MINISTRY OF EDUCATION
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY
COLLEGE OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

وزارة التعليم
جامعة الإمام عبدالرحمن بن فيصل
كلية علوم الحاسب وتقنية المعلومات

جامعة الإمام عبد الرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

Imam Abdulrahman bin Faisal University

College of Computer Science and Information Technology

CIS 517 - Data Mining & Data Warehousing

Academic Year (2023-2024) – First Semester

Project Title: Heart Attack Prediction Models using Machine Learning Algorithms

Authors:

Project Superviser: Dr. Irfan Ullah Abdur Rab

Department of Computer Science, Imam Abdulrahman bin Faisal University

Students:

Zeina Moammer Alabido(Leader)

Department of Computer Science, Imam Abdulrahman bin Faisal University

Reem Ali Alaqeel

Department of Computer Science, Imam Abdulrahman bin Faisal University

Nourah Hesham Aldhuwaihi

Department of Computer Science, Imam Abdulrahman bin Faisal University

Afnan Turki Rayyani

Department of Computer Science, Imam Abdulrahman bin Faisal University

Table of Content

Table of Contents

Table of Content: 2

Table of Figures 3

Table of Tables: 4

1. Abstract 5
2. Introduction 6
3. Review of Related Literatures 8
4. Description of Proposed Model 10
5. Description of Dataset 14
6. Experimental setup 28
7. Performance Measure 28
8. Results and Discussion 30
9. Conclusion 30
10. Acknowledgment 31
11. Datasets 32
12. References: 33

Table of Figures

Figure 1: decision tree diagram..... 10
Figure 2: the NB mathematical formula 11
Figure 3: the working of K-Means clustering..... 12
Figure 4: Dataset after has been combined into a single sheet. dataset in this step is still not processed, and it contains noise, and incorrect data should be handled. 14
Figure 5: Python on Google Collab to start work on the dataset. 15
Figure 6: In this figure, dataset has been imported to start work on it. Notice that data at the beginning it contains 901 record. 15
Figure 7: Description about dataset. 16
Figure 8: Drop unnecessary column. 17
Figure 9: Change data type from Object type to Int Type for some columns, and then fill in missing values using fillna() function. 17
Figure 10: Remove tuples that contains ' ? ' Symbol. 17
Figure 11: Description of dataset. 18
Figure 12: Box-Plot before removing outliers. 18
Figure 13: after trying to remove outlier values. 19
Figure 14: Z-Score calculation and replacing outlier with median value. 19
Figure 15: This figure shows the number of data samples their result is positive for the heart attack which is higher than negative result represented by 0..... 21

Table of Tables

Table 1: Table shows the relationship between the target column and each attribute in dataset. 21

Table 2: Datasets used in Machine Learning models. 32

1. Abstract

Machine learning and data mining-based techniques to heart disease prediction and diagnosis might be extremely useful in the clinic, but they would be extremely difficult to build. Most countries suffer from a lack of cardiovascular competence and a high rate of misdiagnosis, which might be addressed by establishing reliable and efficient early-stage heart disease prediction through analytical support of clinical decision-making using digitized patient information. The goal of this work was to find the most accurate machine learning classifiers for such diagnostic applications. Several supervised machine-learning and unsupervised machine-learning algorithms were used and compared for prediction of heart disease performance and accuracy. Except for MLP and KNN, all applied algorithms estimated feature significance ratings for each feature. To select people with high heart disease predictions, all of the features were rated based on their importance score. The RF technique obtained 100% accuracy, sensitivity, and specificity utilizing a heart disease dataset collected from Kaggle three-classification based on k-nearest neighbor (KNN), decision tree (DT), and random forests (RF) algorithms. As a result, we discovered that a relatively basic supervised machine learning algorithm can be used to predict heart disease with very high accuracy and excellent potential use.

2. Introduction

The term "Big Data" has emerged due to the rapid increase in the volume of data that is difficult to manage. Big Data encompasses both structured and unstructured data and requires the adoption of data mining and data warehousing technologies to derive useful information from it.

Data mining is a process that involves extracting meaningful and valuable insights from large datasets using statistical, mathematical, and analytical techniques. These techniques enable the identification of patterns, trends, and relationships within the data .Data mining can be applied to specific domains to conduct studies and gain insights into various topics. In our project, we will be focusing on heart attack diseases, which is a critically important health issue that needs to be addressed. By studying these diseases and understanding the underlying causes, we aim to minimize or eliminate the probability of heart attacks and promote better heart health. A heart attack occurs when the blood flow to the heart is obstructed or reduced by fat deposits around the coronary arteries. This blockage prevents the artery from effectively supplying blood and oxygen to the heart, leading to tissue damage and possible death. There are several risk factors associated with heart attacks, including age, tobacco use, high blood pressure, high cholesterol, obesity, diabetes, metabolic syndrome, family history, unhealthy lifestyle, stress, illegal drug use, history of preeclampsia, and autoimmune conditions. To conduct our study, we require a dataset that contains data from which we can predict the possibility of a heart attack .

In a previous study titled "Heart Attack Prediction using Data Mining Techniques," the researchers aimed to design and develop a model for predicting heart attack diseases using data mining techniques. They employed an unsupervised classification model called Fuzzy C Means and utilized a dataset consisting of 13 attributes derived from patients' medical records to determine the risk of a heart attack. For our project, we will employ data mining techniques to extract valuable information from three selected datasets obtained from Kaggle. The first dataset contains 295 samples, while the second and third datasets contain 304 samples each. Preprocessing tools will be utilized to remove any abnormalities from the data, enabling effective data analysis and mining techniques to be applied. The insights gained from this study are significant in terms of

understanding the prevalence of heart attacks, raising public awareness, and providing guidance on preventive measures and self-care. In summary, our project focuses on utilizing data mining techniques to study heart attack diseases. By analyzing and extracting insights from the selected datasets, we aim to gain a better understanding of the prevalence and risk factors associated with heart attacks, ultimately promoting heart health and well-being.

3. Review of Related Literatures

The paper [1] suggests an approach for feature extraction and ensemble deep learning for identifying and predicting the outcome of heart disease. The application of classifiers like Random Forest and Gaussian NB for classification is examined in this article. The findings demonstrate that the suggested method outperforms existing systems in terms of accuracy, achieving 98.5% for heart disease prediction. The paper also includes confusion matrices for various features and classifiers. Using ALEXANET features, for instance, the Random Forest classifier achieves an overall classification accuracy of 96.43%. With SIFT features, the Gaussian NB classifier obtains an overall classification accuracy of 45.74%. In summary, the suggested method predicts heart disease with excellent accuracy, and the Random Forest classifier that makes use of ALEXANET characteristics exhibits encouraging outcomes with an accuracy of 96.43%.

The paper [2] addresses how different algorithms and techniques are used to classify heart disease. Multi-Layer Perceptron (MLP), Fluffy Unordered Standard Acceptance Calculation (FURIA), Multinomial Strategic Relapse (MLR), Sequential Minimal Optimization (SMO), Bayes Net, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Artificial Neural Network (ANN), and C4.5 are some of the classifiers that have been mentioned. Depending on the study and dataset used, these classifiers produce different results and levels of accuracy. As an illustration, one study improved the prediction accuracy of MLP by 11%, MLR by 9.3%, FURIA by 9.2%, and C4.5 by 9.4%. Another study found that a hybrid random forest model could predict heart disease with an accuracy of 88.7%. Furthermore, in identifying patients at higher risk, the decision tree classifier demonstrated a sensitivity rate of 93.3% and a specificity rate of 63.6%. keep in mind that the accuracy and outcomes can change based on the dataset, feature choices, preprocessing methods, and assessment metrics applied in each study.

The goal of the paper [3] is to employ machine learning techniques to forecast heart disease. It investigates how supervised learning algorithms can be used to forecast a patient's risk of acquiring heart disease. Several classifiers, including Artificial Neural Networks (ANN), Decision Trees (DT), Support Vector Machines (SVM), Naive Bayes, and Gradient Boosting Classifiers, are mentioned in the study. The study's findings show that, out of the four evaluated algorithms, the Gaussian Naive Bayes classifier had the best accuracy. It was discovered that the Gaussian Naive Bayes classifier had an accuracy of 81.9%. In summary, the study article discusses the characteristics associated with heart disease and how machine learning algorithms are used to make predictions. The most accurate classifier for predicting heart disease was the Gaussian Naive Bayes model.

A comparative analysis of various classifiers for the Heart Disease dataset's classification is covered in the paper [4]. K-Nearest Neighbor (K-NN), Naive Bayes, Decision tree J48, JRip, SVM, Adaboost, Stochastic Gradient Descent (SGD), and Decision Table (DT) classifiers are among the ones that were employed in the investigation. The K-NN (K = 1), Decision tree J48, and JRip classifiers exhibit promising classification accuracy, according to the results. With an accuracy of 99.7073%, the JRip classifier obtained 97.2683%, the Decision tree J48 classifier reached 98.0488%, and the K-NN classifier achieved 99.7073%. These classifiers performed better than Adaboost, Naive Bayes, SGD, SVM, and Decision Table classifiers. For the K-NN, Decision tree J48, and JRip classifiers, the corresponding Kappa statistics were 0.9941, 0.961, and 0.9454. To summarize, the heart disease dataset was classified with high accuracy by the K-NN (K = 1), JRip, and Decision tree J48 classifiers. At 99.7073%, the K-NN classifier had the best accuracy.

4. Description of Proposed Model

1. **Decision tree:** A decision tree is a non-parametric supervised learning approach that can be used for classification as well as regression applications. It has a tree structure that is hierarchical and consists of a root node, branches, internal nodes, and leaf nodes.

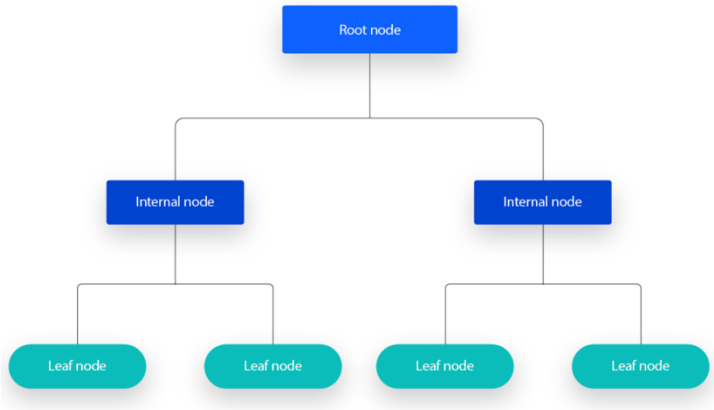


Figure 1: decision tree diagram

A decision tree, as shown in the diagram above, begins with a root node that has no incoming branches. The root node's outgoing branches then feed into the internal nodes, also known as decision nodes. Both node types evaluate the available features to generate homogeneous subsets, which are denoted by leaf nodes or terminal nodes. The leaf nodes represent all the dataset's conceivable outcomes.

Types of decision tree: Hunt's algorithm, which was developed in the 1960s in Psychology to simulate human learning, serves as the foundation for many popular decision tree algorithms, including the following:

1.1: ID3: Ross Quinlan is credited with creating ID3, which stands for "Iterative Dichotomies 3." To evaluate potential splits, this technique uses entropy and information gain as metrics. Some of Quinlan's 1986 research on this algorithm

1.2: C4.5: This algorithm is a subsequent iteration of ID3, which was also created by Quinlan. It may assess split points within decision trees using information gain or gain ratios.

1.3: CART: The acronym CART stands for "classification and regression trees" and was coined by Leo Bierman. This algorithm often makes use of Gini impurity to determine the best characteristic to split on. Gini impurity quantifies how frequently a randomly selected attribute is misclassified. When evaluating Gini impurity, a lower value is preferable [5].

2. **GaussianNB:** Gaussian Naive Bayes is a probabilistic classification algorithm that uses the Bayes theorem with strong independence requirements. Independence in classification refers to the idea that the presence of one value of a feature has no effect on the presence of another (unlike independence in probability theory). When we assume that all the continuous variables associated with each feature have a Gaussian distribution, we utilize Gaussian Nave Bayes. The Gaussian distribution is sometimes known as the Normal distribution [6].

   The Bayes theorem is a formula that provides a conditional probability of an event A occurring if another event B has already occurred. Its mathematical formula is as follows:

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

Figure 2: the NB mathematical formula

3. **K-Means Algorithm:** K-Means Clustering is an Unsupervised Learning technique that divides an unlabeled dataset into clusters. K denotes the number of pre-defined clusters that must be produced during the procedure; for example, if K=2, there will be two clusters; if K=3, there will be three clusters, and so on. The k-means clustering technique is primarily responsible for two tasks:

Iteratively determines the optimal value for K center points or centroids. Each data point is assigned to the nearest k-center. A cluster is formed by data points that are close to a specific K-center. As a result, each cluster comprises datapoints that share some characteristics and is separated from the others.

The graphic below depicts how the K-means Clustering Algorithm works:

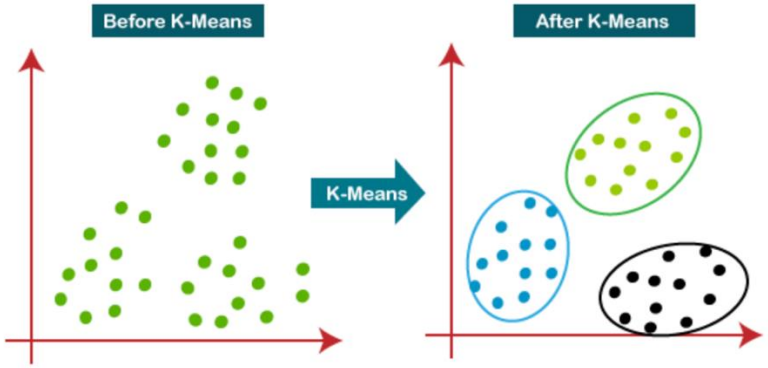


Figure 3: the working of K-Means clustering

How does the K-Means algorithm work?

The following steps describe how the K-Means algorithm works:

Step 1: To determine the number of clusters, choose K.

Step 2: Choose a random set of K points or centroids. (It could be something other than the input dataset).

Step 3: Assign each data point to its nearest centroid, which will result in the formation of the preset K clusters.

Step 4: Determine the variance and assign a new centroid to each cluster.

Step 5: Repeat the third step, reassigning each datapoint to the cluster's new nearest centroid.

Step-6: If reassignment occurs, proceed to step-4; otherwise, proceed to FINISH.

Step 7: The model is complete [7].

5. Description of Dataset

The dataset used in this project was retrieved from Kaggle, and it consists of 3 sub-datasets combined into a single CSV file consisting of 901 data sample. The dataset has 13 attributes 12 are the main attributes, and the last one is the target column to specify if the heart attack is positive or not. Parameters used in this model are age, gender, chest pain, rest blood pressure, cholesterol, fast blood sugar, resting electrocardiographic results, max heart rate, exercise-induced angina, ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels, heart tissue absorb thallium, and label column as the target column. After the 3 datasets has been integrated together, data preprocessing techniques such as imputation, filling the missing values, remove or replace incorrect values, and detect outliers has been implemented on dataset before being used in feature selection, and data splitting step.

Here is a screenshot of the excel sheet before being processed:

Age	Gender	Chest Pain	Rest Blood Pressure	Cholesterol	Fast Blood Sugar	RER	Max Heart Rate	EIA	Oldpeak	Slope	numMajorVessels	Thal	Label
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
44	1	1	120	263	0	1	173	0	0	2	0	3	1
52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
58	0	3	150	283	1	0	162	0	1	2	0	2	1
50	0	2	120	219	0	1	158	0	1.6	1	0	2	1
58	0	2	120	340	0	1	172	0	0	2	0	2	1
66	0	3	150	226	0	1	114	0	2.6	0	0	2	1
43	1	0	150	247	0	1	171	0	1.5	2	0	2	1

Figure 4: Dataset after has been combined into a single sheet. dataset in this step is still not processed, and it contains noise, and incorrect data should be handled.

Using Python on Google Collab to start discovering the dataset more, upload the needed libraries:

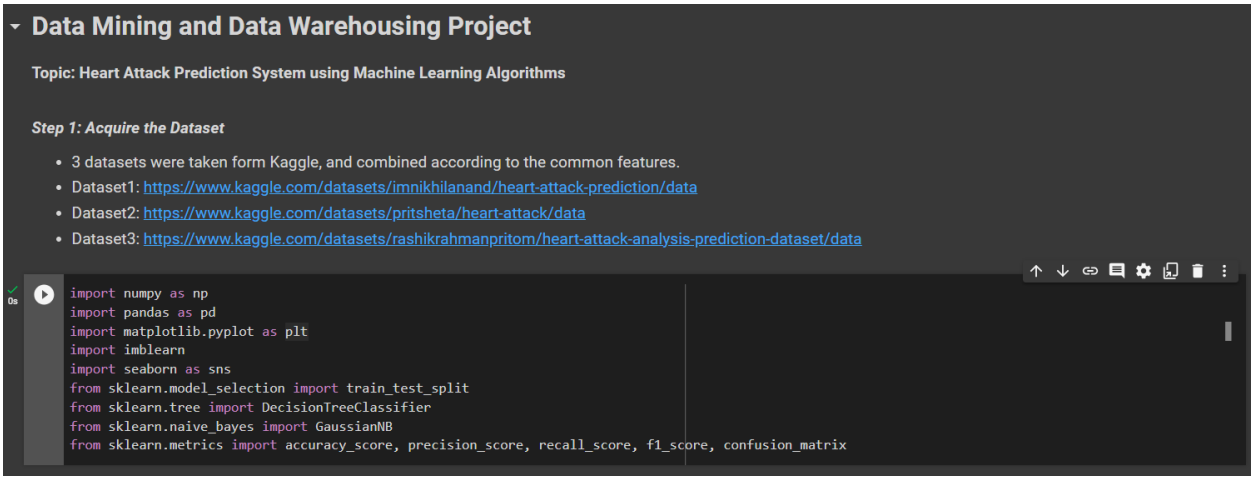
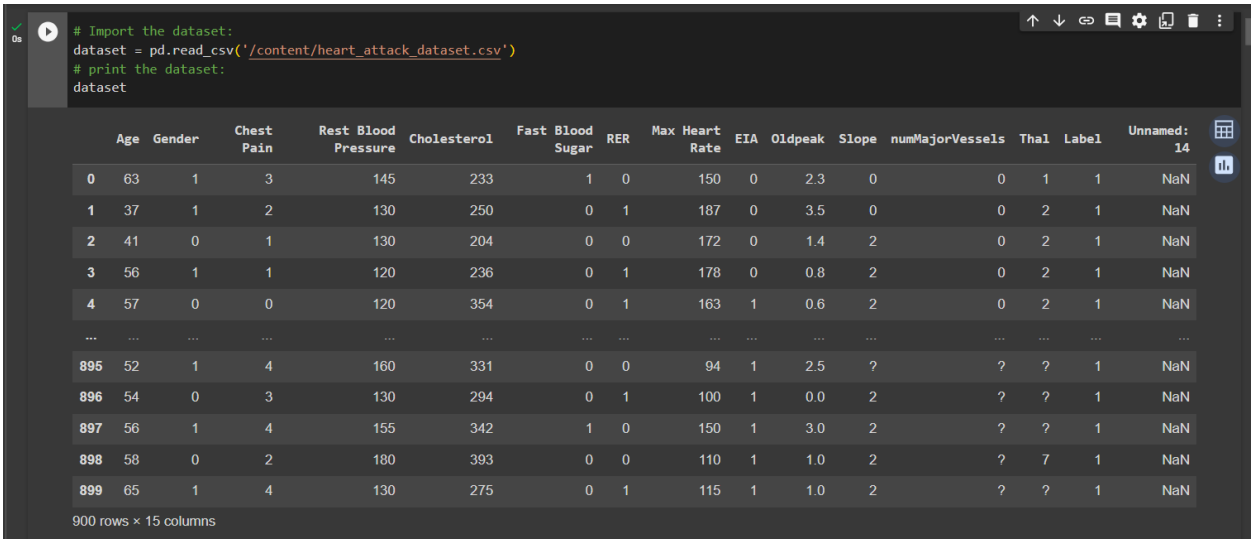


Figure 5: Python on Google Collab to start work on the dataset.

use Pandas to read the excel sheet file.csv:



```
# Import the dataset:
dataset = pd.read_csv('/content/heart_attack_dataset.csv')
# print the dataset:
dataset
```

	Age	Gender	Chest Pain	Rest Blood Pressure	Cholesterol	Fast Blood Sugar	RER	Max Heart Rate	EIA	Oldpeak	Slope	numMajorVessels	Thal	Label	Unnamed: 14
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	NaN
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1	NaN
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1	NaN
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1	NaN
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
895	52	1	4	160	331	0	0	94	1	2.5	?	?	?	1	NaN
896	54	0	3	130	294	0	1	100	1	0.0	2	?	?	1	NaN
897	56	1	4	155	342	1	0	150	1	3.0	2	?	?	1	NaN
898	58	0	2	180	393	0	0	110	1	1.0	2	?	7	1	NaN
899	65	1	4	130	275	0	1	115	1	1.0	2	?	?	1	NaN

900 rows × 15 columns

Figure 6: In this figure, dataset has been imported to start work on it. Notice that data at the beginning it contains 901 record.

The main Data Pre-processing:

- Some Values were "?", so the rows that have these values were removed.
- Object Type Columns converted to INT, which makes the work easy.

- Null values have been handled using the Mean and the Mode of columns.
- The drop column named Unnamed: 14.
- Detect any Outliers and replace them with the Median Value.

The following image description about the dataset, since we have object type data, we want to be able to find the description of this dataset that includes mean, median, mode, and other statistical measurements, consequently, some modifications must be done first.

```
[207] dataset.columns

Index(['Age', 'Gender', 'Chest Pain', 'Rest Blood Pressure', 'Cholesterol',
       'Fast Blood Sugar', 'RER', 'Max Heart Rate', 'EIA', 'Oldpeak', 'Slope',
       'numMajorVessels', 'Thal', 'Label', 'Unnamed: 14'],
      dtype='object')

[208] dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Age                  900 non-null    int64
 1   Gender               900 non-null    int64
 2   Chest Pain           900 non-null    int64
 3   Rest Blood Pressure  900 non-null    object
 4   Cholesterol          900 non-null    object
 5   Fast Blood Sugar     900 non-null    object
 6   RER                  900 non-null    object
 7   Max Heart Rate       900 non-null    object
 8   EIA                  900 non-null    object
 9   Oldpeak              900 non-null    float64
 10  Slope                900 non-null    object
 11  numMajorVessels      900 non-null    object
 12  Thal                 900 non-null    object
 13  Label                900 non-null    int64
 14  Unnamed: 14          0 non-null      float64
dtypes: float64(2), int64(4), object(9)
memory usage: 105.6+ KB

0s completed at 00:23
```

Figure 7: Description about dataset.

Remove “?” from the dataset by applying the imputation technique of deleting rows that contain those symbols, then convert object datatype to INT Type, and fill in missing values:

```
0s
# Remove '?' - romve rows that has this symbol.
dataset = dataset[dataset['Rest Blood Pressure'] != '?']
dataset = dataset[dataset['Cholesterol'] != '?']
dataset = dataset[dataset['Fast Blood Sugar'] != '?']
dataset = dataset[dataset['RER'] != '?']
dataset = dataset[dataset['Max Heart Rate'] != '?']
dataset = dataset[dataset['EIA'] != '?']
dataset = dataset[dataset['Slope'] != '?']
dataset = dataset[dataset['numMajorVessels'] != '?']
dataset = dataset[dataset['Thal'] != '?']
```

Figure 10: Remove tuples that contains ' ? ' Symbol.

```
[179] # Change the type of object to int for specific columns
dataset['Rest Blood Pressure'] = dataset['Rest Blood Pressure'].astype('int')
dataset['Cholesterol'] = dataset['Cholesterol'].astype('int')
dataset['Fast Blood Sugar'] = dataset['Fast Blood Sugar'].astype('int')
dataset['RER'] = dataset['RER'].astype('int')
dataset['Max Heart Rate'] = dataset['Max Heart Rate'].astype('int')
dataset['EIA'] = dataset['EIA'].astype('int')
dataset['Slope'] = dataset['Slope'].astype('int')
dataset['numMajorVessels']= dataset['numMajorVessels'].astype('int')
dataset['Thal'] = dataset['Thal'].astype('int')

# Fill in missing values using the Mean and the Mode
dataset['Age'].fillna(dataset['Age'].mean(), inplace=True)
dataset['Gender'].fillna(dataset['Gender'].mode(), inplace=True)
dataset['Chest Pain'].fillna(dataset['Chest Pain'].mode(), inplace=True)
dataset['Oldpeak'].fillna(dataset['Oldpeak'].mean(), inplace=True)
dataset['Label'].fillna(dataset['Label'].mode(), inplace=True)
dataset['Rest Blood Pressure'].fillna(dataset['Rest Blood Pressure'].mean(), inplace=True)
dataset['Cholesterol'].fillna(dataset['Cholesterol'].mean(), inplace=True)
dataset['Fast Blood Sugar'].fillna(dataset['Fast Blood Sugar'].mode(), inplace=True)
dataset['RER'].fillna(dataset['RER'].mode(), inplace=True)
dataset['Max Heart Rate'].fillna(dataset['Max Heart Rate'].mean(), inplace=True)
dataset['EIA'].fillna(dataset['EIA'].mode(), inplace=True)
dataset['Slope'].fillna(dataset['Slope'].mode(), inplace=True)
dataset['Fast Blood Sugar'].fillna(dataset['Fast Blood Sugar'].mode(), inplace=True)
dataset['numMajorVessels'].fillna(dataset['numMajorVessels'].mode(), inplace=True)
dataset['Thal'].fillna(dataset['Thal'].mode(), inplace=True)
```

Figure 8: Change data type from Object type to Int Type for some columns, and then fill in missing values using fillna() function.

```
[181] # Drop a column by specifying the column name Unnamed: 14  and axis
dataset = dataset.drop('Unnamed: 14', axis=1)
```

Figure 9: Drop unnecessary column.

Finally, we got an accurate Data Description as shown below:

```
dataset.describe()
```

	Age	Gender	Chest Pain	Rest Blood Pressure	Cholesterol	Fast Blood Sugar	RER	Max Heart Rate	EIA	Oldpeak	Slope	numMajorVessels	Thal	Label
count	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000
mean	54.354201	0.683690	0.971993	131.654036	246.230643	0.148270	0.527183	149.561779	0.327842	1.040362	1.400329	0.728171	2.321252	0.545305
std	9.072030	0.465419	1.037676	17.525057	51.751687	0.355661	0.525428	22.963216	0.469814	1.159308	0.615691	1.021347	0.640178	0.498354
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	7.000000	1.000000

Figure 11: Description of dataset.

One of the major options that can be used to detect an outlier is to draw a box plot and see how data are represented. The graph of the box plot shows some outlier values in some columns. These outliers can be handled in different ways, and the one used in this dataset is the Z-Score measurement. By finding a Z-score of each column and specifying the threshold value, outlier values can be detected. These outliers can be replaced with the median value of that specified column. This is the Box-Plot before removing the outliers:

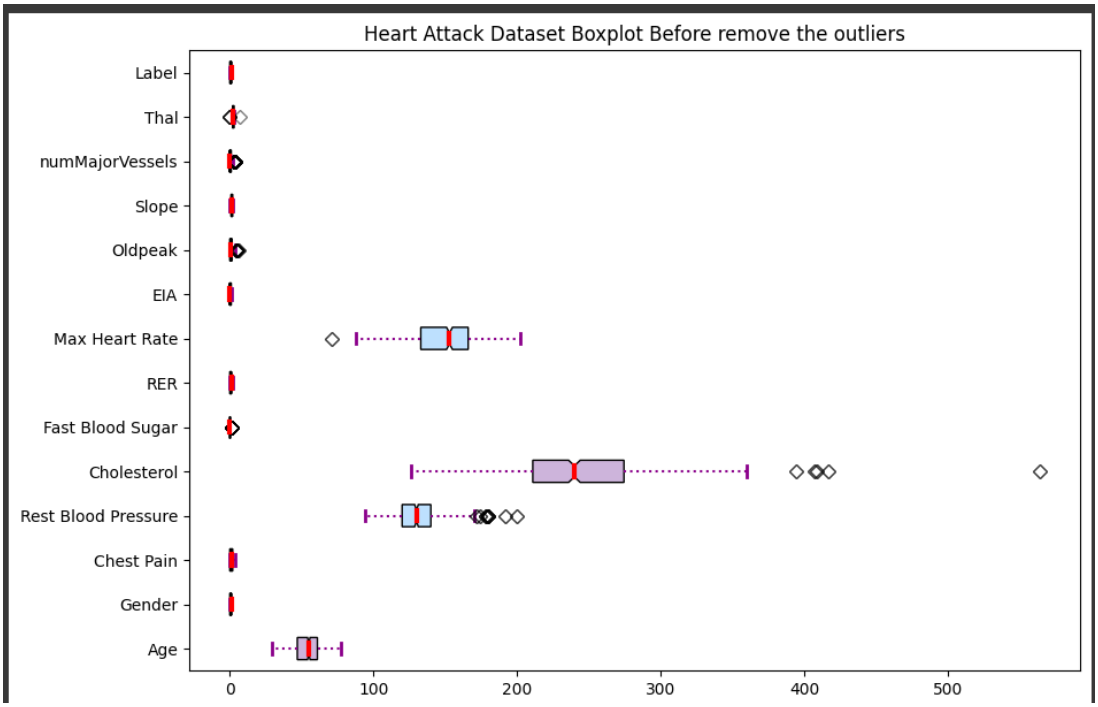


Figure 12: Box-Plot before removing outliers.

```
[184] # Replace outliers with the median
      # Calculate the z-score
      from scipy import stats
      z = np.abs(stats.zscore(dataset['Max Heart Rate']))
      threshold = 1
      dataset.loc[z > threshold, 'Max Heart Rate'] = dataset['Max Heart Rate'].median()

      z1 = np.abs(stats.zscore(dataset['Cholesterol']))
      dataset.loc[z > threshold, 'Cholesterol'] = dataset['Cholesterol'].median()

      z2 = np.abs(stats.zscore(dataset['Rest Blood Pressure']))
      dataset.loc[z > threshold, 'Rest Blood Pressure'] = dataset['Rest Blood Pressure'].median()
```

Figure 14: Z-Score calculation and replacing outlier with median value.

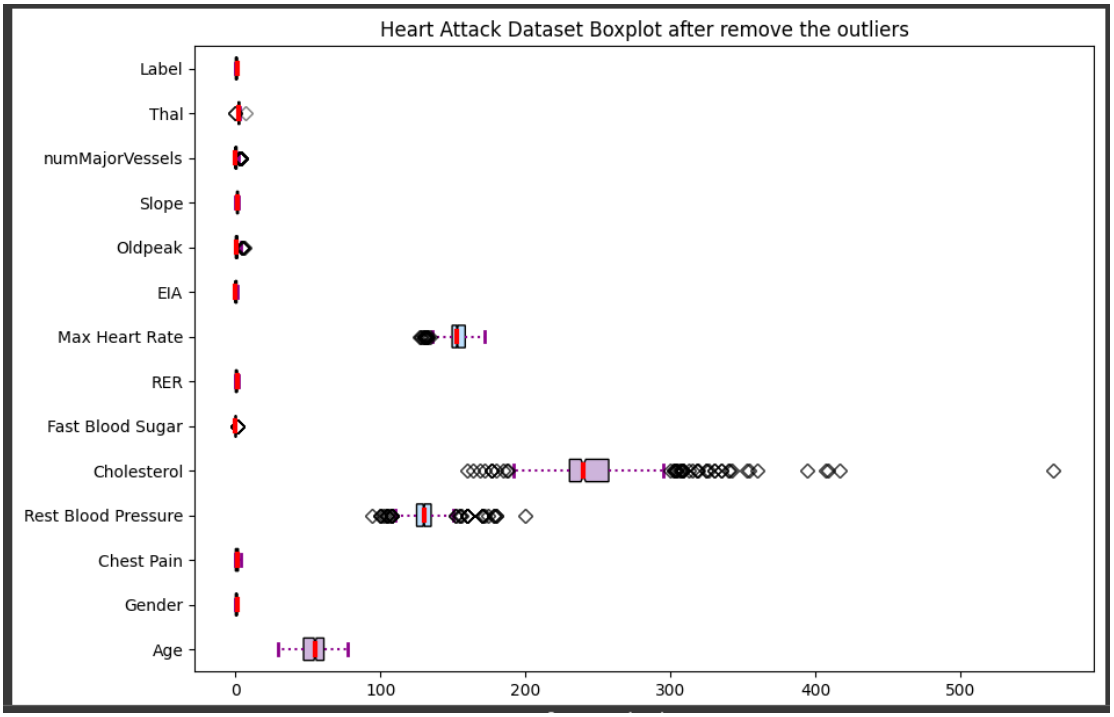
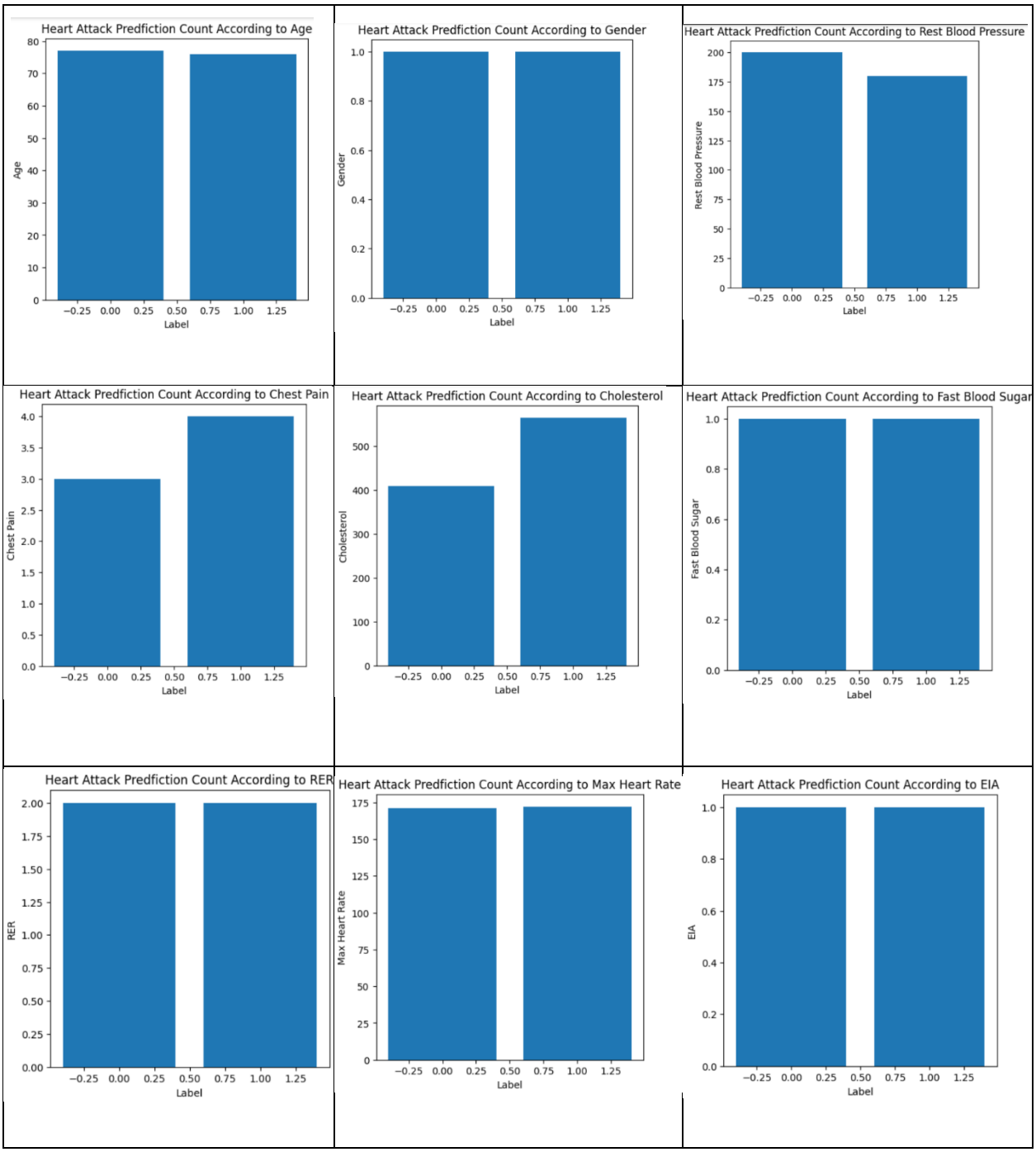


Figure 13: after trying to remove outlier values.

checking if the data is imbalanced or not is a crucially important step before getting started with data splitting, and there are two options, under-sampling, and oversampling. But before that, we plot the label column with each attribute to check how they are related and see which label index has a higher probability, according to that we can choose between under or over-sampling techniques.

The table below shows the relationship between the target column and each attribute:



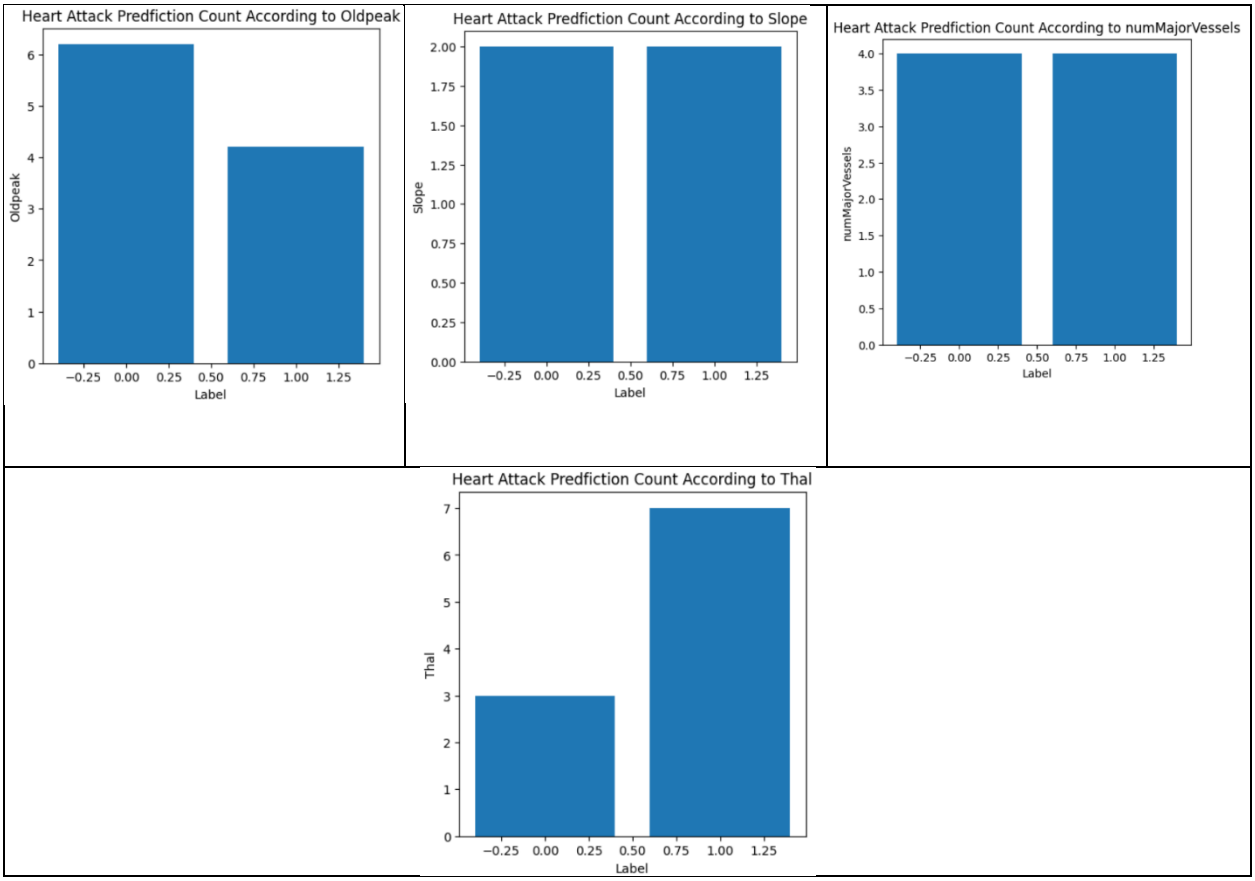


Table 1: Table shows the relationship between the target column and each attribute in dataset.

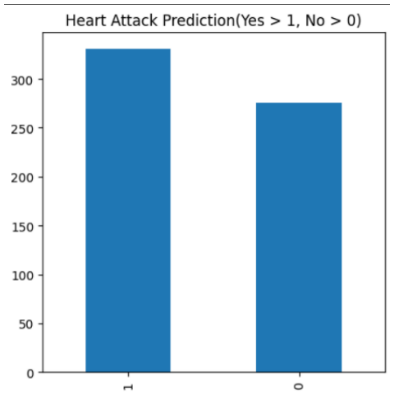
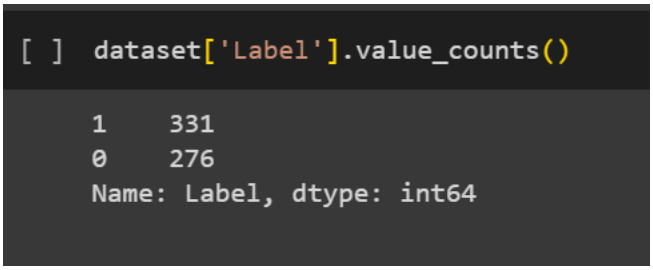


Figure 15: This figure shows the number of data samples their result is positive for the heart attack which is higher than negative result represented by 0.

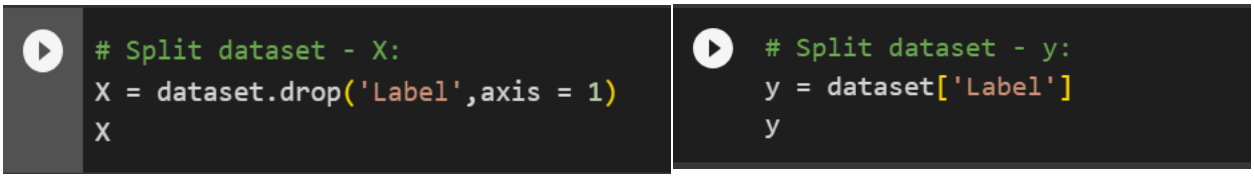


```
[ ]  dataset['Label'].value_counts()

     1    331
     0    276
     Name: Label, dtype: int64
```

Figure 13: The following results show imbalance, and this can be solved using under-sampling or over-sampling.

Figure 13 highlights the idea that under-sampling or over-sampling must be applied to avoid imbalance, and then balance the dataset records. Before applying this, we need to implement a dataset split into X, and y. X represents the selected attributes, or it could be the whole attribute except the target column. Also, y is the target column, where we need to solve the **imbalance** issue using the learning package from Python.

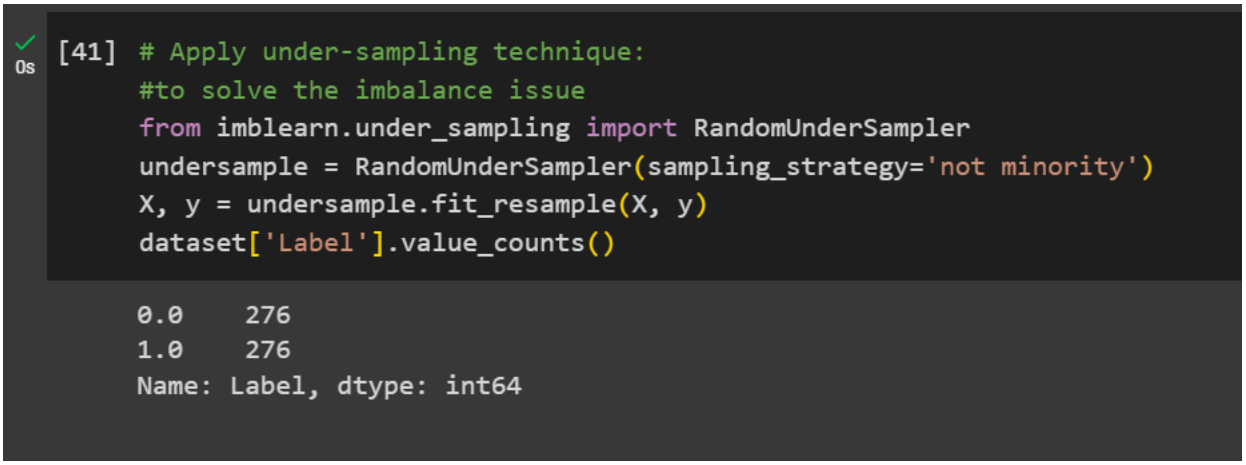


```
# Split dataset - X:
X = dataset.drop('Label',axis = 1)
X
```

```
# Split dataset - y:
y = dataset['Label']
y
```

Figure 14: Split X and y using the dataset after the pre-processing done on it.

Then, apply ***RandomUnderSampler*** from the package imblearn:



```
[41] # Apply under-sampling technique:
     #to solve the imbalance issue
     from imblearn.under_sampling import RandomUnderSampler
     undersample = RandomUnderSampler(sampling_strategy='not minority')
     X, y = undersample.fit_resample(X, y)
     dataset['Label'].value_counts()

     0.0    276
     1.0    276
     Name: Label, dtype: int64
```

Figure 15: After applying the under-sampling, y has been balanced.

Our dataset is ready to apply to Machine Learning algorithms. The first model we are going to apply is the Decision Tree Algorithm. Before starting to train our model, we need to specify X_train, X_test, y_train, and y_test.

```
#spliting data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.2, random_state = 0, stratify = y)
```

Figure16: X_train, X_test, y_train, and y_test parameters.

Apply the Decision Tree model on the dataset:

```
# Train the Decision Tree model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

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

 #Calculate the accuracy, precision, and recall of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
```

Figure 17: Apply Decision Tree model on dataset.

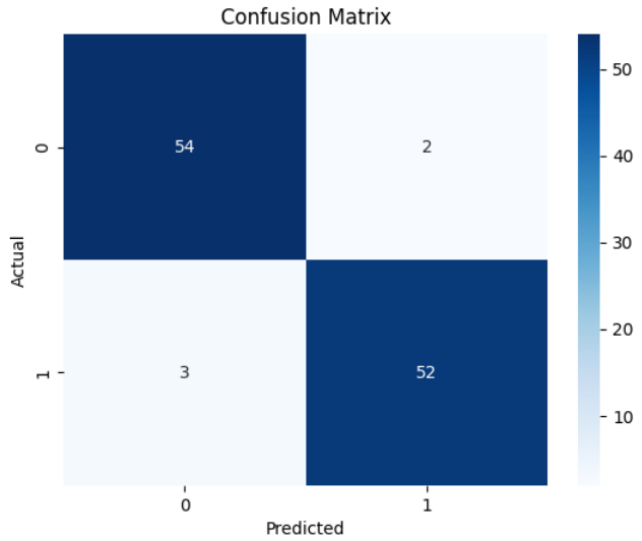


Figure 18: Confusion Matrix of Decision Tree Algorithm.

```
Decision Tree Algorithm Performance results:
Accuracy:  0.954954954954955
Precision:  0.9629629629629629
Recall:  0.9454545454545454
F1-Score:  0.9541284403669724
```

Figure 19: Performance measures of Decision Tree algorithm.

Applying the next model which is GaussianNB:

```
from sklearn.naive_bayes import GaussianNB

# Build a Gaussian Classifier
model2 = GaussianNB()

# Model training
model2.fit(X_train, y_train)

# Predict Output
y_pred2 = model.predict(X_test)

#Calculate the accuracy, precision, and recall of the classifier
accuracy2 = accuracy_score(y_test, y_pred2)
precision2 = precision_score(y_test, y_pred2)
recall2 = recall_score(y_test, y_pred2)
f12 = f1_score(y_test, y_pred2)
confusion_mat2 = confusion_matrix(y_test, y_pred2)
```

Figure 20: Applying GaussianNB.

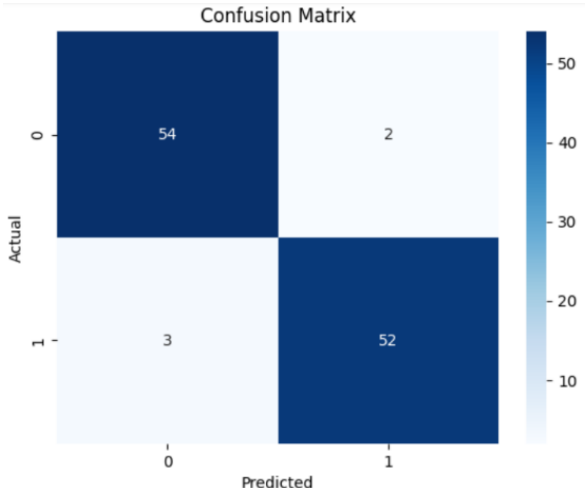


Figure 21: Confusion Matrix for GaussianNB.

```
GaussianNB Algorithm Performance results:
Accuracy:  0.954954954954955
Precision:  0.9629629629629629
Recall:  0.9454545454545454
F1-Score:  0.9541284403669724
```

Figure 22: GaussianNB performance.

Applying Over-sampling technique:

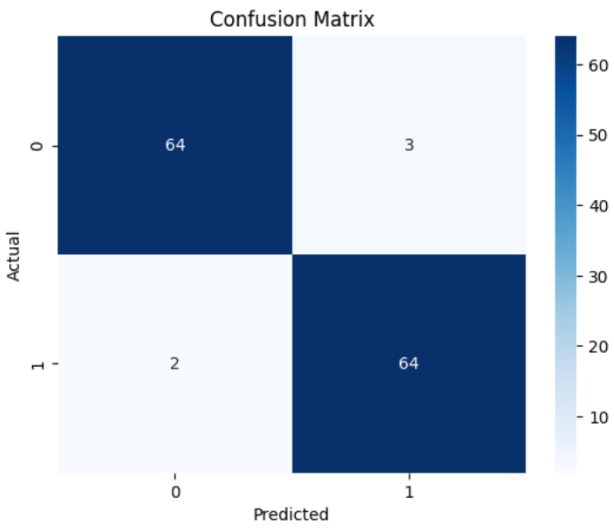


Figure 23: Confusion Matrix of Decision Tree after Over-sampling technique.

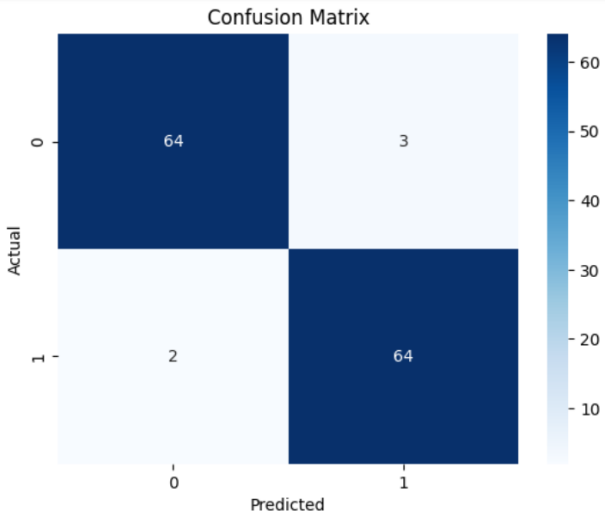


Figure 24: Confusion Matric of GaussianNB using Over-Sampling.

```
Decision Tree Algorithm Performance results:
Accuracy:  0.9624060150375939
Precision:  0.9552238805970149
Recall:  0.9696969696969697
F1-Score:  0.9624060150375939
```

```
GaussianNB Algorithm Performance results:
Accuracy:  0.9624060150375939
Precision:  0.9552238805970149
Recall:  0.9696969696969697
F1-Score:  0.9624060150375939
```

Figure 25: Their measure of performance results after over-sampling.

Now, apply K-Mean Clustering

```python
from sklearn.cluster import KMeans
# Kmeans model
kmeans = KMeans(n_clusters = 2,  random_state = 42)

# Fit and predict on the data
y_kmeans = kmeans.fit_predict(X)
```

Figure 26: Applying K-Mean Clustering Algorithm on dataset.

Notice in figure 26, we specified the n_clusters to be 2, since we have two output values: 0 for negative infection, and 1 for positive infection.

```
[107] # Save the predictions as a column
      dataset['y_kmeans']=y_kmeans

      # Check the distribution
      dataset['y_kmeans'].value_counts()
      #print(len(y_kmeans))
```

```
0    501
1    106
Name: y_kmeans, dtype: int64
```

Figure 27: Results after applying the model.

Plotting of K-Mean Clustering Algorithm:

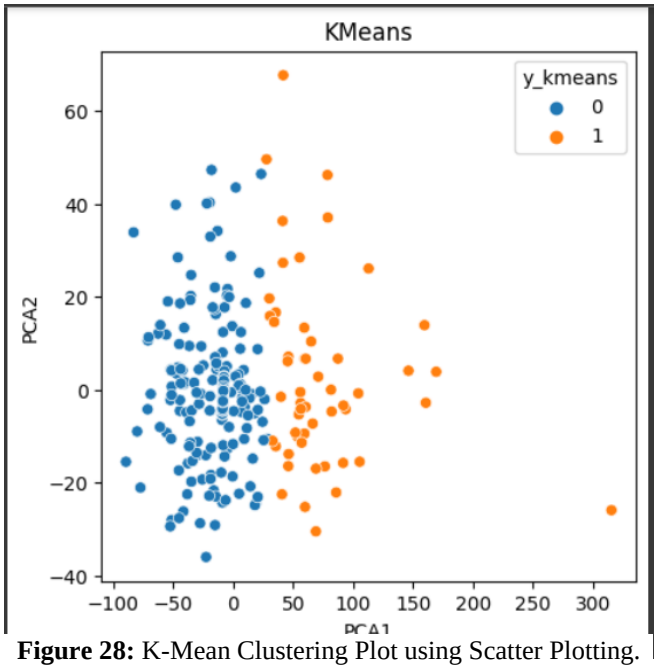


Figure 28: K-Mean Clustering Plot using Scatter Plotting.

Not that in the plot we have some names called PCA1 and PCA2. In this step, we will use two popular algorithms, PCA (Principal Component Analysis) and t-SNE (t-distributed stochastic neighbor embedding) to reduce the dimensionality of the dataset for visualization. There are four features in the dataset. We need to convert the features from a 4-dimensional space to a 2-dimensional space. The output from PCA and t-SNE are saved in the data frame as columns.

```
[145] # Dimensionality reduction
      from sklearn.decomposition import PCA
      from sklearn.manifold import TSNE
      # PCA with 2 components
      pca=PCA(n_components=2).fit_transform(X)

      # Create columns for the 2 PCA components
      dataset['PCA1'] = pca[:, 0]
      dataset['PCA2'] = pca[:, 1]

      # TSNE with 2 components
      tsne=TSNE(n_components=2).fit_transform(X)

      # Create columns for the 2 TSNE components
      dataset['TSNE1'] = tsne[:, 0]
      dataset['TSNE2'] = tsne[:, 1]
```

Figure 29: Dimensionality Reduction before plotting the data.

To measure the performance of the K-mean Cluster, Elbow Criterion used to evaluate the performance of this algorithm:

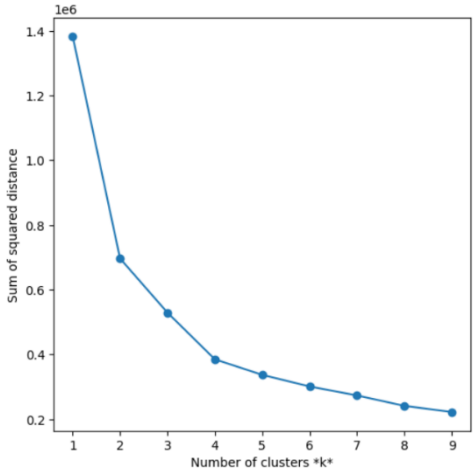


Figure 30: Evaluating the performance of K-mean Clustering using Elbow Criterion.

6. Experimental setup

The following section describes the important details of the experimental study. We used the interactive workplace Google Collab Notebook to execute the data pre-processing techniques and then implement the models. Google Collab requires zero configuration setup and has an easy way to access GPUs. Google Collab is a fully Integrated Development Environment that has important packages and libraries that assist in data science and machine learning models. Google Collab has the Python version 3.10.12.

In the beginning, we divided the dataset into training and testing set using the function *train_test_split* from the package *sklearn.model_selection*. 20% of dataset was split for testing, and the remaining 80% for training. The Algorithm has been used to work on the dataset are Decision Tree, GaussianNB, and K-Mean Clustering algorithms. For Decision Tree and GaussianNB were applied on the dataset and evaluated using the best performance measures. And for the K-Mean Clustering, is an unsupervised algorithm applied on data and the result was 2 cluster and evaluated using Elbow Criterion.

7. Performance Measure

The following measure has been used to evaluate the performance of the 3 algorithms used:

- Decision Tree Algorithm and GaussianNB [8]:
  - Accuracy:
    - $Accuracy = \frac{Correct\ Predications}{Total\ Predications}$
  - Precision:
    - $Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$
  - Recall:
    - $Precision = \frac{True\ Positive}{True\ Positive + False\ Negative}$
  - F1-Score[9]:
    - $F1 - Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall}$

- Confusion Matrix:

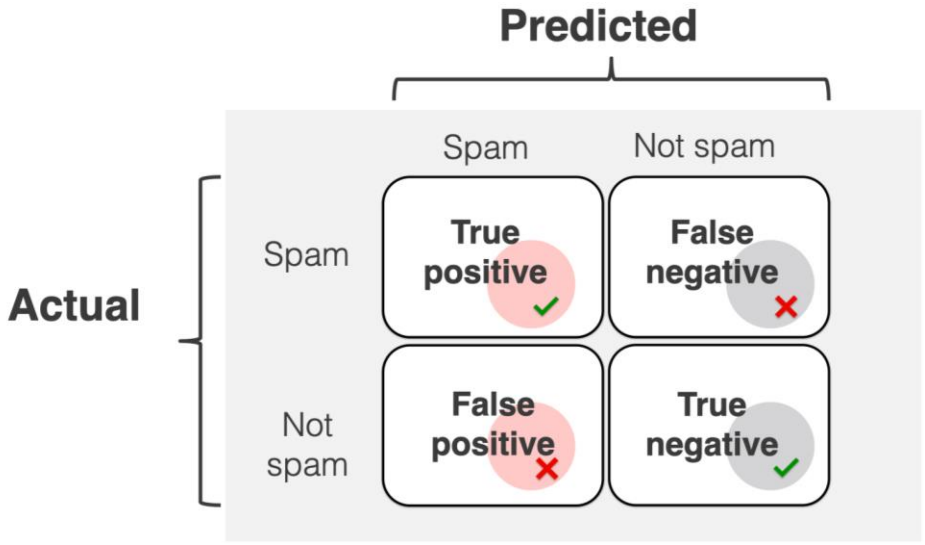


Figure 31: Confusion Matrix representation.

- Elbow Criterion: applying it as a code and then plotting the result.

```
sse = []
list_k = list(range(1, 10))

for k in list_k:
    km = KMeans(n_clusters=k)
    km.fit(X)
    sse.append(km.inertia_)

# Plot sse against k
plt.figure(figsize=(6, 6))
plt.plot(list_k, sse, '-o')
plt.xlabel(r'Number of clusters *k*')
plt.ylabel('Sum of squared distance');
```

Figure 32: Code of Elbow Criterion.

- Figure 30 has the representation of this code.

8. Results and Discussion

After applying the three models which are: Decision Tree, GaussianNB, and K-mean Clustering, the results of the three algorithms were compared and show which one is the best. In the above section, Decision Tree and GaussianNB with the under-sampling technique, outperformed the K-Mean Clustering with an accuracy equal to 0.95 to both Decision Tree and GaussianNB, both two algorithm produces the same accuracy. Also, the precision was equal to 0.96 for both two algorithms, recall equal to 0.94, and F1-Score equal to 0.95. in the case of over-sampling the results for Decision Tree and GaussianNB were the same as well, with an accuracy equal to 0.96. precision equals 0.95, 0.96 for the recall value, and f1-score with 0.96, and we can see much better accuracy with the over-sampling technique. For the K-mean algorithm, it shows the classification of the dataset into two main classes, 0 class for negative attack infection, and 1 class for Positive Heart Attack infection. 501 data samples were classified as 0 class, and the remaining 106 as class 1. The Elbow Criterion is used to test K-mean performance, and we can see the best number of clusters that can fit the data. However, we think working with an unsupervised learning algorithm requires a huge data set since the deep learning algorithms work best with big data, and get much more improved performance, so that's why we think the Decision Tree and GaussianNB outperformed the K-mean.

9. Conclusion

In conclusion, our project aims to address the critical issue of heart attack diseases by utilizing data mining techniques. Through the analysis and extraction of insights from selected datasets, we have gained a better understanding of the prevalence and risk factors associated with heart attacks. By identifying patterns, trends, and relationships within the data, we have been able to develop models for predicting the likelihood of a heart attack.

The findings from our study have significant implications for public health. By raising awareness about the risk factors and promoting preventive measures, we can minimize or eliminate the probability of heart attacks, ultimately improving heart health and well-being in the population.

It is important to note that our study relied on datasets obtained from Kaggle, and preprocessing tools were utilized to ensure data quality. However, further research and validation are necessary to strengthen the predictive models and ensure their applicability in real-world scenarios. Additionally, expanding the scope of the study to include larger and more diverse datasets would enhance the generalizability of the findings.

Overall, our project contributes to the growing field of data mining in healthcare and demonstrates its potential in providing valuable insights for addressing critical health issues. By harnessing the power of big data and employing data mining techniques, we can make significant strides in understanding, preventing, and managing heart attack diseases.

10. Acknowledgment

Dr. Irfan Ullah Abdur Rab, we would like to express our sincere gratitude for your invaluable guidance and supervision throughout the duration of our project. Your expertise and unwavering support have been instrumental in ensuring the success of this endeavor. Your commitment to excellence and commitment to our team have been truly inspiring. Thank you for sharing your knowledge, providing insightful feedback, and fostering a positive and productive working environment. Your dedication to the project has not only enriched our professional experience but has also significantly contributed to its success. It has been an honor and a privilege to have you as our supervising doctor, and we are immensely grateful for your mentorship. With heartfelt thanks.

11. Datasets

Name	Source	Link
Heart Attack Analysis & Prediction Dataset	Kaggle	Kaggle_link
Heart Attack	Kaggle	Kaggle_link
Heart Attack Prediction	Kaggle	Kaggle_link

Table 2: Datasets used in Machine Learning models.

12. References:

[1] P. B. Patil, P. M. Mallikarjun Shastry, and A. P. S Assistant Professor, "Heart Attack Detection Based On Mask Region Based Convolutional Neural Network Instance Segmentation and Hybrid Classification Using Machine Learning Techniques," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 9, pp. 2228-2244–2228– 2244, May 2021, Accessed: Nov. 30, 2023. [Online]. Available: https://turcomat.org/index.php/turkbilmat/article/view/3697

[2] M. Thangamani, R. Vijayalakshmi, M. Ganthimathi, M. Ranjitha, P. Malarkodi, and S. Nallusamy, "Efficient classification of heart disease using K-means clustering algorithm," *International Journal of Engineering Trends and Technology*, vol. 68, no. 12, pp. 48–53, Dec. 2020, doi: 10.14445/22315381/IJETT-V68I12P209.

[3] D. Shah, S. Patel, and S. K. Bharti, "Heart Disease Prediction using Machine Learning Techniques," *SN Comput Sci*, vol. 1, no. 6, Nov. 2020, doi: 10.1007/S42979-020-00365-Y.

[4] K. M. Almustafa, "Prediction of heart disease and classifiers' sensitivity analysis," *BMC Bioinformatics*, vol. 21, no. 1, Jul. 2020, doi: 10.1186/S12859-020-03626-Y.

[5] "What is a Decision Tree | IBM." Accessed: Nov. 30, 2023. [Online]. Available: https://www.ibm.com/topics/decision-trees

[6] "Gaussian Naive Bayes: What You Need to Know? | upGrad blog." Accessed: Nov. 30, 2023. [Online]. Available: https://www.upgrad.com/blog/gaussian-naive-bayes/

[7] "K-Means Clustering Algorithm - Javatpoint." Accessed: Nov. 30, 2023. [Online]. Available: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning

[8] "Accuracy vs. precision vs. recall in machine learning: what's the difference?" Accessed: Nov. 30, 2023. [Online]. Available: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

[9] "What is the F1-score?" Accessed: Nov. 30, 2023. [Online]. Available: https://www.educative.io/answers/what-is-the-f1-score