

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**BONAFIDE CERTIFICATE**

Certified that this project report "Cyber bullying detection and classification" is the bonafide work of V.N Gopi krishna who carried out the Project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr. S. MALATHI, M.E., Ph.D.,**  **HEAD OF THE DEPARTMENT**  DEPARTMENT OF AI&DS,  PANIMALAR ENGINEERING COLLEGE,  NAZARATHPETTAI, POONAMALLEE,  CHENNAI-600 123. | **SIGNATURE**  **Dr. C. Bharanidharan , M.E., Ph.D**  **ASSISTANT** **PROFESSOR**  DEPARTMENT OF AI&DS,  PANIMALAR ENGINEERING COLLEGE,  NAZARATHPETTAI, POONAMALLEE,  CHENNAI-600 123. |

**TABLE OF CONTENTS**

**CHAPTER NO.**  **TITLE**  **PAGE NO.**

**1. Abstract 4**

**2.** **Existing System 5**

**3. Proposed System 6**

**4. LITERATURE SURVEY 7**

**5. System study 11**

**6. Workflow diagram 20 7. Usecasediagarm 21**

**8. Class diagram 21**

**9. Architecture diagram 22**

**10. Sequence diagram 23**

**11. Module description 24**

**12. Module diagarm 25**

**13. Algorithm implementation 28**

**14. RNN architecture 31**

**15. LSTMN 34**

**16. CODING 38**

**17. Deployment code 45**

**18. Output images 48**

**19. CONCLUSION 49**

**20. Future work 50**

**CYBERBULLYING TEXT CLASSIFICATION USING NLP ALGORITHM-BASED TECHNIQUES**

**ABSTRACT:**

Cyberbullying is a growing concern in the digital age, necessitating effective detection and prevention measures. This research focuses on addressing cyberbullying through text classification using Natural Language Processing (NLP) algorithm-based techniques. The study involves the compilation of a diverse dataset containing social media posts, messages, and comments, which are labelled as either cyberbullying or non-cyberbullying content. The text data is pre-processed to handle noise, remove stop words, and tokenize the text for NLP analysis. Experimental results demonstrate the efficacy of NLP algorithm-based techniques in cyberbullying text classification, outperforming traditional rule-based methods. The system's ability to identify harmful content aids in early detection and intervention, promoting safer online environments. Experimental results demonstrate the efficacy of NLP algorithm-based techniques in cyberbullying text classification, outperforming traditional rule-based methods. The system's ability to identify harmful content aids in early detection and intervention, promoting safer online environments.

**Keywords**: Cyberbullying, Text Classification, Natural Language Processing, NLP Algorithms, Naive Bayes, Support Vector Machines, Recurrent Neural Networks, Social Media, Online Safety.

**EXISTING SYSTEM:**

Cyber ranges require networked applications to test cyberspace events effectively. As testing becomes more advanced, it involves multiple real-world applications with flexible execution orders. However, it is increasingly challenging to orchestrate large-scale, chained, and heterogeneous Internet applications. State-of-the-art orchestration techniques face scalability issues due to inefficient representation models and entangled scheduling of events and applications. To address these issues, we present Wukong, a disentangled orchestration system in cyber ranges that disaggregates the scheduling and execution of workflows and their applications in a decentralized coordination approach. First, we overcome the heterogeneity of events with a workflow model that encodes event chains with compositional Directed Acyclic Graphs (DAGs) and unified event triggers. Second, Wukong disaggregates the execution of DAGs and applications with push-pull decentralized coordination over distributed agents. Our evaluation of Wukong on a real-world cyber range demonstrates its expressive, scalable, and efficient abilities for automatically emulating diverse event chains. The storage footprint of compositional modeling is up to 57 times smaller than that of baseline models. Wukong’s response delay is 1.52 to 2.74 times shorter than state-of-theart orchestration engines, and the scheduling delay is up to 2.16 times smaller than the baseline approach.

**DEMERITS:**

* They focused on only cyber ranges.
* They did not apply machine learning techniques.
* Limited scalability.
* High time complexity.
* Complex process to do that.

**PROPOSED SYSTEM:**

The proposed system aims to detect cyberbullying in text using NLP algorithm-based techniques. A diverse dataset of social media posts, messages, and comments is collected and pre-processed. NLP algorithms like Naive Bayes, SVM, and RNN are evaluated for cyberbullying classification. Algorithms learn patterns and linguistic cues indicative of cyberbullying behaviour. The dataset is split into training and testing sets; model hyperparameters are tuned for optimization. System performance is assessed using metrics like accuracy, precision, recall, F1-score, and ROC-AUC. Real-time integration of the system allows prompt cyberbullying detection and reporting. Early identification aids in intervention and creating safer online environments. The proposed system outperforms traditional rule-based methods for cyberbullying detection. Integration of NLP algorithm-based techniques enhances communication safety and user protection.

**MERITS:**

* We build a framework based application for deployment purpose.
* High scalability.
* We build a neural network based language model.
* Higher scope.

**LITERATURE REVIEW:**

**Title**: Cyberbullying Detection through Sentiment Analysis

**Author**: Jalal Omer Atoum

**Year**: 2020

In recent years with the widespread of social media platforms across the globe especially among young people, cyberbullying and aggression have become a serious and annoying problem that communities must deal with. Such platforms provide various ways for bullies to attack and threaten others in their communities. Various techniques and methodologies have been used or proposed to combat cyberbullying through early detection and alerts to discover and/or protect victims from such attacks. Machine learning (ML) techniques have been widely used to detect some language patterns that are exploited by bullies to attack their victims. Also. Sentiment Analysis (SA) of social media content has become one of the growing areas of research in machine learning. SA provides the ability to detect cyberbullying in real-time. SA provides the ability to detect cyberbullying in real-time. This paper proposes a SA model for identifying cyberbullying texts in Twitter social media. Support Vector Machines (SVM) and Naïve Bayes (NB) are used in this model as supervised machine learning classification tools. The results of the experiments conducted on this model showed encouraging outcomes when a higher n-grams language model is applied on such texts in comparison with similar previous research. Also, the results showed that SVM classifiers have better performance measures than NB classifiers on such tweets.

**Title**: A Drift-Sensitive Distributed LSTM Method for Short Text Stream Classification

**Author**: Peipei Li, Yingying Liu, Yang Hu, Yuhong Zhang, Xuegang Hu, and Kui Yu

**Year**: 2023

Real-world applications especially in the fields of social media have produced massive short text streams. Unlike traditional normal texts, these data present the characteristics of short length, high-volume, high-velocity and variable data distribution etc, which lead to the issues of data sparsity and concept drift. It is hence very challenging for existing short text classification algorithms. Therefore, we propose a flexible Long Short-Term Memory (LSTM) ensemble network based short text stream classification approach, which is implemented in a distributed mode while maintaining the high-accuracy advantage of deep learning models. More specifically, external resource based short text embedding using a pretrained embedding model and CNN is first proposed for the solution to the data sparsity of short texts. Second, to adapt to the high-volume and high-velocity short text streams, a flexible LSTM network is developed and implemented in a distributed mode for classifying short text data streams. Third, a concept drift factor is introduced for adapting to the concept drifts caused by the changing of data distributions. Finally, experiments conducted on three real short text data sets demonstrate that as compared with several state-of-the-art short text (stream) classification approaches, the proposed approach can classify short text streams effectively and efficiently while adapting to concept drifts.

**Title**: Cyberbullying Detection Using Machine Learning

**Author**: Nideeksha B K, P Shreya, Sudharani Reddy P, Mohamadi Ghousiya Kousar

**Year**: 2021

The advent of the digital age has enabled people to a new form of bullying which often results in social stigma. This new form of bullying is Cyberbullying which is a crime in which a perpetrator targets a person with online harassment and hate. Social networks provide a rich environment for bullies to find and harass vulnerable victims. Messages or comments concerning sensitive topics that are personal to an individual are more likely to be internalized by a victim, often ending in tragic outcomes. This phenomenon is creating a demand for automated, data-driven techniques for analyzing and detecting such behaviour on the internet. In this paper, a machine learning-based approach is proposed to detect cyberbullying activities from social network data. Naïve Bayes classifier is used to classify the type of message i.e., cyberbullying and non-cyberbullying message. Finally, a chatbot can be implemented to warn bullies about the consequences of their cyberbullying messages and take necessary actions. Our evaluation of performance results reveals that the accuracy of the proposed approach increases with more classification data

**Title**: A Systematic Review on the Visualization of Avatars and Agents in AR & VR displayed using Head-Mounted Displays

**Author**: Florian Weidner, Gerd Boettcher, Stephanie Arevalo Arboleda, Chenyao Diao, Luljeta Sinani, Christian Kunert, Christoph Gerhardt, Wolfgang Broll, Alexander Raake

**Year**: 2023

Augmented Reality (AR) and Virtual Reality (VR) are pushing from the labs towards consumers, especially with social applications. These applications require visual representations of humans and intelligent entities. However, displaying and animating photo-realistic models comes with a high technical cost while low-fidelity representations may evoke eeriness and overall could degrade an experience. Thus, it is important to carefully select what kind of avatar to display. This article investigates the effects of rendering style and visible body parts in AR and VR by adopting a systematic literature review. We analyzed 72 papers that compare various avatar representations. Our analysis includes an outline of the research published between 2015 and 2022 on the topic of avatars and agents in AR and VR displayed using head-mounted displays, covering aspects like visible body parts (e.g., hands only, hands and head, full-body) and rendering style (e.g., abstract, cartoon, realistic); an overview of collected objective and subjective measures ( e.g., task performance, presence, user experience, body ownership); and a classification of tasks where avatars and agents were used into task domains (physical activity, hand interaction, communication, gamelike scenarios, and education/training). We discuss and synthesize our results within the context of today’s AR and VR ecosystem, provide guidelines for practitioners, and finally identify and present promising research opportunities to encourage future research of avatars and agents in AR/VR environments. Index Terms—Virtual reality, augmented reality, avatars, visualization.

**Title**: Cyber Bullying Detection on Social Media using Machine Learning

**Author**: Aditya Desai , Shashank Kalaskar , Omkar Kumbhar , and Rashmi Dhumal

**Year**: 2021

Usage of internet and social media backgrounds tends in the use of sending, receiving and posting of negative, harmful, false or mean content about another individual which thus means Cyberbullying. Bullying over social media also works the same as threatening, calumny, and chastising the individual. Cyberbullying has led to a severe increase in mental health problems, especially among the young generation. It has resulted in lower self-esteem, increased suicidal ideation. Unless some measure against cyberbullying is taken, self-esteem and mental health issues will affect an entire generation of young adults. Many of the traditional machine learning models have been implemented in the past for the automatic detection of cyberbullying on social media. But these models have not considered all the necessary features that can be used to identify.

**SYSTEM STUDY:**

**OVERVIIEW OF THE SYSTEM:**

Cyberbullying is a significant problem in today's world, becoming more extensive and challenging to identify. Detecting unwanted messages in its early phases is a major challenge, exacerbated by the shortage of labelled data for training detection models. The proposed cyberbullying detection framework aims to address these challenges, emphasizing problem definition, data preparation, algorithm evaluation, result improvement, and result prediction.

**FEASIBILITY STUDY:**

**Data Wrangling:**

The report emphasizes loading, checking, trimming, and cleaning the dataset for analysis. Careful steps are taken in data cleaning, ensuring transparency and justification for the cleaning decisions.

**Data collection:**

The collected dataset for predicting cyberbullying data is split into Training and Test sets, with a standard 7:3 ratio. The cyberbullying data has a classification like religion, age, gender and ethinicity based bullying records.

**Pre-processing:**

Addressing missing values and inconsistencies in the collected data through pre-processing. The report stresses the importance of data pre-processing to enhance algorithm efficiency, emphasizing the removal of outliers and variable conversion.

**Building a classification Modes:**

For predicting cyber bullying classifications, the report advocates for a high-accuracy prediction model due to its effectiveness in classification problems. The model's strengths include robust pre-processing, handling outliers, irrelevant variables, and a mix of continuous, categorical, and discrete variables. Additionally, it produces an out-of-bag estimate error, proven to be unbiased in tests, and is relatively easy to tune.

**Construction of a Predictive Model**

## Machine learning needs data gathering have lot of past data’s. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can’t be used directly. It’s used to pre-process then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.

Data Pre-Processing

Data Gathering

Choose model

Train model

Test model

Tune model

Prediction

Process of dataflow diagram

**PROJECT REQUIREMENTS:**

**General:**

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements

2. Non-Functional requirements

3. Environment requirements

A. Hardware requirements

B. software requirements

**Functional requirements:**

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

**Non-Functional Requirements:**

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

**ENVIRONMENTAL REQUIREMENTS:**

1. Software Requirements:

Operating System : Windows 10 or later

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Intel i3

Hard disk : minimum 80 GB

RAM : minimum 2 GB

**LIST OF MODULES:**

* Data Pre-processing
* Data Analysis of Visualization
* Long Short-term memory networks
* Simple RNN
* Deployment

**SOFTWARE DESCRIPTION**

Anaconda is a free and open-source distribution of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)) programming languages for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) ([data science](https://en.wikipedia.org/wiki/Data_science), [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications, large-scale data processing, [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics), etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and deployment. Package versions are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager) “Conda”.

**ANACONDA NAVIGATOR**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

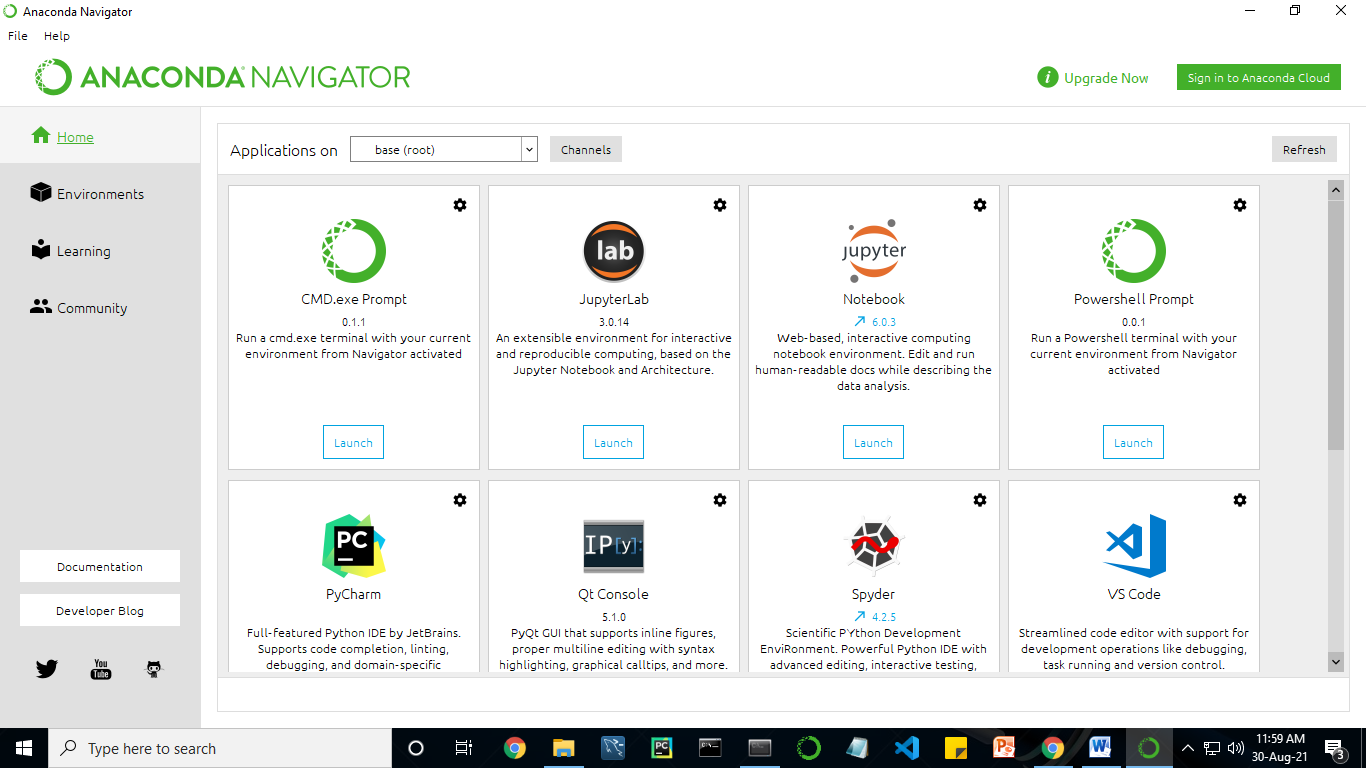
Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

The following applications are available by default in Navigator:

* [JupyterLab](https://jupyterlab.readthedocs.io/en/stable/)
* [Jupyter Notebook](https://jupyter.readthedocs.io/en/latest/)
* [Spyder](https://www.spyder-ide.org/)
* [PyCharm](https://www.jetbrains.com/pycharm/documentation/)
* [VSCode](https://code.visualstudio.com/docs)
* [Glueviz](http://glueviz.org/en/stable/)
* [Orange 3 App](http://orange.biolab.si/docs/)
* [RStudio](http://docs.rstudio.com/)
* Anaconda Prompt (Windows only)
* Anaconda PowerShell (Windows only)



Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with conda list on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

**JUPYTER NOTEBOOK**

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app), which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

## Installation: The easiest way to install the Jupyter Notebook App is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

**JUPYTER Notebook App:**

The Jupyter Notebook App is a server-client application that allows editing and running [notebook documents](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document) via a web browser.

The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” ([Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard)), a “control panel” showing local files and allowing to open notebook documents or shutting down their [kernels](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel).

Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down

[**Notebook Dashboard**](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#id8)**:** The Notebook Dashboard is the component which is shown first when you launch [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app). The Notebook Dashboard is mainly used to open [notebook documents](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-document), and to manage the running [kernels](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel) (visualize and shutdown).

The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files

**Working Process:**

* Download and install anaconda and get the most useful package for machine learning in Python.
* Load a dataset and understand its structure using statistical summaries and data visualization.

The best way to get started using Python for machine learning is to complete a project.

* It will force you to install and start the Python interpreter (at the very least).
* It will give you a bird’s eye view of how to step through a small project.
* It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

* Define Problem.
* Prepare Data.
* Evaluate Algorithms.
* Improve Results.
* Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

**Work flow diagram:**

Source Data

Data Processing and Cleaning

Training Dataset

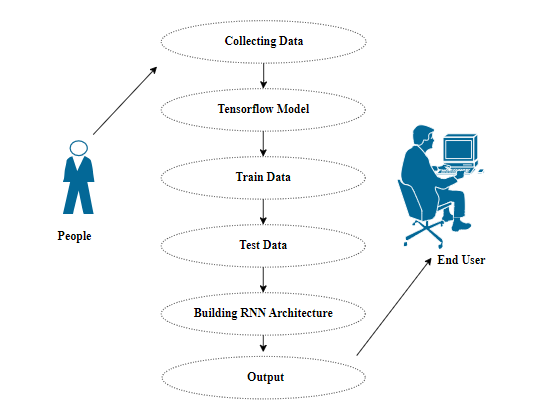
Testing Dataset

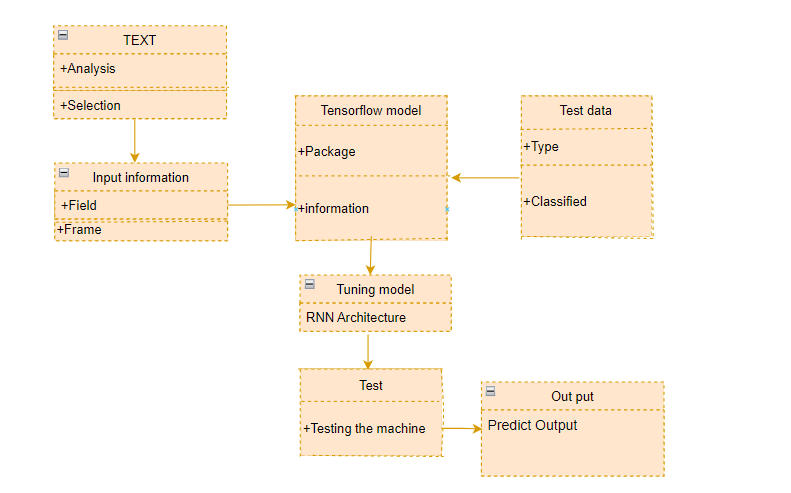
Best Model by Accuracy

Classification Algorithms

Cyber Bullying detection

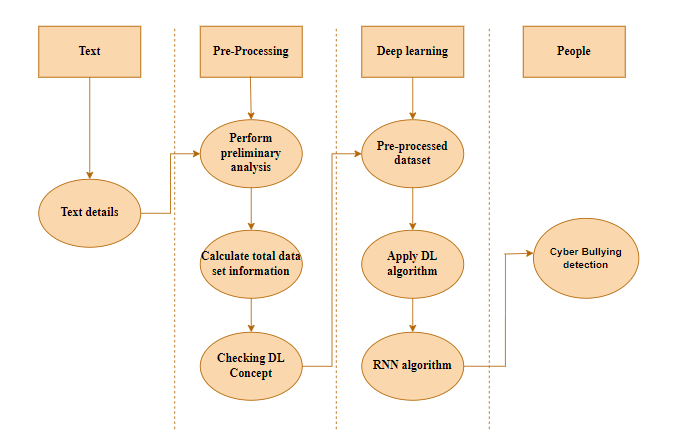
**Use Case Diagram:**

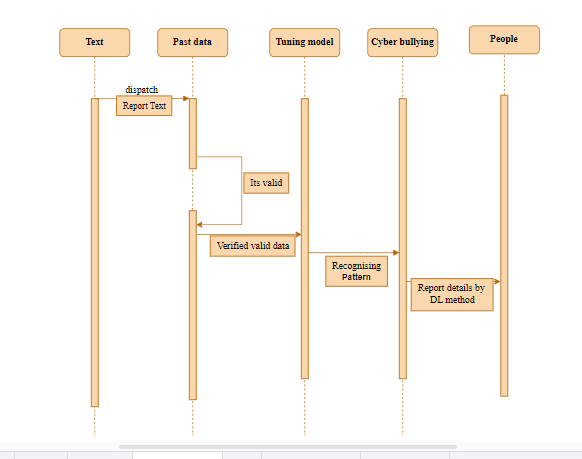


**ClASS Diagram:** 

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system.

**Activity Diagram:**



**Sequence diagra****m**

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes.

**MODULE DESCRIPTION:**

**Data Pre-processing:**

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

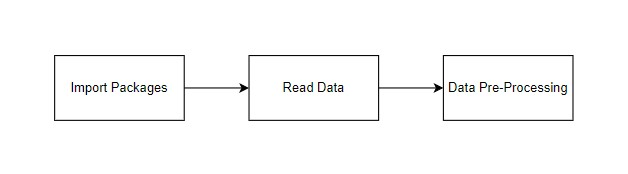
A number of different **data cleaning** tasks using Python’s [Pandas library](https://pandas.pydata.org/) and specifically, it focus on probably the biggest data cleaning task, **missing values** and it able to **more**[**quickly clean data**](https://www.dataoptimal.com/data-cleaning-with-python-2018/). It wants to **spend less time cleaning data**, and more time exploring and modeling.

* User forgot to fill in a field.
* Data was lost while transferring manually from a legacy database.
* There was a programming error.
* Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

* import libraries for access and functional purpose and read the given dataset
* General Properties of Analyzing the given dataset
* Display the given dataset in the form of data frame
* show columns
* shape of the data frame
* To describe the data frame
* Checking data type and information about dataset
* Checking for duplicate data
* Checking Missing values of data frame
* Checking unique values of data frame
* Checking count values of data frame
* Rename and drop the given data frame
* To specify the type of values
* To create extra columns

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

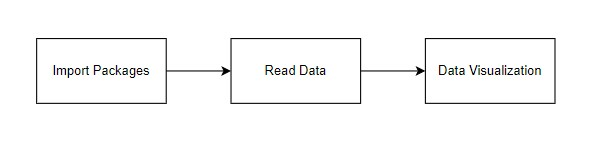
output : removing noisy data

**Data visualization:**

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

* How to chart time series data with line plots and categorical quantities with bar charts.
* How to summarize data distributions with histograms and box plots.

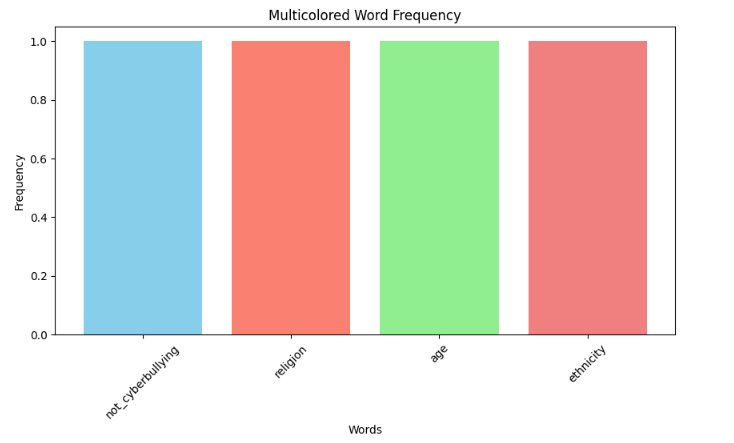
MODULE DIAGRAM

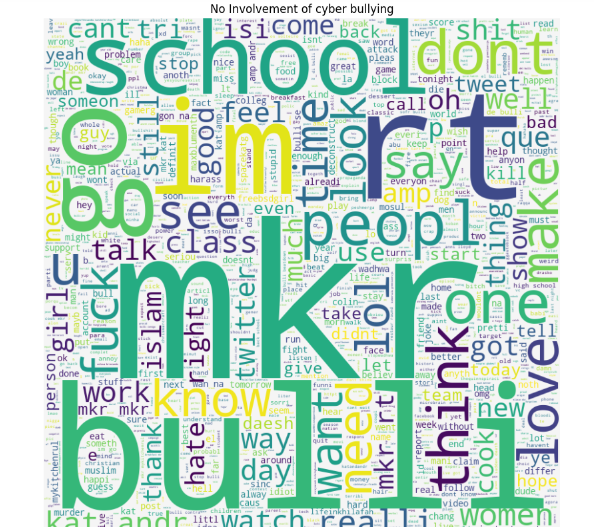


GIVEN INPUT EXPECTED OUTPUT

Input: data

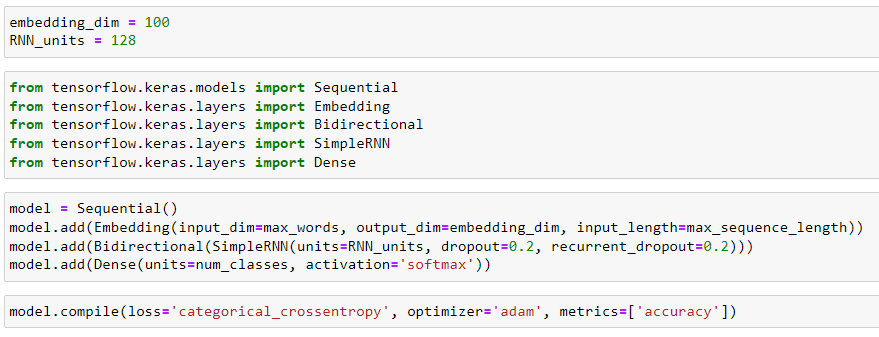
Output: visualized data





**Algorithm implementation:**

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data.



**Performance Metrics to calculate:**

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

True Positive Rate (TPR) = TP / (TP + FN)

False Positive Rate (FPR) = FP / (FP + TN)

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

**Accuracy calculation:**

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct.

Precision = TP / (TP + FP)

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labelled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

Recall = TP / (TP + FN)

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

General Formula:

F- Measure = 2TP / (2TP + FP + FN)

F1-Score Formula:

F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

The below 2 different algorithms are compared:

* RNN Architecture
* Long-Short term memory networks

**RNN ARCHITECTURE:**

A Recurrent Neural Network (RNN) is a type of artificial neural network designed for processing sequential data. Unlike traditional feedforward neural networks, RNNs have connections that loop back on themselves, allowing them to maintain a hidden state that captures information from previous time steps in the sequence. This looping mechanism makes RNNs well-suited for tasks involving sequences, such as natural language processing, speech recognition, and time series prediction.

Here is a detailed explanation of the architecture and key components of an RNN:

**Basic Structure:**

An RNN consists of a series of interconnected layers. At each time step t, it takes an input vector (or sequence) and produces an output vector (or sequence).

The key feature of an RNN is its hidden state, denoted as "h." This hidden state is a representation of the network's memory, and it is updated at each time step.

**Input and Output:**

At each time step t, the RNN takes an input vector or element x(t). This input can be a single element of a sequence, a word in a sentence, a pixel in an image, etc.

The RNN produces an output vector or element y(t) at each time step. The output can be used for various tasks, such as predicting the next element in a sequence or classifying the sequence as a whole.

**Hidden State:**

The hidden state h(t) is a vector that captures information from previous time steps. It serves as the memory of the network.

The hidden state is computed at each time step using the current input x(t) and the previous hidden state h(t-1).

**Output Computation:**

The output at each time step can be computed based on the current hidden state or a combination of the hidden state and the input at that time step.

**Backpropagation Through Time (BPTT):**

Training an RNN involves using a variant of backpropagation called Backpropagation through Time.

It is similar to standard backpropagation but accounts for the sequential nature of the data.

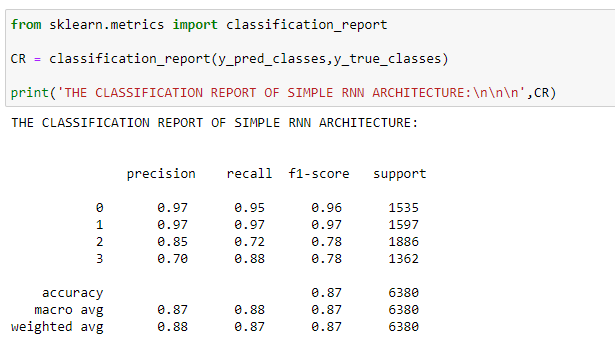
The gradients are computed at each time step and accumulated over the entire sequence to update the network's weights.

**Issues with Standard RNNs:**

Standard RNNs have limitations, including the vanishing gradient problem, which makes it challenging to capture long-range dependencies in sequences.

To address these issues, more advanced RNN architectures have been developed, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which are designed to better capture long-term dependencies.

In summary, an RNN is a neural network architecture that can process sequential data by maintaining a hidden state that captures information from previous time steps. It is a fundamental building block for various sequence-based tasks in machine learning and deep learning.



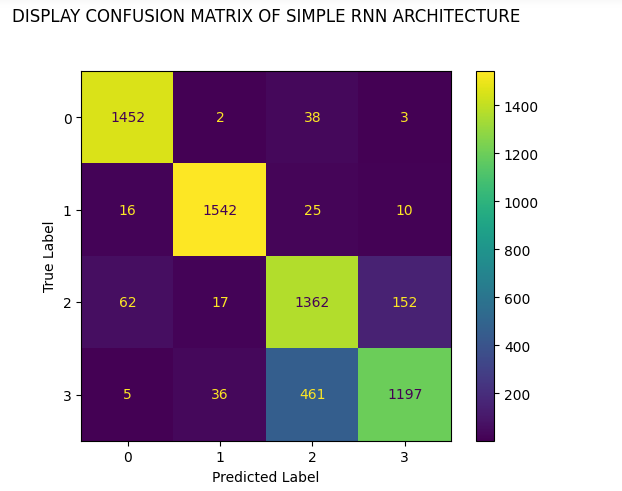
MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

Input: data

Output: getting accuracy



**Long-Short term memory networks:**

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem and better capture long-range dependencies in sequential data. They were introduced to overcome the limitations of standard RNNs and have become a fundamental building block in many applications involving sequences, such as natural language processing, speech recognition, and time series analysis. Here's a comprehensive explanation of LSTM networks:

**Basic Structure:**

An LSTM network is composed of LSTM cells arranged in a sequence. Each LSTM cell has an internal structure that enables it to store and retrieve information over long sequences.

Like standard RNNs, LSTM networks take input vectors or elements sequentially and produce output vectors or elements at each time step.

The key innovation in LSTM cells is their ability to maintain a cell state, which can capture long-term dependencies in the data.

**Components of an LSTM Cell:**

An LSTM cell consists of three main gates and a cell state:

Forget Gate: Decides what information from the cell state should be thrown away or kept.

Input Gate: Determines what new information should be added to the cell state.

Output Gate: Controls what information from the cell state should be used to generate the output.

Cell State: The cell state runs throughout the entire sequence and can carry information over long distances.

**Information Flow:**

The forget gate (f(t)) controls what information from the previous cell state (C(t-1)) should be retained.

The input gate (i(t)) determines what new information from the candidate cell state (ĉ(t)) should be added to the cell state.

The cell state (C(t)) is updated based on the forget gate, input gate, and candidate cell state.

The output gate (o(t)) controls what information from the cell state should be used to produce the hidden state (h(t)).

**Backpropagation Through Time (BPTT):**

LSTM networks are trained using Backpropagation Through Time, similar to standard RNNs. BPTT computes gradients for the network's parameters to minimize a loss function.

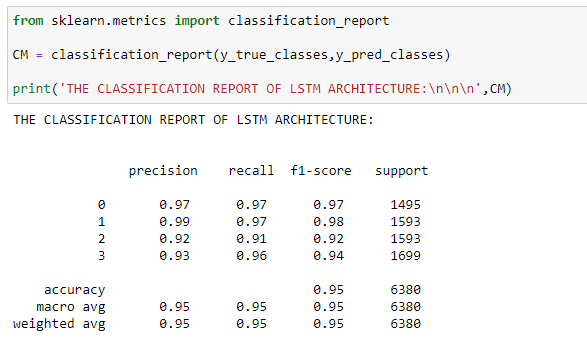
**Advantages of LSTMs:**

LSTMs can capture long-range dependencies in sequences.

They mitigate the vanishing gradient problem, allowing for more effective training on long sequences.

They are suitable for a wide range of sequence-based tasks and have been extended into more advanced variants like Gated Recurrent Units (GRUs).

In summary, LSTM networks are a type of recurrent neural network that incorporates memory cells with gates to selectively store, update, and retrieve information over long sequences. This architecture has proven effective in capturing complex patterns in sequential data and has become a cornerstone of deep learning in fields that involve sequences.



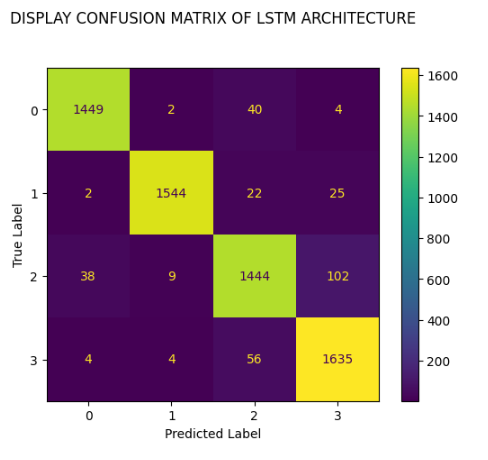
MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy



**CODING:**

# DATA PREPROCESSING AND DATA CLEANING:

**import** pandas **as** pd

**import** numpy **as** np

Data **=** pd**.**read\_csv('CYBER.csv', encoding**=**'latin-1')

Data**.**head()

Data**.**tail()

Data**.**shape

Data **=** Data**.**dropna()

Data**.**shape

Data**.**size

Data**.**isnull()**.**sum()

Data**.**info()

Data**.**columns

Data['cyberbullying\_type']**.**unique()

Data['cyberbullying\_type']**.**value\_counts()

Data**.**groupby('cyberbullying\_type')**.**describe()

#### BEFORE LABEL ENCODER

Data**.**head()

**from** sklearn.preprocessing **import** LabelEncoder

var\_mod **=** ['tweet\_text','cyberbullying\_type']

le **=** LabelEncoder()

**for** i **in** var\_mod:

Data[i] **=** le**.**fit\_transform(Data[i])**.**astype(int)

#### AFTER LABEL ENCODER

Data**.**head()

Data**.**duplicated()

Data**.**duplicated()**.**sum()

Data **=** Data**.**drop\_duplicates()

Data**.**duplicated()**.**sum()

# DATA VISUALIZATION AND DATA ANALYSIS

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

Data **=** pd**.**read\_csv('CYBER.csv', encoding**=**'latin-1')

Data**.**head()

Data**.**tail()

Data **=** Data**.**dropna()

Data['cyberbullying\_type']**.**unique()

Data['cyberbullying\_type'] **=** Data['cyberbullying\_type']**.**map({'not\_cyberbullying': 1, 'religion':2,'age':3,'ethnicity':4})

**import** matplotlib.pyplot **as** plt

**from** collections **import** Counter

*# Example data:*

your\_data **=** ['not\_cyberbullying', 'religion', 'age', 'ethnicity']

*# Join the text data into a single string*

text\_data **=** ' '**.**join(your\_data)

*# Split the text into individual words*

words **=** text\_data**.**split()

*# Count the frequency of each word*

word\_counts **=** Counter(words)

*# Generate a list of unique colors for each bar*

colors **=** ['skyblue', 'salmon', 'lightgreen', 'lightcoral']

*# Plot the multicolored bar chart*

plt**.**figure(figsize**=**(10, 5))

plt**.**bar(word\_counts**.**keys(), word\_counts**.**values(), color**=**colors)

plt**.**xlabel('Words')

plt**.**ylabel('Frequency')

plt**.**title('Multicolored Word Frequency')

plt**.**xticks(rotation**=**45)

plt**.**show()

**import** matplotlib.pyplot **as** plt

**from** collections **import** Counter

*# Example data:*

your\_data **=** ['not\_cyberbullying', 'religion', 'age', 'ethnicity']

*# Join the text data into a single string*

text\_data **=** ' '**.**join(your\_data)

*# Split the text into individual words*

words **=** text\_data**.**split()

*# Count the frequency of each word*

word\_counts **=** Counter(words)

*# Get a list of unique colors for each word*

unique\_colors **=** ['skyblue', 'lightgreen', 'lightcoral', 'orange']

*# Plot the horizontal bar chart with custom colors*

plt**.**figure(figsize**=**(10, 6))

bars **=** plt**.**barh(list(word\_counts**.**keys()), list(word\_counts**.**values()), color**=**unique\_colors)

plt**.**xlabel('Frequency')

plt**.**ylabel('Words')

plt**.**title('Word Frequency')

*# Adding a legend*

plt**.**legend(bars, your\_data, loc**=**'upper right')

plt**.**show()

X,X\_test,y,y\_test **=** train\_test\_split(Data**.**loc[:,'tweet\_text':],Data['cyberbullying\_type'],test\_size**=**0.2)

*# !pip install wordcloud*

**from** wordcloud **import** WordCloud

**import** matplotlib.pyplot **as** plt

religion **=** ' '**.**join(Data**.**loc[Data['cyberbullying\_type'] **==** 1, 'tweet\_text']**.**values)

religion\_text **=** WordCloud(background\_color**=**'whitesmoke',max\_words**=**2000,width **=** 800, height **=** 800)**.**generate(religion)

plt**.**figure(figsize**=**[10,30])

plt**.**imshow(religion\_text,interpolation**=**'bilinear')

plt**.**title('No Involvement of cyber bullying')

plt**.**axis('off')

age **=** ' '**.**join(Data**.**loc[Data['cyberbullying\_type'] **==** 2, 'tweet\_text']**.**values)

age\_text **=** WordCloud(background\_color**=**'thistle',max\_words**=**2000,width **=** 800, height **=** 800)**.**generate(age)

plt**.**figure(figsize**=**[10,30])

### SIMPLE RNN ARCHITECTURE

**import** pandas **as** pd

**import** numpy **as** np

Data **=** pd**.**read\_csv('CYBER.csv', encoding**=**"latin-1")

Data**.**head()

Data**.**tail()

Data['cyberbullying\_type']**.**unique()

Data**.**drop(Data**.**index[Data['cyberbullying\_type'] **==** 'other\_cyberbullying'], inplace**=True**)

Data**.**drop(Data**.**index[Data['cyberbullying\_type'] **==** 'gender'], inplace**=True**)

Data['cyberbullying\_type']**.**value\_counts()

Data['tweet\_text'] **=** Data['tweet\_text']**.**apply(**lambda** x: x**.**lower() **if** pd**.**notna(x) **else** "")

**from** sklearn.preprocessing **import** LabelEncoder

label\_encoder **=** LabelEncoder()

Data['cyberbullying\_type'] **=** label\_encoder**.**fit\_transform(Data['cyberbullying\_type'])

num\_classes **=** len(label\_encoder**.**classes\_)

x **=** Data['tweet\_text']

y **=** Data['cyberbullying\_type']

**from** tensorflow.keras.utils **import** to\_categorical

y **=** to\_categorical(y, num\_classes**=**num\_classes)

**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**=**42)

max\_words **=** 10000

max\_sequence\_length **=** 100

**from** tensorflow.keras.preprocessing.text **import** Tokenizer

tokenizer **=** Tokenizer(num\_words**=**max\_words)

tokenizer**.**fit\_on\_texts(x\_train)

x\_train\_sequences **=** tokenizer**.**texts\_to\_sequences(x\_train)

x\_test\_sequences **=** tokenizer**.**texts\_to\_sequences(x\_test)

**from** tensorflow.keras.preprocessing.sequence **import** pad\_sequences

x\_train\_padded **=** pad\_sequences(x\_train\_sequences, maxlen**=**max\_sequence\_length)

x\_test\_padded **=** pad\_sequences(x\_test\_sequences, maxlen**=**max\_sequence\_length)

embedding\_dim **=** 100

RNN\_units **=** 128

**from** tensorflow.keras.models **import** Sequential

**from** tensorflow.keras.layers **import** Embedding

**from** tensorflow.keras.layers **import** Bidirectional

**from** tensorflow.keras.layers **import** SimpleRNN

**from** tensorflow.keras.layers **import** Dense

model **=** Sequential()

model**.**add(Embedding(input\_dim**=**max\_words, output\_dim**=**embedding\_dim, input\_length**=**max\_sequence\_length))

model**.**add(Bidirectional(SimpleRNN(units**=**RNN\_units, dropout**=**0.2, recurrent\_dropout**=**0.2)))

model**.**add(Dense(units**=**num\_classes, activation**=**'softmax'))

model**.**compile(loss**=**'categorical\_crossentropy', optimizer**=**'adam', metrics**=**['accuracy'])

**from** tensorflow.keras.callbacks **import** ModelCheckpoint

model\_path **=** "SIMPLERNN.h5"

M **=** ModelCheckpoint(model\_path, monitor**=**'accuracy', verbose**=**1, save\_best\_only**=True**, mode**=**'max')

epochs **=** 10

batch\_size **=** 32

model**.**fit(x\_train\_padded, y\_train, epochs**=**epochs, batch\_size**=**batch\_size, validation\_split**=**0.1, callbacks**=**[M])

y\_pred **=** model**.**predict(x\_test\_padded)

y\_pred\_classes **=** np**.**argmax(y\_pred, axis**=**1)

y\_true\_classes **=** np**.**argmax(y\_test, axis**=**1)

**from** sklearn.metrics **import** accuracy\_score

data**=**[AC]

alg**=**"LSTM ARCHITECTURE"

plt**.**figure(figsize**=**(5,5))

b**=**plt**.**bar(alg,data,color**=**("MAROON"))

plt**.**title("THE ACCURACY SCORE OF LSTM ARCHITECTURE IS\n\n\n")

plt**.**legend(b,data,fontsize**=**9)

graph()

**DEPLOYMENT CODE:**

from django.shortcuts import render, redirect

from . models import UserPersonalModel

from . forms import UserRegisterForm, UserPersonalForm

from django.contrib.auth import authenticate, login,logout

from django.contrib import messages

import numpy as np

import tensorflow

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

from keras.utils import to\_categorical

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

def Landing\_1(request):

return render(request, '1\_Landing.html')

def Register\_2(request):

form = UserRegisterForm()

if request.method =='POST':

form = UserRegisterForm(request.POST)

if form.is\_valid():

form.save()

user = form.cleaned\_data.get('username')

messages.success(request, 'Account was successfully created. ' + user)

return redirect('Login\_3')

context = {'form':form}

return render(request, '2\_Register.html', context)

def Login\_3(request):

if request.method =='POST':

username = request.POST.get('username')

password = request.POST.get('password')

user = authenticate(username=username, password=password)

if user is not None:

login(request, user)

return redirect('Home\_4')

else:

messages.info(request, 'Username OR Password incorrect')

context = {}

return render(request,'3\_Login.html', context)

def Home\_4(request):

return render(request, '4\_Home.html')

def Teamates\_5(request):

return render(request,'5\_Teamates.html')

def Domain\_Result\_6(request):

return render(request,'6\_Domain\_Result.html')

def Problem\_Statement\_7(request):

return render(request,'7\_Problem\_Statement.html')

def Per\_Info\_8(request):

if request.method == 'POST':

fieldss = ['firstname','lastname','age','address','phone','city','state','country']

form = UserPersonalForm(request.POST)

if form.is\_valid():

print('Saving data in Form')

form.save()

return render(request, '4\_Home.html', {'form':form})

else:

print('Else working')

form = UserPersonalForm(request.POST)

return render(request, '8\_Per\_Info.html', {'form':form})

model1 = tensorflow.keras.models.load\_model('C:/Users/SPIRO25/Desktop/ITPRN06 - CYBER BULLYING/DEPLOYMENT/PROJECT/APP/LSTM.h5')

models = UserPersonalModel.objects.all()

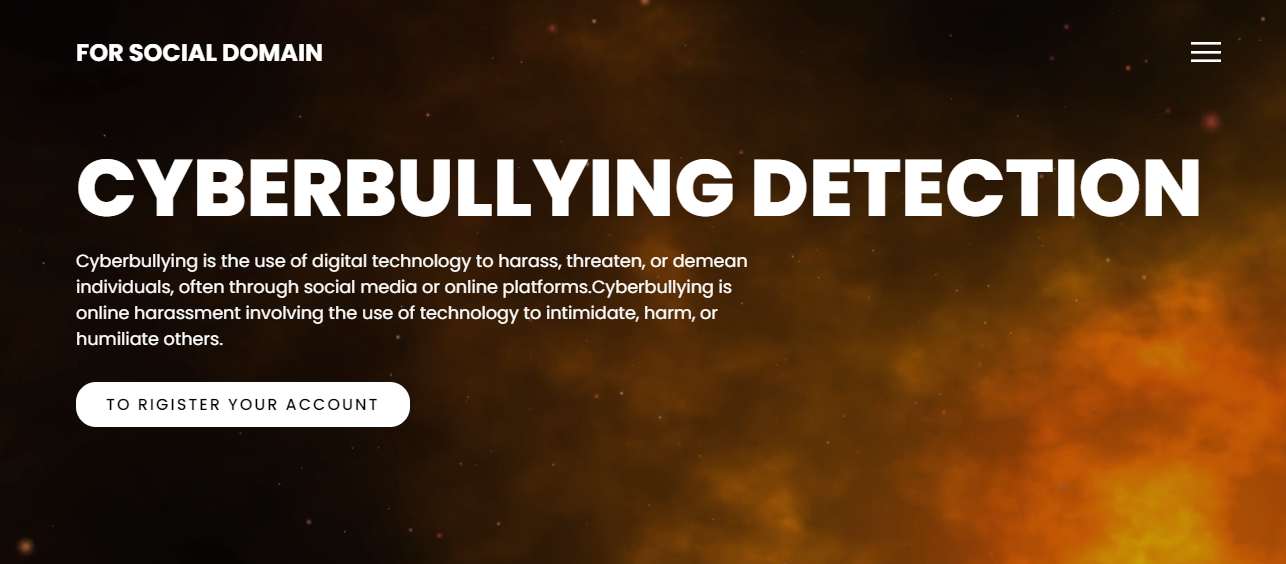
return render(request, '10\_Per\_Database.html', {'models':models})

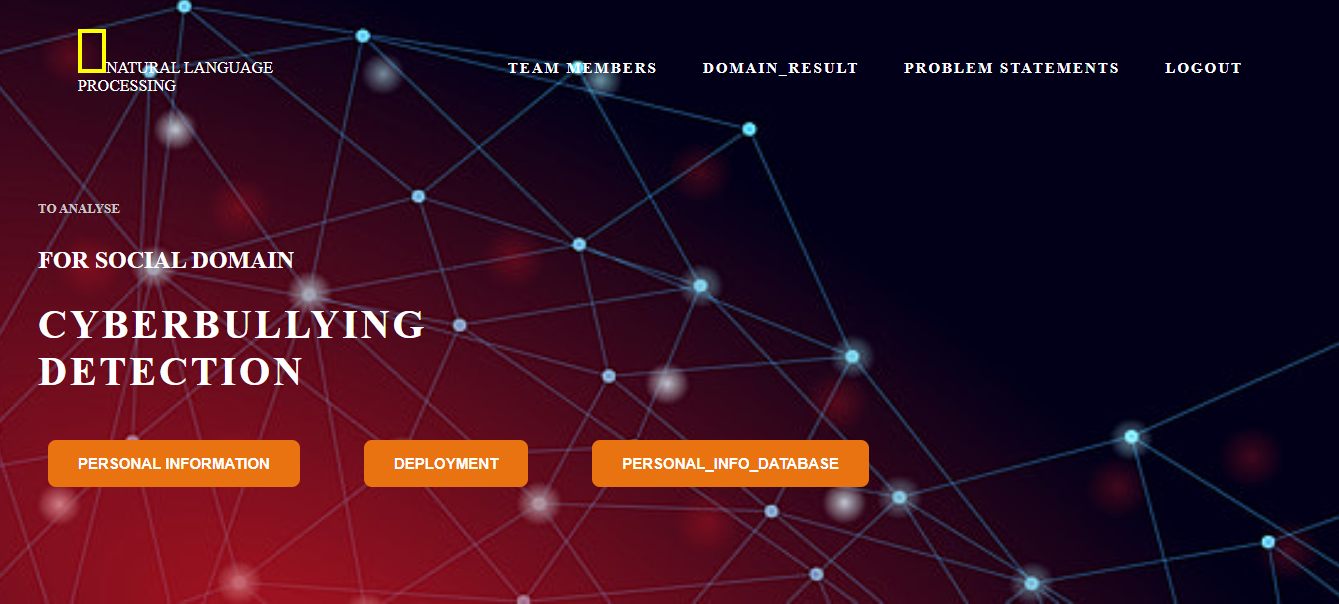
def Logout(request):

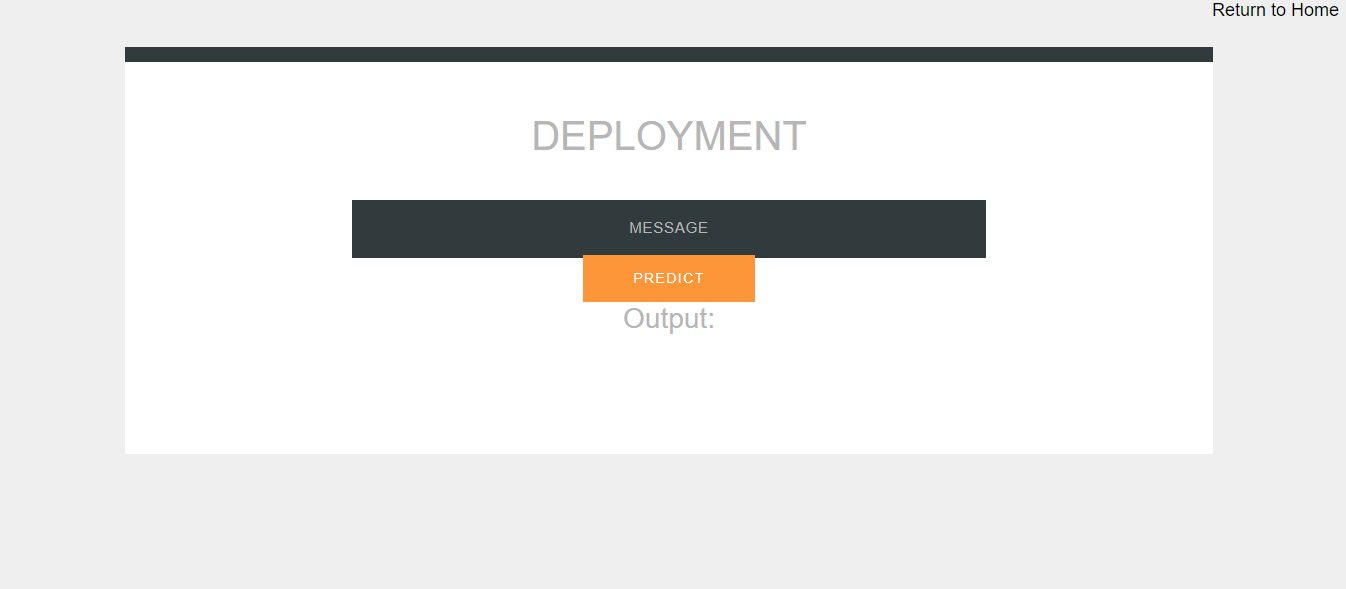
logout(request)

return redirect('Landing\_1')

**OUTPUT IMAGES:**

****

****

****

**CONCLUSION:**

In conclusion, the development and implementation of cyber bullying detection systems represent a significant stride towards fostering safer online environments for individuals of all ages. By leveraging advanced technologies such as machine learning, natural language processing, and data analytics, these systems can effectively identify and mitigate instances of cyber bullying in various online platforms. However, it's crucial to acknowledge that cyber bullying detection is an ongoing challenge that requires continuous refinement and adaptation to keep pace with evolving digital landscapes and emerging forms of online harassment. Additionally, ethical considerations surrounding data privacy, algorithm bias, and intervention strategies must be carefully addressed to ensure that these systems operate responsibly and equitably. Ultimately, through collaborative efforts among technology developers, researchers, educators, policymakers, and community stakeholders, we can work towards creating a more inclusive and respectful online ecosystem where everyone can participate safely and confidently.

While significant strides have been made in the field, there remains a pressing need for continued advancement. Future efforts should focus on refining detection algorithms to improve accuracy, incorporating diverse data sources and contextual information to enhance understanding, and developing real-time monitoring capabilities to enable timely intervention.

Moreover, it is crucial to approach this work with a user-centric perspective, prioritizing the well-being, privacy, and autonomy of individuals affected by cyber bullying. By centering the experiences and needs of users, we can design detection systems that empower individuals to navigate online spaces safely and confidently.

Furthermore, addressing cyber bullying requires a multi-faceted approach that extends beyond technological solutions. It necessitates collaboration among stakeholders including technology developers, researchers, educators, policymakers, mental health professionals, and community advocates. Together, we can work towards fostering a culture of empathy, respect, and accountability in online communities.

Ultimately, the goal of cyber bullying detection efforts is not only to identify and mitigate instances of harm but also to cultivate a digital environment where everyone feels valued, heard, and supported. By continuing to innovate and collaborate, we can move closer to realizing this vision and creating a safer, more inclusive online world for all.

Top of Form

**FUTURE WORK:**

1. **Multimodal Detection**: Integrating multiple sources of data, such as text, images, and videos, to enhance the accuracy and comprehensiveness of cyber bullying detection systems. This approach would allow for a more nuanced understanding of online interactions and enable the detection of cyber bullying across different mediums.
2. **Contextual Analysis**: Incorporating contextual information, such as social dynamics, relationships between users, and cultural nuances, into detection algorithms to improve the contextual understanding of online communication and reduce false positives.
3. **Real-time Monitoring**: Developing real-time monitoring systems capable of detecting and addressing instances of cyber bullying as they occur, thereby providing timely support and intervention to victims and preventing escalation of harmful behavior.
4. **User-Centric Approaches**: Designing detection systems with a focus on user experience and privacy, ensuring that individuals feel empowered and supported rather than surveilled or stigmatized. This could involve incorporating user feedback mechanisms, providing personalized support resources, and respecting user consent and autonomy.
5. **Adversarial Robustness**: Enhancing the robustness of detection models against adversarial attacks and evasion techniques employed by perpetrators to evade detection. This involves developing techniques to identify and mitigate adversarial behavior while maintaining the efficacy and reliability of detection systems.
6. **Longitudinal Studies**: Conducting longitudinal studies to examine the long-term impacts of cyber bullying detection and intervention efforts on individuals' well-being, mental health, and online behavior. This research can inform the refinement of detection strategies and the development of evidence-based intervention approaches.
7. **Cross-platform Integration**: Exploring methods for integrating cyber bullying detection across multiple online platforms and social media networks to create a unified approach to addressing online harassment and promoting a safer online environment.