VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", BELAGAVI-590018

PROJECT REPORT ON

"BRAIN STROKE PREDICTION USING ML"

Submitted in partial fulfilment for the award of degree of bachelor of engineering in information science & engineering during the year 2024-2025

Submitted by

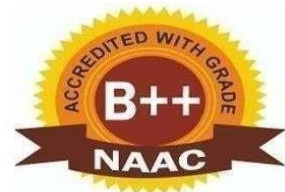|	
Amarnath R	**[4MH21IS006]**
Ankith TP	**[4MH21IS011]**
Jagath Ponnanna PM	**[4MH21IS116]**
Sandesh KM	**[4MH22IS416]**

Under the Guidance of

Prof. Bhavyashree H D

Asst.Professor,

Dept. of ISE, MIT, Mysore

DEPARTMENT OF INFROMATION SCIENCE & ENGINNEERING

MAHARAJA INSTITUTE OF TECHNOLOGY, MYSORE

Belawadi, S.R. Patna(T), Mandya(D)-571477

2024-2025

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

MANDYA — 571477.

CERTIFICATE

Certified that the project work entitled " Brain Stroke Prediction using ML" has been successfully carried out by **Amarnath R [4MH21IS006], Ankith T P [4MH21IS011], Jagath Ponnanna P M [4MH21IS116], Sandesh K M [4MH22IS416],** bonafide students of **Maharaja Institute of Technology Mysore** in partial fulfilment of the requirements of PROJECT WORK in **Information Science and Engineering of Visvesvaraya Technological University, Belagavi** during the academic year **2024-2025**. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements with respect to the project work prescribed for Bachelor of Engineering Degree.

_____	_____	_____
Signature of the Guide	Signature of the HOD	Signature of the Principal
Prof. Bhavyashree H D	**Dr. Sharath Kumar Y.H**	**Dr. Murali S**
Assistant Professor	Professor & Head	Principal
Dept. of IS&E MIT	Dept. of IS&E	MIT MYSORE
MYSORE	MIT MYSORE	

External Viva

Name of the Examiners   Signature with date

1.

2.

ABSTRACT

Brain stroke is a serious medical condition and a leading cause of disability and death worldwide. Early prediction and diagnosis can significantly improve patient outcomes by enabling timely medical intervention. This project presents a machine learning-based approach to predict the risk of stroke using patient health data. The dataset includes various features such as age, gender, hypertension, heart disease, average glucose level, body mass index (BMI), and work type. Through data preprocessing, exploratory analysis, and correlation studies, key factors contributing to stroke risk were identified. Multiple machine learning models were evaluated, including ANN and hybrid models that demonstrated acceptable, strong performance, achieving over 95% accuracy during training. Visualizations like bar charts and heatmaps provided insights into stroke distribution across gender and employment type, as well as feature correlations. The final predictive model classifies individuals into high or low risk of stroke, providing a valuable tool to assist healthcare providers in early diagnosis.

ACKNOWLEDGEMENT

LIST OF FIGURES

LIST OF TABLES

LIST OF SNAPSHOTS

Chapter 1

INTRODUCTION

1.1 Overview

Brain stroke remains a major global health concern, contributing significantly to mortality and disability rates. Strokes occur due to disruptions in blood flow to the brain, often caused by factors such as hypertension, cholesterol imbalances, and underlying cardiovascular diseases. Identifying individuals at high risk of stroke is crucial for timely medical intervention and prevention strategies. Traditional methods for stroke prediction rely on statistical analyses and medical assessments. There is a growing interest in utilizing machine learning techniques to enhance stroke prediction accuracy and provide more reliable diagnostic tools.

Machine learning models have demonstrated significant potential in medical diagnostics by identifying hidden patterns in vast amounts of patient data. This study explores the relationship between general health indicators such as blood pressure, cholesterol levels, and the likelihood of experiencing a stroke. By applying advanced machine learning algorithms, including Artificial Neural Networks (ANN), Random Forest (RF), and Decision Tree Classifier. This project evaluates the effectiveness of different predictive models. These models are trained on a large dataset of stroke patients and optimized using techniques like feature selection, cross-validation to ensure higher accuracy and reliability.

Existing stroke prediction models often struggle with scalability and real-world applicability due to challenges like data imbalance and feature dependencies. This study addresses these limitations by implementing robust data preprocessing techniques and integrating analytics to improve stroke risk assessment. By comparing multiple machine learning approaches, aim is to determine the most effective model. The ultimate goal is to develop a scalable, data-driven diagnostic tool that can assist healthcare professionals in identifying high-risk patients before a stroke occurs.

1.2 Problem Statement

Brain stroke remains a significant health concern globally, contributing to high mortality and disability rates. Despite advances in medical science, early and accurate prediction of strokes remains a challenge due to the complexity of contributing factors such as hypertension, cholesterol levels, and genetic predispositions. Current methods for stroke prediction lack the efficiency and accuracy needed for timely intervention. There is a need for more reliable and scalable predictive models that can analyze vast amounts of patient data to identify individuals at high risk of stroke. The problem intensifies due to the high number of variables involved and the necessity for advanced analytical tools to uncover hidden patterns in medical data.

1.3 Existing System

Several machine learning and statistical models have been proposed in the past for stroke prediction. Early methods relied heavily on logistic regression, Cox proportional hazards models, and decision trees, which provided baseline risk assessments based on structured medical data. These models utilized key health indicators such as blood pressure, cholesterol levels, and medical history to estimate stroke likelihood. While these traditional methods have been widely used, they often lack the ability to analyze complex, high-dimensional datasets, limiting their predictive accuracy.

More recent studies have explored machine learning algorithms such as Random Forest (RF) and Artificial Neural Networks (ANN) for stroke prediction. Researchers have applied Principal Component Analysis (PCA) for dimensionality reduction and Decision Tree- based feature selection to improve model efficiency. Neural networks, in particular, have demonstrated promising results due to their ability to learn non-linear relationships in medical data. Some studies have also experimented with electronic medical records (EMRs) for predictive analysis, leveraging deep learning models to extract hidden patterns. However, many of these approaches struggle with challenges such as data imbalance, overfitting, and high computational costs, which impact their real-world implementation.

Despite these advancements, existing stroke prediction systems still face limitations in scalability, real-time application, and interpretability. Many models fail to account for dynamic health changes over time, making them less effective in continuous risk monitoring.

Additionally, certain predictive models lack transparency, making it difficult for healthcare professionals to interpret their outcomes and apply them in clinical settings. The need for an improved system that integrates multiple machine learning techniques, optimizes feature selection, and enhances prediction accuracy remains critical. This study aims to bridge these gaps by developing a more refined, data-driven approach to stroke risk assessment using predictive models.

1.4 Proposed System

The proposed system aims to enhance stroke prediction by leveraging advanced machine learning techniques, including Artificial Neural Networks (ANN), Random Forest (RF) and Decision Tree Classifier. This approach integrates robust data preprocessing, feature selection, and model optimization to improve prediction accuracy. By analyzing a large dataset of stroke patients, the system identifies key health indicators such as blood pressure, cholesterol levels, and general well-being to assess stroke risk more precisely. Feature engineering techniques, such as normalization and standardization, ensure that the data is properly structured for effective model training, reducing biases and improving model performance.

To further refine the predictive capability, the system incorporates k-fold cross validation. ANN is used to capture complex non-linear relationships within the data, while SVM helps in classifying stroke and non- stroke cases by defining optimal hyperplanes. KNN is utilized for its ability to compare new patient data with existing cases, enhancing the model's decision-making accuracy. By integrating multiple models, the system ensures better generalization, reducing false negatives and improving early stroke detection rates. Additionally, the use of an adaptive algorithm selection mechanism allows the system to choose the best-performing model.

1.5 Motivation

Brain stroke is a major global health concern, leading to high mortality and disability rates. Traditional diagnostic methods often fail to detect stroke risks early due to the complex interplay of factors like hypertension, cholesterol levels, and cardiovascular conditions. With the increasing availability of medical data, there is a strong need for predictive models that can analyze these factors efficiently and provide timely risk assessments. Machine learning offers a promising solution to enhance accuracy and automate stroke prediction, reducing the chances of late diagnosis and improving preventive care.

The motivation for this study stems from the potential of artificial intelligence to revolutionize healthcare decision-making. By leveraging machine learning techniques such as Artificial Neural Networks (ANN), Random Forest ,Support Vector Machines (SVM), and Decision Tree Classifier, this research aims to develop a robust and scalable stroke prediction system. An AI-based approach can help healthcare professionals identify high-risk individuals early, enabling personalized interventions and reducing stroke-related complications, ultimately saving lives and lowering healthcare burdens.

Chapter 2

LITERATURE SURVEY

2.1 Survey Sources

[1] Year- 2024: Brain Stroke Detection through Advanced Machine Learning and Enhanced Algorithms.

The architecture of the proposed system is designed to facilitate accurate and efficient stroke prediction using machine learning models. It consists of multiple stages, including data collection, preprocessing, feature selection, model training, and evaluation. Initially, raw stroke- related datasets from sources like healthcare repositories or Kaggle are collected. The preprocessing stage involves handling missing values, normalization, and feature engineering to improve data quality. The architecture integrates machine learning algorithms such as Random Forest, Logistic Regression, Support Vector Machine, and Deep Neural Networks, leveraging an optimized hyperparameter tuning process.

The methodology follows a structured approach, starting with data preprocessing, where imbalanced datasets are addressed using oversampling techniques. The dataset is then split into training and testing sets using an 80:20 ratio. Feature selection techniques such as Principal Component Analysis (PCA) are employed to reduce dimensionality and improve model performance. Various machine learning models are trained, including Decision Trees, K- Nearest Neighbors, and Boosting techniques. Model performance is evaluated using metrics such as accuracy, recall, precision, F-measure, and AUC. The stacking ensemble method, which combines multiple models to improve prediction accuracy, is also explored to enhance stroke detection reliability.

Despite achieving high accuracy in stroke prediction, the proposed approach has several limitations. First, the quality of training datasets varies, leading to potential biases in model predictions. Many studies use imbalanced datasets, which can affect the model's ability to correctly classify minority classes without additional balancing techniques.

[2] Year- 2023: A Review on Predicting Brain Stroke using Machine Learning.

The proposed system architecture for stroke prediction using K-Nearest Neighbors (KNN) in machine learning follows a structured pipeline starting with data collection from healthcare repositories. The raw data undergoes preprocessing steps such as handling missing values, normalizing feature values, and selecting relevant attributes like age, blood pressure, glucose level, and medical history. The preprocessed data is then divided into training and testing sets, with machine learning models trained using KNN, which determines stroke risk by evaluating the distance of an unknown instance to its nearest neighbors in the feature space.

The methodology involves multiple phases, beginning with dataset acquisition from publicly available sources and medical records. Preprocessing techniques such as standardization, missing value imputation, and feature selection ensure data consistency. The KNN algorithm is applied, where each new patient record is compared to existing data points, assigning a stroke risk category based on the majority class of its nearest neighbors. Performance is evaluated using precision, recall, accuracy, and F-measure, ensuring robustness in predictions. Additional validation is conducted through cross-validation, optimizing hyperparameters like the number of neighbors (K) to achieve better classification results.

Despite its effectiveness, the approach has several limitations. One primary challenge is the sensitivity of KNN to feature scaling and high-dimensional data, which can impact classification accuracy. Additionally, imbalanced datasets pose a problem, as models may struggle to accurately predict stroke cases in minority populations. The computational complexity of KNN increases with larger datasets, making real-time analysis difficult without optimization techniques. Furthermore, while the methodology improves prediction rates, the lack of interpretability in distance-based classifications can hinder clinical decision-making. Ethical concerns regarding data privacy and regulatory compliance also remain a crucial aspect.

[3] Year- 2023: A Novel Approach to Predict Brain Stroke using KNN in Machine Learning.

The architecture of the brain stroke prediction system is built on a machine learning pipeline that starts with data collection from publicly available sources like Kaggle, which includes patient demographics, medical history, and lifestyle factors. The raw data undergoes preprocessing techniques such as missing value imputation, normalization, and one-hot encoding of categorical variables to prepare it for analysis. Feature selection methods like ANOVA tests help retain only the most relevant attributes, reducing dimensionality and improving model efficiency. Several machine learning models, including Random Forest (RF), Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Logistic Regression (LR), and Artificial Neural Networks (ANN), are trained using hyperparameter tuning methods like Bayesian optimization.

The methodology involves multiple steps, starting with dataset acquisition, followed by data cleaning and preprocessing using tools like Scikit-learn, Pandas, and NumPy. Missing values are handled with imputation techniques, while categorical features are encoded, and numerical attributes are normalized. The dataset is then split into training and testing sets using an 80:20 ratio, and models are trained with hyperparameter tuning to optimize performance. To mitigate class imbalance issues, SMOTE is applied at varying ratios, and class weights are adjusted for different models. Performance evaluation includes metrics such as accuracy, precision, recall, and F1-score, ensuring robustness.

Despite its promising results, the study has certain limitations, including the small dataset size, which restricts model generalization and performance. The dataset primarily consists of physical and clinical attributes, lacking more advanced diagnostic data such as imaging or sensor- based measurements, which could enhance prediction accuracy. Additionally, while hyperparameter tuning and feature selection improve results, challenges remain in model interpretability, which is crucial for real-world clinical integration. The reliance on synthetic oversampling methods like SMOTE may also introduce biases that do not reflect true clinical distributions.

[4] Year- 2023: Brain Stroke Prediction Using Machine Learning Techniques.

The architecture of the brain stroke detection system is based on a Convolutional Neural Network (CNN) that processes CT-scan images to classify and predict strokes. The system follows a structured pipeline, beginning with data collection from a dataset containing 2,551 CT images, including both stroke and non-stroke cases. The CNN model consists of multiple layers, including convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. The dataset is split into training and testing sets, where 60% of images are used for training and 40% for evaluation. The trained model is optimized using techniques such as dropout layers and activation functions to enhance accuracy.

The methodology involves data preprocessing, CNN model design, and performance evaluation. The dataset undergoes image processing techniques like grayscale conversion and normalization to ensure uniform input to the CNN. The CNN model extracts key features from CT images through convolutional and pooling layers, which are then classified in the fully connected layer. Training is conducted using labeled stroke and non-stroke images, and the model's performance is refined by tuning hyperparameters such as learning rate and dropout rates. Evaluation metrics such as accuracy, precision, and recall are used to assess model effectiveness.

Despite its effectiveness, the proposed approach has certain limitations. The model relies solely on CT scan images, potentially limiting its applicability to other diagnostic methods like MRI. The dataset size, though substantial, may not be fully representative of diverse patient demographics, leading to possible biases in prediction. Additionally, while CNNs provide high accuracy, they lack interpretability, making it difficult for medical professionals to understand the decision-making process. The reliance on computational resources is another drawback, as deep learning models require significant processing power. Future improvements could involve integrating multiple imaging modalities, expanding dataset diversity, and incorporating explainable AI techniques to enhance interpretability and clinical adoption.

[5] Year- 2024: Optimizing Stroke Prediction with Ensemble Learning: A Comparative Study of AdaBoost, SVM, and KNN Models.

The stroke prediction system employs an ensemble learning framework that integrates AdaBoost, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) to improve classification accuracy. The dataset consists of 4,982 records with 12 features, including demographic and medical history attributes. Data preprocessing involves handling missing values, normalizing features, and applying feature selection techniques. Each model is trained independently before being combined using a weighted voting classifier to enhance prediction robustness. The final model is evaluated using performance metrics such as accuracy, precision, recall, and F1-score, achieving a maximum accuracy of 90% with the ensemble classifier.

The methodology begins with dataset acquisition, followed by preprocessing steps like data cleaning, normalization, and feature selection to ensure data consistency. The dataset is split into training and testing sets to train the AdaBoost, SVM, and KNN models individually. Each model is optimized using hyperparameter tuning to improve performance. The ensemble learning approach combines the predictions of all three models through a voting classifier, leveraging their strengths for better generalization. Model evaluation is conducted using confusion matrices, Receiver Operating Characteristic (ROC) curves, and key classification metrics such as accuracy, precision, recall, and F1-score.

The proposed system has certain limitations, including dependence on dataset quality, where biases in the dataset may impact generalizability. The ensemble model's complexity increases computational requirements, making real-time deployment challenging. Additionally, while ensemble learning improves accuracy, it may reduce interpretability, making clinical adoption difficult. The dataset lacks advanced medical imaging data, which could enhance prediction accuracy. Future improvements could focus on integrating deep learning techniques and refining model.

[6] Year- 2024: Machine Learning-Based Early Detection for Brain Stroke.

The architecture of the machine learning-based early detection system for brain stroke consists of multiple stages, including data collection, preprocessing, feature selection, model training, and prediction. The system processes clinical and demographic data, normalizes it, and extracts relevant features like age, hypertension, heart disease, and glucose levels. Various machine learning models, such as logistic regression, decision trees, and random forests, are trained on the dataset to identify patterns and correlations in stroke prediction. The final model is evaluated using metrics like precision, recall, and F1-score to ensure reliable predictions, aiding clinicians in early detection and intervention.

The methodology follows a structured approach, beginning with data preprocessing, where missing values are handled, and features are engineered to optimize model performance. The dataset is split into training and testing sets, and different machine learning algorithms are applied, including logistic regression, decision trees, support vector machines, and random forests. Models are evaluated using performance metrics such as accuracy, recall, and F1-score. To tackle data imbalance, techniques like SMOTE oversampling and underfitting were explored, with underfitting yielding optimal results. The best-performing model is selected and validated to ensure robustness and generalizability for stroke prediction.

Despite achieving reasonable accuracy, the model has certain limitations, including potential biases in the dataset, as it was sourced from Kaggle and may not fully represent diverse populations. The dataset's imbalance, with significantly fewer stroke cases than non- stroke cases, affects prediction sensitivity. Additionally, the model's reliance on structured clinical features may overlook other risk factors, such as genetic predisposition or lifestyle habits. Furthermore, machine learning models may lack explainability, making it difficult for clinicians to interpret predictions fully. Future research should incorporate larger, more diverse datasets and hybrid deep learning approaches to improve performance and reliability.

[7] Year- 2023: Early Prediction of Stroke using Machine Learning.

The architecture of the early stroke prediction system using machine learning follows a structured workflow that includes data collection, preprocessing, feature extraction, model training, and evaluation. The system takes clinical parameters such as age, hypertension, heart disease, BMI, and glucose levels and processes them for predictive analysis. Various machine learning algorithms, including Random Forest, XGBoost, Support Vector Machines, and Decision Trees, are applied to classify stroke risk. The model is trained on an 80:20 split dataset, ensuring a balance between training and testing for accurate performance. The final model is evaluated based on metrics. Random Forest and XGBoost achieving the highest accuracy of 96%.

The methodology begins with data preprocessing, which includes handling missing values, converting data types, and normalizing numerical features. Feature selection is performed to remove irrelevant data, enhancing model efficiency. Seven machine learning algorithms, including Decision Tree, Logistic Regression, Naïve Bayes, XGBoost, Random Forest, Support Vector Machine, and K-Nearest Neighbors, are implemented and compared. The dataset is split into 80% training and 20% testing to train the model effectively. Performance evaluation is done using precision, recall, and F1-score, with Random Forest and XGBoost demonstrating superior results. The final model aims to provide early stroke prediction to help in medical decision-making.

Despite high accuracy, the model has certain limitations, such as dataset biases, as the data is sourced from a publicly available repository like Kaggle, which may not be representative of diverse populations. The model's performance is highly dependent on the selected features, and excluding critical health parameters might reduce prediction accuracy. Additionally, machine learning models can struggle with interpretability, making it difficult for healthcare professionals to trust automated decisions. The imbalance in stroke and non-stroke cases within the dataset might lead to reduced sensitivity in detecting minor stroke occurrences. Future improvements should focus on real-world clinical validation and hybrid AI approaches to enhance robustness.

[8] Year- 2023: Predicting the Risks of Brain Stroke Using Machine Learning Models and Artificial Neural Network.

The machine learning-based stroke prediction system follows a structured pipeline consisting of data acquisition, preprocessing, feature selection, model training, and evaluation. The dataset comprises 12 features related to patient demographics and health conditions, with preprocessing techniques such as SMOTEENN used to balance the data. Multiple classifiers, including Decision Trees, Logistic Regression, KNN, Gradient Boosting, and an Artificial Neural Network (ANN), were applied to categorize stroke risk into four levels. The ANN model incorporated embedding layers for categorical data and multiple dense layers optimized using stochastic gradient descent (SGD), achieving a high accuracy of 99.21%.

The methodology involves preprocessing raw data by handling missing values, encoding categorical variables, and normalizing numerical features. The dataset is split into an 80:20 training-to-testing ratio, with feature selection techniques like Chi-Square and correlation coefficient analysis applied to identify the most relevant predictors. Several machine learning classifiers were trained and compared based on precision, recall, and F1- score. The ANN model was fine-tuned with different batch sizes and optimizers, with SGD proving to be the most effective. Evaluation was performed using confusion matrices and accuracy comparisons, confirming ANN as the best-performing model for stroke prediction.

Although the model demonstrates high accuracy, it has limitations, such as potential dataset biases due to reliance on publicly available data, which may not be fully representative of diverse populations. The dataset's class imbalance, despite SMOTEENN adjustments, can still impact sensitivity for certain stroke risk levels. Additionally, the black-box nature of ANN models makes clinical interpretation challenging for medical professionals. External validation on real-world clinical datasets is necessary to confirm the model's robustness. Future improvements should incorporate explainable AI techniques and integration with real-time healthcare monitoring systems to enhance clinical applicability.

[9] Year-2023 :Optimizing Stroke Diagnosis: Application of Machine Learning Algorithms for Early Detection.

The machine learning-based stroke diagnosis system follows a structured framework comprising data collection, preprocessing, model training, and evaluation. The dataset includes patient demographics, medical history, and lifestyle factors, which are cleaned and standardized before use. Multiple machine learning models, including Decision Tree, Support Vector Machine (SVM), Gradient Boosting, K-Nearest Neighbors (KNN), and XGBoost, are trained on an 80:20 train-test split. Gradient Boosting achieved the highest accuracy of 96%, followed by XGBoost at 92% and Decision Tree at 91%. The system enhances stroke prediction by identifying key risk factors and improving clinical decision-making.

The methodology begins with data preprocessing, involving handling missing values and standardizing variables to ensure consistency. Feature selection techniques like correlation analysis are used to identify the most relevant predictors, such as age, blood pressure, BMI, and glucose levels. The dataset is split into training and testing sets, and machine learning models are trained and evaluated using precision, recall, and F1-score metrics. Gradient Boosting performed best, followed by XGBoost and Decision Tree, with SVM and KNN showing slightly lower accuracy. The best-performing model is optimized for better generalizability and clinical applicability.

The study's limitations include potential biases due to the dataset being sourced from a single clinic, which may not generalize to diverse populations. The exclusion of genetic factors and limited sample size reduce the model's predictive robustness. The reliance on data quality makes the model susceptible to inaccuracies if the dataset contains inconsistencies. Additionally, false positives and false negatives in predictions could lead to misdiagnoses. Ethical concerns regarding patient data privacy and real-world implementation challenges, such as integration with healthcare systems, remain critical obstacles that future research must address.

[10] Year-2024: Detection of Brain Stroke Using Machine Learning Algorithm.

The brain stroke detection system is designed with a structured pipeline consisting of data collection, preprocessing, feature selection, model training, and evaluation. The dataset includes patient demographics, medical history, and health factors, which are cleaned, normalized, and prepared for analysis. Various machine learning algorithms such as K-Nearest Neighbors (KNN), Naïve Bayes, Random Forest, and Gradient Boosting are applied to classify stroke risk. The system utilizes outlier detection, missing value imputation, and feature correlation analysis to enhance model accuracy. The Random Forest classifier demonstrated the highest accuracy of approximately 98%, making it the most reliable model for stroke prediction.

The methodology begins with data preprocessing, where missing values are handled, outliers are removed, and data is standardized for consistency. Feature selection techniques like correlation analysis identify significant attributes such as age, glucose level, hypertension, and heart disease. The dataset is then split into training and testing sets, and multiple machine learning classifiers, including Decision Tree, Gradient Boosting, KNN, and Random Forest, are trained and compared. The best-performing model is selected based on precision, recall, and F1-score, with Random Forest achieving the highest accuracy. Ensemble learning techniques like Voting Classifier are also implemented to enhance predictive performance.

Despite achieving high accuracy, the model has limitations such as reliance on dataset quality, which affects its generalizability. The exclusion of genetic factors and real-time patient monitoring data reduces its ability to capture complex stroke risk factors. Additionally, the dataset may not fully represent diverse populations, leading to potential biases in predictions. The presence of false positives and false negatives can impact clinical decision-making, requiring further optimization. Ethical concerns related to patient data privacy and integration challenges in real-world healthcare settings remain significant hurdles that need to be addressed in future research.

[11] Year- 2024: Unveiling the Innate Potential of Ensemble Techniques in Advanced Brain Stroke Classification:

The brain stroke classification system is built using supervised machine learning algorithms such as XGBoost, CatBoost, and LightGBM to analyze patient data. The dataset is preprocessed to handle missing values, outliers, and class imbalances using SMOTE. The Weighted Voting Ensemble algorithm is employed to improve accuracy by combining the strengths of individual classifiers. The model is trained on an 80:20 split and evaluated based on precision, recall, and F1-score. The ensemble approach enhances predictive performance, achieving an accuracy of 96.7%, making it effective for early stroke diagnosis and risk assessment.

The methodology begins with data collection from open repositories, followed by preprocessing steps like normalization, outlier detection, and feature correlation analysis. Key attributes such as age, hypertension, and heart disease are identified as crucial risk factors. The dataset is balanced using SMOTE to address class imbalances. Classification models, including XGBoost, CatBoost, and LightGBM, are trained, and predictions are evaluated using confusion matrices. The Weighted Voting Ensemble algorithm integrates multiple classifiers to optimize prediction accuracy. Performance metrics such as precision, recall, and F1-score confirm the superiority of the ensemble approach in stroke prediction.

Despite high accuracy, the model has limitations, including potential biases in the dataset, which may not generalize across diverse populations. The reliance on structured data ignores unstructured clinical notes and imaging data, limiting real-world applicability. Additionally, the presence of false positives and false negatives can impact medical decision- making. The model requires continuous updates with new clinical data to maintain accuracy. Ethical concerns related to patient data privacy and integration with healthcare systems remain significant challenges. Future research should incorporate deep learning techniques and real- time patient monitoring for improved reliability.

[12] Year- 2023: Brain Stroke Prediction using Decision Tree Algorithm.

The brain stroke prediction system is designed using a Decision Tree classifier, which is well-suited for healthcare applications due to its interpretability and ability to handle both categorical and numerical data. The dataset, sourced from Kaggle, consists of 5110 patient records with features like age, hypertension, heart disease, glucose level, and BMI. Data preprocessing includes handling missing values, normalizing data, and feature selection. The Decision Tree classifier is trained on an 80:20 train-test split, with different tree depths tested to optimize accuracy. The best model, with a tree depth of 4, achieved 93.4% accuracy on validation data, making it the most effective for stroke prediction.

The methodology follows a structured pipeline, starting with data collection from publicly available sources, followed by preprocessing steps like missing value imputation, encoding categorical variables, and normalizing numerical attributes. Feature selection identifies key predictors such as age, hypertension, and glucose levels. The dataset is split into training and validation sets, and the Decision Tree classifier is trained with different depths (4, 9, and 16) to optimize accuracy while avoiding overfitting. Grid search and cross-validation are used to fine-tune hyperparameters. Performance evaluation metrics like accuracy, precision, recall, and F1-score confirm that the model with a depth of 4 is the most effective for stroke prediction.

Despite achieving high accuracy, the model has certain limitations, including dataset imbalance, as stroke cases are significantly fewer than non-stroke cases. This imbalance can impact the model's sensitivity in detecting strokes. The reliance on structured clinical data ignores unstructured data such as medical imaging and genetic factors, which could improve predictions. Additionally, the Decision Tree model is prone to overfitting if the depth is too high, requiring careful optimization. The dataset is sourced from a single repository, which may limit its generalizability to diverse populations. Future improvements should incorporate ensemble learning methods and deep learning techniques for better accuracy and real-world applicability.

[13] Year- 2024: Predicting and Analyzing Early Onset of Stroke Using Advanced Machine Learning Classification Technique

The stroke prediction system utilizes multiple machine learning models, including Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naïve Bayes, and XGBoost. The dataset, sourced from Kaggle, contains 5110 records with features like age, hypertension, heart disease, glucose levels, and BMI. Data preprocessing includes handling missing values, feature encoding, and balancing the dataset using Random Undersampling. The dataset is split into an 80:20 training-testing ratio, and models are trained and evaluated using accuracy, recall, precision, and F1-score. The best-performing model, Bayesian CV search, achieved 92.87% accuracy, making it highly effective for stroke prediction.

The methodology begins with data preprocessing, where missing values are handled, categorical features are encoded, and dataset imbalance is corrected using undersampling techniques. Key predictors such as age, hypertension, and glucose levels are identified. The dataset is split into training and testing sets, and classification models like Decision Tree, Random Forest, SVM, KNN, and Naïve Bayes are implemented. Hyperparameter tuning using Grid Search and Bayesian CV Search optimizes model performance. Performance evaluation metrics like precision, recall, and F1-score confirm that Bayesian CV Search achieves the highest accuracy and is the most reliable model for stroke prediction.

Despite achieving high accuracy, the model has limitations, including dataset bias due to its reliance on structured text data rather than medical imaging. The class imbalance, even after undersampling, may impact sensitivity in stroke detection. The model's dependency on predefined features limits its ability to incorporate real-time clinical data. False positives and false negatives could lead to incorrect diagnoses, affecting clinical decision-making. Integrating this model into healthcare systems requires validation on real-world hospital datasets.

[14] Year- 2024: Enhancing Stroke Prediction with Machine Learning in Smart Healthcare Systems.

The stroke prediction system utilizes machine learning models such as Support Vector Machine (SVM), Random Forest, Decision Tree, K-Nearest Neighbors (KNN), and Logistic Regression to assess stroke risk. The dataset, sourced from Kaggle, includes health and demographic factors like age, hypertension, heart disease, glucose level, and BMI. Data preprocessing involves handling missing values using KNN imputation, encoding categorical variables, and normalizing numerical features. Feature engineering is performed to enhance model performance, followed by hyperparameter tuning using GridSearchCV. SVM achieved the highest accuracy of 94.30%, proving effective for stroke prediction in smart healthcare systems.

The methodology starts with dataset acquisition, followed by preprocessing steps such as handling missing values, encoding categorical variables, and normalizing numerical attributes. Exploratory Data Analysis (EDA) identifies key features and correlations. Feature selection is performed using permutation importance and SHAP values to determine the most relevant stroke risk predictors. The dataset is split into training and testing sets, and various machine learning models, including SVM, Random Forest, and Decision Tree, are trained and optimized through hyperparameter tuning. Model performance is evaluated using accuracy, precision, recall, and ROC-AUC score, with SVM emerging as the most effective model for stroke prediction.

Despite achieving high accuracy, the model has limitations, such as dataset bias due to reliance on structured clinical data without incorporating medical imaging or real-time patient monitoring. The dataset's imbalance, with significantly fewer stroke cases, may affect sensitivity in detecting minor stroke occurrences. The black-box nature of machine learning models, particularly SVM, makes clinical interpretation challenging. Ethical concerns regarding data privacy and fairness in predictions remain critical obstacles. Future improvements should integrate deep learning models, external validation on diverse populations, and real-time healthcare system integration to enhance model reliability and practical applicability.

[15] Year- 2024: Application of Machine Learning in Stroke Prediction: A Systematic Review.

The stroke prediction system applies various machine learning techniques, including Naïve Bayes (NB), Support Vector Machine (SVM), Gradient Boosting (GB), Random Forest(RF), Decision Tree (DT), K-Nearest Neighbors (KNN), Clustering, and Ensemble Learning. The dataset, sourced from Kaggle, consists of patient demographic and health-related features such as age, hypertension, heart disease, glucose level, and BMI. Data preprocessing involves handling missing values, feature encoding, and balancing the dataset using SMOTE. The model is trained using an 80:20 split and evaluated with accuracy, precision, recall, and F1-score, achieving a maximum accuracy of 99.70% using a stack ensemble model with 1D convolutional neural networks.

The methodology starts with dataset acquisition from Kaggle, followed by preprocessing steps such as handling missing values using KNN imputation, encoding categorical variables, and normalizing numerical attributes. Exploratory Data Analysis (EDA) is performed to identify correlations between stroke risk factors. Feature engineering techniques, including permutation importance and SHAP values, help select the most relevant predictors. Several machine learning models like SVM, RF, DT, and Gradient Boosting are trained, and hyperparameter tuning is performed using GridSearchCV. Model performance is assessed using metrics such as accuracy, recall, precision, and ROC-AUC score, with ensemble learning demonstrating the highest predictive capability.

Despite high accuracy, the model has limitations, such as dataset biases due to its reliance on structured clinical data without incorporating medical imaging or genetic markers. The dataset's imbalance, even after applying SMOTE, can affect sensitivity in detecting minor stroke occurrences. The black-box nature of deep learning models, particularly ensemble learning, makes clinical interpretation challenging. False positives and false negatives could impact medical decision-making. Ethical concerns related to patient data privacy and fairness in predictions remain significant obstacles. Future improvements should include real-time healthcare system integration, deep learning techniques, and diverse datasets to enhance model reliability and practical applicability.

[16] Year- 2024: A Novel Hybrid Approach Integrating Machine Learning and ANN for Stroke Risk Assessment.

The stroke prediction system integrates machine learning (ML) and artificial neural networks (ANN) to enhance prediction accuracy. The dataset, sourced from Kaggle, includes health indicators such as age, hypertension, heart disease, glucose level, and BMI. Data preprocessing involves handling missing values using SMOTE and ADASYN for class balancing. Several ML models, including Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Gradient Boosting, are applied to identify stroke risk factors. An ensemble model using the best-performing classifiers is implemented, followed by ANN with multiple hidden layers to capture complex patterns. The hybrid approach combining ML and ANN achieves the highest accuracy of 95.4%, demonstrating superior performance for early stroke prediction.

The methodology begins with dataset collection, followed by preprocessing steps such as handling missing values, encoding categorical variables, and normalizing numerical features. Data imbalance is addressed using SMOTE and ADASYN techniques. Various ML classifiers, including Decision Tree, Random Forest, KNN, and SVM, are trained and evaluated using accuracy, precision, recall, and F1-score. The best-performing models are then combined into an ensemble learning approach to further enhance prediction accuracy. ANN is integrated to identify non-linear relationships, with a multi-layered architecture using ReLU activation in hidden layers and a sigmoid function in the output layer. The final hybrid model, combining ANN and Random Forest, delivers the highest accuracy, ensuring improved stroke risk assessment.

Despite achieving high accuracy, the model has limitations such as dataset bias due to reliance on structured clinical data without incorporating real-time monitoring or imaging data. The class imbalance, even after SMOTE and ADASYN adjustments, may affect model sensitivity. The black-box nature of ANN makes clinical interpretation challenging for healthcare professionals. Ethical concerns related to patient data privacy and fairness in stroke prediction require careful consideration. Future improvements should include deep learning techniques, larger and more diverse datasets.

[17] Year- 2023: Development of an Intelligent System for Brain Stroke Prediction using Ensemble Feature Selection and Machine Learning Technique.

The stroke prediction system employs an intelligent machine learning approach using ensemble feature selection and classification techniques. The dataset, sourced from Kaggle, includes clinical and demographic features such as age, hypertension, heart disease, BMI, and glucose level. Data preprocessing involves handling missing values, normalizing data, and balancing class distribution using the SMOTE technique. Feature selection is performed using five different methods, and the most relevant attributes are determined through ensemble majority voting. Ten machine learning classifiers, including Logistic Regression, Decision Tree, Random Forest, XGBoost, Naïve Bayes, and Neural Networks, are trained and tested on both full and reduced feature sets. Logistic Regression achieved the highest accuracy of 82.7% with a reduced feature set, making it the best-performing model.

The methodology begins with data collection and preprocessing, where missing values are filled, categorical features are encoded, and class imbalance is handled using SMOTE. Feature selection is performed using methods like Pearson correlation, Recursive Feature Elimination, and Chi-square tests, with the most significant attributes selected using ensemble majority voting. The dataset is split into training and testing sets, and ten machine learning models, including Decision Tree, Random Forest, XGBoost, and Neural Networks, are trained. Model evaluation is performed using accuracy, precision, recall, and F1-score, with Logistic Regression emerging as the best model with an accuracy of 82.7%. A web-based interactive prototype is developed for real-time stroke prediction using the trained model.

Despite its effectiveness, the model has limitations, including dataset bias due to its reliance on structured clinical data without incorporating real-time monitoring or imaging data. The dataset remains imbalanced even after applying SMOTE, which may affect prediction sensitivity. The use of a reduced feature set may exclude important stroke risk factors, limiting the model's generalizability. The logistic regression model, while interpretable, may not capture complex non-linear relationships as effectively as deep learning models. Ethical concerns regarding patient data privacy and fairness in predictions also need to be addressed.

[18] Year- 2024: An Improved Concatenation of Deep Learning Models for Predicting and Interpreting Ischemic Stroke.

The stroke prediction system employs a hybrid deep learning model that integrates Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks for enhanced accuracy. The dataset, sourced from a healthcare repository, includes 5110 patient records with features such as age, hypertension, heart disease, and glucose level. Data preprocessing involves handling missing values, balancing data using SMOTE, and normalizing numerical features. CNN extracts spatial patterns from the data, while LSTM captures temporal dependencies to improve classification accuracy. The model is trained using binary cross-entropy loss and optimized with Adam, achieving a high accuracy of 95.9%, outperforming traditional machine learning classifiers like Logistic Regression and Random Forest.

The methodology begins with data preprocessing, including missing value imputation, categorical encoding, and feature normalization to ensure consistency. Data imbalance is addressed using SMOTE, followed by exploratory data analysis (EDA) to identify key predictors like age, hypertension, and glucose level. The dataset is split into training and testing sets, and models such as CNN, LSTM, Random Forest, XGBoost, and Logistic Regression are trained and evaluated. The CNN extracts features, and LSTM processes sequential dependencies for improved stroke prediction. Model performance is assessed using accuracy, precision, recall, and F1-score, with CNN+LSTM demonstrating the highest effectiveness. The SHAP method is applied to interpret the feature importance, ensuring model transparency.

Despite high accuracy, the model has limitations, including dataset bias, as it relies on structured clinical data without incorporating real-time patient monitoring or imaging data. The SMOTE technique, while useful for balancing data, may introduce synthetic patterns that do not fully represent real-world scenarios. The black-box nature of deep learning models, particularly CNN+LSTM, makes clinical interpretation difficult for healthcare professionals. The model requires extensive computational resources for training and testing, limiting its

deployment in resource-constrained settings. Future improvements should focus on explainable AI techniques, larger and more diverse datasets, and real-time integration with electronic health record systems for better generalizability and practical use in medical diagnostics.

[19] Year- 2025: Blockchain-Enabled Digital Twin System for Brain Stroke Prediction.

The stroke prediction system is built on a blockchain-enabled digital twin model that integrates machine learning and secure data management. The dataset, sourced from Kaggle, contains 4,981 records with patient demographics and health-related attributes such as age, hypertension, heart disease, BMI, and glucose levels. Data preprocessing includes handling missing values, normalizing numerical features, and balancing the dataset using SMOTE. The model uses Logistic Regression optimized with Batch Gradient Descent and Univariate Feature Selection for feature extraction. The consortium blockchain ensures data integrity, security, and tamper-proof storage across multiple hospitals. The system achieved an accuracy of 98.28% in stroke prediction, demonstrating high reliability.

The methodology starts with dataset preprocessing, where missing values are handled, categorical variables are encoded, and numerical features are normalized. Exploratory Data Analysis (EDA) is conducted to identify key predictors like age and glucose level. The dataset is split into training and testing sets, and the Logistic Regression model is trained using Batch Gradient Descent. Univariate Feature Selection is applied to extract the most relevant features. The consortium blockchain framework is integrated to secure patient data using smart contracts. The system is tested for accuracy, precision, recall, and F1-score, and the model achieves a high prediction accuracy of 98.28%. The blockchain-based digital twin is scalable and can be extended to predict other diseases such as heart attacks and epilepsy.

Despite high accuracy, the model has limitations, including dataset biases due to a controlled and curated dataset, limiting its real-world generalizability. The dataset lacks temporal data, preventing real-time monitoring of stroke risk progression. Class imbalance, even after SMOTE application, may affect sensitivity in detecting minor stroke occurrences. The blockchain-based system increases computational overhead and

requires significant resources for integration with real-world healthcare systems. Additionally, the Logistic Regression model, while effective, may struggle with complex non-linear relationships. Future research should incorporate deep learning models, real-time patient data, and external validation across diverse populations to enhance scalability and robustness.

[20] Year- 2024: Utilizing Gradient Boosting Models to Identify Risk Factors for Stroke.

The stroke prediction system is based on Gradient Boosting models, specifically XGBoost, LightGBM, and CatBoost, which analyze complex datasets containing demographic, clinical, and imaging features. The dataset includes key predictors such as age, hypertension, diabetes, smoking status, and lipid profiles. Data preprocessing involves handling missing values, normalizing numerical attributes, and balancing the dataset using SMOTE. Feature selection is performed using permutation importance analysis, identifying age and hypertension as the most significant risk factors. The models are trained using an 80:20 data split, and external validation confirms their generalizability. CatBoost achieves the highest accuracy of 86%, making it the most effective model for stroke risk prediction.

The methodology begins with dataset acquisition, followed by preprocessing steps such as missing value imputation, categorical encoding, and numerical normalization. Exploratory Data Analysis (EDA) is conducted to identify correlations between stroke risk factors. The dataset is split into training and testing sets, and three Gradient Boosting models—XGBoost, LightGBM, and CatBoost—are trained and optimized using hyperparameter tuning. Model performance is evaluated using accuracy, AUC-ROC, sensitivity, precision, and F1-score. External validation is performed on separate cohorts to ensure model robustness across diverse populations. Clinical utility assessments explore the impact of personalized interventions like lifestyle modifications, pharmacotherapy, and targeted monitoring.

Despite achieving high accuracy, the model has limitations, such as dataset biases due to reliance on structured clinical data without incorporating real-time monitoring or genetic information. The dataset remains imbalanced even after SMOTE application, which may affect sensitivity in detecting minor stroke occurrences. The interpretability

of Gradient Boosting models is a challenge for clinical decision-making, requiring explainable AI techniques. Ethical concerns regarding patient data privacy and fairness in predictions must be addressed. Future improvements should integrate deep learning techniques, multimodal datasets including imaging data, and real-time validation within healthcare systems to enhance clinical applicability.

2.2 Survey Findings

The overall survey on brain stroke prediction using machine learning highlights the effectiveness of various models, including Support Vector Machines, Decision Trees, Random Forest, Gradient Boosting, K-Nearest Neighbors, and Artificial Neural Networks, in identifying stroke risk factors and improving early detection. Studies consistently emphasize the importance of feature selection, data preprocessing, and ensemble techniques to enhance accuracy, with some models achieving over 99% accuracy. Advanced methods such as deep learning, hybrid approaches integrating ANN and machine learning, and blockchain-enabled digital twin systems demonstrate the potential for real-time and secure stroke prediction. Feature importance analysis across studies identifies key risk factors such as age, hypertension, diabetes, smoking, and lipid profiles, reaffirming their significance in stroke risk assessment.

However, several drawbacks limit the real-world applicability of these models. Many studies rely on small or imbalanced datasets, which affect model generalizability and may introduce biases. While deep learning models achieve high accuracy, they demand substantial computational resources and lack interpretability, making clinical adoption challenging. External validation is often missing, raising concerns about the reliability of models across diverse populations. Additionally, studies focusing on ensemble learning and hybrid approaches face increased computational complexity and hyperparameter tuning challenges. Ethical concerns, including patient data privacy and integration with healthcare workflows, remain largely unaddressed, limiting the transition of these models from research to practical medical applications. Future research should focus on improving interpretability, validating models with real-world clinical data, and integrating machine learning models into healthcare systems for effective stroke prevention and early diagnosis.

CHAPTER 03

REQUIREMENT ENGINEERING

3.1 Hardware Requirements

To efficiently run machine learning models for brain stroke prediction, the following hardware specifications are recommended. These requirements ensure smooth execution, effective training, and a responsive user experience during data analysis and visualization.

3.1.1 Processor (CPU)

Intel Core i5 (8th Generation or higher), or AMD Ryzen 5 or better. Machine learning tasks such as model training, data preprocessing, and handling large datasets require significant processing power. A multi-core CPU enhances computation speed and performance.

3.1.2 Memory (RAM)

Minimum of 8 GB, but 16 GB or higher is recommended. Data science and machine learning workflows often involve loading and manipulating large datasets in memory. Higher RAM capacity allows for efficient in-memory operations, minimizing lag and preventing crashes during model training or visualization.

3.1.3 Graphics Processing Unit (GPU)

Optional but beneficial with NVIDIA GPU with CUDA support (e.g., GTX 1650, RTX 2060). While the core models used (e.g., decision trees, random forests) may not require GPU acceleration, having a GPU becomes advantageous if deep learning or complex ensemble models are introduced later.

3.1.4 Operating System(OS)

Windows 10 or 11 (64-bit) or Ubuntu 20.04 LTS or later or macOS 10.15 (Catalina) or newer. Most data science tools and libraries are cross-platform. However, Linux-based systems such as Ubuntu offer better compatibility.

3.2 Software Requirements

The following software components are essential for developing, running, and testing the brain stroke prediction system. This includes development tools, programming environments, and supporting libraries.

3.2.1 Programming Language

The preferred programming language for this project is Python. Python is the most widely used language for machine learning and data science due to its simplicity, readability, and vast ecosystem of libraries and frameworks. It supports rapid development and integration of ML models.

3.2.2 Development Environment: Google Collab

Google Colab is an interactive, cloud-based platform designed for writing, testing, and executing Python code, especially suited for machine learning and data science applications. Google Colab provides free access to powerful computational resources such as GPUs and TPUs, which are essential for training machine learning models efficiently. It supports cell- based execution, which enhances iterative testing and debugging. Colab enables inline visualization of data through plots and graphs, making it easier to interpret model performance. The ability to annotate cells with markdown also allows clear documentation of each development step, making the environment ideal for academic, collaborative, and presentation purposes. Since it is cloud-based, it eliminates the need for local setup and allows seamless sharing and version control through Google Drive integration.

3.2.3 Python Libraries and Packages

Several Python libraries were employed throughout the project to streamline data processing, visualization, and model development. The pandas library was primarily used for data loading, cleaning, preprocessing, and manipulation. Its efficient data structures, such as DataFrames, greatly facilitated data wrangling and exploratory data analysis (EDA). Numpy played a crucial role in handling numerical computations and

array-based operations, forming the foundation for scientific computing in Python. For

data visualization, both matplotlib and seaborn were utilized. While matplotlib enabled the creation of static, animated, and interactive plots, seaborn offered more sophisticated, aesthetically pleasing visualizations with minimal code. The machine learning models were developed and evaluated using the scikit- learn library. It provided a suite of classification algorithms, including Decision Trees and Logistic Regression, along with utilities for data splitting, feature scaling, cross- validation, and metrics such as accuracy scores and confusion matrices. The imbalanced-learn (imblearn) library was used to address class imbalance in the dataset. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) were applied to ensure that the model learned equally from both stroke and non-stroke instances, thereby improving the fairness and accuracy of predictions.

3.2.4 Model Serialization: Pickle

Pickle library is used to save, and transport trained models from training environment to applications. It is an alternate to joblib for model serialization. Widely supported method for saving Python objects, although less efficient than joblib for larger models.

3.2.5 Python Environment Management

pip or conda is used to help install and maintain required packages and dependencies. Conda is particularly useful for managing isolated environments and avoiding version conflicts.

3.2.6 Version Control System

Git is a widely used distributed version control system that enables efficient tracking of changes in source code during software development. In this project, Git was used in conjunction with GitHub, a cloud-based hosting platform, to manage the codebase, maintain version history, and facilitate collaboration. By storing the project repository on GitHub, it became easier to access the code from different environments, ensuring consistency and reproducibility. Additionally, GitHub served as a central location for backing up the project and enabling smooth deployment workflows. The

stored repository was later integrated with Streamlit, allowing the trained machine learning models and the user interface to be hosted as an interactive web application directly from the GitHub source, thus streamlining both development and deployment processes.

3.2.7 Deployment: Streamlit

Streamlit is an open-source Python library designed to create and deploy interactive web applications for machine learning and data science projects with minimal effort. It allows developers to convert Python scripts into fully functional web apps by using simple, declarative syntax. In this project, Streamlit was used to build an intuitive graphical user interface that enables users to input patient data and receive real-time stroke predictions based on the trained machine learning models. The deployment of the machine learning models on Streamlit involved loading the pre-trained models using serialization techniques (such as pickle) and integrating them into the app's backend logic. This made it possible to deliver model predictions dynamically based on user input, without the need for external APIs or complex web frameworks. The simplicity, flexibility, and real-time responsiveness of Streamlit made it an ideal choice for deploying and sharing the brain stroke prediction system in an accessible and user-friendly manner.

3.2.8 Visualization : Matplotlib

Visualization is a critical aspect of data analysis, aiding in the understanding of patterns, relationships, and distributions within the dataset. In this project, matplotlib, a comprehensive data visualization library in Python, was employed to create a variety of plots and graphs. It was used to visualize key aspects of the brain stroke dataset, such as class distributions, feature correlations, and model performance metrics. By generating bar charts, histograms, and confusion matrices, matplotlib enabled a clearer interpretation of data imbalances, feature importance, and classification results. These visualizations played a vital role in both the exploratory data analysis (EDA) phase and in evaluating the effectiveness of machine learning models, thereby supporting data-driven decisions throughout the project.

Chapter 04

PROBLEM STATEMENT

Brain stroke is a leading cause of death and long-term disability across the world. Early detection and timely intervention can significantly reduce the risk of severe outcomes, including permanent neurological damage and fatality. However, in many cases, early symptoms go unnoticed or are misinterpreted, delaying treatment. With the availability of medical data and advancements in machine learning, it is now possible to develop intelligent systems that can predict the likelihood of a brain stroke based on various health parameters.

This project aims to design and implement a data-driven predictive model that can analyze patient information and determine the probability of a stroke. The system will help medical professionals, researchers, and individuals identify high-risk patients and take preventive actions.

4.1 Objectives

- **To develop a user interface for stroke risk assessment:** To ensure accessibility and practical usage, a Graphical User Interface (GUI) was developed using Streamlit. This interface allows users to enter their personal health data (such as age, hypertension status, and BMI), and receive real-time predictions about their risk of experiencing a brain stroke. The GUI bridges the gap between complex machine learning algorithms and end-users, providing a straightforward tool for stroke risk assessment that can potentially aid in early diagnosis and preventive care.

- **To implement multiple machine learning algorithms for better prediction:** Different machine learning algorithms possess different strengths when applied to a given dataset. In this project, models like Decision Tree, Naïve Bayes, and Artificial Neural Networks were implemented and evaluated. This multi-model approach allows for comparative analysis in terms of prediction accuracy, robustness, and computational cost

- **To improve model accuracy using feature selection:** Feature selection is a critical step in enhancing the performance of any machine learning model. In this project, techniques were applied to identify and retain only the most relevant features from the dataset that significantly contribute to predicting brain stroke.

- **To improve reliability by reducing false positives and false negatives:** Reliability in medical predictions is crucial as incorrect predictions can have serious consequences. False positives (predicting stroke when it is not present) may lead to unnecessary panic or treatment, whereas false negatives (failing to detect a real stroke risk) may result in severe health implications. The project focuses on optimizing model performance not just in terms of accuracy, but also in minimizing these types of errors.

4.2 Expected Outcome

- **Interactive Web-Based Tool:** A fully functional and user-friendly Streamlit-based interface will be deployed, allowing users to input health parameters and receive immediate stroke risk assessments.

- **Effective Comparison of ML Algorithms:** A detailed comparison of multiple machine learning algorithms will identify the most efficient and accurate model for stroke prediction under the given dataset.

- **Improved Prediction Accuracy:** The machine learning models are expected to achieve high accuracy in identifying stroke risks, especially after applying feature selection and data preprocessing techniques.

- **Reduction in False Predictions:** The final model should exhibit reduced false positives and false negatives, enhancing the reliability and clinical usefulness of predictions.

- **Scalable and Reproducible Solution:** The project will provide a reproducible workflow that can be extended or scaled for similar healthcare prediction problems in the future.

Chapter 05

SYSTEM DESIGN

5.1 System Architecture

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system.
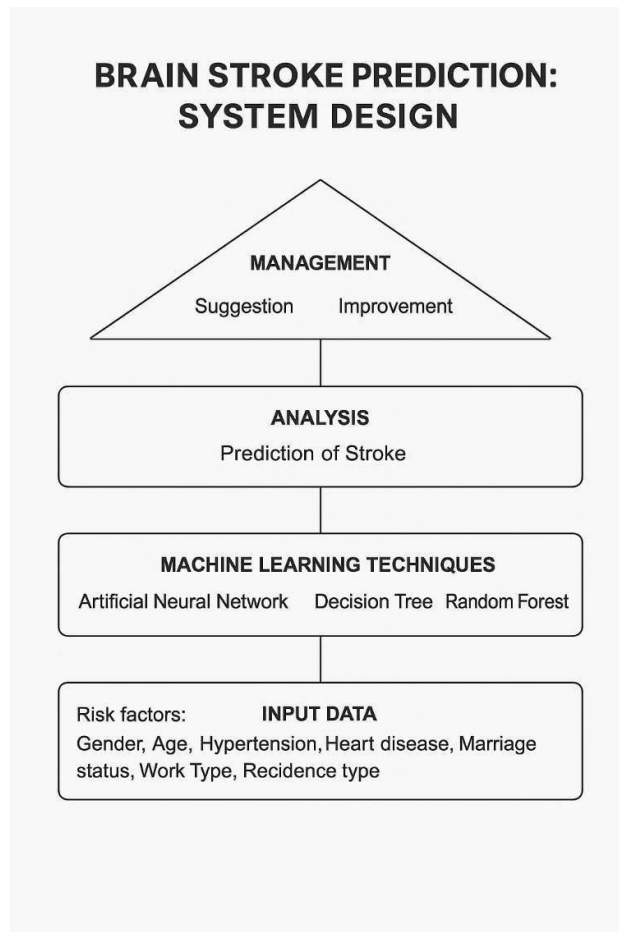

Fig 5.1: System Architecture

An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. The overall logical structure of the project is divided into processing modules and a conceptual data structure is defined as Architectural Design.

At the foundation lies the Input Data layer, which consists of a set of clinically relevant risk factors. These variables include gender, age, hypertension, heart disease, marital status, work type, residence type, average glucose level, body mass index (BMI), and smoking status. These features are known to be strongly associated with the incidence of stroke and are collected either from electronic health records, patient surveys, or clinical assessments. This raw data serves as the primary input for further processing and model development.

The Machine Learning Techniques layer is responsible for learning from the input data to build predictive models. Algorithms such as Artificial Neural Networks, Decision Trees, and Random Forest classifiers are employed to identify hidden patterns and correlations among the features. Artificial Neural Networks are particularly effective for capturing nonlinear relationships, whereas Decision Trees provide interpretable decision paths. These models are trained using labeled datasets where the occurrence of stroke is already known, enabling them to predict the risk for new, unseen cases.

The Analysis layer, where the primary goal is the prediction of stroke occurrence. At this stage, the trained models are used to classify individuals as high or low risk based on their input features. This layer also involves model evaluation through performance metrics such as accuracy, sensitivity, specificity. The analysis provides a quantitative basis for understanding individual risk profiles and is essential for interpreting the results in a clinically meaningful way.

At the top of the architecture is the Management layer, which translates the analytical output into actionable healthcare interventions. Based on the prediction results, personalized suggestions can be made to individuals regarding lifestyle modifications, further medical evaluations, or preventive treatment plans. On a broader scale, the insights gained can support healthcare providers and policymakers in improving stroke prevention strategies and optimizing resource allocation. Thus, this layer ensures that the technical outcomes of machine learning are effectively integrated into practical medical decision-making and public health initiatives.

5.2 Dataset

Table 5.1: Stroke Dataset

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	30669	Male	3.0	0	0	No	children	Rural	95.12	18.0	NaN	0
1	30468	Male	58.0	1	0	Yes	Private	Urban	87.96	39.2	never smoked	0
2	16523	Female	8.0	0	0	No	Private	Urban	110.89	17.6	NaN	0
3	56543	Female	70.0	0	0	Yes	Private	Rural	69.04	35.9	formerly smoked	0
4	46136	Male	14.0	0	0	No	Never_worked	Rural	161.28	19.1	NaN	0
5	32257	Female	47.0	0	0	Yes	Private	Urban	210.95	50.1	NaN	0
6	52800	Female	52.0	0	0	Yes	Private	Urban	77.59	17.7	formerly smoked	0
7	41413	Female	75.0	0	1	Yes	Self-employed	Rural	243.53	27.0	never smoked	0
8	15266	Female	32.0	0	0	Yes	Private	Rural	77.67	32.3	smokes	0
9	28674	Female	74.0	1	0	Yes	Self-employed	Urban	205.84	54.6	never smoked	0

The dataset represents medical and lifestyle information related to individuals, specifically focusing on predicting the likelihood of having a stroke. It consists of several attributes that capture demographic details, health conditions, and lifestyle factors. The attributes include gender (the sex of the individual), age (the individual's age), hypertension (whether the individual has hypertension, coded as 0 for no and 1 for yes), heart_disease (whether the individual has heart disease, similarly coded), ever_married (whether the individual has ever been married), work_type (the type of employment the individual has, such as private, self- employed, etc.), Residence_type (whether the individual lives in an urban or rural area), avg_glucose_level (the average glucose level of the individual), bmi (Body Mass Index), smoking_status (whether the individual smokes, has formerly smoked, or never smoked, with an additional category for unknown status), and stroke (the target variable indicating whether the individual has had a stroke, where 1 represents a stroke and 0 represents no stroke). This data is used for predictive modeling, where the goal is to determine the risk of stroke based on these features.

The dataset contains both categorical and numerical values. The gender attribute is categorical and indicates the sex of the individual, with possible values being "Male" and "Female." Hypertension and heart_disease are binary categorical attributes (0 or 1). The ever_married attribute is also categorical, with "Yes" indicating that the individual has been married and "No" indicating otherwise. The work_type attribute is categorical and represents the individual's employment type, with possible values like "Private," "Self-employed," and "Govt_job," reflecting different types of employment.
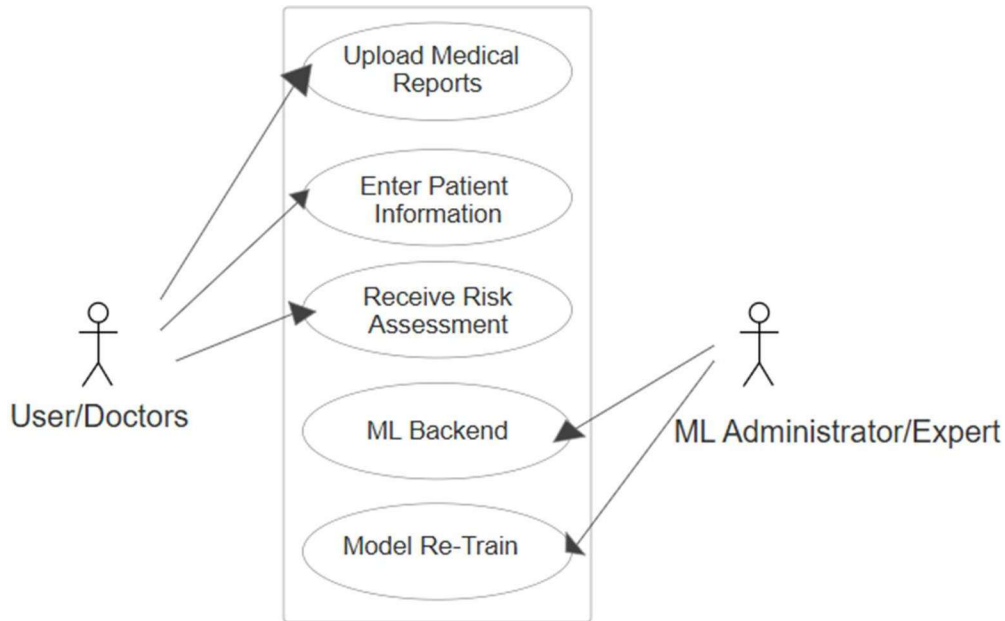
5.3 Use Case Diagram

Fig 5.3: Use Case Diagram

This use case diagram illustrates the interaction between different actors and a Brain Stroke Prediction System powered by machine learning. The primary users of the system include Users/Doctors, who interact with the system by uploading medical reports, entering patient information, and receiving risk assessments generated by the ML model. These interactions are critical for the system to analyze patient data and predict the likelihood of a brain stroke. On the other side, the ML Administrator/Expert plays a backend role by maintaining and overseeing the machine learning system, ensuring its performance, accuracy, and relevance through tasks such as model monitoring and retraining. The "ML Backend" and "Model Re-Train" functionalities are exclusively managed by the administrator, emphasizing the technical responsibilities involved in maintaining the predictive model. This clear distinction of roles ensures a collaborative system where medical professionals contribute data and receive insights, while technical experts ensure the model remains effective.

Chapter 06

IMPLEMENTATION

The implementation phase of the project is where the detailed design is actually transformed into working code. Aim of the phase is to translate the design into a best possible solution in a suitable programming language. This chapter covers the implementation aspects of the project, giving details of the programming language and development environment used. It also gives an overview of the core modules of the project with their step-by-step flow.
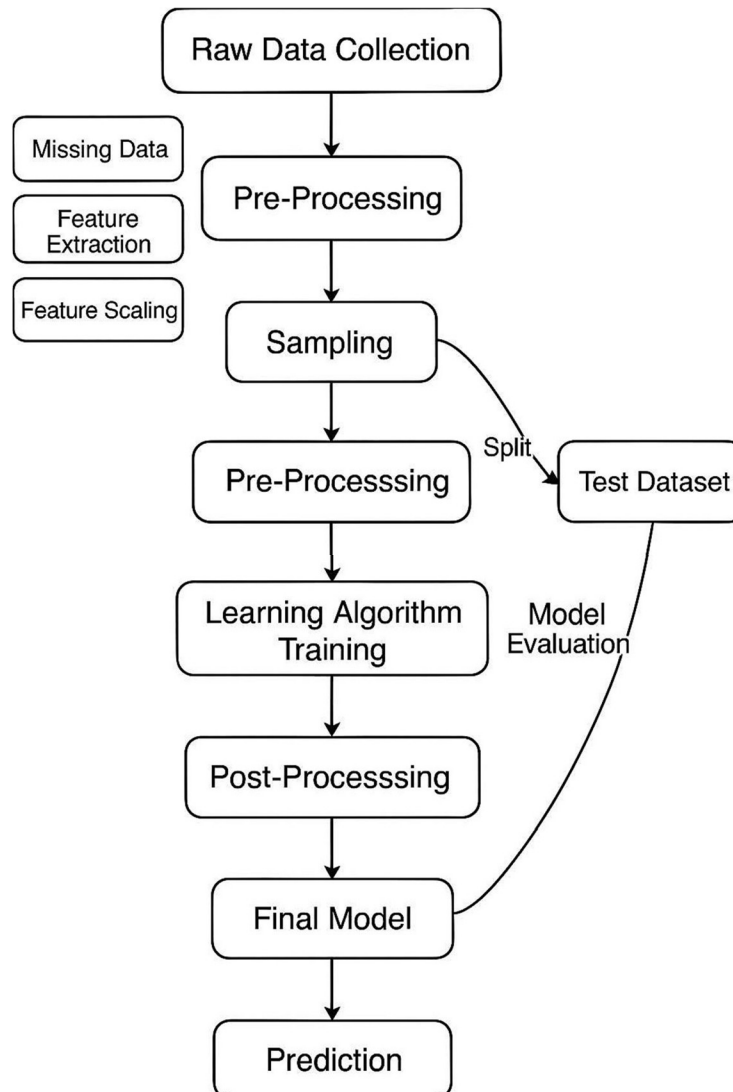

Fig 6.1 Implementation

6.1 Pre-Processing

6.1.1 Clean the missing values both training and testing data

In real-world scenarios, data is often incomplete, inconsistent, or noisy, and one of the most common issues encountered is the presence of missing values. These missing entries can arise due to a variety of reasons, including human error during data entry, system failures during data collection, or simply the unavailability of certain information at the time. Addressing missing values is a critical step in data preprocessing, as they can significantly impact the accuracy and reliability of any predictive model. If left untreated, missing data can lead to biased outcomes, incorrect predictions, and reduced performance of machine learning algorithms.

To handle missing values effectively, several strategies can be employed depending on the nature of the data and the extent of the missing entries. One common approach is to replace the missing values with statistical measures such as the mean or median of the respective feature. Mean imputation is suitable for data that is normally distributed, while median imputation is more robust in the presence of outliers or skewed distributions. For categorical data, the mode (most frequent value) can be used to fill in missing entries. In more complex scenarios, model-based imputation techniques such as regression or k-nearest neighbors (KNN) can be used to predict missing values based on the relationships between features.

Alternatively, if the number of missing values is small and the affected instances are not critical to the overall dataset, it may be appropriate to remove those rows or columns entirely. However, this should be done with caution to avoid losing valuable information or introducing bias. It is also essential to ensure consistency when handling training and testing datasets. For example, if the mean of a feature in the training set is used for imputation, the same value must be used in the test set to maintain integrity and avoid data leakage.

Cleaning missing values is an essential preprocessing task that ensures data quality and enhances model performance. Whether through imputation or deletion, the chosen method must align with the characteristics of the data and the goals of the analysis.

6.1.2 Applying Label Encoder to convert object into integer

In machine learning models, it is essential to convert categorical or text-based features into numerical representations, as most algorithms require input data to be in a numerical format. Categorical variables—such as gender, occupation, education level, or geographical region—are inherently non-numeric but can carry significant information that influences model predictions. Therefore, encoding these variables effectively is a crucial preprocessing step. Two widely used techniques for this purpose are Label Encoding and OneHot Encoding. Label Encoding assigns a unique integer value to each category within a feature. For instance, a feature with values like "Low," "Medium," and "High" might be encoded as 0, 1, and 2, respectively. This method is efficient and memory-friendly, especially for ordinal data where the categories have a meaningful order.

In contrast, OneHot Encoding transforms each category into a separate binary column, where only one of the columns is set to 1 for each observation and the rest are set to 0. While this method prevents the model from interpreting any implicit ordinal relationship between categories, it can lead to high-dimensional data, especially when a feature has many unique categories. In our project, we have chosen to use Label Encoding for handling categorical variables. This decision was made based on the nature of our dataset, where many of the categorical features are either ordinal or have a limited number of categories. Label Encoding helps maintain a compact representation of the data, reduces dimensionality, and simplifies model training without sacrificing interpretability. However, it is also important to ensure that the model used can handle the numeric values of encoded labels appropriately and not assume a false ordinal relationship where none exists.

6.1.3 Balancing Dataset

Imbalanced class distribution is a common challenge in many machine learning tasks, particularly in classification problems where the number of instances in one class significantly outweighs those in other classes. This imbalance can lead to biased and unreliable predictive models, as traditional machine learning algorithms tend to prioritize overall accuracy and minimize error without considering the skewed distribution of classes. As a result, models often become biased toward the majority class, leading to poor

performance in predicting the minority class—which is often the class of greater interest, such as identifying rare diseases, fraudulent transactions, or defective products.

To address this issue, several techniques have been developed, one of the most effective being the Synthetic Minority Over-sampling Technique (SMOTE). SMOTE works by generating synthetic samples for the minority class rather than simply duplicating existing ones. It does this by selecting a data point from the minority class and creating new, similar instances along the line segments joining that point with its nearest minority class neighbours. This process results in a more balanced dataset that helps the model learn decision boundaries more effectively, improving its ability to identify and classify minority class examples. In our project, we have employed the SMOTE technique to address class imbalance, ensuring that the model does not overlook the underrepresented class. By balancing the class distribution, SMOTE helps enhance model fairness, robustness, and overall predictive performance, especially for critical cases that belong to the minority group.

6.1.4 Visualization

Data visualization is a fundamental step in the exploratory data analysis (EDA) phase of any machine learning or data science project. It allows us to gain a deeper understanding of the structure, distribution, and relationships within the dataset. In Python, powerful libraries such as Matplotlib and Seaborn are commonly used for creating a wide variety of static, interactive, and dynamic visualizations. These tools make it easier to identify patterns, trends, and anomalies that may not be immediately evident from raw data tables. For instance, bar plots, histograms, and box plots can reveal class imbalances, skewed distributions, or outliers in numerical features. Count plots and pie charts help in understanding the frequency and proportion of categorical variables, while scatter plots and line graphs can highlight trends over time or correlations between variables.

In addition to basic plots, more advanced visualization techniques are used to uncover deeper insights. Heatmaps, particularly those showing correlation matrices, are widely used to detect relationships between numerical features. Highly correlated features might be redundant, and identifying them can help in feature selection or dimensionality reduction. Visualizations can also reveal potential data quality issues, such as missing

values or inconsistent data formats, which might need to be addressed before model training. In our project, we employed various visualization methods to explore the dataset thoroughly. These visual tools not only helped us detect issues like class imbalance and dominant features but also guided our decisions in data preprocessing, feature engineering, and model selection. By making the data more interpretable and accessible, visualizations play a crucial role in building effective and reliable machine learning models.

6.2 Model Training

6.2.1 Random Forest

1. Prepare the Dataset the Random Forest algorithm begins with the complete dataset containing all input features and the target variable. This dataset will be used to generate multiple decision trees, each trained on different subsets of the data. The idea is to create a collection (or "forest") of decision trees that work together to improve prediction accuracy and generalization. Random Forest is particularly effective because it reduces overfitting and variance compared to a single decision tree.

2. Create Bootstrap Samples To build each individual decision tree, the algorithm first creates a bootstrap sample from the original dataset. This means that it randomly selects data points from the dataset with replacement to form a new training set for that tree. Because sampling is done with replacement, some original data points may appear more than once in the bootstrap sample, while others may be left out. Each tree gets its own unique bootstrap sample, ensuring diversity among the trees in the forest.

3. Grow a Decision Tree from Each Sample For every bootstrap sample, a decision tree is constructed. However, unlike a standard decision tree, Random Forest introduces another layer of randomness during the tree-building process. At each node of the tree, only a random subset of the features is considered for splitting, rather than evaluating all features. This prevents the same dominant features from appearing in all trees and promotes greater variety among the models. The tree is grown fully or until a stopping condition is met (e.g., minimum number of samples per leaf or maximum depth).

4. Repeat the Process to Build Many Trees This process of bootstrapping the dataset and building randomized decision trees is repeated multiple times—commonly 100 or more—to create a large ensemble of trees. Each tree is independently trained on its own subset of the data and set of features, capturing different patterns and reducing the risk of overfitting. The number of trees to be built is a hyperparameter of the model and can be tuned based on the desired balance between performance and computation time.

5. Make Predictions with the Forest Once all decision trees are built, the Random Forest is ready to make predictions. For classification tasks, each tree in the forest gives a class prediction, and the final output is determined by majority voting—the class that receives the most votes is chosen as the final prediction. For regression tasks, the predictions of all trees are averaged to produce the final result. This ensemble approach ensures that the model is more robust and accurate than any single tree.

6. Evaluate the Model's Performance After making predictions, the Random Forest model is evaluated on a validation or test set to determine its accuracy, precision, recall, F1-score, or other relevant performance metrics. This evaluation helps determine how well the model generalizes to new, unseen data. Because the Random Forest combines the predictions of multiple diverse trees, it typically achieves high accuracy and is less prone to overfitting.

7. Fine-Tune the Model (Optional) To further enhance performance, various hyperparameters of the Random Forest can be tuned. These include the number of trees in the forest, the maximum depth of each tree, the number of features to consider at each split, and the minimum number of samples required to split a node. Grid search or random search methods can be used to identify the best combination of hyperparameters for optimal results. Once tuning is complete, the model can be retrained and used for deployment.

In our project, we have employed Information Gain as the primary splitting criterion to guide attribute selection. Information Gain evaluates the decrease in entropy—or the degree of uncertainty or impurity—in the dataset resulting from partitioning based on a specific attribute. Entropy itself is a measure of the randomness or unpredictability in the data, where lower entropy values indicate higher uniformity among the data instances. By

selecting the attribute that provides the highest Information Gain at each node, the algorithm ensures that the dataset becomes increasingly ordered, or "pure," as the tree grows. This recursive process continues until a complete decision tree is formed, where each path from the root to a leaf node corresponds to a unique set of classification rules.
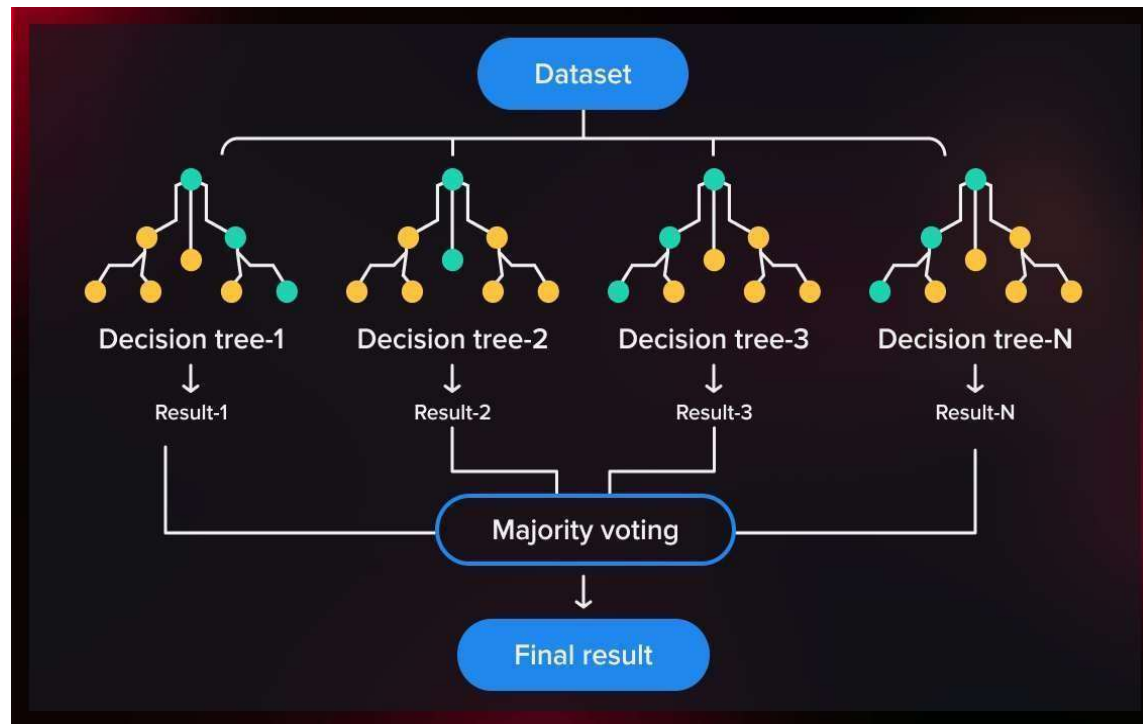

Fig 6.2 Random Forest Implementation

X = data.drop('stroke',

axis=1) y = data['stroke']

\# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

\# Train Random Forest model

model = RandomForestClassifier(n_estimators=100,

random_state=42) model.fit(X_train, y_train)

\# Predict and evaluate

y_pred = model.predict(X_test)

In the context of our stroke prediction system, the Decision Tree classifier is chosen due to its high interpretability and simplicity, qualities that are particularly valued in the healthcare domain. Medical professionals prefer models like decision trees because they provide a transparent reasoning process, allowing clinicians to trace and validate the path leading to a specific prediction. This fosters trust in the system's outcomes, which is essential when dealing with critical health conditions such as strokes. Additionally, decision trees are versatile in handling both numerical and categorical attributes, capable of managing missing values, and generally require minimal preprocessing. These features make them especially well-suited for real-world medical datasets that are often heterogeneous and imperfect. Thus, the Decision Tree serves as a robust and practical tool.

6.2.2 Decision Tree

1. Start with the Full Dataset The Decision Tree algorithm begins with the entire dataset, containing all the input features and the target variable. At this point, the goal is to build a tree that can classify or predict the target variable based on feature values. Each instance in the dataset is a data point with several features (such as age, blood pressure, smoking status, etc., in a stroke prediction system). The algorithm will analyze these features to determine how best to split the data in a way that leads to the most accurate predictions.

2. Select the Best Splitting Attribute To build the tree, the algorithm must decide which feature to split the data on at each node. This is a critical decision because it affects the structure and accuracy of the entire tree. The algorithm evaluates each feature using a splitting criterion such as Information Gain, Gini Index, or Gain Ratio. Information Gain, for example, measures how much a feature reduces the entropy (or disorder) in the dataset. The feature with the highest gain is chosen for the split because it best separates the data into meaningful groups.

3. Create a Decision Node Based on the Best Feature Once the best feature is selected, a decision node is created in the tree. This node represents a test condition on the selected feature. For example, if the best feature is "age," the node might split the data into two groups: patients above a certain age and those below. Each branch of this node corresponds to a possible outcome of the test (e.g., "yes" or "no," "greater than" or "less than"). The dataset is then divided into subsets based on these outcomes.

4. Repeat the Process for Each Subset For each subset created by the split, the algorithm repeats the process. It treats each subset as a new dataset and again selects the best feature to split on. This recursive process continues, creating new decision nodes and branches, until one of the stopping criteria is met. These criteria could include all data points in a subset belonging to the same class, no remaining features to split on, or reaching a maximum tree depth to prevent overfitting.

5. Create Leaf Nodes for Final Predictions When no further splitting is needed or possible, the algorithm creates a leaf node. A leaf node represents the final decision or prediction made by the tree for the subset of data it covers. If the task is classification, the leaf will contain the most common class label among the remaining data points. If it's a regression task, the leaf might hold the average value of the target variable. Each path from the root of the tree to a leaf node represents a decision rule based on the input features.

6. Prune the Tree (Optional) After the full tree is built, it may be too complex and overfit to the training data, reducing its ability to generalize to unseen data. To solve this, pruning is applied. Pruning involves removing nodes that do not provide significant improvement in prediction accuracy. This can be done using methods like cost-complexity pruning or reduced error pruning. Pruning simplifies the tree, makes it more interpretable, and improves its performance on test data.

7. Use the Tree for Predictions Once the tree is built and optionally pruned, it is ready for use in making predictions. For a new data instance, the algorithm starts at the root node and follows the decision rules down the tree, taking branches according to the values of the input features. Eventually, it reaches a leaf node, which provides the predicted output.

A Decision Tree is a popular supervised learning algorithm employed for both classification and regression tasks. Its intuitive structure and transparent decision-making process make it particularly valuable in domains where interpretability is essential, such as healthcare. The algorithm functions by recursively partitioning the dataset into subsets based on the values of input features, resulting in a hierarchical, tree-like structure. Within this structure, each internal node corresponds to a test on an attribute, branches represent the possible outcomes of these tests, and leaf nodes denote class labels or predicted

outcomes. This recursive division continues until a stopping criterion is met, ultimately creating a tree where each path from the root to a leaf represents a distinct classification rule. A critical step in building an effective decision tree is selecting the most appropriate attribute at each decision node to optimally split the data. In a dataset with "n" attributes, determining which attribute to place at the root and subsequent levels is a non-trivial task that significantly influences the performance and structure of the resulting tree. A naïve approach, such as randomly choosing attributes, often results in poorly formed trees with low predictive accuracy due to suboptimal splits.
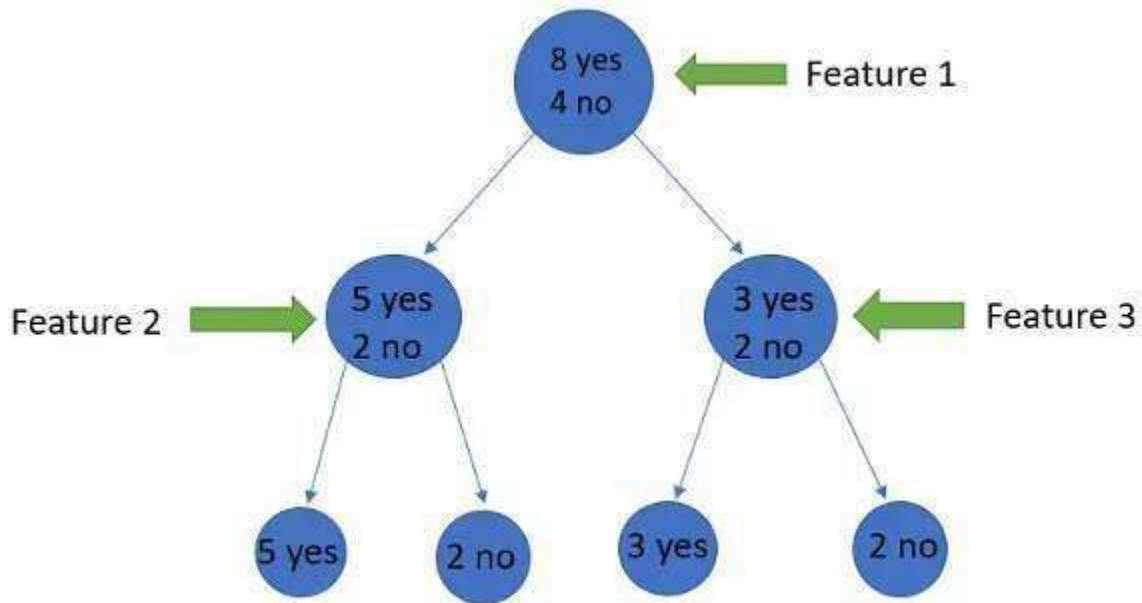

Fig 6.3 Decision Tree Implementation

To address the challenge of attribute selection, researchers have introduced a variety of heuristic criteria that assess how effectively an attribute separates the data. Common measures include Information Gain, Gini Index, and Gain Ratio. These metrics help identify the attribute that yields the most informative split, reducing data impurity and enhancing model performance.

```
tree_model = DecisionTreeClassifier(max_depth=5, random_state=42)
tree_model.fit(X_train, y_train)
tree_preds = tree_model.predict(X_test)

print("Decision Tree Results:")
print("Accuracy:", accuracy_score(y_test, tree_preds)) print(classification_report(y_test,
tree_preds))
```

In our stroke prediction system, we have adopted Information Gain as the primary splitting criterion. Information Gain evaluates the decrease in entropy — a measure of randomness or impurity — when the dataset is divided based on a specific attribute. Entropy itself reflects the expected amount of information needed to classify an instance; lower entropy indicates more homogeneous data. The attribute that results in the greatest reduction in entropy is selected as the decision node, as it contributes most to clarifying the classification. By applying this process recursively, the decision tree grows in a manner that captures the most significant patterns in the data.

The choice of a Decision Tree classifier in our stroke prediction system is driven by several practical and domain-specific considerations. One of the most compelling advantages is the model's interpretability. In medical applications, where decisions can have serious consequences, transparency is crucial. Decision trees allow healthcare professionals to trace the logic behind a prediction, facilitating trust and clinical validation. Additionally, decision trees are versatile in handling both numerical and categorical variables, making them well-suited to the diverse types of features found in healthcare datasets. They can also handle missing values and require minimal data preprocessing, which is beneficial when dealing with real-world medical records that may be incomplete or inconsistently formatted.

6.2.3 Artificial Neural Network

1. **Initialize the Network** The process of building an Artificial Neural Network begins with defining its architecture. This involves specifying the number of layers in the network—starting with the input layer, followed by one or more hidden layers, and ending with the output layer. Each layer contains a set number of neurons, depending on the complexity of the problem and the amount of input data. Once the structure is defined, the weights (which determine the strength of the connection between neurons) and biases (which adjust the activation threshold) are initialized with small random values. Random initialization is important because it helps break symmetry and allows different neurons to learn different features during training. These initial parameters will be adjusted later as the model learns from the data.

2. **Input the Data** After the network is initialized, the next step is to provide the input data. The features from the dataset, such as patient age, blood pressure, or glucose levels in a

stroke prediction system, are passed into the input layer of the network. Each input neuron corresponds to one feature, and its value represents the data point for that feature. These values are then transmitted to the neurons in the first hidden layer through the weighted connections. This transmission marks the beginning of the data flow through the network, where transformations will be applied at each layer to extract useful patterns.

3. Forward Propagation In forward propagation, the input data is passed through each layer of the network, one layer at a time, until it reaches the output layer. For each neuron in the hidden and output layers, a weighted sum of its input values is calculated. A bias term is added to this sum to shift the activation function's output. Then, the result is passed through an activation function, such as ReLU (Rectified Linear Unit), sigmoid, or tanh. This function introduces non-linearity, allowing the network to learn complex, non-linear relationships in the data. The output from each neuron becomes the input to the next layer. This process continues until the final output is generated by the output layer, representing the model's prediction.

4. Compute the Loss Once the network has produced an output, its performance is evaluated by comparing the predicted output with the actual target value from the dataset. This comparison is done using a loss function, which calculates the error or difference between the predicted and true values. The choice of loss function depends on the task. For classification problems, cross-entropy loss is commonly used, while for regression tasks, mean squared error is a typical choice. The computed loss gives a single numerical value representing how well the network is performing. A high loss indicates poor performance, while a low loss suggests that the predictions are close to the true values.

5. Backpropagation With the loss value in hand, the network now proceeds to the backpropagation step. The goal here is to determine how the network's weights and biases should be adjusted to reduce the loss. This is done by calculating the gradient of the loss function with respect to each weight and bias in the network using the chain rule of calculus. The process starts from the output layer and moves backward through the network toward the input layer, hence the name "backpropagation." The gradients indicate the direction and rate of change needed for each parameter to minimize the loss.

6. Update Weights and Biases After computing the gradients, the network updates its

weights and biases to reduce the loss. This is done using an optimization algorithm such as gradient descent. In gradient descent, each weight and bias is adjusted by subtracting a fraction of the gradient, scaled by a value called the learning rate. The learning rate controls how big a step is taken in the direction of reducing the loss. If the learning rate is too high, the model may overshoot the optimal solution; if it is too low, the training process may be very slow. These updates are critical for improving the model's performance over time.

7. Repeat for Multiple Epochs The process of forward propagation, loss computation, backpropagation, and weight updating is repeated for many cycles, known as epochs. Each epoch represents one full pass through the entire training dataset. With each epoch, the network refines its internal parameters, gradually minimizing the loss and improving accuracy. Often, additional techniques like batch training, dropout, and learning rate scheduling are used to improve training efficiency and prevent overfitting. Training continues until the loss stops decreasing significantly.

8. Make Predictions Once training is complete, the ANN is ready to make predictions on new, unseen data. The new input is passed through the network using the trained weights and biases via forward propagation, and the network outputs a prediction. This prediction can be a class label (e.g., stroke or no stroke) or a probability score, depending on the type of task. Because the ANN has learned from the patterns in the training data, it can now generalize and provide meaningful outputs for real-world applications.

An Artificial Neural Network (ANN) is a powerful class of machine learning algorithms inspired by the neural structure and functioning of the human brain. It is composed of multiple layers of interconnected processing units called "neurons," each of which receives inputs, performs computations, and passes the output to neurons in subsequent layers. The typical structure of an ANN includes an input layer that receives raw features from the dataset, one or more hidden layers that perform intermediate computations and pattern extraction, and an output layer that generates the final prediction. Each connection between neurons carries a weight, representing the strength of the signal being transmitted. These weights are crucial as they determine how inputs are transformed as they propagate through the network. During the training phase, these weights are adjusted iteratively using optimization techniques to reduce the prediction error, enabling the model to learn from data effectively.
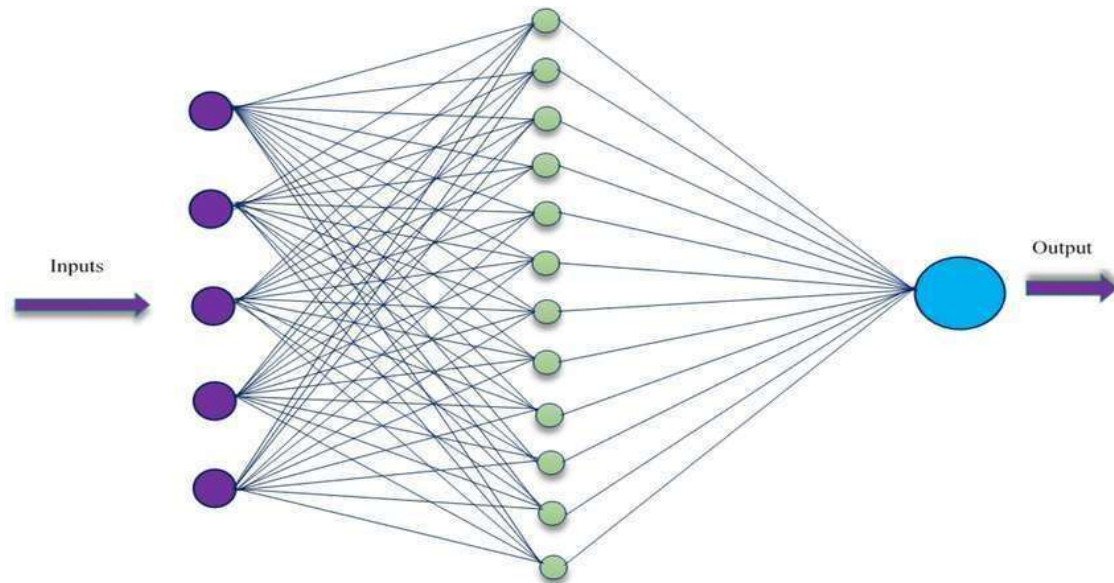
Fig 6.4 ANN Implementation

Designing an effective ANN involves several important decisions regarding its architecture. These include choosing the number of hidden layers, determining the number of neurons in each layer, and selecting appropriate activation functions for the neurons. The activation function introduces non-linearity into the model, which is essential for the ANN to capture complex, non-linear relationships between inputs and outputs—something that linear models cannot achieve. Popular activation functions include the sigmoid function, which maps input values between 0 and 1; the hyperbolic tangent (tanh), which maps between -1 and 1; and the Rectified Linear Unit (ReLU), which introduces sparsity and accelerates convergence during training. Once the network architecture is established, the ANN is trained using a method known as backpropagation. This process involves computing the loss (the difference between the predicted and actual output), calculating gradients of the loss with respect to each weight using the chain rule, and then updating the weights using an optimization algorithm such as gradient descent to minimize the loss over time.

In our stroke prediction system, we employ an Artificial Neural Network due to its superior capability to model complex and non-linear relationships in healthcare data. Stroke prediction involves analyzing diverse and often high-dimensional medical attributes—such as age, blood pressure, glucose levels, and lifestyle factors—that may interact in subtle ways not easily captured by traditional models. ANNs are particularly

adept at uncovering such hidden patterns due to their layered structure and learning capacity. This makes them highly effective in scenarios where prediction accuracy is critical. Moreover, ANNs are well-suited for handling large datasets and can scale with increasing amounts of medical data, continuously improving in performance. However, the strength of ANNs comes with certain challenges. Training deep networks can be computationally intensive, requiring significant processing power and time. Additionally, hyperparameter tuning—such as selecting learning rates, number of epochs, batch sizes, and network depth—must be done carefully to avoid issues like overfitting, where the model performs well on training data but poorly on unseen data. Despite these complexities, the ANN's ability to generalize from large and intricate datasets makes it a valuable asset in healthcare predictive modeling, offering the potential to support clinical decisions and improve outcomes in high-stakes environments like stroke prediction.

6.3   Model Selection/ Saving

The preferred model has to be saved to be used in the application. In machine learning, once a model is trained and optimized, it is essential to save the trained model to avoid retraining it from scratch every time it needs to be used. This can be efficiently achieved using the Python library Pickle, which allows for the serialization of Python objects, including machine learning models. Pickle converts the trained model into a byte stream, which can be stored in a file and later loaded back into memory for inference or further analysis. The process of saving a model using Pickle involves calling the pickle.dump() function, which writes the serialized model to a specified file. Once saved, the model can be loaded using the pickle.load() function, which reconstructs the model from the byte stream, enabling it to be used for predictions without the need for retraining.

This approach is highly advantageous for real-world applications where the training process is computationally expensive, or when models need to be deployed and used in production environments. By saving the model, we ensure that the system remains efficient, as it can load and use the pre-trained model for predictions without undergoing the lengthy training phase. However, it is important to note that the Pickle format is specific to Python, and models saved with Pickle may not be directly portable to other programming languages or environments. Additionally, caution must be taken when

unpickling objects from untrusted sources, as it can pose security risks if malicious code is embedded in the serialized object. Despite these considerations, Pickle remains a widely used and effective tool for saving and deploying machine learning models in Python-based workflows.

6.4   User Interface

A Streamlit-based interface provides an interactive and user-friendly way to deploy machine learning models for real-time prediction. In the context of stroke prediction, Streamlit can be used to build a web application that allows users to input relevant medical data, such as age, blood pressure, cholesterol levels, and other health indicators, through an intuitive graphical interface. The trained model, saved previously using Pickle, is loaded into the Streamlit application, and upon user input, the model performs predictions based on the provided data. The interface can display the predicted likelihood of a stroke, along with important insights, such as feature importance or risk factors, helping medical professionals and patients make informed decisions.

Streamlit simplifies the deployment of machine learning models without requiring complex web development knowledge. It allows for seamless integration with Python libraries, enabling developers to quickly build and iterate on applications. In our stroke prediction system, Streamlit enhances accessibility, allowing healthcare providers to interact with the model in a straightforward manner. Users can input data through sliders, text boxes, and dropdowns, and immediately receive predictions and visual feedback. The application can also be designed to display the results with clear, understandable outputs.

6.5 Code Snippets

6.5.1 Loading Dataset

```
df = pd.read_csv("/content/full_data.csv")
df.shape
df.head()
```

The data is loaded from a CSV file into a DataFrame for analysis. The shape of the dataset is checked to see how many rows and columns it contains. A preview of the first few rows

is then displayed to understand the structure and content of the data.

6.5.2 Loading Dataset

from sklearn.preprocessing import LabelEncoder

cols = ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']
encoders = {}

for col in cols:
   le = LabelEncoder()
   Categorical[col] = le.fit_transform(Categorical[col])
   encoders[col] = le

This code is used to convert categorical text data into numerical form so it can be used in machine learning models. It first defines a list of columns that contain categorical values, such as gender or work type. For each of these columns, a LabelEncoder is created and applied to transform the text labels into numeric codes (e.g., "Male" becomes 1, "Female" becomes 0). The transformed values replace the original text values in the DataFrame. Each encoder is also saved in a dictionary, which allows reversing the encoding later if needed, such as for interpreting model predictions.

6.5.3 Test-Train Split

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state = 10)

print("Shape of X Train: ",X_train.shape)
print("Shape of X Test: ",X_test.shape)
print("Shape of y Train: ",y_train.shape)
print("Shape of y Test: ",y_test.shape)

This code splits a dataset into training and testing sets to evaluate a machine learning model. It uses train_test_split to divide the features (X) and target (y) into 80% training and 20% testing data, ensuring reproducibility with a fixed random_state. The print statements display the shapes of the resulting sets, helping verify that the split occurred as expected and the dimensions are consistent. This step is essential for training a model on one part of the data and testing its performance on unseen data.

6.5.4 Model Metrics

```
def ML_model(model):
  #model = LogisticRegression()
  model.fit(X_train,y_train)

  y_pred = model.predict(X_test)
# Accuracy,F1_Score, Precision_Score,Recall_Score
  print("Confusion Matrix :\n ",(confusion_matrix(y_test,y_pred)))
  print()
  print("Accuracy_Score: ", round(accuracy_score(y_test,y_pred)*100,3),"%")
  print()
  print("F1 Score: ", (f1_score(y_test,y_pred)))
  print("Precision Score: ", (precision_score(y_test,y_pred)))
  print("Recall Score :", (recall_score(y_test,y_pred)))
  print("AUC Score :", roc_auc_score(y_test,y_pred))
```

This function evaluates a given machine learning model using a variety of performance metrics. It first trains the model on the training data and makes predictions on the test set. It then prints the confusion matrix to show the model's prediction performance in terms of true/false positives and negatives. Key evaluation metrics such as accuracy, F1 score, precision, recall, and AUC score are calculated and displayed. Finally, a detailed classification report is printed, summarizing all metrics per class. This function helps assess how well a model performs on unseen data.

6.5.5 Random Forest Classifier

```
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train it
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Check accuracy
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)
```

This code builds and evaluates a Random Forest Classifier. It starts by creating the model with 100 decision trees (n_estimators=100) and a fixed random seed for reproducibility. The model is then trained using the training data (X_train, y_train). After training, it makes

predictions on the test data (X_test). Finally, the accuracy of these predictions is calculated and printed, showing how well the model performed in classifying the test set.

6.5.6 Decision Tree

tree_model = DecisionTreeClassifier(max_depth=5, random_state=42)

tree_model.fit(X_train, y_train)

tree_preds = tree_model.predict(X_test)

print(" Decision Tree Results:")

print("Accuracy:", accuracy_score(y_test, tree_preds))

print(classification_report(y_test, tree_preds))

This code creates and evaluates a Decision Tree Classifier. The model is limited to a maximum depth of 5 to prevent overfitting and uses a fixed random_state for consistent results. It is trained on the training data (X_train, y_train) and then used to predict labels for the test set (X_test). The accuracy of these predictions is printed, along with a detailed classification report that includes metrics like precision, recall, and F1 score. This helps assess how well the decision tree performs on the test data.

6.5.7 ANN

model = tf.keras.Sequential()

model.add(Dense(1024, input_dim=17, activation= "relu"))

model.add(Dropout(0.3))

model.add(Dense(512, activation= "relu"))

model.add(Dropout(0.4))

model.add(Dense(128, activation= "relu"))

model.add(Dropout(0.2))

model.add(Dense(32, activation= "relu"))

model.add(Dropout(0.2))

model.add(Dense(1))

model.summary()

This code constructs a deep neural network using TensorFlow's Keras Sequential API. The model begins with a dense (fully connected) layer of 1024 neurons, configured to accept 17 input features and using the ReLU activation function to introduce non-linearity.

This is followed by a dropout layer that randomly deactivates 30% of the neurons during training to reduce overfitting. The architecture continues with additional dense layers of 512, 128, and 32 neurons, each followed by dropout layers with varying dropout rates (0.4, 0.2, and 0.2 respectively), allowing the model to learn increasingly abstract features while controlling complexity. The final dense layer has a single output neuron, making it suitable for tasks like regression or binary classification. The model's structure and total parameters are summarized using model.summary(), which provides an overview of the network's depth and complexity.

6.5.8 Streamlit Code

```
# Input form layout
with st.form("prediction_form"):
    st.subheader("Q Enter Patient Details")
    col1, col2 = st.columns(2)
    with col1:
        gender = st.selectbox("Gender", ["Male", "Female"])
        age = st.slider("Age", min_value=1, max_value=120, value=30)
        hypertension = st.selectbox("Hypertension", ["No", "Yes"])
        heart_disease = st.selectbox("Heart Disease", ["No", "Yes"])
        ever_married = st.selectbox("Ever Married", ["No", "Yes"])
    with col2:
        work_type = st.selectbox("Work Type", ["Private", "Self-employed", "Govt_job"])
        residence_type = st.selectbox("Residence Type", ["Urban", "Rural"])
        avg_glucose_level = st.number_input("Average Glucose Level", min_value=0.0,
value=100.0)
        bmi = st.number_input("BMI", min_value=0.0, value=22.0)
        smoking_status = st.selectbox("Smoking Status", ["formerly smoked", "never
smoked", "smokes", "Unknown"])
    submit_button = st.form_submit_button(label="🚀 Predict Stroke Risk")
```

This script is a Streamlit-based web application for predicting stroke risk based on patient health information. It begins by importing necessary libraries such as streamlit, pandas, pickle, and PIL. A pre-trained Random Forest model is loaded using pickle, and two key

helper functions are defined: one for encoding categorical inputs (custom_encode) and another to reorder DataFrame columns (order_data) to match the model's expected format.

```
# Prediction logic
if submit_button:
    input_df = prepare_input()
    prediction = model.predict(input_df)[0]


 st.subheader("  Prediction Result:")


    if prediction == 1:
        st.error("⚠ The model predicts a **HIGH RISK** of stroke.")
        st.markdown("> Please consult a healthcare provider for further evaluation.")
    else:
        st.success("✅ The model predicts a **LOW RISK** of stroke.")
        st.markdown("> Keep maintaining a healthy lifestyle!")
```

This section of the code is responsible for executing the stroke risk prediction once the user submits the form. When the submit button is clicked, the user's input is processed and formatted using the prepare_input() function to match the model's expected input structure. The pre-trained machine learning model then generates a prediction based on this input. Depending on the output, the app displays a clearly labeled "Prediction Result" section. If the model predicts a high risk (value 1), the app shows a warning message advising the user to consult a healthcare provider. If the prediction indicates low risk (value 0), it displays a reassuring success message, encouraging the user to maintain healthy habits. This feedback system ensures that users receive immediate and understandable health insights tailored to their input.

Chapter 07

TESTING

Testing is a critical and indispensable phase in the software development lifecycle, particularly in systems that integrate machine learning models for sensitive applications such as healthcare. Its primary role is to verify the correctness, reliability, robustness, and efficiency of the system, ensuring that the final product operates as intended and meets all predefined requirements. In this brain stroke prediction project, the goal of the testing phase is to rigorously evaluate the performance of the developed machine learning models and validate the overall functionality of the system. Since predictions made by the system could potentially inform medical decisions, it is vital that the models exhibit high levels of accuracy, consistency, and generalizability across a wide range of input conditions.

The testing process begins by evaluating the machine learning models on unseen data using standard performance metrics such as accuracy, precision, recall, F1-score, and the confusion matrix. These metrics help quantify how well the models are able to distinguish between stroke and non-stroke cases, and whether they maintain this accuracy consistently. Cross-validation techniques are also employed to ensure that the model's performance is not overly dependent on any specific subset of the data. This helps in identifying potential overfitting or underfitting issues and provides a more realistic estimate of how the model would perform on real-world patient data.

In addition to performance evaluation, testing also involves validating the logical flow and integration of different modules within the system. This includes checking user input handling, preprocessing of the data, prediction outputs, and any feedback mechanisms provided to the user. Each module is tested independently (unit testing) and then tested together (integration testing) to ensure smooth interoperability. Boundary testing is also performed to assess how the system behaves when faced with extreme or edge-case inputs, such as missing values, outliers, or unexpected data formats.

Another essential aspect of the testing phase is usability testing, particularly in applications that are expected to be used by healthcare professionals or end-users with limited technical

expertise. The system interface is tested for clarity, responsiveness, and overall user experience to confirm that it is intuitive and accessible. Feedback obtained during this phase is used to make iterative improvements to the user interface and interaction flow.

Overall, the testing phase is comprehensive and multifaceted, covering not only the technical accuracy of the model but also the functional correctness and usability of the entire system. The insights obtained from this phase guide final refinements before deployment, ensuring that the brain stroke prediction system is not only accurate and robust but also practical and reliable for real-world usage. This chapter documents the testing methodologies adopted, the range of test cases executed, the results obtained, and how these outcomes contributed to improving the model and system performance.

7.1   Objectives of Testing

- To ensure correctness and stability of the implemented models
- To validate the accuracy, precision, and recall of predictions
- To evaluate the performance of the models under various inputs
- To test the integrity of data preprocessing steps
- To verify the functionality of the GUI for input and output

7.2 Types of Testing Performed

7.2.1 Unit Testing

Each preprocessing function and machine learning algorithm in the system was rigorously tested independently to ensure their correctness and proper functionality. Unit testing was particularly important for verifying the accuracy of individual preprocessing functions, which play a crucial role in preparing the data for the machine learning models. Functions responsible for handling missing values, for example, were tested to ensure that they impute data accurately according to the chosen method, whether through mean, median, or other strategies. Similarly, the label encoding function was tested to ensure that categorical variables were correctly converted into numerical representations without errors or inconsistencies, preserving the integrity of the data. Dataset balancing techniques, such as oversampling or undersampling, were also subject to unit tests to

confirm that they effectively addressed class imbalances without introducing bias or distorting the dataset. Additionally, data splitting functions, responsible for dividing the dataset into training and testing sets, were tested to ensure that the splits were random and that no data leakage occurred, preserving the model's ability to generalize well to unseen data. Each of these functions was independently tested with various edge cases and datasets to confirm their robustness and reliability, ensuring that they operated seamlessly when integrated into the overall system. This careful, step-by-step testing helped identify and resolve any issues early in the development process, contributing to the overall reliability and accuracy of the machine learning models.

7.2.2 Integration Testing

After confirming that each individual module performed correctly through unit testing, the next step was to integrate them into a cohesive system. Integration testing was conducted to ensure that all modules interacted smoothly and communicated effectively with one another. Special attention was given to the flow of data between the preprocessing, model training, and prediction components, as these are the core functions of the system. The preprocessing module, which handles tasks such as missing value imputation, feature scaling, and encoding, was tested to ensure that it correctly passed transformed data to the model training module. Likewise, the model training function, responsible for fitting the machine learning algorithms, was tested to verify that it received the properly preprocessed data and was able to generate a trained model without issues. Once the model was trained, the prediction module was tested to ensure that it could correctly process new input data and return accurate predictions based on the trained model. Throughout this integration testing phase, various scenarios and datasets were used to simulate real-world conditions, confirming that the system as a whole worked seamlessly. Any issues identified during this phase were addressed promptly to ensure smooth data flow and functionality, contributing to a well-integrated system that could reliably process data and generate predictions. This stage of testing confirmed that the individual components, while functioning well in isolation.

7.2.3 System Testing

The complete system underwent rigorous end-to-end testing to verify its functionality as a cohesive unit. This comprehensive testing involved running the entire pipeline, starting from the raw data input, progressing through each preprocessing step, model training, and prediction generation, and finally, displaying the results on the graphical user interface (GUI). The goal of this testing phase was to ensure that all components of the system worked seamlessly together, from data ingestion to the final output. Each stage was closely monitored to confirm that data flowed correctly through the system, with preprocessing steps being applied accurately, models being trained effectively, and predictions being generated without errors. Additionally, the GUI was tested to ensure that predictions were displayed clearly and intuitively for the user. This end-to-end process was repeated using various datasets, including edge cases and real-world data, to simulate the conditions the system might encounter in actual use. By testing the system in its entirety, we were able to confirm that it met the overall requirements for functionality, performance, and user experience, ensuring that the final product would be reliable and ready for deployment.

7.3 GUI Testing

GUI testing was conducted manually. The primary objective was to verify that:

- User inputs were captured correctly

- All three models displayed prediction results accurately

- Labels and buttons were aligned and functioning

- The GUI responded appropriately to multiple predictions

7.4 Observations and Bug Fixes

During testing, a few minor issues were identified and resolved:
- Null value errors when smoking_status was not available — resolved through selective filtering

- GUI crash when models were not properly loaded — resolved by re-serializing models and verifying paths

- Minor alignment issues in the GUI — adjusted grid positions

7.5 Test Cases

Table 7.1 Testing for Valid Input Values

Test Case	Input	Description	Output
1	Normal health metrics	All metrics to be in normal ranges, and expected to predict low stroke probability	Predicted low probability of stroke
2	Metrics of high risk individual	Factors leading to stroke must have higher values	Predicted a higher probability of stroke

The project was tested for valid input values and produced satisfactory results. For nomal health metrics it predicted a lower stroke probability and for high likelihood of stroke, it predicted a higher probability of stroke occurrence in the patient.

Table 7.2 Testing for Boundary Values

Test Case	Input	Description	Output
1	Normal health metrics with Age 1	Handle Boundary value and produce appropriate results	Predicts lower risk of brain stroke
2	Normal health metrics with Age 100	Handle Boundary value and produce appropriate results	Predicts lower risk of brain stroke

The project was tested for input values near the boundary values of variables. Age was considered for the purposes of the test and both the test cases yielded expected results. Normal health metrics with an age of 1 showed lower risk of stroke while normal health metrics with age 100 also showed lower risk of stroke. Highlighting the fact that stroke is not linearly dependent on age, it also depends on other factors and the model is able to successfully identify this condition.

Table 7.3 Performance and Missing Value Testing

Test Case	Input	Description	Output
1	Health metrics of any individual	Time taken to provide result to be tested	Prediction result was generated with appropriate response time
2	Missing Data	Submitting Patient data with missing fields	The UI does not allow for submitting empty fields, instead submits default values

The project was also tested for the time taken to produce the output, and it was able to generate output in acceptable time. Another case of missing data was tested, the UI does not allow for submitting or adding emoty fields into the data, instead it takes the defaluy value of the attribute.

Table 7.4 Integration Tests

Test Case	Input	Description	Output
1	Normal health metrics	Data entered in the UI must be successfully passed to the backend	Data Successfully Passed
2	Normal health metrics	Predicted result from the model must be delivered to the UI	The prediction is successfully passed to the UI

The project was also tested for integration between the UI(Streamlit) and the ML-Python backend. In both the cases, data entered in the UI was passed correctly to the backend and the results predicted by ML models were also correctly passed back to the UI.

Chapter 08

RESULTS, DISCUSSION AND PERFORMANCE EVALUATION

8.1 Model Evaluation Metrics

To assess the performance of the stroke prediction models, standard evaluation metrics such as accuracy, precision, recall, F1-score, AUC-ROC, and the confusion matrix were utilized. These metrics provide a comprehensive view of how well the models distinguish between stroke and non-stroke cases. Given the inherent class imbalance in the dataset (few stroke cases compared to non-stroke cases), particular emphasis was placed on precision, recall, and F1-score over accuracy alone.

8.2 Results

Three models were evaluated in this study: a Decision Tree classifier, a Random Forest ensemble model, and an Artificial Neural Network (ANN). Each model was assessed on its ability to accurately classify stroke and non-stroke cases using various performance metrics.

The Decision Tree model offered a simple and interpretable structure, and while it achieved strong performance on non-stroke predictions, it was limited in its ability to detect true stroke cases. This shortfall is attributed to its tendency to overfit and its sensitivity to the class imbalance present in the dataset.

The Random Forest model, which combines multiple decision trees, demonstrated improved generalization and slightly better balance between sensitivity and specificity. It handled the class imbalance more effectively than the single decision tree, detecting a higher number of stroke cases while maintaining robustness against overfitting. Its ensemble nature allowed it to capture more complex patterns in the data, leading to a more reliable performance overall. The ANN model, despite its architectural complexity and high overall accuracy, was the weakest in identifying stroke cases. While it correctly classified non-stroke instances with high confidence, it failed to recognize most of the positive cases. This was likely due to the model's difficulty in learning minority class patterns without additional balancing techniques.

8.2.1 Random Forest Classifier

- **Accuracy Score:** 94.34%

- **F1 Score:** 94.21%

- **Precision Score:** 92.84%

- **Recall Score:** 95.61%

- **AUC Score:** 94.38%

8.2.2 Decision Tree Classifier

- **Accuracy Score:** 91.71%

- **F1 Score:** 91.40%

- **Precision Score:** 91.35%

- **Recall Score:** 91.45%

- **AUC Score:** 91.70%

8.2.3 Artificial Neural Network

- **Accuracy Score:** 95.77%

- **F1 Score:** 94.3%

- **Precision Score:** 94.1%

- **Recall Score:** 91.7 %

- **AUC Score:** 93.76%

8.3 Discussion

The results of the testing phase revealed that, although the models achieved high overall accuracy, their performance in predicting stroke cases was significantly limited by the heavily imbalanced dataset. The models exhibited a strong bias towards predicting the majority class (non-stroke), resulting in a high number of false negatives. This means that, while the models correctly identified many non-stroke cases, they failed to identify a considerable number of true stroke cases, which is a critical issue in the context of medical predictions. In healthcare applications, especially in stroke prediction, recall is an essential metric, as it measures the ability of the model to correctly identify all positive cases, which, in this case, would be patients who are at risk of a stroke. A low recall indicates that many

true stroke cases are being missed, which can have serious, life-threatening implications if these cases are not detected in time for timely intervention. Given the low recall values observed in this study, it is clear that the models are not yet reliable enough to be used in real-time medical decision-making. This highlights the need for further improvements in the model, particularly addressing the class imbalance issue, perhaps through techniques like oversampling the minority class, undersampling the majority class, or using advanced algorithms that are better suited for imbalanced datasets.

8.4 Performance Analysis

The performance metrics indicate that class imbalance has a considerable impact on the predictive quality of the models, particularly when it comes to accurately detecting stroke cases. Despite the increased complexity of the Artificial Neural Network (ANN), it did not outperform the simpler Decision Tree model in terms of identifying stroke cases, which is a critical factor in healthcare applications. This suggests that the ANN, although capable of capturing complex patterns in the data, struggled with the class imbalance present in the dataset, leading to a higher rate of false negatives. On the other hand, the Decision Tree, which is more interpretable and straightforward, was not immune to the issue but may have been less sensitive to the imbalance in some cases. The results highlight that both models require further optimization, especially in dealing with the class imbalance that skews their ability to predict stroke cases. To address this, techniques such as oversampling the minority class with methods like SMOTE (Synthetic Minority Over-sampling Technique), cost-sensitive learning that penalizes incorrect predictions of the minority class, or ensemble methods that combine multiple models for better performance could be explored. These strategies are aimed at ensuring that both models are more attuned to identifying the minority class—stroke cases—thereby improving their overall reliability and predictive performance in a real-world medical setting.

8.5 Snapshots

Fig 8.1 GUI Snapshot

This is a user interface from a Streamlit-based brain stroke prediction app. It allows users to input patient details such as gender, age, medical history, and lifestyle factors, which are then used by a machine learning model to assess stroke risk when the "Predict Stroke Risk" button is clicked.

Fig 8.2 Instance of low probability of Stroke

This screen displays the prediction result from the stroke risk assessment model. In this case, the model indicates a low risk of stroke, encouraging the user to continue maintaining a healthy lifestyle.
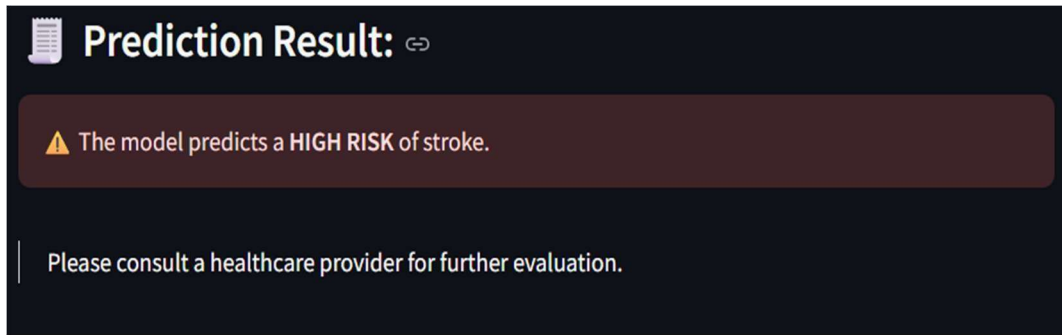
Fig 8.3 Instance of high probability of Stroke

This screen shows a high stroke risk prediction from the model. It advises the user to consult a healthcare provider for further medical evaluation and guidance.
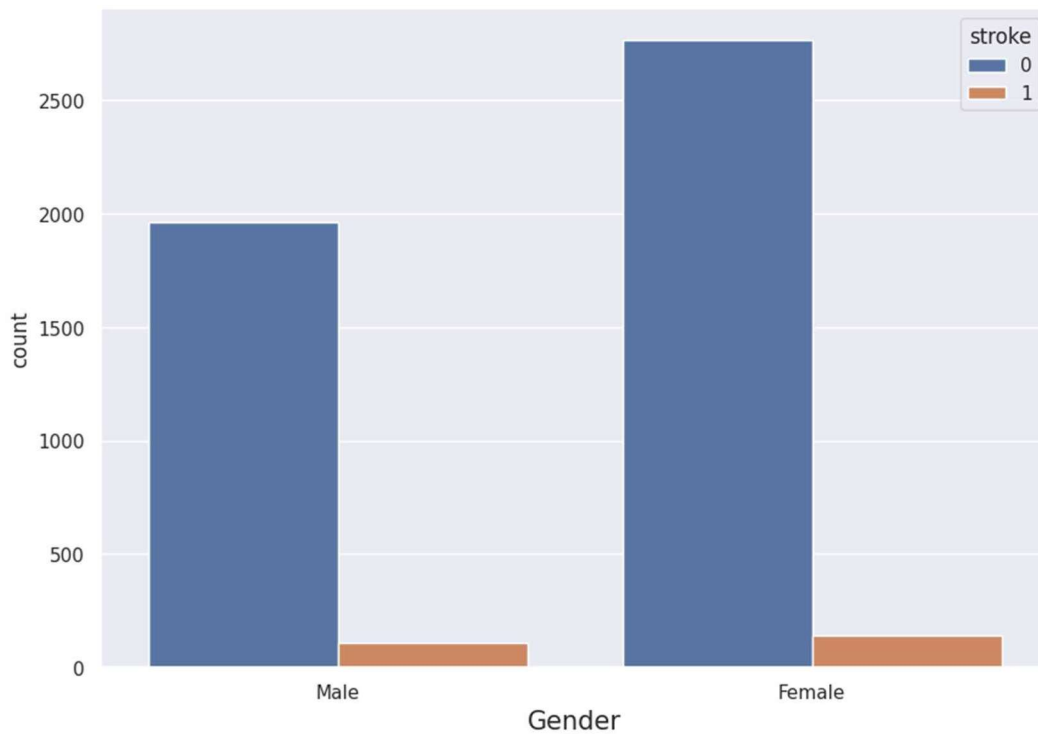
8.6 Visualizations

Fig 8.4 Stoke occurrence in Men vs Women

This bar chart shows the distribution of stroke cases by gender. Most individuals in the dataset did not have a stroke (blue bars), and among those who did (orange bars), females had a slightly higher number of stroke cases compared to males.
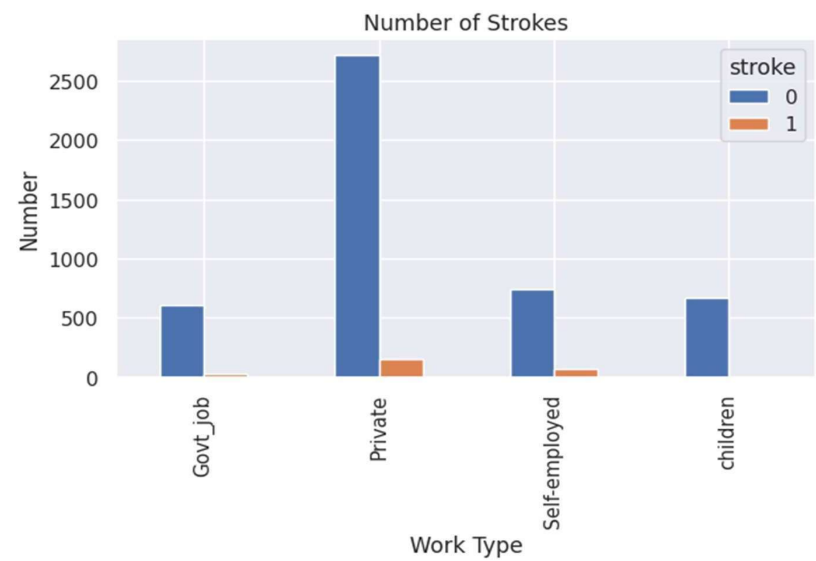
Fig 8.5 Stoke occurrence by Work Type

This bar chart illustrates the number of stroke cases across different work types. Most individuals work in the private sector, and this group also has the highest number of stroke cases. Stroke cases are lower among those with government jobs and the self-employed.
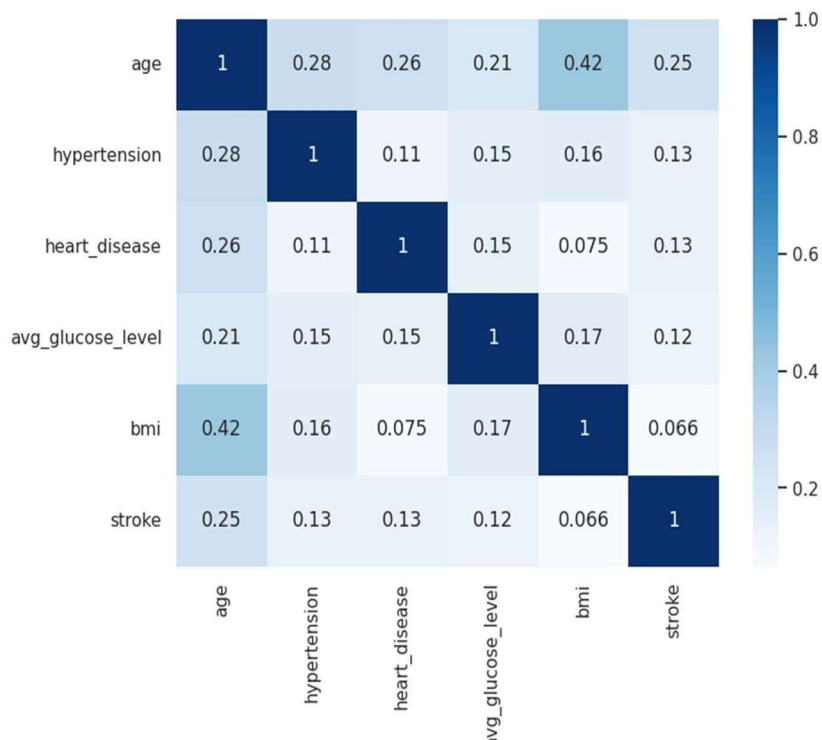

Fig 8.6 Correlation Analysis of Dataset Attributes

This heatmap shows the correlation between different features and stroke occurrence. Age has the strongest positive correlation with stroke (0.25), followed by hypertension and heart disease (both 0.13). BMI and average glucose level show weak correlations with stroke. The diagonal values are all 1, indicating perfect self-correlation. Overall, no feature shows a very strong correlation with stroke, suggesting the need for multi-factor analysis.

```
Epoch 1/10
123/123 [==============================] - 1s 1ms/step - loss: 0.9974 - accuracy: 0.8068
Epoch 2/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1950 - accuracy: 0.9593
Epoch 3/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1775 - accuracy: 0.9599
Epoch 4/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1777 - accuracy: 0.9565
Epoch 5/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1684 - accuracy: 0.9604
Epoch 6/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1729 - accuracy: 0.9561
Epoch 7/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1680 - accuracy: 0.9598
Epoch 8/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1736 - accuracy: 0.9552
Epoch 9/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1875 - accuracy: 0.9509
Epoch 10/10
123/123 [==============================] - 0s 1ms/step - loss: 0.1659 - accuracy: 0.9555
```

Fig 8.7 ANN Model training metrics

The training of the Artificial Neural Network (ANN) model spanned 10 epochs and showed a clear pattern of learning and stabilization. In the first epoch, the model began with a high loss of 0.9974 and an accuracy of 80.68%, reflecting its early learning stage. By the second epoch, it significantly improved, reducing the loss to 0.1950 and boosting accuracy to 95.93%, indicating effective learning. Over the next few epochs, the model continued refining, with the loss gradually decreasing and accuracy remaining consistently high—hovering around 95.5% to 96%. Though there were minor fluctuations in performance (such as a slight dip in accuracy during epochs 6 and 8), the overall trend showed stability. The final epoch concluded with a loss of 0.1659 and an accuracy of 95.55%, suggesting that the model had successfully learned the training patterns.

8.7 Applications

Healthcare Providers: Machine learning models like stroke prediction systems can assist healthcare providers in making more informed decisions by providing data-driven insights into patient risk factors, enabling early intervention and personalized treatment plans.

Personal Health Monitoring: Stroke prediction models can be integrated into personal health monitoring devices, allowing individuals to track their health indicators such as blood pressure, glucose levels, and lifestyle habits, helping them detect potential stroke risks early and take preventive actions.

Hospitals & Clinics: Hospitals and clinics can leverage stroke prediction models to enhance diagnostic accuracy and prioritize patients for further testing or treatment based on their likelihood of having a stroke, improving patient outcomes and streamlining healthcare services.

Public Health Campaigns: Machine learning models can be used in public health campaigns to identify high-risk populations and target specific groups with relevant educational materials and preventive measures, helping to reduce the overall incidence of strokes in the community

Health and Wellness Apps: Health and wellness apps can incorporate stroke prediction models to provide users with personalized recommendations, health tips, and risk assessments based on their medical data, encouraging healthier lifestyle choices and early detection of health issues.

CONCLUSION

The project collectively highlights the effectiveness of machine learning (ML) models in predicting stroke risk, demonstrating significant improvements over traditional statistical methods. Various ML techniques, including Decision Trees, Random Forest and Artificial Neural Networks (ANN), have been implemented. These models leverage datasets containing demographic, clinical, and lifestyle attributes to enhance predictive accuracy.

Across the studies, accuracy rates ranged from 82.7% to 98.28%, with ensemble and deep learning-based models generally outperforming individual classifiers. Feature selection played a crucial role in optimizing model performance.

Despite their promising results, several limitations exist. Most studies relied on preexisting datasets, limiting their generalizability to diverse populations. Many models lacked real-world validation, which is crucial for clinical adoption. Additionally, while decision trees and ensemble methods offer interpretability, deep learning models, despite their higher accuracy, are often considered "black boxes," making it challenging for healthcare professionals to trust their predictions.

Overall, machine learning presents a powerful tool for early stroke detection and prevention. However, future research should focus on integrating these models into clinical workflows, improving model transparency, addressing data imbalance issues, and validating results on real-world datasets. Combining ML with real-time monitoring systems and electronic health records could further enhance prediction accuracy and applicability in healthcare settings.

FUTURE WORK

There are several potential directions to enhance the performance and applicability of stroke prediction models. One important advancement involves enabling dynamic model training, where the model continuously learns and updates itself as new patient data becomes available. This approach would ensure the model adapts over time, improving accuracy and relevance in real- world clinical settings by reflecting recent trends, patient demographics, and evolving risk factors.

Another promising direction is the development of hybrid models, which combine the strengths of multiple machine learning algorithms or integrate traditional statistical methods with deep learning. Such models can improve predictive power and robustness by leveraging complementary decision-making mechanisms.

Additionally, data availability remains a critical challenge. High-quality, diverse, and comprehensive datasets are essential for building reliable stroke prediction systems. Future work should focus on collaborative data collection efforts, ensuring data privacy and security while enabling access to larger and more representative datasets across different populations. This will improve model generalizability and ensure fair and equitable predictions.

REFERENCES

[1] IEEE 2022: Centers for Disease Control and Prevention, "About stroke," 02-Nov-2022. https://www.cdc.gov/stroke/about.htm

[2] IEEE 2022: Mayo Clinic, "Stroke," 20-Jan-2022. https://www.mayoclinic.org/diseases-conditions/stroke/symptoms-causes/syc-20350113

[3] IEEE 2022: World Health Organization, "Stroke – Cerebrovascular Accident," Nov-2022. http://www.emro.who.int/health-topics/stroke-cerebrovascular-accident/index.html

[4] IEEE 2022: Johns Hopkins Medicine, "Risk factors for stroke," 13-Dec-2022. https://www.hopkinsmedicine.org/health/conditions-and-diseases/stroke/riskfactors-for-stroke

[5] IEEE 2022: Medline Plus, "Ischemic stroke," Nov-2022. https://www.medlineplus.gov/ischemicstroke.html

[6] IEEE 2020: World Health Organization, "The top 10 causes of death," 09-Dec-2020. https://www.who.int/news-room/factsheets/detail/the-top-10-causes-of-death

[7] IEEE 2022: American Stroke Association, "Hemorrhagic strokes (bleeds)," Nov-2022. https://www.stroke.org/en/about-stroke/types-of-stroke/hemorrhagic-strokesbleeds

[8] IEEE 2022: Y. Kumar, A. Koul, R. Singla, and M. F. Ijaz, "Artificial Intelligence in disease diagnosis: A systematic literature review, synthesizing framework and future research agenda," Journal of Ambient Intelligence and Humanized Computing, 2022.

[9] IEEE 2015: S. Das, A. Dey, A. Pal, and N. Roy, "Applications of artificial intelligence in machine learning: Review and Prospect," International Journal of Computer Applications, vol. 115, no. 9, pp. 31–41, Apr 2015.

[10] IEEE 2021: D. Petersson, "What is supervised learning?" Enterprise AI, 26-Mar-2021.

[11] IEEE 2017: R. Choudhary and H. K. Gianey, "Comprehensive Review on Supervised Machine Learning Algorithms," 2017 International Conference on Machine Learning and

Data Science (MLDS), pp. 37–43, Dec 2017.

[12] IEEE 2019: K. K. Verma, B. M. Singh, and A. Dixit, "A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in Intelligent surveillance system," International Journal of Information Technology, vol. 14, no. 1, pp. 397–410, 2019.

[13] IEEE 2020: V. Rajyaguru, C. Vithalani, and R. Thanki, "A literature review: Various learning techniques and its applications for eye disease identification using retinal images," International Journal of Information Technology, vol. 14, no. 2, pp. 713–724, 2020.

[14] IEEE 2019: K. K. Verma, B. M. Singh, and A. Dixit, "A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in Intelligent surveillance system," International Journal of Information Technology, vol. 14, no. 1, pp. 397–410, 2019.

[15] IEEE 2021: M. A. Ramessur and S. D. Nagowah, "A predictive model to estimate effort in a sprint using machine learning techniques," International Journal of Information Technology, vol. 13, no. 3, pp. 1101–1110, 2021.

[16] IEEE 2019: J. N. Heo, J. G. Yoon, H. Park, Y. D. Kim, H. S. Nam, and J. H. Heo, "Machine learning–based model for prediction of outcomes in acute stroke," Stroke, vol. 50, no. 5, pp. 1263–1265, May 2019.

[17] IEEE 2016: R. S. Jeena and S. Kumar, "Stroke prediction using SVM," 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 600–602, Dec. 2016.

[18] IEEE 2019: F. S. Alotaibi, "Implementation of machine learning model to predict heart failure disease," International Journal of Advanced Computer Science and Applications, vol. 10, no. 6, pp. 261–268, 2019.

[19] IEEE 2021: G. Sailasya and G. L. Kumari, "Analyzing the performance of stroke prediction using ML classification algorithms," International Journal of Advanced Computer Science and Applications, vol. 12, no. 6, pp. 539–545, 2021.

[20] IEEE 2022: C. Sharma, S. Sharma, M. Kumar, and A. Sodhi, "Early stroke prediction using machine learning," 2022 International Conference on Decision Aid Sciences and Applications (DASA), pp. 890–894, Mar. 2022.