DEMENTIA PREDICTION USING SUPERVISED MACHINE LEARNING

A PROJECT REPORT

Submitted by

DEVANATHAN A	[211419104054]
DHARNISELVAN H	[211419104061]
ARAVIND S	[211419104015]

in partial fulfillment 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

PANIMALAR ENGINEERING COLLEGE

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

BONAFIDE CERTIFICATE

Certified that this project report **"DEMENTIA PREDICTION USING SUPERVISED MACHINE LEARNING"** is the bonafide work of **"DEVANATHAN A (211419104054), DHARNISELVAN H (211419104061), ARAVIND S (211419104015)"** who carried out the project work under my supervision.

SIGNATURE

Dr.L.JABASHEELA,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mr. THYAGARAJAN.C M.E.,(Ph.D),
SUPERVISOR
ASSISTANT PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We **DEVANATHAN A (211419104054), DHARNISELVAN H (211419104061), ARAVIND S (211419104015)** hereby declare that this project report titled **"DEMENTIA PREDICTION USING SUPERVISED MACHINE LEARNING",** under the guidance of **Mr.THYAGARAJAN.C M.E.,(Ph.D.)**, is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

DEVANATHAN A

DHARNISELVAN H

ARAVIND S

ACKNOWLEDGEMENT

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

We express our sincere thanks to our beloved 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 the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.** who facilitated us in completing the project.

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

We would like to thank our parents, friends, project Guide **Mr.THYAGARAJAN.C M.E.,(Ph.D.)**, and coordinator **Dr.N.PUGHAZENDI M.E.,(Ph.D.),** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

DEVANATHAN A

DHARNISELVAN H

ARAVIND S

ABSTRACT

Dementia is a medical disorder in which the blood arteries in the brain are ruptured, causing damage to the brain. When the supply of blood and other nutrients to the brain is interrupted, symptoms might develop. According to the World Health Organization (WHO), Dementia is the greatest cause of death and disability globally. Early recognition of the various warning signs of a Dementia can help to reduce the severity of the Dementia. Different machine learning (ML) models have been developed to predict the likelihood of a Dementia occurring in the brain. The dataset used in the development of the method was the open-access Dementia Prediction dataset. The analysis of the dataset by supervised machine learning technique (SMLT) to capture several information like, variable identification, univariate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparation, and data visualization will be done on the entire given dataset. Additionally, to compare and discuss the performance of various machine learning algorithms from the given hospital dataset with an evaluation classification report, identify the confusion matrix and to categorizing data from priority and the result shows that the effectiveness of the proposed machine learning algorithm technique can be compared with the best accuracy with precision, Recall and F1 Score.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
SMLT	Supervised Machine Learning Technique
UML	Unified Modeling Language
XGB	Extreme Gradient Boosting
Gaussian NB	Gaussian Naïve Bayes
WHO	World Health Organization
GP	Gaussian Process
SES	Socioeconomic Status
MMSE	Mini Mental State Examination Score
CDR	Clinical Dementia Rating
eTIV	Estimated Total Intracranial Volume
nWBV	Normalized Whole Brain Volume
ASF	Atlas Scaling Factor
GUI	Graphical User Interface

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 OVERVIEW

Dementia occurs when the blood flow to various areas of the brain is disrupted or diminished, resulting in the cells in those areas of the brain not receiving the nutrients and oxygen they require and dying. A Dementia is a medical emergency that requires urgent medical attention. Early detection and appropriate management are required to prevent further damage to the affected area of the brain and other complications in other parts of the body. The World Health Organization (WHO) estimates that fifteen million people worldwide suffer from Dementias each year, with one person dying every four to five minutes in the affected population. Dementia is the sixth leading cause of mortality in the United States according to the Centers for Disease Control and Prevention (CDC) [1]. Dementia is a noncommunicable disease that kills approximately 11% of the population. In the United States, approximately 795,000 people suffer from the disabling effects of Dementias on a regular basis [2]. It is India's fourth leading cause of death. Dementias are classified as ischemic or hemorrhagic. In a chemical Dementia, clots obstruct the drainage; in a hemorrhagic Dementia, a weak blood vessel bursts and bleeds into the brain. Dementia may be avoided by leading a healthy and balanced lifestyle that includes abstaining from unhealthy behaviors, such as smoking and drinking, keeping a healthy body mass index (BMI) and an average glucose level, and maintaining an excellent heart and kidney function. Dementia prediction is essential and must be treated promptly to avoid irreversible damage or death. With the development of technology in the medical sector, it is now possible to anticipate the onset of a Dementia by utilizing ML techniques. The algorithms included in ML are beneficial as they allow for accurate prediction and proper analysis. The majority of previous Dementia-related research has focused on, among other things, the prediction of heart attacks. Brain Dementia has been the subject of very few studies. The main motivation of this paper is to demonstrate how ML may be used to forecast the onset of a brain Dementia. The most important aspect of the methods employed and the findings achieved is that among the four distinct classification algorithms tested, Random Forest fared the best, achieving a higher accuracy metric in comparison to the others. One downside of the model is that it is trained on textual data rather than real time brain images. The implementation of four ML classification methods is shown in this paper.

1.2 PROBLEM DEFINITION

Dementia is a progressive neurological disorder that impairs cognitive functions such as memory, communication, and reasoning. Early detection and diagnosis of dementia are critical for providing effective treatment and improving patient outcomes. The use of supervised machine learning algorithms to predict dementia risk based on clinical and demographic data has shown promising results. However, the development of accurate and reliable prediction models requires further investigation and optimization. Therefore, this project aims to explore the feasibility of using supervised machine learning to predict dementia risk and develop an accurate prediction model that can aid in the early detection and management of dementia.

CHAPTER 2

LITERATURE SURVEY

2 LITERATURE SURVEY

2.1 General

In the research conducted by Manisha Sirsat, Eduardo Ferme, Joana Camara, the main aim of the research was to classify state-of-arts on ML techniques for brain Dementia into 4 categories based on their functionalities or similarity, and then review studies of each category systematically. The study further discusses the outcomes and accuracies obtained by using different Machine Learning models using text and image-based datasets. In this study, the authors discussed many Dementia related problems from the state-of-art. The reviewed studies were grouped in several categories based on their similarities. The study notes that it is difficult to compare studies as they employed different performance metrics for different tasks, considering different datasets, techniques, and tuningparameters. Hence, it only mentions the research areas which were targeted in more than one study and the studies which report highest classification accuracy in each section [1]. Harish Kamal, Victor Lopez, Sunil A. Sheth, in their study discus how Machine Learning (ML) through pattern recognition algorithms is currently becoming an essential aid for the diagnosis, treatment, and prediction of complications and patient outcomes in several neurological diseases. The evaluation and treatment of Acute Ischemic Dementia (AIS) haveexperienced a significant advancement over the past few years, increasingly requiring the use of neuroimaging for decisionmaking. This study offers an insight into the recent developments and applications of ML in neuroimaging focusing on acute ischemic Dementia. The implementations of machine learning are numerous, from early identification of imaging diagnostic findings, estimating time of onset, lesion segmentation, and fate of salvageable tissue, to the analysis of cerebral edema, and predicting complications and patient outcomes after treatment.

2.2 Title: Brain Dementia Prediction Using Machine Learning Approach

Author: DR. AMOL K. KADAM , PRIYANKA AGARWAL

Year: 2022

A Dementia is an ailment that causes harm by tearing the veins in the mind. Dementia may likewise happen when there is a stop in the blood stream and different supplements

to the mind. As per the WHO, the World Health Organization, Dementia is one of the world's driving reasons for death and incapacity. The majority of the work has been completed on heart Dementia forecast however not many works show the gamble of a cerebrum Dementia. Subsequently, the AI models are worked to foresee the chance of cerebrum Dementia. The project is pointed towards distinguishing the familiarity with being in danger of Dementia and its determinant factors amongst victims. The research has taken numerous factors and utilized ML calculations like Logistic Regression, Decision Tree Classification, Random Forest Classification, KNN, and SVM for accurate prediction.

2.3 Title: **Brain Dementia Prediction using Machine Learning**

Author: Mrs. Neha Saxena, Mr. Deep Singh Bhamra, Mr. Arvind Choudhary

Year: 2019

A Dementia, also known as a cerebrovascular accident or CVA is when part of the brain loses its blood supply and the part of the body that the blood-deprived brain cells control stops working. This loss of blood supply can be ischemic because of lack of blood flow, or haemorrhagic because of bleeding into brain tissue. A Dementia is a medical emergency because Dementias can lead to death or permanent disability. There are opportunities to treat ischemic Dementias but that treatment needs to be started in the first few hours after the signs of a Dementia begin. The patient, family, or bystanders should activate emergency medical services immediately should a Dementia be suspected. A transient ischemic attack (TIA or mini-Dementia) describes an ischemic Dementia that is short-lived where the symptoms resolve spontaneously. This situation also requires emergency assessment to try to minimize the risk of a future Dementia. By definition, a Dementia would be classified as a TIA if all symptoms resolved within 24 hours. According to the World Health Organization (WHO) Dementia is the 2nd leading cause of death globally, responsible to approximately 11% of total deaths. For survival prediction, our ML model uses dataset to predict whether a patient is likely to get Dementia based on the input parameters like gender, age, various diseases, and smoking status. Unlike most of the datasets, our dataset focuses on attributes that would have a major risk factors of a Brain Dementia.

2.4 Title: Machine Learning for Brain Dementia: A Review

Author: Manisha Sanjay Sirsat, Eduardo Ferme , and Joana Camara,

Year : 2020

Machine Learning (ML) delivers an accurate and quick prediction outcome and it has become a powerful tool in health settings, offering personalized clinical care for Dementia patients. An application of ML and Deep Learning in health care is growing however, some research areas do not catch enough attention for scientific investigation though there is real need of research. Therefore, the aim of this work is to classify state-of-arts on ML techniques for brain Dementia into 4 categories based on their functionalities or similarity, and then review studies of each category systematically. A total of 39 studies were identified from the results of ScienceDirect web scientific database on ML for brain Dementia from the year 2007 to 2019. Support Vector Machine (SVM) is obtained as optimal models in 10 studies for Dementia problems. Besides, maximum studies are found in Dementia diagnosis although number for Dementia treatment is least thus, it identifies a research gap for further investigation. Similarly, CT images are a frequently used dataset in Dementia. Finally SVM and Random Forests are efficient techniques used under each category. The present study showcases the contribution of various ML approaches applied to brain Dementia

2.5 Title: Predicting Dementia Risk With an Interpretable Classifier

Author: Sergio Penafiel, Nelson Baloian, Horacio Sanson, And Jose A. Pino.

Year : 2021

Predicting an individual's risk of getting a Dementia has been a research subject for many authors worldwide since it is a frequent illness and there is strong evidence that early awareness of having that risk can be beneficial for prevention and treatment. Many Governments have been collecting medical data about their own population with the purpose of using artificial intelligence methods for making those predictions. The most accurate ones are based on so called black-box methods which give little or no information about why they make a certain prediction. However, in the medical field the explanations are sometimes more important than the accuracy since they allow specialists

to gain insight about the factors that influence the risk level. It is also frequent to find medical information records with some missing data. In this work, we present the development of a prediction method which not only outperforms some other existing ones but it also gives information about the most probable causes of a high Dementia risk and can deal with incomplete data records. It is based on the Dempster-Shafer theory of plausibility. For the testing we used data provided by the regional hospital in Okayama, Japan, a country in which people are compelled to undergo annual health checkups by law. This article presents experiments comparing the results of the Dempster-Shafer method with the ones obtained using other well-known machine learning methods like Multilayer perceptron, Support Vector Machines and Naive Bayes. Our approach performed the best in these experiments with some missing data. It also presents an analysis of the interpretation of rules produced by the method for doing the classification. The rules were validated by both medical literature and human specialists.

2.6 Title: Towards Dementia prediction using electronic health records
Author: Douglas Teoh

Year: 2020

Background: As of 2014, Dementia is the fourth leading cause of death in Japan. Predicting a future diagnosis of Dementia would better enable proactive forms of healthcare measures to be taken. We aim to predict a diagnosis of Dementia within one year of the patient's last set of exam results or medical diagnoses. Methods: Around 8000 electronic health records were provided by Tsuyama Jifukai Tsuyama Chuo Hospital in Japan. These records contained non-homogeneous temporal data which were first transformed into a form usable by an algorithm. The transformed data were used as input into several neural network architectures designed to evaluate efficacy of the supplied data and also the networks' capability at exploiting relationships that could underlie the data. The prevalence of Dementia cases resulted in imbalanced class outputs which resulted in trained neural network models being biased towards negative predictions. To address this issue, we designed and incorporated regularization terms into the standard cross-entropy loss function. These terms penalized false positive and false negative predictions. We evaluated the performance of our trained models using Receiver Operating Characteristic. Results: The best neural network incorporated and combined

the different sources of temporal data through a dual-input topology. This network attained area under the Receiver Operating Characteristic curve of 0.669. The custom regularization terms had a positive effect on the training process when compared against the standard cross-entropy loss function

CHAPTER 3

SYSTEM ANALYSIS

3 SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Dementia has become a leading cause of death and long-term disability in the world with no effective treatment. Deep learning-based approaches have the potential to outperform existing Dementia risk prediction models, but they rely on large well-labeled data. Due to the strict privacy protection policy in health-care systems, Dementia data is usually distributed among different hospitals in small pieces. In addition, the positive and negative instances of such data are extremely imbalanced. Transfer learning can solve small data issue by exploiting the knowledge of a correlated domain, especially when multiple source of data are available. This work has addressed the issues of SRP with small and imbalanced Dementia data. We have proposed a novel Hybrid Deep Transfer Learning-based Dementia Risk Prediction (HDTL-SRP) framework which consists of three key components: Generative Instance Transfer (GIT) for making use of the external Dementia data distribution among multiple hospitals while preserving the privacy, Network Weight Transfer (NWT) for making use of data from highly correlated diseases (i.e., hypertension or diabetes), Active Instance Transfer (AIT) for balancing the Dementia data with the most informative generated instances. It is found that the proposed HDTL-SRP framework outperforms the state-of-the-art SRP models in both synthetic and real-world scenarios.

3.2 DISADVANTAGES OF EXISTING SYSTEM

- Machine learning is not implemented properly.
- Accuracy is less and low performance.
- They are not using any model deployment process.
- They did not find the output properly.
- They did not find the patient’s food recommendations.

3.3 PROPOSED SYSTEM

The proposed method is to build a machine learning model for the classification of brain Dementia. The process carries from data collection where past data related to brain Dementia are collected. Data mining is a commonly used technique for processing

enormous data in the healthcare domain. A brain Dementia if found before proper treatment can save lives. Machine learning is now applied and mostly used in health care where it reduces the manual effort and a better model makes error less which leads to saving the life. The data analysis is done on the dataset proper variable identification is done that is both the dependent variables and independent variables are found. The proper machine learning algorithms are applied to the dataset where the pattern of data is learned. After applying different algorithms a better algorithm is used for the prediction of the outcome.

3.4 ADVANTAGES OF PROPOSED SYSTEM

- Machine learning will be implemented perfectly.
- More algorithms will be compared.
- Accuracy will be predicted perfectly.
- Deployment is done for getting results.
- The prediction will be better.

3.5 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.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data cannot be used directly. It's used to preprocess 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.

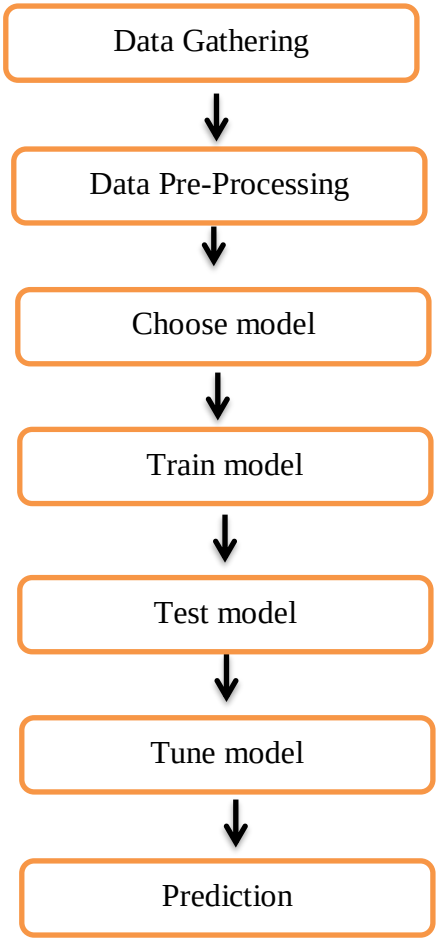

Fig 3.5.1 Process of dataflow diagram

List Of Modules:

- Data pre-processing (Module-01)
- Data Analysis of Visualization (Module-02)
- Comparing Algorithm with prediction in the form of best accuracy result (Module-03 to 06)
- Deployment Using Result (Module-07)

3.6 HARDWARE REQUIREMENT

- Hard Disk : 500GB and Above
- RAM : 4GB and Above
- Processor : I3 and Above

3.7 SOFTWARE REQUIREMENT

- Operating System : Windows 10 (64 bit)
- Software : Python
- Tools : Anaconda

3.8 TECHNOLOGIES USED

- Python
- Machine Learning

3.8.1 Python

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, data science, artificial intelligence, scientific computing, and more. It was first released in 1991 and has since become one of the most popular programming languages in the world. Some key features of Python include:

- Easy to Learn: Python has a simple and easy-to-learn syntax, which makes it an ideal language for beginners.
- Interpreted Language: Python is an interpreted language, which means that the code is executed line by line, making it easier to test and debug.
- Cross-Platform: Python can be run on various platforms, including Windows, macOS, and Linux.
- Large Standard Library: Python has a large standard library that provides a wide range of built-in modules for various tasks, such as file I/O, regular expressions, networking, and more.
- Open Source: Python is open-source software, which means that the source code is freely available to anyone and can be modified and redistributed.

- Object-Oriented: Python is an object-oriented language, which means that it supports object-oriented programming concepts such as encapsulation, inheritance, and polymorphism.

Python has a vast community of developers and users, which has contributed to its popularity and growth. It has a large number of third-party libraries and frameworks that make it easier to develop complex applications quickly. Some popular Python frameworks include Django, Flask, and Pyramid for web development, and NumPy, Pandas, and SciPy for scientific computing and data analysis. In summary, Python's simplicity, ease of use, and versatility have made it a popular programming language in various industries, from web development to scientific computing and artificial intelligence.

3.8.2 MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or

categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). 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 non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

Fig 3.8.2.1 Process of Machine Learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is y = f(X). The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

CHAPTER 4

SYSTEM DESIGN

4 SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM (DFD)

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM). Superficially, DFDs can resemble flow charts or Unified Modeling Language (UML), but they are not meant to represent details of software logic. DFDs make it easy to depict the business requirements of applications by representing the sequence of processsteps and flow of information using a graphical representation or visual representation rather than a textual description.

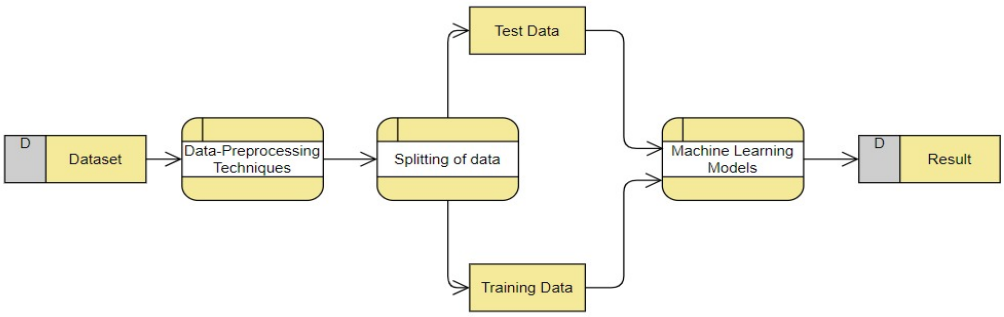

Fig 4.1.1 Level 0 of Data Flow Diagram

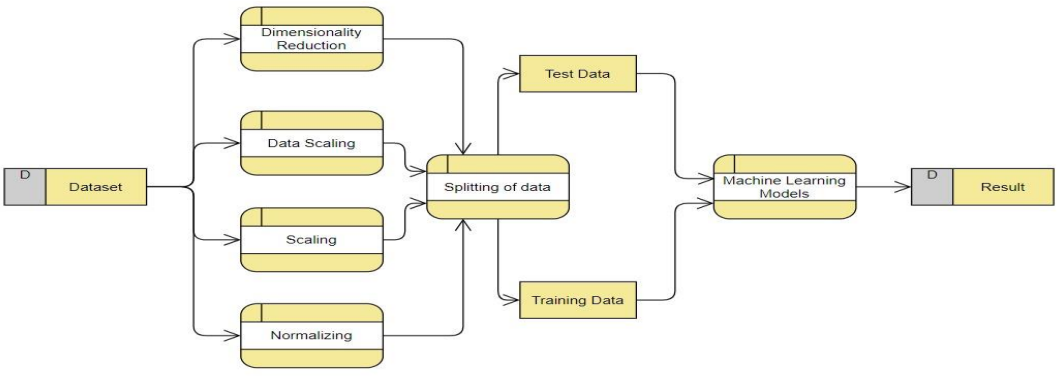

Fig 4.1.2 Level 1 of Data Flow Diagram

4.2 Entity Relationship Diagram (ERD)

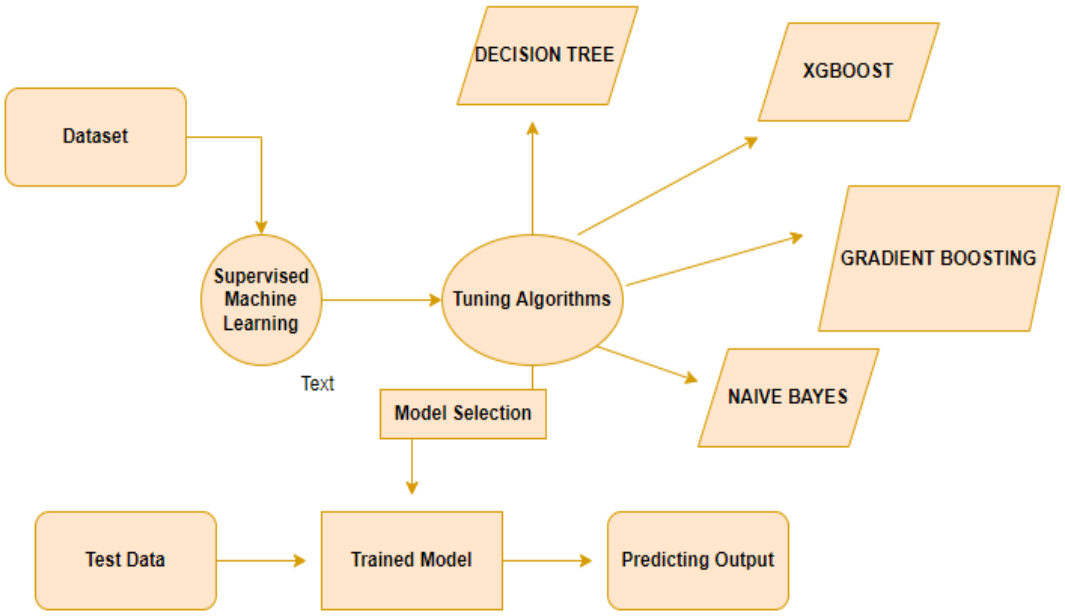

Fig 4.2.1 Entity Relationship Diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

4.3 UML DIAGRAMS

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.OMG is continuously making efforts to create a truly industry standard. UML stands for Unified Modeling Language. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modeling language to

visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well.

For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard.

4.3.1 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.

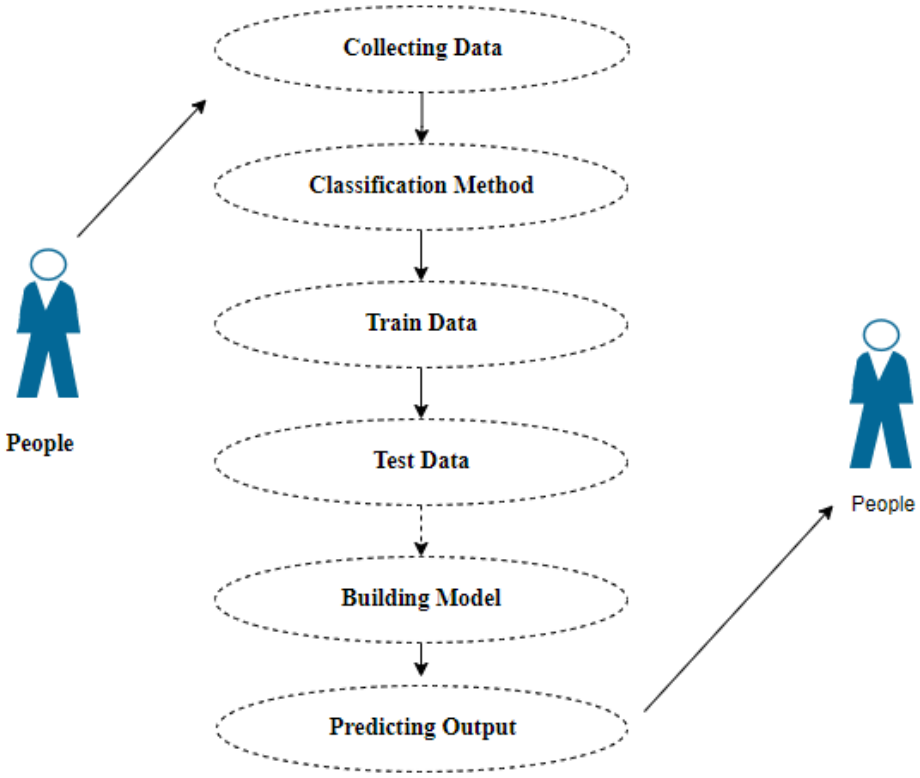

Fig 4.3.1 Use Case Diagram

4.3.2 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. 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.

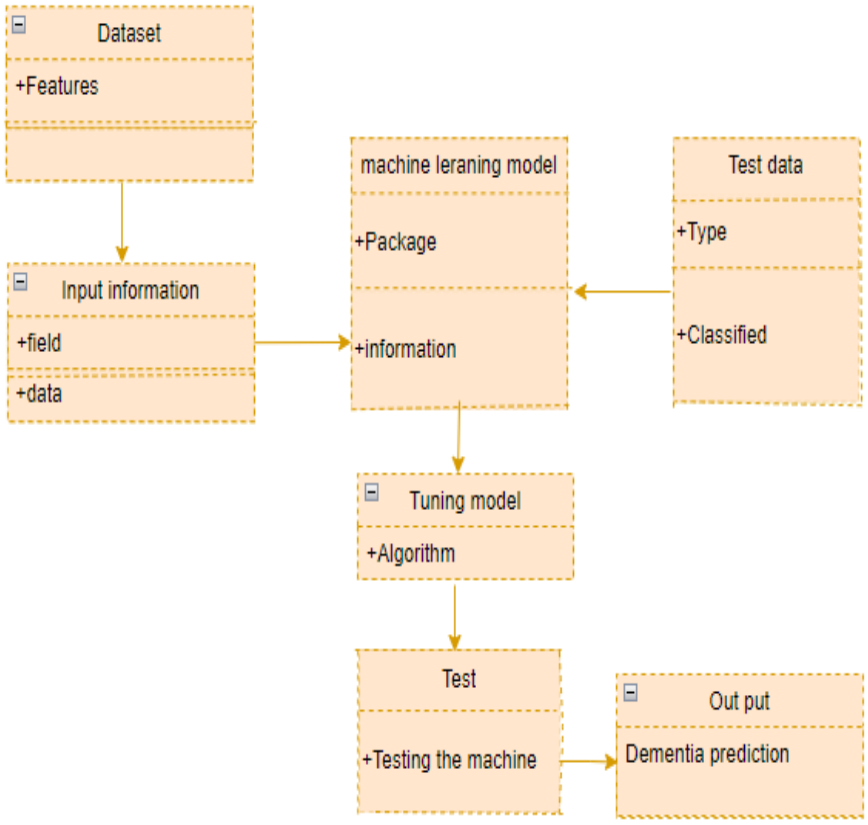

Fig 4.3.2 Class Diagram

4.3.3 Activity Diagram:

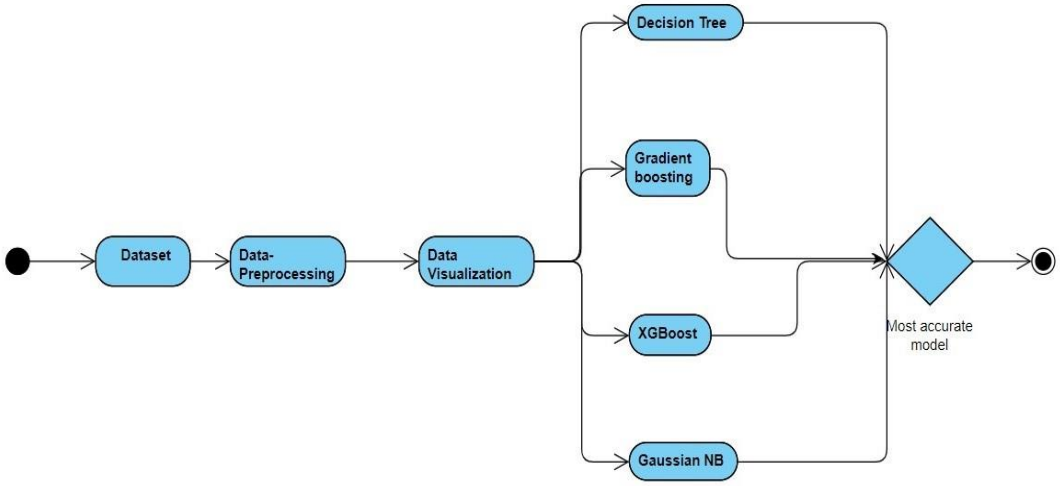

Fig 4.3.3 Activity Diagram

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. It shows different flow like parallel, branched, concurrent and single.

4.3.4 Sequence Diagram:

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. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming.

Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

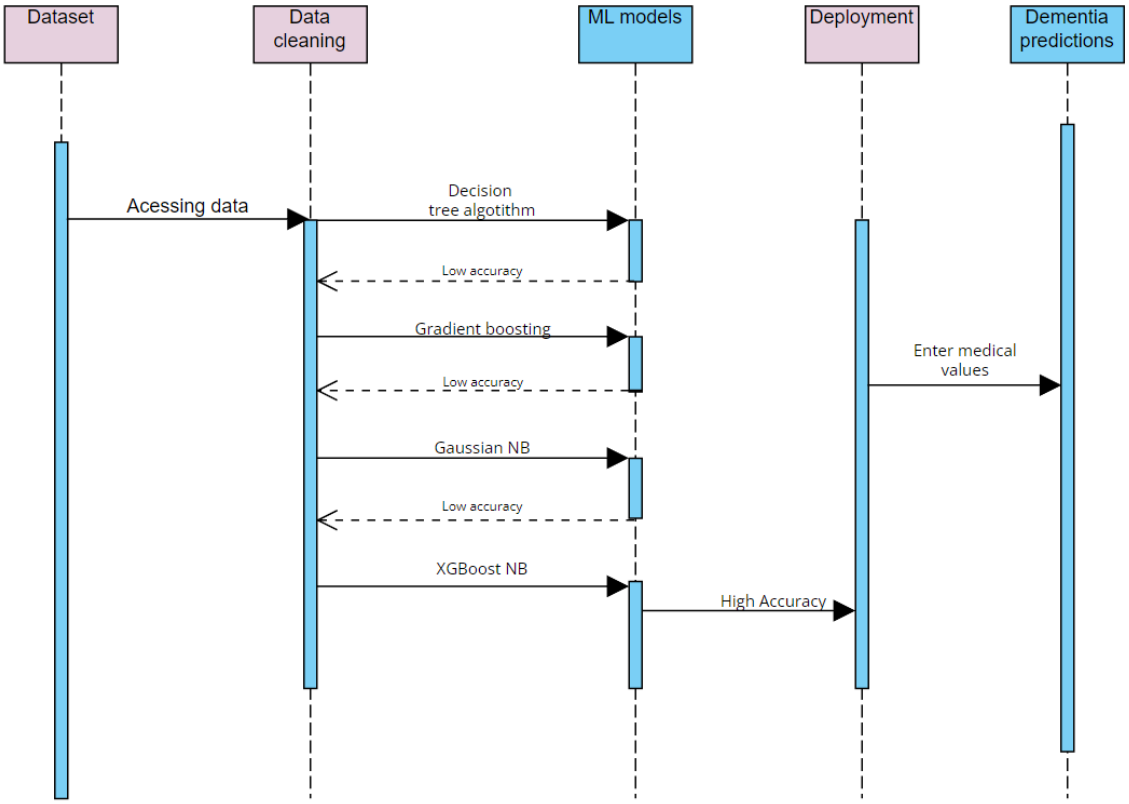

Fig. 4.3.4 Sequence Diagram

CHAPTER 5

SYSTEM ARCHITECTURE

5 SYSTEM ARCHITECTURE

5.1 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts that are part of an architecture, including their principles, elements and components. It is also defined as a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element. Also, an architecture diagram is a network map used to describe the general structure of a software program as well as the interactions, restrictions, and limits between elements. It is a significant tool since it offers a broader picture of the computer underlying physical installation as well as its development plan.

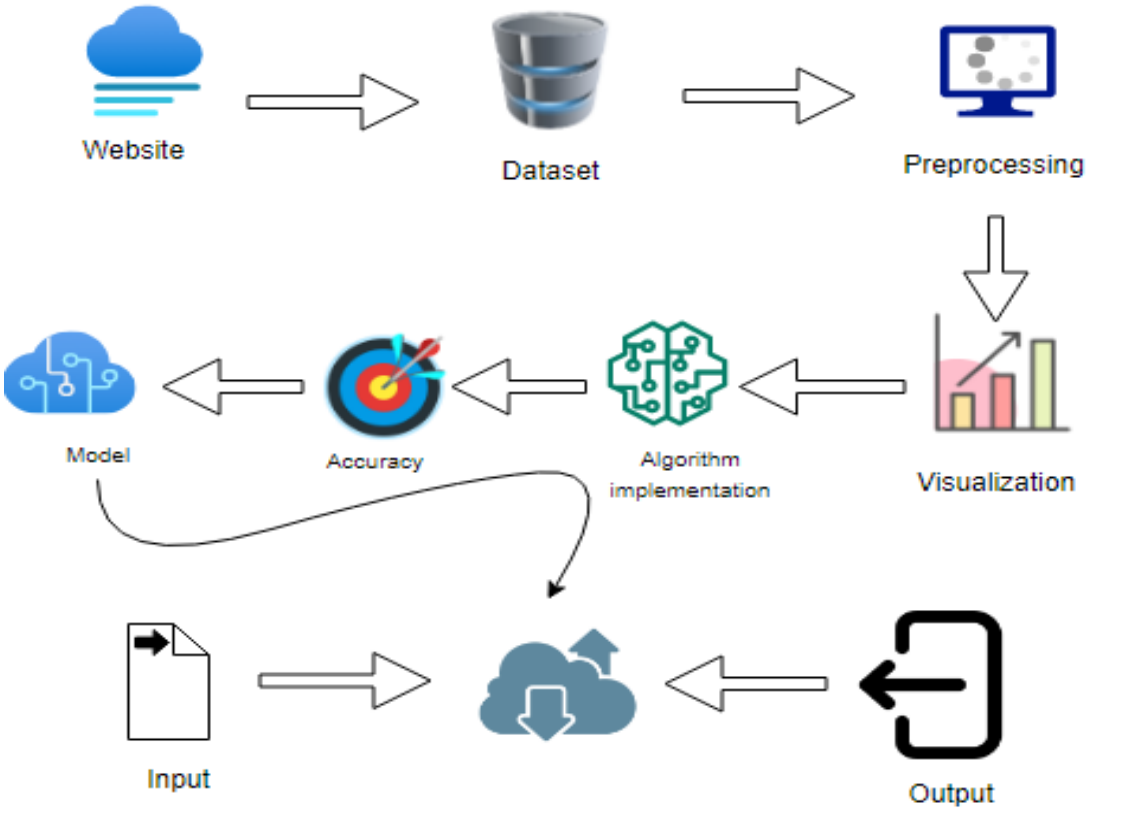

Fig. 5.1 System Architecture

This graphic provides a concise and understandable description of all the entities currently integrated into the system. The diagram shows how the many actions and choices are linked together. You might say that the whole process and how it was carried out is a picture. The figure below shows the functional connections between various entities.

5.2 ALGORITHMS

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 non-spam) 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 labeled 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 unlabeled new data

5.2.1 DECISION TREE CLASSIFIER

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees. Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of

the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returns the classification associated with the particular leaf.

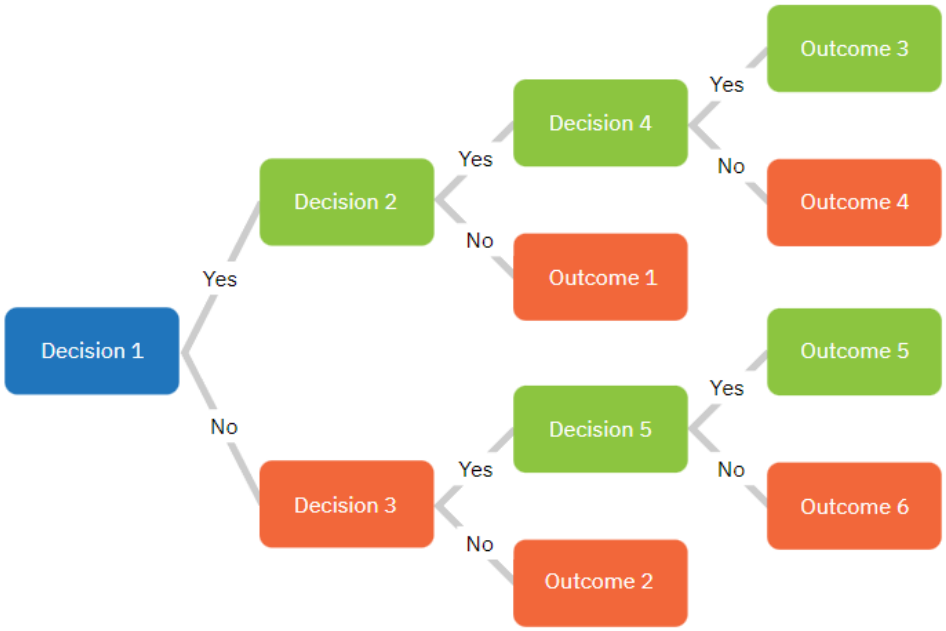

Fig. 5.2.1 Decision Tree Classifier

5.2.2 GRADIENT BOOSTING CLASSIFIER

Gradient boosting is a method standing out for its prediction speed and accuracy, particularly with large and complex datasets. From Kaggle competitions to machine learning solutions for business, this algorithm has produced the best results. We already know that errors play a major role in any machine learning algorithm. There are mainly two types of error, bias error and variance error. Gradient boost algorithm helps us minimize bias error of the model Before getting into the details of this algorithm we must have some knowledge about AdaBoost Algorithm which is again a boosting method. This algorithm starts by building a decision stump and then assigning equal weights to all the

data points. Then it increases the weights for all the points which are misclassified and lowers the weight for those that are easy to classify or are correctly classified. A new decision stump is made for these weighted data points. The idea behind this is to improve the predictions made by the first stump. I have talked more about this algorithm here. The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model. But how do we do that? How do we reduce the error? This is done by building a new model on the errors or residuals of the previous model.When the target column is continuous, we use Gradient Boosting Regressor whereas when it is a classification problem, we use Gradient Boosting Classifier.

Since it is based on loss function hence for regression problems, we'll have different loss functions like Mean squared error (MSE) and for classification, we will have different for e.g log-likelihood. A gradient boosting classifier is used when the target column is binary. All the steps explained in the Gradient boosting regressor are used here, the only difference is we change the loss function. Earlier we used Mean squared error when the target column was continuous but this time, we will use log-likelihood as our loss function.

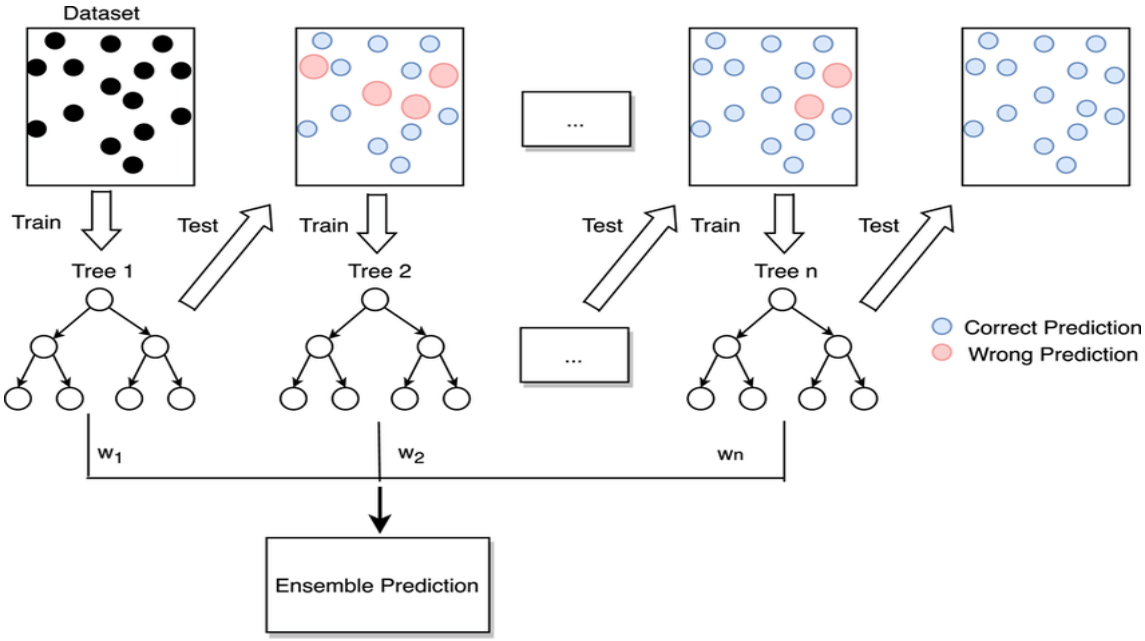

Fig. 5.2.2 Gradient Boosting Classifier

5.2.3 XGBOOST CLASSIFIER

The XGBoost (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models. The XGBoost algorithm performs well in machine learning competitions because of its robust handling of a variety of data types, relationships, distributions, and the variety of hyperparameters that you can fine-tune. You can use XGBoost for regression, classification (binary and multiclass), and ranking problems. You can use the new release of the XGBoost algorithm either as a Amazon SageMaker built-in algorithm or as a framework to run training scripts in your local environments. This implementation has a smaller memory footprint, better logging, improved hyperparameter validation, and an expanded set of metrics than the original versions. It provides an XGBoost estimator that executes a training script in a managed XGBoost environment. The current release of SageMaker XGBoost is based on the original XGBoost versions 1.0, 1.2, 1.3, and 1.5.

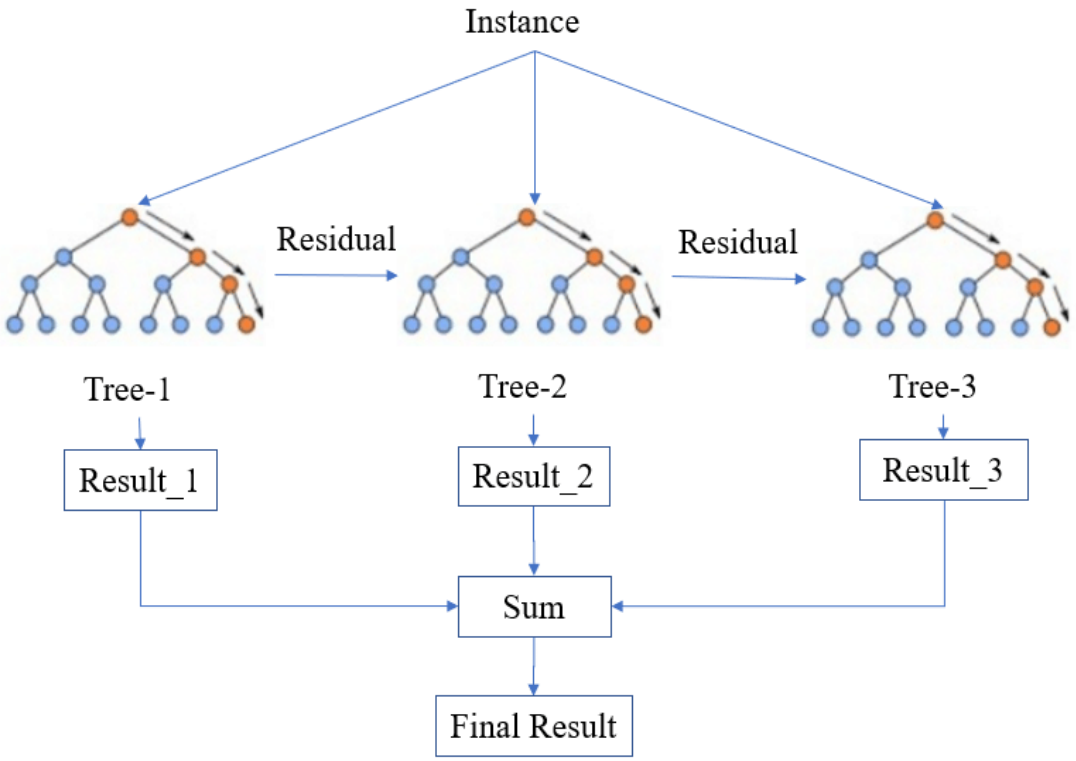

Fig. 5.2.3 Xgboost Classifier

5.2.4 GAUSSIAN NB

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically. Naivebayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method. The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities togeth er for each attribute for a given class value, we have a probability of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. NaiveBayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets. Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

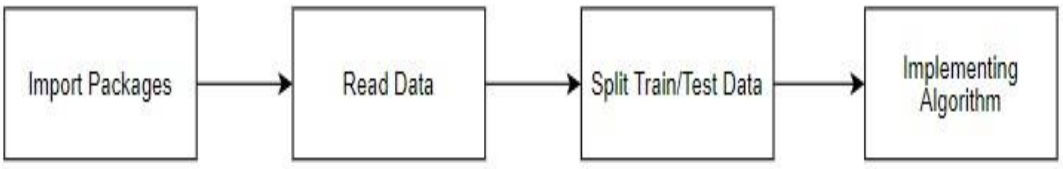

Fig. 5.2.4 Gaussian NB

CHAPTER 6

SYSTEM IMPLEMENTATION

6 SYSTEM IMPLEMENTATION

6.1 DATA PRE-PROCESSING

Loading the supplied dataset and importing library packages to investigate variable's identity by data type and shape and to assess duplicated and values that are missing. while fine-tuning models and using strategies to make the most of validate and test data while analyzing your models, you can use validation datasets, which are samples of data excluded from learning your model and used to gauge model competence. The provided dataset must be renamed, columns must be deleted, and other data cleaning and preparation steps must be completed in order to analyze the uni-variate, bi-variate, and multi-variate processes depending on the dataset, several procedures and techniques will be used to clean the data. To boost the worth of datasets for analysis and decision-making, data cleaning aims to find and remove mistakes and anomalies.

Dataset:

The Dementia Prediction Dataset comprises of a total of 372 rows data of data with 13 columns and had attributes such as "Male/Female", "Hand", "Age" , "EDUC" , "SES" ,"MMSE" , "CDR" , "MR Delay" , "Visit" , "nWBV" , "eTIV" , "ASF" .

```
In [2]:  1 df=p.read_csv('A.csv')
         2 df.head()
```

Out[2]:

	ASF	eTIV	nWBV	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE	CDR	Group
0	0.883	1987	0.696	1	0	M	R	87	14	2.0	27.0	0.0	Nondemented
1	0.876	2004	0.681	2	457	M	R	88	14	2.0	30.0	0.0	Nondemented
2	1.046	1678	0.736	1	0	M	R	75	12	NaN	23.0	0.5	Demented
3	1.010	1738	0.713	2	560	M	R	76	12	NaN	28.0	0.5	Demented
4	1.034	1698	0.701	3	1895	M	R	80	12	NaN	22.0	0.5	Demented

```
In [3]:  1 df.tail()
```

Out[3]:

	ASF	eTIV	nWBV	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE	CDR	Group
368	1.037	1693	0.694	2	842	M	R	82	16	1.0	28.0	0.5	Demented
369	1.040	1688	0.675	3	2297	M	R	86	16	1.0	26.0	0.5	Demented
370	1.331	1319	0.801	1	0	F	R	61	13	2.0	30.0	0.0	Nondemented
371	1.323	1327	0.796	2	763	F	R	63	13	2.0	30.0	0.0	Nondemented
372	1.317	1333	0.801	3	1608	F	R	65	13	2.0	30.0	0.0	Nondemented

Fig. 6.1.1Dataset

6.2 Data visualization

Data visualizations can be used to convey and demonstrate important relationships using plots and graphs that are more visceral and attractive to stakeholders than measures of association or significance. Unless data is graphically represented, such as via charts and graphs, it might not always make sense. Fast visualization of samples of data and other objects is valuable in both statistical applications and applied machine learning. It will demonstrate how to use different plot types to examine your own information as well as the wide variety of plot types you'll encounter while illustrating information using Python.

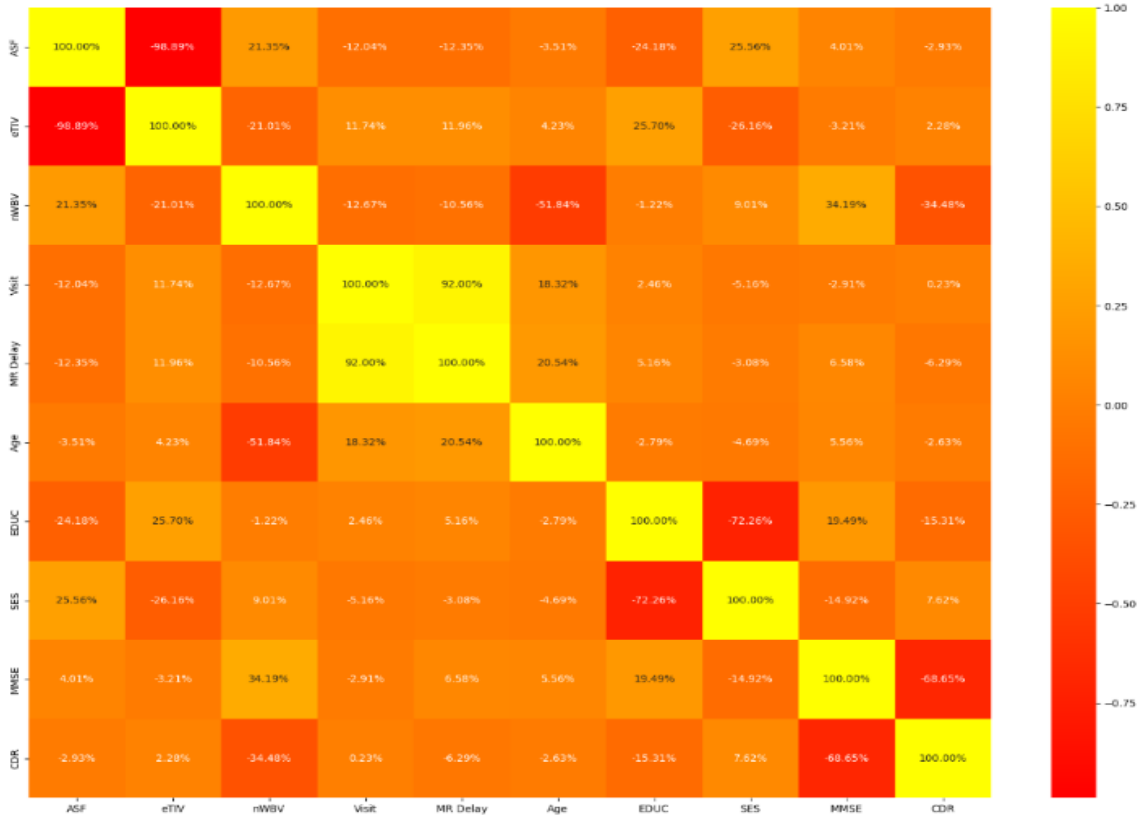

Fig.6.2.1 Data visualization

6.3 Training ML Models

To ensure the system produces the best results and is as accurate as possible, 4 distinct machine learning algorithms have been used in its implementation.

- Decision tree
- Gradient Boosting Classifier

- XGB classifier
- Gaussian NB

The libraries which were used in training these models are

1) Pandas

2) Numpy

3) Matplotlib

4) SKlearn

6.4 DEPLOYMENT

After training all the model a live website is created which predicts the possibilities of dementia. For this the machine learning model which gave high accuracy is used by accessing its as a model file in our python program. The tools used in this deployment model are

1. Front End:

Bootstrap, CSS (Cascading Style Sheets), HTML (Hyper Text Markup Language).

2. Framework:

Flask: Python API for web-applications.

3. Runtime Environment:

Jupyter notebook

6.5 TEST RESULTS

The last stage will be to use our web application to deliver exact and accurate results to the user, allowing them to proceed as needed in accordance with the results. The system will finally produce the desired result after the approach, modules, algorithms, and codes have been implemented. The homepage will assist users in entering the information needed for the Dementia prediction system, and the GUI portion is designed to be user-friendly for regular people. The system has been developed

utilizing four distinct ML algorithms, as specified in the Implementation, to obtain the best accuracy while using the dataset.

The results generated was:

- The lowest accuracy was given by Decision tree classifier i.e. (86.91%)
- The Highest accuracy was given by XGB classifier i.e. (94.36%)

The following table shows the results of the evaluation parameters against the models. This was found using the optimal dataset split, which was 70% for training and 30% for testing.

Evaluation parameters	Accuracy Score	Recall Score	Precision Score	F1 Score
Decision Tree	86.91%	87%	86%	86%
Gradient Boosting	92.95%	92%	93%	92%
XGB Classifier	94.36%	94%	95%	93%
Gaussian NB	91.67%	93%	92%	92%

Table. 3.5.1 Accuracy Table

As shown in Figure 4, the suggested system helped us analyse the optimum manner to receive user inputs for the GUI implementation portion of our project. The XGB Classification, which has been trained with the datasets using the information supplied to the GUI for risk of stroke prediction, was employed, and the new data supplied by the user was checked against by the training sample.

CHAPTER 7

SYSTEM TESTING

7. SYSTEM TESTING

7.1 BLACK BOX TESTING

Black box testing is a software testing technique that focuses on testing the functionalityof a software system without knowing the internal workings of the system. In this type of testing, the tester is not concerned with the code, architecture, or design of the system but instead focuses on the inputs and outputs of the system. The goal of black box testing is to determine whether the software system behaves correctly based on its specifications, requirements, and business logic. The tester typically creates test cases that simulate different scenarios and inputs to verify the expected results. Black box testing can be performed at different levels of software testing, including unit testing, integration testing, system testing, and acceptance testing. It can be automated or performed manually. The advantages of black box testing include the ability to test the software system from the end user's perspective, the ability to detect defects that may be missed in other types of testing, and the fact that it does not require knowledge of the internal workings of the system. The disadvantages of black box testing include the possibility of incomplete testing due to the limited knowledge of the system, and the fact that it may be difficult to determine the root cause of defects.

7.2 WHITE BOX TESTING

White box testing is a software testing technique that focuses on testing the internalworkings of a software system. In this type of testing, the tester has access to the source code, architecture, and design of the system and uses this knowledge to create test cases that verifythe correctness and quality of the system's implementation. The goal of white box testing is toensure that the code is written correctly, follows best practices and coding standards, and meetsthe design specifications. The tester typically creates test cases that target specific areas of the code, such as loops, conditionals, and error handling, to ensure that all possible scenarios have been tested. White box testing can be performed at different levels of software testing, including unit testing, integration testing, and system testing. It can be automated or performed manually. The advantages of white box

testing include the ability to test the system thoroughly and ensure that all code paths have been exercised, the ability to detect defects that may be missed in other types of testing, and the ability to optimize the code forperformance and efficiency.

7.3 TEST CASES

TEST REPORT: 01

PRODUCT : Give required inputs for prediction

USE CASE : Values

TESTCASE ID	**TESTCASE / ACTION TO BE PERFORMED**	**EXPECTED RESULT**	**ACTUAL RESULT**	**PASS/ FAIL**
1	Give Input Values	Dementia is demented	Dementia is demented	PASS
2	Give Input Values	Dementia is Non-demented	Dementia is Non-demented	PASS
3	Give Input Values	Dementia is converted	Dementia is converted	PASS

Table 7.3.1 Test Case for Dimentia Predictions

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENT

8. CONCLUSION & FUTURE ENHANCEMENT

8.1 CONCLUSION

In conclusion, the use of various machine learning techniques, such as decision tree, XGBoost classifier, Naive Bayes, and gradient boosting, in predicting dementia has shown promising results. By analyzing various factors such as demographic information, lifestyle, medical history, and cognitive assessments, machine learning models can accurately predict the risk of developing dementia in individuals. The project has demonstrated that different methods have their own strengths and limitations in predicting dementia. For example, decision trees are easy to interpret and visualize but may suffer from overfitting. XGBoost classifier, on the other hand, is more complex and has a higher computational cost but can provide better accuracy. Naive Bayes is a simple yet effective probabilistic classifier, while gradient boosting can handle large and complex datasets by combining weak learners into a strong one. Regardless of the methods used, it is important to note that machine learning models should not replace clinical diagnosis or medical expertise. These models are meant to be a supplementary tool that can aid in decision-making and improve patient outcomes. Additionally, there are ethical considerations surrounding the use of personal data for prediction models, and it is important to prioritize privacy and consent in the development and implementation of these models. Overall, the application of machine learning in dementia prediction has great potential to improve the lives of individuals at risk of developing this debilitating condition. Further research and development are necessary to refine and improve these models, ensuring their accuracy and usefulness in clinical practice.

After the literature survey, we came to know various pros and cons of different research papers and thus, proposed a system that helps to predict brain Dementias in a cost effective and efficient way by taking few inputs from the user side and predicting accurate results with the help of trained Machine Learning algorithms. Thus, the Brain Dementia Prediction system has been implemented using the given Machine Learning algorithm given a Best accuracy.The system is therefore designed providing simple yet efficient User Interface design with an empathetic approach towards their users and patients.

8.2 FUTURE ENHANCEMENT

The accuracy of a machine learning model depends heavily on the quality and quantity of data used to train it. In order to improve the performance of a dementia prediction model, you could collect more data from a larger and more diverse population, including individuals from different ethnic groups, socioeconomic backgrounds, and age ranges. Recent research has identified several genetic markers that may be associated with an increased risk of developing dementia. Adding genetic data to a machine learning model could improve its accuracy and help identify individuals who are at higher risk for developing the disease. In addition to demographic and medical data, there are several other sources of data that could be incorporated into a dementia prediction model, such as lifestyle factors, social support, and cognitive performance measures. For healthcare providers, patients, and caregivers to effectively use a dementia prediction model, it needs to have a user-friendly interface. This could involve creating an easy-to-use web or mobile application that provides personalized recommendations based on the patient's risk profile. Integrating the dementia prediction model with electronic health records could allow healthcare providers to automatically screen patients for dementia risk during routine check-ups and provide targeted interventions when necessary.

APPENDICES

APPENDICES

A.1 CODING:

MODULE – 1:

```
import pandas as p
import numpy as n

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
df=p.read_csv('A.csv')
df.head()
```

In [ ]:

```
df.tail()
```

In [ ]:

```
df.shape
```

In [ ]:

```
df.size
```

In [ ]:

```
df.columns
```

In [ ]:

```
df.ndim
```

In [ ]:

```
df.isnull()
```

In [ ]:

```
df['Group'].unique()
```

In [ ]:

```
df = df.dropna()
```

In [ ]:

```
df.describe()
```

In [ ]:

```
df.corr()
```

In [ ]:

```
df.info()
```

In [ ]:

```
p.crosstab(df["ASF"], df["eTIV"])
```

In [ ]:

```
df.groupby(["nWBV","Visit"]).groups
```

In [ ]:

```
df["Group"].value_counts()
```

In [ ]:

```
p.Categorical(df["EDUC"]).describe()
```

In [ ]:

```
df.duplicated()
```

In [ ]:

```
sum(df.duplicated())
```

MODULE – 2:

```
import pandas as pd
import numpy as n
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
d = pd.read_csv('A.csv')
d.head()
```

In [ ]:

```
d.columns
```

In [ ]:

```
plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
plt.hist(d['ASF'],color='red')

plt.subplot(1,2,2)
plt.hist(d['eTIV'],color='blue')
```

In [ ]:

```
d.hist(figsize=(15,55),layout=(15,4), color='red')
plt.show()
```

In [ ]:

```
d['nWBV'].hist(figsize=(10,5),color='red')
```

In [ ]:

```
plt.bar(d['Visit'],d['Age'], color='red') # scatter, plot, triplot, stackplot
```

In [ ]:

```
plt.boxplot(d['MMSE'])
```

In [ ]:

```
d['SES'].plot(kind='density')
```

In [ ]:

```
sns.displot(d['Group'], color='red')
```

 # barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot, histplot, kdeplot, pointplot, violinplot, stripplot

In [ ]:

```
sns.residplot(d['MMSE'],d['CDR'], color='red') # residplot, scatterplot
```

In [ ]:

```
sns.pairplot(d)
```

In [ ]:

```
fig, ax = plt.subplots(figsize=(20,15))
sns.heatmap(d.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)
```

In [ ]:

```
def plot(d, variable):
    dataframe_pie = d[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return n.round(dataframe_pie/d.shape[0]*100,2)

plot(d, 'Group')
```

MODULE – 3:

DecisionTreeClassifier

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=pd.read_csv('A.csv')
data.head()
```

In [ ]:

```
df=data.dropna()
df
```

In [ ]:

```
df.columns
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['Group','M/F','Hand']

for i in var:
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
x = df.drop(labels='Group', axis=1)
y = df.loc[:,'Group']
```

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=1,
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))
```
In [ ]:

```
x_train
```
In [ ]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.tree import  DecisionTreeClassifier
```
In [ ]:

```
DTC = DecisionTreeClassifier()
DTC.fit(x_train,y_train)
predicted = DTC.predict(x_test)
```
In [ ]:

```
cr = classification_report(y_test,predicted)
print('Classification report of DecisionTreeClassifier Result is:\n',cr)
print("\n")
```
In [ ]:

```
cm = confusion_matrix(y_test,predicted)
print('Confusion Matrix result of DecisionTreeClassifier is:',cm)
print("\n")
```
In [ ]:

```
accuracy = cross_val_score(DTC, x, y, scoring='accuracy')
print('Cross validation test results of accuracy:', accuracy)
print("\n")
```
In [ ]:

```
a = accuracy.mean() * 100
print("Accuracy Result of DecisionTreeClassifier is:",a)
```
In [ ]:

```
def plot_confusion_matrix(cm, title='Confusion matrix-DecisionTreeClassifier',
cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predicted)
print('Confusion matrix-DecisionTreeClassifier:')
print(cm)
plot_confusion_matrix(cm)
```
In [ ]:

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

import joblib
joblib.dump(DTC, 'model.pkl')
```

MODULE – 4:

GradientBoostingClassifier

```
In [ ]:

import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
In [ ]:

data=pd.read_csv('A.csv')
data.head()
In [ ]:

df=data.dropna()
df
In [ ]:

df.columns
In [ ]:

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['Group','M/F','Hand']

for i in var:
    df[i] = le.fit_transform(df[i]).astype(int)
In [ ]:

x = df.drop(labels='Group', axis=1)
y = df.loc[:,'Group']
In [ ]:
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=1, 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))
In [ ]:

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingClassifier
In [ ]:

LR = GradientBoostingClassifier()
LR.fit(x_train,y_train)
predicted = LR.predict(x_test)
In [ ]:

cr = classification_report(y_test,predicted)
print('Classification report of LogisticRegression Result is:\n',cr)
print("\n")
In [ ]:

cm = confusion_matrix(y_test,predicted)
print('Confusion Matrix result of LogisticRegression is:',cm)
print("\n")
In [ ]:

accuracy = cross_val_score(LR, x, y, scoring='accuracy')
print('Cross validation test results of accuracy:', accuracy)
print("\n")
In [ ]:

a = accuracy.mean() * 100
print("Accuracy Result of LogisticRegression is:",a)
In [ ]:

def plot_confusion_matrix(cm, title='Confusion matrix-LogisticRegression', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predicted)
print('Confusion matrix-LogisticRegression:')
print(cm)
plot_confusion_matrix(cm)
In [ ]:

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

MODULE – 5:

XGBClassifier

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=pd.read_csv('A.csv')
data.head()
```

In [ ]:

```
df=data.dropna()
df
```

In [ ]:

```
df.columns
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['Group','M/F','Hand']

for i in var:
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
x = df.drop(labels='Group', axis=1)
y = df.loc[:,'Group']
```

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=1,
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))
In [ ]:

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.model_selection import cross_val_score
from xgboost import XGBClassifier
In [ ]:

XGB = XGBClassifier()
XGB.fit(x_train,y_train)
predicted = XGB.predict(x_test)
In [ ]:

cr = classification_report(y_test,predicted)
print('Classification report of XGBClassifier Result is:\n',cr)
print("\n")
In [ ]:

cm = confusion_matrix(y_test,predicted)
print('Confusion Matrix result of XGBClassifier is:',cm)
print("\n")
In [ ]:

accuracy = cross_val_score(XGB, x, y, scoring='accuracy')
print('Cross validation test results of accuracy:', accuracy)
print("\n")
In [ ]:

a = accuracy.mean() * 100
print("Accuracy Result of XGBClassifier is:",a)
In [ ]:

def plot_confusion_matrix(cm, title='Confusion matrix-XGBClassifier',
cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predicted)
print('Confusion matrix-XGBClassifier:')
print(cm)
plot_confusion_matrix(cm)
In [ ]:

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"][:100], marker='x', linestyle='dashed', color='red')
```

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

MODULE – 6:

GaussianNB

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=pd.read_csv('A.csv')
data.head()
```

In [ ]:

```
df=data.dropna()
df
```

In [ ]:

```
df.columns
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['Group','M/F','Hand']

for i in var:
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
x = df.drop(labels='Group', axis=1)
y = df.loc[:,'Group']
```

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=1,
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))
```

In [ ]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.model_selection import cross_val_score
```

```
from sklearn.naive_bayes import GaussianNB
In [ ]:

GN = GaussianNB()
GN.fit(x_train,y_train)
predicted = GN.predict(x_test)
In [ ]:

cr = classification_report(y_test,predicted)
print('Classification report of GaussianNB Result is:\n',cr)
print("\n")
In [ ]:

cm = confusion_matrix(y_test,predicted)
print('Confusion Matrix result of GaussianNB is:',cm)
print("\n")
In [ ]:

accuracy = cross_val_score(GN, x, y, scoring='accuracy')
print('Cross validation test results of accuracy:', accuracy)
print("\n")
In [ ]:

a = accuracy.mean() * 100
print("Accuracy Result of GaussianNB is:",a)
In [ ]:

def plot_confusion_matrix(cm, title='Confusion matrix-GaussianNB',
cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predicted)
print('Confusion matrix-GaussianNB:')
print(cm)
plot_confusion_matrix(cm)
In [ ]:

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

import joblib
```

```
joblib.dump(GN, 'model.pkl')
In [ ]:
```

DEPLOYMENT:

```
from django.shortcuts import render

from django.shortcuts import render, redirect

import numpy as np

import joblib


model = joblib.load('C:/Users/SPIRO15/Desktop/TOMORROW/DEMENTIA/deploy/app/model.pkl')


# Create your views here.

def home(request):

    return render(request, "index.html")


def predict(request):

    if request.method == "POST":

        int_features = [x for x in request.POST.values()]

        int_features = int_features[1:]

        print(int_features)

        final_features = [np.array(int_features, dtype=object)]

        print(final_features)

        prediction = model.predict(final_features)

        print(prediction)

        output = prediction[0]

        print(f'output{output}')
```

```
    if output == 0:
        return render(request, 'index.html', {"prediction_text":"DEMENTIA IS NON DEMENTED"})
    elif output == 1:
        return render(request, 'index.html', {"prediction_text":"DEMENTIA IS DEMENTED"})
    print(output)
```

A.2 SAMPLE SCREENSHOT:

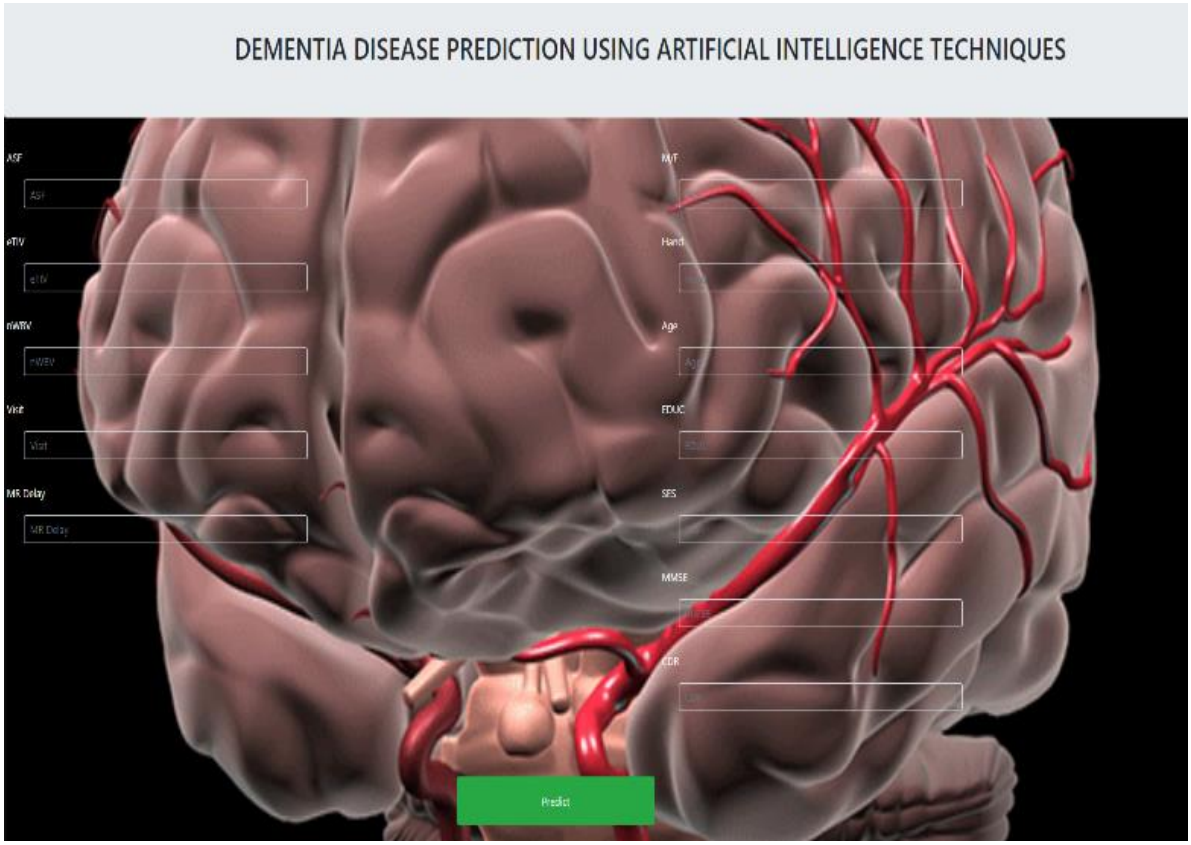

Fig A.2.1: Home Screen of The Application

In Fig A.2.1 The Home Page of The Screen is displayed and then we should be able to predict the Dimentia disease. By entering the values we can tell that dementia is demented, non-demented or converted. The following two screenshot are the results of the different patients.

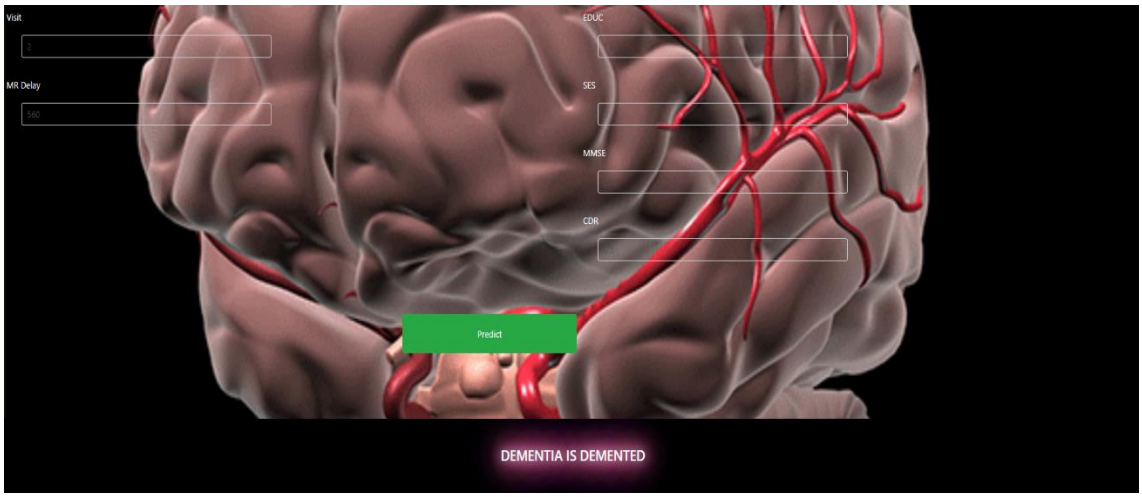

Fig A.2.2: Test result of dementia patient

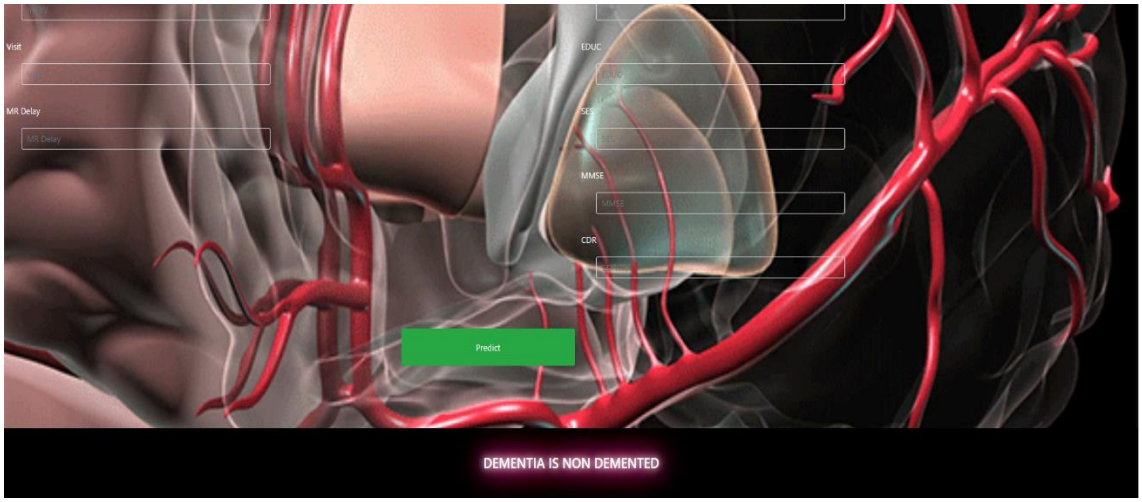

Fig A.2.3: Test result of non-dementia patient

REFERENCES

REFERENCES

[1] Nripendra Narayan Das; Neharika Srivastav; Sourabh Singh Verma, "Magnetic Resonance Imaging based Feature Extraction and Selection Methods for Alzheimer Disease Prediction" 2021, DOI: 10.1109/ICTAI53825.2021.9673337.

[2] D. Deepa; M. Sundar Raj; S. Gowthami; K.Hemalatha; C. Poongodi; P. Thangavel, "Identification and Analysis of Alzheimer's Disease using DenseNet Architecture with Minimum Path Length Between Input and Output Layers" 2022, DOI: 10.1109/STCR55312.2022.10009552.

[3] Carlton Chu; Peter Bandettini; John Ashburner; Andre Marquand; Stefan Kloeppel, "Classification of Neurodegenerative Diseases Using Gaussian Process Classification with Automatic Feature Determination", 2010, doi: 10.1109/WB"D.2010.11.

[4] Roman Filipovych; Bilwaj Gaonkar; Christos Davatzikos,
"A Composite Multivariate Polygenic and Neuroimaging Score for Prediction of Conversion to Alzheimer's Disease", 2012, DOI: 10.1109/PRNI.2012.9.

[5] Hongming Li , Yong Fan ; Alzheimer's Disease Neuroimaging Initiative, "EARLY PREDICTION OF ALZHEIMER'S DISEASE DEMENTIA BASED ON BASELINE HIPPOCAMPAL MRI AND 1-YEAR FOLLOW-UP COGNITIVE MEASURES USING DEEP RECURRENT NEURAL NETWORKS", Proc IEEE Int Symp Biomed Imaging, 2019,Apr;2019:368-371,doi:10.1109/ISBI.2019.8759397. Epub 2019 Jul 11.

[6] Priyanka Agarwal ; Mudit Khandelwal ; Nishtha ; Dr. Amol K. Kadam, "Brain Stroke Prediction using Machine Learning Approach", 2022, Published In: Iconic Research And Engineering Journals , Paper ID: 1703646.

[7] Dharvi Soni; Safvan Vahora , "A Review On Alzheimer Disease Prediction Methodology" 2022, doi: 10.1109/ICAIS53314.2022.9743137.

[8] P. Govindarajan, R. K. Soundarapandian, A. H. Gandomi, R. Patan, P. Jayaraman, and R. Manikandan, stroke disease classification using machine learning algorithms," Neural Computing and Applications in 2019.

[9] Dementia Prediction w/ Tree-based Models. Kaggle.com, 2018,https://www.kaggle.com/code/ruslankl/dementia-prediction-w-tree-based-models/report

[10] Gangavarapu Sailasya and Gorli L Aruna Kumari, "Analyzing the Performance of Stroke Prediction using ML Classification Algorithms," International Journal of Advanced Computer Science and Applications (IJACSA), 2021.