BIRMINGHAM CITY
University

CMP7161
Advnced Data Science Project
2022–2023

Individual Report

Predictions of a Stroke by using Naïve Bayes Model

Course: Master of Science Advanced Computer Science
Student Name: A
Student Number: -

Table of Contents

1 Report Introduction 4

1.1 Dataset identification 4

1.2 Supervised learning task identification 5

1.3 Team Identification 5

2 Exploratory Data Analysis 5

2.1 Question(s) identification 6

2.2 Splitting the dataset 7

2.3 Exploratory Data Analysis process and results 11

2.4 EDA conclusions 12

3 Experimental Design 13

3.1 Identification of your chosen supervised learning algorithm(s) 14

3.2 Identification of appropriate evaluation techniques 14

3.3 Data cleaning and Pre-processing transformations 14

3.4 Limitations and Options 19

4 Predictive Modelling / Model Development 19

4.1 The predictive modelling processes 19

4.2 Evaluation results on “seen” data 20

5 Evaluation and further modelling improvements 23

5.1 Initial evaluation comparison 23

5.2 Further modelling improvements attempted 24

5.3 Final Evaluation results 29

6 Conclusion 30

6.1 Summary of results 30

6.2 Suggested further improvements to the model development process 30

6.3 Reflection on Research Team 31

6.4 Reflection on Individual Learning 31

7 References 32

8 Appendix 1 34

8.1 Exploratory Data Analysis Process and Results 34

8.1.1 All feature Univariate Analysis 34

8.1.2 Categorical Univariate Analysis of the feature variables 39

8.1.3 Categorical Feature Analysis between feature and target 42

8.1.4 Numerical Feature Analysis between the Feature and stroke 47

8.1.5 Multivariate analysis: feature to feature 49

8.1.6 EDA Feature analysis between features 52

9 Identification of appropriate evaluation techniques........53
10 Iterations to the model........54
10.1.1 SMOTE oversampling to balance the imbalance in the data set........54
10.1.2 Feature Scaling........55
10.1.3 Isolation Forest........55
10.1.4 Removing Residental_type........55
10.1.5 Label Encoding........55

1 Report Introduction

This report explores a data set by building models to provide a first-line stroke diagnosis. To be able to understand the data there will be data analysis it will be done through exploratory data analysis, experiment through different data cleaning processes and pre-process through transformations of the variables contained within the data set and then build a variety of models which will be compared against each to see which one performs better. In addition, there will be various iterations to improve the model to see if the models can be modified further to increase the accuracy of predicting a stroke diagnosis.

Worldwide there are 5.5 million deaths linked to strokes, ranking it the second leading cause of death (Donkor, 2018). A stroke is a "sudden loss of blood flow to an area of the brain" (Phipps and Cronin, 2020) which in turn causes a loss in neurological dysfunction. Studies have found a higher chance of developing a stroke within the south-eastern continents than in western countries due to "early identification of patients with stroke" (Phipps and Cronin, 2020).

1.1 Dataset identification

As a group, we discussed the types of data sets and platforms we wanted to obtain the data. It was a group decision to choose data which was more up-to-date and contained ground truth. We all agreed that we wanted to choose data related to health.

As a group, we then discussed choosing between three data sets:

1. Stroke prediction (Fedesoriano, 2021)
2. Body Fat (Fedesoriano, 2021)
3. Breast Cancer (Yasser, 2022)

We chose the Stroke Prediction data set from Kaggle as it was recent data and had a usability score of 10. The usability score is a feature on Kaggle that makes finding high-quality datasets easier.

The Stroke Prediction Dataset was found on the Kaggle website. Researchers at Jahangrinagar University, Dhaka, Bangladesh, obtained the data within the data set. The patient details were collected from many Bangladesh hospitals, and the dataset contains 12 variables listed in the table below (Emon et al., 2020).

The data set is helpful for researchers in the medical field as it contains all the necessary information to study stroke risk factors. Researchers can use this data set to analyse the risk factors related to stroke and use it to develop better strategies for preventing and treating stroke.

Column Title	Description
Id	Unique identification of a patient
Gender	Male, Female, Other
Age	Ages of patients range between 0-80
Hypertension	1 for having already having hypertension and 0 for not having hypertension - Hypertension is when the blood pressure is high, which puts a strain on the arteries
Heart_disease	1 for having already had heart disease and 0 for not having heart disease- heart disease is a range of conditions that affect the heart
Ever_married	Marital status of the patient: Yes married, No not married
Work_type	What occupation the patient has such as: Self-employed, private. Govt_job, Children

Residence_type	Where each patient lives either Urban or Rural
Avg_glucose_level	The average level of the glucose in the blood of the patient
Bmi	Body Mass Index is based on the height and weight of the patient
Smoking_status	A patient Never_smoked, it is not_known if the patient smoke or not, the patient has quit smoking hence the category being - formerly smoked, patient currently smokes
stroke	1 for having been diagnosed with a stroke and 0 for not having been diagnosed with a stroke

Table 1 Identification of the features of the data set

1.2 Supervised learning task identification

The predictive problem of the population data contained within the dataset that the group aims to solve is to ascertain which variables within the dataset in Table 1 will contribute to a patient being given a first-line stroke diagnosis.

As a group, we have chosen the "stroke" as our selected ground truth value from the data set.

Looking at different supervised learning techniques, such as Artificial Neural Networks (ANNs), classification and linear regression, can be used to build predictive models. However, upon further research, ANNs and regression are unsuitable for this dataset.

ANNs are used to help analyse cause-effect relationships in large complex datasets (Pasini, 2015); therefore, ANNs model is not suitable for the project we are undertaking as the stroke prediction data set, and it is not large or complex.

As for regression, a correlation between one or more independent variables and a response or target variable (Data Mining & Big Data, 2021) is better suited to numerical values, and since the stroke prediction dataset we are using is a mixture of categorical and numerical values regression is not the best suited to our data set.

As for classification supervised learning approach seems better suited as the "classification model reads an input and generates an output that classifies the input into a category" (Fanous, 2021). Therefore, classification is a supervised learning approach used to predict nominal values. Since the target variable "stroke" is a nominal value, as a group, we believe that classification would be the best to use as a supervised learning task to build the predictive model.

1.3 Team Identification

The following table details the members of the research group as well as the initial models that have been chosen to develop for the first iteration of supervised learning model development.

Forename	Surname	Student ID	Model(s) developed
A	-		Compliment Naïve Bayes
B			KNN
C			Decision Tree
D			SVM

Table 2 Team Identification

2 Exploratory Data Analysis

2.1 Question(s) identification

The questions that the group are looking at when conducting the data analysis are:

Question Number	Questions to the data analysis of the dataset
1.	Do older patients have an increased likelihood of being diagnosed with a stroke? The older the patient the more likely they to have a stroke, as stroke is a predominantly disease suffered by the elderly after the age of 65 years (Anatskia, 2011).
2.	Does have hypertension or heart disease increase a patient's likelihood of being diagnosed with a stroke? Hypertension being raised creates pressure on blood flow through the heart. So, if a person has hypertension and they have heart disease where the heart is working harder and then having issues pushing the blood through when they have hypertension, then there is more likely hood they will have a stroke. As such, having heart disease and hypertension, there is more of a chance of having a stroke.
3.	Does a specific BMI, along with high glucose levels, increase a patient's likelihood of being diagnosed with a stroke? Having high Body Mass above 30 and raised glucose levels which can cause diabetes mellitus 2 are likely causes of a person developing a stroke (Boot et al, 2020). UNDERWEIGHT <18,5 NORMAL 18,5 - 24,9 OVERWEIGHT 25 - 29,9 OBESE 30 - 34,9 EXTREMLY OBESE 35< Figure 1 Age and Gender varies when it comes to BMI, (Lawler, 2022)
4.	Does a patient's employment type increase their likelihood of being diagnosed with a stroke? For example, are government employees more stressed and more likely to have a stroke than someone unemployed? As stated by WHO and ILO, working long hours may be a factor in a person having a stroke (Descatha. et al., 2020).

5.	Does being retired increase a patient's likelihood of being diagnosed with a stroke? Patients over 65 who are retired are more likely to have a stroke occur as they have other underlying factors correlated with age, such as high glucose levels (Hosseinzadeh et al., 2021).
6.	Does a high workload increase hypertension, increasing a patient's likelihood of being diagnosed with a stroke? Given that people under 65 years are in the workforce within society, they bear a heavy burden that puts them under much stress and causes stroke (Hosseinzadeh et al., 2021).
7.	Does being unemployed increase BMI levels, increasing a patient's likelihood of being diagnosed with a stroke?
8.	Do female patients have an increased likelihood of being diagnosed with a stroke due to balancing their home and work lives? A higher incidence of stroke was found in females under 35 than in men, which could be related to work or other factors such as autoimmune disorders or oral contraceptives (Boot et al., 2020).

Table 3 Questions to be explored through EDA

The above questions will be explored during the Exploratory Data analysis.

2.2 Splitting the dataset

The data set will be split into three groups, the training set to which EDA will be completed, similar to revising for an exam. The validation is like a mock test; the testing set is unseen data to see if the model works like the final exam.

The technique for splitting the data will be used is random sampling to ensure that the data is balanced across the classes in the dataset. "There is a random_state parameter which will have a seed set to 42" (Kharwal, 2020). Setting the random_state to 42 allows the data to run the exact data for each split, and if not set, it will run different code each time, thus seeing the test data and therefore having data leakage occur.

The dataset is split into three sets: training (80%), validation (10%) and testing (10%). We are using the same data split among our team members so that our exploratory data analysis (EDA) and model development are based on the same training data, and the evaluation is based on the same testing data. This approach helps prevent data leakage and ensures that our results are consistent across different models and analysis.

Original Stroke Data set

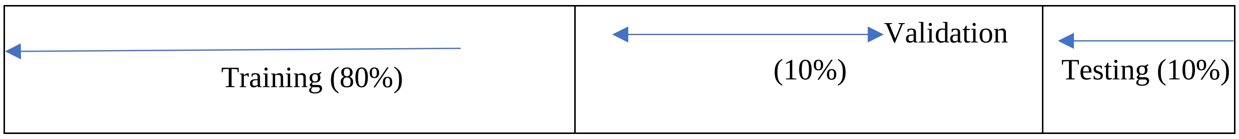
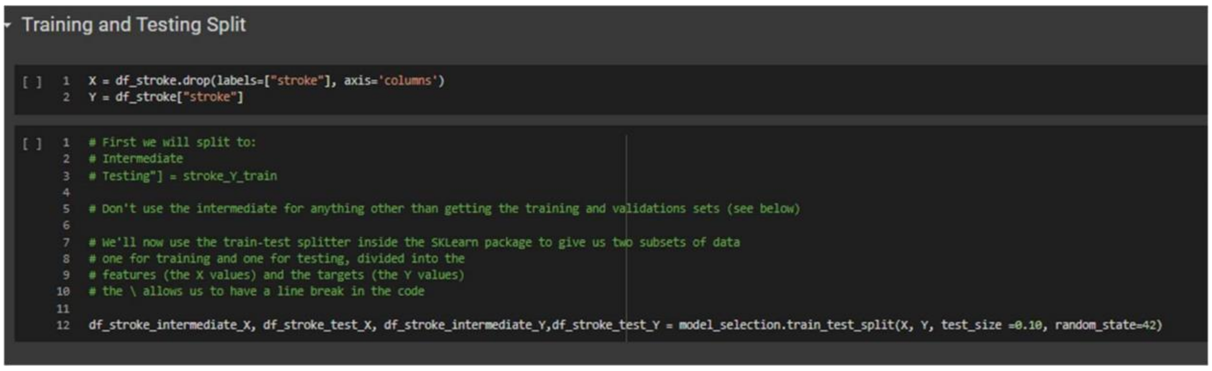


Figure 2 Illustration of data split training (80%), validation (10%) and testing (10%).

We have also ensured that there is no overlap between the three sets. Thus, ensuring that none of the data from the training set is used for validation or testing, and none of the data from the validation set is used for testing. Splitting the data before any analysis helps prevent data leakage and ensures our results are reliable and valid (Werian, 2022).

We have also decided to split the data before the EDA as the test data is then similar to actual world data, which is unseen and not pre-processed, thus removing a bias and giving a realistic accuracy of the models and stopping any occurrence of data leakage. In real life, the test data is patients who have not yet come to the clinics, and in a test situation, it would mean looking at the test paper before the actual test, thereby cheating.



Training and Testing Split

```
X = df_stroke.drop(labels=["stroke"], axis='columns')
Y = df_stroke["stroke"]
```

```
# First we will split to:
# Intermediate
# Testing"] = stroke_Y_train

# Don't use the intermediate for anything other than getting the training and validations sets (see below)

# We'll now use the train-test splitter inside the SKLearn package to give us two subsets of data
# one for training and one for testing, divided into the
# features (the X values) and the targets (the Y values)
# the \ allows us to have a line break in the code

df_stroke_intermediate_X, df_stroke_test_X, df_stroke_intermediate_Y,df_stroke_test_Y = model_selection.train_test_split(X, Y, test_size =0.10, random_state=42)
```

Figure 3 Code showing test and train data split

Figure 3 shows the test, and Intermediate data split, giving intermediate data 90% and the testing data 10%.

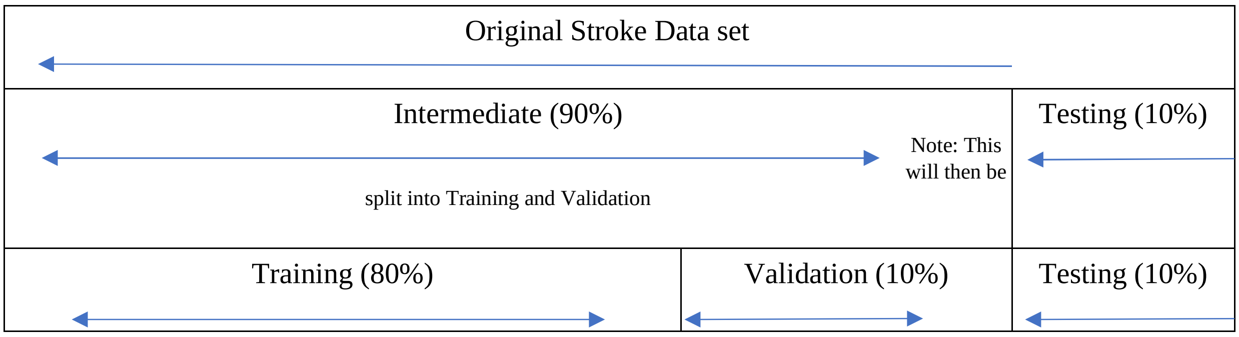
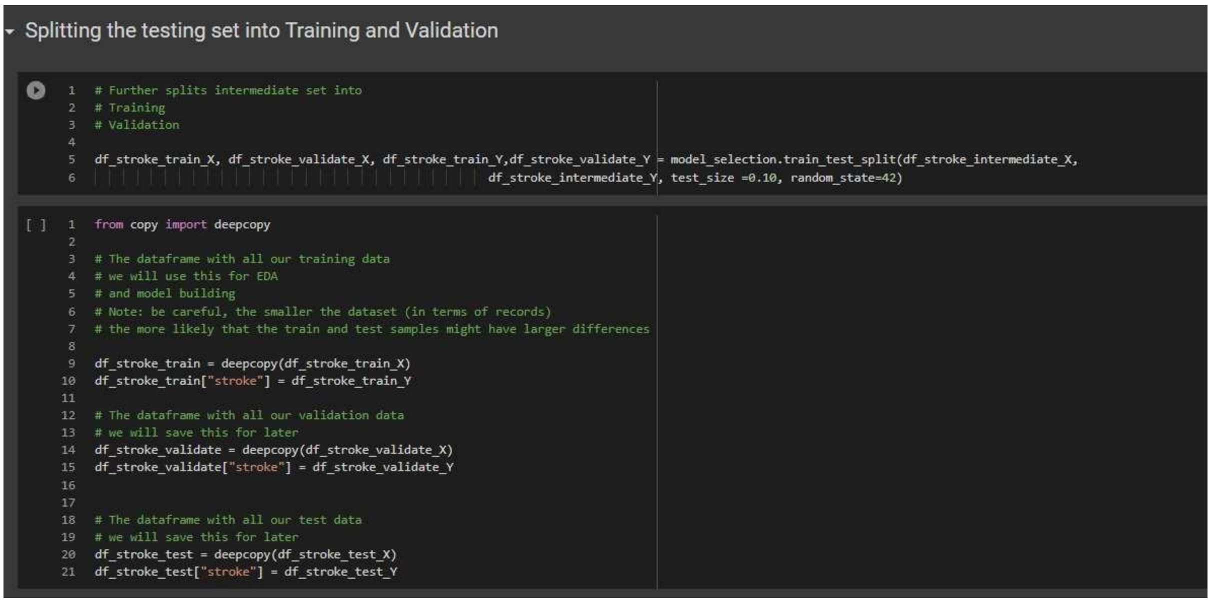


Figure 4 Illustration of how the exact split will take place for the dataset

The intermediate is for the training and validation in code snippet in Figure 5 and an illustration of this is in Figure 4. The 90% intermediate will be split into the training data which will be 80% and the validation as being 10%.

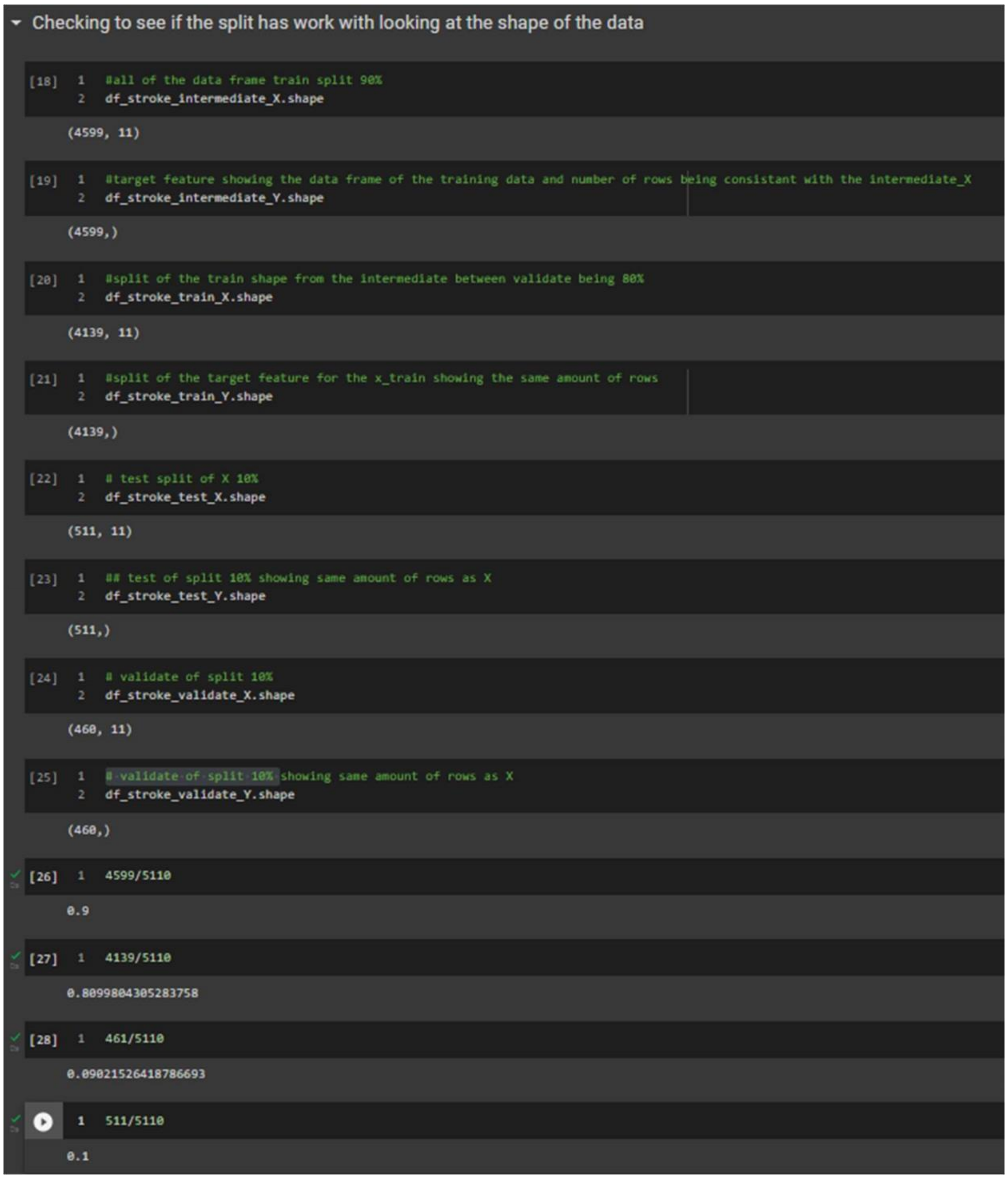

Splitting the testing set into Training and Validation

```
# Further splits intermediate set into
# Training
# Validation

df_stroke_train_X, df_stroke_validate_X, df_stroke_train_Y,df_stroke_validate_Y = model_selection.train_test_split(df_stroke_intermediate_X,
                                                       df_stroke_intermediate_Y, test_size =0.10, random_state=42)
```

```
from copy import deepcopy

# The dataframe with all our training data
# we will use this for EDA
# and model building
# Note: be careful, the smaller the dataset (in terms of records)
# the more likely that the train and test samples might have larger differences

df_stroke_train = deepcopy(df_stroke_train_X)
df_stroke_train["stroke"] = df_stroke_train_Y

# The dataframe with all our validation data
# we will save this for later
df_stroke_validate = deepcopy(df_stroke_validate_X)
df_stroke_validate["stroke"] = df_stroke_validate_Y


# The dataframe with all our test data
# we will save this for later
df_stroke_test = deepcopy(df_stroke_test_X)
df_stroke_test["stroke"] = df_stroke_test_Y
```

Figure 5 training and validation split

I have then checked the shape of the split to ensure that is it is now 80% training 10% testing and 10% validation.



```
Checking to see if the split has work with looking at the shape of the data

[18]  #all of the data frame train split 90%
      df_stroke_intermediate_X.shape

      (4599, 11)

[19]  #target feature showing the data frame of the training data and number of rows being consistant with the intermediate_X
      df_stroke_intermediate_Y.shape

      (4599,)

[20]  #split of the train shape from the intermediate between validate being 80%
      df_stroke_train_X.shape

      (4139, 11)

[21]  #split of the target feature for the x_train showing the same amount of rows
      df_stroke_train_Y.shape

      (4139,)

[22]  # test split of X 10%
      df_stroke_test_X.shape

      (511, 11)

[23]  ## test of split 10% showing same amount of rows as X
      df_stroke_test_Y.shape

      (511,)

[24]  # validate of split 10%
      df_stroke_validate_X.shape

      (460, 11)

[25]  # validate of split 10% showing same amount of rows as X
      df_stroke_validate_Y.shape

      (460,)

[26]  4599/5110

      0.9

[27]  4139/5110

      0.8099804305283758

[28]  461/5110

      0.09021526418786693

      511/5110

      0.1
```

Figure 6 checking the shapes of the split to ensure it is correct

Figure 6 demonstrates testing the data split to ensure it has been done correctly. As seen above, the shape of the train, test and validation is correct as per the group decision to ensure that the EDA, training, and model development are done with the same data split to ensure that we can evaluate our models and compare them against each other.

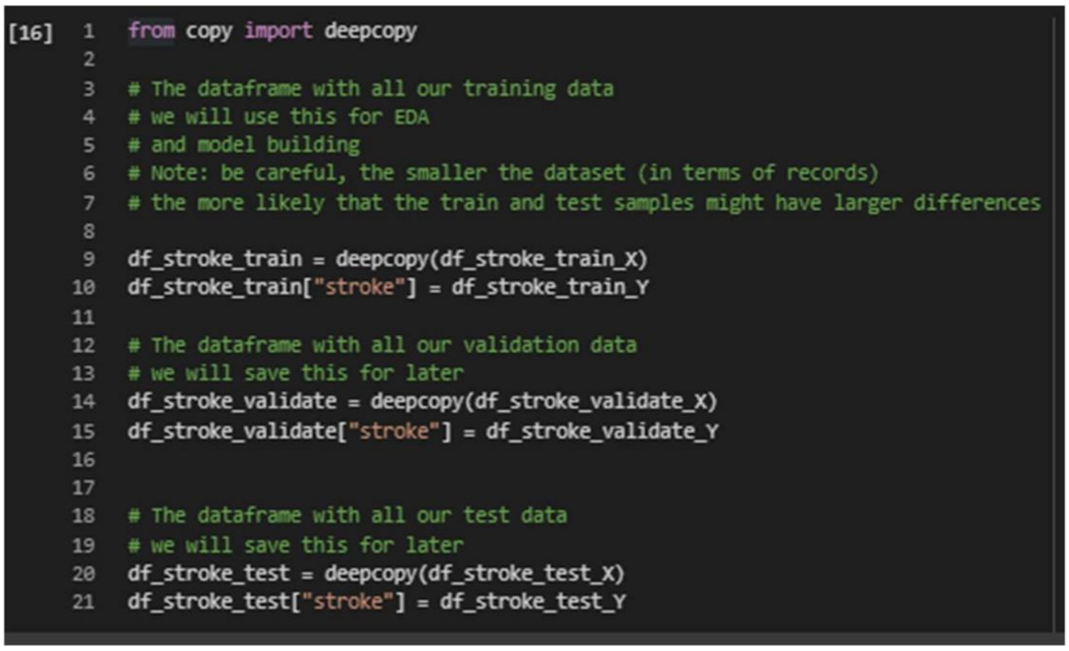
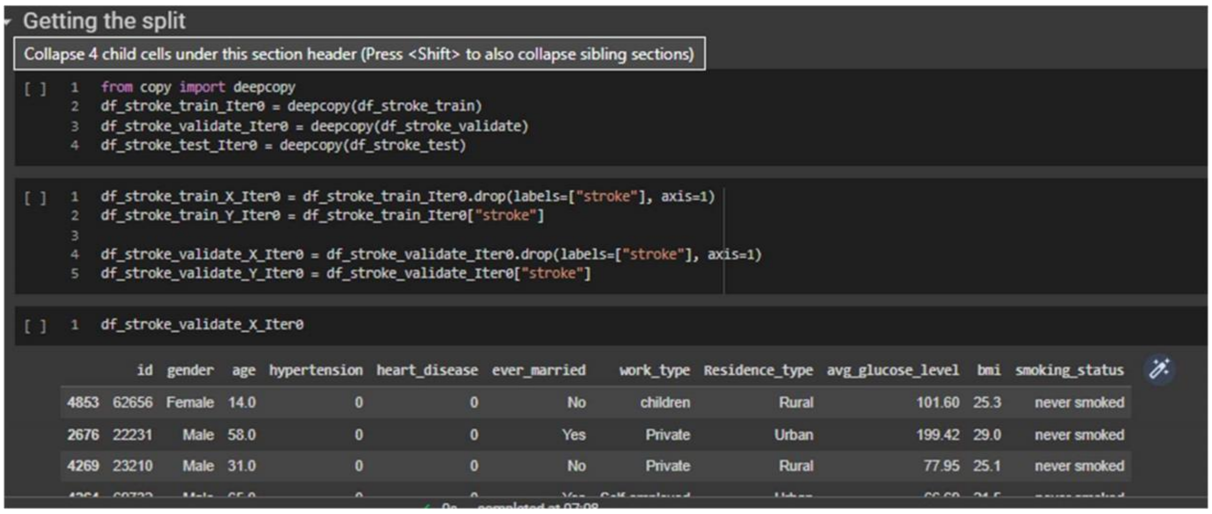

```
[16]
from copy import deepcopy

# The dataframe with all our training data
# we will use this for EDA
# and model building
# Note: be careful, the smaller the dataset (in terms of records)
# the more likely that the train and test samples might have larger differences

df_stroke_train = deepcopy(df_stroke_train_X)
df_stroke_train["stroke"] = df_stroke_train_Y

# The dataframe with all our validation data
# we will save this for later
df_stroke_validate = deepcopy(df_stroke_validate_X)
df_stroke_validate["stroke"] = df_stroke_validate_Y


# The dataframe with all our test data
# we will save this for later
df_stroke_test = deepcopy(df_stroke_test_X)
df_stroke_test["stroke"] = df_stroke_test_Y
```

Figure 7 Data frames deep copies for the training, testing and validation

In Figure 7, a deep copy of the training, validation and testing has been created. The reason for using the deep copy function is that it is independent of the original dataset. Therefore, anything changed within the deep copy does not change the original dataset (Balaji, 2022).


Getting the split

Collapse 4 child cells under this section header (Press <Shift> to also collapse sibling sections)

```
[ ]
from copy import deepcopy
df_stroke_train_Iter0 = deepcopy(df_stroke_train)
df_stroke_validate_Iter0 = deepcopy(df_stroke_validate)
df_stroke_test_Iter0 = deepcopy(df_stroke_test)
```

```
[ ]
df_stroke_train_X_Iter0 = df_stroke_train_Iter0.drop(labels=["stroke"], axis=1)
df_stroke_train_Y_Iter0 = df_stroke_train_Iter0["stroke"]

df_stroke_validate_X_Iter0 = df_stroke_validate_Iter0.drop(labels=["stroke"], axis=1)
df_stroke_validate_Y_Iter0 = df_stroke_validate_Iter0["stroke"]
```

```
[ ]
df_stroke_validate_X_Iter0
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
4853	62656	Female	14.0	0	0	No	children	Rural	101.60	25.3	never smoked
2676	22231	Male	58.0	0	0	Yes	Private	Urban	199.42	29.0	never smoked
4269	23210	Male	31.0	0	0	No	Private	Rural	77.95	25.1	never smoked

Figure 8 Deep copy of the train, validate and test to enable various iterations to run

Figure 8 code snippet shows the deep copy of the train, validation, and test will allow the data to be tested and, with further iterations, enable the same data to be kept as the split, and, therefore, a like-for-like comparison can be made.

2.3 Exploratory Data Analysis process and results

Exploratory data analysis (EDA) is an essential step in the data science process. It is a process of analysing data sets to summarise their main characteristics and get a better insight (Agrawal et al., 2021), often with visual representations. EDA aims to find patterns, relationships and errors in data that can be used to inform decision-making and create predictive models.

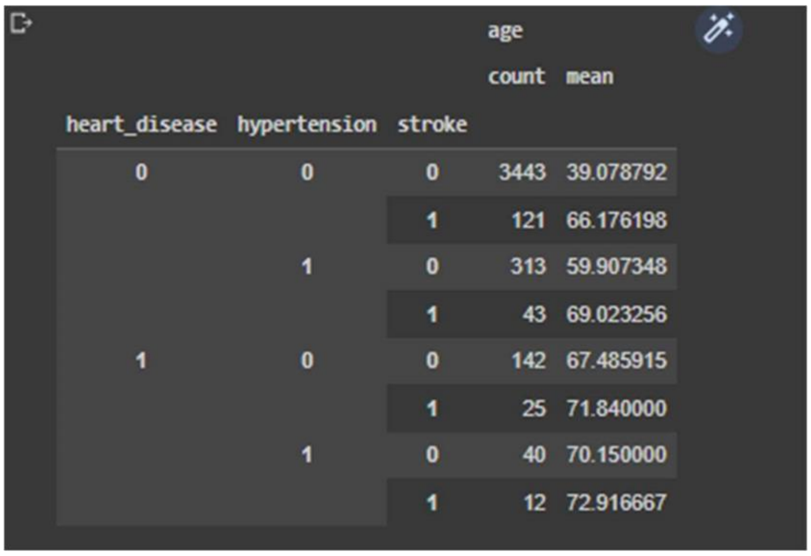
Complete Analysis is in Appendix 1- 8.1

2.4 EDA conclusions

Concluding the EDA I have come to the following conclusions in relation to the questions identification section:

Question 1: Throughout the EDA and in Figure 35, there is positive proof that the assumption of age plays a part in a stroke diagnosis, with the mean age being around 67 and above.

Question 2: Hypertension or heart disease were highly imbalanced; this was expected as there would not be a vast number of patients with either heart disease or hypertension. However, compared to stroke, it is concluded that from this data set, they do not correlate highly to having a stroke and are different than what was expected before the EDA was conducted, as seen in the figure below.



			age	
			count	mean
heart_disease	hypertension	stroke		
0	0	0	3443	39.078792
		1	121	66.176198
	1	0	313	59.907348
		1	43	69.023256
1	0	0	142	67.485915
		1	25	71.840000
	1	0	40	70.150000
		1	12	72.916667

Figure 9 comparison of age with hypertension, heart disease and stroke

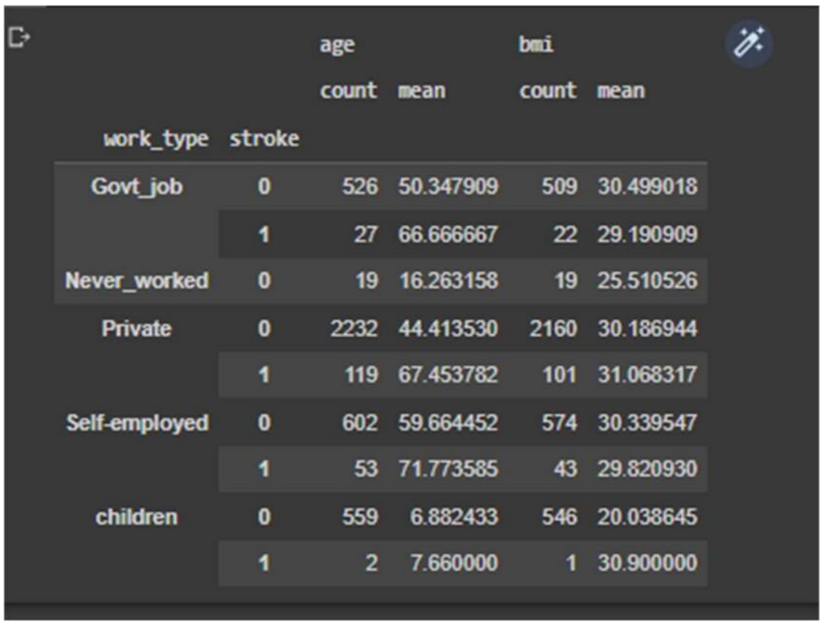
Question 3: Having high glucose levels is a contributory factor of having a stroke but having a high BMI did not significantly influence a stroke diagnosis. Upon looking at the heat map, neither glucose levels nor BMI correlated positively to a stroke diagnosis.

Question 4: - The type of employment a patient has does not play a significant role in the patient having a stroke but working privately does increase the chance of having a stroke the not working at all, which can be seen in Figure 72.

Question 5: Throughout the EDA, it is proven that a patient over 60 is more likely to be diagnosed with a stroke.

Question 6: Since an employed patient's average age is below 60, as seen in Figure 72, there is no correlation between being diagnosed with a stroke for this set of patients even though they contribute to most of the workforce.

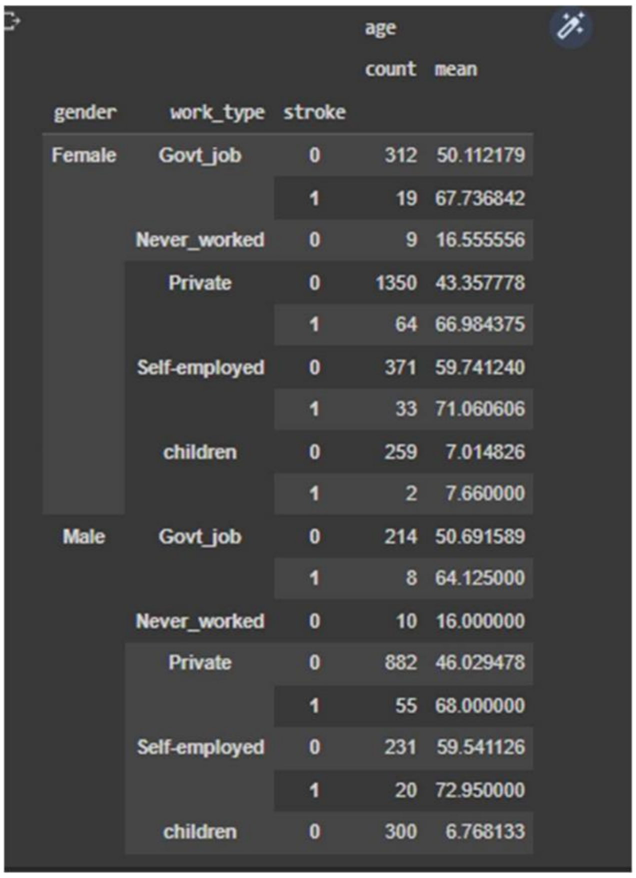
Question 7: this assumption is incorrect as the never_worked was adolescents with a mean age of 16 and not an increased BMI, as seen in the figure below.



		age		bmi	
		count	mean	count	mean
work_type	stroke				
Govt_job	0	526	50.347909	509	30.499018
	1	27	66.666667	22	29.190909
Never_worked	0	19	16.263158	19	25.510526
Private	0	2232	44.413530	2160	30.186944
	1	119	67.453782	101	31.068317
Self-employed	0	602	59.664452	574	30.339547
	1	53	71.773585	43	29.820930
children	0	559	6.882433	546	20.038645
	1	2	7.660000	1	30.900000

Figure 10 BMI, age, Stroke and work_type comparison

Question 8: From the EDA, female patients are more likely to suffer from a stroke than males, but it is impossible to tell what the factors of this may cause this. However, those that are self-employed are more likely to suffer a stroke than those females that have a government job, as can be seen in the figure below.



			age	
			count	mean
gender	work_type	stroke		
Female	Govt_job	0	312	50.112179
		1	19	67.736842
	Never_worked	0	9	16.555556
	Private	0	1350	43.357778
		1	64	66.984375
	Self-employed	0	371	59.741240
		1	33	71.060606
	children	0	259	7.014826
		1	2	7.660000
Male	Govt_job	0	214	50.691589
		1	8	64.125000
	Never_worked	0	10	16.000000
	Private	0	882	46.029478
		1	55	68.000000
	Self-employed	0	231	59.541126
		1	20	72.950000
	children	0	300	6.768133

Figure 11 correlation between gender, work_type, stroke and age

3 Experimental Design

3.1 Identification of your chosen supervised learning algorithm(s)

As a group, we have chosen to look at the problem with our data set as classification. Naïve Bayes is a supervised learning algorithm commonly used for classification tasks. It is based on Bayes' theorem, which states that the probability of an event occurring is equal to the probability of the event given the prior knowledge multiplied by the prior probability of the event.

Therefore, having researched the Naïve Bayes Algorithms, built on the Bayesian theorem "where the data sets features and functions" (Perez and Perez, 2021) are classified independently. This algorithm is based on the assumption that all the features in a dataset are independent of each other, known as the "Naïve" assumption.

The equation for the theorem is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Equation 1 Naïve Bayes Probability Formula

For example, the theorem states the probability of A occurring given that B has occurred. Therefore, within the data set, an example would be the probability of a stroke, given that the patient has hypertension.
To apply this to the data set stroke prediction:

$$P(x|Y) = \frac{P(X|y)P(y)}{P(X)}$$

Equation 2 Probability of a stroke (Y) given the patient has hypertension (X)

In this case, Y is the target variable ‘stroke’ and X is the dependent feature vector (of size n) where:

$$x = (x\ , x\ \ldots\ )$$

Equation 3 dependant feature vector

Compliment Naïve Bayes is an extension of Naïve Bayes, which accounts for the imbalanced class labels in the data set. This algorithm is beneficial because the dataset is highly imbalanced, with most patients not having a stroke. Compliment Naïve Bayes can help accurately classify those who have a stroke, even when there is an imbalance in the data.

3.2 Identification of appropriate evaluation techniques

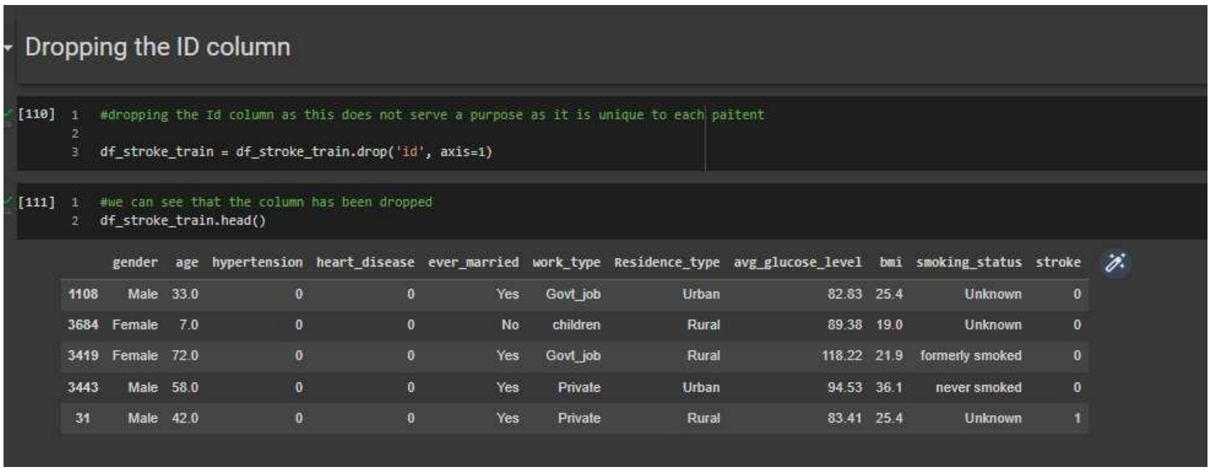
The evaluation metrics as a group we have chosen include Accuracy, F1 Score, Precision, Confusion Metrics, and Recall. However, Accuracy sometimes is misleading when the data set is unbalanced, which is why the other metrics are important.

See full explanation in appendix 1 – 9 Identification of appropriate evaluation techniques

3.3 Data cleaning and Pre-processing transformations

The following are the techniques applied to the data set during data cleaning and pre-processing transformation.

The ID column, which only identified the patient and gave no real value to whether a patient had a stroke as was discovered in the heat map, was dropped in the pre-processing stage as this column created unnecessary noise Figure 12.



Dropping the ID column

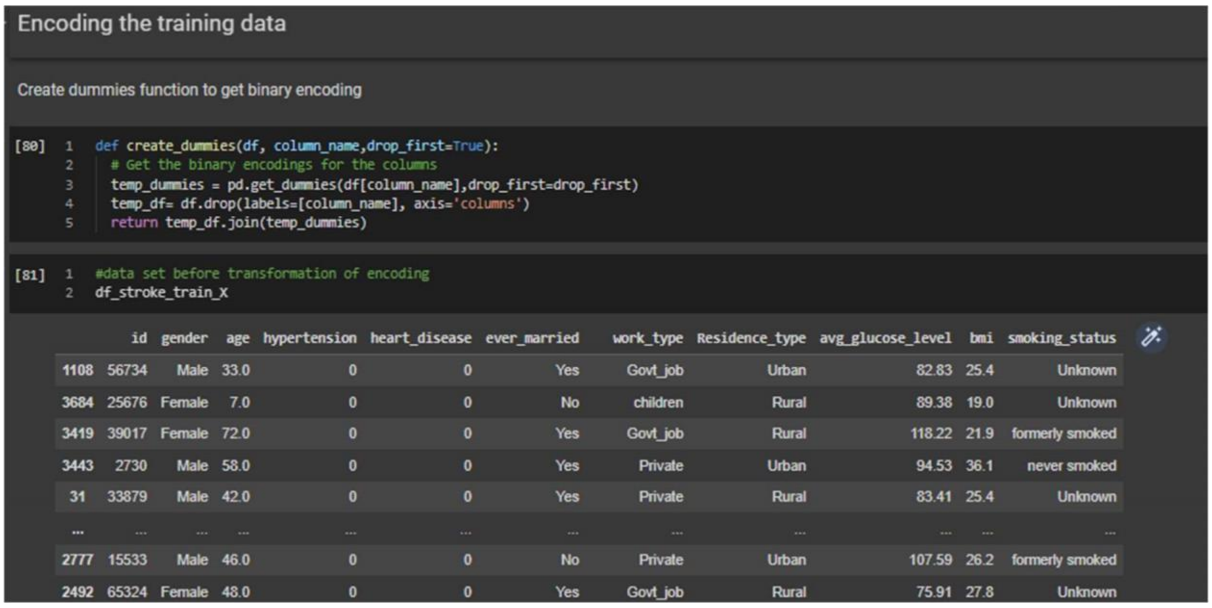
```
[110]
#dropping the Id column as this does not serve a purpose as it is unique to each paitent

df_stroke_train = df_stroke_train.drop('id', axis=1)
```

```
[111]
#we can see that the column has been dropped
df_stroke_train.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
1108	Male	33.0	0	0	Yes	Govt_job	Urban	82.83	25.4	Unknown	0
3684	Female	7.0	0	0	No	children	Rural	89.38	19.0	Unknown	0
3419	Female	72.0	0	0	Yes	Govt_job	Rural	118.22	21.9	formerly smoked	0
3443	Male	58.0	0	0	Yes	Private	Urban	94.53	36.1	never smoked	0
31	Male	42.0	0	0	Yes	Private	Rural	83.41	25.4	Unknown	1

Figure 12 dropping the id column in the training data

Data encoding: The data set contained categorical variables such as gender ever_married, work_type, Residence_type and smoking_status. It was decided to do One Hot Encoding; this is a process by which categorical variables such as the ones mentioned are converted into a form that the machine learning algorithm can better predict. The process was undertaken to convert the categorical variable into a vector of zeros and ones, where each value represents the presence or absence of a particular feature in the data (Jain, 2022). One hot encoding has been used as there are few variables, and creating the extra columns will not increase the time the data is processed.



Encoding the training data

Create dummies function to get binary encoding

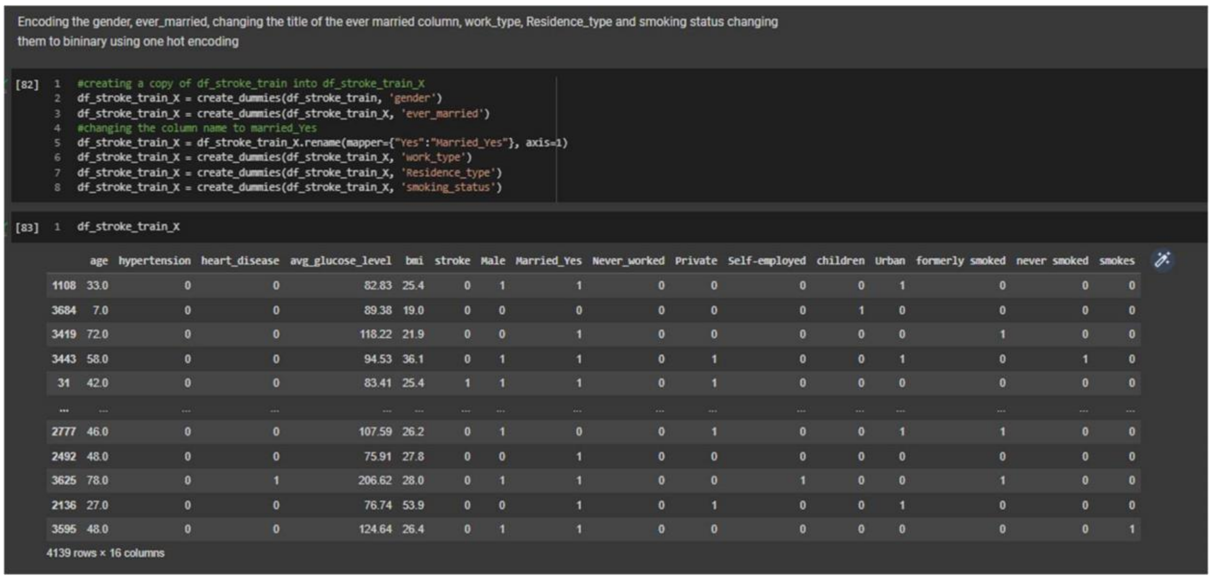
```
[80]
def create_dummies(df, column_name,drop_first=True):
    # Get the binary encodings for the columns
    temp_dummies = pd.get_dummies(df[column_name],drop_first=drop_first)
    temp_df= df.drop(labels=[column_name], axis='columns')
    return temp_df.join(temp_dummies)
```

```
[81]
#data set before transformation of encoding
df_stroke_train_X
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
1108	56734	Male	33.0	0	0	Yes	Govt_job	Urban	82.83	25.4	Unknown
3684	25676	Female	7.0	0	0	No	children	Rural	89.38	19.0	Unknown
3419	39017	Female	72.0	0	0	Yes	Govt_job	Rural	118.22	21.9	formerly smoked
3443	2730	Male	58.0	0	0	Yes	Private	Urban	94.53	36.1	never smoked
31	33879	Male	42.0	0	0	Yes	Private	Rural	83.41	25.4	Unknown
...	...	...	...	...	...	...	...	...	...	...	...
2777	15533	Male	46.0	0	0	No	Private	Urban	107.59	26.2	formerly smoked
2492	65324	Female	48.0	0	0	Yes	Govt_job	Rural	75.91	27.8	Unknown

Figure 13 create dummies function to one hot encode categorical data into binary encoding, also checking the dataset before the encoding takes place

The above function was applied to the categorical data to one hot encode using create_dummies. Initially, the data was encoded separately for each variable, but then it was put into one to make the code more efficient, as can be seen below:


Encoding the gender, ever_married, changing the title of the ever married column, work_type, Residence_type and smoking status changing them to bininary using one hot encoding

```
[82]
#creating a copy of df_stroke_train into df_stroke_train_X
df_stroke_train_X = create_dummies(df_stroke_train, 'gender')
df_stroke_train_X = create_dummies(df_stroke_train_X, 'ever_married')
#changing the column name to married_Yes
df_stroke_train_X = df_stroke_train_X.rename(mapper={"Yes":"Married_Yes"}, axis=1)
df_stroke_train_X = create_dummies(df_stroke_train_X, 'work_type')
df_stroke_train_X = create_dummies(df_stroke_train_X, 'Residence_type')
df_stroke_train_X = create_dummies(df_stroke_train_X, 'smoking_status')
```

```
[83]
df_stroke_train_X
```

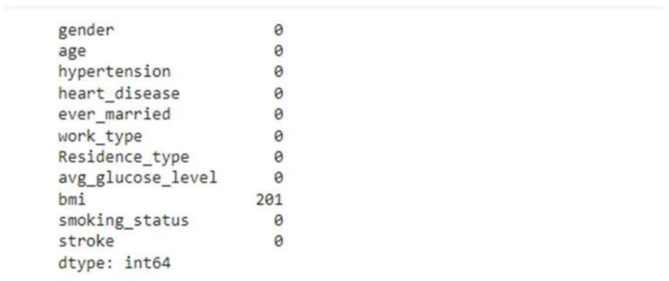
	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	Male	Married_Yes	Never_worked	Private	Self-employed	children	Urban	formerly smoked	never smoked	smokes
1108	33.0	0	0	82.83	25.4	0	1	1	0	0	0	0	1	0	0	0
3684	7.0	0	0	89.38	19.0	0	0	0	0	0	0	1	0	0	0	0
3419	72.0	0	0	118.22	21.9	0	0	1	0	0	0	0	0	1	0	0
3443	58.0	0	0	94.53	36.1	0	1	1	0	1	0	0	1	0	1	0
31	42.0	0	0	83.41	25.4	1	1	1	0	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2777	46.0	0	0	107.59	26.2	0	1	0	0	1	0	0	1	1	0	0
2492	48.0	0	0	75.91	27.8	0	0	1	0	0	0	0	0	0	0	0
3625	78.0	0	1	206.62	28.0	0	1	1	0	0	1	0	0	1	0	0
2136	27.0	0	0	76.74	53.9	0	0	1	0	1	0	0	1	0	0	0
3595	48.0	0	0	124.64	26.4	0	1	1	0	0	0	0	0	0	0	1

4139 rows × 16 columns

Figure 14 code for one hot encoding using create_dummies the categorial features, also the result of the one hot encoding

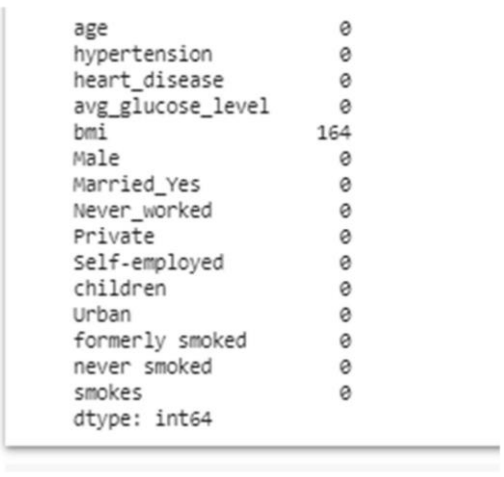
Data Cleaning: Data cleaning is an essential step for any machine learning project; without cleaning the data, it can "result in inaccurate analytics" (Chu et al., 2016). In the Stroke Prediction Dataset, this means dealing with any missing data isna code was used figure below. Depending on how many missing values were found, this could mean discarding any rows containing the missing values or filling in the missing values with a particular method (such as using the mean, median or mode of the data) or using a method such as multiple imputations to generate missing values based on the other values in the dataset.

While exploring the data, the BMI variable had missing data of 201 values of the overall dataset, which is 3.96%.



```
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
```

Figure 15 Data set showing that there are 201 Nan's for BMI within the whole of the data set When looking at the training data the NaN's are just 164 as shown below.

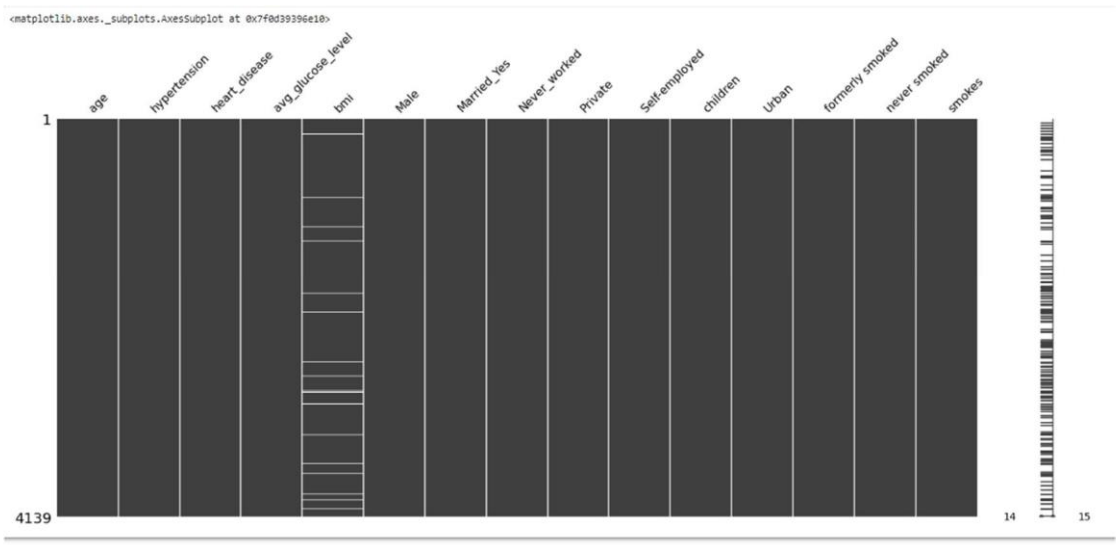


```
age                   0
hypertension          0
heart_disease         0
avg_glucose_level     0
bmi                 164
Male                  0
Married_Yes           0
Never_worked          0
Private               0
Self-employed         0
children              0
Urban                 0
formerly smoked       0
never smoked          0
smokes                0
dtype: int64
```

Figure 16 NaNs in the training set for BMI

To drop this data would mean losing valuable insights as some of these rows also contain patients who have had a stroke, and since the data is so imbalanced, it was decided not to remove the rows but to impute the data into mean values. Using mean values is better as it gives a good representation than if modal values are used.

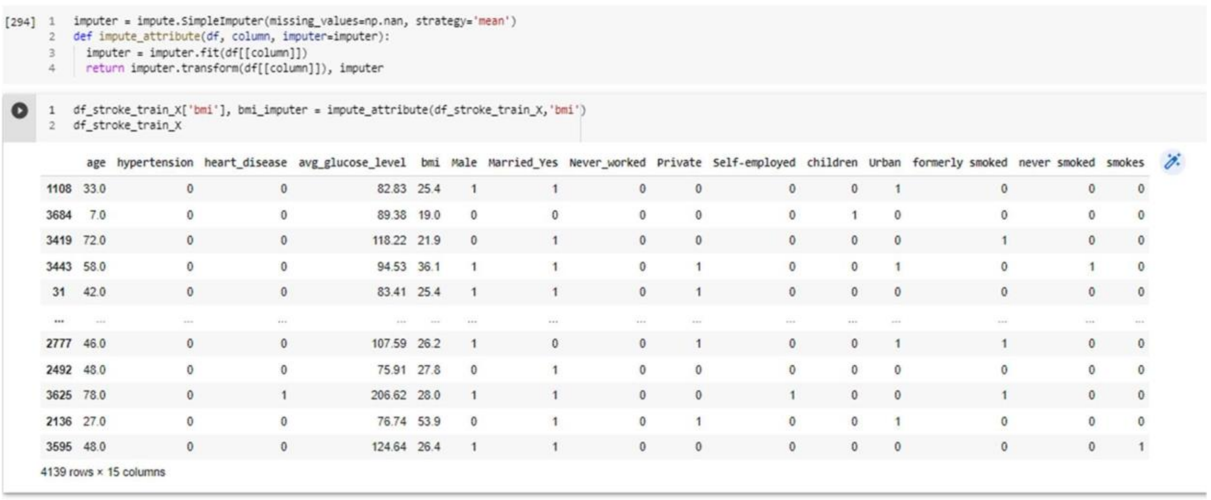
Figure 17 gives a Visual inspection of the data to detect the missing values in the training data set; in the column for BMI, the white lines represent the missing values.



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0d39396e10>
```

Figure 17 visually showing missing data in training data set in the BMI column overall training data is 4139, 15 features with 14 features having no missing values

Therefore, to impute the data to mean values, the imputation function was used. As shown in the figure below



```
[294]
imputer = impute.SimpleImputer(missing_values=np.nan, strategy='mean')
def impute_attribute(df, column, imputer=imputer):
  imputer = imputer.fit(df[[column]])
  return imputer.transform(df[[column]]), imputer
```

```
df_stroke_train_X['bmi'], bmi_imputer = impute_attribute(df_stroke_train_X,'bmi')
df_stroke_train_X
```

	age	hypertension	heart_disease	avg_glucose_level	bmi	Male	Married_Yes	Never_worked	Private	Self-employed	children	Urban	formerly smoked	never smoked	smokes
1108	33.0	0	0	82.83	25.4	1	1	0	0	0	0	1	0	0	0
3684	7.0	0	0	89.38	19.0	0	0	0	0	0	1	0	0	0	0
3419	72.0	0	0	118.22	21.9	0	1	0	0	0	0	0	1	0	0
3443	58.0	0	0	94.53	36.1	1	1	0	1	0	0	1	0	1	0
31	42.0	0	0	83.41	25.4	1	1	0	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2777	46.0	0	0	107.59	26.2	1	0	0	1	0	0	1	1	0	0
2492	48.0	0	0	75.91	27.8	0	1	0	0	0	0	0	0	0	0
3625	78.0	0	1	206.62	28.0	1	1	0	0	1	0	0	1	0	0
2136	27.0	0	0	76.74	53.9	0	1	0	1	0	0	1	0	0	0
3595	48.0	0	0	124.64	26.4	1	1	0	0	0	0	0	0	0	1

4139 rows × 15 columns

Figure 18 function code to impute the missing values using mean

To ensure that the data was correctly imputed, it was rechecked by visualising the training data set and using the isna code as shown in the following two figures:

```
msno.matrix(df_stroke_train_X)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0d38fce890>
```

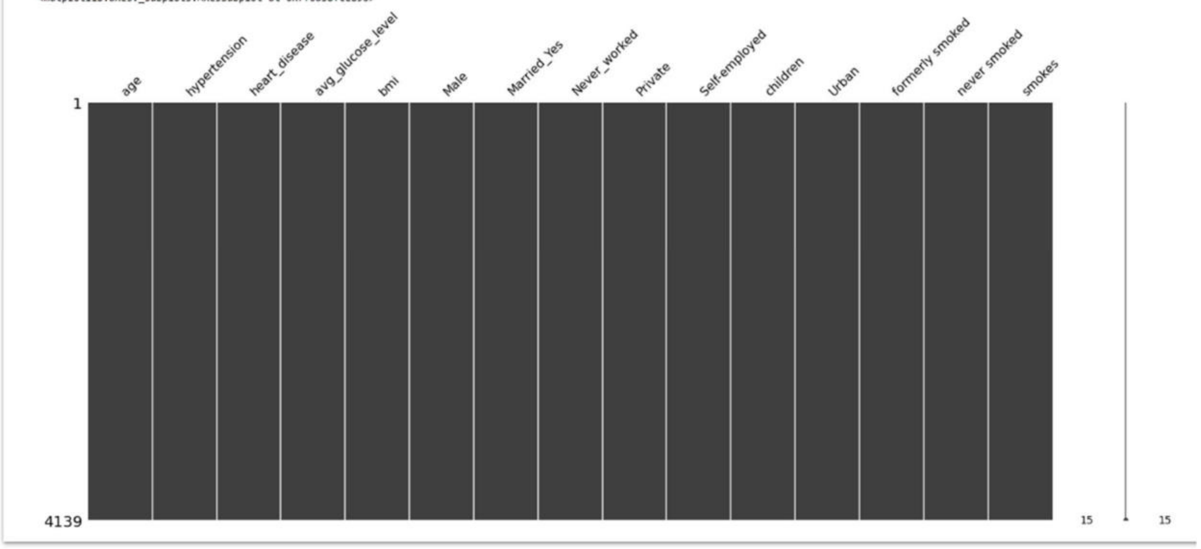
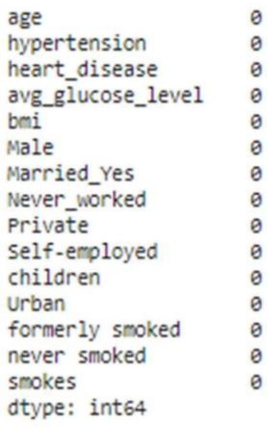


Figure 19 checking to see if the missing values have now been resolved

Figure 19 shows that BMI no longer has missing values, as the columns are all now solid black to show that all values have been filled. Again, Isna is used to verify this is the case, as illustrated in the figure below.



```
age                  0
hypertension         0
heart_disease        0
avg_glucose_level    0
bmi                  0
Male                 0
Married_Yes          0
Never_worked         0
Private              0
Self-employed        0
children             0
Urban                0
formerly smoked      0
never smoked         0
smokes               0
dtype: int64
```

Figure 20 checking to see if the missing values have been resolved

3.4 Limitations and Options

So far, the techniques applied have worked, but the limitation of the data is that it needs to be more balanced. Therefore, applying Smote or isolation forest to either oversample or under-sampling the data and remove outliers may give it a better outcome. We may apply a mixture of oversampling and under-sampling to get a more realistic outcome with the dataset.

Isolation forest is a good measure by which outliers can be removed. For example, in the EDA, it was found that there were a few outliers in BMI and average glucose levels and removing these will improve the accuracy of the model.

4 Predictive Modelling / Model Development

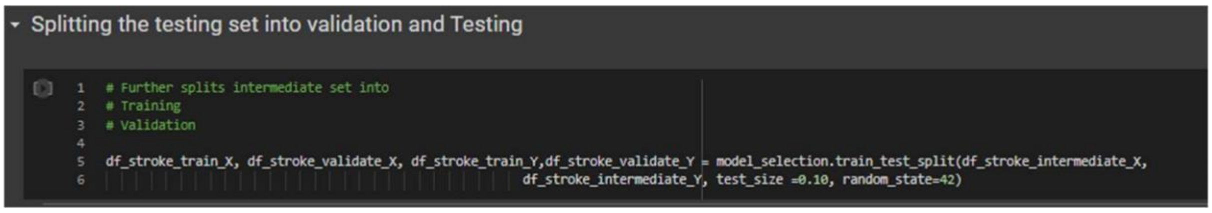
4.1 The predictive modelling processes

The data was pre-processed to create a train/validate/test split. The scikit-learn library in Python split the dataset into training, validation, and testing sets, with 80% of the data used for training, 10% for Validation and 10% used for testing. The splitting of the data enables it to be trained and evaluate the model's performance on unseen data.

In the pre-processing stage, the data with categorical values were one hot encoded using the create_dummies function as there were few variables; this was more logical and would not take up extensive processing time. In addition, there was no link in hierarchy in the categories such as smoking status this then seemed more logical to do one hot encoding.

Missing data was imputed since removing the data would have lost valuable information as in the overall data set there were 201 items missing within the BMI data would have been lost with regards to stroke diagnosis.

Finally, the ID column was removed as it was creating unnecessary noise.

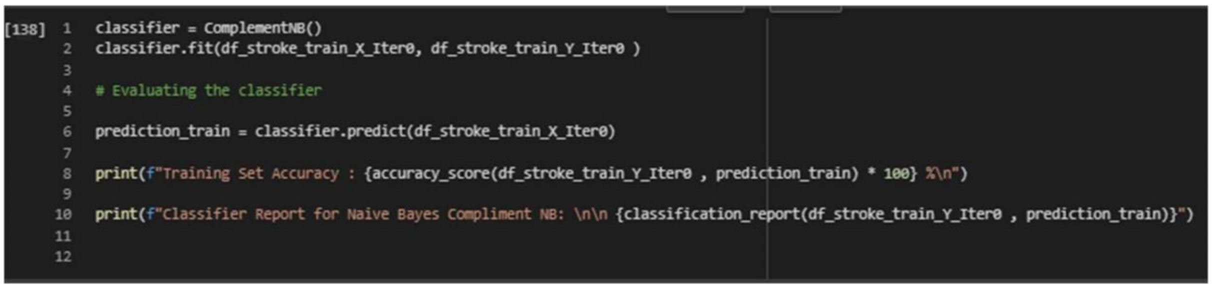


Splitting the testing set into validation and Testing

```
# Further splits intermediate set into
# Training
# Validation

df_stroke_train_X, df_stroke_validate_X, df_stroke_train_Y,df_stroke_validate_Y = model_selection.train_test_split(df_stroke_intermediate_X,
                                                                                   df_stroke_intermediate_Y, test_size =0.10, random_state=42)
```

Figure 21 split of the data training 80% and validation 10% prior to this the data was split 90% intermediate and 10% testing as seen in figure

The next step was to fit the model. The algorithm used was the ComplimentNB algorithm, which is a classification algorithm that is part of the scikit-learn library. For this model, the default parameters of the algorithm were used. These parameters include the prior probabilities of the classes, the features' variance and mean, and the features' correlation. The fit () method was used to fit the model on the training dataset. The fit () method takes in the features and labels of the training dataset as parameters and uses them to "learn" the relationship between the features and labels. The following code snippet shows how the sci-kit-learn library was used to fit the ComplimentNB model on the training dataset:



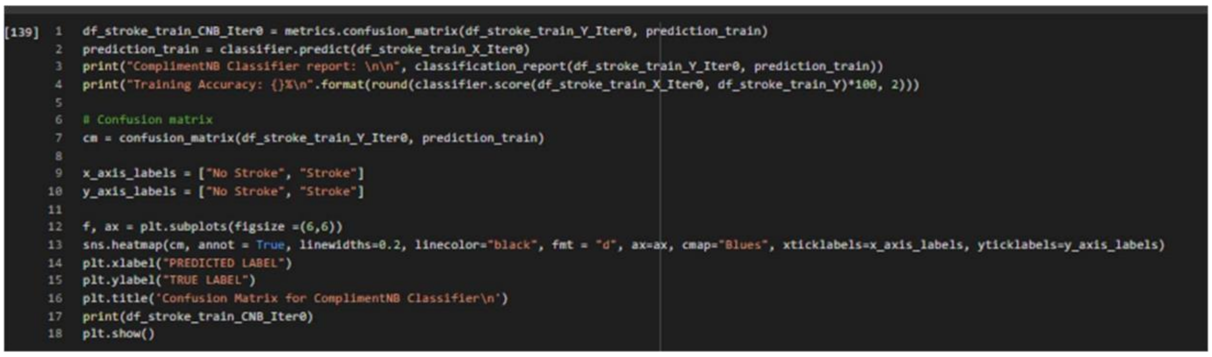
```
[138]
classifier = ComplementNB()
classifier.fit(df_stroke_train_X_Iter0, df_stroke_train_Y_Iter0 )

# Evaluating the classifier

prediction_train = classifier.predict(df_stroke_train_X_Iter0)

print(f"Training Set Accuracy : {accuracy_score(df_stroke_train_Y_Iter0 , prediction_train) * 100} %\n")

print(f"Classifier Report for Naive Bayes Compliment NB: \n\n {classification_report(df_stroke_train_Y_Iter0 , prediction_train)}")


```

Figure 22 snippet showing fit on the training dataset

4.2 Evaluation results on "seen" data

The process of evaluating the ComplimentNB model on previously "seen" data is done by running the model on the training set. First, the model is fit on the training set using the fit () method. This method takes the training data as input and trains the model. Then, the model is tested on the same training set to evaluate its performance. To this end, the predict () method is used, which takes in the training data as its input and returns the predicted labels.

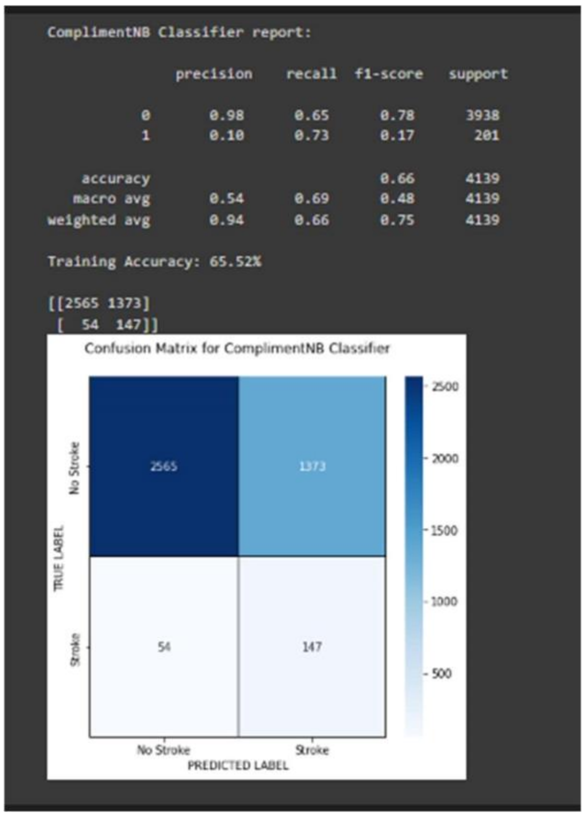
The following code snippet shows Figure 23 the ComplimentNB model on the training set being evaluated by the model:



```
[139]
df_stroke_train_CNB_Iter0 = metrics.confusion_matrix(df_stroke_train_Y_Iter0, prediction_train)
prediction_train = classifier.predict(df_stroke_train_X_Iter0)
print("ComplimentNB Classifier report: \n\n", classification_report(df_stroke_train_Y_Iter0, prediction_train))
print("Training Accuracy: {}%\n".format(round(classifier.score(df_stroke_train_X_Iter0, df_stroke_train_Y)*100, 2)))

# Confusion matrix
cm = confusion_matrix(df_stroke_train_Y_Iter0, prediction_train)

x_axis_labels = ["No Stroke", "Stroke"]
y_axis_labels = ["No Stroke", "Stroke"]

f, ax = plt.subplots(figsize =(6,6))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = "d", ax=ax, cmap="Blues", xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for ComplimentNB Classifier\n')
print(df_stroke_train_CNB_Iter0)
plt.show()
```

Figure 23 code snippet of fitting the model

The model's performance on the training set can be visualized using a confusion matrix Figure 24 shows the training accuracy as being 65.52%.



```
ComplimentNB Classifier report:

              precision    recall  f1-score   support

           0       0.98      0.65      0.78      3938
           1       0.10      0.73      0.17       201

    accuracy                           0.66      4139
   macro avg       0.54      0.69      0.48      4139
weighted avg       0.94      0.66      0.75      4139

Training Accuracy: 65.52%

[[2565 1373]
 [  54  147]]
```

Figure 24 Metrics evaluating the ComplimentNB

The evaluation metrics helps to identify the number of correct and incorrect predictions made by the model. This can be used to evaluate the model and make changes to the model accordingly.

The accuracy is work out as

$$Accuracy = \frac{TP + TN}{Total}$$

Equation 4 Accuracy Equation

In the case of the ComplimentNB model on the stroke data set using the confusion matrix in the above figure:

$$Accuracy = \frac{2565 + 147}{4139}$$

Equation 5 Figures from confusion matrix to work out Accuracy

$$Accuracy = 0.655$$

Equation 6 Accuracy Result

The confusion matrix shows that the True Positive of the 4139 sample is that 2565 had a stroke; this is 61.97 % of the training set.

We have 54 patients out of 4139 with a false positive, which is incorrectly predicted; this is 1.30 % of the training set.

1373 patients out of 4139 with a False-negative which is a positive value predicted as negative for a stroke; this is a large percentage of the overall training set this is 33.17%

The true negative value of 147 of the 4139 was predicted correctly as actual negatives; this is 3.55%.

Since the data set is imbalanced with 95.5% non-stroke patients and 4.5% stroke, the confusion matrix has predicted a good outcome with the ComplimentNB model.

Precision:

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

Equation 7 Formula for precision

$$Precision = \frac{2565}{2565 + 54}$$

Equation 8 Figures from confusion matrix to work out precision

$$Precision = 0.9793$$

Equation 9 Precision Result

The model a very good precision percentage of making correct positive predictions.

Recall:

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

Equation 10 Formula for Recall

$$Recall = \frac{2565}{2565 + 1373}$$

Equation 11 Figures from confusion matrix to work out Recall

$$Recall = 0.6513$$

Equation 12 Recall Result

The model has a reasonably good accuracy of predicting patients with no stroke correctly, but this still needs to be improved, as 65% is just above 50%. We need it to be nearer a hundred as the incorrectly predicted patients could receive the treatment they would not need, which is both costly and timeconsuming and unsuitable for the patient in terms of health.

F1-Score:

$$F1\ Score = \frac{2}{(1/Recall) + (1/Precision)}$$

Equation 13 Formula for F1-Score

$$F1\ Score = 0.78$$

Equation 14 F1-Score Result

5 Evaluation and further modelling improvements

5.1 Initial evaluation comparison

The group used four different models to predict a stroke. The models used were SVM, ComplimentNB, KNN and Decision Tree.

The tables below summarise the evaluation of all four models on the "unseen" data in this case the validation data.

Model	Precision	Recall	F1-Score
SVM	0.00	0.00	0.00
ComplimentNB	0.10	0.73	0.17
KNN	0.00	0.00	0.00
Decision Tree	0.00	0.00	0.00

Table 4 Unseen data comparison between the models where the result was for 1 as stroke

Model	Accuracy	Precision	Recall	F1-Score
SVM	0.96	0.96	1.00	0.98
ComplimentNB	0.66	0.98	0.65	0.78
KNN	0.96	0.96	1.00	0.98
Decision Tree	0.95	0.96	1.00	0.98

Table 5 Unseen data comparison between the models where the result was for 0 no stroke

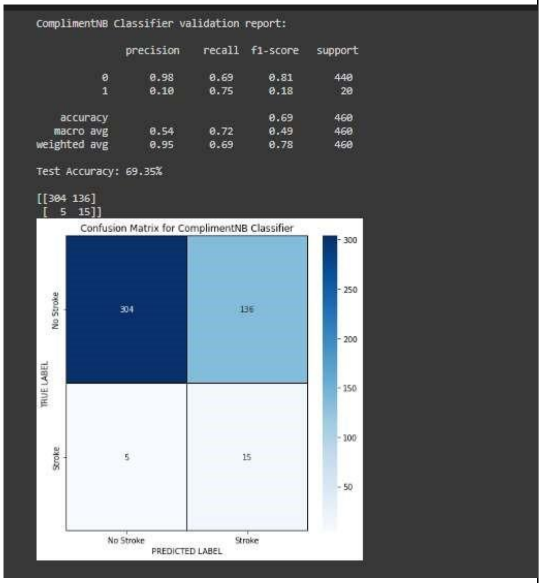
Table 5 shows that Decision Tree is 95.43% accurate at predicting whether a patient has a stroke. This can be backed up with Figure 28, which shows the validation result and heat map.

Both SVM and KNN performed with an accuracy of 95.65%, and the precision, Recall and F1 scores were the same for these two models.

Decision tree, SVM and KNN all had 100% recall, 96% precision and 98% F1 scores. These models outperformed the Complement NB model, which had lower metrics.

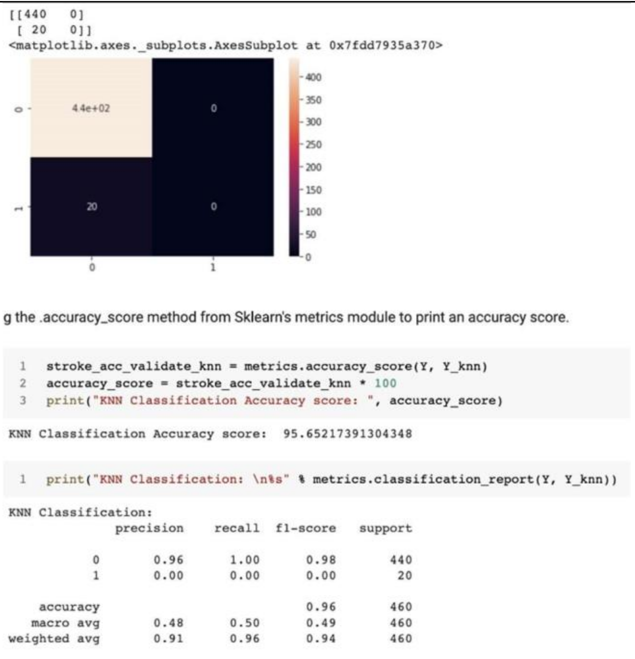
Compliment NB scored the lowest, with an accuracy of 65.52%.

Compliment Naïve Bayes	KNN when K=101



```
ComplimentNB Classifier validation report:

              precision    recall  f1-score   support

           0       0.98      0.69      0.81       440
           1       0.10      0.75      0.18        20

    accuracy                           0.69       460
   macro avg       0.54      0.72      0.49       460
weighted avg       0.95      0.69      0.78       460

Test Accuracy: 69.35%

[[304 136]
 [  5  15]]
```

Figure 25 Compliment Naïve Bayes Validation Result 76.52%



```
[[440   0]
 [ 20   0]]
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd7935a370>
```

g the .accuracy_score method from Sklearn's metrics module to print an accuracy score.

```
stroke_acc_validate_knn = metrics.accuracy_score(Y, Y_knn)
accuracy_score = stroke_acc_validate_knn * 100
print("KNN Classification Accuracy score: ", accuracy_score)
```

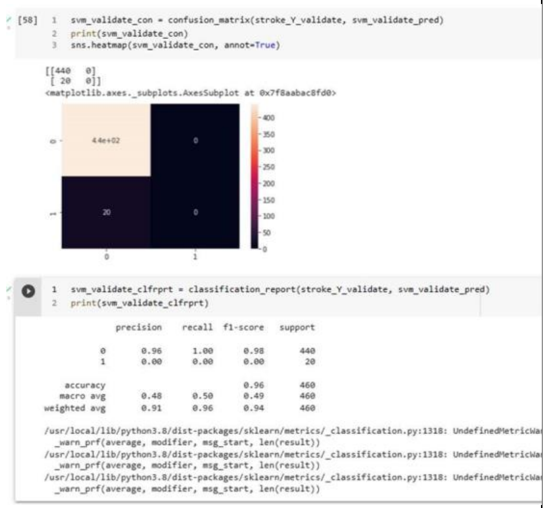
```
KNN Classification Accuracy score:  95.65217391304348
```

```
print("KNN Classification: \n%s" % metrics.classification_report(Y, Y_knn))
```

```
KNN Classification:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       440
           1       0.00      0.00      0.00        20

    accuracy                           0.96       460
   macro avg       0.48      0.50      0.49       460
weighted avg       0.91      0.96      0.94       460
```

Figure 26 KNN validation Result 96% when k = 101

SVM	Decision Tree



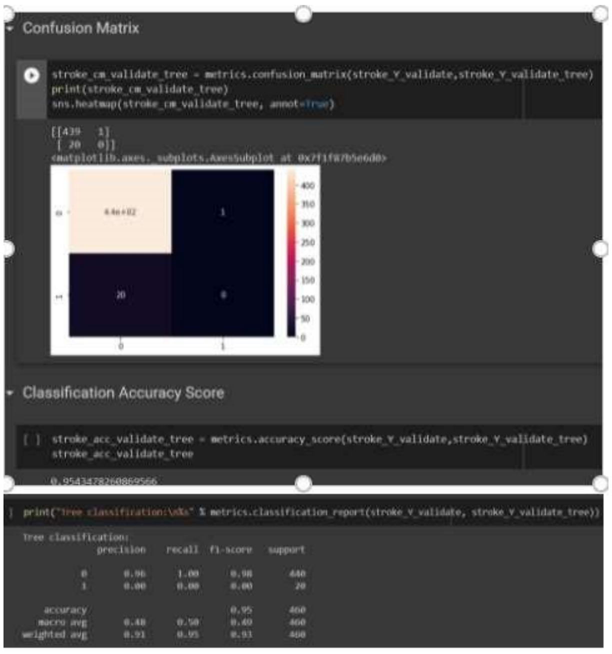
```
[58]
svm_validate_con = confusion_matrix(stroke_Y_validate, svm_validate_pred)
print(svm_validate_con)
sns.heatmap(svm_validate_con, annot=True)
```

```
[[440   0]
 [ 20   0]]
<matplotlib.axes._subplots.AxesSubplot at 0x7f8aabac8fd0>
```

```
svm_validate_clfrprt = classification_report(stroke_Y_validate, svm_validate_pred)
print(svm_validate_clfrprt)
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       440
           1       0.00      0.00      0.00        20

    accuracy                           0.96       460
   macro avg       0.48      0.50      0.49       460
weighted avg       0.91      0.96      0.94       460

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWa
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWa
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWa
  _warn_prf(average, modifier, msg_start, len(result))
```

Figure 27 SVM validation Result accuracy 96%



Confusion Matrix

```
stroke_cm_validate_tree = metrics.confusion_matrix(stroke_Y_validate,stroke_Y_validate_tree)
print(stroke_cm_validate_tree)
sns.heatmap(stroke_cm_validate_tree, annot=True)
```

```
[[439   1]
 [ 20   0]]
<matplotlib.axes._subplots.AxesSubplot at 0x7f1f87b5e6d0>
```

Classification Accuracy Score

```
[ ] stroke_acc_validate_tree = metrics.accuracy_score(stroke_Y_validate,stroke_Y_validate_tree)
    stroke_acc_validate_tree
```

```
0.9543478260869566
```

```
print("Tree classification:\n%s" % metrics.classification_report(stroke_Y_validate, stroke_Y_validate_tree))
```

```
Tree classification:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       440
           1       0.00      0.00      0.00        20

    accuracy                           0.95       460
   macro avg       0.48      0.50      0.49       460
weighted avg       0.91      0.95      0.93       460
```

Figure 28 Decision Tree validation Result accuracy 95%

Table 6 Compliment Naïve Bayes, KNN, SVM and Decision Tree validation results

5.2 Further modelling improvements attempted

For further modelling improvements, there will be five different improvements to see which increase the model's accuracy and can correctly predict and diagnose the patient with a stroke.

These five improvements will be:

1. SMOTE oversampling to balance the imbalance in the data set

2. Feature scaling to standardise the features and re-scale with a distribution value of 1 and value of 0 Mean
3. Isolation Forest- BMI and Average Glucose (removal of outliers)
4. Removing Residential_type as this was discovered within the EDA; this does have any impact on where someone lives if they will have a stroke.
5. Label Encoding as opposed to one hot encoding

Table 7 5 different Iterations

The table of results for further modelling improvements on the Compliment Naïve Bayes Algorithm is as follows:

Changes made to the model before validation	Precision	Recall	F1 Score
1) SMOTE- Over Sampling-training	0.63	0.66	0.65
2) SMOTE- Over Sampling-Validation	0.65	0.76	0.70
3) Feature Scaling Standardise features and re-scale with a distribution value of 0 mean and variance equal to 1-training	0.42	1.00	0.59
4) Feature Scaling Standardise features and re-scale with a distribution value of 0 mean and variance equal to 1-validation	0.48	1.00	0.65
5) Isolation Forest- BMI and Average Glucose (removal of outliers)- training	0.10	0.76	0.18
6) Isolation Forest- BMI and Average Glucose (removal of outliers) - validation	0.11	0.80	0.19
7) Removing Residential_typetraining	0.10	0.74	0.17
8) Removing Residential_type - validation	0.10	0.75	0.17
9) Label Encoding as opposed to one hot encoding-training	0.15	0.99	0.26
10) Label Encoding as opposed to one hot encoding - validation	0.10	0.75	0.17

Table 8 Results table for changes to the model for Stroke (1)

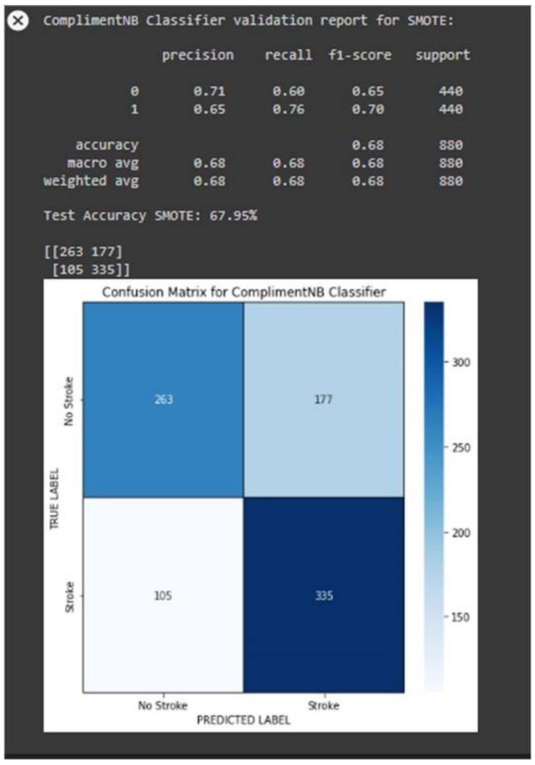
Changes made to the model before validation	Accuracy	Precision	Recall	F1 Score
1) SMOTE- Over Sampling-training	63.57%	0.64	0.61	0.62
2) SMOTE- Over Sampling-Validation	67.95%	0.71	0.60	0.65

3) Feature Scaling Standardise features and re-scale with a distribution value of 0 mean and variance equal to 1training	93.38%	1.00	0.93	0.96
4) Feature Scaling Standardise features and re-scale with a distribution value of 0 mean and variance equal to 1- validation	95.22%	1.00	0.95	0.97
5) Isolation Forest- BMI and Average Glucose (removal of outliers)-training	66.34%	0.98	0.66	0.79
6) Isolation Forest- BMI and Average Glucose (removal of outliers)- validation	69.78%	0.99	0.69	0.81
7) Removing Residential_typetraining	65.33%	0.98	0.65	0.78
8) Removing Residential_typevalidation	68.48%	0.98	0.68	0.81
9) Label Encoding as opposed to one hot encoding-training	73.21%	1.00	0.72	0.84
10) Label Encoding as opposed to one hot encoding - validation	69.13%	0.98	0.69	0.81

Table 9 Results table for changes to the model for no Stroke

Validation results for the 5 Iterations

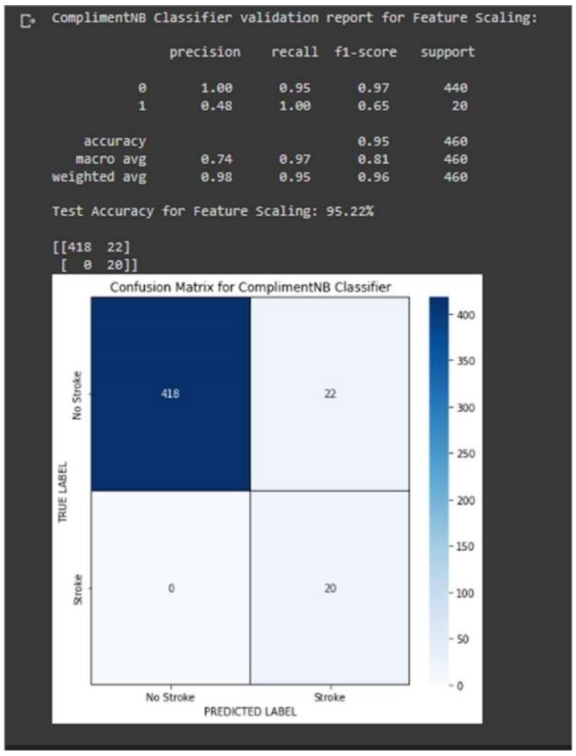
SMOTE- Over Sampling



```
ComplimentNB Classifier validation report for SMOTE:

              precision    recall  f1-score   support

           0       0.71      0.60      0.65       440
           1       0.65      0.76      0.70       440

    accuracy                           0.68       880
   macro avg       0.68      0.68      0.68       880
weighted avg       0.68      0.68      0.68       880

Test Accuracy SMOTE: 67.95%

[[263 177]
 [105 335]]
```

Figure 29 validation for Complement NB with SMOTE over sampling

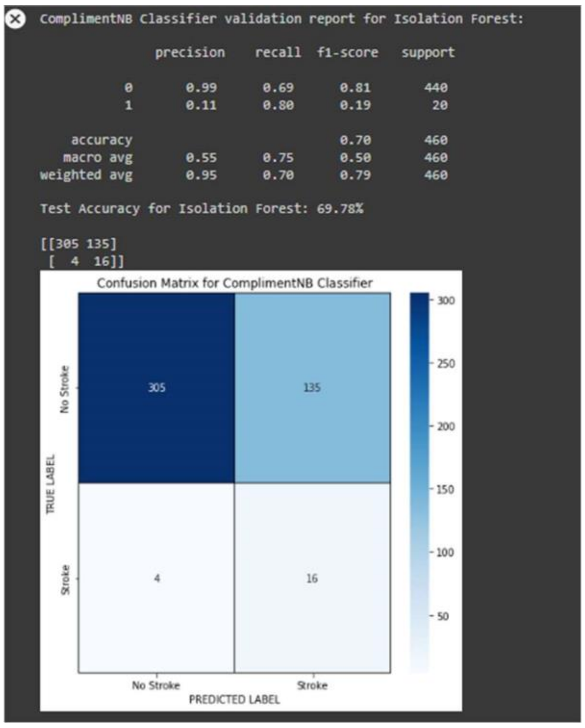
Feature Scaling Standardise features and re-scale with a distribution value of 0 mean and variance equal to 1



```
ComplimentNB Classifier validation report for Feature Scaling:

              precision    recall  f1-score   support

           0       1.00      0.95      0.97       440
           1       0.48      1.00      0.65        20

    accuracy                           0.95       460
   macro avg       0.74      0.97      0.81       460
weighted avg       0.98      0.95      0.96       460

Test Accuracy for Feature Scaling: 95.22%

[[418  22]
 [  0  20]]
```

Figure 30 validation for Complement NB with feature scaling

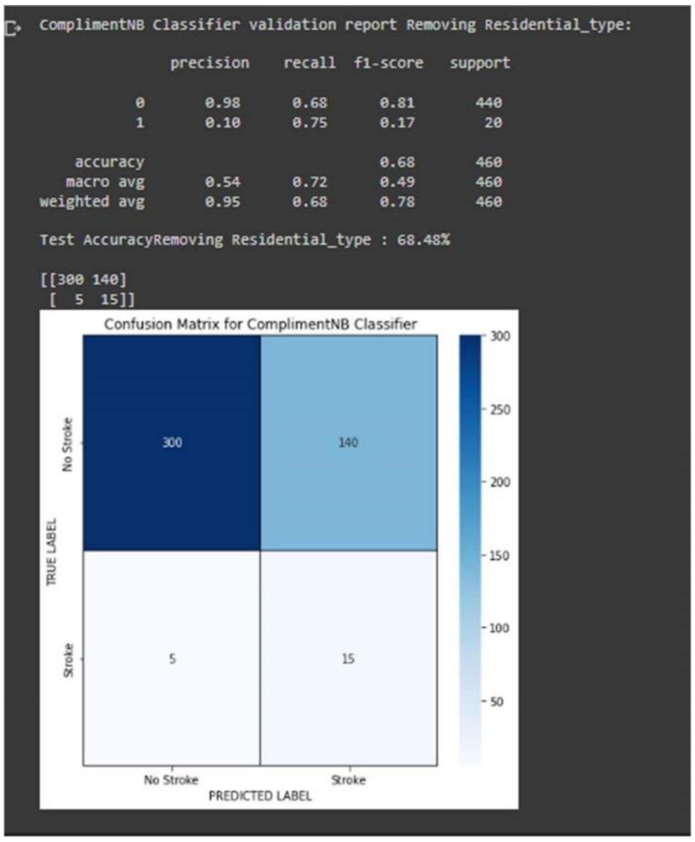
Isolation Forest- BMI and Average Glucose (removal of outliers)



```
ComplimentNB Classifier validation report for Isolation Forest:

              precision    recall  f1-score   support

           0       0.99      0.69      0.81       440
           1       0.11      0.80      0.19        20

    accuracy                           0.70       460
   macro avg       0.55      0.75      0.50       460
weighted avg       0.95      0.70      0.79       460

Test Accuracy for Isolation Forest: 69.78%

[[305 135]
 [  4  16]]
```

Figure 31 validation for Complement NB with isolation forest

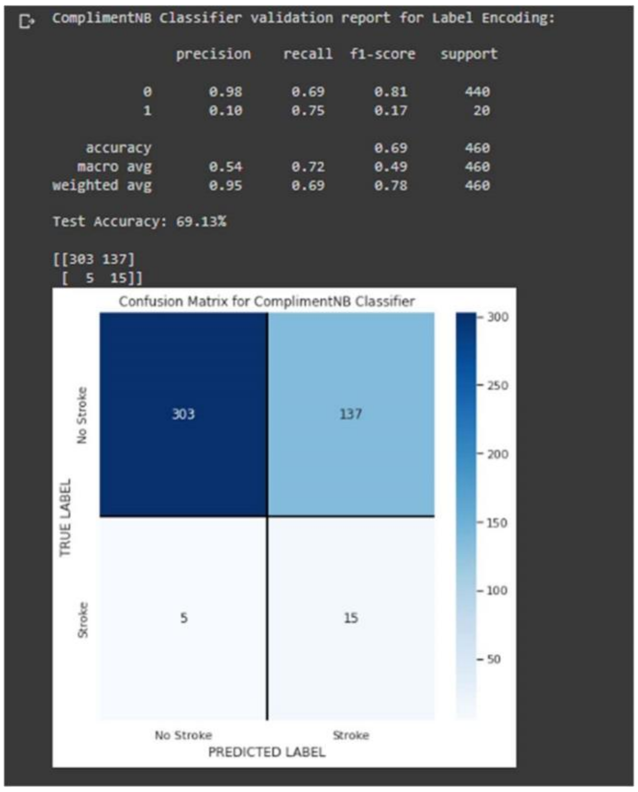
Removing Residential_type



```
ComplimentNB Classifier validation report Removing Residential_type:

              precision    recall  f1-score   support

           0       0.98      0.68      0.81       440
           1       0.10      0.75      0.17        20

    accuracy                           0.68       460
   macro avg       0.54      0.72      0.49       460
weighted avg       0.95      0.68      0.78       460

Test AccuracyRemoving Residential_type : 68.48%

[[300 140]
 [  5  15]]
```

Figure 32validation for Complement NB with removing residential_type feature

Label Encoding as opposed to one hot encoding



```
ComplimentNB Classifier validation report for Label Encoding:

              precision    recall  f1-score   support

           0       0.98      0.69      0.81       440
           1       0.10      0.75      0.17        20

    accuracy                           0.69       460
   macro avg       0.54      0.72      0.49       460
weighted avg       0.95      0.69      0.78       460

Test Accuracy: 69.13%

[[303 137]
 [  5  15]]
```

Figure 33validation for Complement NB with Label Encoding

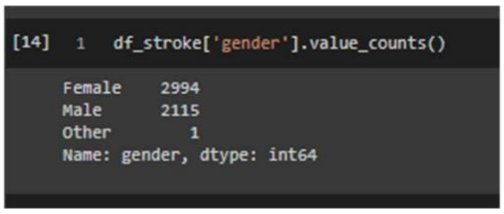
Table 10 Screen shots of results of the iterations with validation results

From the table of results, the feature scaling performed better in accuracy, being 95.22% Precision recall and F1 score. However, SMOTE proves that balancing the data would not increase accuracy or the other metrics but a decrease in all areas. Therefore, moving forward to the final test of the model, feature scaling will be used to see if the accuracy can be increased, and stroke can be positively predicted.

5.3 Final Evaluation results

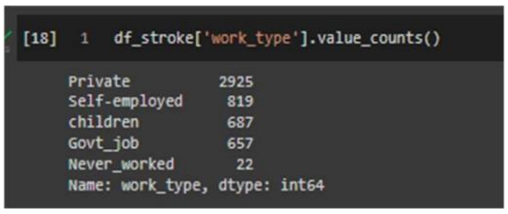
For the final Testing of the model, Feature scaling has been used.

There is, however, an issue concerning minority items within the data set, such as gender Figure 34 and the work_type - never worked Figure 35.



```
[14]  df_stroke['gender'].value_counts()

Female    2994
Male      2115
Other        1
Name: gender, dtype: int64
```

Figure 34 code snippet showing gender feature

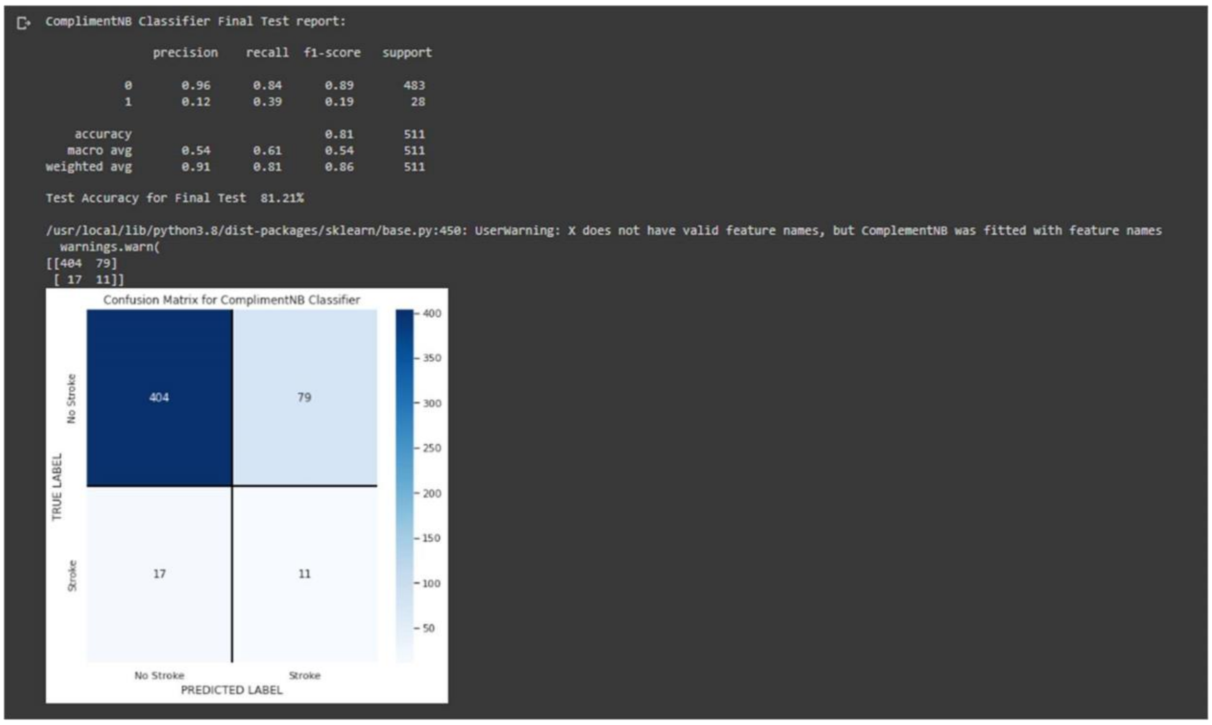


```
[18]  df_stroke['work_type'].value_counts()

Private          2925
Self-employed     819
children          687
Govt_job          657
Never_worked       22
Name: work_type, dtype: int64
```

Figure 35 code snippet showing work_type

Since random sampling has been used to split the data, these items were put into the three splits without giving a proportional; representation of each feature in the train, validate and test group.

Therefore, the other gender has appeared in the unseen testing data and the never worked has not as there was only enough to be split between the training and validation data sets.



```
ComplimentNB Classifier Final Test report:

              precision    recall  f1-score   support

           0       0.96      0.84      0.89       483
           1       0.12      0.39      0.19        28

    accuracy                           0.81       511
   macro avg       0.54      0.61      0.54       511
weighted avg       0.91      0.81      0.86       511

Test Accuracy for Final Test  81.21%

/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but ComplementNB was fitted with feature names
  warnings.warn(
[[404  79]
 [ 17  11]]
```

Figure 36 final test evaluation of Complement Naïve Bayes

The model has got an improved accuracy of 81.21%. However, this is a 2% decrease in precision from the initial validation model. The Recall has increased from 0.69 to 0.84, and F1-score has increased from 0.81 to 0.89. Therefore, by removing outliers and scaling the BMI and average glucose levels, the score has seen a positive outcome. In addition, as per the EDA findings, these two features did not positively correlate with stroke. Therefore, removing the noise of the outliers helped generate better accuracy and metrics in predicting a first-line stroke diagnosis.

6 Conclusion

6.1 Summary of results

The current results of the model development process demonstrated that the model did not perform well on the training data and had low accuracy. However, the model was evaluated on various metrics, including accuracy, precision, recall, and F1 score, and has achieved good results. Additionally, the model has been tested on unseen data and achieved better performance metrics. This indicates that the model is generalising well and can make accurate predictions on unseen data by implementing feature scaling.

The results from the model development process indicate that the model performed relatively well when feature scaling was implemented, with a high accuracy score and improved Recall and F1 scores. Furthermore, the model could accurately predict the outcome of the data set, with a high degree of accuracy being 81.21%. Additionally, the model identified essential features that could be used to make more accurate predictions, such as using stratified sampling rather than the random sampling used where some features failed to appear evenly in the split. Overall, these results demonstrate that the model performs at an acceptable level and could be used for further analysis.

6.2 Suggested further improvements to the model development process

The model development process went well in that the problem was accurately defined, identifying the required data was made, building the model, and evaluating the model's performance was completed. I also tuned the model to improve its performance successfully by using feature scaling.

With my recent experience, I would improve the model development process by ensuring that I have a more thorough understanding of the data before building the model. I would also use the K-fold validation technique to evaluate the model further and identify potential problems. Additionally, I would like to utilise hyperparameter tuning better to improve model performance. Finally, I would like continue, to monitor the model's performance over time to ensure that it is still working as expected.

One thing that went well with the model development process was cross-validation. This allowed us to evaluate the model's performance on unseen data and helped to ensure that the model was not overfitting or underfitting the data. Additionally, the use of various evaluation metrics allowed me to get a better understanding of the model's performance.

Based on my current model evaluation findings and experience with the model development process, I would improve my model development process by being more rigorous in feature engineering and selection. I would also explore different algorithms to see which best suits the data. Additionally, I would explore using more sophisticated techniques such as Key Performance Indicators (KPIs), ensemble methods, and advanced feature engineering to improve the model's performance.

6.3 Reflection on Research Team

Working with the research team was a positive experience because I gained insights and other perspectives, such as completing the EDA. However, there was an issue, and by speaking with the group, it was established that the comparison between the target variable ‘stroke and other features of the visualisations did not look correct. Upon discussing this with the group, someone said to do it another way, and it came out with a more readable comparison.

In addition, to remove further potential issues, I worked with my team to ensure everyone was on the same page.

We communicated expectations, set deadlines, and outlined a plan to address potential issues. We also discussed each step of the project and had teams calls to discuss any issues. Additionally, we discussed our research findings and agreed on a timeline for providing the validation results to enable each member enough time to reflect and compare.

We disagreed with dropping the ID column as two members said they wanted to keep it; this included me, as we both thought it would be helpful to identify the participant. However, the other two members said they wanted to drop it as it served no purpose. After discussion, we decided to drop it.

It was also a challenge since my research team were from the data science course and I was from the computer science course; meeting up and discussing issues was a challenge. There this was mitigated by using teams to talk through issues and sending messages through teams. Overall, the experience was good, and I learnt a lot.

6.4 Reflection on Individual Learning

1. I learned how to apply machine learning and AI to solve various real-world problems. I gained an understanding of the different algorithms and techniques used in the field and how to apply them to a given problem. I also learned how to evaluate the performance of an AI model and adjust parameters to improve its accuracy.

2. I enjoyed most the challenge of solving a real-world problem using machine learning and AI. It was also interesting to explore the different algorithms and techniques available and how they could be applied to the given problem.

3. To further improve my machine learning and AI skills, I would improve my understanding of the algorithms and techniques used in the field. I would also improve my coding and programming skills to develop better models. Additionally, I would like to look into using more advanced techniques, such as deep learning and reinforcement learning to work on more complex problems.

7 References

Agrawal, H., Chandiwala, J., Agrawal, S and Goyal, Y. (2021) Heart Failure Prediction using Machine Learning with Exploratory Data Analysis. International Confrence on Intelligent Technologies, Available at: https://doi: 10.1109/CONIT51480.2021.9498561.

Alam, M., Lee, T.J., Tatbul, N., Zdonik, S. and Gottschlich, J. (2018) Precision and recall for time series.[pdf] 32. Neural Information Processing Systems, Available at: Precision and Recall for Time Series (neurips.cc)[Accessed 25 December 2022].

Anatskia, L.N. (2011) Stroke in elderly patients. Zh Nevrol Psikhiatr Im S S Korsakova, Available at: PMID:22224250.

Balaji, S. (2022) Python deepcopy. Available at: Python deepcopy | Complete Guide to Python deepcopy with Examples (educba.com) [Accessed 12 January 2022].

Baskini, M and Proios, H. (2023) Obesity and Stroke. Available at: Obesity and Stroke - StrokePrevention.info [Accessed 15 January 2023].

Boot, E., Ekker, M.S., Putaala, J., Kittner, S., Leeauw, F.E.D and Tuladhar, A.M. (2020) Ischaemic stroke in young adults: a global perspective. Journal or Neurology, Neurosurgery & Psychiatry, 91(4), Available at: https://10.1136/ jnnp-2019-322424.

Chu, X., Ilyas, I.F., Krishnan, S., Wang, J. (2016) Data Cleaning: Overview and Emerging Challenges. Association for Computing Machinery, 1145. Available at: https://doi.org/10.1145/2882903.2912574.

Data Mining & Big Data (2021) What is Regression Model?. Available at: What Is a Regression Model? | IMSL by Perforce [Accessed 14 January 2022].

Descatha, A., Sembajwe, G., Pega, F., Ujita, Y., Baer, M., Boccuni, F., Tecco, C.D., Duret, C., Evanoff, B.A., Gagliardi, D. et al. (2020) The effect of exposure to long working hours on stroke: A systematic review and meta-analysis from the WHO/ILO Joint Estimates of the Work-related Burden of Disease and Injury. Environment International, 142, Available at: https://doi.org/10.1016/j.envint.2020.105746.

Donker, E.S. (2018) Stroke in the 21st Century: A Snapshot of the Burden, Epidemiology, and Quality of Life. Stroke Research and Treatment, 2018, 3238165. Available at: https://doi.org/10.1155/2018/3238165.

Emon, M.U., Keya, M. S., Meghla, T.I., Rahman, M.M., Mamun, M.S.A., and Kaiser, M. S. (2020) Performance Analysis of Machine Learning Approaches in Stroke Prediction. 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Available at: https://doi:10.1109/ICECA49313.2020.9297525.

Fanous, P. (2021) Classification Models and Thresholds in Machine Learning. Available at: Classification Models and Thresholds in Machine Learning | by Karim Fanous | Towards Data Science [Accessed 12 January 2023].

Fedesoriano (2021) Body Fat Prediction Dataset. [dataset] Available at: https://www.kaggle.com/datasets/fedesoriano/body-fat-prediction-dataset [Accessed 01 December 2022].

Fedesoriano (2021) Stroke Prediction Dataset. [dataset] Available at: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset [Accessed 01 December 2022].

Hosseinzadeh, N., Ala, A., Rahnemayan, S., Sadeghi-Hokmabadi, E., Milani, A.G., Asenjan, M.R. and Vahdati, S.S. (2022) Demographic information and risk factors of a stroke patients younger than 65 years old. Frontiers in Emergency Medicine, 6(1), Available at: https://10.18502/fem.v6i1.7676.

Jain, K. (2022) Simple Methods to deal with Categorical Variables in Predictive Modeling. Available at: How to Deal With Categorical Variable in Predictive Modeling (analyticsvidhya.com) [Accessed 25 December 2022].

Khrawal, A. (2020) Why Random_state=42 in Machine Learning?. Available at : Why Random_state=42 in Machine Learning? | Aman Kharwal (thecleverprogrammer.com) [Accessed 09 January 2023].

Lawler, M. (2022) What’s a Healthy BMI in Adults? Here’s Everything You Need to Know. Available at: What’s a Healthy BMI in Adults? Here’s Everything You Need to Know | Everyday Health [Accessed 11 January 2023].

Miao, J. and Zhu, W. (2022) Precision-recall curve (PRC) Classification tree. Evol.Intel, 15, 1007. Available at: https://doi.org/10.1007/s12065-021-00565-2.

NHS (2022) What is the body mass index (BMI)?. Available at: What is the body mass index (BMI)? - NHS (www.nhs.uk)[Accessed 15 January 2023].

Pasini, A. (2015) Artificial neural networks for small dataset analysis. Journal of Thoracic Disease, 7(5), Available at: https://DOI: 10.3978/j.issn.2072-1439.2015.04.61.

Perez, J.G. and Perez, E.S. (2021) Predicting Student Program Completion. I.J. Modern Education and Computer Science, 3, 5815. Available at: https://doi.org/10.5815/ijmecs.2021.03.05.

Phipps, M.S. and Cronin, C.A. (2020) Management of acute ischemic stroke. BMJ, 368, 16983. Available at: https://doi.org/10.1136/bmj.l6983.

Poole, J. (2023) What is considered a normal blood sugar level. Available at: What is considered a normal blood sugar level? (mymed.com) [Accessed 14 January 2023].

Waskom, M. (2022) Seaborn.histplot. Available at: seaborn.histplot — seaborn 0.12.2 documentation (pydata.org) [Accessed 04 January 2023].

Weiren, S. (2022) Avoid Data Leakage-Split Your Data Before Processing. Available at: Avoid Data Leakage — Split Your Data Before Processing | by Sun Weiran | Towards Data Science [Accessed 12 January 2023].

Yasser, M.H. (2022) Breast Cancer Dataset. [dataset] Available at: https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset [Accessed 01 December 2022].

8 Appendix 1

8.1 Exploratory Data Analysis Process and Results

8.1.1 All feature Univariate Analysis

8.1.1.1 Univariate Analysis of the target variable

The exploratory data analysis was performed on the training data set. Initially, EDA was performed on the target, “stroke”, and then the univariate analysis will take place on the remaining features.

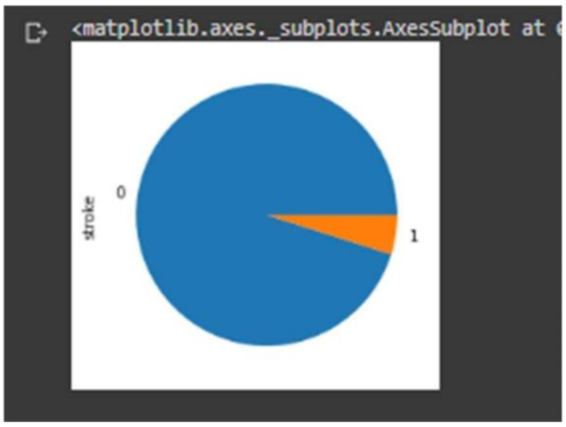


Figure 37 Target variable of the train data set showing 94.14 % no stroke diagnosis and 4.86% stroke diagnosis

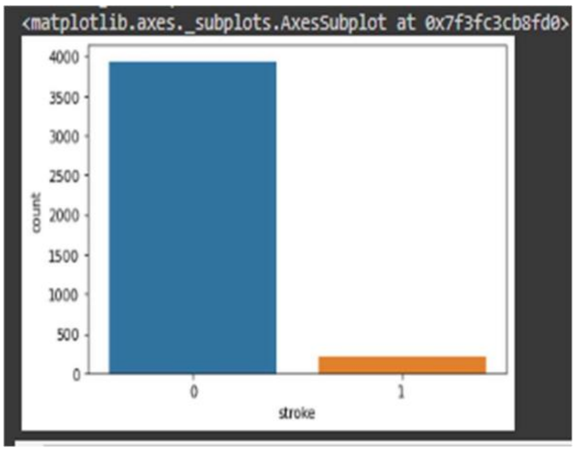
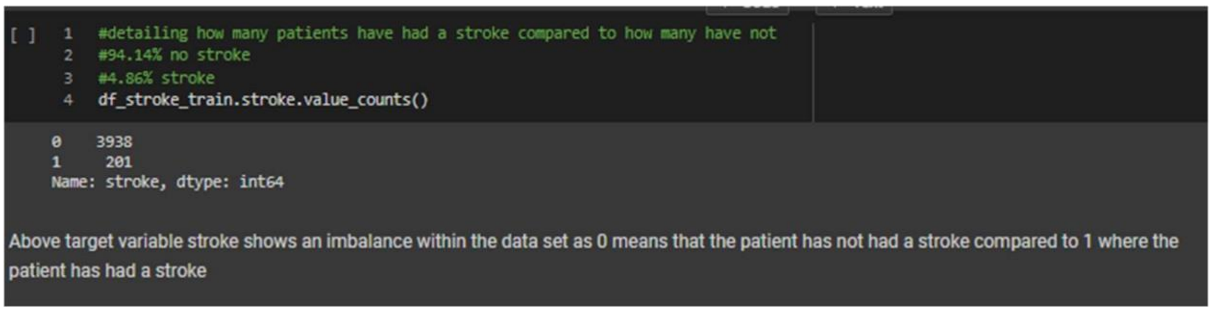


Figure 38 Stroke is the target variable 1 being a patient having a stroke and 0 not having a stroke

As can be seen in Figure 37 and Figure 38, the training data set displays visually that the data is imbalanced, with a large majority of the participants not having a stroke (94.14%) compared to those that have had a stroke (4.86%). However, this imbalance is expected because if the volumes of stroke patients were higher, there would be an issue as the data may need to be corrected or rechecked.

Code snippet Figure 39 below shows the out with the count of how many patients had a stroke diagnosis compared to how many did not.



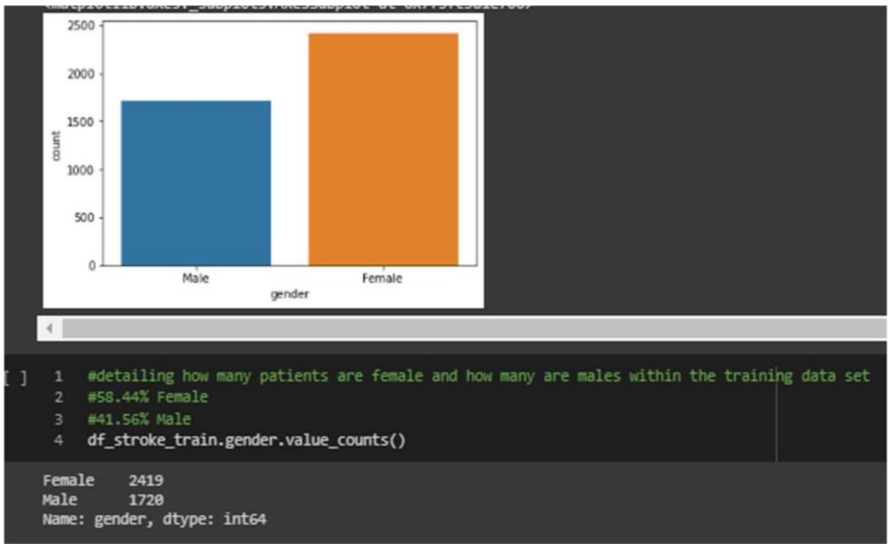
```
#detailing how many patients have had a stroke compared to how many have not
#94.14% no stroke
#4.86% stroke
df_stroke_train.stroke.value_counts()
```

```
0    3938
1     201
Name: stroke, dtype: int64
```

Above target variable stroke shows an imbalance within the data set as 0 means that the patient has not had a stroke compared to 1 where the patient has had a stroke

Figure 39 Count of how many patients had a stroke diagnosis

8.1.1.2 Categorical Univariate Analysis of the feature variables

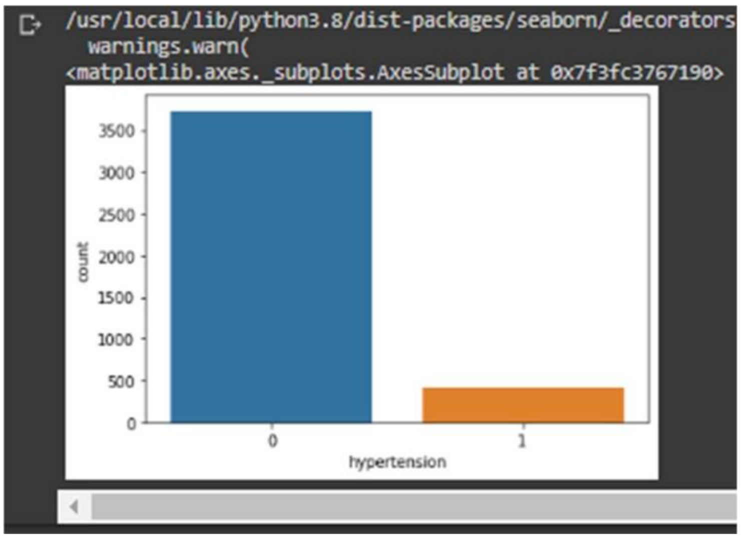


```
#detailing how many patients are female and how many are males within the training data set
#58.44% Female
#41.56% Male
df_stroke_train.gender.value_counts()
```

```
Female    2419
Male      1720
Name: gender, dtype: int64
```

Figure 40 data set showing male to female ratio within the training data set

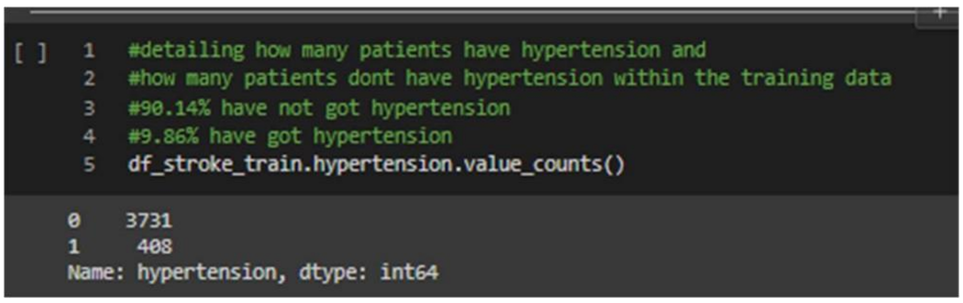
Figure 40 indicates more females than males with the training data set. However, the balance of the data relating to the genders is not too vast or imbalanced, with Females: 2419, 58%, and Males: 1720, which is 42%.



```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc3767190>
```

Figure 41 Hypertension data from the training data set showing an imbalance

Figure 41 shows an imbalance of the training data for variable hypertension. It indicates that most patients with hypertension were 408 (9.86%) in the overall data set. This is correct as most of the population would not have hypertension; therefore, the imbalance is actual of real-world statistics.

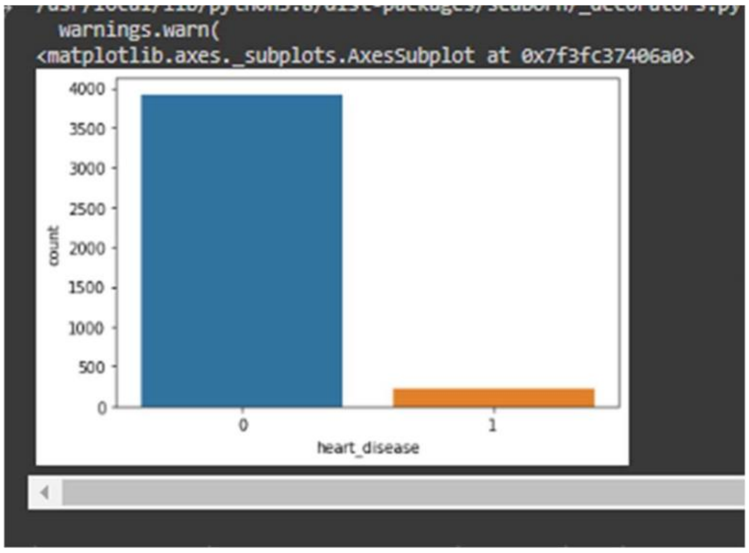


```
#detailing how many patients have hypertension and
#how many patients dont have hypertension within the training data
#90.14% have not got hypertension
#9.86% have got hypertension
df_stroke_train.hypertension.value_counts()
```

```
0    3731
1     408
Name: hypertension, dtype: int64
```

Figure 42 Count of patients with and without hypertension

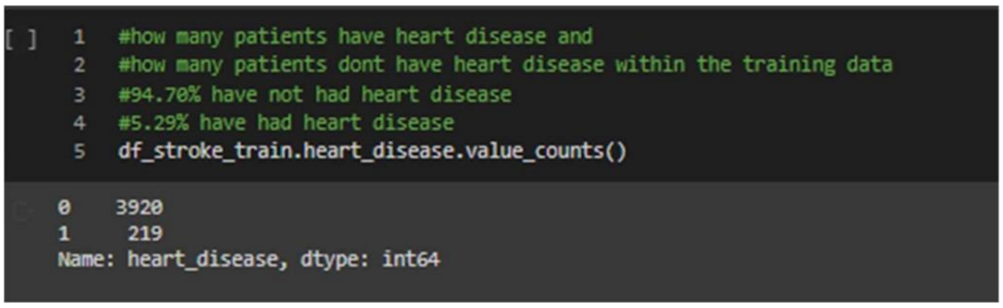
The snippet above Figure 42 shows the number of patients who have do not hypertension 3731 (90.14%) and 408 (9.86%) who do.



```
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc37406a0>
```

Figure 43 heart disease in the training data set showing an imbalance

Figure 43 also indicates, just as the target variable and the hypertension feature, that heart disease is also imbalanced regarding the data contained in the training data. Again, as with the other two variables, this is also correct, as it would not expect the majority of the population to have heart disease.



```
#how many patients have heart disease and
#how many patients dont have heart disease within the training data
#94.70% have not had heart disease
#5.29% have had heart disease
df_stroke_train.heart_disease.value_counts()
```

```
0    3920
1     219
Name: heart_disease, dtype: int64
```

Figure 44 count of the split of how many patients have heart disease compared to those that do not

The snippet Figure 44 shows the number of patients who have do not heart disease 3920 (94.70%) and 219 (5.29%) who do.

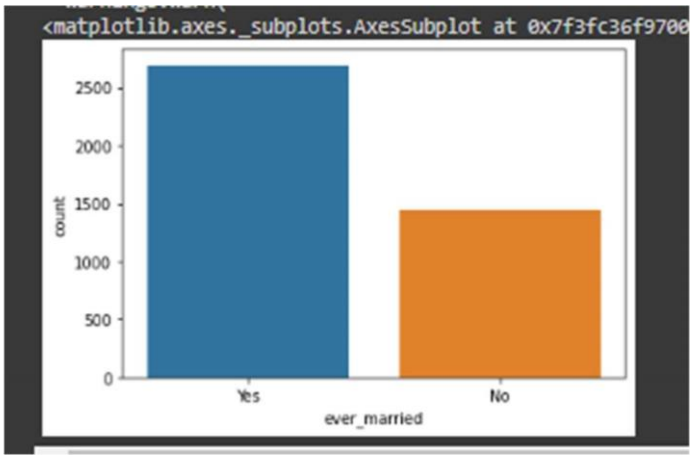


Figure 45 visualisation showing if someone is married or not

Figure 45 shows that there are more people married, which is 65%, then there are not, which is 35% within the data set.

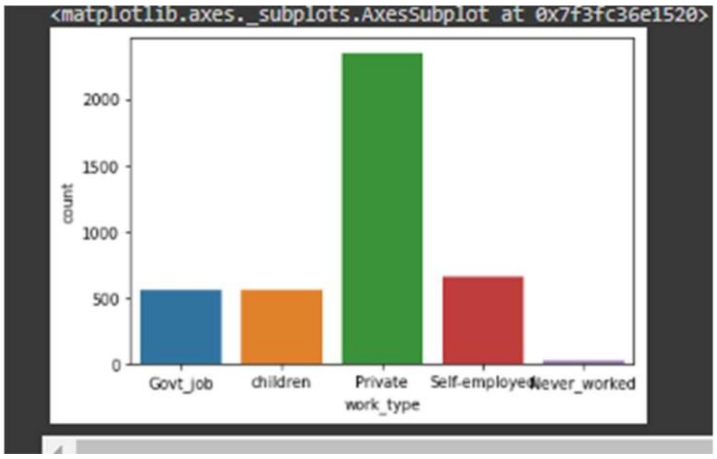
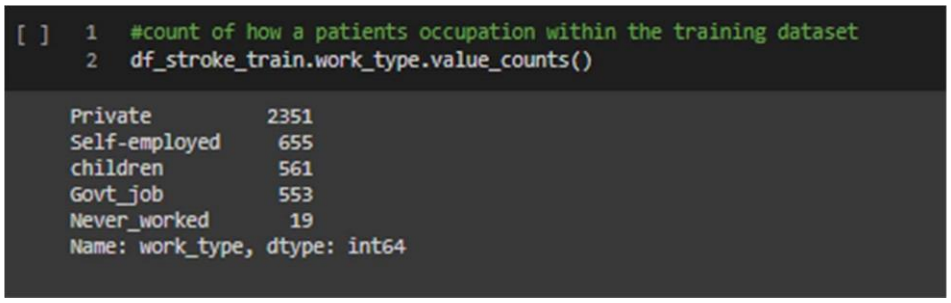


Figure 46 visualisation of what the candidate work type is within the training data set

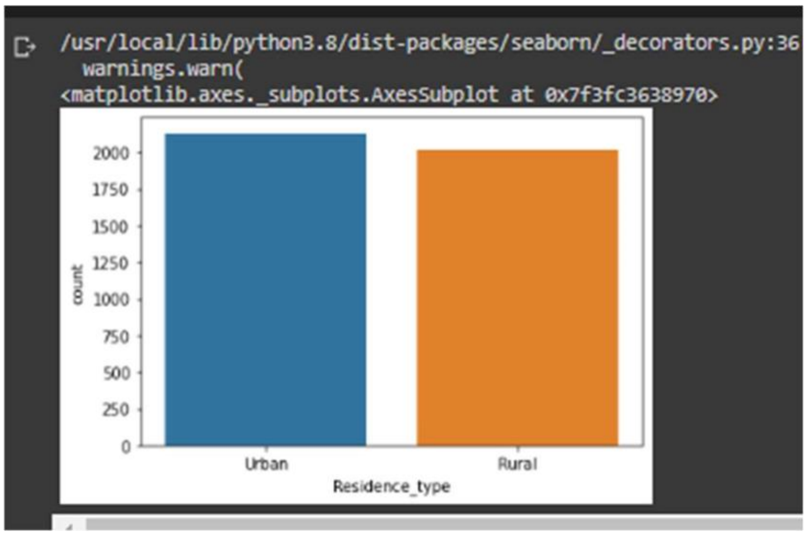
Figure 46 shows that the government job, children and self-employed are similar in number. However, there are more people in private work and very few within the data set that have never worked.



```
[ ]  #count of how a patients occupation within the training dataset
     df_stroke_train.work_type.value_counts()

     Private          2351
     Self-employed     655
     children          561
     Govt_job          553
     Never_worked       19
     Name: work_type, dtype: int64
```

Figure 47 figures showing how many within each category of the work_type

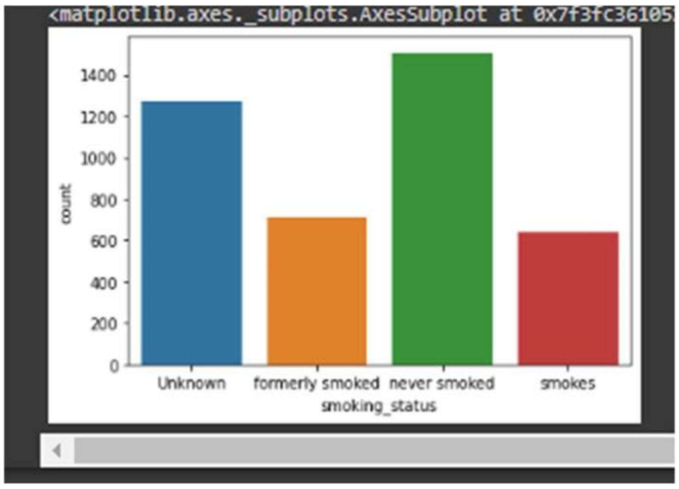
The actual count of these categories in Figure 47 suggests that more private workers have attended the clinics for treatment. Private could either mean they did not want to disclose what type of work they do, or it was that they were working as contractors. There is a small minority of people who attended the clinics for treatment as it never worked; this could either mean they were not of working age or had never worked but needed urgent treatment and had no other option but to seek treatment.



```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc3638970>
```

Figure 48 visualisation shows in the training set there are similar number of people living in rural and urban areas

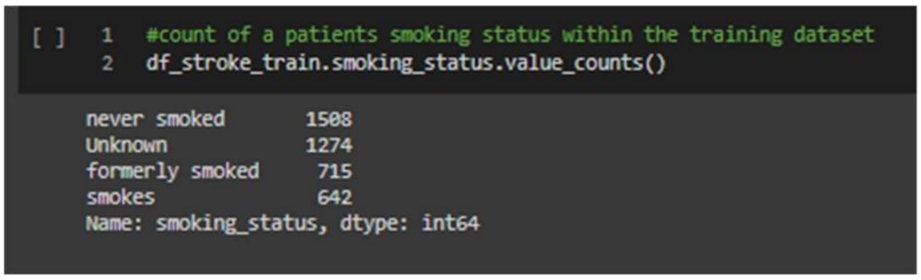
Figure 48 shows where people live nearly balanced within the data set. Therefore, it is safe to assume that this is not a factor. However, it will still need to be explored further.



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc36105
```

Figure 49 visualisation of training data for smoking status

Figure 49 shows that a large portion of the data has candidates that have never smoked. The data also shows that a large proportion of patients have an unknown status; this could either mean that they do not wish to give this information or that the researchers did not collect it.



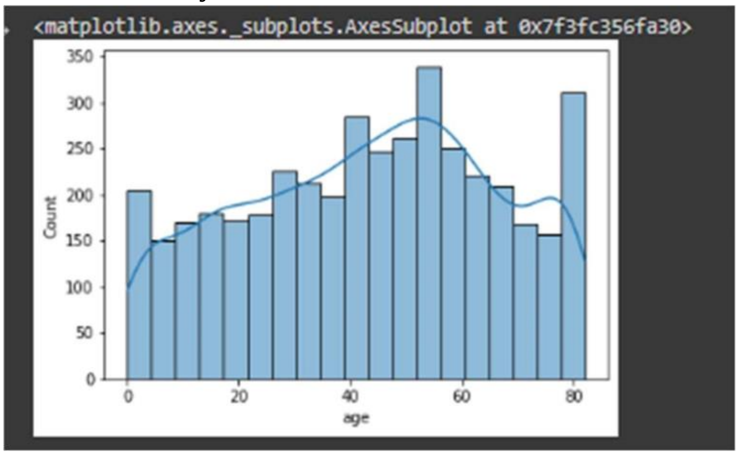
```
#count of a patients smoking status within the training dataset
df_stroke_train.smoking_status.value_counts()
```

```
never smoked       1508
Unknown            1274
formerly smoked     715
smokes              642
Name: smoking_status, dtype: int64
```

Figure 50 Code snippet of smoking_status

Figure 50 shows that 30% of the population of the data set has a smoking status of the unknown; therefore, if this was to be removed through the data cleaning process, we may lose a large amount of training data within the data set, which is already imbalanced. In addition, it is noted that a large majority of patients never smoked, which amounts to 36% of the training data.

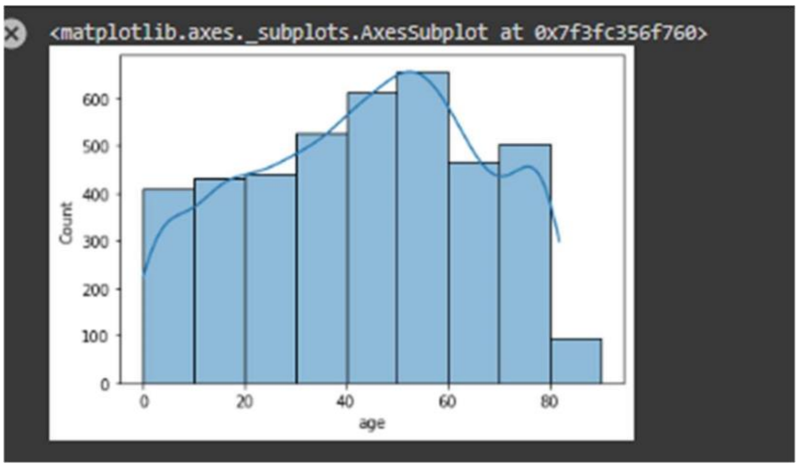
8.1.2 Categorical Univariate Analysis of the feature variables



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc356fa30>
```

Figure 51 univariate analysis of age within the training data set

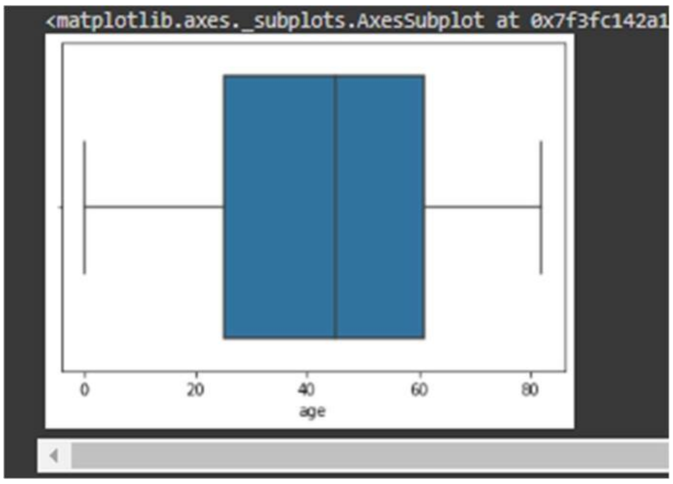
The univariate analysis of the age feature shows that it is not a gaussian bell curve but more skewed towards the right side. Thereby the data suggests that the population of the data set is older and more towards the working age group; this may ring true as in third world counties, access to medicine is expensive and to afford the fees, the patient would need to be earning.



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc356f760>
```

Figure 52 Histoplot of the age feature with a bin width as 10 showing the data more readable

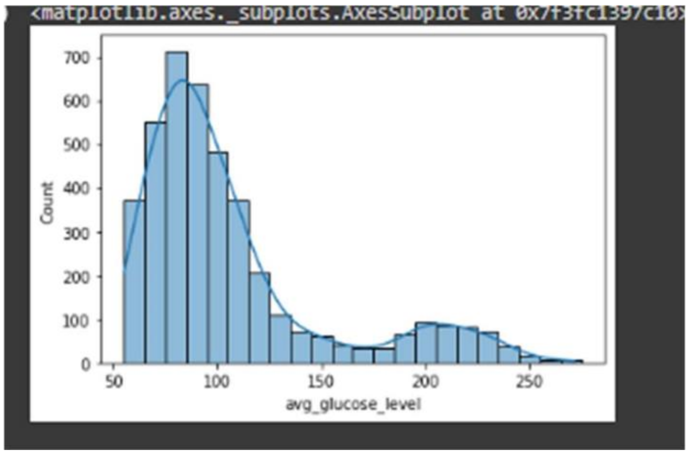
Figure 52 is better for analysis than Figure 51, as we can see in Figure 52 that there is a more extensive distribution of patients in the age range of 30-60. Whereas in Figure 51, it looks like there are 80-year-olds then there are 60-year-olds within the data set.



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc142a1
```

Figure 53 Box plot showing distribution of age of the patients

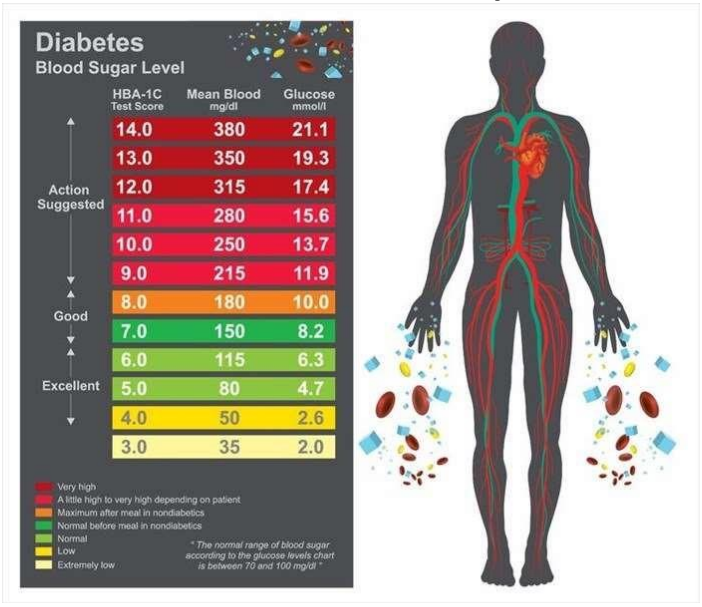
The box plot shows a better representation of the age feature within the data set that seems well distributed among patients between 25-62 years. Since the data was collected from Bangladesh, a third-world country, the people accessing health care would be of working age as healthcare is not free.



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc1397c10>
```

Figure 54 change in width for the bars in the histoplot for glucose levels which show a slightly clearer picture

Figure 54 shows that the distribution of the glucose levels is centred around 59-120, which is in the good hyper glycaemic index levels. However, a few are above this, which are patients who would be considered as having diabetes between 180-380, as shown in Figure 55.



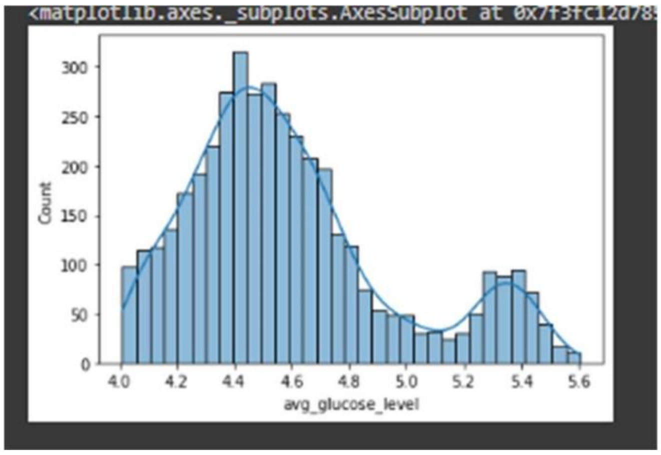
Diabetes
Blood Sugar Level

	HBA-1C Test Score	Mean Blood mg/dl	Glucose mmol/l
Action Suggested	14.0	380	21.1
	13.0	350	19.3
	12.0	315	17.4
	11.0	280	15.6
	10.0	250	13.7
	9.0	215	11.9
Good	8.0	180	10.0
	7.0	150	8.2
Excellent	6.0	115	6.3
	5.0	80	4.7
	4.0	50	2.6
	3.0	35	2.0

🟥 Very high
🟥 A little high to very high depending on patient
🟧 Maximum after meal in nondiabetics
🟩 Normal before meal in nondiabetics
🟩 Normal
🟨 Low
🟨 Extremely low

" The normal range of blood sugar according to the glucose levels chart is between 70 and 100 mg/dl "

Figure 55 illustration showing blood glucose levels (poole, 2023)



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fc12d78
```

Figure 56 transforming the distribution of the data using logarithmic transformation

Figure 56 shows that the logarithmic transformation of the average glucose data gives it more of a bell curve, and the shape of the data is better in terms of readability.

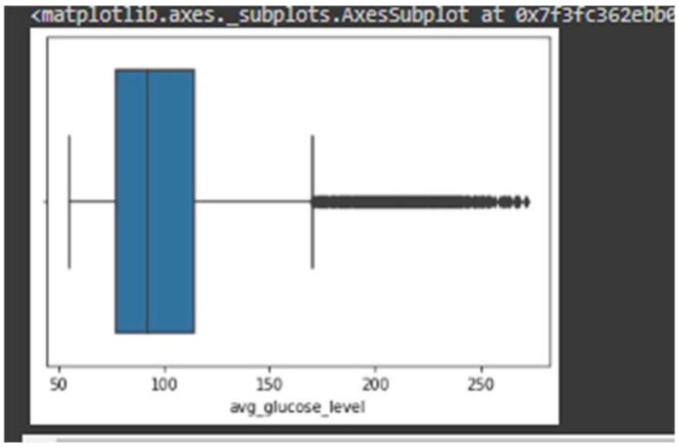


Figure 57 Box plot of the average glucose levels of patients

In Figure 57, we can see from the visualisation that there are outliers past 120 up to and including 270. The histoplot is an excellent visualisation as it demonstrates the "distribution of one or more variables by counting the number of observations that fall within the discreate bins" (Waskom, 2022). In comparison, the box plot indicates where the distribution of the population lies, therefore, is a better representation.

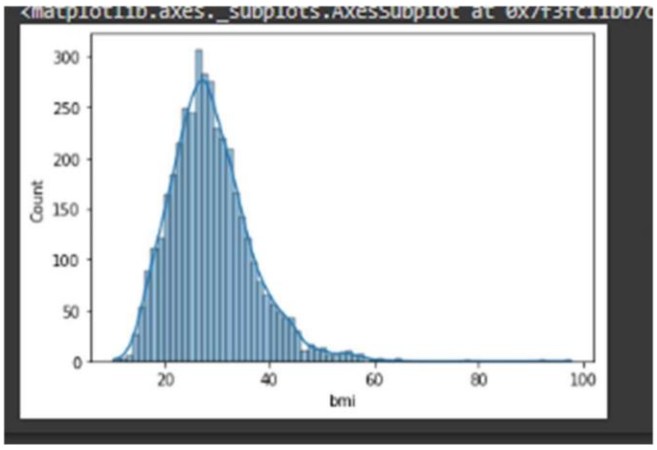


Figure 58 visualisation of BMI within the training data set with the data skewed to the left

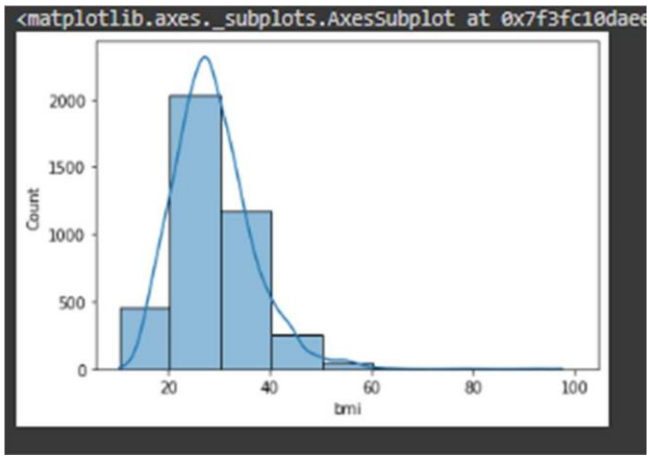


Figure 59 visualisation of BMI data with the bin width being 10 which now clearly shows that the 20 -40 is where most of the BMI lies

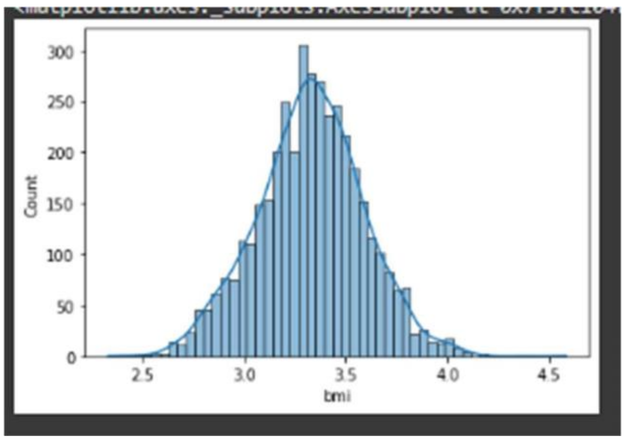


Figure 60 Logarithmic transformation of the data for BMI

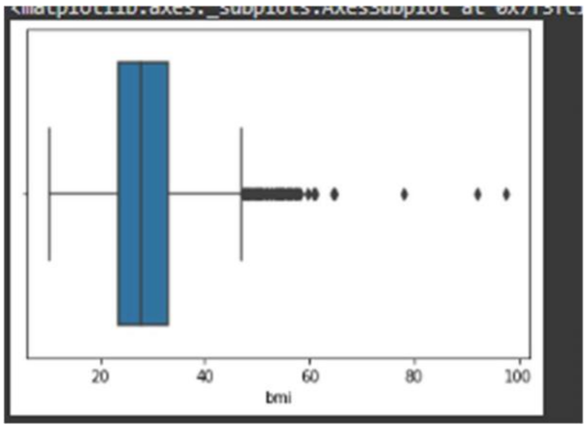
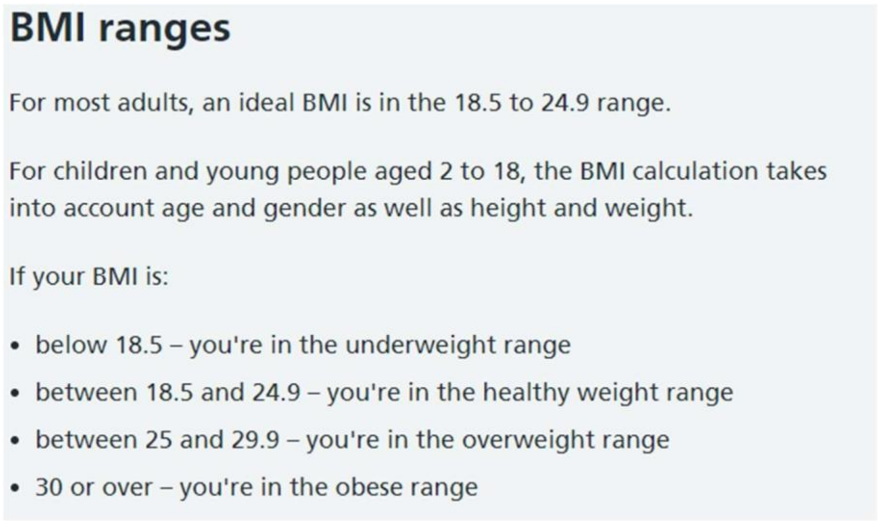


Figure 61 box plot for BMI data

Figure 58 to Figure 61 illustrate that the main distribution of the BMI for the patients lies between 20-35 with a few outliers. As can be seen in Figure 62, any BMI past 30 is in the obese range since a large amount of the patient population past 30 indicates that there are obese and obesity is a contributory factor of stroke (Baskini and Proios, 2023).



BMI ranges

For most adults, an ideal BMI is in the 18.5 to 24.9 range.

For children and young people aged 2 to 18, the BMI calculation takes into account age and gender as well as height and weight.

If your BMI is:

- below 18.5 – you're in the underweight range
- between 18.5 and 24.9 – you're in the healthy weight range
- between 25 and 29.9 – you're in the overweight range
- 30 or over – you're in the obese range

Figure 62 BMI ranges (NHS, 2022)

8.1.3 Categorical Feature Analysis between feature and target

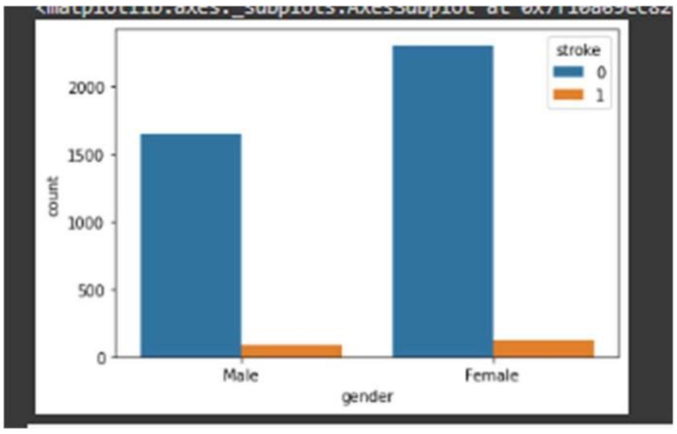
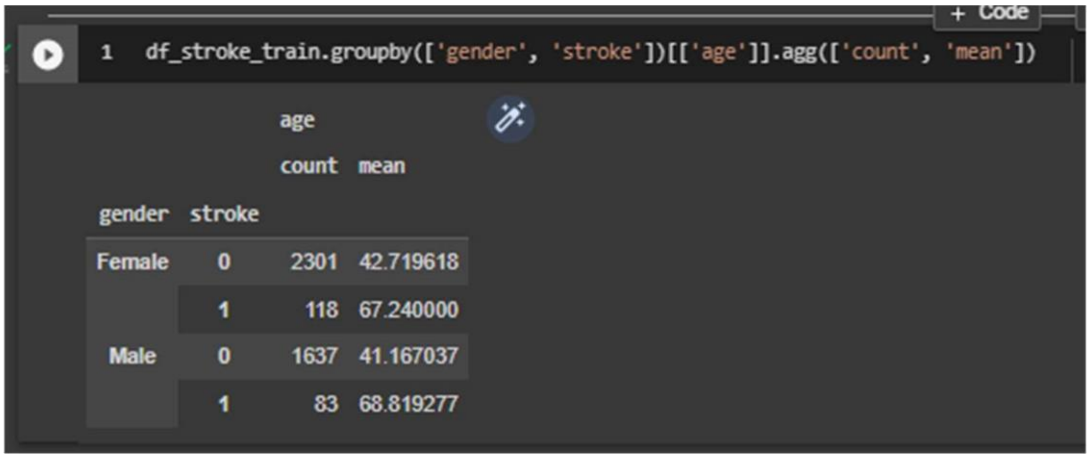


Figure 63 gender in comparison to having a stroke

The genders have an even distribution between them of having a stroke, with females slightly more likely than males but not by a lot. However, this is not surprising as there are more females than males within the data set.



```
df_stroke_train.groupby(['gender', 'stroke'])[['age']].agg(['count', 'mean'])
```

```
                age
                count  mean
gender  stroke
Female  0       2301   42.719618
        1        118   67.240000
Male    0       1637   41.167037
        1         83   68.819277
```

Figure 64 code snippet showing how many males and how many females suffered a stroke

However, Figure 64 shows that 118 females have a stroke diagnosis, significantly more than males who have 83. In addition, the mean age for both genders is in the late 60s, suggesting that patients of retirement age are more likely to suffer a stroke.

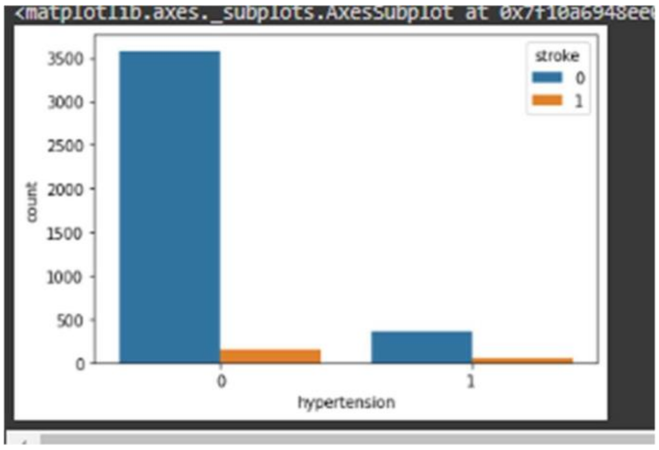
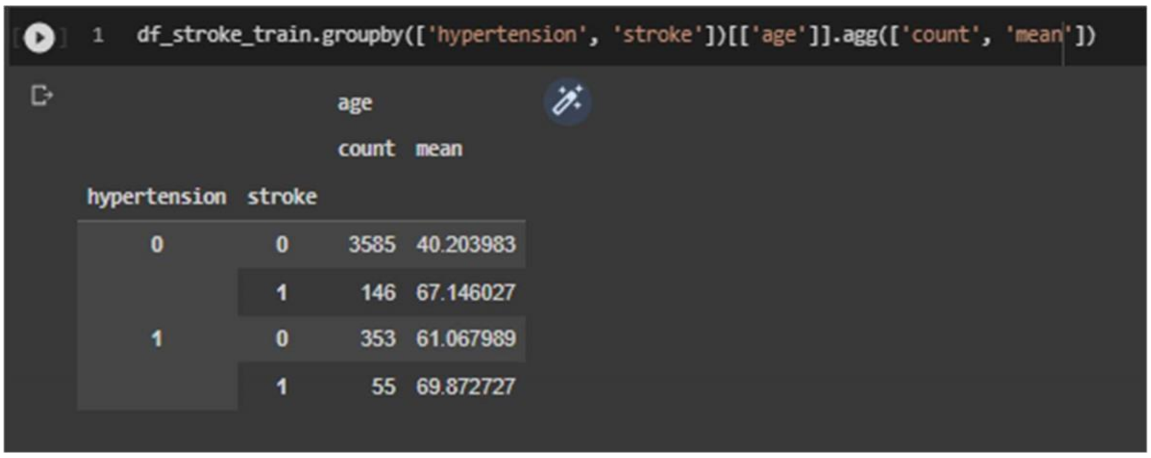


Figure 65 hypertension and having a stroke

Since looking at the data set, we know that it is imbalanced; as suggested before, this is correct as this would not expect a large majority of patients to have hypertension. However, comparing patients with hypertension to stroke, most patients who do not have hypertension seem to suffer from a stroke; this is opposite to what was hypothesised in question 2.



```
df_stroke_train.groupby(['hypertension', 'stroke'])[['age']].agg(['count', 'mean'])
```

```
                          age
                          count  mean
hypertension  stroke
0             0           3585   40.203983
              1            146   67.146027
1             0            353   61.067989
              1             55   69.872727
```

Figure 66 hypertension with stroke comparison and mean age

Figure 66 shows that not having hypertension, the patients in this dataset are more likely to be diagnosed with a stroke then opposed to those with hypertension. In addition, the mean age for stroke remains in the late 60s.

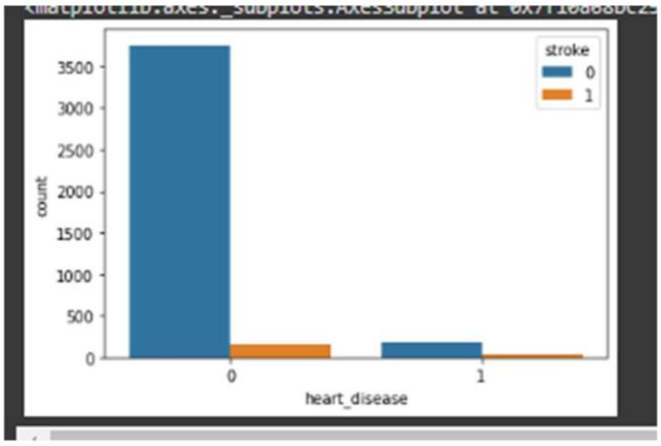
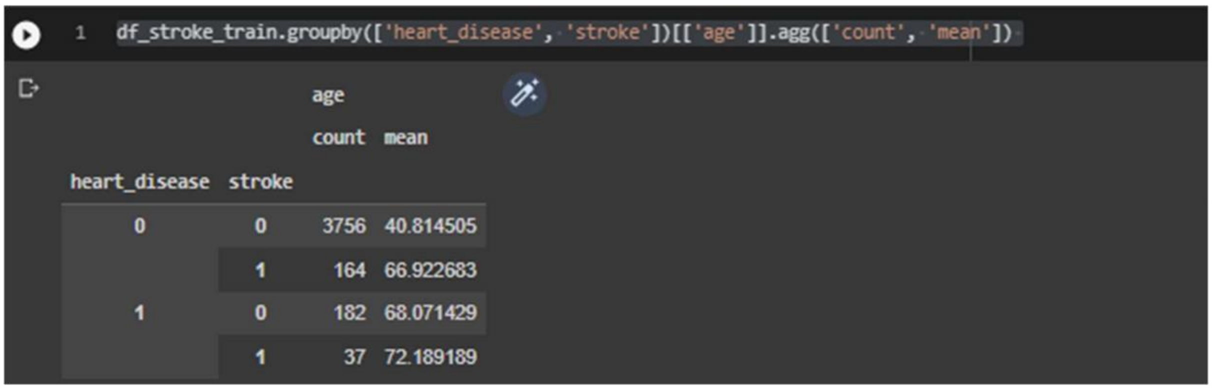


Figure 67 heart disease compared to having a stroke

However, comparing patients with heart disease to stroke, most patients who do not have heart disease seem to suffer from a stroke; this is opposite to what was hypothesised in question 2.



```
df_stroke_train.groupby(['heart_disease', 'stroke'])[['age']].agg(['count', 'mean'])
```

```
                           age
                           count  mean
heart_disease  stroke
0              0           3756   40.814505
               1            164   66.922683
1              0            182   68.071429
               1             37   72.189189
```

Figure 68 heart disease correlation with stroke and the mean age of the patient

Figure 68 shows that not having heart disease, the patients in this dataset are more likely to be diagnosed with a stroke then opposed to those with heart disease. In addition, the mean age for stroke remains in the late 60s which is similar to hypertension.

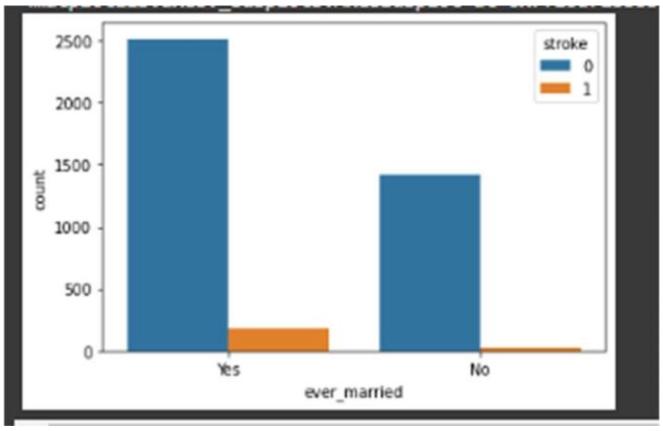
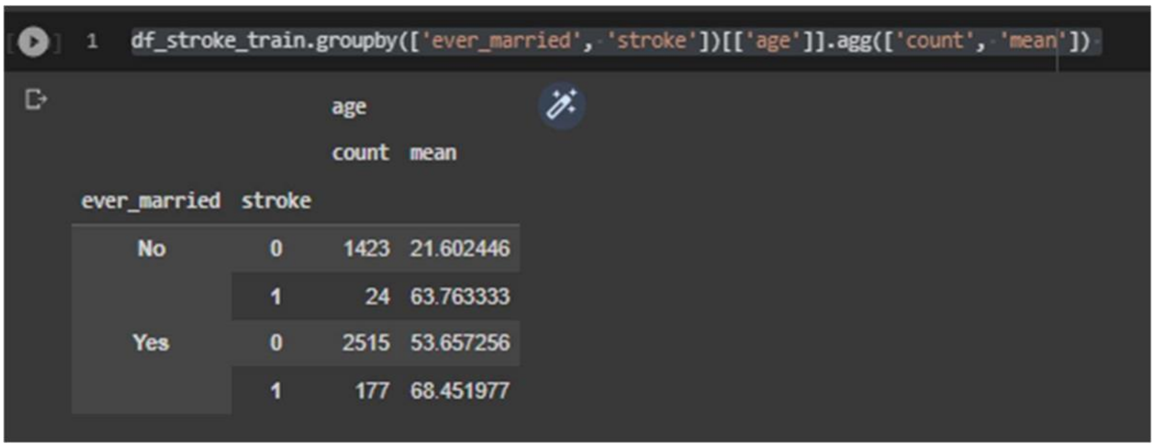


Figure 69 comparison to being married and having a stroke

The graph indicates that being married you are more likely to have a stroke then not being married.



```
df_stroke_train.groupby(['ever_married', 'stroke'])[['age']].agg(['count', 'mean'])
```

```
                         age
                       count       mean
ever_married  stroke
          No       0    1423  21.602446
                   1      24  63.763333
         Yes       0    2515  53.657256
                   1     177  68.451977
```

Figure 70 Code snippet showing high number of patients who are married suffer with a stroke

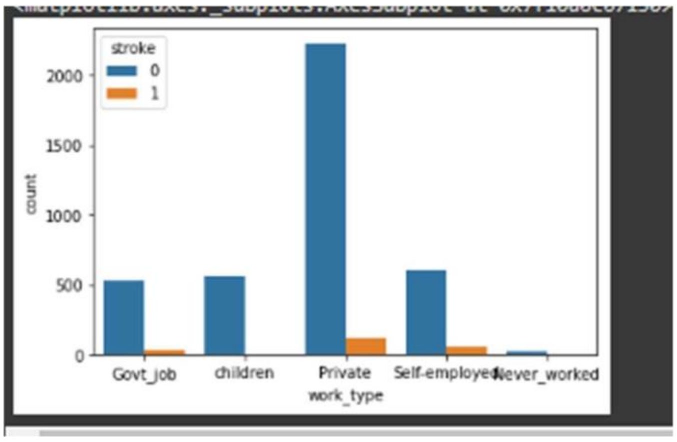
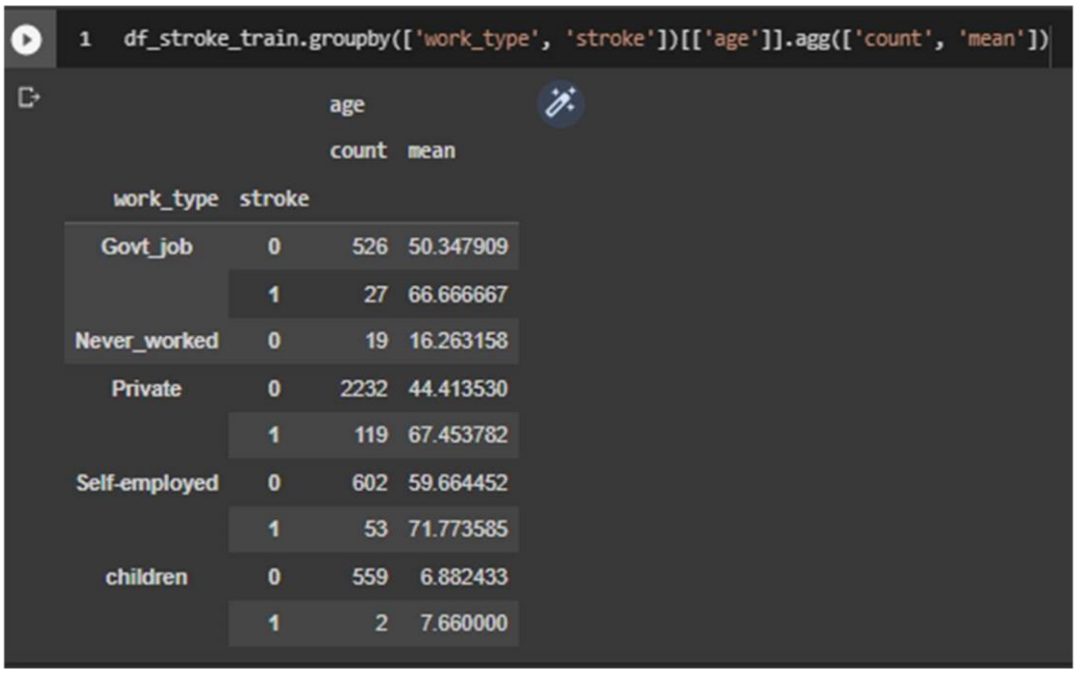


Figure 71 comparison of having a stroke with work type

The distribution indicates that a private work type which is unclear what it is, contributes to having a stroke more than the other types of work. It is also clear that the person is less likely to have a stroke by never working and having children. The never working is understandable as there is no stress; however, having children is surprising as the stress of children would contribute to having a stroke.



```
df_stroke_train.groupby(['work_type', 'stroke'])[['age']].agg(['count', 'mean'])
```

```
                          age
                          count  mean
work_type     stroke
Govt_job      0            526   50.347909
              1             27   66.666667
Never_worked  0             19   16.263158
Private       0           2232   44.413530
              1            119   67.453782
Self-employed 0            602   59.664452
              1             53   71.773585
children      0            559    6.882433
              1              2    7.660000
```

Figure 72 work_type analysis between stroke and mean age

Figure 72 makes the comparison much more precise; the assumption taken from Figure 71 was incorrect as being self-employed has more patients diagnosed with a stroke than the other groups. In addition, Children are not patients with children but actual children whose mean age of 7 years old have a stroke diagnosis. As mentioned previously never worked as no patients have suffered from a stroke and the mean age suggests that this is also adolescents, with a mean age of 19 years old.

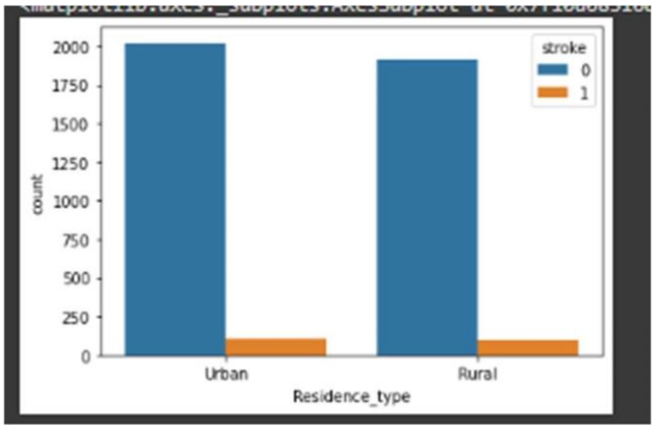
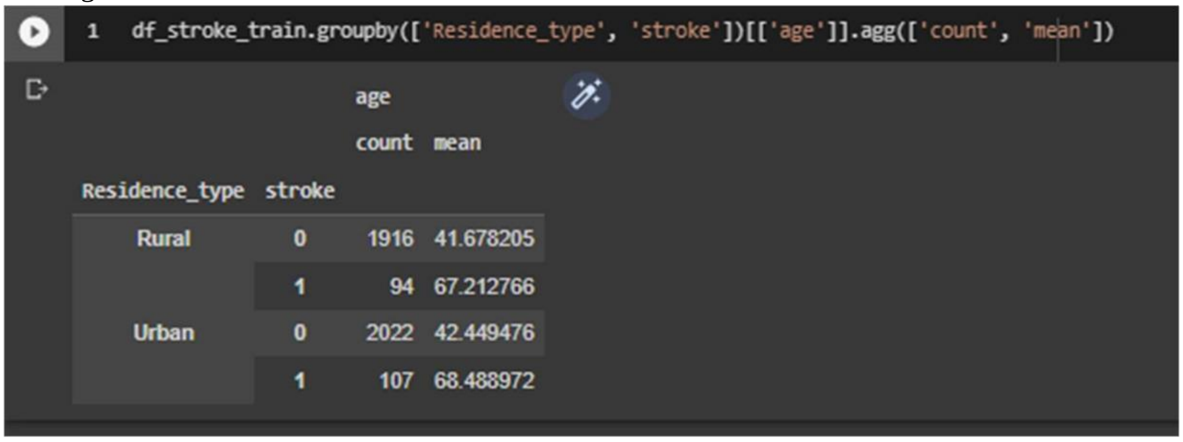


Figure 73 comparison of residence type to having a stroke

The graph is evenly distributed and indicates that where a person resides does not correlate with a stroke occurring.



```
df_stroke_train.groupby(['Residence_type', 'stroke'])[['age']].agg(['count', 'mean'])
```

```
                          age
                          count  mean
Residence_type  stroke
Rural           0          1916   41.678205
                1            94   67.212766
Urban           0          2022   42.449476
                1           107   68.488972
```

Figure 74 Residence_type compared to stroke and mean age

The above code snippet shows no correlation between where a patient lives and suffering from a stroke.

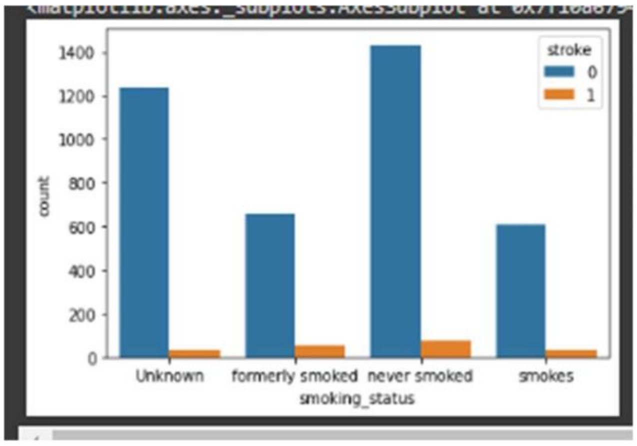
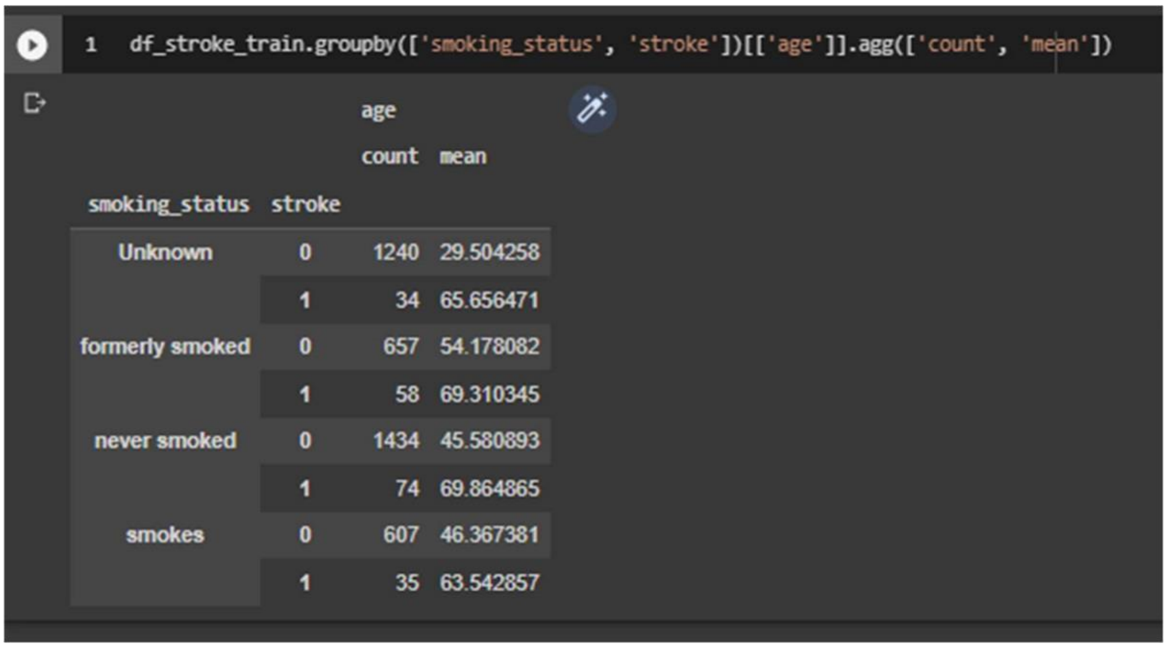


Figure 75 smoking status in comparison to how many people had a stroke

The above figure shows that people who never smoked had a stroke compared to the other categories.



```
df_stroke_train.groupby(['smoking_status', 'stroke'])[['age']].agg(['count', 'mean'])
```

```
                        age
                        count  mean
smoking_status  stroke
Unknown         0       1240   29.504258
                1         34   65.656471
formerly smoked 0        657   54.178082
                1         58   69.310345
never smoked    0       1434   45.580893
                1         74   69.864865
smokes          0        607   46.367381
                1         35   63.542857
```

Figure 76 Smoking_status compared with stroke and mean age

However, looking at Figure 76 it is clear that former smokers are more likely to have a stroke diagnosis than those currently smoking. Also, the mean age for a stroke diagnosis of a former smoker is 69, around retirement age.

8.1.4 Numerical Feature Analysis between the Feature and stroke

`<seaborn.axisgrid.FacetGrid at 0x7f10a6702ee0>`

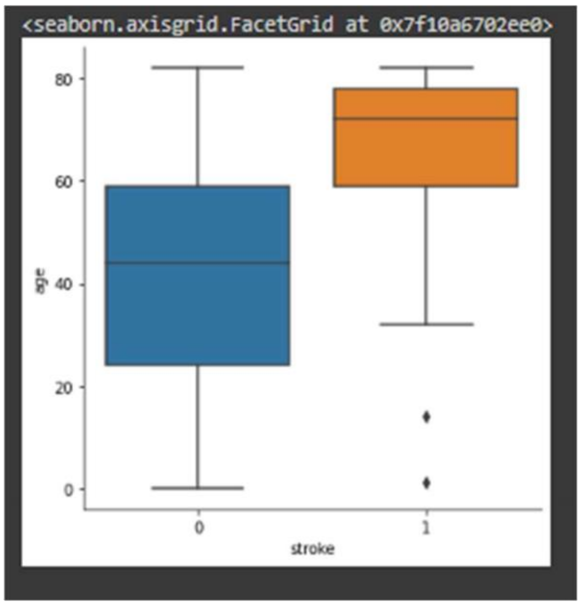
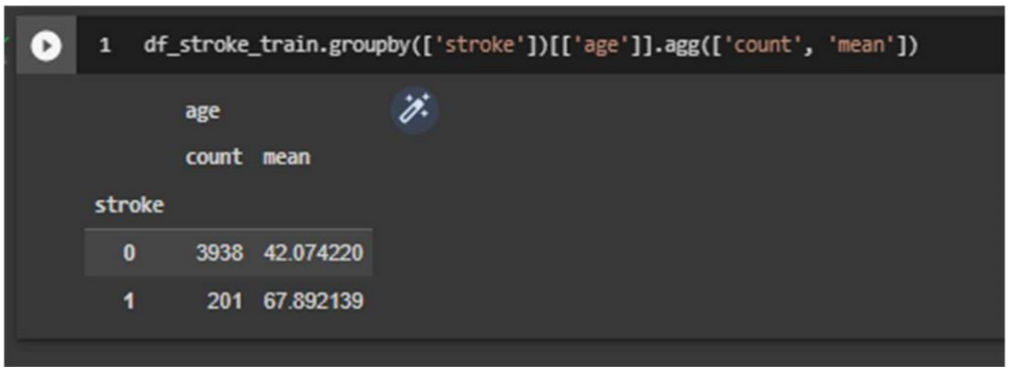


Figure 77 comparison of age with stroke

Age in this data set plays a part in having a stroke, as can be seen in the above box plot. The older the person, as the retirement age is 60 plus, the more likely they are to have a stroke, with a few outliers in the younger ages that are also contained within the data set as having a stroke.



```
df_stroke_train.groupby(['stroke'])[['age']].agg(['count', 'mean'])
```

```
         age
         count  mean
stroke
0        3938   42.074220
1         201   67.892139
```

Figure 78 stroke compared to age

The above code snippet displays that the mean age for a patient to suffer from a stroke is 67, which is the retirement age.

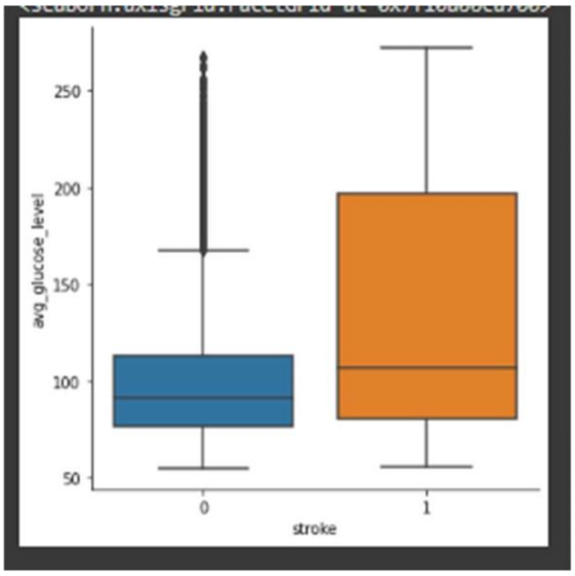
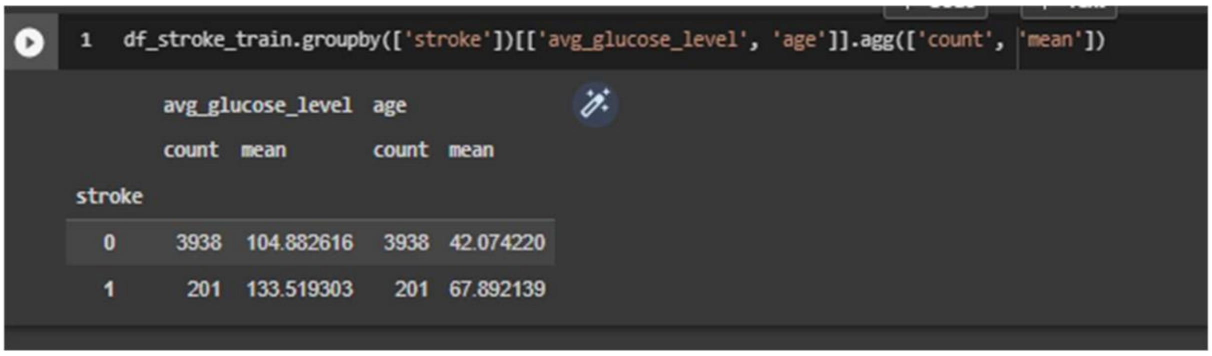


Figure 79 box plot comparison of having high glucose to having a stroke

The box plot demonstrates that having high glucose is a contributory factor to having a stroke. Conversely, those with lower glucose levels have a lower chance of having a stroke, but there are outliers where glucose levels are high but do not have a stroke diagnosis.



```
df_stroke_train.groupby(['stroke'])[['avg_glucose_level', 'age']].agg(['count', 'mean'])
```

```
        avg_glucose_level      age
        count  mean            count  mean
stroke
  0     3938   104.882616      3938   42.074220
  1      201   133.519303       201   67.892139
```

Figure 80 mean glucose levels compared to a stroke and no stroke

As we can understand from the above code snippet, the average glucose level for a stroke diagnosis is 133, with a mean age of 67. Therefore, age is a significant factor in a stroke diagnosis as 60 and above, and there is more chance of a stroke.

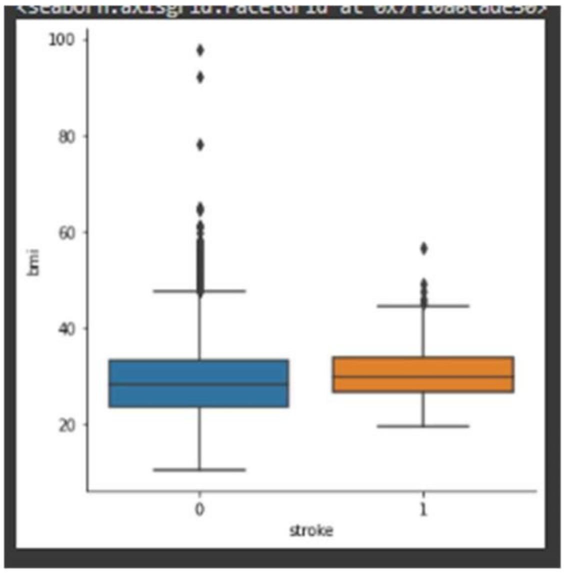


Figure 81 box plot showing bmi in comparison to a stroke

The above figure demonstrates that having a high BMI contributes to having a stroke. Nevertheless, in the box plot, we can see quite a few outliers, which show that a significantly high BMI does not contribute to a stroke.

8.1.5 Multivariate analysis: feature to feature

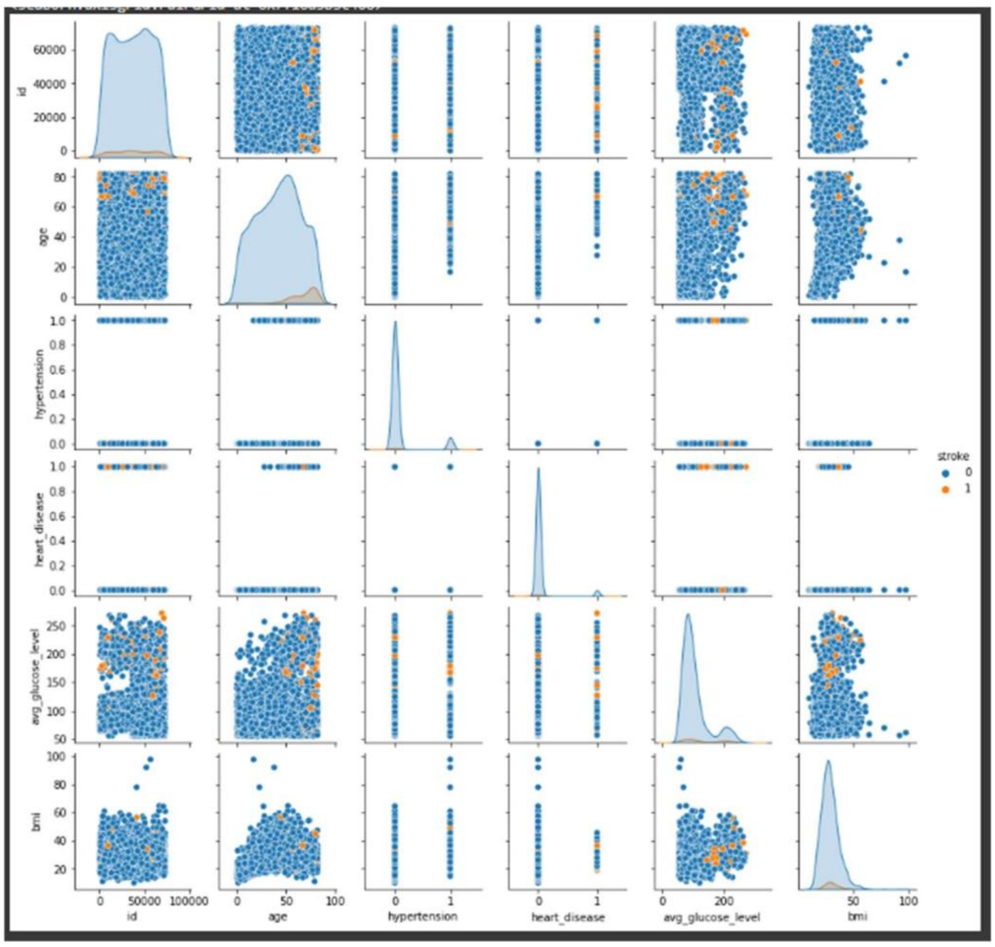


Figure 82 pair plot of the stroke training dataset

The pair plot illustrates the correlation between features and if a stroke diagnosis is given. The observations are listed below from the pair plot:

1. Looking at age and average glucose, we can tell that the older the patient, the higher the glucose and the more chance of suffering a stroke.
2. Also, looking at the plots can be seen in the figure above that the higher the Glucose levels, the higher the BMI more likely the patient is to suffer from a stroke.
3. The higher the Glucose levels, the more patients have heart disease and the more likely the chance of a stroke occurring.

8.1.5.1 Heatmap correlation between numerical features

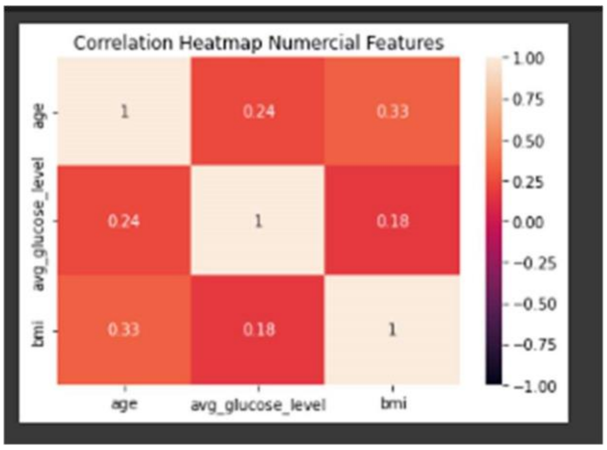


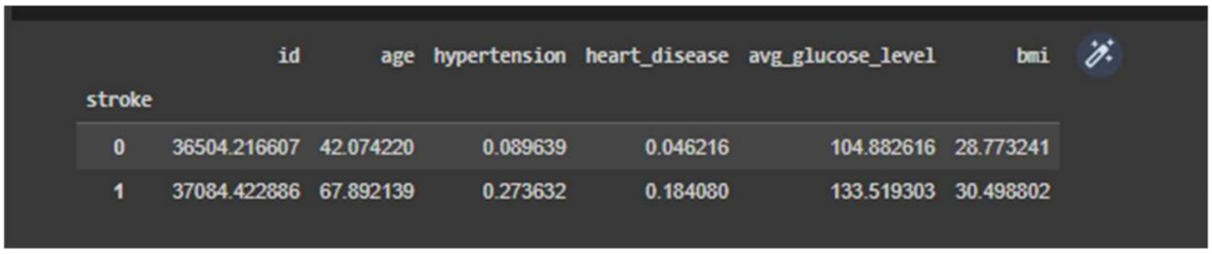
Figure 83 numerical features heat map correlation between age, average glucose and BMI The

heat map shows a strong positive correlation between age and BMI.

As for age and average glucose, there is a positive correlation but not as strong as BMI and age.

There is a weak correlation between average glucose and BMI which is surprising as having a high BMI would expect that glucose levels would be high.

8.1.5.2 Correlation analysis between the features



```
                  id        age  hypertension  heart_disease  avg_glucose_level        bmi
stroke
     0  36504.216607  42.074220      0.089639       0.046216         104.882616  28.773241
     1  37084.422886  67.892139      0.273632       0.184080         133.519303  30.498802
```

Figure 84 Comparison of stroke to age, hypertension, heart disease, average glucose and bmi

Figure 56 shows that the mean value of a person having a stroke is 67, which is the same through this EDA that 60 or above is a highly correlated factor to stroke.

The mean BMI of 30.50 also corresponds to the fact that being obese increases the chances of a stroke; however, this may not always be the case, as the mean BMI for a no-stroke diagnosis is 28.77, which is also classed as obese. Therefore, BMI has a weak correlation with a risk of a stroke occurring.

The average mean for glucose levels for a stroke diagnosis is 133; a patient having this glucose level is classed as a person with diabetes; with the non-occurrence of a stroke diagnosis, the glucose levels are in the Normoglyemia range, which is between 72-108.

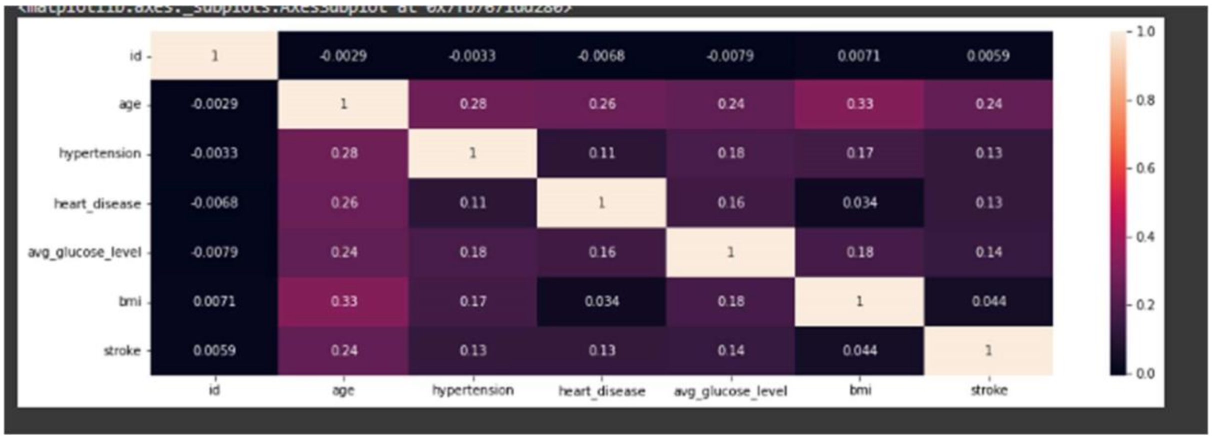


Figure 85 heatmap of feature to feature in the training data set

The heatmap demonstrates that age and stroke are highly correlated.

Hypertension, heart disease and average glucose correlate to stroke, but this is a weak correlation. However, this could be because the dataset is imbalanced.

Whereas BMI and stroke have a low correlation.

The heatmap also provides reassurance that the ID column is unwanted noise as it has a very low correlation between all features.

8.1.6 EDA Feature analysis between features

This is a continuation from the EDA feature analysis.

Comparing age and work_type what the mean age is and what work the patients do.

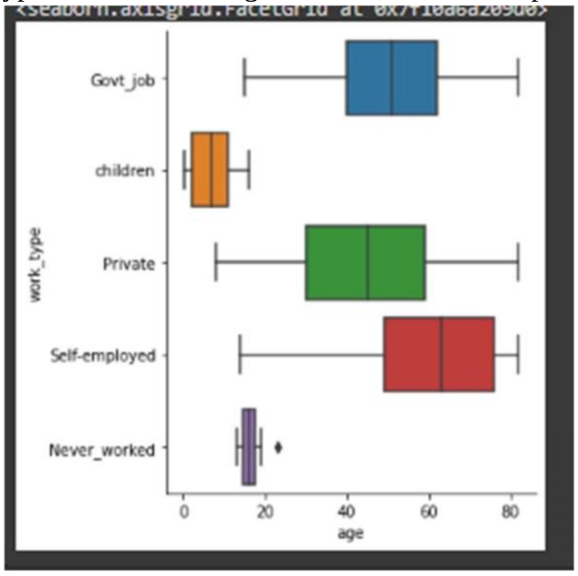


Figure 86 work_type and age

The age range between private and govt_job is similar however the age of self-employed looks past the age of 60 into retirement age.

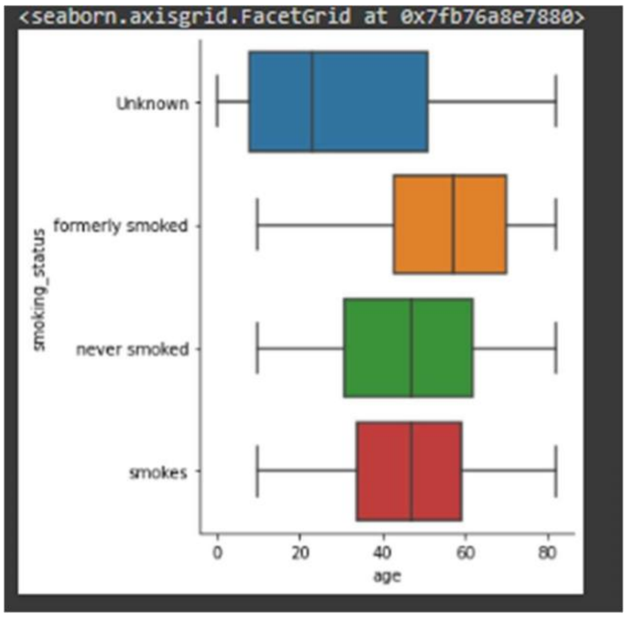


Figure 87 comparision between age and smoking_status

The above figure demonstrates that former smokers are of retirement age, and since smoking is also a contributory factor of a stroke, this may explain why older people have strokes.

9 Identification of appropriate evaluation techniques

F1 Score: The F1 score is a metric that combines both precision and recall into one measure. “It is a harmonic mean of precision and recall” (Zhu and Miao, 2022), so it will consider both false positives and false negatives. Therefore, the F1 score helps evaluate the performance of a classifier as it gives a better indication of the overall performance than either precision or recall alone.

$$F\ Measure = \frac{2}{recall \quad + precision}$$

Equation 15 The harmonic Mean of precision and recall (Zhu and Maio, 2022)

Precision is a metric used to evaluate the performance of a model by measuring the proportion of true positives correctly identified by the model therefore the patient have not stroke diagnosis. It is a valuable metric for evaluating the performance of our models as it helps us understand how well our models can correctly identify cases of the target variable.

Confusion Metrics: Confusion metrics measure how often a classifier predicts a particular class when the actual class is different. Confusion Metrics help evaluate a classifier's performance by indicating how often the model makes incorrect predictions.

Predicted Class

TrueClass		
	🟩 True Positive (TP)	🟥 False Negative (FN)
	🟥 False Positive (FP)	True Negative (TN)

Table 11 Confusion Matrix

Recall: Recall measures how often a classifier correctly identifies an instance of a specific class. It is a “fraction of all real anomalies that are successfully detected” (Alam. et al., 2018). It helps evaluate the performance of a classifier as it indicates how often the model is predicting the correct class.

$$Recall = \frac{TP}{(TP + FN)}$$

Equation 16 Calculation of Recall

$$Recall = \frac{no\ stroke}{(no\ stroke + (false\ stroke\ but\ actually\ the\ patient\ has\ no\ stroke))}$$

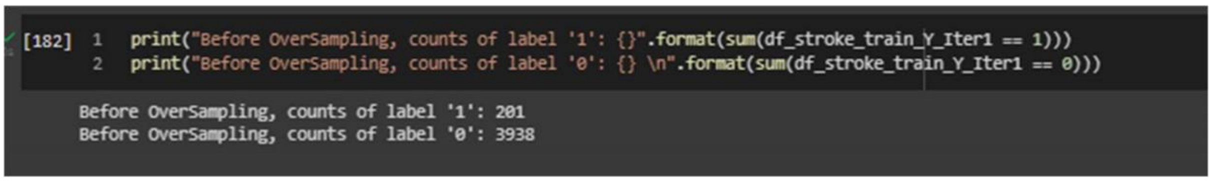
Overall, these evaluation metrics will give us a clearer picture of the relative performance of our models. In addition, they will help us identify areas of improvement and will enable us to select the best model based on its performance on these metrics.

10 Iterations to the model

The model iterations and changes made to see if the accuracy and metrics could be improved.

10.1.1 SMOTE oversampling to balance the imbalance in the data set

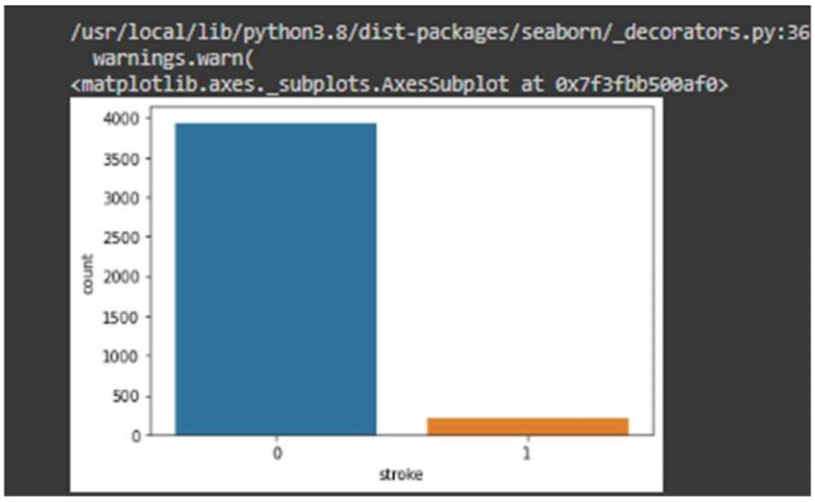
SMOTE over sampling was used to balance the data as the target variable “stroke” was imbalanced.



```
[182] print("Before OverSampling, counts of label '1': {}".format(sum(df_stroke_train_Y_Iter1 == 1)))
      print("Before OverSampling, counts of label '0': {} \n".format(sum(df_stroke_train_Y_Iter1 == 0)))

Before OverSampling, counts of label '1': 201
Before OverSampling, counts of label '0': 3938
```

Figure 88 code snippet of "stroke" variable before over sampling

Figure 88 shows the ‘1’ patient being diagnosed with a stroke as being 201 and ‘0’ not being diagnosed as having a stroke as 3938.

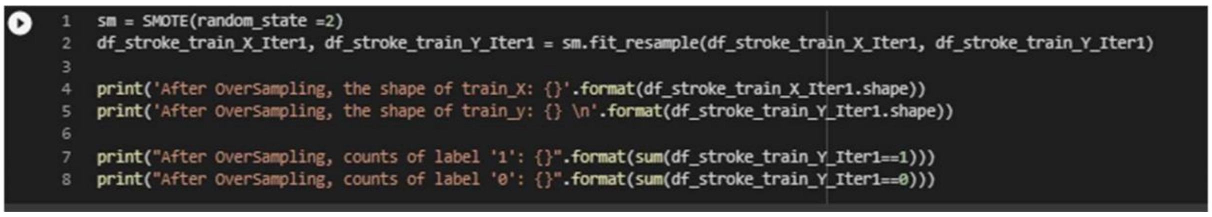


```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f3fbb500af0>
```

Figure 89 target variable "stroke" imbalanced Figure 89 illustrates the imbalance of the ‘stroke’ data.

The imbalanced as discussed in the main document is correct ground truth as we would not expect large volumes of patients in the hospitals to be diagnosed with stroke.

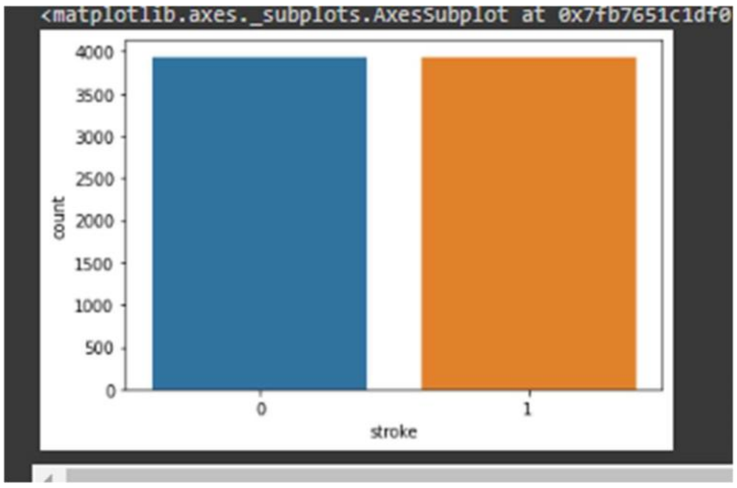
The following code snippet shows that the training data for SMOTE is oversampling the data with a Random_state=2 to ensure that each time it runs it does the same and does not change.



```
sm = SMOTE(random_state =2)
df_stroke_train_X_Iter1, df_stroke_train_Y_Iter1 = sm.fit_resample(df_stroke_train_X_Iter1, df_stroke_train_Y_Iter1)

print('After OverSampling, the shape of train_X: {}'.format(df_stroke_train_X_Iter1.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(df_stroke_train_Y_Iter1.shape))

print("After OverSampling, counts of label '1': {}".format(sum(df_stroke_train_Y_Iter1==1)))
print("After OverSampling, counts of label '0': {}".format(sum(df_stroke_train_Y_Iter1==0)))
```

Figure 90 Code snippet of oversampling the training data for iteration 1

The data is balance occurs by balancing the minority class in this data set being ‘1’ patients who have been diagnosed with a stroke. The process of the balancing of the minority class happens with a random mean values.



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb7651c1df0
```

10.1.2 Feature Scaling

10.1.3 Isolation Forest

10.1.4 Removing Residental_type

10.1.5 Label Encoding