanding the working of the random forest, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest. Bagging, also known as Bootstrap Aggregation, is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

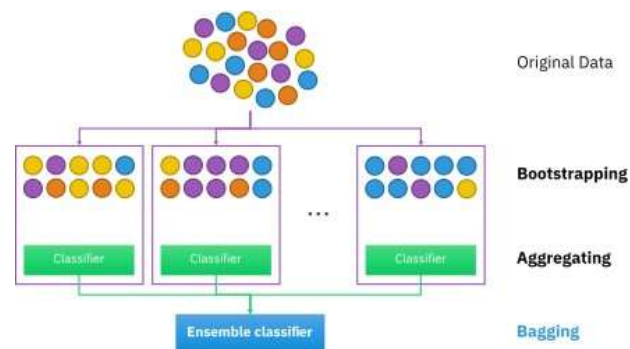


Fig. 4.7: RF Classifier analysis.

Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

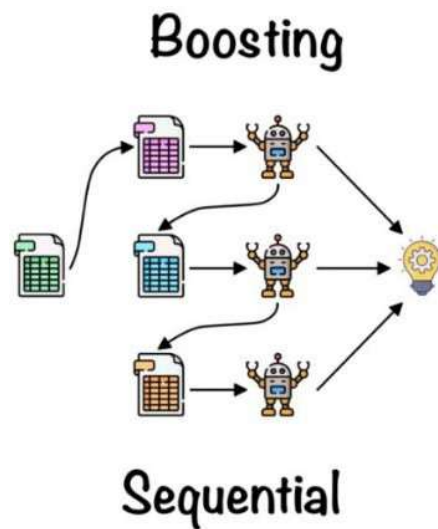


Fig. 4.8: Boosting RF Classifier

CHAPTER 5

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

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

GOALS: The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

Class Diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an “is-a” or “has-a” relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed “methods” of the class. Apart from this, each class may have certain “attributes” that uniquely identify the class.

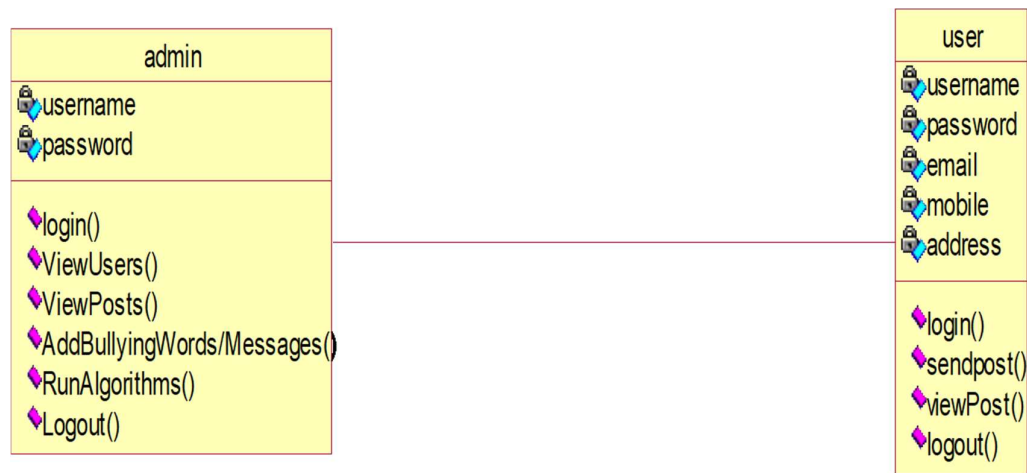


Figure 5.1: Class Diagram

Collaboration Diagram

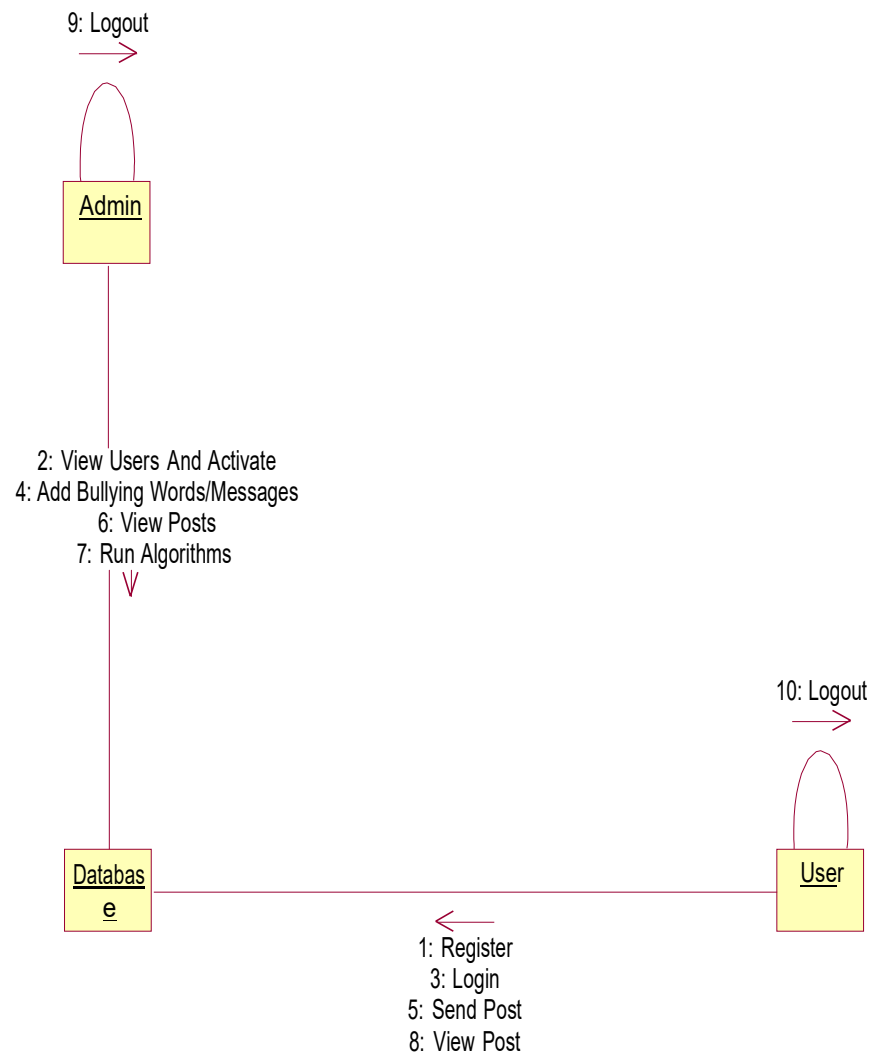


Figure 5.2: Collaboration Diagram

Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (“lifelines”), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

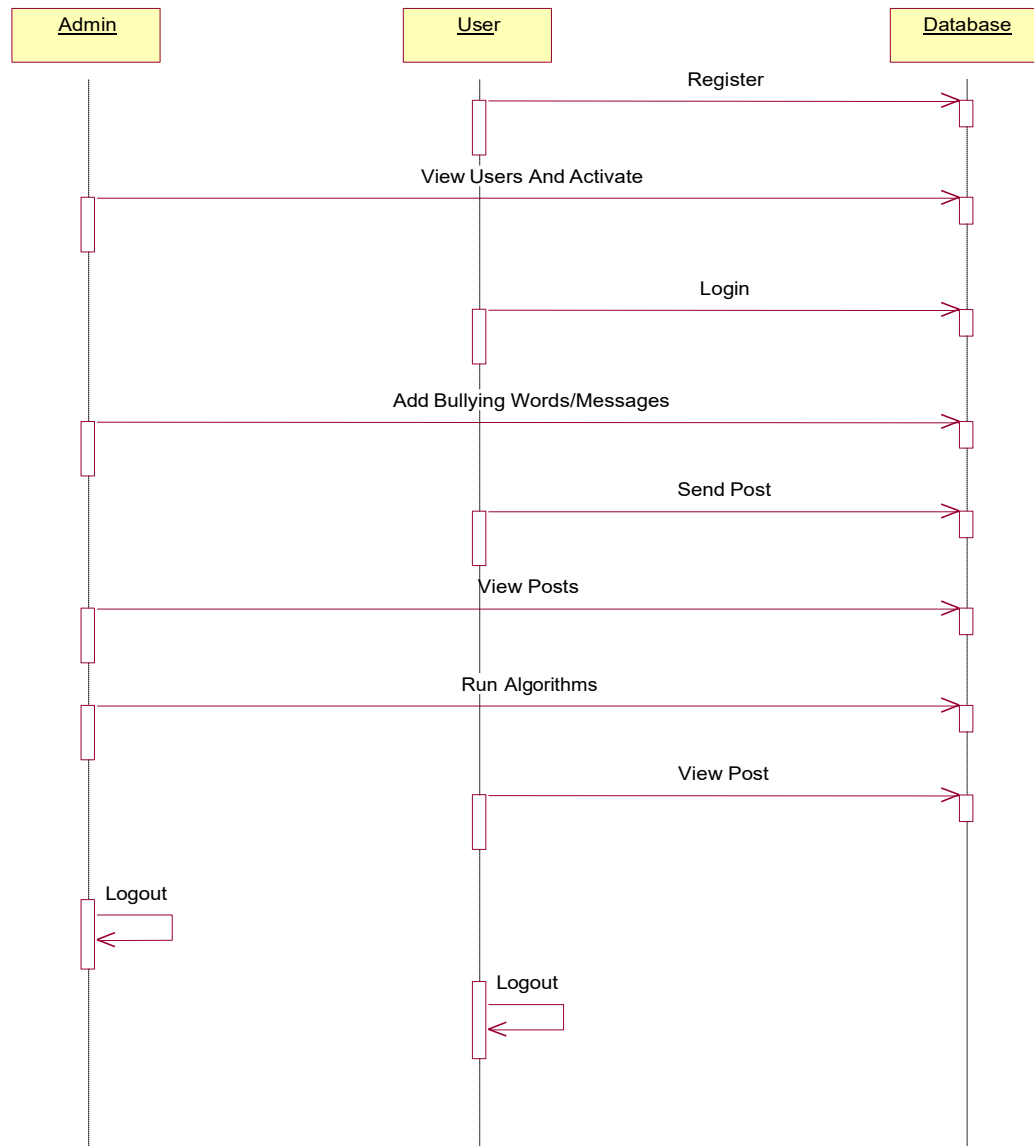
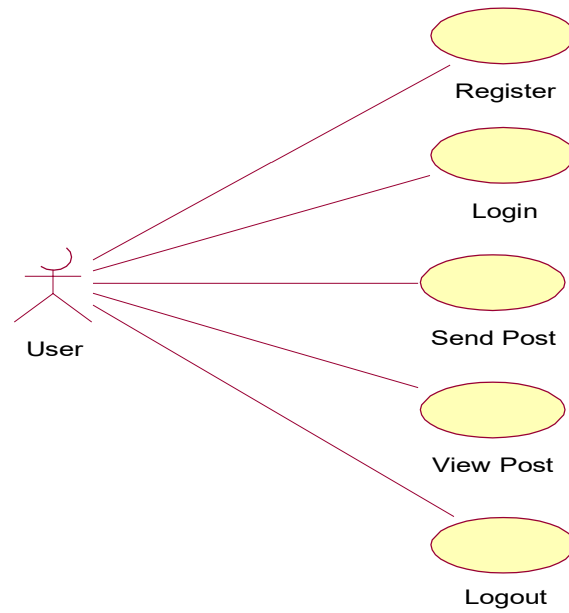


Fig 5.3: Sequence Diagram

Use Case Diagram: The purpose of use case diagram is to capture the dynamic aspect of a system.

User



Admin

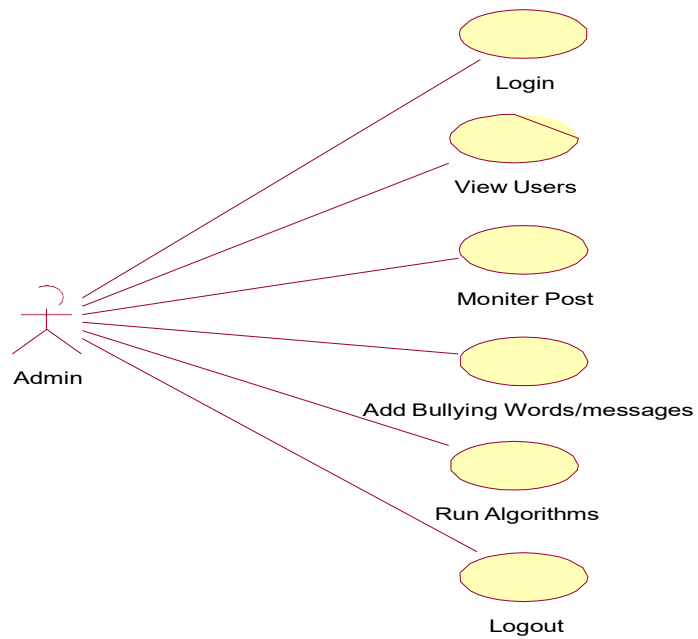


Fig 5.4: Use Case diagram

CHAPTER 6

SOFTWARE ENVIRONMENT

What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.
- The biggest strength of Python is huge collection of standard library which can be used for the following –
 - Machine Learning
 - GUI Applications (like Kivy, Tkinter, PyQt etc.)
 - Web frameworks like Django (used by YouTube, Instagram, Dropbox)
 - Image processing (like Open cv, Pillow)
 - Web scraping (like Scrappy, BeautifulSoup, Selenium)
 - Test frameworks
 - Multimedia

Advantages of Python

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally

build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we have seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple No, we're not kidding.

Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 3.0:

Print is now a function.

- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Modules Used in Project

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyse. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

- Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

- For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet [here](#). The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Figure 6.1: Python Installation and Setup

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Fig. 6.2: Python Download

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 3.7.18	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Fig. 6.3: Python Version Selection

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	OS
Unipped source tarball	Source release		68111671e5b3db4ae770baf010f09be	23017663	36
XZ compressed source tarball	Source release		d13e4aee6297051c3bc445ee3044803	17131432	36
macOS 64-bit installer	Mac OS X	for Mac OS X 10.5 and later	6428b4f975d3daf71a42cbafce0b6e	34898416	36
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5d8603c38217a45723b5e4a938d141f	28082848	36
Windows 32-bit	Windows		08399573a2e56b2ac5acade0b47ed2	8111761	36
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64/x64	9800c0e3a5e0b0b0e0c3a503e5c3400	7954381	36
Windows x86-64 executable installer	Windows	for AMD64/EM64/x64	4702b4b4d75ab0d030c3a503e5c3400	20483368	36
Windows x86-64 web-based installer	Windows	for AMD64/EM64/x64	20c914608bdf73aee053a3baf311b4bd1	1362904	36
Windows x86 embeddable zip file	Windows		9fab1b8138841879fa94123574129d8	6741626	36
Windows x86 executable installer	Windows		71c1802942d5444a5a5b451476294788	25663848	36
Windows x86 web-based installer	Windows		1d670cfa5d117d812c30943ea371887c	1224608	36

Fig. 6.4: Version with Python its Corresponding OS

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Fig. 6.5: Opening the Downloaded File

Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.



Fig. 6.6: Python Installation

Step 3: Click on Install NOW After the installation is successful. Click on Close.



Fig. 6.7: Python Installation Setup Prompt

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

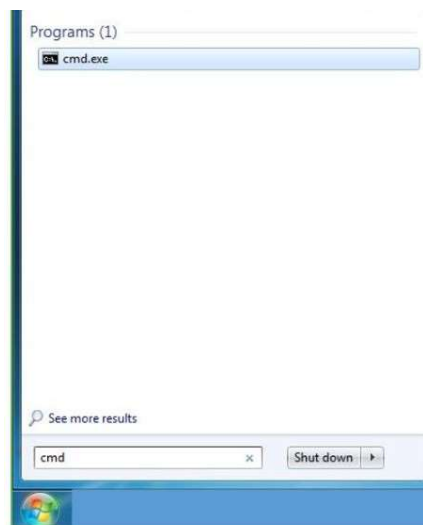
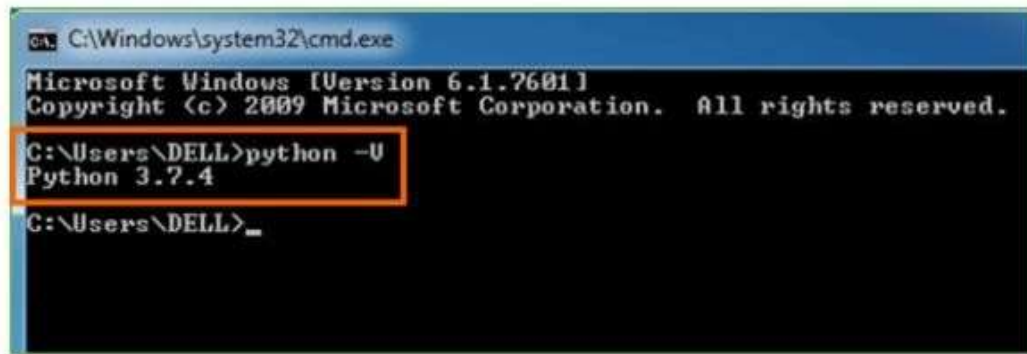


Fig. 6.8: Opening cmd

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type `python -V` and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python -U
Python 3.7.4

C:\Users\DELL>_
```

Fig. 6.9: Checking for Python in cmd

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

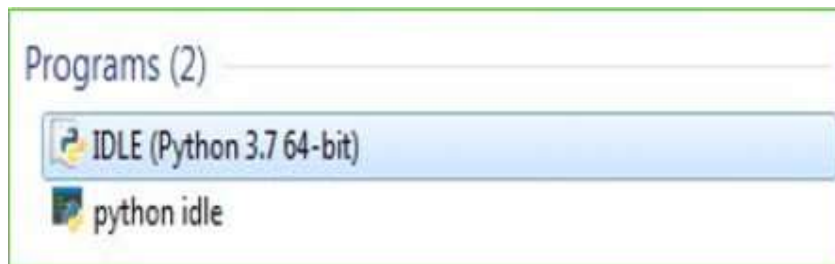


Fig. 6.10: Opening IDLE Interpreter

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File >

Click on Save

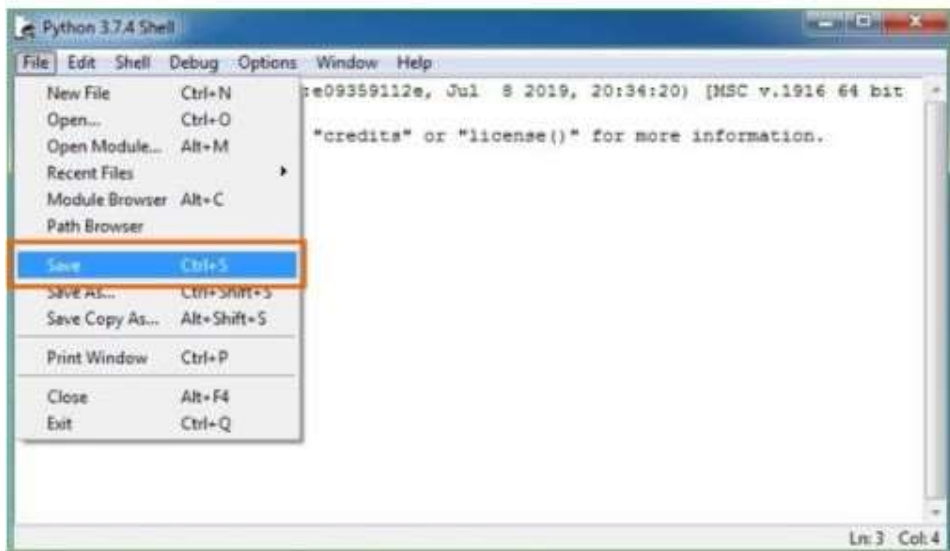


Fig. 6.11: Saving the Python File

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print ("Hey World") and Press Enter.

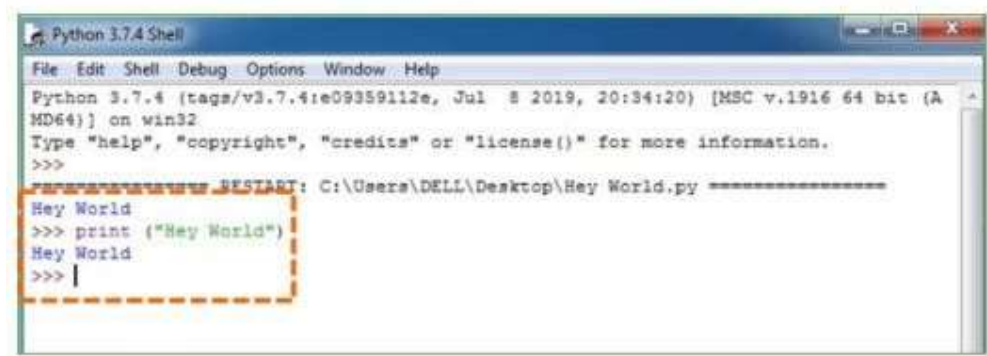


Fig. 6.12: Executing the Program

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

CHAPTER 7

SYSTEM REQUIREMENTS

7.1 SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

7.2 HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- Operating system : Windows, Linux
- Processor : minimum intel i3
- Ram : minimum 4 GB
- Hard disk : minimum 250GB

CHAPTER 8

FUNCTIONAL REQUIREMENTS

Output Design

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

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

Output Definition

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

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

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

Input Design

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

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

Input Stages

The main input stages can be listed as below:

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

Input Types

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

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

Input Media

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

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

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

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

Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Classified As:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Interfaces

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

Performance Requirements

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

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

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

Rank	Participation	Precision	Recall	F1	F0.5
	Our best method	0.9561	0.8583	0.9046	0.9348
1	Villatoro-Tello et al. [9]	0.9804	0.7874	0.8734	0.9346
2	Snider [22]	0.9839	0.7205	0.8318	0.9168
3	Bours and Kulsrud [23]	0.8910	0.8700	0.8800	0.8870
4	Parapar et al. [6]	0.9392	0.6693	0.7816	0.8691

Table. 8.1: Top Performance

CHAPTER 9

SOURCE CODE

```
from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

import pymysql

from django.http import HttpResponse

from django.conf import settings

from django.core.files.storage import FileSystemStorage

import datetime

import matplotlib.pyplot as plt

import re

import numpy as np

from sklearn import svm

import pandas as pd

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

import nltk

from nltk.corpus import stopwords

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix
```



```

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive_bayes import MultinomialNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn_extensions.extreme_learning_machines.elm import GenELMClassifier

from sklearn_extensions.extreme_learning_machines.random_layer import
RBFRandomLayer, MLPRandomLayer

from sklearn.linear_model import LogisticRegression

from django.core.files.storage import FileSystemStorage

import datetime

from numpy.linalg import norm

from numpy import dot

global classifier

global label_count

global X

global Y

corpus = []

def index(request):

    if request.method == 'GET':

        return render(request, 'index.html', {})

def SendPost(request):

    if request.method == 'GET':

        return render(request, 'SendPost.html', {})

def Register(request):

```

```

    if request.method == 'GET':

        return render(request, 'Register.html', {})

def Admin(request):

    if request.method == 'GET':

        return render(request, 'Admin.html', {})

def Login(request):

    if request.method == 'GET':

        return render(request, 'Login.html', {})

def AddCyberMessages(request):

    if request.method == 'GET':

        return render(request, 'AddCyberMessages.html', {})

def RunAlgorithms(request):

    if request.method == 'GET':

        return render(request, 'RunAlgorithms.html', {})

def MonitorPost(request):

    if request.method == 'GET':

        strdata = '<table border=1 align=center width=100%><tr><th>Sender  
Name</th><th>File Name</th><th>Message</th><th>Post Time</th><th>Status</th></tr><tr>'

        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
database = 'cyber',charset='utf8')

    with con:

        cur = con.cursor()

        cur.execute("select * FROM posts")

        rows = cur.fetchall()

        for row in rows:

```

```

        strdata+='<td>'+str(row[0])+'</td><td><img        src=/static/photo/'+str(row[1])+'
width=200
height=200></img></td><td>'+str(row[2])+'</td><td>'+str(row[3])+'</td><td>'+str(row
[4])+'</td></tr>'

```

```

    context= {'data':strdata}

```

```

return render(request, 'MonitorPost.html', context)

```

```

def AddBullyingWords(request):

```

```

    if request.method == 'POST':

```

```

        message = request.POST.get('t1', False)

```

```

        label = request.POST.get('t2', False)

```

```

        message = message.strip("\n")

```

```

        message = message.strip()

```

```

        message = message.lower()

```

```

        message = re.sub(r'^a-zA-Z\s]+', "", message)

```

```

        file = open('dataset.txt','a+')

```

```

        file.write(message+" "+label+"\n")

```

```

        file.close()

```

```

        context= {'data':'Cyber Words added to dataset as '+label}

```

```

return render(request, 'AddCyberMessages.html', context)

```

```

def Signup(request):

```

```

    if request.method == 'POST':

```

```

        username = request.POST.get('t1', False)

```

```

        password = request.POST.get('t2', False)

```

```

        contact = request.POST.get('t3', False)

```

```

        email = request.POST.get('t4', False)

```

```

        address = request.POST.get('t5', False)

```

```

db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password
= 'root', database = 'cyber',charset='utf8')

db_cursor = db_connection.cursor()

student_sql_query = "INSERT INTO
users(username,password,contact_no,email,address,status)
VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"','Accept
ed')"

db_cursor.execute(student_sql_query)

db_connection.commit()

print(db_cursor.rowcount, "Record Inserted")

if db_cursor.rowcount == 1:

    context= {'data':'Signup Process Completed'}

    return render(request, 'Register.html', context)

else:

    context= {'data':'Error in signup process'}

    return render(request, 'Register.html', context)

def UserLogin(request):

if request.method == 'POST':

    username = request.POST.get('t1', False)

    password = request.POST.get('t2', False)

    index = 0

    con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
database = 'cyber',charset='utf8')

    with con:

        cur = con.cursor()

        cur.execute("select * FROM users")

```

```

rows = cur.fetchall()

for row in rows:

    if row[0] == username and password == row[1] and row[5] == 'Accepted':

        index = 1

        break

if index == 1:

    file = open('session.txt','w')

    file.write(username)

    file.close()

    context= {'data':'welcome '+username}

    return render(request, 'UserScreen.html', context)

else:

    context= {'data':'login failed'}

    return render(request, 'Login.html', context)

def AdminLogin(request):

if request.method == 'POST':

    username = request.POST.get('t1', False)

    password = request.POST.get('t2', False)

    if username == 'admin' and password == 'admin':

        context= {'data':'welcome '+username}

        return render(request, 'AdminScreen.html', context)

    else:

        context= {'data':'login failed'}

        return render(request, 'Admin.html', context)

def ViewUsers(request):

```

```

if request.method == 'GET':

    color='<font size="" color=black>'

    strdata          =          '<table          border=1          align=center
width=100%><tr><th>Username</th><th>Password</th><th>Contact
No</th><th>Email ID</th><th>Address</th><th>Status</th></tr><tr>'

    con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
database = 'cyber',charset='utf8')

    with con:

        cur = con.cursor()

        cur.execute("select * FROM users")

        rows = cur.fetchall()

        for row in rows:

strdata+='<td>'+color+row[0]+'</td><td>'+color+row[1]+'</td><td>'+color+row[2]+'</t
d><td>'+color+str(row[3])+'</td><td>'+color+str(row[4])+'</td><td>'+color+row[5]+'</
td></tr>'

        context= {'data':strdata+'</table><br/><br/><br/>'}

        return render(request, 'ViewUsers.html', context)

def ViewUserPost(request):

    if request.method == 'GET':

        strdata      =      '<table      border=1      align=center      width=100%><tr><th>Sender
Name</th><th>File      Name</th><th>Message</th><th>Post      Time</th>
<th>Status</th></tr><tr>'

        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
database = 'cyber',charset='utf8')

        with con:

            cur = con.cursor()

            cur.execute("select * FROM posts")

```

```

        rows = cur.fetchall()

        for row in rows:

            strdata+='<td>'+str(row[0])+'</td><td><img      src=/static/photo/'+str(row[1])+'
width=200
height=200></img></td><td>'+str(row[2])+'</td><td>'+str(row[3])+'</td><td>'+str(row
[4])+'</td></tr>'

            context= {'data':strdata}

            return render(request, 'ViewUserPost.html', context)

def word_count(str):

    counts = dict()

    words = str.split()

    for word in words:

        if word in counts:

            counts[word] += 1

        else:

            counts[word] = 1

    return counts

def prediction(X_test, cls): #prediction done here

    y_pred = cls.predict(X_test)

    for i in range(len(X_test)):

        print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))

    return y_pred

def cal_accuracy(y_test, y_pred, details):

    msg = "

    cm = confusion_matrix(y_test, y_pred)

    accuracy = accuracy_score(y_test,y_pred)*100

```

```

msg+=details+"<br/>"

msg+="Accuracy : "+str(accuracy)+"<br/>"

#msg+="Report : "+str(classification_report(y_test, y_pred))+"<br/>"

#msg+="Confusion Matrix : "+str(cm)+"<br/>"

return msg

def classifyPost(vec1, vec2):

    vector1 = np.asarray(vec1)

    vector2 = np.asarray(vec2)

    return dot(vector1, vector2)/(norm(vector1)*norm(vector2))

def PostSent(request):

    if request.method == 'POST' and request.FILES['t2']:

        output = "

        myfile = request.FILES['t2']

        msg = request.POST.get('t1', False)

        fs = FileSystemStorage()

        filename = fs.save('D:\2023 Mini\SMCE\CSE\Final projects\B12. Identification of

        Child Predators and Cyber Harassers\CyberBullying\static\photo/'+str(myfile),

        myfile)

        now = datetime.datetime.now()

        current_time = now.strftime("%Y-%m-%d %H:%M:%S")

        text = 'msg,label\n'+msg+"?"

        f = open("test.txt", "w")

        f.write(text)

        f.close()

        df = pd.read_csv('test.txt')

```



```

X1 = df.iloc[:, :-1].values

dataset = ""

for k in range(len(corpus)):

    dataset+=corpus[k]+", "

dataset = dataset[0:len(dataset)-1]

dataset+="\n"

for i in range(len(X1)):

    line = str(X1[i]).strip('\n')

    line = line.strip()

    line = line.lower()

    line = re.sub(r'^a-zA-Z\s|'+', ', line)

    print(line)

    wordCount = word_count(line.strip())

    value = ""

    for j in range(len(corpus)):

        if corpus[j] in wordCount.keys():

            value+=str(wordCount[corpus[j]])+", "

        else:

            value+="0,"

    value = value[0:len(value)-1]

    dataset+=value+"\n"

    print(dataset)

f = open("test.txt", "w")

f.write(dataset)

f.close()

```

```

test = pd.read_csv("test.txt")

X1 = test.values[:, 0:label_count]

result = classifier.predict(X1)

#classify = 0

#score = 0

#for i in range(len(X)):

#    cosine = classifyPost(X[i], X1)

#    if cosine > score:

#        score = cosine

#        classify = Y[i]

status = 'Non-Cyber Harassers'

print(result)

if result[0] == 0:

    status = 'Non-Cyber Harassers'

else:

    status = 'Cyber Harassers'

    user = "

with open("session.txt", "r") as file:

    for line in file:

        user = line.strip('\n')

db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password

= 'root', database = 'cyber',charset='utf8')

db_cursor = db_connection.cursor()

student_sql_query = "INSERT INTO posts(sender,filename,msg,posttime,status)

VALUES('"+user+"','"+str(myfile)+"','"+msg+"','"+current_time+"','"+status+"')"
```

```

db_cursor.execute(student_sql_query)

db_connection.commit()

print(db_cursor.rowcount, "Record Inserted")

if db_cursor.rowcount == 1:

    context= {'data': 'Posts details added'}

    return render(request, 'SendPost.html', context)

else:

    context= {'data': 'Error in adding post details'}

    return render(request, 'SendPost.html', context)

def RunAlgorithm(request):

global classifier

global label_count

global X

global Y

msg = "

if request.method == 'POST':

    name = request.POST.get('t1', False)

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

    corpus.clear()

    posts = []

    df = pd.read_csv('dataset.txt')

    X = df.iloc[:, :-1].values

    Y = df.iloc[:, -1].values

    for i in range(len(Y)):

        if Y[i] == 'Non-Bullying':

```

```

        Y[i] = 0

    else:

        Y[i] = 1

for i in range(len(X)):

    line = str(X[i]).strip('\n')

    line = line.strip()

    line = line.lower()

    line = re.sub(r'^a-zA-Z\s|'+', ', line)

    arr = line.split(" ")

    for k in range(len(arr)):

        word = arr[k].strip("\n").strip()

        if len(word) > 2 and word not in corpus and word not in stop_words:

            corpus.append(word)

    posts.append(line)

    dataset = "

for k in range(len(corpus)):

    dataset+=corpus[k]+", "

    dataset+='Label\n'

for k in range(len(posts)):

    text = posts[k];

    wordCount = word_count(text.strip())

    for j in range(len(corpus)):

        if corpus[j] in wordCount.keys():

            dataset+=str(wordCount[corpus[j]])+", "

        else:

```

```

        dataset+="0,"

        dataset+=str(Y[k])+"\n"

f = open("features.txt", "w")

f.write(dataset)

f.close()

train = pd.read_csv("features.txt")

cols = train.shape[1]

features = cols - 2

label = cols - 1

label_count = label

X = train.values[:, 0:label]

Y = train.values[:, label]

print(Y)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state =

0)    output = ""

if name == "SVM Algorithm":

    cls = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 2)

    cls.fit(X_train, y_train)

    prediction_data = prediction(X_test, cls)

    msg = cal_accuracy(y_test, prediction_data,'SVM Accuracy')

    output = 'SVM Algorithm Output details<br/><br/>' + msg

    classifier = cls

if name == "Decision Tree":

    cls=DecisionTreeClassifier(max_depth=None,min_samples_split=2,

    random_state=0)

```

```
cls.fit(X_train, y_train)

prediction_data = prediction(X_test, cls)

msg = cal_accuracy(y_test, prediction_data, 'Decision Tree Accuracy')

output = 'Decision Tree Algorithm Output details<br/><br/>' + msg

if name == "KNearest Neighbors":

    cls = BaggingClassifier(KNeighborsClassifier(), max_samples=0.5,
                           max_features=0.5)

    cls.fit(X_train, y_train)

    prediction_data = prediction(X_test, cls)

    msg = cal_accuracy(y_test, prediction_data, 'KNearest Neighbor Accuracy')

    output = 'KNearest Neighbor Algorithm Output details<br/><br/>' + msg

if name == "Random Forest":

    cls = RandomForestClassifier(n_estimators=1, max_depth=0.9, random_state=None)

    cls.fit(X_train, y_train)

    prediction_data = prediction(X_test, cls)

    msg = cal_accuracy(y_test, prediction_data, 'Random Forest Accuracy')

    output = 'Random Forest Algorithm Output details<br/><br/>' + msg

if name == "Naive Bayes":

    cls = MultinomialNB()

    cls.fit(X_train, y_train)

    prediction_data = prediction(X_test, cls)

    msg = cal_accuracy(y_test, prediction_data, 'Naive Bayes Accuracy')

    output = 'Naive Bayes Algorithm Output details<br/><br/>' + msg

context = {'data': output}

return render(request, 'RunAlgorithms.html', context)
```

CHAPTER 10

RESULTS AND DISCUSSION

10.1 Overview

This project implements a web application that combines Django web development with machine learning techniques to detect and monitor cyberbullying messages. It involves user registration, login, data collection, model training, and post classification. Below are the interactions between various components:

- **User Interface:** This section represents the user interface where users interact with the application. Users can access various pages such as the home page (Index), Send Post, Register, and Login.
- **Database:** The database section represents the MySQL database and file system used to store data and files. Users' requests, including database queries and file storage, are shown here.
- **Admin Panel:** Admins can access the Admin page, Add Bullying Words functionality, and View Users functionality. These actions also involve requests to the database.
- **Machine Learning:** This section depicts machine learning algorithms used for classification. The algorithms are trained and predict based on data supplied by the database. Model outputs and predictions are returned to the database.
- **User Interaction:** This part illustrates user interactions, such as sending posts, logging in (creating sessions), and user registration. These interactions result in data being saved in the database.
- **Monitoring and Reporting:** Admins can access the Monitor Posts and Run Algorithms functionalities, which involve database requests. The database supplies data to these functionalities, and they may generate reports or monitor posts.

10.2 Dataset description

The dataset contains two columns: "Tweet" and "Text Label." Here's an explanation of what each column represents:

- **Tweet:** This column contains short text messages or "tweets" that are typically associated with social media platforms like Twitter. Each row in this column represents

an individual tweet posted by a user. Tweets are typically limited to a certain character count (e.g., 280 characters on Twitter).

- **Text Label:** This column contains labels or categories associated with each tweet. These labels can be used to classify or categorize the content of the tweets. For example, text labels could indicate whether a tweet is positive, negative, or neutral sentiment, or they could represent topics or themes present in the tweet, such as "sports," "politics," "entertainment," etc. Each row in this column likely corresponds to a label that is assigned to the corresponding tweet in the "Tweet" column.

This dataset is commonly used in natural language processing (NLP) tasks, sentiment analysis, text classification, and machine learning. It allows researchers and analysts to train and test algorithms to automatically categorize or analyse the content of tweets based on the provided text labels.

10.3 Database creation

This project first provides SQL statements to create a MySQL database named "cyber" and two tables within that database: "users" and "psts."

Users Table:

- username (varchar(50)): This column is likely intended to store usernames for user accounts. It has a maximum length of 50 characters.
- password (varchar(50)): This column is likely intended to store user passwords. Passwords should be securely hashed and stored, not in plain text, in a production system.
- contact_no (varchar(12)): This column appears to store user contact numbers. It can hold up to 12 characters.
- email (varchar(50)): This column is for storing user email addresses.
- address (varchar(50)): This column seems to be for storing user addresses.
- status (varchar(30)): This column appears to be for storing user status or some additional information. It can hold up to 30 characters.

Posts Table

- sender (varchar(50)): This column is likely intended to store the sender's username or identifier.

- filename (varchar(50)): This column might store the name of a file associated with a post.
- msg (varchar(300)): This column seems to be for storing the message or content of a post. It can hold up to 300 characters.
- post time (timestamp): This column is for storing the timestamp when the post was created. It will automatically record the date and time when a post is added.
- status (varchar(50)): Similar to the "status" column in the "users" table, this column may be for additional information or status related to posts.

10.4 Results Description

Figure 10.1 represents the main landing page of the web application. It's called the "index" page, and it contains modules for different functionalities related to the identification of online child predators and cyber harassers in a social media environment. The modules include:

- Home: A link to the homepage or main dashboard.
- User: A link for users to log in or access their user-specific functionalities.
- Register: A link for new users to create an account.
- Administrator: A link for administrators to login and access admin-specific functionalities.

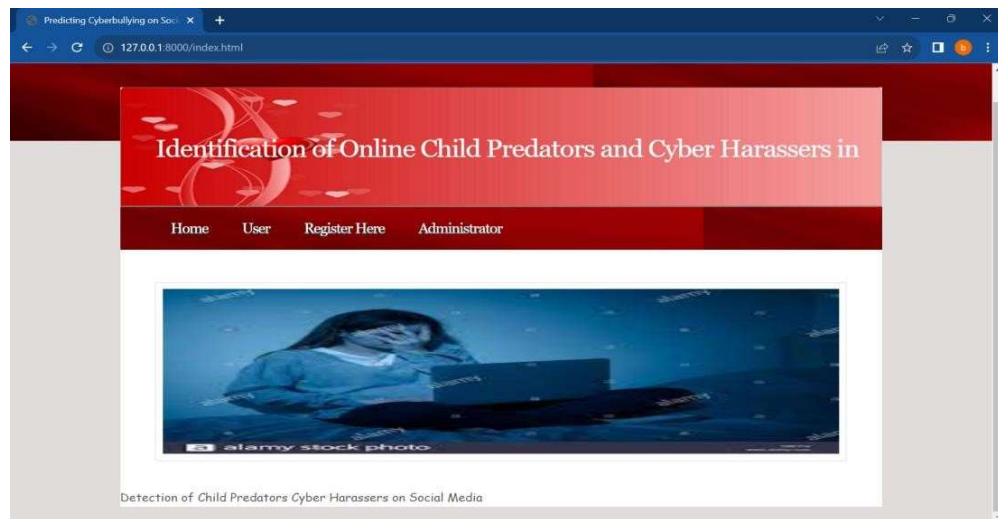
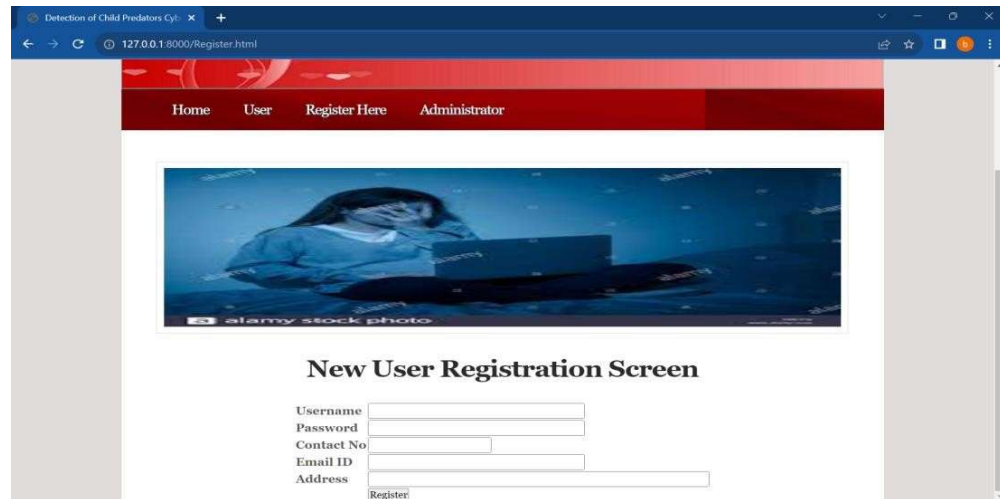


Figure 10.1: Illustration of index URL page with home, user, register, and administrator modules of identification of online child predators and cyber harassers in social media environment.

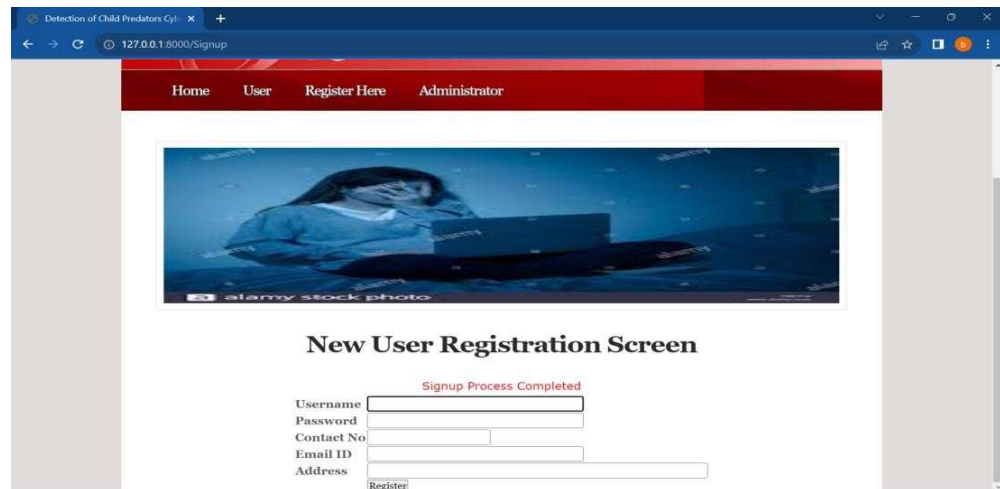
Figure 10.2 represents the registration page of the application. Users can access this page by clicking on the "Register" link from the main page. On this page, new users can sign up for an account by providing their registration details, such as username, password, contact information, email, and address.



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/Register.html". The page has a red navigation bar with links: Home, User, Register Here, and Administrator. Below the navigation bar is a banner image of a person in a blue jacket. The main heading is "New User Registration Screen". Below the heading are five input fields: Username, Password, Contact No, Email ID, and Address. A "Register" button is located at the bottom right of the form.

Figure 10.2: Register URL page for new user registration to create an account.

Figure 10.3 shows a confirmation page that appears after a user successfully registers for an account. It displays a message confirming that the registration process has been completed.



The screenshot shows the same web browser window as Figure 10.2, but the address bar now displays "127.0.0.1:8000/Signup". The page layout is identical, but a red message "Signup Process Completed" is displayed above the input fields. The "Register" button is still present at the bottom right of the form.

Figure 10.3: Signup URL after registering the new user account.

Figure10.4 represents the login page for administrators. Administrators can access this page by clicking on the "Administrator" link from the main page. They need to provide their login credentials, typically with a default username and password (username and password as 'admin') to gain access to admin-specific features and functionalities.



Figure 10.4: Admin URL page for login as an admin with a username as admin and password as admin.

After logging in as an administrator, Figure 10.5 displays various modules and functionalities available to administrators.



Figure 10.5: Admin login page showing the modules like view users, monitor post, add bullying messages/words, and run machine learning algorithms.

These modules include:

- View Users: This module allows administrators to view and manage registered users and their details.
- Monitor Post: Administrators can monitor posts made by users, including details like the messages, uploaded files, posting time, and the status of the bullying (e.g., bullying, or cyber harassment).
- Add Bullying Messages/Words: Administrators can add words or messages to the dataset that are considered bullying or related to cyber harassment.
- Run Machine Learning Algorithms: This module enables administrators to run machine learning algorithms, likely for the purpose of analysing and detecting cyberbullying content.

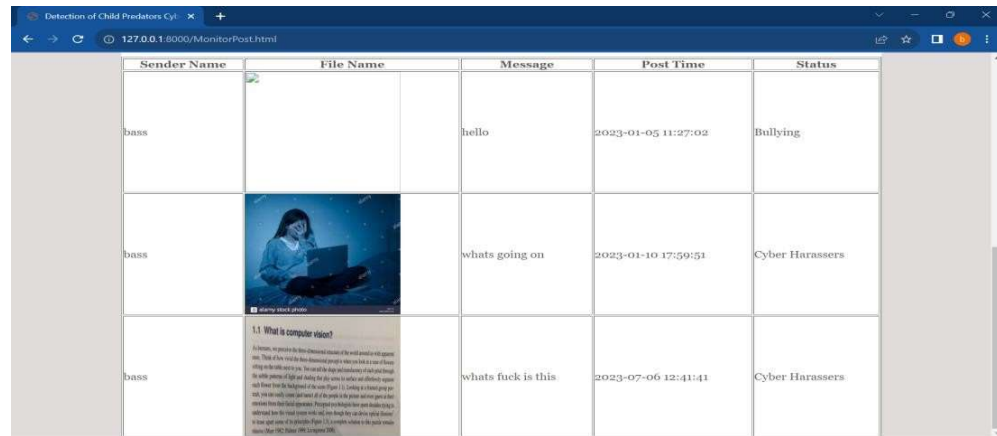
Figure 10.6 represents the "View Users" page accessible to administrators. It displays a list of registered users along with their details, which include usernames, passwords, contact information, email addresses, addresses, and user statuses.



Username	Password	Contact No	Email ID	Address	Status
bass	bass	8978944301	maturirao@gmail.com	hyd	Accepted
bass	bass	8978944301	maturirao@gmail.com	hyd	Accepted
raju	raju	8978944301	raju@gmail.com	hyderabad	Accepted

Figure 10.6: View users page showing the registered users and their details.

Figure 10.7 shows the monitor post page that provides administrators with a view of registered users' posts. It includes information such as the sender's name, uploaded files (possibly images), the messages posted, the posting time, and the status of each post (e.g., categorized as bullying or cyber harassment). F




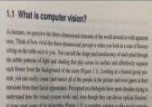
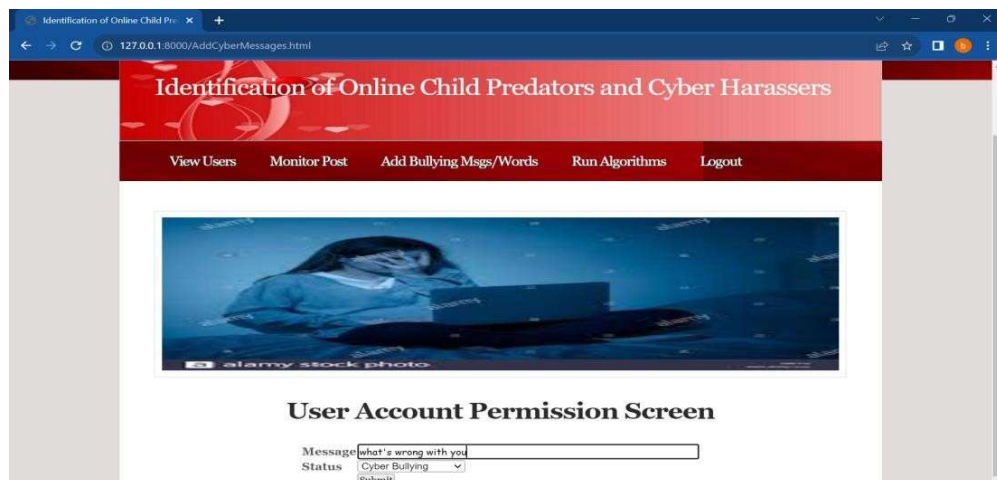
Sender Name	File Name	Message	Post Time	Status
bass		hello	2023-01-05 11:27:02	Bullying
bass		whats going on	2023-01-10 17:59:51	Cyber Harassers
bass		whats fuck is this	2023-07-06 12:41:41	Cyber Harassers

Figure 10.7: Monitor post page showing the registered users with the messages, uploaded files with the posting time and the status of the bullying i.e., bullying, or cyber harassment.

Figure10.8 represents a page or interface where administrators can add specific words or messages to a dataset. These words are typically considered bullying or related to cyber harassment. This dataset is used for training machine learning models to detect such content.



Identification of Online Child Predators and Cyber Harassers

View Users Monitor Post Add Bullying Msgs/Words Run Algorithms Logout


alamy stock photo

User Account Permission Screen

Message

Status

Figure 10.8: Adding the bullying words to the dataset.

In Figure 10.9 admin must select each algorithm and click on ‘Submit’ button to train model and the accuracy will be shown for each algorithm. Admin must repeat this step whenever first time he starts the server or upon adding new bullying messages. He must run at least one algorithm to perform automatic detection of harasser’s or non-harassers.



Figure 10.9: Run machine learning algorithms page showing different ML models i.e., SVM, naïve bayes, random forest, decision tree and KNN classification.

Figure 10.10 represents the login page for regular users who want to access the system. Users need to provide their login credentials (e.g., username and password) to login and gain access to their account.



Figure 10.10: User login URL page for logging into system to send a post and view a post.

The main functionalities accessible to users from this page are:

- Sending a post: Users can compose and send a message along with a photo.
- Viewing posts: Users can access a page to view posts made by themselves or others.
- Logout: There may be an option to log out of the system, terminating the current session.

Figure 10.11 is an illustration of the user login page after successfully logging in. The page presents several modules or options to users:

- Send post: Users can click on this module to create and send a new post. This likely includes composing a message and attaching a photo.
- View post: Users can access a section where they can view posts, which could be their own posts or posts from other users.
- Logout: There may be a logout option to allow users to log out of their account when they're finished with their session.

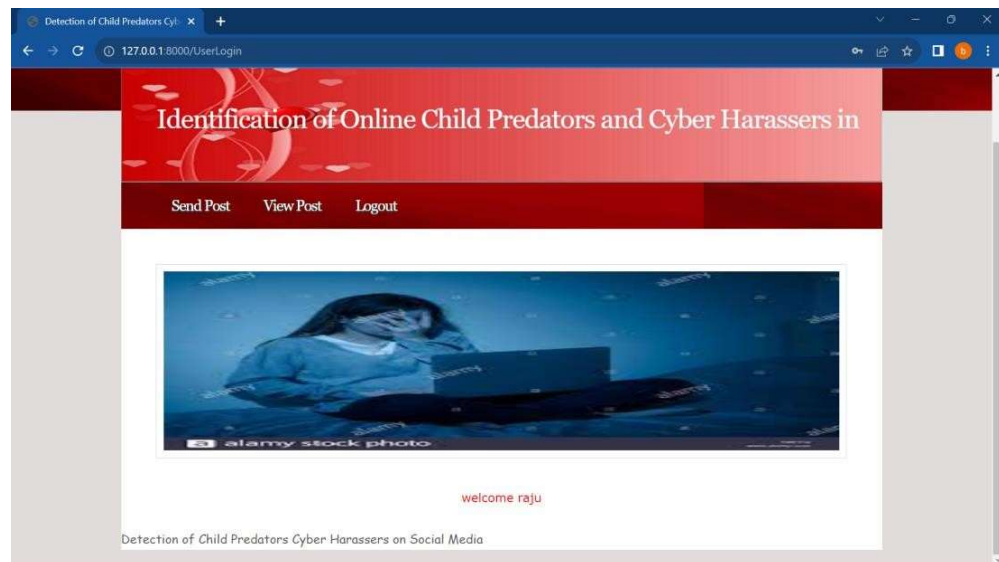


Figure 10.11: User login page showing send post, view post and logout modules.

Figure 10.12 represents the "Send post" page or module within the user interface. On this page, users can compose and send a message along with a photo. Key elements on this page include:

- Text input field: Users can type their message in this field.
- File upload: Users can upload a photo or image to include in their post.
- Send button: A button that allows users to submit their post.

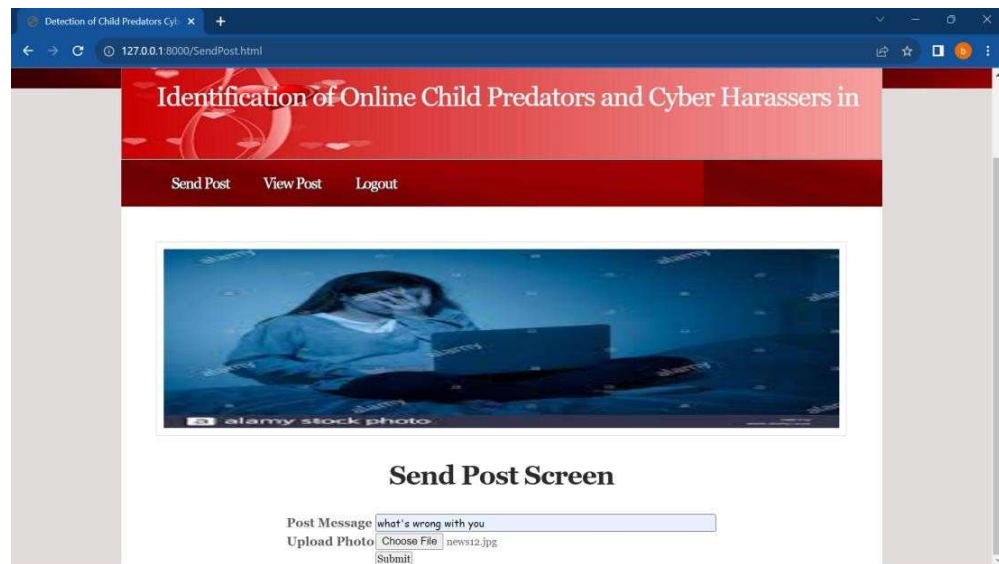


Figure 10.12: Send post page for sending a message with a photo.

Figure 10.13 shows the posts from all users, and it is observed that with the help of machine learning, the proposed system can predict the message as cyber or non-cyber harassers. Here, machine learning models are utilized to predict the harasser or non-harasser word based on the dataset records. So, admin can add all the possible harasser and non-harasser words to dataset by using 'add words' module from admin as discussed in earlier steps. After adding words then run algorithms link to train model and then proposed application predicts the harasser or non-harasser automatically.

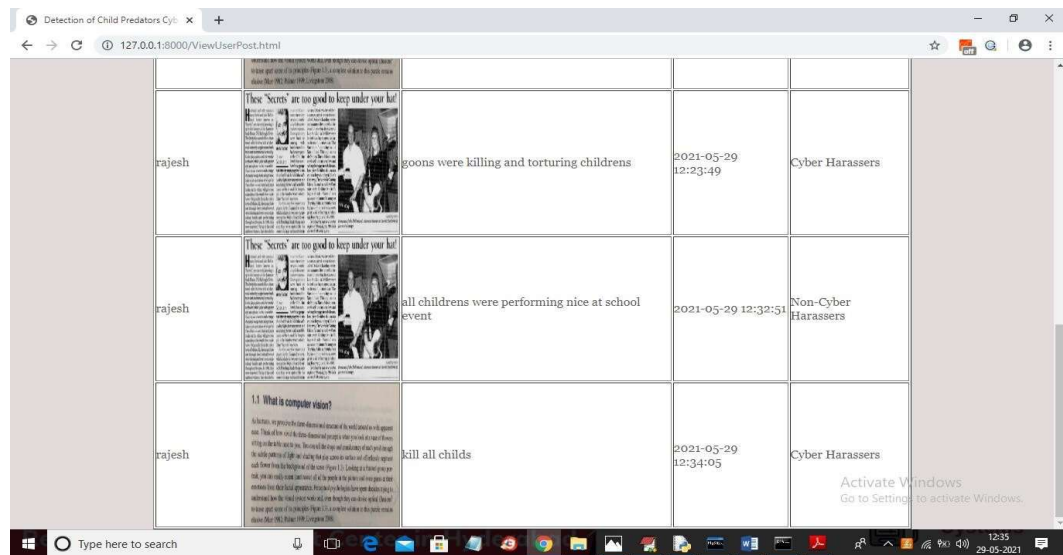


Figure 10.13: View post page for viewing all the messages with uploaded photos posted by users.

CHAPTER 11

CONCLUSION

This research presents the backend logic of a web application focused on user management and cyberbullying detection. It embodies a comprehensive approach to tackling issues related to cyberbullying and enhancing user experiences. First and foremost, the application offers robust user management capabilities. Users can register, log in, and view their profiles. User data, including usernames, passwords (which should be further secured using techniques like hashing and salting), contact information, and status, are meticulously stored within a MySQL database. This feature forms the foundation of the user experience and provides the basis for managing user interactions. One of the standout features of the application is its cyberbullying detection functionality. It leverages a diverse set of machine learning algorithms, including SVM, Decision Trees, KNN, Random Forest, and Naive Bayes, to classify user-submitted text messages or "posts" as either "Cyber Harassers" or "Non-Cyber Harassers." The application extracts relevant features from these text messages and maintains them in a dataset for training and inference. The application enables users to submit posts that encompass sender names, filenames (for attached images), messages, timestamps, and statuses. These posts are meticulously recorded within a database, forming a comprehensive record of user interactions. In terms of user interaction, the application offers a suite of user-friendly web pages. Users can register, log in, view profiles, add words associated with cyberbullying to the dataset, submit posts, run machine learning algorithms for detection, and monitor posts. This intuitive and accessible user interface facilitates smooth user engagement.

CHAPTER 12

FUTURE SCOPE

While the current application is already impressive, there is substantial room for future enhancement and expansion such as enhanced user authentication, continuously refine and optimize machine learning models for cyberbullying detection by exploring diverse algorithms, feature engineering techniques, and hyperparameter tuning.

REFERENCES

- [1] Wachs S, Wolf KD, Pan C. Cyber grooming: Risk factors, coping strategies and associations with cyberbullying. *Psicothema* 2012;24(4):628–33.
- [2] KELLER, N.B.a.M.H. video games and online chats are “hunting grounds” for sexual predators. Available from <https://www.nytimes.com/interactive/2019/12/07/us/video-games-child-sex-abuse.html> [Accessed DEC. 7, 2019].
- [3] Amer, N. Arabic-sexual-harassment-dataset. Available from <https://github.com/Nooramer8/Arabic-sexual-harassment-dataset>. [Accessed 09-10- 2023].
- [4] Nandhini BS, Sheeba J. Online social network bullying detection using intelligence techniques. *Proc Comput Sci* 2015;45:485–92. doi: <https://doi.org/10.1016/j.procs.2015.03.085>.
- [5] Al-Katheri ASA, Siraj MM. Classification of sexual harassment on Facebook using term weighting schemes. *Internat J Innov Comput* 2018;8(1):15–9. doi: <https://doi.org/10.11113/ijic.v8n1.157>.
- [6] M.Saeidi, S.Sousa, E.Milios, N.Zeh, L.Berton. Categorizing online harassment on Twitter. in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2019, 3, 283- 297. https://doi.org/10.1007/978-3-03043887-6_22.
- [7] Liu, D., C.Y. Suen, and O. Ormandjieva. A novel way of identifying cyber predators. 2017, 1712.03903,1-6. <https://doi.org/10.48550/arXiv.1712.03903>
- [8] Pandey, R., et al. Distributional semantics approach to detect intent in Twitter conversations on sexual assaults. in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. 2018, 1 270-277. <https://doi.org/10.1109/wi.2018.00-80>.
- [9] S.Karlekar, and M. Bansal.. Safecity: Understanding diverse forms of sexual harassment personal stories, *arXiv preprint arXiv*. 2018, 2,1-7. <https://doi.org/10.18653/v1/d18-1303>.
- [10] Espinoza I, Weiss F. Detection of harassment on Twitter with deep learning techniques. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* 2019;1168:307–13. doi: <https://doi.org/10.1007/978-3-03043887-6>.
- [11] Liu Y et al. Sexual harassment story classification and key information identification. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. p. 2385–8. <https://doi.org/10.1145/3357384.3358146>.

- [12] Kim, J., et al. Analysis of online conversations to detect cyber predators using recurrent neural networks. in Proceedings for the First International Workshop on Social Threats in Online Conversations: Understanding and Management. 2020,1,15-20. <https://www.aclweb.org/anthology/2020.stoc-1.3>.
- [13] Hamzah NA, Dhannoon BN. The detection of sexual harassment and chat predators using artificial neural network. *Karbala Int J Mod Sci* 2021;7 (4):6–20.
- [14] M. A. Fauzi and P. Bours, "Ensemble Method for Sexual Predators Identification in Online Chats," 2020 8th International Workshop on Biometrics and Forensics (IWBF), Porto, Portugal, 2020, pp. 1-6, doi: 10.1109/IWBF49977.2020.9107945
- [15] T. R. Ringenberg, K. Misra and J. T. Rayz, "Not So Cute but Fuzzy: Estimating Risk of Sexual Predation in Online Conversations," 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 2019, pp. 2946-2951, doi: 10.1109/SMC.2019.8914528.
- [16] P. R. Borj and P. Bours, "Predatory Conversation Detection," 2019 International Conference on Cyber Security for Emerging Technologies (CSET), Doha, Qatar, 2019, pp. 1-6, doi: 10.1109/CSET.2019.8904885.
- [17] P. Bours and H. Kulrud, "Detection of Cyber Grooming in Online Conversation," 2019 IEEE International Workshop on Information Forensics and Security (WIFS), Delft, Netherlands, 2019, pp. 1-6, doi: 10.1109/WIFS47025.2019.9035090.
- [18] N. Pendar, "Toward Spotting the Pedophile Telling victim from predator in text chats," International Conference on Semantic Computing (ICSC 2007), Irvine, CA, USA, 2007, pp. 235-241, doi: 10.1109/ICSC.2007.32.
- [19] D. Michalopoulos and I. Mavridis, "Utilizing document classification for grooming attack recognition," 2011 IEEE Symposium on Computers and Communications (ISCC), Kerkyra, Greece, 2011, pp. 864-869, doi: 10.1109/ISCC.2011.5983950.
- [20] M. C. Seto, "Pedophilia and sexual offenses against children", *Annual Review of Sex Research*, vol. 15, no. 1, pp. 321-361, 2004.
- [21]