

BACS2003/BMCS2003/BACS3074

(Remain only your course code before submission)

ARTIFICIAL INTELLIGENCE

**202401 Session, Year 2023/24**

**Assignment Documentation**

| **Project Title: Supervised Machine Learning on Stroke Prediction** |
| --- |
| **Programme: RSTY2S3** |
| **Tutorial Group: 4** |
| **Tutor: KEAT SOON TIOH** |
| **Team members’ data**   | **No** | **Student Name** | **Student ID** | **Module In Charge** | | --- | --- | --- | --- | | **1** | **TAN WEI JIE** | **23WMR09393** |  | | **2** | **WONG YIK XIANG** | **23WMR09395** | Xiang | | **3** | **ONG TING HAO** | **23WMR09389** |  | |

[**1. Introduction 3**](#_heading=)

[1.1. Problem Background 3](#_heading=h.3znysh7)

[1.2. Objectives/Aims 4](#_heading=h.2et92p0)

[1.3. Motivation 5](#_heading=h.tyjcwt)

[1.4. Timeline/Milestone 6](#_heading=h.3dy6vkm)

[**2. Research Background 7**](#_heading=h.1t3h5sf)

[2.1. Background of the applications 7](#_heading=h.4d34og8)

[2.2. Analysis of selected tool with any other relevant tools 8](#_heading=h.2s8eyo1)

[2.3. Justify why the selected tool is suitable 10](#_heading=h.17dp8vu)

[**3. Methodology 11**](#_heading=h.3rdcrjn)

[3.1. Description of dataset 11](#_heading=h.26in1rg)

[3.1.1. Data Understanding 12](#_heading=h.nzqbdqr0m6e5)

[3.1.2. Data Preprocessing 25](#_heading=h.y26z3ckays1j)

[3.1.2.1. Feature Engineering 29](#_heading=h.qb1lenssjd95)

[3.1.2.2. Detect and Remove Outliers 40](#_heading=h.rvwb837d3i7k)

[3.1.2.3. Split Data Into Training Set and Testing Set, Normalizing Data 67](#_heading=h.x487v2yzndxr)

[3.2. Applications of the algorithm(s) 75](#_heading=h.lnxbz9)

[3.2.1. Random Forest 75](#_heading=h.k409ljr0diy0)

[3.2.1.1. Random Forest Hyperparameter Tuning using GridSearchCV 75](#_heading=h.c7j1xyp4mkfk)

[3.2.1.2. Check Accuracy on Test Data 75](#_heading=h.e5o6mj4lucwk)

[3.2.2. CatBoost 75](#_heading=h.9nl2i9ptpg9d)

[3.2.2.1. Setting up the model 75](#_heading=h.ulrq9rvg3hvo)

[3.2.2.2. Check Accuracy on Test Data 76](#_heading=h.nbj5p2djktl2)

[3.2.3. XGBoost 76](#_heading=h.nof83xsthcik)

[3.2.3.1. Setting up XGBoost model 76](#_heading=h.baizxhika4lj)

[3.2.3.2. Check Accuracy on Test Data 76](#_heading=h.ntcm0vc108xh)

[3.3. System flowchart/activity diagram 77](#_heading=h.35nkun2)

[3.4. Proposed test plan/hypothesis 78](#_heading=h.1ksv4uv)

[**4. Result 79**](#_heading=h.44sinio)

[4.1. Results 79](#_heading=h.2jxsxqh)

[4.1.1. Random Forest 79](#_heading=h.r0au9jdy5f7w)

[4.1.1.1. Classification Report for Random Forest 79](#_heading=h.464ayadw50yd)

[4.1.1.2. Confusion Matrix for Random Forest 79](#_heading=h.fs2m8bmwvh1t)

[4.1.1.3. Conclusion of Random Forest 81](#_heading=h.xra887bib9ut)

[4.1.2. CatBoostClassifier 82](#_heading=h.v5m4p2r8ad8u)

[4.1.2.1. Classification Report for CatBoostClassifier 82](#_heading=h.jk6dlev8k3bu)

[4.1.2.2. Confusion Matrix for CatBoostClassifier 83](#_heading=h.olz6qk32mdvu)

[4.1.2.3. Conclusion of CatBoostClassifier 84](#_heading=h.2f80qefhla07)

[4.1.3. XGBClassifier 85](#_heading=h.cuwpyad9gzm)

[4.1.3.1. Classification report for XGBClassifier 85](#_heading=h.scueknxu55z8)

[4.1.3.2. Confusion Matrix for XGBClassifier 86](#_heading=h.rlun7x794y8g)

[4.1.3.3. Conclusion of XGBClassifier 87](#_heading=h.74zcc5d16rrf)

[4.2. Discussion/Interpretation 88](#_heading=h.z337ya)

[4.2.1. Evaluation 88](#_heading=h.qs59rgr4cp7w)

[4.2.1.1. Accuracy 88](#_heading=h.byiql6ov4g45)

[4.2.1.2. Precision 88](#_heading=h.ourwwmpb0bn9)

[4.2.1.3. Recall 89](#_heading=h.u7b8lg30erpo)

[4.2.1.4. F1-Score 89](#_heading=h.7lygvue9y13c)

[4.2.1.5. Specificity 90](#_heading=h.fiz77jswp1uk)

[4.2.2. Deployment 90](#_heading=h.dwk5jpj81wte)

[4.2.2.1. Selecting the Best Model 90](#_heading=h.89wqi2g626g8)

[4.2.2.2. Hardware Environment 95](#_heading=h.91ce1vl3l2y2)

[4.2.2.3. Software Environment 95](#_heading=h.uoc7b865o28g)

[4.2.2.4. Dataset (csv or from database) 95](#_heading=h.n46hjrtnosxm)

[**5. Discussion and Conclusion 97**](#_heading=h.3j2qqm3)

[5.1. Achievements 97](#_heading=h.1y810tw)

[5.2. Limitations and Future Works 98](#_heading=h.4i7ojhp)

[5.2.1. Limitations 98](#_heading=h.kmxgs2qkualm)

[5.2.2. Future Works 98](#_heading=h.ko9a66pt7tv6)

[**Reference & Source 99**](#_heading=h.3whwml4)

# **Introduction**

## Problem Background

Stroke is a serious medical condition that can lead to severe disabilities or even death if not promptly diagnosed and treated. It is essential to identify individuals at risk of stroke to implement preventive measures and provide timely interventions. Traditional risk assessment methods rely on medical history, physical examinations, and laboratory tests, which may not always capture the full spectrum of risk factors or predict outcomes accurately.

This study aims to leverage supervised machine learning techniques to develop a predictive model for stroke risk assessment. By analyzing a diverse range of health-related attributes and lifestyle factors, we seek to enhance the accuracy and efficiency of stroke prediction. Traditional risk assessment methods often have limitations in capturing complex interactions between various risk factors, leading to suboptimal predictions.

Supervised machine learning offers a promising approach to address these limitations by automatically learning patterns and relationships from data to make predictions. By training the model on historical data of individuals with and without strokes, we can create a predictive algorithm capable of identifying patterns indicative of stroke risk.

Furthermore, the development of such a predictive model can aid healthcare professionals in early detection and intervention, potentially reducing the incidence and severity of strokes. Additionally, it can empower individuals to take proactive measures to mitigate their stroke risk through lifestyle modifications and targeted interventions.

In summary, this study aims to harness the power of supervised machine learning to improve stroke prediction, thereby contributing to more effective preventive healthcare strategies and better patient outcomes.

## 

## Objectives/Aims

The study aims to improve our knowledge and understanding of supervised machine learning algorithms for classification. We should know how supervised machine learning algorithms work to fully understand and utilize the concept and algorithms of supervised machine learning. The second objective of this study is also improving our skills in applying and improving the chosen algorithms. This is because there are many ways to improve our models' accuracy to achieve the correct prediction through our analysis. The third objective is achieving 80% accuracy, which the model selected.

## Motivation

The development of a supervised machine learning model for stroke prediction presents promising opportunities for both commercialization and profound social impact. From a commercial standpoint, companies specializing in healthcare technology and analytics could capitalize on the predictive model by integrating it into comprehensive stroke risk assessment solutions tailored for healthcare providers. These solutions would offer hospitals, clinics, and healthcare systems advanced tools to enhance their preventive care strategies, potentially leading to improved patient outcomes and reduced healthcare costs.

Moreover, the incorporation of the predictive model into consumer health products, such as wearable health monitors or smart scales, could unlock new avenues for innovation. Wearable technology companies could leverage the model to provide users with personalized stroke risk assessment features, empowering individuals to monitor their health more proactively and take preventive action where necessary. This integration could significantly enhance the value proposition of these devices in the consumer market.

Additionally, insurance companies could benefit from integrating the predictive model into their risk assessment processes. By identifying individuals at higher risk of stroke, insurers can tailor their offerings, such as premiums and preventive health programs, more effectively. This targeted approach to risk management could lead to more sustainable business practices for insurers while also improving health outcomes for policyholders.

On a societal level, the social impact of the predictive model for stroke prediction is profound. Early detection and intervention facilitated by the model could lead to fewer strokes, reduced disabilities, and improved quality of life for stroke survivors and their families. Furthermore, by addressing disparities in healthcare access and outcomes, the model has the potential to promote health equity, ensuring that all individuals, regardless of background or socioeconomic status, have access to timely preventive care.

Public health initiatives could also benefit from the predictive model, with governments and public health organizations leveraging its insights to inform population-level stroke prevention strategies. By raising awareness about stroke risk factors and promoting lifestyle modifications, community-based interventions, and educational campaigns, these initiatives could contribute to healthier communities and populations.

Lastly, the predictive model could drive further research and innovation in stroke prevention and management. By providing a robust framework for analyzing risk factors and predicting stroke occurrence, the model could inspire new discoveries in stroke pathophysiology, novel risk factors, and targeted interventions, ultimately advancing the field of stroke research and clinical practice. In summary, the supervised machine learning model for stroke prediction holds immense potential for commercialization and significant social impact, offering opportunities for innovation, improved healthcare outcomes, and greater health equity.

## Timeline/Milestone

| Works | Remarks | Duration |
| --- | --- | --- |
| Planning | Gathering idea and assign taks | Week 4,  11/03/2024 - 17/03/2024 |
| Background studies | Do research for relevant information and references in studies | Week 5,  18/03/2024 - 24/03/2024 |
| Finding dataset | Find a relevant and compatibility dataset for studying | Week 6,  25/03/2024 - 31/03/2024 |
| Data Understanding | Understanding the pattern and correlation relationship between variables. | Week 7,  01/04/2024 - 07/04/2024 |
| Data Pre-Processing | Normalization, detect and remove outliers, split data into training and testing datasets. | Week 8,  08/04/2024 - 14/04/2024 |
| Modeling | To predict labels by training a machine learning algorithm | Week 9,  15/04/2024 - 21/04/2024 |
| Evaluation | Evaluate the models by accuracy, precision, recall, etc. | Week 9,  15/04/2024 - 21/04/2024 |
| Deployment | Select the best model and create a simple programme to predict the result. | Week 10,  22/04/2024 - 28/04/2024 |
| Reporting | Complete the reporting | Week 10,  22/04/2024 - 28/04/2024 |
| Submission | Submit the assignment and report in time | Week 11,  29/04/2024 - 03/05/2024 |

# **Research Background**

## Background of the applications

Traditionally, stroke risk assessments are only done by clinics through periodical blood tests and only factors in factors like age, blood cholesterol level and more. These methods are crucial as they directly reflect the risk but doing so may cost a lot or require time to go to the clinic yearly. With the aforementioned concern, using Machine Learning can greatly reduce the resources needed. With a large amount of data collected, we can then use those collected data and process them through Machine Learning algorithms to obtain a robust predictive model that is good at identifying the subtle patterns of pre-stroke warnings as well as providing recommendations to the patient based on his risk level and whether a real follow up examination by a professional doctor is to be conducted or not.

To create such application, Machine Learning algorithms like Linear Regression, Decision Trees, Support Vector Classification, Random Forest are commonly used because of their ability to process large amounts of data, notice subtle patterns and provide statistical feedback. Other than that, the approach that we are going for is supervised learning where we provide the algorithm with labeled datasets for them to train and learn the relationship between the input and outputs so that the algorithm can predict the outcomes and recognize patterns. With supervised learning, we will be able to easily create complex models that can make accurate predictions for healthcare.

## Analysis of selected tool with any other relevant tools

| **Tools comparison** | **Remark** | **Jupyter Notebook** | **Google Colab** | **RStudio** |
| --- | --- | --- | --- | --- |
| Type of license and open source license | State all types of license | Open-source  Platform: Anaconda  modified BSD license | Closed-source  Not applicable license | Open-source with license Affero General Public License (AGPL) |
| Year founded | When is this tool being introduced? | 2014 | 2012 | 2009 |
| Founding company | Owner | Project maintained by community | Google | RStudio, PBC |
| License Pricing | Compare the prices if the license is used for development and business/commercialization | Free | Free (with limited resources), Paid (Google Workspace) | Free (open-source), Paid (RStudio Pro) |
| Supported features | What features that it offers? | - Interactive development environment  - Multiple programming language support  - Code reproducibility and documentation  - Data visualization capabilities  - Collaboration features | - Interactive development environment  - Integrated with Google Drive  - Provides free GPU/TPU usage (limited)  - Support for Python programming language | - Interactive development environment  - Focused on R programming language  - Integrated development environment for R  - Code debugging and profiling features |
| Common applications | In what areas this tool is usually used? | Data exploration and analysis  Machine learning model development  Data visualization and communication of results | Similar applications as Jupyter Notebook, particularly for machine learning projects | Statistical analysis    Data visualization  Data science projects involving R programming |
| Customer support | How the customer support is given, e.g. proprietary, online community, etc. | Community-driven support forums  Documentation and tutorials available online | Limited official support from Google  Community-driven support through forums and documentation | Official support from RStudio  Documentation, tutorials, and webinars  Community-driven support forums |
| Limitations | The drawbacks of the software | - Resource-intensive for large datasets  - Limited support for real-time collaboration  - Steeper learning curve for beginners | - Limited resources for free usage (e.g., computing power)  - Dependency on Google services  - Limited support for non-Python programming languages | - Focused on R programming language, may not be suitable for projects requiring other languages  - Limited support for non-R programming languages  - May have fewer collaboration features compared to Jupyter Notebook |

## Justify why the selected tool is suitable

The choice of Jupyter Notebook for a project focused on developing a supervised machine learning model for stroke prediction is justified by its versatile and interactive development environment. Jupyter Notebook allows seamless integration of code, visualizations, and explanatory text, making it ideal for exploring data, experimenting with different machine learning algorithms, and iteratively refining the predictive model. Its support for multiple programming languages, including Python, R, and Julia, ensures compatibility with popular machine learning libraries such as scikit-learn, TensorFlow, and PyTorch, enhancing its utility in data science projects.

One of the key strengths of Jupyter Notebook lies in its ability to promote code reproducibility and documentation. By enabling the documentation of code, analysis steps, and results in a single document format, Jupyter Notebook facilitates transparency and reproducibility, allowing other researchers or stakeholders to easily understand and replicate the project's findings. Additionally, its integration with data visualization libraries like Matplotlib, Seaborn, and Plotly enables the creation of insightful visualizations to explore the dataset, understand relationships between variables, and communicate insights effectively.

Moreover, Jupyter Notebook supports collaboration among team members by allowing the sharing of notebooks via platforms like GitHub or JupyterHub. This facilitates collaborative development, peer review, and knowledge sharing throughout the project lifecycle. Furthermore, Jupyter Notebook offers flexibility and customization options, allowing users to incorporate custom code, extensions, and widgets to tailor the environment to their specific needs. This flexibility enables users to adapt the notebook to different stages of the machine learning pipeline, from data preprocessing to model evaluation.

In summary, the versatility, interactivity, collaboration features, and documentation capabilities of Jupyter Notebook make it an ideal choice for developing a supervised machine learning model for stroke prediction. Its seamless integration with the data science ecosystem, support for multiple programming languages, and visualization capabilities empower researchers to explore data, experiment with models, and communicate findings effectively, ultimately contributing to the success of the project.

# **Methodology**

## Description of dataset

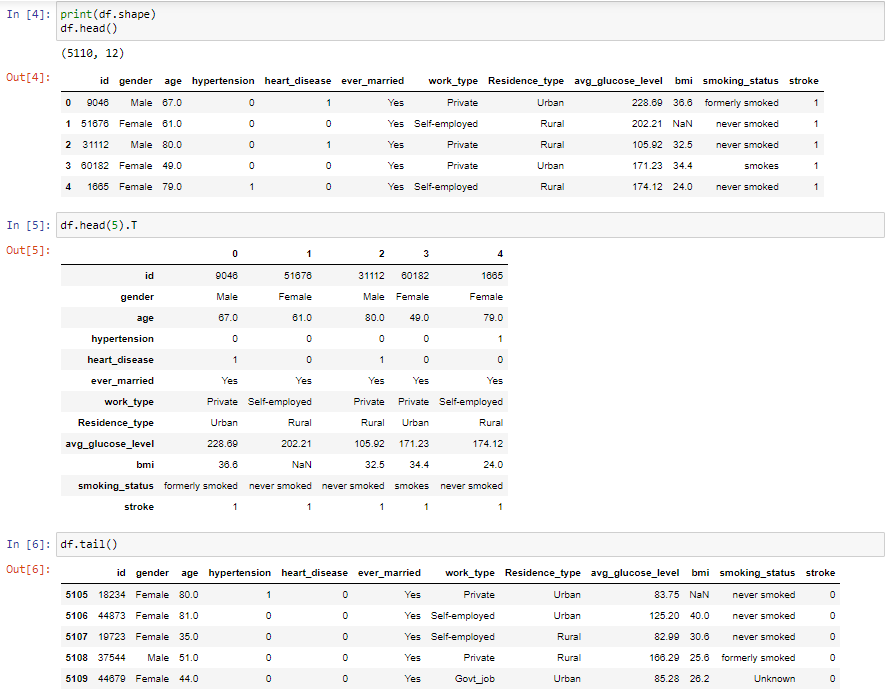
The dataset's source is Kaggle. The dataset consists of a total of 12 columns and 9 of the columns indicate personal details such as id, gender, age, bmi, ever\_married, stroke, avg\_glucose\_level, hypertension, heart\_disease and 3 of the columns indicate the person's lifestyle such as work\_type, Residence\_type, and smoking\_status. There are a total of 5110 rows of data for us to study.

| No. | Name | Type | Dependent / Independent Variable | Description |
| --- | --- | --- | --- | --- |
| 1 | id | Numeric | Independent Variable | The id of the data. |
| 2 | gender | Categorical | Independent Variable | The gender of the person. |
| 3 | age | Numeric | Independent Variable | The age of the person, some data contains 2 decimal points. |
| 4 | hypertension | Categorical | Independent Variable | Indicates whether the person has hypertension. |
| 5 | heart\_disease | Categorical | Independent Variable | Indicates whether the person has heart disease records. |
| 6 | ever\_married | Categorical | Independent Variable | Indicates whether the person has ever married. |
| 7 | work\_type | Categorical | Independent Variable | Indicates the person's work type. |
| 8 | Residence\_type | Categorical | Independent Variable | Indicates what type of area the person lives in. |
| 9 | avg\_glucose\_level | Numeric | Independent Variable | The average glucose level of the person. |
| 10 | bmi | Numeric | Independent Variable | The bmi of the person. |
| 11 | smoking\_status | Categorical | Independent Variable | Indicates the smoking status of the person. |
| 12 | stroke | Categorical | Independent Variable | Indicates whether the person has stroke. |

### Data Understanding

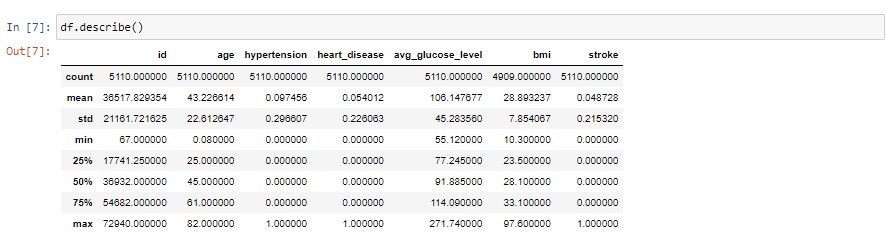


**Figure 3.1.1.1**



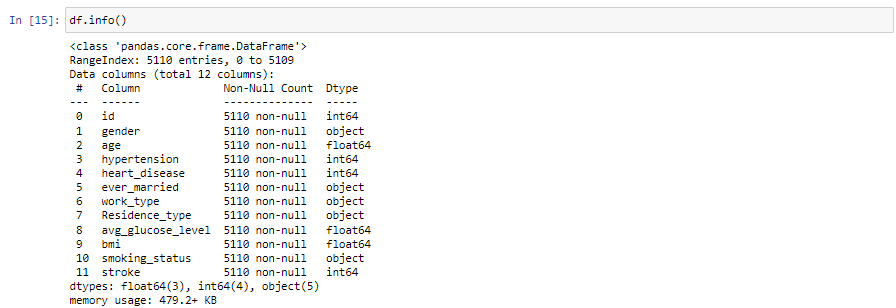
**Figure 3.1.1.2**

The diagrams above, Figure 3.1.1.1 and Figure 3.1.1.2 show that the data have been successfully inserted and retrieved, respectively. The displayed first five rows and last five rows are the data after successfully inserting the 5110 rows of data inside the program.



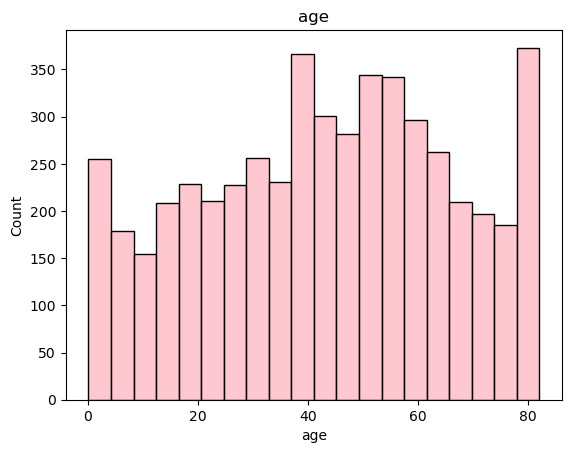
**Figure 3.1.1.3**

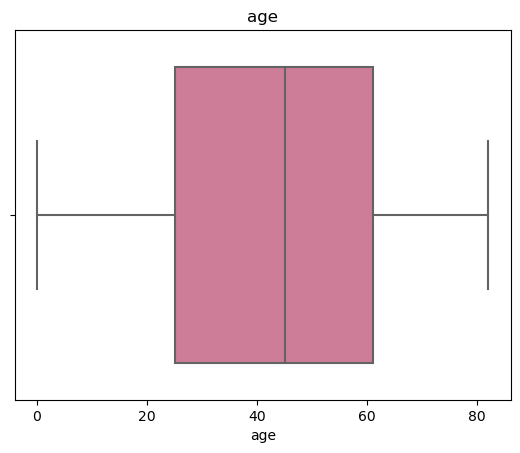
Figure 3.1.1.3 shows the statistics information of our data, including the count, mean, standard deviation, first quartile, second quartile, third quartile and the max value for each of the columns.



**Figure 3.1.1.5**

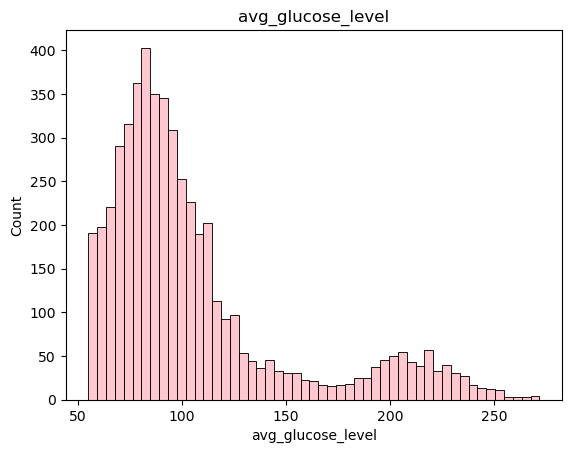
Figure 3.1.1.5 shows the information on each of the columns that have been inserted into our program. It shows that there are a total of 11 columns. It also shows the data type for each column. Most of the data types are float, object and int value.

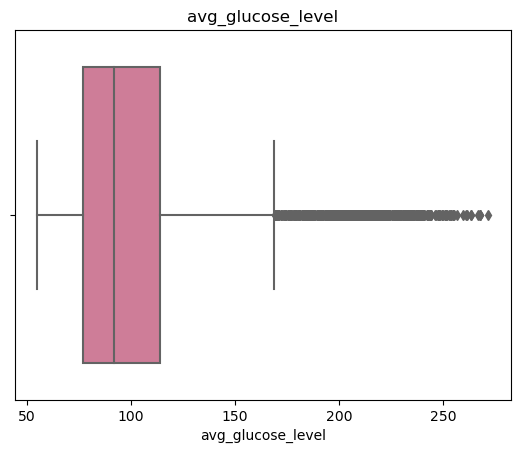




**Figure 3.1.1.6**

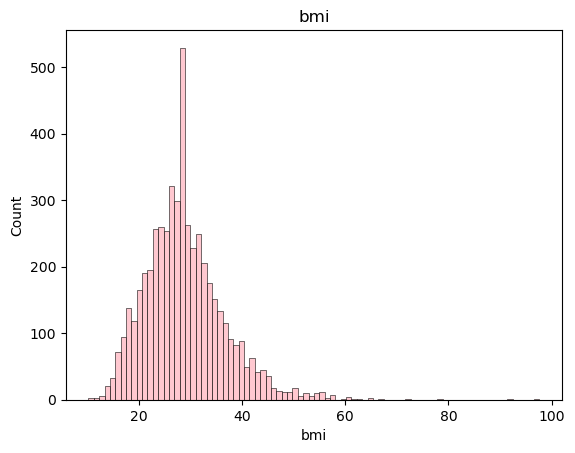
Figure 3.1.1.6 shows the distribution of age values and their count in the dataset.

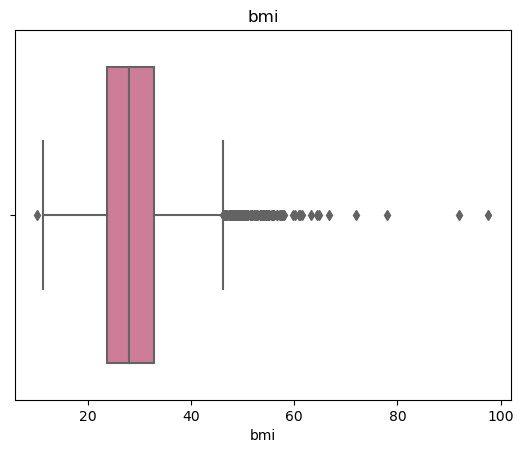




**Figure 3.1.1.7**

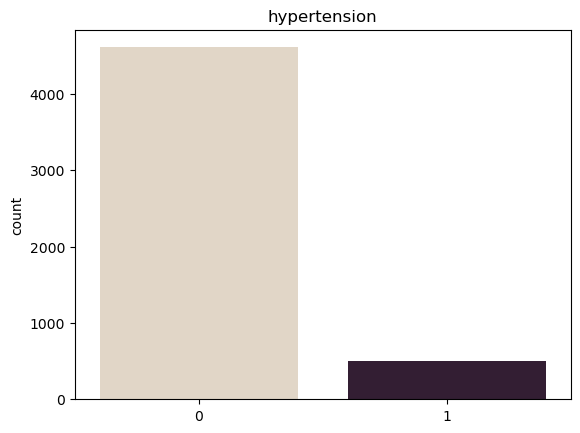
Figure 3.1.1.7 shows the distribution of average glucose level values and their count in the dataset.





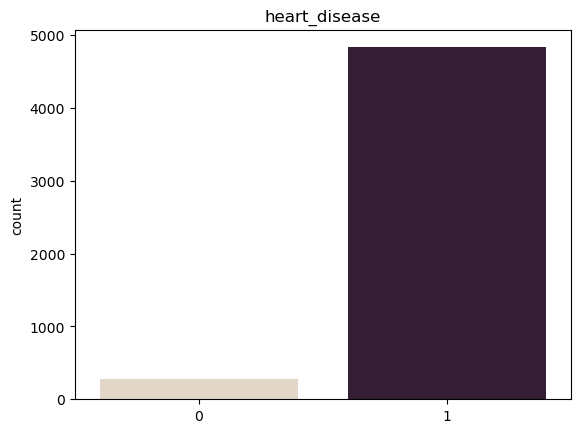
**Figure 3.1.1.8**

Figure 3.1.1.8 shows the distribution of bmi values and their count in the dataset.

****

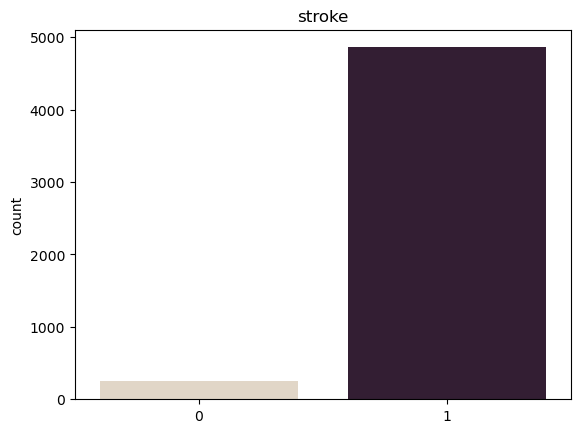
**Figure 3.1.1.9**

Figure 3.1.1.9 shows the amount of persons with and without hypertension in the dataset.

****

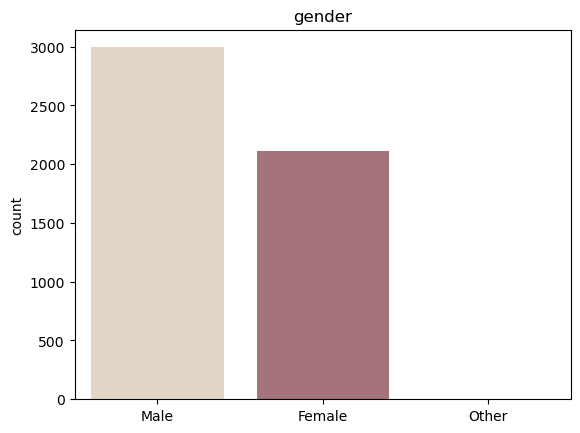
**Figure 3.1.1.10**

Figure 3.1.1.10 shows the amount of persons with and without heart disease in the dataset.



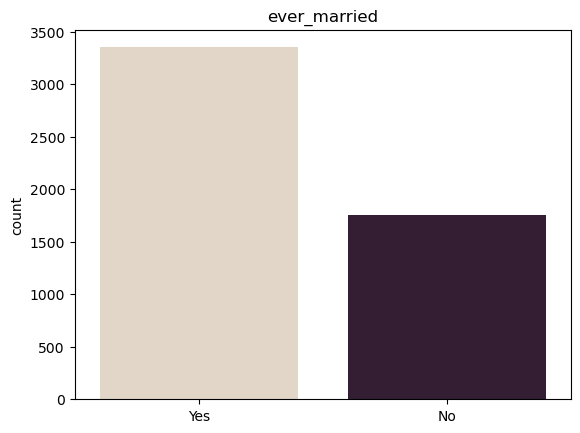
**Figure 3.1.1.11**

Figure 3.1.1.11 shows the amount of persons with and without stroke in the dataset.



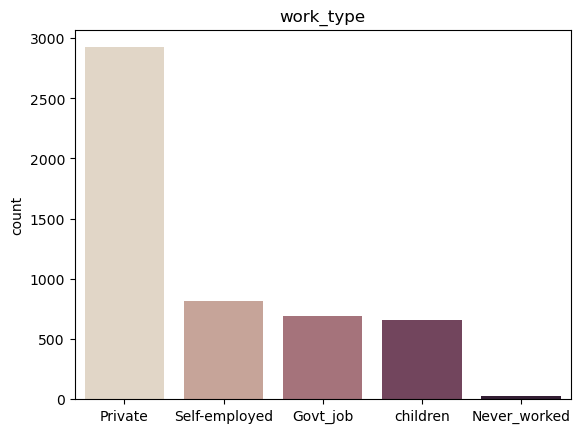
**Figure 3.1.1.12**

Figure 3.1.1.12 shows the count of each gender in the dataset.



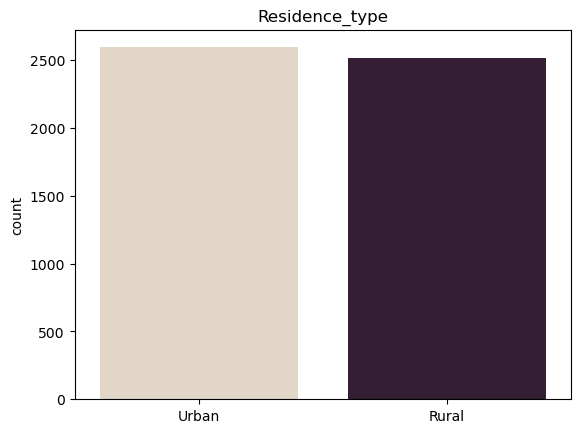
**Figure 3.1.1.13**

Figure 3.1.1.13 shows the amount of persons that have never married and vice versa in the dataset.



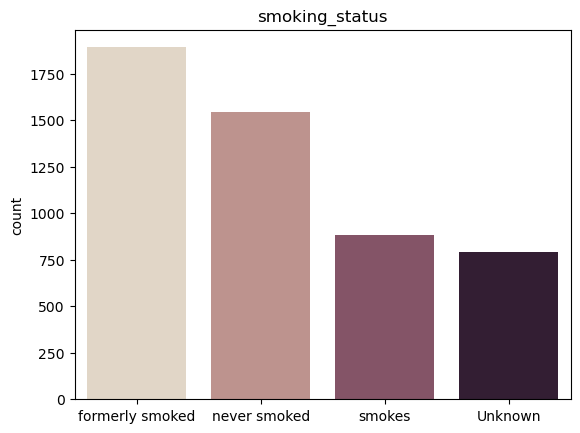
**Figure 3.1.1.14**

Figure 3.1.1.14 shows the different work types and their count in the dataset.



**Figure 3.1.1.15**

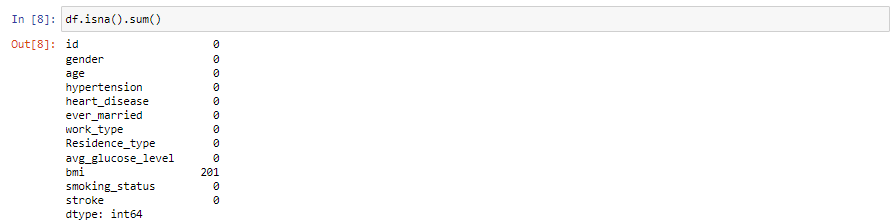
Figure 3.1.1.15 shows the different residence types and their count in the dataset.



**Figure 3.1.1.16**

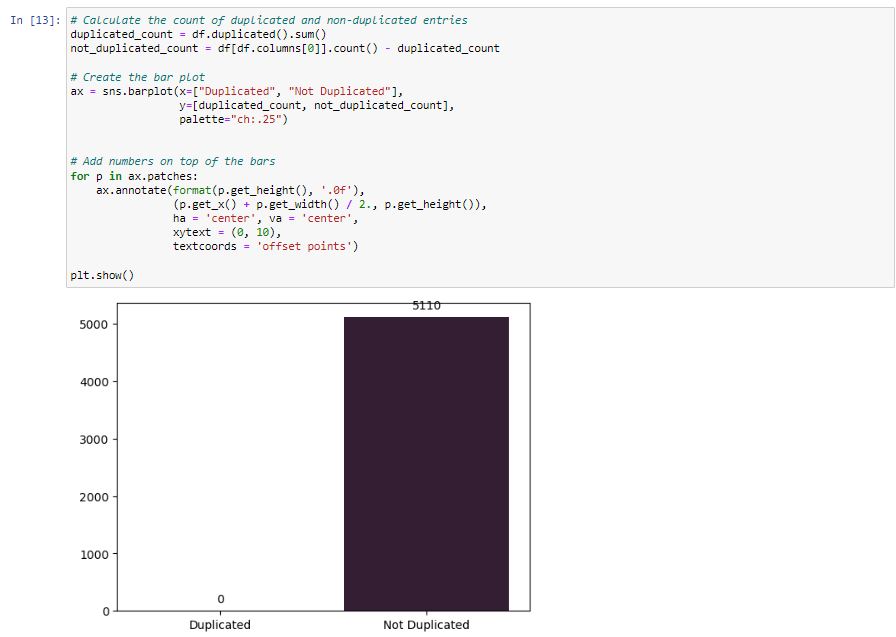
Figure 3.1.1.16 shows the different smoking statuses and their count in the dataset.

### Data Preprocessing

****

**Figure 3.1.2.1**

Figure 3.1.1.4 shows the amount of missing values (NA) in the dataset. We found that there are missing values in the dataset, specifically the bmi column. 201 rows are found with missing bmi.

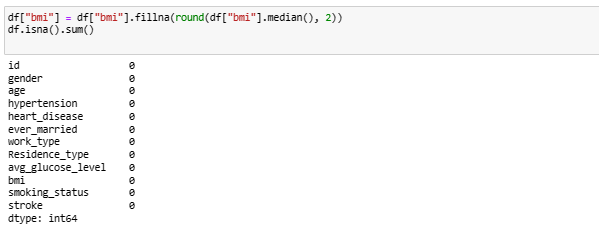


**Figure 3.1.2.2**

Figure 3.1.1.6 shows that there are also no duplicate rows in the dataset. Hence, all the rows in the data are unique.

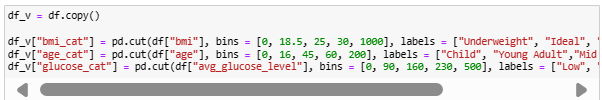
****

**Figure 3.1.2.3**

****

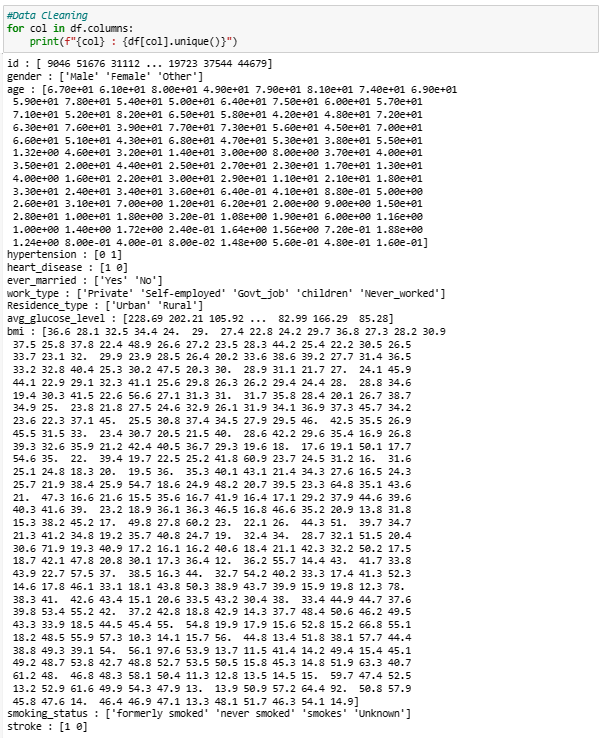
**Figure 3.1.2.4**

Figure 3.1.2.3 and Figure 3.1.2.4 shows the process of resolving the missing values in the dataset. We use the median to fill in the missing values in the bmi column. After filling in the values, there are no more missing values present in the dataset.



**Figure 3.1.2.5**

Figure 3.1.2.5 shows the process of duplicating the dataframe and adding labels for different cut off values. This is used in showing the distribution of data later.

****

**Figure 3.1.2.6**

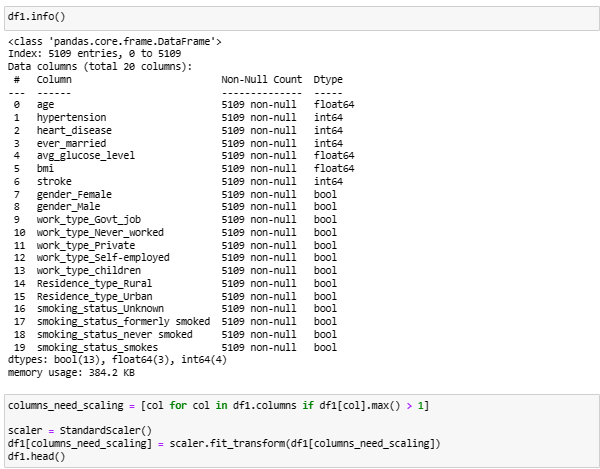
Figure 3.1.2.6 shows the full data that was cleaned, without duplicates and empty values.

#### Feature Engineering

****

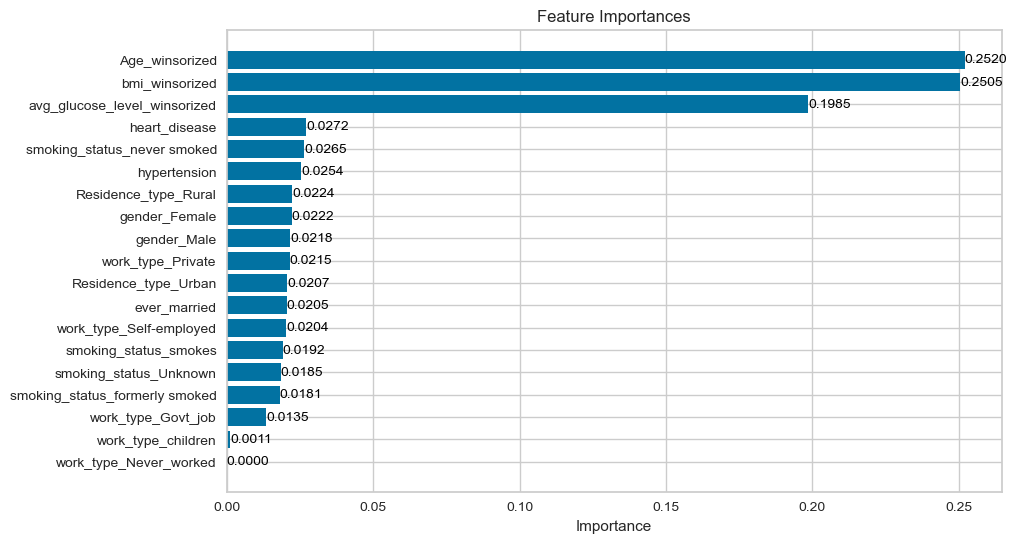
**Figure 3.1.2.1.1**

Figure 3.1.2.1.1 shows various feature engineering methods used. Firstly, the id column is dropped as it is irrelevant to the prediction. Next, we drop the data row containing the 'Other' gender. This is because there is only one row consisting of this value, and thus we want to avoid it affecting the result being biased by it. Finally, we encode the columns that are of categorical values so that they are able to be fed to the models. The ever\_married column is encoded to values of 1 and 0, each representing yes and no respectively, while other columns are encoded by separating them into different columns with boolean values.



**Figure 3.1.2.1.2**

Figure 3.1.2.1.2 shows various data preprocessing methods used. Firstly, the id column is dropped as it is irrelevant to the prediction. Next, we drop the data row containing the 'Other' gender. This is because there is only one row consisting of this value, and thus we want to avoid it affecting the result being biased by it. Finally, we encode the columns that are of categorical values so that they are able to be fed to the models. The ever\_married column is being encoded to values of 1 and 0, each representing yes and no respectively, while other columns are encoded by separating them into different columns with boolean values.

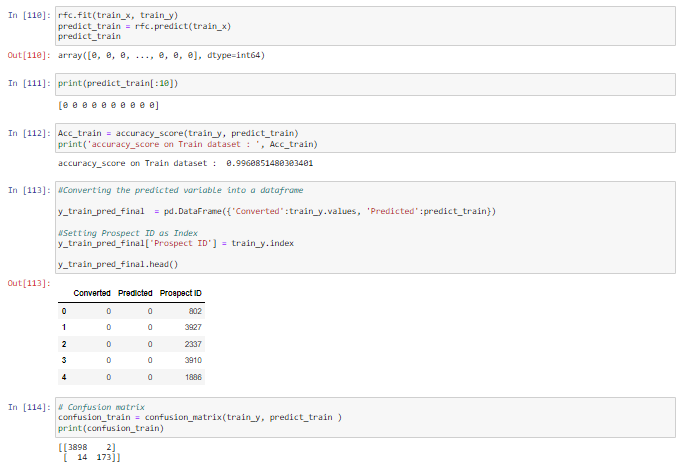


**Figure 3.1.2.1.3**

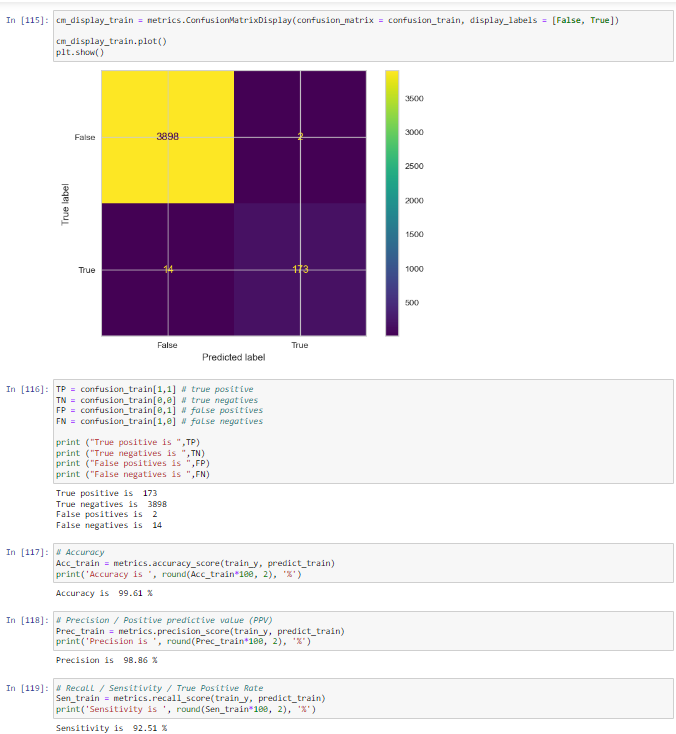
Furthermore, three columns are columns that are of dominant importance, which are illustrated in Figure 3.2.2.1.3. Therefore, we decided to prune off the other columns to see whether it would improve the model.

****

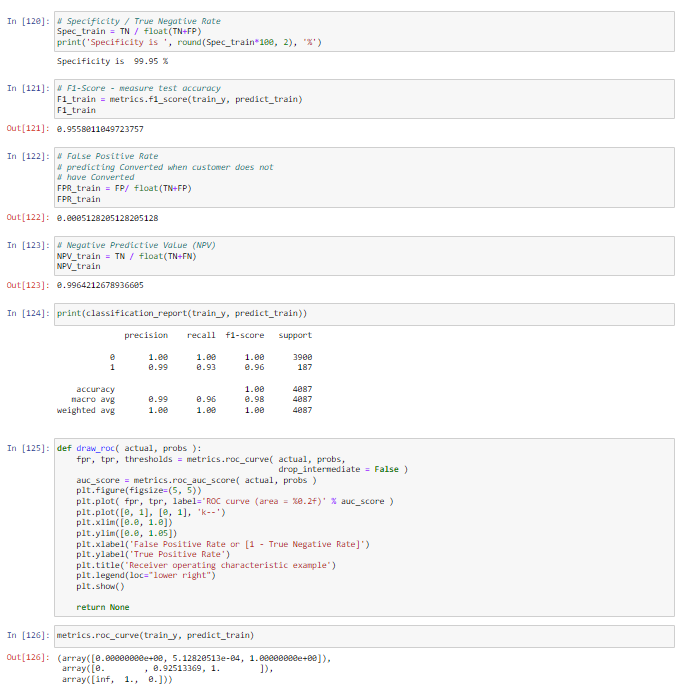
**Figure 3.1.2.1.4**



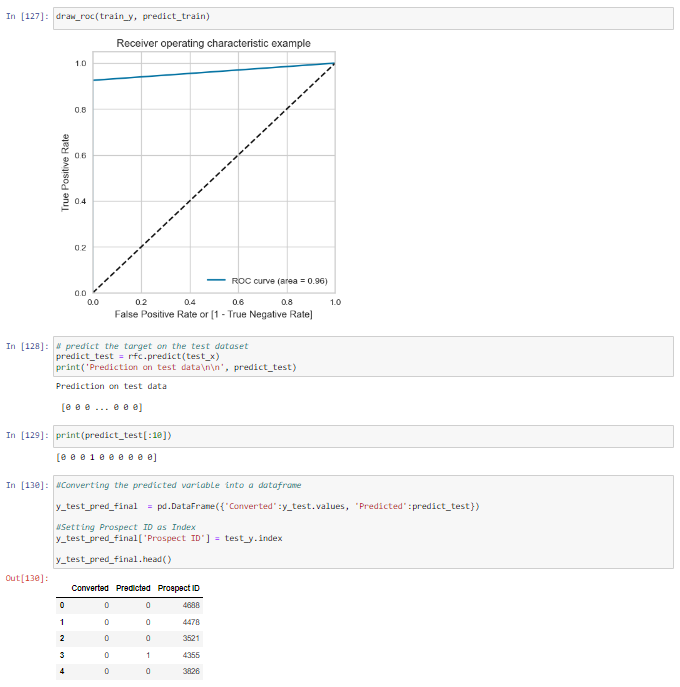
**Figure 3.1.2.1.5**



**Figure 3.1.2.1.6**



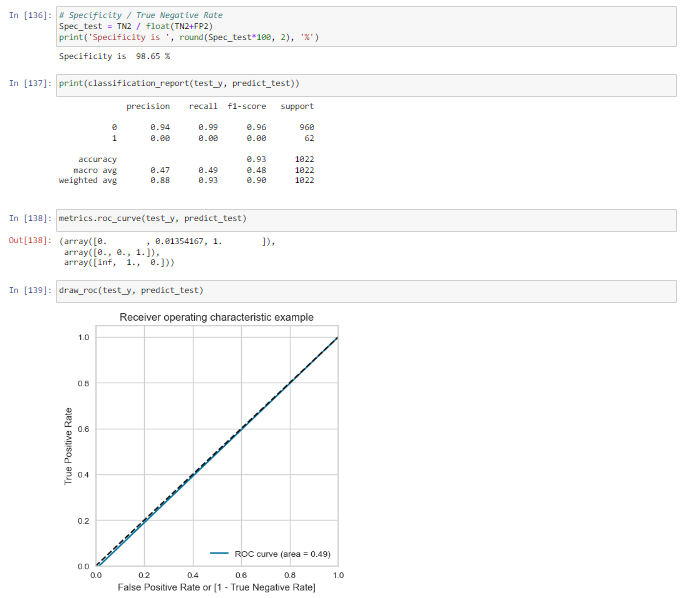
**Figure 3.1.2.1.7**



**Figure 3.1.2.1.8**



**Figure 3.1.2.1.9**



**Figure 3.1.2.1.10**



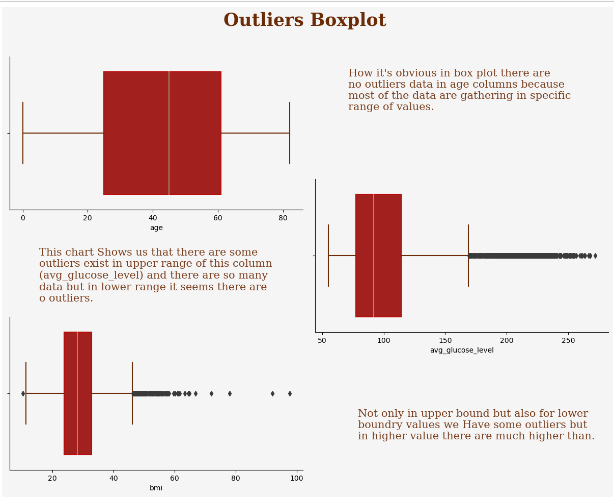
**Figure 3.1.2.1.11**

Figures 3.1.2.1.4 to 3.1.2.1.11 show the process and the results of retraining the random forest model with pruned columns. For the deployment, this model will be used.

#### Detect and Remove Outliers



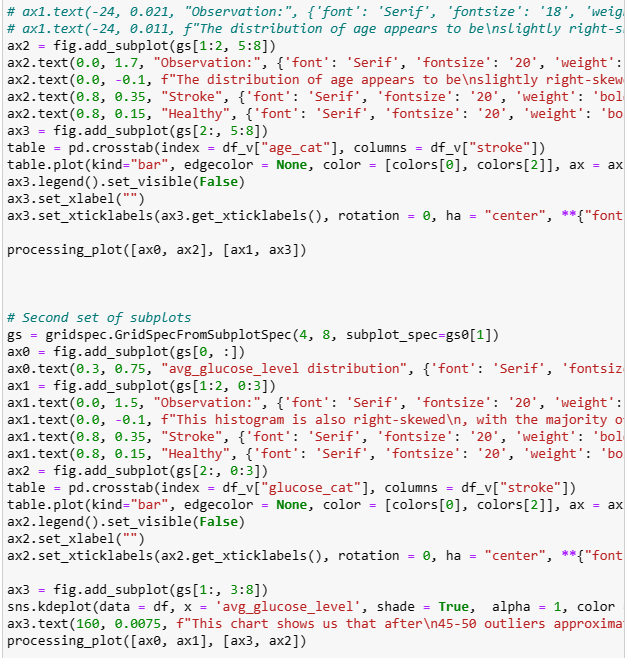
**Figure 3.1.2.2.1**



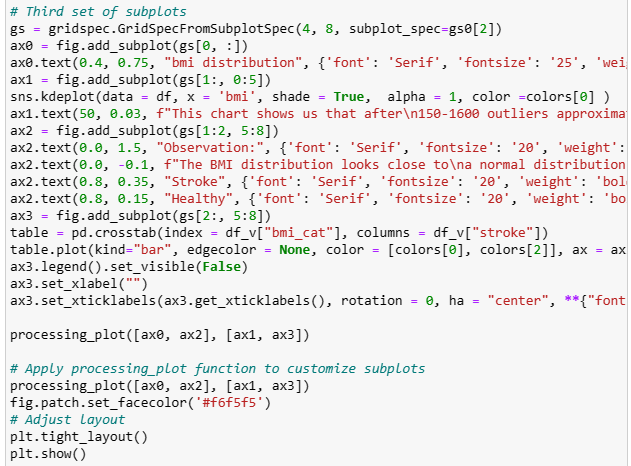
**Figure 3.1.2.2.2**



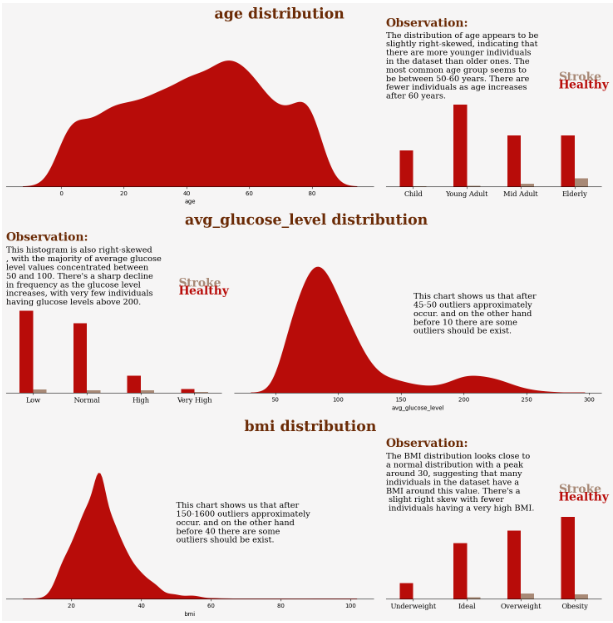
**Figure 3.1.2.2.3**



**Figure 3.1.2.2.4**



**Figure 3.1.2.2.5**



**Figure 3.1.2.2.6**

Figure 3.1.2.2.1 until Figure 3.1.2.2.6 show the outlier detected as having data bias in three columns of data .

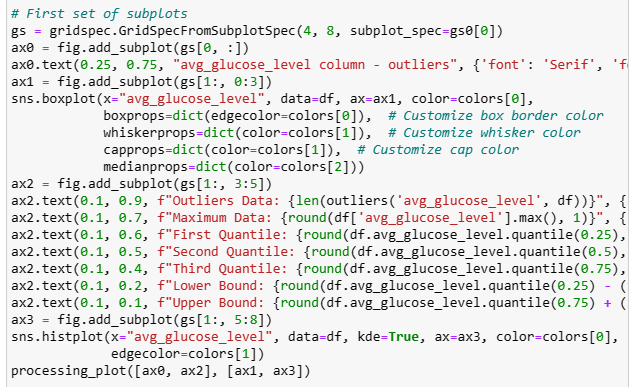


**Figure 3.1.2.2.7**

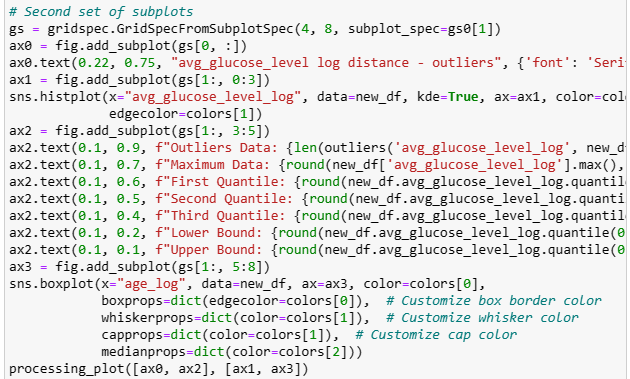
Figure 3.1.2.2.7 shows using Log Normalisation to convert data with a skewed distribution to a nearly normal or more symmetric distribution



**Figure 3.1.2.2.8**



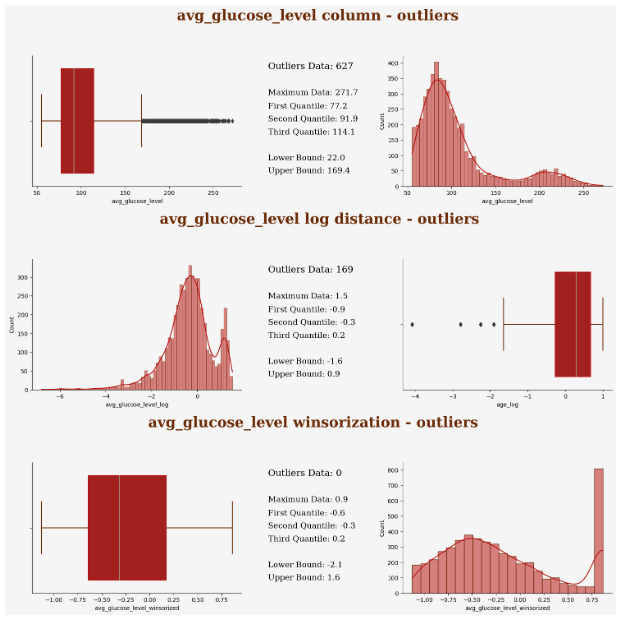
**Figure 3.1.2.2.9**



**Figure 3.1.2.2.10**



**Figure 3.1.2.2.11**

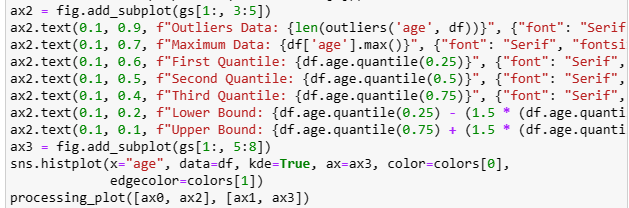


**Figure 3.1.2.2.12**

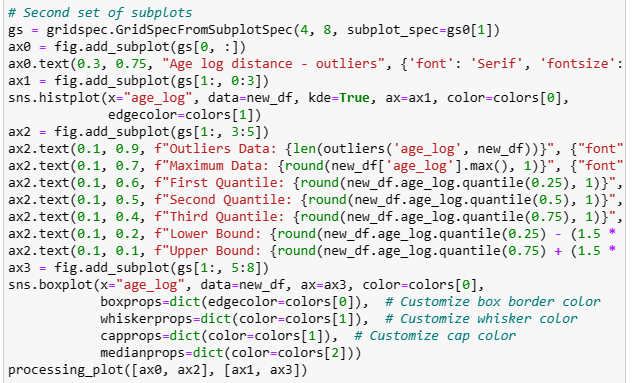
Figure 3.1.2.2.8 until Figure 3.1.2.2.12 shows a process of using winsorization to handle the data bias of average glucose level and its outlier. This technique reduces the impact of outliers on data analysis and modeling by replacing extreme values ​​in the data with the upper and lower bounds of the data. The main idea of ​​Winsorization is to adjust the extreme values ​​of the data (larger or smaller values) to a more reasonable range, rather than completely removing outliers.



**Figure 3.1.2.2.13**



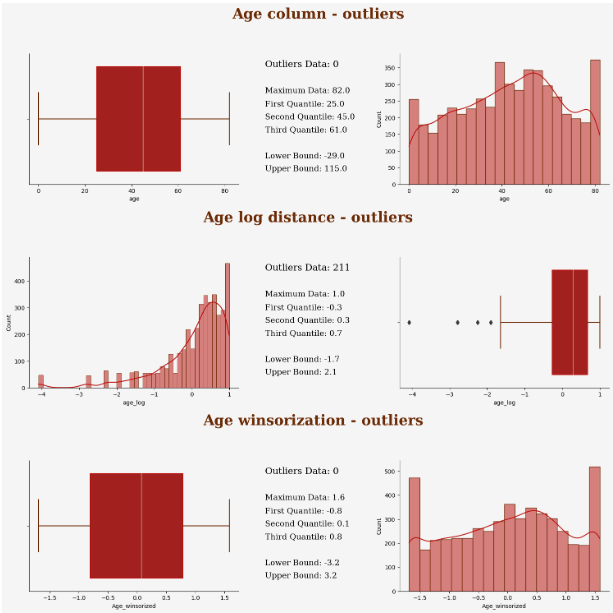
**Figure 3.1.2.2.14**



**Figure 3.1.2.2.15**



**Figure 3.1.2.2.16**

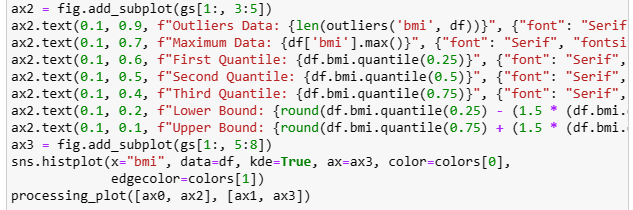


**Figure 3.1.2.2.17**

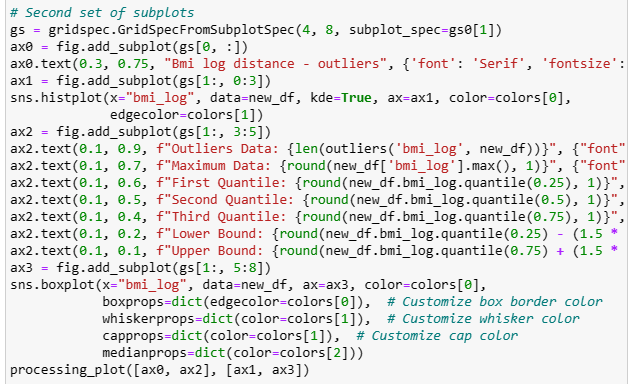
Figure 3.1.2.2.14 until Figure 3.1.2.2.17 shows a process of using winsorization to handle the data bias of age and its outlier.



**Figure 3.1.2.2.18**



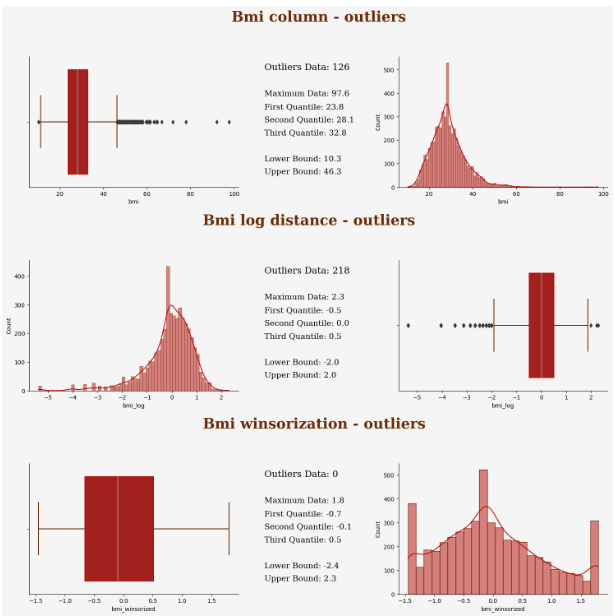
**Figure 3.1.2.2.19**



**Figure 3.1.2.2.20**

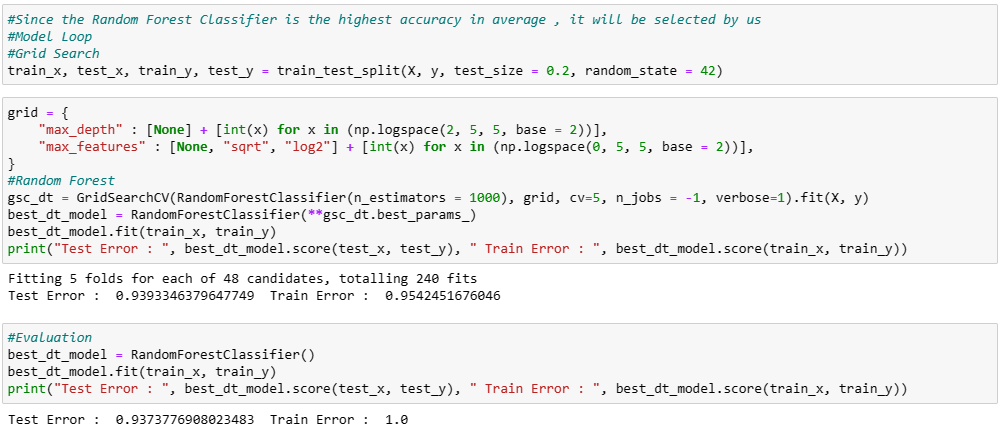


**Figure 3.1.2.2.21**

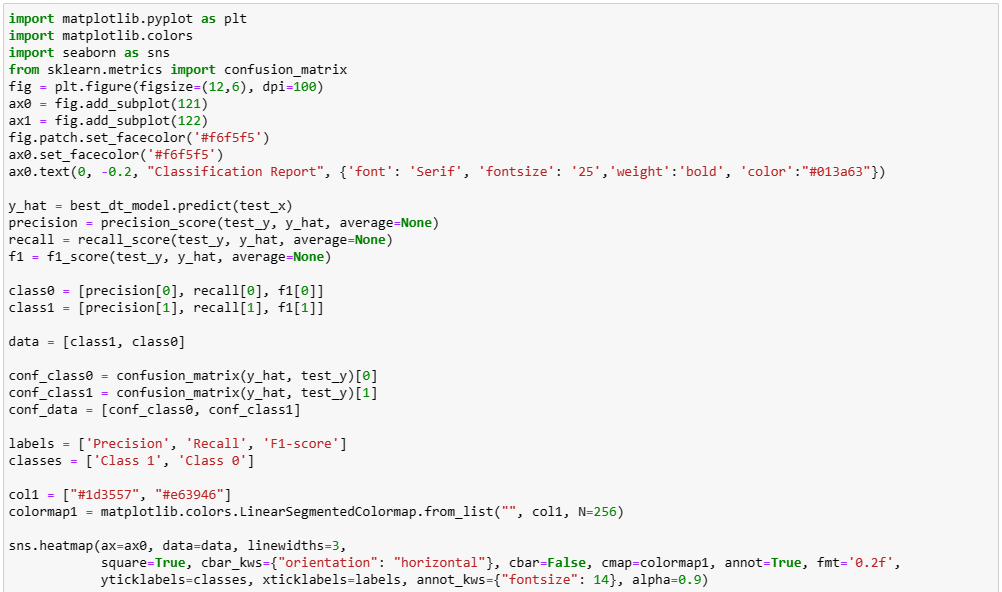


**Figure 3.1.2.2.22**

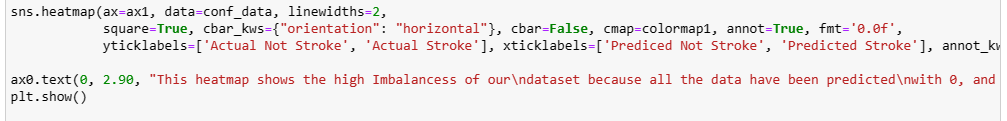
Figure 3.1.2.2.18 until Figure 3.1.2.2.22 shows a process of using winsorization to handle the data bias of age and its outlier.



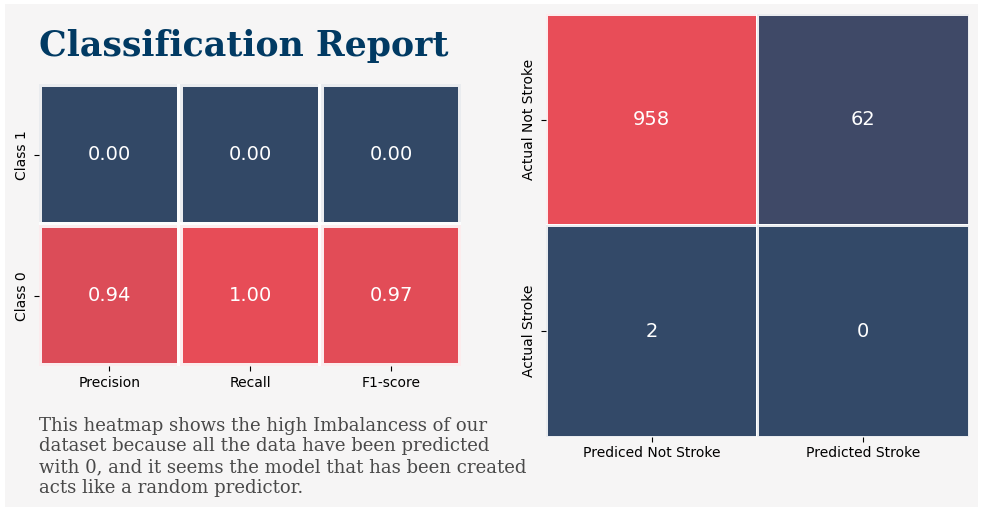
**Figure 3.1.2.2.23**



**Figure 3.1.2.2.24**



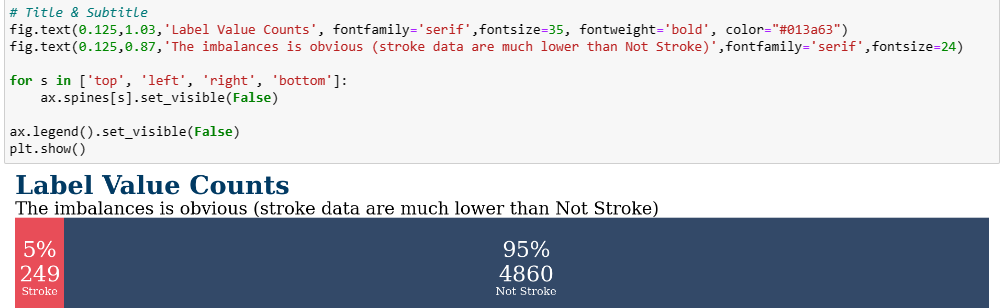
**Figure 3.1.2.2.25**



**Figure 3.1.2.2.26**

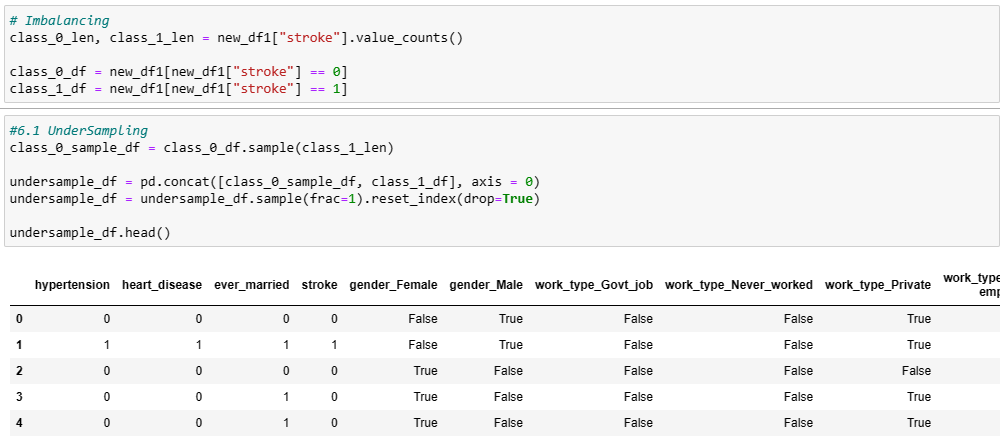


**Figure 3.1.2.2.27**



**Figure 3.1.2.2.28**

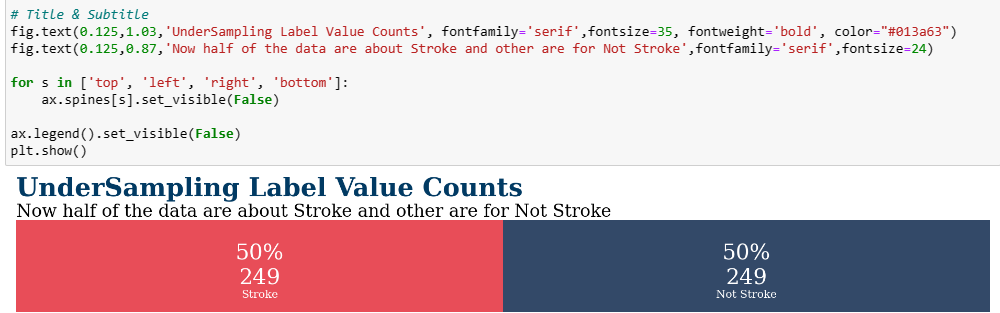
Figure 3.1.2.2.23 until Figure 3.1.2.2.28 showing that an imbalancing stroke data is much lower than not stroke data.



**Figure 3.1.2.2.29**



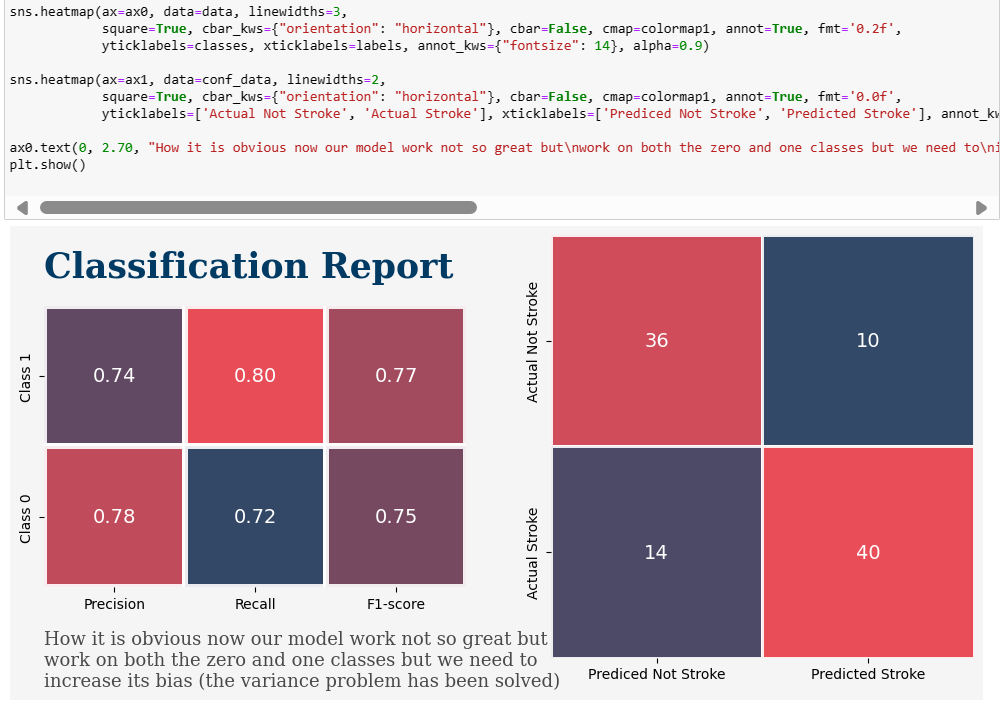
**Figure 3.1.2.2.30**



**Figure 3.1.2.2.31**



**Figure 3.1.2.2.32**



**Figure 3.1.2.2.33**

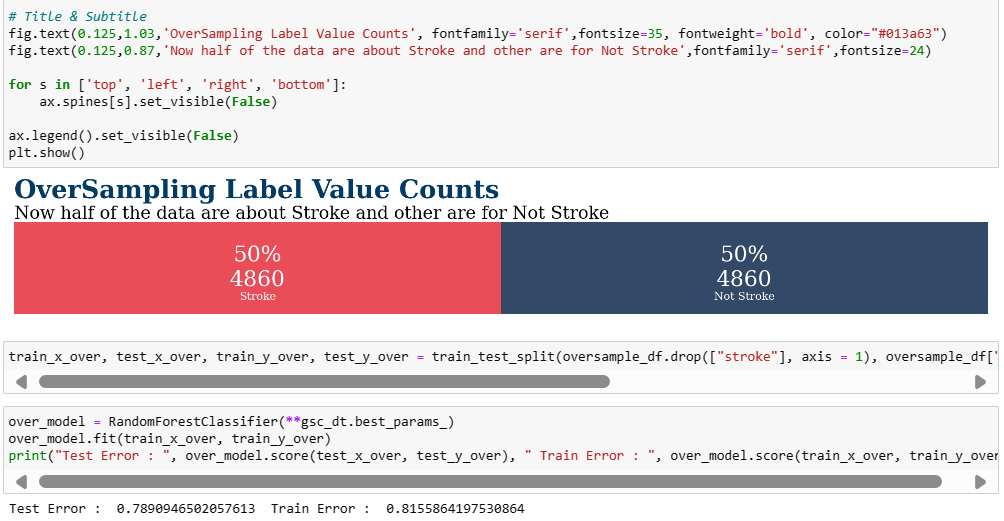
Figure 3.1.2.2.29 until Figure 3.1.2.2.33 is a process of using under sampling to solve the imbalancing but it just solved the variance problem .Undersampling refers to balancing the sample distribution between categories in a dataset by reducing the number of majority class samples. This can be achieved by randomly deleting majority class samples or selecting representative majority class samples. The advantage of undersampling is that it can reduce the size of the dataset, reduce training time and computational cost. It can improve the model's learning effect on majority class samples, thereby improving the model's performance on unbalanced datasets. We are considering adding oversampling method to increase more models performance.



**Figure 3.1.2.2.34**



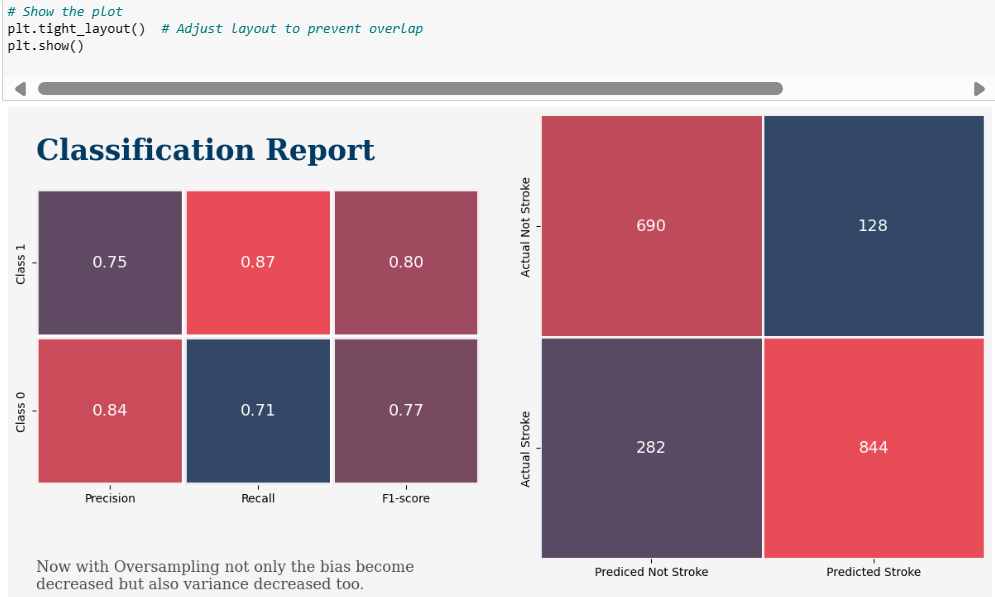
**Figure 3.1.2.2.35**



**Figure 3.1.2.2.36**



**Figure 3.1.2.2.37**



**Figure 3.1.2.2.38**

Figure 3.1.2.2.34 until Figure 3.1.2.2.38 is a process of using oversampling to solve the imbalancing and it does. Oversampling refers to balancing the sample distribution between categories in a dataset by increasing the number of minority class samples. This can be achieved by duplicating minority class samples, synthesizing new minority class samples (such as the SMOTE algorithm), etc. The advantage of oversampling is that it can retain all the information of the original dataset without losing any samples. It can improve the model's ability to learn minority class samples, thereby improving the performance of the model on unbalanced datasets.  
  
In this case, we use these two sampling methods to further improve model performance.

#### Split Data Into Training Set and Testing Set, Normalizing Data



**Figure 3.1.2.3.1**



**Figure 3.1.2.3.2**

Before proceeding to determine the ratio for splitting our dataset into training and testing sets, it's crucial to acknowledge the potential issue of overfitting that arises when feeding all the data into the model for training. Overfitting occurs when a model becomes overly tailored to the training data, rendering it less effective when applied to new datasets. To mitigate this, we must divide our dataset appropriately into training and testing sets. The training set is utilized to feed the model with data, enabling it to learn patterns and relationships within the dataset. Conversely, the testing set is reserved for evaluating the model's performance and making predictions on unseen data.

Traditionally, the dataset is split into an 80-20 ratio between training and testing data, based on established research findings. Hence, we adopt this standard practice and partition our data into two sets: a training set and a testing set. Furthermore, we normalize our data using the MinMaxScaler function in Python. Normalization ensures that each feature's values are scaled to fall within the range of 0 to 1. By normalizing the data, we aim to standardize the range of features, thereby preventing any particular feature from dominating the learning process due to its larger scale. This normalization process contributes to a more balanced and effective training of the model.

* + - 1. Oversampling Processing

****

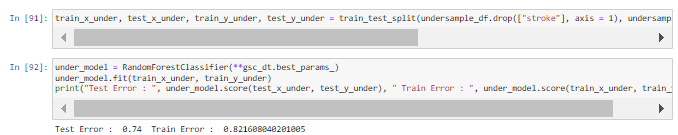
**Figure 3.1.2.4.1**

****

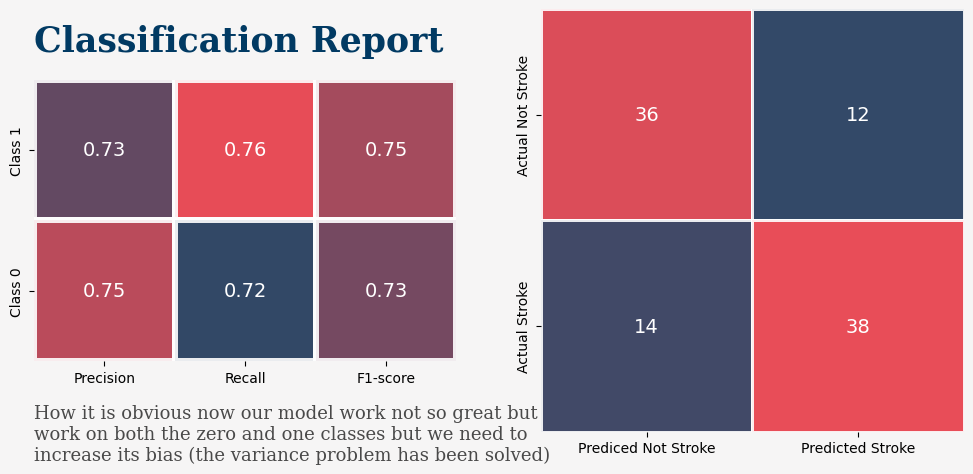
**Figure 3.1.2.4.2**

****

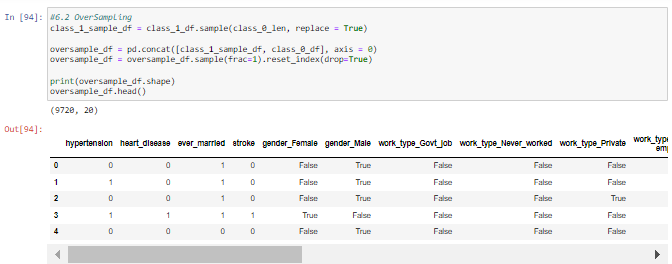
**Figure 3.1.2.4.3**

****

**Figure 3.1.2.4.4**

****

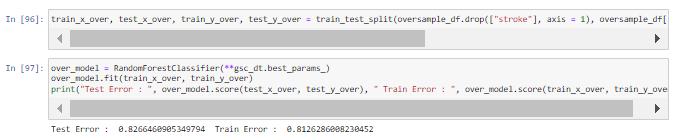
**Figure 3.1.2.4.5**

****

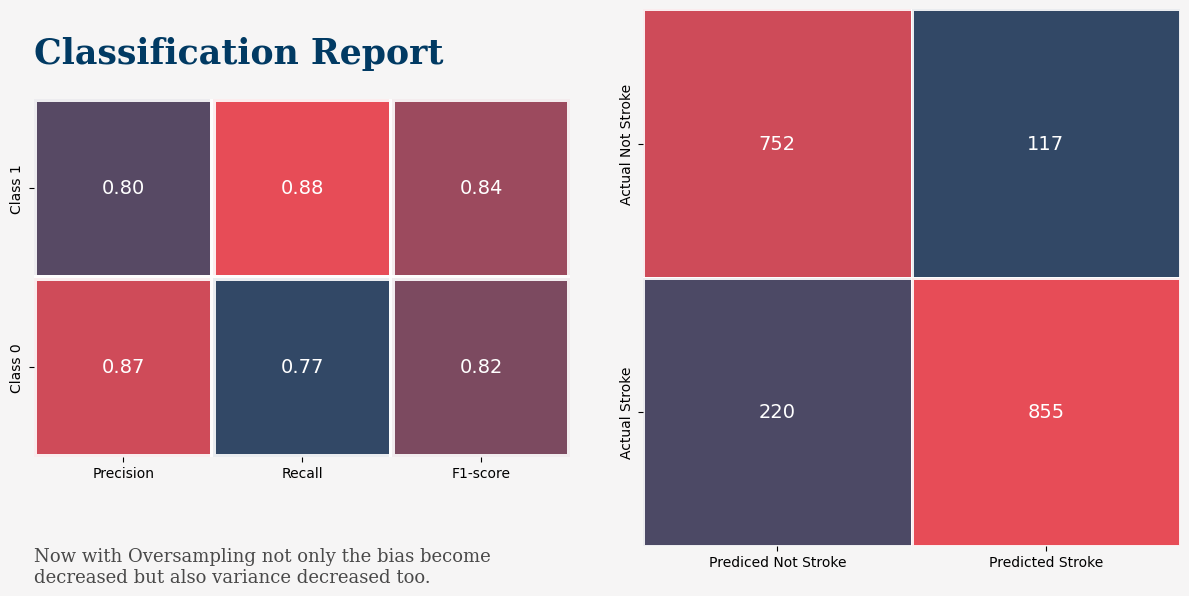
**Figure 3.1.2.4.6**

****

**Figure 3.1.2.4.7**

****

**Figure 3.1.2.4.8**

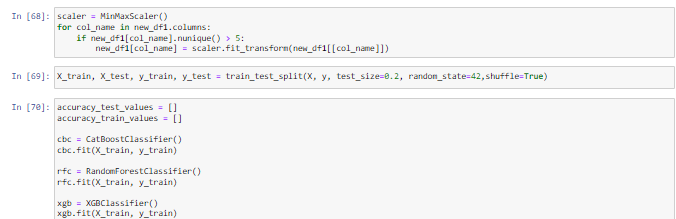
****

**Figure 3.1.2.4.9**

Since our dataset consists of a high imbalance, as stroke is a rare occurrence, we use oversampling to balance out the dataset. Figures 3.1.2.4.1 to 3.1.2.4.9 show the process of applying this method and also their results.

### 

## Applications of the algorithm(s)



### Random Forest

#### Random Forest Hyperparameter Tuning using GridSearchCV

The algorithm is first utilized by selecting the importance from the above test data. Then after that, we will input the data into the algorithm which then we could apply hyperparameter tuning by changing the parameters that are being fed into the algorithm such as our CV which is the number of folds for the K-cross validation and scoring which defines the performance measure of the algorithm. Then we will call the fit function which will provide all the training data and train the model using all hyperparameter combinations and perform cross-validation to evaluate their performance.

#### Check Accuracy on Test Data

We will then create a confusion matrix for both train data and test data to check the accuracy of said data. The Random Forest algorithm after it has been trained with the accuracy of 100% has returned a test accuracy of 93.74%

### CatBoost

#### Setting up the model

To set up CatBoost, we will first need to call the classifier by using CatBoostClassifier(). Then we are to provide two parameters for said function which is iterations which specifies the maximum number of trees that can be built when creating the model. Next is the learning rate which defines the rate of the model learning. After the classifier is called, we will then fit the data obtained from above and fit it into the algorithm.

#### Check Accuracy on Test Data

We will then create a confusion matrix for both train data and test data to check the accuracy of said data. The Gradient Boosting algorithm after it has been trained with the accuracy of 95.94% has returned a test accuracy of 93.84%

### XGBoost

#### Setting up XGBoost model

To utilize this code, first we need to convert our dataset into DMatrix class which in turn can optimize the speed and memory use for effective dataset processing. To use DMatrix, we will then call the DMatrix() function which requires the parameters such as the data and labels. Then we will create the model by calling the function ‘train()’ and providing a few hyper parameters such as the tree depth, learning rate and binary logistic objective. We also need to provide boosting rounds which helps train the model.

#### Check Accuracy on Test Data

We will then create a confusion matrix for both train data and test data to check the accuracy of said data. The XG Boost algorithm after it has been trained with the accuracy of 99.68% has returned a test accuracy of 93.95%

## System flowchart/activity diagram



## Proposed test plan/hypothesis

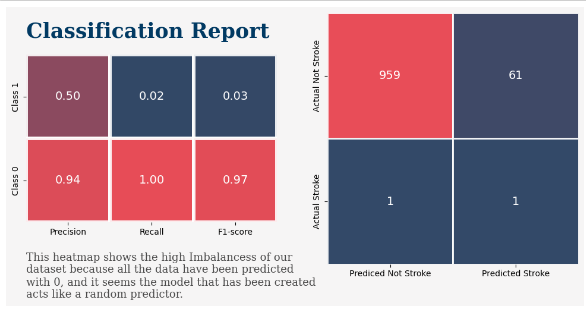
For our test plan, we will compare the accuracy, precision, recall and F1 value of all 3 models and choose the best model out of the three. The system will evaluate them by creating a comparison histogram so that we can compare the results of the three models. After choosing the best model, we will test the systems by inputting said model and its features into deployment to test the best model and acquire its results.

# **Result**

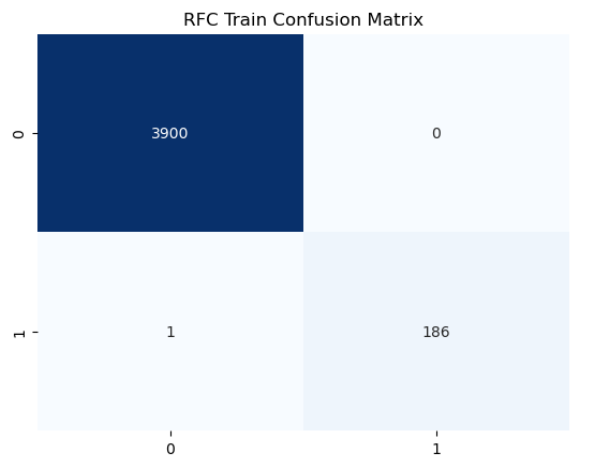
## Results

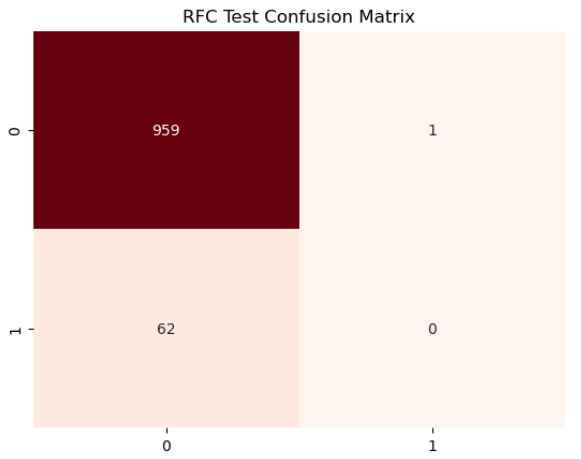
### Random Forest

#### Classification Report for Random Forest



#### Confusion Matrix for Random Forest



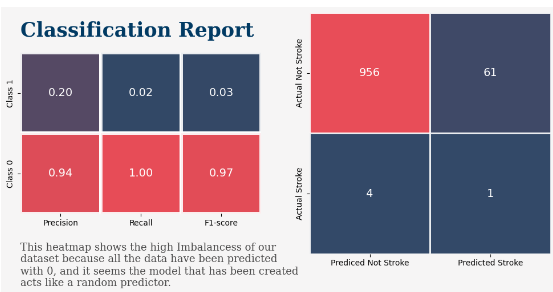


#### Conclusion of Random Forest

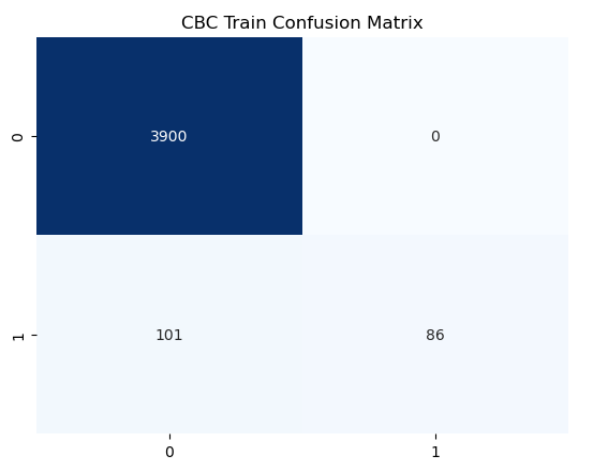
With all the graphs above, we will be able to deduce that the accuracy for the Random Forest algorithm is 93.54%

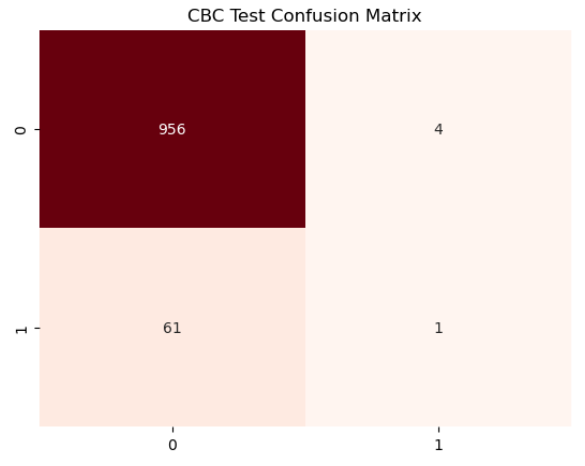
### CatBoostClassifier

#### Classification Report for CatBoostClassifier



#### Confusion Matrix for CatBoostClassifier





#### Conclusion of CatBoostClassifier

With all the graphs above, we will be able to deduce that the accuracy for the CatBoost algorithm is 93.64%.

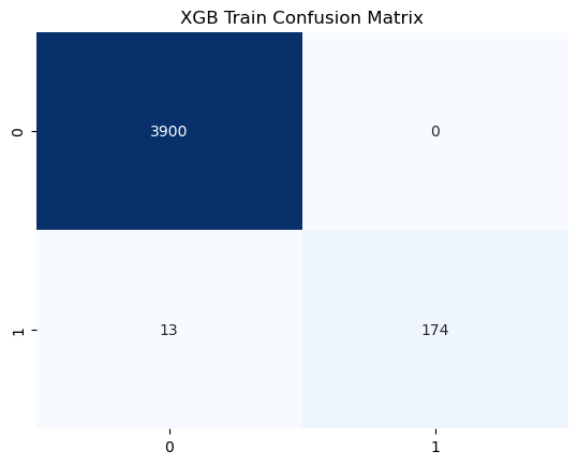
### 

### XGBClassifier

#### Classification report for XGBClassifier



#### Confusion Matrix for XGBClassifier





#### Conclusion of XGBClassifier

With all the graphs above, we will be able to deduce that the accuracy for the XGBoost algorithm is 93.35%.

## Discussion/Interpretation

### Evaluation

#### Accuracy

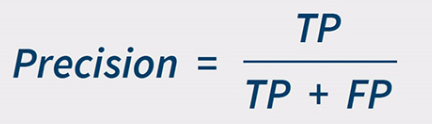


**Figure 4.2.1.1**

Accuracy is a metric of measurement that is used to assess how well the machine learning algorithm performs, the higher the accuracy, the better the model. Figure 4.2.1.1 shows the formula commonly used to calculate the accuracy of an algorithm. As for the accuracy for each of our algorithms, we have compiled it into a table below.

| Algorithm | Accuracy (%) |
| --- | --- |
| Cat Boost | 93.64 |
| Random Forest | 93.84 |
| XGBoost | 93.35 |

#### Precision

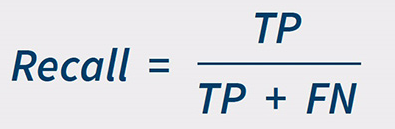


**Figure 4.2.1.2**

Precision describes the ability for the algorithm to correctly predict the data, basically describing the correctness of a model’s positive predictions. And based on that statement, basically high precision means the model is good at identifying the true positive cases and minimizing the number of false positives thus with a higher precision it means that the algorithm is better. Figure 4.2.1.2 shows the formula commonly used to calculate the accuracy of an algorithm. As for the precision for each of our algorithms, we have compiled it into a table below.

| Algorithm | Precision |
| --- | --- |
| Cat Boost | 0.057 |
| Random Forest | 0.075 |
| XGBoost | 0.163 |

#### Recall



**Figure 4.2.1.3**

Recall, or also known as True Positive Rate (TRP) shows the precision in machine learning classification tasks. While precision focuses on the accuracy of positive predictions, recall dives into the completeness of those predictions. Generally, a model with higher recall is a better algorithm. Figure 4.2.1.3 shows the formula used to calculate Recall. As for the recall for each of our algorithms, we have compiled it into a table below.

| Algorithm | Recall |
| --- | --- |
| Cat Boost | 0.008 |
| Random Forest | 0.012 |
| XGBoost | 0.044 |

#### F1-Score

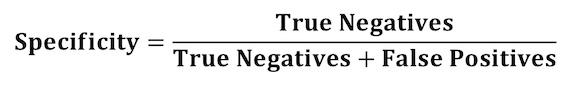


**Figure 4.2.1.4**

F1 score addresses a shortcoming of using just accuracy for evaluating classification models. It combines the strengths of two key metrics which is precision and recall, providing a more balanced view of a model's performance. Figure 4.2.1.4 shows the formula used to calculate F1 score. As for the F1 score for each of our algorithms, we have compiled it into a table below.

| Algorithm | F1 Score |
| --- | --- |
| Cat Boost | 0.014 |
| Random Forest | 0.021 |
| XGBoost | 0.068 |

#### Specificity



**Figure 4.2.1.5**

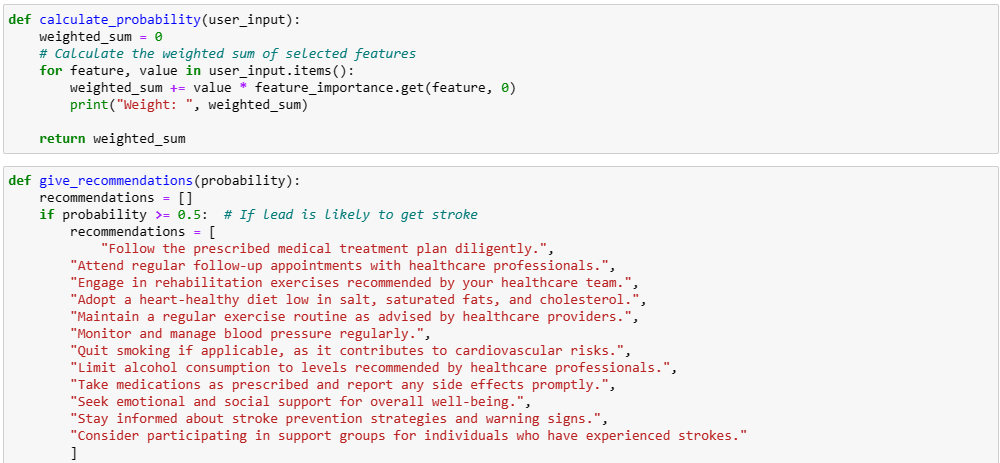
Specificity or False Positive Rate will show the percentage of negative instances are wrongly classified as positive by the algorithm. Hence, a good algorithm should have a low specificity. Figure 4.2.1.5 shows the formula used to calculate specificity. As for the specificity for each of our algorithms, we have compiled it into a table below.

| Algorithm | Specificity |
| --- | --- |
| Cat Boost | 0.2 |
| Random Forest | 0 |
| XGBoost | 0.333 |

### Deployment

#### Selecting the Best Model

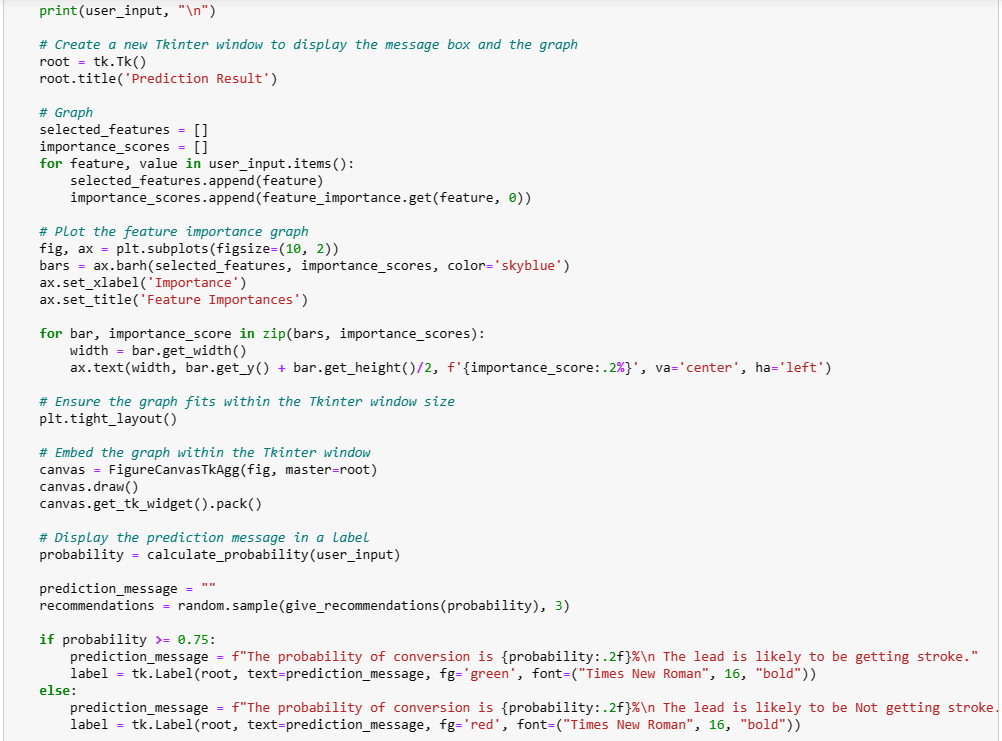
Out of the 3 models we evaluated which are Random Forest, XGBoost, CatBoost and we have decided to use Random Forest model. The reason that we chose Random Forest model is because it has the highest accuracy for both training and testing out of all the algorithms causing it to be highly accurate in our data processing and prediction. Hence, by using the Random Forest algorithm, the users will be able to maintain their health and lifestyle to have a lower risk of having a stroke.



**Figure 4.2.2.1.1**



**Figure 4.2.2.1.2**



**Figure 4.2.2.1.3**



**Figure 4.2.2.1.4**



**Figure 4.2.2.1.5**



**Figure 4.2.2.1.6**

#### Hardware Environment

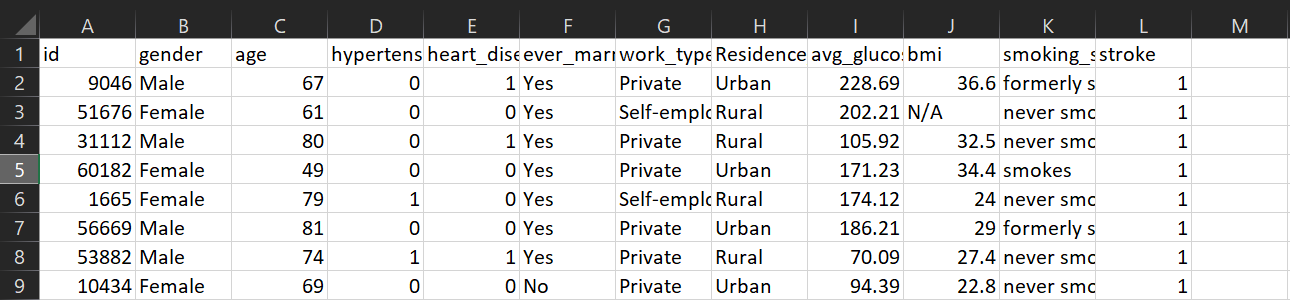
This project is designed to be run on a x86 processor-equipped computer, equipped with at least 8GB of Random Access Memory (RAM) and with a minimum operating system of Windows 10.

#### Software Environment

Jupyter Notebook is being used because this project was created using the Python programming language. Other than that, this project also relies on multiple libraries such as pywaffle, imblearn, missingno, yellowbrick, xgboost, numpy, pandas, seaborn, matplotlib, scikit-learn, catboost. All these libraries are required to be installed before running the project.

#### Dataset (csv or from database)

The dataset is available in the healthcare-dataset-stroke-data.csv from the Kaggle website.



# **Discussion and Conclusion**

## Achievements

Through this project, we aim to provide early detection of stroke before its occurrence, facilitating early intervention and potentially preventing strokes altogether. By doing so, we can reduce the likelihood of long-term complications and improve patient recovery outcomes. Furthermore, by employing machine learning for stroke prediction, this project contributes to advancements in research and development within the field. Leveraging learning models enables the anticipation of possible stroke events, thereby driving progress in stroke prevention strategies. Additionally, integrating this project into healthcare systems can offer automatic stroke risk assessment to the public, enhancing accessibility. Moreover, with its capability to process simple user-provided parameters, the project promotes early detection and preventive measures.

In the project's inception, we delved into data analysis and its significance, uncovering data biases along the way. To mitigate the impact of these biases on data accuracy and fairness, we employed Winsorization to handle outliers, ensuring data balance for more effective predictions. Furthermore, we contrasted the performance of one tree and two gradient boosting classifier models to assess their suitability for the project. Based on data provision and model underlying logic, we opted for the random forest model, which exhibited the highest accuracy. Despite its lower performance in other metrics compared to gradient boosting classifier models, the recursive feature selection behavior and better interpretability of tree-based models influenced our choice.

Subsequently, we identified data imbalance issues and employed data balancing techniques such as oversampling and undersampling. This involved increasing the number of minority class samples and reducing the number of majority class samples to enhance model fairness and mitigate accuracy declines due to data biases. Finally, we developed a user interface where users can input three simple pieces of information for stroke probability prediction. This comprehensive approach ensures that the project not only addresses early detection and prevention but also tackles data biases and fairness concerns for robust and equitable outcomes.

## Limitations and Future Works

#### Limitations

There are a few limitations that are currently present in our system and here we list them out to further discuss the limitations faced. One of the current problems that we face is that our software takes a long time to run due to the complexity of the algorithm and the training resources needed. Other than that, the algorithm that we used, which is Random Forest, does not perform well in extrapolation situations where it means that the algorithm is not good at making predictions outside the observed data range.

#### Future Works

With limitations in the project comes the possibility for future improvement, our project does have a few improvements that can be worked on In the future. Firstly, we hope to further optimize this project so that it is more efficient and easy to run without taking up too many resources and make it so that it will be more accessible to anyone without a high performance computer or without putting too much pressure on the server framework. Secondly, we could work on implementing more data sources that are currently too few to be reliably used such as heart rate or activity levels but in the future if everyone has a wearable health tracker, we might be able to obtain a large amount of data that can be reliably used in our prediction algorithm.

# **Reference & Source**

Tan, K. S., & Venketasubramanian, N. (2022). Stroke burden in Malaysia. Cerebrovascular Diseases. Extra, 12(2), 58–62. https://doi.org/10.1159/000524271

Bhandari, A., 2022. Confusion matrix for Machine Learning. Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/ [Accessed September 19, 2022].

Anon, 2022. Anaconda (python distribution). Wikipedia. Available at: https://en.wikipedia.org/wiki/Anaconda\_(Python\_distribution) [Accessed September 18, 2022].

Koehrsen, W., 2018. Hyperparameter tuning the random forest in python. Medium. Available at: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28 d2aa77dd74 [Accessed September 19, 2022].

Wikimedia Foundation. (2024, April 18). *XGBoost*. Wikipedia. https://en.wikipedia.org/wiki/XGBoost

Stroke Prediction Dataset. (2021, January 26). Kaggle. https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

Wikipedia contributors. (2024b, April 23). CatBoost. Wikipedia. https://en.wikipedia.org/wiki/CatBoost