

A MACHINE LEARNING MODEL TO PREDICT A DIAGNOSIS OF BRAIN STROKE PREDICTION

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering

By

SUBBAMMAL A P (Reg. No. 40130227)

THASLIMA BANU H (Reg. No. 40130235)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119**

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

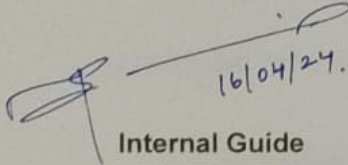
Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SUBBAMMAL A P** (40130227) and **THASLIMA BANU H** (40130235) who carried out the Project entitled "A MACHINE LEARNING MODEL TO PREDICT A DIAGNOSIS OF BRAIN STROKE PREDICTION" under our supervision from June 2023 to April 2024.


16/04/24.
Internal Guide

Dr. M.SUBRAMONIAM , M.E., Ph.D.


Head of the Department

Dr. T. RAVI, M.E., Ph.D.

Submitted for Viva voce Examination held on 18.04.2024

Internal Examiner

External Examiner

DECLARATION

We Subbammal A P (Reg. No. 40130227), Thaslima Banu H (Reg. No. 40130235) hereby declare that the Project Report entitled “**A MACHINE LEARNING MODEL TO PREDICT A DIAGNOSIS OF BRAIN STROKE PREDICTION**” done by us under the guidance of **Dr.M.SUBRAMONIAM, M.E., Ph.D.** Department of Electronics and Communication Engineering, **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE: 18.04.2024

PLACE: Chennai

SIGNATURE OF THE CANDIDATES

1. A.P.SUBBAMMAL

2. H.THASLIMA BANU

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph. D., Dean, School of Electrical and Electronics** and **Dr. T. RAVI, M.E., Ph. D., Head of the Department, Dept. of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr. M. SUBRAMONIAM, M.E., Ph.D., Department of Electronics and Communication Engineering** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In this study, we aimed to predict the occurrence of strokes using machine learning techniques. The dataset consists of various demographic and health-related features such as age, gender, hypertension, heart disease, and smoking status. After pre-processing the data, including handling missing values and encoding categorical variables, we trained several classification algorithms, including Random Forest Classifier, AdaBoost Classifier, and Gradient Boosting Classifier. We evaluated the models' performance using metrics such as accuracy, precision, recall, and F1-score. The Random Forest Classifier achieved the highest accuracy of 94.72% on the test set. Furthermore, we addressed the issue of class imbalance using techniques like SMOTE (Synthetic Minority Over-Sampling Technique) and Random Over Sampler, which helped improve the models' performance on predicting stroke occurrences. Overall, our findings suggest that machine learning models can effectively predict stroke occurrences based on demographic and health-related features, with the Random Forest Classifier showing promising results in terms of accuracy.

TABLE OF CONTENTS

CHAPTER .NO	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF TABLES	vi
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	vii
1.	INTRODUCTION	1
	1.1 BRAIN STROKE	1
	1.2 CEREBROVASCULAR ACCIDENT (CVA)	2
	1.3 OXYGEN AND NUTRIENTS	2
	1.4 ISCHEMIC STROKE	3
	1.5 OBJECTIVES	3
2	LITERATURE SURVEY	4 - 14
3	SYSTEM ANALYSIS	15
	3.1 EXISTING SYSTEM	15
	3.1.1 DRAWBACKS	15
	3.2 PROPOSED SYSTEM	16
	3.2.1 SYSTEM STANDARDS	16
	3.2.2 SYSTEM CONSTRINTS	17
	3.3.3 SYSTEM TRADE OFFS	17
	3.2.4 ADVANTAGES	17
	3.3 FEASIBILITY STUDY	18
	3.3.1 TECHNICAL FEASIBILITY	18
	3.3.2 OPERATIONAL FEASIBILITY	19
	3.3.3 ECONOMICAL FEASIBILITY	19
4	SYSTEM SPECIFICATION	20
	4.1 HARDWARE CONFIGURATION	20
	4.2 SOFTWARE SPECIFICATION	20
5	SOFTWARE DESCRIPTION	21

	5.1 FRONT END	21 - 27
6	PROJECT DESCRIPTION	28
	6.1 PROBLEM DEFINITION	28
	6.2 MODULE DESCRIPTION	28
	6.3 SYSTEM FLOW DIAGRAM	29
	6.4 INPUT DESIGN	30
	6.5 OUTPUT DESIGN	30
7	SYSTEM TESTING AND IMPLEMENTATION	31
	7.1 SYSTEM TESTING	31
	7.2 SYSTEM IMPLEMENTATION	31
8	SYSTEM MAINTENANCE	32
	8.1 CORRECTIVE MAINTENANCE	33
	8.2 ADAPTIVE MAINTENANCE	33
	8.3 PERFECTIVE MAINTENANCE	33
9	CONCLUSION AND FUTURE ENHANCEMENT	35
10	APPENDICES	36
	10.1 SOURCE CODE	37 – 71
	10.2 SCREEN SHOTS	72 - 78
11	REFERENCES	79

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NUMBER
1	SYSTEM FLOW DIAGRAM	28

LIST OF ABBREVIATIONS

ABBREVIATIONS	
SMOTE	Synthetic Minority Over-Sampling Technique
CVA	cerebrovascular accident
MRI	magnetic resonance imaging
CT	computed tomography
ML	Machine Learning
NSP	novel Stroke Prediction
SVM	Support Vector Machine
FNR	false-negative rate
DTC	Decision Tree Classifier

CHAPTER 1

1. INTRODUCTION

Brain stroke, also known as cerebrovascular accident (CVA), is a critical medical condition that occurs when there is a sudden disruption of blood supply to the brain, leading to a lack of oxygen and nutrients. This interruption can result from either a blockage in the blood vessels supplying the brain (ischemic stroke) or the rupture of a blood vessel within the brain (hemorrhagic stroke). The timely and accurate detection of a brain stroke is crucial for initiating prompt medical intervention, as the brain is highly sensitive to oxygen deprivation, and every minute counts in minimizing potential damage. Advances in medical technology, including imaging techniques such as computed tomography (CT) scans and magnetic resonance imaging (MRI), have significantly improved the ability to diagnose and assess the extent of brain strokes. Recognizing the signs and symptoms of a stroke is equally important for early intervention, as quick medical attention can make a substantial difference in the prognosis and recovery of individuals affected by this life-threatening condition.

1.1 BRAIN STROKE

A brain stroke, medically referred to as a cerebrovascular accident (CVA), is a critical and potentially life-threatening event characterized by a sudden disruption of blood supply to the brain. This interruption can occur through either the blockage of blood vessels supplying the brain, known as an ischemic stroke, or the rupture of a blood vessel within the brain, leading to a hemorrhagic stroke. The brain's functioning relies heavily on a constant supply of oxygen and nutrients, and when this supply is abruptly cut off, it can result in severe consequences. The timely detection and swift response to a brain stroke are paramount as the brain is highly sensitive to oxygen deprivation, and the duration of the interruption plays a crucial role in determining the extent of damage. Technological advancements in medical imaging, such as computed tomography (CT) scans and magnetic resonance imaging (MRI), have significantly enhanced our ability to diagnose and understand the intricacies of brain strokes.

1.2 CEREBROVASCULAR ACCIDENT (CVA)

A cerebrovascular accident (CVA), commonly known as a stroke, is a sudden and often devastating medical event that occurs when there is a disruption in the blood supply to the brain. This interruption can result from either the blockage of blood vessels supplying the brain (ischemic stroke) or the rupture of a blood vessel within the brain (hemorrhagic stroke). The consequences of a CVA can be severe, as the brain, being the control center for the body's functions, relies heavily on a constant flow of oxygen and nutrients to sustain its intricate processes. The occurrence of a CVA can lead to a range of impairments, including motor, sensory, and cognitive deficits, depending on the affected area of the brain. Timely recognition and intervention are crucial to minimize the potential damage and enhance the prospects for recovery. As a leading cause of long-term disability and mortality, understanding the nature of cerebrovascular accidents is paramount for both medical professionals and the general public, emphasizing the importance of preventative measures and proactive healthcare.

1.3 OXYGEN AND NUTRIENTS

Oxygen and nutrients are fundamental elements that sustain life, playing a crucial role in the intricate dance of physiological processes within the human body. Oxygen, a vital gas, serves as the catalyst for cellular respiration, providing the energy needed for the body's myriad functions. Simultaneously, nutrients encompass a diverse array of substances, including carbohydrates, proteins, fats, vitamins, and minerals, which collectively fuel the body's growth, maintenance, and repair. The symbiotic relationship between oxygen and nutrients is particularly evident in the brain, where these elements are essential for the organ's optimal functioning. The brain, accounting for a mere 2% of body weight, demands an astonishing 20% of the body's oxygen supply to support its energy-intensive activities. This intricate balance of oxygen and nutrients is integral to maintaining cellular health, and any disruption to this delicate equilibrium can have profound consequences on overall well-being, underscoring the significance of a lifestyle and healthcare practices that prioritize adequate oxygenation and nutritional support.

1.4 ISCHEMIC STROKE

An ischemic stroke stands as a critical manifestation of vascular compromise within the intricate landscape of the human brain. This medical event occurs when blood flow to a specific part of the brain is obstructed, often by a blood clot or plaque formation within the arteries supplying the organ. Deprived of its necessary oxygen and nutrients, the affected brain tissue begins to suffer damage, giving rise to potentially severe consequences. Ischemic strokes account for the majority of stroke cases, highlighting their significance as a leading cause of disability and mortality globally. Understanding the underlying factors that contribute to the development of ischemic strokes, such as hypertension, atherosclerosis, or cardiac conditions, is crucial for both preventative measures and targeted interventions. Advances in medical technologies, particularly imaging techniques like computed tomography (CT) scans and magnetic resonance imaging (MRI), have greatly enhanced our ability to diagnose and manage ischemic strokes promptly, emphasizing the critical importance of timely intervention to minimize long-term neurological impairments.

1.5 OBJECTIVES

1. Predict stroke occurrences using machine learning techniques.
2. Evaluate classification algorithms' performance on stroke prediction.
3. Address class imbalance in stroke prediction models.
4. Assess the effectiveness of demographic and health-related features in stroke prediction.

CHAPTER 2

2. LITERATURE REVIEW

2.1 PREDICTION OF BRAIN STROKE SEVERITY USING MACHINE LEARNING

Yong Yuet.al has proposed this paper in recent years, strokes have emerged as a prominent threat to global public health, primarily due to their profound impact on the central nervous system. Ischemic and hemorrhagic strokes, two major types of strokes, pose significant risks and challenges. The World Health Organization (WHO) reports that 87% of stroke cases are attributed to ischemic strokes, with subarachnoid hemorrhage affecting 3% of the population and intracerebral hemorrhage affecting 10%. The need for accurate identification, classification, and prediction of strokes has led to a growing interest in the application of Machine Learning (ML) techniques in the medical field. Despite the increasing use of ML in healthcare, existing research has limitations in predicting risk factors associated with different types of strokes comprehensively. To address this gap, this research introduces a novel Stroke Prediction (SPN) algorithm. The algorithm leverages an improved random forest approach to analyse and assess risk levels associated with strokes. The research specifically focuses on enhancing the prediction accuracy of the Stroke Predictor (SPR) model using advanced machine learning techniques. The proposed SPN algorithm exhibits significant advancements over existing models, achieving an impressive prediction accuracy of 96.97%. This substantial improvement underscores the potential of machine learning in revolutionizing stroke prediction, enabling more effective risk assessment and preventive measures. The research not only contributes to the field of medical informatics but also holds promise for enhancing patient outcomes and reducing the global burden of strokes through early and accurate prediction. As technology continues to evolve, the integration of machine learning in healthcare is likely to play a pivotal role in advancing our understanding of complex medical conditions, such as strokes, and improving the overall quality of patient care.

2.2 PERFORMANCE ANALYSIS OF MACHINE LEARNING APPROACHES IN STROKE PREDICTION

UsmanWazir et.al has proposed this paper The focus on early prediction and prevention of strokes is paramount in addressing the potential catastrophic consequences of these events. Recognizing the urgency of this matter, the current research proposes a comprehensive approach to stroke prediction by integrating various machine learning techniques. The study considers crucial factors such as hypertension, body mass index, heart disease, average glucose levels, smoking status, previous stroke history, and age, all of which contribute significantly to the occurrence of strokes. Ten different machine learning classifiers, including Logistics Regression, Stochastic Gradient

Jian Lou et.al has proposed this paper Stroke, a medical disorder characterized by the rupture of blood arteries in the brain, poses a serious threat to health as it can lead to significant brain damage. Interruption in the supply of blood and essential nutrients to the brain can result in various symptoms. The World Health Organization (WHO) identifies stroke as the leading cause of death and disability on a global scale. Recognizing the warning signs of a stroke early on is crucial for minimizing its severity and improving outcomes. In response to this challenge, different machine Descent, Decision Tree Classifier, AdaBoost Classifier, Gaussian Classifier, Quadratic Discriminant Analysis, Multi-layer Perceptron Classifier, KNeighbors Classifier, Gradient Boosting Classifier, and XGBoost Classifier, were employed to harness the predictive power of the selected features. The utilization of a diverse set of classifiers enables a more robust analysis of the data, capturing different patterns and relationships that contribute to stroke prediction. The results of the individual classifiers were then aggregated using a weighted voting approach, demonstrating the efficacy of this ensemble method in enhancing prediction accuracy. The study achieved an impressive overall accuracy of 97%, with the weighted voting classifier outperforming the individual base classifiers. The high accuracy and robust performance of the weighted voting classifier make it a promising tool for predicting strokes, offering a reliable means for physicians and patients to detect and address potential stroke risks. Notably, the study evaluated the performance of the model using additional metrics such as the area under the curve (AUC) value. The high AUC value further validates the model's effectiveness in

distinguishing between positive and negative instances. Moreover, the weighted voting classifier demonstrated lower false positive and false negative rates compared to individual classifiers, underscoring its reliability in clinical applications. In conclusion, the proposed machine learning model, particularly the weighted voting classifier, emerges as a powerful tool for predicting strokes with a high degree of accuracy. The potential clinical implications of this model are significant, providing healthcare professionals and patients alike with a valuable tool for early detection and intervention, ultimately contributing to the prevention and mitigation of the impact of strokes on individuals' health.

2.3 A MACHINE LEARNING APPROACH TO DETECT THE BRAIN STROKE DISEASE

Priti Narwal et al. has proposed this paper. The intricate nature of the human brain, comprising the cerebrum, cerebellum, and brainstem and enclosed within the protective skull, underscores its significance in human physiology. Given the brain's complexity, strokes pose a critical threat, ranking as the second-leading cause of mortality worldwide. Swift and effective treatment is essential to mitigate brain damage, making early detection a pivotal factor in preventing or minimizing the severity of strokes and consequently reducing mortality rates. In this context, leveraging machine learning algorithms for identifying risk variables associated with strokes emerges as a promising avenue. The current research introduces a model designed to forecast brain strokes accurately. The success of this model relies on a meticulous approach to data handling, including efficient data collection, pre-processing, and transformation techniques. The use of a dedicated "brain stroke dataset" serves as the foundation for constructing and training the model. To ensure the reliability of the information provided by the model, standardization techniques are applied to normalize the data. The training and testing phases employ three distinct classifiers—Random Forest (RF), Support Vector Machine (SVM), and Decision Tree (DT). The model's performance is rigorously evaluated using key metrics such as accuracy, sensitivity (SEN), error rate, false-positive rate (FPR), false-negative rate (FNR), root mean square error, and log loss. Notably, the Random Forest (RF) classifier emerges as a standout performer in the evaluation. The results indicate that the proposed model achieves a remarkable accuracy of 95.30% when utilizing the RF classifier. This high level of accuracy attests to the efficacy of the model in predicting brain

strokes. The findings suggest that the integration of machine learning algorithms, particularly when coupled with meticulous data handling procedures, holds considerable promise for enhancing early stroke detection and improving patient outcomes. In conclusion, the proposed model not only demonstrates the potential of machine learning in forecasting brain strokes but also underscores the critical role of accurate data handling in the success of such predictive models. As technological advancements continue, the integration of machine learning approaches in healthcare holds the potential to revolutionize the field of neurological disorders, ultimately contributing to more effective stroke prevention and treatment strategies.

2.4 STROKE PREDICTION ANALYSIS USING MACHINE LEARNING CLASSIFIERS AND FEATURE TECHNIQUE

Nitin Kumar Sharma et al. has proposed this paper Stroke, characterized as a fatal brain disease with a rapid onset that can lead to death within 3 to 10 hours, emphasizes the critical need for timely intervention to prevent mortality. Smart health systems present a promising avenue for addressing this urgency by enabling prompt identification and reaction to the nature of a stroke. In this paper, the focus is on developing a machine learning model for predicting the existence of a stroke in a patient, with a specific emphasis on the superior performance of the Random Forest classifier compared to state-of-the-art models such as Logistic Regression, Decision Tree Classifier (DTC), and K-NN. The experimental framework involved the use of datasets comprising 5110 observations and 12 attributes. To enhance the robustness of the model, exploratory data analysis (EDA) was employed for pre-processing, and feature engineering techniques were applied to balance the datasets. The emphasis on dataset balancing is crucial for avoiding biases and ensuring the model's generalizability across diverse instances. In a forward-thinking approach, the research integrates a cloud-based mobile application into the system. This mobile app serves as a data collection tool, allowing users to input relevant information for analysis. The model's predictions, based on user-provided data, offer the possibility of stroke detection. The system boasts an impressive accuracy with precision, recall, and F1-score all standing at 96%. This high level of accuracy is crucial in ensuring reliable predictions, minimizing false alarms, and optimizing the system's

utility in real-world scenarios. The user-friendly nature of the proposed system is highlighted as a key advantage. Users can receive essential warnings easily by providing minimal information from anywhere, leveraging the convenience of a mobile device. This accessibility makes the system a potential lifesaver, as individuals can receive timely alerts, facilitating prompt action and potentially preventing severe consequences associated with strokes. In conclusion, the integration of a machine learning model, particularly the Random Forest classifier, within a smart health system offers a promising solution for predicting strokes. The combination of advanced algorithms, thorough data pre-processing, and a user-friendly mobile application enhances the overall effectiveness of the system in providing timely warnings. This innovative approach represents a significant step towards leveraging technology for stroke prevention and underscores the potential of such systems in saving lives.

2.5 PREDICTING STROKE RISK WITH AN INTERPRETABLE CLASSIFIER

Ziad Ismailel.al has proposed this paper Predicting the risk of stroke in individuals has garnered considerable attention globally, with researchers and governments recognizing the potential benefits of early awareness for prevention and treatment. Governments, in particular, have been actively collecting medical data to harness artificial intelligence methods for making accurate predictions. While black-box methods have shown high accuracy in predictions, they often lack transparency, providing limited insights into why a specific prediction is made. In the medical field, explanations are deemed crucial as they allow specialists to understand the influencing factors behind the assessed risk level. This research addresses the need for both accuracy and interpretability in stroke risk prediction. The proposed method, based on the Dempster-Shafer theory of plausibility, not only outperforms existing methods but also offers information about the most probable causes of a high stroke risk. Furthermore, the method demonstrates its robustness in dealing with incomplete medical records, a common challenge in real-world healthcare datasets. The dataset used for testing originates from the regional hospital in Okayama, Japan, a country where annual health check-ups are mandatory by law. The utilization of real-world data enhances the applicability of the findings to practical healthcare settings. The experiments conducted in this research compare the results of the Dempster-Shafer

method with other well-known machine learning techniques, including Multilayer Perceptron, Support Vector Machines, and Naive Bayes. Remarkably, the Dempster-Shafer method outperformed the other methods, showcasing its effectiveness in handling missing data. This is a significant advantage in healthcare settings where incomplete records are common. The article also delves into the interpretation of rules generated by the method for classification. The rules were validated through alignment with medical literature and consultation with human specialists, establishing the method's reliability and relevance in a clinical context. In conclusion, this research not only contributes to the improvement of stroke risk prediction but also underscores the importance of interpretability, especially in medical applications. The Dempster-Shafer method emerges as a powerful tool, offering both accuracy and transparency, thus presenting a promising avenue for enhancing stroke prediction models in healthcare.

2.6 PREDICTION OF HEMORRHAGIC TRANSFORMATION SEVERITY IN ACUTE STROKE FROM SOURCE PERFUSION MRI

Evrin Furuncu et al. has proposed this paper Hemorrhagic transformation (HT) poses a significant challenge as the most severe complication of reperfusion therapy in acute ischemic stroke (AIS) patients. The effective management of AIS patients would greatly benefit from accurate predictions not only of HT occurrence but also of its severity and the affected brain territory. This study addresses these critical issues and seeks to advance the prediction of spatial HT occurrence in AIS by utilizing perfusion-weighted magnetic resonance imaging (PWI) in conjunction with diffusion-weighted imaging (DWI). The retrospective analysis included 165 AIS patients treated with reperfusion therapy at a single stroke center. The integration of machine learning models within the framework involved the application of various algorithms, including Support Vector Machines (SVM), linear regression, decision trees, neural networks, and kernel spectral regression. Notably, kernel spectral regression emerged as the top-performing model, achieving an accuracy of $83.7 \pm 2.6\%$. The significance of this framework lies in its formalization of the HT prediction problem as a machine learning task. The model is designed to directly extract imaging markers of HT from source PWI images, avoiding reliance on pre-established metrics. This approach enhances the model's adaptability and potential for

capturing nuanced patterns in the imaging data that might be crucial for accurate predictions. The outcomes of the study offer valuable insights for neuro-interventionists by providing spatial likelihood predictions of HT in various brain territories. These predictions were evaluated against follow-up gradient recalled echo (GRE), shedding light on the potential utility of the model in guiding decisions prior to endovascular therapy. The ability to predict not only the occurrence but also the spatial distribution and severity of HT represents a significant advancement in the field, offering a more comprehensive understanding of the potential complications associated with reperfusion therapy in AIS patients. In conclusion, the presented framework leverages machine learning techniques to enhance the prediction of hemorrhagic transformation in acute ischemic stroke patients. The success of the model, particularly the kernel spectral regression approach, showcases its potential clinical utility in guiding treatment decisions and improving patient outcomes in the challenging context of reperfusion therapy.

2.7 EARLY STROKE PREDICTION METHODS FOR PREVENTION OF STROKES

Jin Liet.al has proposed this paper The rapid advancement of technology has paved the way for non-invasive techniques in the realm of healthcare, offering promising avenues for early detection and diagnosis. Among the various cardiovascular diseases, stroke stands out as a particularly dangerous and life-threatening condition. Detecting strokes at an early stage is crucial for saving lives, and research suggests that patients often experience ministrokes, known as transient ischemic attacks (TIA), before a full-blown stroke occurs. However, much of the existing literature relies on expensive diagnostic methods such as MRI and CT scans for classifying cardiovascular diseases, including strokes. In the context of rising stroke cases in India and the need for cost-effective diagnostic methods, this study aims to explore non-invasive approaches for the early diagnosis of strokes. The paper proposes a novel method that utilizes processed EEG data, avoiding the time-consuming nature of cascaded prediction algorithms and leveraging the inherent properties of EEG signals. The primary objective is to develop mechanisms for forecasting strokes based on time series analysis, employing advanced techniques such as Long Short-Term Memory (LSTM), bidirectional LSTM (biLSTM), Gated Recurrent Unit (GRU), and Feedforward Neural Network (FFNN). The

experimental research outcomes demonstrate the efficacy of these time series-based approaches in predicting early strokes. Notably, GRU emerges as the top-performing algorithm, achieving an impressive accuracy of 95.6%. The results also highlight the strong performance of biLSTM (91% accuracy), LSTM (87% accuracy), and FFNN (83% accuracy). These findings indicate that the proposed methods can effectively analyse brain wave patterns to predict the signs of strokes. The potential clinical impact of this research is significant, as the accurate and early detection of strokes can empower physicians to take timely interventions and save lives. The non-invasive nature of the proposed approach makes it particularly appealing for widespread application, especially in regions like India where cost-effective healthcare solutions are critical. In conclusion, this paper contributes to the field of stroke detection by introducing a non-invasive approach based on processed EEG data and time series-based prediction algorithms. The impressive accuracy rates achieved by these algorithms, especially GRU, underscore the potential of the proposed methods in aiding physicians in the early detection of strokes, thereby improving patient outcomes.

2.8 PREDICTIVE ANALYSIS FOR RISK OF STROKE USING MACHINE LEARNING TECHNIQUES

Xun Yiet.al has proposed this paper Recently, Stroke, characterized by the interruption or reduction of blood flow to the brain, depriving brain tissue of essential oxygen and nutrients, underscores the critical importance of early intervention and treatment. Timely prediction of stroke risk is pivotal in facilitating proactive measures for prevention and intervention. Recognizing the significance of predicting strokes, this experiment focuses on implementing various machine learning algorithms to predict stroke risk using a comprehensive dataset. The initial steps of data imputation, feature selection, and data pre-processing lay the groundwork for this study. Key physical parameters such as heart disease, age, BMI, gender, and hypertension are identified as features for training and testing the predictive models. Leveraging AdaBoost, artificial neural network, decision tree classifier, k-nearest neighbor (KNN) classifier, random forest, stochastic gradient descent (SDG), support vector machine (SVM), and XGBoost classifier, the study explores a diverse range of machine learning algorithms to predict the risk of stroke. In

the experimentation phase, a voting classifier is introduced, combining the predictions from the eight traditional classifiers. This ensemble approach aims to enhance the overall predictive performance. The results reveal that after analysing different machine learning algorithms with voting classifiers, the model achieves an impressive accuracy of 98% in predicting the risk factor of stroke. This high level of accuracy positions the proposed model as superior to other existing models. The significance of this research lies in its ability to contribute to the field of stroke prediction by evaluating and comparing multiple machine learning algorithms. The achieved accuracy of 98% indicates the robustness and effectiveness of the proposed approach in identifying individuals at risk of stroke. This could have profound implications for early intervention strategies, leading to improved patient outcomes. In conclusion, the study not only underscores the critical nature of stroke prediction for early intervention but also demonstrates the superiority of the proposed model over other existing models. The utilization of various machine learning algorithms and the integration of a voting classifier showcase the potential for accurate and reliable stroke risk prediction, marking a significant advancement in the field of healthcare analytics.

2.9 STROKE DISEASE DETECTION AND PREDICTION USING ROBUST LEARNING APPROACHES

learning (ML) models have been developed to predict the likelihood of a stroke occurrence in the brain. This research specifically employs a variety of physiological parameters and machine learning algorithms, including Logistic Regression (LR), Decision Tree (DT) Classification, Random Forest (RF) Classification, and a Voting Classifier. These algorithms are used to train four distinct models with the aim of achieving reliable stroke prediction. Notably, the Random Forest algorithm emerges as the top performer in this task, demonstrating an accuracy of approximately 96 percent. The dataset utilized for developing and testing these models is the open-access Stroke Prediction dataset. The high accuracy percentage achieved in this investigation surpasses that of previous studies, indicating the enhanced reliability of the models employed. The use of machine learning algorithms in predicting strokes showcases the potential of advanced analytical techniques in healthcare. The study not only presents

successful model comparisons that establish their robustness but also emphasizes the reliability and effectiveness of the proposed scheme. In conclusion, this research contributes to the growing body of work focused on stroke prediction using machine learning. The utilization of physiological parameters and advanced algorithms demonstrates the potential for accurate and reliable predictions. The high accuracy rates achieved by the models, particularly with Random Forest, underscore the effectiveness of this approach in enhancing stroke prediction. The study's findings hold promise for improving early detection and intervention strategies, ultimately contributing to better stroke outcomes.

2.10 PREDICTION OF BRAIN STROKE USING MACHINE LEARNING ALGORITHMS AND DEEP NEURAL NETWORK TECHNIQUES

Mohamed Nabee et al. has proposed this paper. The brain, as the primary organ of the human body, plays a crucial role in various physiological functions. Stroke, a medical disorder characterized by the rupture of blood arteries in the brain, has serious implications as it can lead to damage and interruption of blood supply, causing symptoms and potential long-term complications. Recognized as a medical emergency, strokes can result in neurological damage, complications, and, in severe cases, death. The World Health Organization (WHO) highlights stroke as the leading cause of death and disability globally, emphasizing the urgent need for effective detection and intervention strategies. In this study, the primary objective is to forecast the possibility of a brain stroke occurring at an early stage, leveraging both deep learning and machine learning techniques. The research utilizes a reliable dataset for stroke prediction obtained from the Kaggle website. A diverse set of classification models, including Extreme Gradient Boosting (XGBoost), Ada Boost, Light Gradient Boosting Machine, Random Forest, Decision Tree, Logistic Regression, K Neighbors, SVM - Linear Kernel, Naive Bayes, and deep neural networks (3-layer and 4-layer ANN), are employed for classification tasks. The Random Forest classifier emerges as the standout performer, achieving an impressive 99% classification accuracy, which is the highest among the machine learning classifiers employed in the study. Additionally, the three-layer deep neural network (4-Layer ANN) demonstrates a higher accuracy of 92.39% compared to the three-layer ANN method utilizing selected

features as input. The research findings highlight that machine learning techniques outperformed deep neural networks in this particular context. The significance of this study lies in its exploration of diverse techniques for early stroke prediction, shedding light on the strengths and effectiveness of different algorithms. The high accuracy achieved by the Random Forest classifier suggests its potential as a robust tool for stroke prediction. Additionally, the nuanced comparison between machine learning and deep neural networks offers valuable insights into the most effective methodologies for this specific predictive task. In conclusion, this research underscores the importance of early stroke prediction and compares the efficacy of various machine learning and deep learning techniques. The findings contribute to the growing body of knowledge in healthcare analytics, emphasizing the potential of machine learning algorithms, particularly the Random Forest classifier, in enhancing stroke prediction accuracy.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

A stroke is a medical condition in which poor blood flow to the brain results in cell death. It is now a day a leading cause of death all over the world. Several risk factors believe to be related to the cause of stroke has been found by inspecting the affected individuals. Using these risk factors, a number of works have been carried out for predicting the stroke diseases. Most of the models are based on data mining and machine learning algorithms. In this work, we have used five machine learning algorithms to detect the stroke that can possibly occur or occurred form a person's physical state and medical report data. We have collected a good number of entries from the hospitals and use them to solve our problem. The classification result shows that the result is satisfactory and can be used in real time medical report. We believe that machine learning algorithms can help better understanding of diseases and can be a good healthcare companion

3.1.1 DRAWBACKS

1. Machine learning algorithms may struggle to generalize well to diverse populations, potentially leading to biases and reduced effectiveness across different demographic groups.
2. The performance of machine learning models heavily relies on the quality and representativeness of the input data, making them susceptible to inaccuracies or biases inherent in the training data.
3. Many machine learning algorithms, especially complex ones like deep neural networks, often lack transparency, making it challenging for healthcare professionals to interpret and trust the decision-making process. privacy, data

security, and the responsible handling of sensitive medical information, necessitating careful consideration and robust safeguards.

3.2 PROPOSED SYSTEM

The proposed system encompasses a comprehensive approach to predicting strokes using machine learning techniques. Initially, the dataset undergoes thorough exploration and preprocessing to handle missing values, drop irrelevant features, and encode categorical variables for compatibility with machine learning algorithms. Feature selection techniques are then applied to transform raw data into meaningful features, potentially including feature scaling and outlier handling to further refine the dataset. Subsequently, multiple classification algorithms such as Random Forest Classifier, AdaBoost Classifier, and Gradient Boosting Classifier are trained and evaluated for their efficacy in stroke prediction. Hyper parameter tuning using techniques like Randomized SearchCV optimizes model performance, with evaluation metrics such as accuracy, precision, recall, and F1-score employed to assess effectiveness. Given the imbalanced nature of the dataset, oversampling techniques like SMOTE and Random Over Sampler are explored to address class imbalance and enhance model accuracy.

3.2.1 SYSTEM STANDARDS

Accuracy & Metrics: The model should be demonstrably accurate, with high sensitivity (catching true strokes) and specificity (avoiding false positives). Metrics like AUC provide a more comprehensive picture.

Data Quality & Generalizability: Training data should be high-quality, diverse, and reflect real-world scenarios. The model's ability to perform well on unseen data (generalizability) is crucial.

Interpretability & Explainability: Understanding how the model arrives at a conclusion is essential for doctors to trust its recommendations. Transparency allows informed decision-making alongside the model.

Regulatory Approval: Regulatory bodies have guidelines for ensuring safety and efficacy. ML models for medical diagnosis need to go through these approval processes.

3.2.2 SYSTEM CONSTRAINTS

Data Limitations:

Quality & Quantity: Acquiring massive amounts of high-quality labeled medical data is expensive and time-consuming. Biases in the data can lead to biased models.

Data Diversity: Stroke presentations can vary. If the training data lacks diversity, the model might struggle with identifying atypical strokes.

Model Performance:

Accuracy vs. Explainability: Highly accurate models, especially deep learning, can be difficult to interpret ("black boxes"). This lack of transparency can be problematic in medicine.

False Positives & Negatives: Even with high accuracy, mistakes can happen. False positives lead to unnecessary procedures and anxiety. False negatives can have devastating consequences.

3.2.3 SYSTEM TRADE OFFS

Accuracy vs. Explainability: As mentioned earlier, there's often a trade-off between achieving high accuracy and understanding the model's reasoning.

Sensitivity vs. Specificity: Focusing heavily on catching all strokes (high sensitivity) might increase false positives. Conversely, prioritizing avoiding false positives (high specificity) might lead to missed strokes.

3.2.4 ADVANTAGES

1. Enhances early detection of stroke risk factors. Enables personalized preventive interventions for at-risk individuals.
2. Facilitates more accurate and timely medical decision-making.
3. Contributes to improved public health outcomes through targeted interventions.

3.3 FEASIBILITY STUDY

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility

3.3.1 TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at DB2 Database. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified.

Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

3.3.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

3.3.3 ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS

▮ Processor Type	: AMD RYZEN 7
▮ Speed	: 4.40GHZ
▮ RAM	:16 GB RAM
▮ Hard disk	: 1 TB
▮ Keyboard	: 101/102 Standard Keys
▮ Mouse	: Optical Mouse

4.2 SOFTWARE REQUIREMENTS

▮ Operating System	: Windows 10
▮ Front End	: Jupyter Notebook/ Anaconda tool
▮ Coding Language	: Python

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 FRONT END: JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. The software requirement specification is created at the end of the analysis task. The function and performance allocated to software as part of system engineering are developed by establishing a complete information report as functional representation, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

FEATURES OF JUPYTER NOTEBOOK

- In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.
- The ability to execute code from the browser, with the results of computations attached to the code which generated them.
- Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc.
- For example, publication-quality figures rendered by the matplotlib library, can be included inline.
- In-browser editing for rich text using the Markdown markup language, which can provide commentary for the code, is not limited to plain text.
- The ability to easily include mathematical notation within markdown cells using LaTeX, and rendered natively by MathJax.

NOTEBOOK DOCUMENTS

Notebook documents contains the inputs and outputs of a interactive session as well as additional text that accompanies the code but is not meant for execution.

In this way, notebook files can serve as a complete computational record of a session, interleaving executable code with explanatory text, mathematics, and rich representations of resulting objects. These documents are internally JSON files and are saved with the .ipynb extension. Since JSON is a plain text format, they can be version-controlled and shared with colleagues. Notebooks may be exported to a range of static formats, including HTML (for example, for blog posts), reStructuredText, LaTeX, PDF, and slide shows, via the nbconvert command.

Furthermore, any .ipynb notebook document available from a public URL can be shared via the Jupyter Notebook Viewer (nbviewer). This service loads the notebook document from the URL and renders it as a static web page. The results may thus be shared with a colleague, or as a public blog post, without other users needing to install the Jupyter notebook themselves. In effect, nbviewer is simply nbconvert as a web service, so you can do your own static conversions with nbconvert, without relying on nbviewer.

PYTHON DEFINITION

Python is a high-level programming language designed to be easy to read and simple to implement. It is open source, which means it is free to use, even for commercial applications. Python can run on Mac, Windows, and Unix systems and has also been ported to Java and .NET virtual machines.

Python is considered a scripting language, like Ruby or Perl and is often used for creating Web applications and dynamic Web content. It is also supported by a number of 2D and 3D imaging programs, enabling users to create custom plug-ins and extensions with Python. Examples of applications that support a Python API include GIMP, Inkscape, Blender, and Autodesk Maya.

Scripts written in Python (.PY files) can be parsed and run immediately. They can also be saved as a compiled programs (.PYC files), which are often used as programming modules that can be referenced by other Python programs.

PYTHON FEATURES

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.

1) Easy to Learn and Use

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.

2) Expressive Language

Python can perform complex tasks using a few lines of code. A simple example, the hello world program you simply type `print("Hello World")`. It will take only one line to execute, while Java or C takes multiple lines.

3) Interpreted Language

Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language. It enables programmers to develop the software for several competing platforms by writing a program only once.

5) Free and Open Source

Python is freely available for everyone. It is freely available on its official website www.python.org. It has a large community across the world that is dedicatedly working towards make new python modules and functions. Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

6) Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence. It supports inheritance, polymorphism, and encapsulation, etc. The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code. It converts the program into byte code, and any platform can use that byte code.

8) Large Standard Library

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting. There are various machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

9) GUI Programming Support

Graphical User Interface is used for the developing Desktop application. PyQt5, Tkinter, Kivy are the libraries which are used for developing the web application.

10) Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C, C++ Java. It makes easy to debug the code.

11. Embeddable

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

12. Dynamic Memory Allocation

In Python, we don't need to specify the data-type of the variable. When we assign some value to the variable, it automatically allocates the memory to the variable at run time. Suppose we are assigned integer value 15 to x, then we don't need to write `int x = 15`. Just write `x = 15`.

ANACONDA

Anaconda Cloud is a package management service by Anaconda. Cloud makes it easy to find, access, store and share public notebooks, environments, and conda and PyPI packages. Cloud also makes it easy to stay current with updates made to the packages and environments you are using. Cloud hosts hundreds of useful Python packages, notebooks, projects and environments for a wide variety of applications. You do not need to log in, or even to have a Cloud account, to search for public packages, download and install them.

You can build new conda packages using `conda-build`, then upload the packages to Cloud to quickly share with others or access yourself from anywhere. The Anaconda Cloud command line interface (CLI), `anaconda-client`, allows you to manage your account - including authentication, tokens, upload, download, remove and search. Connect to and manage your Anaconda Cloud account. Upload packages you have created. Generate access tokens to allow access to private packages.

For developers, Cloud is designed to make software development, release and maintenance easy by providing broad package management support. Cloud allows for free public package hosting, as well as package channels, providing a flexible and scalable service for groups and organizations of all sizes.

APPLICATIONS PROVIDED IN ANACONDA DISTRIBUTION

The Anaconda distribution comes with the following applications along with Anaconda Navigator.

- JupyterLab
- Jupyter Notebook
- Qt Console
- Spyder
- Glueviz
- Orange3
- RStudio
- Visual Studio Code

JupyterLab: This is an extensible working environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Jupyter Notebook: This is a web-based, interactive computing notebook environment. We can edit and run human-readable docs while describing the data analysis.

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

Spyder: Spyder is a scientific Python Development Environment. It is a powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.

VS Code: It is a streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz: This is used for multidimensional data visualization across files. It explores relationships within and among related datasets.

Orange 3: It is a component-based data mining framework. This can be used for data visualization and data analysis. The workflows in Orange 3 are very interactive and provide a large toolbox.

Rstudio: It is a set of integrated tools designed to help you be more productive with R. It includes R essentials and notebooks

CHAPTER 6

PROJECT DESCRIPTION

6.1 PROBLEM DEFINITION

The escalating global burden of strokes poses a significant challenge to public health, with a pressing need for effective predictive tools to identify individuals at risk. While numerous risk factors associated with stroke have been identified, the development of accurate prediction models remains a complex task. Existing efforts predominantly rely on data mining and machine learning algorithms, yet the potential limitations and biases inherent in these models, such as challenges in generalizability and interpretability, raise concerns about their real-world applicability. Additionally, the ethical implications surrounding patient data privacy and security demand careful consideration.

6.2 MODULE DESCRIPTION

6.2.1 DATA EXPLORATION AND PREPROCESSING:

The initial phase involves exploring the dataset to understand its structure and characteristics. Missing values are handled appropriately, and irrelevant features are dropped to streamline the dataset for further analysis. Categorical variables are encoded, ensuring compatibility with machine learning algorithms.

6.2.2 FEATURE SELECTION:

This module focuses on transforming raw data into meaningful features that can enhance the predictive power of the model. Categorical variables are encoded, and certain categories are renamed for simplification. Additionally, feature scaling and outlier handling techniques may be explored to further improve model performance.

6.2.3 MODEL BUILDING AND EVALUATION:

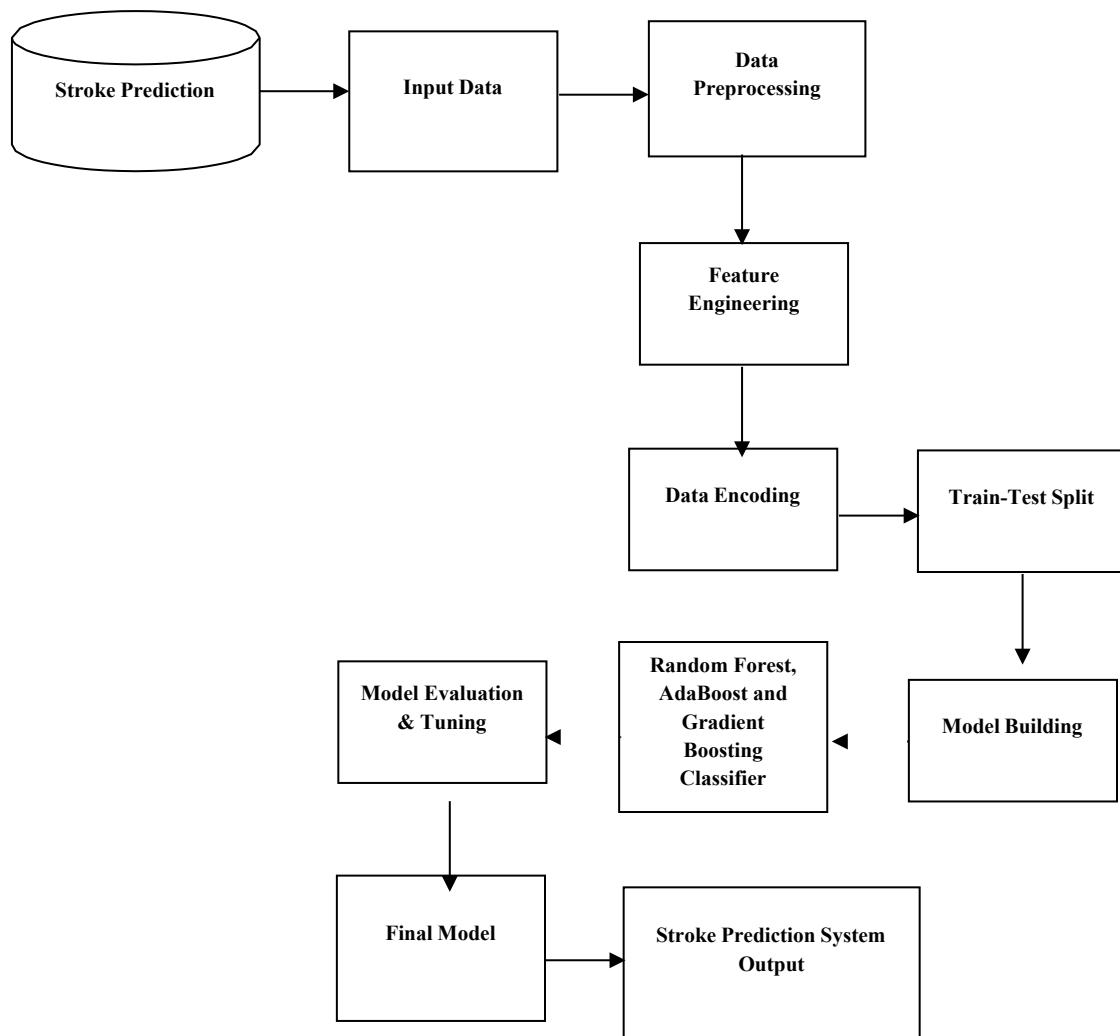
Multiple classification algorithms, including Random Forest Classifier, AdaBoost Classifier, and Gradient Boosting Classifier, are trained and evaluated for their ability to predict strokes. Techniques like Randomized SearchCV are employed for hyper

parameter tuning, optimizing model performance. Model evaluation metrics such as accuracy, precision, recall, and F1-score are utilized to assess model effectiveness.

6.2.4 OVERSAMPLING TECHNIQUES:

Given the imbalanced nature of the dataset, oversampling techniques like SMOTE and Random Over Sampler are explored to address class imbalance and enhance model performance. These techniques generate synthetic samples of the minority class, thereby improving the model's ability to predict strokes accurately.

6.3 SYSTEM FLOW DIAGRAM



6.4 INPUT DESIGN

The input design for predicting stroke occurrences using machine learning involves collecting various demographic and health-related data points from individuals, including age, gender, hypertension status, heart disease history, and smoking habits. These data are typically gathered through surveys, medical records, or electronic health records systems. Additionally, preprocessing steps such as handling missing values and encoding categorical variables are applied to ensure data quality and compatibility with the chosen machine learning algorithms. Special attention may be given to addressing class imbalance issues through techniques like SMOTE or Random Over Sampler to ensure the models are trained on balanced datasets.

6.5 OUTPUT DESIGN

The output design for predicting stroke occurrences using machine learning involves generating actionable insights and predictions based on the trained models. Once the input data is processed and fed into the model, the output typically consists of a binary classification indicating the likelihood of stroke occurrence for each individual in the dataset. These predictions can be presented in various formats, such as probability scores or categorical labels (e.g., "high risk" or "low risk"). Additionally, the output may include metrics evaluating the model's performance, such as accuracy, precision, recall, and F1-score, to provide an assessment of the model's reliability.

CHAPTER 7

7. SYSTEM TESTING AND IMPLEMENTATION

7.1 SYSTEM TESTING

System testing for the stroke prediction machine learning system involves thoroughly assessing its functionality, accuracy, and reliability. This includes testing various components such as data preprocessing, model training, and prediction generation. Functional testing ensures that all features and functionalities of the system work as intended, such as handling missing data, encoding categorical variables, and implementing class imbalance techniques. Accuracy testing involves comparing the model's predictions against known outcomes to evaluate its predictive performance, using metrics like accuracy, precision, recall, and F1-score. Reliability testing assesses the system's stability and consistency in producing accurate predictions over time and across different datasets.

7.2 SYSTEM IMPLEMENTATION

System implementation for the stroke prediction machine learning involves deploying the trained model into a production environment where it can be used to make real-time predictions. This typically involves integrating the model into existing healthcare systems or developing standalone applications for use by healthcare professionals. Implementation also includes setting up infrastructure to support data input, model inference, and output visualization. Furthermore, considerations for scalability, security, and compliance with healthcare regulations are essential during implementation. Regular monitoring and maintenance of the system are necessary to ensure continued accuracy and reliability in predicting stroke occurrences.

CHAPTER 8

SYSTEM MAINTENANCE

The objectives of this maintenance work are to make sure that the system gets into work all time without any bug. Provision must be for environmental changes which may affect the computer or software system. This is called the maintenance of the system. Nowadays there is the rapid change in the software world. Due to this rapid change, the system should be capable of adapting these changes. In this project the process can be added without affecting other parts of the system. Maintenance plays a vital role. The system is liable to accept any modification after its implementation. This system has been designed to favor all new changes. Doing this will not affect the system's performance or its accuracy.

Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environment. It has been seen that there are always some errors found in the system that must be noted and corrected. It also means the review of the system from time to time.

The review of the system is done for:

- Knowing the full capabilities of the system.
- Knowing the required changes or the additional requirements.
- Studying the performance.

TYPES OF MAINTENANCE:

- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance
- Preventive maintenance

8.1 CORRECTIVE MAINTENANCE

Changes made to a system to repair flaws in its design coding or implementation. The design of the software will be changed. The corrective maintenance is applied to correct the errors that occur during that operation time. The user may enter invalid file type while submitting the information in the particular field, then the corrective maintenance will display the error message to the user in order to rectify the error.

Maintenance is a major income source. Nevertheless, even today many organizations assign maintenance to unsupervised beginners, and less competent programmers.

The user's problems are often caused by the individuals who developed the product, not the maintainer. The code itself may be badly written maintenance is despised by many software developers unless good maintenance service is provided, the client will take future development business elsewhere. Maintenance is the most important phase of software production, the most difficult and most thankless.

8.2 ADAPTIVE MAINTENANCE:

It means changes made to system to evolve its functionalities to change business needs or technologies. If any modification in the modules the software will adopt those modifications. If the user changes the server then the project will adapt those changes. The modification server work as the existing is performed.

8.3 PERFECTIVE MAINTENANCE:

Perfective maintenance means made to a system to add new features or improve performance. The perfective maintenance is done to take some perfect measures to maintain the special features. It means enhancing the performance or modifying the programs to respond to the users need or changing needs. This proposed system could be added with additional functionalities easily. In this project, if the user wants to improve the performance further then this software can be easily upgraded.

8.4 PREVENTIVE MAINTENANCE:

Preventive maintenance involves changes made to a system to reduce the chances of system failure. The possible occurrence of error that might occur are forecasted and prevented with suitable preventive problems. If the user wants to improve the performance of any process then the new features can be added to the system for this project.

CHAPTER 9

9. CONCLUSION

In conclusion, our study demonstrates the effectiveness of machine learning models in predicting stroke occurrences based on demographic and health-related features. Through comprehensive data exploration, preprocessing, feature selection, model building, and evaluation, we have showcased the potential of algorithms such as Random Forest Classifier, AdaBoost Classifier, and Gradient Boosting Classifier in accurately identifying individuals at risk of strokes. Furthermore, by employing oversampling techniques like SMOTE and Random Over Sampler to address class imbalance, we have enhanced the robustness of our predictive models.

FUTURE WORK

In future work, we aim to further refine and extend our predictive models for stroke occurrences by incorporating additional features and exploring more advanced machine learning techniques. Specifically, integrating genetic and biomarker data could provide deeper insights into individual susceptibility to strokes and facilitate the development of more personalized risk prediction models. Additionally, we plan to investigate the potential impact of socio-economic factors and environmental variables on stroke risk, enabling a more holistic understanding of the underlying determinants.

CHAPTER 10

APPENDICES

10.1 SOURCE CODE

```
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "\n",
        "# Stroke Predication"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {},
      "outputs": [],
      "source": [
        "# Importing Libraries:\n",
        "import pandas as pd\n",
        "import numpy as np\n",
        "import seaborn as sns\n",
        "import matplotlib.pyplot as plt"
```

```

]
},
{
  "cell_type": "code",
  "execution_count": 2,
  "metadata": {},
  "outputs": [],
  "source": [
    "# for displaying all feature from dataset:\n",
    "pd.pandas.set_option('display.max_columns', None)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 3,
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/html": [
          "<div>\n",
          "<style scoped>\n",
          "  .dataframe tbody tr th:only-of-type {\n",
          "    vertical-align: middle;\n",
          "  }\n",

```

```

"\n",
"  .dataframe tbody tr th {\n",
"    vertical-align: top;\n",
"  }\n",
"\n",
"  .dataframe thead th {\n",
"    text-align: right;\n",
"  }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>id</th>\n",
"      <th>gender</th>\n",
"      <th>age</th>\n",
"      <th>hypertension</th>\n",
"      <th>heart_disease</th>\n",
"      <th>ever_married</th>\n",
"      <th>work_type</th>\n",
"      <th>Residence_type</th>\n",
"      <th>avg_glucose_level</th>\n",
"      <th>bmi</th>\n",
"      <th>smoking_status</th>\n",
"      <th>stroke</th>\n",

```

```

"   </tr>\n",
" </thead>\n",
" <tbody>\n",
"   <tr>\n",
"     <th>0</th>\n",
"     <td>9046</td>\n",
"     <td>Male</td>\n",
"     <td>67.0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>Yes</td>\n",
"     <td>Private</td>\n",
"     <td>Urban</td>\n",
"     <td>228.69</td>\n",
"     <td>36.6</td>\n",
"     <td>formerly smoked</td>\n",
"     <td>1</td>\n",
"   </tr>\n",
" <tr>\n",
"   <th>1</th>\n",
"   <td>51676</td>\n",
"   <td>Female</td>\n",
"   <td>61.0</td>\n",
"   <td>0</td>\n",
"   <td>0</td>\n",

```

```

"    <td>Yes</td>\n",
"    <td>Self-employed</td>\n",
"    <td>Rural</td>\n",
"    <td>202.21</td>\n",
"    <td>NaN</td>\n",
"    <td>never smoked</td>\n",
"    <td>1</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>2</th>\n",
"    <td>31112</td>\n",
"    <td>Male</td>\n",
"    <td>80.0</td>\n",
"    <td>0</td>\n",
"    <td>1</td>\n",
"    <td>Yes</td>\n",
"    <td>Private</td>\n",
"    <td>Rural</td>\n",
"    <td>105.92</td>\n",
"    <td>32.5</td>\n",
"    <td>never smoked</td>\n",
"    <td>1</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>3</th>\n",

```

```

"    <td>60182</td>\n",
"    <td>Female</td>\n",
"    <td>49.0</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>Yes</td>\n",
"    <td>Private</td>\n",
"    <td>Urban</td>\n",
"    <td>171.23</td>\n",
"    <td>34.4</td>\n",
"    <td>smokes</td>\n",
"    <td>1</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>4</th>\n",
"    <td>1665</td>\n",
"    <td>Female</td>\n",
"    <td>79.0</td>\n",
"    <td>1</td>\n",
"    <td>0</td>\n",
"    <td>Yes</td>\n",
"    <td>Self-employed</td>\n",
"    <td>Rural</td>\n",
"    <td>174.12</td>\n",
"    <td>24.0</td>\n",

```

```

"    <td>never smoked</td>\n",
"    <td>1</td>\n",
"  </tr>\n",
" </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"  id gender  age hypertension heart_disease ever_married \\n",
"0  9046  Male  67.0          0          1      Yes \n",
"1  51676 Female  61.0          0          0      Yes \n",
"2  31112  Male  80.0          0          1      Yes \n",
"3  60182 Female  49.0          0          0      Yes \n",
"4  1665  Female  79.0          1          0      Yes \n",
"\n",
"  work_type Residence_type avg_glucose_level  bmi smoking_status \\n",
"0    Private      Urban      228.69 36.6 formerly smoked \n",
"1 Self-employed      Rural      202.21 NaN  never smoked \n",
"2    Private      Rural      105.92 32.5  never smoked \n",
"3    Private      Urban      171.23 34.4      smokes \n",
"4 Self-employed      Rural      174.12 24.0  never smoked \n",
"\n",
"  stroke \n",
"0    1 \n",
"1    1 \n",

```

```

"2    1 \n",
"3    1 \n",
"4    1 "
]
},
"execution_count": 3,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"# Reading Dataset:\n",
"dataset = pd.read_csv(\"Stroke_data.csv\")\n",
"# Top 5 records:\n",
"dataset.head()"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"### Attribute Information\n",
"1) id: unique identifier <br>\n",
"2) gender: \"Male\", \"Female\" or \"Other\" <br>\n",
"3) age: age of the patient <br>\n",

```

"4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
\n",

"5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
\n",

"6) ever_married: \"No\" or \"Yes\"
\n",

"7) work_type: \"children\", \"Govt_jov\", \"Never_worked\", \"Private\" or \"Self-employed\"
\n",

"8) Residence_type: \"Rural\" or \"Urban\"
\n",

"9) avg_glucose_level: average glucose level in blood
\n",

"10) bmi: body mass index
\n",

"11) smoking_status: \"formerly smoked\", \"never smoked\", \"smokes\" or \"Unknown\"*
\n",

"12) stroke: 1 if the patient had a stroke or 0 if not
\n",

"*Note: \"Unknown\" in smoking_status means that the information is unavailable for this patient"

]

},

{

"cell_type": "code",

"execution_count": 4,

"metadata": {},

"outputs": [],

"source": [

"# Dropping unneccsary feature :\n",

"dataset = dataset.drop('id', axis=1)"

]

},

```

{
  "cell_type": "code",
  "execution_count": 5,
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/plain": [
          "(5110, 11)"
        ]
      },
      "execution_count": 5,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "# Shape of dataset:\n",
    "dataset.shape"
  ]
},
{
  "cell_type": "code",
  "execution_count": 6,
  "metadata": {},

```

```

"outputs": [
{
  "data": {
    "text/plain": [
      "gender          0\n",
      "age             0\n",
      "hypertension     0\n",
      "heart_disease    0\n",
      "ever_married     0\n",
      "work_type        0\n",
      "Residence_type   0\n",
      "avg_glucose_level 0\n",
      "bmi              201\n",
      "smoking_status    0\n",
      "stroke           0\n",
      "dtype: int64"
    ]
  },
  "execution_count": 6,
  "metadata": {},
  "output_type": "execute_result"
}
],
"source": [
  "# Cheaking Missing (NaN) Values:\n",

```

```

"dataset.isnull().sum()"
]
},
{
"cell_type": "code",
"execution_count": 7,
"metadata": {},
"outputs": [],
"source": [
"# Filling NaN Values in BMI feature using mean:\n",
"dataset['bmi'] = dataset['bmi'].fillna(dataset['bmi'].median())"
]
},
{
"cell_type": "code",
"execution_count": 8,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"gender      0\n",
"age         0\n",
"hypertension 0\n",
"heart_disease 0\n",

```

```

    "ever_married      0\n",
    "work_type         0\n",
    "Residence_type     0\n",
    "avg_glucose_level  0\n",
    "bmi                0\n",
    "smoking_status     0\n",
    "stroke             0\n",
    "dtype: int64"
]
},
"execution_count": 8,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
    "# After filling Missing (NaN) Values in BMI feature:\n",
    "dataset.isnull().sum()"
]
},
{
    "cell_type": "code",
    "execution_count": 9,
    "metadata": {},
    "outputs": [

```

```

{
  "data": {
    "text/html": [
      "<div>\n",
      "<style scoped>\n",
      "  .dataframe tbody tr th:only-of-type {\n",
      "    vertical-align: middle;\n",
      "  }\n",
      "\n",
      "  .dataframe tbody tr th {\n",
      "    vertical-align: top;\n",
      "  }\n",
      "\n",
      "  .dataframe thead th {\n",
      "    text-align: right;\n",
      "  }\n",
      "</style>\n",
      "<table border='1' class='dataframe'>\n",
      "  <thead>\n",
      "    <tr style='text-align: right;'>\n",
      "      <th></th>\n",
      "      <th>age</th>\n",
      "      <th>hypertension</th>\n",
      "      <th>heart_disease</th>\n",
      "      <th>avg_glucose_level</th>\n",

```

```

"    <th>bmi</th>\n",
"    <th>stroke</th>\n",
"  </tr>\n",
" </thead>\n",
" <tbody>\n",
"   <tr>\n",
"     <th>count</th>\n",
"     <td>5110.000000</td>\n",
"     <td>5110.000000</td>\n",
"     <td>5110.000000</td>\n",
"     <td>5110.000000</td>\n",
"     <td>5110.000000</td>\n",
"     <td>5110.000000</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>mean</th>\n",
"     <td>43.226614</td>\n",
"     <td>0.097456</td>\n",
"     <td>0.054012</td>\n",
"     <td>106.147677</td>\n",
"     <td>28.862035</td>\n",
"     <td>0.048728</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>std</th>\n",

```

```

"    <td>22.612647</td>\n",
"    <td>0.296607</td>\n",
"    <td>0.226063</td>\n",
"    <td>45.283560</td>\n",
"    <td>7.699562</td>\n",
"    <td>0.215320</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>min</th>\n",
"    <td>0.080000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>55.120000</td>\n",
"    <td>10.300000</td>\n",
"    <td>0.000000</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>25%</th>\n",
"    <td>25.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>77.245000</td>\n",
"    <td>23.800000</td>\n",
"    <td>0.000000</td>\n",
"  </tr>\n",

```

```

" <tr>\n",
"   <th>50%</th>\n",
"   <td>45.000000</td>\n",
"   <td>0.000000</td>\n",
"   <td>0.000000</td>\n",
"   <td>91.885000</td>\n",
"   <td>28.100000</td>\n",
"   <td>0.000000</td>\n",
" </tr>\n",
" <tr>\n",
"   <th>75%</th>\n",
"   <td>61.000000</td>\n",
"   <td>0.000000</td>\n",
"   <td>0.000000</td>\n",
"   <td>114.090000</td>\n",
"   <td>32.800000</td>\n",
"   <td>0.000000</td>\n",
" </tr>\n",
" <tr>\n",
"   <th>max</th>\n",
"   <td>82.000000</td>\n",
"   <td>1.000000</td>\n",
"   <td>1.000000</td>\n",
"   <td>271.740000</td>\n",
"   <td>97.600000</td>\n",

```

```

"    <td>1.000000</td>\n",
"  </tr>\n",
" </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"      age hypertension heart_disease avg_glucose_level \\n",
"count 5110.000000 5110.000000 5110.000000 5110.000000 \n",
"mean  43.226614  0.097456  0.054012  106.147677 \n",
"std   22.612647  0.296607  0.226063  45.283560 \n",
"min   0.080000  0.000000  0.000000  55.120000 \n",
"25%   25.000000  0.000000  0.000000  77.245000 \n",
"50%   45.000000  0.000000  0.000000  91.885000 \n",
"75%   61.000000  0.000000  0.000000  114.090000 \n",
"max   82.000000  1.000000  1.000000  271.740000 \n",
"\n",
"      bmi      stroke \n",
"count 5110.000000 5110.000000 \n",
"mean  28.862035  0.048728 \n",
"std   7.699562  0.215320 \n",
"min   10.300000  0.000000 \n",
"25%   23.800000  0.000000 \n",
"50%   28.100000  0.000000 \n",
"75%   32.800000  0.000000 \n",

```

```

    "max    97.600000    1.000000 "
  ]
},
"execution_count": 9,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
  "# Description:\n",
  "dataset.describe()"
]
},
{
  "cell_type": "code",
  "execution_count": 10,
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/plain": [
          "gender          object\n",
          "age             float64\n",
          "hypertension     int64\n",
          "heart_disease   int64\n",

```

```

    "ever_married      object\n",
    "work_type        object\n",
    "Residence_type    object\n",
    "avg_glucose_level  float64\n",
    "bmi               float64\n",
    "smoking_status     object\n",
    "stroke            int64\n",
    "dtype: object"
]
},
"execution_count": 10,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
    "# Datatypes:\n",
    "dataset.dtypes"
]
},
{
    "cell_type": "code",
    "execution_count": 11,
    "metadata": {},
    "outputs": [

```

```

{
  "name": "stdout",
  "output_type": "stream",
  "text": [
    "Stroke People    : 249\n",
    "Non-Stroke People : 4861\n"
  ]
},
{
  "source": [
    "# Target feature:\n",
    "print(\"Stroke People    : \", dataset['stroke'].value_counts()[1])\n",
    "print(\"Non-Stroke People : \", dataset['stroke'].value_counts()[0])"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "- By seeing target feature, We clearly say we have imbalanced dataset."
  ]
},
{
  "cell_type": "code",
  "execution_count": 12,

```

```

"metadata": {},
"outputs": [
  {
    "data": {
      "text/plain": [
        "Female    2994\n",
        "Male       2115\n",
        "Other        1\n",
        "Name: gender, dtype: int64"
      ]
    },
    "execution_count": 12,
    "metadata": {},
    "output_type": "execute_result"
  },
  {
    "source": [
      "# Gender\n",
      "dataset['gender'].value_counts()"
    ],
    "cell_type": "code",
    "execution_count": 13,
    "metadata": {},

```

```

"outputs": [
{
  "data": {
    "text/html": [
      "<div>\n",
      "<style scoped>\n",
      "  .dataframe tbody tr th:only-of-type {\n",
      "    vertical-align: middle;\n",
      "  }\n",
      "\n",
      "  .dataframe tbody tr th {\n",
      "    vertical-align: top;\n",
      "  }\n",
      "\n",
      "  .dataframe thead th {\n",
      "    text-align: right;\n",
      "  }\n",
      "</style>\n",
      "<table border='1' class='dataframe'>\n",
      "  <thead>\n",
      "    <tr style='text-align: right;'>\n",
      "      <th></th>\n",
      "      <th>gender</th>\n",
      "      <th>age</th>\n",
      "      <th>hypertension</th>\n",

```

```

"   <th>heart_disease</th>\n",
"   <th>ever_married</th>\n",
"   <th>work_type</th>\n",
"   <th>Residence_type</th>\n",
"   <th>avg_glucose_level</th>\n",
"   <th>bmi</th>\n",
"   <th>smoking_status</th>\n",
"   <th>stroke</th>\n",
" </tr>\n",
" </thead>\n",
" <tbody>\n",
"   <tr>\n",
"     <th>3116</th>\n",
"     <td>Other</td>\n",
"     <td>26.0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>No</td>\n",
"     <td>Private</td>\n",
"     <td>Rural</td>\n",
"     <td>143.33</td>\n",
"     <td>22.4</td>\n",
"     <td>formerly smoked</td>\n",
"     <td>0</td>\n",
"   </tr>\n",

```

```

" </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"  gender  age hypertension heart_disease ever_married work_type \\n",
"3116 Other 26.0      0      0      No  Private  \n",
"\n",
"  Residence_type avg_glucose_level  bmi smoking_status stroke \n",
"3116      Rural      143.33 22.4 formerly smoked      0 "
]
},
"execution_count": 13,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"dataset[dataset['gender']=='Other']"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [

```

"- We Seen that in our **Gender** feature, we have only one **Other** gender, So instead of taking we drop that record."

```
]
},
{
  "cell_type": "code",
  "execution_count": 14,
  "metadata": {},
  "outputs": [],
  "source": [
    "# Dropping Other gender\n",
    "Other_gender = dataset[dataset['gender'] == 'Other'].index[0]\n",
    "dataset = dataset.drop(Other_gender, axis=0)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 15,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "Male   : 2115\n",

```

```

    "female : 2994\n"
  ]
}
],
"source": [
  "# Gender:\n",
  "print(\"Male   : \", dataset['gender'].value_counts()[1])\n",
  "print(\"female : \", dataset['gender'].value_counts()[0])"
]
},
{
  "cell_type": "code",
  "execution_count": 16,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "Hypertension People   : 498\n",
        "Non-hypertension People : 4611\n"
      ]
    }
  ],
  "source": [

```

```

"# Hypertension:\n",
"print(\"Hypertension People   : \", dataset['hypertension'].value_counts()[1])\n",
"print(\"Non-hypertension People : \", dataset['hypertension'].value_counts()[0])"
]
},
{
"cell_type": "code",
"execution_count": 17,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Heart Disease People   : 276\n",
"Non-Heart Disease People : 4833\n"
]
}
],
"source": [
"# Heart Disease:\n",
"print(\"Heart Disease People   : \", dataset['heart_disease'].value_counts()[1])\n",
"print(\"Non-Heart Disease People : \", dataset['heart_disease'].value_counts()[0])"
]
},

```

```

{
  "cell_type": "code",
  "execution_count": 18,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "Single   : 1756\n",
        "Married  : 3353\n"
      ]
    }
  ],
  "source": [
    "# Single VS Married:\n",
    "print(\"Single   : \", dataset['ever_married'].value_counts()[1])\n",
    "print(\"Married  : \", dataset['ever_married'].value_counts()[0])"
  ]
},
{
  "cell_type": "code",
  "execution_count": 19,
  "metadata": {},
  "outputs": [

```

```

{
  "name": "stdout",
  "output_type": "stream",
  "text": [
    "Private      : 2924\n",
    "Self-employed : 819\n",
    "children     : 687\n",
    "Govt_job     : 657\n",
    "Never_worked : 22\n"
  ]
}
],
"source": [
  "# Work Type:\n",
  "print(\"Private      : \", dataset['work_type'].value_counts()[0])\n",
  "print(\"Self-employed : \", dataset['work_type'].value_counts()[1])\n",
  "print(\"children     : \", dataset['work_type'].value_counts()[2])\n",
  "print(\"Govt_job     : \", dataset['work_type'].value_counts()[3])\n",
  "print(\"Never_worked  : \", dataset['work_type'].value_counts()[4])"
]
},
{
  "cell_type": "code",
  "execution_count": 20,
  "metadata": {},

```

```

"outputs": [],
"source": [
    "# Rename some names in worktype feature for simplicity nothing else:\n",
    "dataset.replace({'Self-employed' : 'Self_employed'}, inplace=True)"
]
},
{
    "cell_type": "code",
    "execution_count": 21,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "Urban   : 2596\n",
                "Rural   : 2513\n"
            ]
        }
    ],
    "source": [
        "# Residence Type:\n",
        "print(\"Urban   : \", dataset['Residence_type'].value_counts()[0])\n",
        "print(\"Rural   : \", dataset['Residence_type'].value_counts()[1])"
    ]
}

```

```

from flask import Flask, render_template, request

import numpy as np

import pickle


app = Flask(__name__)

model = pickle.load(open('Stroke.pkl', 'rb'))


@app.route('/', methods=['GET'])
def Home():
    return render_template('index.html')


@app.route("/predict", methods=['POST'])
def predict():
    if request.method == 'POST':
        gender = request.form['gender']
        if gender == 'Male':
            gender_Male = 1
            gender_Female = 0
        else:
            gender_Male = 0
            gender_Female = 1

        age = float(request.form['age'])
        hypertension = int(request.form['hypertension'])

```

```
heart_disease = int(request.form['heart_disease'])
ever_married = int(request.form['ever_married'])
Residence_type = int(request.form['Residence_type'])
avg_glucose_level = float(request.form['avg_glucose_level'])
bmi = float(request.form['bmi'])
```

```
work_type = request.form['work_type']
```

```
if work_type == 'Never_worked':
```

```
    work_type_Never_worked = 1
    work_type_Private = 0
    work_type_Self_employed = 0
    work_type_children = 0
    work_type_Govt_job = 0
```

```
if work_type == 'Private':
```

```
    work_type_Never_worked = 0
    work_type_Private = 1
    work_type_Self_employed = 0
    work_type_children = 0
    work_type_Govt_job = 0
```

```
elif work_type == "Self_employed":
```

```
    work_type_Never_worked = 0
```

```
work_type_Private = 0
work_type_Self_employed = 1
work_type_children = 0
work_type_Govt_job = 0
```

```
elif work_type == "children":
```

```
    work_type_Never_worked = 0
    work_type_Private = 0
    work_type_Self_employed = 0
    work_type_children = 1
    work_type_Govt_job = 0
```

```
else:
```

```
    work_type_Never_worked = 0
    work_type_Private = 0
    work_type_Self_employed = 0
    work_type_children = 0
    work_type_Govt_job = 1
```

```
smoking_status = request.form['smoking_status']
```

```
if smoking_status == "formerly_smoked":
```

```
    smoking_status_formerly_smoked = 1
    smoking_status_never_smoked = 0
```

```
smoking_status_Smokes = 0  
smoking_status_Unknown = 0
```

```
elif smoking_status == "never_smoked":  
    smoking_status_formerly_smoked = 0  
    smoking_status_never_smoked = 1  
    smoking_status_Smokes = 0  
    smoking_status_Unknown = 0
```

```
elif smoking_status == "Smokes":  
    smoking_status_formerly_smoked = 0  
    smoking_status_never_smoked = 0  
    smoking_status_Smokes = 1  
    smoking_status_Unknown = 0
```

```
else:  
    smoking_status_formerly_smoked = 0  
    smoking_status_never_smoked = 0  
    smoking_status_Smokes = 0  
    smoking_status_Unknown = 1
```

```
values = np.array([[gender_Male,age, hypertension, heart_disease, ever_married,  
Residence_type, avg_glucose_level, bmi,
```

```
work_type_Never_worked, work_type_Private, work_type_Self_employed,  
work_type_children,
```

```
smoking_status_formerly_smoked, smoking_status_never_smoked,  
smoking_status_Smokes]])
```

```
prediction = model.predict(values)
```

```
return render_template('result.html', prediction=prediction)
```

```
if __name__ == "__main__":
```

```
    app.run(debug=True)
```

10.2 SCREEN SHOTS

The screenshot shows a Jupyter Notebook titled "Stroke - Jupyter Notebook" running on a local host. The notebook contains three code cells and one output cell. The first code cell imports libraries: pandas, numpy, seaborn, and matplotlib. The second code cell sets the display options for pandas. The third code cell reads the "Stroke_data.csv" file and displays the first five records. The output cell shows a table with 12 columns: id, gender, age, hypertension, heart_disease, ever_married, work_type, Residence_type, avg_glucose_level, bmi, smoking_status, and stroke.

```
In [1]: # Importing Libraries:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

In [2]: # for displaying all feature from dataset:
pd.pandas.set_option("display.max_columns", None)

In [3]: # Reading Dataset:
dataset = pd.read_csv("Stroke_data.csv")
# Top 5 records:
dataset.head()

Out[3]:
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.89	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

The screenshot shows a Jupyter Notebook titled "Stroke - Jupyter Notebook" running on a local host. The notebook contains three code cells and one output cell. The first code cell drops the 'id' column. The second code cell displays the shape of the dataset. The third code cell checks for missing values in the 'gender' column. The output cell shows the result of the missing value check.

```
In [4]: # Dropping unnecessary feature :
dataset = dataset.drop('id', axis=1)

In [5]: # Shape of dataset:
dataset.shape

Out[5]: (5110, 11)

In [6]: # Checking Missing (NaN) Values:
dataset.isnull().sum()

Out[6]: gender 0
```

Attribute Information

1. id: unique identifier
2. gender: "Male", "Female" or "Other"
3. age: age of the patient
4. hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
5. heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
6. ever_married: "No" or "Yes"
7. work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
8. Residence_type: "Rural" or "Urban"
9. avg_glucose_level: average glucose level in blood
10. bmi: body mass index
11. smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"
12. stroke: 1 if the patient had a stroke or 0 if not

Note: "Unknown" in smoking_status means that the information is unavailable for this patient

Desktop/stroke_app/ x Stroke - Jupyter Notebook x +

localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
hypertension      0
heart_disease     0
ever_married      0
work_type         0
Residence_type    0
avg_glucose_level 0
bmi              201
smoking_status    0
stroke            0
dtype: int64
```

In [7]: # Filling NaN Values in BMI feature using mean:
dataset['bmi'] = dataset['bmi'].fillna(dataset['bmi'].median())

In [8]: # After filling Missing (NaN) Values in BMI feature:
dataset.isnull().sum()

Out[8]:

```
gender      0
age         0
hypertension 0
heart_disease 0
ever_married 0
work_type    0
Residence_type 0
avg_glucose_level 0
bmi         0
smoking_status 0
stroke      0
dtype: int64
```

In [9]: # Description:

Desktop/stroke_app/ x Stroke - Jupyter Notebook x +

localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Out[9]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000
mean	43.226614	0.097456	0.054012	106.147677	28.862035	0.048728
std	22.612647	0.296607	0.226063	45.283560	7.699562	0.215320
min	0.000000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	25.000000	0.000000	0.000000	77.245000	23.800000	0.000000
50%	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	61.000000	0.000000	0.000000	114.090000	32.800000	0.000000
max	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

In [10]: # Datatypes:
dataset.dtypes

Out[10]:

```
gender      object
age         float64
hypertension int64
heart_disease int64
ever_married object
work_type    object
Residence_type object
avg_glucose_level float64
bmi          float64
smoking_status object
stroke      int64
dtype: object
```

Desktop/stroke_app/ Stroke - Jupyter Notebook

localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Out[13]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
3116	Other	26.0	0	0	No	Private	Rural	143.33	22.4	formerly smoked	0

• We Seen that in our Gender feature, we have only one Other gender, So instead of taking we drop that record.

In [14]:

```
# Dropping Other gender
Other_gender = dataset[dataset['gender'] == 'Other'].index[0]
dataset = dataset.drop(Other_gender, axis=0)
```

In [15]:

```
# Gender:
print("Male : ", dataset['gender'].value_counts()[1])
print("female : ", dataset['gender'].value_counts()[0])
```

Male : 2115
female : 2994

In [16]:

```
# Hypertension:
print("Hypertension People : ", dataset['hypertension'].value_counts()[1])
print("Non-hypertension People : ", dataset['hypertension'].value_counts()[0])
```

Hypertension People : 498
Non-hypertension People : 4611

In [17]:

```
# Heart Disease:
print("Heart Disease People : ", dataset['heart_disease'].value_counts()[1])
print("Non-Heart Disease People : ", dataset['heart_disease'].value_counts()[0])
```

Desktop/stroke_app/ Stroke - Jupyter Notebook

localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [18]:

```
# Single VS Married:
print("Single : ", dataset['ever_married'].value_counts()[1])
print("Married : ", dataset['ever_married'].value_counts()[0])
```

Single : 1756
Married : 3353

In [19]:

```
# Work Type:
print("Private : ", dataset['work_type'].value_counts()[0])
print("Self-employed : ", dataset['work_type'].value_counts()[1])
print("children : ", dataset['work_type'].value_counts()[2])
print("Govt_job : ", dataset['work_type'].value_counts()[3])
print("Never_worked : ", dataset['work_type'].value_counts()[4])
```

Private : 2924
Self-employed : 819
children : 687
Govt_job : 657
Never_worked : 22

In [20]:

```
# Rename some names in worktype feature for simploty nothing else:
dataset.replace({'Self-employed': 'Self_employed'}, inplace=True)
```

In [21]:

```
# Residence Type:
print("Urban : ", dataset['Residence_type'].value_counts()[0])
print("Rural : ", dataset['Residence_type'].value_counts()[1])
```

Urban : 2596
Rural : 2513

Desktop/stroke_app/ Stroke - Jupyter Notebook localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
Never Smoked : 1892
Unknown      : 1544
Formerly Smoked : 884
Smokes       : 789
```

In [23]: *# Rename some names in smokers feature for simplicity nothing else:*
dataset.replace({'never smoked':'never_smoked', 'formerly smoked':'formerly_smoked'}, inplace=True)

In [24]: never_smoked, Unknown, formerly_smoked, Smokes

NameError Traceback (most recent call last)
Cell In[24], line 1
----> 1 never_smoked, Unknown, formerly_smoked, Smokes
NameError: name 'never_smoked' is not defined

In []: *# Boxplot:*
plt.figure(figsize=(15,12))
dataset.plot(kind='box', subplots=True, layout=(2,3), figsize=(20, 10))
plt.show()

In []: *# Correlation using Heatmap:*
plt.figure(figsize=(12,8))
sns.heatmap(dataset.corr(), annot=True, cmap='YlGnBu')
plt.show()

In [25]: dataset.head()

Desktop/stroke_app/ Stroke - Jupyter Notebook localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
plt.show()
```

In [25]: dataset.head()

Out[25]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly_smoked	1
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.1	never_smoked	1
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never_smoked	1
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never_smoked	1

In [26]: *# Dependent & Independent Feature:*
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

In [27]: X.head()

Out[27]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly_smoked
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.1	never_smoked
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never_smoked
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never_smoked

Desktop/stroke_app/ Stroke - Jupyter Notebook

localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [36]: # Importing Performance Metrics:
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

In [37]: # RandomForestClassifier:
from sklearn.ensemble import RandomForestClassifier
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(X_train,y_train)

# Predictions:
y_pred = RandomForest.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

Accuracy: 0.9471624266144814
[[968 0]
[54 0]]

	precision	recall	f1-score	support
0	0.95	1.00	0.97	968
1	0.00	0.00	0.00	54
accuracy			0.95	1022
macro avg	0.47	0.50	0.49	1022
weighted avg	0.90	0.95	0.92	1022

C:\Users\KOKUSHI\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics_classification.py:1509: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.

Desktop/stroke_app/ Stroke - Jupyter Notebook

localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [38]: # AdaBoostClassifier:
from sklearn.ensemble import AdaBoostClassifier
AdaBoost = AdaBoostClassifier()
AdaBoost = AdaBoost.fit(X_train,y_train)

# Predictions:
y_pred = AdaBoost.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

C:\Users\KOKUSHI\AppData\Roaming\Python\Python311\site-packages\sklearn\ensemble_weight_boosting.py:519: FutureWarning: The SAM ME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
warnings.warn(

Accuracy: 0.9461839530332681
[[966 2]
[53 1]]

	precision	recall	f1-score	support
0	0.95	1.00	0.97	968
1	0.33	0.02	0.04	54
accuracy			0.95	1022
macro avg	0.64	0.51	0.50	1022
weighted avg	0.92	0.95	0.92	1022

In [39]: # GradientBoostingClassifier:

Desktop/stroke_app/ Stroke - Jupyter Notebook

localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [39]:

```
# GradientBoostingClassifier:
from sklearn.ensemble import GradientBoostingClassifier
GradientBoost = GradientBoostingClassifier()
GradientBoost = GradientBoost.fit(X_train,y_train)

# Predictions:
y_pred = GradientBoost.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

Accuracy: 0.9461839530332681

```
[[966  2]
 [ 53  1]]
```

	precision	recall	f1-score	support
0	0.95	1.00	0.97	968
1	0.33	0.02	0.04	54
accuracy			0.95	1022
macro avg	0.64	0.51	0.50	1022
weighted avg	0.92	0.95	0.92	1022

RandomizedSearchCV

Desktop/stroke_app/ Stroke - Jupyter Notebook

localhost:8888/notebooks/Desktop/stroke_app/Stroke.ipynb

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Stroke Last Checkpoint: 06/01/2024 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

RandomizedSearchCV

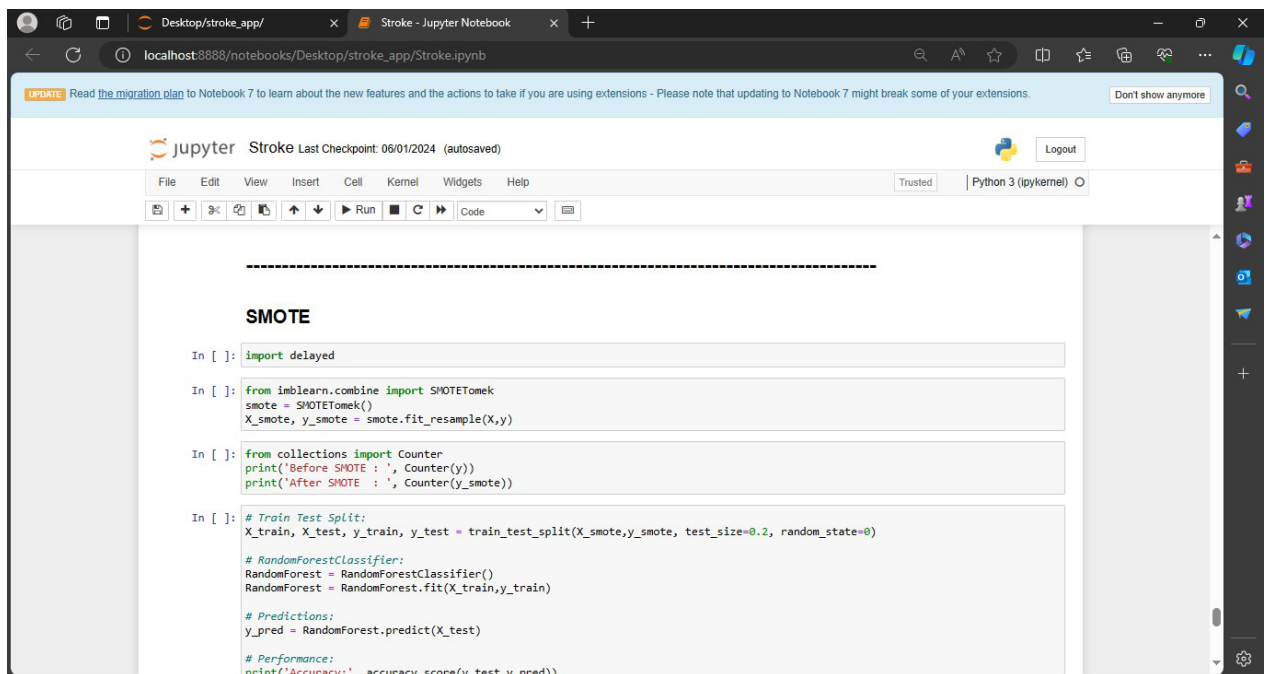
In [40]:

```
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 200, num = 20)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt', 'log2']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 200,20)]
# Minimum number of samples required to split a node
min_samples_split = [1,2,3,4,5,6,7,8,9,10,12,14,16,18,20]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1,2,3,4,5,6,7,8,9,10,12,14,16,18,20]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'criterion':['entropy','gini']}
print(random_grid)

rf = RandomForestClassifier()
rf_randomcv = RandomizedSearchCV(estimator=rf,param_distributions=random_grid,n_iter=100,cv=10,verbose=2,
                                random_state=0,n_jobs=-1)

## fit the randomized model
rf_randomcv.fit(X_train,y_train)
```

```
{'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200], 'max_features': ['auto', 'sqrt', 'log2'], 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200], 'min_samples_split': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20], 'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20]}
```



The screenshot shows a Jupyter Notebook interface with the title 'Stroke'. The notebook is running on a Python 3 (ipykernel) environment. The code in the notebook implements SMOTE (Synthetic Minority Over-sampling Technique) using the `SMOTETomek` class from `imblearn.combine`. It includes imports for `delayed`, `SMOTETomek`, `Counter`, and `train_test_split`. The code splits the data into training and testing sets, fits a `RandomForestClassifier` on the training data, and prints the accuracy score on the test data.

```
In [ ]: import delayed

In [ ]: from imblearn.combine import SMOTETomek
smote = SMOTETomek()
X_smote, y_smote = smote.fit_resample(X,y)

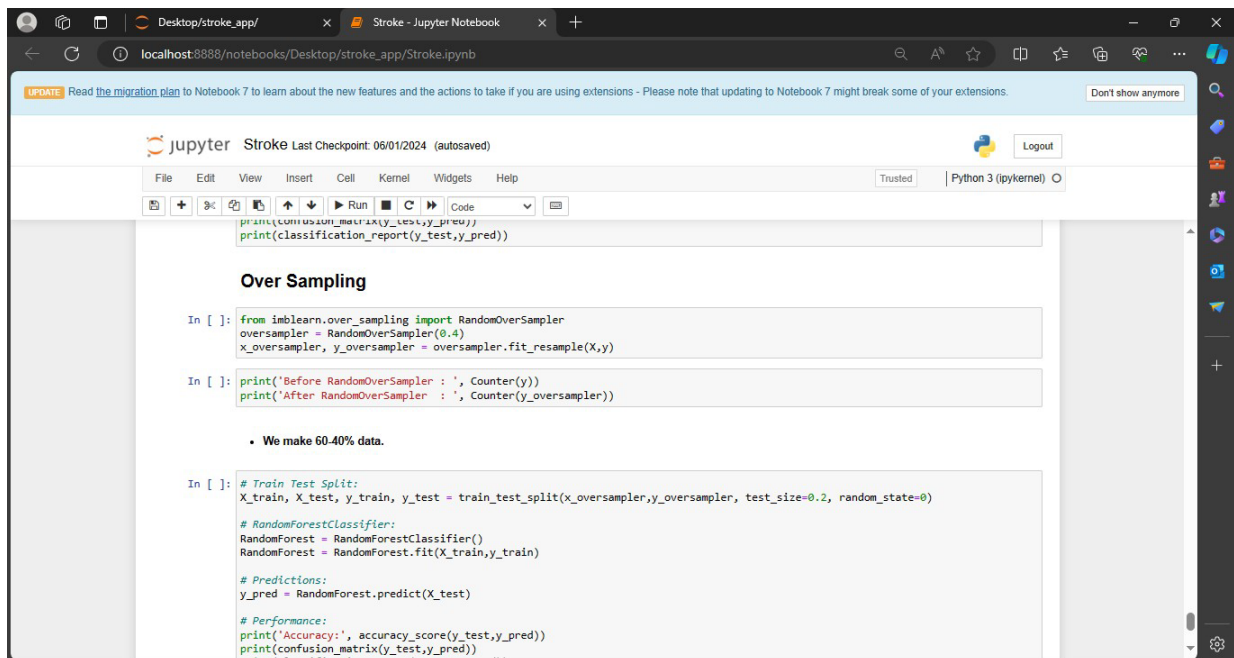
In [ ]: from collections import Counter
print('Before SMOTE : ', Counter(y))
print('After SMOTE : ', Counter(y_smote))

In [ ]: # Train Test Split:
X_train, X_test, y_train, y_test = train_test_split(X_smote,y_smote, test_size=0.2, random_state=0)

# RandomForestClassifier:
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(X_train,y_train)

# Predictions:
y_pred = RandomForest.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
```



The screenshot shows a Jupyter Notebook interface with the title 'Stroke'. The notebook is running on a Python 3 (ipykernel) environment. The code in the notebook implements Over Sampling using the `RandomOverSampler` class from `imblearn.over_sampling`. It includes imports for `RandomOverSampler`, `train_test_split`, `RandomForestClassifier`, `accuracy_score`, and `confusion_matrix`. The code splits the data into training and testing sets, fits a `RandomForestClassifier` on the training data, and prints the accuracy score and confusion matrix on the test data.

```
In [ ]: from imblearn.over_sampling import RandomOverSampler
oversampler = RandomOverSampler(0.4)
x_oversampler, y_oversampler = oversampler.fit_resample(X,y)

In [ ]: print('Before RandomOverSampler : ', Counter(y))
print('After RandomOverSampler : ', Counter(y_oversampler))

• We make 60-40% data.

In [ ]: # Train Test Split:
X_train, X_test, y_train, y_test = train_test_split(x_oversampler,y_oversampler, test_size=0.2, random_state=0)

# RandomForestClassifier:
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(X_train,y_train)

# Predictions:
y_pred = RandomForest.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

CHAPTER 11

11. REFERENCES

- [1] V. Singh and A. K. Misra, “Prediction of Brain Stroke Severity Using Machine Learning,” *Inf. Process. Agricult.*, vol. 4, pp. 41–49, Mar. 2017.
- [2] S. H. Lee, C. S. Chan, S. J. Mayo, and P. Remagnino, “Performance Analysis of Machine Learning Approaches in Stroke Prediction,” *Pattern Recognit.*, vol. 71, pp. 1–13, Nov. 2017
- [3] G. Farjon, O. Krikeb, A. B. Hillel, and V. Alchanatis, “A Machine Learning Approach to Detect the Brain Stroke Disease” *Precis. Agricult.*, vol. 21, pp. 1–19, Aug. 2019.
- [4] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Stroke prediction analysis using machine learning classifiers and feature technique,” *Frontiers Plant Sci.*, vol. 7, p. 1419, Sep. 2016
- [5] F. Ren, W. Liu, and G. Wu, “Predicting Stroke Risk With an Interpretable Classifier,” *IEEE Access*, vol. 7, pp. 122758–122768, 2019
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Prediction of Hemorrhagic Transformation Severity in Acute Stroke from Source Perfusion MRI,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25, Dec. 2012, pp. 1097–1105.
- [7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Early Stroke Prediction Methods for Prevention of Strokes,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
- [8] K. Simonyan and A. Zisserman, “Predictive Analysis for Risk of Stroke Using Machine Learning Techniques,” 2014, arXiv:1409.1556
- [9] K. He, X. Zhang, S. Ren, and J. Sun, “Stroke Disease Detection and Prediction Using Robust Learning Approaches,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778
- [10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Prediction of Brain Stroke using Machine Learning Algorithms and Deep Neural Network Techniques” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.