CLASSIFICATION OF HATE SPEECH USING MACHINE LEARNINGWITH NLP

A PROJECT REPORT

Submitted by

MACHA BHARATH M (211419104153)

RAHUL C (211419104208)

RAKESH R (211419104208)

in partial fulfilment for the award

of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION,AFFILIATED TO ANNA UNIVERSITY,CHENNAI)

APRIL – 2023

PANIMALAE ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO ANNA UNIVERSITY)

BONAFIDE CERTIFICATE

Certified that this project report "**CLASSIFICATION OF HATE SPEECH USING MACHINE LEARNING WITH NLP"** is the bonafide work OF "**MACHA BHARATH M (211419104153), RAHUL C (211419104208), RAKESH R (211419104211)"** Who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA M.E.,Ph.D.,	**MR.KRISHNAMOORTHY,M.E,M.B.A,(PH.D.)**
HEAD OF THE DEPARTMENT,	**ASSOCIATE PROFESSOR**
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE COLLEGE	PANIMALAR ENGINEERING COLLEGE
NAZARATHPETTAI,	NAZARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI-600 123.	CHENNAI-600 123.

Certified that the above mentioned students were examined in the End Semester project Viva-Voice held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **MACHABHARATH M (211419104153) , RAHUL C (211419104208) , RAKESH R (211419104211) hereby declare that this project report titled "CLASSIFICATION OF HATE SPEECH USING MACHINE LEARNING WITH NLP"** under the guidance of **Mr.KRISHNAMOORTHY,M.E.,M.B.A.,(PH.D),** is the original work done by us and we have not plagarzised or submitted to any other degree in many university bu us.

MACHABHARATH M

RAHUL C

RAKESH R

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.**, for this kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHI KUMAR, M.E., Ph.D., and Dr.SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for providing us with necessary facilities toundertake this project.

We also express our gratitude to our Principa**l Dr.K.Mani, M.E., Ph.D.,**
Who facilitated us in completing the project.

We thank the Head of the Department**, Dr.L.JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my project coordinator **Mr M.MOHAN, M.E., (Ph.D).,** and Project Guide **Mr. M. KRISHNAMOORTHY, M.E., M.B.A, (Ph.D.)**., and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

MACHA BHARATH M

RAHUL C

RAKESH R

ABSTRACT

Hate speech is a complex and multifaceted form of harmful or offensive content that targets individuals or groups based on their race, ethnicity, gender, religion, sexual orientation, or other characteristics. Online toxic discourses could result in conflicts between groups or harm to online communities. Due to the increase in the online social network development in the past few years due to different purposes, hate speech appears in large numbers and the online world has widespread. By these online hate speech online social networks users can get affected easily.

The impact of hate speech is enormous as it can cause psychological distress, anxiety, depression, and even physical harm to individuals and groups. Moreover, it can result in the marginalization and exclusion of targeted individuals and communities from society. In some cases, hate speech can also spread more rapidly than true information, leading to misinformation and a distorted public discourse.

To address the problem of hate speech, there is a need for machine learning models that can detect these speech automatically. Machine learning models are designed to build using algorithms so that they can classify whether a speech is hate speech, offensive speech, or not hate and offensive speech.

Machine learning models can be trained using various techniques such as supervised, unsupervised, and reinforcement learning. The supervised learning method requires a dataset that contains labeled examples of hate speech, offensive speech, and non-offensive speech. The model is then trained to classify new examples based on the features extracted from the dataset.

Unsupervised learning methods are used when the dataset is not labeled. The model is trained to discover patterns and relationships in the dataset and categorize new examples accordingly. Reinforcement learning is a method that uses feedback from

the environment to train the model. The model learns by trial and error, and it receives rewards for correct classification.

In conclusion, the development of machine learning models for detecting hate speech is crucial for addressing the problem of online toxicity. speech, enabling online platforms to take appropriate measures to prevent the spread of hate speech and protect their users.

CHAPTER 2
INTRODUCTION

2. INTRODUCTION

2.1 Data Science

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

2.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa). Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

2.3Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation.

CHAPTER 3

PROPOSED SYSTEM

3.1 Proposed System

The proposed model is to build a machine learning model that is capable of classifying whether the speech is hate speech or not. The hate speech are considered to be widespread and controlling them is very difficult as the world is developing toward digital everyone now has access to internet and they can post whatever they want. So there is a greater chance for the people to get misguided. The machine learning is generally build to tackle these type of complicated task. It takes more amount of time to analyse these type of data manually. The machine learning can be used to classify the speech is hate speech or not by using the previous data and make them to understand the pattern and improve the accuracy of the model by adjusting parameters and use that model as the classification model. Different algorithms can be compared and the best model can be used for classification purpose.

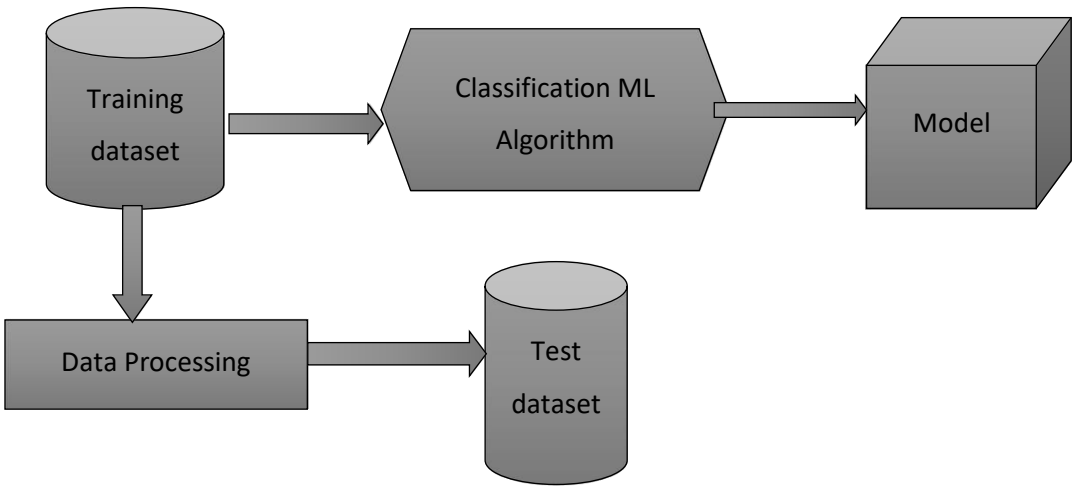


FIG NO : 3.1 Architecture of Proposed modeL

3.2 Advantages:

- Machine learning algorithms are used.
- Accuracy may be improved.

CHAPTER 4

LITERATURE SURVEY

4.1 Literature survey

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources.

Title : Using Machine Learning for Detection of Hate Speech and Offensive Code-Mixed Social Media text

Author: Varsha Pathak, Manish Joshi

Year : 2020

This paper describes the system submitted by our team KBCNMUJAL for Task 2 of the shared task "Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC)" at FIRE 2020. The datasets of two Dravidian languages viz Malayalam and Tamil of size **4000 rows** each, are shared by **HASOC** organizers. These datasets are used to train the machine using different machine learning algorithms, based on classification and regression models. The datasets consist of twitter messages with two class labels "offensive" and "not offensive". The machine is trained to classify such social media messages in these two categories. Appropriate n-gram feature sets are extracted to learn the specific characteristics of the Hate Speech text messages. These feature models are based on TF-IDF weights of n-gram. The referred work and respective experiments show that the features such as word, character and combined model of word and character n-grams could be used to identify the term patterns of offensive text contents.

4.2 Title : Detection of Hate Speech in Videos Using Machine Learning

Author: Ching Seh Wu Unnathi Bhandary

Year : 2020

With the progression of the Internet and social media, people are given multiple platforms to share their thoughts and opinions about various subject matters freely. However, this freedom of speech is misused to direct hate towards individuals or group of people due to their race, religion, gender etc. The rise of hate speech has led to conflicts and cases of cyber bullying, causing many organizations to look for optimal solutions to solve this problem. Developments in the field of machine learning and deep learning have piqued the interest of researchers, leading them to research and implement solutions to solve the problem of hate speech.

4.3 Title : Automatic Hate Speech Detection using Machine Learning: A Comparative Study

Author: Sindhu Abro, Zahid Hussain

Year : 2020

The increasing use of social media and information sharing has given major benefits to humanity. However, this has also given rise to a variety of challenges including the spreading and sharing of hate speech messages. Thus, to solve this emerging issue in social media sites, recent studies employed a variety of feature engineering techniques and machine learning algorithms to automatically detect the hate speech messages on different datasets. However, to the best of our knowledge, there is no study to compare the variety of feature engineering techniques and machine learning algorithms to evaluate which feature engineering technique and machine learning algorithm outperform on a standard publicly available dataset.

CHAPTER 5

SYSTEM STUDY

5.1 Objectives

The goal is to develop a machine learning model for Hate speech classification, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

5.2 Project Goals

- Exploration data analysis of variable identification
- Loading the given dataset
- Import required libraries packages
- Analyze the general properties
- Find duplicate and missing values
- Checking unique and count values
- Uni-variate data analysis
- Rename, add data and drop the data
- To specify data type
- Exploration data analysis of bi-variate and multi-variate
- Plot diagram of pairplot, heatmap, bar chart and Histogram
- Method of Outlier detection with feature engineering
- Pre-processing the given dataset
- Splitting the test and training dataset
- Comparing the Decision tree and Logistic regression model and random forest etc.

5.3 Scope of the Project

Here the scope of the project is that integration of social media hate speech with computer-based prediction could reduce errors and improve prediction outcome. This suggestion is promising as data modeling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of Hate speech classification.

CHAPTER 6

FEASIBILITY STUDY

6.1 Feasibility study

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

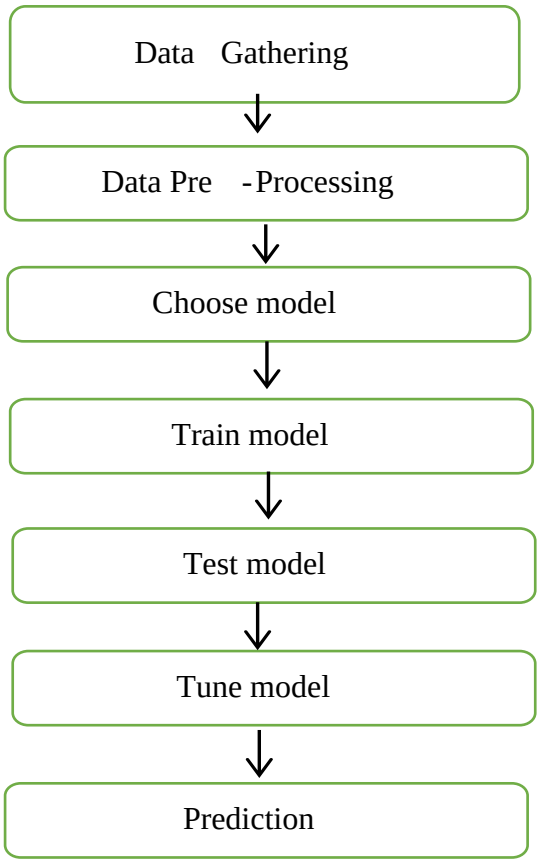
Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

Building the classification model

The prediction of Heart attack, a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

Construction of a Predictive Model



Running the Jupyter Notebook

Launching *Jupyter Notebook App:* The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): "jupyter notebook".

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

.

.

CHAPTER 7
PROJECT REQUIREMENTS

Project Requirements

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

7.1 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.

7.2 Non-Functional Requirements

Process of functional steps,

1.Problem define

2.Preparing data

3.Evaluating algorithms

4.Improving results

5.Prediction the result

7.3 Environmental Requirements 1.

Software Requirements

Operating System : Windows 10 or later

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

CHAPTER 8
SOFTWARE DESCRIPTION

8.1 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.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

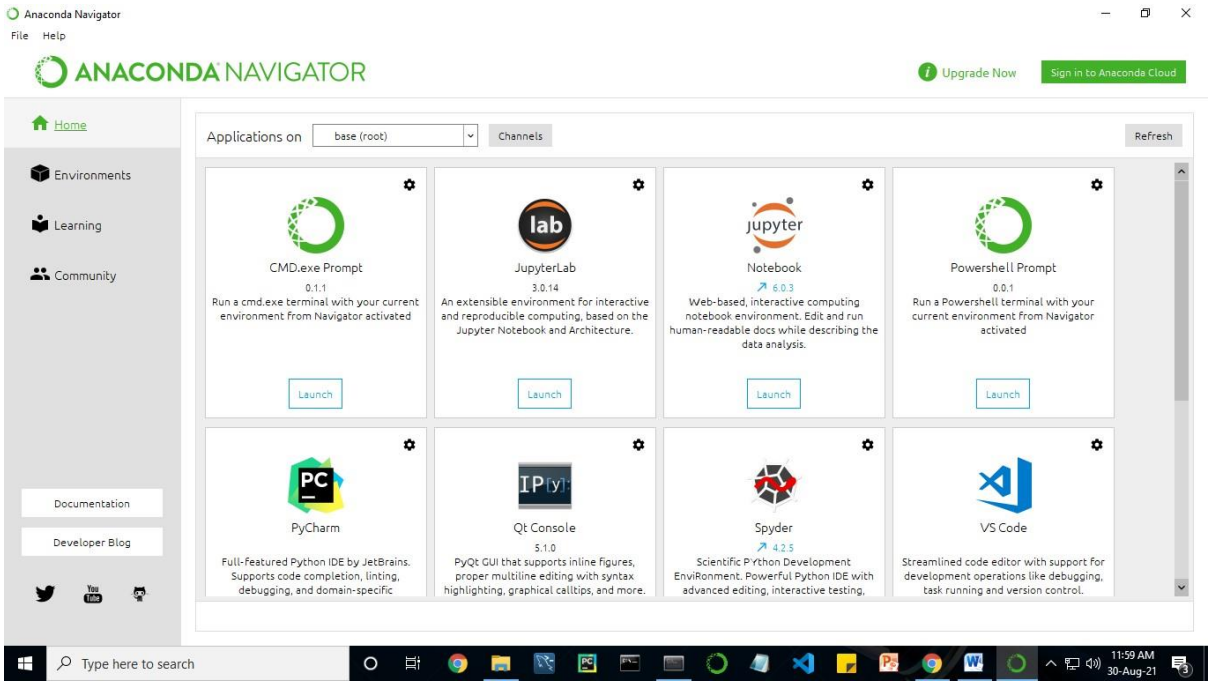


FIG NO : 8.1.1

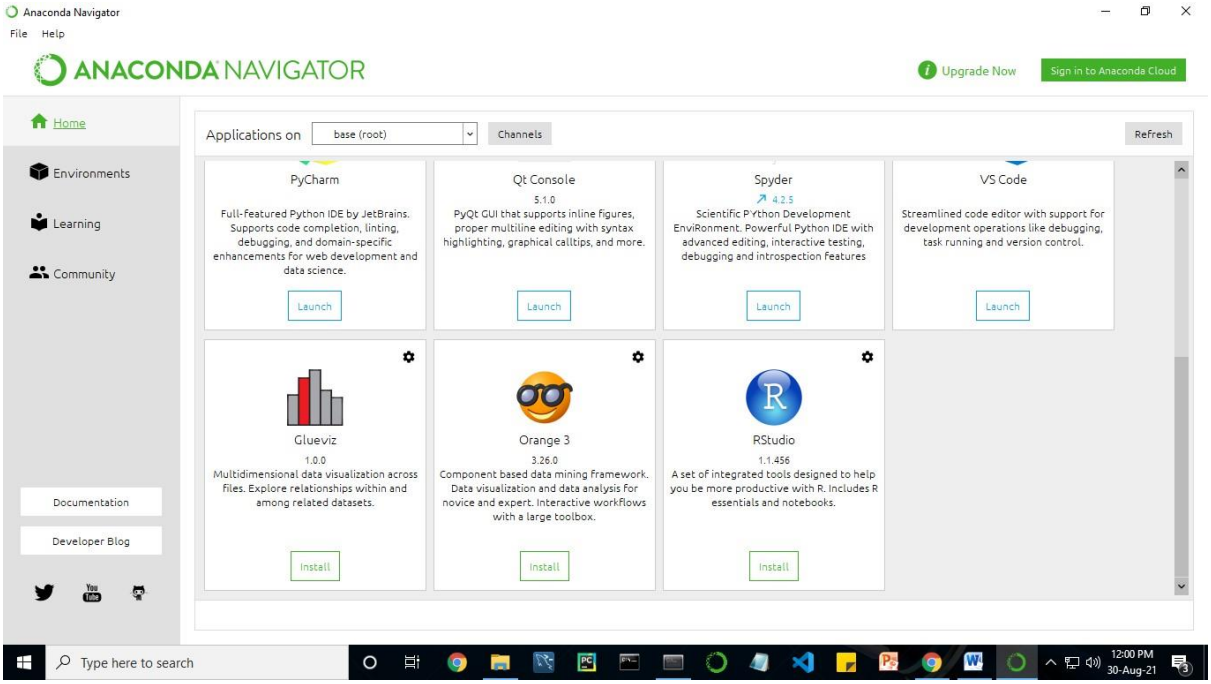


FIG NO : 8.1.2

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.

8.2 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, 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.

8.2.1. JUPYTER NOTEBOOK APP

The JUPYTER NOTEBOOK App is a server-client application that allows editing and running notebook documents 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.

Kernel: A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

CHAPTER 9

SYSTEM ARCHITECTURE

9.1 System Architecture

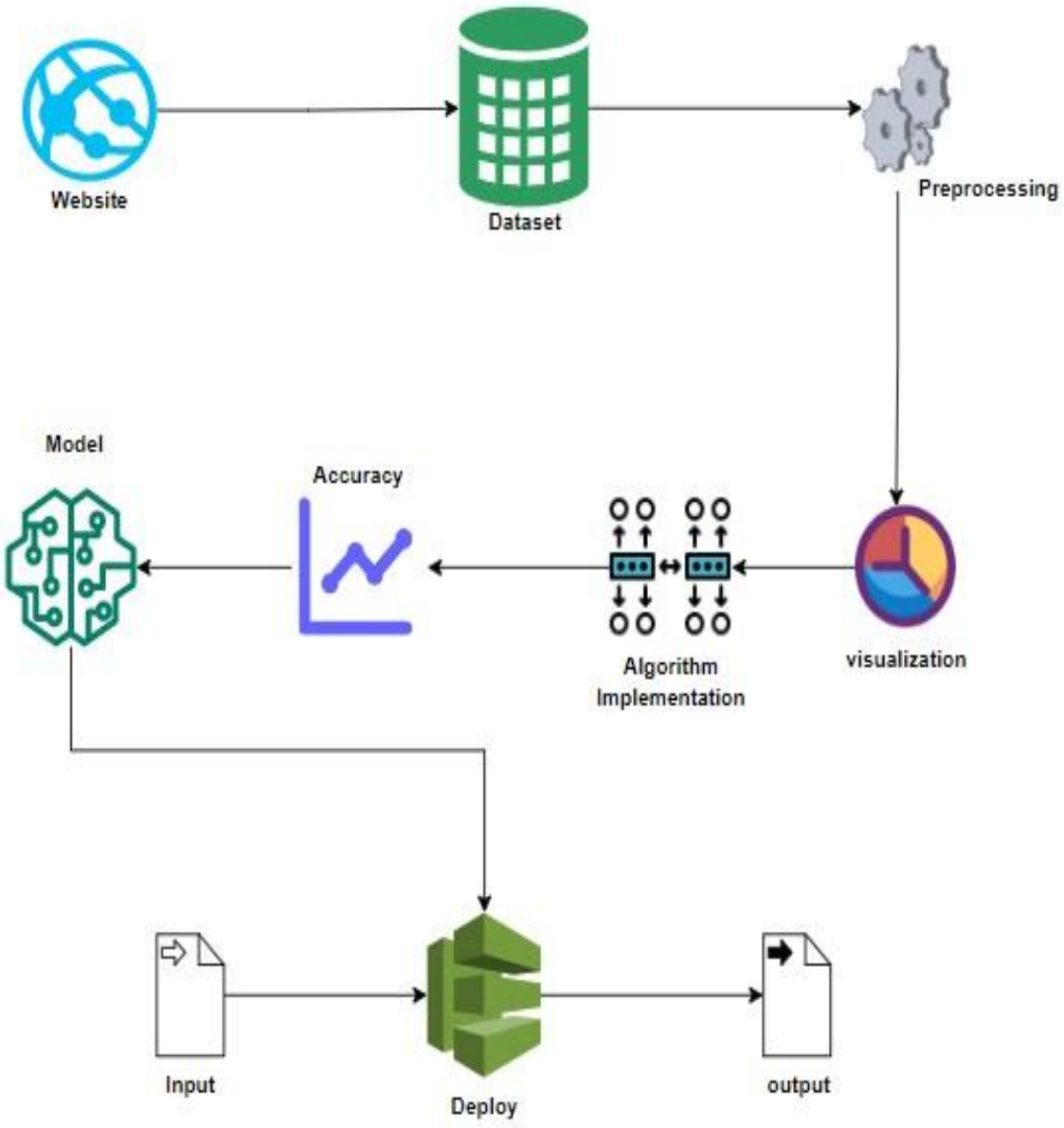


FIG NO : 9.1.1

CHAPTER 10

WORKFLOW DIAGRAM

10.1 Work flow diagram

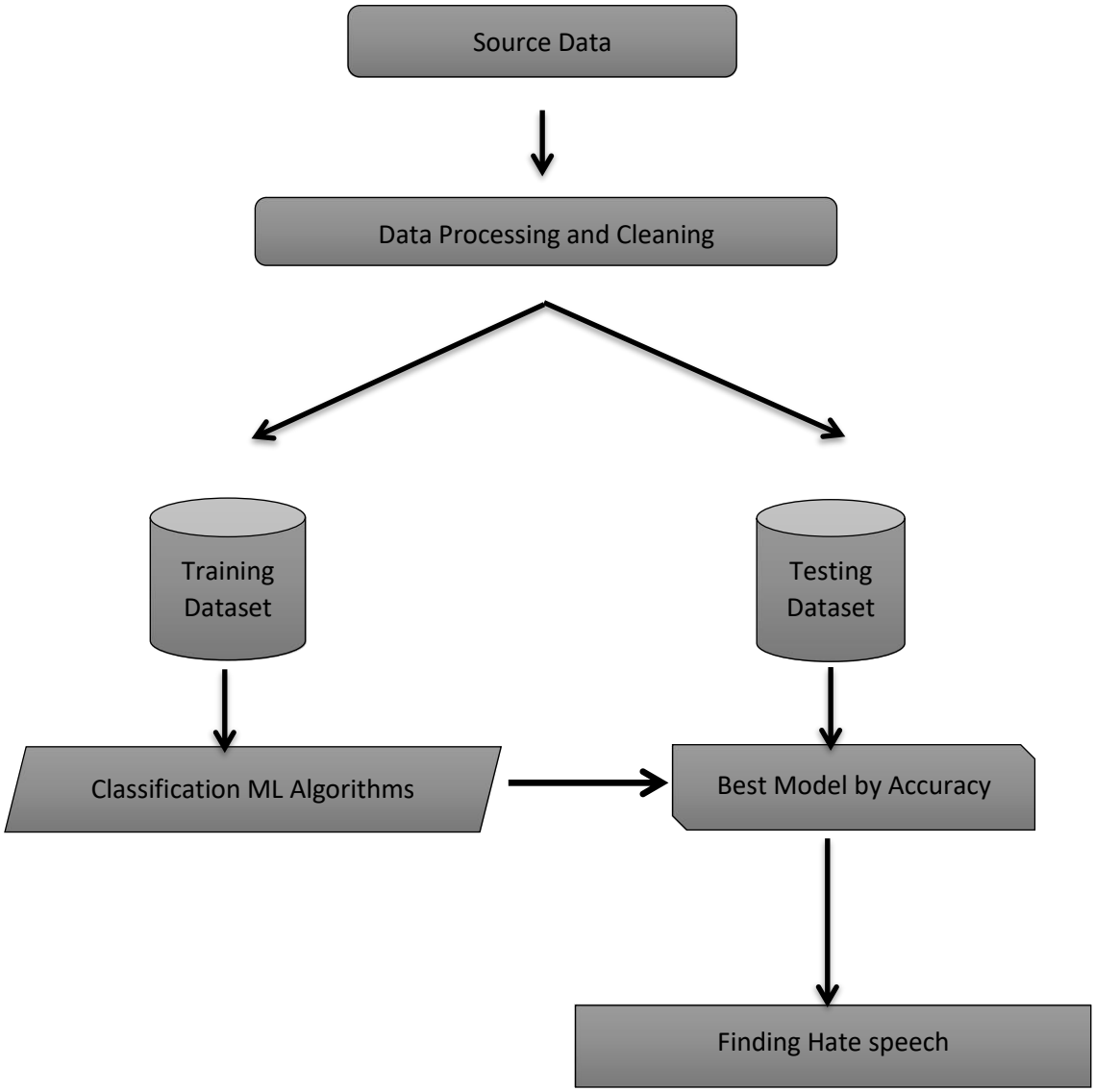


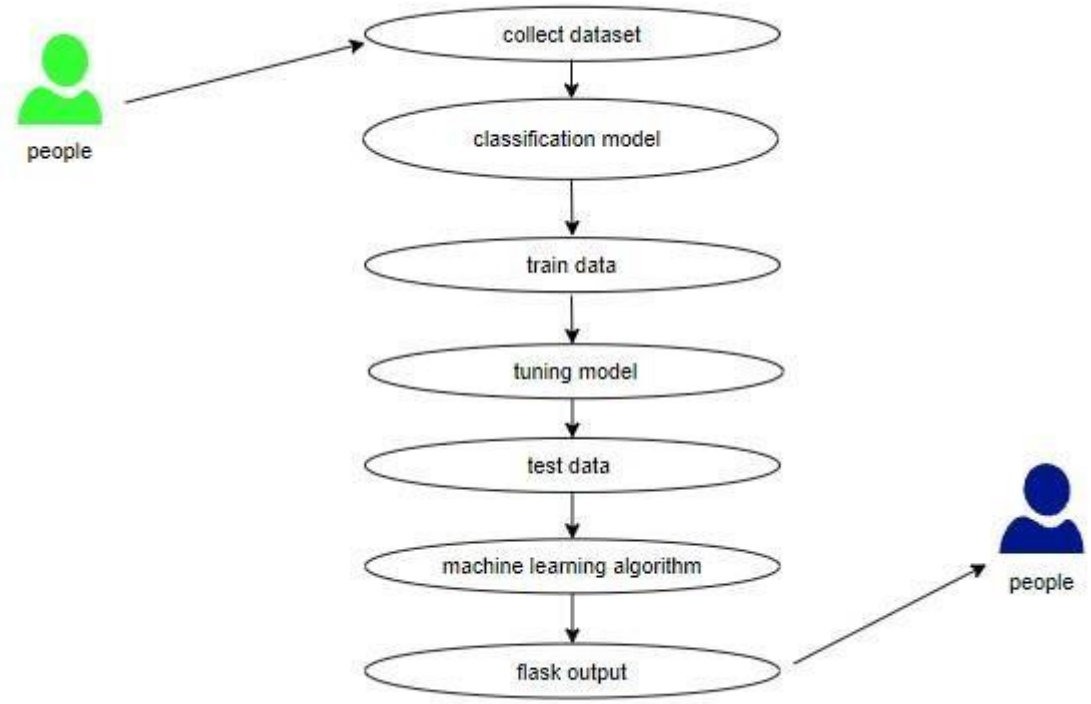
FIG NO :10.1.1

A workflow diagram, sometimes called a **workflow chart**, is a visual representation of a business process (or workflow), usually done through a flowchart. It is a visual way for business analysis to show how work gets accomplished.

CHAPTER 11

USE CASE DIAGRAM

11.1 Use Case Diagram



Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

CHAPTER 12

CLASS DIAGRAM

12.1 CLASS DIAGRAM

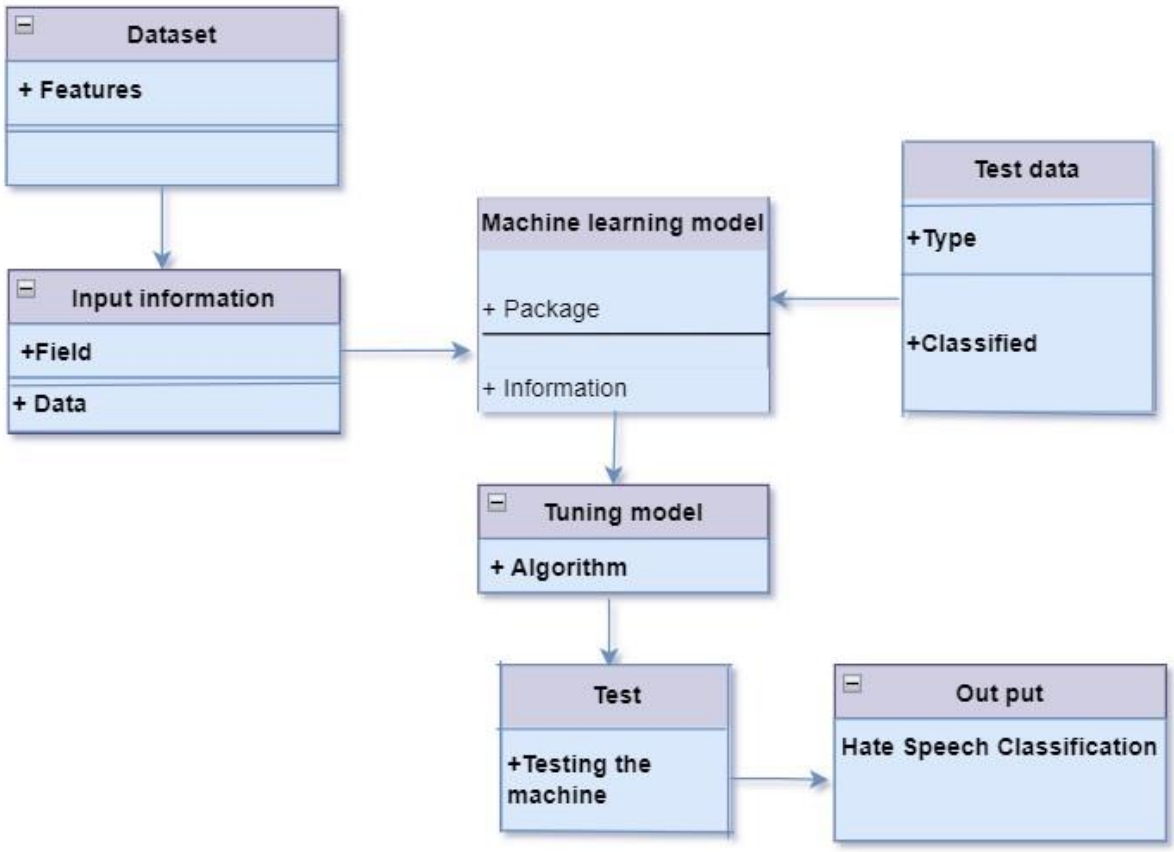


FIG NO : 12.1.1

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. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

CHAPTER 13

ACTIVITY DIAGRAM

13.1 Activity Diagram

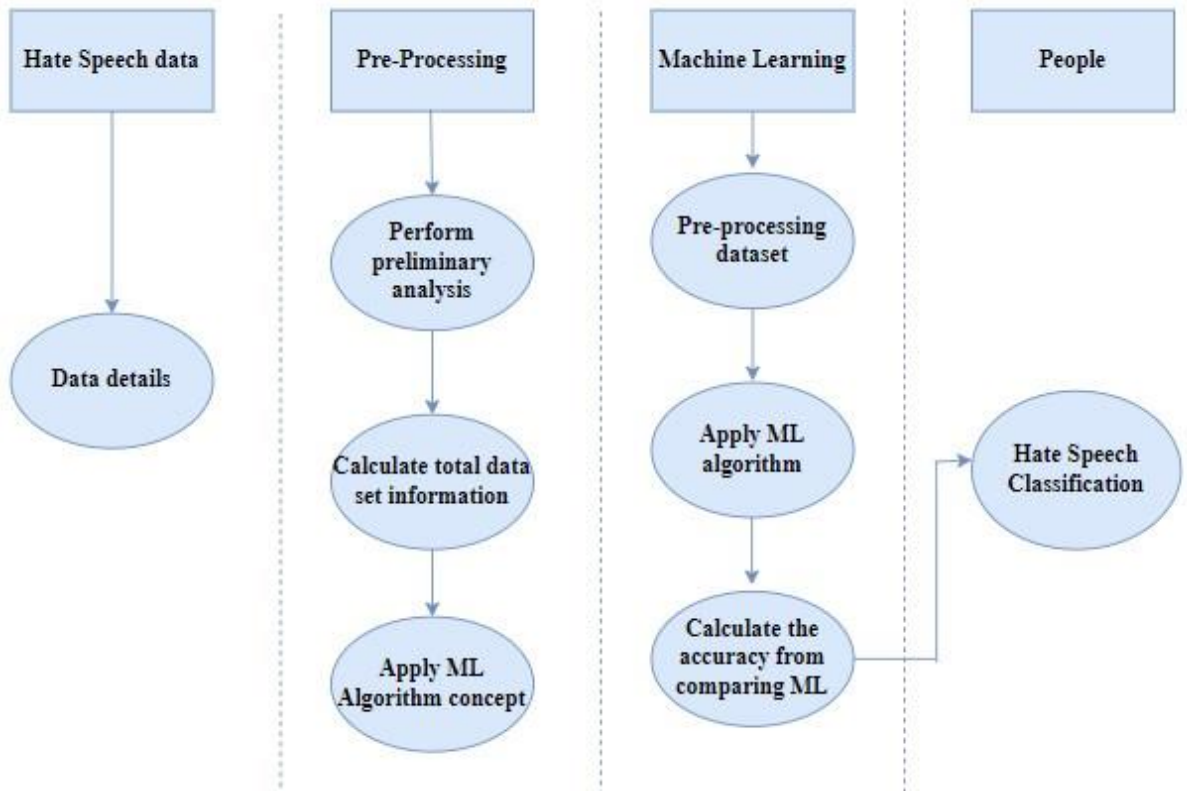


FIG NO 13.1.1

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not.

CHAPTER 14

MODULE DESCRIPTION

14.1 Module description

14.1.1 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.

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

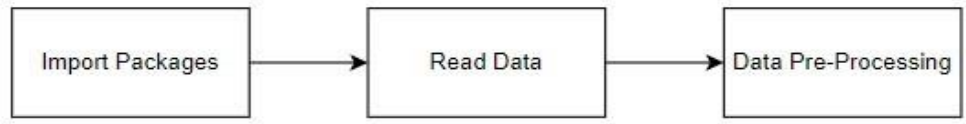
- ➢ import libraries for access and functional purpose and read the given dataset
- ➢ General Properties of Analyzing the given dataset
- ➢ Display the given dataset in the form of data frame
- ➢ show columns
- ➢ shape of the data frame
- ➢ To describe the data frame
- ➢ Checking data type and information 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

14.1.2 Data Validation/ Cleaning/Preparing Process

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning models and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value data in analytics and decision making.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

14.1.3 Comparing Algorithm with prediction in the form of best accuracy result

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. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 5 different algorithms are compared:

- ➢ Logistic Regression
- ➢ Random Forest
- ➢ Decision Tree Classifier

- ➢ Naive Bayes
- ➢ AdaBoost

14.1.4 Prediction result by accuracy

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

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)

14.1.5 ALGORITHM AND TECHNIQUES

Algorithm Explanation

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or nonspam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labelled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabelled new data.

sklearn

- ➢ In python, sklearn is a machine learning package which include a lot of ML algorithms.
- ➢ Here, we are using some of its modules like train_test_split, DecisionTreeClassifier or Logistic Regression and accuracy_score.

NumPy

- ➢ It is a numeric python module which provides fast maths functions for calculations.
- ➢ It is used to read data in numpy arrays and for manipulation purpose.

Pandas

- ➢ Used to read and write different files.
- ➢ Data manipulation can be done easily with data frames.

14.1.6 Gaussian Naïve Bayes

Naïve Bayes is a probabilistic machine learning algorithm used for many classification functions and is based on the Bayes theorem. Gaussian Naïve Bayes is the extension of naïve Bayes. While other functions are used to estimate data distribution, Gaussian or normal distribution is the simplest to implement as you will need to calculate the mean and standard deviation for the training data.

Naive Bayes is a probabilistic machine learning algorithm that can be used in several classification tasks. Typical applications of Naive Bayes are classification of documents, filtering spam, prediction and so on. This algorithm is based on the discoveries of Thomas Bayes and hence its name.

The name "Naïve" is used because the algorithm incorporates features in its model that are independent of each other. Any modifications in the value of one feature do not directly impact the value of any other feature of the algorithm. The main advantage of the Naïve Bayes algorithm is that it is a simple yet powerful algorithm.

It is based on the probabilistic model where the algorithm can be coded easily, and predictions did quickly in real-time. Hence this algorithm is the typical choice to solve real-world problems as it can be tuned to respond to user requests instantly. But before we dive deep into Naïve Bayes and Gaussian Naïve Bayes, we must know what is meant by conditional probability.

14.1.7 AdaBoost

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. Boosting is used to reduce bias as well as variance for supervised learning. It works on the principle of learners growing sequentially.

Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference

First, let us discuss how boosting works. It makes ‘n’ number of decision trees during the data training period. As the first decision tree/model is made, the incorrectly classified record in the first model is given priority. Only these records are sent as input for the second model. The process goes on until we specify a number of base learners we want to create. Remember, repetition of records is allowed with all boosting techniques.

14.1.8 K-Nearest Neighbour

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a nonparametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered “plurality voting”, the term, “majority vote” is more commonly used in literature. The distinction between these terminologies is that “majority voting” technically requires a majority of greater than 50%, which primarily works when there are only two categories.
When you have multiple classes—e.g. four categories, you don’t necessarily need 50% of the vote to make a conclusion about a class; you could assign a class label with a vote of greater than 25%.

CHAPTER 15

DEPLOYMENT

15.1 Deployment

15.1.1 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.[2] It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004.[6] According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application.[7][8][9] The name is a play on the earlier Bottle framework.[7]

When Ronacher and Georg Brandl created a bulletin board system written in Python in 2004, the Pocoo projects Werkzeug and Jinja were developed.[10]

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.[11][12] Since 2018, Flask-related data and objects can be rendered with Bootstrap.[13]

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django,[14] and was voted the most popular web framework in the Python Developers Survey 2018, 2019, 2020 and 2021

Features

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching

- Uses Jinja templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Complete documentation
- Google App Engine compatibility
- Extensions available to extend functionality

15.1.2 Parameters

- **rule** (str) – The URL rule string.
- **endpoint** (Optional[str]) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to view_func.__name__.
- **view_func** (Optional[Callable]) – The view function to associate with the endpoint name.
- **provide_automatic_options** (Optional[bool])–Add OPTIONS method and respond to OPTIONS requests automatically.
- **options** (Any) – Extra options passed to the Rule object. Return type -- None

After_Request(f)

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object.

This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use teardown_request() for that. **Parameters**

f (Callable[[Response], Response]) Return
type

Callable[[Response], Response] after_request_funcs:
t.Dict[AppOrBlueprintKey,

t.List[AfterRequestCallable]]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the after_request() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time. app_context()

Create an AppContext. Use as a with block to push the context, which will make current_app point at this application.

An application context is automatically pushed by RequestContext.push() when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With app.app_context():

Init_db()

CHAPTER 16

HTML

16.1 HTML INTRODUCTION

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.

HTML OUTLOOK DIAGRAM

<!DOCTYPE html> — This tag **specifies the language** you will write on the page. In this case, the language is HTML 5.

<html> — This tag signals that from here on we are going to write in HTML code.

<head> — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

<body> — This is where the **content of the page** goes.

16.1.1 Head Tag

```
<head>
<title>My First Webpage</title>
<meta charset="UTF-8">
<meta name="description" content="This field contains information about your page. It is usually around two sentences long.">.
<meta name="author" content="Conor Sheils">
</header>
```

16.1.2 Add Text In HTML

Adding text to our HTML page is simple using an element opened with the tag <p> which **creates a new paragraph**. We place all of our regular text inside the element <p>.

When we write text in HTML, we also have a number of other elements we can use **to control the text or make it appear in a certain way.**

Add Links In HTML

As you may have noticed, the internet is made up of lots of links.

Almost everything you click on while surfing the web is a link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the **<a>** tag. This element is the first that we've met which uses an attribute and so it **looks different to previously mentioned tags.**

Google

16.1.3 Image Tag

In today's modern digital world, images are everything. The <img> tag has everything you need to display images on your site. Much like the <a> anchor element, <img> also contains an attribute.

The attribute features information for your computer regarding the source, height, width and alt text of the image

<img src="yourimage.jpg" alt="Describe the image" height="X" width="X">

CHAPTER 17

CSS (CASCADING STYLE SHEET)

17.1 CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

17.1.1 CSS Comment

- Comments don't render on the browser
- Helps to understand our code better and makes it readable.
- Helps to debug our code

17.1.2 Inline CSS

- Before CSS this was the only way to apply styles
- Not an efficient way to write as it has a lot of redundancy
- Self-contained
- Uniquely applied on each element • The idea of separation of concerns was lost
- Example:

<h3 style = "color:red"> Have a great day </h3>

 I did this, I did that

17.1.3 Internal CSS

- With the help of style tag, we can apply styles within the HTML file
- Redundancy is removed
- But the idea of separation of concerns still lost
- Uniquely applied on a single document

17.1.4 External CSS

- With the help of <link> tag in the head tag, we can apply styles
- Reference is added
- File saved with .css extension
- Redundancy is removed
- The idea of separation of concerns is maintained
- Uniquely applied to each document

17.1.5 CSS Colors

- There are different colouring schemes in CSS
- **RGB**-This starts with RGB and takes 3 parameter
- **HEX**-Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
- **RGBA**-This starts with RGB and takes 4 parameter

17.1.6 CSS BoxModel

- Every element in CSS can be represented using the BOX model
- It allows us to add a border and define space between the content
- It helps the developer to develop and manipulate the elements
- It consists of 4 edges
    - Content edge – It comprises of the actual content

- Padding edge – It lies in between content and border edge ○ Border edge – Padding is followed by the border edge ○ Margin edge – It is an outside border and controls the margin of the element

CHAPTER 18

CODING

18.1 Module – 1 Pre-

Processing

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd data =
pd.read_csv('heart.csv') data
Before removing the null data
data.shape After removing
the null data df =
data.dropna() df.shape
df.columns df.isnull().sum()
df.info() df.columns
df.duplicated()
df.duplicated().sum()
df.describe() df.columns
df.Gender.unique()
df.Locality.unique()
df.BP.unique()
df.Age.unique()
df.Smoking.unique()
df.Mortality.unique()
df.Mortality.value_counts()
pd.crosstab(df.Age,df.Mortali
ty)
pd.crosstab(df.Gender,df.Mor
tality) df.corr() df.columns
Before LabelEncoder
df.head()
```

After LabelEncoder from sklearn.preprocessing import LabelEncoder var_mod = ['Gender', 'Locality', 'Marital_status', 'Sleep', 'Depression', 'Smoking', 'Hypersensitivity'] le
= LabelEncoder()
for i in var_mod:
df[i] = le.fit_transform(df[i]).astype(int) df.head()

18.1.1 Module – 2 Visualization

import warnings warnings.filterwarnings('ignore')
import pandas as pd import matplotlib.pyplot as plt
import seaborn as sns sns.set() data =
pd.read_csv('heart.csv') df = data.dropna()
df.columns sns.countplot(x="Mortality", data=df,
palette="husl") df['Age'].hist(figsize=(5,5),
color='brown', alpha=1) plt.xlabel('Age')
plt.ylabel('Count') plt.title('Age and its Count')
df['chol'].hist(figsize=(5,5), color='brown', alpha=1)
plt.xlabel('Chol') plt.ylabel('Count') plt.title('Age and
its Count') def piechart(df, variable):
dataframe_pie = df[variable].value_counts() ax =
dataframe_pie.plot.pie(figsize=(8,8), autopct='%1.2f%%', fontsize = 12)
ax.set_title(variable + ' \n', fontsize = 15) piechart(df, 'Smoking') piechart(df,
'Gender') g = sns.catplot("Mortality", data=df, kind='count', hue='Age',
margin_titles=False)
g.set_ylabels('Number of persons')

g.fig.set_figheight(5)

g.fig.set_figwidth(20) sns.pairplot(df) fig,
ax = plt.subplots(figsize=(25,12))
sns.heatmap(df.corr(),annot = True, ax=ax)
plt.show() df.head()
#preprocessing, split test and dataset, split response variable

X = df.drop(labels='Mortality', axis=1)

```
#Response variable y =
df.loc[:,'Mortality']
#Splitting for train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42, stratify=y) print("Number of training dataset: ",
len(X_train)) print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))
Implementing Logistic Algo from sklearn.linear_model import
LogisticRegression from sklearn.metrics import confusion_matrix,
classification_report, accuracy_score, plot_confusion_matrix
Training lr = LogisticRegression() lr.fit(X_train,y_train) predicted =
lr.predict(X_test) Finding Accuracy accuracy =
accuracy_score(y_test,predicted)
print('Accuracy of Logistic Regression',accuracy*100) Finding
Clasiification Report cr = classification_report(y_test,predicted)
print('Classification report\n\n',cr) Finding Confusion matrix cm =
confusion_matrix(y_test,predicted) print('Confusion
matrix\n\n',cm) import matplotlib.pyplot as plt fig, ax =
plt.subplots(figsize=(10,10)) plot_confusion_matrix(lr, X_test,
y_test, ax=ax) plt.title('Confusion matrix of Logistic Regression')
plt.show() df2 = pd.DataFrame() df2["y_test"] = y_test
df2["predicted"] = predicted df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5)) plt.plot(df2["predicted"], marker='x',
linestyle='dotted', color='red') plt.plot(df2["y_test"],  marker='o',
linestyle='dotted', color='green') plt.show()
```

18.1.2 Module – 3

Logistic Regression Algorithm

```
#import library packages import
pandas as pd import warnings
warnings.filterwarnings('ignore')
# Load given dataset data =
pd.read_csv("heart.csv") df =
data.dropna() df.columns del
df['Age.Group'] del
df['Locality'] del df['Gender']
df.columns df.head() from
sklearn.preprocessing import
LabelEncoder var_mod =
['Marital_status', 'Sleep',
'Depression', 'Smoking',
'Hypersensitivity'] le =
LabelEncoder() for i in
var_mod:
df[i] = le.fit_transform(df[i]).astype(int) df.head()
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='Mortality', axis=1)
#Response variable y =
df.loc[:,'Mortality']
#Splitting for train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42, stratify=y) print("Number of training dataset: ",
len(X_train)) print("Number of test dataset: ", len(X_test))
```

```
print("Total number of dataset: ", len(X_train)+len(X_test))
Implementing Logistic Algo from sklearn.linear_model
import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, plot_confusion_matrix Training lr =
LogisticRegression() lr.fit(X_train,y_train) predicted =
lr.predict(X_test) Finding Accuracy accuracy =
accuracy_score(y_test,predicted) print('Accuracy of Logistic
Regression',accuracy*100) Finding Clasiification Report cr =
classification_report(y_test,predicted) print('Classification
report\n\n',cr) Finding Confusion matrix cm =
confusion_matrix(y_test,predicted) print('Confusion matrix\n\n',cm)
import matplotlib.pyplot as plt fig, ax =
plt.subplots(figsize=(10,10)) plot_confusion_matrix(lr, X_test,
y_test, ax=ax) plt.title('Confusion matrix of Logistic Regression')
plt.show() df2 = pd.DataFrame() df2["y_test"] = y_test
df2["predicted"] = predicted df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5)) plt.plot(df2["predicted"], marker='x',
linestyle='dotted', color='red') plt.plot(df2["y_test"],  marker='o',
linestyle='dotted', color='green') plt.show()
```

18.1.3 Module – 4

MLPCLASSIFIER ALGORITHM

```
#import library packages import

pandas as pd import warnings

warnings.filterwarnings('ignore')

# Load given dataset data = pd.read_csv("heart.csv") df = data.dropna() del

df['Age.Group'] del df['Locality'] del df['Gender'] df.columns df.head() from

sklearn.preprocessing import LabelEncoder var_mod = ['Marital_status',

'Sleep', 'Depression', 'Smoking', 'Hypersensitivity'] le = LabelEncoder() for i in

var_mod:

 df[i] = le.fit_transform(df[i]).astype(int) df.head()

#preprocessing, split test and dataset, split response variable

X = df.drop(labels='Mortality', axis=1)

#Response variable y = df.loc[:,'Mortality']

#Splitting for train and test from

sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,

random_state=42, stratify=y) print("Number of training dataset: ",

len(X_train)) print("Number of test dataset: ", len(X_test))
```

```
print("Total number of dataset: ", len(X_train)+len(X_test))
```

Implementing MLPClassifier Algo from sklearn.neural_network

```
import MLPClassifier

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, plot_confusion_matrix
```

Training

```
mlp = MLPClassifier() mlp.fit(X_train,y_train)

predicted = mlp.predict(X_test) Finding Accuracy

accuracy = accuracy_score(y_test,predicted)

print('Accuracy of MLPClassifier',accuracy*100)
```

Finding Clasiification Report cr =

```
classification_report(y_test,predicted)

print('Classification report\n\n',cr) Finding
```

Confusion matrix cm =

```
confusion_matrix(y_test,predicted)

print('Confusion matrix\n\n',cm) import

matplotlib.pyplot as plt fig, ax =

plt.subplots(figsize=(10,10))

plot_confusion_matrix(mlp, X_test, y_test, ax=ax)
```

```
plt.title('Confusion matrix of MLPClassifier')

plt.show() df2 = pd.DataFrame()

df2["y_test"] = y_test df2["predicted"] = predicted

df2.reset_index(inplace=True) plt.figure(figsize=(20, 5))

plt.plot(df2["predicted"], marker='x', linestyle='dotted', color='red')

plt.plot(df2["y_test"],  marker='o', linestyle='dotted', color='green')

plt.show()
```

18.1.4 Module – 5

RANDOM FOREST ALGORITHM

```
#import library packages import
pandas as pd import warnings
warnings.filterwarnings('ignore')
# Load given dataset data =
pd.read_csv("heart.csv") df =
data.dropna() del
df['Age.Group'] del
df['Locality'] del df['Gender']
df.columns df.head() from sklearn.preprocessing import LabelEncoder
var_mod = ['Marital_status', 'Sleep', 'Depression', 'Smoking', 'Hypersensitivity']
le = LabelEncoder() for i in var_mod:
df[i] = le.fit_transform(df[i]).astype(int) df.head()
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='Mortality', axis=1)
#Response variable y = df.loc[:,'Mortality']
#Splitting for train and test from
sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42, stratify=y) print("Number of training dataset: ",
```

```
len(X_train)) print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))
Implementing RandomForestClassifier Algo from sklearn.ensemble
import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, plot_confusion_matrix Training rf =
RandomForestClassifier() rf.fit(X_train,y_train) predicted =
rf.predict(X_test) Finding Accuracy accuracy =
accuracy_score(y_test,predicted) print('Accuracy of Random Forest
Classifier',accuracy*100) Finding Clasiification Report cr =
classification_report(y_test,predicted) print('Classification
report\n\n',cr) Finding Confusion matrix cm =
confusion_matrix(y_test,predicted) print('Confusion matrix\n\n',cm)
import matplotlib.pyplot as plt fig, ax =
plt.subplots(figsize=(10,10)) plot_confusion_matrix(rf, X_test,
y_test, ax=ax) plt.title('Confusion matrix of Random Forest
Classifier') plt.show() import matplotlib.pyplot as plt

df2 = pd.DataFrame() df2["y_test"] = y_test df2["predicted"] =
predicted df2.reset_index(inplace=True) plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"], marker='x', linestyle='dashed', color='red')
```

plt.plot(df2["y_test"], marker='o', linestyle='dashed', color='green')

plt.show() from joblib import dump dump(rf, 'RF.pkl')

18.1.5 Module – 6

CATBOOST CLASSIFIER ALGORITHM

#import library packages

#import library packages import

pandas as pd import warnings

warnings.filterwarnings('ignore')

Load given dataset data =

pd.read_csv("heart.csv") df =

data.dropna()

del df['Age.Group'] del df['Locality'] del df['Gender'] df.columns df.head()

from sklearn.preprocessing import LabelEncoder var_mod = ['Marital_status',

'Sleep', 'Depression', 'Smoking', 'Hypersensitivity'] le = LabelEncoder() for i in

var_mod:

df[i] = le.fit_transform(df[i]).astype(int) df.head()

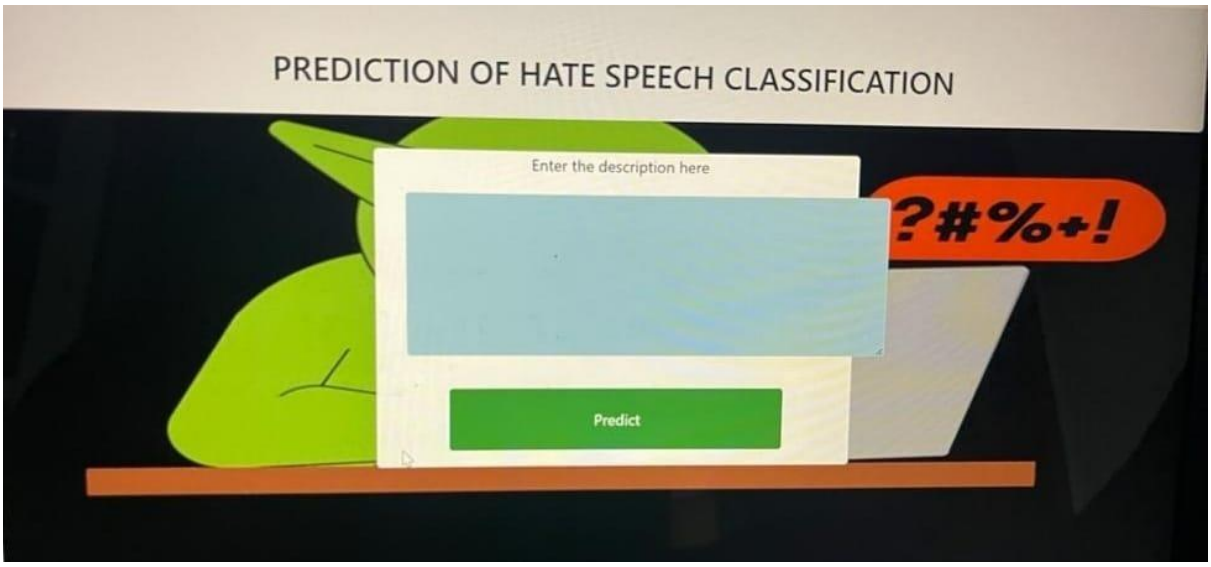
#preprocessing, split test and dataset, split response variable

X = df.drop(labels='Mortality', axis=1)

```
#Response variable y = df.loc[:,'Mortality']
#Splitting for train and test from
sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42, stratify=y) print("Number of training dataset: ",
len(X_train)) print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))
Implementing CatBoostClassifier Algo from catboost import
CatBoostClassifier from sklearn.metrics import confusion_matrix,
classification_report, accuracy_score, plot_confusion_matrix
Training cat = CatBoostClassifier(verbose=0) cat.fit(X_train,y_train)
predicted = cat.predict(X_test) Finding Accuracy accuracy =
accuracy_score(y_test,predicted) print('Accuracy of CatBoost
Classifier',accuracy*100) Finding Clasiification Report cr =
classification_report(y_test,predicted) print('Classification
report\n\n',cr) Finding Confusion matrix cm =
confusion_matrix(y_test,predicted) print('Confusion matrix\n\n',cm)
import matplotlib.pyplot as plt fig, ax = plt.subplots(figsize=(10,10))
plot_confusion_matrix(cat, X_test, y_test, ax=ax) plt.title('Confusion
matrix of CatBoost Classifier') plt.show()
df2 = pd.DataFrame() df2["y_test"] = y_test df2["predicted"] = predicted
df2.reset_index(inplace=True) plt.figure(figsize=(20, 5))
```

```
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100],  marker='o', linestyle='dashed', color='green')
plt.show()
```

Output Sample Screenshot




PREDICTION OF HATE SPEECH CLASSIFICATION
?#%+!
Result for Hate Speech Classification
No Hate and Offensive
back


PREDICTION OF HATE SPEECH CLASSIFICATION
?#%+!
Result for Hate Speech Classification
Offensive Language
Go back

CHAPTER 19

CONCLUSION

19.1 Conclusion

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set of higher accuracy score algorithm will be find out. The founded one is used in the application which can help to find the Hate Speech

CHAPTER 20

FUTURE WORK

20.1 Future Work

➢ Deploying the project in the cloud.

➢ To optimize the work to implement in the IOT system.

➢ Researchers can explore the ethical and social implications of hate speech classification and develop frameworks for responsible AI. This can include developing explainable AI models, ensuring fairness and transparency, and involving diverse stakeholders in the development and implementation of hate speech classification models.