COURSEWORK 1

MODULE TITLE- PROGRAMMING FOR DATA ANALYTICS AND AI

MODULE CODE- M33145

TOPIC- Comprehensive Statistical Analysis and Machine Learning for Stroke Dataset: A Multi-Methodological Approach to Classification, Regression, and Clustering

Submitted By-
Student ID: UP2280648

DATE OF SUBMISSION- 15th January, 2025

Table of Contents

Chapter 1. Descriptive Analytics **4**
1.1 Basic Statistics of all Attributes 4
1.1.1 Basic Statistics of ‘gender’ Attribute 4
1.1.2 Basic Statistics of ‘age’ Attribute 5
1.1.3 Basic Statistics of ‘hypertension’ Attribute 6
1.1.4 Basic Statistics of ‘heart_disease’ Attribute 7
1.1.5 Basic Statistics of ‘ever_married’ Attribute 8
1.1.6 Basic Statistics of ‘work_type’ Attribute 9
1.1.7 Basic Statistics of ‘Residence_type’ Attribute 10
1.1.8 Basic Statistics of ‘avg_glucose_level’ Attribute 11
1.1.9 Basic Statistics of ‘bmi’ Attribute 12
1.1.10 Basic Statistics of ‘smoking_status’ Attribute 14
1.1.11 Basic Statistics of ‘stroke’ Attribute 15
1.2. Visualisations 16
1.2.1 Boxplot to Analyse the Relationship between ‘Stroke’ and ‘Age’ 16
1.2.2 Violinplot to Analyse the Relationship between ‘Stroke’ and ‘BMI’ (EXTRA) 18
1.2.3 Scatterplot to Analyse the Relationship between ‘bmi’ and ‘avg_glucose_level’ 20
1.2.4 Boxplot to Analyse the Relationship between ‘Hypertension’ and ‘Age’ 21
1.2.5 Grouped Bar Chart to Analyse the Relationship between ‘Heart Disease’ and ‘Stroke’ 23
Chapter 2. Data Preparation **24**
2.1 Deleting the ‘id’ Column 24
2.2 Conversion of Necessary Columns to Numerical 24
2.3 Treating the Outliers 25
2.3.1 Treating the outliers for the ‘bmi’ Attribute 25
2.3.2 Treating the outliers for the ‘avg_glucose_level’ Attribute 26
2.4 Imputing missing values for Regression 27
2.5 Segregating ‘Features’ and ‘Target’ column for Regression 28
2.6 Normalisation of the Features for Regression 28
2.7 Splitting the Data into Train and Test for Regression 28
2.8 Imputing missing values for Classification 28
2.9 Segregating ‘Features’ and ‘Target’ for Classification 30
2.10 Normalisation of the Features for Classification 30
2.11 Tackling Class Imbalance of the target column ('stroke') using SMOTE (Synthetic Minority Oversampling Technique) (EXTRA) 30
2.12 Splitting the Balanced Dataset into Train and Test set for Classification 31
2.13 Normalisation of the Features for Clustering 32
Chapter 3. Classification **33**

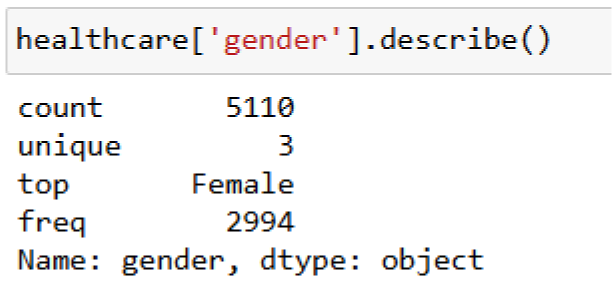
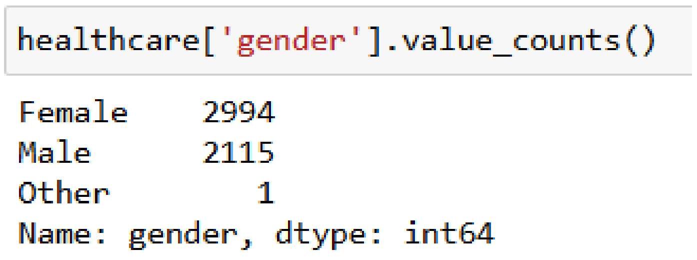
3.1 Random Forest Classifier: Hyperparameter Tuning and Model Evaluation................................ 33
3.2 Bagging Classifier: Hyperparameter Tuning and Model Evaluation.. 37
3.3 KNeighborsClassifier: Hyperparameter Tuning and Model Evaluation...................................... 41
3.4 AdaBoostClassifier: Hyperparameter Tuning and Model Evaluation (EXTRA).........................45
3.5 Comparison and Analysis of the Classifiers used based on F1 Score...49
3.5.1 Visualization of the classifiers and ranking them on the basis of F1 Score of each class (class 0 and class 1)..49
3.5.2 Analysis and Comparision... 49
Chapter 4. Regression...51
4.1 Gradient Boosting Regressor: Hyperparameter Tuning and Model Evaluation (EXTRA)..........51
4.2 Random Forest Regressor: Hyperparameter Tuning and Model Evaluation................................53
4.3 Comparision and Analysis of the Regression Models..56
Chapter 5. Clustering.. 58
5.1 K-Means Clustering...58
5.2 KMedoids Clustering (EXTRA)...68
5.3 Comparision and Analysis of the Clustering Models...76
References.. 77

Chapter 1. Descriptive Analytics

1.1 Basic Statistics of all Attributes

Before starting with the basic statistics of each attribute, it's important to mention that the default data types of 'hypertension', 'heart_disease', and 'stroke' columns are given as numeric data type (int64). However, the actual nature of the above-mentioned columns is categorical. As a result, these columns are first converted from numerical (int64) to categorical (object). This conversion is done because our analysis will only be correct and logical when these categorical attributes are treated as categorical while describing using the describe() function.
Now, we proceed to describe each attribute present in our dataset.

1.1.1 Basic Statistics of 'gender' Attribute

Fig 1.1.1 (A)	*Fig 1.1.1 (B)*
	

```
healthcare['gender'].describe()

count       5110
unique         3
top       Female
freq        2994
Name: gender, dtype: object
```

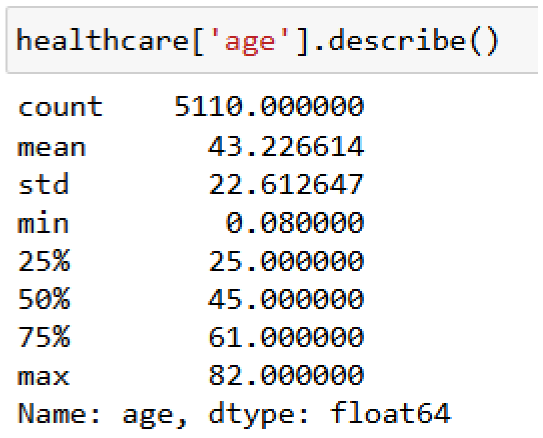
```
healthcare['gender'].value_counts()

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

Table 1.1.1: Basic Statistics of 'gender' Attribute (Refer Fig 1.1.1(A) and 1.1.1 (B))		
STATISTIC	**VALUE**	**DESCRIPTION**
count	5110	There are 5110 entries in the 'gender' column
unique	3	There are 3 unique values in the 'gender' column
top	Female	'Female' is the most frequently occurring value in the 'gender' column
frequency	2994	The number of times 'Female' appears in the 'gender' column is 2994. This represents the mode.

Male	2115	The number of times ‘Male’ appears in the ‘gender’ column is 2115
Other	1	The number of times ‘other’ appears in the ‘gender’ column is 1.

1.1.2 Basic Statistics of ‘age’ Attribute

Fig 1.1.2



```
healthcare['age'].describe()
```

```
count    5110.000000
mean       43.226614
std        22.612647
min         0.080000
25%        25.000000
50%        45.000000
75%        61.000000
max        82.000000
Name: age, dtype: float64
```

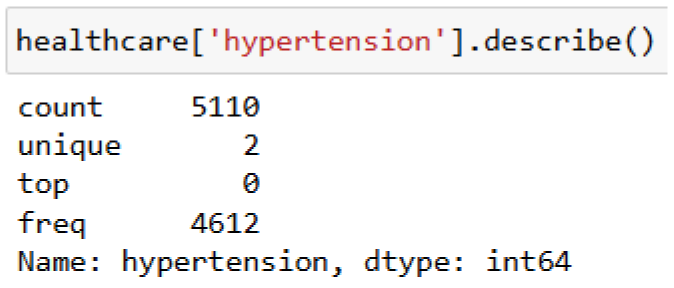
Table 1.1.2: Basic Statistics of ‘age’ Attribute (Refer Fig 1.1.2)

STATISTIC	**VALUE**	**DESCRIPTION**
count	5110	There are 5110 entries in the ‘age’ column
mean	43.226614	The average age is 43.226614 years
std	22.612647	The standard deviation shows that the age varies by around 22.612647 years around the mean which basically means that it varies below or above average (43.226614) by 22.612647.
min	0.08	The minimum age is 0.08 which is approximately 29 days old.

25%	25	It means that 25% of the people’s age is 25 years or less. It refers to the first quartile, which indicates that around 25% of the dataset has values less than or equal to 25 years and 75% of the dataset has values greater than 25 years.
50%	45	It means that 50% of the people’s age is 45 years or less. It refers to the median, which indicates that around 50% of the dataset has values less than or equal to 45 years and 50% of the dataset has values greater than 45 years.
75%	61	It means that 75% of the people’s age is 61 years or less. It refers to the third quartile, which indicates that around 75% of the dataset has values less than or equal to 61 years, and 25% of the dataset has values greater than 61 years.
max	82	The maximum age is 82 years.

1.1.3 Basic Statistics of ‘hypertension’ Attribute

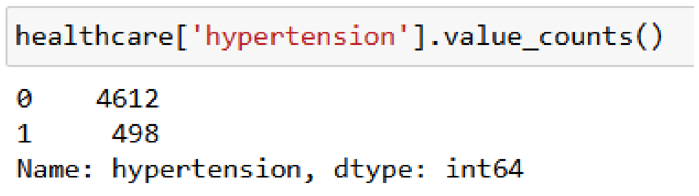
Fig 1.1.3 (A)



```
healthcare['hypertension'].describe()
```

```
count      5110
unique        2
top           0
freq       4612
Name: hypertension, dtype: int64
```

Fig 1.1.3 (B)



```
healthcare['hypertension'].value_counts()
```

```
0    4612
1     498
Name: hypertension, dtype: int64
```

Table 1.1.3: Basic Statistics of 'hypertension' Attribute (Refer Fig 1.1.3(A) and 1.1.3 (B))		
STATISTIC	**VALUE**	**DESCRIPTION**
count	5110	There are 5110 entries in the 'hypertension' column
unique	2	There are 2 unique values in the 'hypertension' column
top	0	'0'(refers to people with no hypertension) is the most frequently occurring value in the 'hypertension' column
frequency	4612	The number of times '0' appears in the 'hypertension' column is 4612. This represents the mode.
1	498	The number of times '1' (refers to people with hypertension) appears in the 'hypertension' column is 498

1.1.4 Basic Statistics of 'heart_disease' Attribute

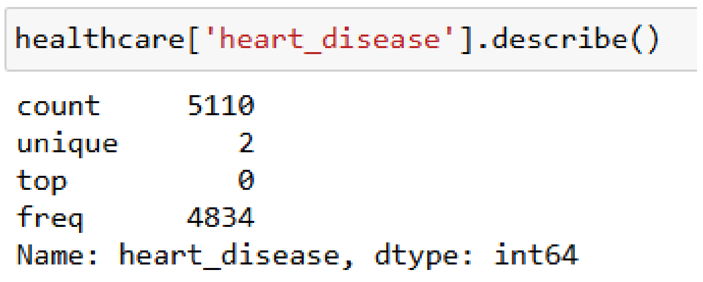
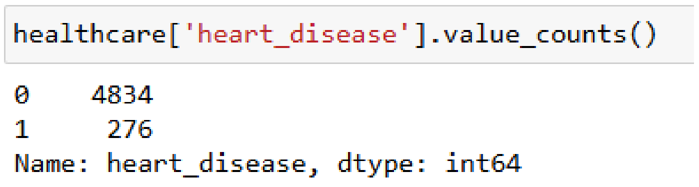
Fig 1.1.4 (A)	*Fig 1.1.4 (B)*
	

Fig 1.1.4 (A):

```
healthcare['heart_disease'].describe()

count      5110
unique        2
top           0
freq       4834
Name: heart_disease, dtype: int64
```

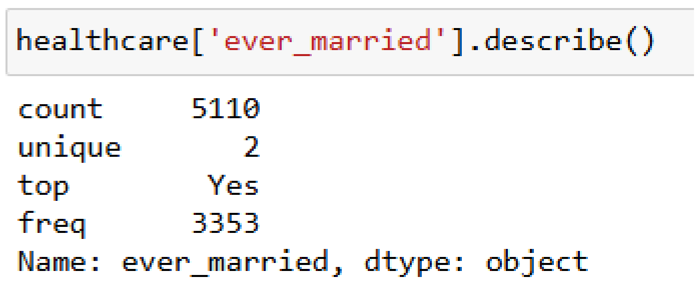
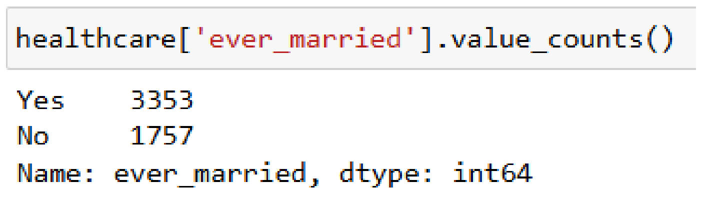
Fig 1.1.4 (B):

```
healthcare['heart_disease'].value_counts()

0    4834
1     276
Name: heart_disease, dtype: int64
```

Table 1.1.4: Basic Statistics of 'heart_disease' Attribute (Refer Fig 1.1.4(A) and 1.1.4 (B))		
STATISTIC	**VALUE**	**DESCRIPTION**
count	5110	There are 5110 entries in the 'heart_disease' column
unique	2	There are 2 unique values in the 'heart_disease' column

top	0	‘0’(refers to people with no heart disease) is the most frequently occurring value in the ‘heart_disease’ column
frequency	4834	The number of times ‘0’ appears in the ‘heart_disease’ column is 4834. This represents the mode.
1	276	The number of times ‘1’(refers to people with heart disease) appears in the ‘heart_disease’ column is 276

1.1.5 Basic Statistics of ‘ever_married’ Attribute

Fig 1.1.5 (A)	***Fig 1.1.5 (B)***
	

```
healthcare['ever_married'].describe()
```

```
count      5110
unique        2
top         Yes
freq       3353
Name: ever_married, dtype: object
```

```
healthcare['ever_married'].value_counts()
```

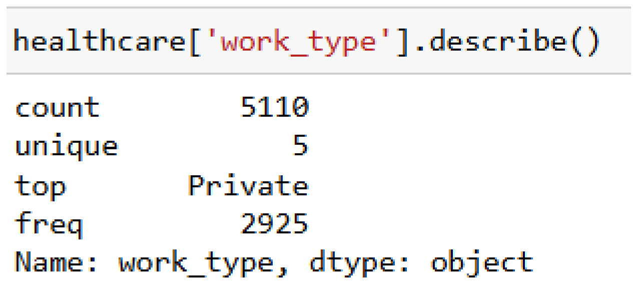
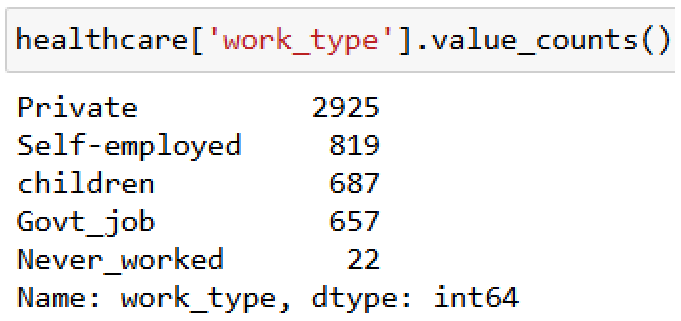
```
Yes     3353
No      1757
Name: ever_married, dtype: int64
```

Table 1.1.5: Basic Statistics of ‘ever_married’ Attribute (Refer Fig 1.1.5(A) and 1.1.5 (B))

STATISTIC	**VALUE**	**DESCRIPTION**
count	5110	There are 5110 entries in the ‘ever_married’ column
unique	2	There are 2 unique values in the ‘ever_married’ column
top	Yes	‘Yes’(refers to people who are married) is the most frequently occurring value in the ‘ever_married’ column
frequency	3353	The number of times ‘Yes’ appears in the ‘ever_married’ column is 3353. This represents the mode.

No	1757	The number of times ‘No’(refers to people who are not married) appears in the ‘ever_married’ column is 1757

1.1.6 Basic Statistics of ‘work_type’ Attribute

Fig 1.1.6 (A)	*Fig 1.1.6 (B)*
	

```
healthcare['work_type'].describe()

count         5110
unique           5
top        Private
freq          2925
Name: work_type, dtype: object
```

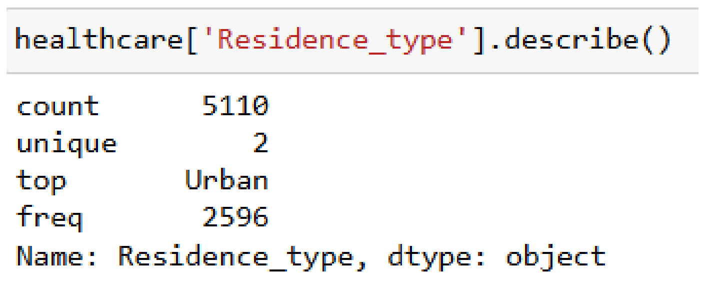
```
healthcare['work_type'].value_counts()

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

Table 1.1.6: Basic Statistics of ‘work_type’ Attribute (Refer Fig 1.1.6(A) and 1.1.6 (B))		
STATISTIC	**VALUE**	**DESCRIPTION**
count	5110	There are 5110 entries in the ‘work_type’ column.
unique	5	There are 5 unique values in the ‘work_type’ column.
top	Private	‘Private’ is the most frequently occurring value in the ‘work_type’ column.
frequency	2925	The number of times ‘Private’ appears in the ‘work_type’ column is 2925. This represents the mode.
Self-employed	819	The number of times ‘Self_employed’ appears in the ‘work_type’ column is 819.
children	687	The number of times ‘children’ appears in the ‘work_type’ column is 687.

Govt_job	657	The number of times ‘Govt_job’ appears in the ‘work_type’ column is 657.
Never_worked	22	The number of times ‘Never_worked’ appears in the ‘work_type’ column is 22.

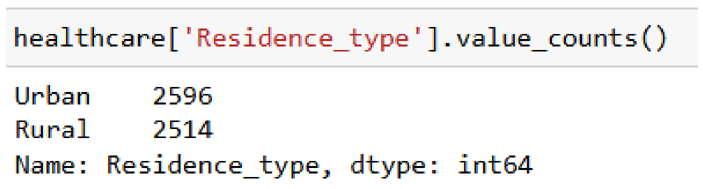
1.1.7 Basic Statistics of ‘Residence_type’ Attribute

Fig 1.1.7 (A)



```
healthcare['Residence_type'].describe()

count       5110
unique         2
top        Urban
freq        2596
Name: Residence_type, dtype: object
```

Fig 1.1.7 (B)



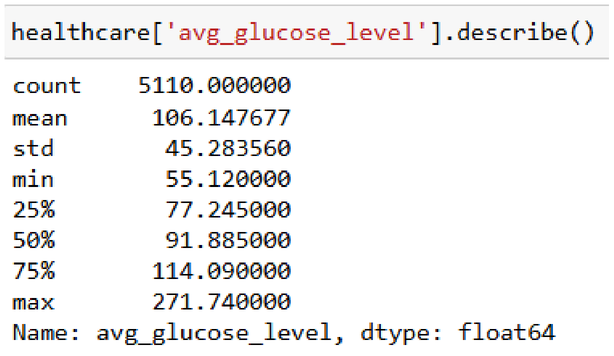
```
healthcare['Residence_type'].value_counts()

Urban    2596
Rural    2514
Name: Residence_type, dtype: int64
```

Table 1.1.7: Basic Statistics of ‘Residence_type’ Attribute (Refer Fig 1.1.7(A) and 1.1.7 (B))

STATISTIC	VALUE	DESCRIPTION
count	5110	There are 5110 entries in the ‘Residence_type’ column
unique	2	There are 2 unique values in the ‘Residence_type’ column
top	Urban	‘Urban’ is the most frequently occurring value in the ‘Residence_type’ column
frequency	2596	The number of times ‘Urban’ appears in the ‘Residence_type’ column is 2596. This represents the mode.
Rural	2514	The number of times ‘Rural’ appears in the ‘Residence_type’ column is 2514

1.1.8 Basic Statistics of ‘avg_glucose_level’ Attribute

Fig 1.1.8



```
healthcare['avg_glucose_level'].describe()
```

```
count    5110.000000
mean      106.147677
std        45.283560
min        55.120000
25%        77.245000
50%        91.885000
75%       114.090000
max       271.740000
Name: avg_glucose_level, dtype: float64
```

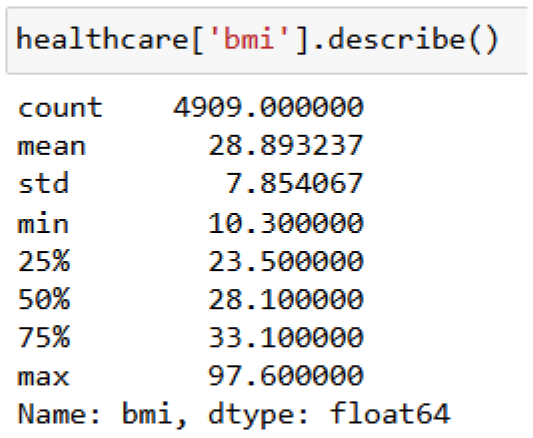
Table 1.1.8: Basic Statistics of ‘avg_glucose_level’ Attribute (Refer Fig 1.1.8)

STATISTIC	VALUE	DESCRIPTION
count	5110	There are 5110 entries in the ‘avg_glucose_level’ column
mean	106.147677	The average of ‘avg_glucose_level’ is 106.147677 units.
std	45.283560	The standard deviation shows that the ‘avg_glucose_level’ varies by around 45.283560 units around the mean which basically means that it varies below or above average (106.147677) by 45.283560.
min	55.12	The minimum ‘avg_glucose_level’ is 55.12 units.
25%	77.245	It means that 25% of the people have an ‘avg_glucose_level’ of 77.245 or less. It refers to the first quartile, which indicates that around 25% of the dataset has values less than or equal to 77.245 units and 75% of the

		dataset has values greater than 77.245 units.
50%	91.885	It means that 50% of the people have an ‘avg_glucose_level’ of 91.885 units. It refers to the median, which indicates that around 50% of the dataset has values less than or equal to 91.885 units and 50% of the dataset has values greater than 91.885 units.
75%	114.09	It means that 75% of the people have an ‘avg_glucose_level’ of 114.09 units. It refers to the third quartile, which indicates that around 75% of the dataset has values less than or equal to 114.09 units, and 25% of the dataset has values greater than 114.09 units.
max	271.74	The maximum ‘avg_glucose_level’ is 271.74 units.

1.1.9 Basic Statistics of ‘bmi’ Attribute

Fig 1.1.9



```
healthcare['bmi'].describe()
```

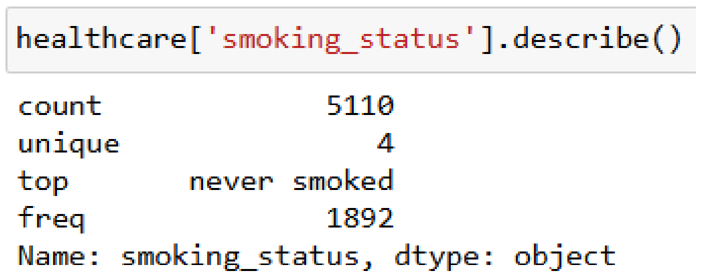
```
count    4909.000000
mean       28.893237
std         7.854067
min        10.300000
25%        23.500000
50%        28.100000
75%        33.100000
max        97.600000
Name: bmi, dtype: float64
```

Table 1.1.9: Basic Statistics of ‘bmi’ Attribute (Refer Fig 1.1.9)		
STATISTIC	**VALUE**	**DESCRIPTION**
count	4909	There are 4909 entries in the ‘bmi’ column.
mean	28.893237	The average bmi is 28.893237 units.
std	7.854067	The standard deviation shows that the ‘bmi’ varies by around 7.854067 units around the mean which basically means that it varies below or above average (28.893237) by 7.854067.
min	10.3	The minimum ‘bmi’ is 10.3 units.
25%	23.5	It means that 25% of the people have a ‘bmi’ of 23.5 or less. It refers to the first quartile, which indicates that around 25% of the dataset has values less than or equal to 23.5 units and 75% of the dataset has values greater than 23.5 units.
50%	28.1	It means that 50% of the people have a ‘bmi’ of 28.1 units. It refers to the median, which indicates that around 50% of the dataset has values less than or equal to 28.1 units and 50% of the dataset has values greater than 28.1 units.
75%	33.1	It means that 75% of the people have a ‘bmi’ of 33.1 units. It refers to the third quartile, which indicates that around 75% of the dataset has values less than or equal to 33.1 units, and 25% of the dataset has values greater than 33.1 units.
max	97.6	The maximum ‘bmi’ is 97.6

		units.

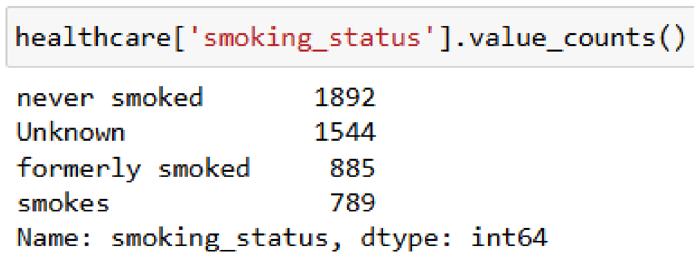
1.1.10 Basic Statistics of ‘smoking_status’ Attribute

Fig 1.1.10 (A)



```
healthcare['smoking_status'].describe()

count              5110
unique                4
top        never smoked
freq               1892
Name: smoking_status, dtype: object
```

Fig 1.1.10 (B)



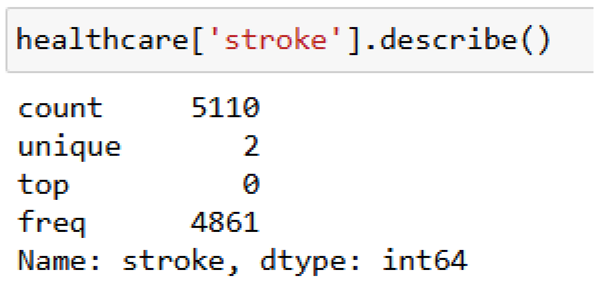
```
healthcare['smoking_status'].value_counts()

never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
Name: smoking_status, dtype: int64
```

Table 1.1.10: Basic Statistics of ‘smoking_status’ Attribute (Refer Fig 1.1.10(A) and 1.1.10 (B))

STATISTIC	VALUE	DESCRIPTION
count	5110	There are 5110 entries in the ‘smoking_status’ column
unique	4	There are 4 unique values in the ‘smoking_status’ column
top	never smoked	‘never smoked’ is the most frequently occurring value in the ‘smoking_status’ column
frequency	1892	The number of times ‘never smoked’ appears in the ‘smoking_status’ column is 1892. This represents the mode.
Unknown	1544	The number of times ‘Unknown’ appears in the ‘smoking_status’ column is 1544.
formerly smoked	885	The number of times ‘formerly smoked’ appears in the ‘smoking_status’ column is 885.
smokes	789	The number of times ‘smokes’ appears in the ‘smoking_status’ column is 789.

1.1.11 Basic Statistics of ‘stroke’ Attribute

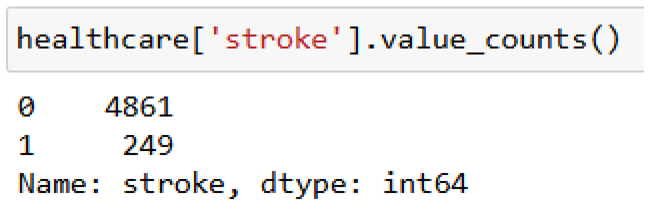
Fig 1.1.11 (A)



```
healthcare['stroke'].describe()
```

```
count      5110
unique        2
top           0
freq       4861
Name: stroke, dtype: int64
```

Fig 1.1.11 (B)



```
healthcare['stroke'].value_counts()
```

```
0    4861
1     249
Name: stroke, dtype: int64
```

Table 1.11: Basic Statistics of ‘stroke’ Attribute (Refer Fig 1.11(A) and 1.11 (B))

STATISTIC	**VALUE**	**DESCRIPTION**
count	5110	There are 5110 entries in the ‘stroke’ column
unique	2	There are 2 unique values in the ‘stroke’ column
top	0	‘0’ (refers to people who didn’t have stroke) is the most frequently occurring value in the ‘stroke’ column
frequency	4861	The number of times ‘0’ appears in the ‘stroke’ column is 4861. This represents the mode.
1	249	The number of times ‘1’(refers to people who had stroke) appears in the ‘stroke’ column is 249.

1.2. Visualisations

1.2.1 Boxplot to Analyse the Relationship between ‘Stroke’ and ‘Age’

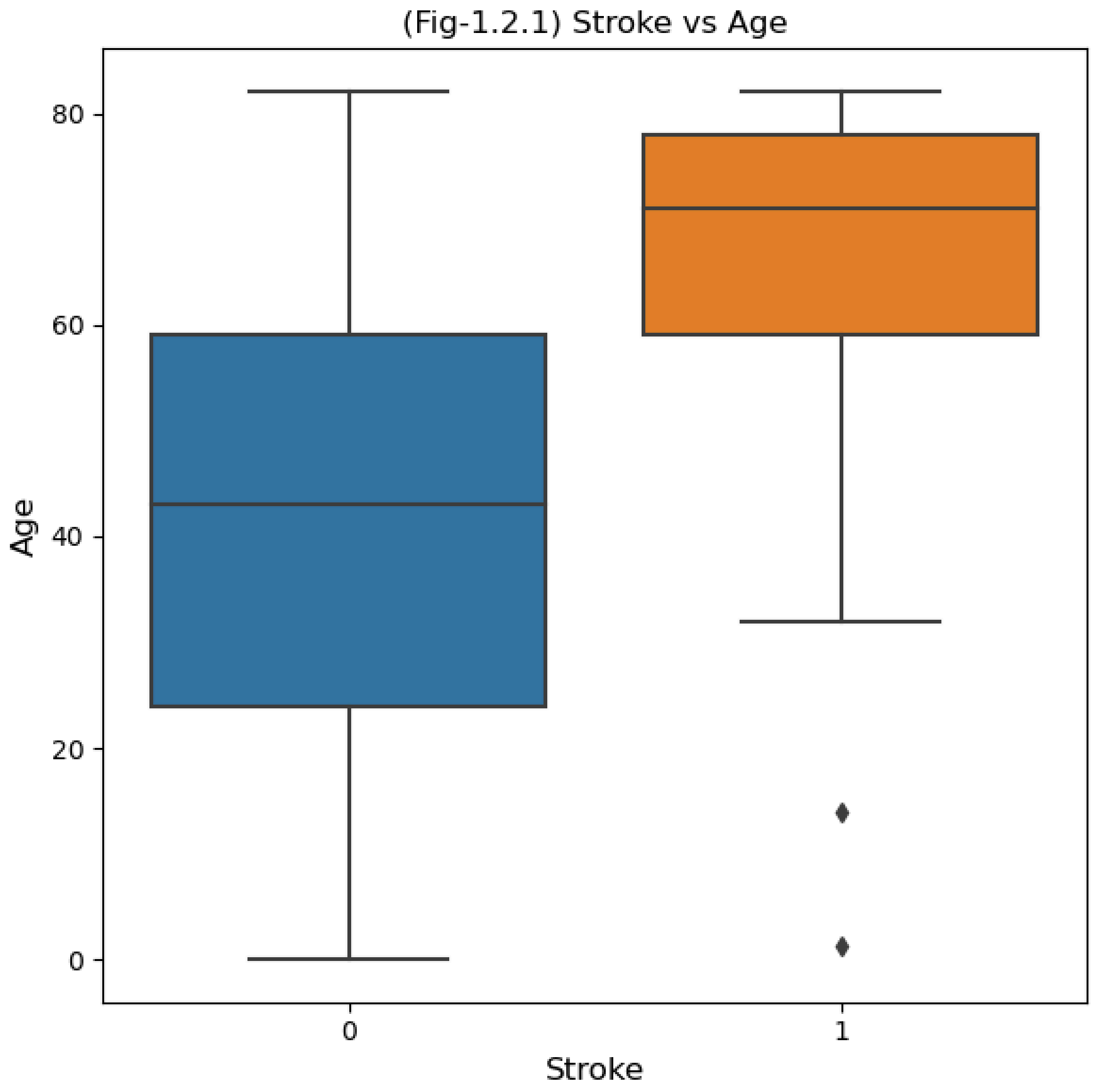


Table 1.2.1: Key Observations and Interpretations from the Graph (Ref Fig 1.2.1)	
Stroke	**Age**
0 (individuals who had no stroke)	● The median age of individuals who had no stroke appears to be around 40 years. ● The minimum age of individuals who had no stroke is 0 years (less than 1 year old)

	• The maximum age of individuals who had no stroke is 80 years. • The majority of non-stroke cases occur in the age range of approximately 20 to slightly less than 60 years (Inter Quartile Range)
1 (individuals who had a stroke)	• The median age of individuals who had a stroke appears to be around 65 years. • The minimum age of individuals who had a stroke is approximately 30 years (without considering outliers), but stroke cases can also be observed in individuals younger than 30 years which suggests potential outliers. • The maximum age of individuals who had a stroke is approximately 80 years. • It can be inferred from the graph that stroke cases are more common in older individuals as the majority of stroke cases occur in the age range of approximately 60 to slightly less than 80 years (Inter Quartile Range)

1.2.2 Violinplot to Analyse the Relationship between ‘Stroke’ and ‘BMI’ (EXTRA)

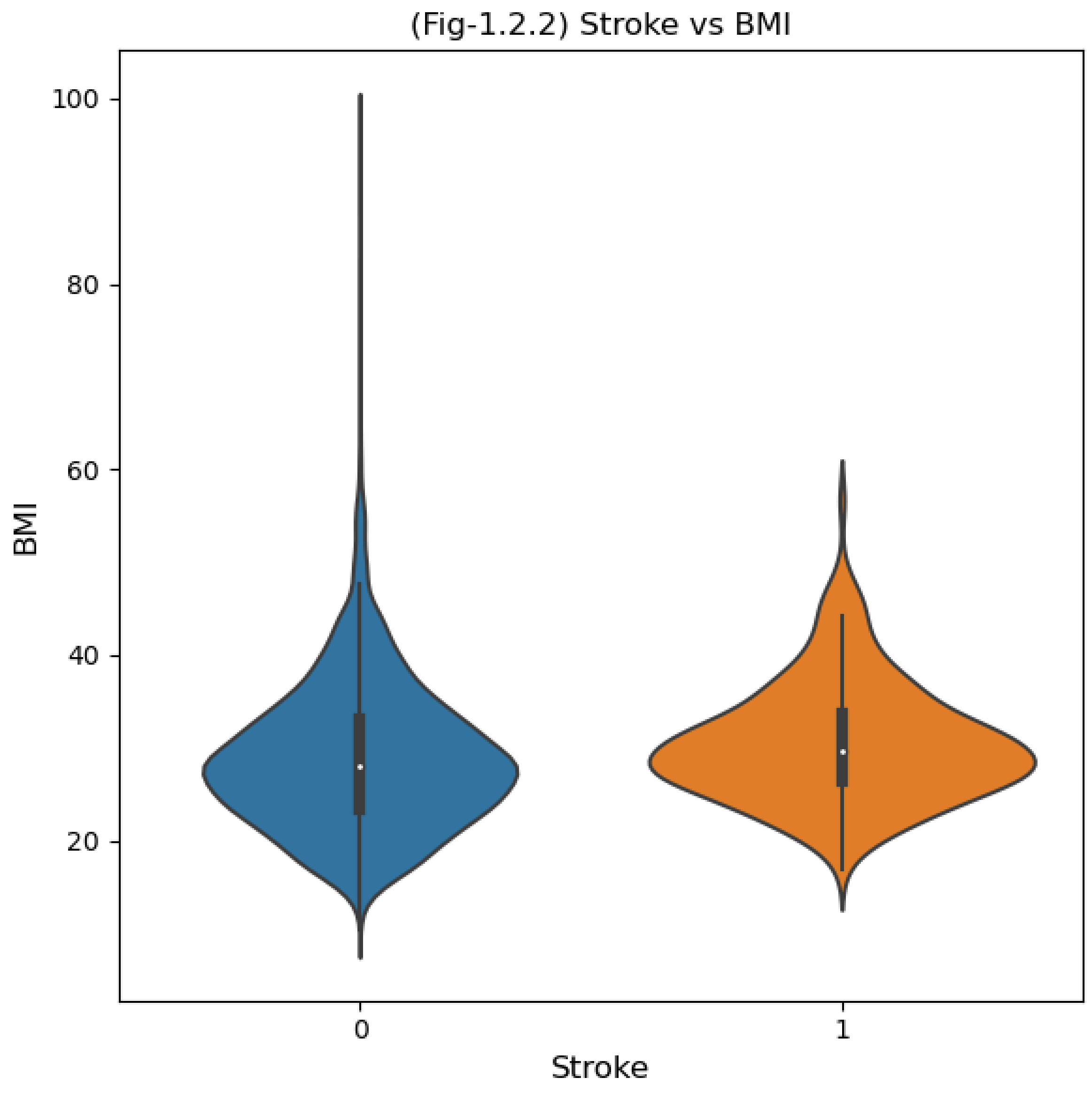


Table 1.2.2: Key Observations and Interpretations from the Graph (Ref Fig 1.2.2)	
Stroke	**BMI**
0 (individuals who had no stroke)	• The median BMI of individuals who had no stroke is slightly less than 30. • The minimum BMI of individuals who had no stroke is around 10. • The maximum BMI of individuals who had no stroke is slightly less than 50 (excluding outliers), but individuals with a BMI beyond 50 suggest serious outliers.

	• The majority of non-stroke cases occur in the BMI range of around 20 to less than 40 as the "violin" plot is widest in this range.
1 (individuals who had a stroke)	• The median BMI of individuals who had a stroke appears to be around 30. • The minimum BMI of individuals who had a stroke is approximately 15. • The maximum BMI of individuals who had a stroke is slightly less than 50, individuals with BMI beyond this value suggest alarming outliers. • The majority of stroke cases occur in the BMI range of around 25 to less than 40 as the "violin" plot is widest in this range.

1.2.3 Scatterplot to Analyse the Relationship between 'bmi' and 'avg_glucose_level'

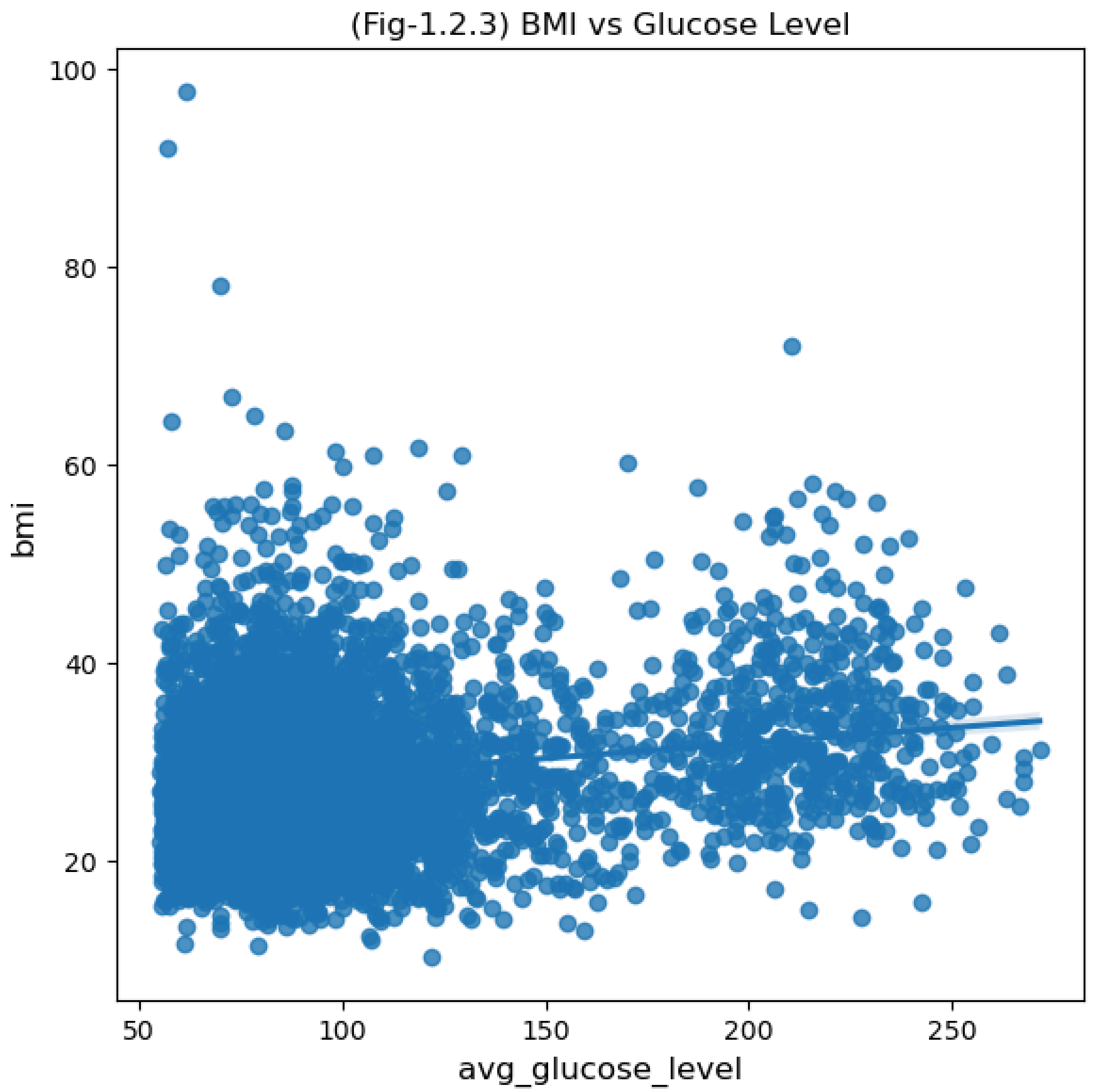


Table 1.2.3: Key Observations and Interpretations from the Graph (Ref Fig 1.2.3)
• There is a very weak positive correlation between 'avg_glucose_level' and 'bmi' as indicated by the regression line. • As glucose level increases, BMI tends to slightly increase. • It can be observed from the graph that most data points are concentrated in the range of 20-40 BMI and 50-150 glucose levels, this indicates that most individuals fall in this range in our dataset. • There are serious outliers with high BMI values beyond 60 with lower glucose levels. • Some outliers can also be observed for glucose levels beyond 250. • It can be observed from the graph that for a particular glucose level, BMI shows high variability, this indicates that there might be some other factors that influence BMI more strongly than glucose levels.

1.2.4 Boxplot to Analyse the Relationship between 'Hypertension' and 'Age'

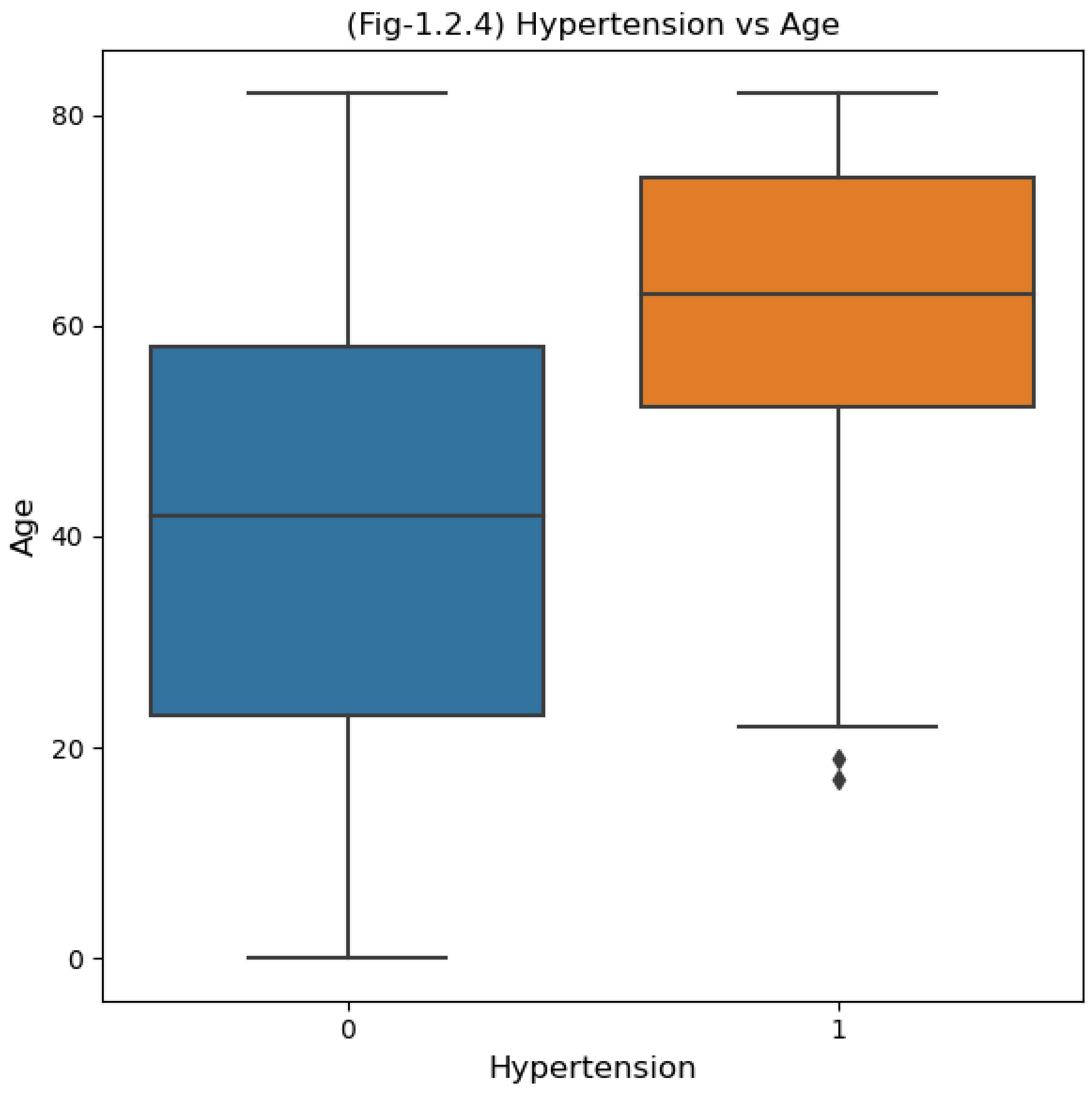


Table 1.2.4: Key Observations and Interpretations from the Graph (Ref Fig 1.2.4)	
Hypertension	**Age**
0 (individuals who have no hypertension)	● The median age of individuals who have no hypertension appears to be around 40 years. ● The minimum age of individuals who have no hypertension is 0 years (less than 1 year old) ● The maximum age of individuals who have no hypertension is 80 years.

	● The majority of non-hypertension cases occur in the age range of approximately 20 to slightly less than 60 years (Inter Quartile Range)
1 (individuals who have hypertension)	● The median age of individuals who have hypertension appears to be around 60 years. ● The minimum age of individuals who have hypertension is approximately 20 years (without considering outliers), but hypertension cases can also be observed in individuals younger than 20 years which suggests potential outliers. ● The maximum age of individuals who have hypertension is approximately 80 years. ● It can be inferred from the graph that hypertension cases are more common in older individuals as the majority of hypertension cases occur in the age range of approximately 50 to slightly more than 70 years (Inter Quartile Range)

1.2.5 Grouped Bar Chart to Analyse the Relationship between 'Heart Disease' and 'Stroke'

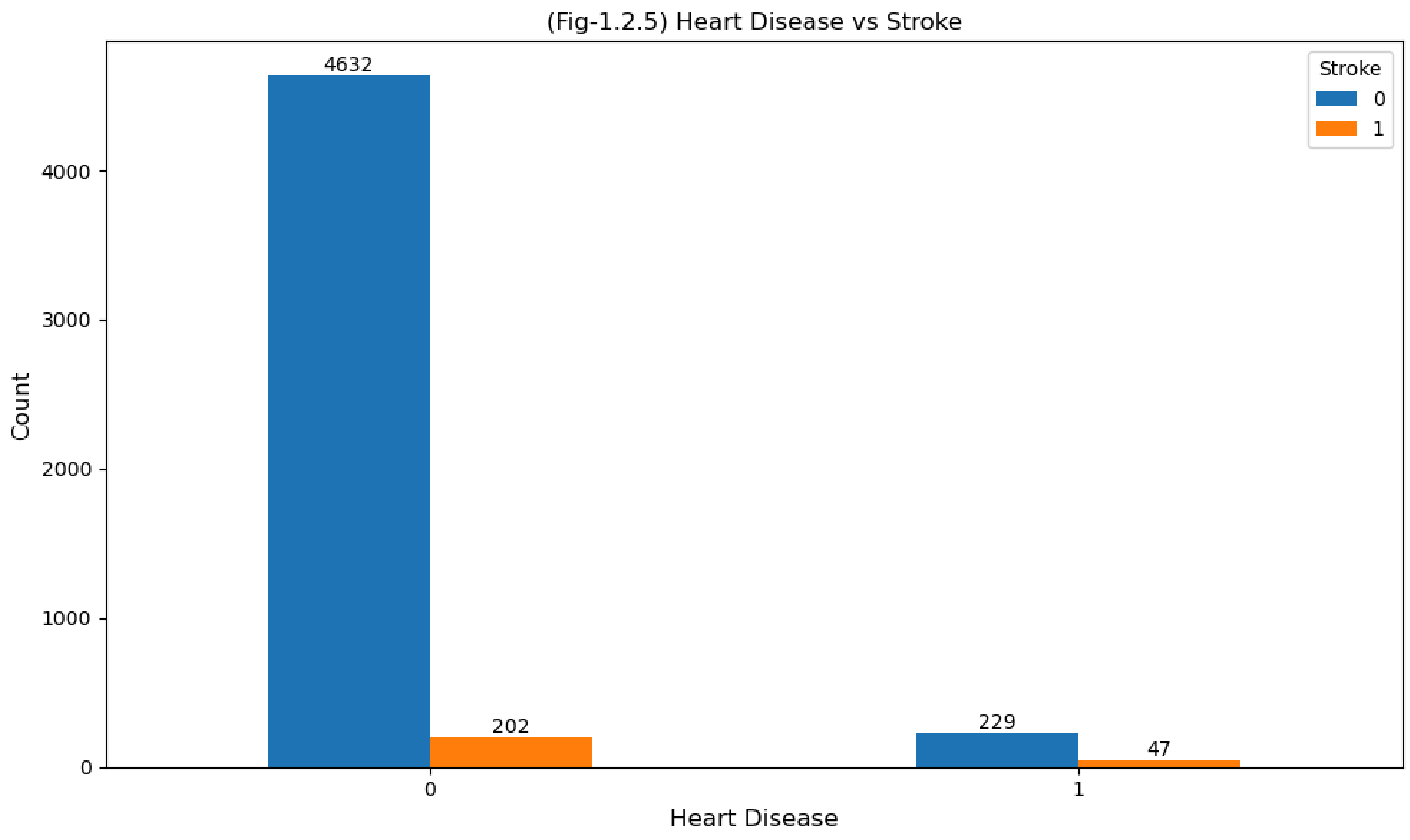


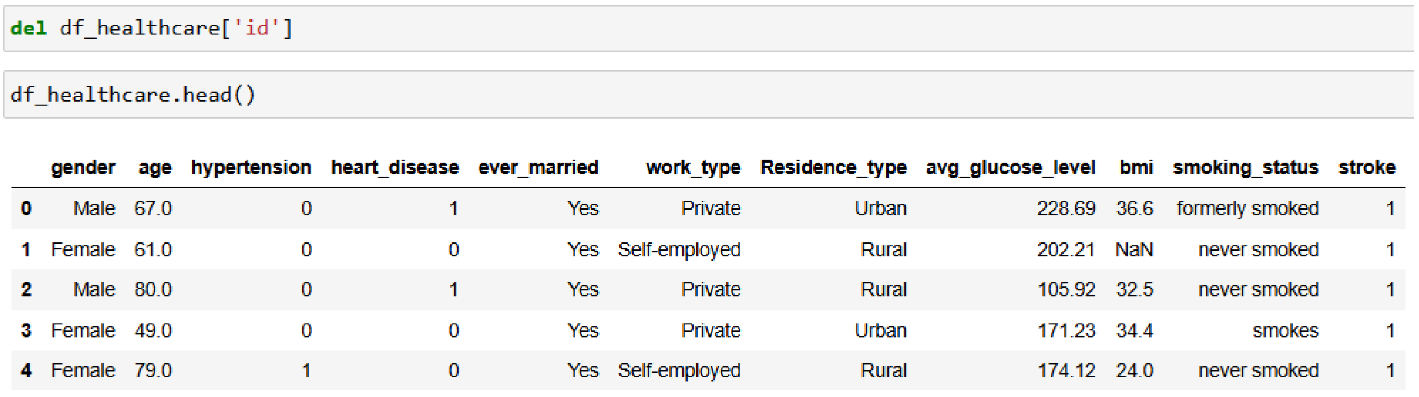
Table 1.2.5: Key Observations and Interpretations from the Graph (Ref Fig 1.2.5)	
Heart Disease	**Stroke**
0 (individuals who have no heart disease)	Out of 4834(4632+202) individuals, 202 had reported a stroke, which means the percentage of individuals reporting a stroke without heart disease is 4.17%
1 (individuals who have heart disease)	Out of 276 (229+47) individuals, 47 had reported a stroke, which means the percentage of individuals reporting a stroke with heart disease is 17.02%. This tells us that stroke cases are more common in individuals with heart disease.

Chapter 2. Data Preparation

2.1 Deleting the 'id' Column

The 'id' column is deleted because it is simply a unique identification of each record in the dataset and has no statistical significance and meaning on its own. The below figure (Fig-2.1) shows the successful deletion of the 'id' column.

Fig-2.1



```
del df_healthcare['id']
```

```
df_healthcare.head()
```

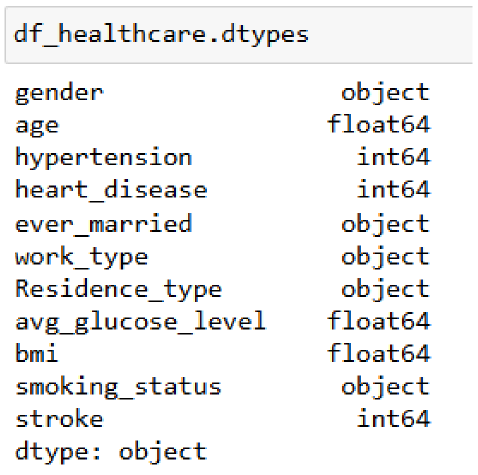
```
   gender   age  hypertension  heart_disease  ever_married      work_type  Residence_type  avg_glucose_level   bmi   smoking_status  stroke
0    Male  67.0             0              1           Yes        Private           Urban             228.69  36.6  formerly smoked       1
1  Female  61.0             0              0           Yes  Self-employed           Rural             202.21   NaN     never smoked       1
2    Male  80.0             0              1           Yes        Private           Rural             105.92  32.5     never smoked       1
3  Female  49.0             0              0           Yes        Private           Urban             171.23  34.4           smokes       1
4  Female  79.0             1              0           Yes  Self-employed           Rural             174.12  24.0     never smoked       1
```

2.2 Conversion of Necessary Columns to Numerical

The default data types of **'gender'**, **'ever_married'**, **'work_type'**, **'Residence_type'**, and **'smoking_status'** is object, these columns are converted to numerical data type using fit_transform method of the LabelEncoder class from the preprocessing module in the sklearn library. The conversion is done because machine learning algorithms require numerical inputs. One-hot encoding is not used to convert categorical to numerical data type because it creates multiple columns for each category and increases the dimensionality and complexity of the dataset. The below figure (Fig-2.2 A) shows the successful conversion of the above-mentioned columns.

Fig-2.2 A

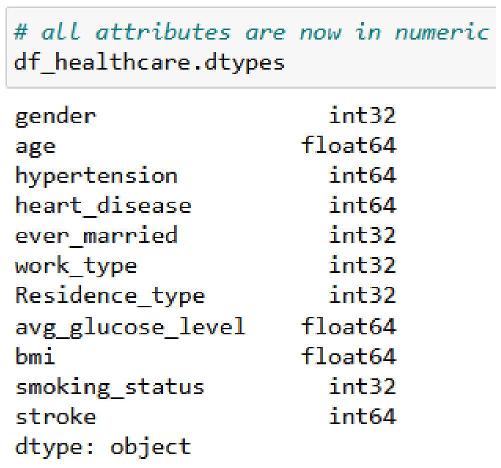
Before Conversion



```
df_healthcare.dtypes
```

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

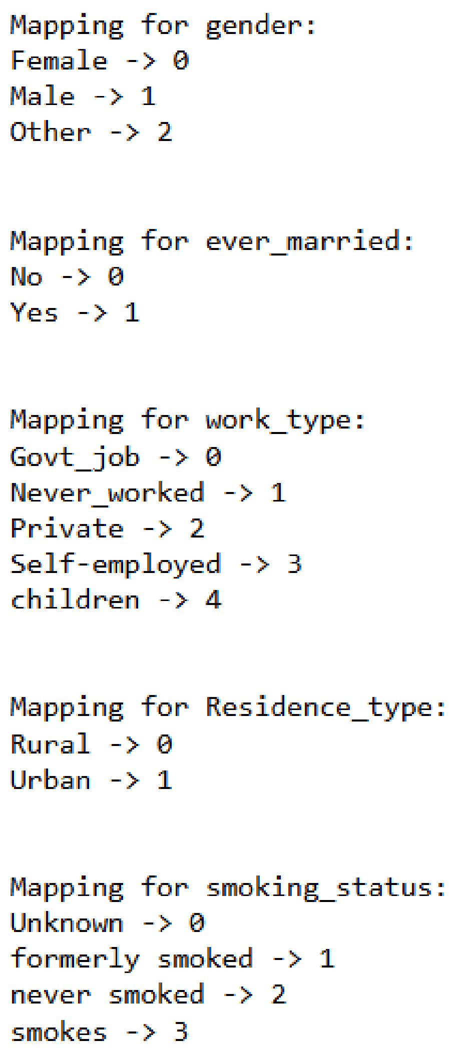
After Conversion to Numerical Data Type



```
# all attributes are now in numeric
df_healthcare.dtypes
```

```
gender                 int32
age                  float64
hypertension           int64
heart_disease          int64
ever_married           int32
work_type              int32
Residence_type         int32
avg_glucose_level    float64
bmi                  float64
smoking_status         int32
stroke                 int64
dtype: object
```

Fig- 2.2 B-**Categorical Names associated with each LabelEncoded Categories**



```
Mapping for gender:
Female -> 0
Male -> 1
Other -> 2


Mapping for ever_married:
No -> 0
Yes -> 1


Mapping for work_type:
Govt_job -> 0
Never_worked -> 1
Private -> 2
Self-employed -> 3
children -> 4


Mapping for Residence_type:
Rural -> 0
Urban -> 1


Mapping for smoking_status:
Unknown -> 0
formerly smoked -> 1
never smoked -> 2
smokes -> 3
```

2.3 Treating the Outliers

The IQR (Inter Quartile Range) method is used to treat outliers in the 'bmi' and 'avg_glucose_level' attributes. All values that are greater than Q3 + 1.5 * (Q3 - Q1) are clipped to this upper limit, and values below Q1 - 1.5 * (Q3 - Q1) are clipped to this lower limit for both the above-mentioned attributes, where Q1 is the 25th percentile and Q3 is the 75th percentile.

2.3.1 Treating the outliers for the 'bmi' Attribute

The 25th percentile (Q1) for 'bmi' attribute is 23.5 and the 75th percentile (Q3) is 33.1. All values that are greater than 47.5 (Q3 + 1.5 * (Q3 - Q1)) are clipped to this upper limit, and values below 9.1 (Q1 - 1.5 * (Q3 - Q1)) are clipped to this lower limit. The figure (Fig-2.3 (A)) shows the successful treatment of the outliers in the 'bmi' attribute.

2.3.2 Treating the outliers for the 'avg_glucose_level' Attribute

The 25th percentile (Q1) for 'avg_glucose_level' attribute is 77.245 and the 75th percentile (Q3) is 114.09. All values that are greater than 169.357 (Q3 + 1.5 * (Q3 - Q1)) are clipped to this upper limit, and values below 21.977 (Q1 - 1.5 * (Q3 - Q1)) are clipped to this lower limit. The figure (Fig-2.3(B)) shows the successful treatment of the outliers in the 'avg_glucose_level' attribute.

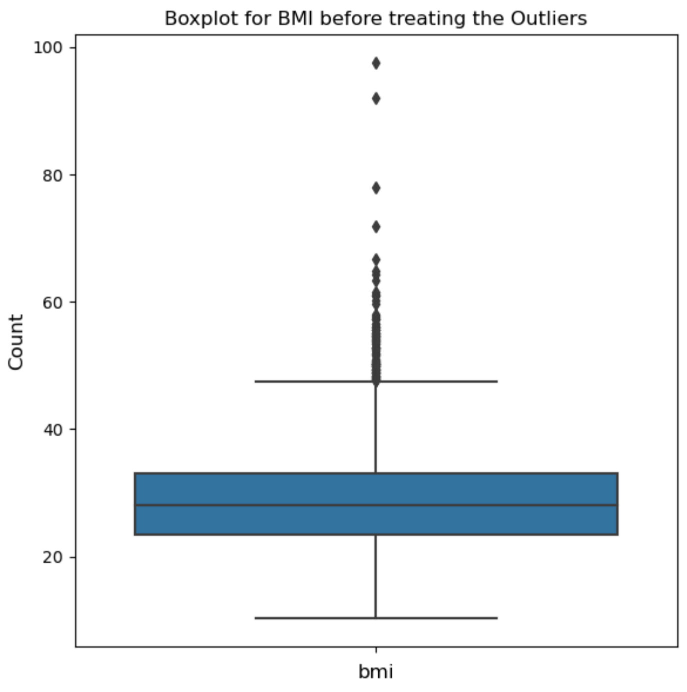
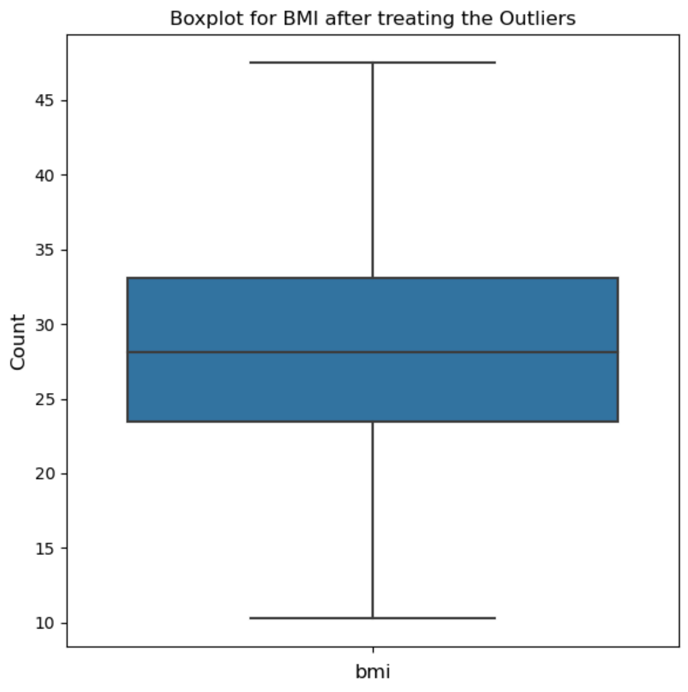
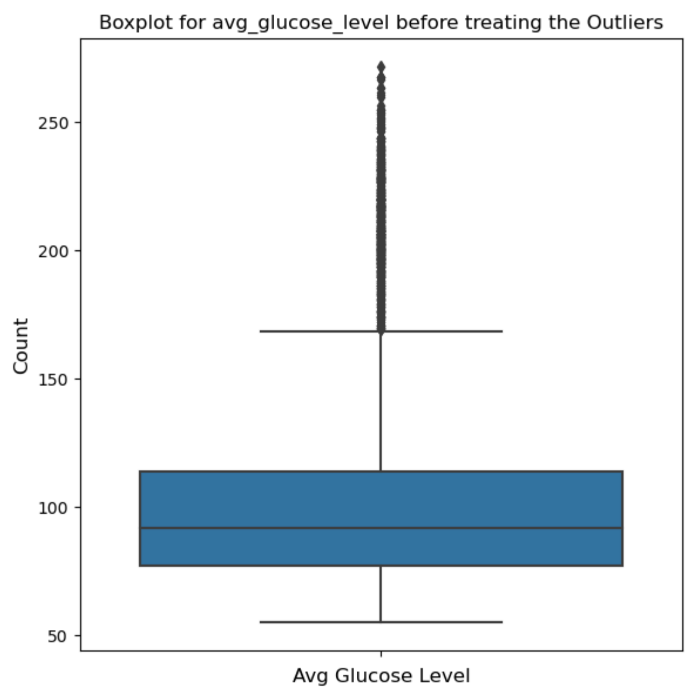
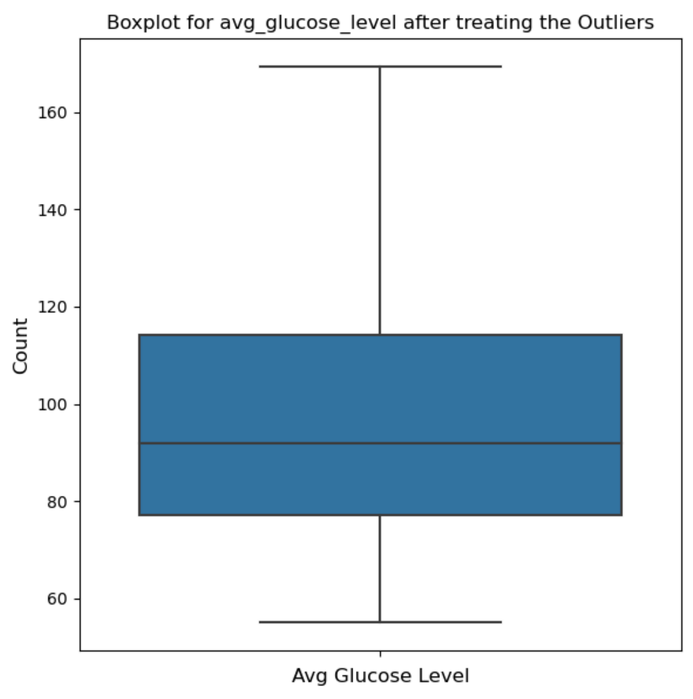
Fig-2.3 (A)	
Before Treatment	**After Treatment**
	

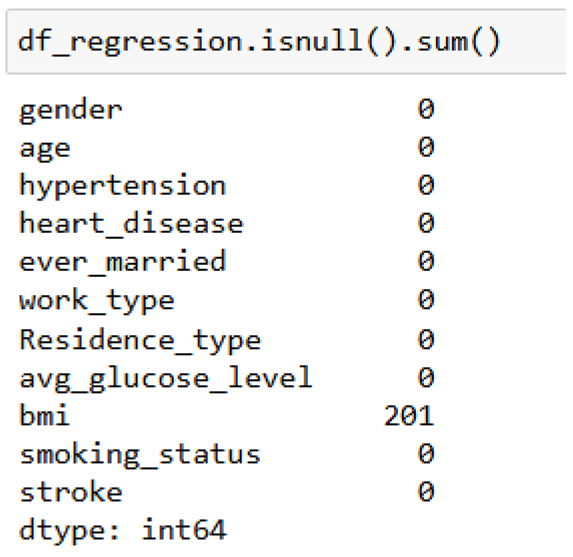
Fig-2.3 (B)	
Before Treatment	**After Treatment**
	

2.4 Imputing missing values for Regression

A copy of the ‘df_healthcare’ dataframe is created as ‘df_regression’ to specifically use for regression tasks. There are 201 missing values in the ‘bmi’ attribute. These missing values are dropped using the dropna() function from the ‘df_regression’ dataframe. Refer Fig-2.4.

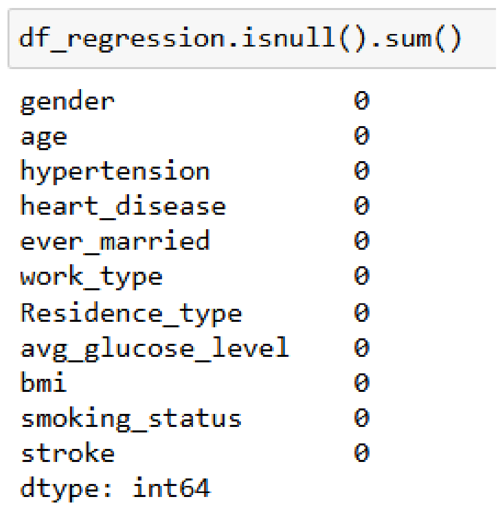
Fig-2.4

Before Treatment



```
df_regression.isnull().sum()

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
```

After Treatment

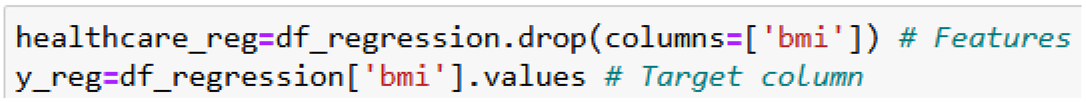


```
df_regression.isnull().sum()

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

2.5 Segregating ‘Features’ and ‘Target’ column for Regression

The ‘bmi’ attribute is the target column for regression. The features include all the columns of the dataset except the ‘bmi’ column. In the below figure (Fig-2.5) **healthcare_reg** represents all the features and **y_reg** represents the target column.

Fig-2.5

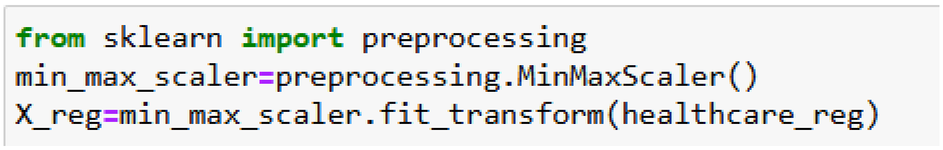


```
healthcare_reg=df_regression.drop(columns=['bmi']) # Features
y_reg=df_regression['bmi'].values # Target column
```

2.6 Normalisation of the Features for Regression

The **healthcare_reg** variable which contains all the features are normalised using MinMaxScaler and the normalised features are stored in **X_reg** variable. See Fig-2.6 for the implemented code.

Fig-2.6

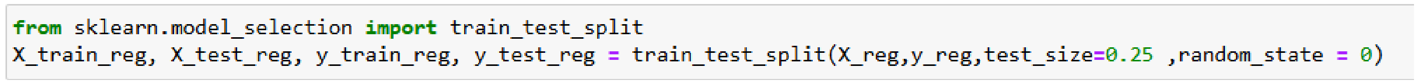


```
from sklearn import preprocessing
min_max_scaler=preprocessing.MinMaxScaler()
X_reg=min_max_scaler.fit_transform(healthcare_reg)
```

2.7 Splitting the Data into Train and Test for Regression

The normalised features **(X_reg)** and target column **(y_reg)** are split into **75%** for training and **25%** for testing where **X_train_reg** contains **75%** of the features for training, **X_test_reg** contains **25%** of the features for testing, **y_train_reg** contains **75%** of the target data for training and **y_test_reg** contains **25%** of the target data for testing. See Fig-2.7 for the implemented code.

Fig-2.7



```
from sklearn.model_selection import train_test_split
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg,y_reg,test_size=0.25 ,random_state = 0)
```

2.8 Imputing missing values for Classification

A copy of the ‘df_healthcare’ dataframe is created as ‘df_classification’ to specifically use for classification tasks. There are 201 missing values in the ‘bmi’ attribute. These missing values are treated based on the mean ‘bmi’ grouped by the ‘age’ attribute. The ‘age’ attribute is chosen for grouping because ‘bmi’ has high positive correlation of 0.36 (refer Fig-2.3(A)) with the ‘age’ attribute as compared to other attributes. Although, the ‘ever_married’ attribute also has the same correlation of 0.36 (Refer Fig-2.8 (A)) but it is not chosen because it doesn't seem directly related to ‘bmi’ based on domain knowledge. The figure (Fig-2.8 (B)) shows the successful treatment of the missing values in the ‘bmi’ attribute.

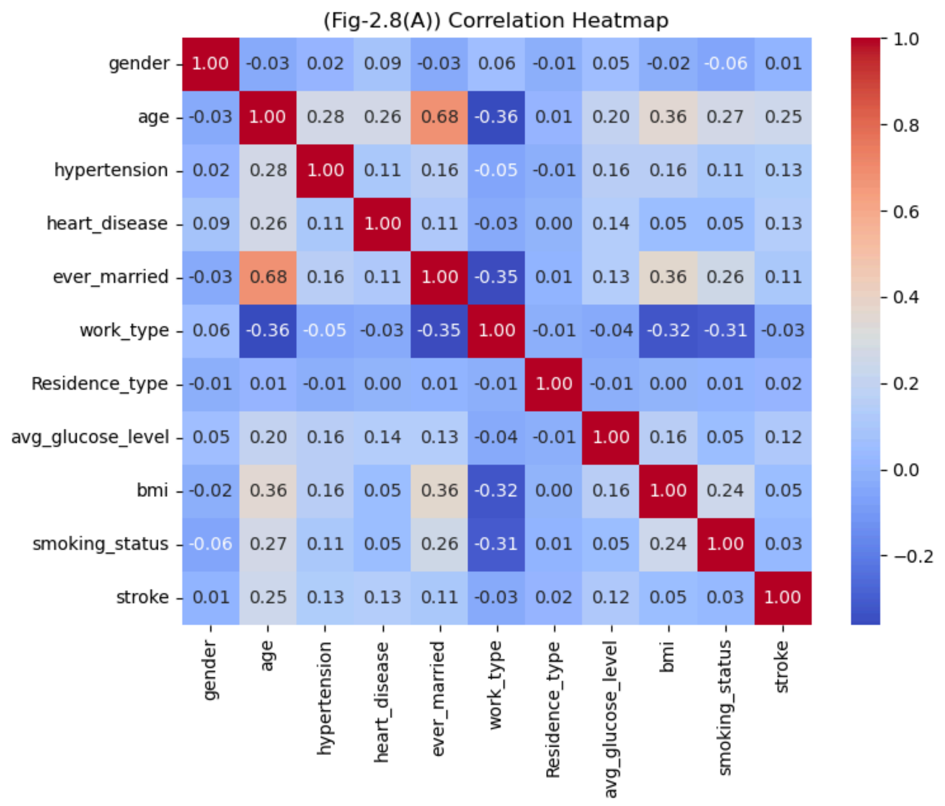


Fig-2.8 (B)

Before Treatment	**After Treatment**
`df_classification.isnull().sum()`	`df_classification.isnull().sum()`

Before Treatment

```
df_classification.isnull().sum()

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
```

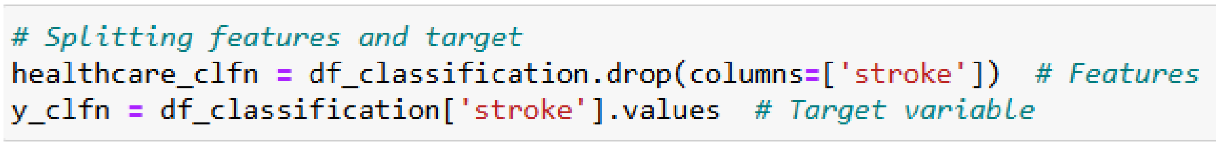
After Treatment

```
df_classification.isnull().sum()

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

2.9 Segregating ‘Features’ and ‘Target’ for Classification

The ‘stroke’ attribute is the target column for classification. The features include all the columns of the dataset except the ‘stroke’ column. In the below figure (Fig-2.9) **healthcare_clfn** represnts all the features and **y_clfn** represents the target column.

Fig-2.9

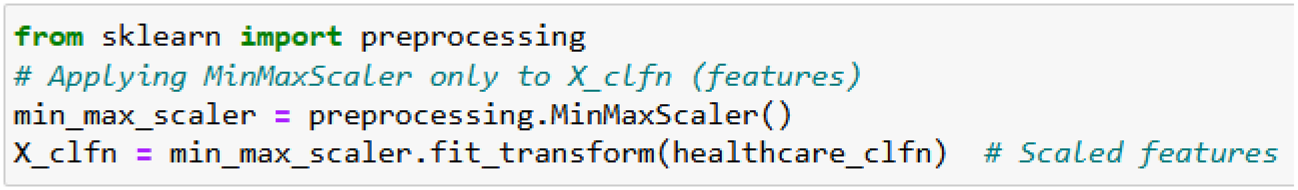


```
# Splitting features and target
healthcare_clfn = df_classification.drop(columns=['stroke'])  # Features
y_clfn = df_classification['stroke'].values  # Target variable
```

2.10 Normalisation of the Features for Classification

The **healthcare_clfn** variable which contains all the features are normalised using MinMaxScaler and the normalised features are stored in **X_clfn** variable. See Fig-2.10 for the implemented code.

Fig-2.10

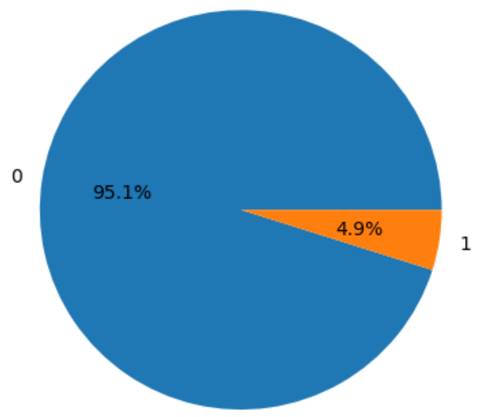


```
from sklearn import preprocessing
# Applying MinMaxScaler only to X_clfn (features)
min_max_scaler = preprocessing.MinMaxScaler()
X_clfn = min_max_scaler.fit_transform(healthcare_clfn)  # Scaled features
```

2.11 Tackling Class Imbalance of the target column ('stroke') using SMOTE (Synthetic Minority Oversampling Technique) (EXTRA)

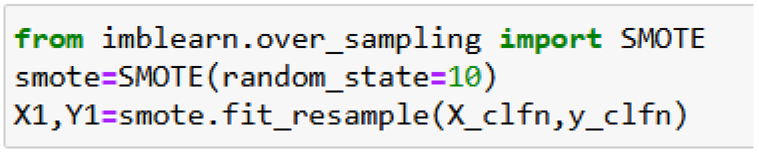
It can be observed from the below pie chart (Fig-2.11 A) that the classes in the ‘stroke’ column is highly imbalanced. The class ‘0’ is the majority class which has 95.1% values in the dataset and class ‘1’ is the minority class which has only 4.9% values in the dataset.

(Fig-2.11 A) Stroke Column Class Distribution



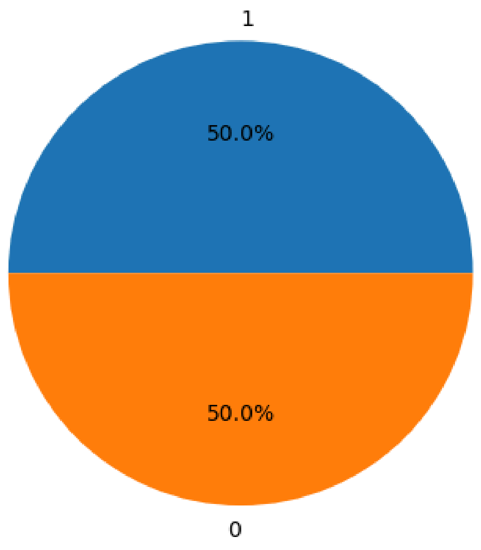
The imbalance nature of the target column ('stroke') is tackled by oversampling the minority class (class '1') to the number of majority class (class '0') using SMOTE (Synthetic Minority Oversampling Technique) of the imblearn library. X1 and Y1 contains the balanced features and the target column respectively. See Fig-2.11 B for the implemented code. The Fig-2.11 C shows the balance between both the classes (class '0' and '1') after applying SMOTE.

Fig-2.11 B



```
from imblearn.over_sampling import SMOTE
smote=SMOTE(random_state=10)
X1,Y1=smote.fit_resample(X_clfn,y_clfn)
```

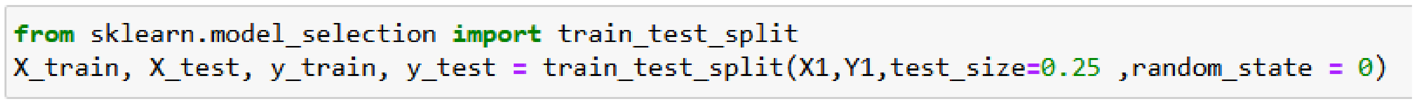
(Fig-2.11 C) Stroke Column Class Distribution after oversampling using SMOTE



2.12 Splitting the Balanced Dataset into Train and Test set for Classification

The normalised and balanced features **(X1)** and target column **(Y1)** are split into **75%** for training and **25%** for testing where **X_train** contains **75%** of the features for training, **X_test** contains **25%** of the features for testing, **y_train** contains **75%** of the target data for training and **y_test** contains **25%** of the target data for testing. See Fig-2.12 for the implemented code.

Fig-2.12

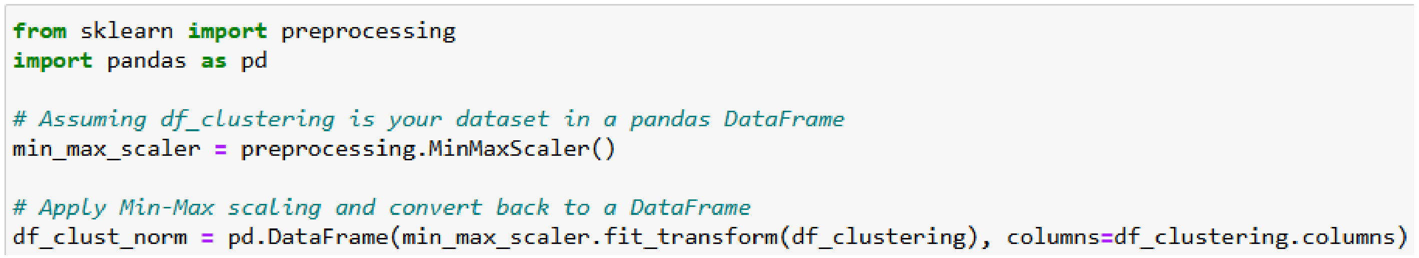


```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X1,Y1,test_size=0.25 ,random_state = 0)
```

2.13 Normalisation of the Features for Clustering

A copy of the 'df_classification' dataframe is created as 'df_clustering'. This dataframe is normalised using MinMaxScaler and is stored as a new dataframe 'df_clust_norm' which will extensively be used in the clustering tasks. Refer Fig 2.13

Fig 2.13



```
from sklearn import preprocessing
import pandas as pd

# Assuming df_clustering is your dataset in a pandas DataFrame
min_max_scaler = preprocessing.MinMaxScaler()

# Apply Min-Max scaling and convert back to a DataFrame
df_clust_norm = pd.DataFrame(min_max_scaler.fit_transform(df_clustering), columns=df_clustering.columns)
```

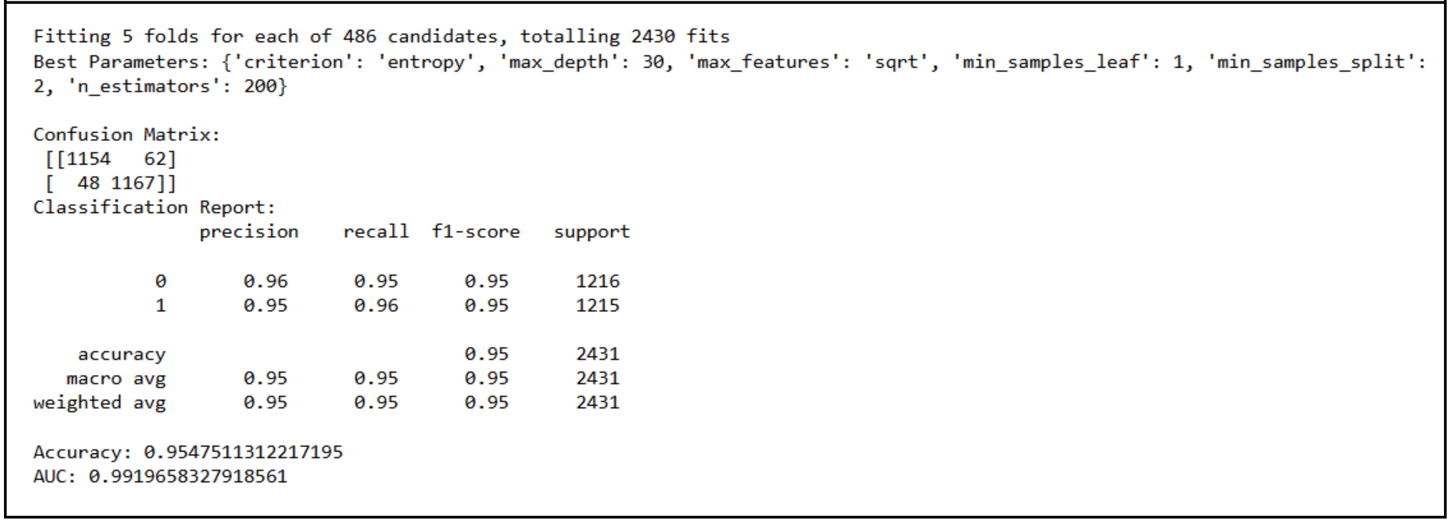
Chapter 3. Classification

3.1 Random Forest Classifier: Hyperparameter Tuning and Model Evaluation

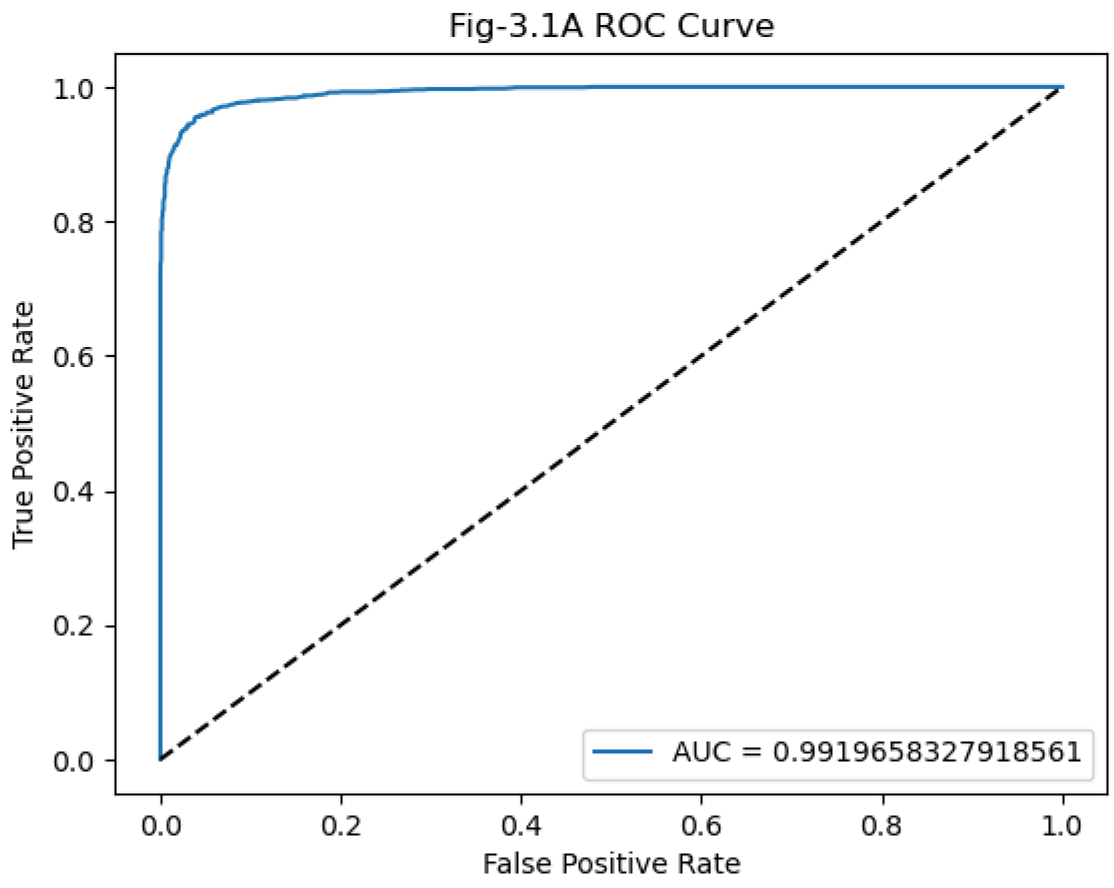
- The RandomForestClassifier and GridSearchCV are imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named ‘parameters’ which are as follows-
    - n_estimators: [100, 200, 300].
    - max_depth: [10, 20, 30].
    - min_samples_split: [2, 5, 10].
    - min_samples_leaf: [1, 2, 4].
    - criterion: ['gini', 'entropy', 'log_loss'].
    - max_features: ['sqrt', 'log2']
- The RandomForestClassifier is initialized as RF with random_state as 0 for the reproducibility of the code.
- A GridSearchCV object is created with the Random Forest model (RF), parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is accuracy.
- The grid search is performed on training data i.e X_train and y_train to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score with the help of y_test (actual values) and y_pred (predicted values) and Receiver Operating Characteristics curve.

Result:

Fig-3.1



```
Fitting 5 folds for each of 486 candidates, totalling 2430 fits
Best Parameters: {'criterion': 'entropy', 'max_depth': 30, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 200}

Confusion Matrix:
 [[1154   62]
 [  48 1167]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      1216
           1       0.95      0.96      0.95      1215

    accuracy                           0.95      2431
   macro avg       0.95      0.95      0.95      2431
weighted avg       0.95      0.95      0.95      2431

Accuracy: 0.9547511312217195
AUC: 0.9919658327918561
```



Analysis of the Result:

1. **Best Hyperparameters**

Table 3.1 A (Refer Fig-3.1)		
Hyperparameter	**Description**	**Best Value**
criterion	Function which is used to measure the quality of a split	entropy
max_depth	It is the maximum depth of each tree	30
max_features	Number of features for best split	sqrt
min_samples_leaf	Minimum number of samples required to be at the leaf node	1
min_samples_split	Minimum number of samples required to split an internal node	2

n_estimators	Total number of trees in the forest	200

2. **Analysis of the Confusion Matrix**

Table-3.1 B (Refer Fig-3.1)		
Actual/ Predicted	**Predicted Label 0 (Individuals who didn't report stroke)**	**Predicted Label 1 (individuals who reported stroke)**
Actual Label 0 (Individuals who didn't report stroke)	1154 instances of label 0 are correctly classified (True Negatives)	62 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (individuals who reported stroke)	48 instances of label 1 are misclassified as label 0 (False Negatives)	1167 instances of label 1 are correctly classified (True Positives)

3. **Analysis of Precision, Recall, and F1-Score**

Table-3.1 C (Refer Fig-3.1)	
Target Labels (Stroke)	**Analysis**
0 (Individuals who didn't report stroke)	A precision of 0.96 suggests that when the model classifies label 0, it is correct 96% of the time. A recall of 0.95 suggests that the model is able to capture 95% of actual label 0 instances. An f-1 score of 0.95 suggests the overall balance between precision and recall. There are 1216 actual instances of label 0 in total.
1 (individuals who reported stroke)	A precision of 0.95 suggests that when the model classifies label 1, it is correct 95% of the time. A recall of 0.96 suggests that the model is able to capture 96% of actual label 1 instances. An f-1 score of 0.95 suggests the overall balance between precision and recall. There are 1215 actual instances of label 1 in total.

4. **Analysis of Accuracy, Macro Average, and Weighted Average**

Table-3.1 D (Refer Fig-3.1)	
Accuracy	An accuracy of **95.47%** suggests that the model correctly classifies **95.47%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **95%**, for precision, recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **95%**, reflecting the overall performance weighted by class size.

5. **Analysis of the ROC Curve (Receiver Operating Characteristic Curve)**

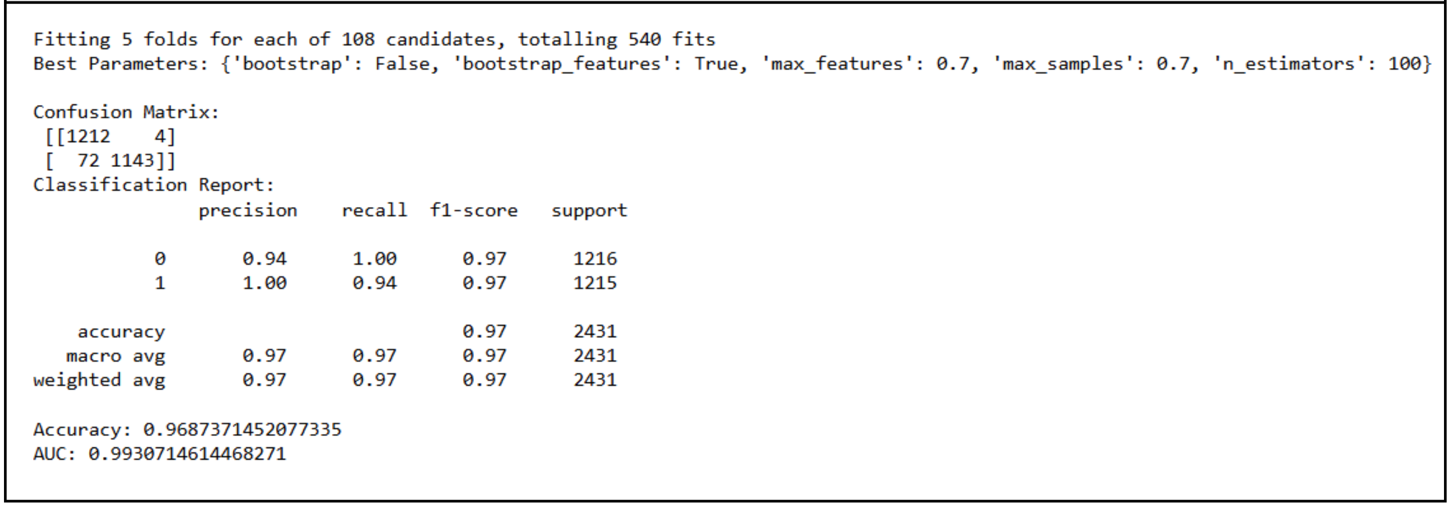
Table 3.1 E (Refer Fig-3.1A)	
Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.991 (AUC score). This high score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

3.2 Bagging Classifier: Hyperparameter Tuning and Model Evaluation

- The BaggingClassifier is imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named 'parameters' which are as follows-
    - n_estimators: [50, 100, 150]
    - max_samples: [0.5, 0.7, 1.0]
    - max_features: [0.5, 0.7, 1.0]
    - bootstrap: [True, False]
    - bootstrap_features: [True, False]
- The BaggingClassifier is initialized as BC with random_state as 0 for the reproducibility of the results.
- A GridSearchCV object is created with the BaggingClassifier (BC), the above-mentioned parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is accuracy.
- The grid search is performed on training data i.e X_train and y_train to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score with the help of y_test (actual values) and y_pred (predicted values) and Receiver Operating Characteristics curve.

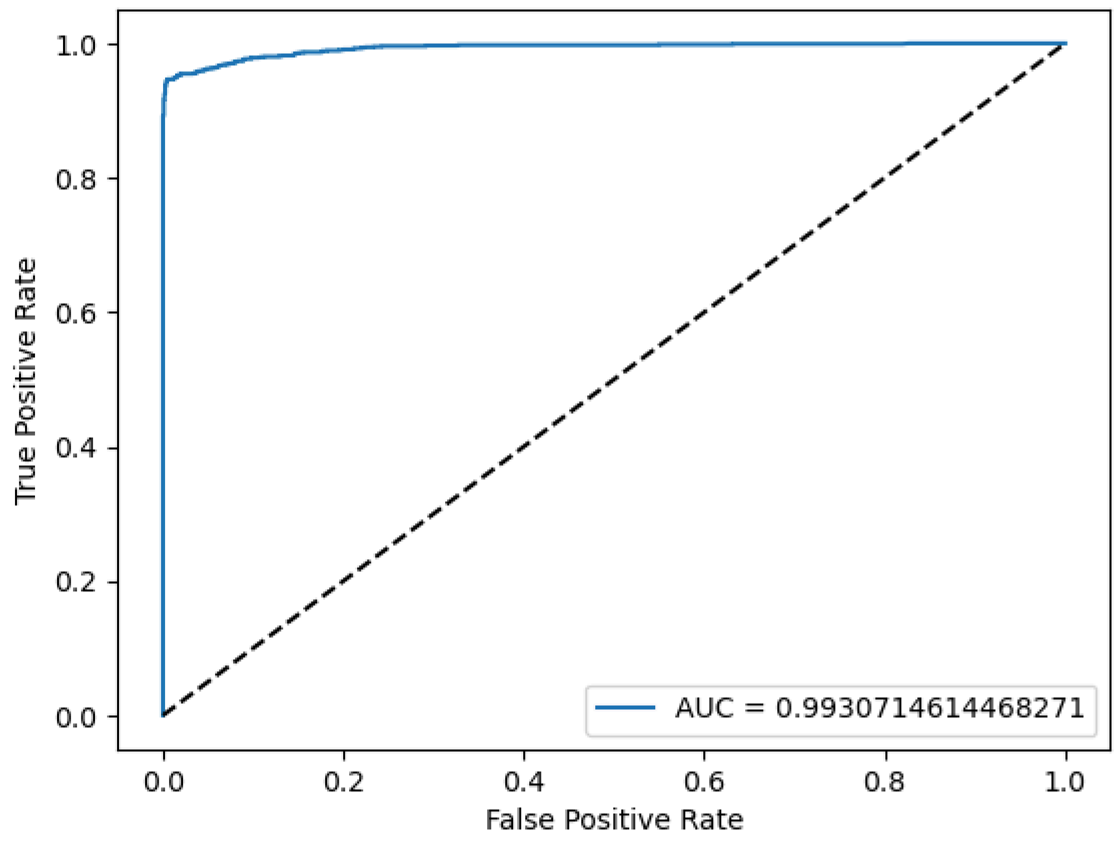
Result:

Fig-3.2



```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Parameters: {'bootstrap': False, 'bootstrap_features': True, 'max_features': 0.7, 'max_samples': 0.7, 'n_estimators': 100}

Confusion Matrix:
 [[1212    4]
 [  72 1143]]
Classification Report:
               precision    recall  f1-score   support

           0       0.94      1.00      0.97      1216
           1       1.00      0.94      0.97      1215

    accuracy                           0.97      2431
   macro avg       0.97      0.97      0.97      2431
weighted avg       0.97      0.97      0.97      2431

Accuracy: 0.9687371452077335
AUC: 0.9930714614468271
```

Fig-3.2A ROC Curve



Analysis of the Result:

1. **Best Hyperparameters**

Table 3.2 A (Refer Fig-3.2)		
Hyperparameter	**Description**	**Best Value**
bootstrap	Whether to sample data with replacement for training	False
bootstrap_features	Whether to sample features with replacement for training	True
max_features	Proportion of features sampled for training each base estimator (Decision Tree)	0.7
max_samples	Proportion of dataset sampled for training each base estimator (Decision Tree)	0.7
n_estimators	Total number of trees in the	100

	ensemble model	

2. **Analysis of the Confusion Matrix**

Table-3.2 B (Refer Fig-3.2)		
Actual/ Predicted	**Predicted Label 0 (Individuals who didn't report stroke)**	**Predicted Label 1 (individuals who reported stroke)**
Actual Label 0 (Individuals who didn't report stroke)	1212 instances of label 0 are correctly classified (True Negatives)	4 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (individuals who reported stroke)	72 instances of label 1 are misclassified as label 0 (False Negatives)	1143 instances of label 1 are correctly classified (True Positives)

3. **Analysis of Precision, Recall, and F1-Score**

Table-3.2 C (Refer Fig-3.2)	
Target Labels (Stroke)	**Analysis**
0 (Individuals who didn't report stroke)	A precision of 0.94 suggests that when the model classifies label 0, it is correct 94% of the time. A recall of 1.0 suggests that the model is able to capture 100% of actual label 0 instances. An f-1 score of 0.97 suggests the overall balance between precision and recall. There are 1216 actual instances of label 0 in total.
1 (individuals who reported stroke)	A precision of 1.0 suggests that when the model classifies label 1, it is correct 100% of the time. A recall of 0.94 suggests that the model is able to capture 94% of actual label 1 instances. An f-1 score of 0.97 suggests the overall balance between precision and recall. There are 1215 actual instances of label 1 in total.

4. **Analysis of Accuracy, Macro Average, and Weighted Average**

Table-3.2 D (Refer Fig-3.2)	
Accuracy	An accuracy of **0.9687** suggests that the model correctly classifies **96.87%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **97%**, for precision, recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **97%**, reflecting the overall performance weighted by class size.

5. **Analysis of the ROC Curve (Receiver Operating Characteristic Curve)**

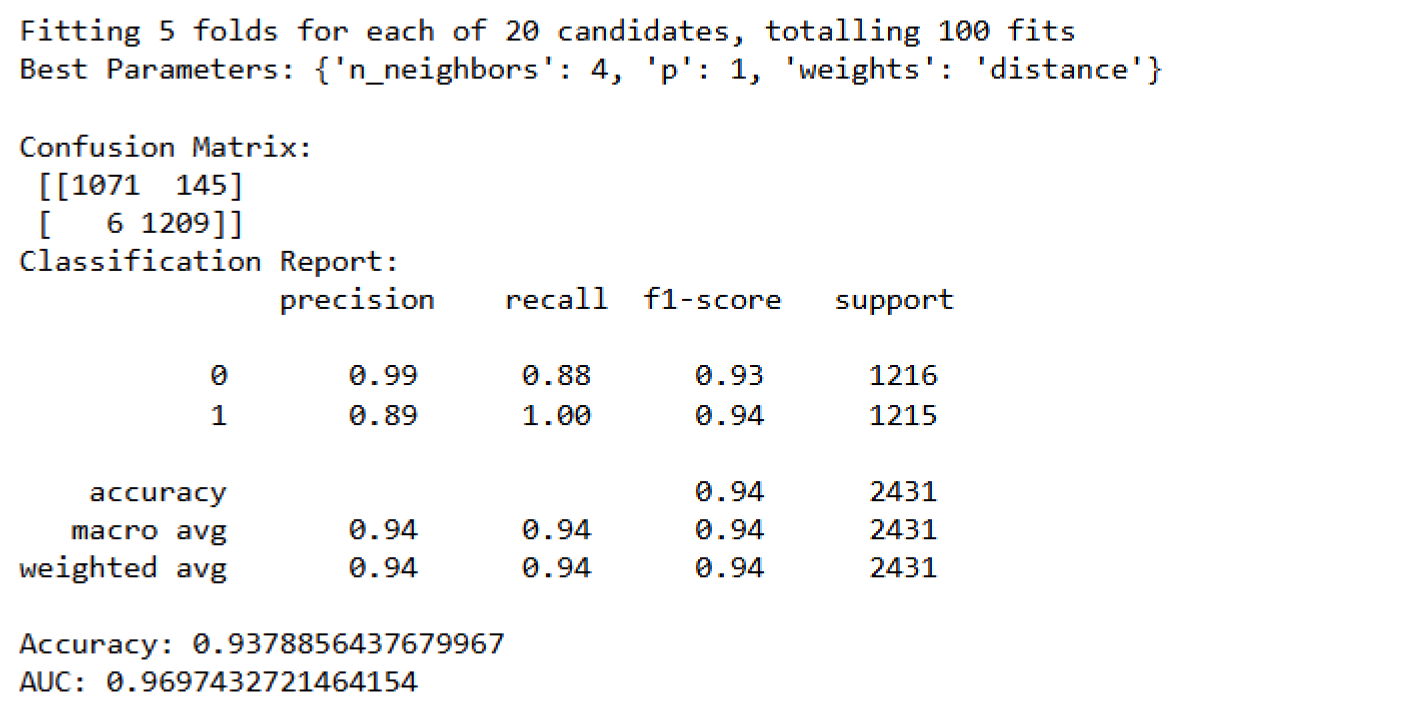
Table 3.2 E (Refer Fig-3.2A)	
Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.993 (AUC score). This high score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

3.3 KNeighborsClassifier: Hyperparameter Tuning and Model Evaluation

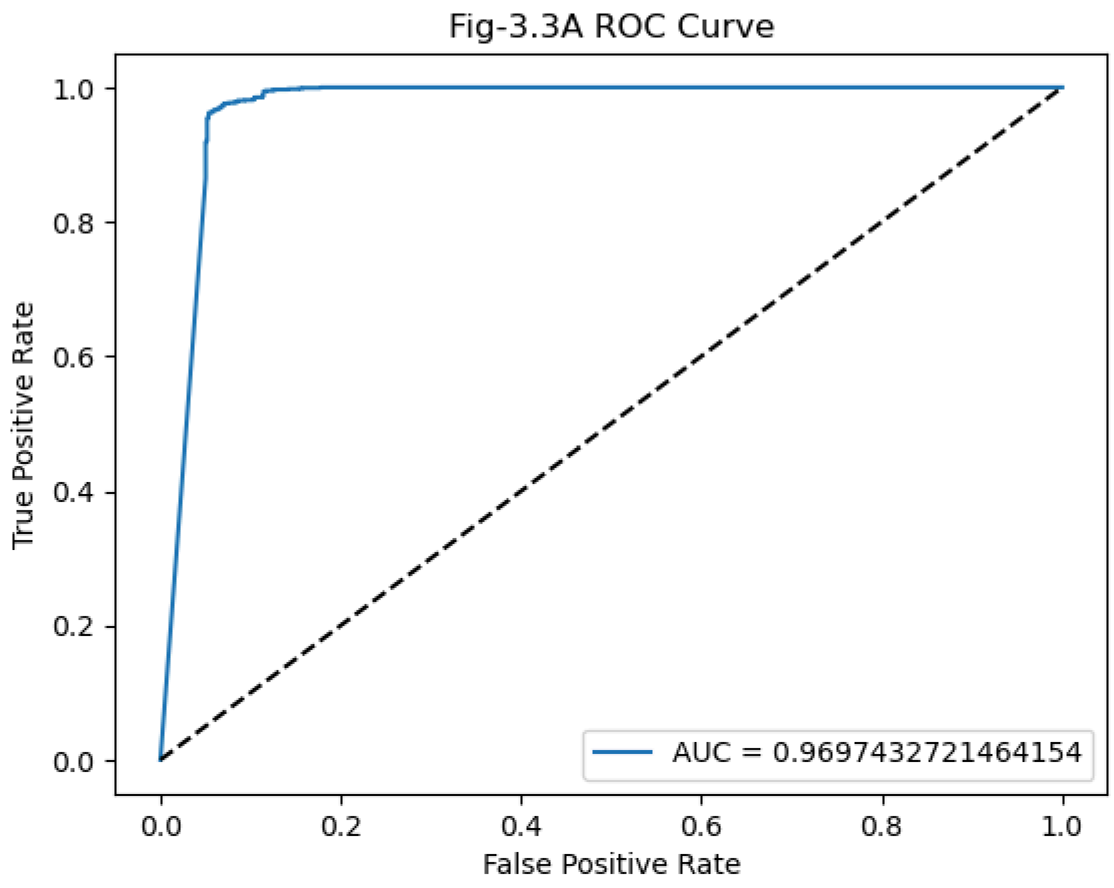
- The KNeighborsClassifier is imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named ‘parameters’ which are as follows-
  - n_neighbors: [3,4,5,6,7]
  - weights: [‘uniform’, ‘distance’]
  - p: [1,2]
- The KNeighborsClassifier is initialized as KNN.
- A GridSearchCV object is created with the KNeighborsClassifier (KNN), the above-mentioned parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is accuracy.
- The grid search is performed on training data i.e X_train and y_train to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score with the help of y_test (actual values) and y_pred (predicted values) and Receiver Operating Characteristics curve.

Result:

Fig-3.3



```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best Parameters: {'n_neighbors': 4, 'p': 1, 'weights': 'distance'}

Confusion Matrix:
 [[1071  145]
 [   6 1209]]
Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.88      0.93      1216
           1       0.89      1.00      0.94      1215

    accuracy                           0.94      2431
   macro avg       0.94      0.94      0.94      2431
weighted avg       0.94      0.94      0.94      2431

Accuracy: 0.9378856437679967
AUC: 0.9697432721464154
```



Analysis of the Result:

1. **Best Hyperparameters**

Table 3.3 A (Refer Fig-3.3)		
Hyperparameter	**Description**	**Best Value**
n_neighbors	Number of nearest neighbors	4
p	Distance metric used: 1 for Manhattan distance or 2 for Euclidean distance	1
weights	It is the criteria to weight the neighbor's votes: **uniform** (equal votes) or **distance** (closer ones count more)	distance

2. **Analysis of the Confusion Matrix**

Table-3.3 B (Refer Fig-3.3)		
Actual/ Predicted	**Predicted Label 0 (Individuals who didn't report stroke)**	**Predicted Label 1 (individuals who reported stroke)**
Actual Label 0 (Individuals who didn't report stroke)	1071 instances of label 0 are correctly classified (True Negatives)	145 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (individuals who reported stroke)	6 instances of label 1 are misclassified as label 0 (False Negatives)	1209 instances of label 1 are correctly classified (True Positives)

3. **Analysis of Precision, Recall, and F1-Score**

Table-3.3 C (Refer Fig-3.3)	
Target Labels (Stroke)	**Analysis**
0 (Individuals who didn't report stroke)	A precision of 0.99 suggests that when the model classifies label 0, it is correct 99% of the time. A recall of 0.88 suggests that the model is able to capture 88% of actual label 0 instances. An f-1 score of 0.93 suggests the overall balance between precision and recall. There are 1216 actual instances of label 0 in total.
1 (individuals who reported stroke)	A precision of 0.89 suggests that when the model classifies label 1, it is correct 89% of the time. A recall of 1.0 suggests that the model is able to capture 100% of actual label 1 instances. An f-1 score of 0.94 suggests the overall balance between precision and recall. There are 1215 actual instances of label 1 in total.

4. **Analysis of Accuracy, Macro Average, and Weighted Average**

Table-3.3 D (Refer Fig-3.3)	
Accuracy	An accuracy of **0.9378** suggests that the model correctly classifies **93.78%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **94%**, for precision, recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **94%**, reflecting the overall performance weighted by class size.

5. **Analysis of the ROC Curve (Receiver Operating Characteristic Curve)**

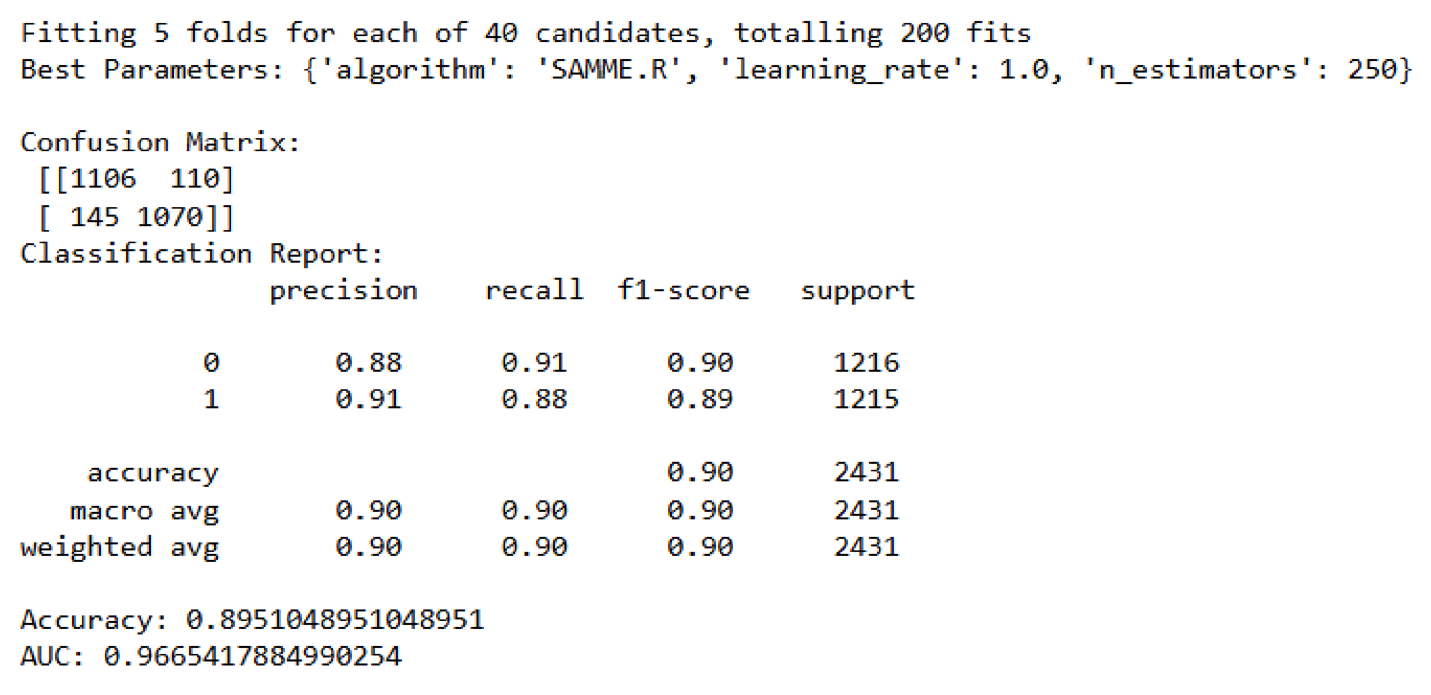
Table 3.3 E (Refer Fig-3.3 A)	
Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.969 (AUC score). This score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

3.4 AdaBoostClassifier: Hyperparameter Tuning and Model Evaluation (EXTRA)

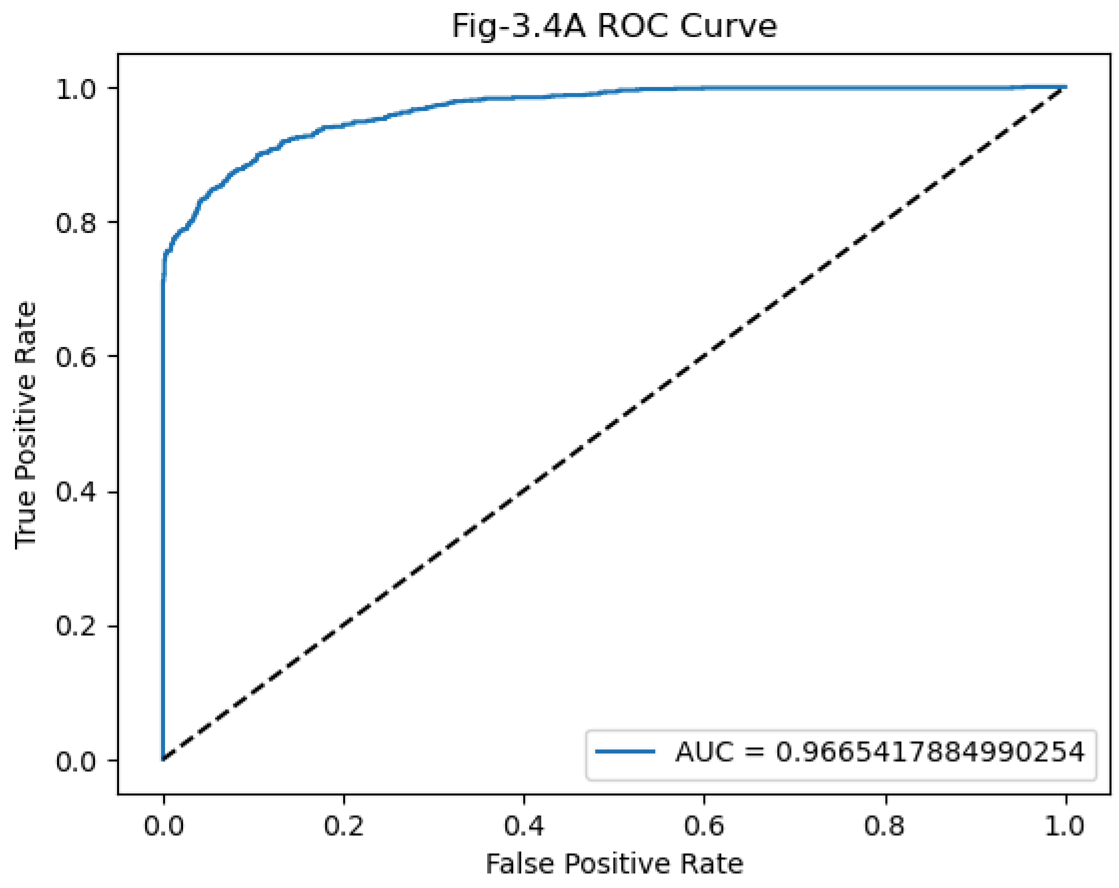
- The AdaBoostClassifier is imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named ‘parameters’ which are as follows-
  - n_estimators: [100, 150, 200, 250]
  - learning_rate: [0.001, 0.01, 0.1, 1.0, 2.0]
  - algorithm: [‘SAMME’, ‘SAMME.R’]
- The AdaBoostClassifier is initialized as ABC with random_state as 0 for the reproducibility of the results
- A GridSearchCV object is created with the AdaBoostClassifier (ABC), the above-mentioned parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is accuracy.
- The grid search is performed on training data i.e X_train and y_train to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using a confusion matrix, classification report, accuracy score with the help of y_test (actual values) and y_pred (predicted values) and Receiver Operating Characteristics curve.

Result:

Fig-3.4



```
Fitting 5 folds for each of 40 candidates, totalling 200 fits
Best Parameters: {'algorithm': 'SAMME.R', 'learning_rate': 1.0, 'n_estimators': 250}

Confusion Matrix:
 [[1106  110]
 [ 145 1070]]
Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.91      0.90      1216
           1       0.91      0.88      0.89      1215

    accuracy                           0.90      2431
   macro avg       0.90      0.90      0.90      2431
weighted avg       0.90      0.90      0.90      2431

Accuracy: 0.8951048951048951
AUC: 0.9665417884990254
```



Analysis of the Result:

1. **Best Hyperparameters**

Table 3.4 A (Refer Fig-3.4)		
Hyperparameter	**Description**	**Best Value**
n_estimators	Number of boosting rounds	250
learning_rate	The rate at which the model learns, smaller values make the model learn slower	1.0
algorithm	The boosting algorithm: **SAMME** (discrete boosting) or **SAMME.R** (real boosting, weighted).	SAMME.R

2. **Analysis of the Confusion Matrix**

Table-3.4 B (Refer Fig-3.4)		
Actual/ Predicted	**Predicted Label 0 (Individuals who didn’t report stroke)**	**Predicted Label 1 (individuals who reported stroke)**
Actual Label 0 (Individuals who didn’t report stroke)	1106 instances of label 0 are correctly classified (True Negatives)	110 instances of label 0 are misclassified as label 1 (False Positives)
Actual Label 1 (individuals who reported stroke)	145 instances of label 1 are misclassified as label 0 (False Negatives)	1070 instances of label 1 are correctly classified (True Positives)

3. **Analysis of Precision, Recall, and F1-Score**

Table-3.4 C (Refer Fig-3.4)	
Target Labels (Stroke)	**Analysis**
0 (Individuals who didn’t report stroke)	A precision of 0.88 suggests that when the model classifies label 0, it is correct 88% of the time. A recall of 0.91 suggests that the model is able to capture 91% of actual label 0 instances. An f-1 score of 0.90 suggests the overall balance between precision and recall. There are 1216 actual instances of label 0 in total.
1 (individuals who reported stroke)	A precision of 0.91 suggests that when the model classifies label 1, it is correct 91% of the time. A recall of 0.88 suggests that the model is able to capture 88% of actual label 1 instances. An f-1 score of 0.89 suggests the overall balance between precision and recall. There are 1215 actual instances of label 1 in total.

4. **Analysis of Accuracy, Macro Average, and Weighted Average**

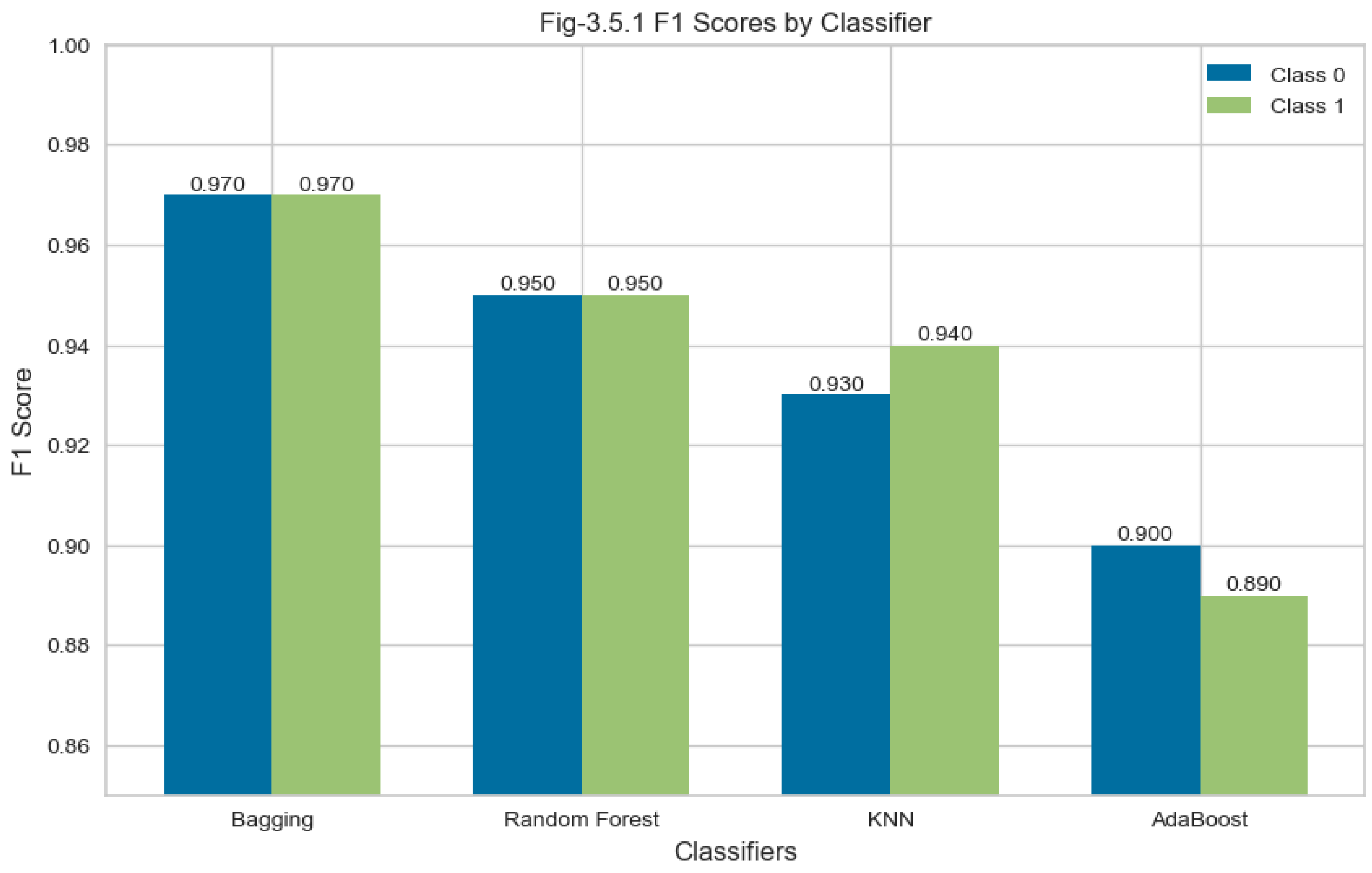
Table-3.4 D (Refer Fig-3.4)	
Accuracy	An accuracy of **0.8951** suggests that the model correctly classifies **89.51%** of the total instances.
Macro Average	This is the average of precision, recall, and F1-score across all classes, it is calculated without considering the support of each class. The macro average of **90%**, for precision, recall and f1 score suggests that on average, both the classes perform at a similar level.
Weighted Average	This is the average of precision, recall, and F1-score and it takes into account the support of each class. The weighted average in this case is **90%**, reflecting the overall performance weighted by class size.

5. **Analysis of the ROC Curve (Receiver Operating Characteristic Curve)**

Table 3.4 E (Refer Fig-3.4 A)	
Aspect	**Observation**
AUC (Area Under the Curve)	The area under the ROC curve is 0.966 (AUC score). This score indicates that the model can effectively distinguish between the positive and negative class in our dataset.
Low False Positive Rate	The ROC curve remains close to the top-left which indicates minimum false positives
High True Positive Rate	The model effectively captures large number of true positive instances.
Above the Diagonal	The ROC curve lies significantly above the diagonal line which means that the model is not randomly guessing the outcomes.

3.5 Comparison and Analysis of the Classifiers used based on F1 Score

3.5.1 Visualization of the classifiers and ranking them on the basis of F1 Score of each class (class 0 and class 1)



3.5.2 Analysis and Comparision

Table 3.5.2 (Refer *Fig-3.5.1)*

Classifier	**F1 Score (Class 0)**	**F1 Score (Class 1)**	**Analysis**	**Rank**
Bagging	0.970	0.970	It performs the best among all the classifiers used and handles both the classes very well	**1**
Random Forest	0.950	0.950	It performs slightly less than the Bagging Classifier and classifies both the classes very well	**2**
KNN	0.930	0.940	It performs good but less than Bagging and Random	**3**

			Forest. The F1 score of class 1 is slightly more than class 0 indicating better classification performance for class 1	
AdaBoost	0.900	0.890	It performs decent but ranks the lowest among all the above classifiers used. The F1 score of class 0 is slightly more than class 1 indicating better classification performance for class 0	**4**

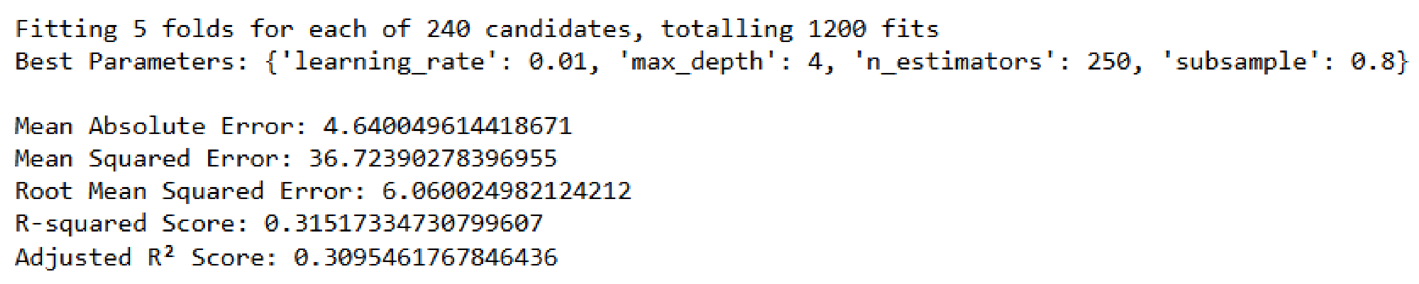
Chapter 4. Regression

4.1 Gradient Boosting Regressor: Hyperparameter Tuning and Model Evaluation (EXTRA)

- The GradientBoostingRegressor is imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named 'parameters' which are as follows-
  - n_estimators: [100, 150, 200, 250]
  - learning_rate: [0.001, 0.01, 0.1, 1.0, 2.0]
  - max_depth: [3, 4, 5, 6]
  - subsample: [0.6, 0.8, 1.0]
- The GradientBoostingRegressor is initialized as GBR with random_state as 0 for the reproducibility of the results
- A GridSearchCV object is created with the GradientBoostingRegressor (GBR), the above-mentioned parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is R-squared score (r2).
- The grid search is performed on training data i.e X_train_reg and y_train_reg to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test_reg) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using Mean Absolute Error, Mean Squared Error, Root Mean Squared Error, R-Sqaured score, and Adjusted R-Squared score with the help of y_test_reg (actual values) and y_pred (predicted values).

Result:

Fig-4.1



```
Fitting 5 folds for each of 240 candidates, totalling 1200 fits
Best Parameters: {'learning_rate': 0.01, 'max_depth': 4, 'n_estimators': 250, 'subsample': 0.8}

Mean Absolute Error: 4.640049614418671
Mean Squared Error: 36.72390278396955
Root Mean Squared Error: 6.060024982124212
R-squared Score: 0.31517334730799607
Adjusted R² Score: 0.3095461767846436
```

Analysis of the Result:

1. **Best Hyperparameters**

Table 4.1 A (Refer Fig-4.1)		
Hyperparameter	**Description**	**Best Value**
learning_rate	The rate at which the model learns	0.01
max_depth	The maximum depth of each tree	4
n_estimators	The number of boosting rounds (iterations)	250
subsample	The fraction of samples used to train each base learner.	0.8

2. **Interpretation of Mean Absolute Error, Mean Squared Error, Root Mean Square Error, R-Squared Score and Adjusted R-Squared Score**

Table-4.1 B (Refer Fig-4.1)	
Mean Absolute Error	The predictions made by our model vary by around **4.64** units from the actual values.
Mean Squared Error	This metric suggests that on average the squared difference between the actual values and the predicted values is **36.723** units. This measure has more impact on the large errors made by our model.
Root Mean Square Error	This metric suggests that the square root of the mean squared error is **6.060.** This measure gives more weight to the larger errors made by our model making it more sensitive to large prediction mistakes.
R-Squared Score	This measure suggests that **31.5173%** of the fluctuations or variance in the target variable can be explained by our trained model.

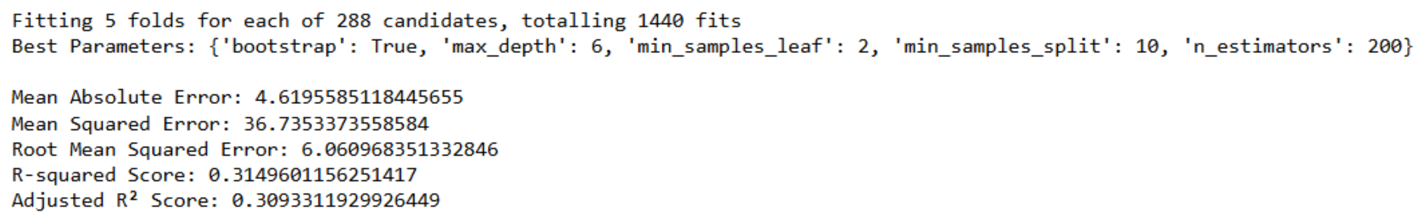
Adjusted R-Squared Score	This measure adjusts the R-squared score based on the number of features in the model and it penalizes the score for unwanted features which makes this measure more reliable. An adjusted R2 score of **30.954 %** suggests that our model is able to explain the fluctuations or variance taking into consideration the features used to train the model.

4.2 Random Forest Regressor: Hyperparameter Tuning and Model Evaluation

- The RandomForestRegressor is imported from the sklearn library.
- The hyperparameters chosen for GridSearchCV and the corresponding values are stored in a variable named ‘parameters’ which are as follows-
    - n_estimators: [100, 150, 200, 250]
    - max_depth: [3, 4, 5, 6]
    - min_samples_split: [2, 5, 10]
    - min_samples_leaf: [1, 2, 4]
    - bootstrap: [True, False]
- The RandomForestRegressor is initialized as RF with random_state as 0 for the reproducibility of the results
- A GridSearchCV object is created with the RandomForestRegressor (RF), the above-mentioned parameters, and a 5-fold cross-validation (cv=5). The scoring metric used for evaluation is R-squared score (r2).
- The grid search is performed on training data i.e X_train_reg and y_train_reg to find the optimal combination of hyperparameters
- The best model is then identified using grid_search.best_estimator_, and predictions are made using the test data (X_test_reg) based on the best parameters. The predictions made are stored as y_pred.
- The performance of the model is evaluated using Mean Absolute Error, Mean Squared Error, Root Mean Squared Error, R-Sqaured score, and Adjusted R-Squared score with the help of y_test_reg (actual values) and y_pred (predicted values).

Result:

Fig-4.2



```
Fitting 5 folds for each of 288 candidates, totalling 1440 fits
Best Parameters: {'bootstrap': True, 'max_depth': 6, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 200}

Mean Absolute Error: 4.6195585118445655
Mean Squared Error: 36.7353373558584
Root Mean Squared Error: 6.060968351332846
R-squared Score: 0.3149601156251417
Adjusted R² Score: 0.3093311929926449
```

Analysis of the Result:

1. **Best Hyperparameters**

Table 4.2 A (Refer Fig-4.2)

Hyperparameter	**Description**	**Best Value**
bootstrap	Whether to sample data with replacement for training	True
max_depth	The maximum depth of each tree	6
min_samples_leaf	The minimum number of samples required to be at a leaf node.	2
min_samples_split	The minimum number of samples required to split an internal node.	10
n_estimators	The number of trees in the forest	200

2. **Interpretation of Mean Absolute Error, Mean Squared Error, Root Mean Square Error, R-Squared Score and Adjusted R-Squared Score**

Table-4.2 B (Refer Fig-4.2)	
Mean Absolute Error	The predictions made by our model vary by around **4.619** units from the actual values.
Mean Squared Error	This metric suggests that on average the squared difference between the actual values and the predicted values is **36.735** units. This measure has more impact on the large errors made by our model.
Root Mean Square Error	This metric suggests that the square root of the mean squared error is **6.06.** This measure gives more weight to the larger errors made by our model making it more sensitive to large prediction mistakes.
R-Squared Score	This measure suggests that **31.49%** of the fluctuations or variance in the target variable can be explained by our trained model.
Adjusted R-Squared Score	This measure adjusts the R-squared score based on the number of features in the model and it penalizes the score for unwanted features which makes this measure more reliable. An adjusted R2 score of **30.93 %** suggests that our model is able to explain the fluctuations or variance taking into consideration the features used to train the model.

4.3 Comparision and Analysis of the Regression Models

Table-4.3

Metric	Gradient Boosting Regressor	Random Forest Regressor	Analysis
Mean Absolute Error	4.640049614418671	4.6195585118445655	The Gradient Boosting Regressor has a slightly higher Mean Absolute Error than the Random Forest Regressor, indicating it makes marginally larger errors on average
Mean Squared Error	36.72390278396955	36.7353373558584	The Mean Squared Error is almost identical between the two models, with Gradient Boosting Regressor having a slightly lower value
Root Mean Square Error	6.060024982124212	6.060968351332846	The Root Mean Square Error is also very close between the two models, suggesting both models perform similarly in terms of penalizing larger errors
R-Squared Score	0.31517334730799607	0.3149601156251417	Both models have very similar R-Squared Score with Gradient Boosting having a slightly higher score, indicating that it is able to explain a marginally higher proportion of the variance in the target variable
Adjusted R-Squared Score	0.3095461767846436	0.3093311929926449	Both models have very similar Adjusted

			R-Squared Score with Gradient Boosting having a slightly higher score, indicating that it is able to explain a marginally higher proportion of the variance in the target variable
Conclusion: The differences between the two models are very minimal. Gradient Boosting Regressor slightly performs better in terms of Mean Squared Error, Root Mean Square Error, R-Squared and Adjusted R-Squared scores, while Random Forest Regressor performs slightly better in terms of Mean Absolute Error.			

Chapter 5. Clustering

5.1 K-Means Clustering

- The KMeans clustering algorithm is first initialised with two clusters.
- The model is trained on df_clust_norm (a normalised dataframe using MinMaxScaler() which contains all the features)
- After modelling, the cluster centers are viewed.
- The silhouette score is computed for the clustering.
- The elbow curve is plotted to determine the optimum number of clusters for better clustering (point appeared at k=2; suggesting 2 as the optimum number of clusters)
- The silhouette visualiser is plotted to graphically analyse the performance of clustering.
- A new column named 'cluster' is appended to the 'df_clust_norm' to store the cluster labels assigned to each data point by the KMeans algorithm.
- Properties of both the clusters are viewed.
- For both the cluster, mean values of all attributes are computed from the cluster centers.
- A bar chart is plotted to analyse the mean values for the clusters.
- Statistical summary of the two clusters are analysed.

Results:

***Fig-5.1 (A)-* Elbow Curve**

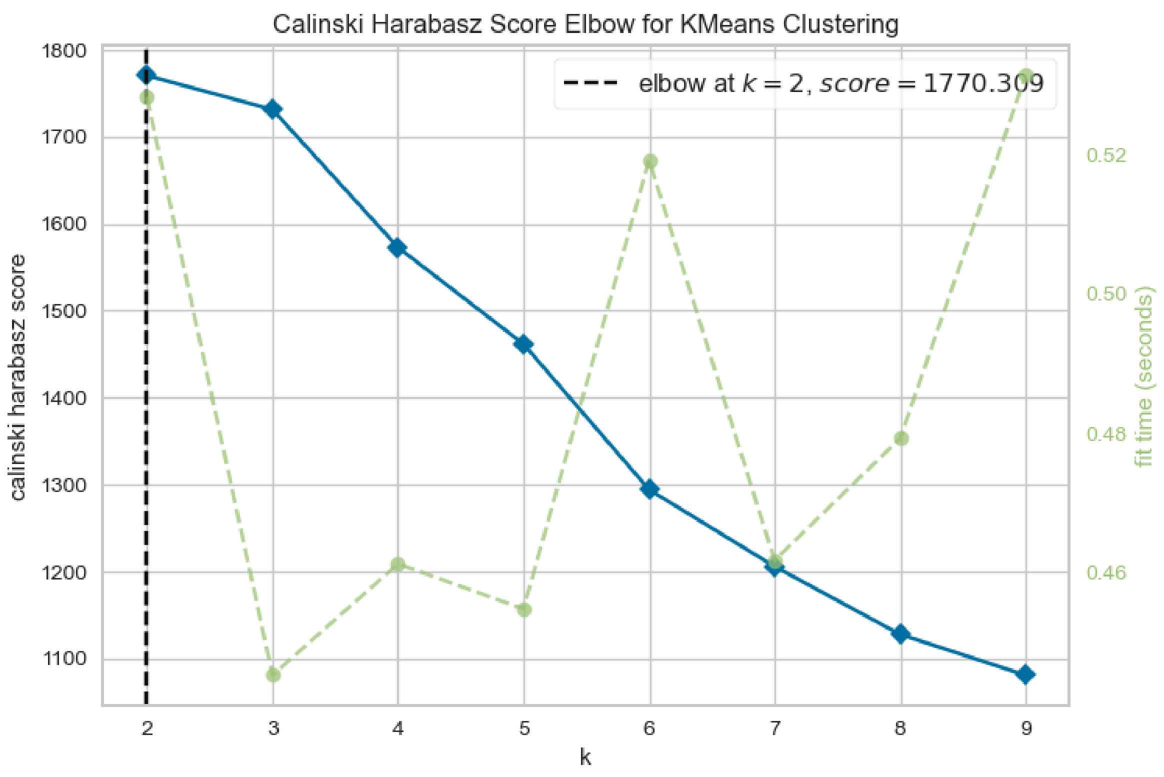


***Fig-5.1 (B)*- Silhouette Plot**

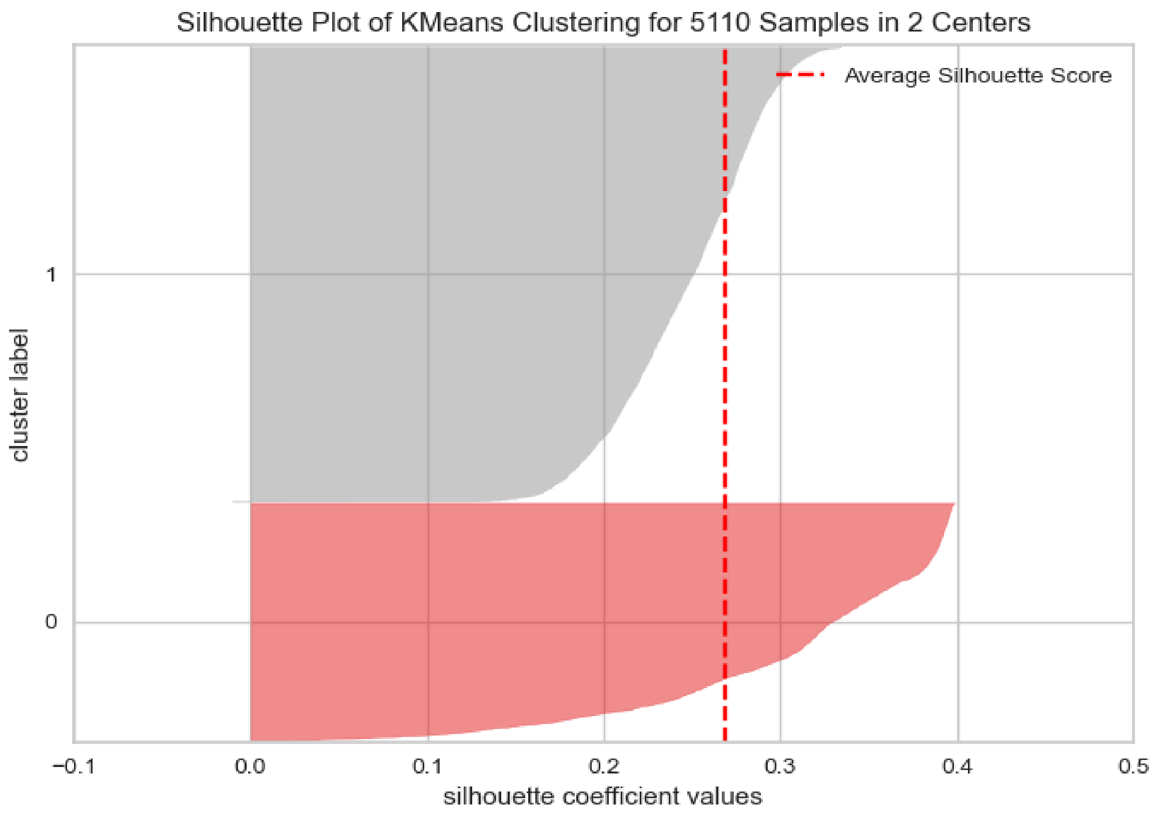
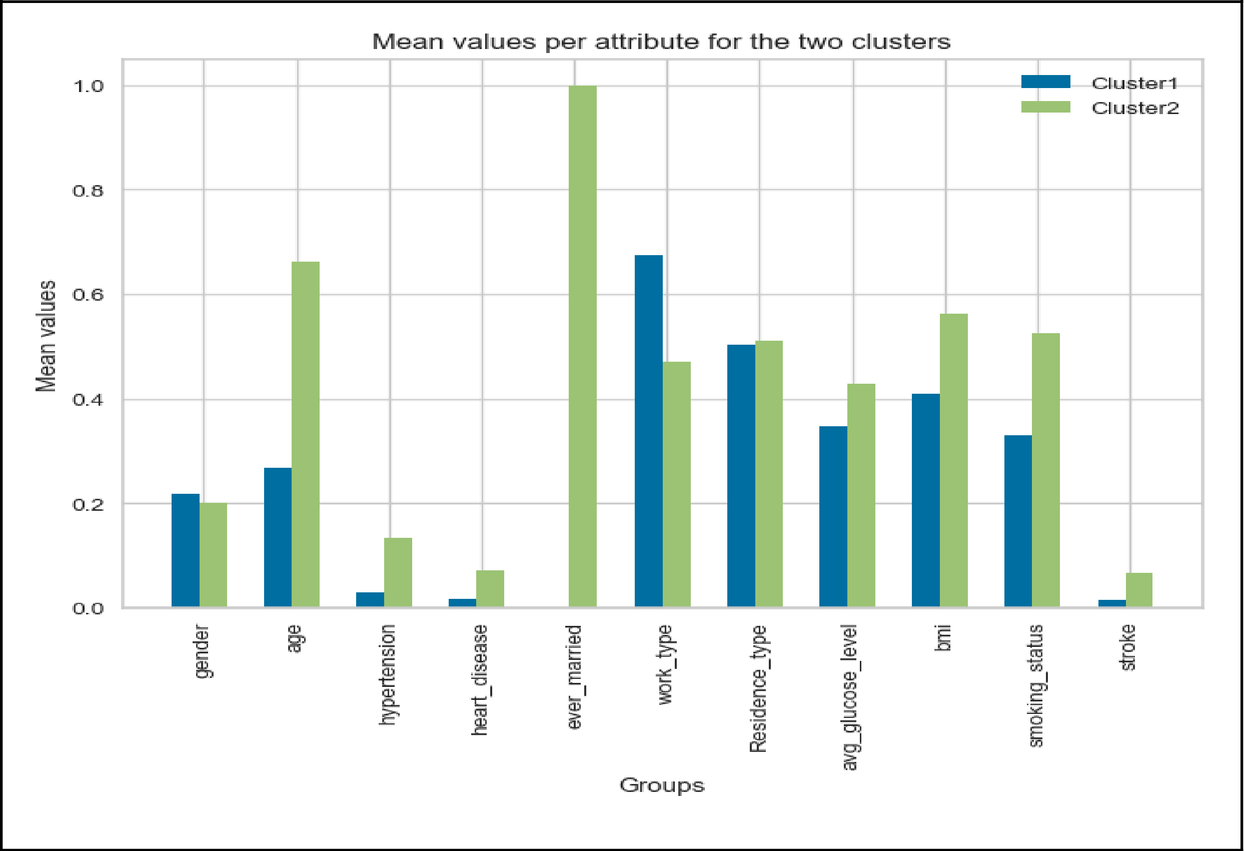


***Fig-5.1 (C)*- Mean values per Attribute for the two Clusters**



Interpretation and Analysis of the above Results:

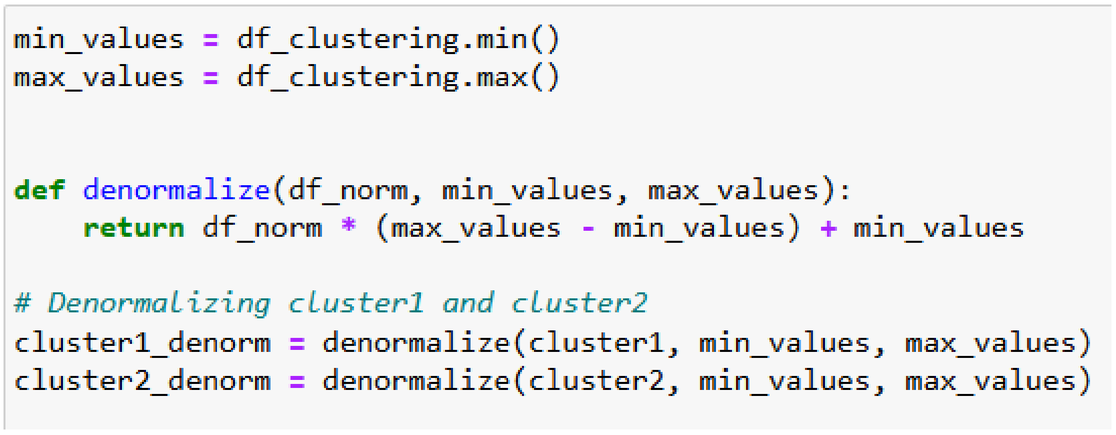
Result	**Interpretation**
Elbow Curve (Refer Fig- 5.1 (A))	The elbow curve suggests that the optimal number of clusters for better clustering performance is 2 with the highest Calinski Harabasz score of 1770.309; this point is the elbow point. This score suggests the best trade-off between compactness of the cluster (how tightly packed the points are in each cluster) and the separation of the clusters (how distinct the clusters are from each other). The elbow point suggests that adding more clusters beyond 2 doesn't significantly improve the clustering quality.
Silhouette Plot (Refer Fig- 5.1 (B))	This plot helps us visualise the silhouette scores for the two clusters; indicating an average silhouette score of 0.27 which shows decent clustering quality. A score nearer to 1 is more desirable. For both the clusters (cluster 0 and cluster 1) almost all the data points have

	silhouette scores in the positive range which indicates good intra-cluster cohesion and inter-cluster seperation. However, an extremely small number of data points have silhouette scores in the negative range for cluster 1 which demonstrates that these points are randomly and ambiguously assigned to this cluster.
Visualisation of mean values per attribute for the two clusters (Refer Fig-5.1 C))	We can see that some attributes show a significant difference in mean values between the two clusters which indicates better clustering and more distinct separation between the groups. These include- ‘age,’ ‘avg_glucose_level,’ ‘bmi,’ ‘ever_married,’ ‘heart_disease,’ ‘hypertension,’ and ‘stroke.’

Denormalization and Conversion of specific attributes to Categorical

While performing clustering, all the features were transformed into numerical values and normalized. After successfully performing clustering, we now aim to analyze the statistical summary of the two clusters. To ensure accurate and meaningful interpretation, all features are denormalized within both clusters. Since the denormalized features are actually categorical in nature but represented as numerical, they are converted back to their categorical form for proper statistical analysis. All columns, except 'age,' 'avg_glucose_level,' and 'bmi,' are converted to categorical in both clusters. Refer Fig- *5.1 **(D)*** and Fig- 5.1 (E)

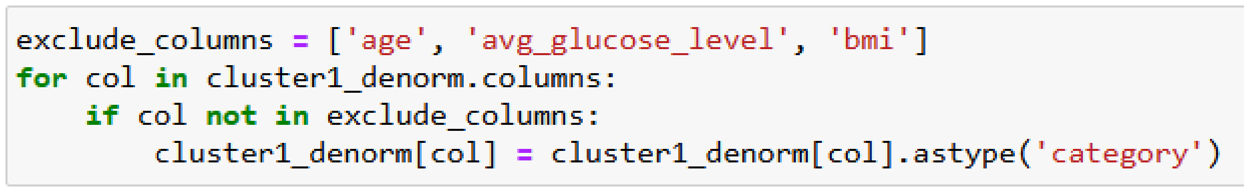
Fig- 5.1 (D) Denormalization



```
min_values = df_clustering.min()
max_values = df_clustering.max()


def denormalize(df_norm, min_values, max_values):
    return df_norm * (max_values - min_values) + min_values

# Denormalizing cluster1 and cluster2
cluster1_denorm = denormalize(cluster1, min_values, max_values)
cluster2_denorm = denormalize(cluster2, min_values, max_values)
```

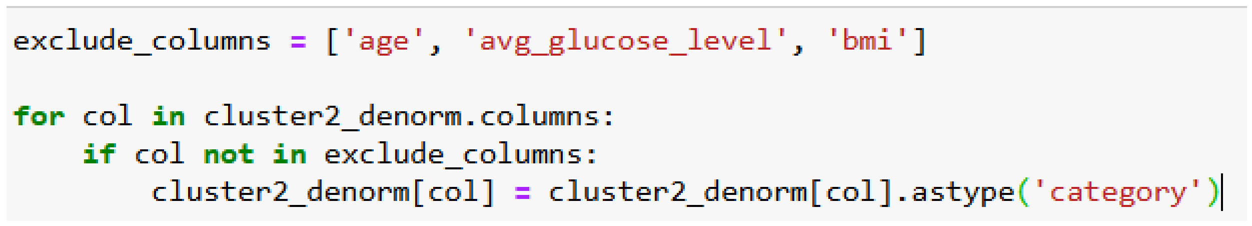
Fig 5.1 (E) Conversion to Categorical

Conversion of Cluster 1 Features



```
exclude_columns = ['age', 'avg_glucose_level', 'bmi']
for col in cluster1_denorm.columns:
    if col not in exclude_columns:
        cluster1_denorm[col] = cluster1_denorm[col].astype('category')
```

Conversion of Cluster 2 Features

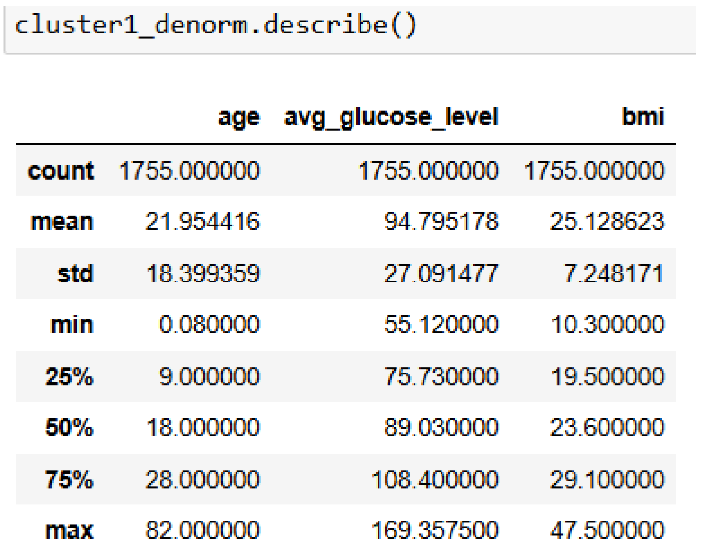


```
exclude_columns = ['age', 'avg_glucose_level', 'bmi']

for col in cluster2_denorm.columns:
    if col not in exclude_columns:
        cluster2_denorm[col] = cluster2_denorm[col].astype('category')
```

Statistical Summary of the two Clusters:

Table 5.1 (i)- Statistical Summary of the Numerical Attributes

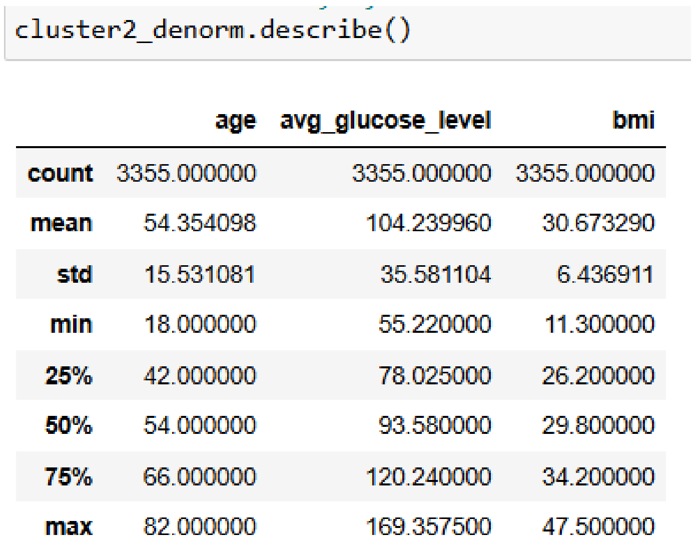
Cluster 1 (Numerical Attributes)



```
cluster1_denorm.describe()
```

```
               age  avg_glucose_level          bmi
count  1755.000000        1755.000000  1755.000000
mean     21.954416          94.795178    25.128623
std      18.399359          27.091477     7.248171
min       0.080000          55.120000    10.300000
25%       9.000000          75.730000    19.500000
50%      18.000000          89.030000    23.600000
75%      28.000000         108.400000    29.100000
max      82.000000         169.357500    47.500000
```

Cluster 2 (Numerical Attributes)

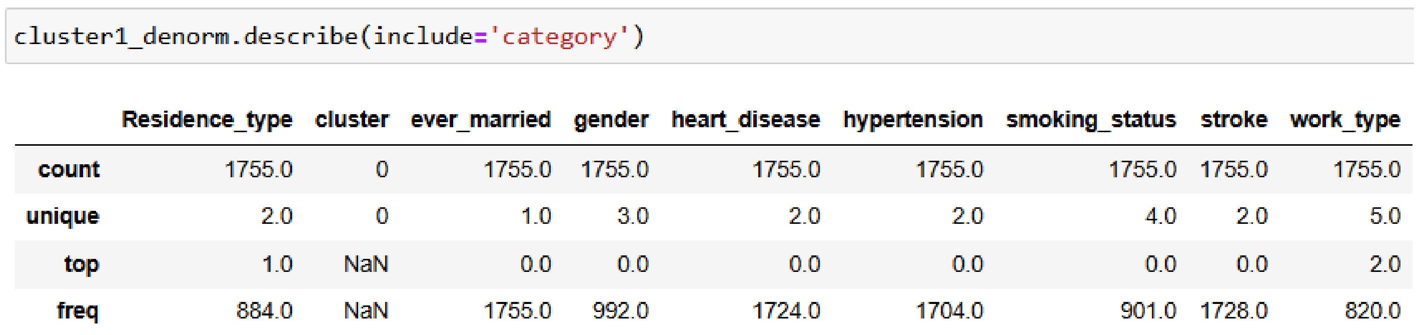


```
cluster2_denorm.describe()
```

```
               age  avg_glucose_level          bmi
count  3355.000000        3355.000000  3355.000000
mean     54.354098         104.239960    30.673290
std      15.531081          35.581104     6.436911
min      18.000000          55.220000    11.300000
25%      42.000000          78.025000    26.200000
50%      54.000000          93.580000    29.800000
75%      66.000000         120.240000    34.200000
max      82.000000         169.357500    47.500000
```

Table 5.1 (ii)- Statistical Summary of the Categorical Features

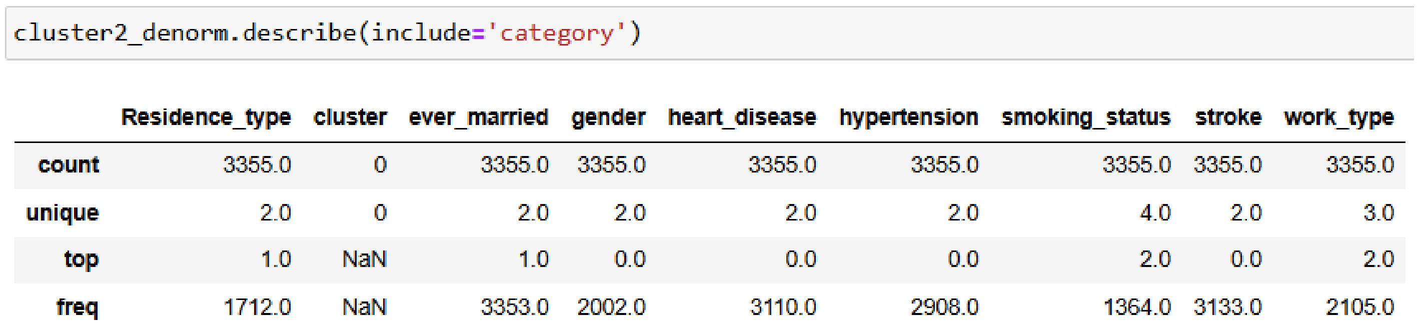
Cluster 1 (Categorical Attributes)



```
cluster1_denorm.describe(include='category')
```

```
        Residence_type  cluster  ever_married  gender  heart_disease  hypertension  smoking_status  stroke  work_type
 count          1755.0        0        1755.0  1755.0         1755.0        1755.0          1755.0  1755.0     1755.0
unique             2.0        0           1.0     3.0            2.0           2.0             4.0     2.0        5.0
   top             1.0      NaN           0.0     0.0            0.0           0.0             0.0     0.0        2.0
  freq           884.0      NaN        1755.0   992.0         1724.0        1704.0           901.0  1728.0      820.0
```

Cluster 2 (Categorical Attributes)



```
cluster2_denorm.describe(include='category')
```

```
        Residence_type  cluster  ever_married  gender  heart_disease  hypertension  smoking_status  stroke  work_type
 count          3355.0        0        3355.0  3355.0         3355.0        3355.0          3355.0  3355.0     3355.0
unique             2.0        0           2.0     2.0            2.0           2.0             4.0     2.0        3.0
   top             1.0      NaN           1.0     0.0            0.0           0.0             2.0     0.0        2.0
  freq          1712.0      NaN        3353.0  2002.0         3110.0        2908.0          1364.0  3133.0     2105.0
```

Analysis of the Statistical Summary of the Numerical Attributes for the two Clusters (Refer Table 5.1 (i))

Table- 5.1 (iii)

Features	Analysis
age	In **cluster 1**, the average age of individuals is 21.95 years and the median (50%) age is 18 years which means 50% of the individuals in cluster 1 are 18 years or less. This cluster contains younger individuals.
	In **cluster 2**, the average age of individuals is 54.35 years and the median (50%) age is 54 years which means 50% of the individuals in cluster 2 are 54 years or less. This cluster contains older individuals as compared to cluster 1.
avg_glucose_level	In **cluster 1**, the mean of avg_glucose_level is 94.79 units and the median (50%) glucose level is 89.03 units which means 50% of the

	individuals in cluster 1 have a glucose level of 89.03 units or less. This cluster contains individuals with lower glucose levels
	In **cluster 2**, the mean of avg_glucose_level is 104.23 units and the median (50%) glucose level is 93.58 units which means 50% of the individuals in cluster 2 have a glucose level of 93.58 units or less. This cluster contains individuals with higher glucose levels as compared to cluster 1.
bmi	In **cluster 1**, the mean bmi is 25.12 units and the median (50%) bmi is 23.6 units which means 50% of the individuals in cluster 1 have a bmi value of 23.6 units or less. This cluster contains individuals with lower bmi values.
	In **cluster 2**, the mean bmi is 30.67 units and the median (50%) bmi is 29.8 units which means 50% of the individuals in cluster 2 have a bmi value of 29.8 units or less. This cluster contains individuals with higher bmi values as compared to cluster 1.

Analysis of the Statistical Summary of the Categorical Attributes for the two Clusters (Refer Table 5.1 (ii)

Table 5.1 (iv)	
Features	**Analysis**
Residence_type	In **cluster 1**, the most frequently occurring value is '1', which represents urban (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times '1' (i.e., urban) appears in cluster 1 is 884 out of a total of 1755 instances. The number of times the other category, '0' (Rural), appears is 871. This cluster is a mixture of the two categories (884 instances of '1' and 871 instances of '0'), indicating that the cluster is not well defined for the 'Residence_type' feature.
	In **cluster 2**, the most frequently occurring value

	is ‘1,’ which represents urban (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘1’ (i.e., urban) appears in cluster 2 is 1712 out of a total of 3355 instances. The number of times the other category, ‘0’ (Rural), appears is 1643 . This cluster is a mixture of the two categories (1712 instances of ‘1’ and 1643 instances of ‘0’), indicating that the cluster is not well defined for the ‘Residence_type’ feature.
ever_married	In **cluster 1**, the most frequently occurring value is ‘0’, which represents ‘No’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘0’ (i.e., No) appears in cluster 1 is 1755 out of a total of 1755 instances. This cluster predominantly contains individuals who are not married which indicates that the cluster is well-defined for the ‘ever_married’ feature.
	In **cluster 2**, the most frequently occurring value is ‘1’, which represents ‘Yes’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘1’ (i.e., Yes) appears in cluster 2 is 3353 out of a total of 3355 instances(only two individuals belong to the non-married class). This cluster predominantly contains individuals who are married which indicates that the cluster is well-defined for the ‘ever_married’ feature.
gender	In **cluster 1**, the most frequently occurring value is ‘0’, which represents Female (refer to section 2.2 for the categorical names associated with each label-encoded category). The value ‘0’ (i.e., Female) appears 992 times out of a total of 1755 instances. The other category, ‘1’ (Male), appears 762 times, while there is a single instance of ‘2’ (Other). This cluster is a mixture of the three categories (992 instances of ‘0’, 762 instances of ‘1,’ and 1 instance of ‘2’), indicating that the cluster is not well-defined for the ‘gender’ feature.
	In **cluster 2**, the most frequently occurring value is ‘0’, which represents Female (refer to section

	2.2 for the categorical names associated with each label-encoded category). The value ‘0’ (i.e., Female) appears 2202 times out of a total of 3355 instances. The other category, ‘1’ (Male), appears 1353 times. This cluster is a mixture of the two categories (2202 instances of ‘0’, 1353 instances of ‘1’), indicating that the cluster is not well-defined for the ‘gender’ feature.
heart_disease	In **cluster 1**, the most frequently occurring value is ‘0’, which represents ‘no heart disease’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 1 is 1724 out of a total of 1755 instances. Only 31 instances of ‘1’ (individuals with heart disease) occurs in this cluster. This cluster predominantly contains individuals without heart disease.
	In **cluster 2**, the most frequently occurring value is ‘0’, which represents ‘no heart disease’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 2 is 3110 out of a total of 3355 instances. There are 245 instances of ‘1’ (individuals with heart disease) occurs in this cluster. The majority of individuals with heart disease belong to this cluster as compared to cluster 1.
hypertension	In **cluster 1**, the most frequently occurring value is ‘0’, which represents ‘no hypertension’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 1 is 1704 out of a total of 1755 instances. There are 51 instances of ‘1’ (individuals with hypertension) occurs in this cluster. This cluster predominantly contains individuals without hypertension.
	In **cluster 2**, the most frequently occurring value is ‘0’, which represents ‘no hypertension’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 2 is 2908 out of a total of 3355 instances. There are 447 instances of ‘1’ (individuals with hypertension) in this cluster. The majority of individuals with hypertension belong to this cluster as compared to cluster 1.

smoking_status	In **cluster 1**, the most frequently occurring value is ‘0’, which represents ‘Unknown’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘0’ appears in cluster 1 is 901 out of a total of 1755 instances. There are 528 instances of ‘2’(never smoked), 179 instances of ‘3’(smokes) and 147 instances of ‘1’ (formerly smoked).
	In **cluster 2**, the most frequently occurring value is ‘2’, which represents ‘never smoked’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘2’ appears in cluster 2 is 1364 out of a total of 3355 instances. There are 643 instances of ‘0’(unknown), 610 instances of ‘3’(smokes) and 738 instances of ‘1’ (formerly smoked).
stroke	In **cluster 1**, the most frequently occurring value is ‘0’, which represents ‘no stroke’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 1 is 1728 out of a total of 1755 instances. Only 27 instances of ‘1’ (individuals who reported stroke) occur in this cluster. This cluster predominantly contains individuals who didn’t report stroke.
	In **cluster 2**, the most frequently occurring value is ‘0’, which represents ‘no stroke’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 2 is 3133 out of a total of 3355 instances. There are 222 instances of ‘1’ (individuals who reported stroke) in this cluster. The majority of individuals who reported stroke belong to this cluster as compared to cluster 1.
work_type	In **cluster 1**, the most frequently occurring value is ‘2’, which represents ‘Private’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘2’ appears in cluster 1 is 820 out of a total of 1755 instances. There 687 instances of ‘4’(children), 116 instances of ‘0’(Govt_job), 110 instances of ‘3’(self-employed) and 22 instances of ‘1’(Never_worked)

	In **cluster 2**, the most frequently occurring value is ‘2’, which represents ‘Private’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘2’ appears in cluster 2 is 2105 out of a total of 3355 instances. There are 709 instances of ‘3’ (self-employed) and 541 instances of ‘0’ (Govt_job).

Best Features for KMeans Clustering

From Tables 5.1 (iii) and 5.1 (iv), it is observed that the most suitable features for K-Means Clustering are: ‘age,’ ‘avg_glucose_level,’ ‘bmi,’ ‘ever_married,’ ‘heart_disease,’ ‘hypertension,’ and ‘stroke.’ These features are well-defined between the two clusters and are well-separated and distinguished.

5.2 KMedoids Clustering (EXTRA)

- The KMedoids clustering algorithm is first initialised with two clusters.
- The model is trained on df_clust_norm1 (a normalised dataframe using MinMaxScaler() which contains all the features)
- After modelling, the cluster centers are viewed.
- The silhouette score is computed for the clustering.
- The elbow curve is plotted to determine the optimum number of clusters for better clustering (point appeared at k=2; suggesting 2 as the optimum number of clusters)
- A new column named ‘cluster’ is appended to the ‘df_clust_norm1’ to store the cluster labels assigned to each data point by the KMedoids algorithm.
- Properties of both the clusters are viewed.
- For both the cluster, medoid values of all attributes are computed from the cluster centers.
- A bar chart is plotted to analyse the medoid values for the clusters.
- Statistical summary of the two clusters are analysed.

Results:

***Fig-5.2 (A)-* Elbow Curve**

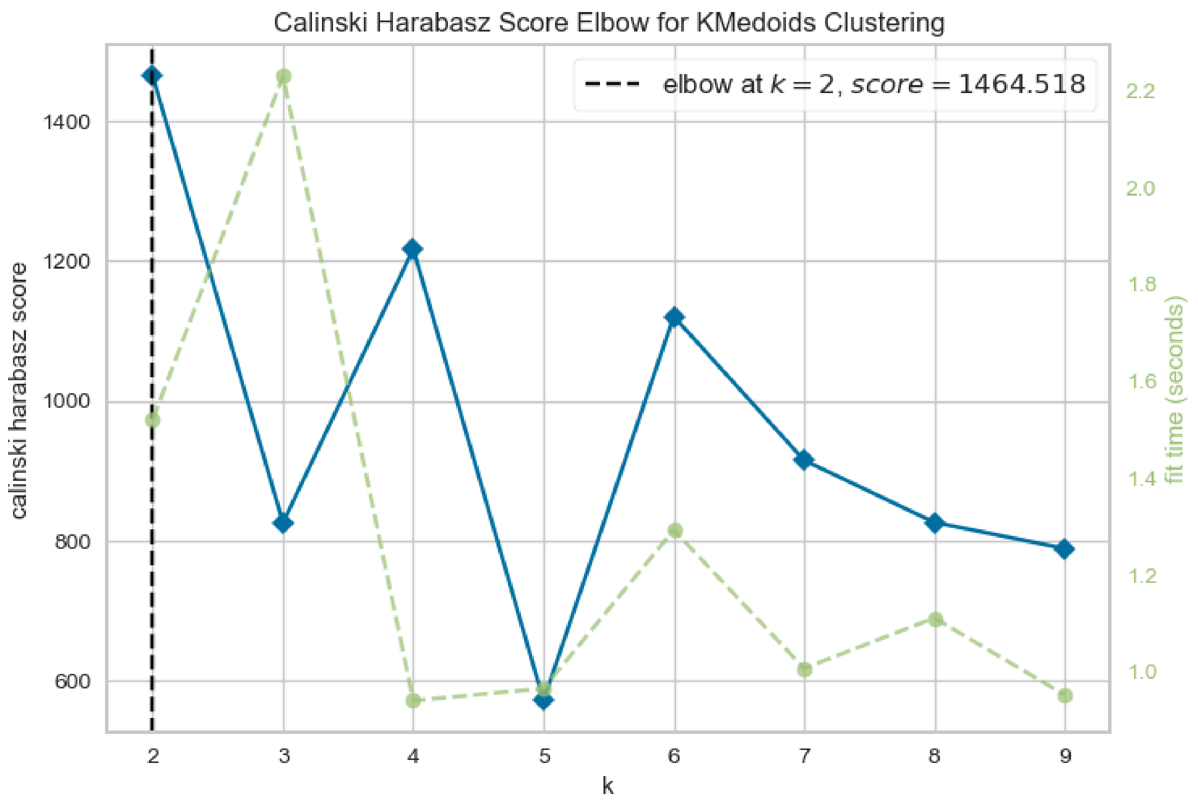
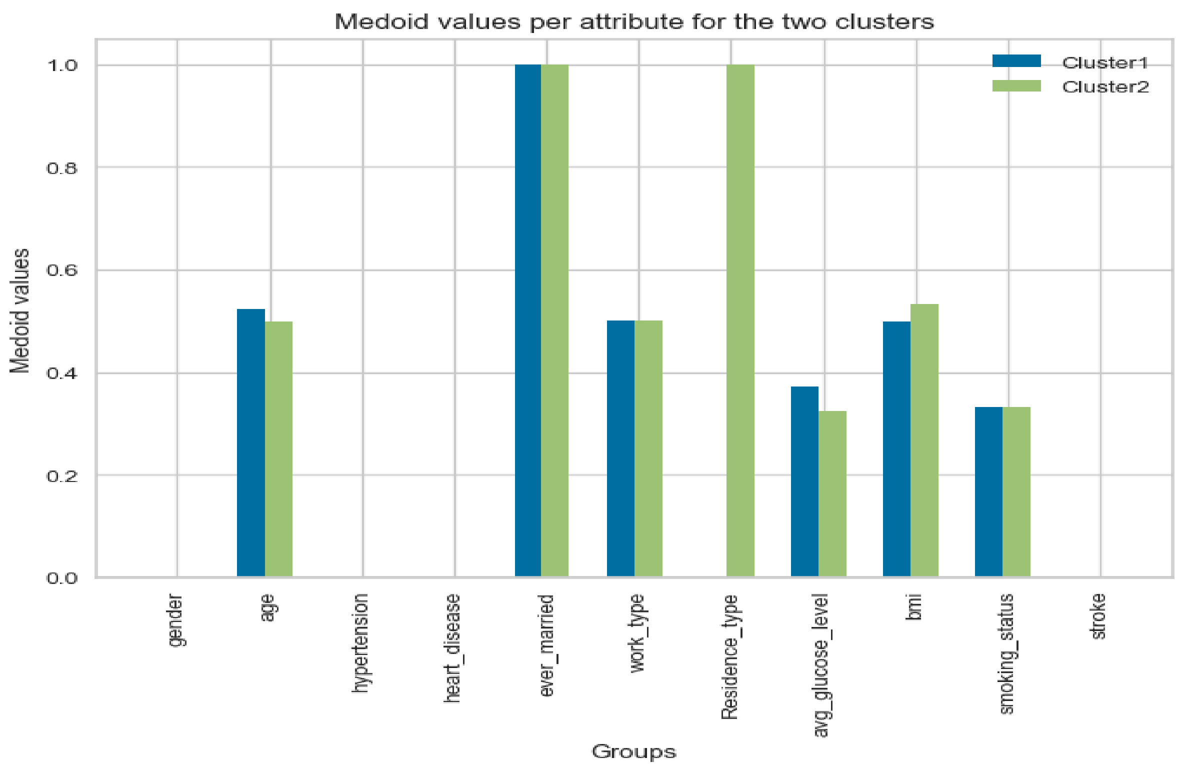


Fig-5.2 (B)- **Medoid values per Attribute for the two Clusters**


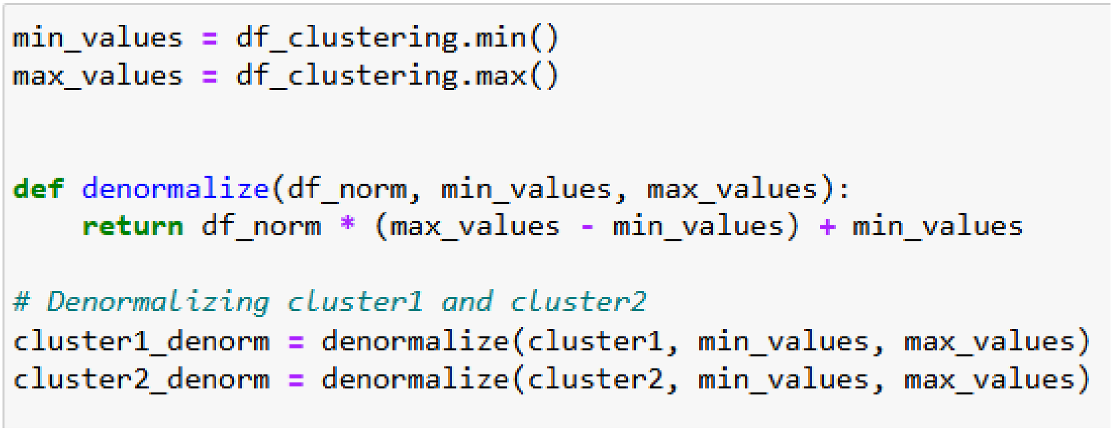
Interpretation and Analysis of the above Results:

Result	**Interpretation**
Elbow Curve (Refer Fig- 5.2 (A))	The elbow curve suggests that the optimal number of clusters for better clustering performance is 2 with the highest Calinski Harabasz score of 1464.518; this point is the elbow point. This score suggests the best trade-off between compactness of the cluster (how tightly packed the points are in each cluster) and the separation of the clusters (how distinct the clusters are from each other). The elbow point suggests that adding more clusters beyond 2 doesn’t significantly improve the clustering quality.
Visualisation of mean values per attribute for the two clusters (Refer Fig-5.2 B))	We can see that some attributes show a significant difference in medoid values between the two clusters which indicates better clustering and more distinct separation between the groups.The only feature that shows significant difference in the medoid values between the two clusters is ‘Residence_type’ attribute whereas

	others show minimal or no difference.

Denormalization and Conversion of specific attributes to Categorical

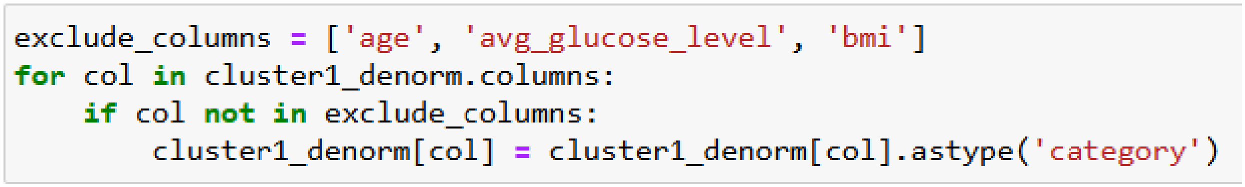
Fig- 5.2 (C) Denormalization



```python
min_values = df_clustering.min()
max_values = df_clustering.max()


def denormalize(df_norm, min_values, max_values):
    return df_norm * (max_values - min_values) + min_values

# Denormalizing cluster1 and cluster2
cluster1_denorm = denormalize(cluster1, min_values, max_values)
cluster2_denorm = denormalize(cluster2, min_values, max_values)
```

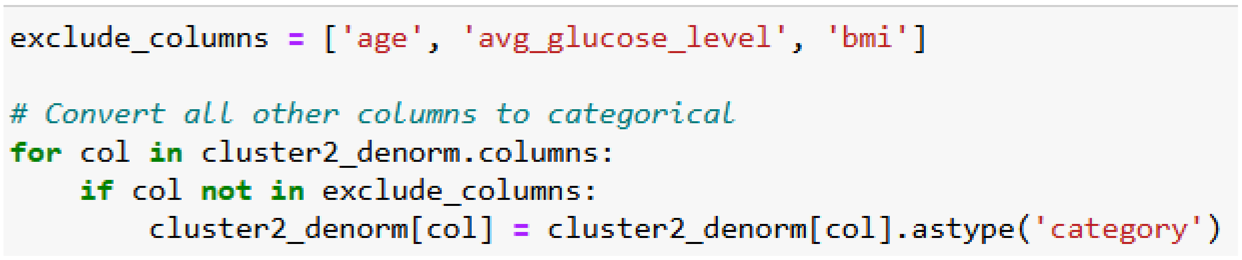
Fig 5.2 (D) Conversion to Categorical

Conversion of Cluster 1 Features



```python
exclude_columns = ['age', 'avg_glucose_level', 'bmi']
for col in cluster1_denorm.columns:
    if col not in exclude_columns:
        cluster1_denorm[col] = cluster1_denorm[col].astype('category')
```

Conversion of Cluster 2 Features



```python
exclude_columns = ['age', 'avg_glucose_level', 'bmi']

# Convert all other columns to categorical
for col in cluster2_denorm.columns:
    if col not in exclude_columns:
        cluster2_denorm[col] = cluster2_denorm[col].astype('category')
```

Statistical Summary of the two Clusters:

Table 5.2 (i)- Statistical Summary of the Numerical Attributes

Cluster 1 (Numerical Attributes)

	age	avg_glucose_level	bmi
count	2514.000000	2514.000000	2514.000000
mean	42.900811	101.200759	28.743369
std	22.462089	33.048070	7.153297
min	0.080000	55.120000	10.300000
25%	25.000000	77.420000	23.800000
50%	44.000000	92.955000	28.300000
75%	61.000000	114.515000	32.567059
max	82.000000	169.357500	47.500000

Cluster 2 (Numerical Attributes)

	age	avg_glucose_level	bmi
count	2596.000000	2596.000000	2596.000000
mean	43.542126	100.798111	28.793833
std	22.757380	33.380519	7.291051
min	0.080000	55.220000	11.300000
25%	26.000000	77.042500	23.600000
50%	45.000000	90.770000	28.200000
75%	61.000000	113.767500	33.100000
max	82.000000	169.357500	47.500000

Table 5.2 (ii)- Statistical Summary of the Categorical Features

Cluster 1 (Categorical Attributes)

	Residence_type	cluster	ever_married	gender	heart_disease	hypertension	smoking_status	stroke	work_type
count	2514.0	0	2514.0	2514.0	2514.0	2514.0	2514.0	2514.0	2514.0
unique	1.0	0	2.0	3.0	2.0	2.0	4.0	2.0	5.0
top	0.0	NaN	1.0	0.0	0.0	0.0	2.0	0.0	2.0
freq	2514.0	NaN	1642.0	1465.0	2380.0	2263.0	961.0	2400.0	1462.0

Cluster 2 (Categorical Attributes)

	Residence_type	cluster	ever_married	gender	heart_disease	hypertension	smoking_status	stroke	work_type
count	2596.0	0	2596.0	2596.0	2596.0	2596.0	2596.0	2596.0	2596.0
unique	1.0	0	2.0	2.0	2.0	2.0	4.0	2.0	5.0
top	1.0	NaN	1.0	0.0	0.0	0.0	2.0	0.0	2.0
freq	2596.0	NaN	1711.0	1529.0	2454.0	2349.0	931.0	2461.0	1463.0

Analysis of the Statistical Summary of the Numerical Attributes for the two Clusters (Refer Table 5.2 (i))

Table- 5.2 (iii)	
Features	**Analysis**
age	In **cluster 1**, the average age of individuals is 42.9 years and the median (50%) age is 44 years which means 50% of the individuals in cluster 1 are 44 years or less.
	In **cluster 2**, the average age of individuals is 43.54 years and the median (50%) age is 45 years which means 50% of the individuals in cluster 2 are 45 years or less.
avg_glucose_level	In **cluster 1**, the mean of avg_glucose_level is 101.2 units and the median (50%) glucose level is 92.95 units which means 50% of the individuals in cluster 1 have a glucose level of 92.95 units or less.
	In **cluster 2**, the mean of avg_glucose_level is 100.79 units and the median (50%) glucose level is 90.77 units which means 50% of the individuals in cluster 2 have a glucose level of 90.77 units or less.
bmi	In **cluster 1**, the mean bmi is 28.74 units and the median (50%) bmi is 28.3 units which means 50% of the individuals in cluster 1 have a bmi value of 28.3 units or less.
	In **cluster 2**, the mean bmi is 28.79 units and the median (50%) bmi is 28.2 units which means 50% of the individuals in cluster 2 have a bmi value of 28.2 units or less.

Analysis of the Statistical Summary of the Categorical Attributes for the two Clusters (Refer Table 5.2 (ii)

Table 5.2 (iv)

Features	**Analysis**
Residence_type	In **cluster 1**, the most frequently occurring value is ‘0’, which represents Rural (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘0’ (i.e., Rural) appears in cluster 1 is 2514 out of a total of 2514 instances making this cluster well-defined for this feature. This cluster contains individuals whose residence type is rural.
	In **cluster 2**, the most frequently occurring value is ‘1’, which represents Urban (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘1’ (i.e., urban) appears in cluster 2 is 2596 out of a total of 2596 instances, making this cluster well-defined for this feature. This cluster contains all the individuals whose residence type is urban.
ever_married	In **cluster 1**, the most frequently occurring value is ‘1’, which represents ‘Yes’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘1’ (i.e., Yes) appears in cluster 1 is 1642 out of a total of 2514 instances. The number of times the other category appears which is ‘0’ (‘No’) is 872.
	In **cluster 2**, the most frequently occurring value is ‘1’, which represents ‘Yes’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘1’ (i.e., Yes) appears in cluster 2 is 1711 out of a total of 2596 instances. The number of times the other category appears which is ‘0’ (‘No’) is 885.
gender	In **cluster 1**, the most frequently occurring value is ‘0’, which represents Female (refer to section 2.2 for the categorical names associated with each label-encoded category). The value ‘0’ (i.e., Female) appears 1465 times out of a total of 2514

	instances. The other category, ‘1’ (Male), appears 1048 times, while there is a single instance of ‘2’ (Other).
	In **cluster 2**, the most frequently occurring value is ‘0’, which represents Female (refer to section 2.2 for the categorical names associated with each label-encoded category). The value ‘0’ (i.e., Female) appears 1529 times out of a total of 2596 instances. The other category, ‘1’ (Male), appears 1067 times.
heart_disease	In **cluster 1**, the most frequently occurring value is ‘0’, which represents ‘no heart disease’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 1 is 2380 out of a total of 2514 instances. There are 134 instances of ‘1’ (individuals with heart disease) in this cluster.
	In **cluster 2**, the most frequently occurring value is ‘0’, which represents ‘no heart disease’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 2 is 2454 out of a total of 2596 instances. There are 142 instances of ‘1’ (individuals with heart disease) in this cluster.
hypertension	In **cluster 1**, the most frequently occurring value is ‘0’, which represents ‘no hypertension’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 1 is 2263 out of a total of 2514 instances. There are 251 instances of ‘1’ (individuals with hypertension) in this cluster.
	In **cluster 2**, the most frequently occurring value is ‘0’, which represents ‘no hypertension’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 2 is 2349 out of a total of 2596 instances. There are 247 instances of ‘1’ (individuals with hypertension) in this cluster.
smoking_status	In **cluster 1**, the most frequently occurring value is ‘2’, which represents ‘never smoked’ (refer to section 2.2 for the categorical names associated

	with each label-encoded category). The number of times ‘2’ appears in cluster 1 is 961 out of a total of 2514 instances. There are 762 instances of ‘0’(Unknown), 363 instances of ‘3’(smokes) and 428 instances of ‘1’ (formerly smoked).
	In **cluster 2**, the most frequently occurring value is ‘2’, which represents ‘never smoked’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘2’ appears in cluster 2 is 931 out of a total of 2596 instances. There are 782 instances of ‘0’(Unknown), 426 instances of ‘3’(smokes) and 457 instances of ‘1’ (formerly smoked).
stroke	In **cluster 1**, the most frequently occurring value is ‘0’, which represents ‘no stroke’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 1 is 2400 out of a total of 2514 instances. There are 114 instances of ‘1’ (individuals who reported stroke) in this cluster.
	In **cluster 2**, the most frequently occurring value is ‘0’, which represents ‘no stroke’ (as per the data dictionary provided). The number of times ‘0’ appears in cluster 2 is 931 out of a total of 2596 instances. There are 135 instances of ‘1’ (individuals who reported stroke) in this cluster.
work_type	In **cluster 1**, the most frequently occurring value is ‘2’, which represents ‘Private’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘2’ appears in cluster 1 is 1462 out of a total of 2514 instances. There 340 instances of ‘4’(children), 312 instances of ‘0’(Govt_job), 393 instances of ‘3’(self-employed) and 7 instances of ‘1’(Never_worked)
	In **cluster 2**, the most frequently occurring value is ‘2’, which represents ‘Private’ (refer to section 2.2 for the categorical names associated with each label-encoded category). The number of times ‘2’ appears in cluster 2 is 1463 out of a total of 2596 instances. There are 462 instances of ‘3’ (self-employed), 345 instances of ‘0’ (Govt_job), 347 instances of ‘4’ (children) and

	15 instances of ‘1’ (Never_worked’)

Best Features for KMedoids Clustering

From Tables 5.2 (iii) and 5.2 (iv), it is observed that the most significant feature for K-Medoids Clustering is ‘Residence_type’. The remaining features appear to be randomly assigned to one of the two clusters, with minimal or no distinguishable separation between the clusters.

5.3 Comparision and Analysis of the Clustering Models

Table 5.3		
Clustering Model	**Silhouette Score**	**Analysis**
K-Means	0.27	K-Means clustering outperforms K-Medoids with a higher Silhouette score. Most features are well-defined, indicating better separation between clusters and more distinguishable features within each cluster.
K-Medoids	0.23	K-Medoids clustering performs below K-Means, with many points randomly assigned. The lower Silhouette score suggests poorer separation between clusters and less distinguishable features.

References

1. imbalanced-learn. (n.d.). *imblearn.over_sampling.SMOTE*. Retrieved January 4, 2025, from https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
2. Seaborn developers. (n.d.). *seaborn.violinplot*. Retrieved January 4, 2025, from https://seaborn.pydata.org/generated/seaborn.violinplot.html
3. scikit-learn developers. (n.d.). *sklearn.ensemble.AdaBoostClassifier*. Retrieved January 4, 2025, from https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
4. scikit-learn developers. (n.d.). *sklearn.ensemble.GradientBoostingRegressor*. Retrieved January 4,2025,fromhttps://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html
5. scikit-learn developers. (n.d.). *scikit-learn: Machine learning in Python*. Retrieved January 4, 2025, from https://scikit-learn.org/stable/
6. JavaTpoint. (n.d.). *K-medoids clustering: Theoretical explanation*. JavaTpoint. Retrieved January 4, 2025, from https://www.javatpoint.com/k-medoids-clustering-theoretical-explanation