BRAIN STROKE PREDICTION BY USING CLASSIFICATION TECHNIQUE

A PROJECT REPORT

Submitted by

SRIVAS C [REGISTER NO:211419104269]

HARSHA VARDHAN B [REG:211419104096]

YUVA PRAKASH M [REGISTER NO:211419104315]

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 **"BRAIN STROKE PREDICTION BY USING CLASSIFICATION TECHNIQUE"** is the bonafide work of "**SRIVAS C (211419104269), HARSHA VARDHAN B (211419104096), YUVA PRAKASH M(211419104315)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA,M.E.,Ph.D., **HEAD OF THE DEPARTMENT**	**Mrs. VIJAYALAKSHMI.P,** **M.C.A.,M.Phil.,M.Tech.,** **ASSISTANT PROFESSOR** **GRADE 1**
DEPARTMENT OF CSE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, POONAMALLEE, CHENNAI-600 123.	DEPARTMENT OF CSE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, POONAMALLEE, CHENNAI-600 123.

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

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **SRIVAS C (211419104269), HARSHA VARDHAN B (211419104096), YUVA PRAKASH M** (**211419104315**) hereby declare that this project report titled **“BRAIN STROKE PREDICTION BY USING CLASSIFICATION TECHNIQUE” ,** under the guidance of **Mrs.VIJAYALAKSHMI.P ,M.C.A., M.Phil., M.Tech.,** is the Original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

SRIVAS C

B HARSHA VARDHAN

YUVA PRAKASH M

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 project coordinator **Mr.M.MOHAN, M.Tech.,(Ph.D.),** and our project guide **Mrs.VIJAYALAKSHMI.P,M.C.A,M.Phil.,M.Tech** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

SRIVAS C

HARSHA VARDHAN B

YUVA PRAKASH M

ABSTRACT

Stroke 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), stroke is the greatest cause of death and disability globally. Early recognition of the various warning signs of a stroke can help to reduce the severity of the stroke. Different machine learning (ML) models have been developed to predict the likelihood of a stroke occurring in the brain. The dataset used in the development of the method was the open-access Stroke 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

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**v**
	LIST OF TABLES	**ix**
	LIST OF FIGURES	**x**
	LIST OF ABBREVATIONS	**xv**
1	**INTRODUCTION**	**1**
	1.1 INTRODUCTION	**2**
	1.1.1 DOMAIN OVERVIEW	**3**
	1.2 PROBLEM STATEMENT	**10**
2	**LITERATURE SURVEY**	**12**
3	**SYSTEM ANALYSIS**	**17**
	3.1 EXISTING SYSTEM	**18**
	3.2 PROPOSED SYSTEM	**19**
	3.3 FEASIBILITY STUDY	**19**
	3.4 REQUIREMENT SPECIFICATION	**22**
	3.4.1 SOFTWARE REQUIREMENTS	**22**
	3.4.2 HARDWARE REQUIREMENTS	**22**
	3.5 SOFTWARE SPECIFICATION	**22**
	3.5.1 ANACONDA DISTRIBUTION	**22**
	3.5.2 JUPITER NOTEBOOK	**25**
	3.5.3 PYTHON 3.6	**26**
	3.5.4 KAGGLE	**28**

4	**SYSTEM DESIGN**	**29**
	4.1 SYSTEM ARCHITECTURE	**30**
	4.2 DATA FLOW DIAGRAM	**31**
	4.3 UML DIAGRAMS	**32**
	4.3.1 Use Case Diagram	**32**
	4.3.2 Sequence Diagram	**33**
	4.3.3 Activity Diagram	**34**
	4.3.4 Class Diagram	**35**
5	**SYSTEM ARCHITECTURE**	**36**
	5.1 MODULES	**37**
	5.1.1 DESCRIPTION ABOUT EACH MODULE	**37**
6	**SYSTEM IMPLEMENTATION**	**51**
	6.1 NUMPY	**52**
	6.2 OS MODULE	**52**
	6.3 OPENCV	**52**
	6.4 SCIKIT-LEARN	**53**
	6.4.1 Shuffle	**53**
	6.4.2 Confusion Matrix	**53**
	6.4.3 Train_Test_Split	**53**
	6.4.4 Classification_Report	**53**
	6.5 MATPLOTLIB	**53**

	6.6 PANDAS	**53**
	6.7 WARNINGS	**53**
	6.8 SOURCE CODE	**54**
7	**SYSTEM TESTING**	**64**
	7.1 Unit Testing	**65**
	7.2 Integration Testing	**65**
8	**CONCLUSION**	**66**
	APPENDIX	**68**
	SCREENSHOT	**69**
	BIBLIOGRAPHY	**73**

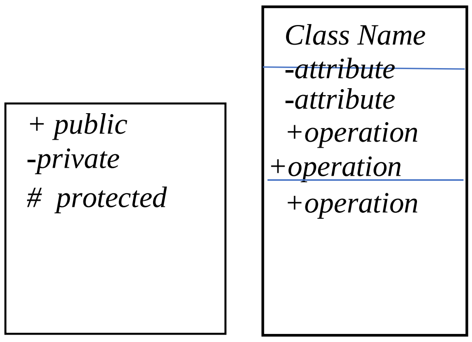
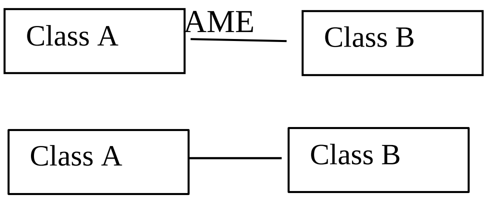
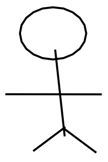
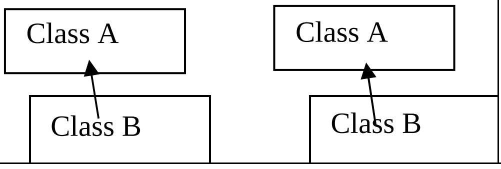
LIST OF TABLES

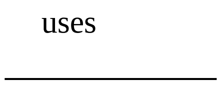
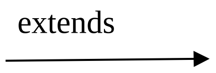
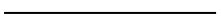
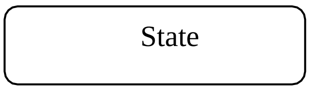
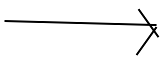
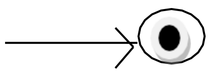
TABLE NO.	TABLE NAME	PAGE NO.
Table 4.1	DFD Symbols	27

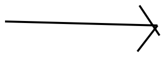
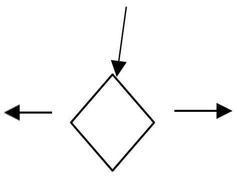
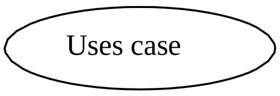
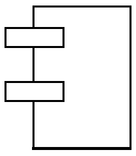
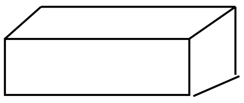
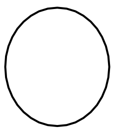
LIST OF FIGURES

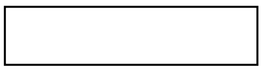
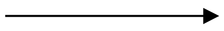
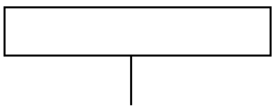
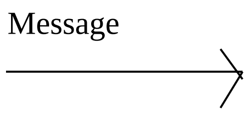
FIGURE NO.	FIGURE NAME	PAGE NO.
Figure 1.1	PROCESS OF MACHINE LEARNING	8
Figure 3.1	PROCESS OF DATAFLOW DIAGRAM	19
Figure 3.2	ANACONDA NAVIGATOR	21
Figure 4.1	SYSTEM ARCHITECTURE	27
Figure 4.2	FLOW CHART	28
Figure 4.3.1	USE CASE DIAGRAM	29
Figure 4.3.2	SEQUENCE DIAGRAM	30
Figure 4.3.3	ACTIVITY DIAGRAM	31
Figure 4.3.4	CLASS DIAGRAM	32
Figure 5.1,5.2	DATA PREPROCESSING	36
Figure 5.3	MODULE 1 DIAGRAM	36
Figure 5.4	MODULE 2 DIAGRAM	37
Figure 5.5	CONFUSION MATRIX FOR MLP	40
Figure 5.6	MODULE 3 DIAGRAM	41
Figure 5.7	IMPLEMENTATION OF MLP CLASSIFIER	42
Figure 5.8	MODULE 4 DIAGRAM	42
Figure 5.9	MODULE 5 DIAGRAM	43
Figure 5.10	CONFUSION MATRIX FOR DTC	44
Figure 5.11	MODULE 6 DIAGRAM	45
Figure 5.12	CONFUSION MATRIX FOR SVM	45

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	 + public -private # protected Class Name -attribute -attribute +operation +operation +operation	Represents a collection of similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the

			system and external environment
5.	Relation(uses)		Used for additional process communication.
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the process.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object

11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Use case		Interaction between the system and external environment.
14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which are a collection of components
16.	DataProcess/State		A circle in DFD represents a state or process which

			has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard, sensors etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message		Represents the message exchanged.

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CNN	Convolutional Neural Network
GNN	Graph Neural Network
CT	Computer Tomography
AHDCNN	Adaptive Hybridized Deep CNN
RELU	Rectified Linear Unit

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION

Stroke 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 stroke 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 strokes each year, with one person dying every four to five minutes in the affected population. Stroke is the sixth leading cause of mortality in the United States according to the Centers for Disease Control and Prevention (CDC) [1]. Stroke 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 strokes on a regular basis [2]. It is India's fourth leading cause of death. Strokes are classified as ischemic or hemorrhagic. In a chemical stroke, clots obstruct the drainage; in a hemorrhagic stroke, a weak blood vessel bursts and bleeds into the brain. Stroke 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. Stroke 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 stroke by utilizing ML techniques. The algorithms included in ML are beneficial as they allow for accurate prediction and proper analysis. The majority of previous stroke-related research has focused on, among other things, the prediction of heart attacks. Brain stroke 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 stroke. 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.

1.1.1 DOMAIN OVERVIEW

DATA SCIENCE

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

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.

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.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect.

For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding.

AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals.

To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning.AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors. 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.

Natural 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. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

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.

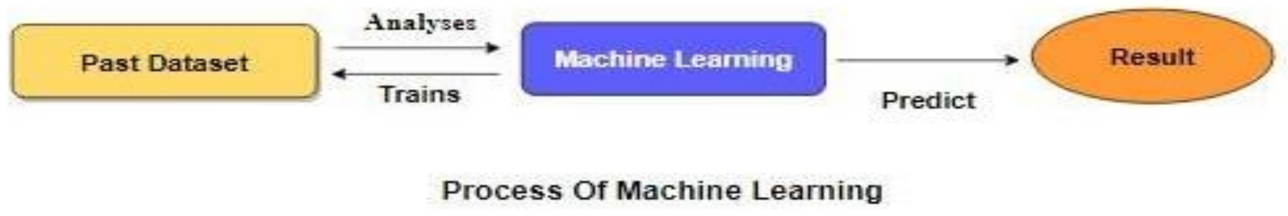


Figure 1.1

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”.

1.2 PROBLEM STATEMENT

Stroke is the second leading cause of death worldwide and remains an important health burden both for the individuals and for the national healthcare systems. Potentially modifiable risk factors for stroke include hypertension, cardiac disease, diabetes, and dysregulation of glucose metabolism, atrial fibrillation, and lifestyle factors.

Therefore, the goal of our project is to apply principles of machine learning over large existing data sets to effectively predict the stroke based on potentially modifiable risk factors. Then it intended to develop the application to provide a personalized warning based on each user's level of stroke risk and a lifestyle correction message about the stroke risk factors.

The goal is to develop a novel decision-making tool for predicting strokes. This study compares the accuracy, precision, and execution time of convolution neural networks, densenet, and VGG 16 for stroke prediction. The performance of the CNN classifier is superior than Densent and VGG 16, according to the findings of the experiments.

SCOPE

The scope of this project is to investigate Brain stroke prediction classification technique using machine learning technique. To identifying Brain stroke, occur or NOT.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

1.Title: Brain Stroke Prediction Using Machine Learning Approach

Author: DR. AMOL K. KADAM , PRIYANKA AGARWAL

Year: 2022

A Stroke is an ailment that causes harm by tearing the veins in the mind. Stroke 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, stroke is one of the world's driving reasons for death and incapacity. The majority of the work has been completed on heart stroke forecast however not many works show the gamble of a cerebrum stroke. Subsequently, the AI models are worked to foresee the chance of cerebrum stroke. The project is pointed towards distinguishing the familiarity with being in danger of stroke 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.Title: **Brain Stroke Prediction using Machine Learning**

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

Year: 2014

A stroke, 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 stroke is a medical emergency because strokes can lead to death or permanent disability. There are opportunities to treat ischemic strokes but that treatment needs to be started in the first few hours after the signs of a stroke begin. The patient, family, or bystanders should activate emergency medical services immediately should a stroke be suspected.

A transient ischemic attack (TIA or mini-stroke) describes an ischemic stroke that is short-lived where the symptoms resolve spontaneously. This situation also requires emergency assessment to try to minimize the risk of a future stroke. By definition, a stroke would be classified as a TIA if all symptoms resolved within 24 hours. According to the World Health Organization (WHO) stroke 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 stroke 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 Stroke.

3. Title: **Machine Learning for Brain Stroke: 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 stroke 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 stroke 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 stroke from the year 2007 to 2019. Support Vector Machine (SVM) is obtained as optimal models in 10 studies for stroke problems. Besides, maximum studies are found in stroke diagnosis although number for stroke treatment is least thus, it identifies a research gap for further investigation. Similarly, CT images are a frequently used dataset in stroke. 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 stroke.

4. Title: **Predicting Stroke Risk With an Interpretable Classifier**

Author: SERGIO PEÑAFIEL, NELSON BALOIAN, HORACIO SANSON, AND JOSÉ A. PINO

**Year : **2021

Predicting an individual's risk of getting a stroke 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 stroke 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.

5. Title: **Towards stroke prediction using electronic health records**

Author: Douglas Teoh

Year: 2014

Background: As of 2014, stroke is the fourth leading cause of death in Japan. Predicting a future diagnosis of stroke would better enable proactive forms of healthcare measures to be taken. We aim to predict a diagnosis of stroke 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 stroke 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

Stroke 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 stroke risk prediction models, but they rely on large well-labeled data. Due to the strict privacy protection policy in health-care systems, stroke 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 stroke data. We have proposed a novel Hybrid Deep Transfer Learning-based Stroke Risk Prediction (HDTL-SRP) framework which consists of three key components: Generative Instance Transfer (GIT) for making use of the external stroke 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 stroke 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.

Disadvantages:

1. Accuracy is low.
2. They are not using any machine learning algorithms.
3. They are not implementing the performance metrics.

3.2 PROPOSED SYSTEM

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

Advantages:

1. Accuracy will be increased.
2. Machine Learning Algorithms are compared and the best model is evaluated for better prediction.
3. Performance metrics of different algorithms are compared and a better prediction is done.

3.3 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 wsn attack, A Random Forest Algorithm prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.
- **Construction of a Predictive Model**

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

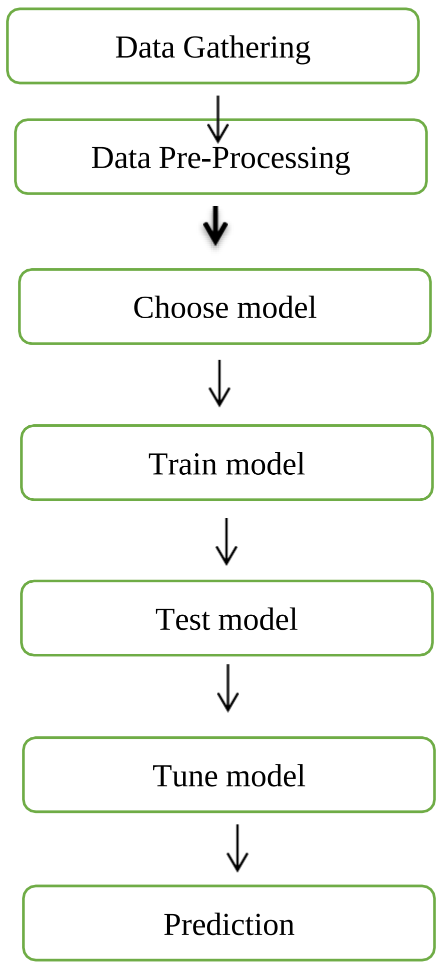


FIGURE 3.1 Process of dataflow diagram

3.4 REQUIREMENT SPECIFICATION

3.4.1 SOFTWARE REQUIREMENTS

Operating System : Windows 2007

Tool : Anaconda with Jupyter Notebook

Coding Language : Python

3.4.2 HARDWARE REQUIREMENTS

Processor	: Pentium IV/III
Hard disk	: minimum 100 GB
RAM	: minimum 4 GB

3.5 SOFTWARE SPECIFICATION

3.5.1 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.

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

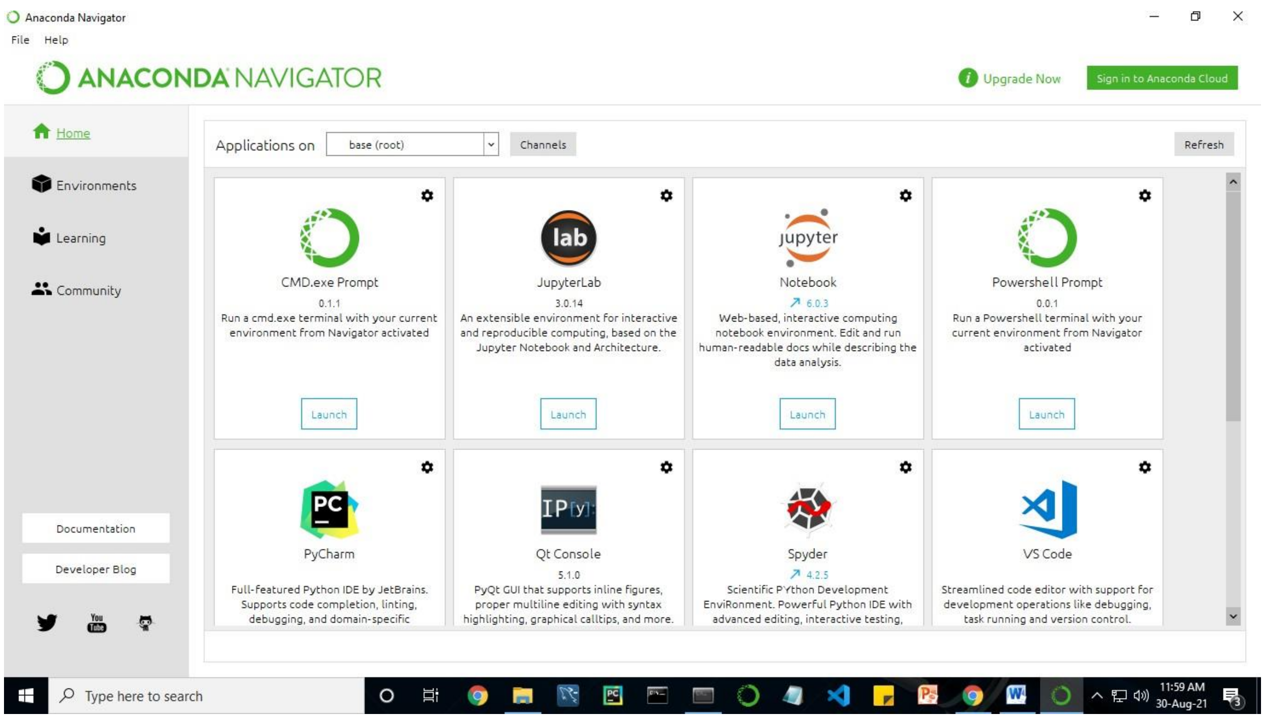
In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

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

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)



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
3.0.14
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

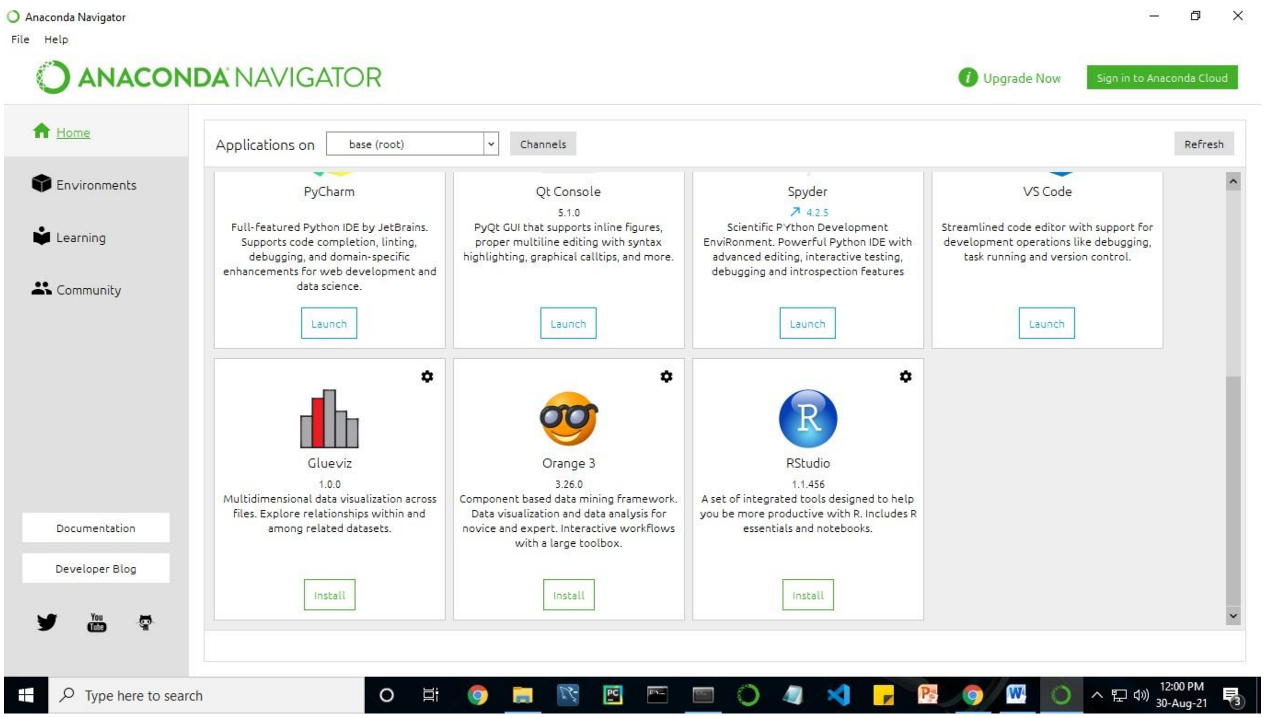
PyCharm
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific

Qt Console
5.1.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
4.2.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing,

VS Code
Streamlined code editor with support for development operations like debugging, task running and version control.

FIGURE 3.2



PyCharm

Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.

Qt Console

5.1.0

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder

4.2.5

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code

Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz

1.0.0

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3

3.26.0

Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio

1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

FIGURE 3.3

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

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

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

3.5.2 JUPYTER NOTEBOOK

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.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels

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.

3.5.3 PYTHON

Introduction

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.Python consistently ranks as one of the most popular programming languages.

History

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC Programming Language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning

3.5.4 KAGGLE

Kaggle is an online community platform for data scientists and machine learning enthusiasts. Kaggle allows users to collaborate with other users, find and publish datasets, use GPU integrated notebooks, and compete with other data scientists to solve data science challenges.

Kaggle provides powerful resources on cloud and allows us to use a maximum of 30 hours of GPU and 20 hours of TPU per week. We can upload our datasets to Kaggle and download others' datasets as well. Additionally, we can check other people's datasets and notebooks and start discussion topics on them. All our activity is scored on the platform and our score increases as we help others and share useful information. Once we start earning points, we will be placed on a live leaderboard of 8 million Kaggle users.

CHAPTER 4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

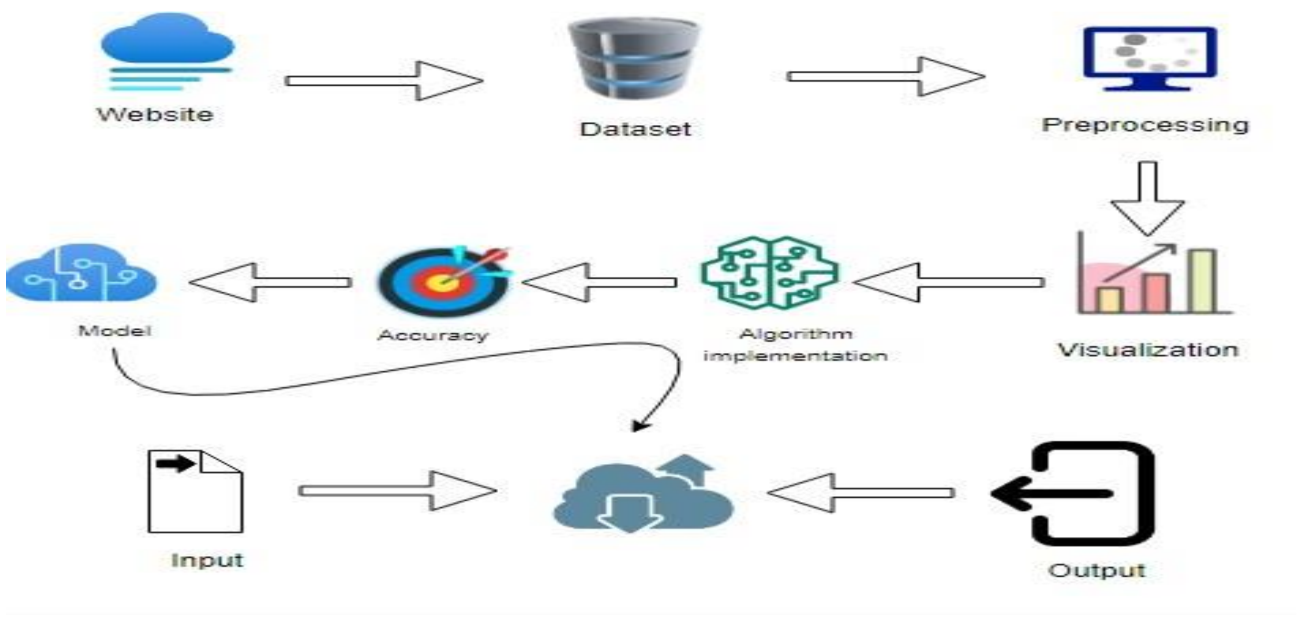


FIGURE NO 4.1

4.2 DATA FLOW DIAGRAM

SYMBOLS	NAME	DESCRIPTION
□	External Entity	They are outside the system and they either supplyinput data into the system or use system output.
═	Data Store	Huge collection of data and used for storagepurpose.
○	Process	Shows transformation manipulation of Data Flowswithin the system.
→	Data Flow	Shows flow of information from input to output.

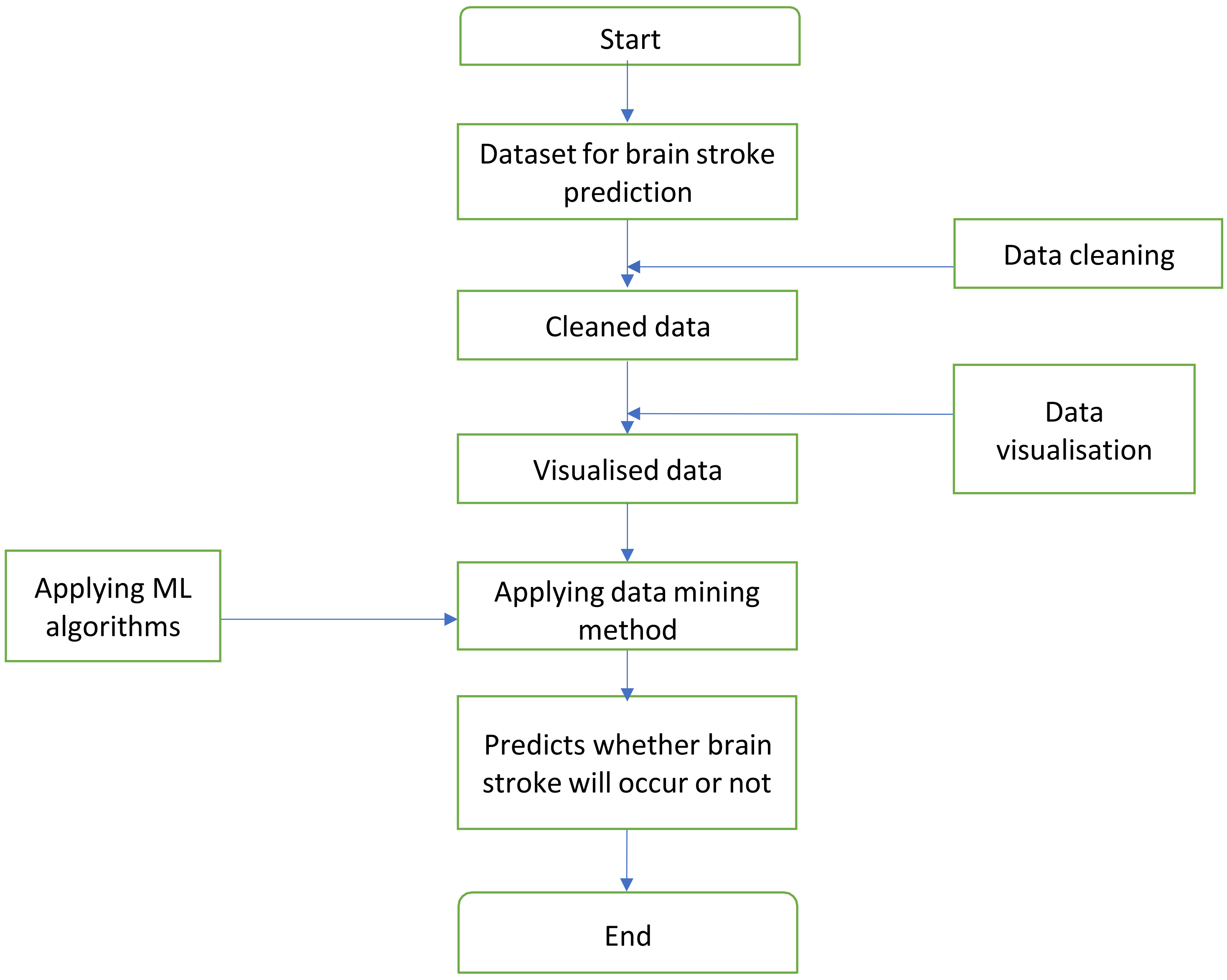

Start
Dataset for brain stroke prediction
Data cleaning
Cleaned data
Data visualisation
Visualised data
Applying ML algorithms
Applying data mining method
Predicts whether brain stroke will occur or not
End

FIGURE NO 4.2

4.3 UML DIAGRAMS

4.3.1 USE CASE DIAGRAM

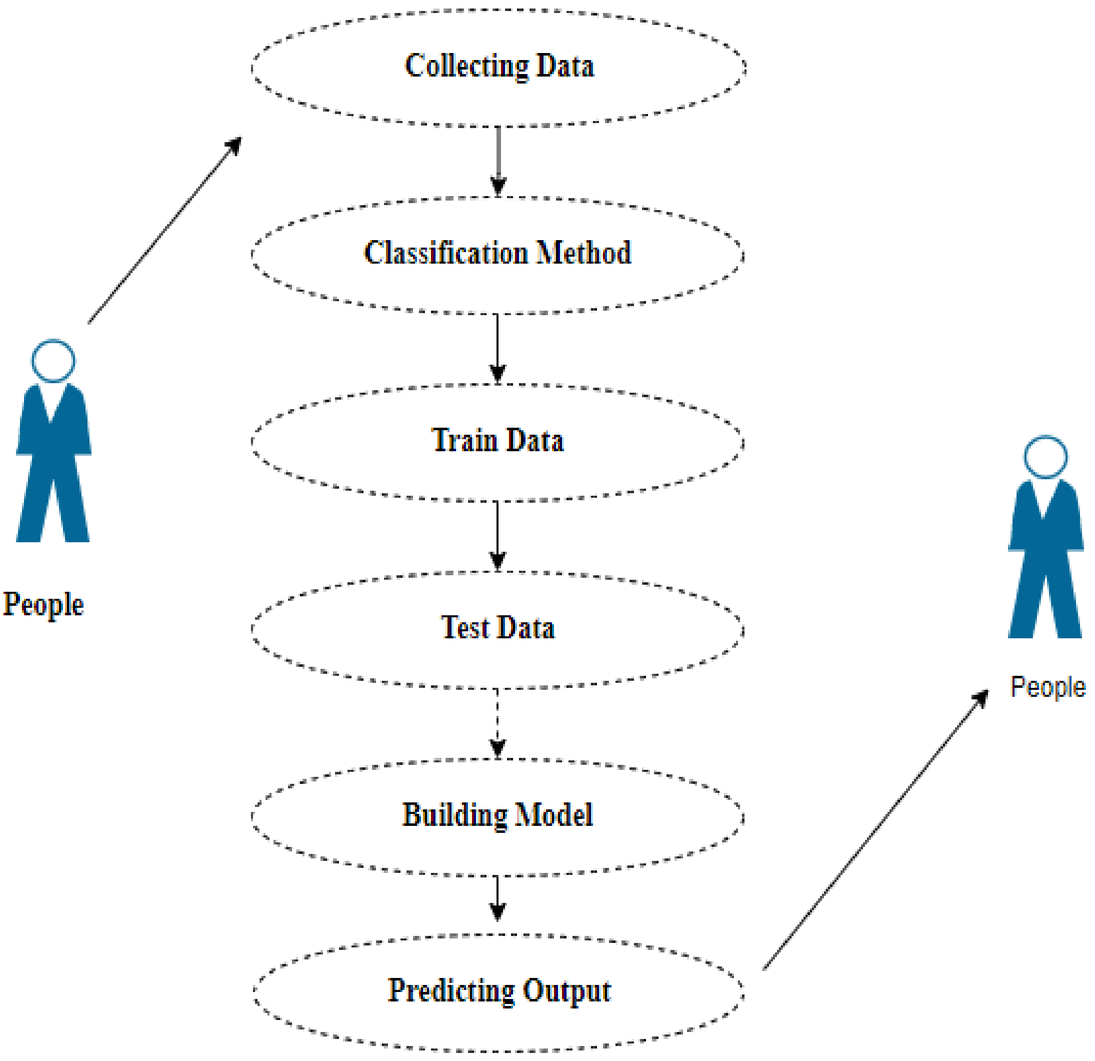


FIGURE NO 4.3.1

In this use case diagram, we will collect data from the user and train and test the data using classification method for predicting output.

4.3.2 SEQUENCE DIAGRAM

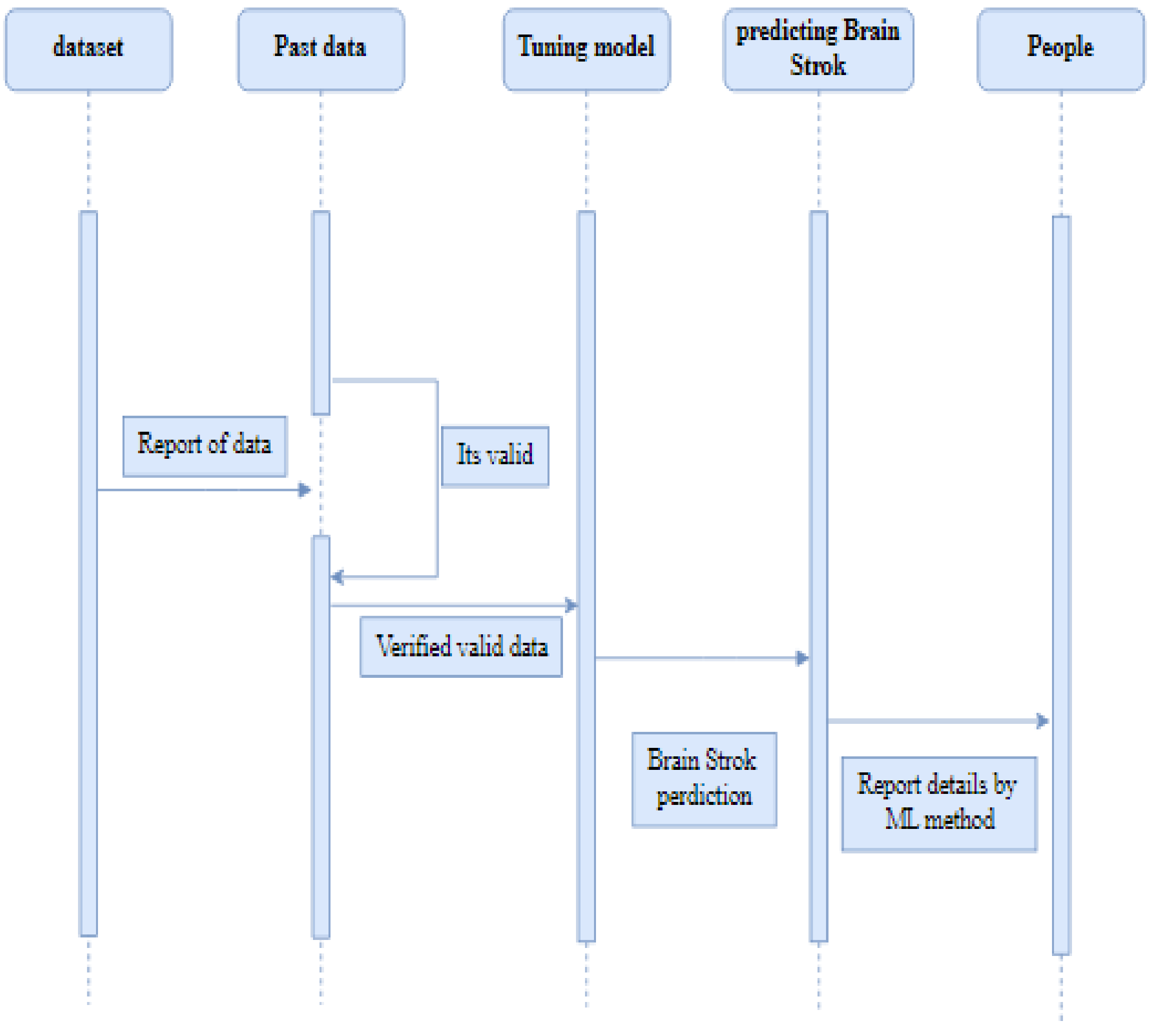


FIGURE NO 4.3.2

The collected set of data will be validated and the validated data will be predicted by using the ML method.

4.3.3 ACTIVITY DIAGRAM

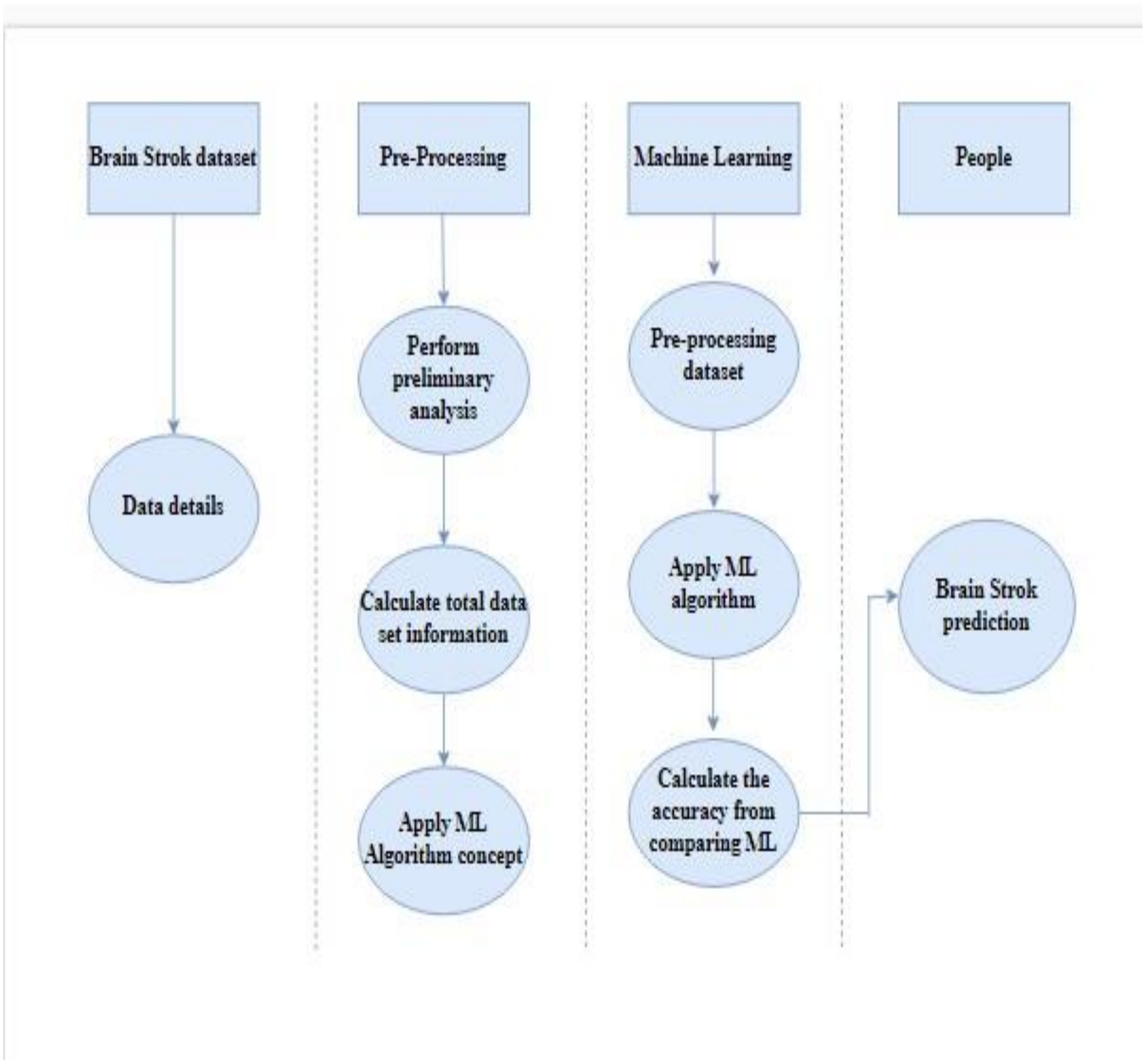

Brain Strok dataset
Pre-Processing
Machine Learning
People
Data details
Perform preliminary analysis
Calculate total data set information
Apply ML Algorithm concept
Pre-processing dataset
Apply ML algorithm
Calculate the accuracy from comparing ML
Brain Strok prediction

FIGURE NO 4.3.3

The detailed information collected from the people performing preliminary analysis and calculating the total set of information and applying the ML algorithm.

Pre processing the data by applying ML algorithm and calculating the accuracy from comparing ML algorithm and predicting the result.

4.3.4 CLASS DIAGRAM

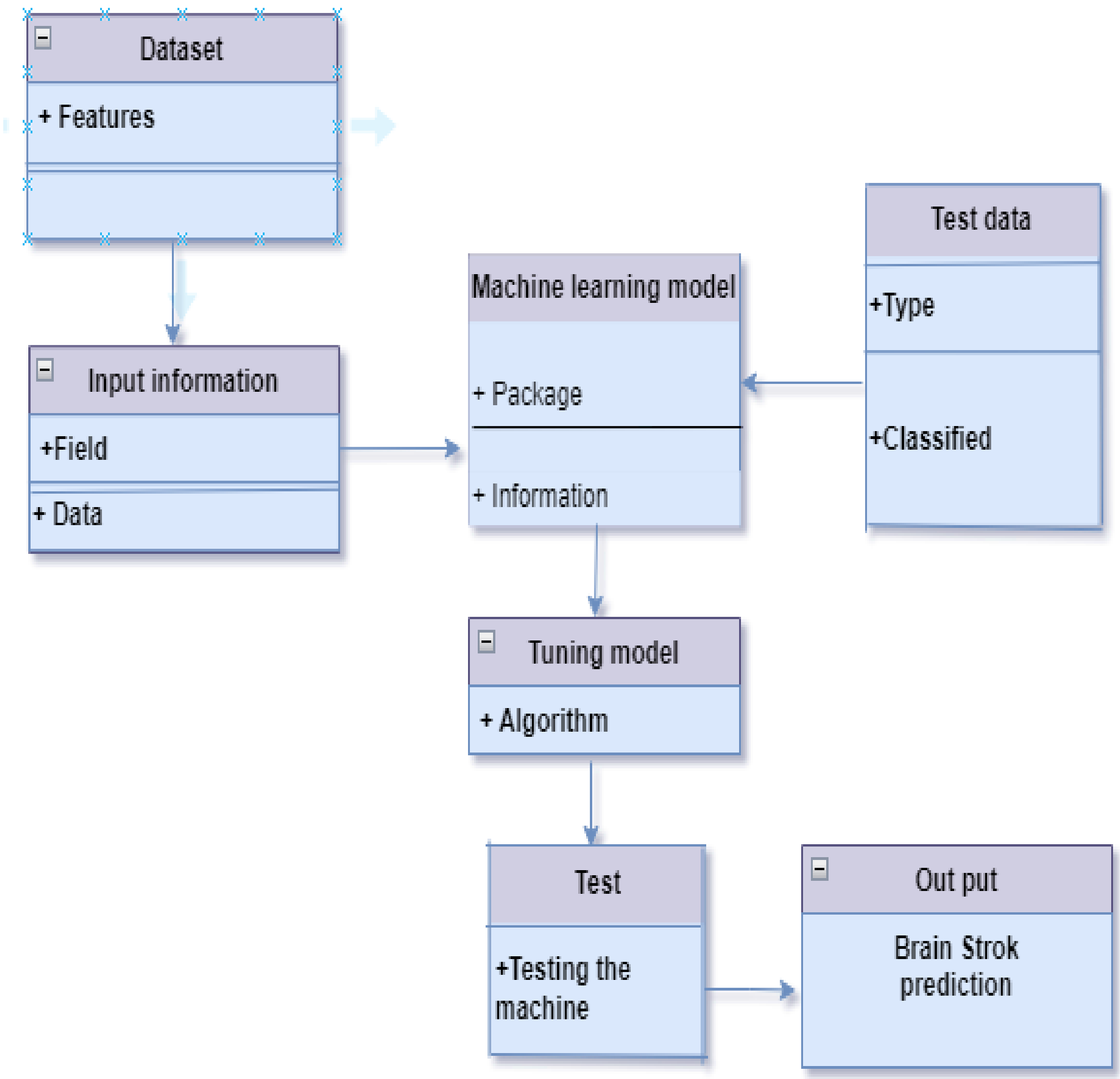


FIGURE NO 4.3.4

This class diagram represents about the patient being evaluated for the risk of stroke and represents the various symptoms that a patient may experience the stroke and different risk factors that can increase the livelihood of the patient and represents the diagnosis of risk and also represents the health care professional responsible for evaluating the patients risk of stroke and the algorithm that calculates the patients risk of stroke based on their symptoms.

CHAPTER 5

SYSTEM ARCHITECTURE

5. SYSTEM ARCHITECTURE

5.1 MODULES

5.1.1 DESCRIPTION ABOUT EACH MODULE

MODULE-1

DATA PRE-PROCESSING

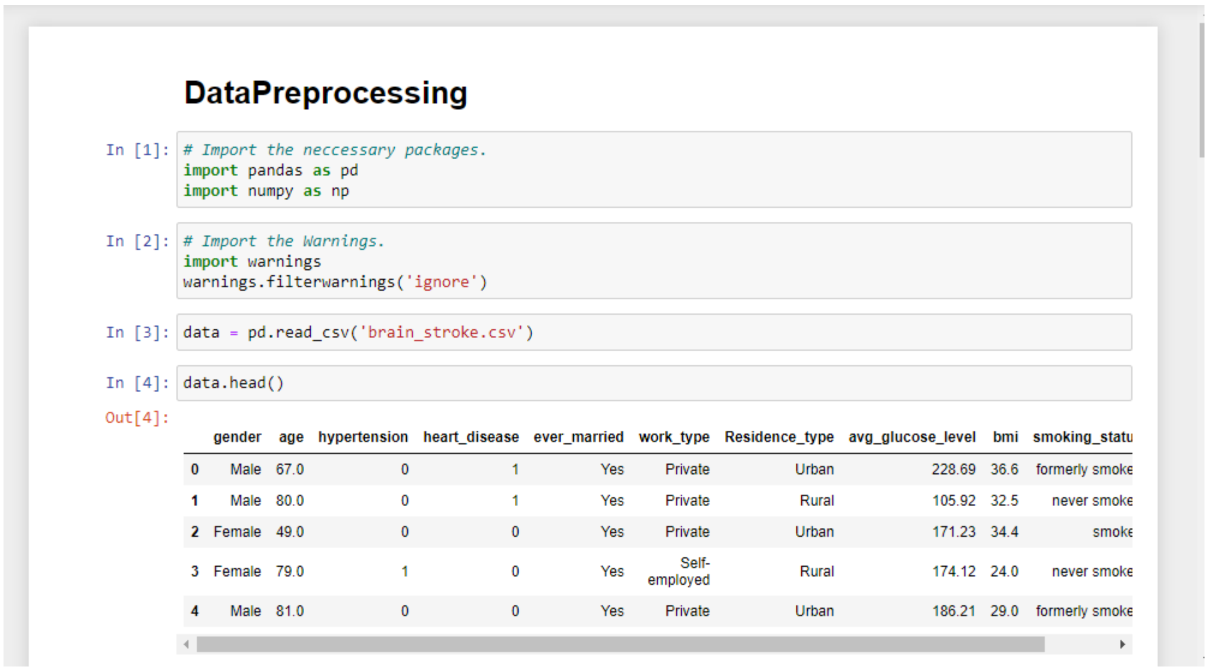
Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, 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.

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

A number of different data cleaning tasks using Python’s Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data**.** It wants to spend less time cleaning data**,** and more time exploring and modeling.

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 model's 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 of data in analytics and decision making.



DataPreprocessing

In [1]:
```
# Import the neccessary packages.
import pandas as pd
import numpy as np
```

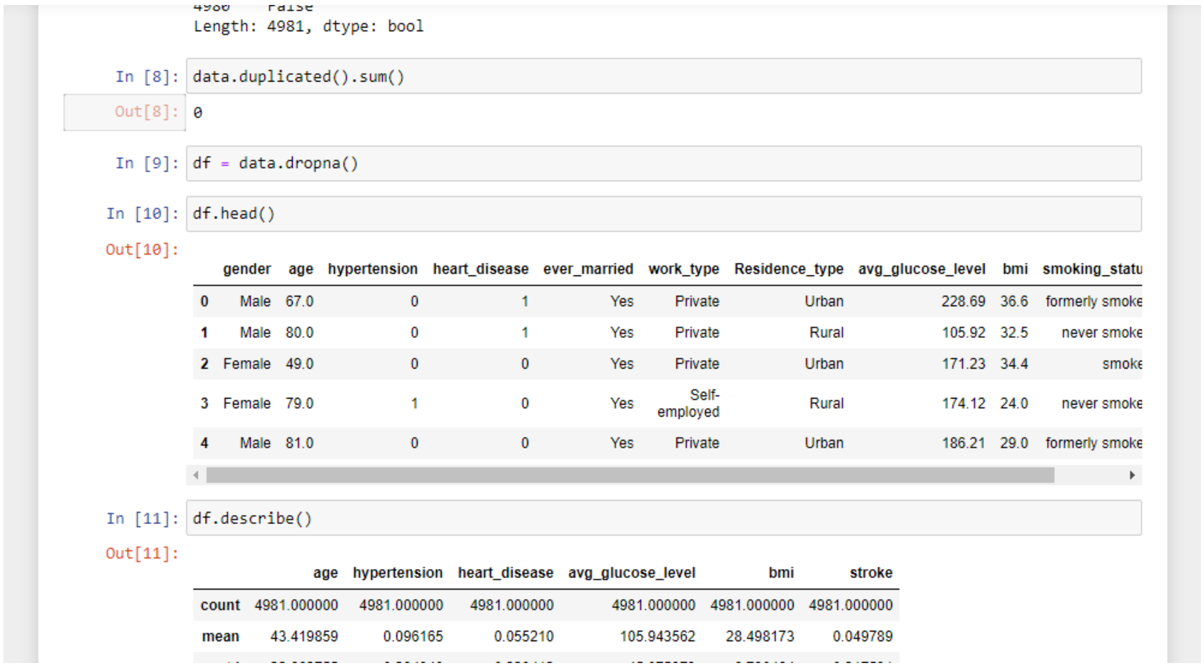
In [2]:
```
# Import the Warnings.
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
data = pd.read_csv('brain_stroke.csv')
```

In [4]:
```
data.head()
```

Out[4]:
```
   gender  age   hypertension  heart_disease  ever_married  work_type  Residence_type  avg_glucose_level  bmi   smoking_statu
0    Male  67.0             0              1           Yes    Private           Urban             228.69  36.6  formerly smoke
1    Male  80.0             0              1           Yes    Private           Rural             105.92  32.5     never smoke
2  Female  49.0             0              0           Yes    Private           Urban             171.23  34.4           smoke
3  Female  79.0             1              0           Yes      Self-           Rural             174.12  24.0     never smoke
                                                             employed
4    Male  81.0             0              0           Yes    Private           Urban             186.21  29.0  formerly smoke
```

FIGURE NO 5.1



```
4980    False
Length: 4981, dtype: bool

In [8]: data.duplicated().sum()

Out[8]: 0

In [9]: df = data.dropna()

In [10]: df.head()

Out[10]:
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_statu
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoke
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoke
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smoke
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoke
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoke

```
In [11]: df.describe()

Out[11]:
```

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	4981.000000	4981.000000	4981.000000	4981.000000	4981.000000	4981.000000
mean	43.419859	0.096165	0.055210	105.943562	28.498173	0.049789

FIGURE NO 5.2

Module Diagram

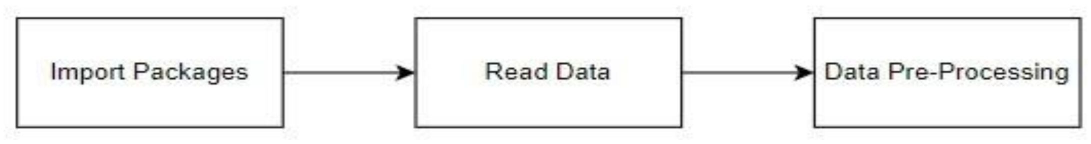


FIGURE NO 5.3

GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

MODULE-2

DATA VISUALIZATION

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

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

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

Module Diagram:

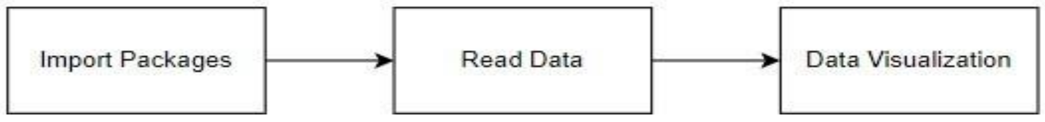


FIGURE NO 5.4

GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

False Positive (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.

MODULE-3

IMPLEMENTING NAÏVE BAYES ALGORITHM

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. Naive Bayes 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 together 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. Naive Bayes 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.

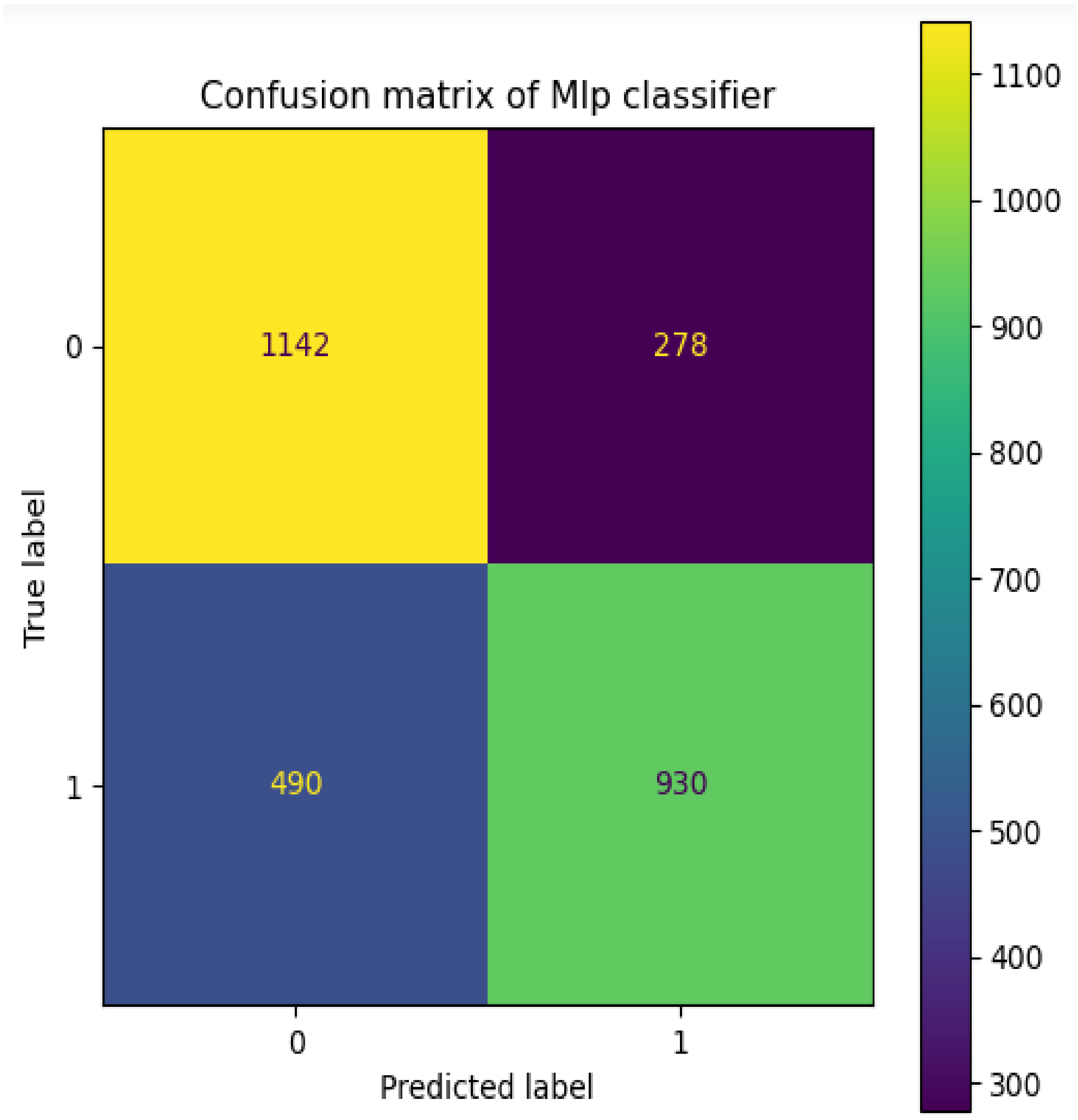

Confusion matrix of Mlp classifier
True label
Predicted label
0
1
0
1
1142
278
490
930
1100
1000
900
800
700
600
500
400
300

FIGURE NO 5.5

MODULE DIAGRAM

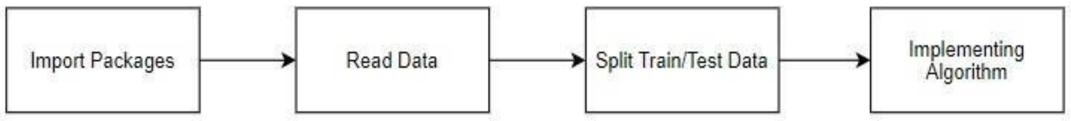


FIGURE NO 5.6

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

MODULE-4

IMPLEMENTING MLP (MULTI LAYER PERCEPTRON)

A multilayer perceptron (MLP) is a feedforward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input and output layers. MLP classifier is a very powerful neural network model that enables the learning of non-linear functions for complex data. The method uses forward propagation to build the weights and then it computes the loss. Next, back propagation is used to update the weights so that the loss is reduced. A multilayer perceptron (MLP) is a fully connected class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to mean any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation); see § Terminology.

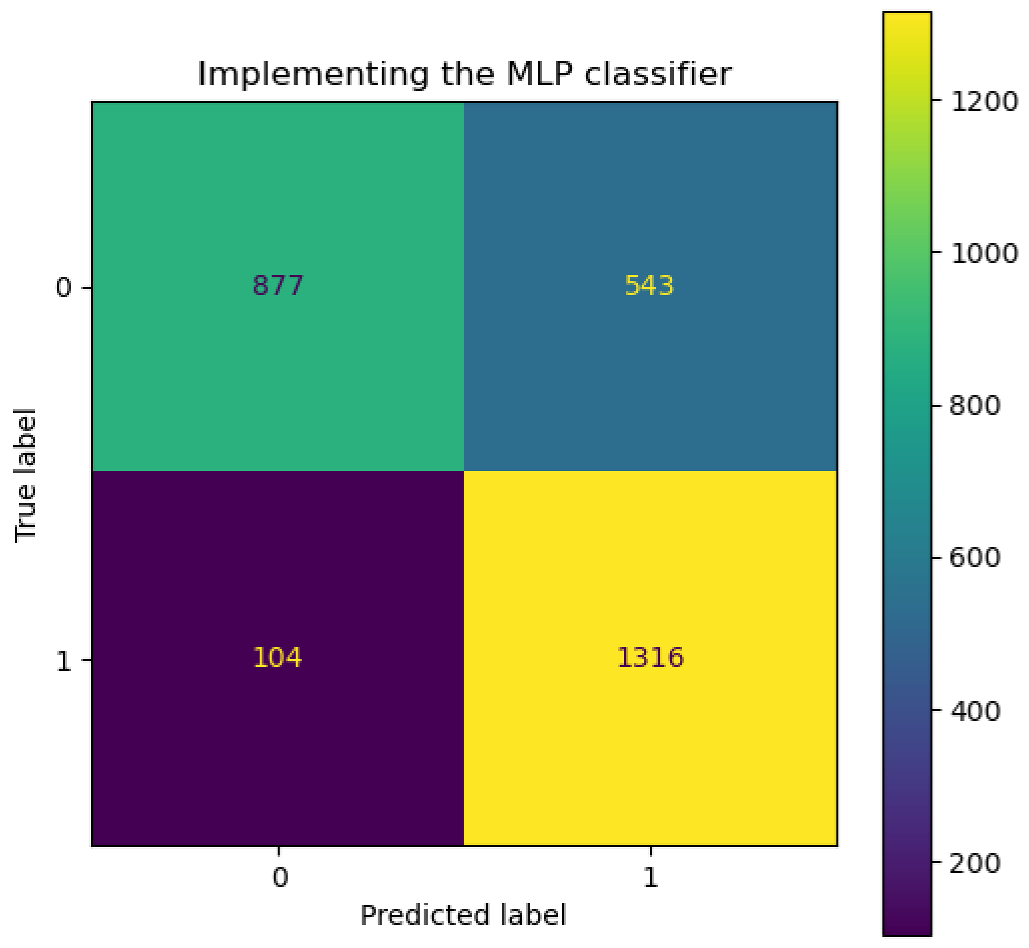


FIGURE NO 5.7

MODULE DIAGRAM

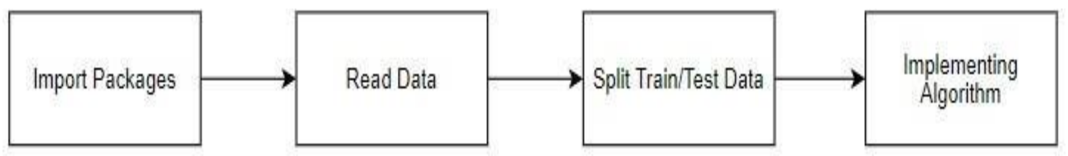


FIGURE NO 5.8

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

MODULE-5

IMPLEMENTING 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.

MODULE DIAGRAM

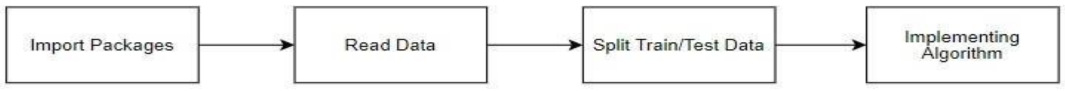


FIGURE NO 5.9

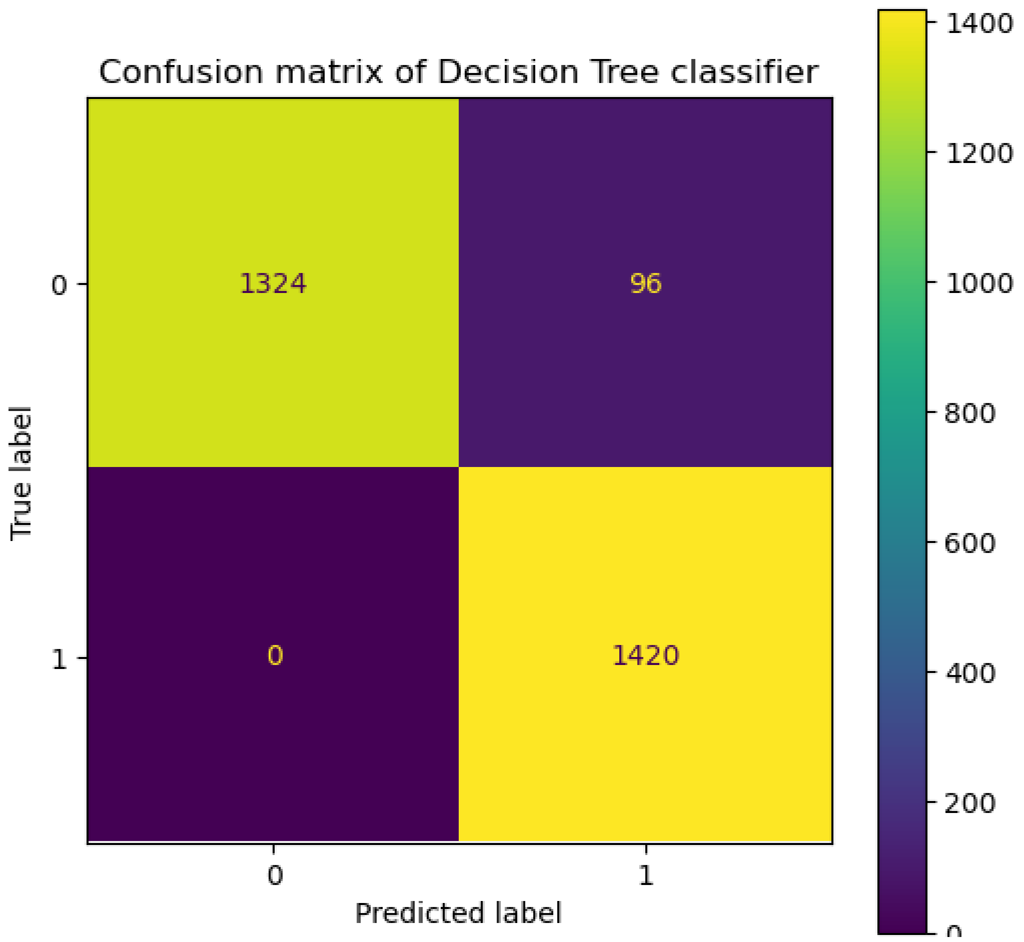


FIGURE NO 5.10

GIVEN INPUT EXPECTED OUTPUT

input : data

output: getting accuracy

MODULE-6

IMPLEMENTING SUPPORT VECTOR MACHINE

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

MODULE DIAGRAM

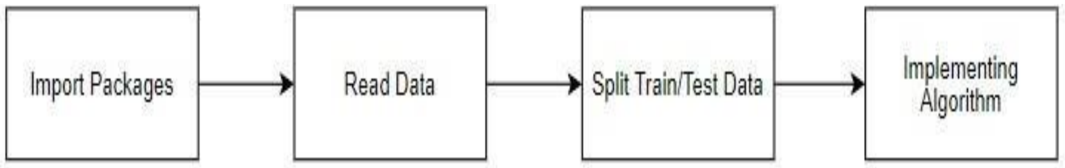


FIGURE NO 5.11

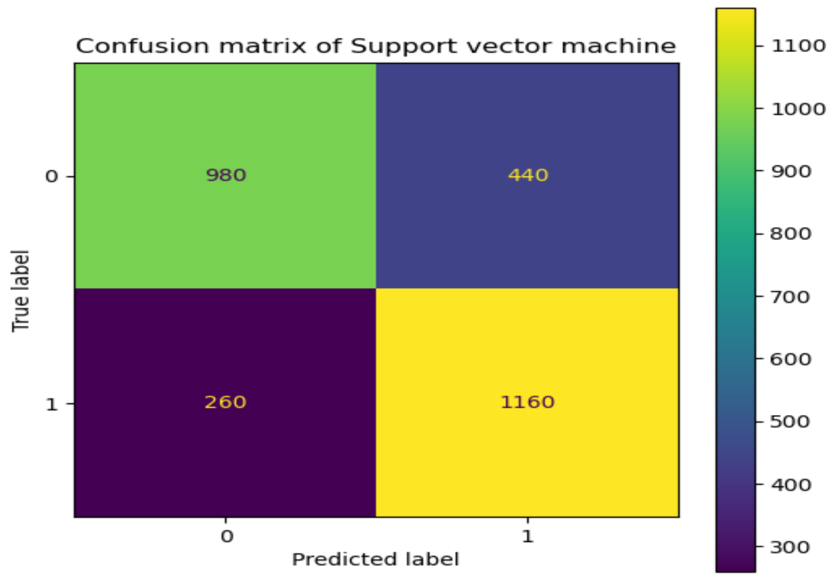


FIGURE NO 5.12

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

CHAPTER 6

SYSTEM IMPLEMENTATION

6. SYSTEM IMPLEMENTATION

a. 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.

b. OS MODULE

This module is a standard library within python3 coming up only with the installation of python allowing us to interface with the underlying operating system. The main motive of this OS module is to make interactions with our own operating systems so as to create folders, remove them, move folders and make some changes inthe working directory. The major purpose is to access the names of files within a file path by doing listdir().

c. OPENCV

Open source Computer Vision is a library of python bindings aimed to solve the computer vision problems. It is an artificial intelligence technology developed by Intel as a cross platform and open-source library of programing functions mainly for supporting real time computer vision and image processing in python. It is a discipline under a broad area of artificial intelligence applied in deep learning, computer vision,Image analysis, artificial intelligence that is used across developing industries like Amazon Go, Google lens, Autonomous Vehicles, Face recognition influenced by a number of factors like the mobile technology with built-in cameras which saturated the world, computing power that are more affordable and easily accessible, Hardware designed for computer vision and analysis is more widely available, new algorithms like convolution neural network can take advantage of the hardware and software capabilities. Within a decade, the accuracy rates for object identification have been raised from 50 percent to 99 percent. This technology makes the computer systems tobe more accurate than human at detection and reaction to the visual inputs.

d. SCIKIT-LEARN

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.

e. MATPLOTLIB

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with data frames.

f. PANDAS

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

g. WARNINGS

This class is a subclass of Exception, which is one among a number of built-in exceptions that throws out warning categories which is useful to filter out the group ofwarnings. This class acts as a base class of all warning category classes.

6.8 SOURCE CODE

Module 1: Data validation and pre-processing technique

```
# Import the neccessary packages.
import pandas as pd
import numpy as np
                                                            In [ ]:
# Import the Warnings.
import warnings
warnings.filterwarnings('ignore')
                                                            In [ ]:
data = pd.read_csv('brain_stroke.csv')
                                                            In [ ]:
data.head()
                                                            In [ ]:
data.shape
                                                            In [ ]:
data.info()
                                                            In [ ]:
data.duplicated()
                                                            In [ ]:
data.duplicated().sum()
                                                            In [ ]:
df = data.dropna()
                                                            In [ ]:
df.head()
                                                            In [ ]:
df.describe()
                                                            In [ ]:
df.columns
                                                            In [ ]:
df['work_type'].unique()
                                                            In [ ]:
df['smoking_status'].unique()
                                                            In [ ]:
```

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

In [ ]:

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

In [ ]:

In [ ]:

Module 2: Exploration data analysis of visualization and training a model by given attributes

```
# Import the neccessary packages.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
# Import the warnings
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data = pd.read_csv('brain_stroke.csv')
```

In [ ]:

```
data.head()
```

In [ ]:

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

In [ ]:

```
pd.crosstab(df.stroke, df['age'])
```

In [ ]:

```
df['bmi'].hist(figsize= (7,8), color='green')
plt.title("Bmi distribution")
plt.xlabel("Bmi range")
plt.ylabel("Bmi weight")
```

In [ ]:

```
sns.distplot(x = df['age'], bins = 10)
```

In [ ]:

```
sns.barplot(x = df['gender'], y = df['stroke'])
```

In [ ]:

```
sns.boxplot(x = df['age'], y = df['gender'])
```

In [ ]:

```
sns.violinplot(y = df['avg_glucose_level'], x =df['stroke']
```

Module 3 : Performance measurements of Naive Bayes

```
# Import the Neccesary packages.
import pandas as pd
import numpy as np
```

In [ ]:

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

In [ ]:

```
data = pd.read_csv('brain_stroke.csv')
```

In [ ]:

```
data.head()
```

In [ ]:

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

In [ ]:

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

var = ['gender','smoking_status']

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

In [ ]:

```
df.columns
```

In [ ]:

```
df.head()
```

In [ ]:

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

In [ ]:

```
del df['ever_married']
del df['work_type']
del df['Residence_type']
```

In [ ]:

```
df.head()
```

In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='stroke', axis=1)
#Response variable
Y = df.loc[:,'stroke']
```

```
                                                                In [ ]:
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=1)
x_ros,y_ros=ros.fit_resample(X,Y)



print("OUR DATASET COUNT         : ", Counter(Y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
                                                                In [ ]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.30,
random_state=1, stratify=y_ros)
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,plot_confusion_matrix
from sklearn.naive_bayes import GaussianNB
                                                                In [ ]:
NB = GaussianNB()
NB.fit(x_train,y_train)
predictNb =NB.predict(x_test)
                                                                In [ ]:
accuracy = accuracy_score(y_test, predictNb)
print("Accuracy of naive bayes classifier:", accuracy*100)
                                                                In [ ]:
cr = classification_report(y_test, predictNb)
print("Classification report \n\n:", cr)
                                                                In [ ]:
cm = confusion_matrix(y_test, predictNb)
print("Confusion matrix:\n", cm)
                                                                In [ ]:
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(6,6))
plot_confusion_matrix(NB, x_test, y_test, ax=ax)
plt.title('Confusion matrix of Mlp classifier')
```

Module 4 : Implementing Decision Tree classifier

```
# Import the neccsary packages.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

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

In [ ]:

```
data = pd.read_csv('brain_stroke.csv')
```

In [ ]:

```
data.head()
```

In [ ]:

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

In [ ]:

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

var = ['gender','smoking_status']

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

In [ ]:

```
df.columns
```

In [ ]:

```
df.head()
```

In [ ]:

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

In [ ]:

```
del df['ever_married']
del df['work_type']
del df['Residence_type']
```

In [ ]:

```
df.head()
```

In [ ]:

```
#preprocessing, split test and dataset, split response variable
```

```
X = df.drop(labels='stroke', axis=1)
#Response variable
Y = df.loc[:,'stroke']
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=1)
x_ros,y_ros=ros.fit_resample(X,Y)
print("OUR DATASET COUNT         : ", Counter(Y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.30,
random_state=1, stratify=y_ros)
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 the DecisionTree classifier

In [ ]:

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

In [ ]:

```
Dt = DecisionTreeClassifier()
Dt.fit(x_train,y_train)
predictDt =Dt.predict(x_test)
```

Finding the accuracy of Decision Tree classifier.

In [ ]:

```
accuracy = accuracy_score(y_test, predictDt)
print("Accuracy of DecisionTree classifier:", accuracy *100)
```

Finding the classification Report Decision tree classifier.

In [ ]:

```
cr = classification_report(y_test, predictDt)
print("Classification report \n\n:", cr)
```

Finding the Confusion matrix Decision tree classifier.

In [ ]:

```
cm = confusion_matrix(y_test, predictDt)
```

```
print("Confusion matrix:\n", cm)
```

In [ ]:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(6,6))
plot_confusion_matrix(Dt, x_test, y_test, ax=ax)
plt.title('Confusion matrix of Decision Tree classifier')
plt.show()
```

In [ ]:

```
import joblib
joblib.dump(Dt,'Dt.pkl')
```

In [ ]:

In [ ]:

Module – 5 Implementing the MLP classifier

```
# Import the neccessary packages
import pandas as pd
import numpy as np
```

In [ ]:

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

In [ ]:

```
data = pd.read_csv('brain_stroke.csv')
```

In [ ]:

```
data.head()
```

In [ ]:

```
data.columns
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
del df['ever_married']
del df['work_type']
del df['Residence_type']
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()

var = ['gender','smoking_status']

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

In [ ]:

```
df.head()
```

In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='stroke', axis=1)
#Response variable
Y = df.loc[:,'stroke']
```

In [ ]:

```
!pip install imblearn
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=1)
x_ros,y_ros=ros.fit_resample(X,Y)
print("OUR DATASET COUNT         : ", Counter(Y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```

Split the dataset in training and testing

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.30,
random_state=1, stratify=y_ros)
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 the MLP classifier

In [ ]:

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

In [ ]:

```
MLP = MLPClassifier(random_state=1)
MLP.fit(x_train, y_train)
```

In [ ]:

```
predictMLP = MLP.predict(x_test)
```

Find the accuracy of MLP classifier

In [ ]:

```
accuracy = accuracy_score(y_test,predictMLP)
print("The accuracy of MLP classifier:" ,accuracy*100)
```

Find the classification report of MLP classifier.

In [ ]:

```
cr = classification_report(y_test, predictMLP)
print("Classification report \n\n:", cr)
```

Finding the Confusion matrix

In [ ]:

```
cm = confusion_matrix(y_test, predictMLP)
print("Confusion matrix:\n", cm)
```

In [ ]:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(6,6))
plot_confusion_matrix(MLP, x_test, y_test, ax=ax)
plt.title('Implementing the MLP classifier')
plt.show()
```

In [ ]:

In [ ]:

Module – 6 Implementing the Support vector machine.

```
# Implementing necessary packages.
import pandas as pd
import numpy as np
```

In [ ]:

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

In [ ]:

```
data = pd.read_csv('brain_stroke.csv')
```

In [ ]:

```
data.head()
```

In [ ]:
```
df = data.dropna()
```
In [ ]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['gender','smoking_status']

for i in var:
    df[i] = le.fit_transform(df[i]).astype(int)
```
In [ ]:
```
df.columns
```
In [ ]:
```
df.head()
```
In [ ]:
```
df['stroke'].unique()
```
In [ ]:
```
del df['ever_married']
del df['work_type']
del df['Residence_type']
```
In [ ]:
```
df.head()
```
In [ ]:
```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='stroke', axis=1)
#Response variable
Y = df.loc[:,'stroke']
```
In [ ]:
```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=1)
x_ros,y_ros=ros.fit_resample(X,Y)
print("OUR DATASET COUNT         : ", Counter(Y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```
In [ ]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.30,
random_state=1, stratify=y_ros)
```

```
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 Support vector machine

In [ ]:

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

In [ ]:

```
svc = SVC()
svc.fit(x_train,y_train)
predictSvc =svc.predict(x_test)
```

Finding the accuracy score

In [ ]:

```
accuracy = accuracy_score(y_test, predictSvc)
print("Accuracy of Support vector machine classifier:", accuracy*100)
```

Finding the classification Report

In [ ]:

```
cr = classification_report(y_test, predictSvc)
print("Classification report \n\n:", cr)
```

Finding the Confusion matrix

In [ ]:

```
cm = confusion_matrix(y_test, predictSvc)
print("Confusion matrix:\n", cm)
```

In [ ]:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(6,6))
plot_confusion_matrix(svc, x_test, y_test, ax=ax)
plt.title('Confusion matrix of Support vector machine')
plt.show()
```

CHAPTER 7

SYSTEM TESTING

7. SYSTEM TESTING

7.1 UNIT TESTING

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation.Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

7.2 INTEGRATION TESTING

It is defined to be set of interaction, all defined interaction among components need to be tested. The architecture and design can give the detail ofinteraction within the system and explain how they interact with each other. Thetypes of integration testing are Top-Down testing, Bottom-Up testing, Bi- Directional testing and System integration.

TEST CASE ID	INPUT	OUTPUT	PASS/FAIL
Test case 1	Age: 65 Gender: Male Blood pressure: 160/90 mmHg Cholesterol level: 220 mg/dL Smoking history: Non-smoker Outcome: No brainstroke	Brainstroke	pass
Test case 2	Age: 52 Gender: Female Blood pressure: 130/80 mmHg Cholesterollevel:180 mg/dL Smokinghistory: Smoker (10 pack-years)	No brainstroke	pass
Test case 3	Age: 72 Gender: Male Blood pressure: 140/85 mmHg	Brainstroke	pass

	Cholesterol level: 200 mg/dL Smoking history: Non-smoker		
Test case 4	Age: 45 Gender: Female Blood pressure: 120/70 mmHg Cholesterol level: 160 mg/dL Smoking history: Non-smoker	No brainstroke	pass

CHAPTER 8

CONCLUSION

8. CONCLUSION

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 strokes 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 Stroke 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.

FUTURE WORK

- The added background knowledge from other datasets can also possibly improve the accuracy of stroke prediction models as well.
- We intend to collect our institutional dataset for further benchmarking of these machine learning methods for stroke prediction.
- We also plan to perform external validation of our proposed method, as a part of our upcoming planned work.

APPENDIX

APPENDIX

OUTPUT SCREENSHOT

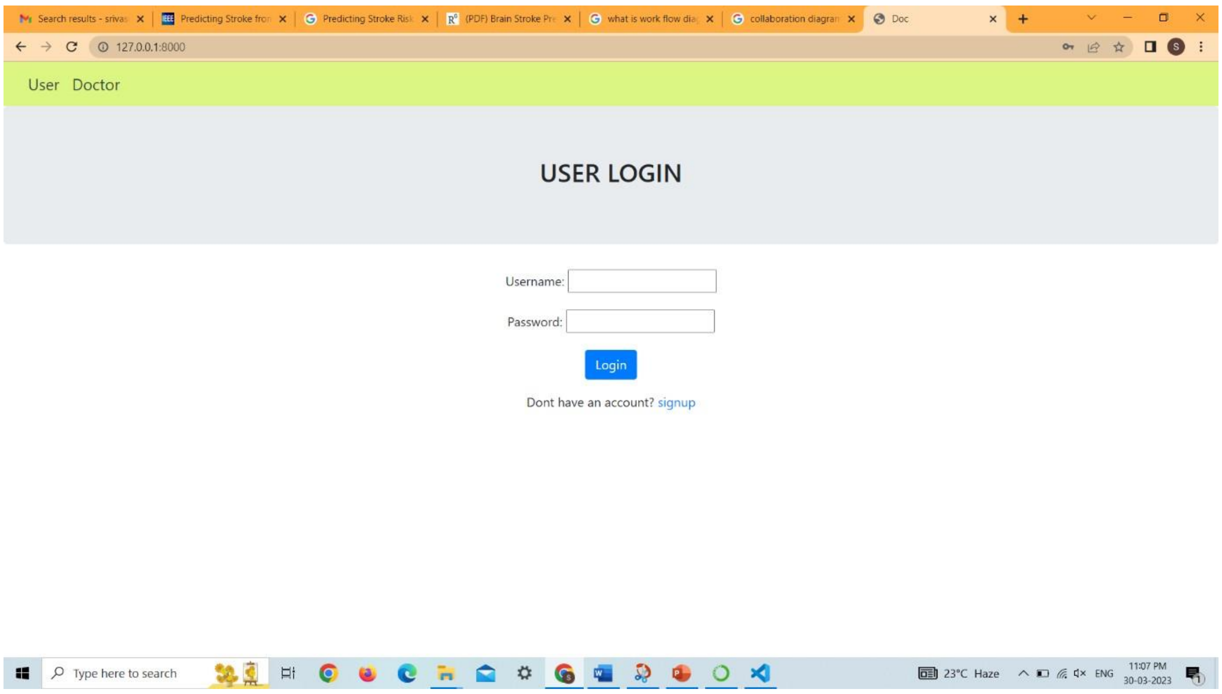


FIGURE A.1.1

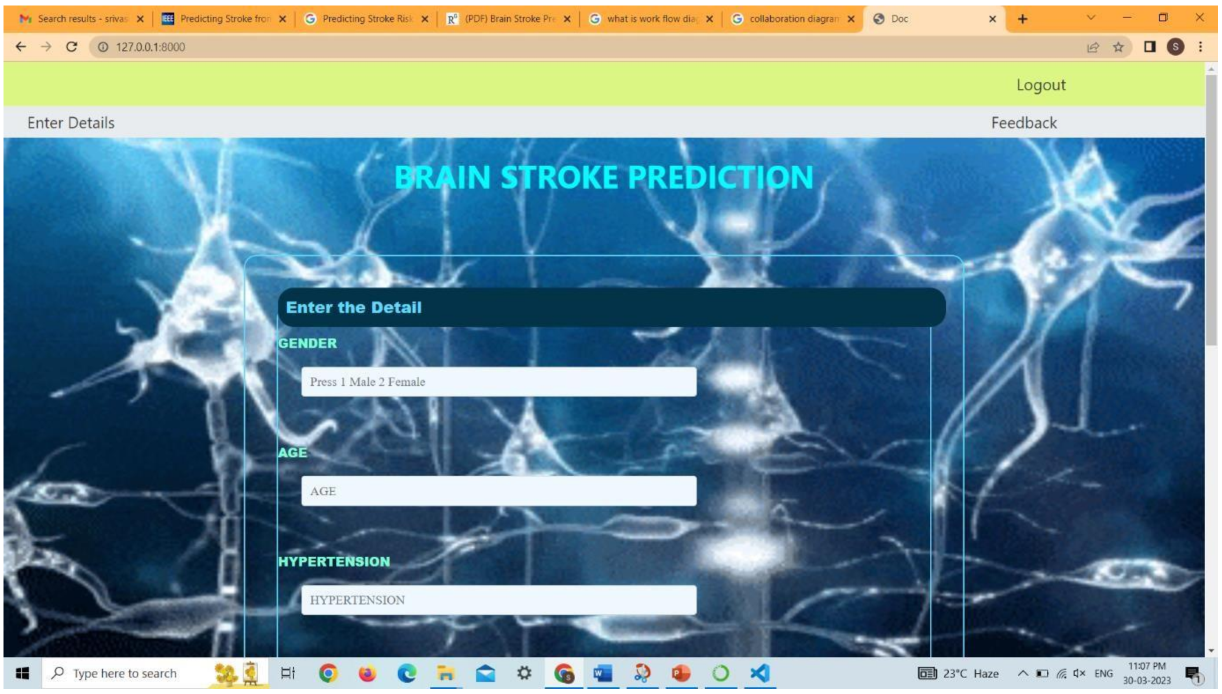
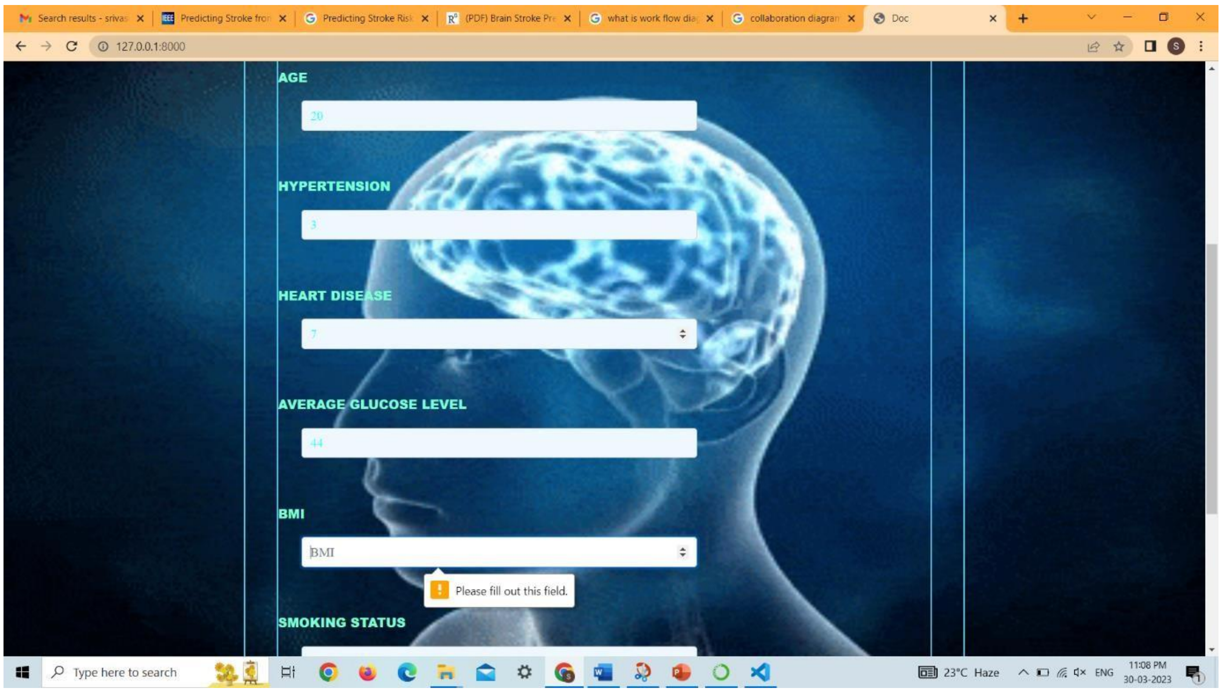


FIGURE A.1.2



AGE

20

HYPERTENSION

3

HEART DISEASE

7

AVERAGE GLUCOSE LEVEL

44

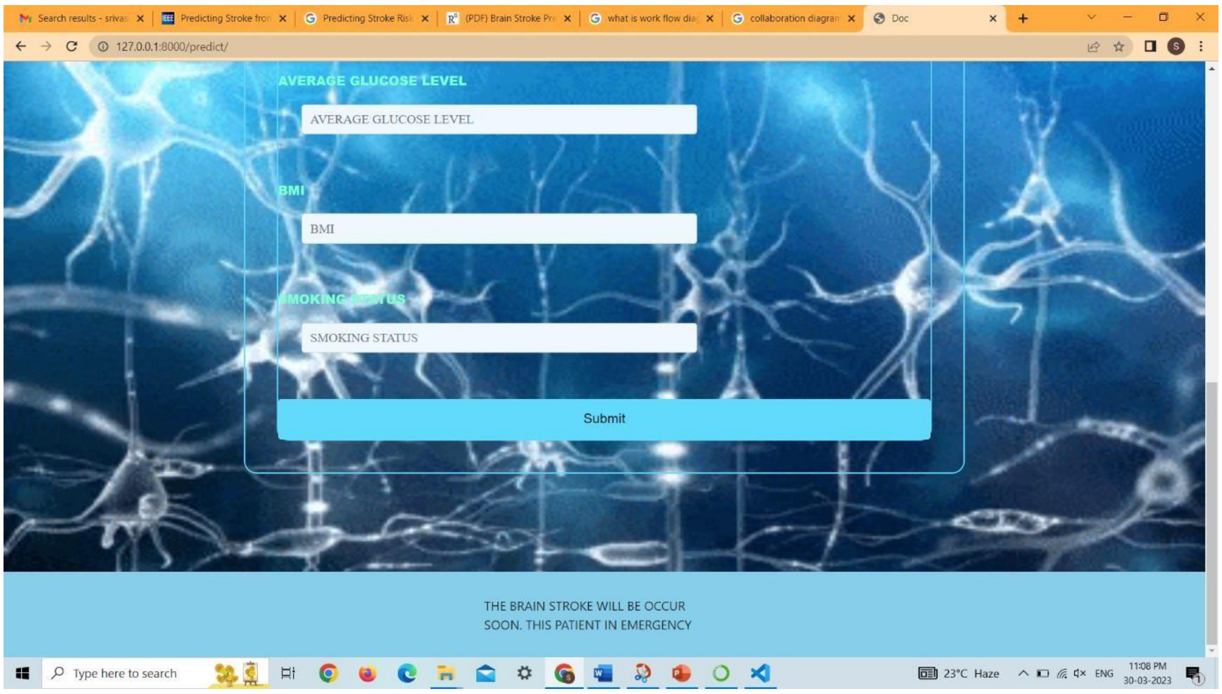
BMI

BMI

Please fill out this field.

SMOKING STATUS

FIGURE A.1.3



AVERAGE GLUCOSE LEVEL

AVERAGE GLUCOSE LEVEL

BMI

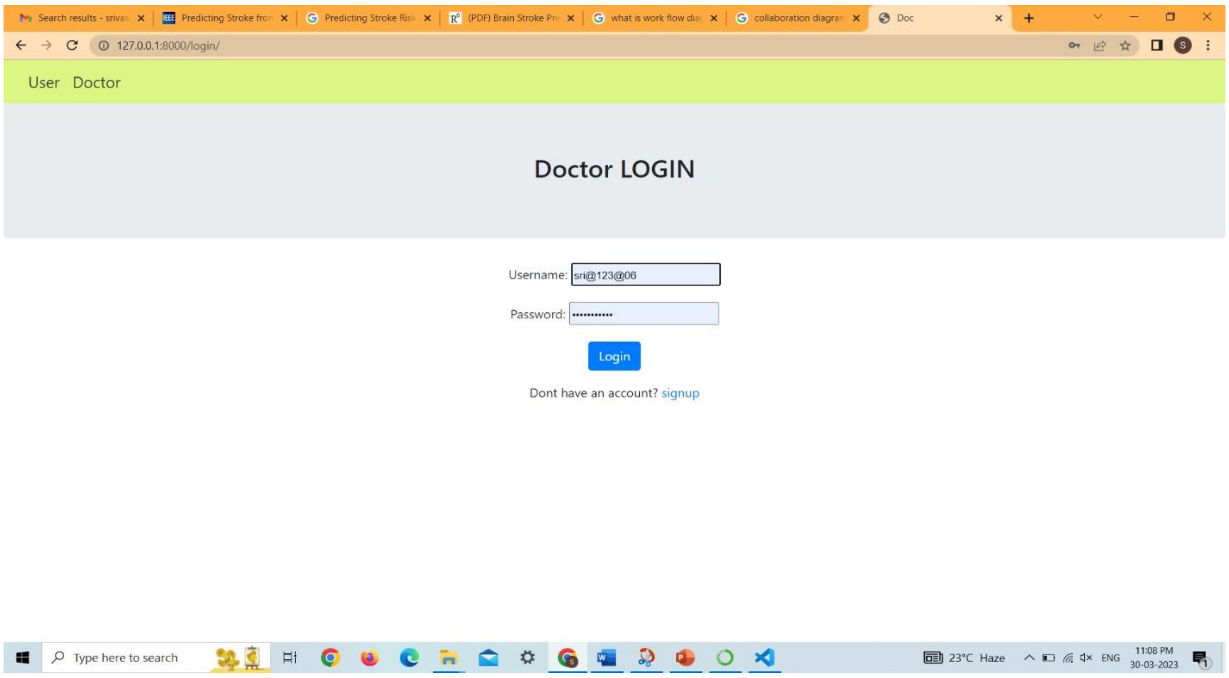
BMI

SMOKING STATUS

SMOKING STATUS

THE BRAIN STROKE WILL BE OCCUR
SOON. THIS PATIENT IN EMERGENCY

FIGURE A.1.4



Doctor LOGIN

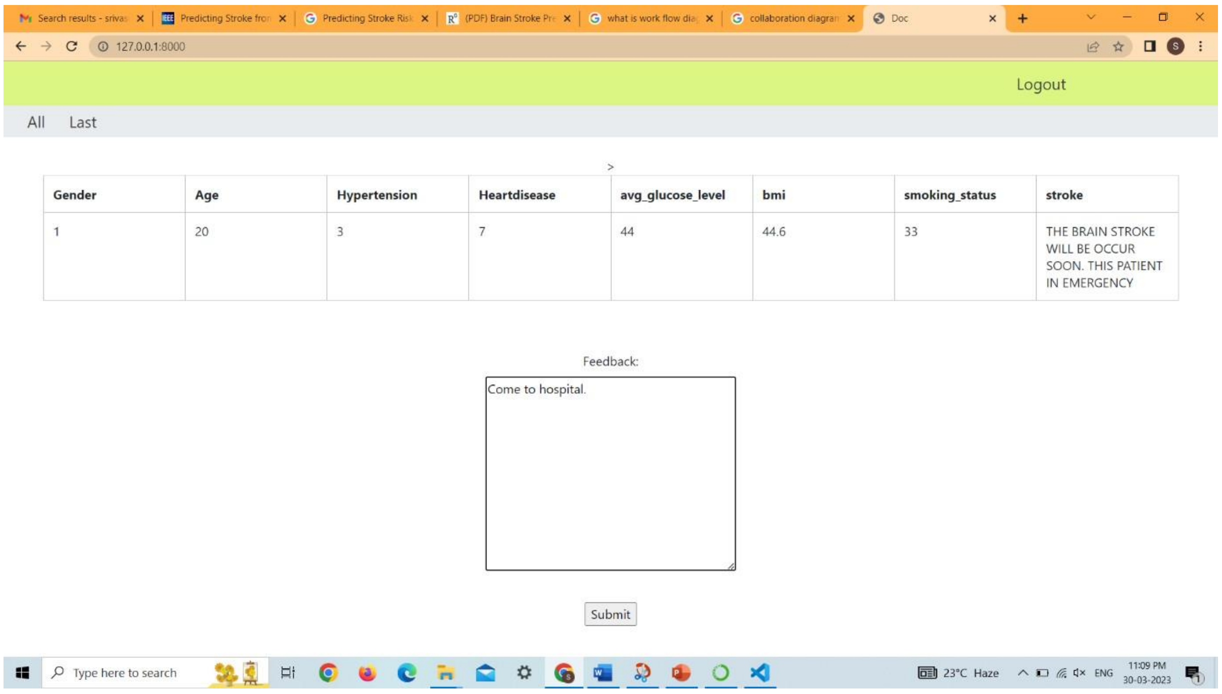
Username: sri@123@06

Password: ••••••••••

Login

Dont have an account? signup

FIGURE A.1.5



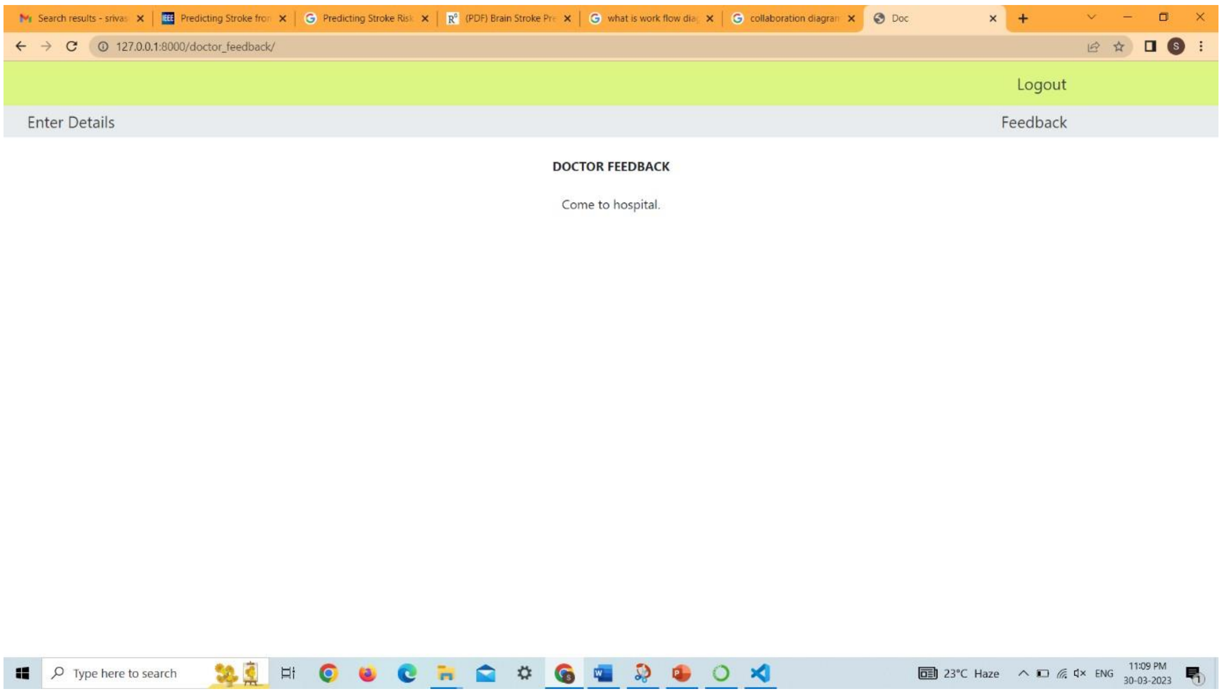
Gender	Age	Hypertension	Heartdisease	avg_glucose_level	bmi	smoking_status	stroke
1	20	3	7	44	44.6	33	THE BRAIN STROKE WILL BE OCCUR SOON. THIS PATIENT IN EMERGENCY

Feedback:

Come to hospital.

Submit

FIGURE A.1.6



DOCTOR FEEDBACK

Come to hospital.

FIGURE A.1.7

BIBILOGRAPHY

BIBILOGRAPHY

1. "Concept of stroke by healthline,"
   Available: https://www.cdc.gov/stroke/index.htm.

2. S. Y. Adam, A. Yousif, and M. B. Bashir, "Classification of ischemic stroke using machine learning algorithms," *International Journal of Computer Application*, vol. 149, no. 10, pp. 26–31, 2016.

3. P. Govindarajan, R. K. Soundarapandian, A. H. Gandomi, R. Patan, P. Jayaraman, and R. Manikandan, "Classification of stroke disease using machine learning algorithms," *Neural Computing & Applications*, vol. 32, no. 3, pp. 817–828, 2020.

4. L. Amini, R. Azarpazhouh, M. T. Farzadfar et al., "Prediction and control of stroke by data mining," *International Journal of Preventive Medicine*, vol. 4, no. 2, pp. S245–S249, 2013.

5. "Documentation for decision tree classification from scikit-learn,".

6. G. Sailasya and G. L. A. Kumari, "Analyzing the performance of stroke prediction using ML classification algorithms," *International Journal Of Advanced Computer Science And Applications*, vol. 12, no. 6, pp. 539–545, 2021.

7. "Prediction of ischemic stroke subtypes using machine learning algorithms" by D. K. Kumar, K. S. Sujatha, and P. S. Nair. Neural Computing and Applications, 2020.

8. "Prediction of stroke outcomes using machine learning algorithms: a systematic review" by M. Liang, Z. Zhang, and D. Sun. Neural Regeneration Research, 2020.

9. "Predicting early neurological deterioration after acute ischemic stroke using machine learning techniques" by C. Wang, Y. Li, and L. Li. Journal of Stroke and Cerebrovascular Diseases, 2021.

10. "A machine learning approach for the prediction of hemorrhagic stroke" by R. Sharifi and M. Khadem. Journal of Medical Systems, 2019.

11. "Machine learning-based prediction of hemorrhagic transformation in ischemic stroke" by L. Tang, Y. Leng, and J. Xiong. Journal of Stroke .and Cerebrovascular Diseases, 2020.