

# **CEREBRAL STROKE CLASSIFICATION USING SUPERVISED LEARNING APPROACHES**

**A PROJECT REPORT**

*Submitted by*

**VARSHASHREE A      [REGISTER NO: 211420104296]**

**KOMEL P                      [REGISTER NO:211420104137]**

*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)**

**MARCH 2024**

# **PANIMALAR ENGINEERING COLLEGE**

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

## **BONAFIDE CERTIFICATE**

Certified that this project report “**CEREBRAL STROKE CLASSIFICTION USING SUPERVISED LEARNING APPROACHES**” is the bonafide work of “**VARSHASHREE(211420104296) , KOMEL(211420104137)**” who carried out the project work under my supervision.

**SIGNATURE**

**Dr.L.JABASHEELA,M.E.,Ph.D.,**

**HEAD OF THE DEPARTMENT**

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

**SIGNATURE**

**Dr.K.VALARMATHI M.E.,Ph.D.,**

**PROFESSOR**

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

Certified that the above candidate(s) was examined in the End Semester Project

Viva-Voce Examination held on.....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **DECLARATION**

We **VARSHASHREE(211420104296)** , **KOMEL(211420104137)**, hereby declare that this project report titled **“CEREBRAL STROKE CLASSIFICATION USING SUPERVISED LEARNING APPROACHES”**, under the guidance of **Dr. K.VALARMATHI M.E,Ph.D.** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**VARSHASHREE A**

**KOMEL P**

## **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 guide **Dr.K.VALARMATHI M.E,Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

**VARSHASHREE A**

**KOMEL P**

## TABLE OF CONTENTS

CHAPTERNO	TITLE	PAGENO
	<b>ABSTRACT</b>	iii
	<b>LIST OF FIGURES</b>	iv
	<b>LIST OF TABLES</b>	v
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 Overview	2
	1.2 Problem Definition	3
<b>2</b>	<b>LITERATURE SURVEY</b>	4
<b>3</b>	<b>SYSTEM ANALYSIS</b>	
	3.1 Existing System	11
	3.2 Proposed System	11
	3.3 Project Requirements	13
<b>4</b>	<b>SYSTEM DESIGN</b>	
	4.1 UML Diagrams	22
	4.1.1 Use Case Diagram	22
	4.1.2 Activity Diagram	23
	4.1.3 Class Diagram	24
	4.2 Data Flow Diagram	25

<b>5</b>	<b>SYSTEM ARCHITECTURE</b>	
	5.1 System Architecture Overview	27
<b>6</b>	<b>SYSTEM IMPLEMENTATION</b>	
	6.1 Algorithm	29
	6.1.1 Random Forest Algorithm	29
	6.1.2 AdaBoost Classifier	31
	6.1.3 KNN Classifier	34
	6.2 Module Design Specification	37
	6.3 Module Description	37
<b>7</b>	<b>RESULTS AND DISCUSSIONS</b>	
	7.1 Unit Testing	42
	7.2 Test Cases	43
<b>8</b>	<b>CONCLUSION</b>	
	8.1 Conclusion	45
	8.2 Future Enhancements	45
<b>9</b>	<b>APPENDICES</b>	
	1. Screenshots	47
	2. Source Code	49
	3. Plagiarism Report	57
	<b>REFERENCES</b>	60

## **ABSTRACT**

Cerebral stroke is a life-threatening medical condition that requires immediate diagnosis and intervention to minimize its devastating effects. Timely and accurate classification of stroke subtypes is crucial for determining appropriate treatment strategies and improving patient outcomes. In this research, we explore the application of supervised learning approaches to classify cerebral stroke subtypes based on relevant medical data. The study involves the collection of a diverse dataset comprising anonymized patient records, including clinical indicators, medical history, risk factors, and diagnostic imaging results. The data is carefully pre-processed to handle missing values, normalize numerical features, and address any potential biases. The experimental results demonstrate the effectiveness of supervised learning approaches in accurately classifying cerebral stroke subtypes. The model with the highest performance metrics is identified, and its potential integration into clinical settings is discussed. Additionally, the research explores potential challenges and limitations of the proposed approach, along with potential strategies to improve model accuracy and generalizability.

## **LIST OF FIGURE**

<b>FIGURE NO.</b>	<b>FIGURE NAME</b>	<b>PAGE NO.</b>
Fig 4.1.1	Use case Diagram	22
Fig 4.1.2	Activity Diagram	23
Fig 4.1.3	Class Diagram	24
Fig 4.2.1	Level 0 DFD Diagram	25
Fig 4.2.2	Level 1 DFD Diagram	26
Fig 5.1	System Architecture	27
Fig 9.1.1	Screenshot of Signup Page	47
Fig 9.1.2	Screenshot of Home Page	47
Fig 9.1.3	Screenshot of Prediction From Page	48
Fig 9.2.4	Screenshot of Result Page	48
Fig 9.3	Screenshot of Plagiarism Report	57



# **CHAPTER-1**

## **INTRODUCTION**

# **CHAPTER-1**

## **INTRODUCTION**

### **1.1 OVERVIEW**

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).

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.

## **1.2 PROBLEM DEFINITION**

This research focuses on employing supervised learning techniques to classify cerebral stroke subtypes accurately, a critical aspect for timely treatment and improved patient outcomes. By assembling a diverse dataset of anonymized patient records containing clinical indicators, medical history, and diagnostic imaging results, the study aims to develop robust models capable of distinguishing between ischemic stroke, hemorrhagic stroke, and transient ischemic attack (TIA). Through meticulous data pre-processing to handle missing values and biases, followed by model training and evaluation, the research identifies the most effective model for stroke subtype classification. Discussions include the potential integration of this model into clinical settings to aid healthcare professionals in diagnosis. Additionally, the study explores challenges and strategies to enhance model accuracy and applicability, contributing to the advancement of stroke management practices.

# **CHAPTER-2**

## **LITERATURE SURVEY**

## **CHAPTER-2**

### **LITERATURE SURVEY**

**1. TITLE:** Brain Stroke Prediction Using Machine Learning Approach  
**YEAR:** 2022

**AUTHORS:** DR. AMOL K. KADAM , PRIYANKA AGARWAL

#### **ABSTRACT**

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 Logical Regression, Decision Tree Classification , Random Forest Classification,KNN and SVM for accurate prediction.

**2. TITLE:** Brain Stroke Prediction using Machine Learning  
**YEAR:** 2020

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

## **ABSTRACT**

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. 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

**YEAR:** 2020

**AUTHORS :** Manisha Sanjay Sirsat, Eduardo Ferme, and Joana Camara

## **ABSTRACT**

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

**YEAR:** 2021

**AUTHORS:** Sergio peñafiel, Nelson baloian, Horacio sanson, and José a. pin

## **ABSTRACT**

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

**YEAR:** 2020

**AUTHORS:** Douglas Teoh



## ABSTRACT

**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**

## **CHAPTER-3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

Neural-like P systems are membrane computing models inspired by natural computing and are viewed as third-generation neural network models. Although real neurons have complex structures, classical neural-like P systems simplify the structures and corresponding mechanisms to two-dimensional graphs or tree-based firing and forgetting communications, which limit the real applications of these models. In this paper, we propose a hypergraph-based numerical neural-like (HNN) P system containing five types of neurons to describe the high-order correlations among neuron structures. Three new kinds of communication mechanisms among neurons are also proposed to address numerical variables and functions. Based on the new neural-like P system, a tumor/organ segmentation model for medical images is developed. The experimental results indicate that the proposed models outperform the state-of-the-art methods based on two hippocampal datasets and a multiple brain metastases dataset, thus verifying the effectiveness of the HNN P system in correctly segmenting tumors/organs.

#### **DEMERITS**

- They did not implement the deployment process.
- They only disease segmentation.
- Accuracy & performance was low
- They did not using machine learning techniques.

#### **3.2 PROPOSED WORK**

The proposed system for cerebral stroke classification using supervised learning approaches aims to enhance the accuracy and efficiency of stroke

diagnosis through advanced machine learning techniques. Leveraging the power of supervised learning algorithms, such as Support Vector Machines (SVM), Random Forest, or Neural Networks, the system will be trained on a diverse dataset of medical images and patient data. This comprehensive approach allows the model to learn patterns and relationships within the data, enabling it to accurately classify different types of cerebral strokes based on imaging features, clinical indicators, and other relevant parameters. By harnessing the capabilities of supervised learning, the system can provide timely and reliable predictions, aiding healthcare professionals in making quicker and more informed decisions for effective stroke management. Moreover, the proposed system will contribute to the ongoing efforts in leveraging artificial intelligence for medical diagnosis and treatment planning. The incorporation of supervised learning not only enhances the accuracy of stroke classification but also enables continuous learning and adaptation as new data becomes available. This adaptive nature ensures that the system stays up-to-date with the latest medical knowledge and can effectively handle variations in stroke presentations. Ultimately, the implementation of this system has the potential to significantly improve the speed and precision of stroke diagnosis, leading to better patient outcomes and more efficient allocation of healthcare resources.

## **MERITS**

- We build framework based application for deployment purposes.
- We using machine learning techniques to build a predictive model.
- Accuracy & performance level improved.
- We implemented the Data analysis process by using histograms, Plots, and Graphs.
- We compared more than two algorithms to get a better accuracy level

### **3.3 PROJECT REQUIREMENTS**

#### **General**

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

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
  - A. Hardware requirements
  - B. software requirements

#### **1. Functional requirements**

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

## **2. Non-Functional Requirements**

Process of functional steps,

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

## **3.Environmental Requirements**

### **A.Hardware system configuration**

1. Processor : Pentium IV/III
2. Hard disk : minimum 100 GB
3. RAM : minimum 4 GB

### **B.Software system configuration**

1. Operating System : Windows
2. Tool : Anaconda with Jupyter Notebook

## **SOFTWARE DESCRIPTION**

### **INTRODUCTION TO ANACONDA**

Anaconda is a free and open-source 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. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment

manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

## **ANACONDA NAVIGATOR**

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

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

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

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

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

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:

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

## **Conda**

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for



you. It consists of many softwares which will help you to build your machine learning project and deep learning project. these softwares have great graphical user interface and these will make your work easy to do. you can also use it to run your python script. These are the software carried by anaconda navigator.

## **JUPYTER NOTEBOOK**

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

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

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

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

## **Running the Jupyter Notebook**

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

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

When started, the Jupyter Notebook App can access only files within its

start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

**Save notebooks** Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

**Executing a notebook** Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- Launch the jupyter notebook app
- In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- Click on the menu Help -> User Interface Tour for an overview of the Jupyter Notebook App user interface.
- You can run the notebook document step-by-step (one cell a time) by pressing shift + enter.
- You can run the whole notebook in a single step by clicking on the menu Cell -> Run All.
- To restart the kernel (i.e. the computational engine), click on the menu Kernel -> Restart. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...).

## **JUPYTER Notebook App**

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser.

The Jupyter Notebook App can be executed on a local desktop requiring

no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

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.

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

When you open a Notebook document, the associated kernel is automatically launched . When the notebook is executed the kernel performs the computation and produces the results.

**Notebook Dashboard** The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

## WORKING PROCESS

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

## PANDAS

### 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.

# **CHAPTER-4**

## **SYSTEM DESIGN**

## CHAPTER-4

### SYSTEM DESIGN

#### 4.1 UML DIAGRAMS

Unified Modeling Language (UML) is a general purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

##### 4.1.1 USE CASE DIAGRAM

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

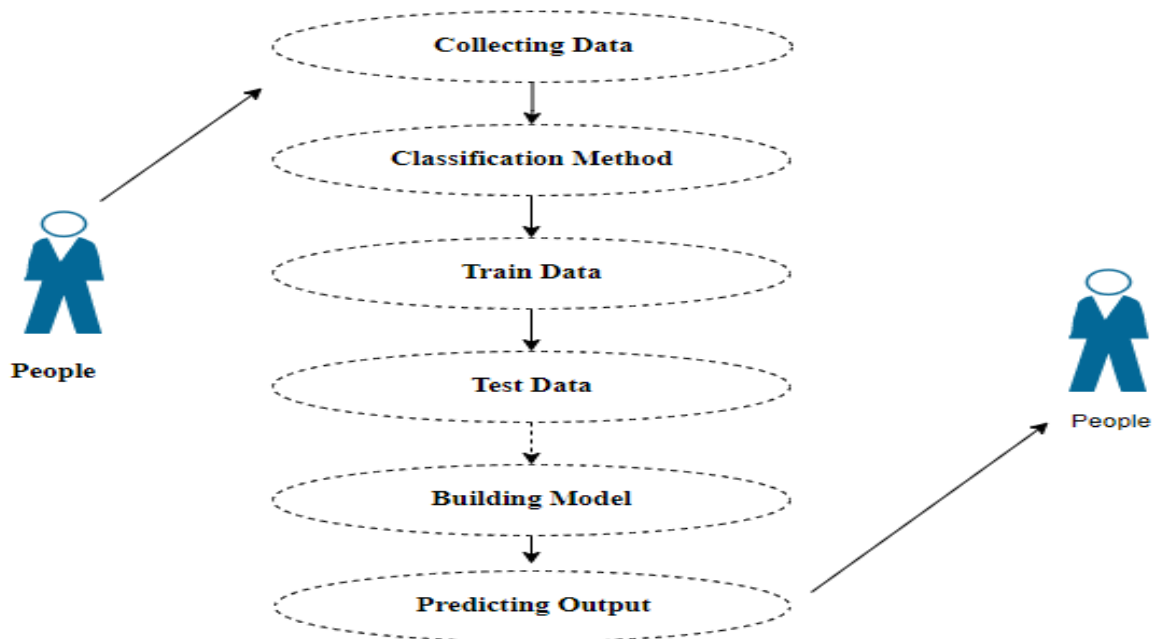


Figure 4.1.1 Use Case Diagram

## 4.1.2 ACTIVITY DIAGRAM

A graphical representation of an executed set of procedural system activities and considered a state chart diagram variation. Activity diagrams describe parallel and conditional activities, use cases and system functions at a detailed level.

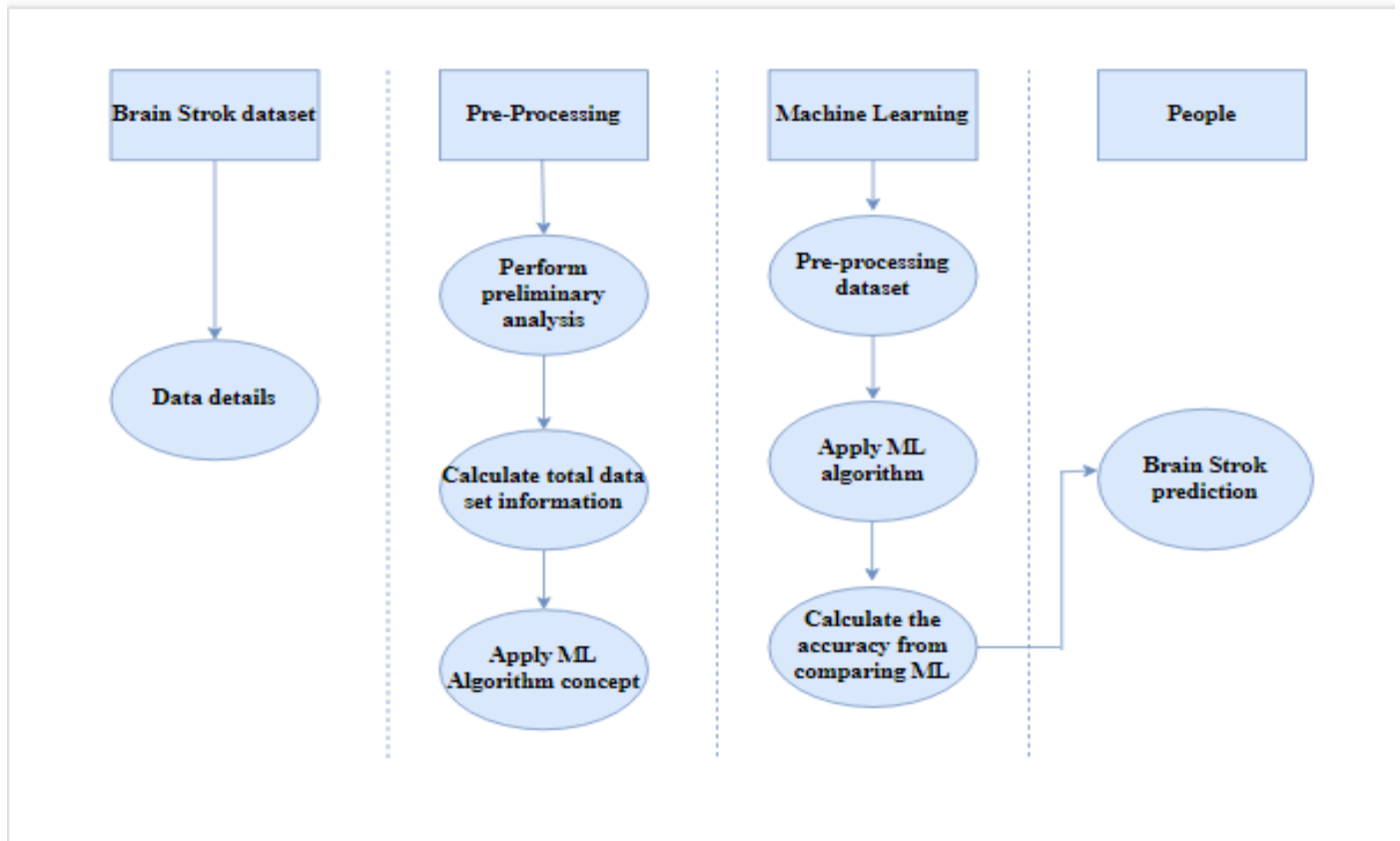


Figure 4.1.2 Activity Diagram

### 4.1.3 CLASS DIAGRAM

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance.

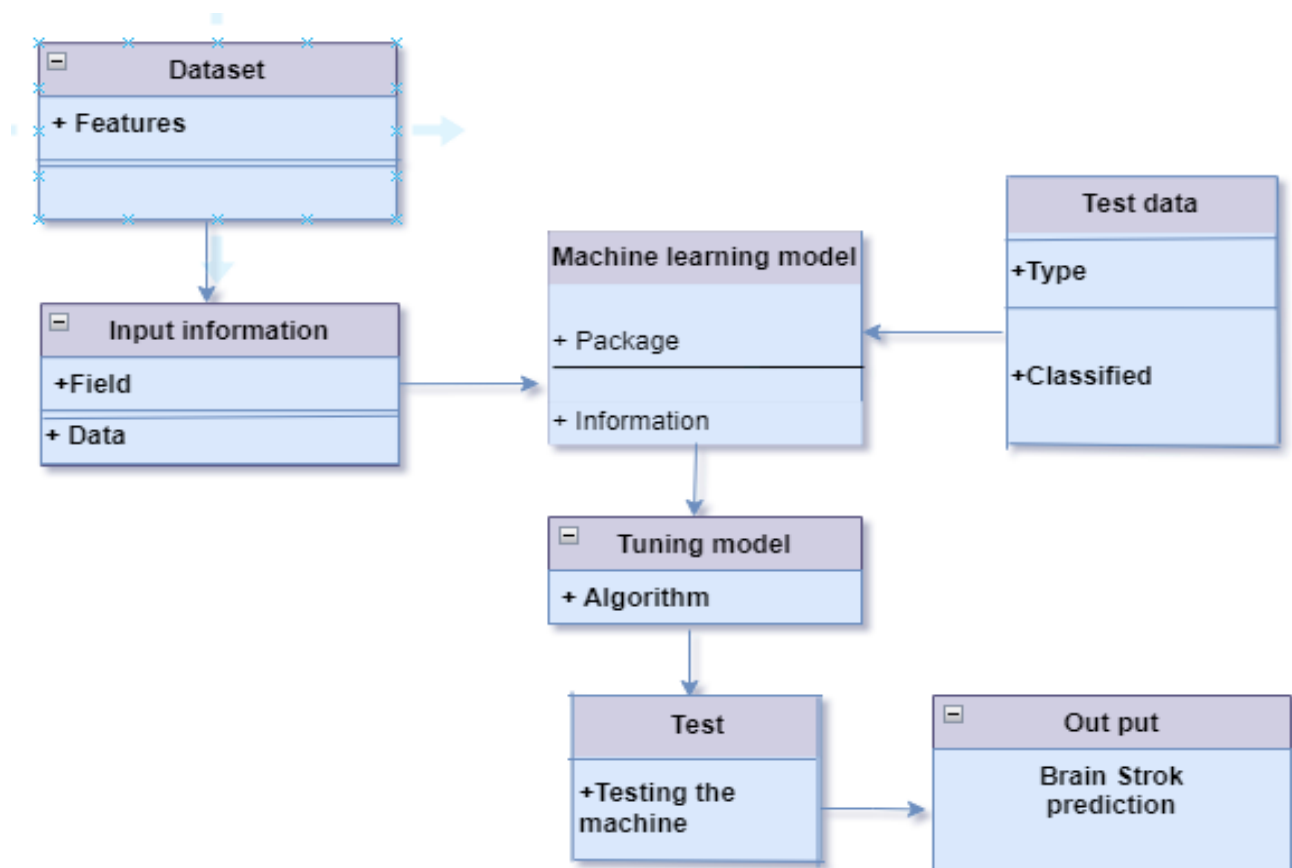


Figure 4.1.3 Class Diagram



## 4.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. It can be used for the visualization of data processing (structured design). Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. DFD Level 0 is also called a Context Diagram.

Level 0

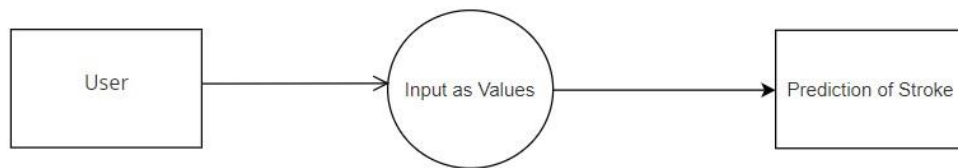


Figure 4.2.1 Level 0 DFD Diagram

Level 1:

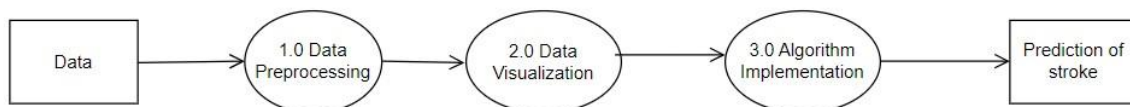


Figure 4.2.1 Level 1 DFD Diagram

# **CHAPTER-5**

## **SYSTEM ARCHITECTURE**

## CHAPTER-5

### SYSTEM ARCHITECTURE

#### 5.1 SYSTEM ARCHITECTURE OVERVIEW

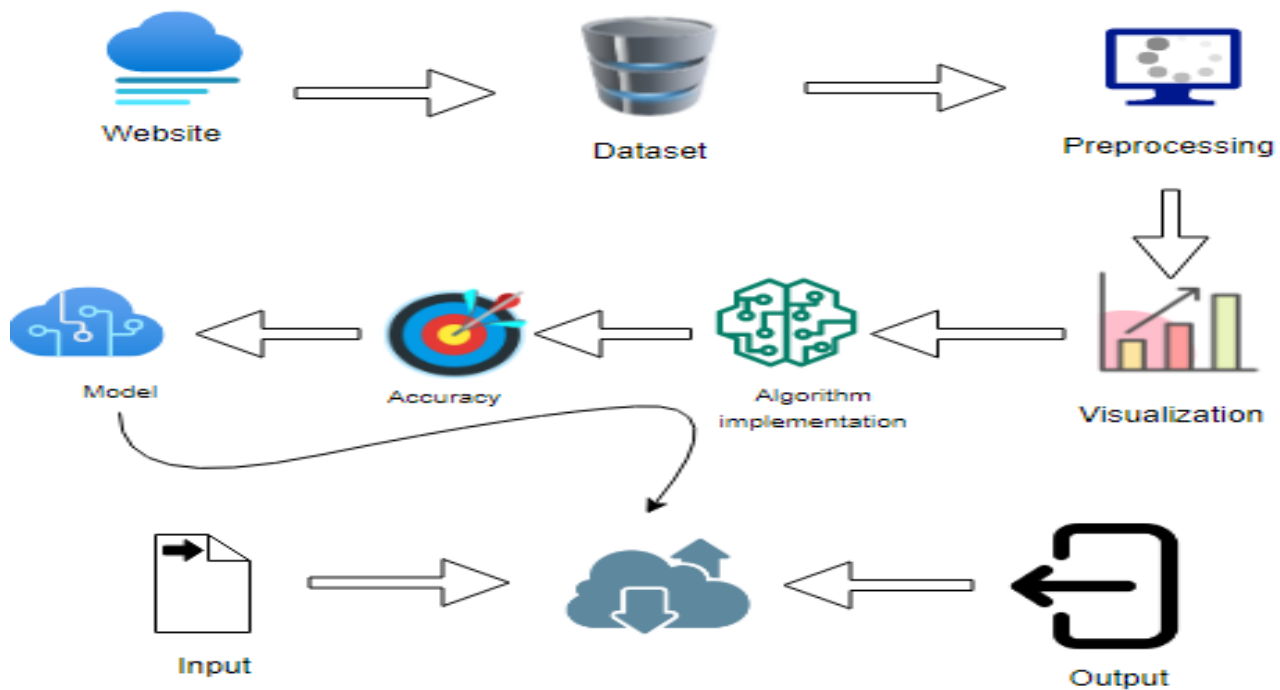


Figure 5.1 System Architecture

The architecture diagram shows the processes involved for building the project. It involves collecting dataset from website, the processing it to remove the noisy data, visualizing it and then implementing algorithms and finding the best model based on accuracy and then deploying it in the form of webpage.

# **CHAPTER-6**

## **SYSTEM IMPLEMENTATION**

## CHAPTER-6

### SYSTEM IMPLEMENTATION

#### 6.1 ALGORITHMS

1. Random Forest Algorithm
2. Adaboost Classifier
3. KNN

##### 6.1.1 RANDOM FOREST ALGORITHM

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

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

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

#### 1. Decision Trees

Random Forest is built upon the foundation of decision trees. Decision trees are simple models that recursively split the data based on features to make predictions..

#### 2. Ensemble Learning

Random Forest is an ensemble learning method that combines the predictions

of multiple decision trees to improve overall performance and robustness.

### **3. Bagging (Bootstrap Aggregating)**

Random Forest uses bagging to build diverse trees. It randomly selects subsets of the training data (with replacement) to train each tree. This helps in reducing overfitting and capturing different patterns in the data.

### **4. Random Feature Selection**

In addition to using random subsets of data, Random Forest also randomly selects a subset of features for each split in a tree. This introduces further randomness, prevents overfitting, and ensures that each tree focuses on different aspects of the data.

### **5. Decision Tree Training**

Each decision tree in the Random Forest is trained independently on its subset of data. The training process involves recursively splitting the data based on features, considering the best split at each node according to a specified criterion (commonly Gini impurity or information gain).

### **6. Voting (Classification)**

For classification tasks, each tree in the forest "votes" for a class. The class with the most votes becomes the final prediction of the Random Forest.

### **7. Averaging (Regression)**

For regression tasks, each tree predicts a numerical value, and the final prediction is the average of these values.

### **8. Out-of-Bag (OOB) Score**

Because each tree is trained on a subset of the data, some data points are not included (out-of-bag) in the training of certain trees. The OOB score is calculated by evaluating each tree on the data it did not see during training. This provides an estimate of the model's performance without the need for a separate validation set.

## **9. Hyperparameter Tuning**

Random Forest has hyperparameters such as the number of trees, the depth of each tree, and the number of features considered at each split. Proper hyperparameter tuning is crucial for achieving optimal performance.

## **10. Feature Importance**

Random Forest provides a feature importance score, indicating the contribution of each feature to the model's predictions. This information is valuable for feature selection and understanding the impact of variables on the model.

## **11. Parallel Training**

Random Forest is suitable for parallel processing, as each tree can be trained independently. This makes it computationally efficient and scalable.

### **6.1.2 ADABOOST CLASSIFIER**

This method operates iteratively, identifying misclassified data points and adjusting their weights to minimize the training error. The model continues to optimize sequentially until it yields the strongest predictor.

AdaBoost is implemented by combining several weak learners into a single strong learner. The weak learners in AdaBoost take into account a single input feature and draw out a single split decision tree called the decision stump. Each observation is weighted equally while drawing out the first decision stump.

The results from the first decision stump are analyzed, and if any observations are wrongfully classified, they are assigned higher weights. A new decision stump is drawn by considering the higher-weight observations as more significant. Again if any observations are misclassified, they're given a higher weight, and this process continues until all the observations fall into the right class.

#### **1. Weak Learners**

Adaboost starts by training a weak learner on the entire dataset. A weak learner is a model that performs slightly better than random chance. Common weak learners are decision stumps (simple decision trees with a single split).

## **2. Weighted Data**

Each instance in the training dataset is assigned an initial weight. Initially, all weights are set equally, so each instance has the same importance.

## **3. Training the Weak Learner**

The first weak learner is trained on the original data using the initial weights. It produces a model and predicts the target variable.

## **4. Compute Error**

The algorithm computes the error of the weak learner by comparing its predictions to the actual targets. Instances that are misclassified receive higher weights.

## **5. Compute Learner Weight**

The weight of the weak learner is calculated based on its classification error. The lower the error, the higher the weight assigned to the weak learner.

## **6. Update Instance Weights**

The instance weights are updated based on whether they were correctly or incorrectly classified by the weak learner. Misclassified instances receive higher weights.

## **7. Repeat**

Steps 3-6 are repeated for a predefined number of iterations or until a perfect model is achieved.

## **8. Final Model**

The final model is an ensemble of weak learners, each contributing to the final prediction based on their individual weights.



## **9. Weighted Voting**

During prediction, each weak learner "votes" on the class of a new instance. The weight of each learner influences its contribution to the final prediction.

## **10. Adaptive Learning**

Adaboost is adaptive, meaning it focuses more on instances that are difficult to classify correctly. As the algorithm progresses, it assigns higher weights to misclassified instances, forcing subsequent weak learners to focus on these challenging cases.

## **11. Combining Weak Models**

The final prediction is a weighted sum of the weak learners' predictions. The weights are determined by the accuracy of each weak learner on the training data.

## **12. Hyperparameters**

Adaboost has hyperparameters such as the number of weak learners (base estimators), the learning rate (which controls the contribution of each weak learner), and the choice of the weak learner.

## **13. Avoiding Overfitting**

Adaboost is less prone to overfitting compared to some other algorithms. The focus on misclassified instances and the adaptive nature of the algorithm help generalize well to unseen data.

## **14. Applications**

Adaboost is widely used in practice for binary classification problems. It is effective in scenarios where weak models can be combined to create a strong and accurate predictor.

### **6.1.3 KNN CLASSIFIER**

K-Nearest Neighbour is one of the simplest Machine Learning algorithms

based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

## **1. Overview**

KNN is a non-parametric, lazy learning algorithm that makes predictions by finding the majority class of the k-nearest data points in the feature space.

## **2. Distance Metric**

KNN relies on a distance metric (e.g., Euclidean distance, Manhattan distance) to measure the similarity between data points in the feature space. The choice of distance metric depends on the nature of the data.

## **3. Training**

KNN is an instance-based algorithm, meaning it memorizes the entire

training dataset. The training process involves storing the feature vectors and corresponding class labels.

#### **4. Prediction**

To make a prediction for a new instance, KNN calculates the distance between the instance and all data points in the training set.

#### **5. Nearest Neighbors**

The algorithm identifies the k-nearest neighbors of the new instance based on the computed distances. The value of k is a user-defined hyperparameter.

#### **6. Majority Voting**

For classification, KNN uses majority voting among the k-nearest neighbors to determine the predicted class of the new instance. The class with the most occurrences among the neighbors is assigned.

#### **7. Regression**

For regression tasks, the algorithm takes the average of the target values of the k-nearest neighbors as the predicted output.

#### **8. Hyperparameter Tuning**

The choice of the hyperparameter k is critical. A smaller k value may lead to noisy predictions, while a larger k value may smooth the decision boundaries but could ignore local patterns. Cross-validation is often used to find the optimal value of k.

#### **9. Feature Scaling**

KNN is sensitive to the scale of features, as features with larger scales can dominate the distance calculations. Therefore, it is common to scale or normalize features before applying KNN.

#### **10. Decision Boundaries**

KNN tends to have non-linear decision boundaries that can adapt to the shape of the data. The decision boundaries are formed by the regions where the majority class changes.

### **11. Curse of Dimensionality**

KNN's performance may degrade in high-dimensional spaces due to the curse of dimensionality. In high-dimensional spaces, instances become more sparse, making it harder to find meaningful nearest neighbors.

### **12. Computational Complexity**

The prediction step in KNN involves calculating distances for each test instance against all training instances. As a result, the algorithm can be computationally expensive for large datasets.

### **13. Applications**

KNN is used in various applications, including image recognition, text mining, and recommendation systems. It is particularly useful when the decision boundaries are complex and non-linear. Each decision tree in the forest is built using a random subset of the training data and a random subset of the input features. This helps to reduce overfitting and increase the diversity of the trees in the forest.

## **6.2 MODULE DESIGN SPECIFICATION**

1. Data collection
2. Data preprocessing
3. Data Validation/Cleaning/Preparing Process
4. Data Visualization
5. Evaluation Model

## **6.3 MODULE DESCRIPTION**

### **6.3.1 DATA COLLECTION**

The data collection process involved gathering anonymized patient records from diverse sources, encompassing various demographics and medical backgrounds. Patient records included comprehensive clinical indicators such as blood pressure, cholesterol levels, glucose levels, and neurological assessments. Medical history data, including previous strokes, cardiovascular diseases, diabetes, and other relevant conditions, were incorporated to provide a holistic view of each patient's health status. Risk factors such as smoking history, alcohol consumption, family history of strokes, and medication usage were also documented to assess their potential impact on stroke subtypes. Data collection protocols ensured adherence to ethical standards and patient privacy regulations, with all identifiable information removed or encrypted to maintain anonymity.

### **6.2.2 DATA PREPROCESSING**

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.

### **6.2.3 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.

### **6.2.3 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.

### 6.2.3 EVALUATION MODEL

Here , the machine is trained using classifier algorithms such as Random forest algorithm , Adaboost classifier , KNN classifier. Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Performance of each classification model is estimated base on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs. **Accuracy** is the Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

# **CHAPTER-7**

## **RESULTS AND DISCUSSIONS**



## CHAPTER-7

### 7.TESTING

To keep the system error free during the phases of development and during the time when new features are added, the following testing strategies are applied:

#### 7.1.Unit Testing

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

TEST CASE ID	SCENARIO	TEST CASE	PRECONDITION	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	Diverse dataset of medical images and patient data	Initial Model Performance Evaluation	A set of 100 labeled medical images of cerebral strokes along with corresponding patient data	Evaluation of the model's performance metrics	The SVM model demonstrates promising performance metrics, indicating its potential for accurate cerebral stroke classification	PASS
2	Assess the robustness of the trained Random Forest model for cerebral stroke classification	Model Robustness Testing	Additional dataset of 50 labeled medical images and patient data, including variations in imaging features and clinical indicators	Evaluation of the model's performance on unseen data	The Random Forest model consistently and accurately classifies the new dataset, showcasing its robustness against	PASS

					variations in cerebral stroke presentations	
3	Deploy the Neural Network model in a real-world clinical setting for practical evaluation	Deployment and Real-world Performance	continuous stream of medical images and patient data obtained from patients presenting with suspected cerebral strokes	Timely and reliable predictions for effective stroke management	The Neural Network model offers timely and precise predictions, assisting healthcare professionals in informed decision-making for effective stroke management	PASS

# **CHAPTER-8**

# **CONCLUSION**

## **8.1 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.

## **8.2 FUTURE ENHANCEMENT**

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.

# **APPENDICES**

## CHAPTER-9 APPENDICES

### 9.1 SAMPLE SCREENSHOTS

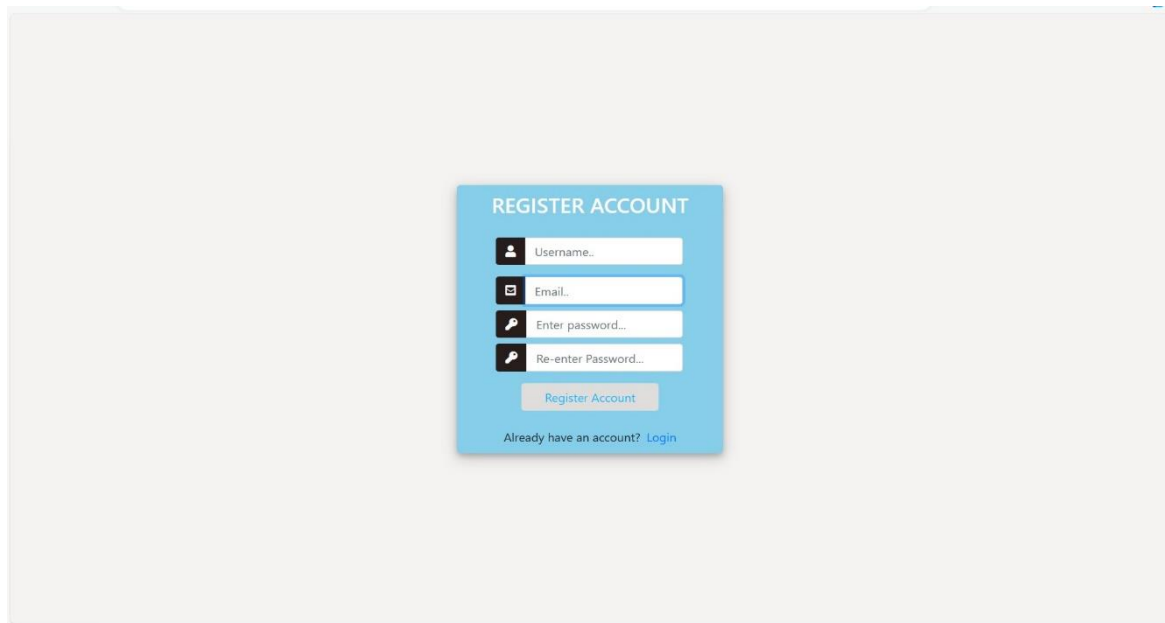


Figure 9.1.1 Screenshot of Signup Page

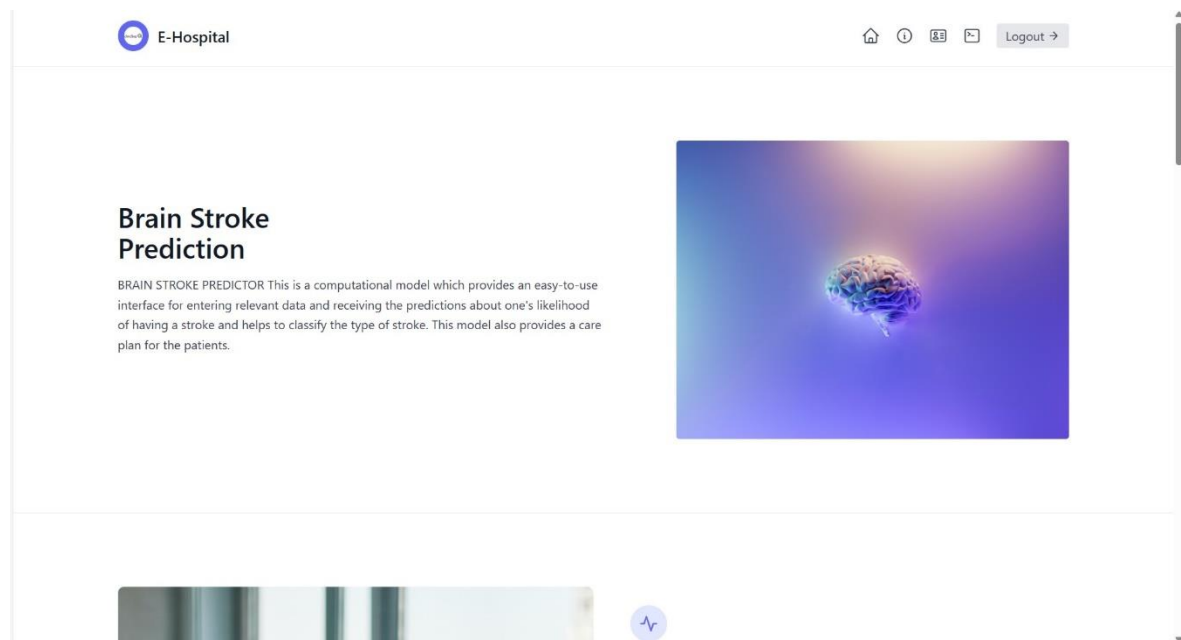



Figure 9.1.2 Screenshot of Home Page

<b>AGE</b>		<b>LDL</b>	
<input type="text" value="Please enter your age"/>		<input type="text" value="Please enter LDL value"/>	
<b>CHOLESTROL</b>		<b>SYSTOLIC</b>	
<input type="text" value="Please enter your Cholesterol"/>		<input type="text" value="Please enter systolic value"/>	
<b>GLUCOSE LEVEL</b>		<b>DIASTOLIC</b>	
<input type="text" value="Please enter your Glucose level"/>		<input type="text" value="Please enter dialostic value"/>	
<b>HDL</b>		<b>BMI</b>	
<input type="text" value="Please enter HDL value"/>		<input type="text" value="Please enter bmi value"/>	
<b>GENDER</b>	<b>SMOKING STATUS</b>	<b>ALCOHOL STATUS</b>	<b>STROKE</b>
<input type="text" value="Male"/>	<input type="text" value="Formerly smoked"/>	<input type="text" value="Formerly Drinks"/>	<input type="text" value="Yes"/>
<b>HEART DISEASE</b>			
<input type="text" value="Yes"/>			
<b>HYPER TENSION</b>			
<input type="text" value="Yes"/>			
<b>FAMILY HISTORY</b>			
<input type="text" value="Yes"/>			
<b>DIABETS</b>			
<input type="text" value="Yes"/>			

Figure 9.1.3 Screenshot of Prediction Form Page


E-Hospital

Home
Info
Help
Logout →

## Symptoms for hemorrhagic stroke

- Sudden, severe headache near the back of the head. Many people have described this as the “worst headache of your life.”
- Losing consciousness
- Inability to move or feel
- Confusion and irritability
- Muscle pain in neck and shoulders
- Sensitivity to light
- Seizure
- Vision problems
- One eye pupil larger than the other
- Nausea and vomiting

### Care plan for hemorrhagic stroke




TREATMENT	PROCEEDURE	
 <span>Medications</span>	<ul style="list-style-type: none"> <li>• Blood pressure medication with nicardipine drip to lower blood pressure to 140/90 mmHg or lower</li> <li>• Anti-seizure medication with levetiracetam (500mg twice daily) to prevent seizures.</li> <li>• Tranexamic acid (1g IV every 6 hours for 24 hours) to control bleeding.</li> <li>• Vitamin K antagonist (warfarin) should be stopped if present, if not then no anticoagulation should be given.</li> <li>• Antiemetic medication with ondansetron (4mg three times daily) to prevent nausea and vomiting.</li> </ul>	<b>important</b>
 <span>Rehabilitation</span>	<ul style="list-style-type: none"> <li>• Physical therapy 3 times per week for 12 weeks to improve mobility and strength .</li> <li>• Occupational therapy 2 times per week for 12 weeks to relearn daily living skills</li> <li>• Speech therapy 2 times per week for 12 weeks to improve communication skills.</li> </ul>	<b>must</b>
 <span>Diagnosis</span>	<ul style="list-style-type: none"> <li>• CT scan or MRI to confirm the diagnosis and identify the location and extent of the hemorrhagic stroke.</li> </ul>	<b>must</b>

Figure 9.1.4 Screenshot of Result Page

## 9.2 SOURCE CODING

### Data validation and pre-processing technique

```
# Import the neccessary packages.
```

```
import pandas as pd
```

```
import numpy as np
```

```
import warnings
```

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

```
In [ ]:
```

```
df = pd.read_csv('CEREBRAL.csv')
```

```
del df['id']
```

```
df.head()
```

```
In [ ]:
```

```
df.tail()
```

```
In [ ]:
```

```
df.shape
```

```
In [ ]:
```

```
df.size
```

```
In [ ]:
```

```
df.columns
```

```
In [ ]:
```

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

```
le = LabelEncoder()
```

```
var = ['gender','ever_married','work_type','Residence_type','smoking_status']
```

```
for i in var:
```

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

```
In [ ]:
```

```
df.isnull()
```

```
In [ ]:
```

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

```
In [ ]:
```

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

```
In [ ]:
```

```
df.describe()
```

```
In [ ]:
```

```
df.corr()
```

```
In [ ]:
```

```
df.info()
```



```

In [ ]:
pd.crosstab(df["Residence_type"], df["avg_glucose_level"])
In [ ]:
df.groupby(["heart_disease", "hypertension"]).groups
In [ ]:
df["stroke"].value_counts()
In [ ]:
pd.Categorical(df["gender"]).describe()
In [ ]:
df.duplicated()
In [ ]:
sum(df.duplicated())

```

## 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 [ ]:
df = pd.read_csv('CEREBRAL.csv')
del df['id']
df.head()

In [ ]:
df.columns

In [ ]:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']

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

In [ ]:
plt.figure(figsize=(12,7))
sns.countplot(x='stroke', data=df)

In [ ]:
plt.figure(figsize=(15,5))

plt.subplot(1,2,1)

```

```

plt.hist(df['work_type'],color='red')

plt.subplot(1,2,2)
plt.hist(df['ever_married'],color='blue')

In [ ]:
df.hist(figsize=(15,55),layout=(15,4), color='green')
plt.show()

In [ ]:
df['smoking_status'].hist(figsize=(10,5),color='yellow')

In [ ]:
sns.lineplot(df['gender'], color='brown') # scatter, plot, triplot, stackplot

In [ ]:
sns.violinplot(df['age'], color='purple')

In [ ]:
df['bmi'].plot(kind='density')

In [ ]:
sns.displot(df['smoking_status'], color='purple')
# barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot, histplot,

In [ ]:
sns.displot(df['smoking_status'], color='coral') # residplot, scatterplot

In [ ]:
fig, ax = plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)

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

plot(df, 'stroke')

```

## Performance measurements of Adaboost Classifier

```

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

import warnings
warnings.filterwarnings('ignore')

```

```

In [ ]:
df = pd.read_csv('CEREBRAL.csv')
del df['id']
df.head()

In [ ]:
df.columns

In [ ]:
df=df.dropna()

In [ ]:
df.columns

In [ ]:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['gender','ever_married','work_type','Residence_type','smoking_status']

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

In [ ]:
df.tail()

In [ ]:
x1 = df.drop(labels='stroke', axis=1)
y1 = df.loc[:, 'stroke']

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

ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT      : ", Counter(y1))
print("OVER SAMPLING DATA COUNT : ", Counter(y))

In [ ]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
print("NUMBER OF TRAIN DATASET   : ", len(x_train))
print("NUMBER OF TEST DATASET    : ", len(x_test))
print("TOTAL NUMBER OF DATASET   : ", len(x_train)+len(x_test))

In [ ]:
print("NUMBER OF TRAIN DATASET   : ", len(y_train))
print("NUMBER OF TEST DATASET    : ", len(y_test))
print("TOTAL NUMBER OF DATASET   : ", len(y_train)+len(y_test))

```

```

In [ ]:
from sklearn.ensemble import AdaBoostClassifier

In [ ]:
ADB = AdaBoostClassifier(random_state=42)
ADB.fit(x_train,y_train)

In [ ]:
predicted = ADB.predict(x_test)

In [ ]:
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF ADABOOST CLASSIFIER)

In [ ]:
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(ADB, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY )

In [ ]:
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF ADABOOST CLASSIFIER IS :",a*100)

In [ ]:
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF ADABOOST CLASSIFIER IS :",hl*100)

In [ ]:
from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF ADABOOST CLASSIFIER IS :",P*100)

In [ ]:
from sklearn.metrics import recall_score
R = recall_score(y_test,predicted)
print("THE RECALL SCORE OF ADABOOST CLASSIFIER IS :",R*100)

In [ ]:
from sklearn.metrics import f1_score
f1 = f1_score(y_test,predicted)
print("THE PRECISION SCORE OF ADABOOST CLASSIFIER IS :",f1*100)

In [ ]:
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE ,
cmap=plt.cm.Blues):
    target_names=["]
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

```

```

tick_marks = np.arange(len(target_names))
plt.xticks(tick_marks, target_names, rotation=45)
plt.yticks(tick_marks, target_names)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

cm=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF ADABOOST CLASSIFIER:\n\n')
print(cm)

sns.heatmap(cm/np.sum(cm), annot=True, cmap = 'Blues')
plt.show()

In [ ]:
def graph():
    import matplotlib.pyplot as plt
    data=[a]
    alg=" ADABOOST CLASSIFIER"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("GREEN"))
    plt.title("THE ACCURACY SCORE OF ADABOOST CLASSIFIER IS\n\n\n")
    plt.legend(b,data,fontsize=9)
graph()

```

## Implementing the RandomForest Classifier

```

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

```

```

import warnings
warnings.filterwarnings('ignore')

```

```

In [ ]:
df = pd.read_csv('CEREBRAL.csv')
del df['id']
df.head()

```

```

In [ ]:
df.columns

```

```

In [ ]:
df=df.dropna()

```

```

In [ ]:

```

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

```
var = ['gender','ever_married','work_type','Residence_type','smoking_status']
```

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

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

```
In [ ]:
x1 = df.drop(labels='stroke', axis=1)
y1 = df.loc[:, 'stroke']
```

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

```
ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT      :", Counter(y1))
print("OVER SAMPLING DATA COUNT :", Counter(y))
```

```
In [ ]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, )
print("NUMBER OF TRAIN DATASET  :", len(x_train))
print("NUMBER OF TEST DATASET   :", len(x_test))
print("TOTAL NUMBER OF DATASET  :", len(x_train)+len(x_test))
```

```
In [ ]:
print("NUMBER OF TRAIN DATASET  :", len(y_train))
print("NUMBER OF TEST DATASET   :", len(y_test))
print("TOTAL NUMBER OF DATASET  :", len(y_train)+len(y_test))
```

```
In [ ]:
from sklearn.ensemble import RandomForestClassifier
```

```
In [ ]:
RFC = RandomForestClassifier(random_state=42)
RFC.fit(x_train,y_train)
```

```
In [ ]:
predicted = RFC.predict(x_test)
```

```
In [ ]:
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER)
```

```

In [ ]:
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(RFC, x, y, scoring='accuracy')
print("THE CROSS VALIDATION TEST RESULT OF ACCURACY)

In [ ]:
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS :",a*100)

In [ ]:
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF RANDOM FOREST CLASSIFIER IS :",hl*100)

In [ ]:
from sklearn.metrics import precision_score
P = precision_score(y_test,predicted)
print("THE PRECISION SCORE OF RANDOM FOREST CLASSIFIER IS :",P*100)

In [ ]:
from sklearn.metrics import recall_score
R = recall_score(y_test,predicted)
print("THE RECALL SCORE OF RANDOM FOREST CLASSIFIER IS :",R*100)

In [ ]:
from sklearn.metrics import f1_score
f1 = f1_score(y_test,predicted)
print("THE PRECISION SCORE OF RANDOM FOREST CLASSIFIER IS :",f1*100)

In [ ]:
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF RANDOM
FOREST CLASSIFIER\n\n', cmap=plt.cm.Blues):
    target_names=[""]
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm=confusion_matrix(y_test, predicted)
print("THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:\n\n")
print(cm)

sns.heatmap(cm/np.sum(cm), annot=True, cmap = 'Blues',)

```

```
plt.show()
```

```
In [ ]:
```

```
def graph():
```

```
    import matplotlib.pyplot as plt
```

```
    data=[a]
```

```
    alg="RANDOM FOREST CLASSIFIER"
```

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

```
    b=plt.bar(alg,data,color=("YELLOW"))
```

```
    plt.title("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER ")
```

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

```
graph()
```

```
In [ ]:
```

```
import joblib
```

```
joblib.dump(RFC, 'RFC.pkl')
```



## 9.3 PLAGIARISM REPORT

### RE-2022-220256-plag-report

#### ORIGINALITY REPORT

<b>7</b> %	<b>2</b> %	<b>5</b> %	<b>3</b> %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

#### PRIMARY SOURCES

<b>1</b>	Jie Xue, Liwen Ren, Bosheng Song, Yujie Guo, Jie Lu, Xiyu Liu, Guanzhong Gong, Dengwang Li. "Hypergraph-based Numerical Neural-like P Systems for Medical Image Segmentation", IEEE Transactions on Parallel and Distributed Systems, 2023 Publication	<b>1</b> %
<b>2</b>	Rajib Mia, Shapla Khanam, Amira Mahjabeen, Nazmul Hoque Ovy et al. "Exploring Machine Learning for Predicting Cerebral Stroke: A Study in Discovery", Electronics, 2024 Publication	<b>1</b> %
<b>3</b>	Rohan Kokate, Dishant Kohad, Amita Hiwarkar, Pooja Godbole, Rutik Bhasarkar. "Real Time Parking System using ML", Research Square Platform LLC, 2024 Publication	<b>1</b> %
<b>4</b>	Submitted to Institute of Aeronautical Engineering (IARE) Student Paper	<b>1</b> %

5	Submitted to Higher Education Commission Pakistan Student Paper	1 %
6	Submitted to University of North Texas Student Paper	<1 %
7	Submitted to University of Bristol Student Paper	<1 %
8	Goulikar Laxmi Narasimha Deva, Ramesh Ponnala. "Diagnosis Of Alzheimer's Disease Using Machine Learning", international journal of engineering technology and management sciences, 2022 Publication	<1 %
9	Submitted to The NorthCap University, Gurugram Student Paper	<1 %
10	bmcpregnancychildbirth.biomedcentral.com Internet Source	<1 %
11	Guangming Guo, Feida Zhu, Enhong Chen, Qi Liu, Le Wu, Chu Guan. "From Footprint to Evidence", ACM Transactions on the Web, 2016 Publication	<1 %
12	www.jatit.org Internet Source	<1 %
13	www.pecteam.in	

Internet Source

<1 %

14

Submitted to University of Huddersfield

Student Paper

<1 %

15

Pattharaporn Thongnim, Vasin Yuvanatemiya, Phaatoon Srinil. "Chapter 29 Smart Agriculture: Transforming Agriculture with Technology", Springer Science and Business Media LLC, 2024

Publication

<1 %

16

[www.ijcaonline.org](http://www.ijcaonline.org)

Internet Source

<1 %

Exclude quotes On

Exclude matches Off

Exclude bibliography On

# REFERENCES

- [1] G. Păun, “Computing with membranes,” *J. Comput. System Sci.*, vol. 61, no. 1, pp. 108–143, 2000.
- [2] M. Ionescu and G. Păun, and T. Yokomori, “Spiking neural P systems,” *Fundamenta Informaticae*, vol. 71, no. 23, pp. 279–308, 2006.
- [3] T. Wu, A. Paun, Z. Zhang, and L. Pan, “Spiking neural P systems with polarizations,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp.3349–3360, Aug. 2018.
- [4] T. Pan et al., “Cell-like spiking neural P systems with evolution rules,” *Soft Comput.*, vol. 23, no. 14, pp. 5401–5409, 2018.
- [5] L. Pan et al., “Cell-like P systems with polarizations and minimal rules,” *Theor. Compute. Sci.*, vol. 816, pp. 1–18, 2020.
- [6] Y. Gao et al., “Hypergraph learning: Methods and practices,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 5, pp. 2548–2566, May 2022.  
XUE et al.: HYPERGRAPH-BASED NUMERICAL NEURAL-LIKE P SYSTEMS FOR MEDICAL IMAGE SEGMENTATION 1213
- [7] H. Shi et al., “Hypergraph-Induced convolutional networks for visual classification,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 10, pp. 2963–2972, Oct. 2019.
- [8] H. Peng et al., “Spiking neural P systems with inhibitory rules,” *Knowl. Based Syst.*, vol. 188, 2020, Art. no. 105064.
- [9] H. Peng et al., “Nonlinear spiking neural P systems,” *Int. J. Neural Syst.*, vol. 30, no. 10, 2020, Art. no. 2050008.
- [10] Y. Jiang, F. Luo, and Y. Su, “An improved universal spiking neural P system with generalized use of rules,” *J. Membrane Comput.*, vol. 4, pp. 41 –

55,2019.

- [11] T. Wu, L. Pan, Q. Yu, and K. C. Tan, "Numerical spiking neural P systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2443–2457, Jun. 2021.
- [12] K. J. Ballesteros et al., "Matrix representation and simulation algorithm of numerical spiking neural P systems," *J. Membrane Comput.*, vol. 4, pp. 41–55, 2022.
- [13] H. Peng et al., "Fuzzy reasoning spiking neural P system for fault diagnosis," *Inf. Sci.*, vol. 235, pp. 106–116, 2013.
- [14] B. Song et al., "A survey of nature-inspired computing: Membrane computing," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–31, 2021.