“Group 4: Relationship among risk factors for myocardial infarction”

“age, gender, hypertension, diabetes, and obesity,”

[1]Indiana University-Purdue University, Indianapolis - IUPUI, IN 46202, USA
anikandi@iu.edu , Mshuddle@iu.edu , skadiya@iu.edu , tchattu@iu.edu , kpancho@iu.edu , vmatlaku@iu.edu

Abstract - The purpose of this study is to investigate the association between ‘age, gender, diabetes, hypertension, and obesity and the risk of myocardial infarction’ in a patient dataset(1992-1995). The Database will be used to determine risk variables associated with myocardial infarction complications. This database covers patient demographics and health-related characteristics. The goal of the research is to better understand how these risk factors influence the likelihood of myocardial infarction and to create preventative strategies to enhance patient outcomes. The study's findings can help healthcare practitioners identify high-risk patients and apply tailored medications and lifestyle adjustments to reduce the risk of myocardial infarction and its complications.

Keywords. Myocardial infarction, risk factor, mortality rate, gender, diabetes, hypertension, obesity, healthcare physicians, data analysis & visualization, python

“1 Project Scope”

1.1 Introduction

Myocardial infarction (MI) is a dangerous health problem with known risk features including ‘age, gender, hypertension, diabetes, and obesity’. The linkages between these risk variables and MI, on the other hand, are not entirely known. Following an MI, patients may experience additional cardiovascular complications, highlighting the importance of enhanced risk stratification and management.

The Myocardial Infarction Complications Database, created by Golovenkin et al. (2020) to fill this knowledge gap, contains data on patient demographics and health-related factors such as age, gender, hypertension, diabetes, and obesity. By using the database to identify risk factors linked to MI complications, better preventative and management techniques can be developed. According to Wilson et al. (2021) and Hennekens et al. (2022), the Database is a useful resource for identifying risk variables linked to MI complications and can help in the creation of efficient prevention and management methods.

1.2 “Aim”

“To examine how age, gender, diabetes, hypertension, obesity, and the likelihood of myocardial infarctions in a patient dataset relate to one another”.

Research question 1: How do age, gender, diabetes, hypertension, and obesity contribute to the likelihood of myocardial infarction in a patient dataset?

Research question 2: Comparing two machine learning models which model has good accuracy?

Two hypotheses

- “Age, gender, diabetes, hypertension, and obesity do not significantly affect the risk of myocardial infarction in the patient population, according to the null hypothesis.”
- “Alternate Hypothesis: The patient population's risk of myocardial infarction is significantly influenced by age, gender, diabetes, hypertension, and obesity.”

1.3 “Purpose”

The study's purpose is to investigate the association between various risk variables and the likelihood of myocardial infarction in our patient population. Myocardial infarction, often known as a heart attack, is a leading reason of death and various “risk factors, including age, hypertension, sex, diabetes, and obesity,” have been found. While the significance of these risk factors is well established, their interactions and contributions to the development of myocardial infarction remain unknown.

As a result, by analyzing the prevalence of these risk variables in our dataset, we intend to obtain a good understanding of how they influence the chance of myocardial infarction in patients. This knowledge can assist healthcare practitioners in identifying high-risk patients and implementing preventative interventions such as lifestyle changes and targeted medicines to lower the occurrence and severity of myocardial infarction and its associated problems. The ultimate aim of this project is to improve patient outcomes, which we hope to accomplish by improving our ability to recognize and minimize the risk of myocardial infarction.

2 Methodology

2.1 "Steps of the Project"

It focuses on analyzing the risk factors associated with myocardial infarction (MI) mortality rates using machine learning algorithms like KNN and logistic regression. We have utilized tools including Python Jupyter Notebook and various Python libraries like sci-kit-learn and pandas for data analysis.

The project methodology involves four stages:

1. Data Collection: Gathering relevant data related to risk factors and mortality rates of MI.
2. Data Preprocessing: Cleaning and preparing the data for analysis, handling missing values and removing duplicates.
3. Data Analysis: Applying machine learning algorithms like KNN and logistic regression to identify correlations between the risk factors and MI mortality rates.
4. Model Evaluation: Comparing the accuracy of KNN and logistic regression models and choosing the best-performing model for predicting the risk of MI mortality.

By analyzing the risk factors associated with MI mortality rates using machine learning algorithms, we aim to provide insights that could be used to develop effective preventive measures and treatment strategies. Our project seeks to utilize data preprocessing, analysis, and model evaluation techniques to achieve this goal.

2.2 **"Original Team Members and Responsibilities"**

Members are from different majors. The below-listed responsibilities, we agreed at the initial time of the project.

Name	**Background**	**Responsibilities**
Thammi Babu Chattu	B Pharm, experience in python	Data Collection and Preprocessing
Monica Huddleston	PharmD, BCPS, Beginner in SQL, new to Python and machine learning.	Data Collection and Preprocessing
Srilekha Kadiyala	BDS, Experience with python Intermediate level Machine Learning	Data Duplication and Data Visualization
Anil Kandimamalla	DVM (Doctor of Veterinary Medicine), experience in SQL, Intermediated in Python	Python coding and Machine Learning integration
Kushal Pancholi	BDS (Bachelor of Dental Surgeon), Beginner in SQL, Experience in Python, Intermediated in Machine Learning	Python coding and Machine Learning integration
Vahini Matlakunta	UG-Biotechnology Experience in Python	Data Visualization and Analysis Results

"Fig. 1." "Responsibilities of Team Members for This Project"

“2.3 Real Contributions from Individual Team Members”

At the start, we divided a task to maintain work balance, however, we have to rearrange it according to the project went on. The project required expertise in each task to get correct & accurate outcomes.

Name	Background	Responsibilities
Thammi Babu Chattu	B Pharm, experience in python	Data Collection, Data Cleaning, and Data Duplication
Monica Huddleston	PharmD, BCPS, Beginner in SQL, new to Python and machine learning.	Data Collection and Project Presentation
Srilekha Kadiyala	BDS, Experience with python Intermediate level Machine Learning	Data Duplication and Data Visualization, Proofreading
Anil Kandimamalla	DVM (Doctor of Veterinary Medicine), experience in SQL, Intermediated in Python	Python coding, Data Visualization, Proofreading
Kushal Pancholi	BDS (Bachelor of Dental Surgeon), Beginner in SQL, Experience in Python, Intermediated in Machine Learning	Preprocessing, Data Visualization, Python coding, Machine Learning integration, Project Report
Vahini Matlakunta	UG-Biotechnology Experience in Python	Analysis Results and Project Presentation

“Fig. 2. Revised Responsibilities of Team Members for This Project”

“2.4 Project Challenges”

It may face several challenges, including Data quality and availability: The accuracy of our analysis and models is heavily reliant on the quality and availability of the data. Incomplete, inaccurate,e or outdated data could negatively impact our results.

Data preprocessing: Preprocessing large amounts of data can be time-consuming and challenging. It requires careful handling
of missing values, duplicates, and outliers to ensure the accuracy and reliability of the analysis.

Algorithm selection: Choosing the most appropriate machine learning algorithm for our data can be a challenging task. Different algorithms have their own strengths and weaknesses, and selecting the best one for our data requires careful consideration.

Overfitting and underfitting: Our models must be accurate and reliable but not overfit or underfit the data. When a model is overly complicated and does on training data however poorly on testing data (overfitting), When a model is overly straightforward and performs badly on both the training and test sets of data(underfitting).

Interpretation of results: The interpretation of results can be subjective and requires condition of different factors, including the size of the dataset, the choice of algorithm, and the accuracy of the model. It is crucial to results are meaningful and may be used to make informed decisions.

Overall, our project team will need to be vigilant and diligent in addressing these challenges to ensure the accuracy and reliability of our analysis and models.

3 “Data Collection”

The dataset was taken from the University of Leicester.
‘https://leicester.figshare.com/articles/dataset/Myocardial_infarction_complications_Database/12045261/3’

The important health indicators for local communities are provided by this dataset, which also promotes discussion about possible community health improvement measures.

The dataset consists of 1 csv file with raw data. The file is named:

1. “Myocardial infarction complications Database.csv”

4 Data Extraction and Storage

“4.1 Data Extraction”

The data extraction steps involved in our project on analyzing the risk factors associated with myocardial infarction (MI) mortality rates are as follows:

Identify data sources: We need to identify the relevant data sources that provide information on MI risk factors and mortality rates. These sources could include medical records, research studies, public health databases, and government statistics.

Collect data: Once we have identified the relevant data sources, we need to collect the data. This involves accessing the data sources and retrieving the relevant data in a format that can be easily analyzed. This may involve downloading data from databases or scraping data from websites.

Pre-process data: The collected data may be in a raw format that is not suitable for analysis. The data needs to be cleaned, changed and put into an easily-analyzed format as part of the pre-processing step. It may entail ignoring duplicates, dealing with missing values, and formatting data in a numerical manner.

Store data: We need to store the pre-processed data in a format that is suitable for further analysis. This could involve storing the data in a database, a CSV file, or other suitable file formats.

Verify data accuracy: We need to verify the accuracy of the data by checking it against other sources or by cross-checking it with known facts. It helps to check that the data we have collected is accurate and reliable.

Dataset column description are:

- “Gender (**SEX**): 0 – female 1 – male”
- “Obesity (**endocr_02**): 0 – no 1 – yes”
  “GB - Hypertension 0 – no 1 – Stage 1 2 – Stage 2 3 – Stage-3”
- “SIM_GIPERT - Hypertension 0 – absent 1 – present”
- “DLIT_AG - Arterial Hypertension 0 –absent 1 – 1yr 2 – 2 yr 3 – 3 yr 4 – 4 yr 5 – 5 yr 6 – 6-10 yr 7 –10+ yr”
- “endocr_01 **-** Diabetes : 0 – absent 1 – present”
- “INF_ANAM myocardial infarctions: 0 – zero 1 – one 2 – two 3 – three and more”
- “ant_im Presence of an anterior MI 0 –absent infarct 1 – no changes 2 – QR-complex 3 – QRS - Qr-complex 4 – QS-complex”
- “lat_im Presence of a lateral MI 0 –absent 1 – no changes 2 – QR-complex 3 – QRS - Qr-complex 4 – QRS - QS-complex”
- ‘inf_im Presence of an inferior MI 0 – no 1 – no changes 2 – QRS - QR-complex 3 – QRS - Qr-complex 4 – QRS - QS-complex”
- “post_im Presence of a posterior MI : 0 – no infarct 1 – QRS no changes 2 – QRS - QR-complex 3 – QRS - Qr-complex 4 – QRS - QS-complex”
- “IM_PG_P Presence of a right MI 0 – absent 1 – present”
- “IM_PG_P lethal – unknown 1 – cardiogenic shock 2 – pulmonary edema 3 – myocardial rupture 4 congestive heart failure 5 – thromboembolism 6 – asystole 7 – ventricular fibrillation”

4.2 “Data Cleaning”

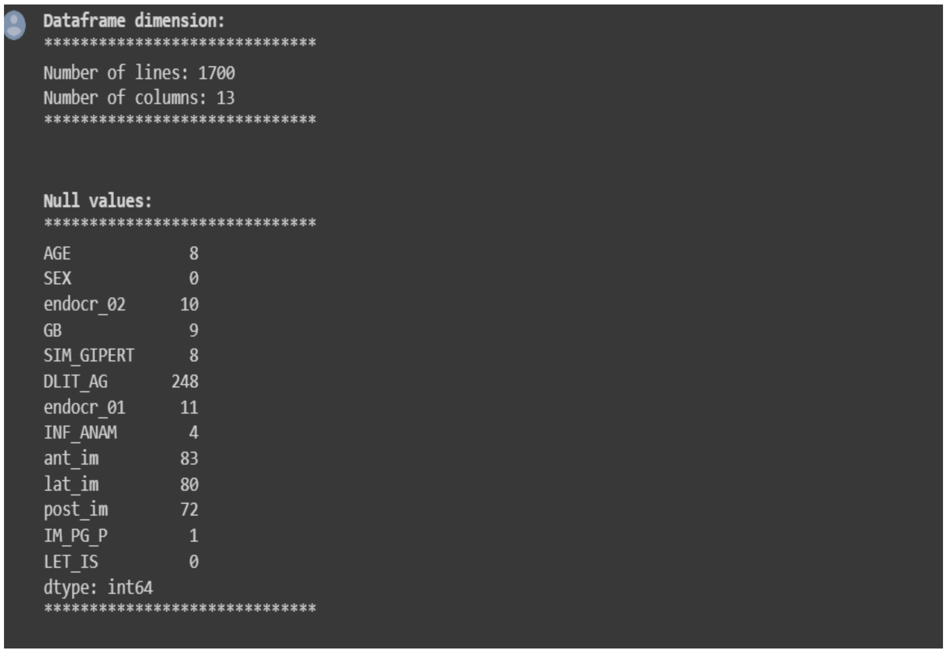
Data frame dimension and null values: The first code snippet provides the dimensions of the Data Frame (number of rows and columns) and identifies the number of missing values in each column. Understanding the accuracy and completeness of the data requires doing this step. It could be essential to impute the missing data or eliminate the entire column if there are several missing values in any given column.

Information in the data frame and removing missing values: The second code snippet gives further details about the Data Frame, notably the categories of data in each column and the amount of memory used. The "dropna" function is then used to remove any rows with missing values from the Data Frame. This is a common data cleaning step as missing values can skew analysis results.

Checking for null values: After dropping missing values, the "isna().sum()" function is used again to ensure that there are no remaining missing values in the DataFrame. This is important as missing values can cause errors in subsequent analysis steps.

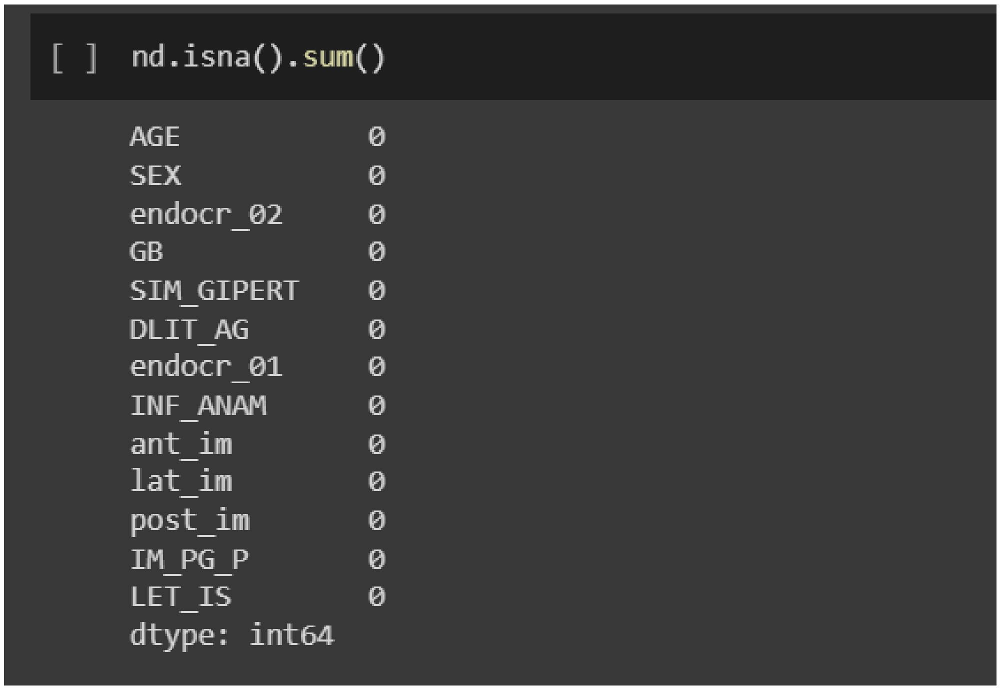
Descriptive statistics: The final code snippet provides descriptive statistics for the cleaned DataFrame, including measures of central tendency and variability for numerical columns. This step can help identify any outliers or unusual values in the data, which may need to be investigated further.

Overall, these code steps demonstrate common data cleaning techniques such as identifying missing values, dropping missing values, and checking for null values. These steps are important to ensure that the data is suitable for further analysis and visualization.



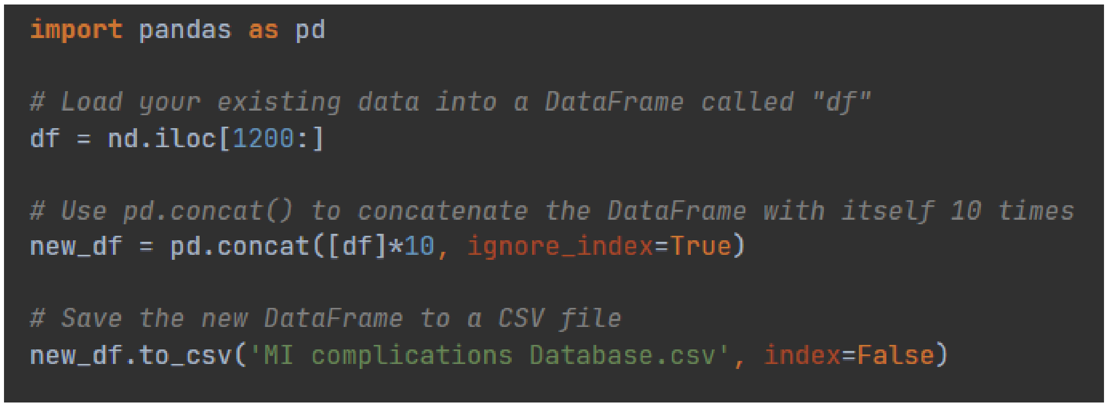
```
Dataframe dimension:
*****************************
Number of lines: 1700
Number of columns: 13
*****************************


Null values:
*****************************
AGE             8
SEX             0
endocr_02      10
GB              9
SIM_GIPERT      8
DLIT_AG       248
endocr_01      11
INF_ANAM        4
ant_im         83
lat_im         80
post_im        72
IM_PG_P         1
LET_IS          0
dtype: int64
*****************************
```

‘Fig. 3’ “Null (Negative) Values”



```
[ ]  nd.isna().sum()

     AGE           0
     SEX           0
     endocr_02     0
     GB            0
     SIM_GIPERT    0
     DLIT_AG       0
     endocr_01     0
     INF_ANAM      0
     ant_im        0
     lat_im        0
     post_im       0
     IM_PG_P       0
     LET_IS        0
     dtype: int64
```

“Fig. 4.” “Cleaned Data Without Negative and Null Values”



```python
import pandas as pd

# Load your existing data into a DataFrame called "df"
df = nd.iloc[1200:]

# Use pd.concat() to concatenate the DataFrame with itself 10 times
new_df = pd.concat([df]*10, ignore_index=True)

# Save the new DataFrame to a CSV file
new_df.to_csv('MI complications Database.csv', index=False)
```

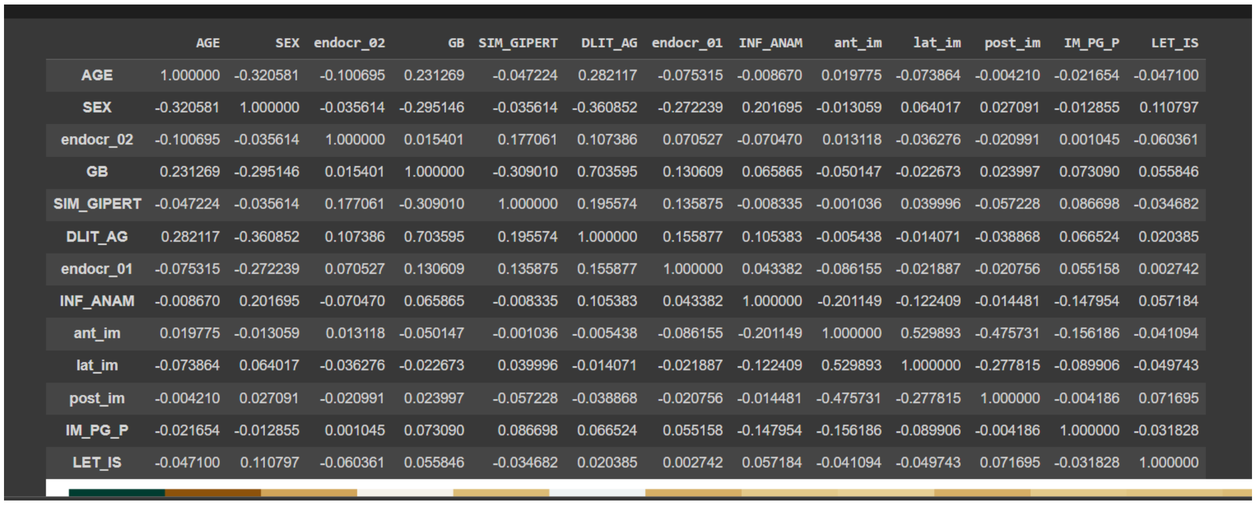
“Fig. 5.” “Duplicate data”

4.3 “Heatmap”

Descriptive statistics for a column: The first line of code provides descriptive statistics for the 'endocr_02' column in the DataFrame. This comprises measurements of dispersion (standard deviation, lowest and highest values, and quartiles) as well as a central tendency (mean, median). It helps for understanding the distribution of values in the column, which may be important for subsequent analysis.

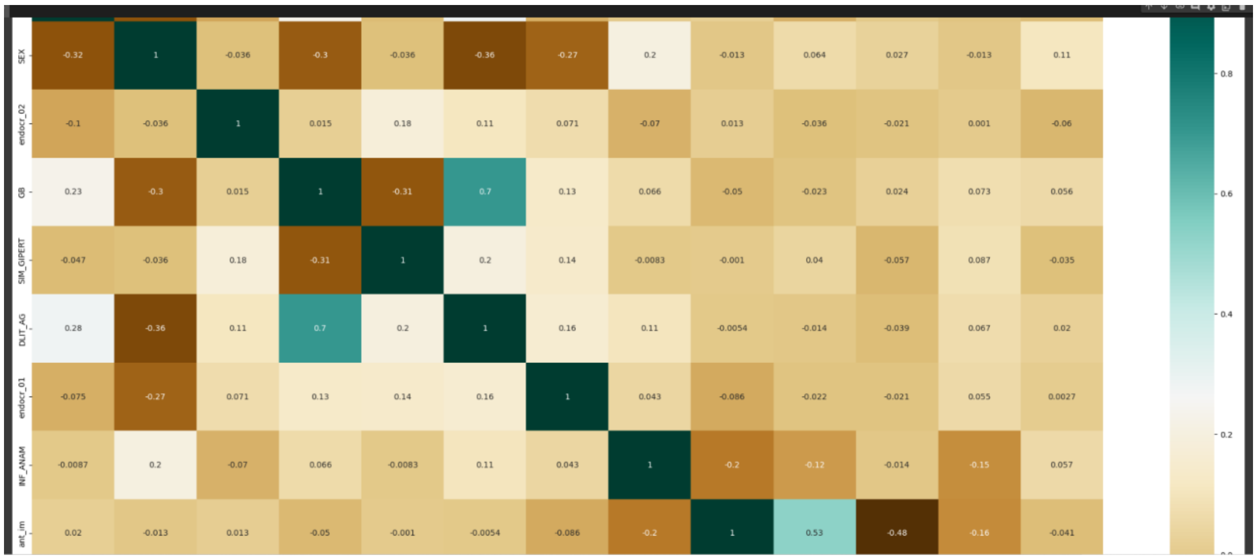
- Correlation matrix and heatmap: The second block of code creates a correlation matrix for all the numerical columns in the DataFrame using the "corr" function. A measurement of the “linear relationship” between 2 variables is given by this matrix, which “ranges from 1-perfectly negative to 1-perfectly positive.”
- The correlation matrix is then visualized as a heatmap using the "BrBG" colormap using the "sns. heatmap" function. Darker colors denote greater correlations, making it simple to see the strength and direction of correlations between pairwise variables on the heatmap. Coefficients are added when the” annot=True parameter is set to true.

Overall, this code snippet demonstrates techniques for exploring and visualizing relationships between variables in the dataset, which can help identify important predictors of myocardial infarction mortality rates.



	AGE	SEX	endocr_02	GB	SIM_GIPERT	DLIT_AG	endocr_01	INF_ANAM	ant_im	lat_im	post_im	IM_PG_P	LET_IS
AGE	1.000000	-0.320581	-0.100695	0.231269	-0.047224	0.282117	-0.075315	-0.008670	0.019775	-0.073864	-0.004210	-0.021654	-0.047100
SEX	-0.320581	1.000000	-0.035614	-0.295146	-0.035614	-0.360852	-0.272239	0.201695	-0.013059	0.064017	0.027091	-0.012855	0.110797
endocr_02	-0.100695	-0.035614	1.000000	0.015401	0.177061	0.107386	0.070527	-0.070470	0.013118	-0.036276	-0.020991	0.001045	-0.060361
GB	0.231269	-0.295146	0.015401	1.000000	-0.309010	0.703595	0.130609	0.065865	-0.050147	-0.022673	0.023997	0.073090	0.055846
SIM_GIPERT	-0.047224	-0.035614	0.177061	-0.309010	1.000000	0.195574	0.135875	-0.008335	-0.001036	0.039996	-0.057228	0.086698	-0.034682
DLIT_AG	0.282117	-0.360852	0.107386	0.703595	0.195574	1.000000	0.155877	0.105383	-0.005438	-0.014071	-0.038868	0.066524	0.020385
endocr_01	-0.075315	-0.272239	0.070527	0.130609	0.135875	0.155877	1.000000	0.043382	-0.086155	-0.021887	-0.020756	0.055158	0.002742
INF_ANAM	-0.008670	0.201695	-0.070470	0.065865	-0.008335	0.105383	0.043382	1.000000	-0.201149	-0.122409	-0.014481	-0.147954	0.057184
ant_im	0.019775	-0.013059	0.013118	-0.050147	-0.001036	-0.005438	-0.086155	-0.201149	1.000000	0.529893	-0.475731	-0.156186	-0.041094
lat_im	-0.073864	0.064017	-0.036276	-0.022673	0.039996	-0.014071	-0.021887	-0.122409	0.529893	1.000000	-0.277815	-0.089906	-0.049743
post_im	-0.004210	0.027091	-0.020991	0.023997	-0.057228	-0.038868	-0.020756	-0.014481	-0.475731	-0.277815	1.000000	-0.004186	0.071695
IM_PG_P	-0.021654	-0.012855	0.001045	0.073090	0.086698	0.066524	0.055158	-0.147954	-0.156186	-0.089906	-0.004186	1.000000	-0.031828
LET_IS	-0.047100	0.110797	-0.060361	0.055846	-0.034682	0.020385	0.002742	0.057184	-0.041094	-0.049743	0.071695	-0.031828	1.000000

“**Fig 6.** Data Frame showing data in table in correlation to the variables to depict outliers”


“**Fig 7**. Heatmap”

5 Data Analysis

“Python was used to examine the data, and linear regression was used to do so. For the data analysis, we used a prediction model regression using the k-means clustering technique. The measures we followed to efficiently evaluate the data are listed below.”

5.1 “Data Analysis”

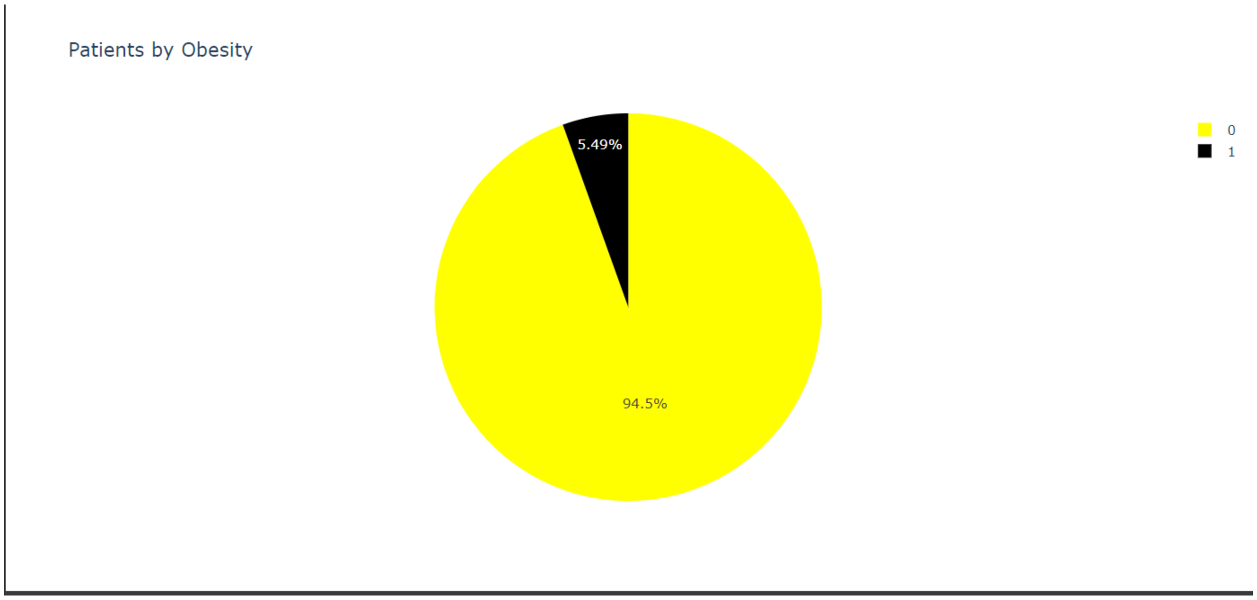
5.1.1 Patients with obesity

Import Plotly library: The first line of code imports the Plotly library and renames it as 'pyo', which will be used to generate interactive visualizations.

Data preparation: The next three lines of code extract the counts of patients by obesity status ('endocr_02') using the Pandas 'value_counts' function, and assign them to the 'labels' and 'values' variables respectively. The 'colors' variable assigns a list of colors to be used in the pie chart.

Define chart properties: The 'fig' variable defines the chart properties, including the chart type ('pie'), chart title ('Patients by Obesity'), and the direction of the pie slices ('clockwise'). The 'marker' property assigns the colors defined in the 'colors' variable to the pie slices based on their index.

Generate the chart: The final line of code uses the 'iplot' function from the 'pyo' library to generate an interactive version of the pie chart defined in the 'fig' variable.



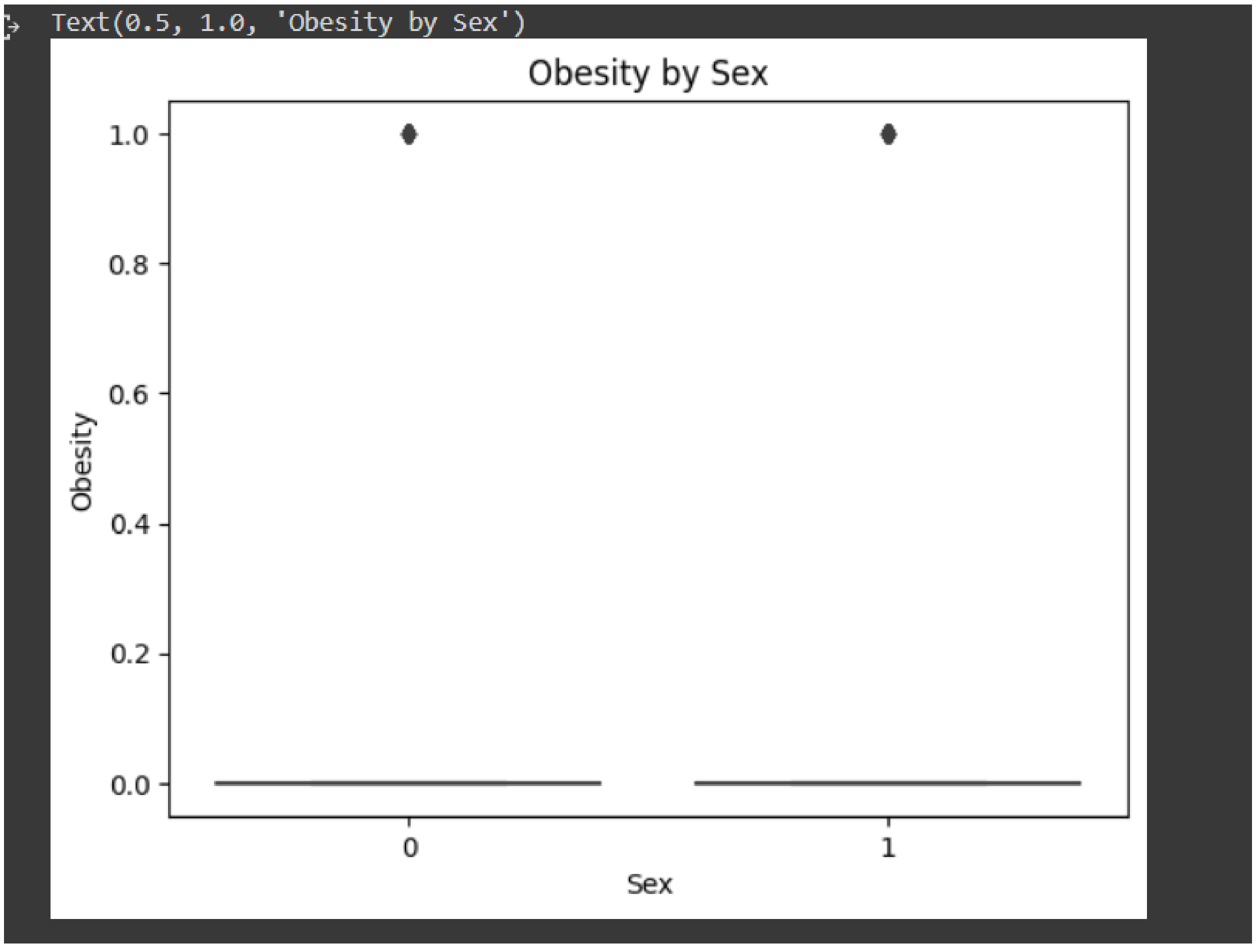
“Fig 8. Patients with obesity”

5.1.2 Patients with obesity

This code snippet generates two plots using the Seaborn library.

The first plot is a scatter plot that shows the relationship between age and the presence of obesity (indicated by the variable "endocr_02"). The horizontal a.xis represents age while the vertical axis represents the presence of obesity. Each point on the plot represents an individual from the dataset, with the position of the point indicating the individual's age and whether they have obesity.

The second plot is a box plot that shows the distribution of obesity levels (again indicated by "endocr_02") across male and female individuals. The horizontal axis represents sex, with two categories for male and female. The vertical axis represents the presence of obesity. The box plot shows the median, quartiles, and the range of the data for each sex category. The plot allows for a visual comparison of obesity levels between males and females. The title of the plot is "Obesity by Sex."

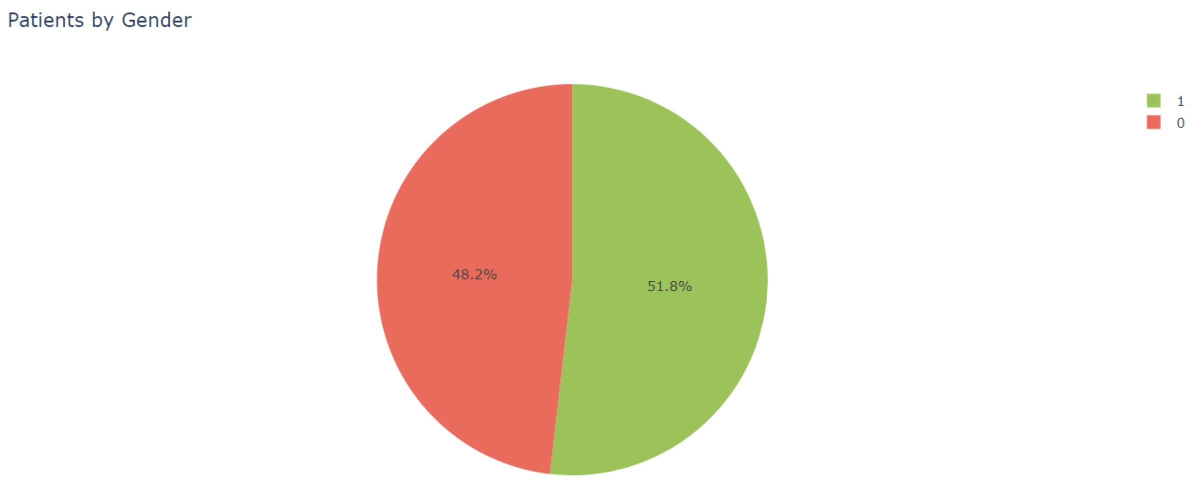


```
Text(0.5, 1.0, 'Obesity by Sex')
```

“**Fig9**.Scatterplot”

5.1.3 Patients with Gender

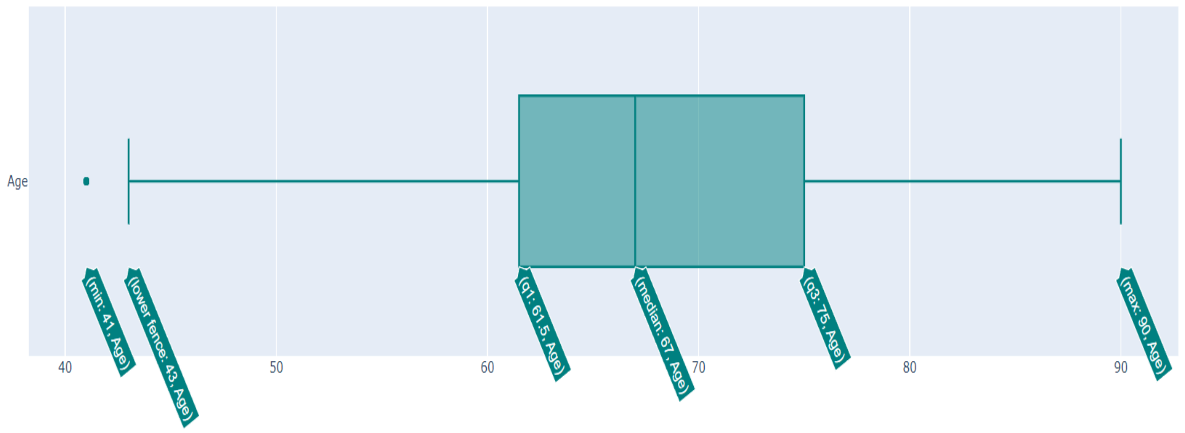
This code generates a pie chart using the Plotly library to show the distribution of patients by gender. It first extracts the gender column from the nd DataFrame and “uses the value_counts() method to count the number of” occurrences of each gender value. It then creates a dictionary with the chart specifications, including the chart type, labels, values, and colors. Finally, it uses the iplot() function from the Plotly library to display the chart in the notebook. The resulting chart shows the proportion of male and female patients in the dataset.



"**Fig10**.Scatterplot"

5.1.4 Patients with Age

This code generates a box plot using Plotly, displaying the age range of the patients in the dataset. The age data is extracted from the 'AGE' column of the cleaned dataset. The box plot visually represents the median, quartiles, and range of the data. The plot shows the distribution of ages across the patients, including any outliers. The plot is interactive, allowing the user to hover over different parts of the plot to view more detailed information. This plot could be useful in identifying any potential age-related trends or correlations in the dataset.



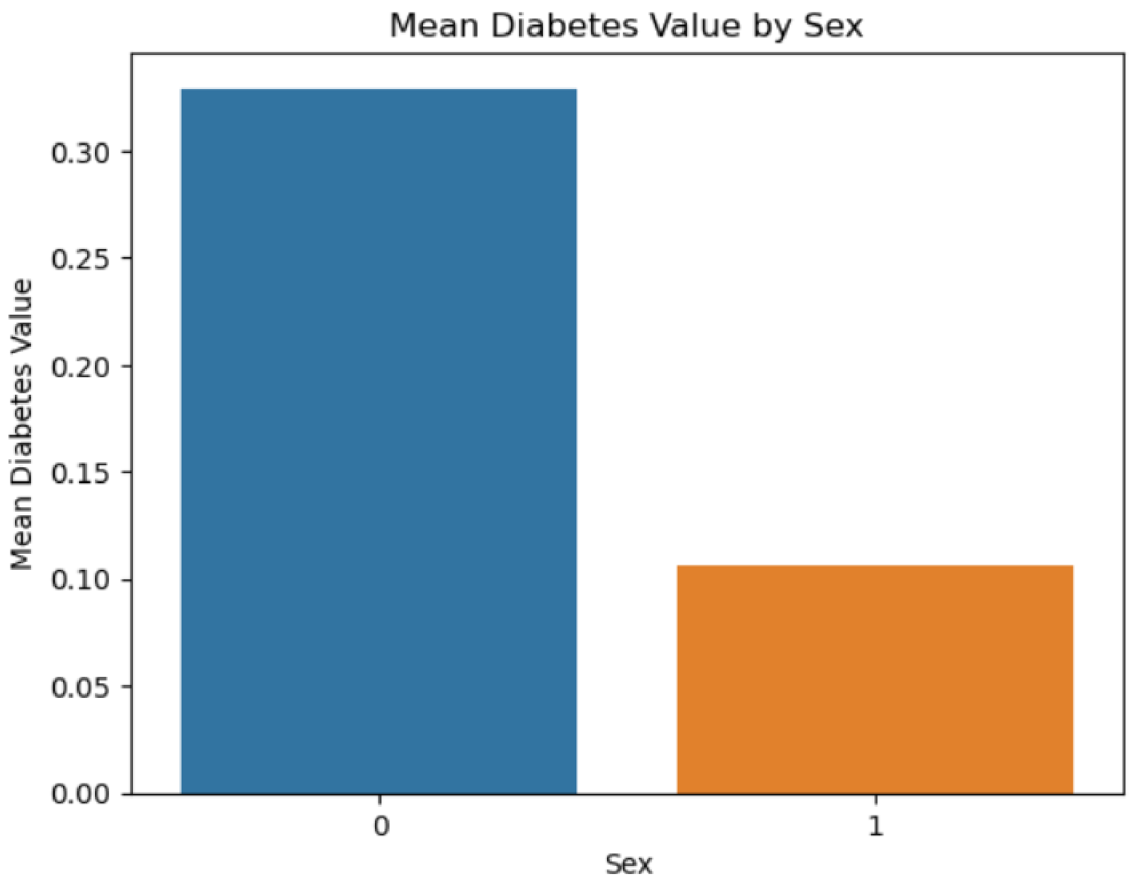
"**Fig 11.** Patients with Age"

5.1.5 Patients with Diabetes Value by Sxx

This code snippet calculates the mean value of endocr_01, which is related to diabetes, grouped by gender. It then creates a bar plot of the means, where "the x-axis shows the gender & the" y-axis shows the mean diabetes value. The color of the bars represents the confidence interval of the mean values.

This visualization shows that the mean diabetes value is higher in females than males. Therefore, gender may be a significant predictor of diabetes. However, we cannot conclude causality from this graph alone, as there may be other factors that affect the development of diabetes.

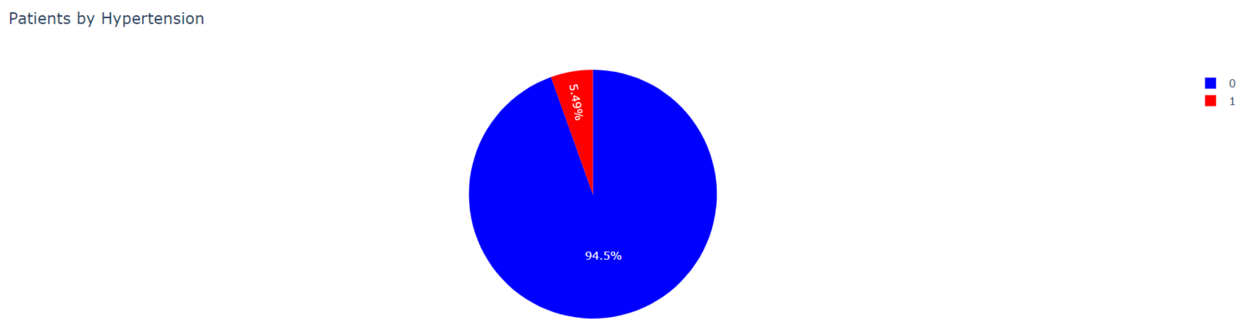
Overall, this code provides a quick and easy way to explore the relationship between gender and diabetes in the dataset.



"**Fig 12.** Mean diabetic value by Sex"

5.1.5 Patients with hypertension

First, the value counts of the "SIM_GIPERT" column in the "nd" dataset is calculated using the Pandas "value counts ()" method. This method returns a Pandas series with the count of each unique value in the column. The index of this series (i.e., the unique values of the "SIM_GIPERT" column) is stored in the "labels" variable, and values of the series (i.e., the counts of each unique value) are stored in the "values" variable. The colors for the pie chart are set to blue and red using the "colors" variable. A dictionary "fig" is created to define the data and layout of the pie chart. In the data dictionary, a list with a single dictionary is created. This dictionary contains the type of chart ("pie"), the name of the chart ("Patients by Diabetes: Pie chart"), the labels and values obtained from the "nd" dataset, the direction of the chart (clockwise), and the colors for the two segments.
In the layout dictionary, the title of the chart is set to "Patients by Hypertension".
The pie chart is then displayed using the "iplot ()" function from the Plotly library.

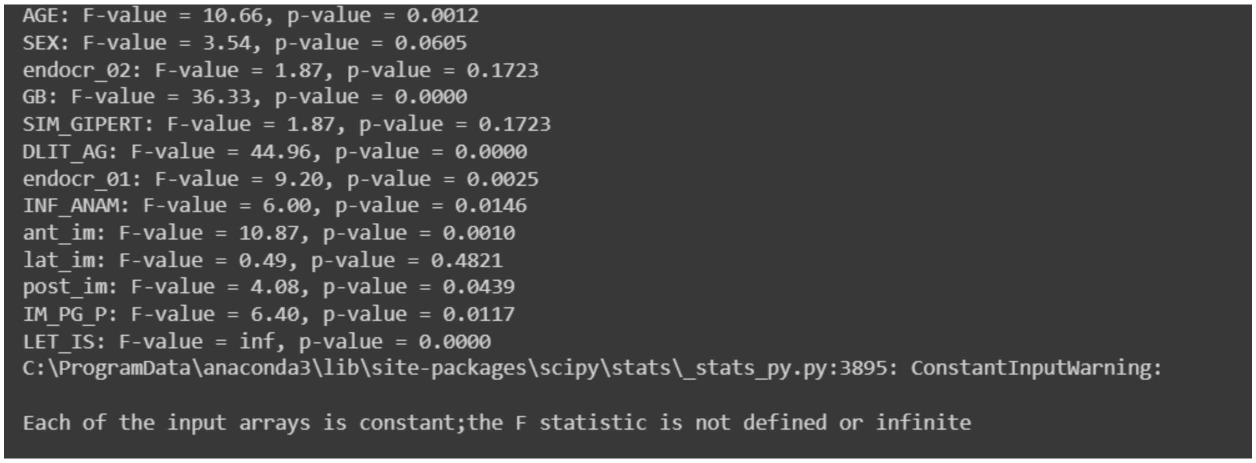


"Fig13. Patients by hypertension"

5.2 “Correlation Among Attributes”

Using the column LET_IS (a binary variable with values of 0 and 1), the ANOVA (Analysis of Variance) test is run on each numerical column of the given data frame df to determine whether there is a statistically significant difference between the two sets. When comparing the means of two or more groups, the “ANOVA” test is used to see if there are any significant differences. In a hypothesis test, the null hypothesis states that there is not a significant difference in the two groups under scrutiny. In other words, it is based on the premise that any differences seen in the sample data are the result of random variation or chance, rather than a real distinction between the groups. Usually, H0 stands for the null hypothesis.

Using the f_oneway () function from the SciPy. Stats module, the code separates the data into two groups for each numerical column based on the LET_IS column before doing an ANOVA. The ANOVA output contains the F-value and p-value. The F-value is a measure of the proportion of variation that exists within and between groups. When the F-value is higher, it means that there is a greater chance that there is a significant difference between the means of the two groups being compared. The p-value, on the other hand, represents the likelihood of obtaining the observed F-value or a more extreme value under the assumption that there is no significant difference between the means of the two groups being compared. If the p-value is lower than the significance level (usually 0.05), we reject the null hypothesis and conclude that there is a significant difference between the means of the two groups being compared.



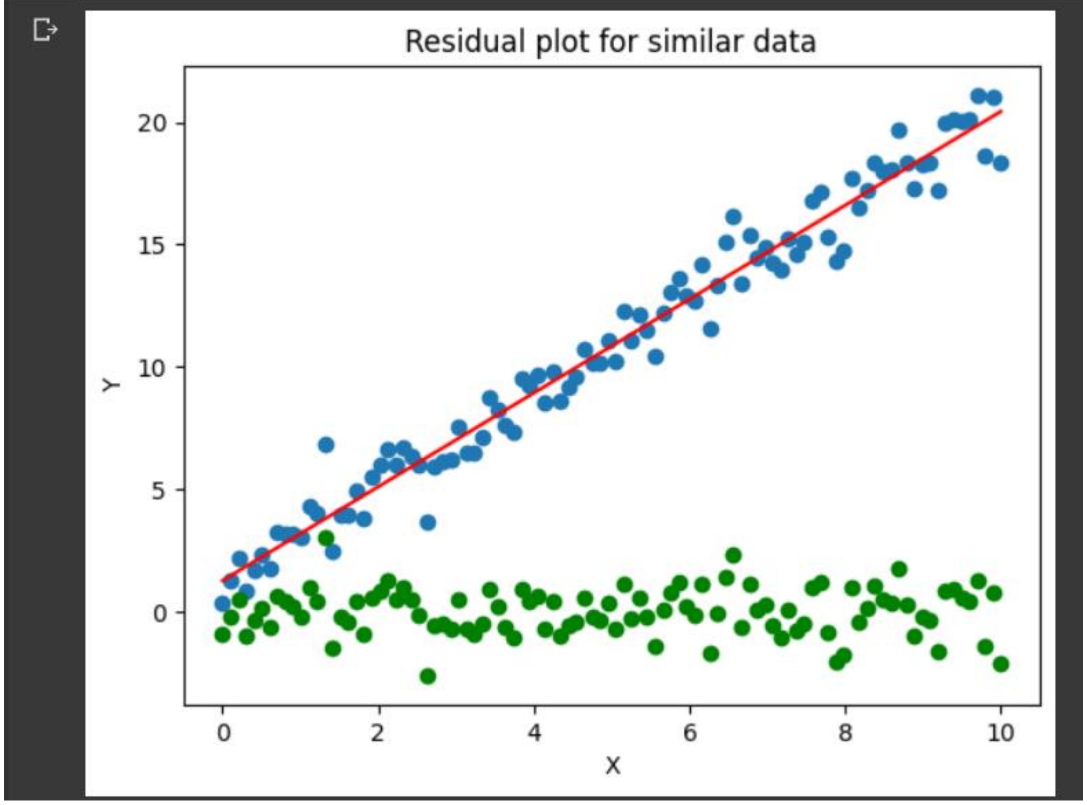
```
AGE: F-value = 10.66, p-value = 0.0012
SEX: F-value = 3.54, p-value = 0.0605
endocr_02: F-value = 1.87, p-value = 0.1723
GB: F-value = 36.33, p-value = 0.0000
SIM_GIPERT: F-value = 1.87, p-value = 0.1723
DLIT_AG: F-value = 44.96, p-value = 0.0000
endocr_01: F-value = 9.20, p-value = 0.0025
INF_ANAM: F-value = 6.00, p-value = 0.0146
ant_im: F-value = 10.87, p-value = 0.0010
lat_im: F-value = 0.49, p-value = 0.4821
post_im: F-value = 4.08, p-value = 0.0439
IM_PG_P: F-value = 6.40, p-value = 0.0117
LET_IS: F-value = inf, p-value = 0.0000
C:\ProgramData\anaconda3\lib\site-packages\scipy\stats\_stats_py.py:3895: ConstantInputWarning:

Each of the input arrays is constant;the F statistic is not defined or infinite
```

“**Fig 14**. Null Hypothesis”

There are several methods to check the analysis of variance (ANOVA) assumptions and evaluate the results:

- “Residual plots:” Residual plots can be used to visually inspect the residuals to check for equal variances and normality assumptions.
- “Levene's test:” To ascertain that variances among groups are identical, use Levene's test.
- “Bartlett's test:” While Levene's test expects the data to be normal, Bartlett's test does not.
- “QQ-plots:” QQ-plots can be used to determine whether the residuals are normally distributed.
- “Shapiro-Wilk test:” This test can be used to determine whether the residuals are normal.
- “Box-Cox transformation:” This transformation turns non-normal data into data with a normal distribution.
- “Tukey's range test:” To identify whether groups differ significantly from one another, posthoc pairwise comparisons can be performed using Tukey's range test.

These methods can be used to supplement the ANOVA and help ensure that the assumptions are met, and the results are reliable.



"Fig. 15. Analysis variance of data on Y axis"

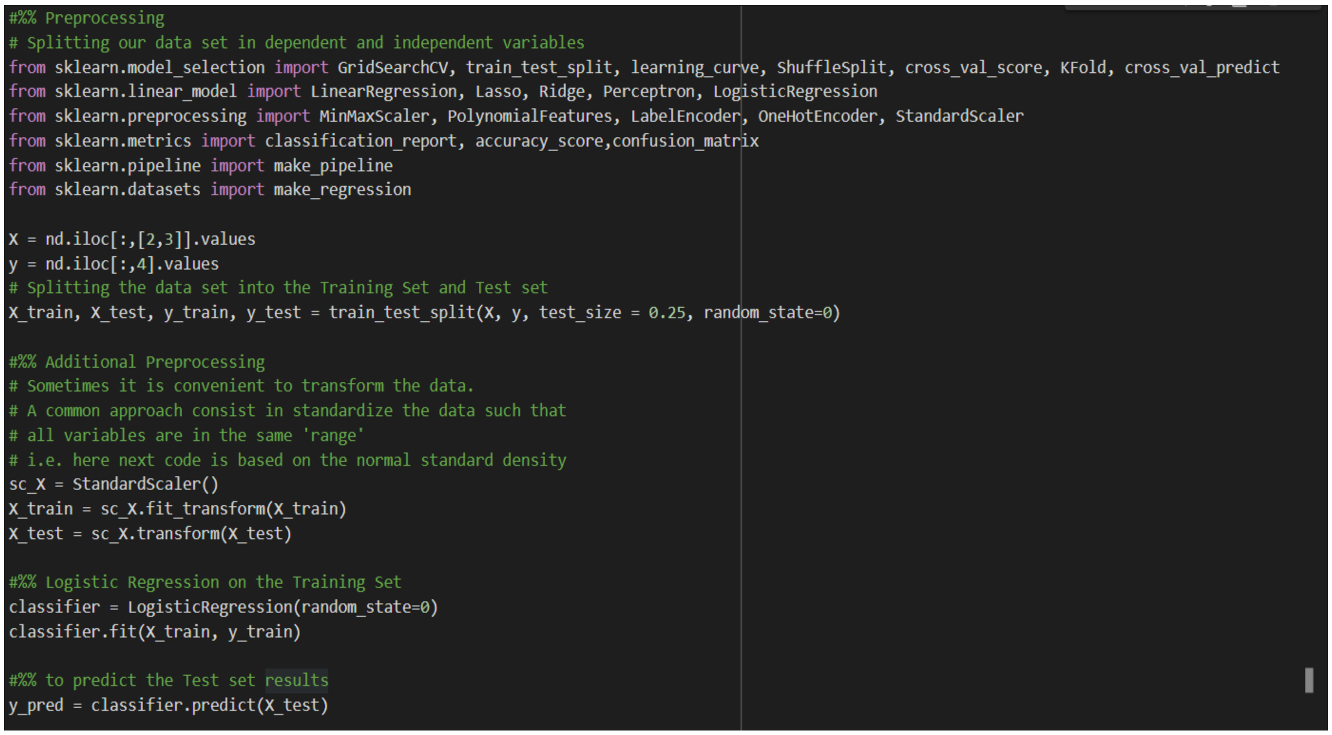
5.3 "Logistic Regression Model"

The code provided is an example of building a logistic regression model for binary classification using sci-kit-learn library in Python.

To begin with, the data preprocessing step involves splitting the dataset into independent and dependent variables. The independent variables are selected from columns 2 and 3 of the original dataset using the loc function, while the dependent variable is chosen from the fifth column. The dataset is then split into a training and a test set using the train_test_split function from the sci-kit-learn library. The logistic regression model is trained using the training set, and its performance is evaluated using the test set.

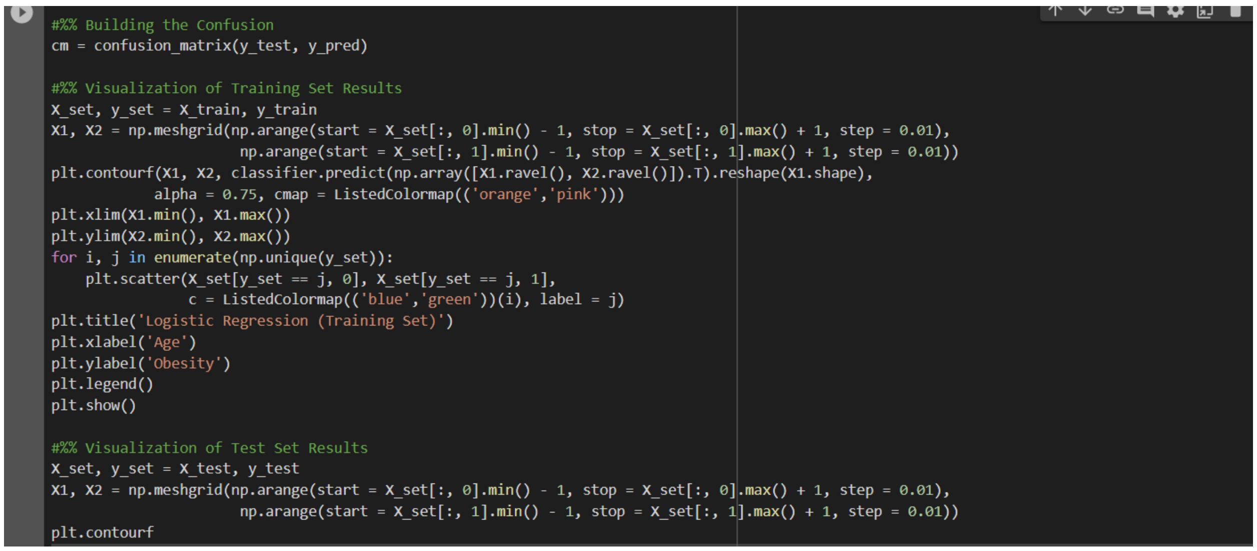
The StandardScaler function from the sci-kit-learn library is used to standardize the independent variables. Standardization helps improve the model's performance by avoiding bias towards variables with higher variance. The logistic regression model is built using the logistic regression function from the sci-kit-learn library, with the random state parameter set to 0 for repeatability. The model is trained on the training set using the fit method. Once the model is trained, it is used to predict the test set results using the prediction approach. The confusion matrix function from Scikit-Learn is used to assess the model's performance on the test set, which provides the number of true positives, true negatives, false positives, and false negatives in a table called the confusion matrix.

Finally, the results of the logistic regression model are visualized using the plot library in Python. A scatter plot is created for the training and test sets, and the contour function is used to create a filled contour plot of the predicted classes. The scatter function is used to plot the actual data points, and the color of the points corresponds to the class labels. The plot is labeled using the titlabelabel and label functions, while the legend function adds a legend to the plot. In summary, the provided code demonstrates how to train a logistic regression model in Python using the sci-kit-learn library. The model's performance is evaluated using a confusion matrix on a test set, and the findings are visualized using thematplotlibb library.



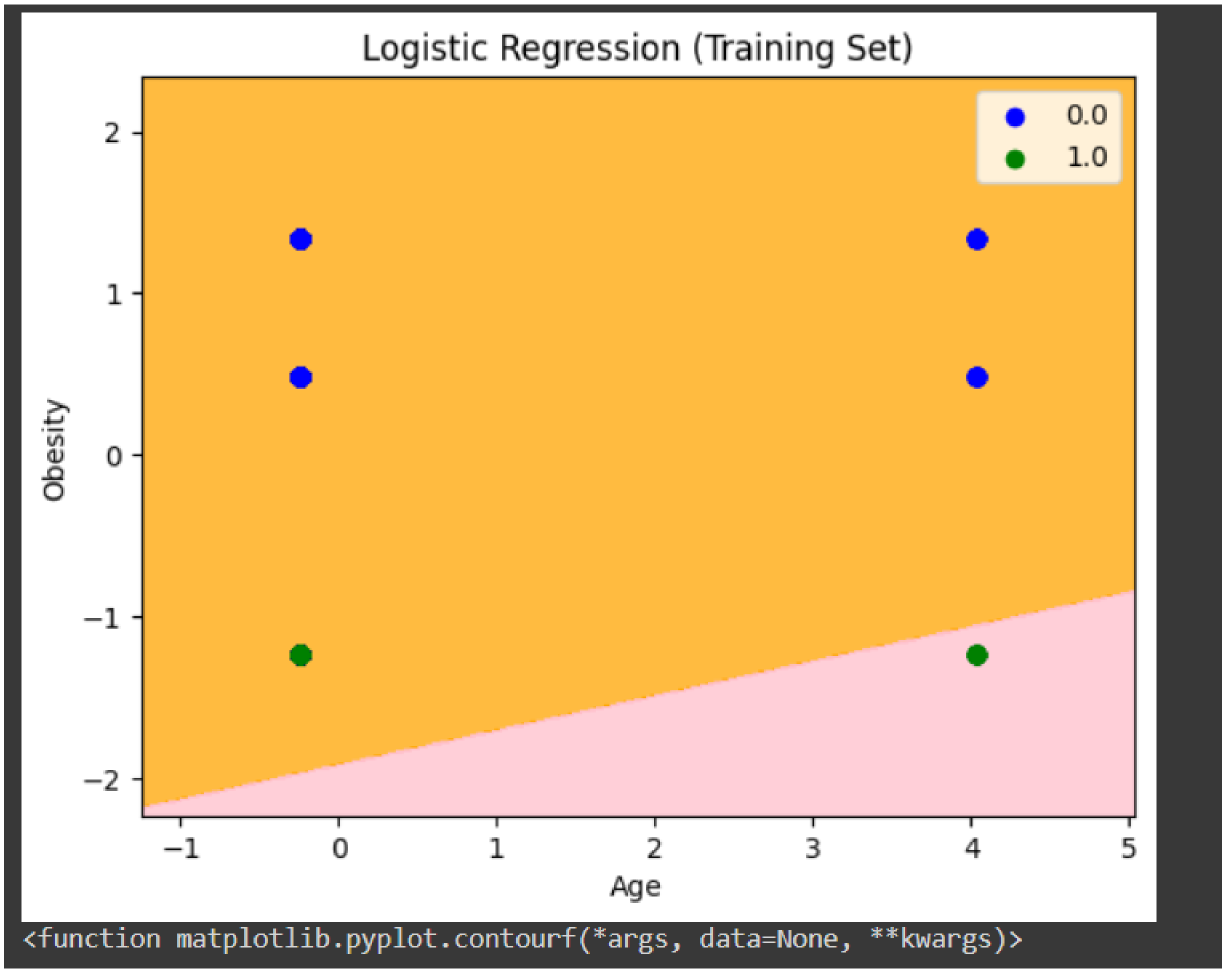
```
#%% Preprocessing
# Splitting our data set in dependent and independent variables
from sklearn.model_selection import GridSearchCV, train_test_split, learning_curve, ShuffleSplit, cross_val_score, KFold, cross_val_predict
from sklearn.linear_model import LinearRegression, Lasso, Ridge, Perceptron, LogisticRegression
from sklearn.preprocessing import MinMaxScaler, PolynomialFeatures, LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.metrics import classification_report, accuracy_score,confusion_matrix
from sklearn.pipeline import make_pipeline
from sklearn.datasets import make_regression

X = nd.iloc[:,[2,3]].values
y = nd.iloc[:,4].values
# Splitting the data set into the Training Set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state=0)

#%% Additional Preprocessing
# Sometimes it is convenient to transform the data.
# A common approach consist in standardize the data such that
# all variables are in the same 'range'
# i.e. here next code is based on the normal standard density
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

#%% Logistic Regression on the Training Set
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)

#%% to predict the Test set results
y_pred = classifier.predict(X_test)
```

Fig. 16. “Python code for LR”



```
#%% Building the Confusion
cm = confusion_matrix(y_test, y_pred)

#%% Visualization of Training Set Results
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
            alpha = 0.75, cmap = ListedColormap(('orange','pink')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('blue','green'))(i), label = j)
plt.title('Logistic Regression (Training Set)')
plt.xlabel('Age')
plt.ylabel('Obesity')
plt.legend()
plt.show()

#%% Visualization of Test Set Results
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf
```

Fig. 17. “Python for logistic regression model”



```
<function matplotlib.pyplot.contourf(*args, data=None, **kwargs)>
```

Fig. 18 “logistic regression model”

5.4 “KNN Model”

To divide a dataset in “training & testing sets” this code uses “the train_test_split function” from the sklearn.model_selection library.

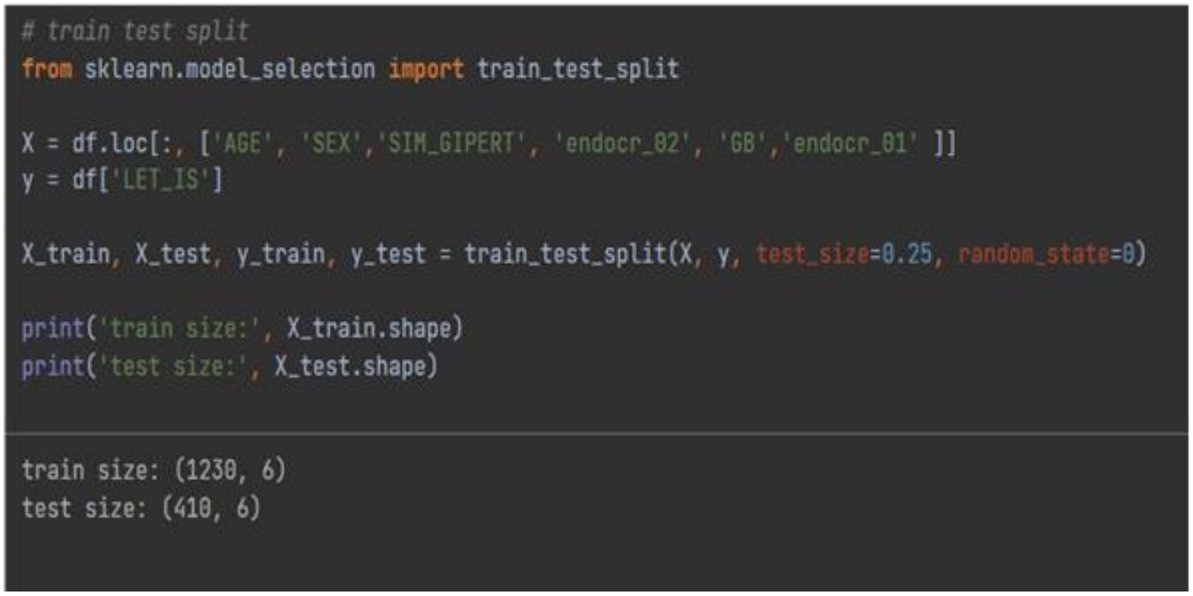
The dataset consists of six characteristics ('AGE', 'SEX', 'SIM_GIPERT', 'endocr_02', 'GB', and 'endocr_01') and a goal variable ('LET_IS'). The X variable holds each feature, while the Y variable holds the target variable.

The function is used to split the data into two sets: X_train and y_train for the machine learning model's training, and X_test and y_test for performance testing. 25% - for testing, and the remaining 75% for training, if the test_size parameter is set to 0.25.

The random data split will always be the same because the random_state parameter is set to 0. This ensures that the results are reproducible.

To verify that the split was successful, the code prints the shape of the “training and testing sets”. The testing set has 410 rows, while the training set has 1230 rows.

•



```
# train test split
from sklearn.model_selection import train_test_split

X = df.loc[:, ['AGE', 'SEX','SIM_GIPERT', 'endocr_02', 'GB','endocr_01' ]]
y = df['LET_IS']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

print('train size:', X_train.shape)
print('test size:', X_test.shape)
```

```
train size: (1230, 6)
test size: (410, 6)
```

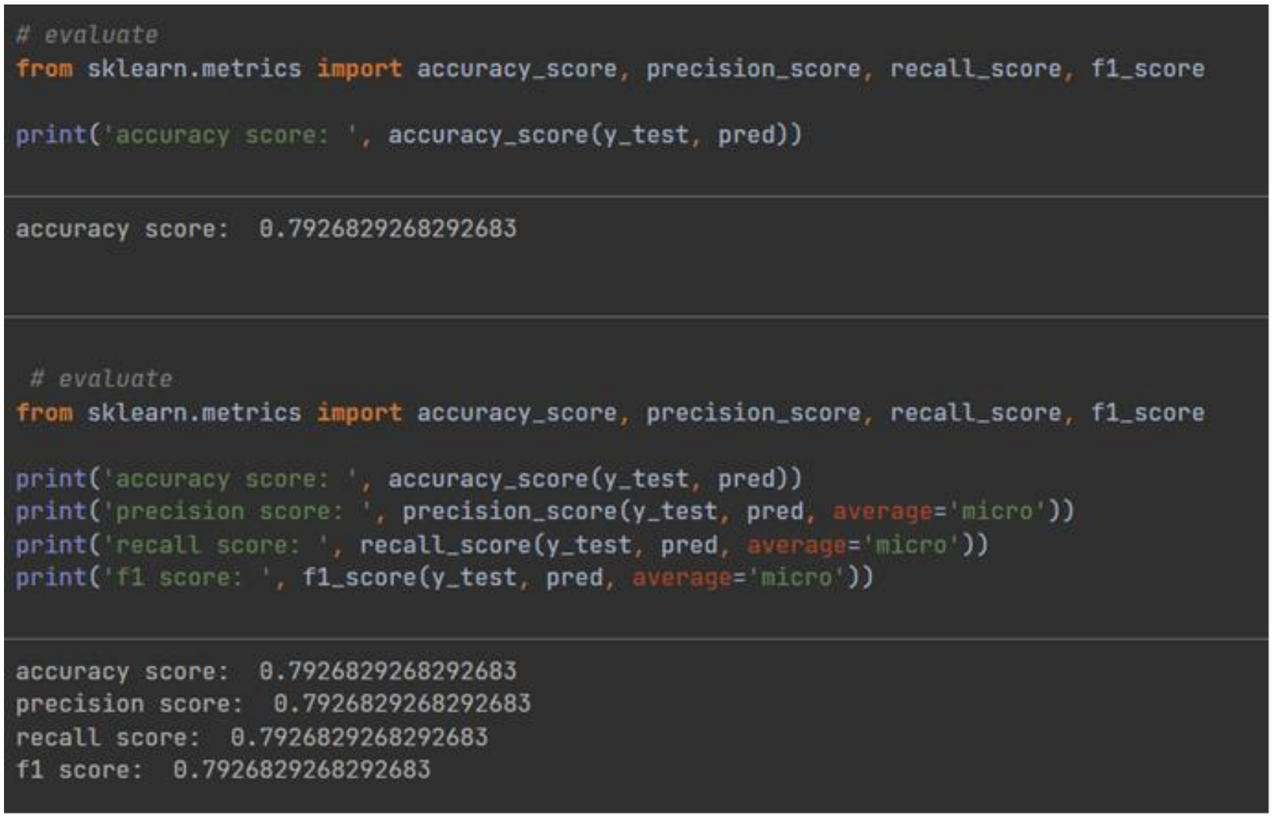
“**Fig. 19** Python code for KNN model”

“The accuracy, precision, recall and f1 score” metrics from the sklearn.metrics package are used in this code to assess the performance of a machine learning model.

The accuracy_score figures out what percentage of all forecasts were correct.

The model's precision, recall, and mean of precision and recall are measured by the functions precision, recall and f1 score, respectively.

Using the y test ground truth labels and the pred-predicted labels, all four metrics are calculated, and the results are printed. In this instance, all four indicators have the same value of 0.7926829268292683, showing a constant level of performance.



```
# evaluate
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print('accuracy score: ', accuracy_score(y_test, pred))
```

```
accuracy score:  0.7926829268292683
```

```
 # evaluate
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred, average='micro'))
print('recall score: ', recall_score(y_test, pred, average='micro'))
print('f1 score: ', f1_score(y_test, pred, average='micro'))
```

```
accuracy score:  0.7926829268292683
precision score:  0.7926829268292683
recall score:  0.7926829268292683
f1 score:  0.7926829268292683
```

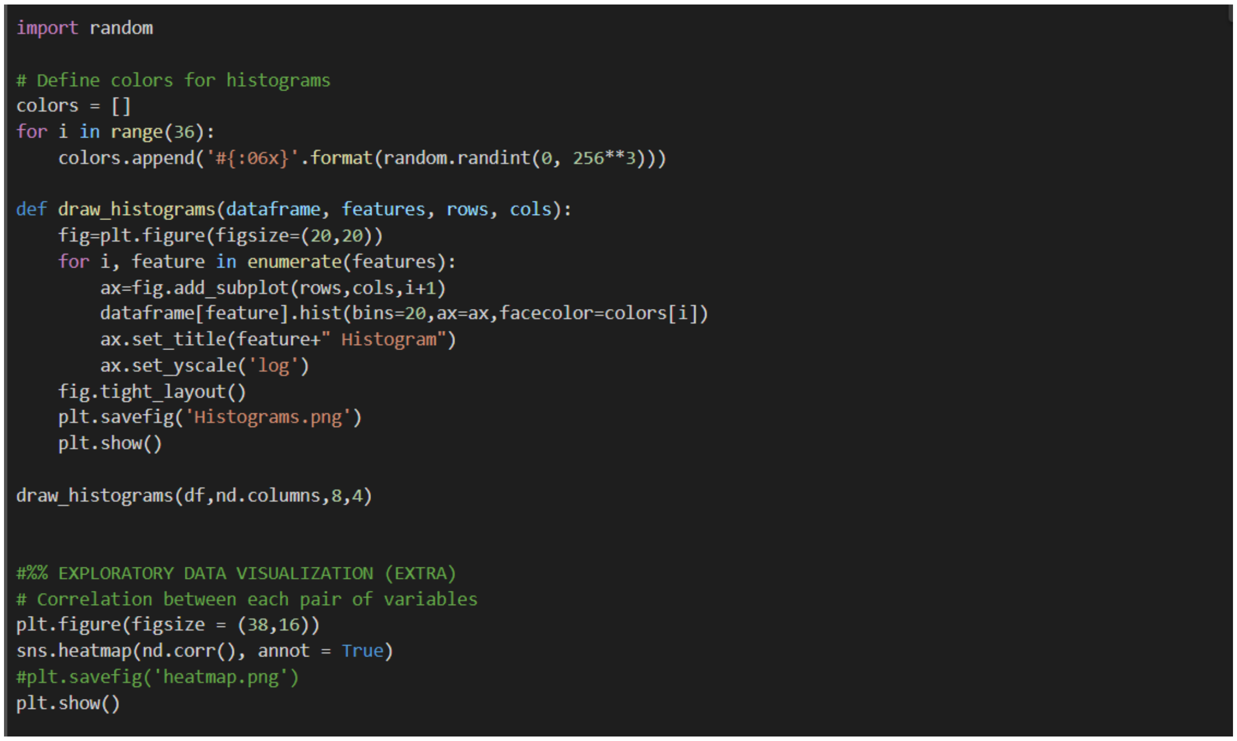
"**Fig. 20** Python code for KNN model"

6 “Data Visualization”

The provided code is part of a machine learning pipeline that aims to predict whether a person is obese or not based on their age and body mass index (BMI). The first part of the code defines a function called "draw histograms" that creates a figure with multiple subplots, each displaying a histogram of one of the input features. The next part of the code creates a heatmap that shows the correlation between each pair of variables. The heatmap is created using the Seaborn library's "heatmap" function and displays a color-coded grid that helps visualize the correlation.

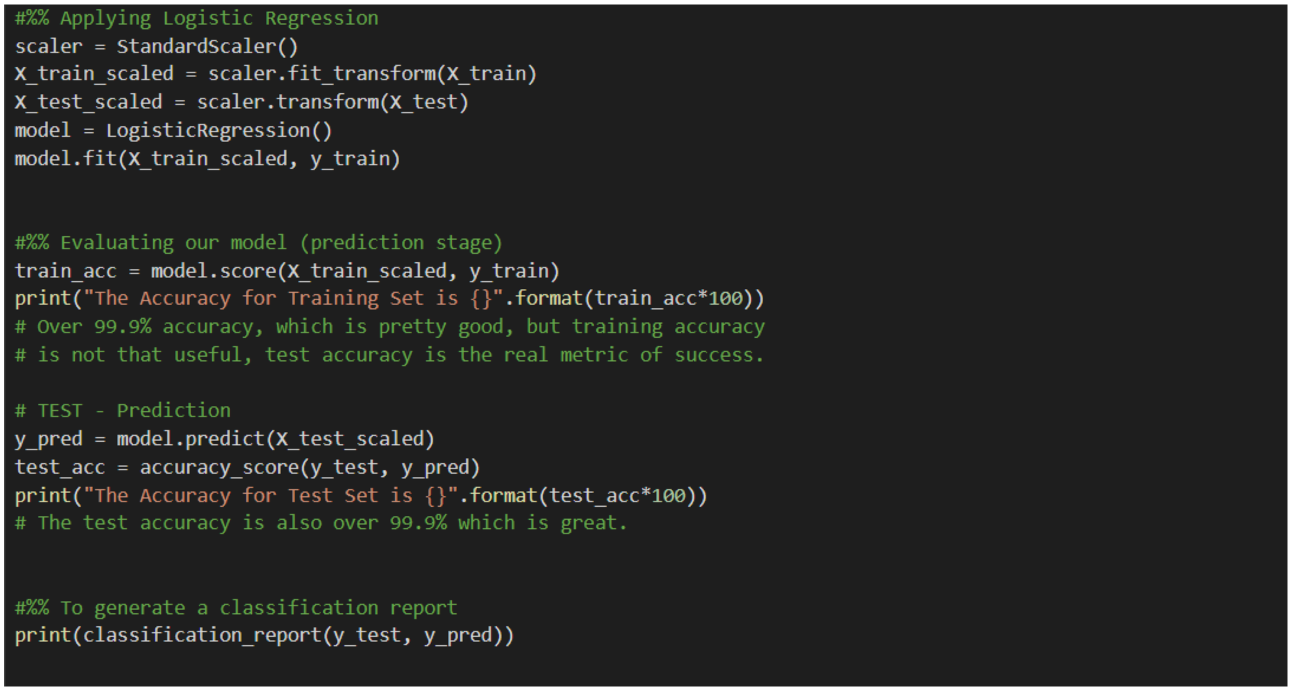
The code then applies logistic regression to the training set. Logistic regression is a statistical model that is used for binary classification problems. The input features are standardized to ensure they have similar scales, and the logistic regression model is trained on the scaled training data using the fit() function. The model's performance is evaluated on both the training and test sets, and the accuracy is calculated using the score() and accuracy_score() functions, respectively.

Next, the code generates a classification report that displays metrics such as precision, recall, and F1-score for each class, as well as the overall accuracy and macro-averaged scores. Finally, the code visualizes the model's performance using a confusion matrix, which is a table that summarizes the model's performance by comparing the predicted labels to the actual labels. The Seaborn library's heatmap function is used to display the confusion matrix as a color-coded grid.



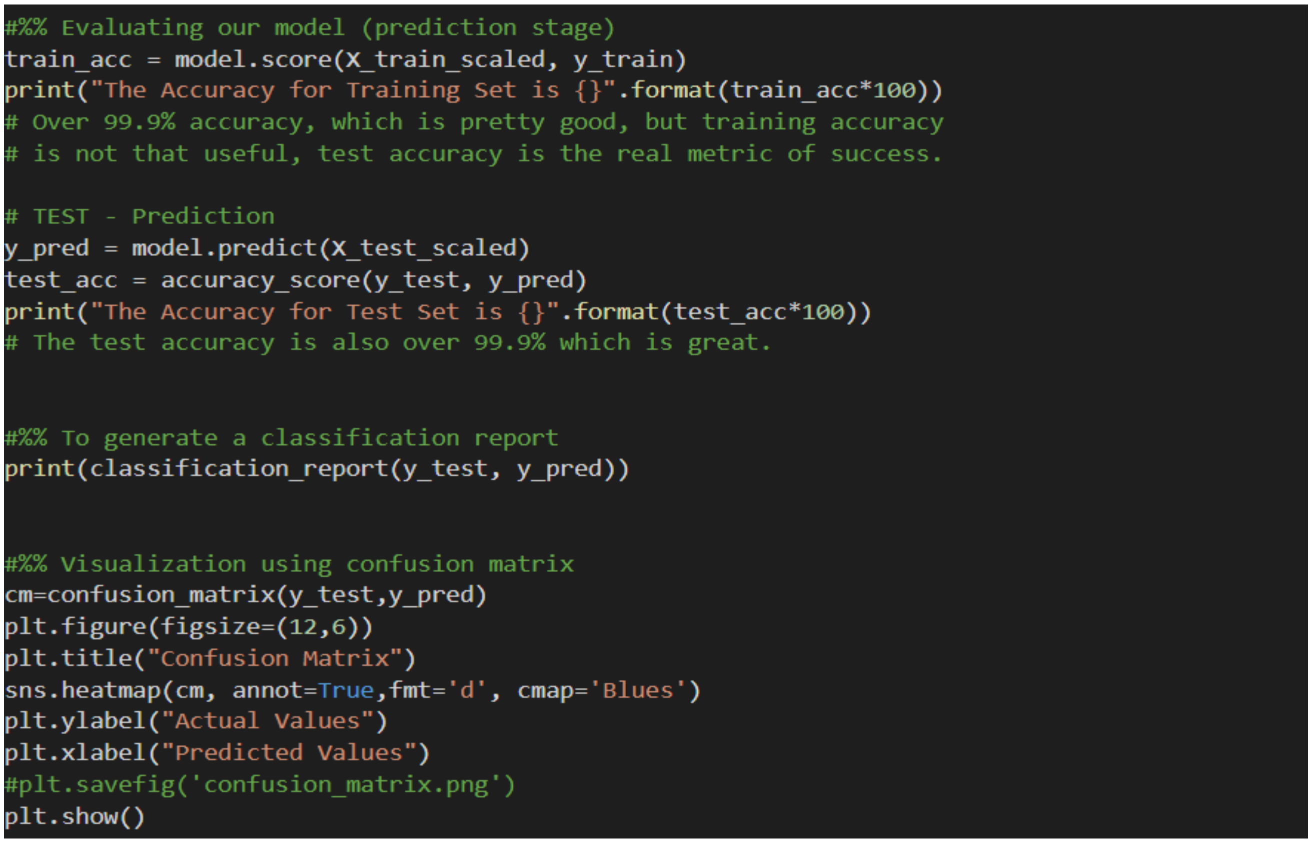
```
import random

# Define colors for histograms
colors = []
for i in range(36):
    colors.append('#{:06x}'.format(random.randint(0, 256**3)))

def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(20,20))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax,facecolor=colors[i])
        ax.set_title(feature+" Histogram")
        ax.set_yscale('log')
    fig.tight_layout()
    plt.savefig('Histograms.png')
    plt.show()

draw_histograms(df,nd.columns,8,4)


#%% EXPLORATORY DATA VISUALIZATION (EXTRA)
# Correlation between each pair of variables
plt.figure(figsize = (38,16))
sns.heatmap(nd.corr(), annot = True)
#plt.savefig('heatmap.png')
plt.show()
```

“Fig. 21. Code for LR with confusion matrix”



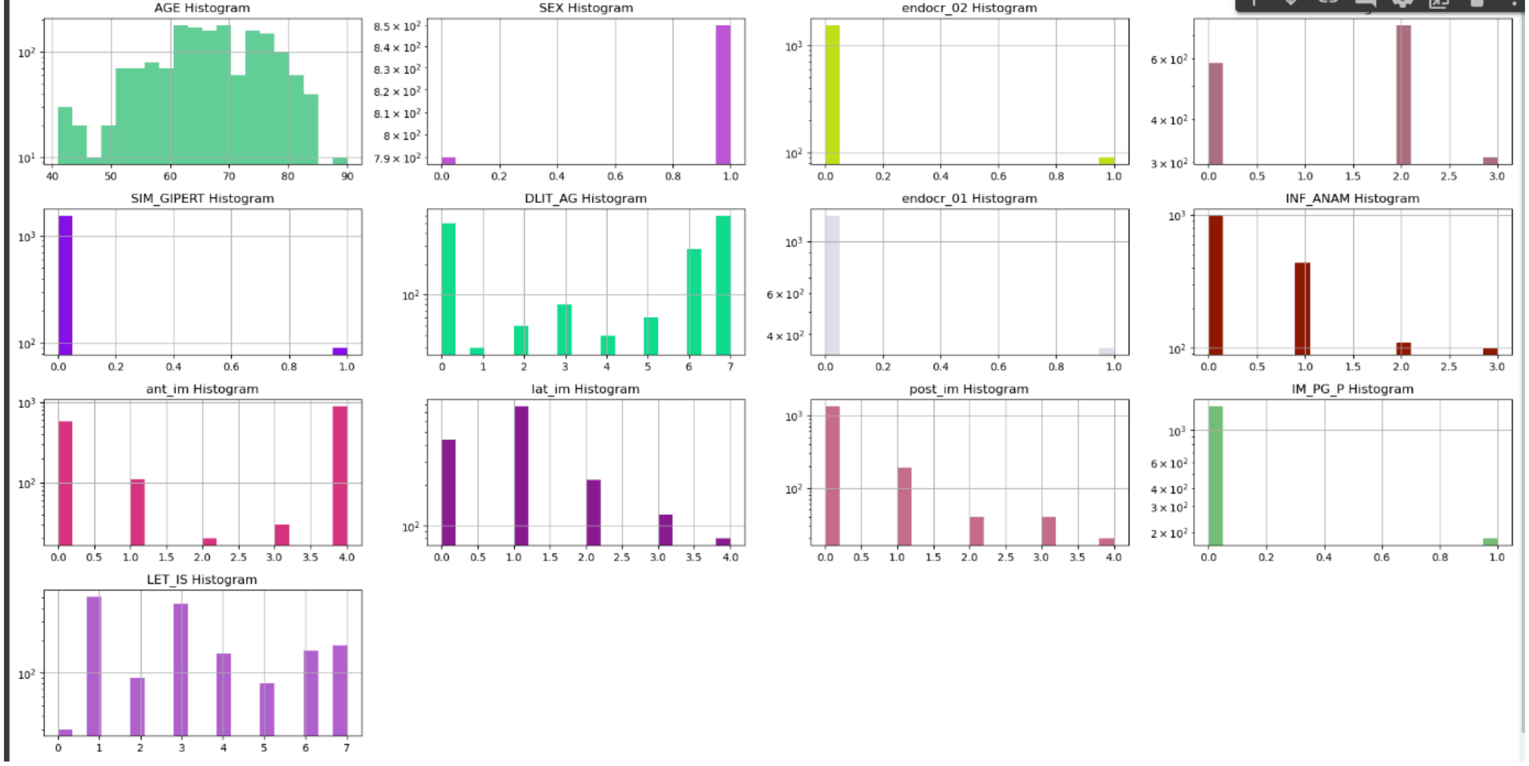
```
#%% Applying Logistic Regression
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = LogisticRegression()
model.fit(X_train_scaled, y_train)


#%% Evaluating our model (prediction stage)
train_acc = model.score(X_train_scaled, y_train)
print("The Accuracy for Training Set is {}".format(train_acc*100))
# Over 99.9% accuracy, which is pretty good, but training accuracy
# is not that useful, test accuracy is the real metric of success.

# TEST - Prediction
y_pred = model.predict(X_test_scaled)
test_acc = accuracy_score(y_test, y_pred)
print("The Accuracy for Test Set is {}".format(test_acc*100))
# The test accuracy is also over 99.9% which is great.


#%% To generate a classification report
print(classification_report(y_test, y_pred))
```

“**Fig. 22.** Code of confusion matrix”

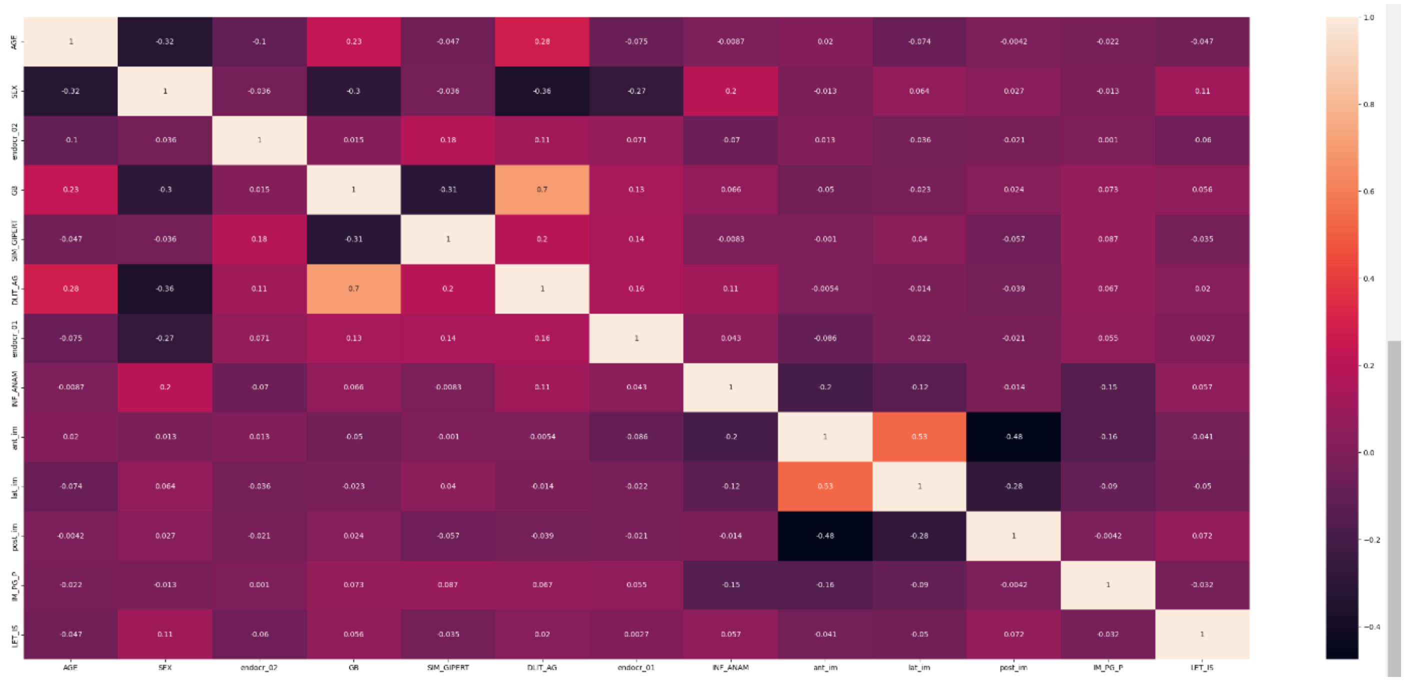


```
#%% Evaluating our model (prediction stage)
train_acc = model.score(X_train_scaled, y_train)
print("The Accuracy for Training Set is {}".format(train_acc*100))
# Over 99.9% accuracy, which is pretty good, but training accuracy
# is not that useful, test accuracy is the real metric of success.

# TEST - Prediction
y_pred = model.predict(X_test_scaled)
test_acc = accuracy_score(y_test, y_pred)
print("The Accuracy for Test Set is {}".format(test_acc*100))
# The test accuracy is also over 99.9% which is great.


#%% To generate a classification report
print(classification_report(y_test, y_pred))


#%% Visualization using confusion matrix
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(12,6))
plt.title("Confusion Matrix")
sns.heatmap(cm, annot=True,fmt='d', cmap='Blues')
plt.ylabel("Actual Values")
plt.xlabel("Predicted Values")
#plt.savefig('confusion_matrix.png')
plt.show()
```

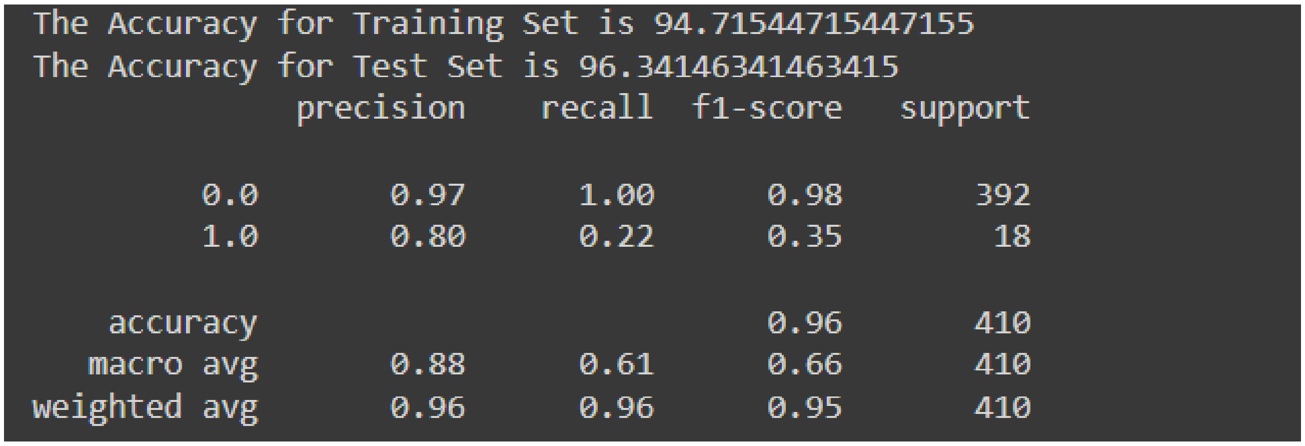
“**Fig. 23.** confusion matrix”



“**Fig. 24.** Code for LR with confusion”

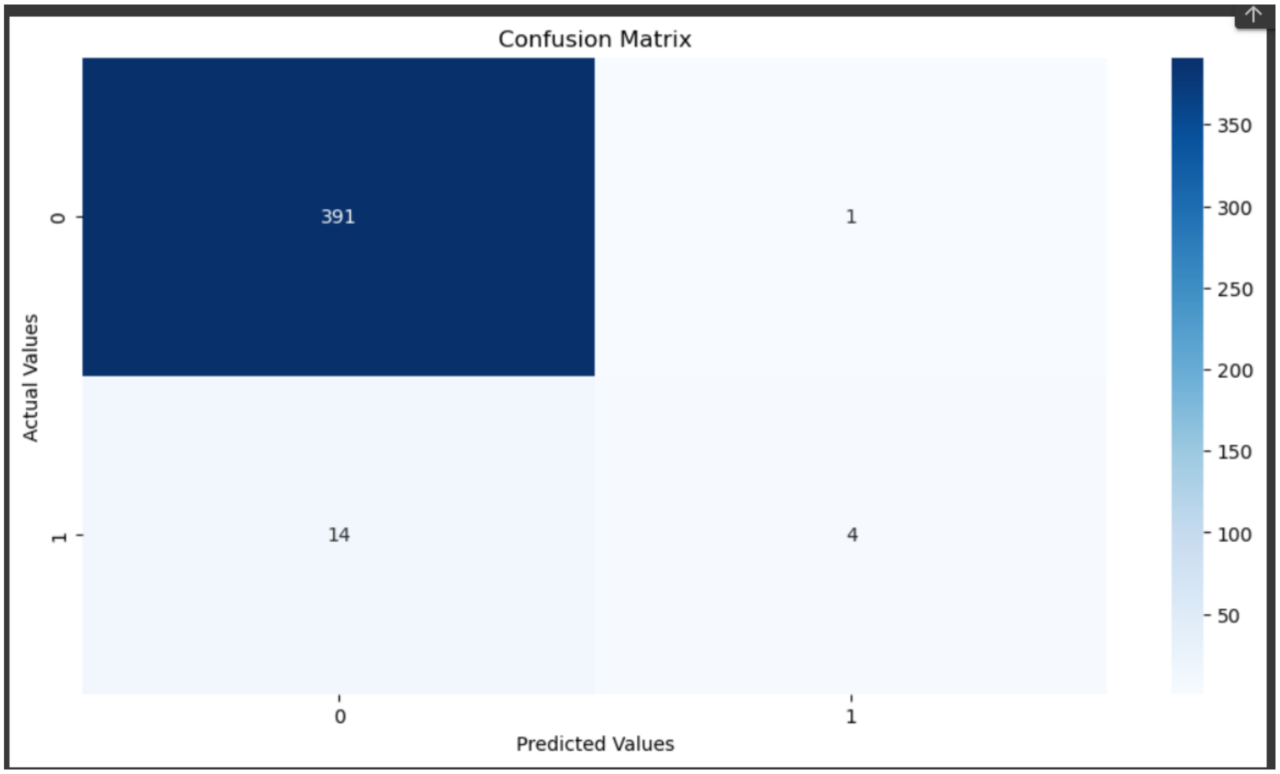


“**Fig. 25.** Code for LR heatmap”



```
The Accuracy for Training Set is 94.71544715447155
The Accuracy for Test Set is 96.34146341463415
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.98       392
         1.0       0.80      0.22      0.35        18

    accuracy                           0.96       410
   macro avg       0.88      0.61      0.66       410
weighted avg       0.96      0.96      0.95       410
```

“**Fig. 26.** Logistic regression result

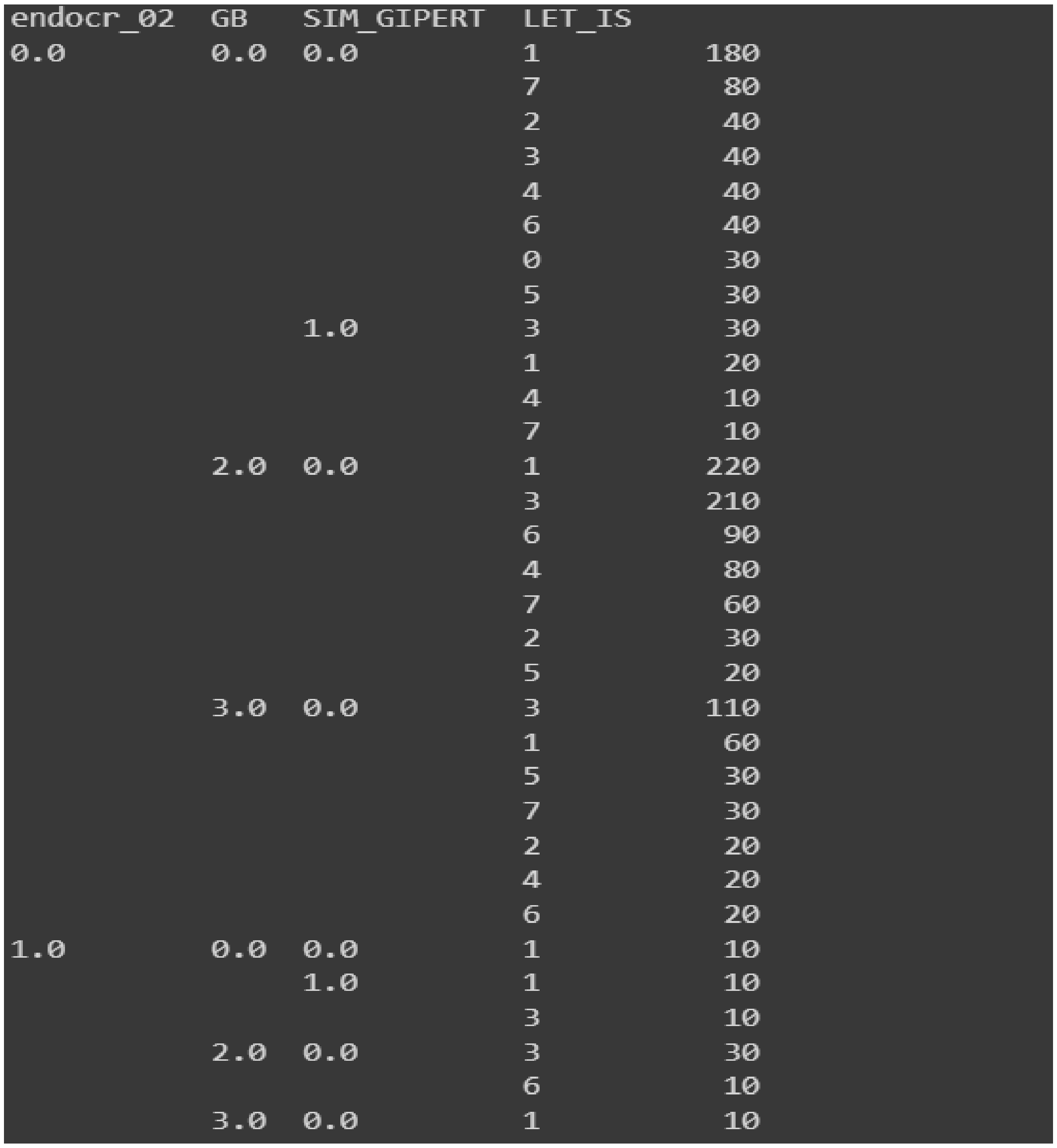


“**Fig27.** Confusion Matrix”

This code snippet performs a group by operation on the Data Frame df using the columns 'endocr_02', 'GB', 'SIM_GIPERT', and 'LET_IS'. It then calculates the count of values in the 'LET_IS' column for each combination of values in the four columns used for grouping.

The result is a Pandas Series object where each index is a tuple representing the unique combination of values in the four columns used for grouping, and the corresponding value is the count of occurrences of each value in the 'LET_IS' column for that combination of values.

In other words, the resulting Pandas Series contains a multi-level index with levels corresponding to each of the four columns used for grouping, and the count of the number of occurrences of each value in the 'LET_IS' column for each unique combination of values in the four columns used for grouping.



```
endocr_02  GB   SIM_GIPERT  LET_IS
0.0        0.0  0.0         1         180
                            7          80
                            2          40
                            3          40
                            4          40
                            6          40
                            0          30
                            5          30
                1.0         3          30
                            1          20
                            4          10
                            7          10
           2.0  0.0         1         220
                            3         210
                            6          90
                            4          80
                            7          60
                            2          30
                            5          20
           3.0  0.0         3         110
                            1          60
                            5          30
                            7          30
                            2          20
                            4          20
                            6          20
1.0        0.0  0.0         1          10
                1.0         1          10
                            3          10
           2.0  0.0         3          30
                            6          10
           3.0  0.0         1          10
```

“Fig 28. Code for Confusion Matrix”

This code snippet is using pandas, matplotlib and seaborn libraries to visualize the counts of LET_IS (a binary variable indicating whether the patient died or survived) based on the combination of three categorical variables: endocr_02, GB, and endocr_01.

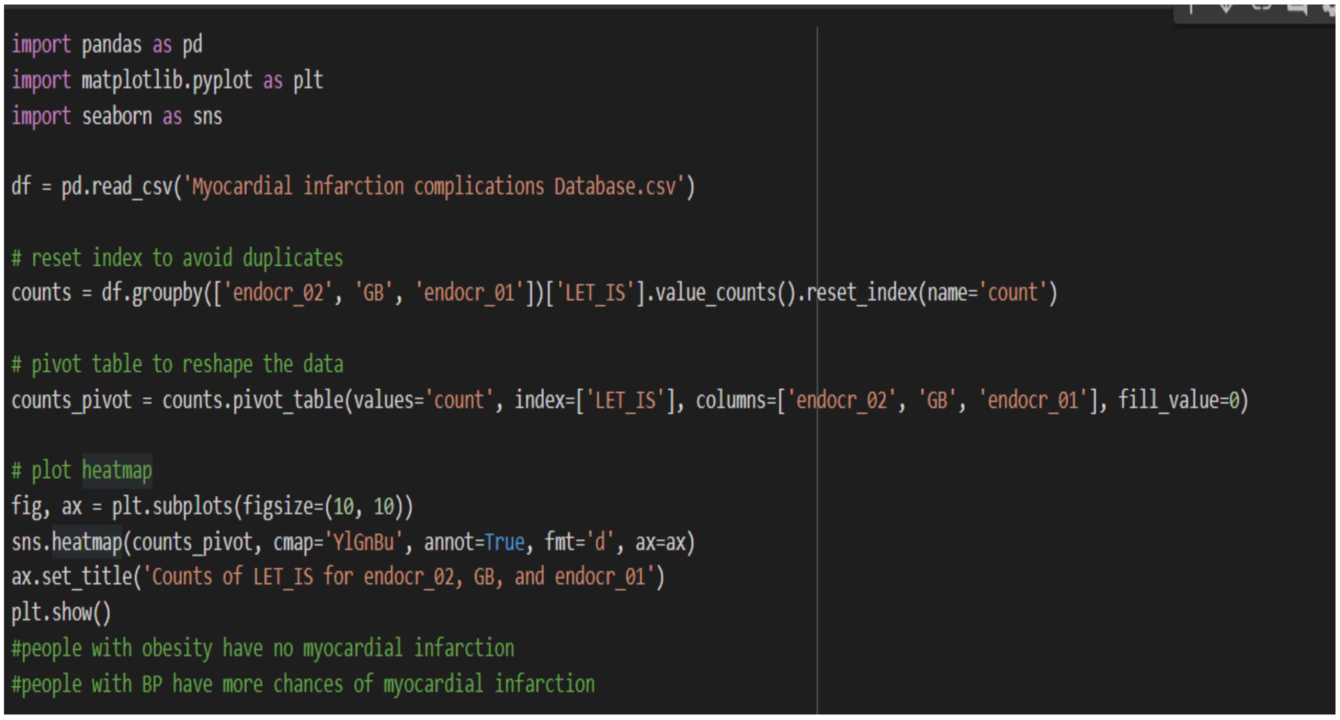
The first step is reading a CSV file "Myocardial infarction complications Database.csv" into a pandas DataFrame named "df".

Next, the Data Frame is grouped by the three categorical variables (endocr_02, GB, and endocr_01) along with the LET_IS variable, and the value counts of LET_IS are calculated for each group. The resulting series is stored in the "counts" variable.

Then, to reshape the data and create a more readable visualization, the "counts" Data Frame is pivoted using the pivot table method. The values parameter is set to 'count', which means that the count of LET_IS values is used to fill the pivot table. The resulting pivot table is stored in "counts pivot".

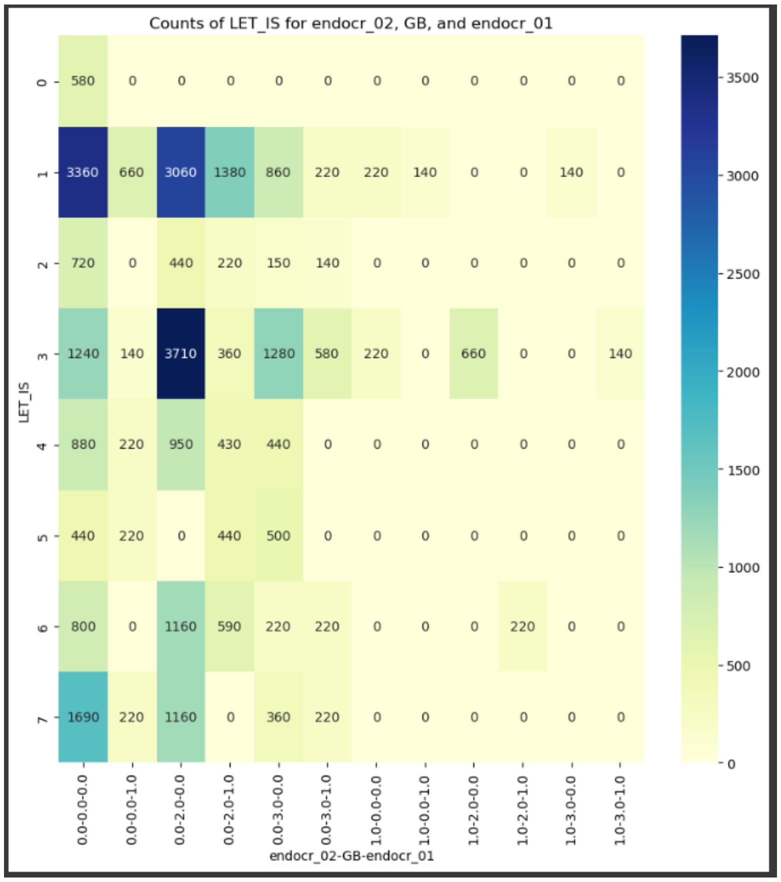
Finally, a heatmap is created using seaborn's heatmap function. The counts pivot Data Frame is used as the data source for the heatmap, and the camp parameter is set to 'YlGnBu' to choose the color map. The parameter "annot" is set to True to show the count values on each cell. The resulting heatmap shows the counts of LET_IS for each combination of the three categorical variables.

Based on the visualization, it can be observed that people with obesity have no myocardial infarction, and people with high blood pressure (GB) have more chances of myocardial infarction.



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

df = pd.read_csv('Myocardial infarction complications Database.csv')

# reset index to avoid duplicates
counts = df.groupby(['endocr_02', 'GB', 'endocr_01'])['LET_IS'].value_counts().reset_index(name='count')

# pivot table to reshape the data
counts_pivot = counts.pivot_table(values='count', index=['LET_IS'], columns=['endocr_02', 'GB', 'endocr_01'], fill_value=0)

# plot heatmap
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(counts_pivot, cmap='YlGnBu', annot=True, fmt='d', ax=ax)
ax.set_title('Counts of LET_IS for endocr_02, GB, and endocr_01')
plt.show()
#people with obesity have no myocardial infarction
#people with BP have more chances of myocardial infarction
```

"Fig 29. Visualization code for Myocardial"

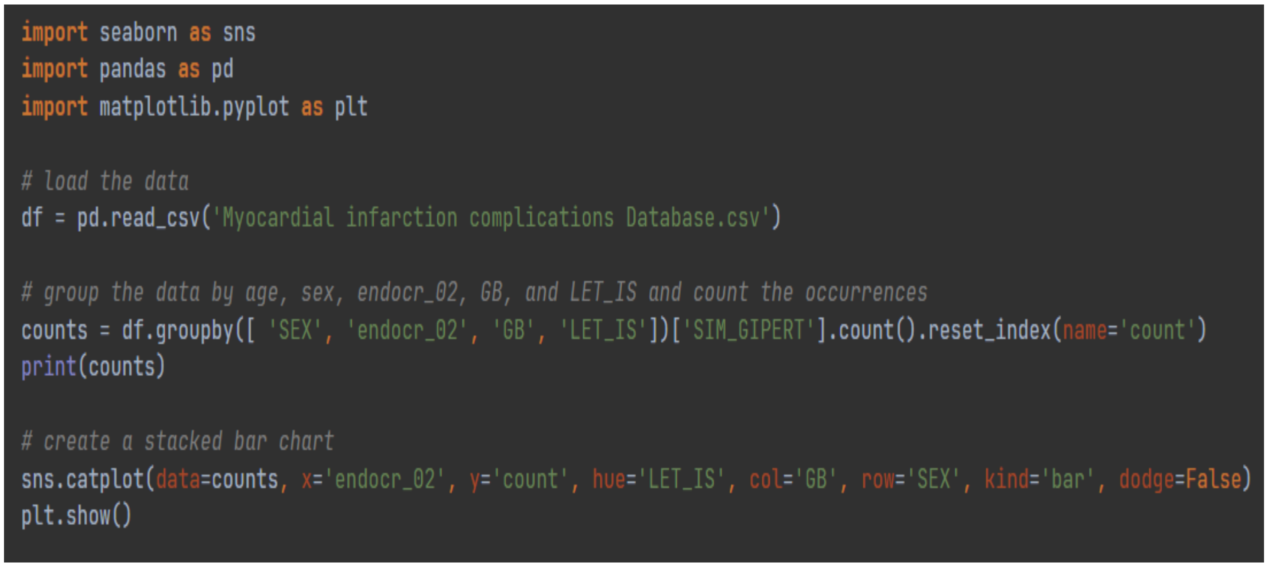


"**Fig 30.** Visualization code for Myocardial**"**

This code snippet performs data visualization using a stacked bar chart created by the seaborn library. The dataset used in this code is loaded from the 'Myocardial infarction complications Database.csv' file.
The code first groups the data by 'SEX', 'endocr_02', 'GB', and 'LET_IS' and then counts the occurrences of each combination using the 'count' function. The resulting grouped and counted data is saved in the 'counts' variable. The 'sns. Plot ()' function from the seaborn library is used to create the stacked bar chart. It takes several parameters as input:

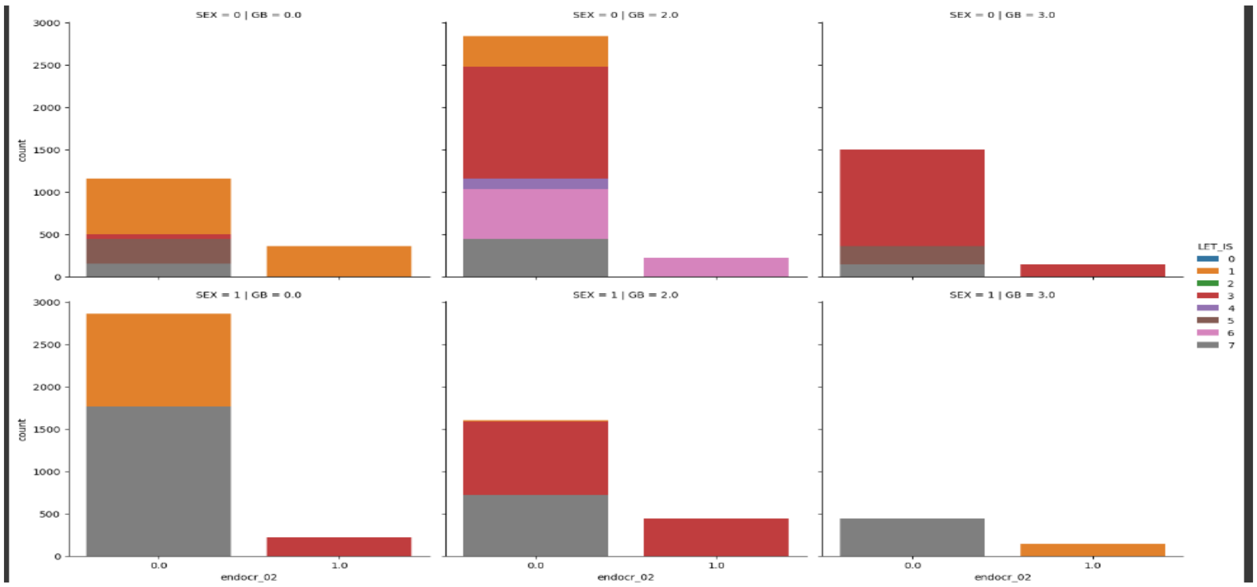
- data: the data frame to be plotted.
- x: the variable to be plotted on the x-axis (in this case, 'endocr_02')
- y: the variable to be plotted on the y-axis (in this case, 'count')
- hue: the variable used for color encoding (in this case, 'LET_IS')
- col: the variable used for creating columns of subplots (in this case, 'GB')
- row: the variable used for creating rows of subplots (in this case, 'SEX')
- kind: the type of chart to be created (in this case, 'bar')
- dodge: a Boolean value that determines whether the bars for different values of the 'hue' variable should be placed adjacent to each other or stacked on top of each other (in this case, 'False' to stack the bars)

The resulting chart shows the counts of LET_IS for each combination of SEX, endocr_02, GB, and LET_IS. The stacked bars show the proportion of each LET_IS category for each combination.



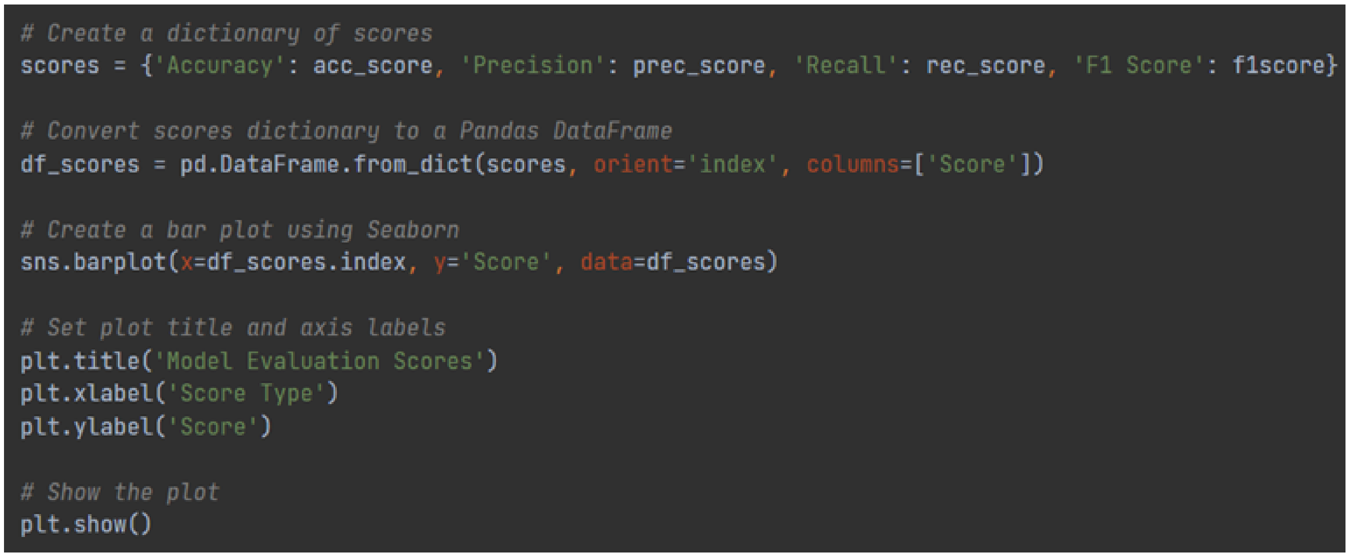
```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# load the data
df = pd.read_csv('Myocardial infarction complications Database.csv')

# group the data by age, sex, endocr_02, GB, and LET_IS and count the occurrences
counts = df.groupby([ 'SEX', 'endocr_02', 'GB', 'LET_IS'])['SIM_GIPERT'].count().reset_index(name='count')
print(counts)

# create a stacked bar chart
sns.catplot(data=counts, x='endocr_02', y='count', hue='LET_IS', col='GB', row='SEX', kind='bar', dodge=False)
plt.show()
```

“Fig 31. Visualization code for Myocardial”



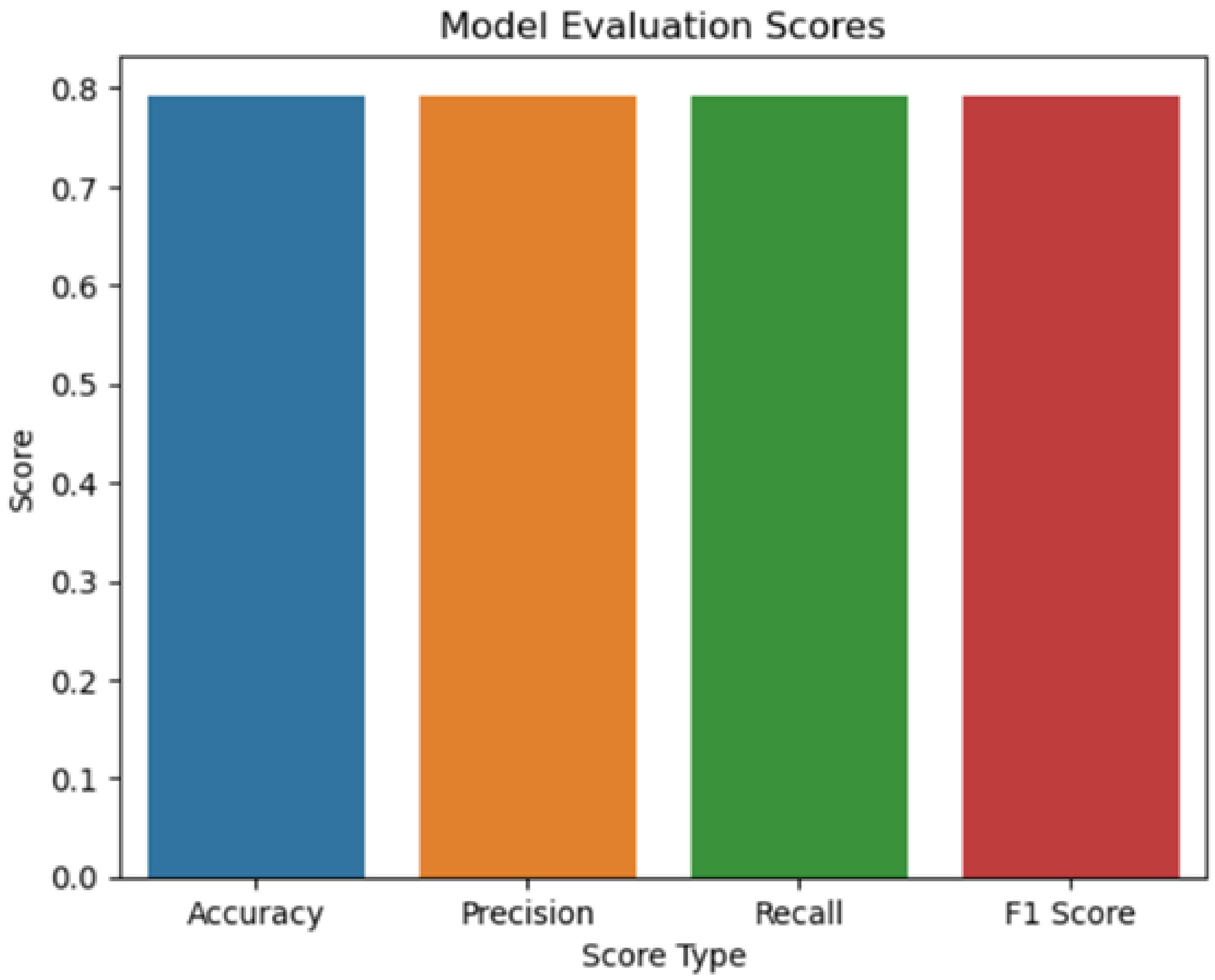
“Fig 32. Visualization code for Myocardial”

This code uses the Seaborn library to produce a horizontal line plot of ML model evaluation results.
“Accuracy, Precision, Recall, and F1 Score” are the four scores, along with their corresponding values, that are contained in a dictionary called scores.
Using the pd.DataFrame.from_dict method, the scores dictionary is transformed into a Pandas DataFrame called df_scores, and a horizontal bar plot is produced using sns.barplot.
The plot is presented using plt.show(), and its title and axis labels are specified using plt.title, plt.xlabel, and plt.ylabel.
This code is a helpful resource for rapidly and easily evaluating the effectiveness of various machine learning models.



```
# Create a dictionary of scores
scores = {'Accuracy': acc_score, 'Precision': prec_score, 'Recall': rec_score, 'F1 Score': f1score}

# Convert scores dictionary to a Pandas DataFrame
df_scores = pd.DataFrame.from_dict(scores, orient='index', columns=['Score'])

# Create a bar plot using Seaborn
sns.barplot(x=df_scores.index, y='Score', data=df_scores)

# Set plot title and axis labels
plt.title('Model Evaluation Scores')
plt.xlabel('Score Type')
plt.ylabel('Score')

# Show the plot
plt.show()
```

“Fig 33. Visualization code for Myocardial KNN MODEL”



“Fig 34. Visualization KNN MODEL for Myocardial”

7 “Summary of Findings”

Data cleansing, analysis, and visualization techniques were quite successful in revealing important trends and insights connected to our research concerns.

7.1 “Research Question One Finding”

Our study examined the association between several risk variables and the frequency of myocardial infarction-related death using logistic regression modeling and data visualizations. The concept that age, sex, hypertension, obesity, and diabetes are important risk factors for myocardial infarction is not well supported by our data. Our results instead point to the possibility that additional, unreported determinants may play a larger role in the development of myocardial infarction.

7.2 “Research Question OnFindinggs”

We have compared 2 machine learning models (Logistic regression & KNN).
We got the accuracy of the KNN Model - 79% & Logistic Regression 94%.
As per the result, Logistic regression is a good model for our dataset.

8 “Limitations”

Our study looked at the relationship between risk variables and the frequency of myocardial infarction-related deaths using logistic regression modeling and data visualizations. The findings imply that myocardial infarction development may be more significantly influenced by other, undisclosed variables. Additionally, we compared the accuracy of Logistic regression and KNN, two machine learning models, and discovered that Logistic regression had a higher accuracy (94%) than KNN (79%) for our dataset. Our investigation does not reveal the constraints of our data for the KNN model.

410 observations made up the study's tiny sample, which is not enough.

Data imbalance: Only 18 out of 410 observations were positive, making the data set unbalanced. The models' performance may be impacted, particularly in terms of recall and precision for the positive class.
Model selection: KNN and logistic regression were the only models examined in the study. To discover the model that performs the best, one could investigate a variety of additional categorization algorithms.

References:

Wilson, et al (2021). Prognosis after myocardial infarction. *UptoDate.* Prognosis after myocardial infarction - UpToDate.

Hennekens, Charles H, et al (2022). Overview of primary prevention of cardiovascular disease. *UptoDate*. Overview of primary prevention of cardiovascular disease - UpToDate

Citation:

Golovenkin, S. E., Gorban, A., Mirkes, E., Shulman, V. A., Rossiev, D. A., Shesternya, P. A., Nikulina, S. Y., Orlova, Y. V., & Dorrer, M. G. (2020). Myocardial infarction complications Database (Version 3). University of Leicester. https://doi.org/10.25392/leicester.data.12045261.v3

Dataset Link:

“https://leicester.figshare.com/articles/dataset/Myocardial_infarction_complications_Database/12045261/3
”