

CLASSIFICATION

Title: Improving the Prediction of Heart Attacks with Machine Learning based on medical test and diagnostics tests.

1.1 Introduction

Classification is the process of arranging, sorting, or dividing objects into groups of similar type, values, labels etc. Classification is an important concept in machine learning and data analysis. It is a supervised learning technique which involves categorizing data into different or predetermined classes based on certain features or attributes (Sarkar, 2018). Examples of supervised learning are Logistic Regression, Decision Trees, Random Forest, K-Nearest Neighbors (KNN), Neural Networks e.t.c. The reason for classification is to determine a model or algorithm that can assign a label or class to a data point which is often represented as a feature vector.

Some of the applications of classification includes finance, healthcare, marketing, image recognition, web page classification etc

Classification in the case of heart attack prediction involves training a machine learning algorithm to predict whether a given individual is likely to experience a heart attack.

Cardiovascular diseases (CVDs) are one of the major cause of death globally (Roth et al., 2020). CVDs is a general term use to describe disorders of the heart and blood vessels and include coronary heart disease, cerebrovascular disease, rheumatic heart disease and other conditions. More than four out of five CVD deaths are due to heart attacks and strokes, and one third of these deaths occur prematurely in people under 70 years of age. Lee & Yoon, (2017) studied a broad overview of machine learning techniques used in healthcare for various diseases.

This predictive modelling is particularly valuable for healthcare professionals aiming to identify individuals who are at a higher risk of experiencing a heart attack. By utilizing a dataset containing relevant health indicators such as blood pressure, age, kcm and other vital factors. A classification algorithm can learn patterns and relationships that contribute to predicting the likelihood of a heart attack.

Tripoliti et al. (2017) conducted a thorough review with an emphasis on machine learning techniques for assessing heart failure. They looked into estimating the severity of heart failure and making predictions about mortality, rehospitalization, and destabilizations.

In the past, a variety of models with various algorithms have been put forth, leading to original discussions regarding accuracy and reliability for heart disease. Numerous data mining prediction models, including SVM, Naïve Bayes, Decision Trees, Bagging and Boosting, and Random Forest for heart disease, have been introduced in the literature review mentioned above. These algorithms produced very high accuracy models for heart disease prediction.

1.2 Aims and Objectives

This study's primary objective is to increase the precision of heart attack risk prediction via the application of machine learning methods. The goal is to find the best model for heart attack prediction by analyzing and comparing various classification algorithms using a sizable dataset of relevant health data and improving patient satisfaction and optimizing healthcare resources.

1.3 EXPLORATION OF DATASET ANALYSIS

1.3.1 Description of the dataset

The heart attack dataset was sourced from google dataset. The size of the dataset is 1319 samples or observations (rows) which have nine features, where eight features (columns) are for the input and one feature (column) is for the output.

This dataset columns are:

Column labels	Description
Age	The age of the person in years
Gender	The biological sex or gender identity of patients. 0 is used for male and 1 for female.
Impulse	The pulse or heart rate of the patients
Pressure hight	The high blood pressure readings or systolic blood pressure measurements. Which is the pressure caused by your heart contracting and pushing out blood
Pressure Low	The low blood pressure readings or diastolic blood pressure measurement which is the pressure when your heart relaxes and fills with blood
Glucose	The sugar level of the blood.
KCM	CK-MB, also known as Creatine Kinase-MB, is an enzyme found in the heart muscle (myocardium). It is a subtype of the enzyme creatine kinase, which is present in various tissues throughout the body, including the heart, brain, and skeletal muscles.
Troponin	Possibly refers to troponin levels, a protein associated with heart muscle contraction and commonly measured in cardiac health assessments.
Class	A categorical class or outcome variable indicating the presence of heart attack. (Positive and negative Value)

Table 1.1: The dataset values

1.3.2 Dependent and independent variables

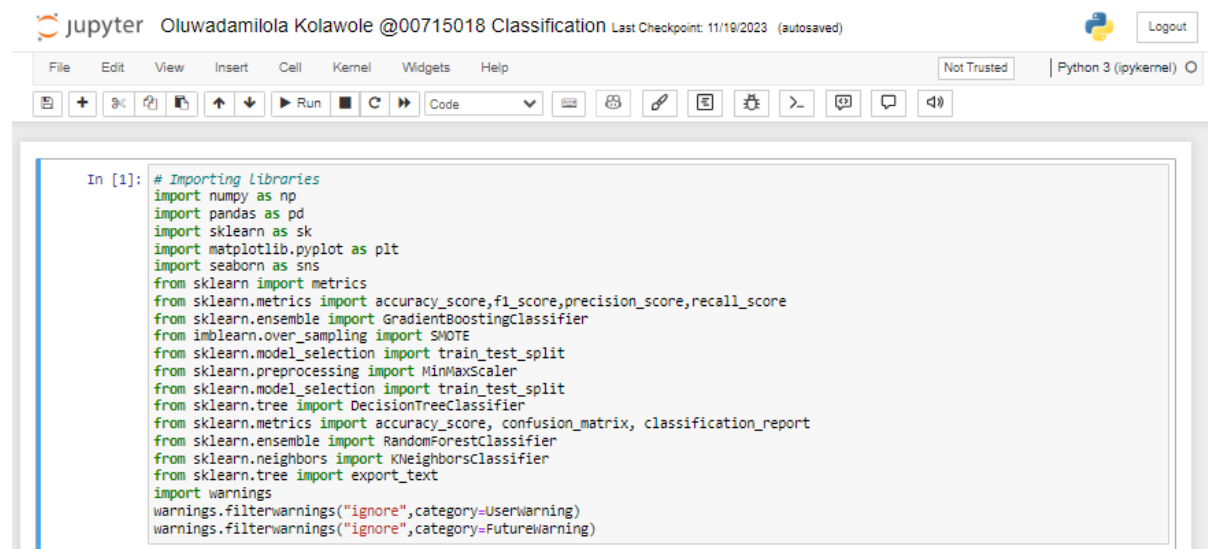
The variable that the model predicts or explains is known as the dependent variable in a statistical or mathematical model. Variables that are used to predict the such variable are known as the independent variables. Since they are not determined by the values of the dependent variable, they are known as independent variables. Stated differently, the independent variable value establish the independent variables' values.

1.3.3 Data Preparation

Data preparation is an important step in the data machine learning. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis and modelling (Witten et al., 2011). Data preparation improves the quality of the data and make it more suitable for the task.

1.3.4 Data integration

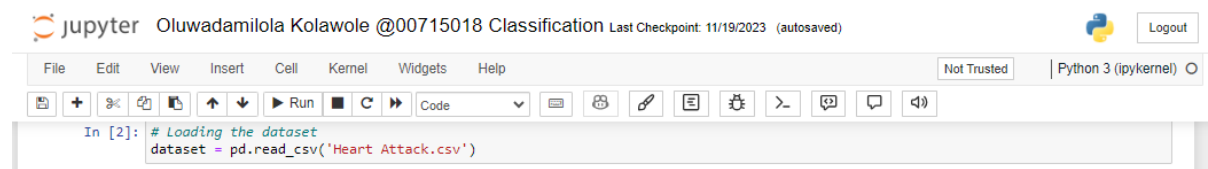
Importing all libraries using the code below



```
In [1]: # Importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
from sklearn.ensemble import GradientBoostingClassifier
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import export_text
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
```

Figure 1.1: Showing the necessary libraries used.

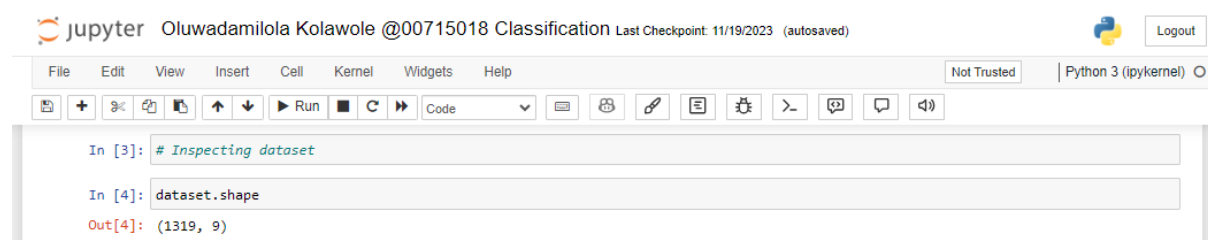
The heart attack dataset was downloaded as a CSV file and imported into python



```
In [2]: # Loading the dataset
dataset = pd.read_csv('Heart Attack.csv')
```

Figure 1.2: Loading of dataset

The dataset was read and examined using the code shown in figure 3. The dataset additionally validate the 1319 variables and the 9 columns depicted in Figure 3.



```
In [3]: # Inspecting dataset

In [4]: dataset.shape
Out[4]: (1319, 9)
```

Figure 1.3: Showing the numbers of observations and features

The Jupyter Notebook interface shows a code cell with the following content:

```
In [5]: # Lets view the first five rows of the dataset
dataset.head()
```

The output of the code cell is displayed as follows:

```
Out[5]:
```

	age	gender	impluse	pressurehigh	pressurelow	glucose	kcm	troponin	class
0	64	1	66	160	83	160.0	1.80	0.012	negative
1	21	1	94	98	46	296.0	6.75	1.060	positive
2	55	1	64	160	77	270.0	1.99	0.003	negative
3	64	1	70	120	55	270.0	13.87	0.122	positive
4	55	1	64	112	65	300.0	1.08	0.003	negative

Figure 1.4: It show the first five rows of dataset.

The Data Frame's brief overview is shown by the info() function. Along with the memory usage, the data displays the number of rows, columns, labels, data types, and non-null values in each column.

The Jupyter Notebook interface shows a code cell with the following content:

```
In [6]: # Lets see the data information
dataset.info()
```

The output of the code cell is displayed as follows:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1319 entries, 0 to 1318
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   age             1319 non-null   int64
1   gender          1319 non-null   int64
2   impluse         1319 non-null   int64
3   pressurehigh    1319 non-null   int64
4   pressurelow     1319 non-null   int64
5   glucose         1319 non-null   float64
6   kcm             1319 non-null   float64
7   troponin        1319 non-null   float64
8   class           1319 non-null   object
dtypes: float64(3), int64(5), object(1)
memory usage: 92.9+ KB
```

Figure 1.5: Data frame summary

The describe() function returns statistical description of the data. It returns the below information for numeric columns:

count - The number of unempty values.

mean - The average value.

std - The standard deviation.

min - the minimum value.

25% - The 25% percentile (The value at which 25% of the values are less than it.)

50% - The 50% percentile (The value at which 50% of the values are less than it.)

75% - The 75% percentile (The value at which 75% of the values are less than it.)

max - the maximum value.

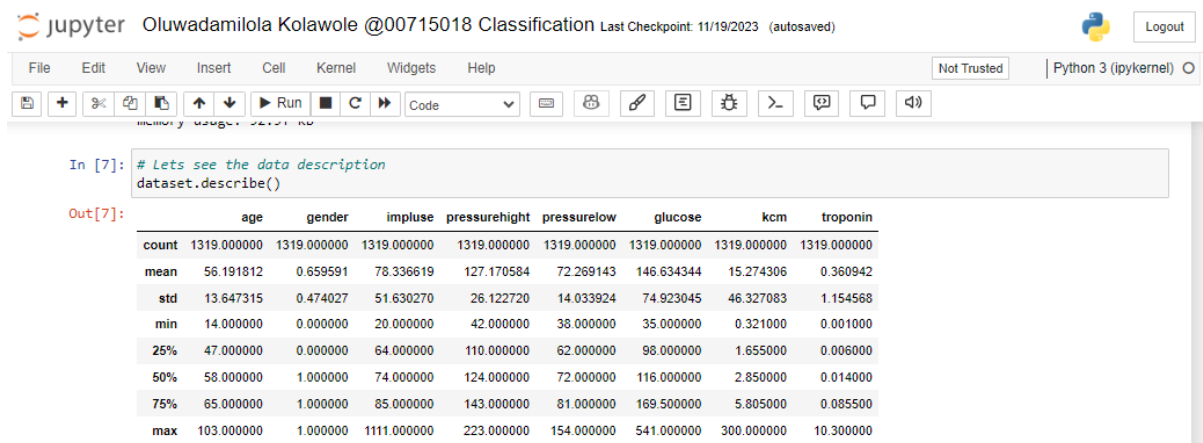


Figure 1.6: The statistical information of the data

1.3.5 Data Cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset

Data cleaning ensures that the data is in an appropriate format for the algorithm to use (Zhao et al., 2019).

There were no missing data or duplicate values, however, the target value column (Class) is a categorical data type so it has to be transformed to numerical. This is achieved by replacing the negative and positive labels with 0 and 1 respectively.

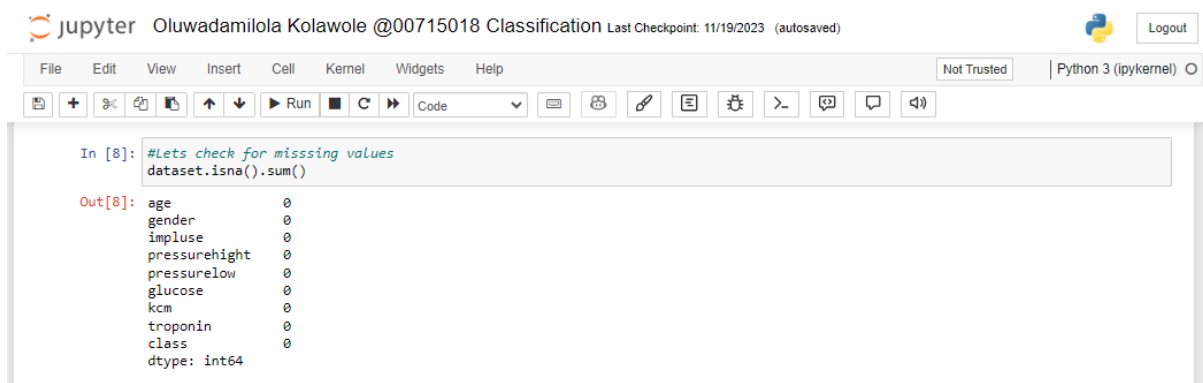


Figure 1.7: Checking for missing values

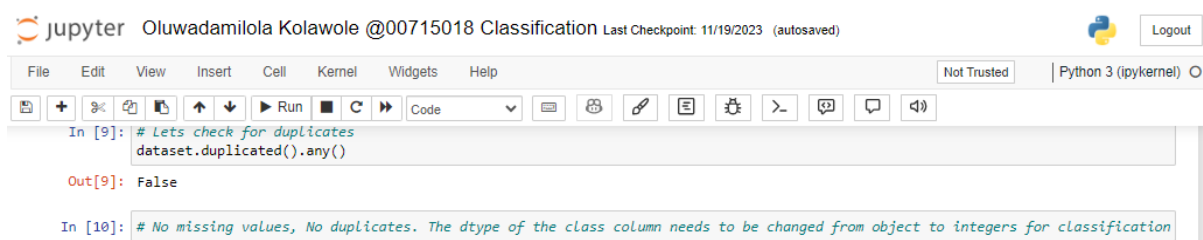


Figure 1.8: Checking for duplicate values

The code below shows the count occurrence of the class distribution

```
jupyter Oluwadamilola Kolawole @00715018 Classification Last Checkpoint: 11/19/2023 (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
In [12]: # df = pd.DataFrame(dataset)

# Use value_counts to count occurrences of each category
class_counts = dataset['class'].value_counts()

print(class_counts)

# The heart attack class
plt.figure(figsize=(10,5))
sns.histplot(x="class",hue="class", data=dataset)
plt.title("Class Distribution")
plt.show()

positive    810
negative    509
Name: class, dtype: int64
```



Figure 1.9: The Values of Negative and positive class of the dataset

Visualizing the count values of each column of the independent variables and comparing it with the class distribution.

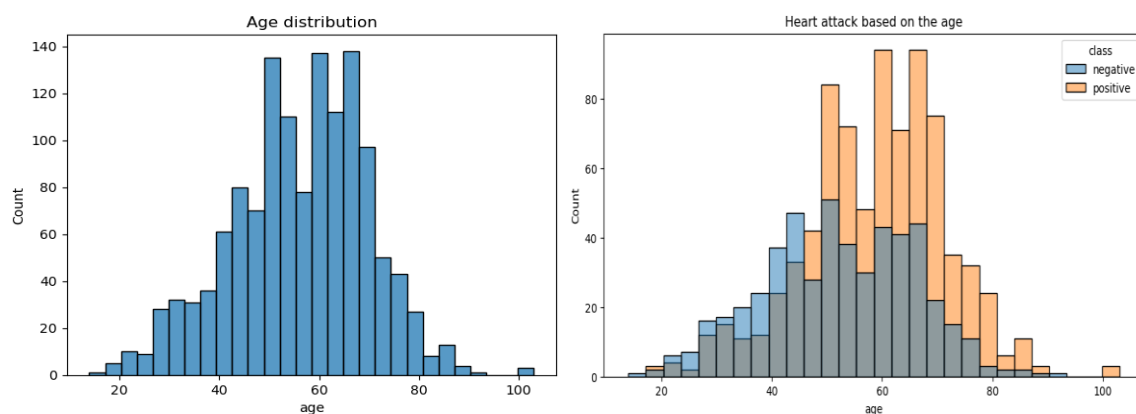


Figure 1.10: The Age distribution of the data

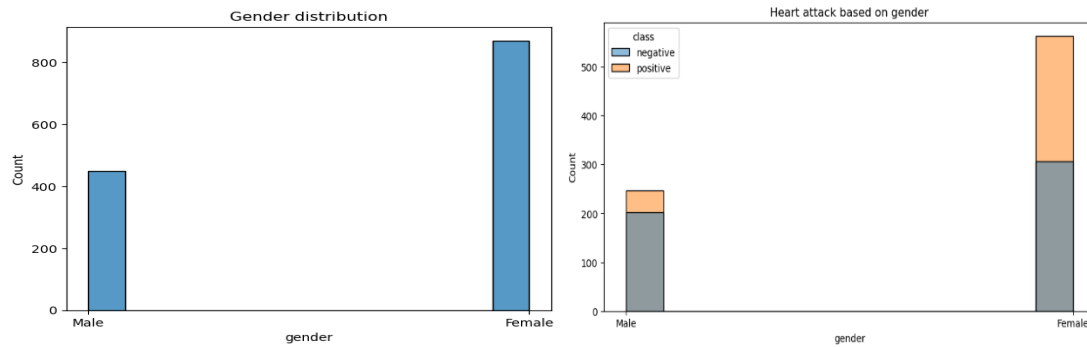


Figure 1.11: The Gender distribution of the data

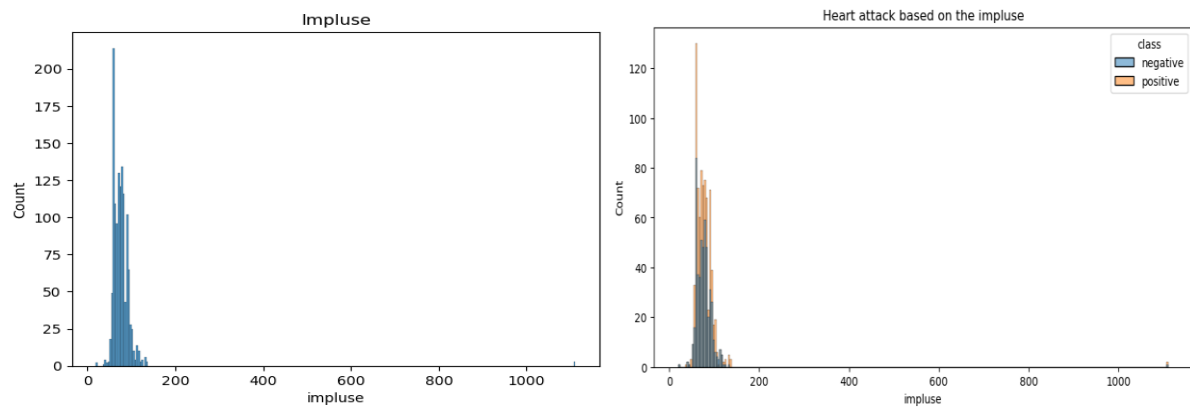


Figure 1.12: The impulse distribution of the data

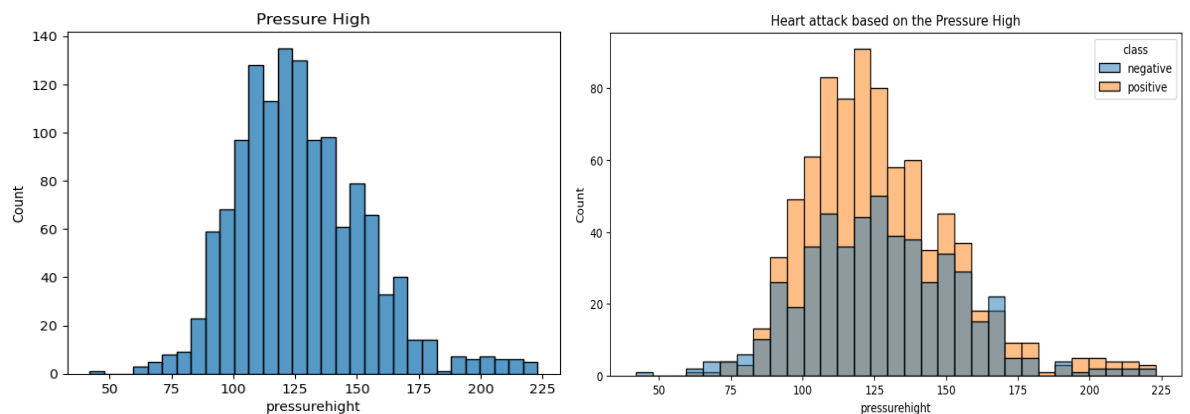


Figure 1.13: The Pressure high of the distribution

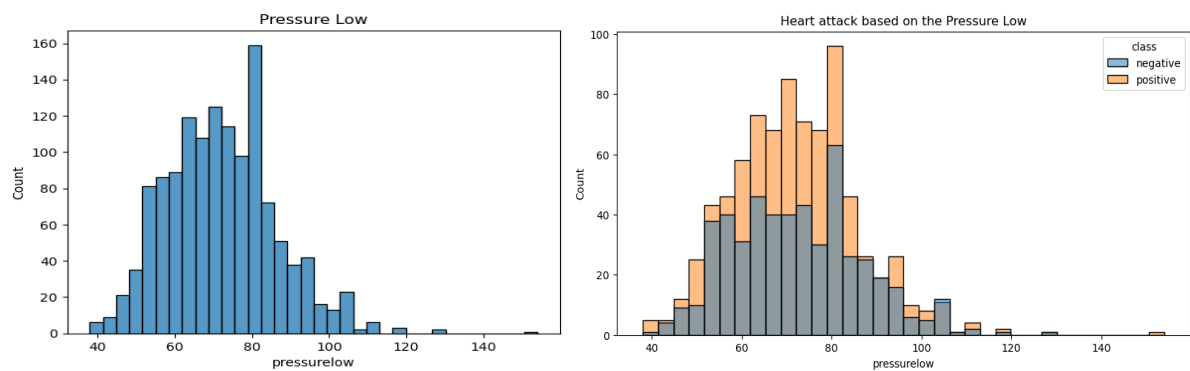


Figure 1.14: The Pressure Low of the distribution

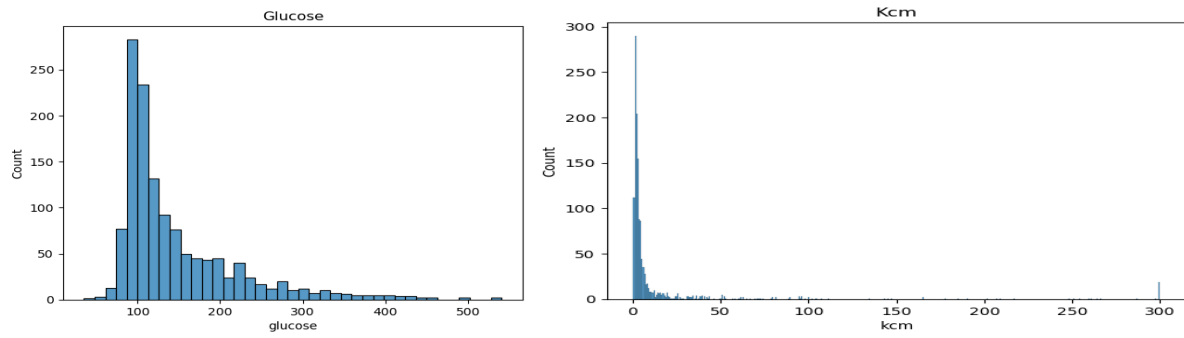


Figure 1.15: The pressure low of the distribution

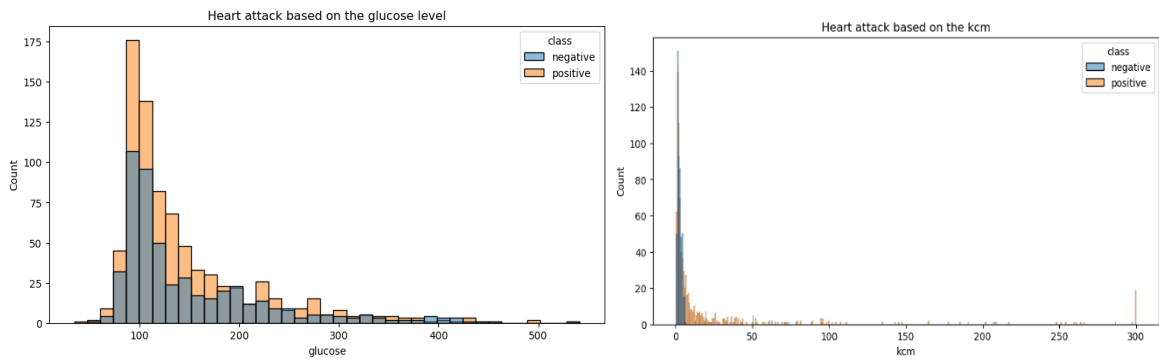


Figure 1.16: The KCM distribution of the data

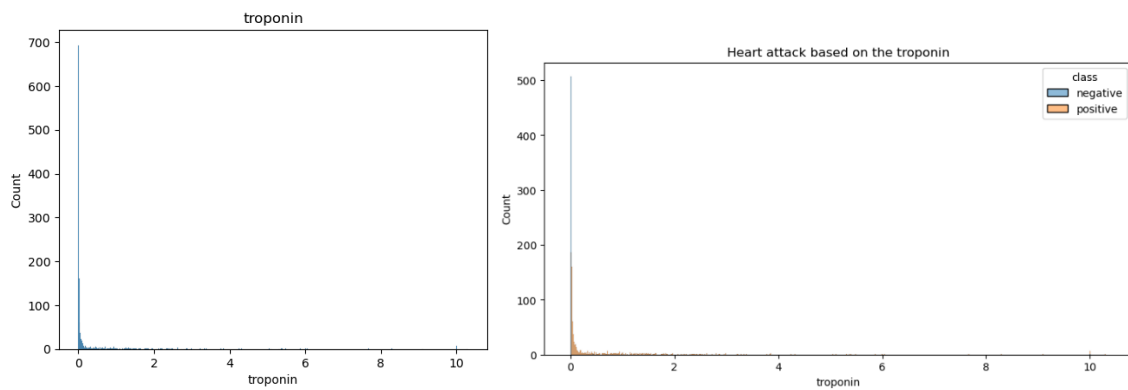


Figure 1.17: The troponin distribution of the data

Calculating the correlation matrix and visualizing the heat map.

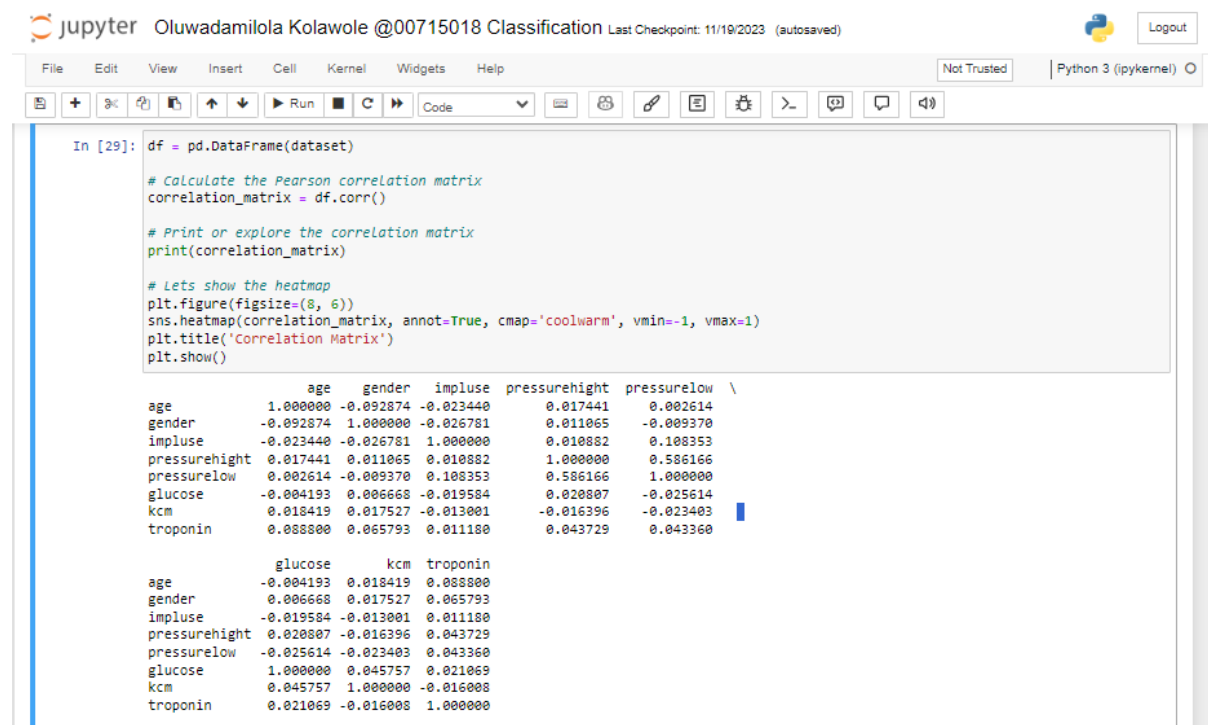


Figure 1.18: Correlation Matrix

From the heat map generated, Age, kcm and troponin have the best correlation with the target or output variable 'Class' (0.24, 0.22 and 0.23) respectively.

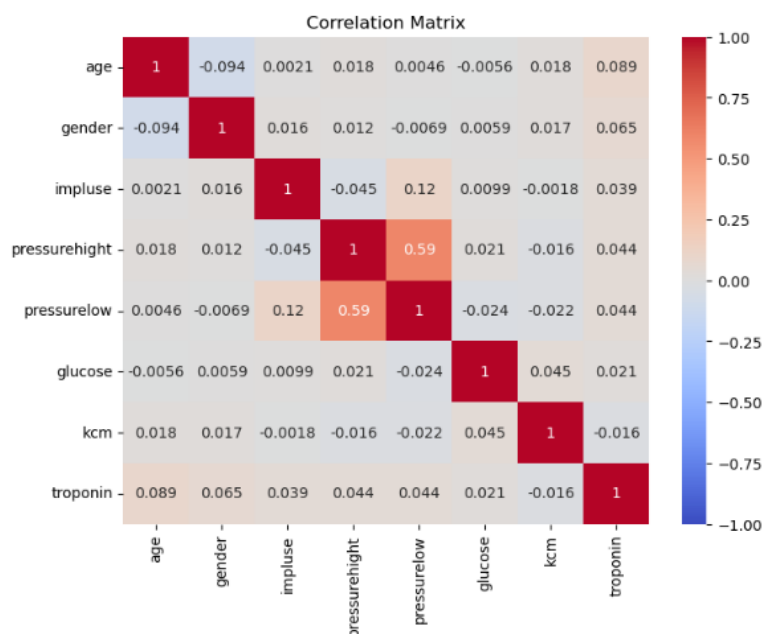
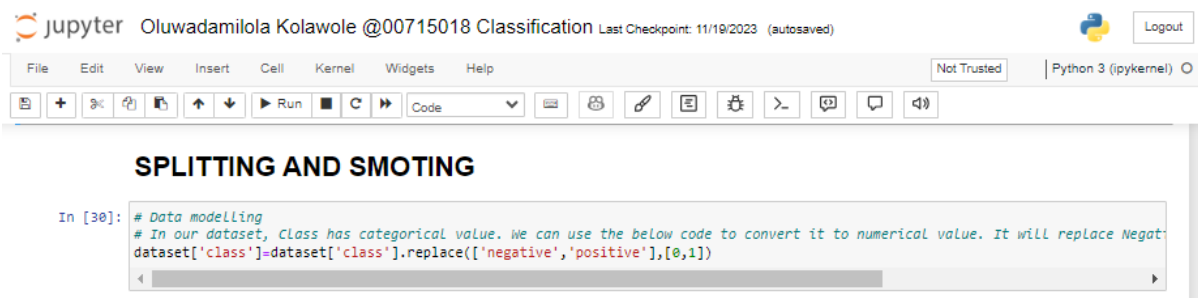


Figure 1.19: Correlation heat map

The Class label or dependent variable was changed to binary (0,1).

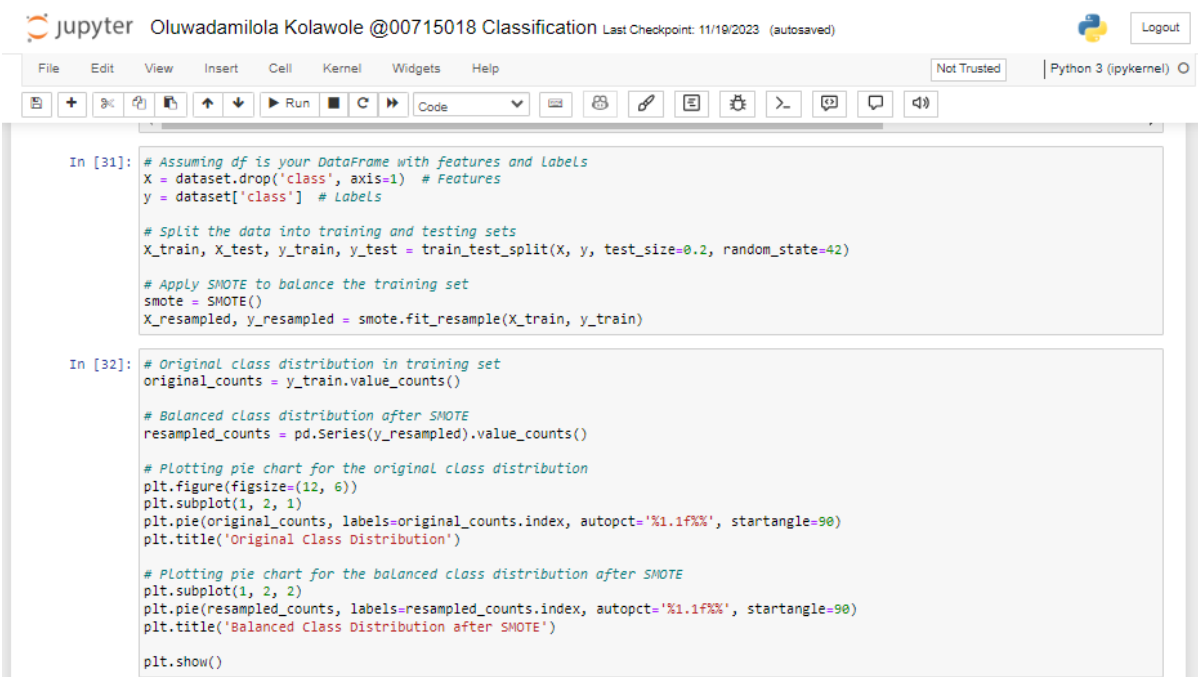


The image shows a Jupyter Notebook interface with a single code cell. The notebook title is "Oluwadamilola Kolawole @00715018 Classification". The code in the cell is as follows:

```
In [30]: # Data modelling
# In our dataset, Class has categorical value. We can use the below code to convert it to numerical value. It will replace Negative
dataset['class']=dataset['class'].replace(['negative','positive'],[0,1])
```

Figure 1.20: Class Labelling

The data was balanced using SMOTE technique and splitted into the X and Y variable using the code below:



The image shows a Jupyter Notebook interface with two code cells. The notebook title is "Oluwadamilola Kolawole @00715018 Classification". The code in the cells is as follows:

```
In [31]: # Assuming df is your DataFrame with features and Labels
X = dataset.drop('class', axis=1) # Features
y = dataset['class'] # Labels

# 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)

# Apply SMOTE to balance the training set
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

In [32]: # Original class distribution in training set
original_counts = y_train.value_counts()

# Balanced class distribution after SMOTE
resampled_counts = pd.Series(y_resampled).value_counts()

# Plotting pie chart for the original class distribution
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.pie(original_counts, labels=original_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Original Class Distribution')

# Plotting pie chart for the balanced class distribution after SMOTE
plt.subplot(1, 2, 2)
plt.pie(resampled_counts, labels=resampled_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Balanced Class Distribution after SMOTE')

plt.show()
```

Figure 1.21: Splitting and balancing the data

The input variables represent X while the output variable represent y. The train test split made the differentiation necessary because training your target variable is illogical.

SMOTE (Synthetic Minority Oversampling Technique) is a technique to oversample the minority class in a dataset (Chawla et al., 2002). When training a classifier on a dataset with imbalanced class distributions—for example, when there are significantly more values of one class than another, to balance the class value, the provided code uses the SMOTE class to generate samples of the minority class, which are then added to the training set.

One useful tool from the Python sklearn library for machine learning tasks is the `train_test_split` function (Pedregosa et al., 2011). It returns four variables after accepting the test size and the input features (x and y) as arguments. The four variables are divided into two subsets: "x_train" and "y_train" for training the model, and "x_test" and "y_test" for assessing its performance. Since the test size in this instance is set at 0.2, 20% of the data will be used for testing and the remaining 80% for training. Setting the random state to 42 guarantees that the data is divided deterministically and stays consistent throughout several code runs.

In summary, this code is a typical experimental process in machine learning tasks; it prepares the dataset for training and testing the model.

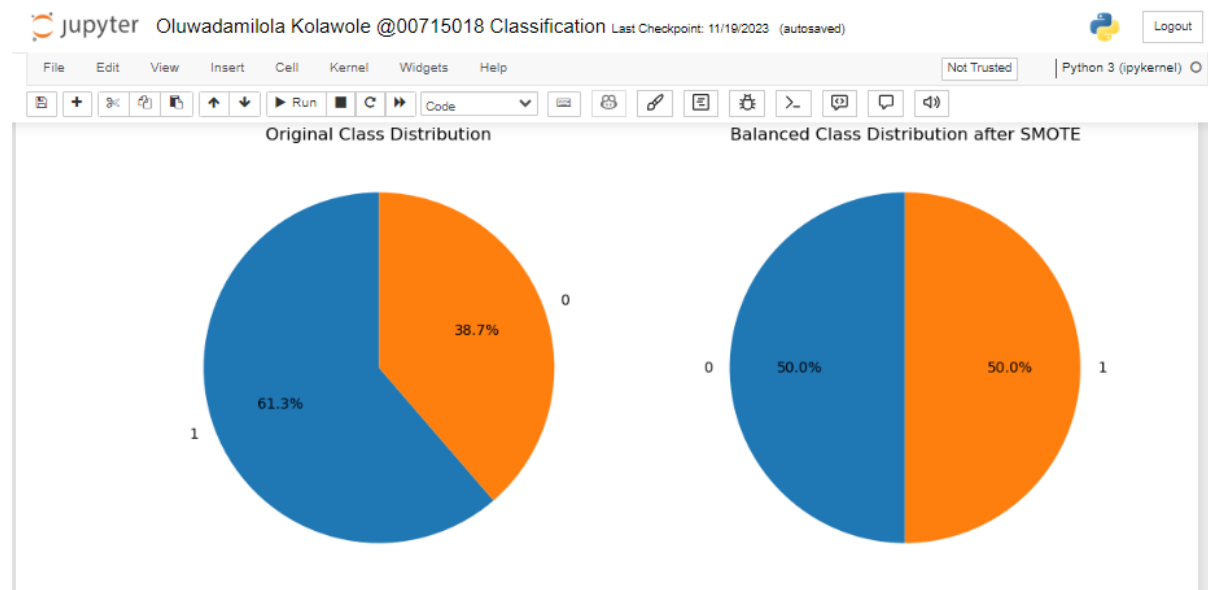


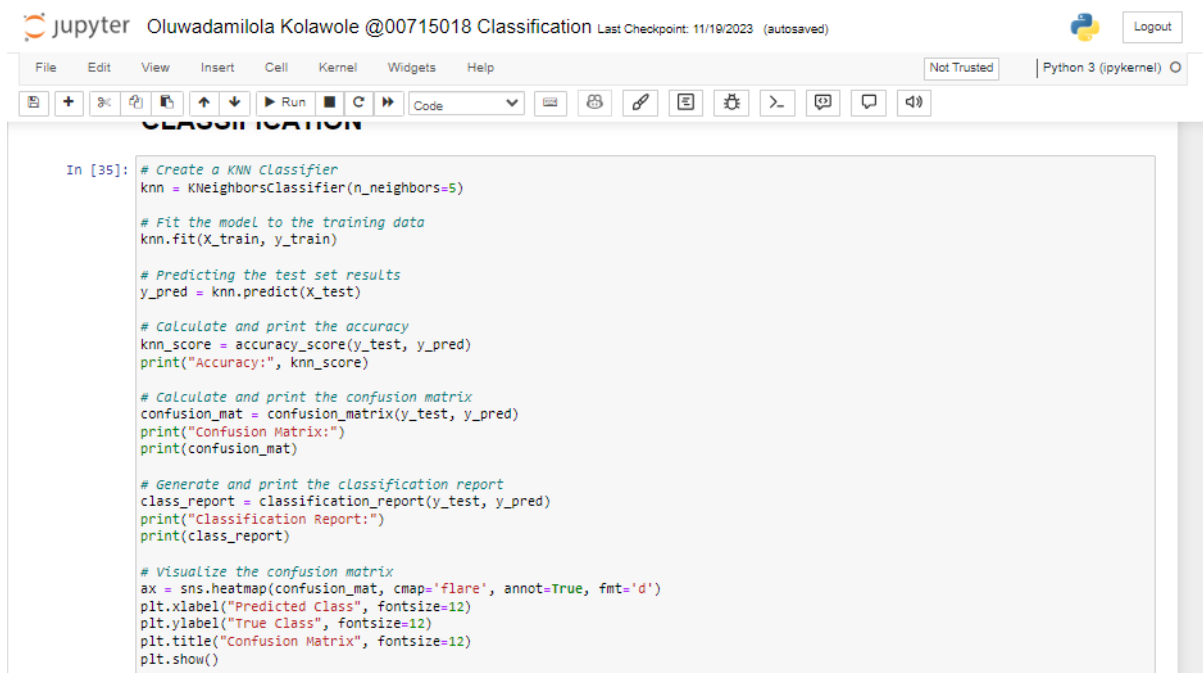
Figure 1.22: Class distribution before and after balancing

1.4 Classification algorithms

The K-Nearest neighbor classifier and decision tree classifier are popular algorithms that can be used for classification tasks. KNN is a supervised learning algorithm that can be applied to classification problems. It is regarded as one of the most basic algorithms for machine learning. In order to classify a new data point, it first locates the k training examples that are closest to it. Then, it uses the average of the k nearest neighbors to determine the most common class. KNN makes predictions using a distance metric like Euclidean distance to find the nearest neighbors.

The decision tree classifier work by recursively partitioning the input space into smaller regions or subsets based on the features that best split the data i.e a tree like model with possible outcomes in a diagram.

1.4.1 Classification using K-Nearest Neighbor



The image shows a Jupyter Notebook interface. At the top, the header includes the Jupyter logo, the username 'Oluwadamilola Kolawole @00715018', the title 'Classification', and the last checkpoint '11/19/2023 (autosaved)'. There is a 'Logout' button on the right. Below the header is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. To the right of the menu bar are 'Not Trusted' and 'Python 3 (ipykernel)' indicators. Below the menu bar is a toolbar with various icons for file operations, running, and viewing. The main area of the notebook contains a code cell with the following Python code:

```
In [35]: # Create a KNN Classifier
knn = KNeighborsClassifier(n_neighbors=5)

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

# Predicting the test set results
y_pred = knn.predict(X_test)

# Calculate and print the accuracy
knn_score = accuracy_score(y_test, y_pred)
print("Accuracy:", knn_score)

# Calculate and print the confusion matrix
confusion_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion_mat)

# Generate and print the classification report
class_report = classification_report(y_test, y_pred)
print("Classification Report:")
print(class_report)

# Visualize the confusion matrix
ax = sns.heatmap(confusion_mat, cmap='flare', annot=True, fmt='d')
plt.xlabel("Predicted Class", fontsize=12)
plt.ylabel("True Class", fontsize=12)
plt.title("Confusion Matrix", fontsize=12)
plt.show()
```

Figure 1.23: K-Nearest Neighbor Classifier

The code above creates the KNN classifier with 5 neighbors, fits the model to the training data ('X_train' and 'y_train'), predicts the label for the test data('X_test'), calculates and print the accuracy score by comparing the predicted labels ('y_pred') against the true label ('y test'), calculates and print the confusion matrix showing the True Positives, True Negatives, False Positives and False Negatives.

Generates and prints the classification report showing the precision, recall, F1-score and support for each class. Visualizing the heatmap for the confusion matrix.

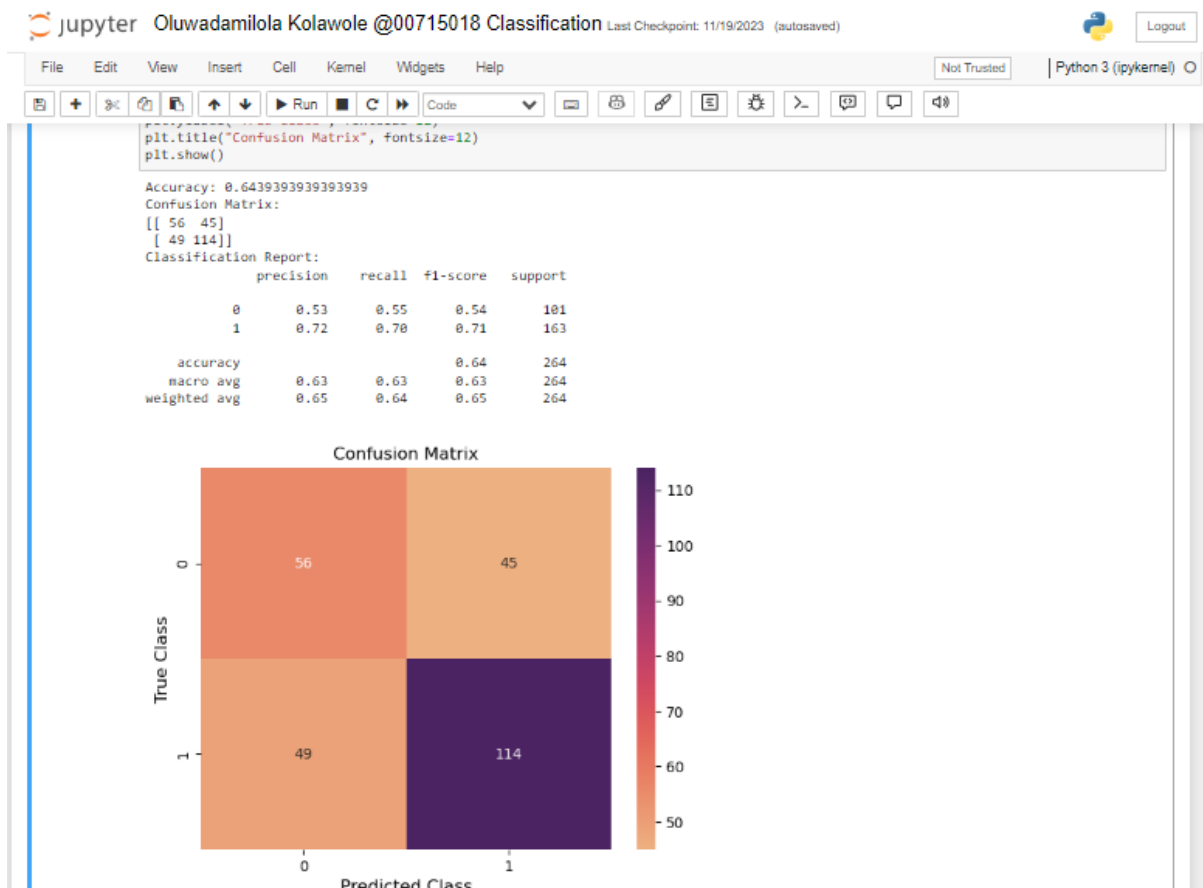


Figure 1.24: K-Nearest Neighbor Classification report and Confusion matrix

1.4.2 Classification using Decision Tree Classifier

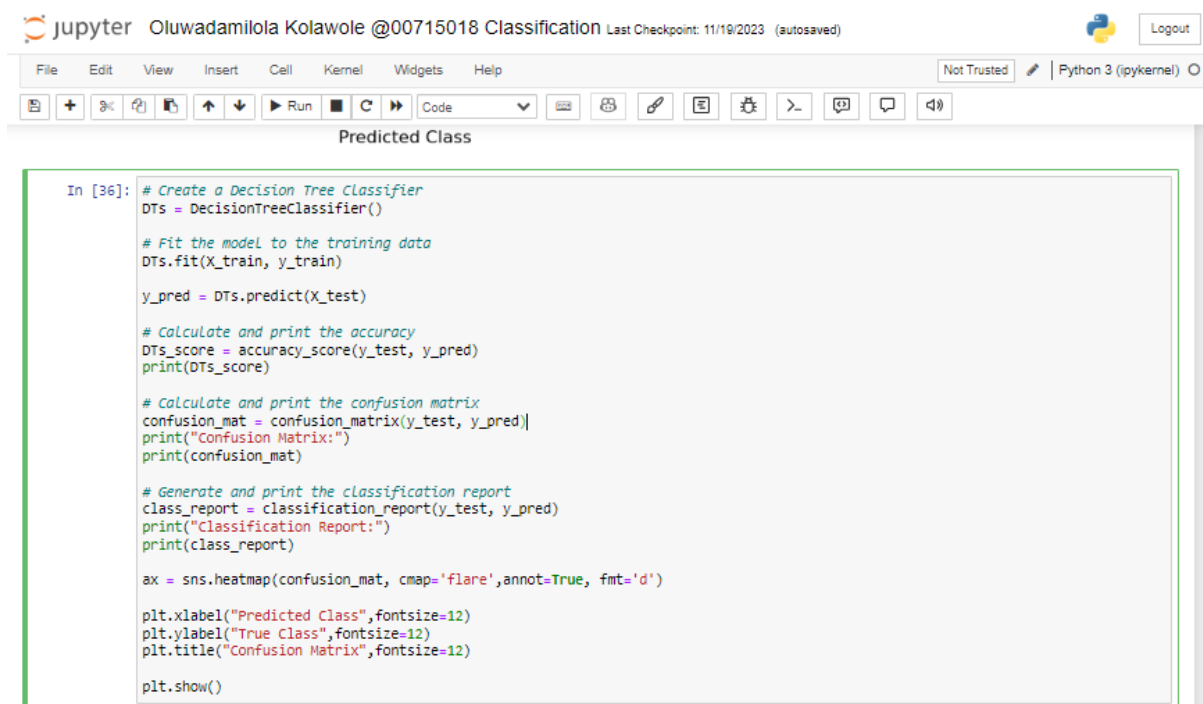


Figure 1.25: Decision Tree Classifier

The code above creates the Decision Tree Classifier, fits the model to the training data ('X_train' and 'y_train'), predicts the label for the test data('X_test'), calculates and print the accuracy score by comparing the predicted labels ('y_pred') against the true label ('y_test'), calculates and print the confusion matrix showing the True Positives, True Negatives, False Positives and False Negatives.

Generates and prints the classification report showing the precision, recall, F1-score and support for each class. Visualizing the heatmap for the confusion matrix.

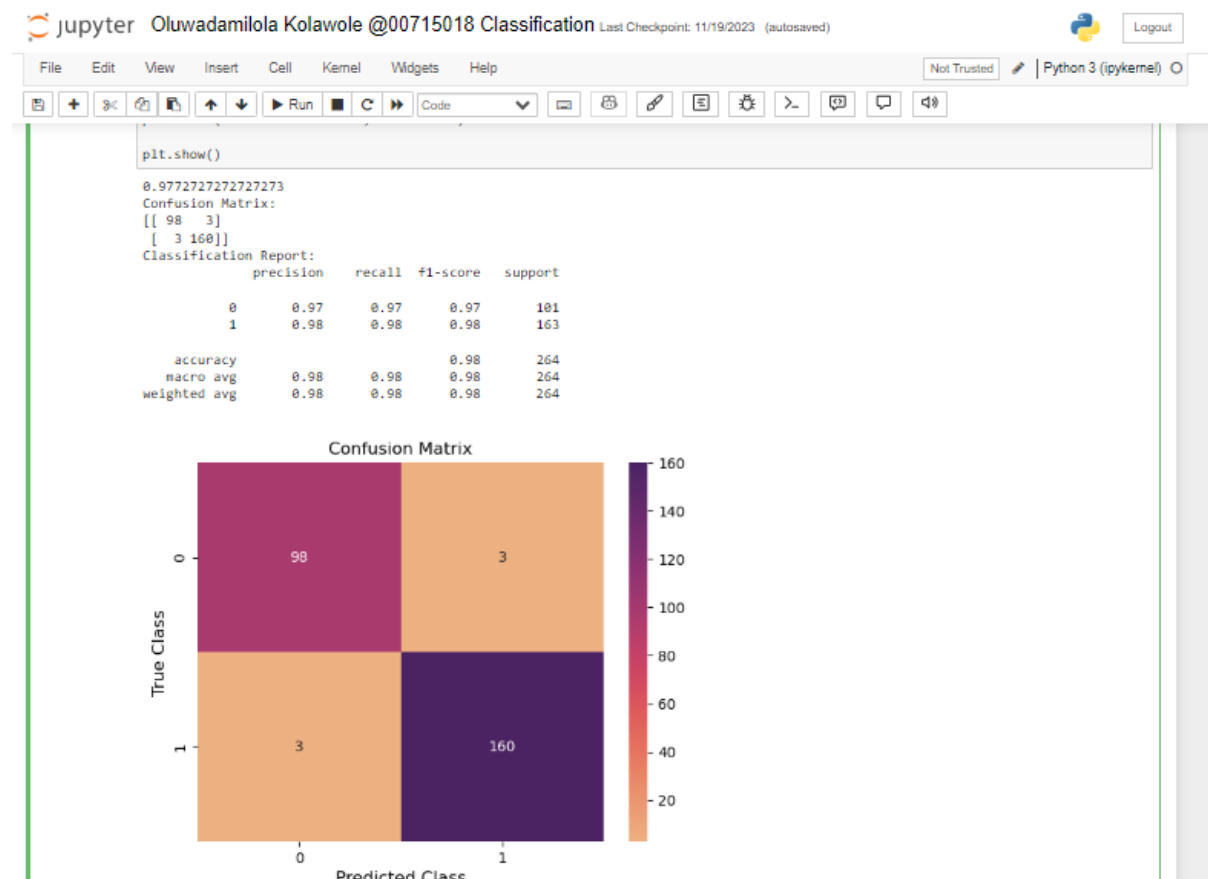


Figure 1.26: Decision Tree Classification report and Confusion matrix

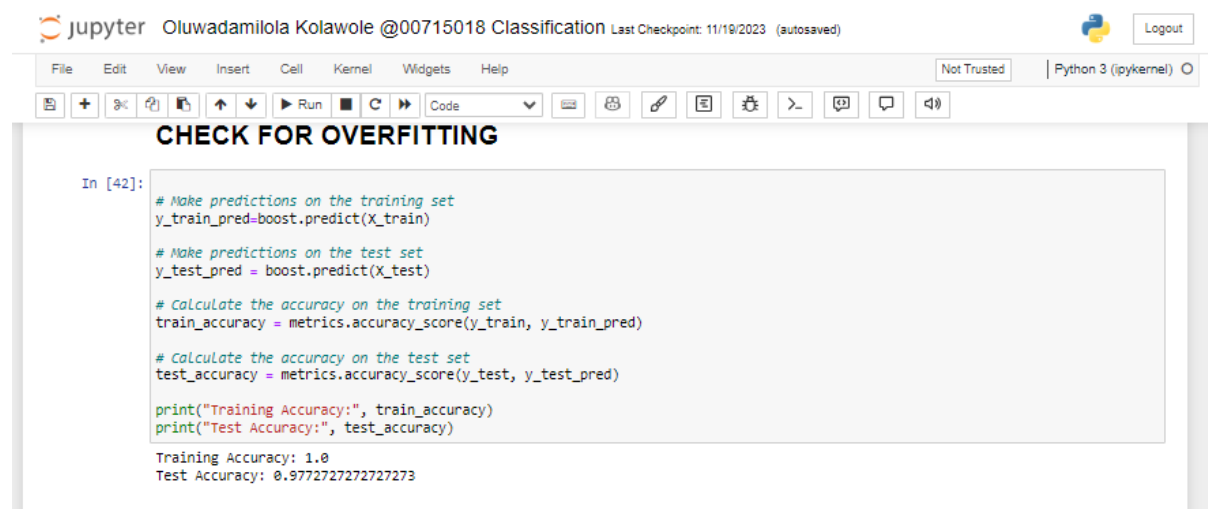


Figure 1.27: Checking for Overfitting in the data

1.5 Comparisons and discussion of results

The decision tree classifier has a prediction accuracy of 97.72% while the K-Nearest Neighbor classifier has a prediction accuracy of 64.39%. One of the potential reasons for the higher accuracy in Decision Trees classifier is it works better with categorical features and clear decision boundaries in the data while KNN works better with continuous features and complex decision spaces.

Moreover, KNN struggles in high dimensional spaces due to the curse of dimensionality. Decision Trees are more robust to large feature spaces. KNN is sensitive to noisy data and irrelevant features since it relies on distances between points. Noisy or irrelevant features can impact its performance negatively by affecting the determination of nearest neighbors.

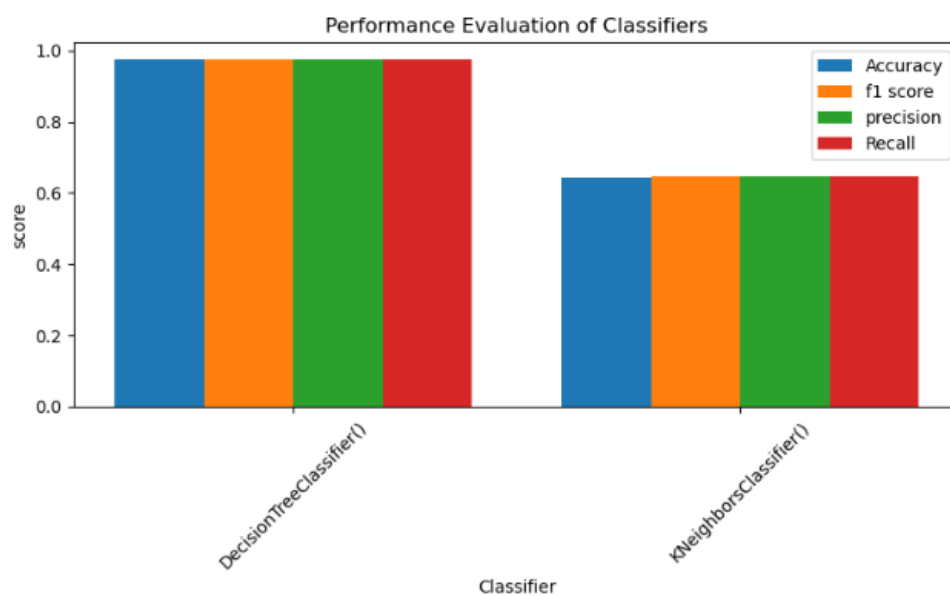


Figure 1.28: The classifier comparison

1.6 Conclusion

Based on the results of the classification using both a K-Nearest Neighbors (KNN) classifier and a Decision tree Classifier, I would propose that using a Decision Tree for predicting the presence of heart attack as it has great classification report. The Decision Tree Classification has a higher prediction accuracy (98%) compared to the K-Nearest Neighbors (KNN) classifier (64%). This means that the model can correctly predict the probability of a patient having heart attack or not.

However, the medical practitioner can predict the presence of heart attack from the age, kcm and troponin due to strong correlation between the three variables and class column. The model's performance is great.

1.7 Introduction: Azure Machine learning

Microsoft offers tools and services to build, deploy, compute, store, network, develop, and manage applications and services through its Azure computing platform and services. These machine learning-focused services include Azure Databricks, Azure Notebooks, Azure Stream Analytics, Azure Functions, and Azure DevOps.

A fully managed cloud service for creating, honing, and implementing machine learning models is called Azure Machine Learning. A visual tool for creating, honing, and implementing machine learning models is Azure Machine Learning Studio. Among its benefits are its global reach, data protection, security and compliance, scalability, flexibility, and cost optimization. When all is said and done, Azure provides a vast array of resources and services for developing and deploying machine learning models in the cloud.

1.8 Preparation of Dataset

The dataset used is the same one used in the classification task under both K-Nearest Neighbors and Decision Tree.

The dependent and independent variables stays the same.

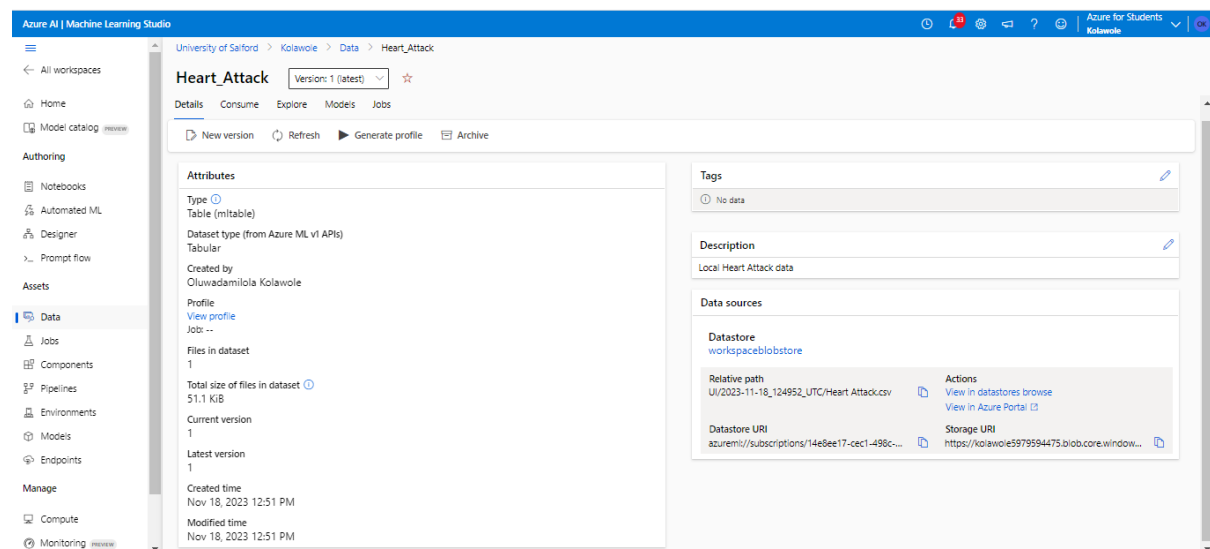


Figure 1.29: Loading the dataset

age	gender	impluse	pressurehight	pressurelow	glucose	kcm	troponin	class
64	1	66	160	83	160	1.8	0.012	negative
21	1	94	96	46	296	6.75	1.06	positive
55	1	64	160	77	270	1.99	0.003	negative
64	1	70	120	55	270	13.87	0.122	positive
55	1	64	112	65	300	1.08	0.003	negative
58	0	61	112	58	87	1.83	0.004	negative
32	0	40	179	68	102	0.71	0.003	negative
63	1	60	214	82	87	300	2.37	positive
44	0	60	154	81	135	2.35	0.004	negative
67	1	61	160	95	100	2.84	0.011	negative
44	0	60	166	90	102	2.39	0.006	negative
63	0	60	150	83	198	2.39	0.013	negative
64	1	60	199	99	92	3.43	5.37	positive

Figure 1.30: Exploring the dataset

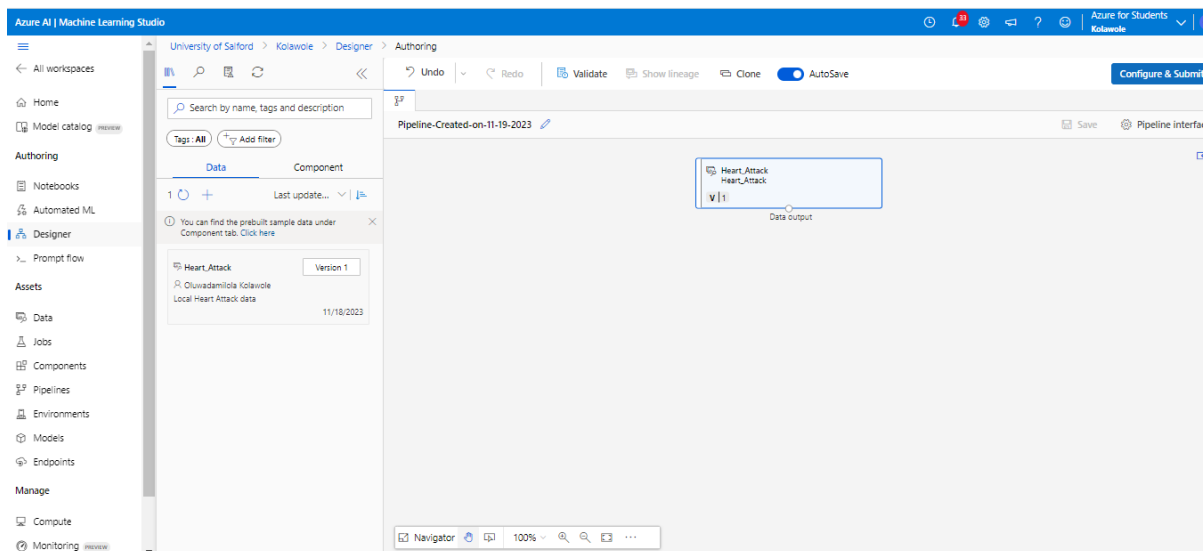


Figure 1.31: Pipeline creation in designer

From the explored dataset, the shows that my dataset dependent/target variable/class label is in categorical form so as to convert it to binary label, the column observations was replaced using the execute python script function as shown below

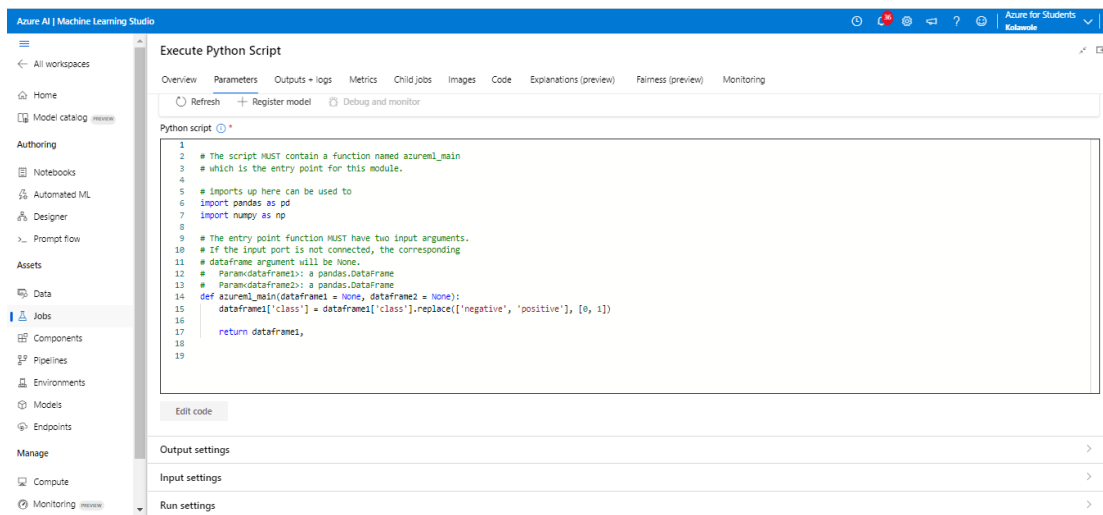


Figure 1.32: The execute python script

Another name for data normalisation is scaling. This is the next stage of preprocessing, and it can improve the performance of the model by scaling the data to have a mean of 0 and a standard deviation of 1. They also selected the columns that would be changed.

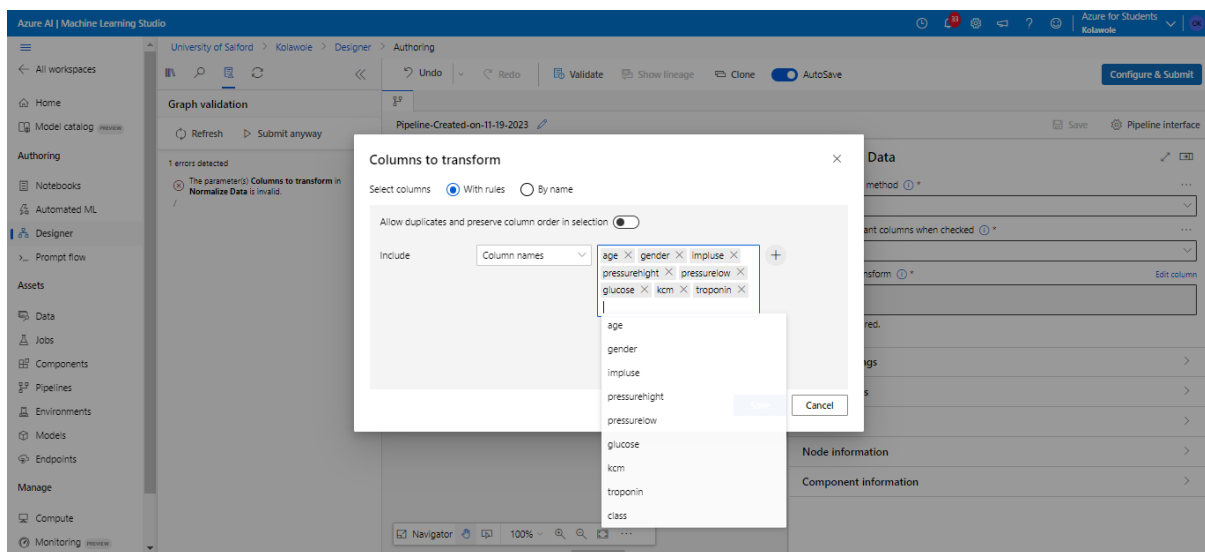


Figure 1.33: The transform columns

The `train_test_split` function is then used to split the data before training it.

Python-based processes and Azure-based machine learning processes are identical.

1.9 Classification Algorithms

1.9.1 Two Class Neural Network

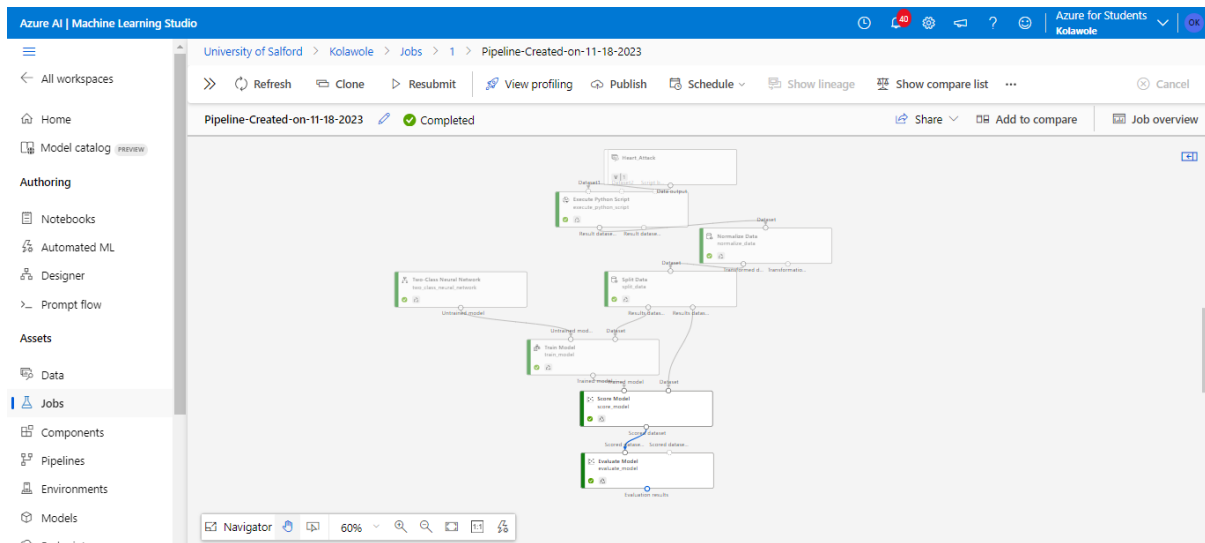


Figure 1.34: Two Class Neural Network Pipeline

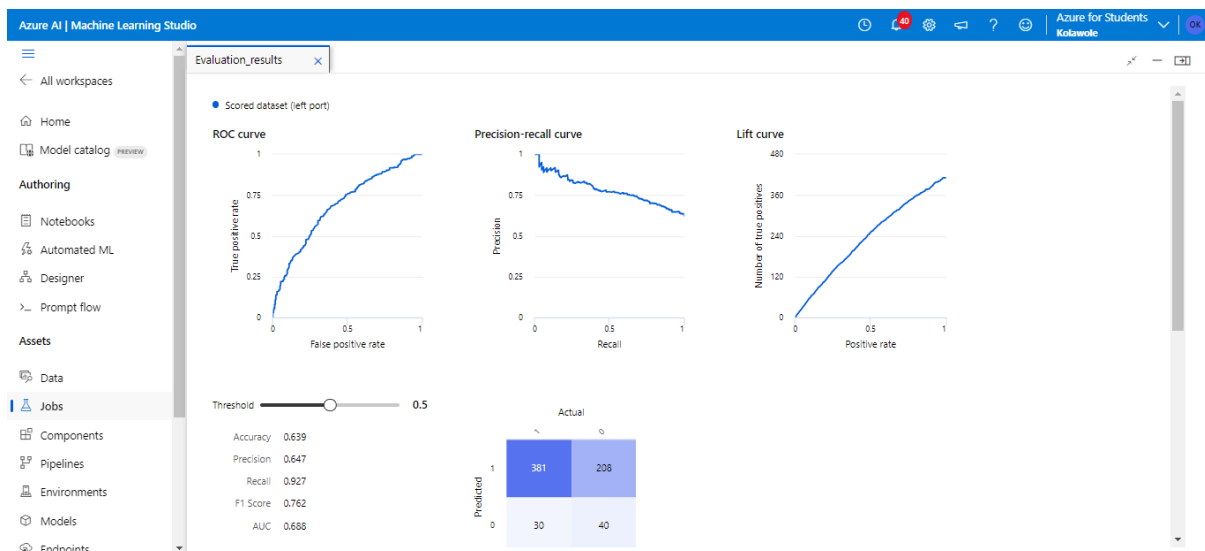


Figure 1.35: Two Class Neural Network Classification report

1.9.2 Two Class Decision Forest

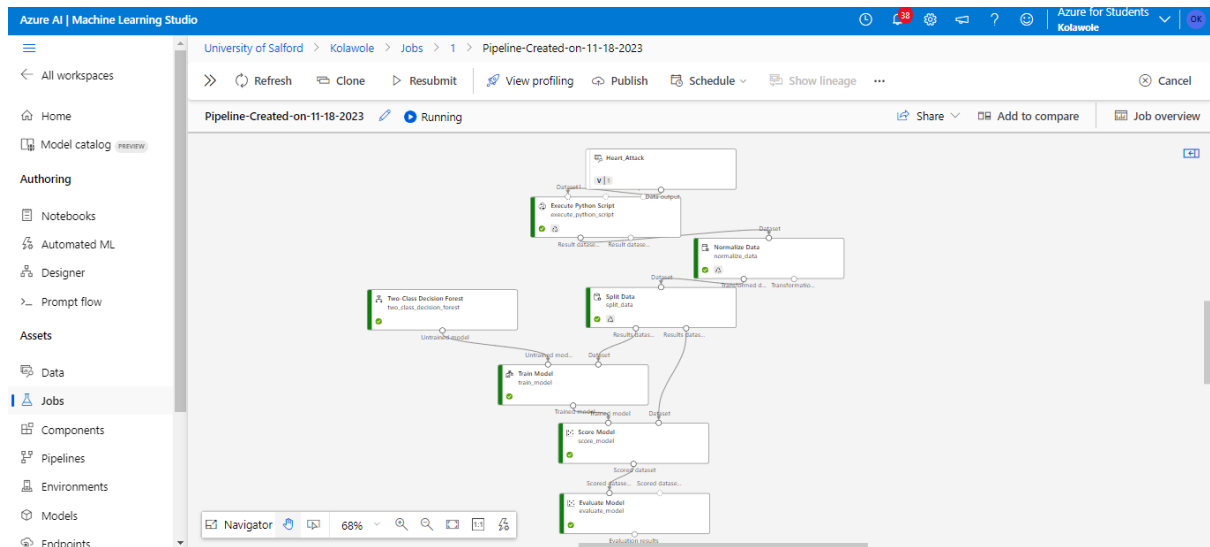


Figure 1.36: Two Class Decision Forest Pipeline

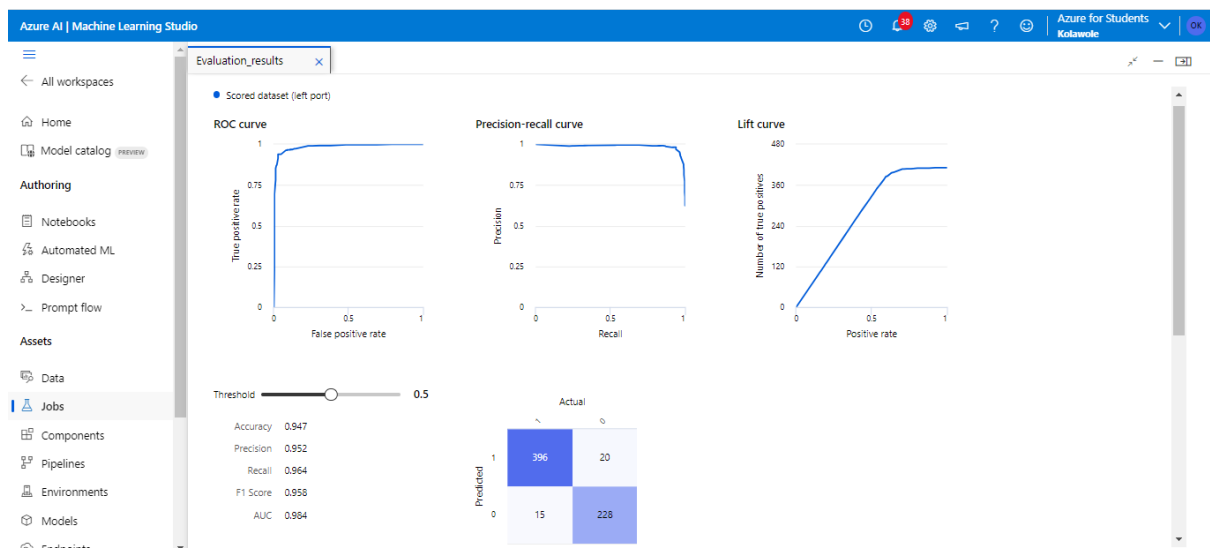


Figure 1.37: Two Class Decision Forest Classification report

The classification report shows the summary of the model's precision, recall, and f1-score for each classification model.

Classification Metrics	Description
Accuracy	The percentage of cases that are correctly classified out of all the instances.
Precision	The ratio of correctly predicted positive observations to the total positive observations that were predicted.

Recall	The ratio of correctly predicted positive observations to the actual positives in the dataset.
F1 Score	The balance between precision and recall
AUC (Area Under Curve)	The ability of a model to differentiate between classes.

Table 1.2: The classification metrics

	Accuracy	Precision	Recall	F1 Score	AUC
Neural Network	0.639	0.647	0.927	0.762	0.688
Decision Forest	0.947	0.952	0.964	0.958	0.984

Table 1.3: The classification algorithms result

The table above shows the classification report used to evaluate the performance of the models on the test set.

The Decision Forest algorithms performs better on the test dataset having higher values across all the evaluation metrics.