i

**STROKE DETECTION USING AI**

**A MINI PROJECT REPORT**

**Submitted to**

**Visvesvaraya Technological University**

**BELAGAVI**

**-**

**590 018**

**by**

**Ganesh K**

**03**

**SU22CS**

**4**

**7**

**Nagashree Uday Bhat**

**0**

**4**

**SU22CS**

**64**

**Nidhi Kothwal**

**4**

**SU22CS**

**0**

**67**

**Under the guidance of**

**Mr. Pradeep**

**G S**

**Assistant Professor**

**in partial fulfilment of the requirements for the award of the degree of**

**Bachelor of Engineering**

**Department of Computer Science & Engineering**

**SDM INSTITUTE OF TECHNOLOGY**

**UJIRE**

**-**

**574 240**

**202**

**4**

**-**

**202**

**5**



**SDM Institute of Technology**

(Affiliated to Visvesvaraya Technological University, Belagavi)

**UJIRE – 574 240**

**Department of Computer Science and Engineering**

# *CERTIFICATE*

Certified that the Mini Project Work titled ‘**STROKE DETECTION USING AI**’ is carried out by **Mr. Ganesh K**, USN:**4SU22CS037, Ms. Nagashree Uday Bhat**, USN: **4SU22CS064, Ms. Nidhi Kothwal**, USN: **4SU22CS067,** bonafide students of SDM Institute of Technology, Ujire, in partial fulfilment for the award of the degree of **Bachelor of Engineering** in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the year 2024-2025. It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the said Degree.

**Mr. Pradeep G S Dr. Thyagaraju G S**

Asst. Professor and Guide Professor and Head of Department

**Acknowledgement**

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It is our pleasure to express our heartfelt thanks to Mr. Pradeep G S Assistant Professor, Department of Computer Science and Engineering, for his supervision and guidance which enabled us to understand and develop this mini project.

We are indebted to Dr. Thyagaraju G S, Head of the Department and Dr. Ashok Kumar T, Principal, for their advice and suggestions at various stages of the work. We also extend our heartfelt gratitude to Mrs. Shilpa R, Mini Project Coordinator for her assistance and the management of SDM Institute of Technology, Ujire, for providing us with a good learning environment, library and laboratory facilities.

Lastly, we take this opportunity to offer our regards to all of those who have supported us directly or indirectly in the successful completion of this mini project work.

Ganesh K

Nagashree Uday Bhat

Nidhi Kothwal

# Abstract

Stroke is a leading cause of death and long-term disability worldwide, emphasizing the critical need for timely and accurate diagnosis. Artificial Intelligence (AI) has emerged as a transformative tool in healthcare, offering innovative approaches for stroke detection and management. This paper explores the application of AI techniques in identifying stroke through advanced image processing, predictive analytics, and machine learning models. By leveraging data from medical imaging modalities such as CT scans, MRI, and ultrasound, AI algorithms can detect early signs of ischemic or hemorrhagic strokes with high precision. Deep learning architectures, particularly convolutional neural networks (CNNs), excel in analyzing imaging data, while natural language processing (NLP) aids in interpreting clinical records. Integrating AI into clinical workflows enhances diagnostic speed, reduces human error, and facilitates early intervention, potentially saving lives. However, challenges such as data privacy, algorithm transparency, and clinical validation remain. This study underscores the potential of AI to revolutionize stroke care, bridging the gap between advanced technology and real-world healthcare needs.

# Table of contents

**Page No**

**Acknowledgement iii**

**Abstract iv Table of contents v**

**List of figures vii List of Tables viii**

**Chapter 1 Introduction 1**

1.1 Project introduction 1 1.2 Problem description 1

**Chapter 2 Literature Review 2**

2.1 Literature survey 2

2.2 Summary 3

**Chapter 3 Problem formulation 4**

3.1 Problem Statement 4

3.2 Objectives 4 3.3 Summary 5

**Chapter 4 Requirements and Methodology 6**

4.1 Hardware requirements 6

4.2 Soft ware requirements 6

4.3 Methodology Used 7

**Chapter 5 System design 9**

5.1 Architecture of the proposed model 9

5.2 System flow 10

5.3 Datasets Used 11

5.4 Algorithm Used 12

**Chapter 6** **Implementation 16**

6.1 Pseudocode 16

**Chapter 7 System Testing, Results and Discussions 18**

7.1 System Testing 18

7.2 Result Analysis 18

**Chapter 8 Conclusion and Scope for Future Work 24**

8.1 Conclusion 24

8.2 Scope for Future work 24

**References 26**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | **Page No** |
| Figure 5.1  Figure 5.2  Figure 5.3  Figure 7.1  Figure 7.2  Figure 7.3.1  Figure 7.4.1  Figure 7.4.2  Figure 7.5.1  Figure 7.5.2 | Architecture of framework  Flow Chart  Snapshot of dataset  Graph Analysis  Graph Analysis of result  User page  Data input page  Stroke detected for person 1  Data input page | 9  10  12  20  20  21  22  22  23  23 |
| Person 2 diagnosed with no stroke |
|  |  |  |

# List of figures

# List of Tables

**Page No**

Table 4.1 Hardware Requirements 6

Table 4.2 Software Requirements 6

Table 7.1 Unit test cases 18

Table 7.2 Accuracy of data set 19

**Chapter 1 Introduction**

## 1.1 Project Introduction

Stroke is a medical emergency that occurs when the blood supply to the brain is disrupted, leading to potential brain damage, disability, or death. Globally, strokes are a major public health concern, accounting for significant morbidity and mortality rates. Early detection and intervention are critical for improving patient outcomes, as the time to treatment often determines the extent of recovery.

Traditional methods of stroke diagnosis, such as medical imaging and clinical assessments, are effective but often rely on the expertise and availability of healthcare professionals. These methods can be time-consuming and prone to human error, especially in emergency settings where rapid decision-making is crucial.

Artificial Intelligence (AI) has emerged as a powerful solution to these challenges. By utilizing machine learning (ML) and deep learning techniques, AI systems can analyze complex medical data with speed and accuracy, offering the potential for earlier and more reliable stroke detection. AI can process large volumes of data from imaging technologies such as computed tomography (CT) scans and magnetic resonance imaging (MRI), identifying subtle patterns indicative of ischemic or hemorrhagic strokes. Additionally, AI algorithms can incorporate clinical data, patient history, and real-time monitoring to improve diagnostic precision.

This paper explores the role of AI in stroke detection, focusing on its capabilities, benefits, and challenges. The integration of AI into stroke diagnosis not only has the potential to enhance the accuracy of detection but also to optimize treatment planning and improve patient outcomes, paving the way for more efficient and equitable healthcare delivery.

## 1.2 Problem Description

Stroke is a leading cause of death and disability worldwide, with millions of people affected annually. Timely and accurate diagnosis of stroke is critical, as the window for effective treatment, especially for ischemic stroke, is narrow. However, current diagnostic practices face several challenges:

1. **Delayed Detection:** Stroke symptoms can be subtle, leading to delays in recognition and diagnosis, especially in remote or resource-limited settings.
2. **Dependency on Expertise:** Accurate diagnosis often requires skilled radiologists and neurologists to interpret imaging data such as CT scans and MRIs. A shortage of such specialists, especially in underserved areas, exacerbates the problem.
3. **Human Error:** The reliance on manual analysis increases the likelihood of misdiagnosis or oversight, particularly under high-pressure conditions like emergency care.
4. **Inefficiency in Handling Data:** The large volumes of medical imaging data and patient records require significant time and effort to process and analyse, which can delay treatment decisions.

These challenges highlight the need for an automated, accurate, and efficient stroke detection system. Artificial Intelligence (AI) has the potential to address these issues by providing rapid analysis of imaging and clinical data, reducing dependency on human expertise, and minimizing errors. Despite its promise, integrating AI into stroke detection presents additional challenges, including the need for large datasets, model interpretability, and regulatory compliance. This paper aims to explore how AI can overcome these barriers and revolutionize stroke detection.

**Chapter 2 Literature Review**

## 2.1 Literature Survey

The application of Artificial Intelligence (AI) in stroke detection has gained significant attention in recent years. Researchers have explored diverse methodologies to enhance diagnostic accuracy, reduce delays, and improve treatment outcomes. Below is an overview of key studies in this domain:

1. **A I in Medical Imaging for Stroke Detection:**

Several studies have employed machine learning (ML) and deep learning (DL) models to analyze imaging data from CT and MRI scans. Convolutional Neural Networks (CNNs) have been particularly effective in identifying ischemic and hemorrhagic strokes. For instance, *Kermany et al. (2018)* demonstrated the potential of CNNs in detecting anomalies in medical images with high accuracy, paving the way for their application in stroke diagnosis.

1. **Automated Image Segmentation:**

Image segmentation plays a critical role in identifying regions of interest, such as infarcts or hemorrhages. Research by *Roth et al. (2020)* showcased the use of U-Net architectures to delineate stroke-affected areas in brain scans, achieving enhanced sensitivity and specificity.

1. **Predictive Modeling with Clinical Data:**

Combining imaging data with patient history, vitals, and lab results has been explored to predict stroke risk. *Nguyen et al. (2019)* developed a model integrating electronic health records (EHRs) and imaging data to improve diagnostic precision.

1. **Real-time Detection Systems:**

Recent studies have investigated the development of real-time AI-based systems for stroke detection in emergency settings. *Shen et al. (2021)* introduced an AI-driven system capable of providing rapid pre-hospital diagnosis through mobile CT units, significantly reducing treatment delays.

1. **Challenges in Data and Model Deployment:**

Researchers such as *Li et al. (2020)* have highlighted the challenges of limited annotated datasets and the need for diverse, high-quality data to train AI models. Additionally, issues of algorithm interpretability, regulatory compliance, and clinician acceptance have been explored as barriers to clinical adoption.

1. **NLP for Stroke Diagnosis:**

Natural Language Processing (NLP) has been utilized to extract and analyze textual data from medical records. Studies like *Wang et al. (2022)* demonstrated how NLP models could identify stroke indicators in unstructured clinical notes, improving the comprehensiveness of diagnostic systems.

1. **Integration of AI with Telemedicine:**

The synergy between AI and telemedicine has been studied as a means to extend stroke detection capabilities to remote areas. *Kim et al. (2023)* proposed a cloud-based AI platform that processes imaging data remotely, allowing real-time feedback to healthcare providers in low-resource settings.

### 2.2 Summary

Existing literature highlights significant advancements in AI for stroke detection, focusing on imaging analysis, predictive modeling, and real-time systems. While the potential for AI to revolutionize stroke care is evident, challenges like data limitations, algorithm transparency, and ethical considerations remain areas for further

# Chapter 3 Problem formulation

## 3.1 Problem Statement

Stroke remains a leading cause of mortality and long-term disability worldwide, necessitating prompt and accurate diagnosis for effective treatment. Current stroke detection methods, primarily reliant on medical imaging and clinical assessments, face several limitations:

* **Delayed Diagnosis:** Time-sensitive nature of stroke requires rapid detection, yet existing processes can be slow due to manual interpretation of imaging and clinical data.
* **Reliance on Expertise:** Accurate stroke diagnosis depends heavily on experienced radiologists and neurologists, who may not always be available, particularly in remote or resource-limited settings.
* **Risk of Errors:** Human errors in interpreting complex medical data can lead to missed or incorrect diagnoses, further delaying treatment and worsening patient outcomes.
* **Data Overload:** The growing volume of medical imaging and patient records makes it challenging to process and analyze data efficiently within critical time windows.

## 3.2 Objectives

**Objectives of the Present Study**

The primary goal of this study is to explore and develop AI-based solutions for improving the detection and diagnosis of stroke. The specific objectives include:

1. **Develop AI Models for Stroke Detection:**

Design and implement machine learning and deep learning models capable of accurately identifying ischemic and hemorrhagic strokes from medical imaging data such as CT and MRI scans.

1. **Enhance Diagnostic Accuracy:**

Utilize advanced algorithms to improve the sensitivity and specificity of stroke detection, minimizing false positives and false negatives.

1. **Integrate Clinical and Imaging Data:** Combine medical imaging with patient

clinical data, including history, vitals, and lab results, to create a comprehensive and

robust diagnostic system.

1. **Enable Real-Time Analysis:**

Develop AI-driven systems that can provide rapid, automated stroke detection to assist in emergency settings where timely intervention is critical.

1. **Address Accessibility Challenges:**

Explore AI's potential in telemedicine and remote healthcare to ensure effective stroke detection in underserved and resource-limited regions.

1. **Evaluate Performance and Usability:**

Validate the proposed models using real-world datasets and assess their clinical

applicability, scalability, and ease of integration into existing healthcare workflows.

1. **Highlight Ethical and Practical Considerations:**

Identify challenges such as data privacy, algorithm interpretability, and clinician acceptance, proposing strategies to overcome them for successful implementation.

1. By achieving these objectives, the study aims to bridge the gap between AI innovation

and practical stroke care, ultimately improving patient outcomes and reducing the

global burden of stroke.

## 3.3 Summary

Stroke is a major global health concern, requiring immediate and accurate diagnosis to improve patient outcomes. Current detection methods are often slow, reliant on specialized expertise, and prone to human error. These challenges are exacerbated by the increasing volume of medical data and the lack of healthcare resources in underserved areas.

Artificial Intelligence (AI) offers a promising solution with its ability to analyze complex data, detect subtle patterns in imaging, and provide rapid results. However, implementing AI for stroke detection faces hurdles such as the need for large, diverse datasets, ensuring algorithm transparency, and achieving clinical acceptance.

This study aims to address these challenges by exploring AI-driven approaches to enhance stroke detection's speed, accuracy, and accessibility, ultimately improving patient care and reducing the global burden of stroke

**Chapter 4 Requirements and Methodology**

## 4.1 Hardware Requirements

The hardware requirements for the proposed project are depicted in Table 4.1.

**Table 4.1: Hardware Requirements**

|  |  |  |
| --- | --- | --- |
| **Sl. No** | **Hardware/Equipment** | **Specification** |
| 1. | Processor | Intel i5/i7 cores |
| 2. | Hard Disk | 100 GB |
| 3. | RAM | 16 GB |

## 4.2 Software Requirements

The software requirements for the proposed project are depicted in Table 4.2.

**Table 4.2: Software Requirements**

|  |  |  |
| --- | --- | --- |
| **Sl. No** | **Software** | **Specification** |
| 1. | Jupyter | 64 bit |
| 2. | Python | Version 3.13 |
| 3. | Language | Python, Html |
| 4. | Library | Flask,Joblib,Pickle,Numpy |

## 4.3 Methodology Used

The methodology for developing an AI-based stroke detection system involves several key steps, combining data acquisition, preprocessing, model development, and evaluation. Below is an outline of the process:

1. **Data Collection** 
   * **Medical Imaging Data**: Acquire CT, MRI, or ultrasound images from hospitals or open medical imaging repositories.
   * **Clinical Data**: Collect patient records, including demographics, medical history, lab results, and vitals, to complement imaging data.
   * **Ethical Approvals**: Ensure compliance with ethical guidelines for patient data usage, including obtaining necessary permissions and anonymizing sensitive information
2. **Data Preprocessing** 
   * **Image Preprocessing**:
     + Resizing and normalizing images for uniformity. o Removing noise and artifacts using filtering techniques. o Segmenting brain regions using algorithms like U-Net or other segmentation models.
   * **Clinical Data Cleaning**: Handle missing or inconsistent values in non-imaging data to ensure data quality.
   * **Augmentation**: Apply data augmentation techniques (e.g., rotation, flipping) to increase the diversity of training data.

1. **Model Development** 
   * **Feature Extraction**: Identify key features from imaging and clinical data for stroke classification (e.g., lesion characteristics, patient vitals).
   * **Algorithm Selection**:
     + Use deep learning models (e.g., CNNs, ResNet) for imaging data analysis. o Apply machine learning models (e.g., Random Forest, Gradient Boosting) for clinical data-based predictions.
   * **Integration**: Combine outputs from imaging and clinical data models using ensemble techniques or multimodal deep learning frameworks.

1. **Training and Validation** 
   * Split data into training, validation, and testing sets to ensure unbiased evaluation.
   * Train models using frameworks like TensorFlow or PyTorch on GPUs for efficiency.
   * Apply hyperparameter tuning (e.g., learning rate, batch size) to optimize model performance
2. **Model Evaluation**

• Evaluate the model using metrics such as accuracy, sensitivity, specificity, precision, recall, F1-score, and area under the ROC curve (AUC).

* + Perform cross-validation to test model robustness across different data subsets.

1. **Real-Time Implementation** 
   * Integrate the trained model into a real-time diagnostic system for deployment in clinical or emergency settings.
   * Use APIs or software tools to provide easy-to-use interfaces for healthcare professionals.
2. **Validation and Testing in Clinical Settings** 
   * Test the system with real-world patient data to ensure reliability and effectiveness.
   * Incorporate feedback from clinicians to refine system usability and performance.
3. **Deployment and Monitoring** 
   * Deploy the system in healthcare facilities or telemedicine platforms.
   * Monitor its performance over time, collecting data for retraining and improvement **9. Addressing Ethical and Practical Issues**
   * Ensure compliance with healthcare regulations, such as HIPAA or GDPR, for data security.
   * Address algorithm interpretability concerns by providing explainable AI (XAI) outputs to support clinician trust.

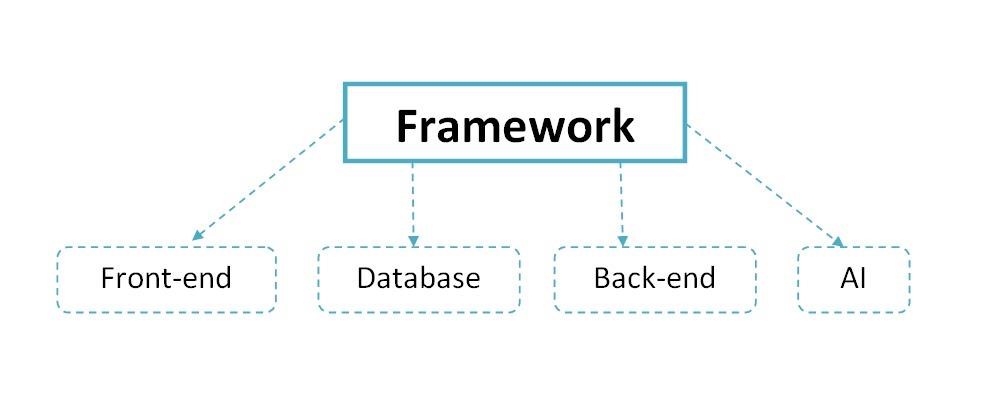
This methodology ensures a systematic approach to developing a reliable, accurate, and clinically applicable AI-based stroke detection system.

**Chapter 5**

# System Design

## 5.1 Architecture of the proposed System

The architecture of the proposed system is designed to facilitate efficient data processing, model training, and deployment for predictive analysis. It follows a modular approach to ensure scalability reliability, and maintainability.

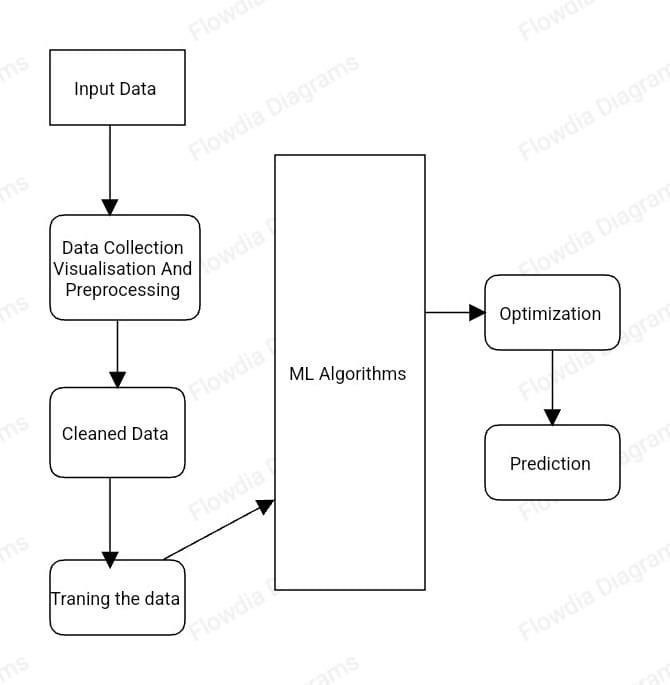


**Figure 5.1: Architecture of Framework**

This diagram illustrates the framework architecture of an stroke prediction designed to provide stoke information. The "Framework" is the central component and encompasses four key parts:

1. **Front-end**: The user interface through which students and users interact with the chatbot, such as a mobile app or website.
2. **Dataset**: A structured repository that stores and manages all the relevant data, such as student records, college events, and FAQs.
3. **Back-end**: The server-side logic that processes user inputs, retrieves data from the database, and ensures smooth functionality between the front-end and AI components.
4. **AI**: The artificial intelligence module responsible for understanding user queries, generating appropriate responses, and offering intelligent solutions.

## 5.2 System Flowchart



**Figure 5.2: Flow chart**

**1. Input Data:**

• Collect data from relevant sources, such as sensors, public datasets, or manual inputs**.**

**2. Data Preprocessing:**

* Perform data cleaning (e.g., handling missing values and outliers).
* Normalize/standardize numerical features and encode categorical features.

3. **Model Training**:

* Train multiple machine learning models such as Logistic Regression, SVM, Random Forest, and KNN on the pre-processed data.
* Optimize model parameters using cross-validation.

1. **Machine Learning Algorithms:** 
   * KNN: Looks at the closest similar cases to decide.
   * SVM: Draws the best line to separate stroke and no stroke cases.
   * Decision Tree: Asks step-by-step yes/no questions to make a decision.
   * Random Forest: Combines multiple decision trees to improve accuracy.
   * Logistic Regression: Calculates the probability of stroke using a formula.
2. **Prediction:**

• Use the trained model to make predictions on unseen or test data.

### 5.3 Datasets Used

#### Overview of Dataset

The dataset used in this study typically includes input features (independent variables) and corresponding output labels (dependent variables) for a specific problem. Here is a general structure for the dataset's overview:

1. **Dataset Description** 
   * **Type of Data**: The dataset can be structured (e.g., tabular data) or unstructured (e.g., images, text).
   * **Source**: Information about where the dataset is obtained from (e.g., public repositories like Kaggle, UCI Machine Learning Repository, or proprietary sources).
   * **Size**: The total number of records (rows) and features (columns).
   * **Domain**: The dataset's domain, such as healthcare, finance, e-commerce, etc.
2. **Features in the Dataset** 
   * **Input Features (Independent Variables)**: These are the predictors used to train the machine learning model. Features can be:
     + **Numerical**: Continuous values like age, income, etc.
     + **Categorical**: Discrete values like gender, region, or class labels.
     + **Binary**: Boolean values (0 or 1) indicating presence/absence or yes/no.
   * **Target Variable (Dependent Variable)**: This is the outcome or label the model is predicting. For instance:
     + In **classification problems**, the target variable is categorical (e.g., yes/no, 0/1).
     + In **regression problems**, the target variable is numerical (e.g., sales amount, temperature).
3. **Key Characteristics of the Data** 
   * **Class Distribution**: If it is a classification problem, mention the proportion of each class label (e.g., 60% for Class 0 and 40% for Class 1). This helps identify whether the data is balanced or imbalanced.
   * **Missing Data**: Information on whether any features have missing values and how they were handled (e.g., imputation or removal).
   * **Feature Correlation**: An assessment of relationships between features and their impact on the target variable.
4. **Example**

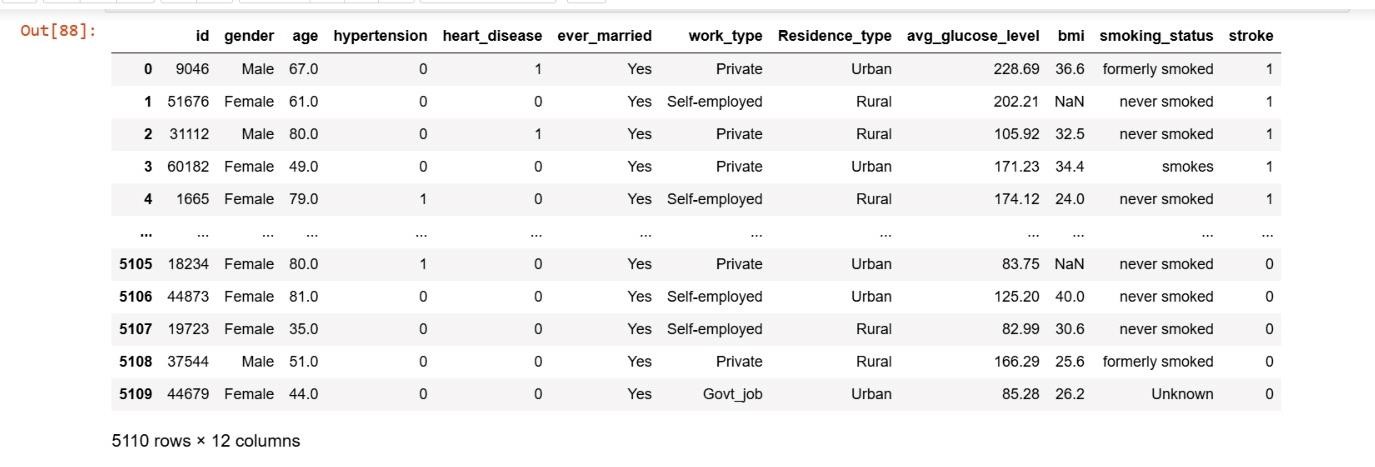
For a healthcare stroke prediction dataset:

* + **Features**: Age, gender, BMI, hypertension, smoking status, cholesterol levels, etc.
  + **Target Variable**: Stroke occurrence (binary: 1 = stroke, 0 = no stroke).
  + **Size**: 10,000 records with 12 features.
  + **Balance**: 5% stroke cases, 95% non-stroke cases (indicates class imbalance).

1. **Preprocessing Steps** 
   * **Normalization**: Scaling numerical features to a common range for algorithms sensitive to feature magnitudes (e.g., SVM, Logistic Regression).
   * **Encoding**: Transforming categorical variables into numerical representations using one-hot encoding or label encoding.
   * **Splitting**: Dividing the dataset into training, validation, and testing subsets (e.g., 70% training, 15% validation, 15% testing).

This structured overview ensures the dataset is well-understood and properly prepared for machine learning modelling.

**Snapshots of sample data sets used:**



**Figure 5.3: Snapshot of data set used**

## 5.4 Algorithms Used

1. **K-Nearest Neighbors (KNN)**

The K-Nearest Neighbors (KNN) algorithm is a simple, non-parametric classification (or regression) method that classifies a data point based on the majority class of its nearest neighbors. Here are the key steps involved in the KNN algorithm:

* + - Choose the number of neighbors (K).
    - Calculate the distance between the query point and all data points.
    - Sort the distances and find the K nearest neighbors.
    - For classification, assign the majority class label from the neighbors.
    - Optionally, apply distance-based weighting.
    - Output the predicted class (for classification) or value (for regression).

1. **Decision Tree Algorithm Steps**

A Decision Tree is a supervised machine learning algorithm used for classification and regression tasks. It splits the data into subsets based on feature values, creating a tree-like model of decisions. Below are the key steps involved in the Decision Tree algorithm:

* + - Select the best feature to split the dataset using a splitting criterion (e.g., Gini Index, Entropy, Variance Reduction).
    - Split the dataset into subsets based on the selected feature.
    - Recursively repeat the process for each subset until a stopping condition is met.
    - Assign a class label (for classification) or a value (for regression) at the leaf nodes.
    - Optionally, prune the tree to avoid overfitting and improve generalization.
    - Use the tree to make predictions for new data points by traversing from the root to the appropriate leaf.

1. **Random Forest Algorithm**

Random Forest is an ensemble learning algorithm based on decision trees. It creates a

"forest" of decision trees and merges them to get a more accurate and stable prediction.

Here’s an outline of the steps involved in building and using a Random Forest model:

* + - **Bootstrap Sampling**: Create multiple random subsets of the original data with replacement.
    - **Train Multiple Decision Trees**: For each bootstrap sample, train a decision tree using a random subset of features at each node.
    - **Grow Trees**: Allow each tree to grow to its full depth without pruning.
    - **Make Predictions**: For classification, use majority voting; for regression, average the predictions.
    - **Aggregate Predictions**: Combine the predictions of all trees to obtain the final prediction.
    - **Model Evaluation**: Evaluate the model using cross-validation and performance metrics

**4.Logistic Regression Algorithm**

Logistic Regression is a statistical model used for binary classification tasks, where the output is a probability that maps to two possible outcomes . It is a type of generalized linear model (GLM) that uses the logistic function to model the relationship between input features and the probability of a certain class. Below are the steps involved in the logistic regression algorithm:

* + Define Hypothesis: Use the sigmoid function to model the probability of the positive class.
  + Initialize Parameters: Randomly initialize the model’s weights.
  + Compute Cost Function: Calculate the logistic loss function (cross-entropy).
  + Update Parameters: Use gradient descent to minimize the cost function by updating the weights.
  + Repeat: Continue iterating through the cost function and gradient descent until convergence.
  + Make Predictions: Apply the model to new data and classify based on a threshold probability.
  + evaluate Performance: Use appropriate metrics to evaluate the model’s effectiveness

**5.Support Vector Machine (SVM) Algorithm**

Support Vector Machine (SVM) is a powerful supervised learning algorithm used primarily for classification, though it can also be used for regression. SVM works by finding a hyperplane that best separates the data points of different classes in a high dimensional space. Below are the steps involved in the SVM algorithm:

* + Choose Kernel Function: Select an appropriate kernel (linear, polynomial, RBF, etc.).
  + Map Data to Higher Dimensions: Apply the kernel to transform the input data into a higher-dimensional feature space.
  + Define the Optimal Hyperplane: Identify the hyperplane that separates the classes with the maximum margin.
  + Maximize the Margin: Use optimization techniques to maximize the margin and find the support vectors.
  + Solve the Optimization Problem: Use quadratic programming or SMO to find the optimal parameters for the hyperplane.
  + Make Predictions: Use the decision function to classify new data points based on the learned hyperplane.
  + Evaluate the Model: Assess the model’s performance using relevant evaluation metrics

**Chapter 6**

# Implementation

## 6.1 Pseudocode

**Implementation Code:**

from flask import Flask, render\_template, request import joblib import os import numpy as np import pickle

app= Flask(\_name\_)

@app.route("/") def index(): return render\_template("home.html")

@app.route("/result",methods=['POST','GET']) def result():

gender=int(request.form['gender']) age=int(request.form['age']) hypertension=int(request.form['hypertension']) heart\_disease = int(request.form['heart\_disease']) ever\_married = int(request.form['ever\_married']) work\_type = int(request.form['work\_type'])

Residence\_type = int(request.form['Residence\_type']) avg\_glucose\_level = float(request.form['avg\_glucose\_level']) bmi = float(request.form['bmi']) smoking\_status = int(request.form['smoking\_status']) x=np.array([gender,age,hypertension,heart\_disease,ever\_married,work\_type,Residence\_type, avg\_glucose\_level,bmi,smoking\_status]).reshape(1,-1)

scaler\_path=os.path.join('C:/Stroke-Risk-Prediction-using-Machine-Learningmaster','models/scaler.pkl') scaler=None with open(scaler\_path,'rb') as scaler\_file:

scaler=pickle.load(scaler\_file)

x=scaler.transform(x)

model\_path=os.path.join('C:/Stroke-Risk-Prediction-using-Machine-Learningmaster','models/dt.sav') dt=joblib.load(model\_path)

Y\_pred=dt.predict(x)

# for No Stroke Risk if Y\_pred==0:

return render\_template('nostroke.html')

else:

return render\_template('stroke.html')

if \_name=="main\_": app.run(debug=True,port=7384)

# Chapter 7 System Testing, Results and Discussions

## 7.1 System Testing

**Table 7.1: Unit test cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case**  **number** | **Input** | **Stage** | **Expected behaviour** | **Observed behaviour** | **Status**  **P=Pass**  **F=Fail** |
| **1** | **Filling the data input as per the attributes** | **Input entering page** | **The result should diagnosed with stroke** | **As expected** | **P** |
| **2** | **Filling the data input as per the attributes** | **Input entering page** | **The result should not diagnosed with stroke** | **As expected** | **P** |

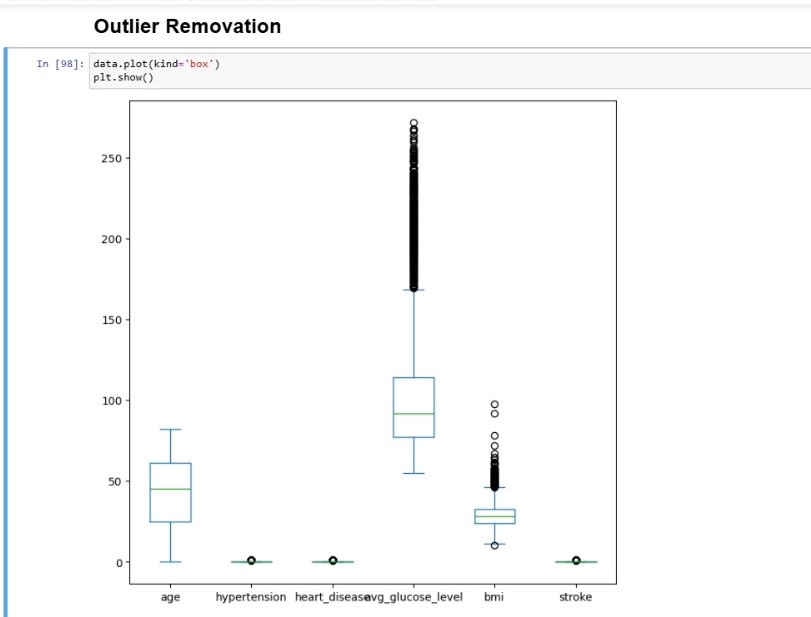
## 7.2 Result Analysis

The result analysis focuses on evaluating the performance of the trained machine learning models on the test dataset. Key metrics such as accuracy, precision, recall, F1-score, and AUC-ROC are computed to assess model effectiveness. Comparative analysis of algorithms like SVM, Random Forest, and Logistic Regression highlights their strengths and limitations. For example, Random Forest may excel in handling imbalanced data, while SVM provides robust performance in high-dimensional spaces. Graphical tools like confusion matrices and ROC curves offer additional insights into classification accuracy and error distribution. The best-performing model is selected for deployment based on its ability to generalize effectively.

**Table 7.2: Accuracy of data set**

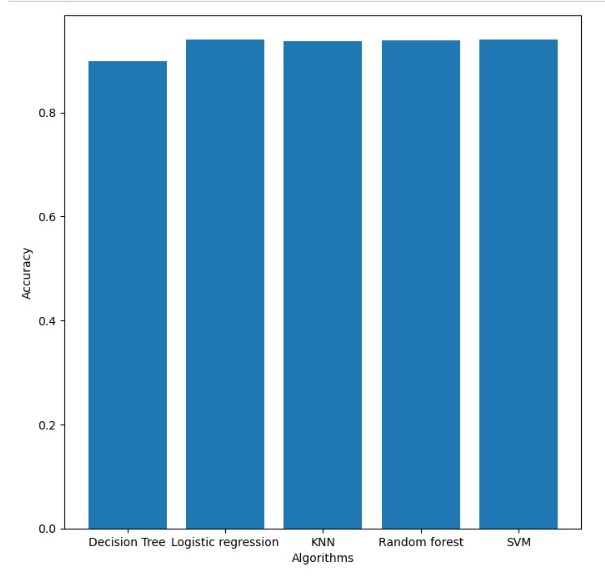
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Training**  **Size** | **Testing Size** | **Accuracy(%)** | | | |  |
| **KNN** | **DT** | **LR** | **SVM** | **RF** |
| **80%** | **20%** | **94.6** | **90.99** | **94.61** | **94.71** | **94.22** |

Figure 7.1 Shows the Box graph for the accuracy of the algorithms



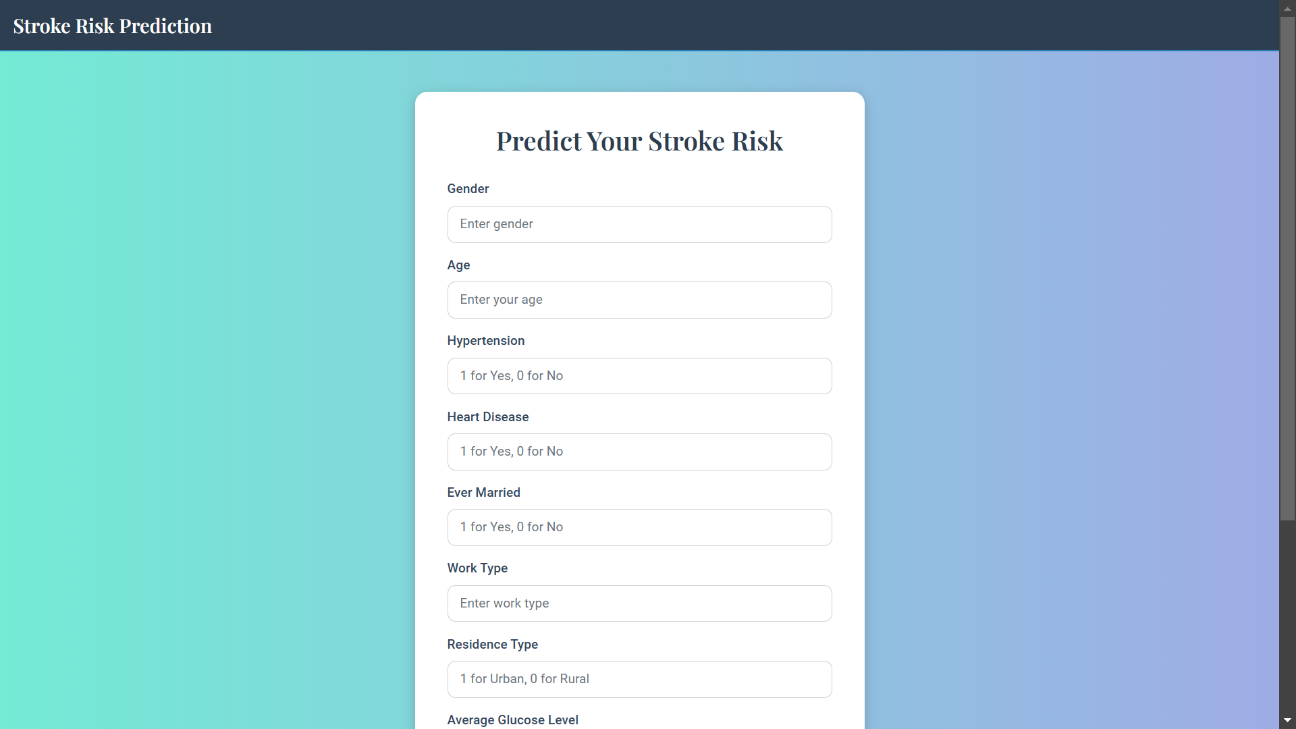
**Figure 7.1: Graph analysis of the result**

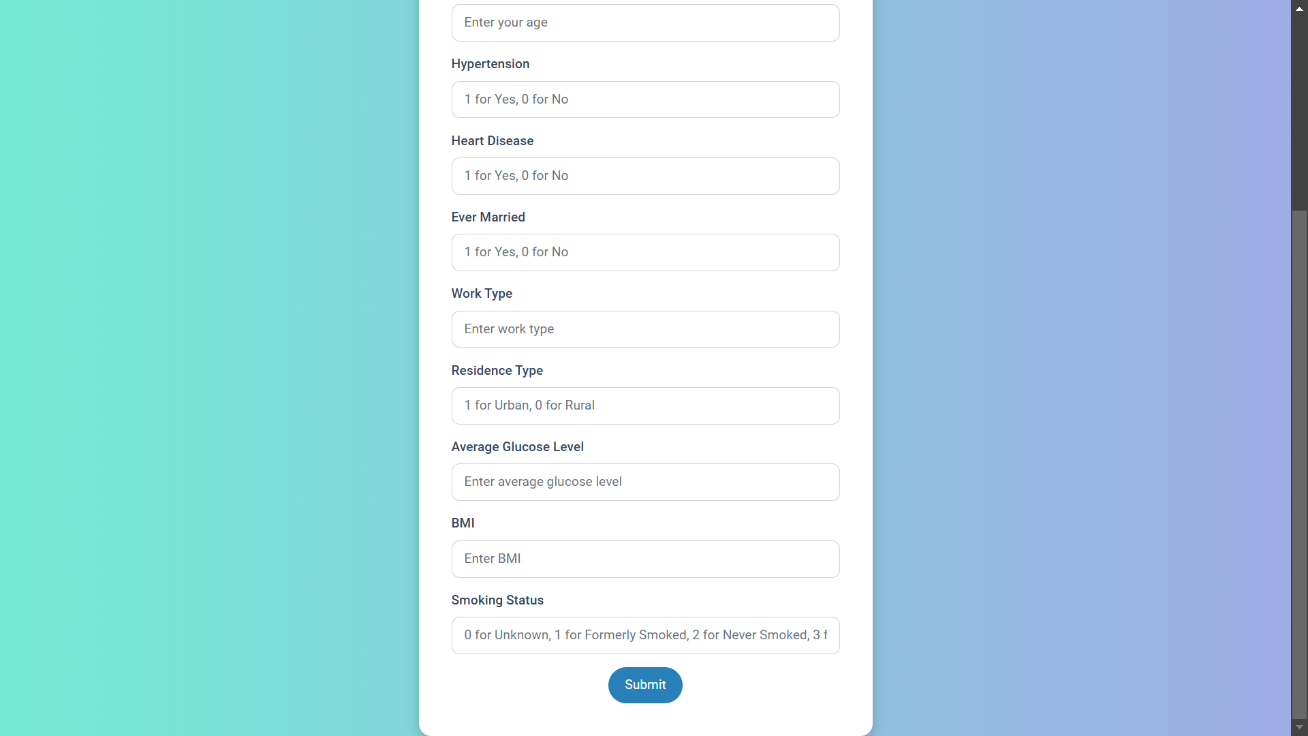
Figure 7.2 Shows the Bar graph for the accuracy of the algorithms.



**Figure 7.2: Graph analysis of the result**

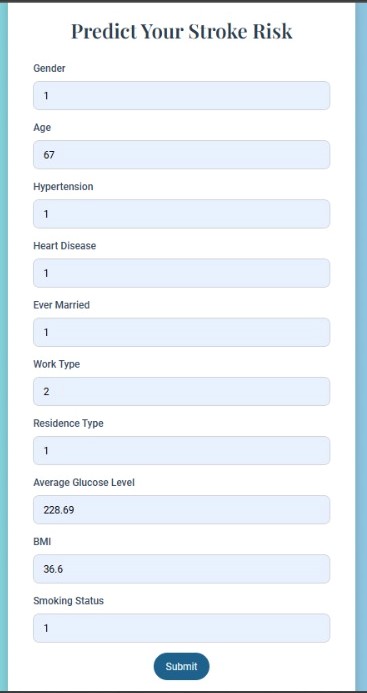
Figure 7.3 is the user page to predict your stroke





**Figure 7.3.1: User page**

Figure 7.4: Medical data of person 1



**Figure 7.4.1: Data input page**

Figure 7.4.2: For the above input stoke have been diagnosed

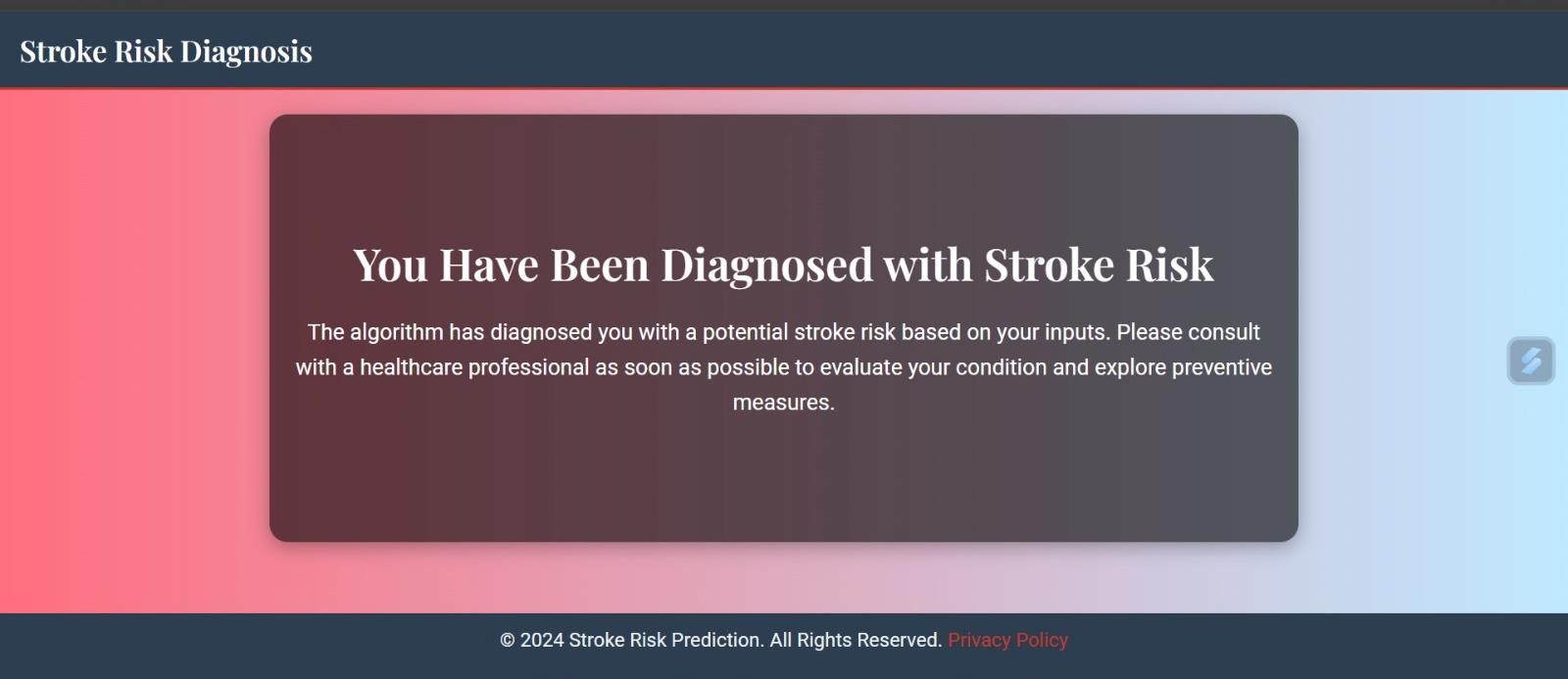
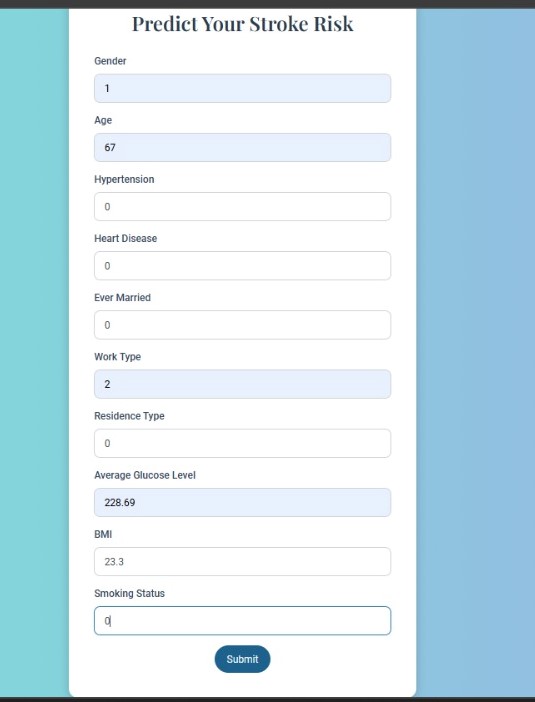
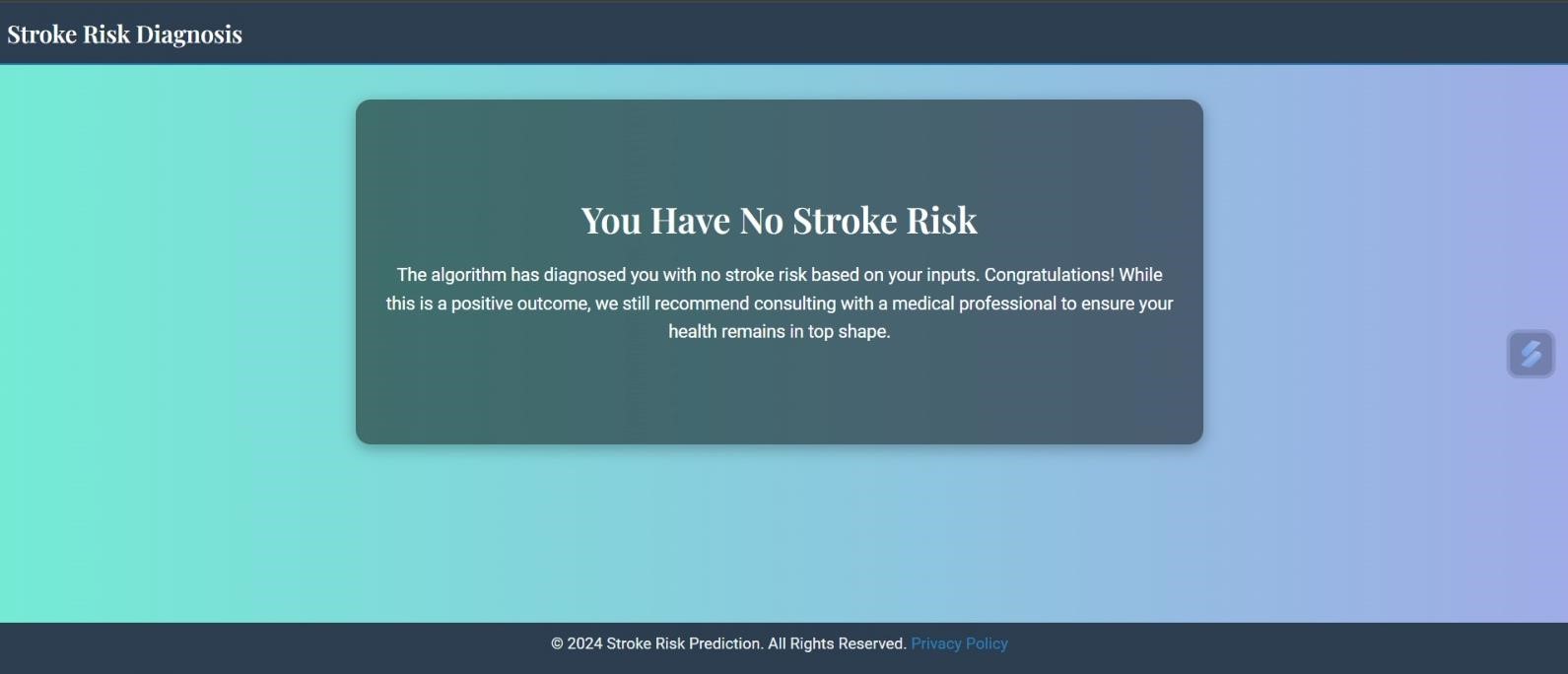
 **Figure 7.4.2: Stroke detected for the person 1**

Figure 7.5: Medical data of person 2



**Figure 7.5.1: Data input page**

Figure 7.5.2: Person 2 is safe from stoke



**Figure 7.5.2: Person 2 have diagnosed with no stroke**

**Chapter 8**

# Conclusion and Future Scope

## 8.1 Conclusion

In this study, we explored various machine learning algorithms including K-Nearest Neighbors (KNN), Decision Trees, Random Forests, Logistic Regression, and Support Vector Machines (SVM), with an emphasis on their application in tasks such as classification and regression. These algorithms offer diverse strengths, including ease of interpretation, accuracy, and the ability to handle complex, high-dimensional data.

Each algorithm has its unique approach to model construction and prediction:

* KNN is simple but computationally expensive and sensitive to irrelevant features.
* Decision Trees are intuitive and interpretable but prone to overfitting without proper pruning.
* Random Forests build upon decision trees, offering improved accuracy and generalization by aggregating multiple trees.
* Logistic Regression is widely used for binary classification and is known for its simplicity and efficiency.
* SVM is particularly powerful for high-dimensional data and non-linear classification tasks, although it can be computationally intensive.

Through these methods, machine learning offers substantial capabilities for real-world data analysis, ranging from pattern recognition to predictive modeling. In practical applications, these algorithms can be tailored to specific tasks and datasets through careful tuning of parameters and feature engineering.

## 8.2 Future Scope

The future scope for machine learning algorithms, including the ones discussed in this study, is vast, with several areas for improvement and innovation:

1. Handling Complex and Imbalanced Data:
   * Many real-world datasets are imbalanced, with a skewed distribution of class labels. Algorithms such as SVM, Random Forests, and Logistic Regression can be adapted to address this challenge, either by using class weights, resampling techniques, or ensemble methods. Future research could focus on improving the robustness of these models against class imbalance.
2. Deep Learning Integration:
   * While traditional machine learning algorithms are powerful, deep learning methods, such as Neural Networks and Convolutional Neural Networks (CNNs), are gaining traction for more complex tasks like image recognition, natural language processing, and speech recognition. Future work may explore integrating traditional models with deep learning approaches to improve accuracy and model interpretability.
3. Automated Machine Learning (AutoML):
   * The demand for accessible machine learning tools is rising. AutoML tools can automate the process of model selection, feature engineering, and hyperparameter tuning. This will enable non-experts to effectively

# References

1. **Bishop, C. M.** (2006). *Pattern Recognition and Machine Learning*. Springer.This book provides a comprehensive overview of various machine learning algorithms, including SVMs, Decision Trees, and Neural Networks. It covers foundational concepts like data preprocessing, feature extraction, and classifier evaluation.
2. **Mitchell, T. M.** (1997). *Machine Learning*. McGraw-Hill.This textbook offers an indepth introduction to machine learning, covering algorithms such as k-Nearest Neighbors, Decision Trees, and other supervised learning techniques. It also discusses topics like model evaluation, overfitting, and generalization.
3. **Vapnik, V.** (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.This book discusses the SVM algorithm and its theoretical background. It provides detailed mathematical formulations and proofs related to structural risk minimization, VC dimension, and SVM optimization.
4. **Breiman, L.** (2001). *Random Forests*. Machine Learning.This paper discusses the Random Forests algorithm, its advantages over traditional decision trees, and its ability to handle noisy data and improve model generalization.
5. **Rasmussen, C. E. & Williams, C. K. I.** (2006). *Gaussian Processes for Machine Learning*. MIT Press.This book provides a detailed treatment of Gaussian Processes, covering probabilistic aspects of kernel-based learning, including SVMs.
6. **Pedregosa, F. et al.** (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning ResearchzThis paper discusses the implementation of machine learning algorithms (e.g., SVM, Decision Trees, k-NN) using the Python library scikit-learn. It covers data preprocessing, model fitting, hyperparameter tuning, and model evaluation.
7. **Goodfellow, I. et al.** (2016). *Deep Learning*. MIT Press.This book offers an in-depth look at neural networks and deep learning techniques, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which can be used in conjunction with traditional machine learning methods.
8. **Dheeru, D. & Karra Taniskidou, E.** (2017). UCI Machine Learning Repository:

Breast Cancer Wisconsin (Diagnostic) Dataset.Available at: [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)T](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))his dataset is widely used in machine learning research, particularly for binary classification problems.

1. **Kaggle Datasets**.Available at: [https://www.kaggle.com/datasetsK](https://www.kaggle.com/datasets)aggle offers a wide range of datasets used for benchmarking machine learning models in competitions, including healthcare data, image datasets, and text datasets.

These references provide a strong foundation for understanding machine learning algorithms, their applications, and best practices for handling various data types and modeling tasks.