Assessing the Risk Factors Associated with Stroke

Bhavana Bethi[1*], Keerthika Sunchu[1], Pallavi Telu[1], Srija Dammanagari[1], Yazna Penmetsa[1]

[1]Indiana University Purdue University Indianapolis, IN 46202, USA
{bbethi, ksunchu, ptelu, srdamma, ypenmets} @iu.edu

Abstract: Our research, conducted at Indiana University Purdue University Indianapolis, is dedicated to investigating the risk factors associated with stroke. We utilized data analysis and machine learning techniques, making use of the Behavioral Risk Factor Surveillance System (BRFSS) survey dataset.From the results, we observed that there is a higher prevalence of smoking, hypertension & heart disease among individuals who have experienced a stroke. Our project's goal is to offer valuable insights for healthcare professionals, policymakers, and public health initiatives, ultimately contributing to the reduction of the burden of this disease and facilitating improved risk prediction and management.

Keywords: Hypertension, Diabetes, Stroke, Risk factors, data visualization, data analysis.

1 Project Scope

1.1 Introduction

Stroke poses a significant public health challenge, resulting in higher rates of illness and death. Key factors contributing to stroke risk that can be modified through lifestyle changes include hypertension, diabetes, atrial fibrillation, dyslipidemia, smoking, and alcohol misuse. Among these factors, diabetes and hypertension are rapidly expanding health concerns, playing a substantial role in the rising incidence of cardiovascular disease and stroke (Volkow et al., 2017). In the 2015 Behavioral Risk Factor Surveillance System (BRFSS) survey dataset, which comprises 70,692 meticulously processed and balanced responses, our project focuses on predicting the likelihood of stroke. The BRFSS survey serves as a valuable resource for understanding the demographic, lifestyle, and health-related factors contributing to the prevalence of stroke within the United States. This dataset represents a valuable source of information, encompassing a wide range of variables, including age, gender, BMI, smoking habits, physical activity, dietary choices, cholesterol level, and

personal and family medical histories (Diabetes, Hypertension, and Stroke Prediction, 2022). One of the important studies from 2018 conducted by Alloubani and their team has significantly contributed to our understanding of how high blood pressure (hypertension), diabetes, and strokes are interconnected. Furthermore, they did extensive research on these health issues and uncovered a wealth of valuable information. This aligns perfectly with the focus of our project (Alloubani et al., 2018).
We utilized machine learning and data analysis to enhance our comprehension of these health challenges. Our objective is to offer valuable insights that can be of benefit not just to healthcare practitioners but also to guide health policy choices and enable individuals to effectively handle these health risks.

1.2 Aim

This project aims to investigate the impact of age, sex, BMI, smoking status, average glucose levels, & heart disease on the risk of developing stroke and hypertension in individuals. By analyzing comprehensive health data, we aim to determine if these demographic and lifestyle factors are significant predictors of these conditions.

Our Research Hypothesis include-

Null hypothesis (H0) - The examined factors (sex, age, hypertension, heart disease, glucose level, BMI, smoking) are not associated with the risk of stroke.

Alternative hypothesis (H1) - The examined factors (sex, age, hypertension, heart disease, glucose level, BMI, smoking) are associated with the risk of stroke.

1.3 Purpose

This research project aims to investigate the influence of various factors on stroke occurrence. It covers variables like age, sex, BMI, smoking, heart conditions & average glucose levels. Through Data Analysis & machine learning, we seek to reveal insights into the relationships between these factors and health conditions. This multifaceted approach will enable us to provide a comprehensive view of measures of central tendency, dispersion, and frequency, and expose the dynamics of these health conditions within the population. This knowledge will serve to inform healthcare providers, policymakers, and public health initiatives in developing tailored interventions for at-risk populations, ultimately reducing the burden of these diseases on individuals and society.

2 Methodology

2.1 Study type – Descriptive & Predictive Statistical Study.

2.2 Steps of our project

The tools we have used for our project are Python Jupyter notebook and phpMyAdmin. Our methodology involves-

1. Data Collection & storage
2. Data Cleaning & Preprocessing
3. Data Analysis
    - Descriptive Statistics
    - Exploratory Data Analysis
    - Data Visualization
    - Statistical Analysis
    - Machine Learning and evaluation metrics

2.3 Team Members Responsibilities

Our team comprises members with varied skills and backgrounds. Initially, we established a list of team members for the project, along with the assigned responsibilities that we collectively agreed upon.

Table 1. Team Member's Responsibilities

Name	**Background**	**Responsibilities**
Bhavana Bethi	Doctor of Pharmacy	Project Draft, Data collection, Data extraction.
Keerthika	Bachelor of Pharmacy	Data storage and analysis using SQL.
Pallavi Telu	Bachelor of Technology in Biotechnology	Machine Learning
Srija	Bachelor of Dental Surgery	Statistical analysis
Yazna	Doctor of Pharmacy	Data Visualization

2.4 Actual Contributions of Team members

Table 2. Actual Responsibilities of Team Members

Name	Background	Responsibilities
Bhavana Bethi	Doctor of Pharmacy	Project Proposal, Data collection, Statistical Analysis, Project Presentation & Report
Keerthika	Bachelor of Pharmacy	Data storage and Data Cleaning, Project Presentation & Report
Pallavi Telu	Bachelor of Technology in Biotechnology	Machine Learning Integration, Project Presentation & Report
Srija	Bachelor of Dental Surgery	Exploratory Data Analysis & Data Visualization, Project Presentation & Report
Yazna	Doctor of Pharmacy	Descriptive Statistics & Data Visualization, Project Presentation & Report

2.5 Project Challenges

The process of data cleaning proved to be both time-consuming and challenging. We encountered the need for an additional round of cleaning when addressing negative values in the age column, prompting their removal.

3 Data Collection & Description

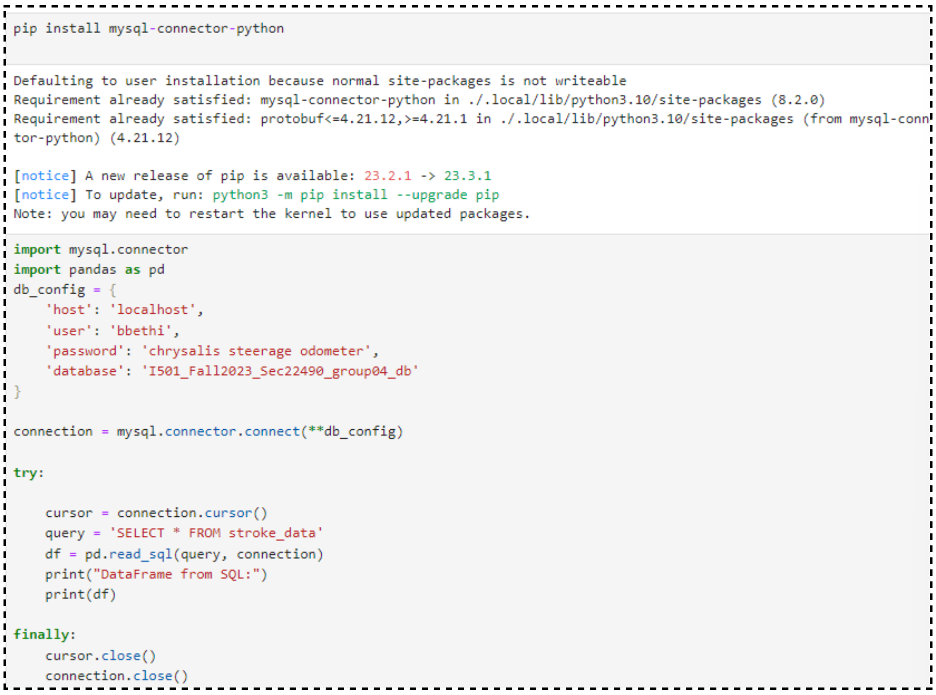
The dataset comprises responses collected through the Behavioral Risk Factor Surveillance System (BRFSS) survey.
source - **https://www.kaggle.com/datasets/prosperchuks/health-dataset/code**
The dataset comprises three distinct files, and we selected the Stoke dataset for analysis.. It consists of
Stroke data.csv
Diabetes data.csv
Hypertension data.csv

Table 3. Classification of Data

Categorical Data		Numerical Data
Binary	**Ordinal**	**Continuous**
Gender, Hypertension, Smoking status, heart disease, stroke, Ever married	Working Status	Average Glucose levels, BMI, Age.

4. Data Storage

The data has been stored in an SQL database.



```
pip install mysql-connector-python
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: mysql-connector-python in ./.local/lib/python3.10/site-packages (8.2.0)
Requirement already satisfied: protobuf<=4.21.12,>=4.21.1 in ./.local/lib/python3.10/site-packages (from mysql-conn
tor-python) (4.21.12)

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python3 -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

```python
import mysql.connector
import pandas as pd
db_config = {
    'host': 'localhost',
    'user': 'bbethi',
    'password': 'chrysalis steerage odometer',
    'database': 'I501_Fall2023_Sec22490_group04_db'
}

connection = mysql.connector.connect(**db_config)

try:

    cursor = connection.cursor()
    query = 'SELECT * FROM stroke_data'
    df = pd.read_sql(query, connection)
    print("DataFrame from SQL:")
    print(df)

finally:
    cursor.close()
    connection.close()
```

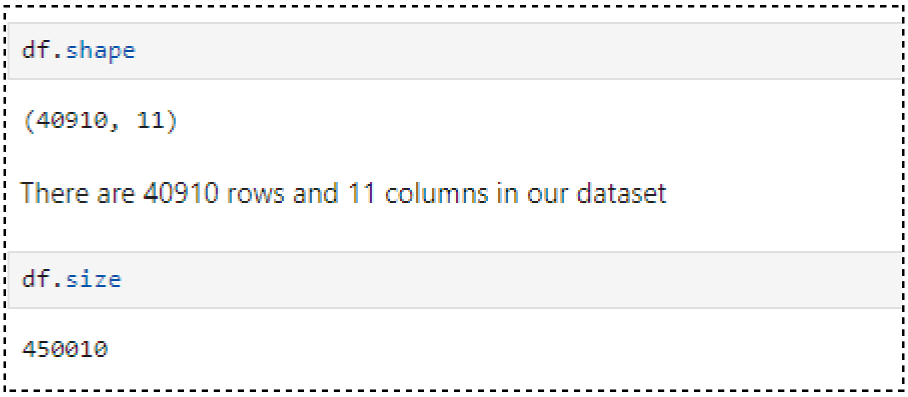
Fig. 1 Python Code for connecting SQL

5. Data Cleaning

The data cleaning process began by loading the dataset into a Pandas DataFrame in Python. After importing the data, We checked the shape & size of the dataframe. There were 40,910 rows and 11 columns. We checked for null values to identify missing data. This process revealed some gender values were missing, so we imputed them with the mode, which was 1 which in our case implies male gender. Beyond filling missing data, additional cleaning entailed replacing numeric columns like BMI with categorical data and dropping invalid values to prepare the dataset. We converted binary values into categorical values for visualization.

```
import pandas as pd
import numpy as np
```

Fig. 2 Python Code for importing Pandas



```
df.shape
```

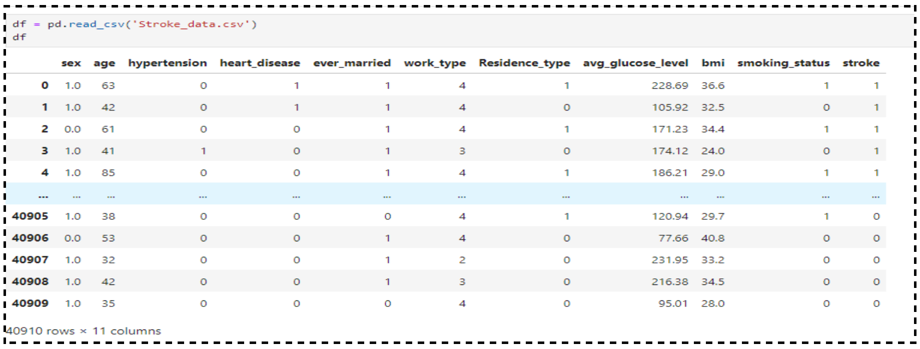
```
(40910, 11)
```

There are 40910 rows and 11 columns in our dataset

```
df.size
```

```
450010
```

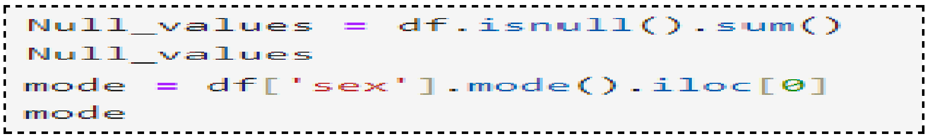
Fig. 3 Python Code for checking number of rows & columns in our data



```
df = pd.read_csv('Stroke_data.csv')
df
```

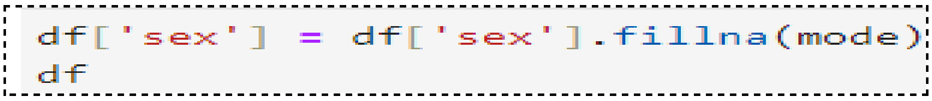
```
       sex  age  hypertension  heart_disease  ever_married  work_type  Residence_type  avg_glucose_level   bmi  smoking_status  stroke
    0  1.0   63             0              1             1          4               1             228.69  36.6               1       1
    1  1.0   42             0              1             1          4               0             105.92  32.5               0       1
    2  0.0   61             0              0             1          4               1             171.23  34.4               1       1
    3  1.0   41             1              0             1          3               0             174.12  24.0               0       1
    4  1.0   85             0              0             1          4               1             186.21  29.0               1       1
  ...  ...  ...           ...            ...           ...        ...             ...                ...   ...             ...     ...
40905  1.0   38             0              0             0          4               1             120.94  29.7               1       0
40906  0.0   53             0              0             1          4               0              77.66  40.8               0       0
40907  1.0   32             0              0             1          2               0             231.95  33.2               0       0
40908  1.0   42             0              0             1          3               0             216.38  34.5               0       0
40909  1.0   35             0              0             0          4               0              95.01  28.0               0       0

40910 rows × 11 columns
```

Fig. 4 Python Code for reading the CSV file



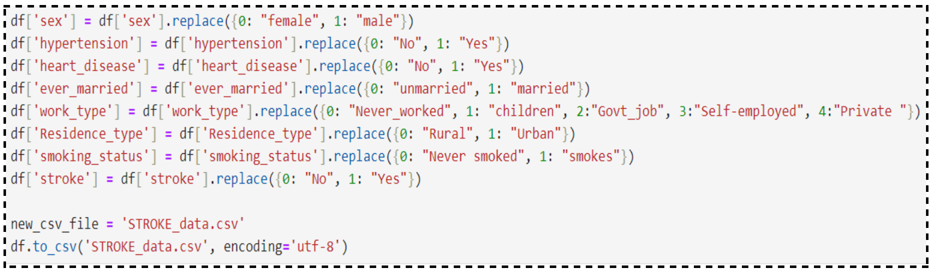
```
Null_values = df.isnull().sum()
Null_values
mode = df['sex'].mode().iloc[0]
mode
```

Fig. 5 Python Code for checking number null values



```
df['sex'] = df['sex'].fillna(mode)
df
```

Fig. 6 Python Code for replacing null values with mode i.e., males



```
df['sex'] = df['sex'].replace({0: "female", 1: "male"})
df['hypertension'] = df['hypertension'].replace({0: "No", 1: "Yes"})
df['heart_disease'] = df['heart_disease'].replace({0: "No", 1: "Yes"})
df['ever_married'] = df['ever_married'].replace({0: "unmarried", 1: "married"})
df['work_type'] = df['work_type'].replace({0: "Never_worked", 1: "children", 2:"Govt_job", 3:"Self-employed", 4:"Private "})
df['Residence_type'] = df['Residence_type'].replace({0: "Rural", 1: "Urban"})
df['smoking_status'] = df['smoking_status'].replace({0: "Never smoked", 1: "smokes"})
df['stroke'] = df['stroke'].replace({0: "No", 1: "Yes"})

new_csv_file = 'STROKE_data.csv'
df.to_csv('STROKE_data.csv', encoding='utf-8')
```

Fig. 7 Python Code for converting binary values into categorical values

5.1 Detection of outliers

A key aspect of the cleaning involved outlier detection. We first visualized a box plot to spot potential outliers and they were only detected in the BMI column. This graphical analysis then informed a mathematical approach of defining BMI outliers quantitatively as values below the 5th or above the 95th percentile. With outliers defined, the final step capped outliers through a process called winsorization without fully removing these legitimate, but extreme data points. By clipping the most extreme BMI values, the cleaning process limited distortion while retaining informative data. Together the missing value imputation, categorical encoding, invalid data removal and outlier capping yielded a cleaned dataset ready for exploratory analysis and modelling.

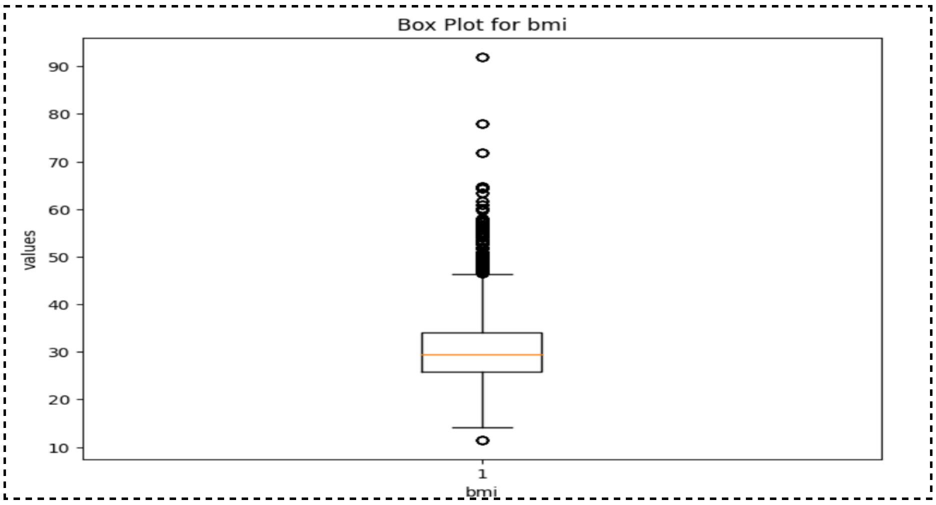
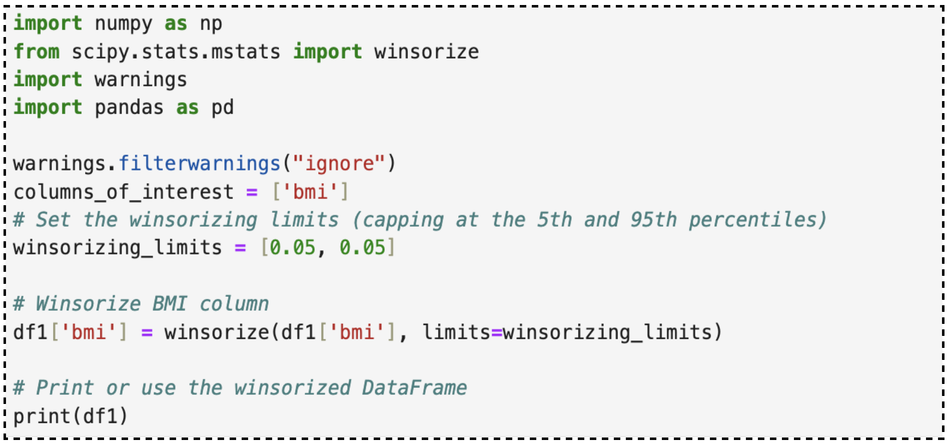


Fig. 8 Checking outliers for BMI column using Boxplot



```
import numpy as np
from scipy.stats.mstats import winsorize
import warnings
import pandas as pd

warnings.filterwarnings("ignore")
columns_of_interest = ['bmi']
# Set the winsorizing limits (capping at the 5th and 95th percentiles)
winsorizing_limits = [0.05, 0.05]

# Winsorize BMI column
df1['bmi'] = winsorize(df1['bmi'], limits=winsorizing_limits)

# Print or use the winsorized DataFrame
print(df1)
```

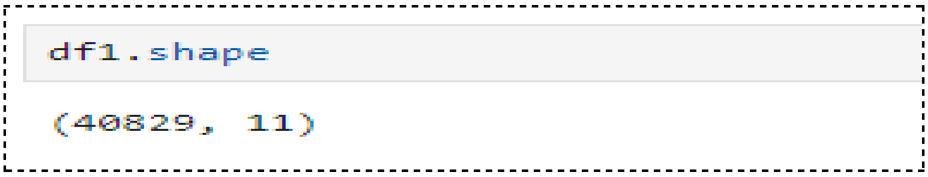
Fig. 9 Python Code for replacing Outliers

6 Data analysis

Python was utilized for data analysis, involving descriptive statistics, Exploratory data analysis, data visualization, the application of chi-square tests and classification models to analyze the data.

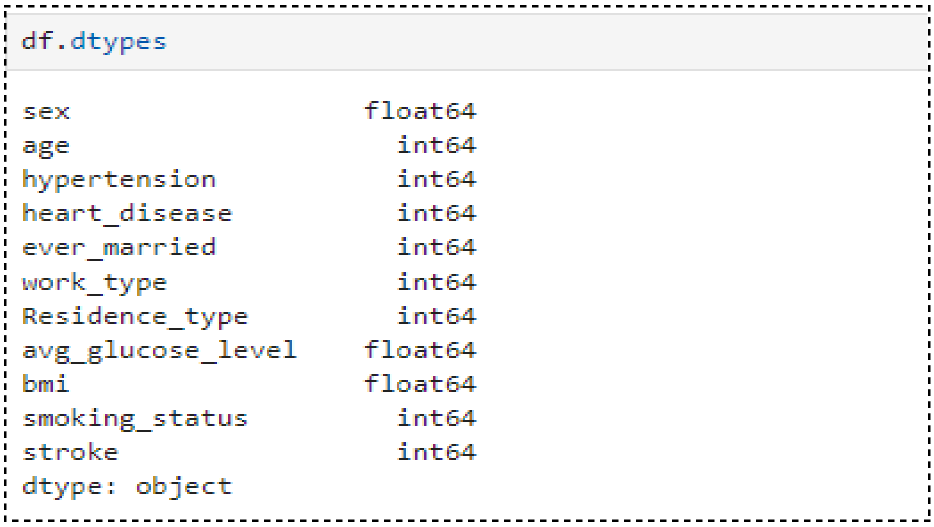
6.1 Descriptive Statistics

There are 40829 rows & 11 columns after data cleaning. 7 columns classified as 'object' type and 4 columns designated as 'float64'. Next, we computed the statistical metrics, including the mean, standard deviation, high value of the variable, and low value of the variable. we then visualized the data using barcharts, pie charts & histograms. Distribution of gender bar graph shows that 22,669 are male, and 18,182 are female.Distribution of Smoking Status shows that 48.9% are smokers. The histogram shows that majority of people have bmi between 25 and 30



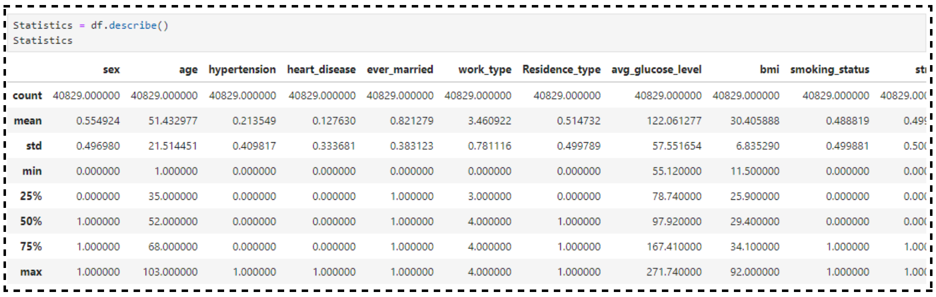
```
df1.shape

(40829, 11)
```

Fig. 10 Python Code for checking number of rows & columns after data cleaning.



```
df.dtypes

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

Fig. 11 Python Code for Verifying Data Types



```python
Statistics = df.describe()
Statistics
```

	sex	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	st
count	40829.000000	40829.000000	40829.000000	40829.000000	40829.000000	40829.000000	40829.000000	40829.000000	40829.000000	40829.000000	40829.00
mean	0.554924	51.432977	0.213549	0.127630	0.821279	3.460922	0.514732	122.061277	30.405888	0.488819	0.49
std	0.496980	21.514451	0.409817	0.333681	0.383123	0.781116	0.499789	57.551654	6.835290	0.499881	0.50
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	55.120000	11.500000	0.000000	0.00
25%	0.000000	35.000000	0.000000	0.000000	1.000000	3.000000	0.000000	78.740000	25.900000	0.000000	0.00
50%	1.000000	52.000000	0.000000	0.000000	1.000000	4.000000	1.000000	97.920000	29.400000	0.000000	0.00
75%	1.000000	68.000000	0.000000	0.000000	1.000000	4.000000	1.000000	167.410000	34.100000	1.000000	1.00
max	1.000000	103.000000	1.000000	1.000000	1.000000	4.000000	1.000000	271.740000	92.000000	1.000000	1.00

Fig. 12 Python Code for Generating Descriptive Statistics

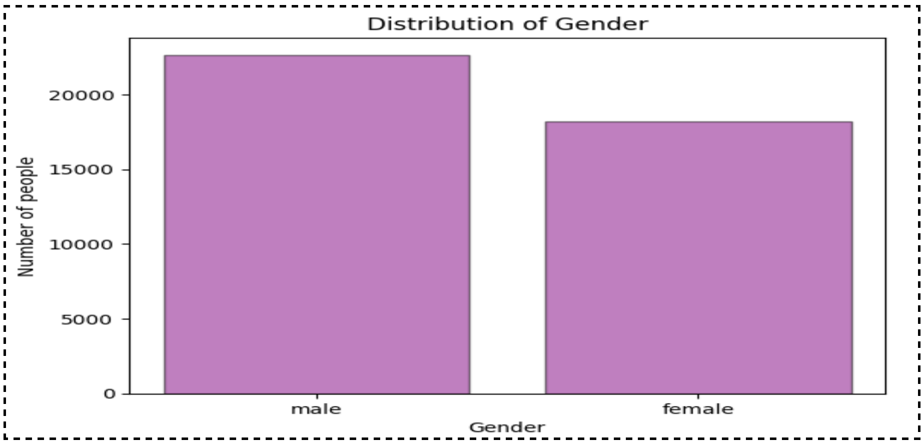


Fig. 13 Bar chart for gender distribution

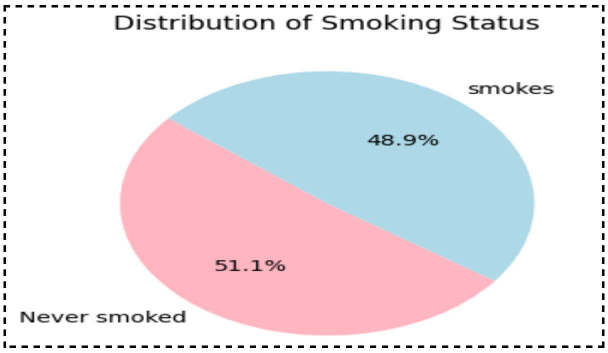


Fig. 14 Pie chart for distribution of Smoking Status

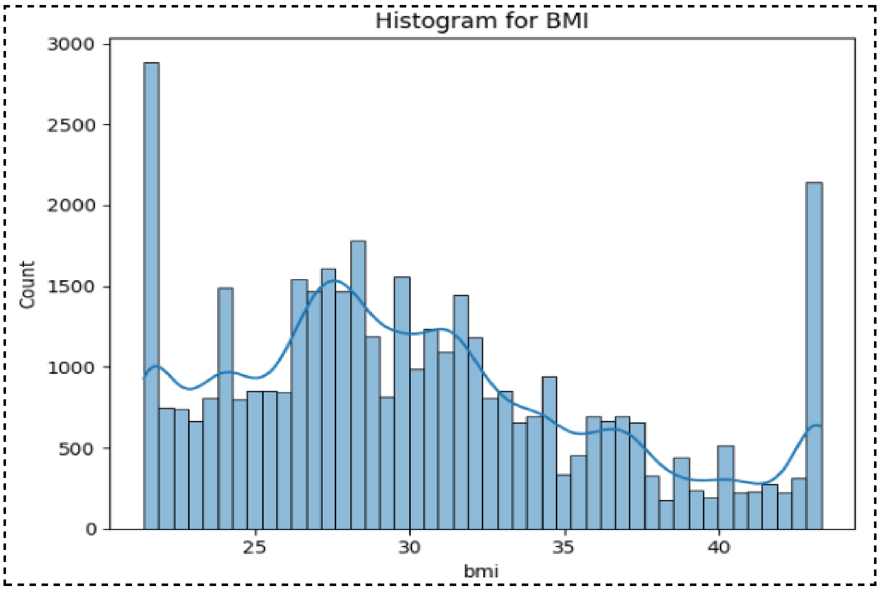


Fig. 15 Histogram for BMI distribution

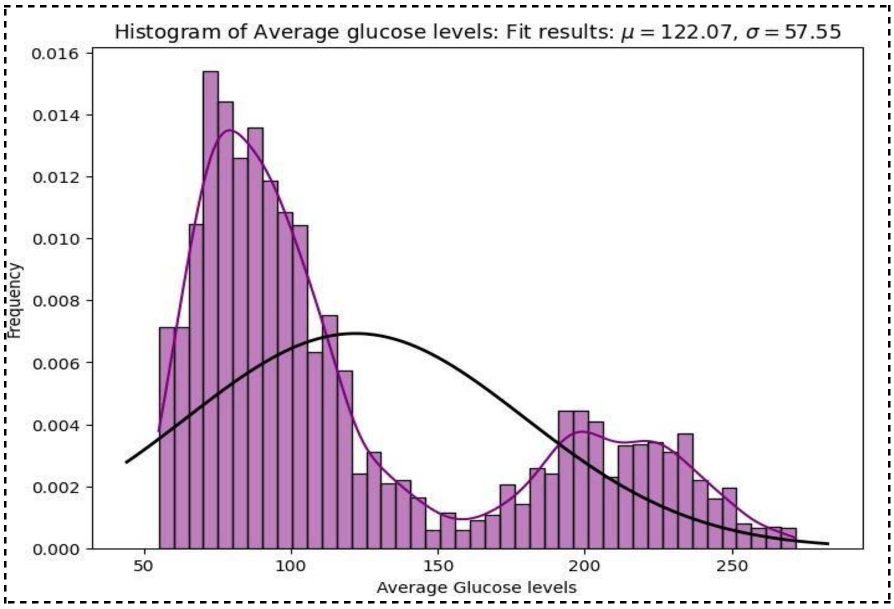


Fig. 16 Histogram for distribution of Average Glucose levels.

6.2 Exploratory Data Analysis

There is a higher prevalence of smoking, hypertension & Heart disease among individuals who have experienced a stroke. Females were more significantly affected by stroke than males.

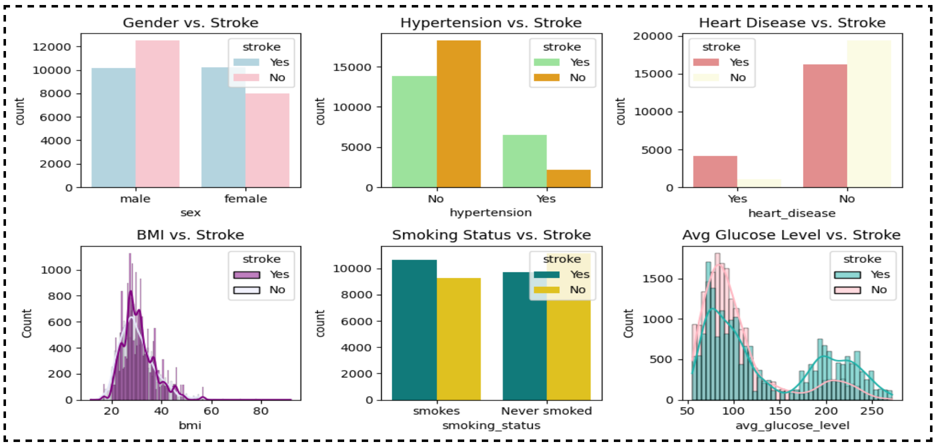
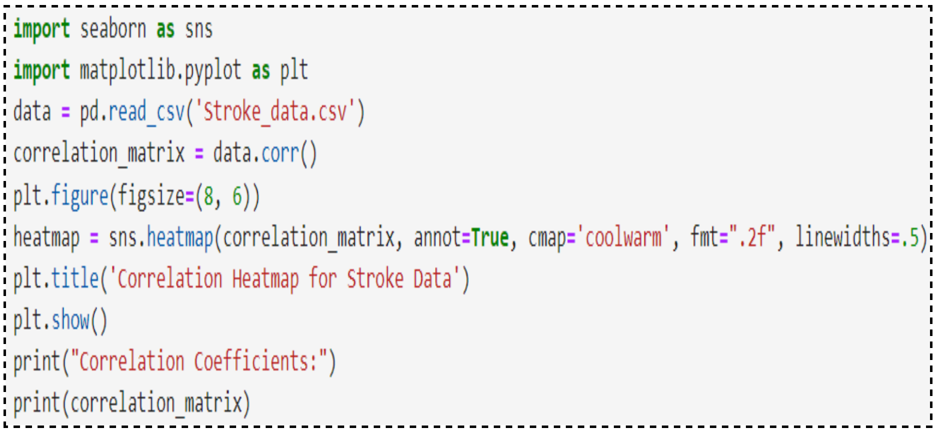


Fig. 17 Bar Charts & Histograms for Comparison of Risk factors with Stroke.

6.3 Correlation Matrix

We found strong positive correlations between stroke and hypertension, heart disease, average glucose level, and smoking status, and a weak negative correlation between stroke and gender.



```
import seaborn as sns
import matplotlib.pyplot as plt
data = pd.read_csv('Stroke_data.csv')
correlation_matrix = data.corr()
plt.figure(figsize=(8, 6))
heatmap = sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Heatmap for Stroke Data')
plt.show()
print("Correlation Coefficients:")
print(correlation_matrix)
```

Fig. 18 Python Code for Heatmap.

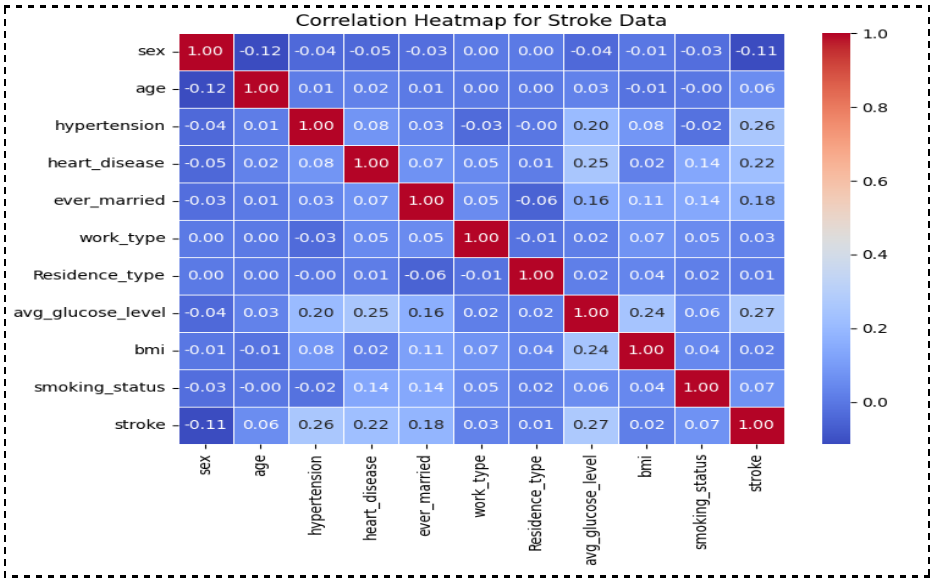
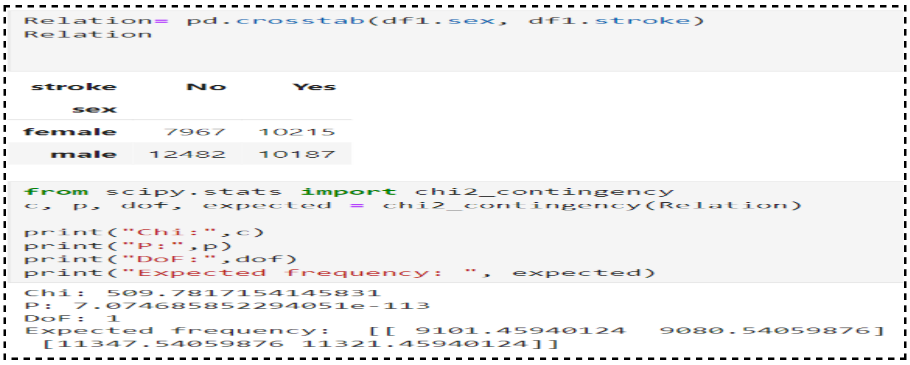


Fig. 19 Heatmap illustrating the correlation between risk factors and stroke.

6.4 Statistical Analysis

As the majority of variables in our dataset are categorical, we conducted chi-square tests to explore potential associations between the dependent variable and independent variables. We utilized the pd.crosstab function from the pandas library to construct contingency tables summarizing the distribution of observed frequencies. Additionally, we imported the chi2_contingency function from the scipy.stats module to facilitate the chi-square tests.

Null Hypothesis (H0): There is no significant association between sex and the occurrence of stroke.
Alternate Hypothesis (H1): There is a significant association between sex and the occurrence of stroke.



```
Relation= pd.crosstab(df1.sex, df1.stroke)
Relation
```

stroke	No	Yes
sex		
female	7967	10215
male	12482	10187

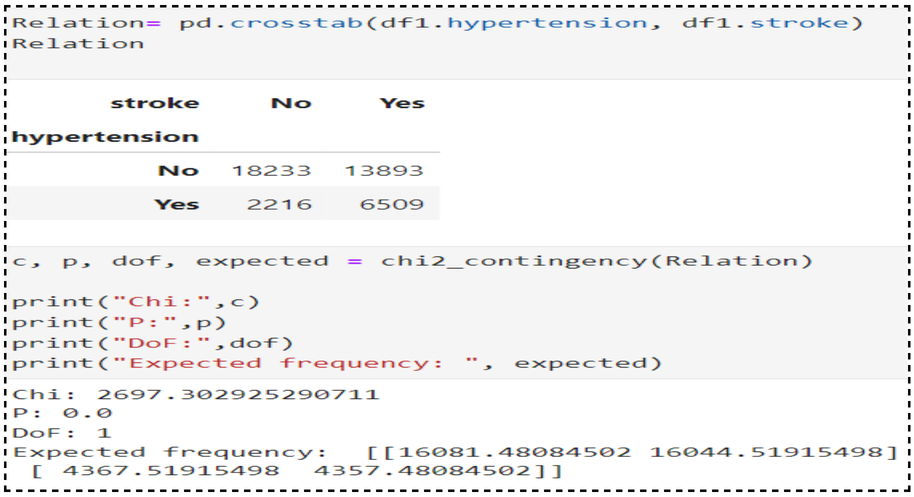
```
from scipy.stats import chi2_contingency
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
Chi: 509.7817154145831
P: 7.074685852294051e-113
DoF: 1
Expected frequency:  [[ 9101.45940124  9080.54059876]
 [11347.54059876 11321.45940124]]
```

Fig. 20 Performing a chi-square test in Python to examine the association between gender and stroke.

The extremely small p-value (7.074685852294051e-111) suggests that the observed association between gender and stroke is highly significant.

Null Hypothesis (H0): There is no significant association between hypertension and the occurrence of stroke.
Alternate Hypothesis (H1): There is a significant association between hypertension and the occurrence of stroke.



```
Relation= pd.crosstab(df1.hypertension, df1.stroke)
Relation
```

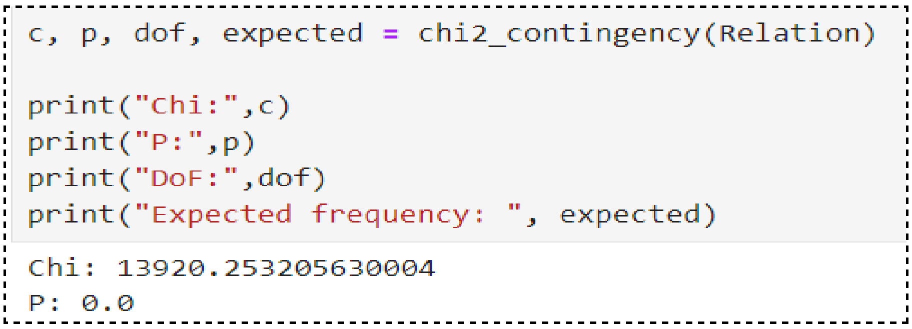
stroke	No	Yes
hypertension		
No	18233	13893
Yes	2216	6509

```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
Chi: 2697.302925290711
P: 0.0
DoF: 1
Expected frequency:  [[16081.48084502 16044.51915498]
 [ 4367.51915498  4357.48084502]]
```

Fig. 21 Performing a chi-square test in Python to examine the association between hypertension and stroke.

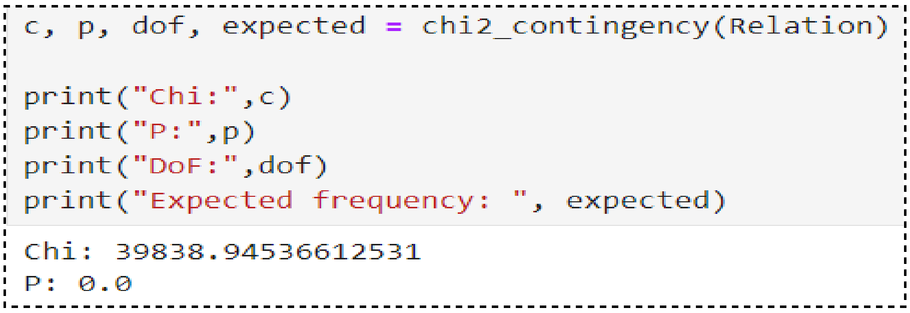
P-value is less than 0.05. Therefore, we reject the null hypothesis and there is a significant association between hypertension and stroke.



```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
Chi: 13920.253205630004
P: 0.0
```

Fig. 22 Performing a chi-square test in Python to examine the association between BMI and stroke.

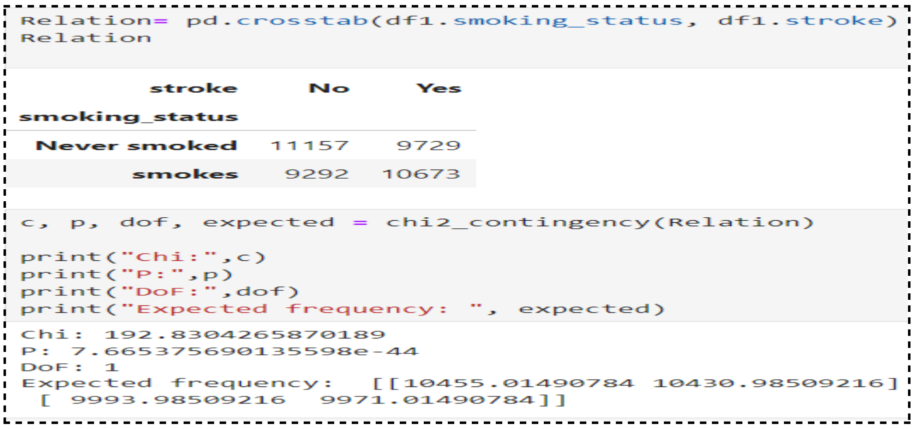


```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
Chi: 39838.94536612531
P: 0.0
```

Fig. 23 Performing a chi-square test in Python to examine the association between Average glucose levels and stroke.

Null Hypothesis (H0): There is no significant association between smoking status and the occurrence of stroke.
Alternate Hypothesis (H1): There is a significant association between smoking status and the occurrence of stroke.



```
Relation= pd.crosstab(df1.smoking_status, df1.stroke)
Relation
```

```
          stroke     No    Yes
smoking_status
   Never smoked   11157   9729
         smokes    9292  10673
```

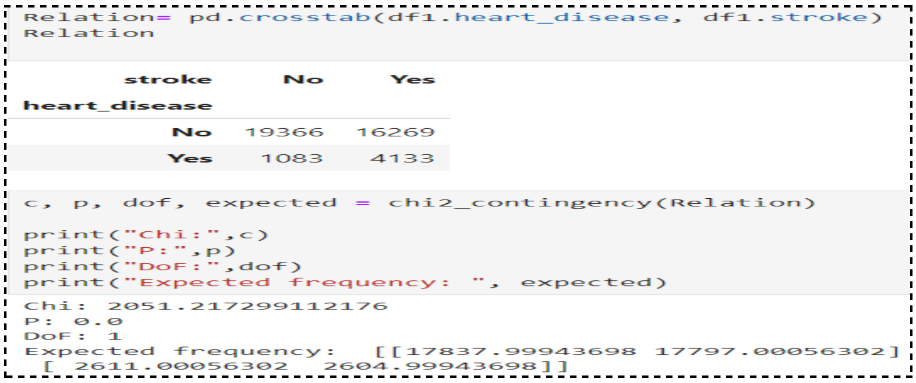
```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
Chi: 192.8304265870189
P: 7.665375690135598e-44
DoF: 1
Expected frequency:  [[10455.01490784 10430.98509216]
 [ 9993.98509216  9971.01490784]]
```

Fig. 24 Performing a chi-square test in Python to examine the association between Smoking Status and stroke.

As P-value (7.665375690135598e-44) is less than 0.05, we reject the null hypothesis. There is a significant association between smoking status and stroke.

Null Hypothesis (H0): There is no significant association between Heart disease and the occurrence of stroke.
Alternate Hypothesis (H1): There is a significant association between Heart Disease and the occurrence of stroke.



```
Relation= pd.crosstab(df1.heart_disease, df1.stroke)
Relation
```

```
        stroke      No     Yes
heart_disease
            No   19366   16269
           Yes    1083    4133
```

```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
Chi: 2051.217299112176
P: 0.0
DoF: 1
Expected frequency:  [[17837.99943698 17797.00056302]
 [ 2611.00056302  2604.99943698]]
```

Fig. 25 Performing a chi-square test in Python to examine the association between Heart disease and stroke.

The extremely small p-value P: 0.0 suggests that the observed association between heart disease and stroke is highly significant. Therefore, based on the data, we reject the null hypothesis.

Upon analyzing the results, we observed that for all variables, the calculated p-values were found to be less than the chosen significance level. Consequently, we rejected the null hypothesis, indicating that there exists a statistically significant association between the risk factors and occurrence of stroke..

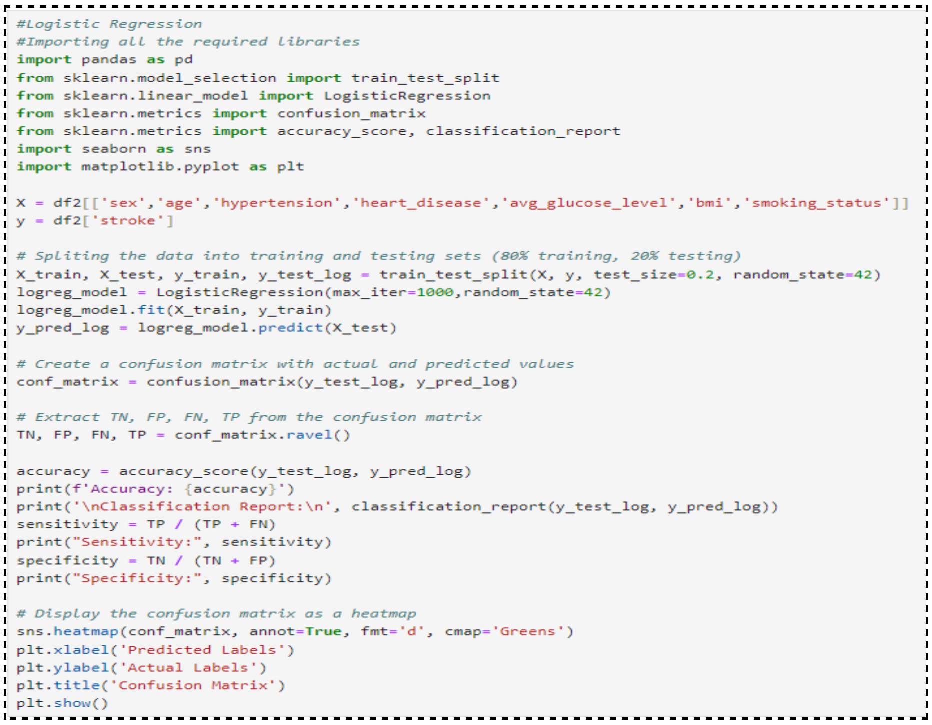
7 Machine Learning Models

Classification Models - As our dependent variable is categorical, we used classification models.

7.1 Logistic Regression

A logistic regression classification model is a statistical approach employed in tasks where the objective is to predict the likelihood of an observation falling into a specific category, especially when the outcome variable is binary, with two potential results. This model is commonly used in various applications where distinguishing between two classes is essential.

We created a logistic regression model using feature columns (sex, age, hypertension, heart disease, avg glucose level, BMI, smoking status) as predictors (X) and stroke as the dependent variable (Y). To split our dataset into training (80%) and testing (20%) data, we utilized the train_test_split function. The logistic regression model was constructed using the LogisticRegression function from the sklearn library.



```
#Logistic Regression
#Importing all the required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']

# Spliting the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test_log = train_test_split(X, y, test_size=0.2, random_state=42)
logreg_model = LogisticRegression(max_iter=1000,random_state=42)
logreg_model.fit(X_train, y_train)
y_pred_log = logreg_model.predict(X_test)

# Create a confusion matrix with actual and predicted values
conf_matrix = confusion_matrix(y_test_log, y_pred_log)

# Extract TN, FP, FN, TP from the confusion matrix
TN, FP, FN, TP = conf_matrix.ravel()

accuracy = accuracy_score(y_test_log, y_pred_log)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', classification_report(y_test_log, y_pred_log))
sensitivity = TP / (TP + FN)
print("Sensitivity:", sensitivity)
specificity = TN / (TN + FP)
print("Specificity:", specificity)

# Display the confusion matrix as a heatmap
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Greens')
plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix')
plt.show()
```

Fig. 26. Python code for Logistic Regression

We built the confusion matrix using actual and predicted values.

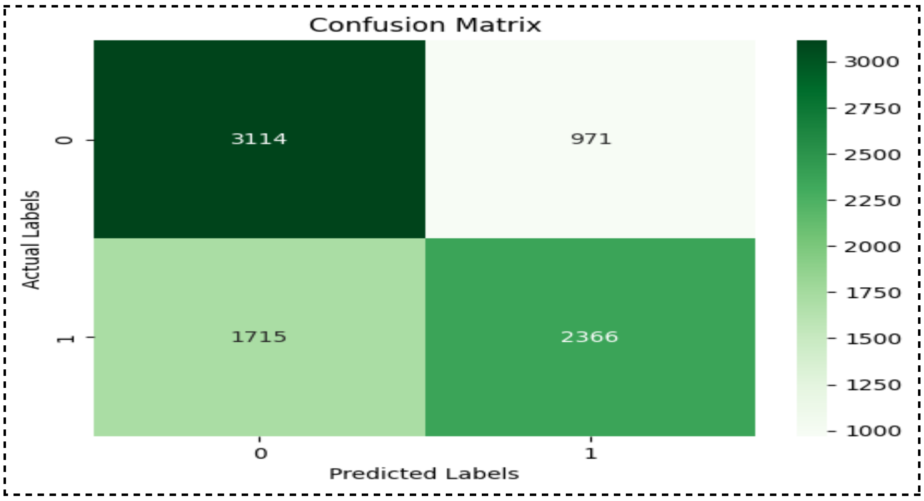


Fig. 27. Confusion Matrix for Logistic Regression

The confusion matrix indicates true positives (2366), true negatives (3114), false positive (971), false negative (1715).

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

           0       0.64      0.76      0.70      4085
           1       0.71      0.58      0.64      4081

    accuracy                           0.67      8166
   macro avg       0.68      0.67      0.67      8166
weighted avg       0.68      0.67      0.67      8166
```

From the classification report we can depict that the accuracy of the model is 67%.

We constructed a receiver operating characteristic curve (ROC) for our logistic regression model. The graph is plotted between true positive rate and false positive rate.

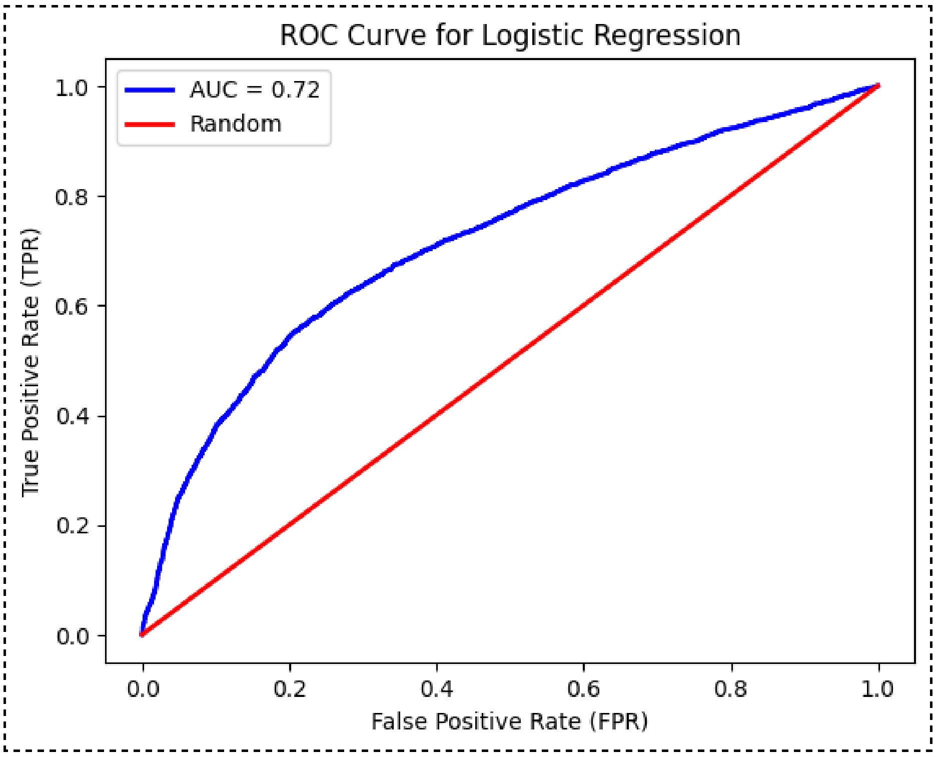
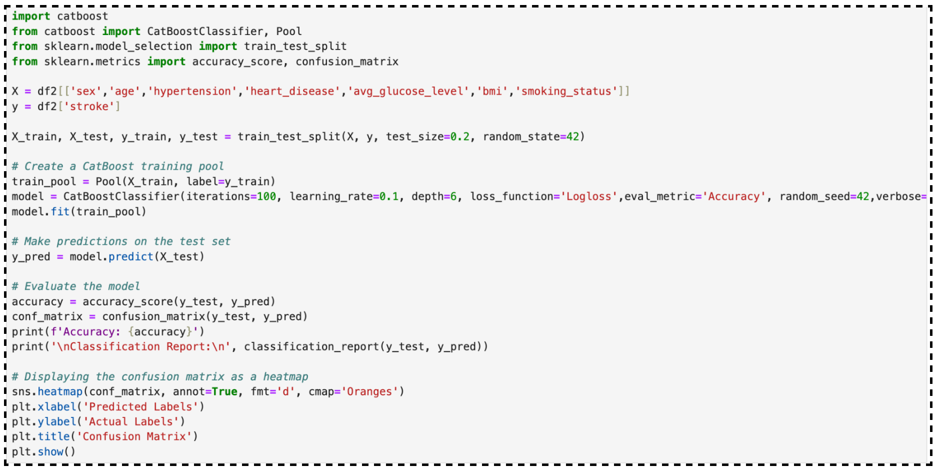


Fig. 28. ROC for Logistic Regression

The area under the curve (AUC) is 72%. It indicates that the model has some ability to distinguish between positive (people with stroke) and negative class (people without stroke).

7.2 CAT BOOST CLASSIFIER

Catboost is a variant of gradient boosting that can handle both categorical and numerical features.CatBoost can handle categorical features without any feature encoding and can easily handle missing values in the dataset.
We created a catboost classifier model using feature columns (sex, age, hypertension, heart disease, avg glucose level, BMI, smoking status) as predictors (X) and stroke as the dependent variable (Y). To split our dataset into training (80%) and testing (20%) data, we utilized the train_test_split function. The CatBoost classifier model was constructed using the CatBoostClassifier function from the sklearn library.



```
import catboost
from catboost import CatBoostClassifier, Pool
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a CatBoost training pool
train_pool = Pool(X_train, label=y_train)
model = CatBoostClassifier(iterations=100, learning_rate=0.1, depth=6, loss_function='Logloss',eval_metric='Accuracy', random_seed=42,verbose=
model.fit(train_pool)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', classification_report(y_test, y_pred))

# Displaying the confusion matrix as a heatmap
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Oranges')
plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix')
plt.show()
```

Fig. 29. Python code for Catboost

We build the confusion matrix using actual and predicted values.

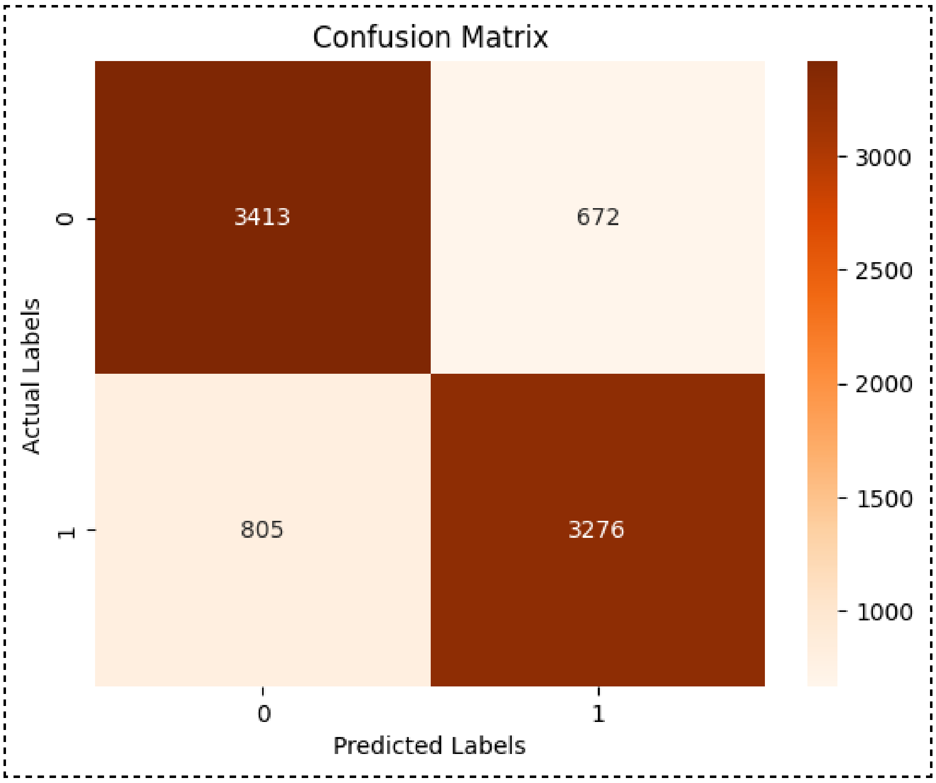
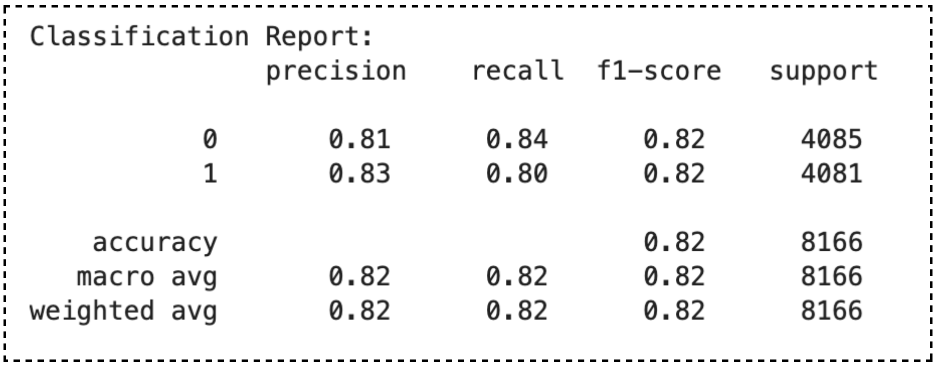


Fig. 30. Confusion Matrix for Catboost Classifier

The confusion matrix indicates true positives (3276), true negatives (3413), false positive (672), false negative (805).



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

           0       0.81      0.84      0.82      4085
           1       0.83      0.80      0.82      4081

    accuracy                           0.82      8166
   macro avg       0.82      0.82      0.82      8166
weighted avg       0.82      0.82      0.82      8166
```

Fig. 31. Classification Report for Cat Boost Classifier

The accuracy of the model is 82%.

We constructed a receiver operating characteristic curve (ROC) for our catboost classifier model. The graph is plotted between true positive rate and false positive rate.

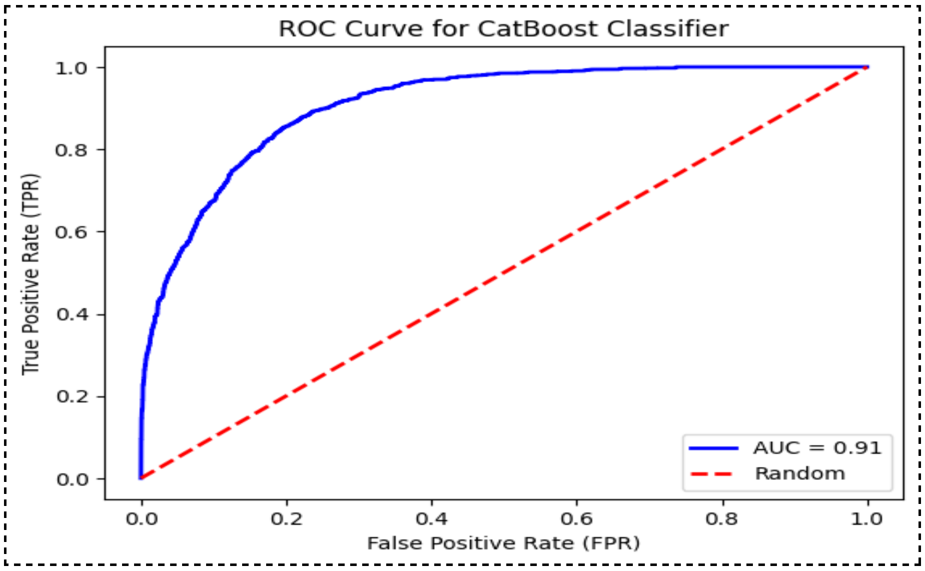


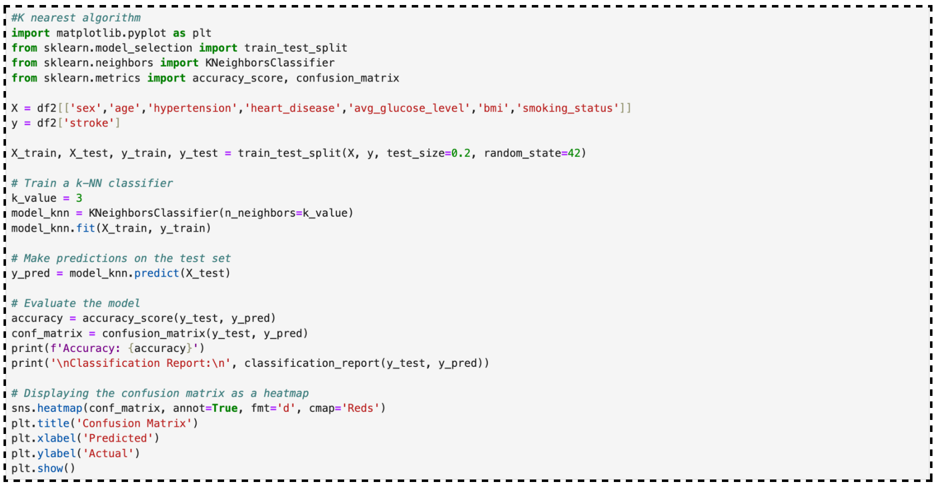
Fig. 32. ROC for Cat boost classifier

The area under the curve (AUC) is 91%. It indicates that the model has excellent ability to distinguish between positive (people with stroke) and negative class (people without stroke).

7.3 K NEAREST NEIGHBOUR ALGORITHM

A k-nearest neighbors (KNN) classification model is a supervised learning classifier used for classification tasks. In KNN, the class of an observation is predicted based on the majority class of its k nearest neighbors in the feature space. The "k" represents the number of neighbors considered, and the class is determined by a voting mechanism among these neighbors.
We created a K nearest neighbour model using feature columns (sex, age, hypertension, heart disease, avg glucose level, BMI, smoking status) as predictors (X) and stroke as the dependent variable (Y). To split our dataset into training (80%) and testing (20%) data, we utilized the train_test_split function. The K nearest neighbour model was constructed using the KNeighborsClassifier function from the sklearn

library.



```
#K nearest algorithm
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a k-NN classifier
k_value = 3
model_knn = KNeighborsClassifier(n_neighbors=k_value)
model_knn.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model_knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', classification_report(y_test, y_pred))

# Displaying the confusion matrix as a heatmap
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Reds')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Fig. 33. Python code for K nearest

We build the confusion matrix using actual and predicted values.

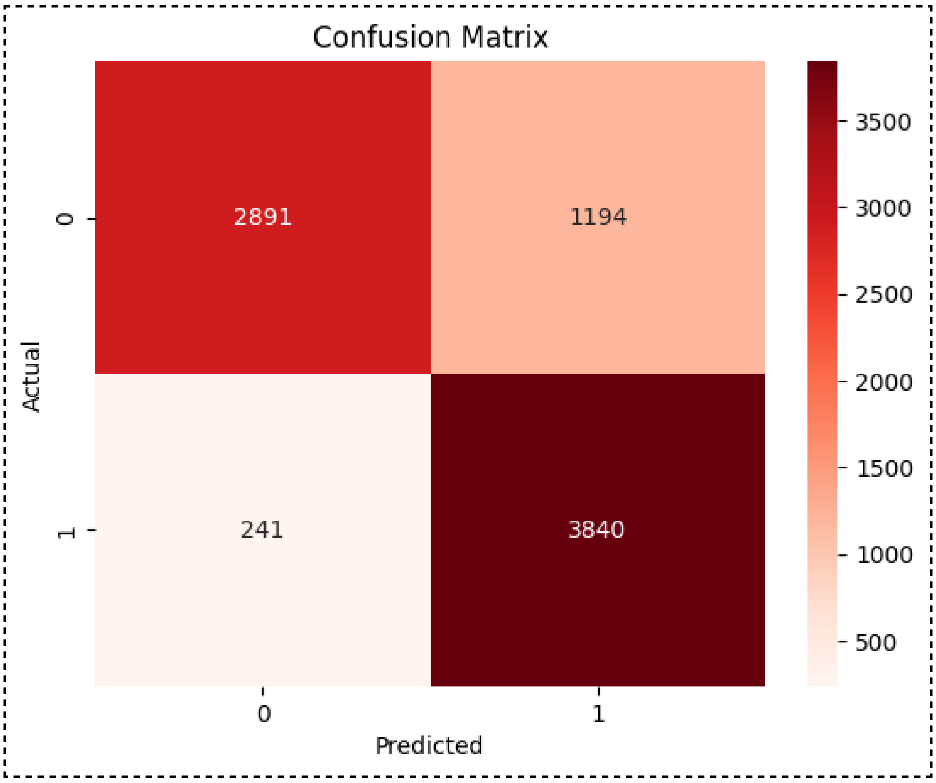
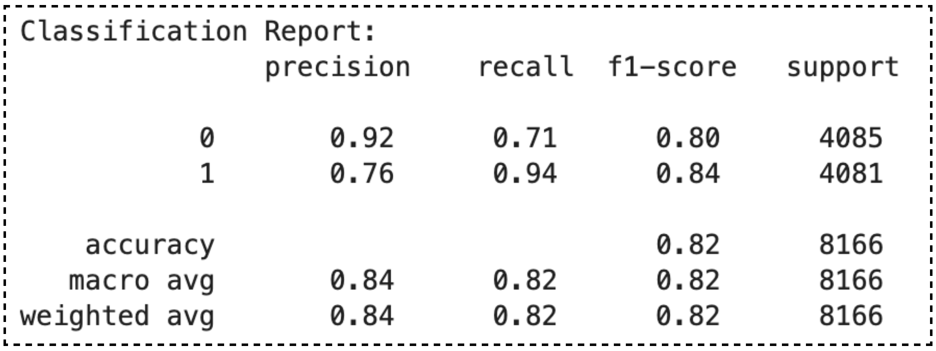


Fig. 34. Confusion Matrix for k nearest

The confusion matrix indicates true positives (3840), true negatives (2891), false positive (1194), false negative (241).



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

           0       0.92      0.71      0.80      4085
           1       0.76      0.94      0.84      4081

    accuracy                           0.82      8166
   macro avg       0.84      0.82      0.82      8166
weighted avg       0.84      0.82      0.82      8166
```

Fig. 35. Classification Report for k-NN Classifier

The accuracy of the model is 82%.

We constructed a receiver operating characteristic curve (ROC) for our K Nearest Neighbour model. The graph is plotted between true positive rate and false positive rate

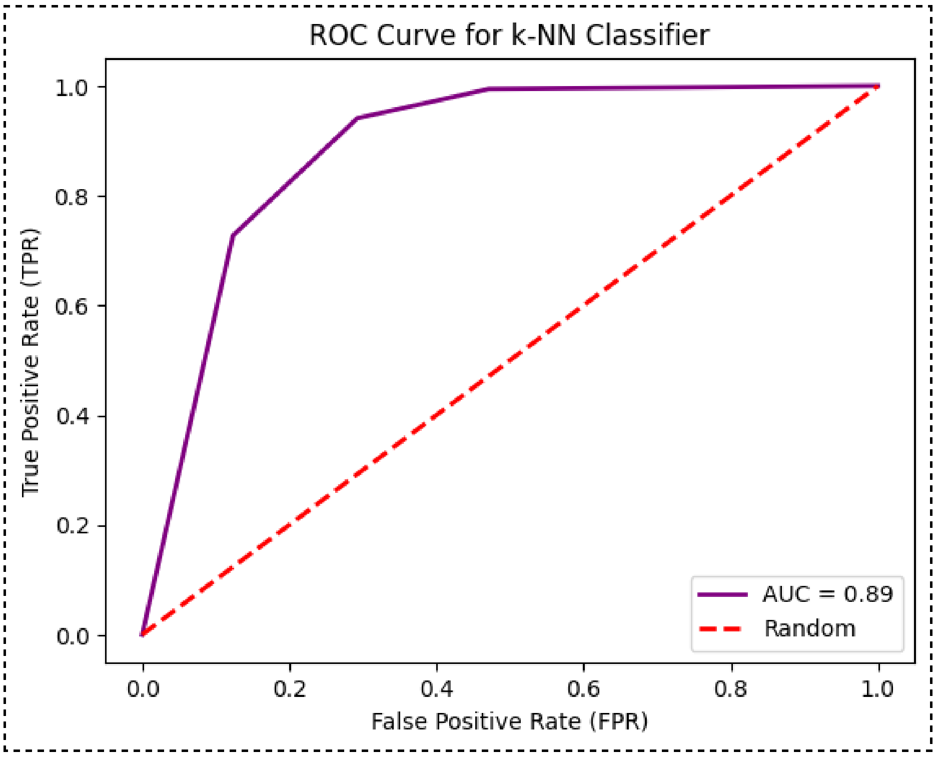


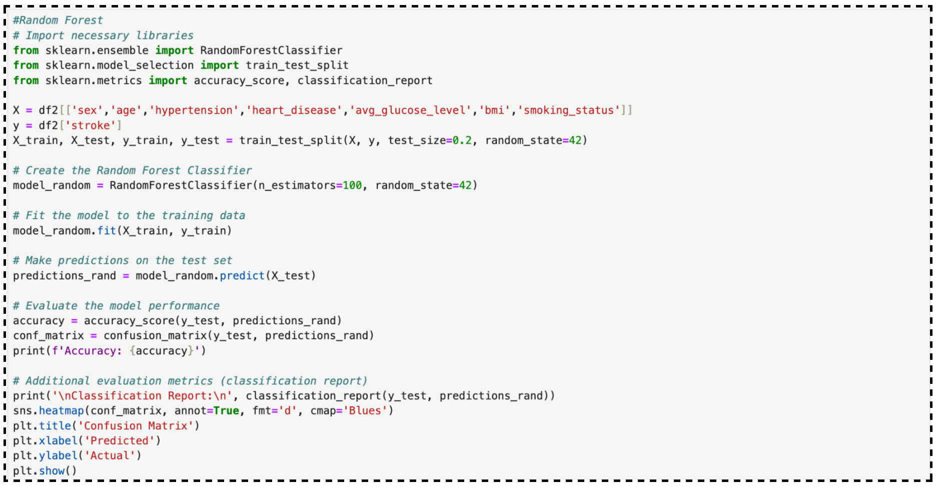
Fig. 36. ROC for k-NN

The area under the curve (AUC) is 89%. It indicates that the model has excellent ability to distinguish between positive (people with stroke) and negative class (people without stroke).

7.4 Random Forest

A Random Forest Classification Model is an ensemble machine learning algorithm designed for classification tasks. It's a versatile and powerful model that combines the strengths of multiple decision trees to make accurate predictions on categorical outcomes.

We created a random forest model using feature columns (sex, age, hypertension, heart disease, avg glucose level, BMI, smoking status) as predictors (X) and stroke as the dependent variable (Y). To split our dataset into training (80%) and testing (20%) data, we utilized the train_test_split function. The random forest model was constructed using the RandomForestClassifier function from the sklearn library.



```python
#Random Forest
# Import necessary libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the Random Forest Classifier
model_random = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model to the training data
model_random.fit(X_train, y_train)

# Make predictions on the test set
predictions_rand = model_random.predict(X_test)

# Evaluate the model performance
accuracy = accuracy_score(y_test, predictions_rand)
conf_matrix = confusion_matrix(y_test, predictions_rand)
print(f'Accuracy: {accuracy}')

# Additional evaluation metrics (classification report)
print('\nClassification Report:\n', classification_report(y_test, predictions_rand))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Fig. 37. Python code for Random forest classifier

We build the confusion matrix using actual and predicted values.

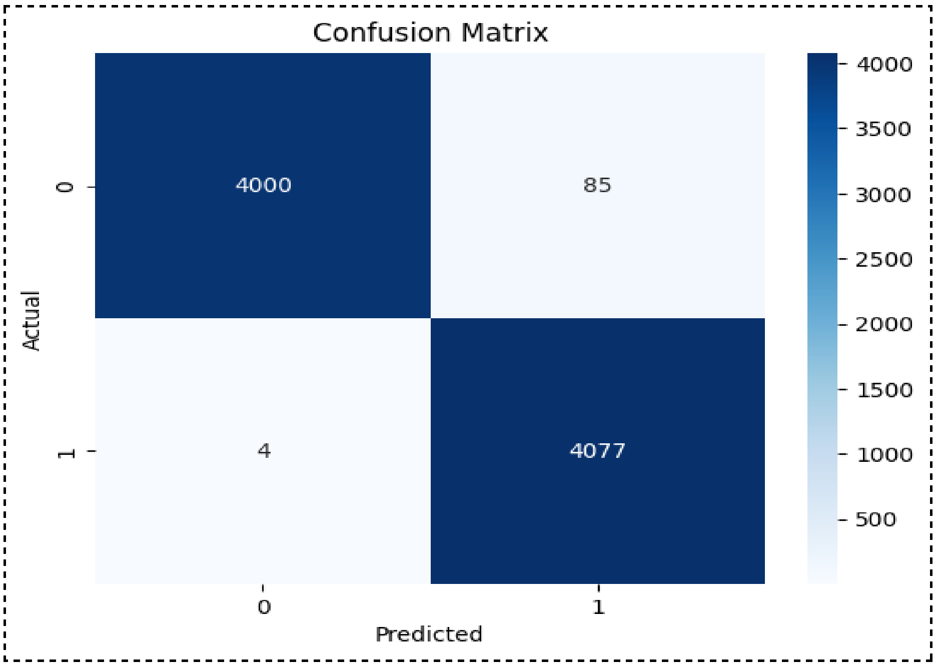
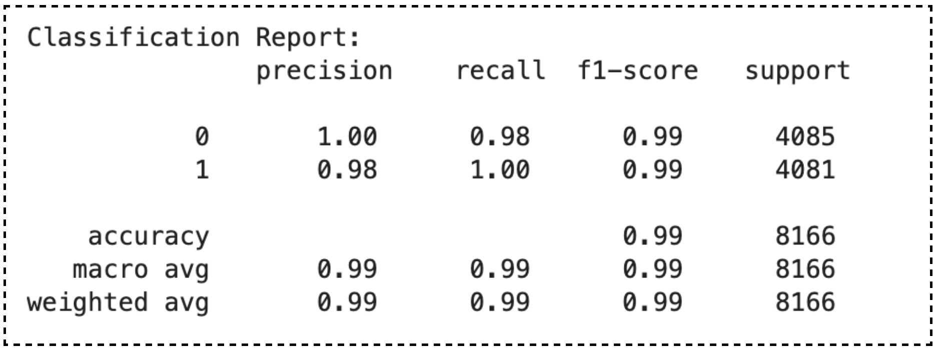


Fig. 38. Confusion Matrix for Random forest classifier

The confusion matrix indicates true positives (4077), true negatives (4000), false positive (85), false negative (4).



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

           0       1.00      0.98      0.99      4085
           1       0.98      1.00      0.99      4081

    accuracy                           0.99      8166
   macro avg       0.99      0.99      0.99      8166
weighted avg       0.99      0.99      0.99      8166
```

Fig. 39. Classification Report for Random forest Classifier

The accuracy of the model is 99%.

We constructed a receiver operating characteristic curve (ROC) for our random forest model. The graph is plotted between true positive rate and false positive rate

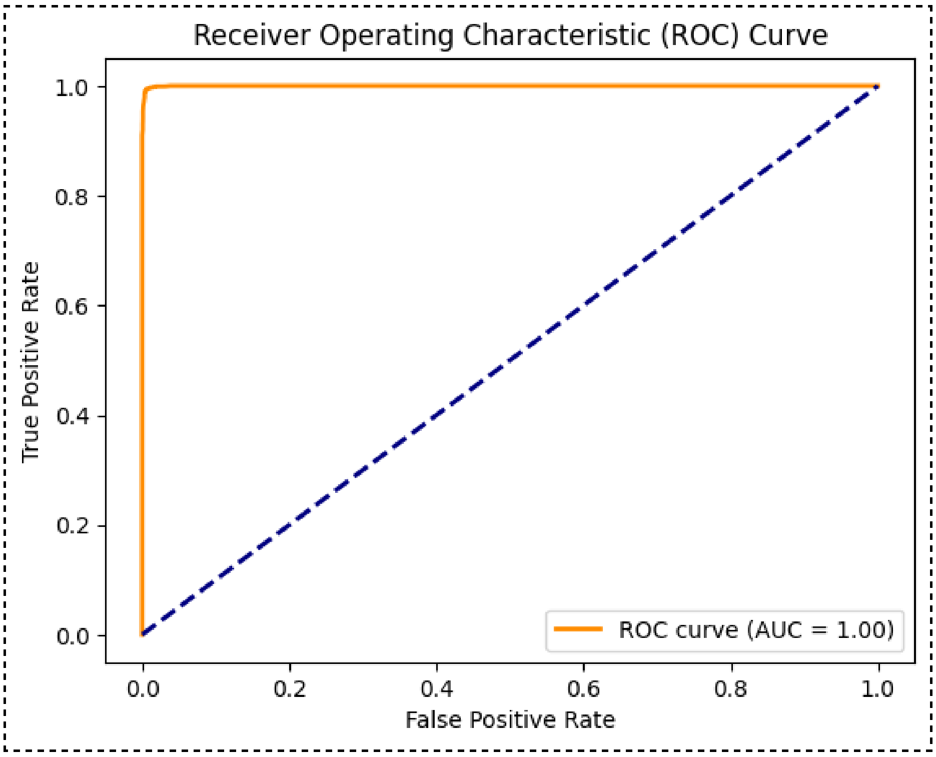
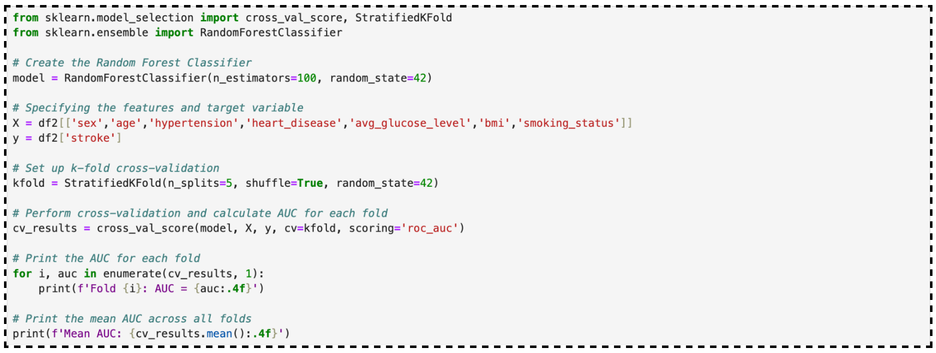


Fig. 40. ROC Curve for Random Forest Classifier

The area under the curve (AUC) is 100%. It indicates that the model has perfect ability to distinguish between positive (people with stroke) and negative class (people without stroke).

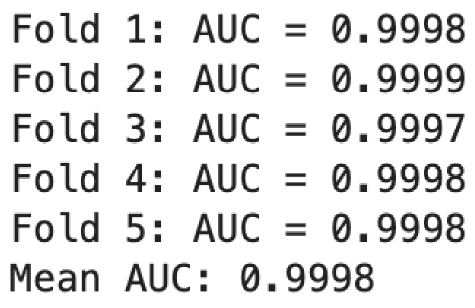
7.4.1 Cross Validation

We obtained an AUC value of 1 for the random forest, indicating perfect performance on the training data. To check for potential overfitting, we implemented a 5-fold cross-validation method. In each iteration, one part of the dataset serves as the test set, while the remaining four parts constitute the training set. We assess the accuracy for each training-testing pair and calculate the average accuracy across all iterations



```python
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier

# Create the Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Specifying the features and target variable
X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']

# Set up k-fold cross-validation
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform cross-validation and calculate AUC for each fold
cv_results = cross_val_score(model, X, y, cv=kfold, scoring='roc_auc')

# Print the AUC for each fold
for i, auc in enumerate(cv_results, 1):
    print(f'Fold {i}: AUC = {auc:.4f}')

# Print the mean AUC across all folds
print(f'Mean AUC: {cv_results.mean():.4f}')
```

Fig. 26. Python code for Cross Validation of Random Forest Classifier



```
Fold 1: AUC = 0.9998
Fold 2: AUC = 0.9999
Fold 3: AUC = 0.9997
Fold 4: AUC = 0.9998
Fold 5: AUC = 0.9998
Mean AUC: 0.9998
```

From the output we conclude that the random forest model is a good model with a mean AUC value of 0.99.

7.5 ROC curve comparison for all our models

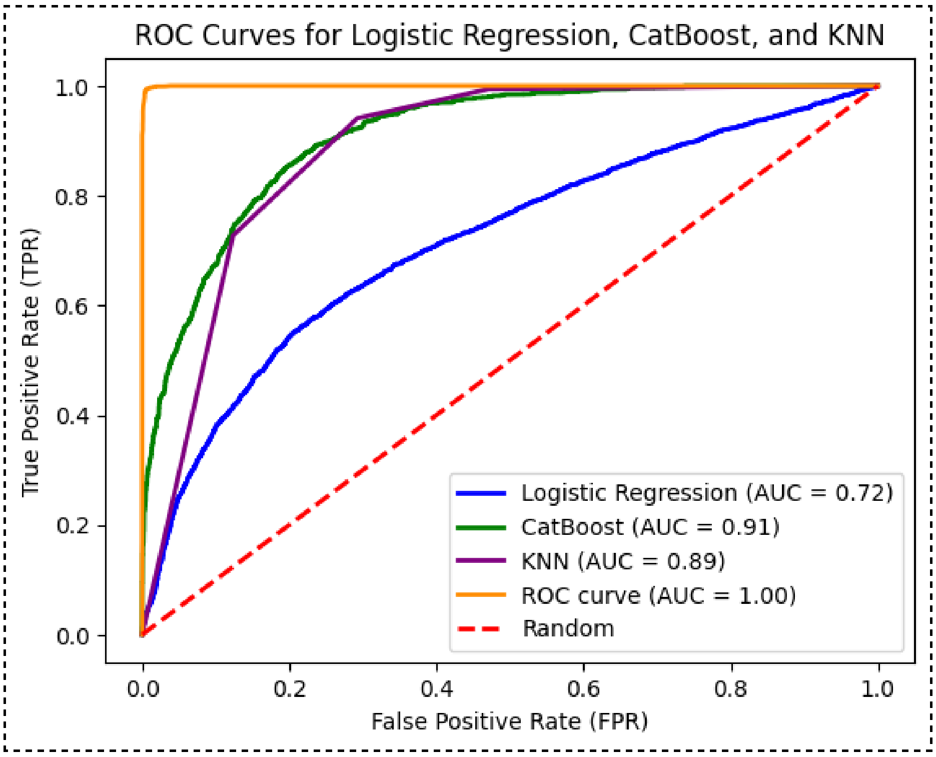


Fig. 26. ROC Curves for all models

From the graph we conclude that Random forest is the best model for our data followed by catboost classifier.

7 Results

• Based on the statistical analysis we reject the null hypothesis, signifying an underlying association among sex, age, hypertension, heart disease, glucose level, BMI, smoking on the incidence of stroke

• The Random Forest model demonstrates superior accuracy, precision, and F1 score compared to other models, underscoring its efficacy in predicting and understanding the complex relationships within the dataset.

8 Limitations

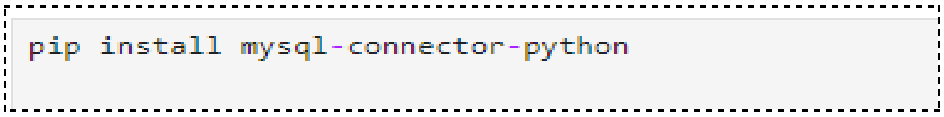
We considered consolidating datasets, but lacking unique columns for merging, we opted to use only one dataset.
Additionally, upon closer examination, we identified inconsistencies in the age values

within our selected dat

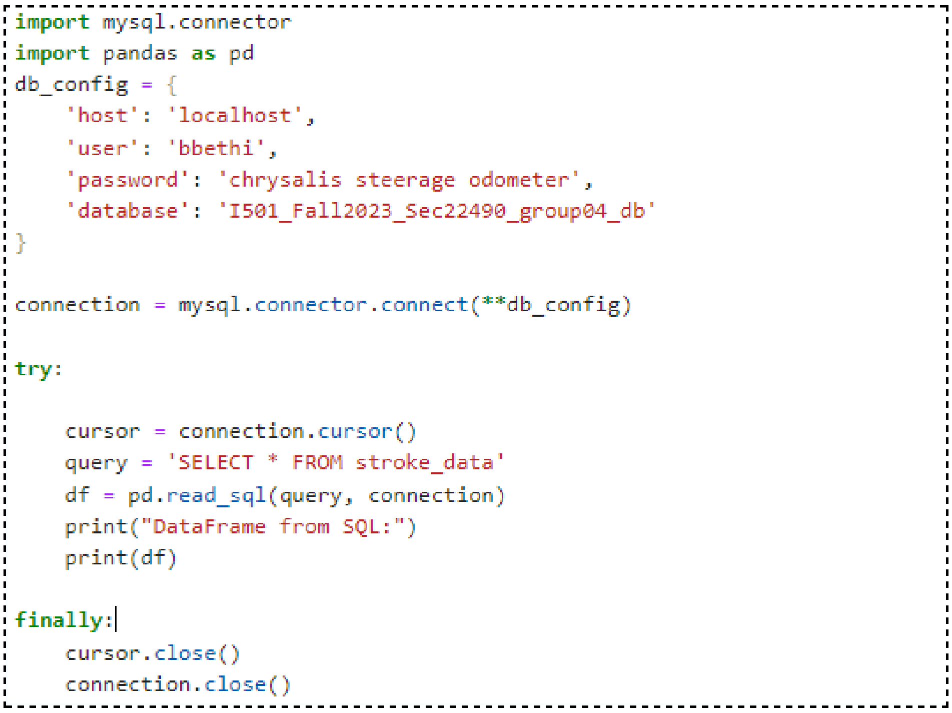
A Appendix

A.1 Data Cleaning

SQL Connection

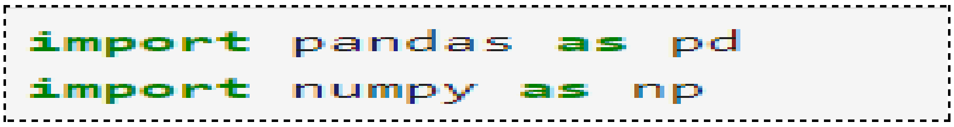


```
pip install mysql-connector-python
```



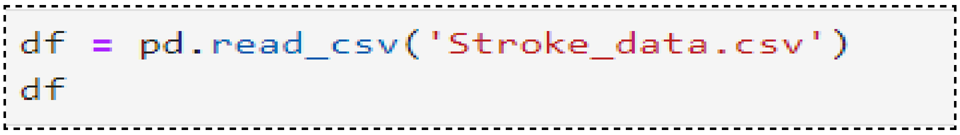
```
import mysql.connector
import pandas as pd
db_config = {
    'host': 'localhost',
    'user': 'bbethi',
    'password': 'chrysalis steerage odometer',
    'database': 'I501_Fall2023_Sec22490_group04_db'
}

connection = mysql.connector.connect(**db_config)

try:

    cursor = connection.cursor()
    query = 'SELECT * FROM stroke_data'
    df = pd.read_sql(query, connection)
    print("DataFrame from SQL:")
    print(df)

finally:
    cursor.close()
    connection.close()
```

Imported libraries



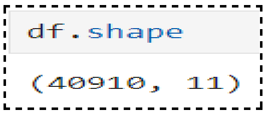
```
import pandas as pd
import numpy as np
```

Read CSV file

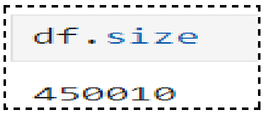


```
df = pd.read_csv('Stroke_data.csv')
df
```

Found Shape & Size of the dataframe

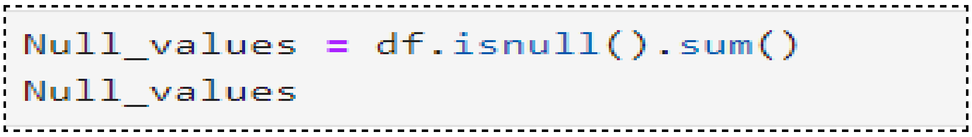


```
df.shape

(40910, 11)
```



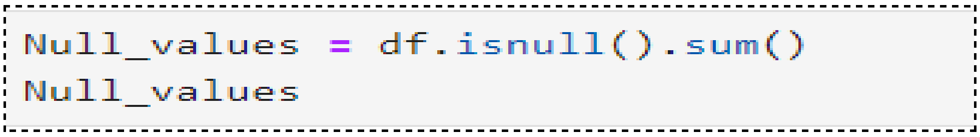
```
df.size

450010
```

Detected Null values

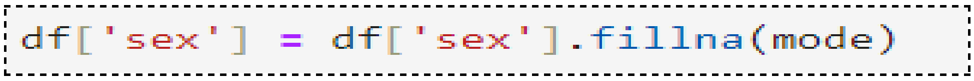


```
Null_values = df.isnull().sum()
Null_values
```

Imputed Null values

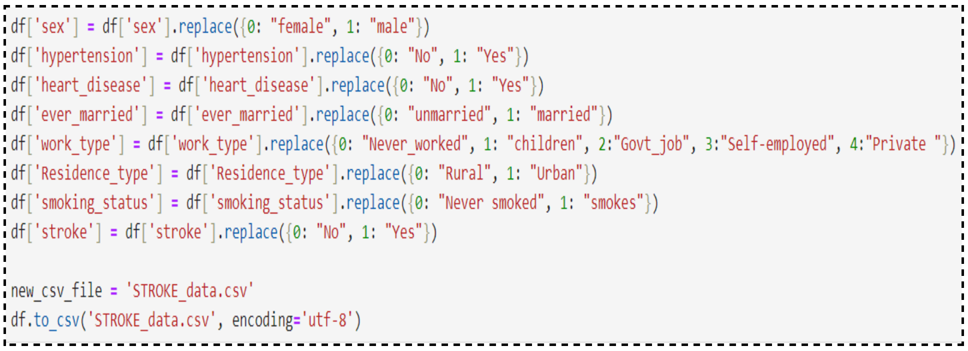


```
Null_values = df.isnull().sum()
Null_values
```



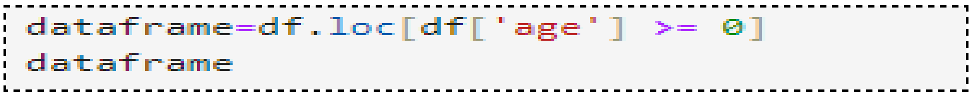
```
df['sex'] = df['sex'].fillna(mode)
```

Replaced Binary values with categorical values



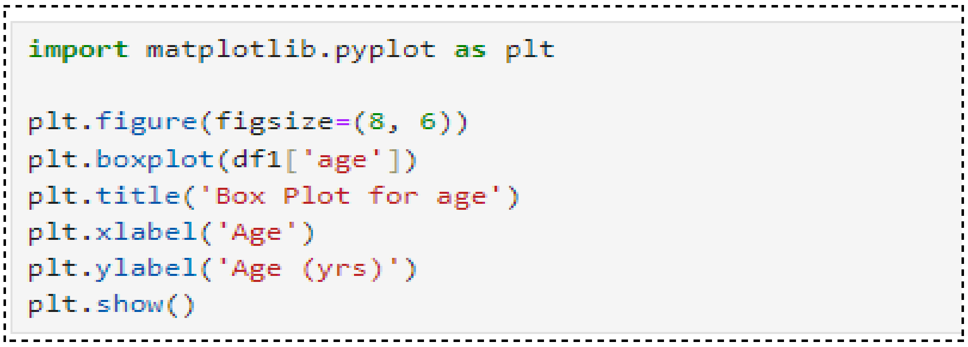
```
df['sex'] = df['sex'].replace({0: "female", 1: "male"})
df['hypertension'] = df['hypertension'].replace({0: "No", 1: "Yes"})
df['heart_disease'] = df['heart_disease'].replace({0: "No", 1: "Yes"})
df['ever_married'] = df['ever_married'].replace({0: "unmarried", 1: "married"})
df['work_type'] = df['work_type'].replace({0: "Never_worked", 1: "children", 2:"Govt_job", 3:"Self-employed", 4:"Private "})
df['Residence_type'] = df['Residence_type'].replace({0: "Rural", 1: "Urban"})
df['smoking_status'] = df['smoking_status'].replace({0: "Never smoked", 1: "smokes"})
df['stroke'] = df['stroke'].replace({0: "No", 1: "Yes"})

new_csv_file = 'STROKE_data.csv'
df.to_csv('STROKE_data.csv', encoding='utf-8')
```

Dropped Negative Values from Age

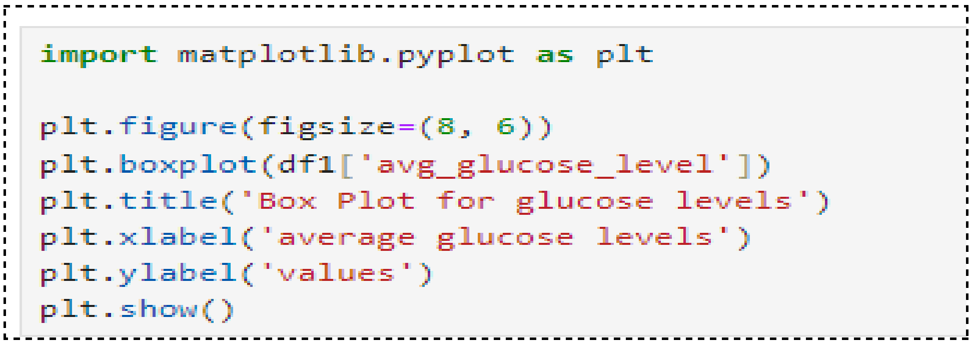


```
dataframe=df.loc[df['age'] >= 0]
dataframe
```

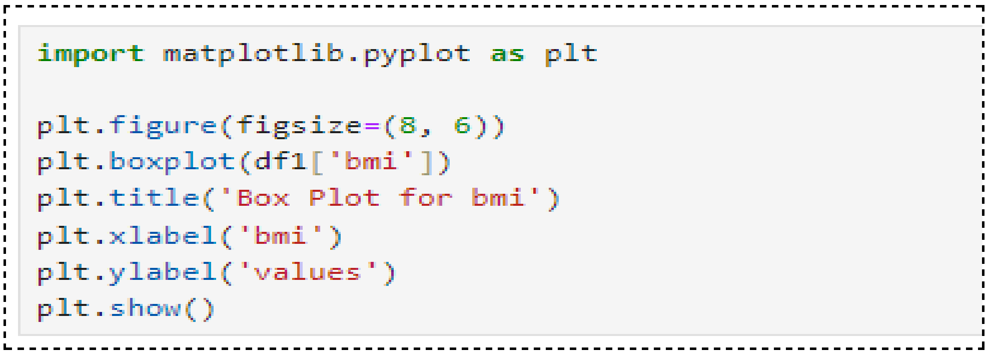
Detected & replaced Outliers



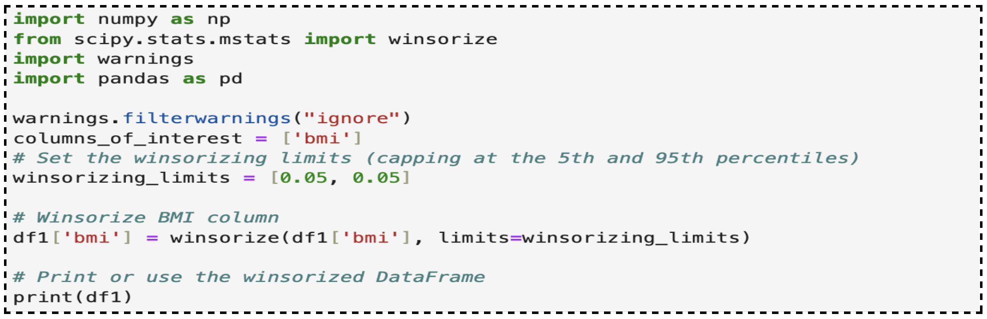
```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.boxplot(df1['age'])
plt.title('Box Plot for age')
plt.xlabel('Age')
plt.ylabel('Age (yrs)')
plt.show()
```



```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.boxplot(df1['avg_glucose_level'])
plt.title('Box Plot for glucose levels')
plt.xlabel('average glucose levels')
plt.ylabel('values')
plt.show()
```

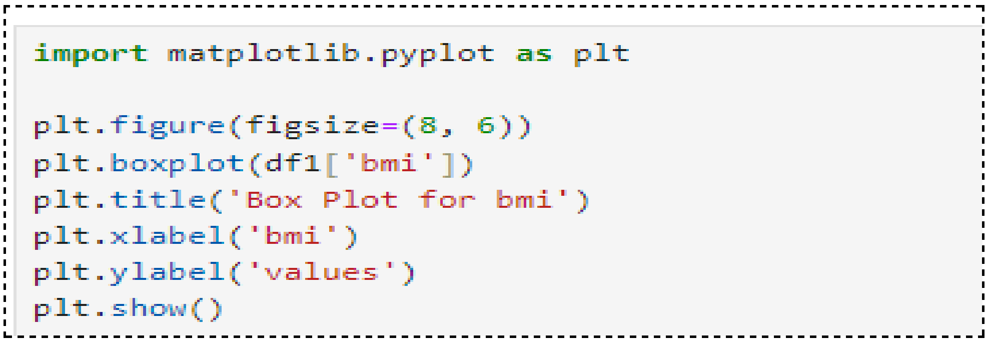


```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.boxplot(df1['bmi'])
plt.title('Box Plot for bmi')
plt.xlabel('bmi')
plt.ylabel('values')
plt.show()
```



```python
import numpy as np
from scipy.stats.mstats import winsorize
import warnings
import pandas as pd

warnings.filterwarnings("ignore")
columns_of_interest = ['bmi']
# Set the winsorizing limits (capping at the 5th and 95th percentiles)
winsorizing_limits = [0.05, 0.05]

# Winsorize BMI column
df1['bmi'] = winsorize(df1['bmi'], limits=winsorizing_limits)

# Print or use the winsorized DataFrame
print(df1)
```

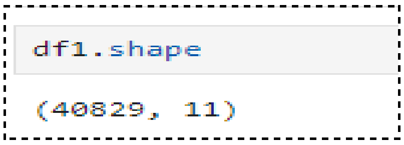
Checked for Outliers again after replacing them



```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.boxplot(df1['bmi'])
plt.title('Box Plot for bmi')
plt.xlabel('bmi')
plt.ylabel('values')
plt.show()
```

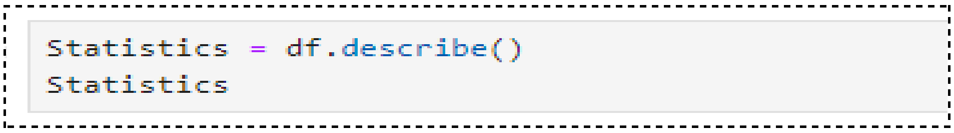
A.2 Descriptive Statistics

Checked shape of the dataframe after data cleaning



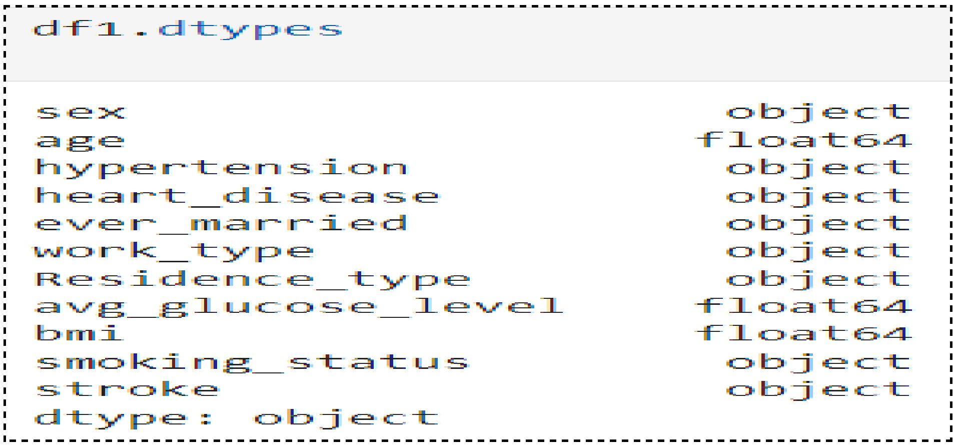
```
df1.shape

(40829, 11)
```

Generated Descriptive statistics



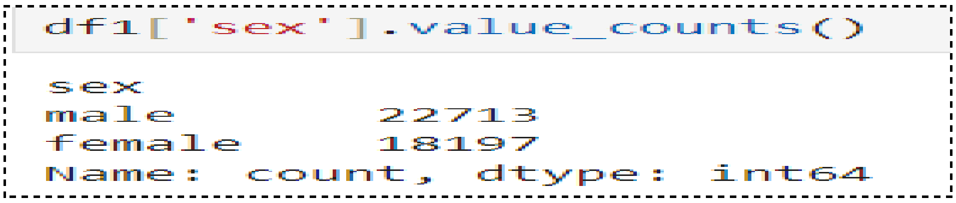
```
Statistics = df.describe()
Statistics
```

Found column types

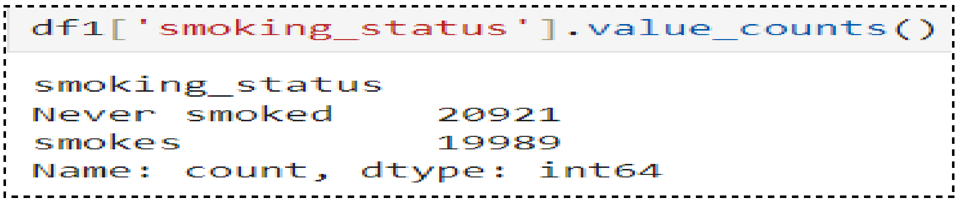


```
df1.dtypes

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

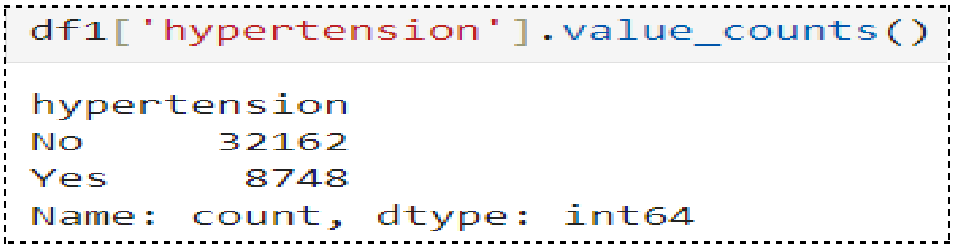
Checked value counts for each column



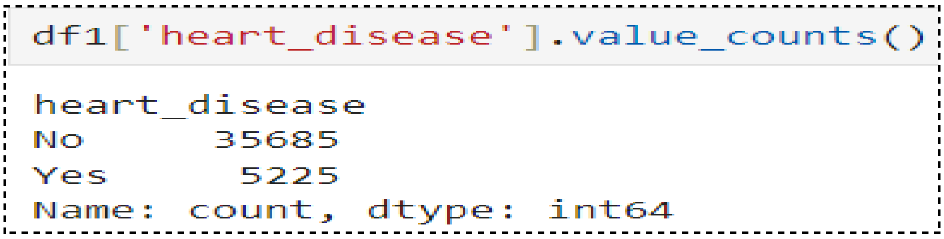
```
df1['sex'].value_counts()

sex
male          22713
female        18197
Name: count, dtype: int64
```



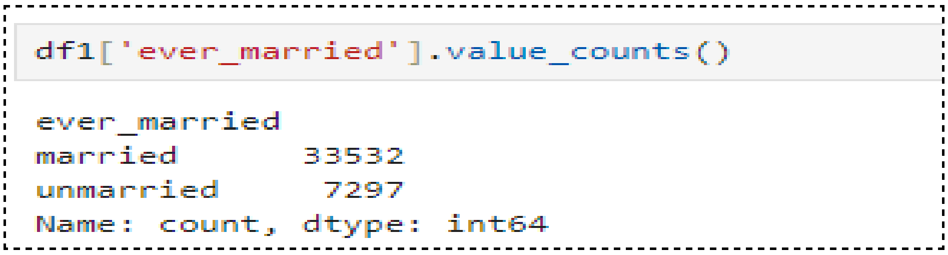
```
df1['smoking_status'].value_counts()

smoking_status
Never smoked     20921
smokes           19989
Name: count, dtype: int64
```



```
df1['hypertension'].value_counts()

hypertension
No      32162
Yes      8748
Name: count, dtype: int64
```



```
df1['heart_disease'].value_counts()

heart_disease
No      35685
Yes      5225
Name: count, dtype: int64
```



```
df1['ever_married'].value_counts()

ever_married
married       33532
unmarried      7297
Name: count, dtype: int64
```

```
df1['work_type'].value_counts()
```

```
work_type
Private          25516
Self-employed     9217
Govt_job          5580
children           431
Never_worked        85
Name: count, dtype: int64
```

```
df1['Residence_type'].value_counts()
```

```
Count = df1['stroke'].value_counts()
Count
```

A.3 Data Visualization

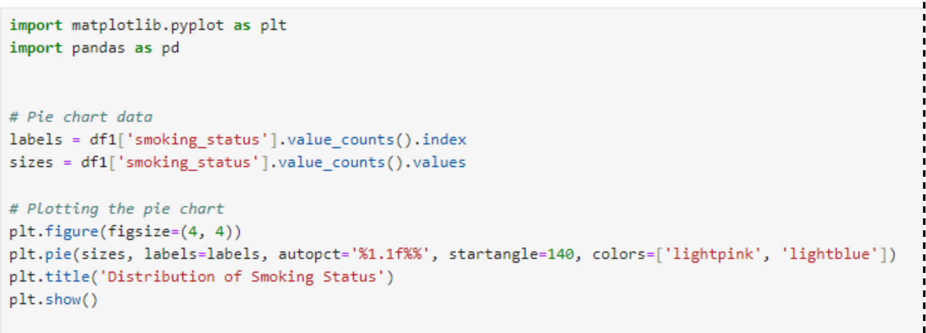
Distribution of Age

```
import matplotlib.pyplot as plt
values = df1['age']
plt.hist(values, bins = 'auto', color='purple', edgecolor='black', alpha = 0.7)
plt.xlim(0, 100)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution')
plt.show()
```

Distribution Of Gender

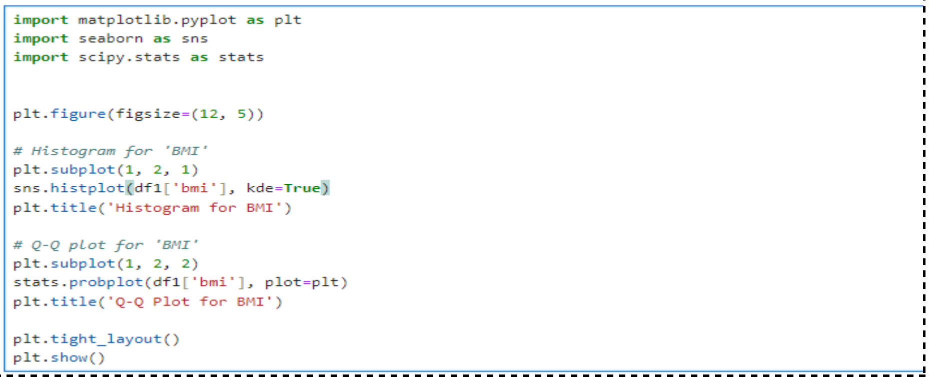
```
import matplotlib.pyplot as plt
gender_proportion = df1['sex'].value_counts()
Gender = gender_proportion.index
Count = gender_proportion.values
plt.bar(Gender, Count, color='purple', edgecolor='black', alpha = 0.5)
plt.title('Distribution of Gender')
plt.xlabel('Gender')
plt.ylabel('Number of people')
plt.show()
```

Distribution of Smoking status

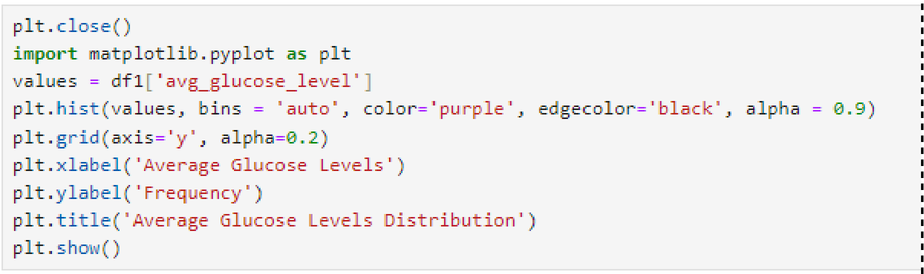


```python
import matplotlib.pyplot as plt
import pandas as pd


# Pie chart data
labels = df1['smoking_status'].value_counts().index
sizes = df1['smoking_status'].value_counts().values

# Plotting the pie chart
plt.figure(figsize=(4, 4))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, colors=['lightpink', 'lightblue'])
plt.title('Distribution of Smoking Status')
plt.show()
```

Distribution of BMI & Average Glucose levels



```python
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats


plt.figure(figsize=(12, 5))

# Histogram for 'BMI'
plt.subplot(1, 2, 1)
sns.histplot(df1['bmi'], kde=True)
plt.title('Histogram for BMI')

# Q-Q plot for 'BMI'
plt.subplot(1, 2, 2)
stats.probplot(df1['bmi'], plot=plt)
plt.title('Q-Q Plot for BMI')

plt.tight_layout()
plt.show()
```



```python
plt.close()
import matplotlib.pyplot as plt
values = df1['avg_glucose_level']
plt.hist(values, bins = 'auto', color='purple', edgecolor='black', alpha = 0.9)
plt.grid(axis='y', alpha=0.2)
plt.xlabel('Average Glucose Levels')
plt.ylabel('Frequency')
plt.title('Average Glucose Levels Distribution')
plt.show()
```

Distribution of Stroke cases

```
plt.close()
import matplotlib.pyplot as plt
labels = ['STROKE', 'NO-STROKE']
Count = [df1['stroke'].value_counts()['Yes'], df1['stroke'].value_counts()['No']]
colors =  ['lightpink', 'lightblue']
plt.figure(figsize=(5,5))
plt.pie(Count, labels=labels, colors=colors, autopct = '%1.1f%%', startangle=75)
plt.title("Distribution of Stroke")
plt.show()
```

A.4 Exploratory Data Analysis

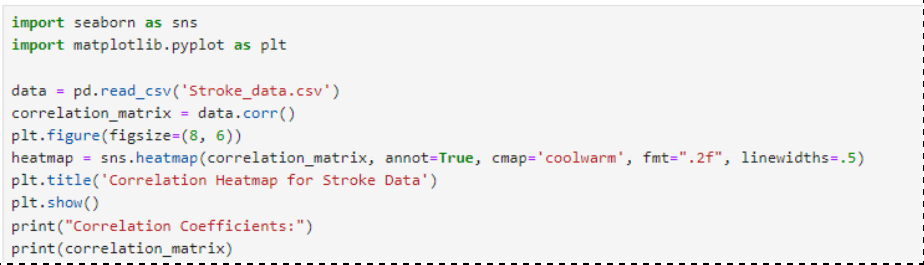
Distribution of Stroke based on Gender

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame({'sex': ['male', 'male', 'female', 'female', 'male', 'female'],
                   'stroke': ['Yes', 'No', 'Yes', 'No', 'Yes', 'No']})
stroke_counts = df.groupby(['sex', 'stroke']).size()
stroke_counts_stroke = stroke_counts['male']['Yes'], stroke_counts['female']['Yes']
gender_labels = ['male', 'female']
colors = ['lightblue', 'pink']
plt.figure(figsize=(5,5))
plt.pie(stroke_counts_stroke, labels=gender_labels, colors=colors, autopct='%1.1f%%', startangle=180)
plt.title("Distribution of Stroke based on Gender")
plt.show()
```

Distribution of Risk factors vs Stroke

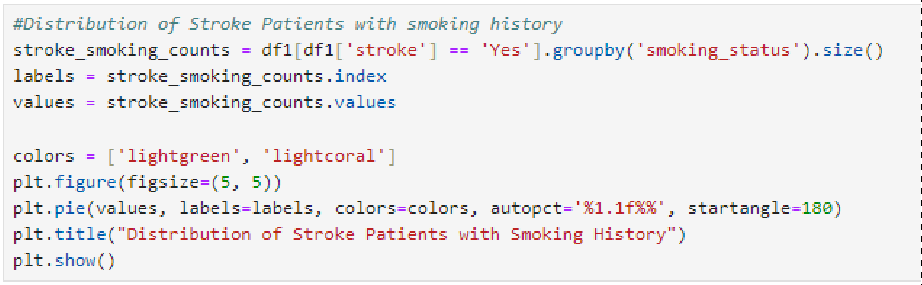
```
import seaborn as sns
palette_0 = ["lightblue", "pink"]
palette_1 = ["lightgreen", "orange"]
palette_2 = ["lightcoral", "lightyellow"]
palette_3 = ["purple", "lavender"]
palette_4 = ["darkcyan", "gold"]
palette_5 = ["lightseagreen", "lightpink"]

fig, axes = plt.subplots(2, 3, figsize=(10, 6))
axes = axes.flatten()
sns.countplot(x='sex', hue='stroke', data=df1, ax=axes[0], palette=palette_0)
axes[0].set_title('Gender vs. Stroke')
sns.countplot(x='hypertension', hue='stroke', data=df1, ax=axes[1], palette=palette_1)
axes[1].set_title('Hypertension vs. Stroke')
sns.countplot(x='heart_disease', hue='stroke', data=df1, ax=axes[2], palette=palette_2)
axes[2].set_title('Heart Disease vs. Stroke')
sns.histplot(x='bmi', hue='stroke', data=df1, kde=True, ax=axes[3], palette=palette_3)
axes[3].set_title('BMI vs. Stroke')
sns.countplot(x='smoking_status', hue='stroke', data=df1, ax=axes[4], palette=palette_4)
axes[4].set_title('Smoking Status vs. Stroke')
sns.histplot(x='avg_glucose_level', hue='stroke', data=df1, kde=True, ax=axes[5], palette=palette_5)
axes[5].set_title('Avg Glucose Level vs. Stroke')

plt.tight_layout()
plt.show()
```

Correlation Matrix



```
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv('Stroke_data.csv')
correlation_matrix = data.corr()
plt.figure(figsize=(8, 6))
heatmap = sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Heatmap for Stroke Data')
plt.show()
print("Correlation Coefficients:")
print(correlation_matrix)
```

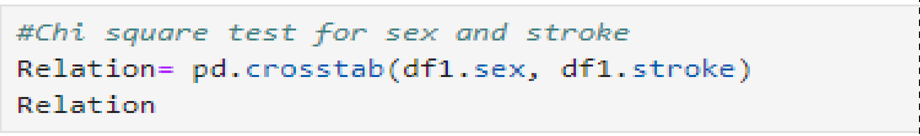
Distribution of Stroke Patients with Smoking History



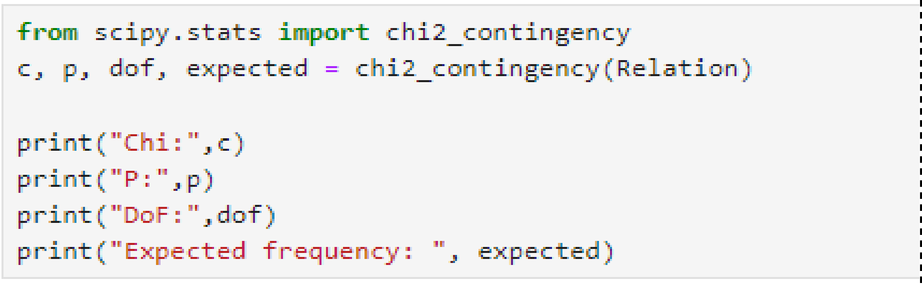
```
#Distribution of Stroke Patients with smoking history
stroke_smoking_counts = df1[df1['stroke'] == 'Yes'].groupby('smoking_status').size()
labels = stroke_smoking_counts.index
values = stroke_smoking_counts.values

colors = ['lightgreen', 'lightcoral']
plt.figure(figsize=(5, 5))
plt.pie(values, labels=labels, colors=colors, autopct='%1.1f%%', startangle=180)
plt.title("Distribution of Stroke Patients with Smoking History")
plt.show()
```

A.5 Statistical Analysis

Chi Square test



```
#Chi square test for sex and stroke
Relation= pd.crosstab(df1.sex, df1.stroke)
Relation
```



```
from scipy.stats import chi2_contingency
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
#Chi square test for hypertension and stroke
Relation= pd.crosstab(df1.hypertension, df1.stroke)
Relation
```

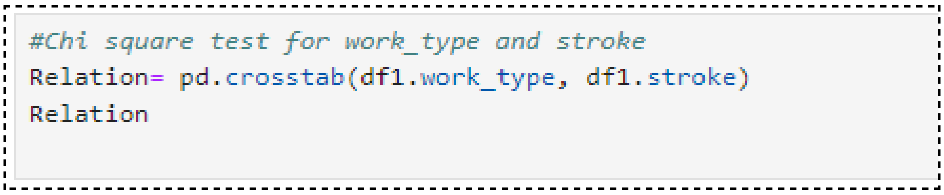
```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
#Chi square test for heart_disease and stroke
Relation= pd.crosstab(df1.heart_disease, df1.stroke)
Relation
```

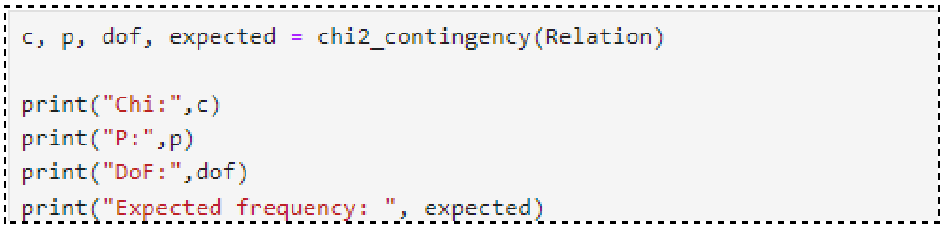
```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

```
#Chi square test for ever_married and stroke
Relation= pd.crosstab(df1.ever_married, df1.stroke)
Relation
```

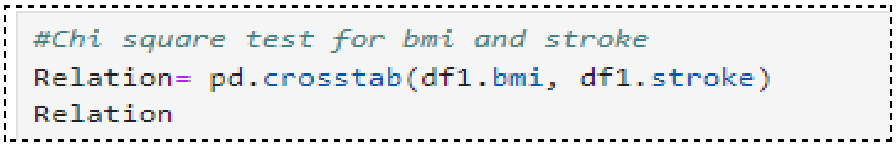
```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```



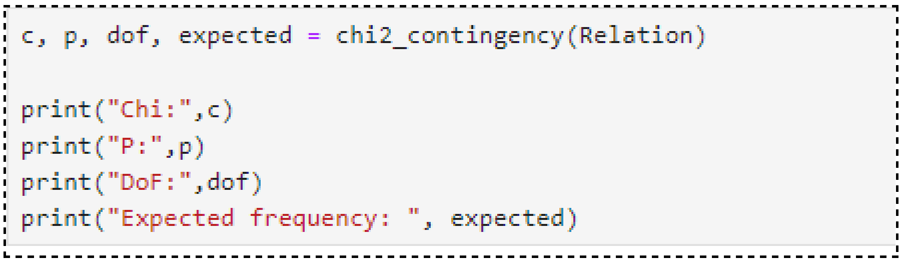
```
#Chi square test for work_type and stroke
Relation= pd.crosstab(df1.work_type, df1.stroke)
Relation
```



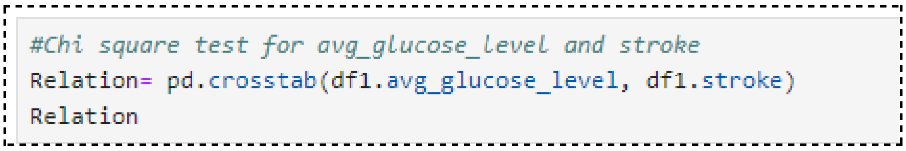
```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```



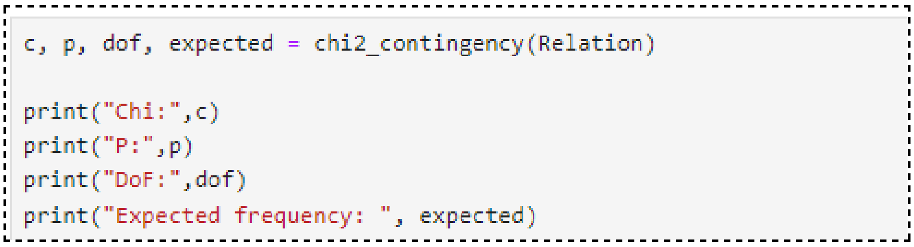
```
#Chi square test for bmi and stroke
Relation= pd.crosstab(df1.bmi, df1.stroke)
Relation
```



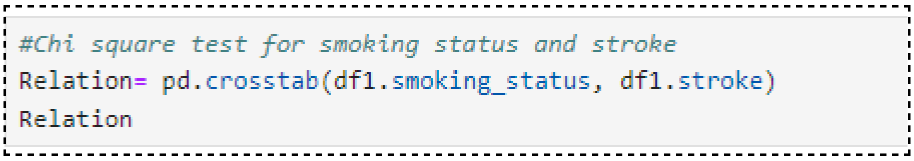
```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```



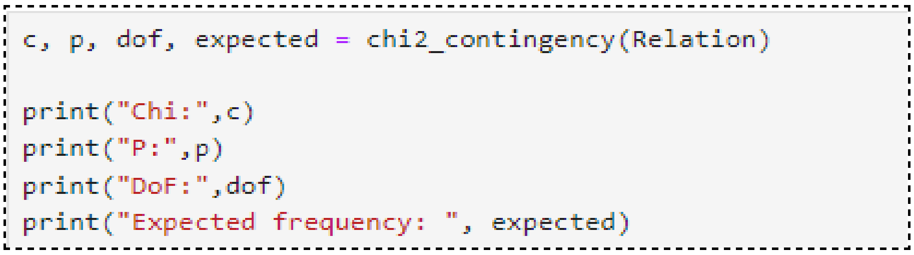
```
#Chi square test for avg_glucose_level and stroke
Relation= pd.crosstab(df1.avg_glucose_level, df1.stroke)
Relation
```



```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```



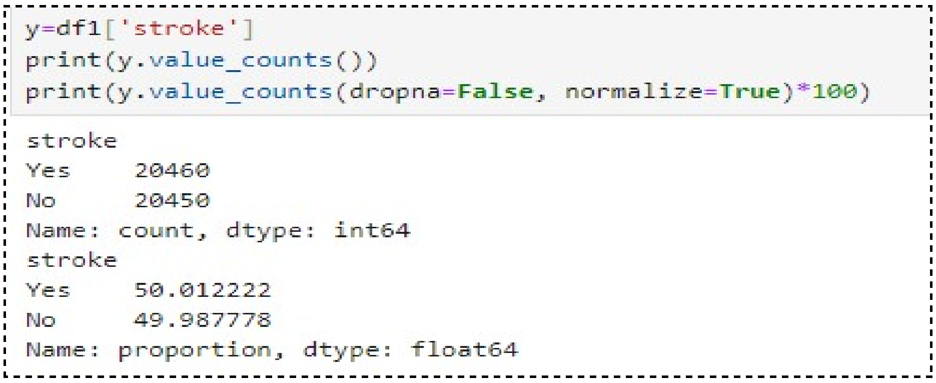
```
#Chi square test for smoking status and stroke
Relation= pd.crosstab(df1.smoking_status, df1.stroke)
Relation
```



```
c, p, dof, expected = chi2_contingency(Relation)

print("Chi:",c)
print("P:",p)
print("DoF:",dof)
print("Expected frequency: ", expected)
```

A.5 Machine Learning Classification Models

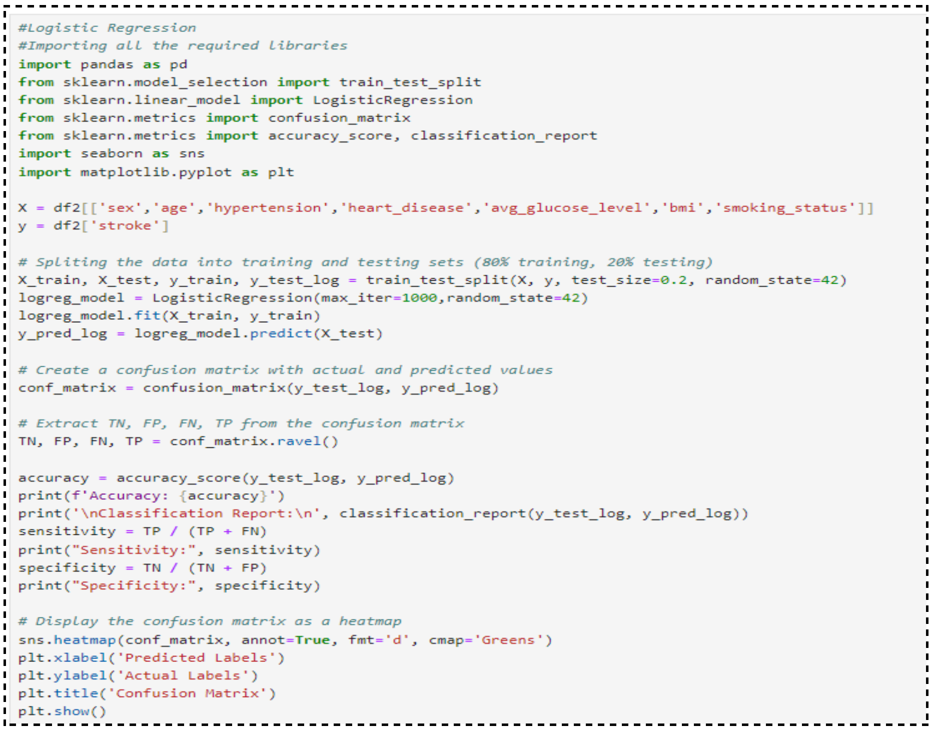
Smote



```
y=df1['stroke']
print(y.value_counts())
print(y.value_counts(dropna=False, normalize=True)*100)
```

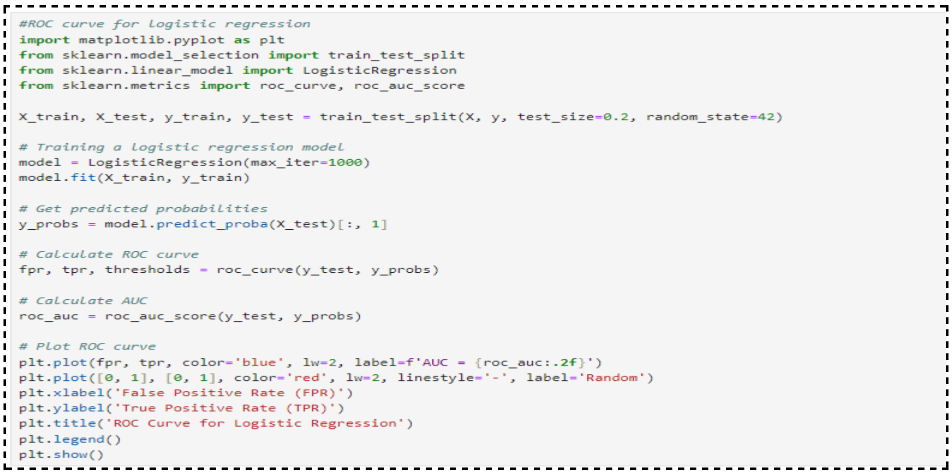
```
stroke
Yes     20460
No      20450
Name: count, dtype: int64
stroke
Yes     50.012222
No      49.987778
Name: proportion, dtype: float64
```

Logistic Regression



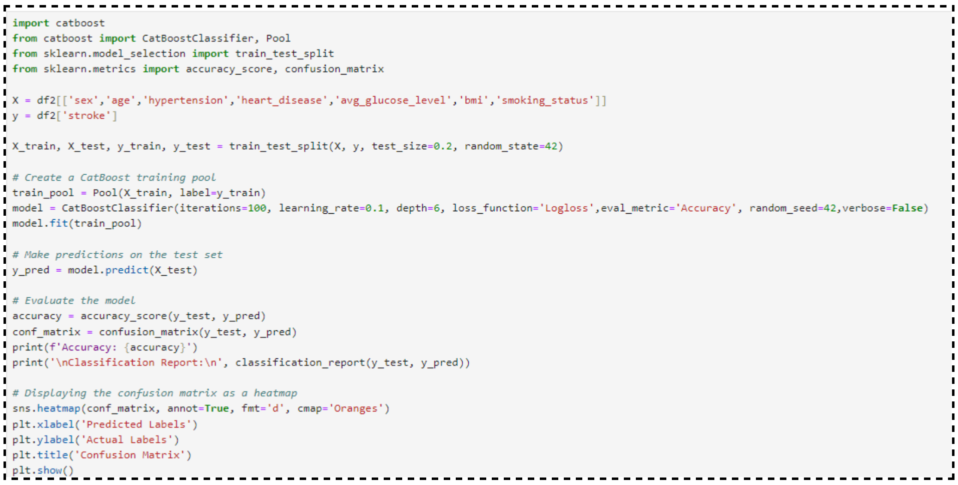
```
#Logistic Regression
#Importing all the required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']

# Spliting the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test_log = train_test_split(X, y, test_size=0.2, random_state=42)
logreg_model = LogisticRegression(max_iter=1000,random_state=42)
logreg_model.fit(X_train, y_train)
y_pred_log = logreg_model.predict(X_test)

# Create a confusion matrix with actual and predicted values
conf_matrix = confusion_matrix(y_test_log, y_pred_log)

# Extract TN, FP, FN, TP from the confusion matrix
TN, FP, FN, TP = conf_matrix.ravel()

accuracy = accuracy_score(y_test_log, y_pred_log)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', classification_report(y_test_log, y_pred_log))
sensitivity = TP / (TP + FN)
print("Sensitivity:", sensitivity)
specificity = TN / (TN + FP)
print("Specificity:", specificity)

# Display the confusion matrix as a heatmap
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Greens')
plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix')
plt.show()
```

ROC curve for Logistic Regression



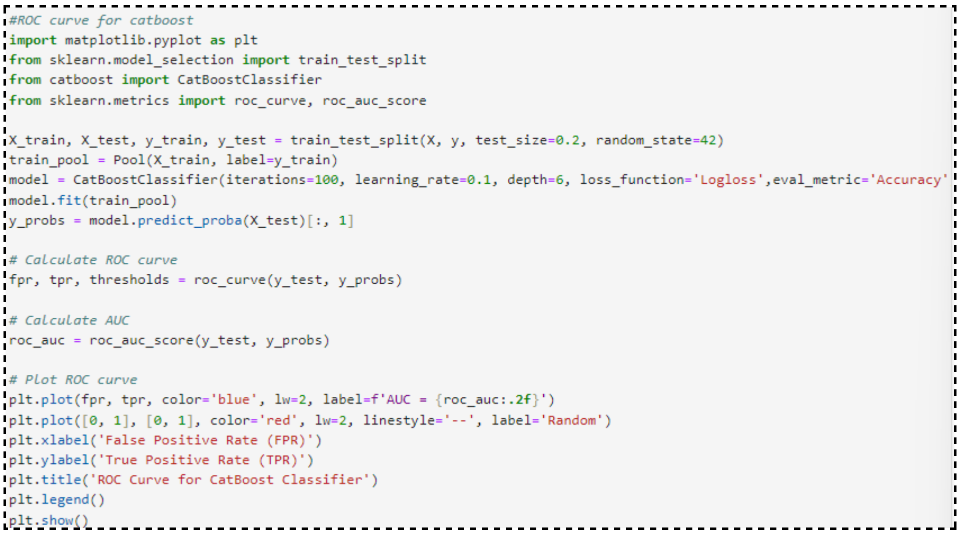
```
#ROC curve for logistic regression
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Training a logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Get predicted probabilities
y_probs = model.predict_proba(X_test)[:, 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_probs)

# Calculate AUC
roc_auc = roc_auc_score(y_test, y_probs)

# Plot ROC curve
plt.plot(fpr, tpr, color='blue', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='-', label='Random')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for Logistic Regression')
plt.legend()
plt.show()
```

Catboost Classifier



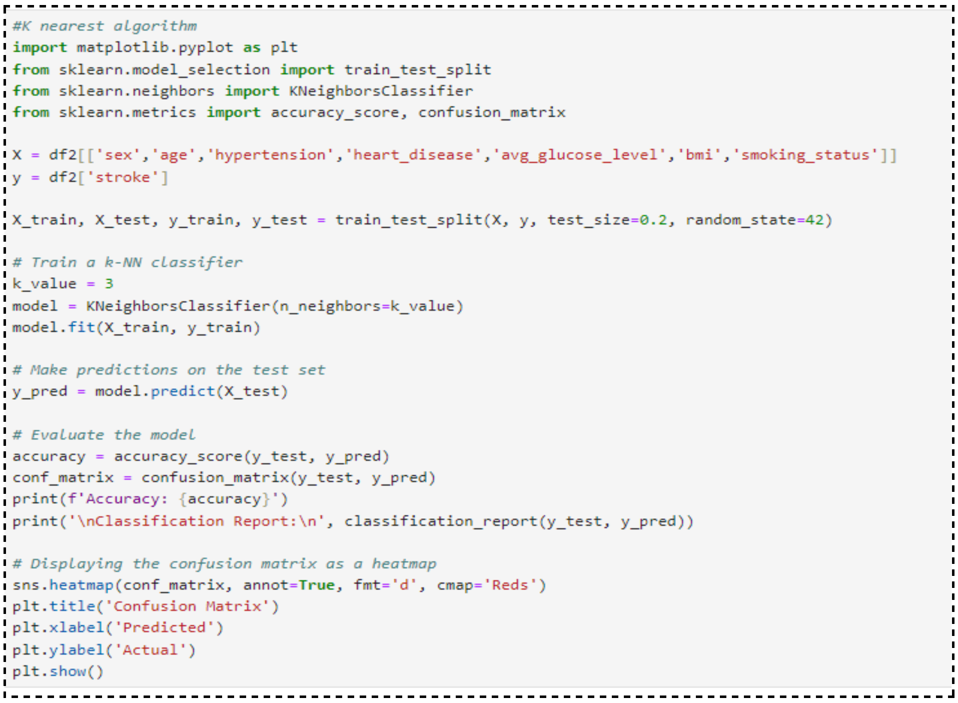
```python
import catboost
from catboost import CatBoostClassifier, Pool
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a CatBoost training pool
train_pool = Pool(X_train, label=y_train)
model = CatBoostClassifier(iterations=100, learning_rate=0.1, depth=6, loss_function='Logloss',eval_metric='Accuracy', random_seed=42,verbose=False)
model.fit(train_pool)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', classification_report(y_test, y_pred))

# Displaying the confusion matrix as a heatmap
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Oranges')
plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix')
plt.show()
```

ROC curve



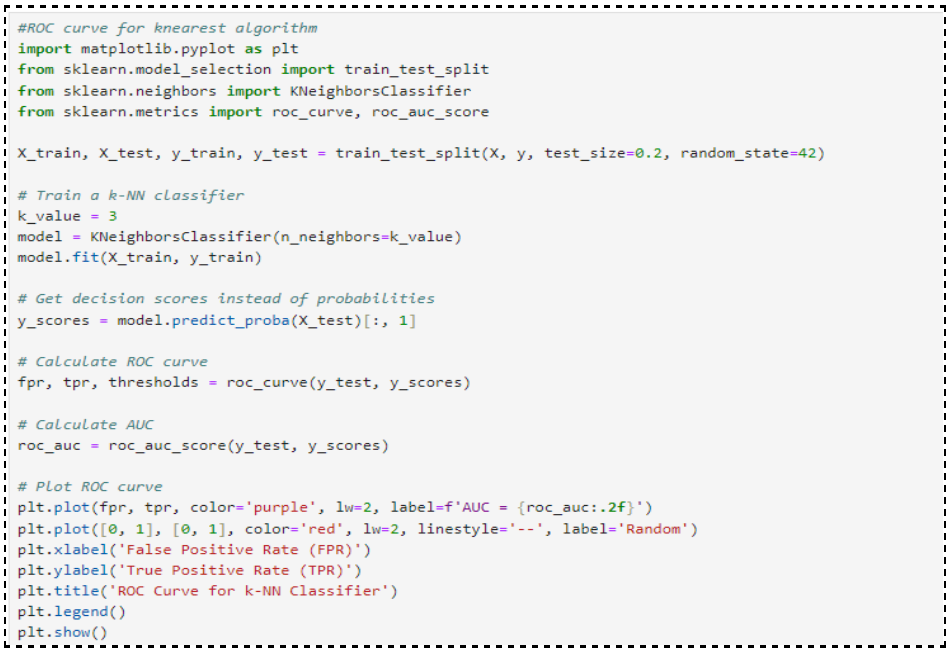
```python
#ROC curve for catboost
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from catboost import CatBoostClassifier
from sklearn.metrics import roc_curve, roc_auc_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
train_pool = Pool(X_train, label=y_train)
model = CatBoostClassifier(iterations=100, learning_rate=0.1, depth=6, loss_function='Logloss',eval_metric='Accuracy'
model.fit(train_pool)
y_probs = model.predict_proba(X_test)[:, 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_probs)

# Calculate AUC
roc_auc = roc_auc_score(y_test, y_probs)

# Plot ROC curve
plt.plot(fpr, tpr, color='blue', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--', label='Random')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for CatBoost Classifier')
plt.legend()
plt.show()
```

KNN Algorithm



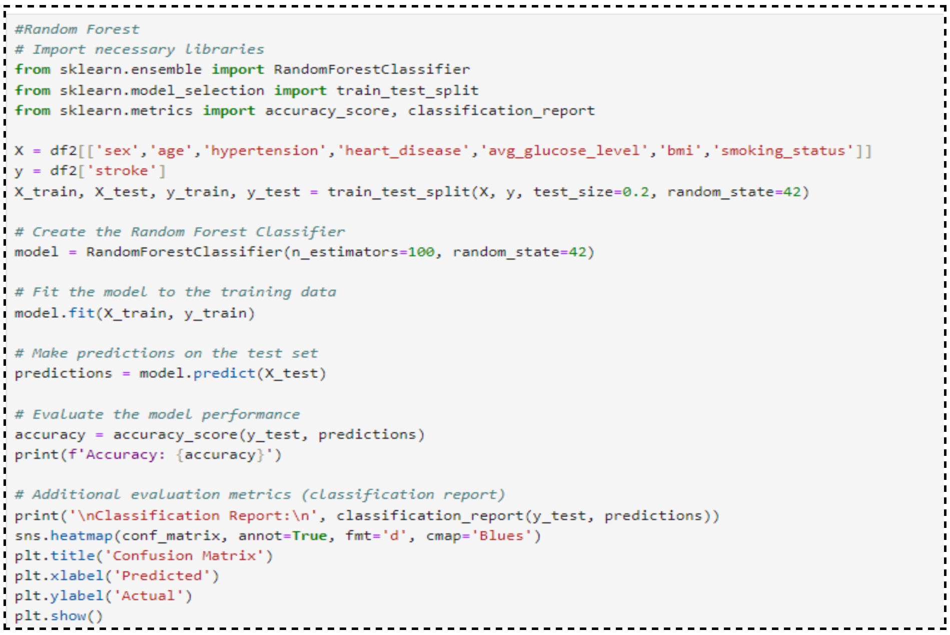
```python
#K nearest algorithm
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a k-NN classifier
k_value = 3
model = KNeighborsClassifier(n_neighbors=k_value)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', classification_report(y_test, y_pred))

# Displaying the confusion matrix as a heatmap
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Reds')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

ROC Curve



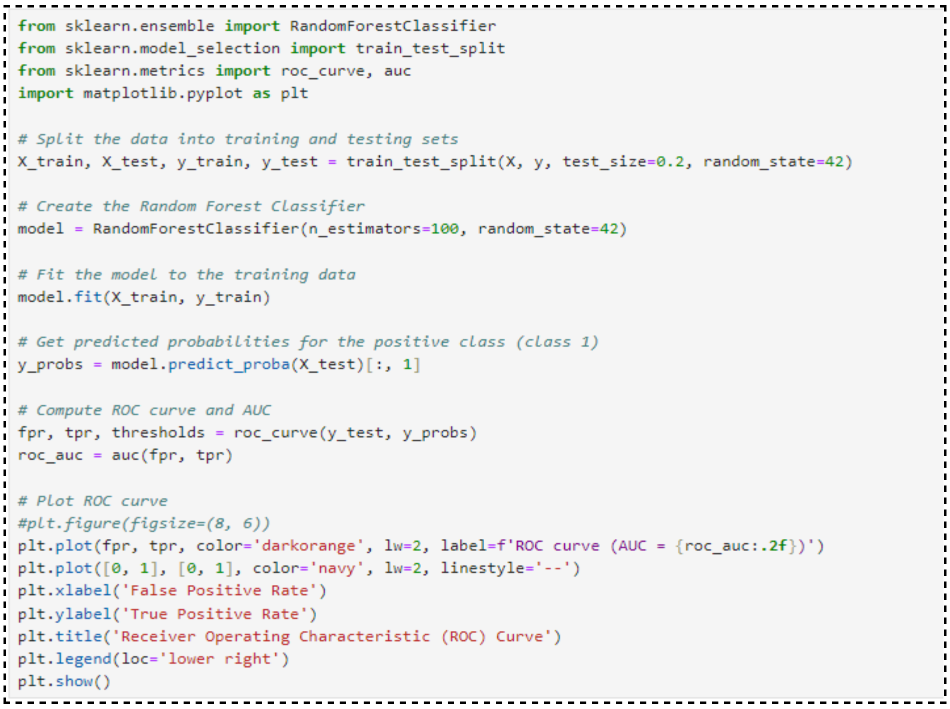
```python
#ROC curve for knearest algorithm
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_curve, roc_auc_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a k-NN classifier
k_value = 3
model = KNeighborsClassifier(n_neighbors=k_value)
model.fit(X_train, y_train)

# Get decision scores instead of probabilities
y_scores = model.predict_proba(X_test)[:, 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_scores)

# Calculate AUC
roc_auc = roc_auc_score(y_test, y_scores)

# Plot ROC curve
plt.plot(fpr, tpr, color='purple', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--', label='Random')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for k-NN Classifier')
plt.legend()
plt.show()
```

Random Forest Classifier



```
#Random Forest
# Import necessary libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test set
predictions = model.predict(X_test)

# Evaluate the model performance
accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy}')

# Additional evaluation metrics (classification report)
print('\nClassification Report:\n', classification_report(y_test, predictions))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

ROC curve



```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model to the training data
model.fit(X_train, y_train)

# Get predicted probabilities for the positive class (class 1)
y_probs = model.predict_proba(X_test)[:, 1]

# Compute ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
#plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

Cross Validation for Random Forest Classifier

```
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier

# Create the Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Specifying the features and target variable
X = df2[['sex','age','hypertension','heart_disease','avg_glucose_level','bmi','smoking_status']]
y = df2['stroke']

# Set up k-fold cross-validation
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform cross-validation and calculate AUC for each fold
cv_results = cross_val_score(model, X, y, cv=kfold, scoring='roc_auc')

# Print the AUC for each fold
for i, auc in enumerate(cv_results, 1):
    print(f'Fold {i}: AUC = {auc:.4f}')

# Print the mean AUC across all folds
print(f'Mean AUC: {cv_results.mean():.4f}')
```

ROC Curve for All Models

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from catboost import CatBoostClassifier
from sklearn.metrics import roc_curve, roc_auc_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train models
# Model 1: Logistic Regression
model_lr = LogisticRegression(max_iter=1000,random_state=42)
model_lr.fit(X_train, y_train)
#Model 2:CatBoost
model_catboost = CatBoostClassifier(iterations=100, learning_rate=0.1, depth=6, loss_function='Logloss',eval_metric='Accuracy', random_seed=42,verbose=F
model_catboost.fit(X_train, y_train)
# Model 3: k-Nearest Neighbors
model_knn = KNeighborsClassifier(n_neighbors=3)
model_knn.fit(X_train, y_train)
#Model 4: Random Forest
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
# Get predicted probabilities for the positive class
y_probs_lr = model_lr.predict_proba(X_test)[:, 1]
y_probs_catboost = model_catboost.predict_proba(X_test)[:, 1]
y_probs_knn = model_knn.predict_proba(X_test)[:, 1]
y_probs_rf = model.predict_proba(X_test)[:, 1]
# Calculate ROC curves and AUCs
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_probs_lr)
fpr_catboost, tpr_catboost, _ = roc_curve(y_test, y_probs_catboost)
fpr_knn, tpr_knn, _ = roc_curve(y_test, y_probs_knn)
fpr_rf, tpr_rf, thresholds_rf = roc_curve(y_test, y_probs_rf)
roc_auc_lr = roc_auc_score(y_test, y_probs_lr)
roc_auc_catboost = roc_auc_score(y_test, y_probs_catboost)
roc_auc_knn = roc_auc_score(y_test, y_probs_knn)
roc_auc_rf = roc_auc_score(y_test, y_probs_rf)
# Plot ROC curves
plt.plot(fpr_lr, tpr_lr, color='blue', lw=2, label=f'Logistic Regression (AUC = {roc_auc_lr:.2f})')
plt.plot(fpr_catboost, tpr_catboost, color='green', lw=2, label=f'CatBoost (AUC = {roc_auc_catboost:.2f})')
plt.plot(fpr_knn, tpr_knn, color='purple', lw=2, label=f'KNN (AUC = {roc_auc_knn:.2f})')
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'Random Forest (AUC = {roc_auc_rf:.2f})')
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--', label='Random')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curves for Logistic Regression, CatBoost, Random Forest and KNN')
```

References

Volkow, N. D., Koob, G. F., & McLellan, A. T. (2016). Neurobiologic Advances from the Brain Disease Model of Addiction. *The New England Journal of Medicine*, *374*(4), 363–371. https://doi.org/10.1056/nejmra1511480

Diabetes, hypertension and stroke prediction. (2022, December 19). Kaggle. https://www.kaggle.com/datasets/prosperchuks/health-dataset

Alloubani, A., Saleh, A., & Abdelhafiz, I. (2018). Hypertension and diabetes mellitus as a predictive risk factor for stroke. *Diabetes & Metabolic Syndrome: Clinical Research and Reviews*, *12*(4), 577–584. https://doi.org/10.1016/j.dsx.2018.03.009