Machine Learning, I DATS 6202

Individual Report

Stroke Prediction

Instructor - Amir Jafari

Presented by Group 2:

Shikha Sharma

Data- 05/01/2023

Table of Contents

1 Introduction

Stroke, a critical global health concern, stands as the second most common cause of death and the third most common cause of disability worldwide, according to the World Health Organization (WHO). Every year, approximately 800,000 individuals in the United States alone experience a stroke, with nearly 75% of these cases being first-time occurrences. Early detection and prevention are crucial, as 80% of strokes can be averted with timely intervention and proper education on risk factors and warning signs.
The primary objective of our project is to leverage the capabilities of machine learning (ML) techniques for the prediction of strokes, an area that has received comparatively less attention than the prediction of heart attacks. By analyzing risk factors such as age, systolic blood pressure, BMI, cholesterol, diabetes, smoking status and intensity, physical activity, alcohol consumption, and family history, we aim to develop a model capable of accurately predicting stroke risk and informing preventive measures.
This project presents the implementation of eight ML classification methods in the context of stroke prediction, contributing to the growing body of knowledge in this critical area of healthcare. Our findings have the potential to enhance early intervention strategies, promote healthier lifestyles, and ultimately save lives by reducing the impact of strokes on individuals and communities worldwide.

2 Description of Individual Work

As a member of the team, I was responsible for several crucial tasks and functions in the project. My main contributions are summarized as follows:

1. Data Cleaning and Preprocessing: I participated in the data cleaning and preprocessing stage, where I worked on preparing the data for the machine learning models.
2. Handling Imbalanced Data: I implemented the SMOTE technique to handle imbalanced data in the dataset, which helped to balance the class distribution.
3. Label Encoding Function: I developed a function to handle missing values in categorical features and performed Label encoding on the same.
4. Exploratory Data Analysis: I developed a function to plot Exploratory Data Analysis (EDA) and visualize the relationship between the features and the target variable.
5. Machine Learning Models: I worked on developing machine learning models on XGBoost Classification, Gradient Boosting, and Random Forest.
6. Modularizing the Code: I was responsible for modularizing the code to make it more organized and easier to maintain.
7. GitHub Repository: I created the GitHub repository and folder structure and was responsible for resolving any issues that my teammates faced.
8. Model Evaluation: I computed several performance metrics, such as accuracy, ROC-AUC, precision, recall, and F1-score, for each model to evaluate their performance.

3 Preprocessing

Before building a predictive model, the dataset was preprocessed to ensure its accuracy and efficiency. The preprocessing stage involved cleaning and preparing the data for model development. Missing values were filled, and string literals were converted to integer values through label encoding.

Before building a predictive model, it is crucial to preprocess the dataset to ensure the model's accuracy and efficiency. The preprocessing stage involves cleaning and preparing the data for model development. In this case, the dataset used has twelve characteristics, and the column ‘id’ is omitted as it does not contribute to the model's construction. Next, the dataset is checked for missing values and any detected missing values are filled. For example, in this case, the missing values in the column ‘BMI’ are filled with the mean of the column. After handling missing values, the next step is to convert the string literals in the dataset to integer values that can be understood by the computer. This process is called label encoding, and in this case, the dataset has five columns with a string data type. All string values are encoded, and the entire dataset is transformed into numerical values. The dataset used for stroke prediction is imbalanced, with 5110 rows in total, 249 rows indicating the possibility of a stroke, and 4861 rows confirming the absence of a stroke. Training a machine-learning model using such an imbalanced dataset can result in high accuracy, but other accuracy measures such as precision and recall may be lacking. To ensure the model's effectiveness, it is necessary to address the imbalance in the data. The SMOTE (Synthetic Minority Over-sampling Technique) is used for this purpose.

4 Model Algorithms

The study evaluated the performance of several commonly used machine learning methods including Random Forest, XGBoost, and Gradient Boosting. These algorithms were selected based on their popularity and effectiveness in solving similar problems in the medical field. The accuracy, precision, and recall of the model are then evaluated using the testing data to determine the most effective algorithm.

1) Random Forest

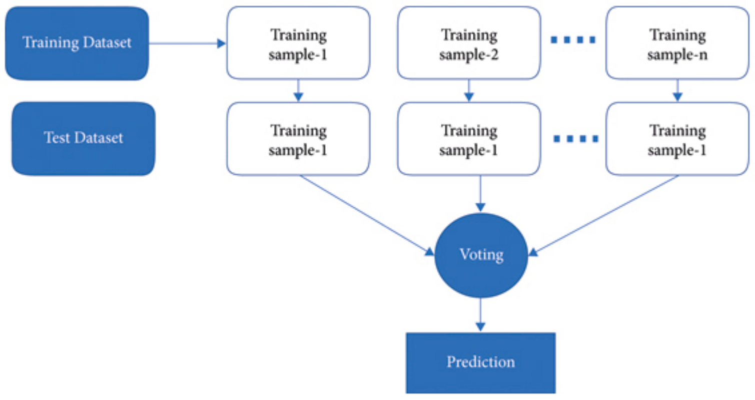
2) Gradient Boosting

3) XGBoost

4.1 Random Forest

The chosen classification algorithm for this project is the Random Forest (RF) classifier. Random Forests are ensemble learning methods that consist of numerous independent decision trees, each trained on a random subset of the data. The core idea behind this approach is to create a "forest" of decision trees that work together to produce more accurate and stable predictions than any single

decision tree could achieve on its own. During the training phase, each decision tree is constructed by selecting a random sample of the data and applying a learning algorithm to generate a tree structure. This random sampling introduces diversity in the decision trees, which helps reduce overfitting and improve the generalization ability of the model. Once the decision trees are trained, the RF classifier combines their outputs through a process called "voting" to arrive at the final prediction. In this specific case, each decision tree must vote for one of the two output classes (stroke or no stroke). The class that receives many votes from the decision trees is then chosen as the final prediction.
The strength of the RF classifier lies in its ability to leverage the collective knowledge of multiple decision trees, effectively reducing the impact of individual tree weaknesses, such as biases or overfitting. This results in a more robust and accurate model that can better handle the complexity and nuances of the data, ultimately leading to improved stroke prediction performance. A block diagram of random forest classification is shown in Figure.



The versatility of the Random Forest is one of its most appealing characteristics. It can be employed for both regression and classification tasks, and the relative importance of input features is easily discernible. Additionally, this method is advantageous because the default hyperparameters often yield clear predictions.
Comprehending the hyperparameters is crucial since there are only a few to consider in the first place. Overfitting is a common issue in machine learning; however, it is rarely a problem with the Random Forest classifier. If there is an adequate number of trees in the forest, the classifier is unlikely to overfit the model.
By constructing a diverse ensemble of decision trees and aggregating their predictions, the Random Forest classifier can achieve high performance and stability across a wide range of tasks. This adaptability, combined with its inherent resistance to overfitting and ease of interpretation, makes the Random Forest classifier a powerful and popular choice in the field of machine learning.

4.2 Gradient Boosting Classifier

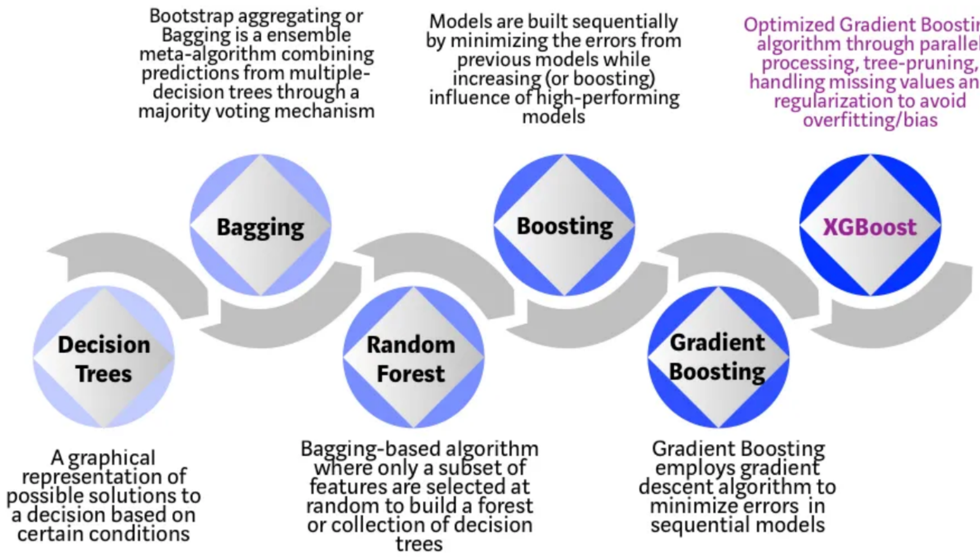
Gradient boosting is one of the variants of ensemble methods where you create multiple weak models and combine them to get better performance.

The gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As we know that the errors in machine learning algorithms are broadly classified into two categories i.e., Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize the bias error of the model.
Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is Mean Square Error (MSE) and when it is used as a classifier then the cost function is Log loss.

Gradient boosting is a highly robust technique for developing predictive models. It applies to several risk functions and optimizes the accuracy of the model's prediction. It also resolves multicollinearity problems where the correlations among the predictor variables are high.

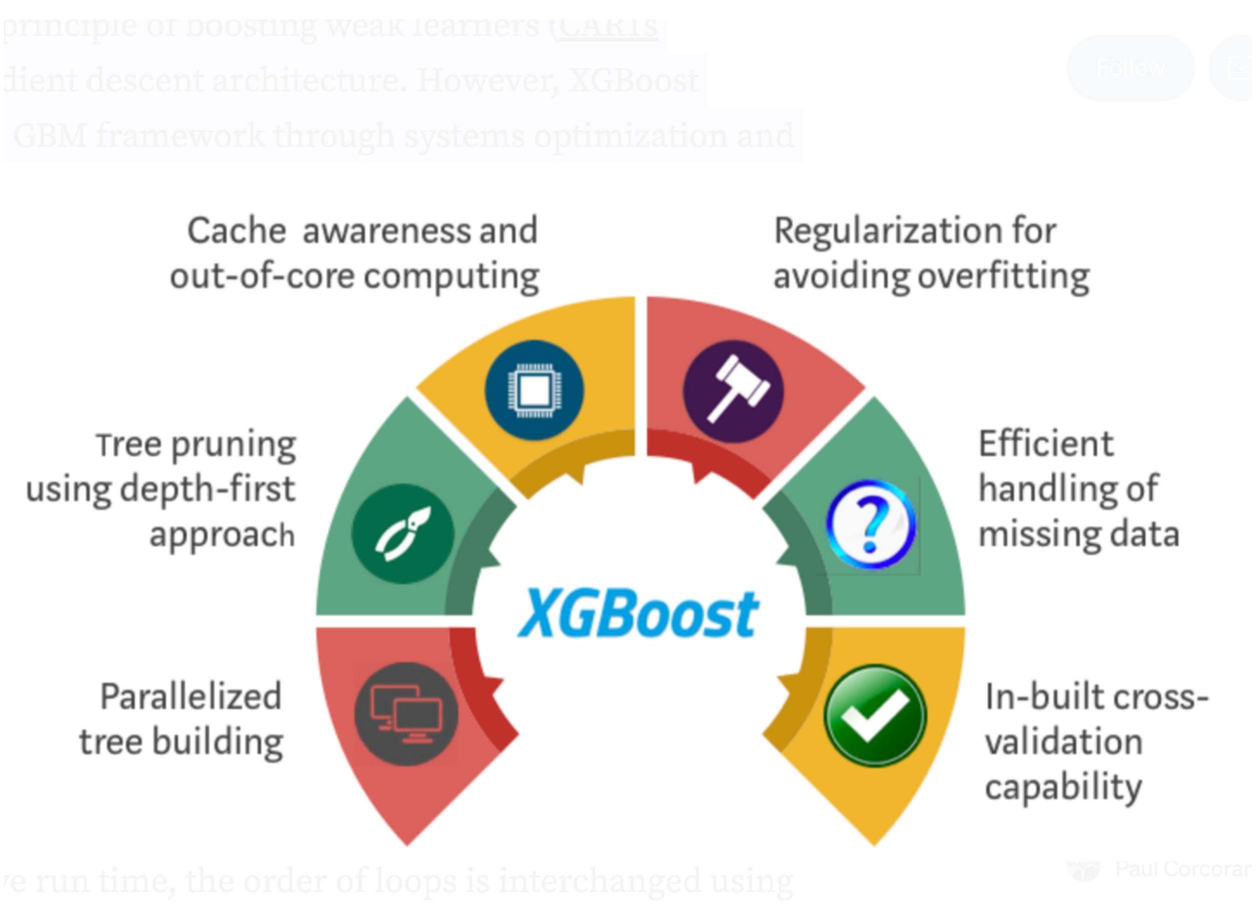
4.3 XGBOOST

XGBoost algorithm is an extended version of the gradient boosting algorithm. It is basically designed to enhance the performance and speed of a Machine Learning model.
XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient-boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now.


Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Each step of the evolution of tree-based algorithms can be viewed as a version of the interview process.

Decision Tree: Every hiring manager has a set of criteria such as education level, number of years of experience, and interview performance. A decision tree is analogous to a hiring manager interviewing candidates based on his or her own criteria.
Bagging: Now imagine instead of a single interviewer, now there is an interview panel where each interviewer has a vote. Bagging or bootstrap aggregating involves combining inputs from all interviewers for the final decision through a democratic voting process.
Random Forest: It is a bagging-based algorithm with a key difference wherein only a subset of features is selected at random. In other words, every interviewer will only test the interviewee on certain randomly selected qualifications (e.g., a technical interview for testing programming skills and a behavioral interview for evaluating non-technical skills).
Boosting: This is an alternative approach where each interviewer alters the evaluation criteria based on feedback from the previous interviewer. This 'boosts' the efficiency of the interview process by deploying a more dynamic evaluation process.


principle of boosting weak learners (CARTs
dient descent architecture. However, XGBoost
GBM framework through systems optimization and
Follow
Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability
e run time, the order of loops is interchanged using
Paul Corcoran

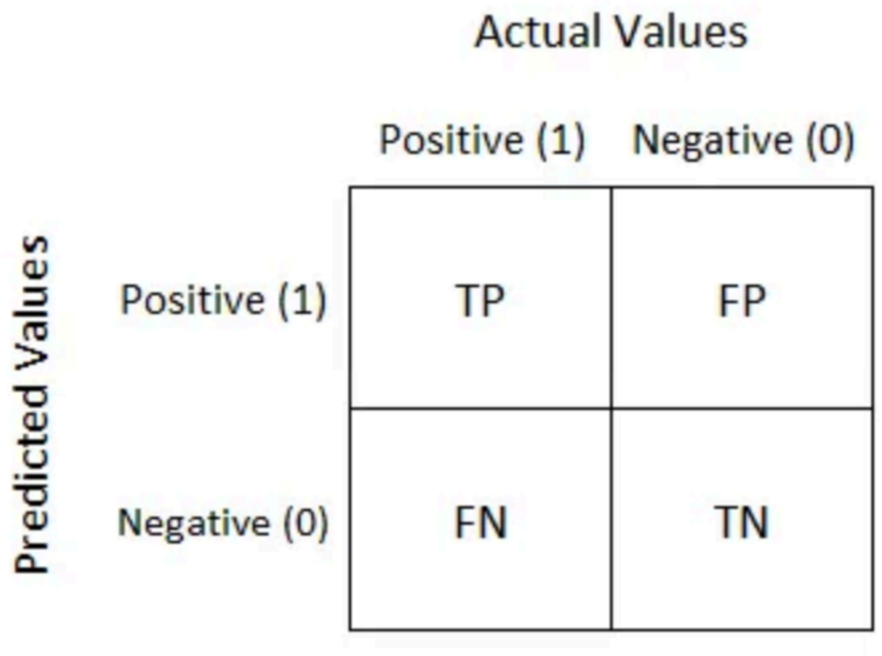
Evaluation Metrics

To assess its performance on test data, every Machine Learning model uses metrics. Metrics are like Loss Functions; except they are used to monitor test data. Accuracy, Binary Accuracy,

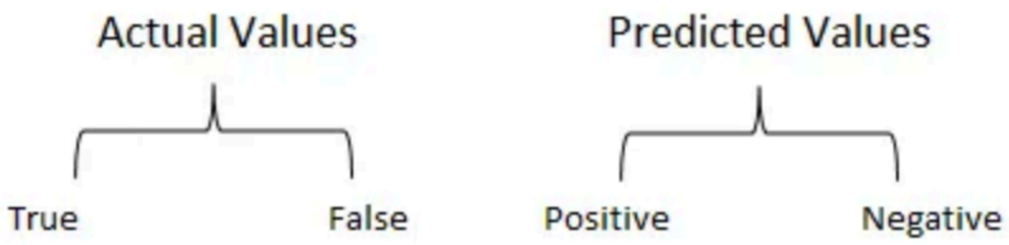
Categorical Accuracy, and other forms of accuracy measures exist. Probabilistic measures such as binary cross-entropy, categorical cross-entropy, and others are also provided.
Different kinds of evaluation metrics available are as follows.
Classification Accuracy, Confusion Matrix, The area under Curve and ROC, F1 Score, Recall

Confusion Matrix

The performance of machine learning algorithms is typically evaluated by a confusion matrix as illustrated in Figure 1 (for a 2-class problem).



It is extremely useful for measuring Recall, Precision, Specificity, Accuracy, and most importantly AUC-ROC curves.
True Positive:
Interpretation: You predicted positive and it’s true.
True Negative:
Interpretation: You predicted negative and it’s true.
False Positive: (Type 1 Error)
Interpretation: You predicted positive and it’s false.
False Negative: (Type 2 Error)
Interpretation: You predicted negative and it’s false.
We describe predicted values as Positive and Negative and actual values as True and False.



Recall

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

The above equation can be explained by saying, from all the positive classes, how many we predicted correctly.
Recall should be high as possible.

Precision

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

The above equation can be explained by saying, from all the classes we have predicted as positive, how many are positive?
Precision should be high as possible.

Accuracy
From all the classes (positive and negative), how many of them we have predicted correctly.
Accuracy should be high as possible.

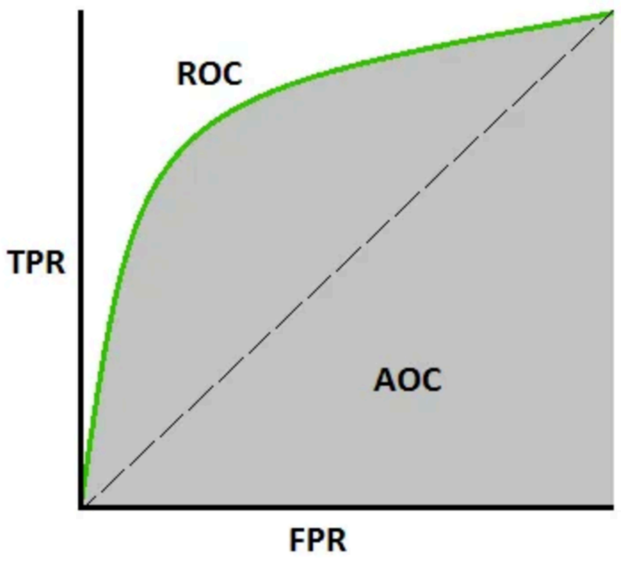
F-measure

$$\textbf{F - measure} = \frac{2*Recall*Precision}{Recall + Precision}$$

It is difficult to compare two models with low precision and high recall or vice versa. So, to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time.

AUC and ROC

AUC - ROC curve is a performance measurement for classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.
The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.



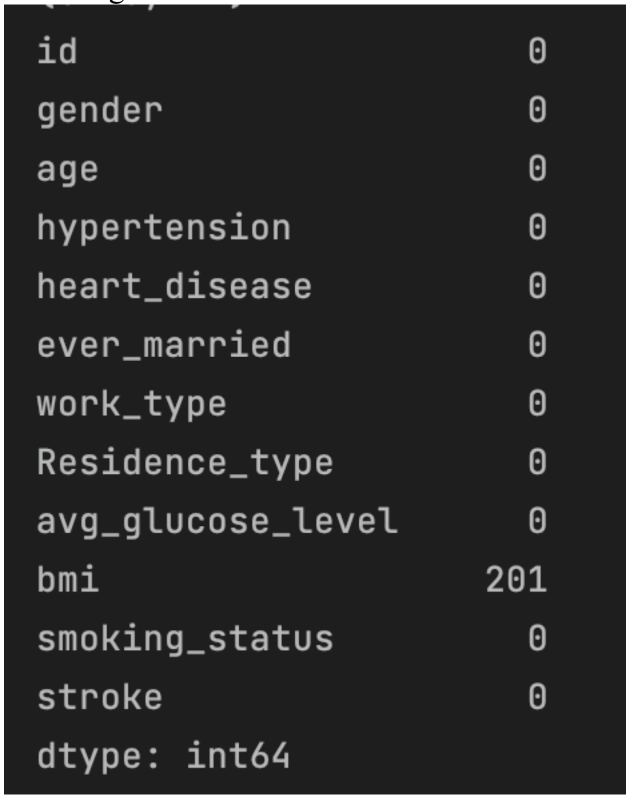
An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means the model has no class separation capacity whatsoever.

5 Result

5.1 Data Cleansing

The data cleansing process helps to ensure that the dataset used for training the machine learning model is of high quality, which in turn increases the likelihood of developing a robust and accurate model. It's essential to carefully examine the data, identify any issues or inconsistencies, and address them appropriately before moving forward with the modeling process.

The data includes NA values and unknown values, and the following steps were taken for data cleansing:



```
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                201
smoking_status       0
stroke               0
dtype: int64
```

1. Handling missing values (N/A): The dataset contains missing values (N/A) in the BMI column, with 201 missing values out of 5109 rows. Missing values can lead to incorrect or biased results when training a machine learning model. In this case, the missing values are filled with the mean of the BMI column. Using the mean is a common method for imputing missing values in numerical columns, as it helps to preserve the overall distribution of the data while minimizing the impact of missing values on the model. However, it's important to note that this approach may not always be appropriate, as it could introduce bias or skew the data distribution in certain cases.

2. Removing irrelevant columns: The 'id' column is removed from the dataset because it doesn't provide any meaningful information for the classification task. Including such columns in the analysis can lead to an unnecessarily complex model and can negatively impact the model's performance. By removing irrelevant columns, we can focus on the features that are important for predicting the target variable, resulting in a more accurate and efficient model.

5.2 Label Encoding Categorical Values

Label encoding is a process of converting categorical variables into numerical values so that machine learning algorithms can work with the data more effectively. Machine Learning algorithms, including the Random Forest classifier used in this project, require numerical inputs, and cannot directly handle categorical data. By converting the categorical variables into numerical values, we can ensure that the model can process the features and make predictions based on them. In this case, label encoding is applied to the following categorical columns: 'gender', 'ever_married', 'work_type', 'Residence_type', and 'smoking_status'. The LabelEncoder from the scikit-learn library is used to perform the encoding. For each column, the LabelEncoder assigns a unique integer value to each distinct category in the column, effectively mapping the categorical values to numerical values. After applying label encoding, you can see that the unique values in each categorical column have been replaced with numerical values, making the data more suitable for input to the machine learning model.

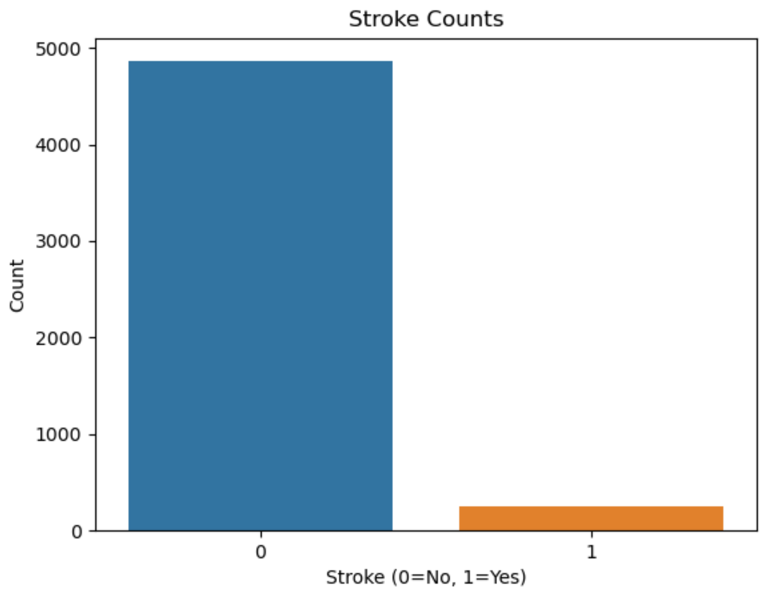
However, it is essential to consider the limitations of label encoding. One issue with label encoding is that it introduces an arbitrary ordering of categories, which might not reflect their true relationship. For example, assigning 0 to 'never smoked' and 1 to 'formerly smoked' in the 'smoking_status' column implies that 'formerly smoked' is greater than 'never smoked', which might not be accurate.

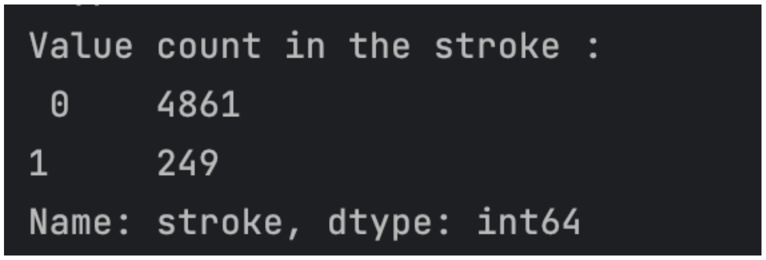
An alternative approach to handle categorical data is one-hot encoding, which creates binary (0 or 1) columns for each category in the original column, avoiding the ordinal relationship imposed by label encoding. However, one-hot encoding can significantly increase the number of columns in the dataset and potentially lead to a higher computational cost.

To overcome this limitation, my teammate used one-hot encoding, which creates binary (0 or 1) features for each category in a categorical variable. One of the most common ways to perform one-hot encoding is by using the pandas' library's get_dummies () function. This function creates a new data frame with binary columns for each category/label present in the categorical variable. Each row will have a 1 in the column corresponding to the category it belongs to and a 0 for the other categories.

5.3 EDA

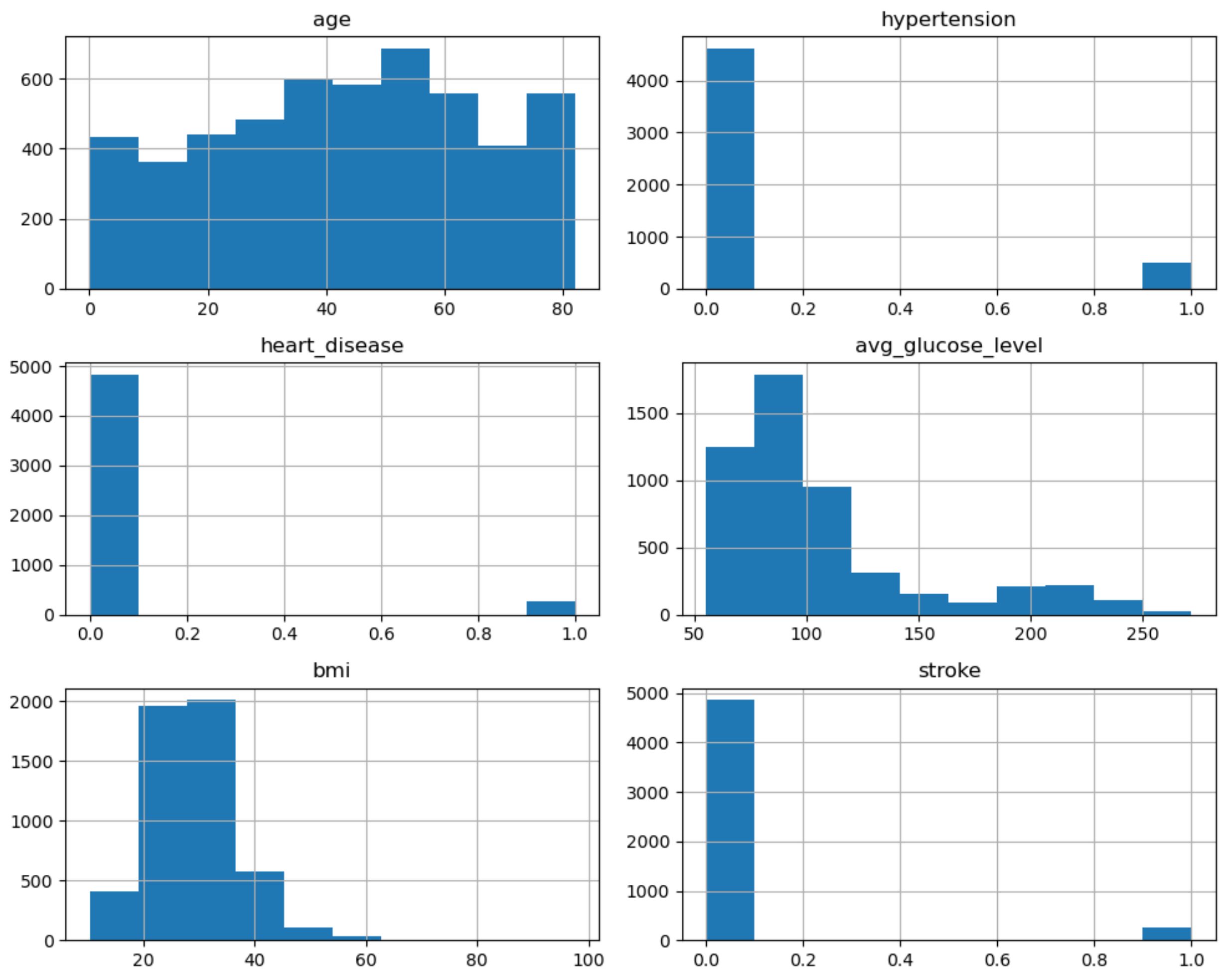
Target feature - **Stroke**: First, the value counts of the stroke variable are printed, showing the number of instances with and without a stroke. A bar chart is plotted to visualize the distribution of stroke cases (0 = No Stroke, 1 = Stroke). This helps in identifying any class imbalance in the dataset.



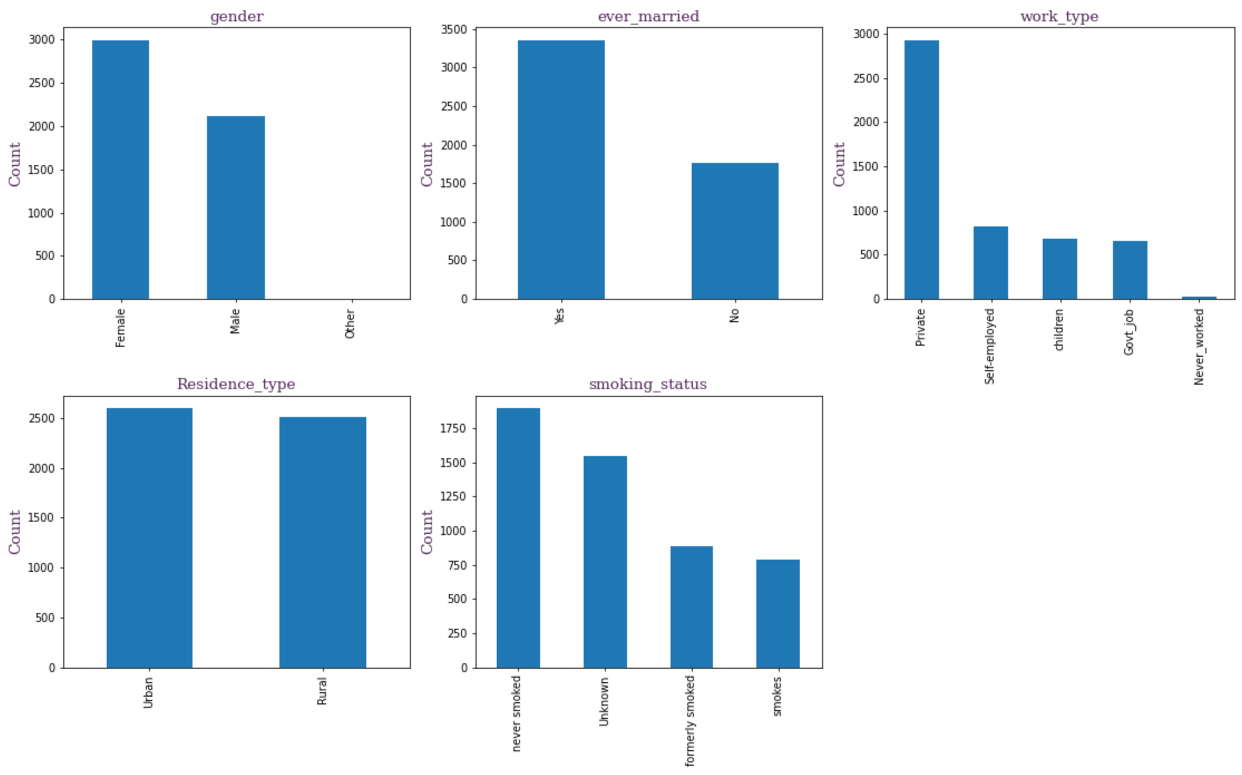


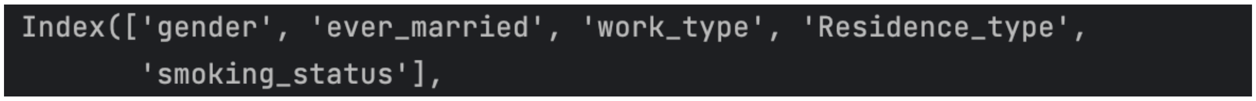
```
Value count in the stroke :
 0    4861
1     249
Name: stroke, dtype: int64
```

Numerical Variable Plot



Categorical Variable Plot

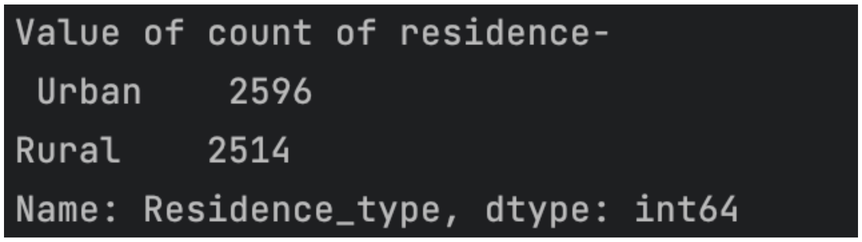




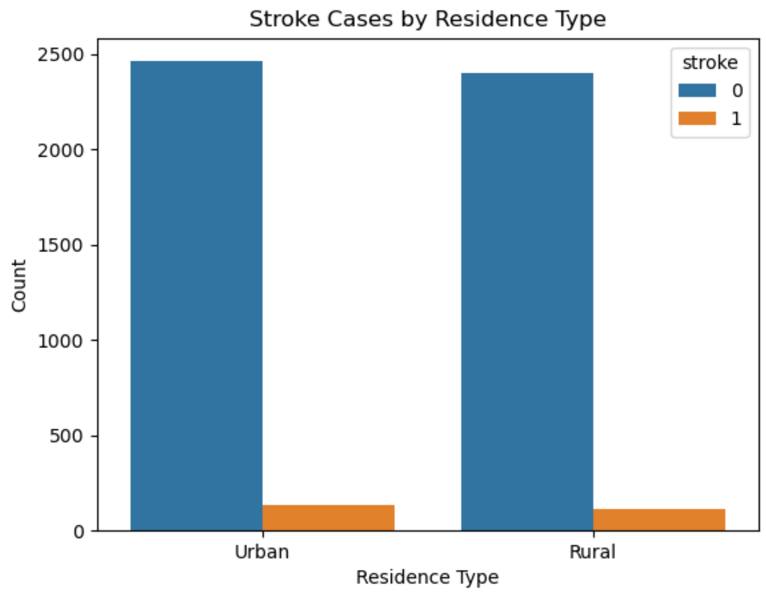
```
Index(['gender', 'ever_married', 'work_type', 'Residence_type',
       'smoking_status'],
```

From the above Plot, we have plotted for Categorical variables.

Residence Type: The value counts of the Residence_type variable are printed, and a bar chart is plotted to show the distribution of stroke cases by residence type (Urban or Rural).



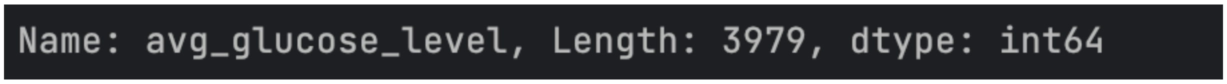
```
Value of count of residence-
 Urban     2596
Rural     2514
Name: Residence_type, dtype: int64
```



Observation:
This attribute is of no use. As we can see there is not much difference in both attribute values. Maybe we must discard it.

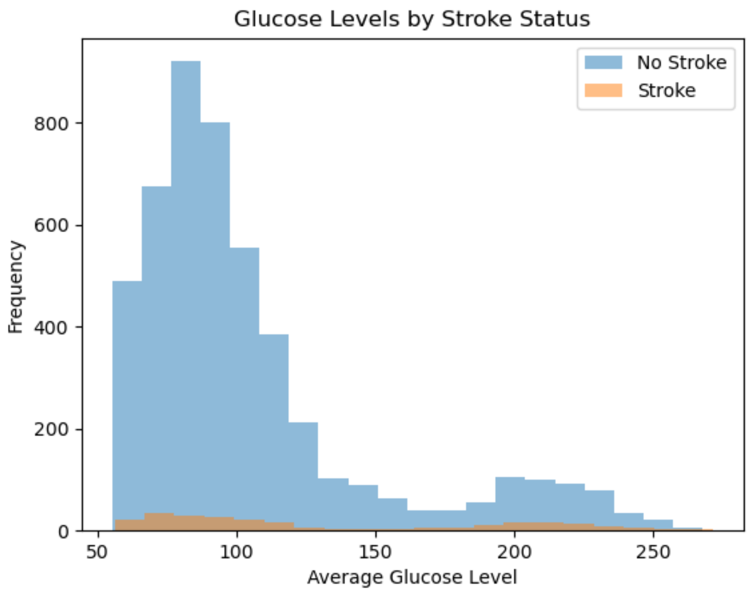
Average Glucose Level

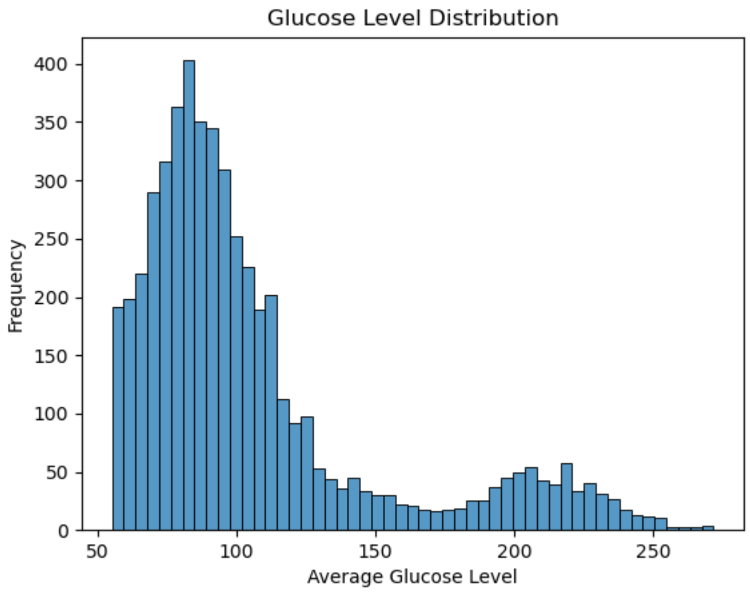
Value and Count



```
Name: avg_glucose_level, Length: 3979, dtype: int64
```

Plot -Glucose level with stroke

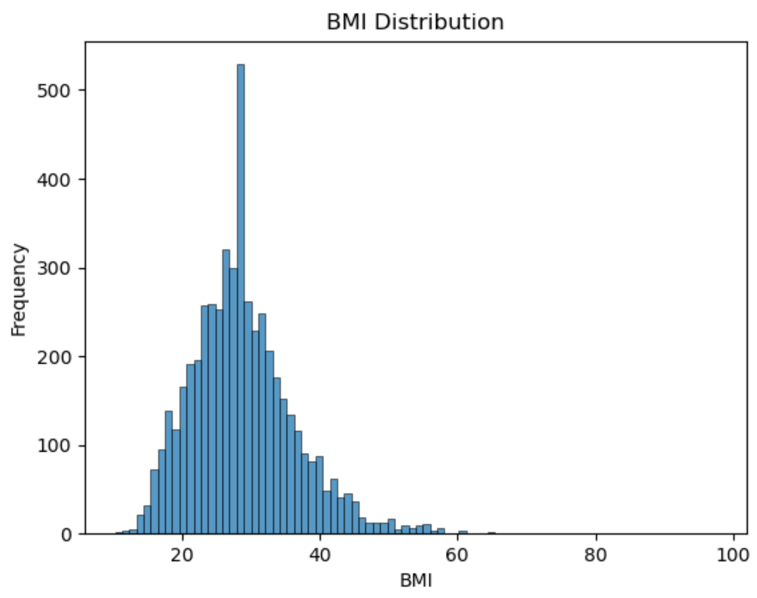




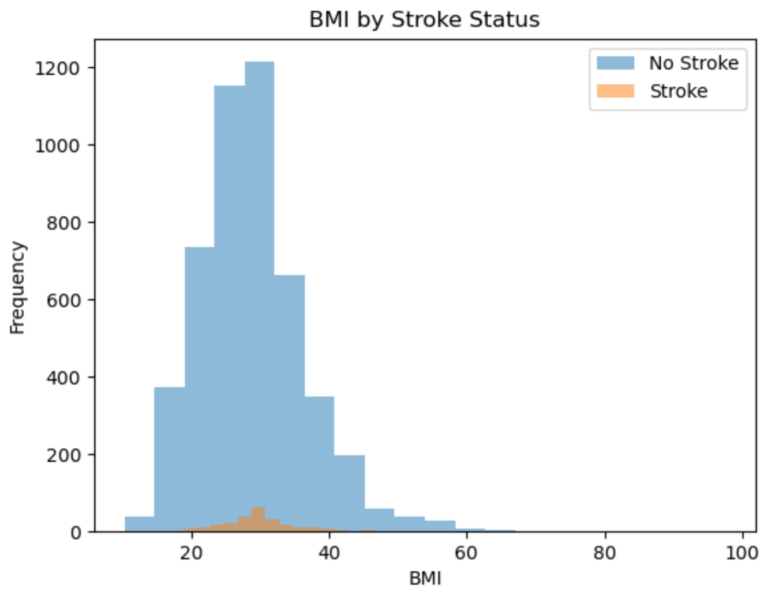
Observation: From the above graph, we can see that people having a stroke have an average glucose level of more than 100.

BMI

It has a null value of 201 and the count is 419.

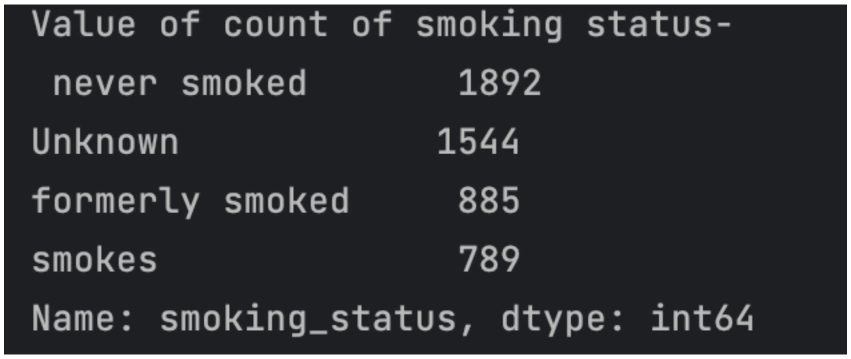


BMI With stroke



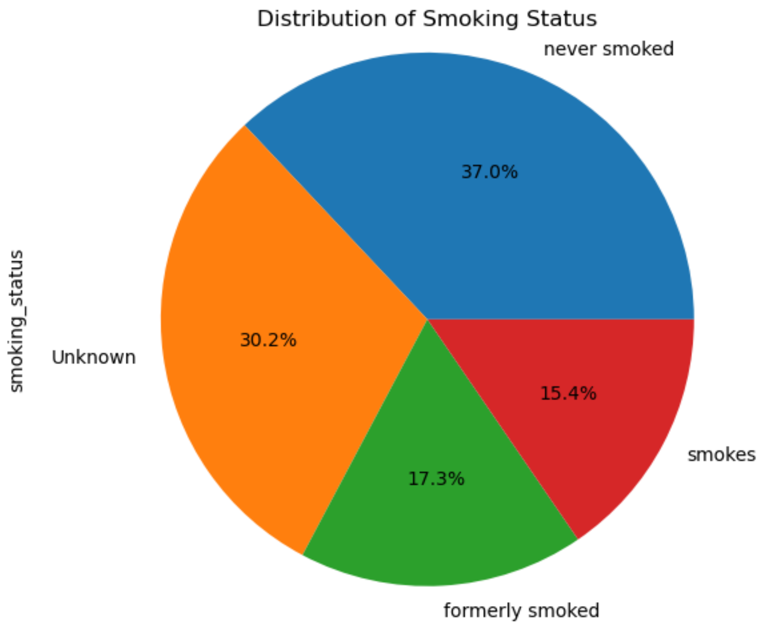
Smoking Status

Value count

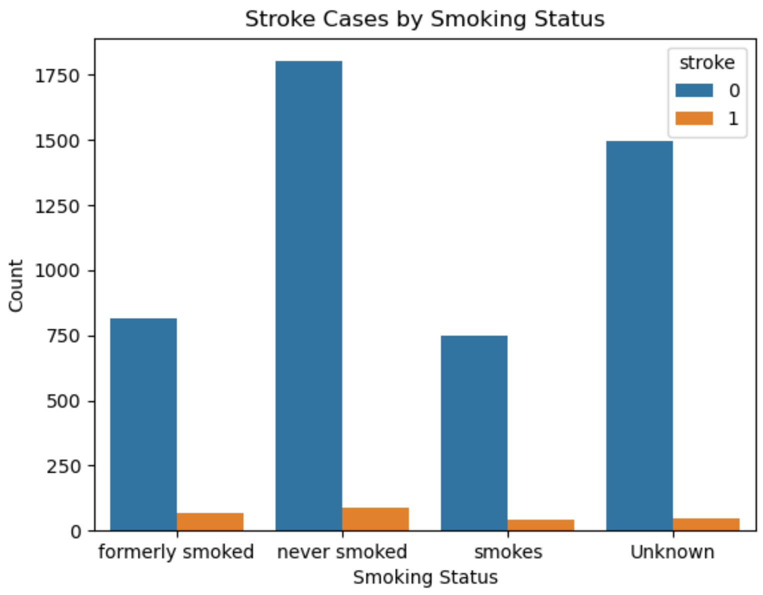


```
Value of count of smoking status-
 never smoked        1892
Unknown             1544
formerly smoked      885
smokes               789
Name: smoking_status, dtype: int64
```

Count plot:



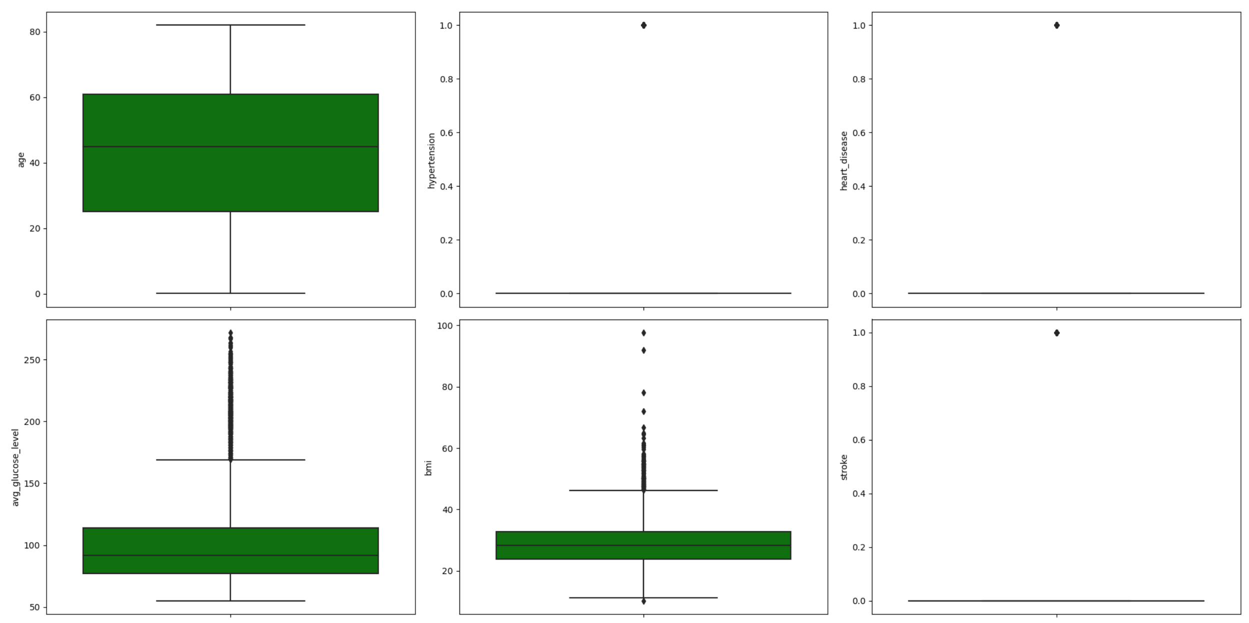
Smoke and Stroke:



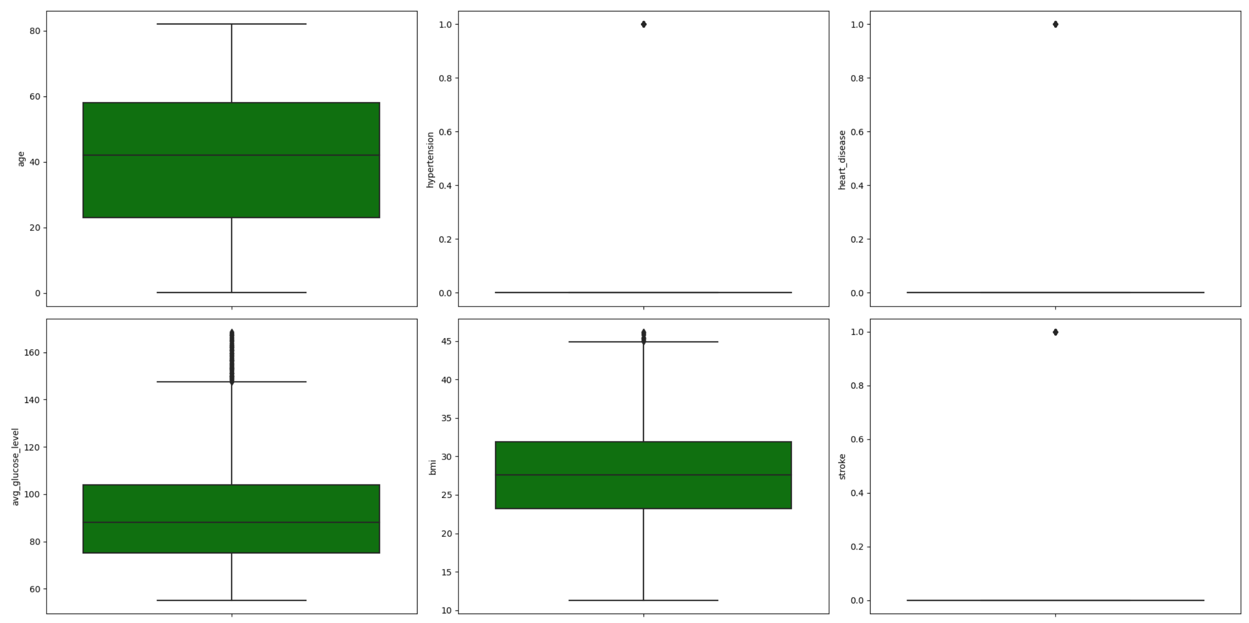
Observation: As per these plots, we can see there is not much difference in the chances of stroke irrespective of smoking status.

Outliers

In this, we identify and treat outliers in the dataset, specifically for the 'avg_glucose_level' and 'bmi' variables. Outliers can have a significant impact on the performance of certain machine learning models, as they might skew the data distribution and affect the model's ability to make accurate predictions. By treating the outliers, the data becomes more representative of the general population and less influenced by extreme values. This can help improve the performance of machine learning models, as they can better capture the underlying patterns in the data.



After Outliers

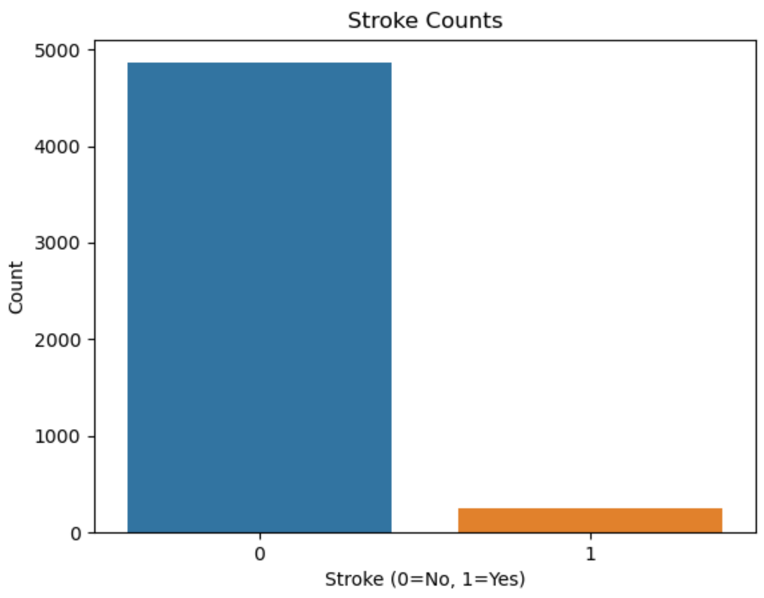


5.4 Data Sampling for Imbalanced Classification

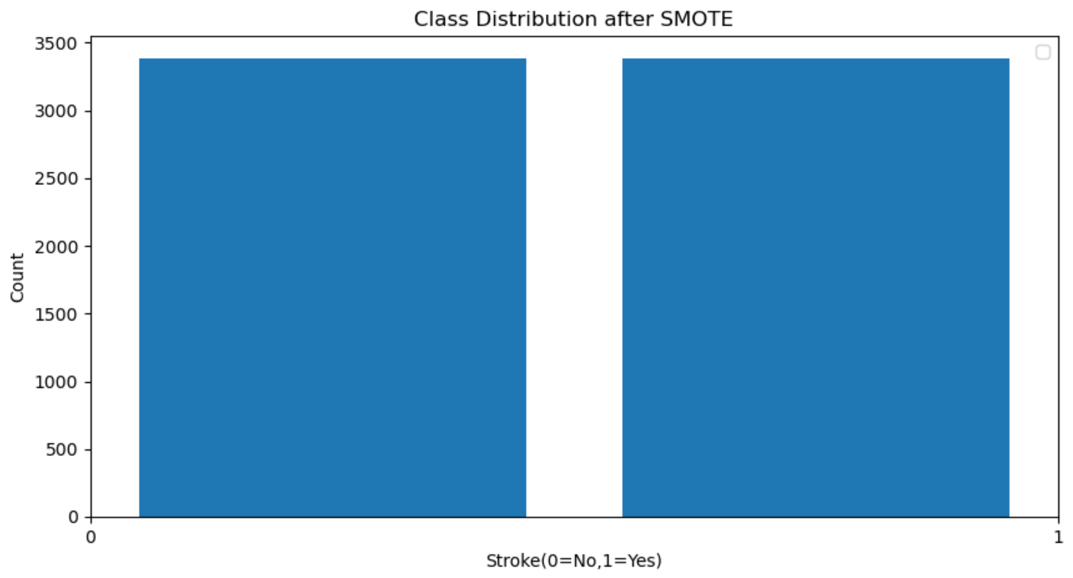
The output column "stroke" has a value of either 1 or 0, where 0 indicates no stroke risk and 1 indicates a stroke risk. In the dataset, the occurrence of 0 in the output column (stroke) is greater than the occurrence of 1. There are 129 rows with a value of 1 in the stroke column and 3383 rows with a value of 0. To enhance the accuracy, data preprocessing was applied to balance the data. There are 3389 patients with a stroke value equal to 0 and 129 patients with a stroke value equal to 1. The ratio of observations in ‘stroke’ is about 1:20, which is highly imbalanced. To resolve this problem, we under-sample the majority class, the class of stroke value being 1, to achieve a balanced class distribution. To resolve this problem, the SMOTE (Synthetic Minority Over-sampling Technique) was used to balance the data. The balance of the output column in the dataset after applying the SMOTE technique was as follows:

Class counts before SMOTE oversampling:
Class 0: 3389 Class 1: 129
Class counts after SMOTE oversampling:
Class 0: 3389 Class 1: 3389

Before using SMOTE



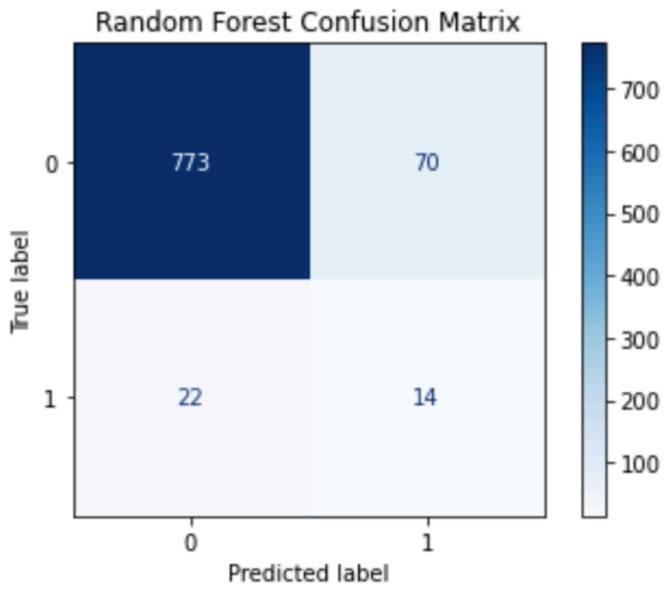
After using SMOTE



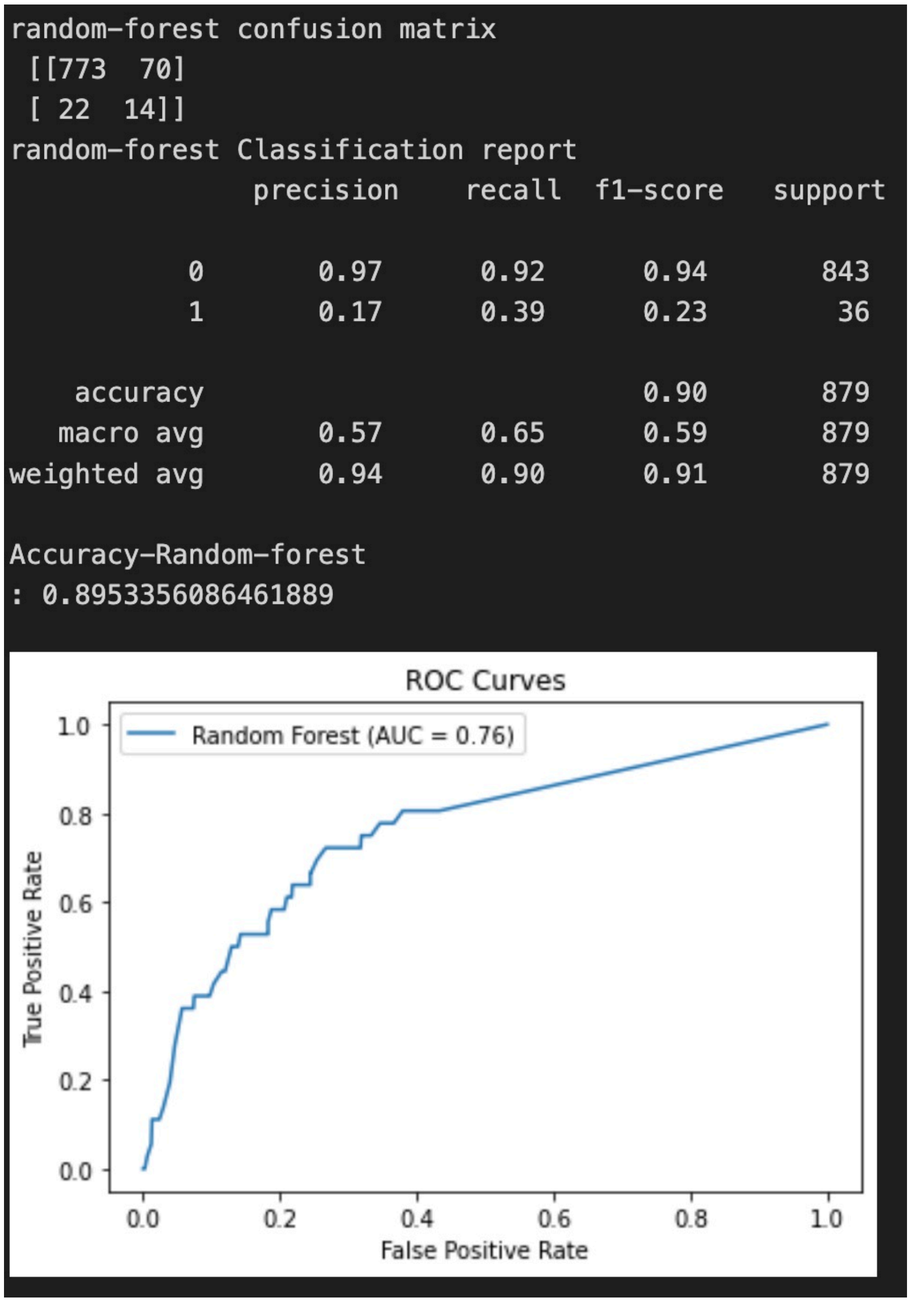
From the above plot, we can see the above distribution is an imbalanced ratio of 1:20, as the number of non-stroke cases (0) is significantly higher than stroke cases (1), with only 249 rows having a value of 1, while 4,861 rows have a value of 0. We have performed SMOTE technique which mainly performs oversampling and helps to handle overfitting.

5.5 Modelling

5.5.1 Random Forest



The confusion matrix shows the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions made by the model. In this case, the model correctly classified 773 instances as class 0 and 14 instances as class 1, while it incorrectly classified 70 instances as class 1 and 22 instances as class 0.

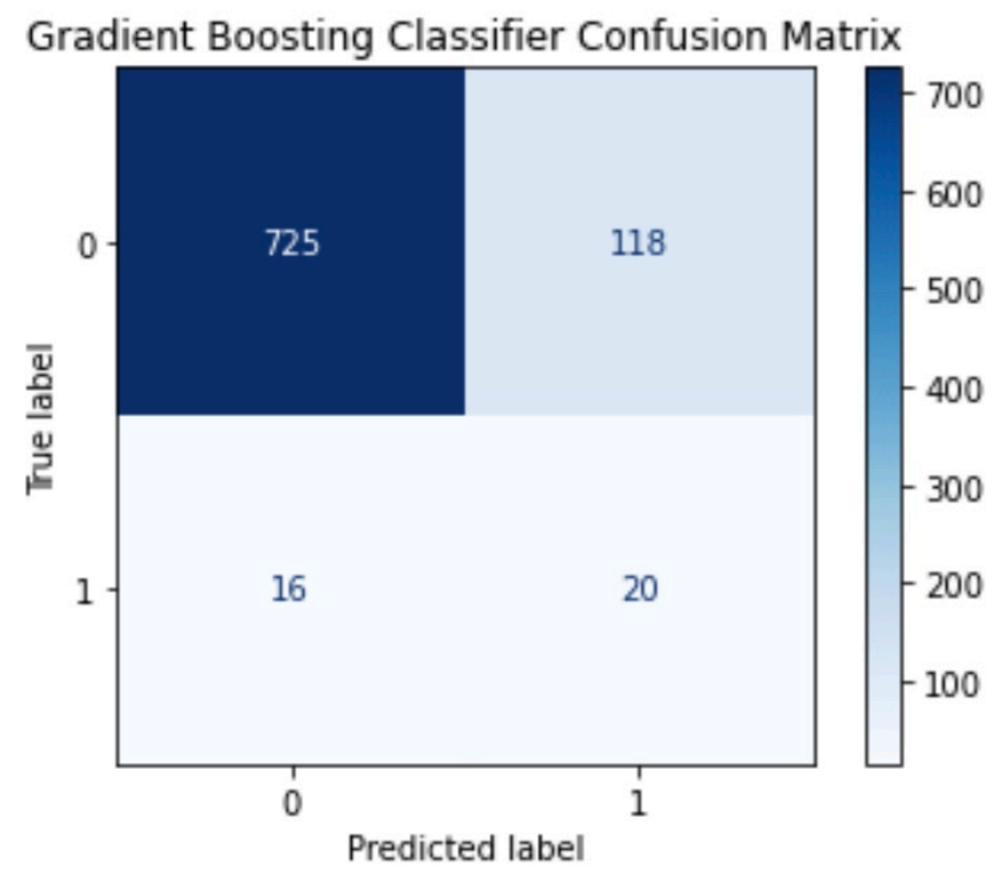


```
random-forest confusion matrix
 [[773  70]
 [ 22  14]]
random-forest Classification report
               precision    recall  f1-score   support

           0       0.97      0.92      0.94       843
           1       0.17      0.39      0.23        36

    accuracy                           0.90       879
   macro avg       0.57      0.65      0.59       879
weighted avg       0.94      0.90      0.91       879

Accuracy-Random-forest
: 0.8953356086461889
```

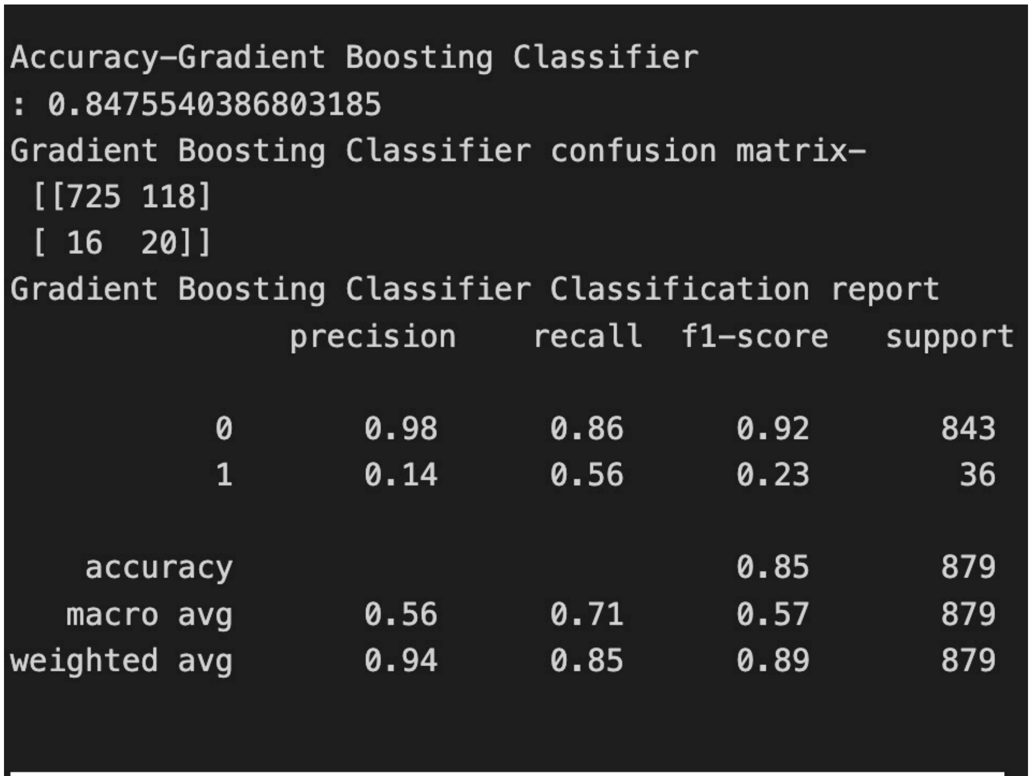
- Precision measures the proportion of positive predictions that are positive. For class 1, the precision is 0.17, which means that only 17% of the instances predicted as class 1 are class 1.

- Recall measures the proportion of actual positive instances that are correctly predicted as positive. For class 1, the recall is 0.39, which means that the model correctly identifies 39% of the actual class 1 instances.
- F1-score is the harmonic mean of precision and recall, and it provides a single score that balances both metrics. For class 1, the F1-score is 0.23, which is a low score indicating poor performance for this class.
- The overall accuracy of the model is 0.895, which means that the model correctly predicts the class for about 89.5% of the instances. However, it is important to note that the recall for class 1 is low, which means that the model is not very good at identifying class 1 instances. This could be a concern if class 1 represents a rare or important event that needs to be accurately detected. In such cases, recall may be a more important metric than accuracy.
- The AUC (Area Under the Curve) score of 0.76 indicates that the model has good performance in separating the positive class from the negative class.

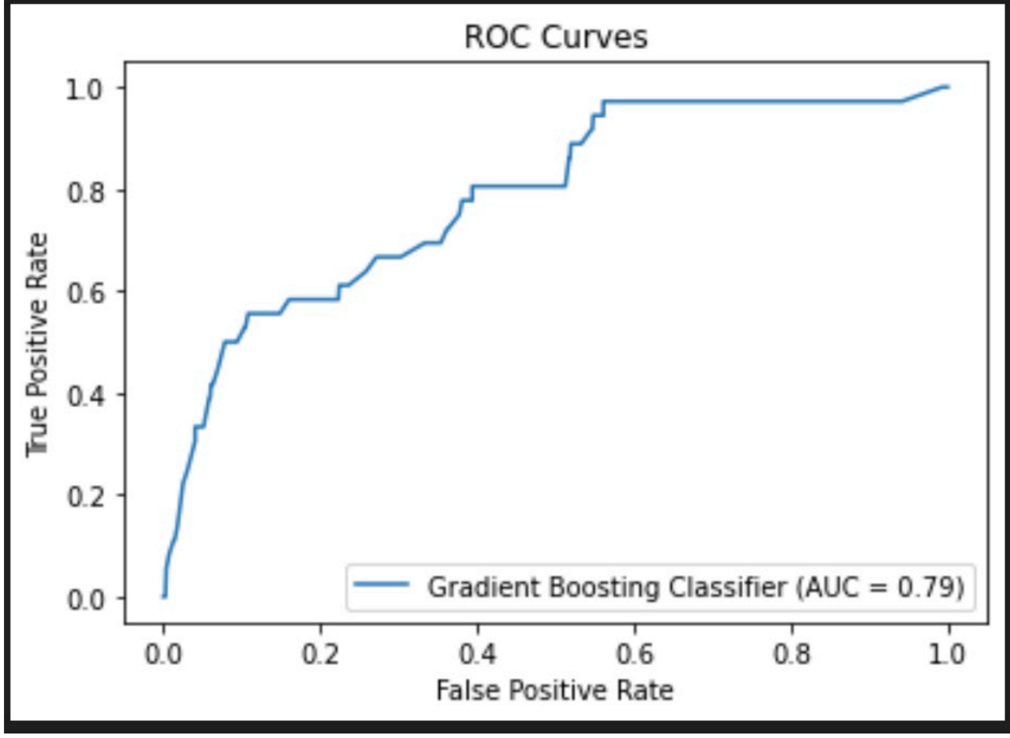
5.5.2 Gradient Boosting



The confusion matrix shows the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions made by the model. In this case, the model correctly classified 725 instances as class 0 and 20 instances as class 1, while it incorrectly classified 118 instances as class 1 and 16 instances as class 0.



```
Accuracy-Gradient Boosting Classifier
: 0.8475540386803185
Gradient Boosting Classifier confusion matrix-
 [[725 118]
 [ 16  20]]
Gradient Boosting Classifier Classification report
               precision    recall  f1-score   support

           0       0.98      0.86      0.92       843
           1       0.14      0.56      0.23        36

    accuracy                           0.85       879
   macro avg       0.56      0.71      0.57       879
weighted avg       0.94      0.85      0.89       879
```

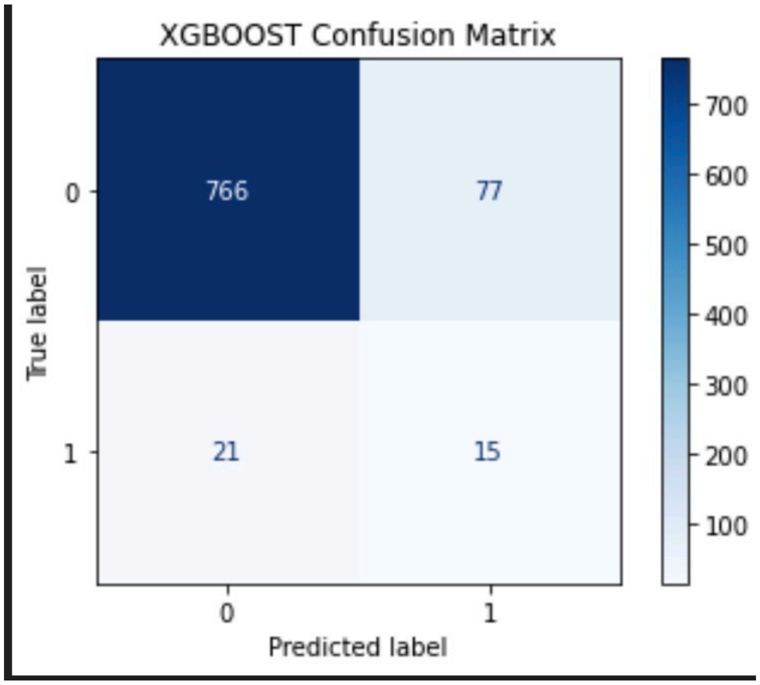


- The classification report provides several metrics that evaluate the performance of the model, including precision, recall, F1-score, and support.
- Precision measures the proportion of positive predictions that are positive. For class 1, the precision is 0.14, which means that only 14% of the instances predicted as class 1 are class 1.

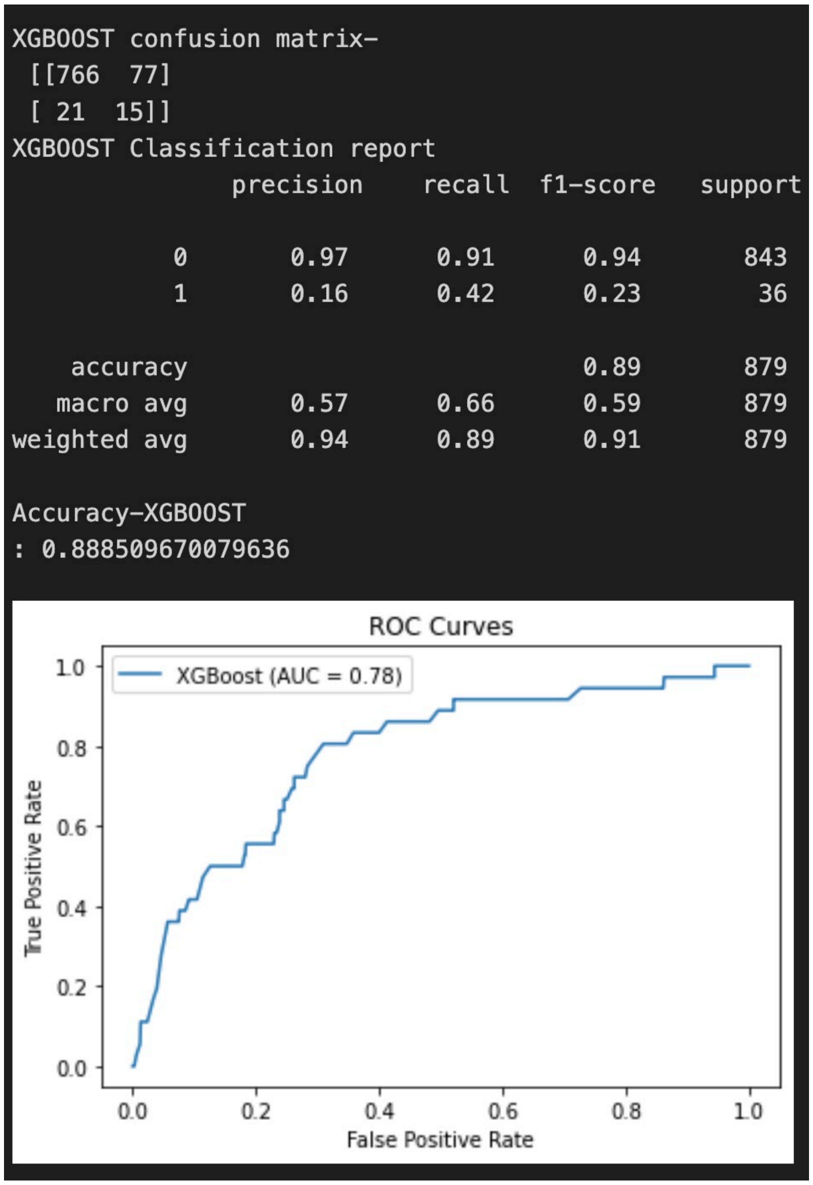
- Recall measures the proportion of actual positive instances that are correctly predicted as positive. For class 1, the recall is 0.56, which means that the model correctly identifies 56% of the actual class 1 instances.
- F1-score is the harmonic mean of precision and recall, and it provides a single score that balances both metrics. For class 1, the F1-score is 0.23, which is a low score indicating poor performance for this class.

This could still be a concern if class 1 represents a rare or important event that needs to be accurately detected. In such cases, recall may be a more important metric than accuracy.
The AUC (Area Under the Curve) score of 0.79 indicates that the model has good performance.

5.5.3 XGBoost



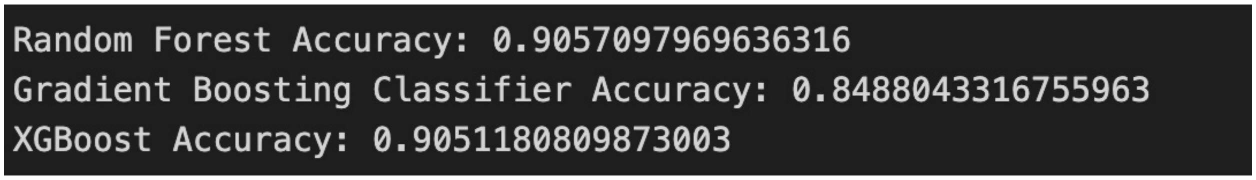
Based on the confusion matrix, the model correctly predicted 766 instances of class 0 and 15 instances of class 1, while it misclassified 77 instances of class 0 as class 1 and 21 instances of class 1 as class 0.



```
XGBOOST confusion matrix-
 [[766  77]
 [ 21  15]]
XGBOOST Classification report
               precision    recall  f1-score   support

           0       0.97      0.91      0.94       843
           1       0.16      0.42      0.23        36

    accuracy                           0.89       879
   macro avg       0.57      0.66      0.59       879
weighted avg       0.94      0.89      0.91       879

Accuracy-XGBOOST
: 0.888509670079636
```

- In this case, the precision for class 0 is 0.97, which means that 97% of the positive predictions for class 0 are correct. The recall for class 0 is 0.91, which means that 91% of the actual instances of class 0 are correctly detected by the model. The F1-score for class 0 is 0.94, which is a weighted average of precision and recall.
- The precision for class 1 is 0.16, which means that only 16% of the positive predictions for class 1 are correct. The recall for class 1 is 0.42, which means that only 42% of the actual instances of class 1 are correctly detected by the model. The F1-score for class 1 is 0.23, which is a low score indicating that the XGBoost model has poor performance for class 1.
- The ROC (Receiver Operating Characteristic) score of 0.78 indicates that the XGBoost model has good performance in separating the positive class from the negative class.

Cross Validation Scores



```
Random Forest Accuracy: 0.9057097969636316
Gradient Boosting Classifier Accuracy: 0.8488043316755963
XGBoost Accuracy: 0.9051180809873003
```

The results you provided are the accuracy scores obtained from cross-validating three different machine learning models: Random Forest, Gradient Boosting Classifier, and XGBoost.
These results suggest that the Random Forest model and the XGBoost model have similar performance in terms of accuracy, while the Gradient Boosting Classifier model has a slightly lower performance.
However, it's important to keep in mind that accuracy is not always the best metric to evaluate the performance of a machine learning model, especially in imbalanced datasets where some classes are underrepresented. In our project, other metrics such as recall should also be considered.

Model Name	Precision	F1 Score	ROC	Accuracy	Recall
Random Forest	0.17	0.23	0.76	0.89	0.39
XGBoost	0.16	0.23	0.78	0.88	0.42
Gradient	0.14	0.23	0.79	0.84	0.56

The tables you provided summarize the performance of three machine learning algorithms (Random Forest, XGBoost, and Gradient Boosting) on a binary classification problem. The precision, F1-score, ROC, accuracy, and recall are commonly used metrics to evaluate the performance of a binary classification model.
According to the table, the Gradient Boosting model has the highest recall value of 0.56, which indicates that the model correctly detects 56% of the positive instances. A recall is an important metric for evaluating the performance of a binary classification model, as it measures the proportion of positive instances that are correctly detected by the model.
The Gradient Boosting model also has a relatively high ROC score of 0.79, indicating good performance in separating the positive class from the negative class. However, the precision of 0.14 is relatively low, indicating that only 14% of the positive predictions made by the model are correct.
The Random Forest and XGBoost models have similar performance, with recall values of 0.39 and 0.42, respectively. Both models have relatively low precision values of 0.17 and 0.16, but relatively high ROC scores of 0.76 and 0.78, respectively.
In conclusion, the Gradient Boosting model has the highest recall value, which is the most important metric to consider in this case. In the stroke prediction project, the MLP model is the

best choice due to its highest recall score. The recall score is essential in stroke prediction as it measures the model's ability to accurately identify patients at risk of having a stroke, minimizing false negatives. A high recall score is crucial in medical applications, as missed positive cases can lead to severe health consequences.

6 Percentage of code

According to the formula, the code percentage is 38%.

7 Summary and Conclusions

Considering the problem statement of stroke prediction, recall is the most important evaluation metric. A high recall score ensures that the model can accurately identify a greater percentage of patients at risk of a stroke. Among all the tested models, the MLP model achieves the highest recall score of 0.81, making it the best choice for this problem.
Stroke is a life-threatening medical illness that should be treated as soon as possible to avoid further complications. The development of an ML model could aid in the early detection of stroke and the subsequent mitigation of its severe consequences. The effectiveness of several ML algorithms in properly predicting stroke based on several physiological variables is investigated in this study. The future scope of this study is that using a larger dataset and machine learning models, such as AdaBoost, SVM, and Bagging, the framework models may be enhanced. This will enhance the dependability of the framework and the framework's presentation. In exchange for just providing some basic information, the machine learning architecture may help the public in determining the likelihood of a stroke occurring in an adult patient. In an ideal world, it would help patients obtain early treatment for strokes and rebuild their lives after the event.

References

1. https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5
2. https://medium.com/dataman-in-ai/sampling-techniques-for-extremely-imbalanced-data-part-i-under-sampling-a8dbc3d8d6d8
3. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
4. G. Sailasya and G. L. A. Kumari, "Analyzing the performance of stroke prediction using ML classification algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 539–545, 2021 Google Scholar