A Comparative Analysis of Machine Learning in Stroke Prediction

Iffat Ara Sanzida (344, iffat.stu2018@juniv.edu) Jannatul Ferdoush Jannati (349, jannatul.stu2018@juniv.edu)
Sumaita Binte Shorif (357, sumaita.stu2018@juniv.edu) Asmaul Shahana Begum (360, shahana.stu2018@juniv.edu)
Mubasher Adnan Jihad (374, adnan.stu2018@juniv.edu)
Department of CSE, Jahangirnagar University
Dhaka, Bangladesh

***Abstract*—Among the leading causes of death, stroke is third which is caused by the blood circulation interruption in the brain. The post-effects of stroke include paralysis, vision problems, etc. An early prediction of stroke, based on different health conditions like blood pressure, diabetes, heart disease, smoking, etc. can lead one to take precautions for it. In this paper, we will predict stoke using some ML algorithms: Logistic Regression(LR), Random Forest(RF), Synthetic Minority Over-sampling Technique(SMOTE), and Extreme Gradient Boosting(XGBoost). Lastly, we will compare the results of these algorithms.**

***Index Terms*—Stroke, Machine Learning, prediction, Random Forest, SMOTE, Logistic Regression, XGBoost**

I. Introduction

Stroke is still a major worldwide health problem because it frequently causes severe disability and death. Predicting strokes matters today because it helps identify individual risks, promotes proactive health measures, and enables early interventions to potentially save lives and reduce the impact of strokes on communities. Thanks to advances in clinical science, it is now possible to predict the likelihood of a stroke by using machine learning algorithms[1]. In this paper, we are considering hypertension, BMI level, heart disease, smoking habit, and average glucose level as the vital parameters for predicting stroke[3].

The motivation for choosing the logistic regression model is it can be used to predict new subjects and it is very effective in health research[5]. SMOTE is a general method that can be used for regular balanced classification problems of small-sized data[6]. Random forest with parameters setting is helpful for anomalies detection[7]. Lastly, the XGBoost algorithm reduces the training time in turn it increases the performance[8].

II. Literature review

A study [11] systematically analyzes electronic health records to identify key factors for stroke prediction, emphasizing the importance of age, heart disease, average glucose level, and hypertension. Employing statistical techniques and neural network models, the research underscores the significance of these factors in achieving accurate predictions, particularly on balanced datasets addressing the inherent imbalance in stroke occurrences. Another study [12] addresses the challenge of predicting stroke in an elderly Chinese population using machine learning models on imbalanced data. Employing techniques such as random over-sampling, random under-sampling, and SMOTE, the research demonstrates significant improvement in prediction accuracy and sensitivity. Sex, hypertension, and uric acid emerged as common predictors across multiple machine-learning methods, highlighting their importance in stroke prediction. In a comprehensive approach in[3] for early stroke prediction, ten classifiers were trained and combined using weighted voting, achieving a remarkable 97% accuracy. The weighted voting classifier outperformed individual classifiers, exhibiting a high area under the curve and the lowest false positive and false negative rates. It is a promising tool for stroke prediction in clinical practice. The research employs various physiological parameters and machine learning algorithms, including Logistic Regression, Decision Tree Classification, Random Forest Classification, and Voting Classifier, to develop reliable stroke prediction models. In this investigation, random Forest emerges as the top-performing algorithm with an impressive 96% accuracy on the Stroke Prediction dataset[13]. Another research employs the SMOTE oversampling method and classifiers like Logistic Regression, Random Forest, and XGBoost. The random forest model achieves an impressive 99.07% accuracy, outperforming previous works. Feature importance analysis highlights key factors influencing model performance, contributing to its effectiveness in stroke prediction[14].

In [15], five different machine learning models were built for accurate stroke prediction, taking into consideration a various physiological factor. Among the models Naïve Bayes attained the highest accuracy of 82% whereas logistic regression was 78% and random forest classification was 66%. The authors additionally built a web page to let users input parameter and returns the predicted output. This research is limited to few accuracy metrics.

The study in [16], a random forest achieved the highest accuracy, showcasing its superiority over traditional models such as logistic regression, decision tree classifier and KNN. The study incorporates exploratory data analysis (EDA) and SMOTE feature engineering for the imbalanced dataset ensuring robust preprocessing. A cloud based mobile application

was built to collect data and provide the likelihood of stroke for a user further emphasizes the practicality of the models.

The research in [17], used a database of 10967 patients and compared the predictive performance of machine learning models and the logistic model. Out of the classification algorithms CatBoost model had the highest AUC of 0.839 followed by XGBoost, Gradient Boosting Decision Tree (GBDT), Random Forest (RF) and AdaBoost (Ada) having AUC of 0.838, 0.835, 0.832, 0.823 respectively. The study further observed that all of the machine learning classifiers had higher than AUC than the Logistic model.

In [18], the studyfocuses on the performance of various machine learning models, including Gaussian Naive Bayes, Logistic Regression, Decision Tree Classifier, K-Nearest Neighbours, AdaBoost Classifier, XGBoost Classifier, and Random Forest Classifier for stroke prediction. Comparing the models, AdaBoost, XGBoost and Random Forest Classifier emerge as the most accurate with the highest accuracy. The study highlights the potential of these models for better predictions.

The study in [19], presents an approach for developing classifiers from imbalanced datasets. The research combines oversampling the minority class and under-sampling the majority class to enhance classifier sensitivity to the minority class highlighting the importance of SMOTE. technique.

III. Dataset

The dataset for the research is obtained from Kaggle for stroke prediction named as ”brain-stroke.csv” [4] consisting 5110 unique records and 12 attributes described in Table 1. Each row includes relevant information about different individuals across many data categories.
Based on the input attributes we predict a patient’s risk of having stroke using the models we developed. The output of target class indicates 0 if no risk is detected and 1 if a patient is at high risk of stroke.

TABLE I: List of dataset attributes

Attribute	**Description**
id	unique identifier
Gender	”Male”, ”Female” or ”Other”
Age	Age of the patient
Hypertension	0 if the patient doesn’t have hypertension. 1 if the patient has hypertension
Heart disease	0 if the patient doesn’t have any heart diseases 1 if the patient has a heart disease
Marital status	”No” or ”Yes”
Work status	Children,Govt_Job, Never_worked,Private or Self-employed
Residential type	”Rural” or ”Urban”
Glucose level of each individual	Average glucose level in blood
BMI	Body mass index
Smoking status	”Formerly smoked”, ”never smoked”, ”smokes” or ”Unknown”*
Stroke	1 if the patient had a stroke or 0 if not

IV. Methodology

For our research topic, we have decided to follow the workflow mentioned below implementing the Logistic Regression, Random Forest, SMOTE, and XGBoost. Before applying the machine learning models, we visualized the data and performed EDA(Exploratory Data Analysis), performed data preprocessing which includes handling missing values, handling outliers, and data normalization, and then balanced the dataset using SMOTE.

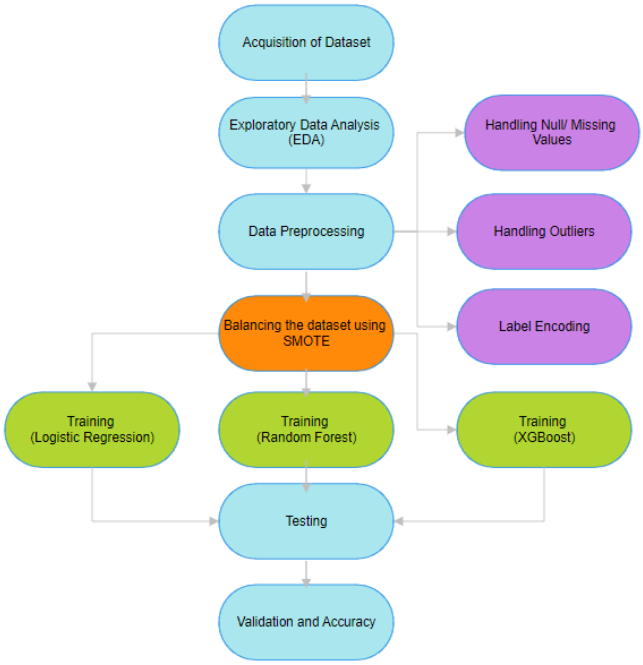

Fig. 1: Workflow diagram of Stroke Prediction.

A. Data Visualization and Preprocessing: We conducted the following for clear data visualization and handling noises:

- **Data Visualization (updated):** We have visualized the dataset using WEKA.

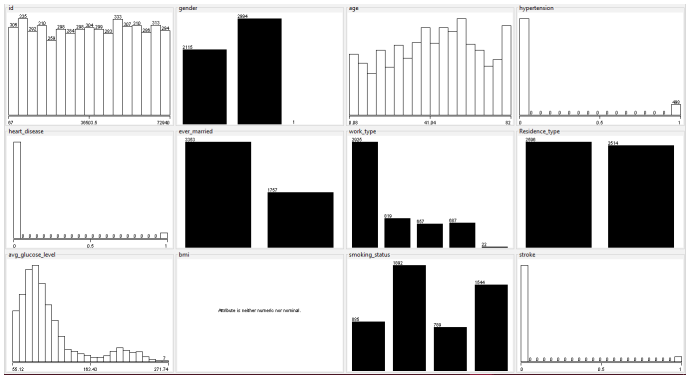

Fig. 2: Visualizing the data.

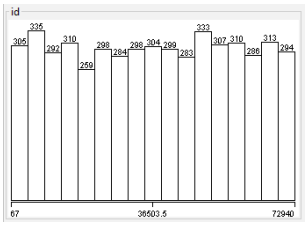

Fig. 3: ID

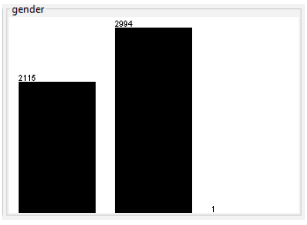

Fig. 4: Gender

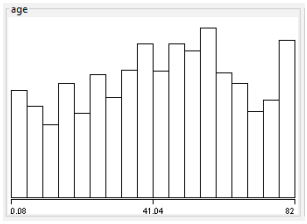

Fig. 5: Age

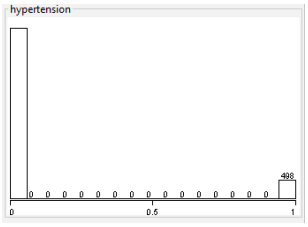

Fig. 6: Hypertension

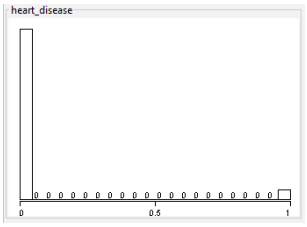

Fig. 7: Heart Disease

Fig. 8: Ever Married

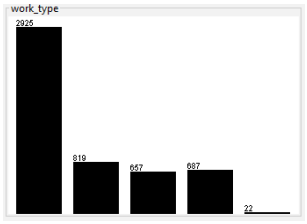

Fig. 9: Work Type

Fig. 10: Residence Type

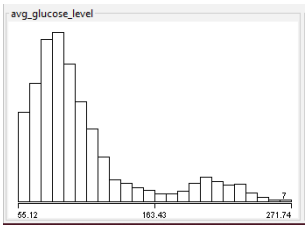

Fig. 11: Average Glucose Level

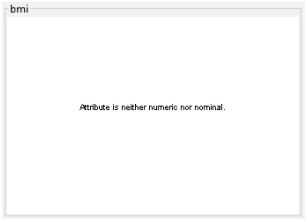

Fig. 12: BMI

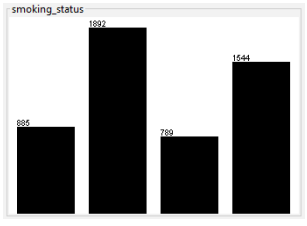

Fig. 13: Smoking Status

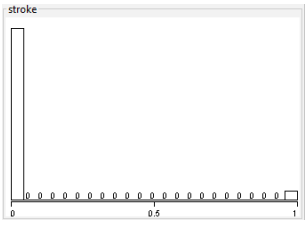

Fig. 14: Stroke

Fig. 15: Visualization of Our Dataset

- **Handling null/missing values:**

```
id                   0
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                188
smoking_status       0
stroke               0
dtype: int64
```

Fig. 16: Searching for missing values

We have 188 null values on bmi column. We have replaced these NaN values with the mean of bmi (28.75).

- **Handling Outliers (updated):**
We have detected the outliers in our dataset based on IQR (InterQuartile Range Method) with Weka. We have calculated the range between the first quartile (Q1) and the third quartile (Q3), known as the IQR. This method utilizes quartiles Q1 (25%) and Q3 (75%) along with the Interquartile Range (IQR), which is the difference between Q1 and Q3. We have found 165 outliers in our dataset. Applying filter, we then removed the outliers and obtained 4945 unique instances.

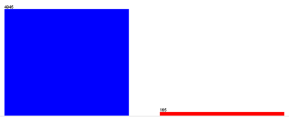

Fig. 17: **Visualizing Outliers**

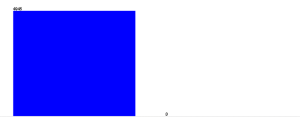

Fig. 18: **After removing Outliers**

- **Exploratory Data Analysis (EDA)(added):** We have used **Correlation** matrix and **Heatmap** function to identify the relevance of all 12 attributes with stroke and sorted the attributes according to their relevance. For this purpose, at first we have mapped the categorical data into numeric data using **label encoding**.Label encoding is commonly used when dealing with algorithms that require numerical input.

1. gender -> Female: 0, Male: 1
2. ever_married -> Yes: 1, No: 0
3. work_type -> Govt_job: 0, children: 1, Private: 2, Self_employed: 3
4. Residence_type -> Rural: 0, Urban: 1
5. smoking_status -> formerly smoked: 0, never smoked: 1, Unknown: 2, smokes: 3

Fig. 19: Corresponding mapping of categorical data to numeric values

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	51676	0	61.0	0	0	1	3	0	202.21	28.751104	1	1
1	31112	1	80.0	0	1	1	2	0	105.92	32.500000	1	1
2	60182	0	49.0	0	0	1	2	1	171.23	34.400000	3	1
3	1665	0	79.0	1	0	1	3	0	174.12	24.000000	1	1
4	56669	1	81.0	0	0	1	2	1	186.21	29.000000	0	1
5	53882	1	74.0	1	1	1	2	0	70.09	27.400000	1	1
6	10434	0	69.0	0	0	0	2	1	94.39	22.800000	1	1
7	27419	0	59.0	0	0	1	2	0	76.15	28.751104	2	1
8	60491	0	78.0	0	0	1	2	1	58.57	24.200000	2	1
9	12109	0	81.0	1	0	1	2	0	80.43	29.700000	1	1

Fig. 20: Dataset after Label Encoding.

After label encoding, we have used **correlation** function and showed the correlation using heatmap function.

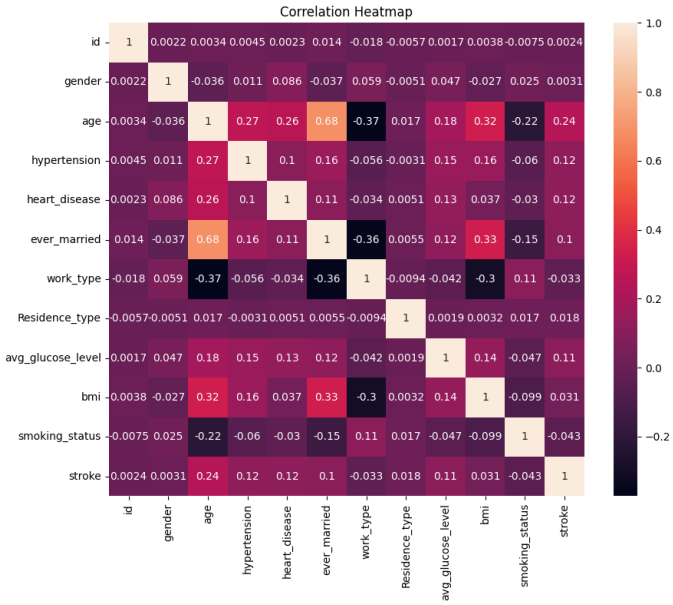

Fig. 21: Correlation of attributes

Then we have sorted attributes according to correlation values with target (stroke).

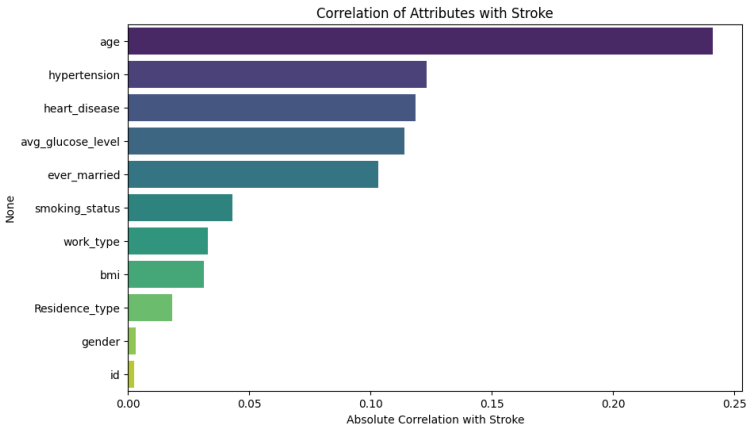

Fig. 22: Attributes sorted according to their correlation with 'stroke'

As 'id' is the least correlated attribute, we have dropped this irrelevant column.

B. Logistic Regression: Logistic regression, a supervised classification model often referred to as the logit model, predicts the probability of an event occurring based on a set of independent variables. This event outcome typically takes on categorical or discrete values, such as 'stroke' or 'no stroke'. Rather than providing exact 0s and 1s, the model generates probabilities ranging from 0 to 1. Unlike linear regression, which fits a straight line, logistic regression employs a logistic function, commonly known as the Sigmoid function (S-curve), to capture the relationship between the predictors and the outcome.[2] It consists of two following functions:

- **Logistic function (Sigmoid function):** Since logistic regression is a binary classification technique, the values predicted should fall close to either 0 or 1. This is why a sigmoid function is convenient. In mathematical terms:

$$p\left(x\right) = \frac{1}{1+e^{-z}} \tag{1}$$

Where: p(x) is the predicted probability that the output for a x is equal to 1. z is the linear function since logistic regression is a linear classifier which translates to:

$$z = b_0 + b_1 x_1 + \ldots + b_r x_r \tag{2}$$

Where: b_0,b_1,...,b_r are the model's predicted weights or coefficients. x is the feature values.
z can be represented the logarithm of the probability of an event occurring (denoted as p(x)=1, such as 'stroke') divided by the probability of the event not occurring (denoted as 1-p(x)=0, such as 'no stroke'). Hence, z is commonly known as the log-odds or the natural logarithm of odds.
The odds mean the probability of success over the probability of failure.

$$\log\left(\frac{p\left(x\right)}{1-p\left(x\right)}\right) \tag{3}$$

Logistic regression uses a threshold value to classify predicted probabilities as either 0 or 1. For instance, with a threshold of 0.5, probabilities above it are classified as 1, and those below as 0.

- **Maximum Likelihood Estimation (MLE):** Maximizing the log-likelihood function (LLF) enables us to obtain the optimal coefficients or predicted weights in logistic regression. This process entails identifying the sigmoid curve that best fits the data, achieved through Maximum Likelihood Estimation.

$$LLF = \sum \left(y_i \log\left(p\left(x_i\right)\right) + \left(1-y_i\right) \log\left(1-p\left(x_i\right)\right)\right) \tag{4}$$

C. Random Forest: Random Forest stands as a widely embraced machine learning algorithm within the realm of

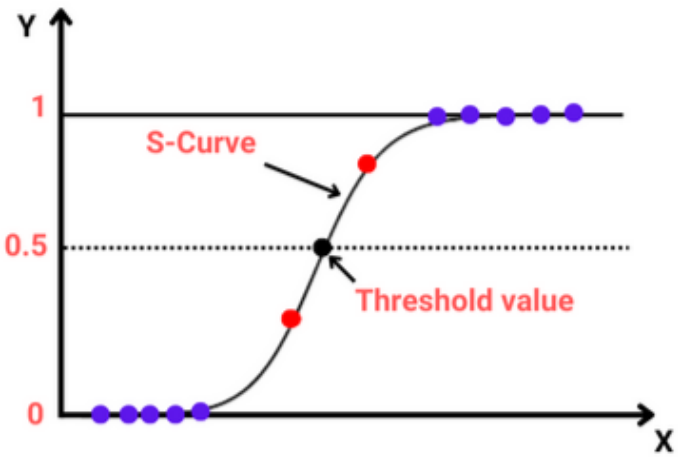

Fig. 23: S curve with threshold for predicting stroke or no stroke.

supervised learning. This versatile tool finds application in tackling both classification and regression challenges in the field of ML. It owes its efficacy to the principle of ensemble learning; wherein multiple classifiers collaborate to address intricate problems and enhance model performance.

In our research work, we have used random forest which follows ensemble of multiple decision trees. Increasing the number of trees in the forest is pivotal for enhancing accuracy and mitigating overfitting concerns.

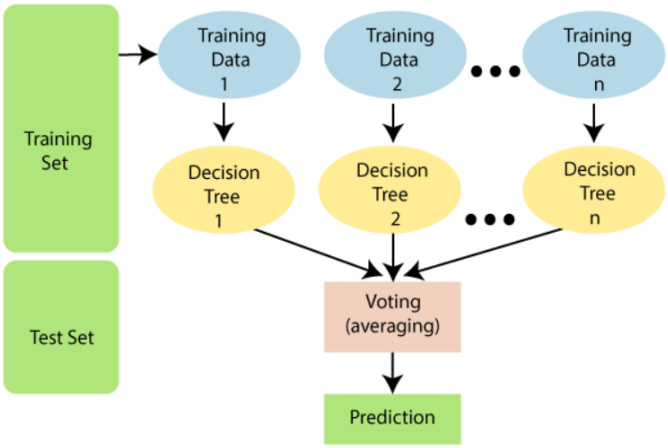

Fig. 24: Steps of Random Forest Classifier

Steps Involved in Random Forest Algorithm

We have followed the following steps in our work to build random forest model for stroke prediction.

- Step 1: Randomly select K data points from the training set.
- Step 2: Construct decision trees based on the selected data points (subsets).
- Step 3: Determine the desired number N of decision trees to build.
- Step 4: Iterate through Steps 1 and 2.
- Step 5: When encountering new data points, predict their outcomes using each decision tree, and classify them into the category with the most votes.

We have used the following hyperparameters to tune our random forest model:

- **n-estimators:** Number of trees the algorithm builds before averaging the predictions.
- **n-jobs:** it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor, but if the value is -1, there is no limit.

D. SMOTE: Imbalanced datasets present a common hurdle for machine learning professionals, particularly in binary classification tasks. This situation is prevalent in real-world scenarios. Machine learning algorithms struggle to effectively learn when there is a significant imbalance between the instances of different classes. To combat this challenge, one widely used technique is the Synthetic Minority Oversampling Technique (SMOTE). SMOTE is tailored to address imbalanced datasets by creating synthetic samples for the minority class.

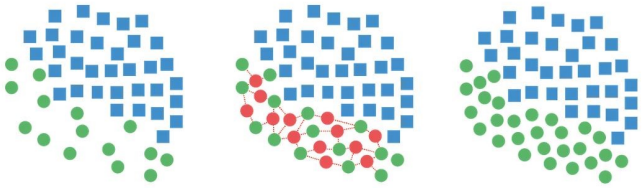

Fig. 25: (a) Original Dataset (b) Generating Samples (c) Resampled Dataset.

SMOTE (Synthetic Minority Over-sampling Technique) is a method used in machine learning to address class imbalance by generating synthetic samples of the minority class. It works by selecting individual instances from the minority class and creating artificial samples that are similar to the existing minority class instances. This technique helps to balance the class distribution in the dataset, which can improve the performance of machine learning models, particularly in scenarios where the minority class is of interest but underrepresented in the data. SMOTE operates by generating artificial samples for the minority class through interpolation among existing minority class instances.

The process involves selecting a minority class instance and then choosing one or more of its nearest minority class neighbors. Subsequently, a new instance is formed by randomly selecting one neighbor and calculating the difference between the feature values of the two instances. This difference is then scaled by a random number between 0 and 1 and added to the feature values of the original minority instance, resulting in a fresh synthetic instance. By iterating through this procedure for several minority class instances, SMOTE constructs a broader

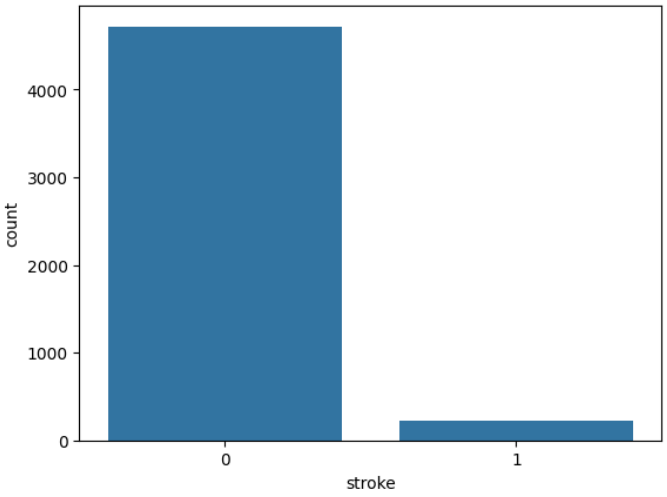

Fig. 26: Class Imbalanced Data.

and more varied set of synthetic minority class instances that can be merged with the original dataset. This leads to a more balanced dataset, which can enhance machine learning model performance by mitigating bias towards the majority class. However, a potential drawback of SMOTE is its tendency to generate synthetic instances closely resembling existing minority class instances, possibly causing overfitting. To address this concern, researchers have introduced variations of SMOTE, such as Borderline-SMOTE, which focuses solely on creating new instances near the boundary separating the minority and majority classes [9].

The synthetic samples are created in a way that preserves the underlying characteristics of the minority class, thereby reducing the risk of overfitting. This technique helps classifiers to better learn from the minority class, improving the overall performance of the model.

The `SMOTE()` function from the `smotefamily` package

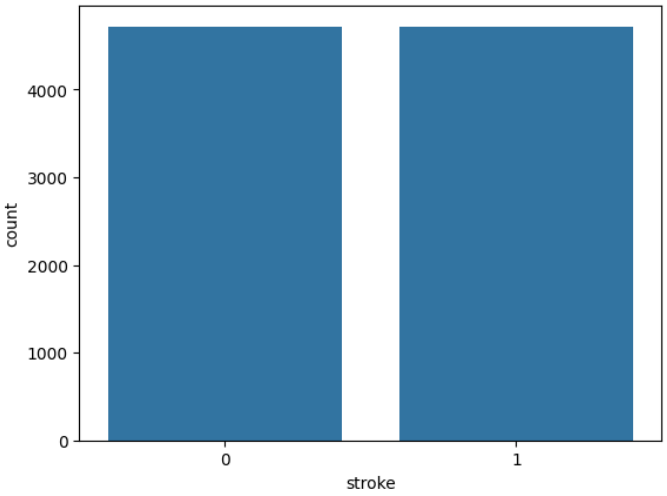

Fig. 27: Balanced data using SMOTE.

requires two parameters: K and *dup_size*. When synthesizing new instances, `SMOTE()` considers neighboring instances from the perspective of existing minority instances. The neighbors considered for each existing minority instance depend on the value of K.

When $K = 1$, only the closest neighbor of the same class is taken into account. For example, if we focus on the bottom right red data point, its closest neighbor is the top left flower. Thus, at $K = 1$, `SMOTE()` generates a new minority instance along the line connecting these two points.

At $K = 2$, both the closest and the second closest neighbors are considered. For each synthesis, one neighbor is randomly chosen between them.

In essence, `SMOTE()` iterates through the existing minority instances. At each iteration, it selects one of the K closest minority class neighbors and synthesizes a new minority instance somewhere along the line between the original instance and that neighbor.

The *dup_size* parameter determines how many times `SMOTE()` iterates through the existing minority instances. For instance, at *dup_size* = 1, only four new data points are generated (one for each existing minority instance). Conversely, at *dup_size* = 200, 800 new data points are synthesized.

Additionally, *dup_size* = 0 is a unique case where `SMOTE()` optimally adjusts the number of minority class reuses to achieve balance with the majority class.

E. XGBoost: XGBoost, short for Extreme Gradient Boosting, is a powerful algorithm designed to enhance predictive accuracy by iteratively refining stroke risk assessment by boosting weak learners. Ever since its introduction, XGBoost has earned praise as the go-to solution for machine learning problems [10]. XGBoost is a machine learning algorithm categorized under ensemble learning, specifically within the gradient boosting framework. It utilizes decision trees as its base learners and incorporates regularization techniques to improve model generalization. Renowned for its computational efficiency, feature importance analysis, and handling of missing values, XGBoost finds extensive application in tasks like regression, classification, and ranking. The algorithm operates by progressively integrating weak learners into the ensemble, where each newcomer aims to rectify the mistakes of its predecessors. It employs gradient descent optimization to minimize a predefined loss function throughout the training process.

XGBoost belongs to the ensemble learning family, which addresses the limitation of relying solely on the outcomes of a single machine learning model. Ensemble learning provides a systematic approach to combine the predictive capabilities of multiple learners, resulting in a single model that aggregates the outputs of several models. In ensemble learning, the models forming the ensemble, termed as base learners, can originate from the same or different learning algorithms. Bagging and boosting are two popular ensemble learning techniques. While these methods can be applied to various statistical models, they are most commonly associated with

decision trees.
Given below is a deeper examination of bagging and boosting.

- **Bagging:** Bagging, also known as bootstrap aggregation, is utilized to address the inherent variability in decision trees, despite their interpretability. This variability becomes apparent when we split a single training dataset into two parts and train a decision tree on each part, resulting in different outcomes. Decision trees are characterized by high variance due to this behavior. To mitigate this variance, bagging aggregates multiple decision trees generated in parallel.

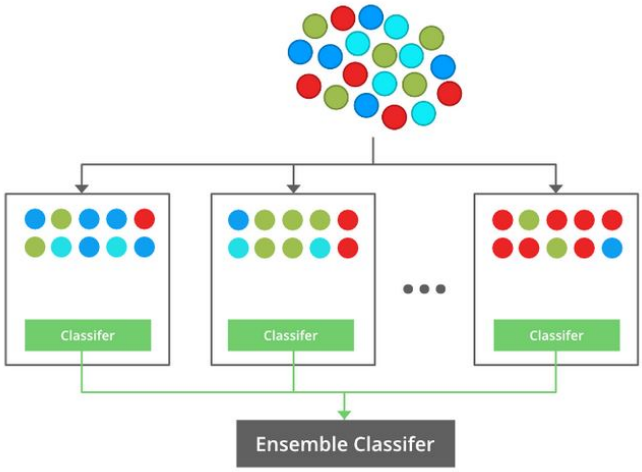

Fig. 28: Bagging procedure.

- **Boosting:** Boosting involves constructing trees sequentially, with each subsequent tree aiming to reduce the errors of its predecessor. As each tree learns from its previous iterations, it updates the residual errors, allowing the next tree in the sequence to learn from an improved version of the residuals. In boosting, the base learners are weak, with high bias and only slightly better predictive power than random guessing. Despite their weaknesses, these learners contribute essential information for prediction. By effectively combining these weak learners, boosting generates a strong learner that reduces both bias and variance. Unlike bagging techniques such as Random Forest, where trees are grown to their maximum depth, boosting employs trees with fewer splits. These smaller, less complex trees are highly interpretable. Parameters like the number of trees, learning rate, and tree depth can be optimized through validation techniques like k-fold cross-validation. It's important to carefully select stopping criteria for boosting to prevent overfitting, especially when using a large number of trees.

The gradient boosting ensemble technique comprises three straightforward steps:

1) Initially, a model F_0 is established to predict the target variable y. This model is associated with the residuals $(y - F_0)$.

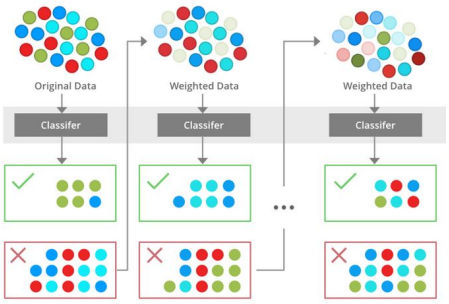

Fig. 29: Boosting procedure.

2) Subsequently, a new model h_1 is trained using the residuals from the previous step.
3) Finally, F_0 and h_1 are combined to produce F_1, the boosted version of F_0. The mean squared error from F_1 will be lower than that from F_0.
$$F_1(x) \leftarrow F_0(x) + h_1(x)$$
4) To improve the performance of F_1, we could model after the residuals of F_1 and create a new model F_2:
$$F_2(x) \leftarrow F_1(x) + h_2(x)$$
5) This can be done for m iterations, until residuals have been minimized as much as possible:
$$F_m(x) \leftarrow F_{m-1}(x) + h_m(x)$$

In this context, the additional learners don't disrupt the functions established in prior steps. Instead, they contribute their own insights to reduce errors.

V. Results and Analysis

(updated)

A. Logistic Regression

1) 5 Fold Cross Validation

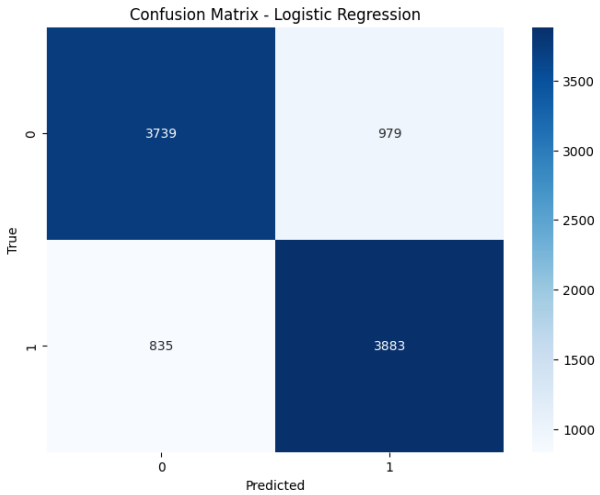

Fig. 30: Confusion Matrix of Logistic Regression with 5 fold Cross Validation

Performance Measures:
Accuracy: 0.807757524374735
Precision: 0.7986425339366516
Recall: 0.8230182280627385
F1 Score: 0.8106471816283926

2) **10 Fold Cross Validation:**

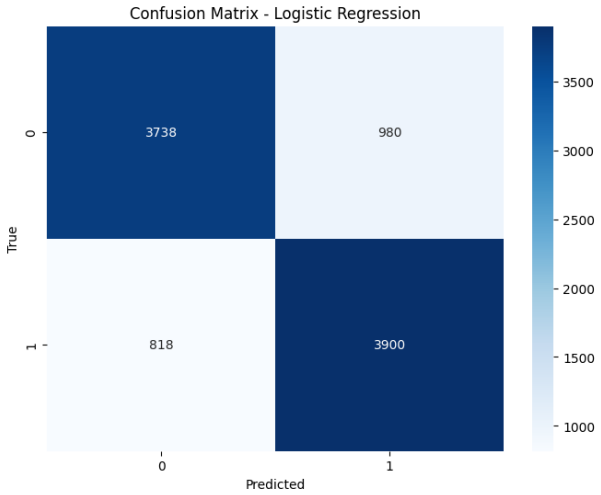

Fig. 31: Confusion Matrix of Logistic Regression with 10 fold Cross Validation

Performance Measures:
Accuracy: 0.8094531581178466
Precision: 0.7991803278688525
Recall: 0.8266214497668504
F1 Score: 0.8126693061054386

B. Random Forest

1) **5 Fold Cross Validation**

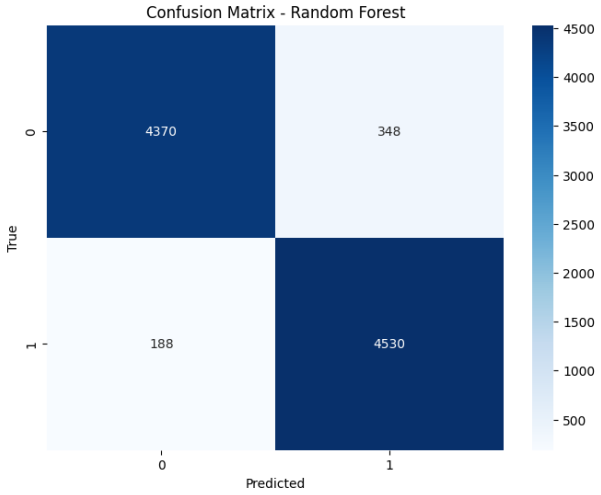

Fig. 32: Confusion Matrix of Random Forest with 5 fold Cross Validation

Performance Measures:
Accuracy: 0.9431962696057652
Precision: 0.9286592865928659
Recall: 0.96015260703688
F1 Score: 0.9441433930804503

2) **10 Fold Cross Validation:**

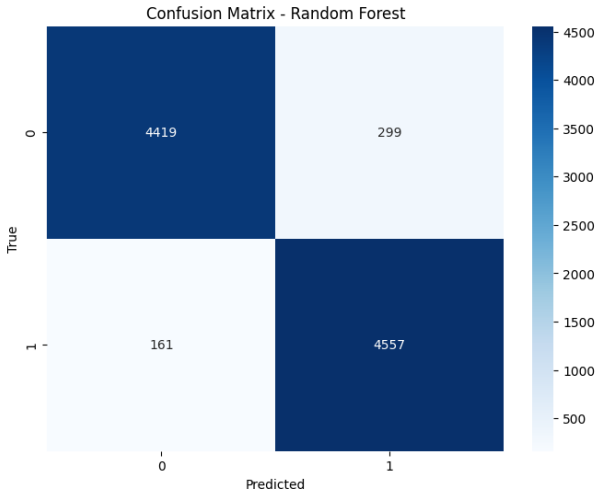

Fig. 33: Confusion Matrix of Random Forest with 10 fold Cross Validation

Performance Measures:
Accuracy: 0.9512505298855447
Precision: 0.9384266886326195
Recall: 0.9658753709198813
F1 Score: 0.9519532066012116

C. XGBoost

1) **5 Fold Cross Validation**

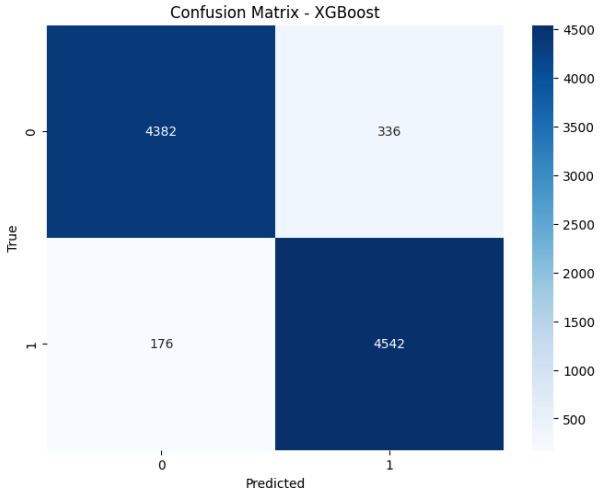

Fig. 34: Confusion Matrix of XGBoost with 5 fold Cross Validation

Performance Measures:
Accuracy: 0.9457397202204324
Precision: 0.931119311193112
Recall: 0.9626960576515473
F1 Score: 0.9466444351813256

2) **10 Fold Cross Validation:**

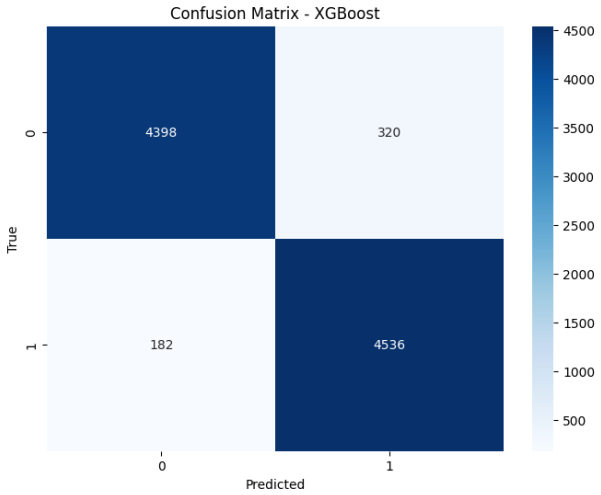

Fig. 35: Confusion Matrix of XGBoost with 10 fold Cross Validation

Performance Measures:
Accuracy: 0.9467994913098771
Precision: 0.9341021416803954
Recall: 0.9614243323442137
F1 Score: 0.9475663254648005

TABLE II: Comparison among 3 algorithms for 5 fold CV

Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.8077	0.7986	0.8230	0.8106
Random Forest	0.9431	0.9286	0.9601	0.9441
XGBoost	0.9457	0.9311	0.9626	0.9466

TABLE III: Comparison among 3 algorithms for 10 fold CV

Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.8094	0.7991	0.8266	0.8126
Random Forest	0.9512	0.9384	0.9658	0.9519
XGBoost	0.9467	0.9341	0.96142	0.9475

Based on their accuracy Random Forest and XGBoost perform impressively well in predicting outcomes. Random Forest and XGBoost demonstrate superior overall predictive accuracy compared to Logistic Regression.
In 5 fold Cross Validation, XGBoost performs better (94.57%) than Random Forest (94.31%) comparing their accuracy while in 10 Fold Cross Validation, Random Forest shows higher accuracy (95.12%) than XGBoost (94.67%). Again we can see much improvement in performance doing 10 fold cross validation than 5 fold cross validation.
Logistic Regression, while achieving lower accuracy and performance metrics compared to Random Forest and XGBoost, may still be suitable for scenarios where simplicity and interpretability are prioritized over predictive accuracy.
Logistic regression gives reliability for simpler datasets and random forest is more effective for handling complex relationships and noisy data. Whereas XGBoost excels particularly in capturing intricate patterns within the data. Overall, each algorithm presents distinct advantages, among which random forest gives the highest accuracy.

VI. Conclusion

In this study, we conducted a comprehensive analysis of machine learning algorithms for brain stroke prediction, employing Logistic Regression, Random Forest, and XGBoost, with the incorporation of SMOTE for handling class imbalance. Our findings provide valuable insights into the effectiveness of these algorithms in predicting stroke occurrence.

Random Forest and XGBoost emerged as the top-performing algorithms, achieving impressive accuracies, outperforming Logistic Regression, which attained an accuracy of 80%. The superior performance of Random Forest can be attributed to its ability to handle complex relationships and noisy data effectively. XGBoost, on the other hand, excelled in capturing intricate patterns within the data.

While Logistic Regression demonstrated lower accuracy and performance metrics compared to Random Forest and XGBoost, it still holds relevance in scenarios prioritizing simplicity and interpretability over predictive accuracy.

Our study underscores the importance of data preprocessing and balancing techniques like SMOTE in improving the robustness and reliability of predictive models. By addressing missing values, outliers, and class imbalance, we ensure more accurate predictions and enhance the utility of the models in real-world applications.

For future research, we aim to extend our analysis by incorporating additional data modalities such as images and CT scans, along with the existing tabular data. Integration of diverse data sources could provide richer insights into the risk factors and predictors associated with stroke occurrence, potentially leading to more accurate and personalized predictive models. We also aim to handle the missing data values using deep learning techniques like TabNet or MICE, instead of simply removing them from our dataset.

Overall, our study highlights the efficacy of machine learning algorithms in stroke prediction and sets the stage for further investigations leveraging multi-modal data integration to enhance predictive accuracy and clinical utility.

References

[1] A. K. Kadam, P. Agarwal, Nishtha, and M. Khandelwal, "Brain Stroke Prediction Using Machine Learning Approach," IRE Journals, vol. 6, no. 1, Jul. 2022, pp. 273, ISSN: 2456-8880.

[2] L. Wang, “Logistic Regression for Stroke Prediction: An Evaluation of its Accuracy and Validity”, HSET, vol. 39, pp. 1086–1092, Apr. 2023, doi: 10.54097/hset.v39i.6712.

[3] M. U. Emon, M. S. Keya, T. I. Meghla, M. M. Rahman, M. S. A. Mamun and M. S. Kaiser, ”Performance Analysis of Machine Learning Approaches in Stroke Prediction,” 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 1464-1469, doi: 10.1109/ICECA49313.2020.9297525.

[4] Dataset from Kaggle: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

[5] Rana, S. O. H. E. L., H. A. B. S. H. A. H. Midi, and S. K. Sarkar. ”Validation and performance analysis of binary logistic regression model.” Proceedings of the WSEAS International Conference on Environmental, Medicine and Health Sciences. WSEAS Press, 2010.

[6] Elreedy, D., and Atiya, A. F. (2019). A Comprehensive Analysis of Synthetic Minority Oversampling TEchnique (SMOTE) for Handling Class Imbalance. Information Sciences. DOI: 10.1016/j.ins.2019.07.070

[7] R. Primartha and B. A. Tama, ”Anomaly detection using random forest: A performance revisited,” 2017 International Conference on Data and Software Engineering (ICoDSE), Palembang, Indonesia, 2017, pp. 1-6, doi: 10.1109/ICODSE.2017.8285847.

[8] Ramraj, Santhanam, et al. ”Experimenting XGBoost algorithm for prediction and classification of different datasets.” International Journal of Control Theory and Applications 9.40 (2016): 651-662.

[9] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao, “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning,” Advances in Intelligent Computing, Sept. 2005, pp. 878–887.

[10] Tianqi Chen and Carlos Guestrin, “XGBoost: A Scalable Tree Boosting System,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016

[11] Soumyabrata Dev, Hewei Wang, Chidozie Shamrock Nwosu, Nishtha Jain, Bharadwaj Veeravalli, Deepu John, “A predictive analytics approach for stroke prediction using machine learning and neural networks”, Healthcare Analytics, Volume 2, 2022, 100032, ISSN 2772-4425, https://doi.org/10.1016/j.health.2022.100032.

[12] Wu, Yafei, and Ya Fang. 2020. ”Stroke Prediction with Machine Learning Methods among Older Chinese” International Journal of Environmental Research and Public Health 17, no. 6: 1828. https://doi.org/10.3390/ijerph17061828

[13] Tazin, Tahia, Md. Nur Alam, Nahian Nakiba Dola, Mohammad Sajibul Bari, Sami Bourouis and Mohammad Monirujjaman Khan. “Stroke Disease Detection and Prediction Using Robust Learning Approaches.” Journal of Healthcare Engineering 2021 (2021): n. pag.

[14] Ferdib-Al-Islam and M. Ghosh, ”An Enhanced Stroke Prediction Scheme Using SMOTE and Machine Learning Techniques,” 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 1-6, doi: 10.1109/ICCCNT51525.2021.9579648.

[15] Sailasya, G. and Kumari, G.L.A., 2021. Analyzing the performance of stroke prediction using ML classification algorithms. International Journal of Advanced Computer Science and Applications, 12(6).

[16] Islam, M.M., Akter, S., Rokunojjaman, M., Rony, J.H., Amin, A. and Kar, S., 2021. Stroke prediction analysis using machine learning classifiers and feature technique. International Journal of Electronics and Communications Systems, 1(2), pp.17-22.

[17] Chen, S.D., You, J., Yang, X.M., Gu, H.Q., Huang, X.Y., Liu, H., Feng, J.F., Jiang, Y. and Wang, Y.J., 2022. Machine learning is an effective method to predict the 90-day prognosis of patients with transient ischemic attack and minor stroke. BMC medical research methodology, 22(1), p.195.

[18] Gupta, S. and Raheja, S., 2022, January. Stroke prediction using machine learning methods. In 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 553-558). IEEE.

[19] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, pp.321-357.