HEART FAILURE PREDICTION

A MINOR PROJECT REPORT
SUBMITTED FOR THE PARTIAL FULFILLMENT OF AWARD OF THE DEGREE OF
Bachelor of Technology in Computer Science and Engineering

By

SAHIL KUMAR SHARMA	19BTCSE69
SWETA CHOPDAR	19BTCSE73

Under the Guidance
of
DR. SIBARAM PANIGRAHI

Department of Computer Science Engineering and Application
SAMBALPUR UNIVERSITY INSTITUTE OF INFORMATION TECHNOLOGY
Jyoti Vihar, Burla, Odisha- 768019

Jan, 2023

CERTIFICATE

This is to certify that the work contained in the project entitled **“HEART FAILURE PREDICTION”**, submitted by **SAHIL KUMAR SHARMA (Regd. No.: 10091/19), SWETA CHOPDAR (Regd. No.: 10097/19),**for the award of **B.Tech degree** to the **SUIIT**, **Burla**, is a record of bonafide project works carried out by students under my direct supervision and guidance.

I considered that the project has reached the standards and fulfilling the requirements of the rules and regulations relating to the nature of the degree. The contents embodied in the project have not been submitted for the award of any other degree or diploma in this or any other university.

External Examiner

Dr. Sibarama Panigrahi	Ms. Sushma Rath	Mr. Pradyumna K Ratha
Asst.Professor & Guide	Faculty in-charge, B.Tech	Asst. Professor & HOD
Dept. of CSE&A	Dept. of CSE&A	Dept. of CSE &A

DECLARATION

I do hereby declare that the work embodied in this minor/major project report entitled "HEART FAILURE PREDICTION" is the outcome of genuine work carried out by me under the direct supervision of Dr. Sibaram Panigrahi, Asst. Professor, Department of Computer Science Engineering and Application is submitted by me to Sambalpur University Institute of Information Technology, Burla for the award of the degree of Bachelor of Technology. The work is original and has not been previously formed the basis for the award of any other degree or diploma.

Date:	SAHIL KUMAR SHARMA
Place: Jyoti Vihar, Burla	19BTCSE69
	SWETA CHOPDAR
	19BTCSE73

ACKNOWLEDGEMENTS

I feel honored to avail myself of this opportunity to express my deep sense of gratitude to my guide Dr. Sibaram Panigrahi, Department of Computer Science Engineering, Sambalpur University Institute of Information Technology, Burla, Odisha, India for his valuable inspiration, guidance, and warm encouragement throughout my research work. His profound knowledge and timely advice were very much helpful in my research work without which my thesis could never be able to see this day. Proud to be work under him and he has given me every freedom for working in this Project.

Sahil Kumar Sharma (10091/19)
Sweta Chopdar (10097/19)

ABSTRACT

In this modern era people are very busy and working hard in order to satisfying their materialistic needs and not able to spend time for themselves which leads to physical stress and mental disorder. There are also reports that heart suffer because of global pandemic corona virus. Inflammation of the heart muscle can be caused by corona virus. Thus, heart disease is very common now a day's particularly in urban areas because of excess mental stress due to corona virus. As a result, heart disease has become one of the most important factors for death of men and women in the so-called material world. It has emerged as the top killer that has affected both urban and rural population. CAD (coronary artery disease) is one of the most common types of heart disease. In the medical field predicting the heart disease has become a very complicated and challenging task, requires patient previous health records and in some cases, they even need Genetic information as well. So, in this contemporary life style there is an urgent need of a system which will predict accurately the possibility getting heart disease. Predicting a heart disease in early stage will save many people's Life. There were many heart disease prediction systems available at present, the Authors have been researched well and proposed different Classification and prediction algorithms but each one has its own limitations. The main objective of this paper is to overcome the limitations and to design a robust system which works efficiently and will able to predict the possibility of heart failure accurately.

KEYWORDS: Heart Disease, Machine Learning, Regression, Classification, Coronary Artery Disease, Logistic Regression, Decision Tree Classifier, Random Forest Classifier, XGB Classifier.

LIST OF FIGURES

CONTENTS

CHAPTER 1

INTRODUCTION

According to the World Health Organization, every year 12 million deaths occur worldwide due to heart disease. Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of data analysis. The load of cardiovascular disease is rapidly increasing all over the world from the past few years. Many researches have been conducted in attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk. Heart Disease is even highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn reduces the complications.

Machine learning proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the health care industry. This project aims to predict future heart disease by analyzing data of patients which classifies whether they have heart disease or not using machine-learning algorithm. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can say that this technique can be very well adapted to do the prediction of heart disease.

1.1 MOTIVATION FOR THE WORK

The main motivation of doing this research is to present a heart disease prediction model for the prediction of occurrence of heart disease. Further, this research work is aimed towards identifying the best classification algorithm for identifying the possibility of heart disease in a patient. This work is justified by performing a comparative study and analysis using three classification algorithms namely Naïve Bayes, Decision Tree, and Random Forest are used at different levels of evaluations. Although these are commonly used machine learning algorithms, the heart disease prediction is a vital task involving highest possible accuracy. Hence, the three algorithms are evaluated at numerous levels and types of evaluation strategies. This will provide researchers and medical practitioners to establish a better.

1.2 PROBLEM STATEMENT

The major challenge in heart disease is its detection. There are instruments available which can predict heart disease but either it is expensive or are not efficient to calculate chance of heart disease in human. Early detection of cardiac diseases can decrease the mortality rate and overall complications. However, it is not possible to monitor patients every day in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more sapience, time and expertise. Since we have a good amount of data in today’s world, we can use various machine learning algorithms to analyze the data for hidden patterns. The hidden patterns can be used for health diagnosis in medicinal data.

CHAPTER 2

METHODOLOGY

2.1. EXISTING SYSTEM

Heart disease is even being highlighted as a silent killer which leads to the death of a person without obvious symptoms. The nature of the disease is the cause of growing anxiety about the disease & its consequences. Hence continued efforts are being done to predict the possibility of this deadly disease in prior. So that various tools & techniques are regularly being experimented with to suit the present-day health needs. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can conclude. This technique can be very well adapted to the do the prediction of heart disease. As the well-known quote says “Prevention is better than cure”, early prediction & its control can be helpful to prevent & decrease the death rates due to heart disease.

2.2. PROPOSED SYSTEM

The working of the system starts with the collection of data and selecting the important attributes. Then the required data is preprocessed into the required format. The data is then divided into two parts training and testing data. The algorithms are applied and the model is trained using the training data. The accuracy of the system is obtained by testing the system using the testing data. This system is implemented using the following modules.

1.) Collection of Dataset
2.) Selection of attributes
3.) Data Pre-Processing
4.) Balancing of Data
5.) Disease Prediction

2.2.1. Collection of datasets

Initially, we collect a dataset for our heart disease prediction system. After the collection of the dataset, we split the dataset into training data and testing data. The training dataset is used for prediction model learning and testing data is used for evaluating the prediction model. For this project, 70% of training data is used and 30% of data is used for testing. The dataset used for this project is Heart Disease UCI. The dataset consists of 76 attributes; out of which, 14 attributes are used for the system.

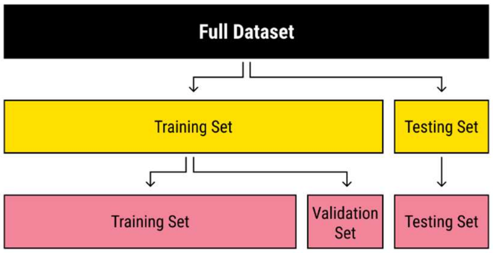


Figure: Collection of Data

2.2.2. Selection of attributes

Attribute or Feature selection includes the selection of appropriate attributes for the prediction system. This is used to increase the efficiency of the system. Various attributes of the patient like gender, chest pain type, fasting blood pressure, serum cholesterol, exam, etc. are selected for the prediction. The Correlation matrix is used for attribute selection for this model.

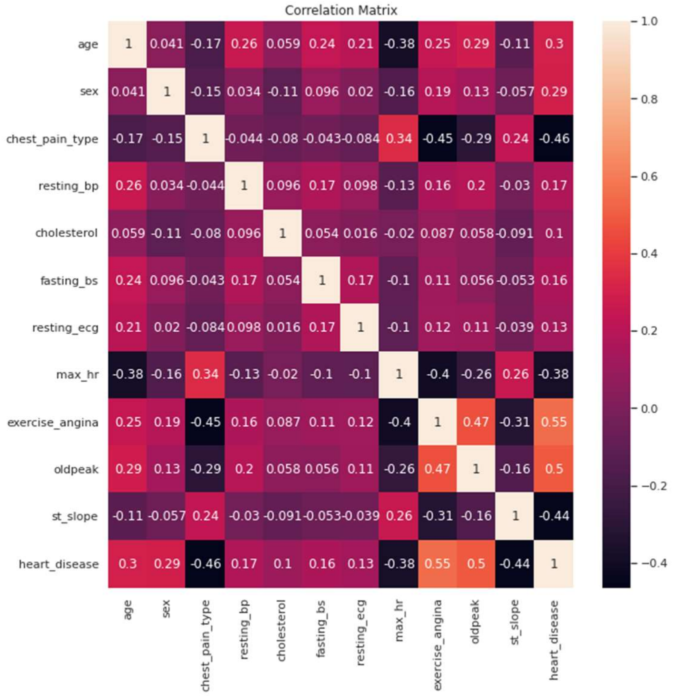


Figure: Correlation matrix

2.2.3. Pre-processing of Data

Data pre-processing is an important step for the creation of a machine learning model. Initially, data may not be clean or in the required format for the model which can cause misleading outcomes. In pre-processing of data, we transform data into our required format. It is used to deal with noises, duplicates, and missing values of the dataset. Data pre-processing has the activities like importing datasets, splitting datasets, attribute scaling, etc. Preprocessing of data is required for improving the accuracy of the model.

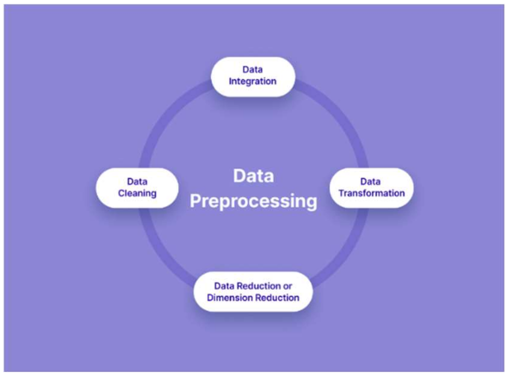


Figure: Data Pre-Processing

2.2.4. Balancing of Data

Imbalanced datasets can be balanced in two ways. They are Under Sampling and Over Sampling

(a) Under Sampling: In Under Sampling, dataset balance is done by the reduction of the size of the ample class. This process is considered when the amount of data is adequate.

(b) Over Sampling: In Over Sampling, dataset balance is done by increasing the size of the scarce samples. This process is considered when the amount of data is inadequate.

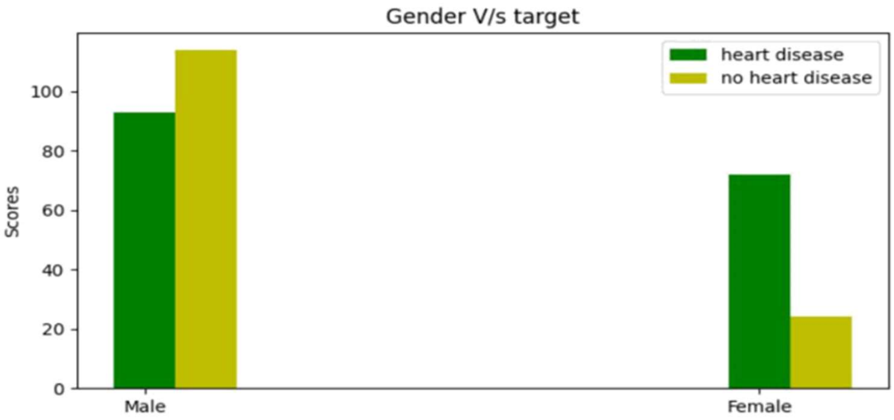


Figure: Data Balancing

2.2.5. Prediction of Disease

Various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Tree, Logistic Regression, Ada-boost, XG-boost are used for classification. Comparative analysis is performed among algorithms and the algorithm that gives the highest accuracy is used for heart disease prediction.

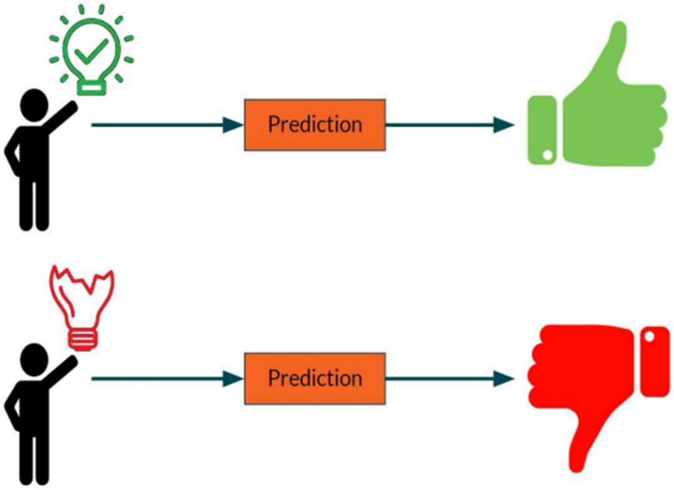


Figure: Prediction of Disease

CHAPTER 3

WORKING OF SYSTEM

3.1. SYSTEM ARCHITECTURE

The system architecture gives an overview of the working of the system.

The working of this system is described as follows:

Dataset collection is collecting data which contains patient details. Attributes selection process selects the useful attributes for the prediction of heart disease. After identifying the available data resources, they are further selected, cleaned, made into the desired form. Different classification techniques as stated will be applied on preprocessed data to predict the accuracy of heart disease. Accuracy measure compares the accuracy of different classifiers.

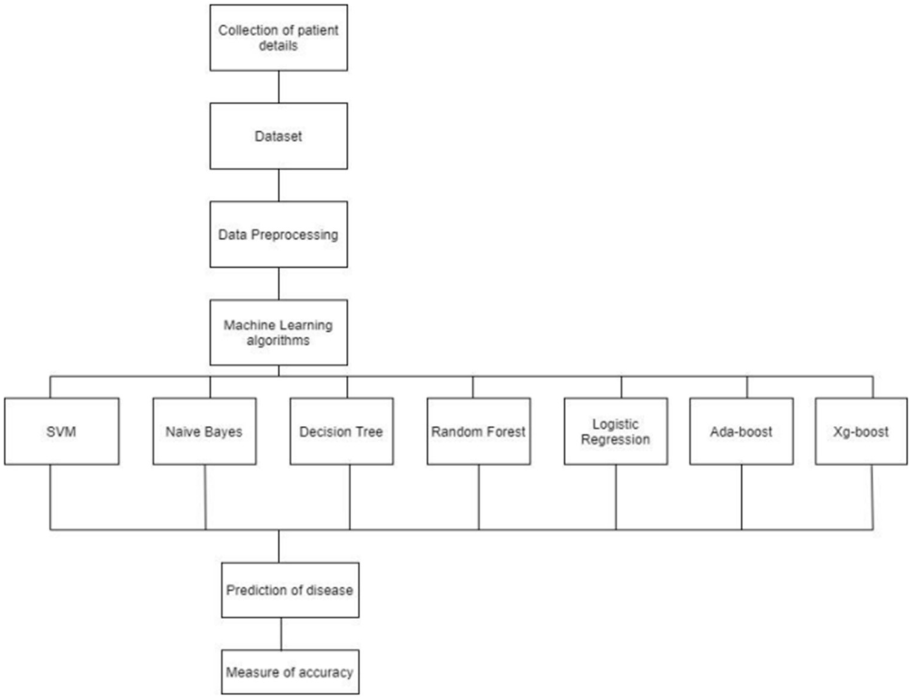


Figure: SYSTEM ARCHITECTURE

3.2. MACHINE LEARNING

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data.

- **Supervised Learning**

  Supervised learning is the type of machine learning in which machines are trained using well "labelled" training data, and on the basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

  In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher. Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

- **Unsupervised Learning**

  Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.
  • Unsupervised learning is helpful for finding useful insights from the data.
  • Unsupervised learning is much similar to how a human learns to think by their own experiences, which makes it closer to the real AI.
  • Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
  • In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

- **Reinforcement Learning**

  Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

3.3. ALGORITHMS

3.3.1. Logistic Regression

Logistic Regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is used to predict a binary outcome **(1 / 0, Yes / No, True / False)** given a set of independent variables. Logistic Regression is an extension of Linear Regression when the outcome variable is binary. The model is represented by an equation of the form $Y = e^{(b0 + b1X)} / (1 + e^{(b0 + b1X)})$ where Y is the outcome, X is the independent variable, b0 and b1 are the parameters of the model and e is the exponential function.

Logistic Regression is a type of supervised learning algorithm that is used for classification problems. It is a statistical method that is used to fit a regression model when the outcome variable is binary (or dichotomous). The outcome can be only two possible values such as 1 or 0, Yes or No, True or False.

Advantages:

1. It is a widely used method, and is a simple and fast algorithm that is easy to implement.
2. It can handle both linear and non-linear relationships between the independent and dependent variables.
3. It can handle categorical independent variables.
4. The model can be updated easily with new data using stochastic gradient descent.
5. It is less prone to overfitting than other more complex models.

Disadvantages:

1. It can only be used for binary classification problems.
2. It assumes linearity of independent variables and log odds, this may not hold in all cases.
3. It requires large sample sizes to achieve stable results.
4. It can be sensitive to outliers, particularly in the case of small datasets.
5. It can be affected by irrelevant features, so feature selection is important.

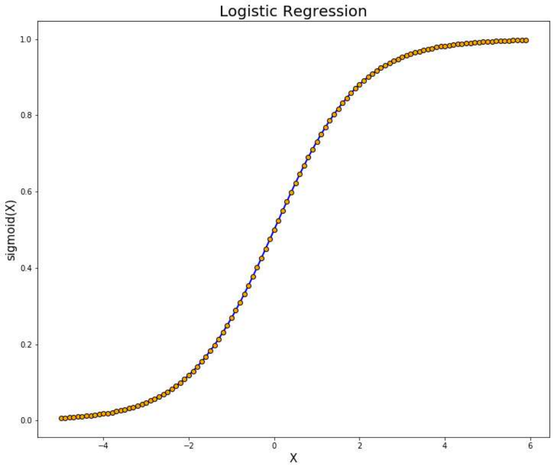
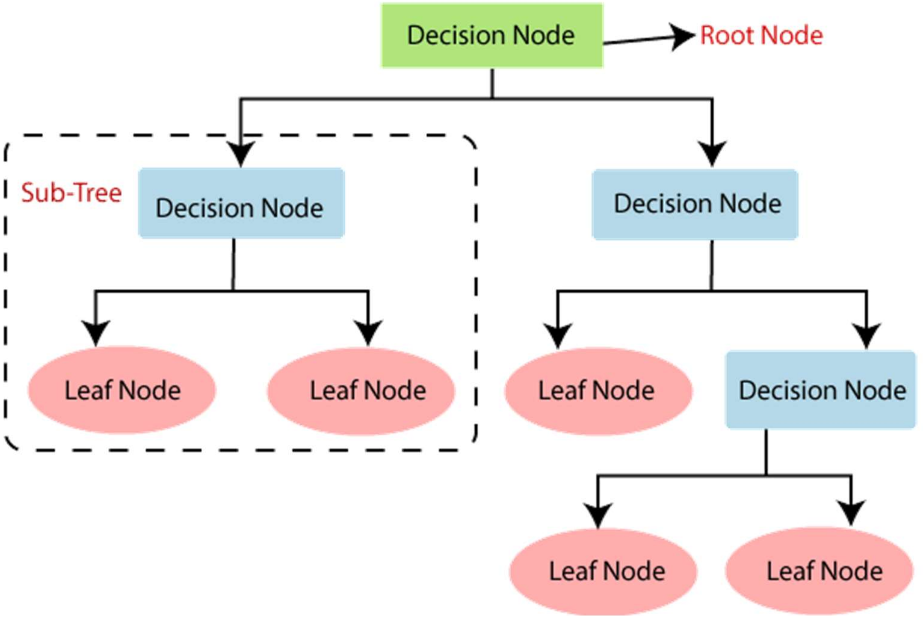


Figure: Logistic Regression

3.3.2. Decision Tree Classifier

A decision tree classifier is a type of supervised learning algorithm that is used for classification problems. It is a tree-based model that is used to make predictions by recursively partitioning the data into subsets based on the values of the input features. At each internal node of the tree, a decision is made based on a single feature, and the tree branches based on the possible outcomes of that decision. The final outcome is determined by traversing the tree from the root to a leaf node.

Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a Decision Tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A Decision Tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.



There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision Tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure. In Decision Tree the major challenge is to identify the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= 1- $\sum_j P_j^2$

3.3.3. Random Forest Classifier

Random Forest is a supervised learning algorithm. It is an extension of machine learning classifiers which include the bagging to improve the performance of Decision Tree. It combines tree predictors, and trees are dependent on a random vector which is independently sampled. The distribution of all trees are the same. Random Forests splits nodes using the best among of a predictor subset that are randomly chosen from the node itself, instead of splitting nodes based on the variables. The time complexity of the worst case of learning with Random Forests is O(M(dnlogn)) , where M is the number of growing trees, n is the number of instances, and d is the data dimension.

It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest consists of trees. It is said that the more trees it has, the more robust a forest is. Random Forests create Decision Trees on randomly selected data samples, get predictions from each tree and select the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random Forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Assumptions:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.

- The predictions from each tree must have very low correlations.

Algorithm Steps:

It works in four steps:

- Select random samples from a given dataset.
- Construct a Decision Tree for each sample and get a prediction result from each Decision Tree.
- Perform a vote for each predicted result.
- Select the prediction result with the most votes as the final prediction.

Advantages:

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages:

Although Random Forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

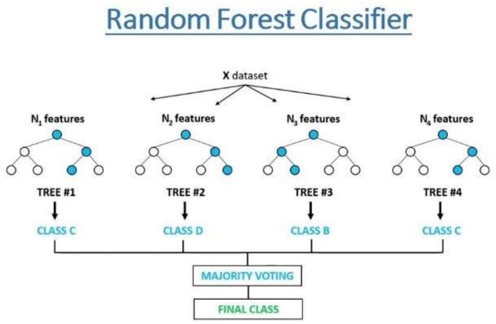


Figure: Random Forest Classifier

3.3.4. XGB Classifier

XG-boost is an implementation of Gradient Boosted decision trees. It is a type of Software library that was designed basically to improve speed and model performance. In this algorithm, decision trees are created in sequential form. Weights play an important role in XG-boost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then assemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined predict.

Regularization: XG-boost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XG-boost is also called regularized form of GBM (Gradient Boosting Machine). While using Scikit Learn library, we pass two hyper-parameters (alpha and lambda) to XG-boost related to regularization. alpha is used for L1 regularization and lambda is used for L2 regularization.

2. Parallel Processing: XG-boost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model. While using Scikit Learn library, thread hyper-parameter is used for parallel processing. thread represents number of CPU cores to be used. If you want to use all the available cores, don't mention any value for thread and the algorithm will detect automatically.

3. Handling Missing Values: XG-boost has an in-built capability to handle missing values. When XG-boost encounters a missing value at a node, it tries both the left- and right-hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

4. Cross Validation: XG-boost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited values can be tested.

5. Effective Tree Pruning: A GBM would stop splitting a node when it encounters a negative loss in the split. Thus, it is more of a greedy algorithm. XG-boost on the other hand make splits up to the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

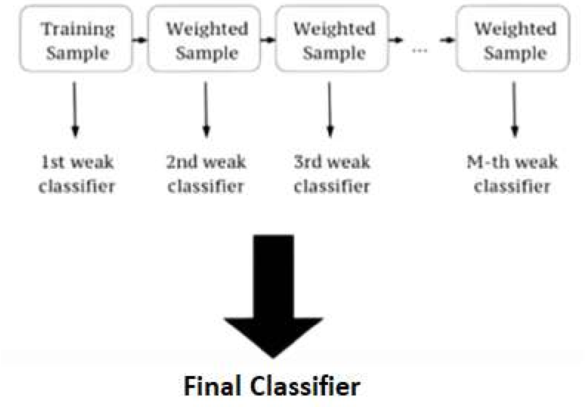


Figure: XG-Boost

3.3.5. Support Vector Machine (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In the 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

The followings are important concepts in SVM –

Support Vectors - Data Points that are closest to the hyperplane are called support vectors. Separating line will be defined with the help of these data points.

Hyperplane - As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

Margin - It may be defined as the gap between two lines on the closest data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

Types of SVM:

SVM can be of two types:

● **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

● **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier. The objective of the support vector machine algorithm is to find a hyperplane in an Ndimensional space (N - the number of features) that distinctly classifies the data points.

The advantages of support vector machines are:

● Effective in high dimensional spaces.
● Still effective in cases where the number of dimensions is greater than the number of samples.
● Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
● Versatile: different kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

● If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

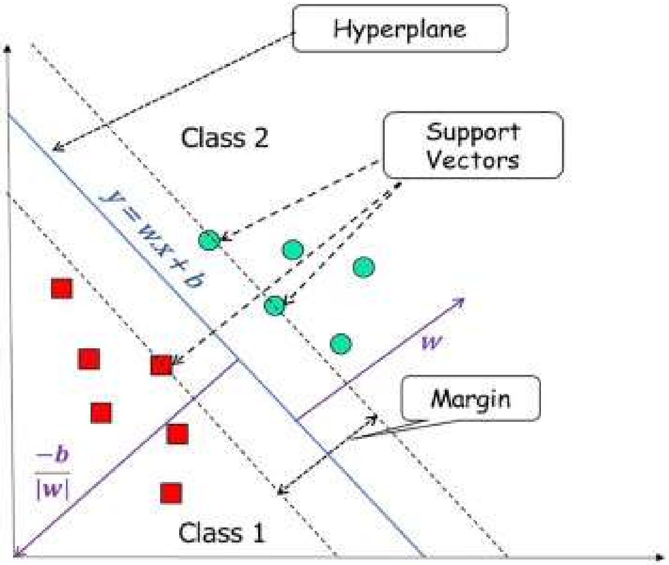


Figure: Support Vector Machine

3.3.6. Naïve Bayes Algorithm

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset.

Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

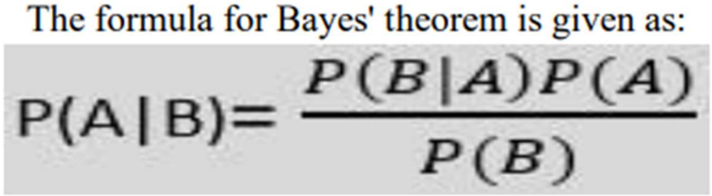
The Naive Bayes algorithm is comprised of two words Naive and Bayes, Which can be described as:

- Naive: It is called Naive because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the basis of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes's Theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:



The formula for Bayes' theorem is given as:

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

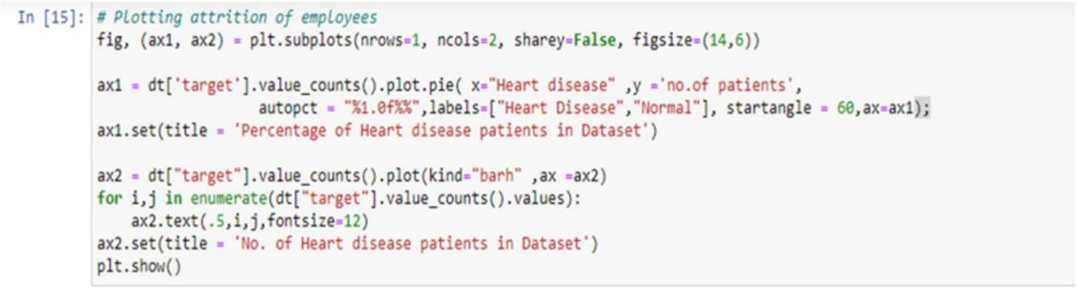
Where,
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.
P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
P(B) is Marginal Probability: Probability of Evidence.

CHAPTER 4

EXPLORATORY DATA ANALYSIS

4.1. Distribution of Heart Disease (Target Value):

Here we will check whether the dataset is balanced or not.



```
In [15]: # Plotting attrition of employees
         fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, sharey=False, figsize=(14,6))

         ax1 = dt['target'].value_counts().plot.pie( x="Heart disease" ,y ='no.of patients',
                            autopct = "%1.0f%%",labels=["Heart Disease","Normal"], startangle = 60,ax=ax1);
         ax1.set(title = 'Percentage of Heart disease patients in Dataset')

         ax2 = dt["target"].value_counts().plot(kind="barh" ,ax =ax2)
         for i,j in enumerate(dt["target"].value_counts().values):
             ax2.text(.5,i,j,fontsize=12)
         ax2.set(title = 'No. of Heart disease patients in Dataset')
         plt.show()
```

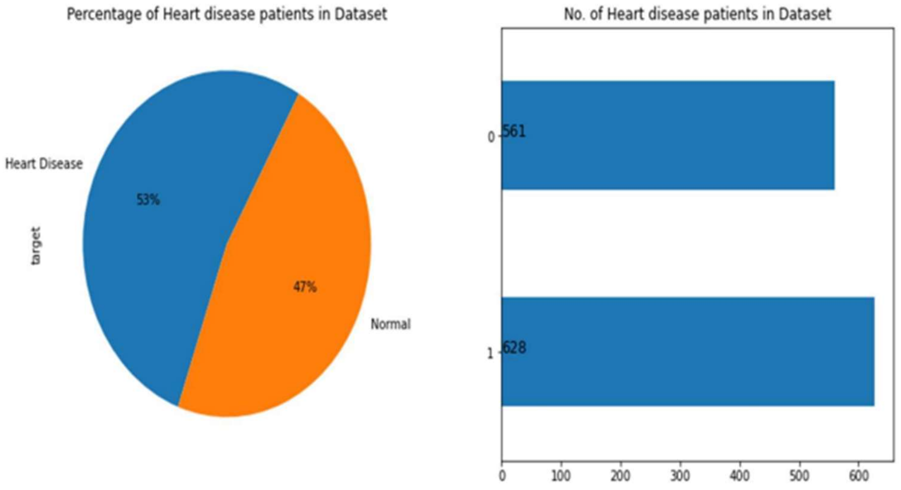
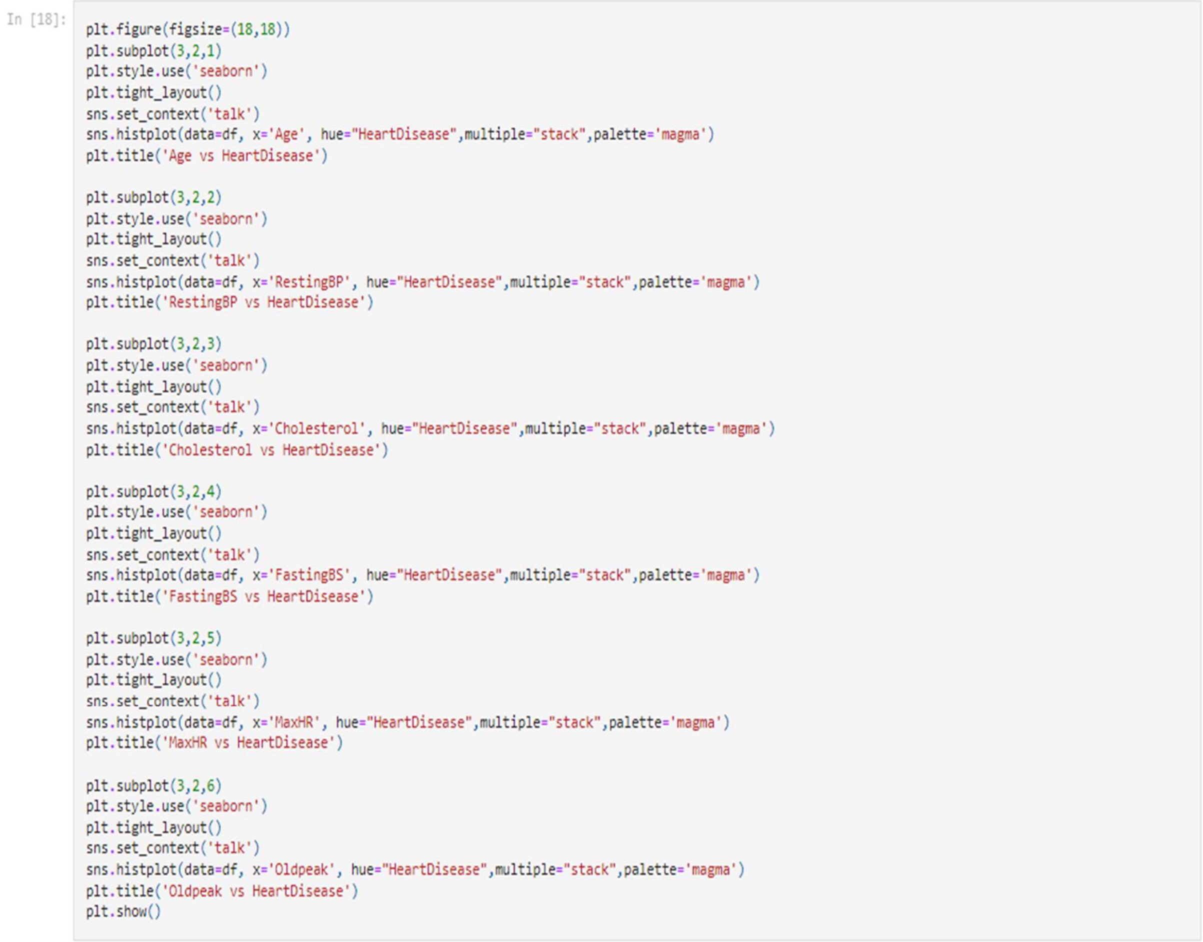


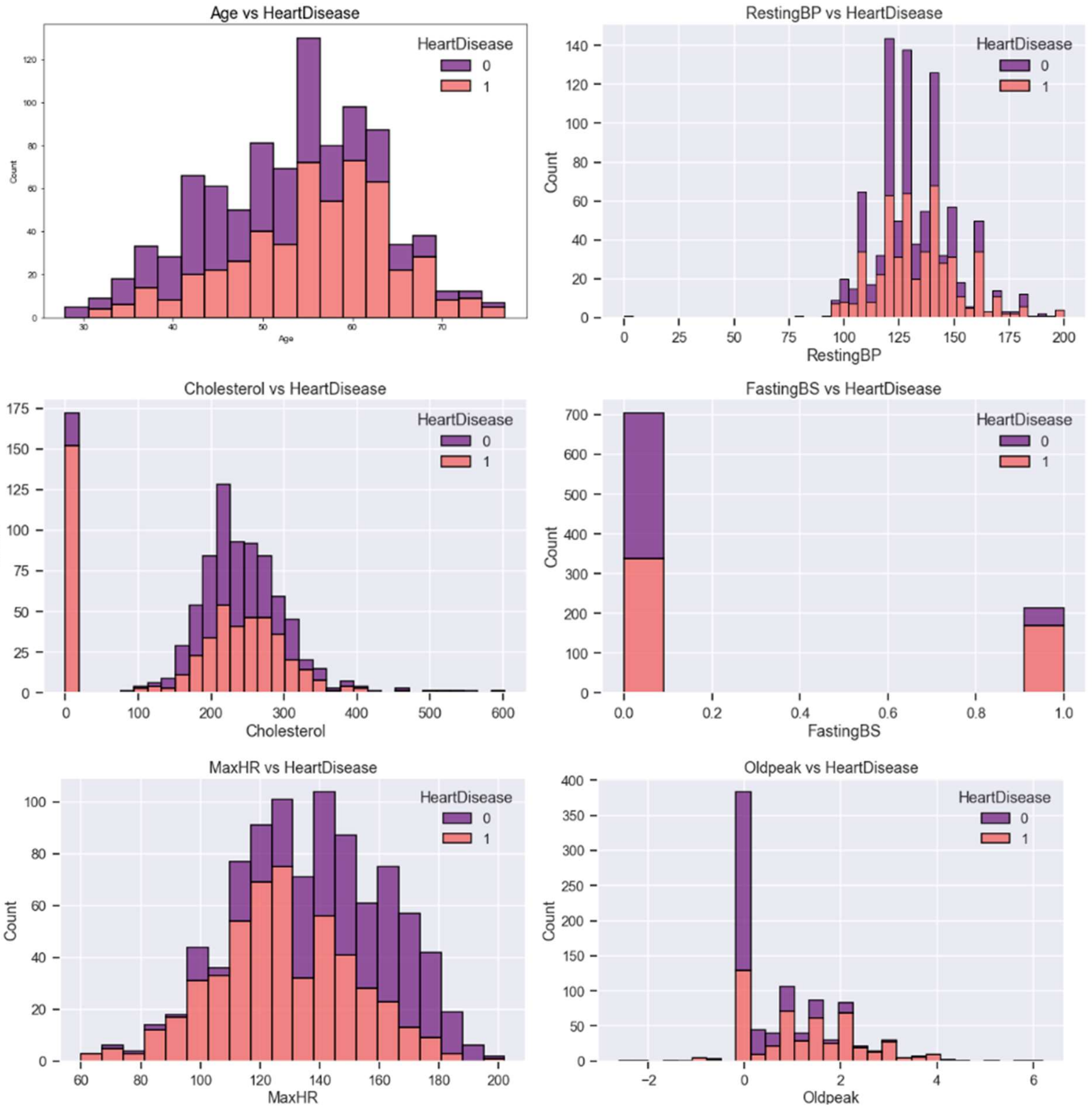
Figure 4.1. Distribution of target value

The dataset is balanced having 628 heart disease patients and 561 normal patients.

4.2. Data Visualization

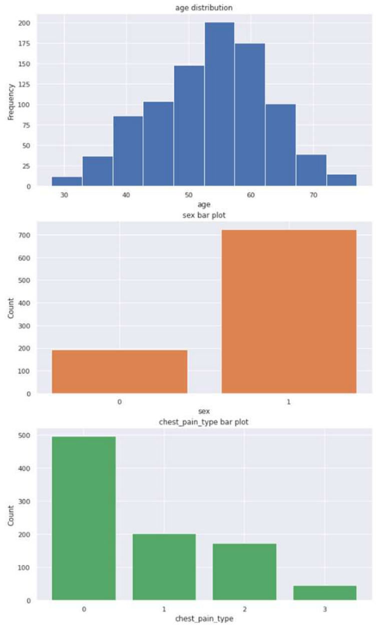


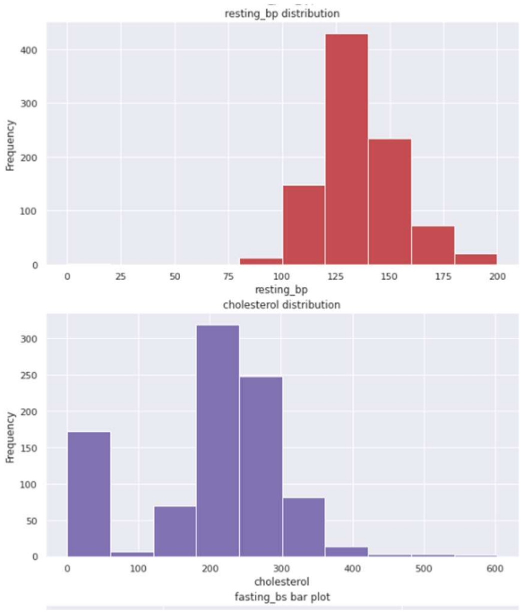
```
plt.figure(figsize=(18,18))
plt.subplot(3,2,1)
plt.style.use('seaborn')
plt.tight_layout()
sns.set_context('talk')
sns.histplot(data=df, x='Age', hue="HeartDisease",multiple="stack",palette='magma')
plt.title('Age vs HeartDisease')

plt.subplot(3,2,2)
plt.style.use('seaborn')
plt.tight_layout()
sns.set_context('talk')
sns.histplot(data=df, x='RestingBP', hue="HeartDisease",multiple="stack",palette='magma')
plt.title('RestingBP vs HeartDisease')

plt.subplot(3,2,3)
plt.style.use('seaborn')
plt.tight_layout()
sns.set_context('talk')
sns.histplot(data=df, x='Cholesterol', hue="HeartDisease",multiple="stack",palette='magma')
plt.title('Cholesterol vs HeartDisease')

plt.subplot(3,2,4)
plt.style.use('seaborn')
plt.tight_layout()
sns.set_context('talk')
sns.histplot(data=df, x='FastingBS', hue="HeartDisease",multiple="stack",palette='magma')
plt.title('FastingBS vs HeartDisease')

plt.subplot(3,2,5)
plt.style.use('seaborn')
plt.tight_layout()
sns.set_context('talk')
sns.histplot(data=df, x='MaxHR', hue="HeartDisease",multiple="stack",palette='magma')
plt.title('MaxHR vs HeartDisease')

plt.subplot(3,2,6)
plt.style.use('seaborn')
plt.tight_layout()
sns.set_context('talk')
sns.histplot(data=df, x='Oldpeak', hue="HeartDisease",multiple="stack",palette='magma')
plt.title('Oldpeak vs HeartDisease')
plt.show()
```

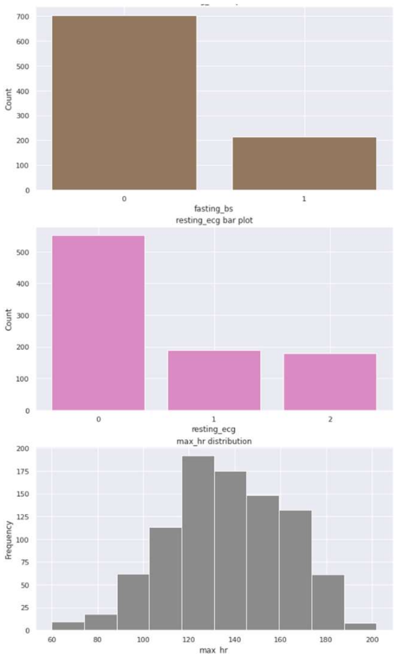


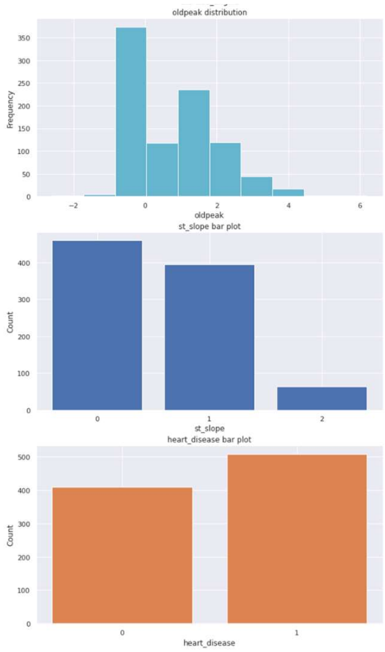
4.3. All Data Distribution:

This is the total distribution of the data.


age distribution
Frequency
0
25
50
75
100
125
150
175
200
30
40
50
60
70
age
sex bar plot
Count
0
100
200
300
400
500
600
700
0
1
sex
chest_pain_type bar plot
Count
0
100
200
300
400
500
0
1
2
3
chest_pain_type


resting_bp distribution
Frequency
0
100
200
300
400
0
25
50
75
100
125
150
175
200
resting_bp
cholesterol distribution
Frequency
0
50
100
150
200
250
300
0
100
200
300
400
500
600
cholesterol
fasting_bs bar plot


Count
0
100
200
300
400
500
600
700
0
1
fasting_bs
resting_ecg bar plot
Count
0
100
200
300
400
500
0
1
2
resting_ecg
max_hr distribution
Frequency
0
25
50
75
100
125
150
175
200
60
80
100
120
140
160
180
200
max_hr


oldpeak distribution
Frequency
0
50
100
150
200
250
300
350
−2
0
2
4
6
oldpeak
st_slope bar plot
Count
0
100
200
300
400
0
1
2
st_slope
heart_disease bar plot
Count
0
100
200
300
400
500
0
1
heart_disease

CHAPTER 5

OUTLIER DETECTION AND REMOVAL

Z-scores can quantify the unusualness of an observation when your data follow the normal distribution. Z-scores are the number of standard deviations above and below the mean that each value falls. For example, a Z-score of 2 indicates that an observation is two standard deviations above the average while a Z-score of -2 signifies it is two standard deviations below the mean. A Z-score of zero represents a value that equals the mean.

To calculate the Z-score for an observation, take the raw measurement, subtract the mean, and divide by the standard deviation. Mathematically, the formula for that process is the following:

$$Z = \frac{X - \mu}{\sigma}$$

The further away an observation's Z-score is from zero, the more unusual it is. A standard cut-off value for finding outliers are Z-scores of +/-3 or further from zero. The probability distribution below displays the distribution of Z-scores in a standard normal distribution. Z-scores beyond +/- 3 are so extreme you can barely see the shading under the curve.

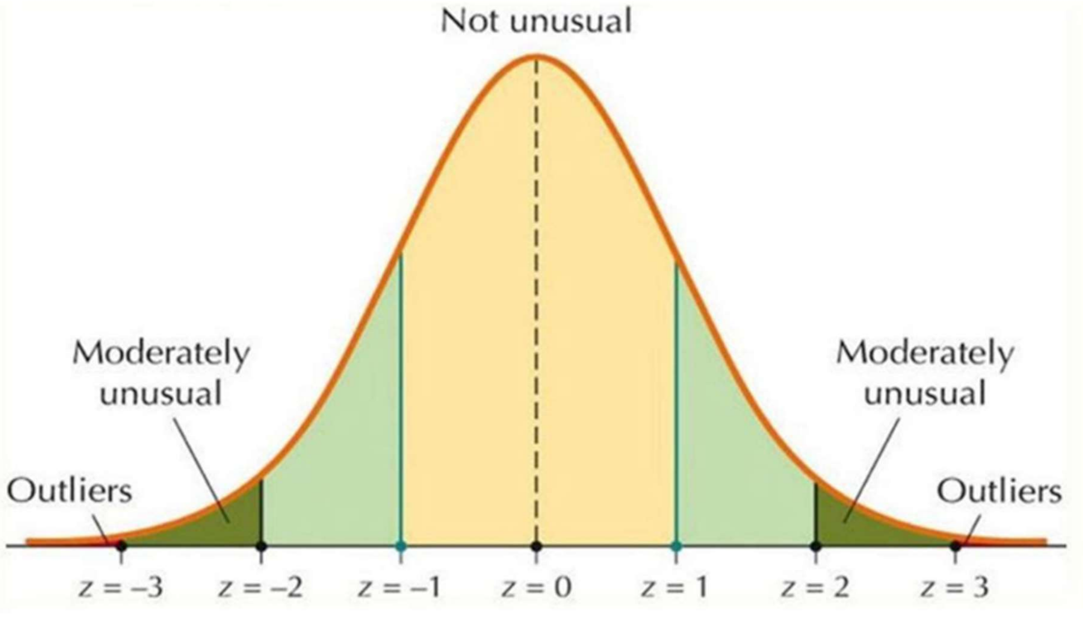
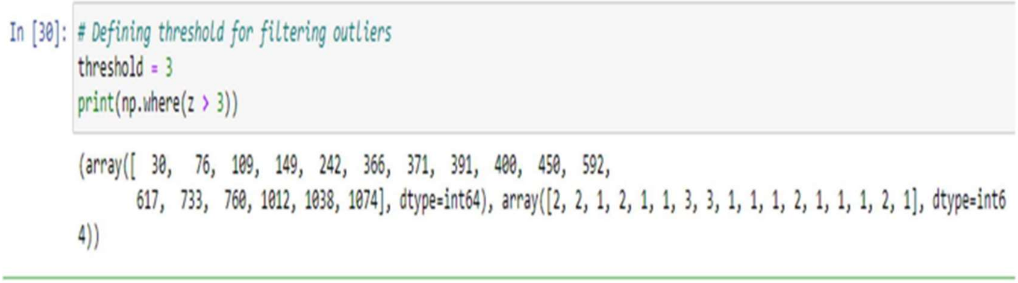


Figure 5.1.: Z-score in normal distribution

Now for removing the outliers we will define a threshold. So, by defining z = 3, wherever the z score of the data ponts will be greater than 3 they will be filtered out.



```
In [30]: # Defining threshold for filtering outliers
         threshold = 3
         print(np.where(z > 3))

         (array([  30,   76,  109,  149,  242,  366,  371,  391,  400,  450,  592,
                 617,  733,  760, 1012, 1038, 1074], dtype=int64), array([2, 2, 1, 2, 1, 1, 3, 3, 1, 1, 1, 2, 1, 1, 1, 2, 1], dtype=int6
         4))
```

Figure 5.2.: Removing the outliers

The first array contains the list of row numbers and second array respective column numbers, which mean z [30][2] have a Z-score higher than 3. There are total 17 data points which are outliers. Now there are total 1172 records and 11 features with 1 target variable.

CHAPTER 6

TRAIN AND TEST SPLIT

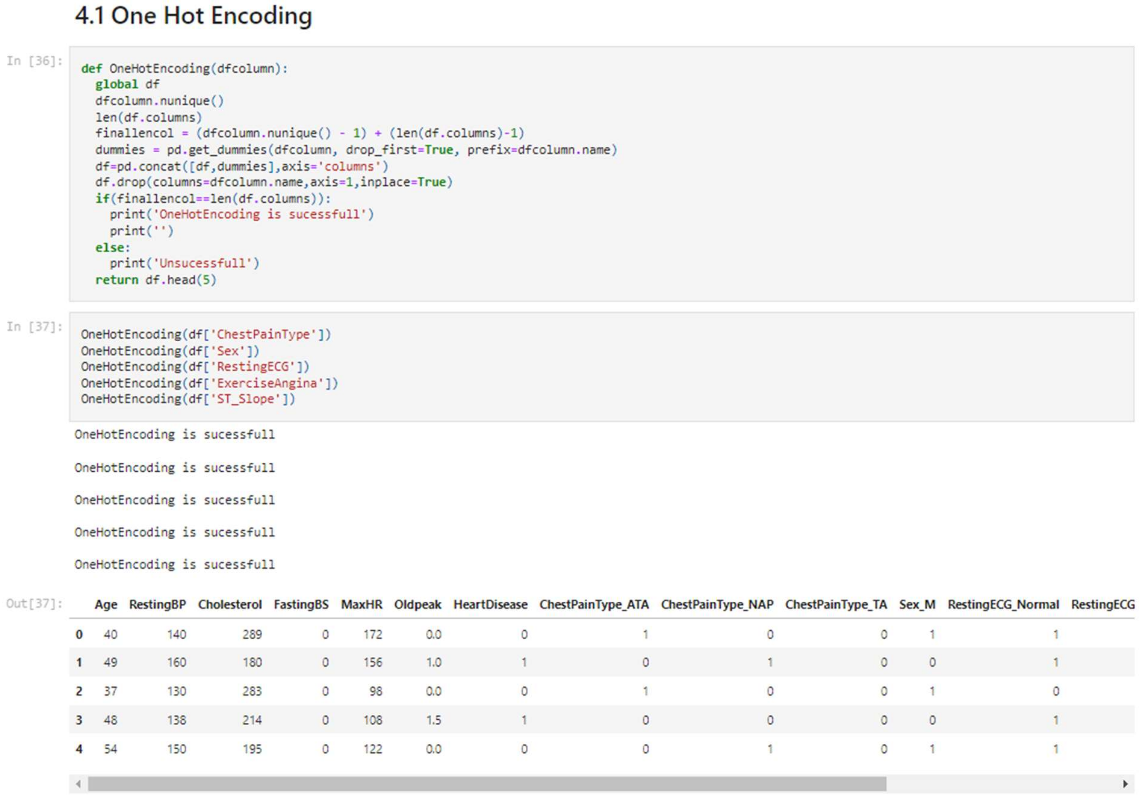
6.1. Training Data:

The observations in the training set form the experience that the algorithm uses to learn. In supervised learning problems, each observation consists of an observed output variable and one or more observed input variables.

6.1. Test Data:

The test set is a set of observations used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are included in the test set. If the test set does contain examples from the training set, it will be difficult to assess whether the algorithm has learned to generalize from the training set or has simply memorized it.

Before splitting dataset into train and test we first encode categorical variables as dummy variables and segregate feature and target variable.



4.1 One Hot Encoding

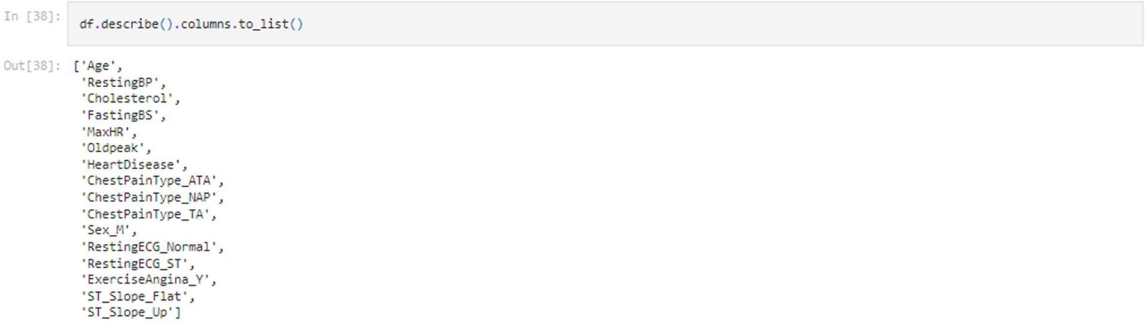
In [36]:
```
def OneHotEncoding(dfcolumn):
  global df
  dfcolumn.nunique()
  len(df.columns)
  finallencol = (dfcolumn.nunique() - 1) + (len(df.columns)-1)
  dummies = pd.get_dummies(dfcolumn, drop_first=True, prefix=dfcolumn.name)
  df=pd.concat([df,dummies],axis='columns')
  df.drop(columns=dfcolumn.name,axis=1,inplace=True)
  if(finallencol==len(df.columns)):
    print('OneHotEncoding is sucessfull')
    print('')
  else:
    print('Unsucessfull')
  return df.head(5)
```

In [37]:
```
OneHotEncoding(df['ChestPainType'])
OneHotEncoding(df['Sex'])
OneHotEncoding(df['RestingECG'])
OneHotEncoding(df['ExerciseAngina'])
OneHotEncoding(df['ST_Slope'])
```

```
OneHotEncoding is sucessfull

OneHotEncoding is sucessfull

OneHotEncoding is sucessfull

OneHotEncoding is sucessfull

OneHotEncoding is sucessfull
```

Out[37]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	Sex_M	RestingECG_Normal	RestingECG
0	40	140	289	0	172	0.0	0	1	0	0	1	1	
1	49	160	180	0	156	1.0	1	0	1	0	0	1	
2	37	130	283	0	98	0.0	0	1	0	0	1	0	
3	48	138	214	0	108	1.5	1	0	0	0	0	1	
4	54	150	195	0	122	0.0	0	0	1	0	1	1	

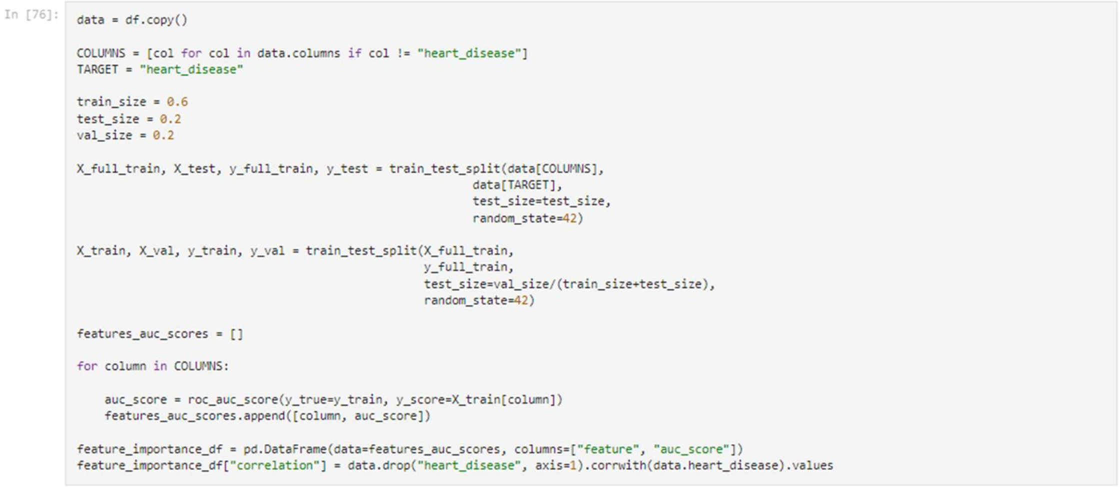


```
In [38]: df.describe().columns.to_list()

Out[38]: ['Age',
          'RestingBP',
          'Cholesterol',
          'FastingBS',
          'MaxHR',
          'Oldpeak',
          'HeartDisease',
          'ChestPainType_ATA',
          'ChestPainType_NAP',
          'ChestPainType_TA',
          'Sex_M',
          'RestingECG_Normal',
          'RestingECG_ST',
          'ExerciseAngina_Y',
          'ST_Slope_Flat',
          'ST_Slope_Up']
```

6.3. Stratified Train Test Splits:

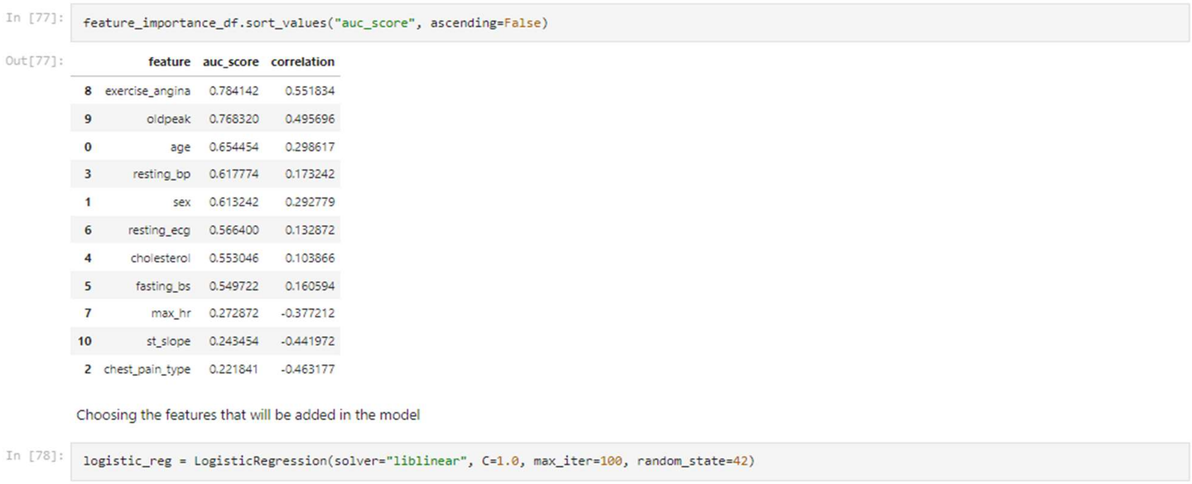
Some classification problems do not have a balanced number of examples for each class label. As such, it is desirable to split the dataset into train and test sets in a way that preserves the same proportions of examples in each class as observed in the original dataset. This is called a stratified train-test split.

We can achieve this by setting the "stratify" argument to the y component of the original dataset. This will be used by the train_test_split () function to ensure that both the train and test sets have the proportion of examples in each class that is present in the provided "y" array.



```
In [76]: data = df.copy()

COLUMNS = [col for col in data.columns if col != "heart_disease"]
TARGET = "heart_disease"

train_size = 0.6
test_size = 0.2
val_size = 0.2

X_full_train, X_test, y_full_train, y_test = train_test_split(data[COLUMNS],
                                                           data[TARGET],
                                                           test_size=test_size,
                                                           random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_full_train,
                                                  y_full_train,
                                                  test_size=val_size/(train_size+test_size),
                                                  random_state=42)

features_auc_scores = []

for column in COLUMNS:

    auc_score = roc_auc_score(y_true=y_train, y_score=X_train[column])
    features_auc_scores.append([column, auc_score])

feature_importance_df = pd.DataFrame(data=features_auc_scores, columns=["feature", "auc_score"])
feature_importance_df["correlation"] = data.drop("heart_disease", axis=1).corrwith(data.heart_disease).values
```

A table with **auc_score** and correlation with **heart_disease** of each column in the dataset



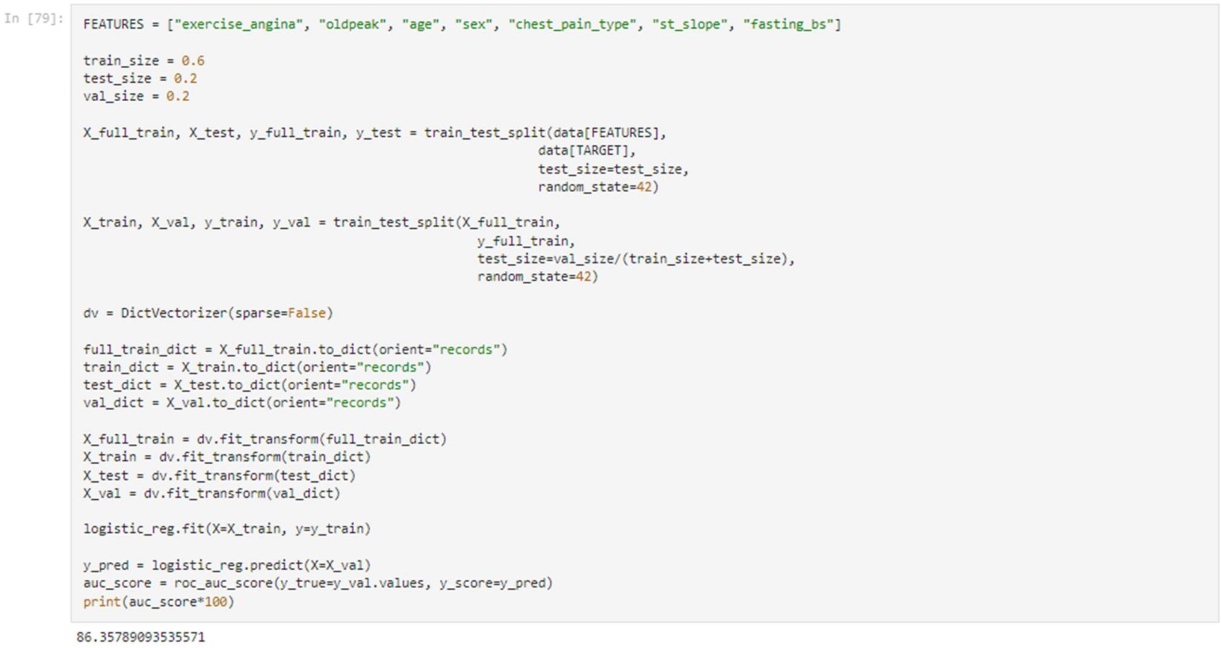
```
In [77]: feature_importance_df.sort_values("auc_score", ascending=False)
```

Out[77]:

	feature	auc_score	correlation
8	exercise_angina	0.784142	0.551834
9	oldpeak	0.768320	0.495696
0	age	0.654454	0.298617
3	resting_bp	0.617774	0.173242
1	sex	0.613242	0.292779
6	resting_ecg	0.566400	0.132872
4	cholesterol	0.553046	0.103866
5	fasting_bs	0.549722	0.160594
7	max_hr	0.272872	-0.377212
10	st_slope	0.243454	-0.441972
2	chest_pain_type	0.221841	-0.463177

Choosing the features that will be added in the model

```
In [78]: logistic_reg = LogisticRegression(solver="liblinear", C=1.0, max_iter=100, random_state=42)
```



```
In [79]: FEATURES = ["exercise_angina", "oldpeak", "age", "sex", "chest_pain_type", "st_slope", "fasting_bs"]

         train_size = 0.6
         test_size = 0.2
         val_size = 0.2

         X_full_train, X_test, y_full_train, y_test = train_test_split(data[FEATURES],
                                                                       data[TARGET],
                                                                       test_size=test_size,
                                                                       random_state=42)

         X_train, X_val, y_train, y_val = train_test_split(X_full_train,
                                                           y_full_train,
                                                           test_size=val_size/(train_size+test_size),
                                                           random_state=42)

         dv = DictVectorizer(sparse=False)

         full_train_dict = X_full_train.to_dict(orient="records")
         train_dict = X_train.to_dict(orient="records")
         test_dict = X_test.to_dict(orient="records")
         val_dict = X_val.to_dict(orient="records")

         X_full_train = dv.fit_transform(full_train_dict)
         X_train = dv.fit_transform(train_dict)
         X_test = dv.fit_transform(test_dict)
         X_val = dv.fit_transform(val_dict)

         logistic_reg.fit(X=X_train, y=y_train)

         y_pred = logistic_reg.predict(X=X_val)
         auc_score = roc_auc_score(y_true=y_val.values, y_score=y_pred)
         print(auc_score*100)

         86.35789093535571
```

CHAPTER 7

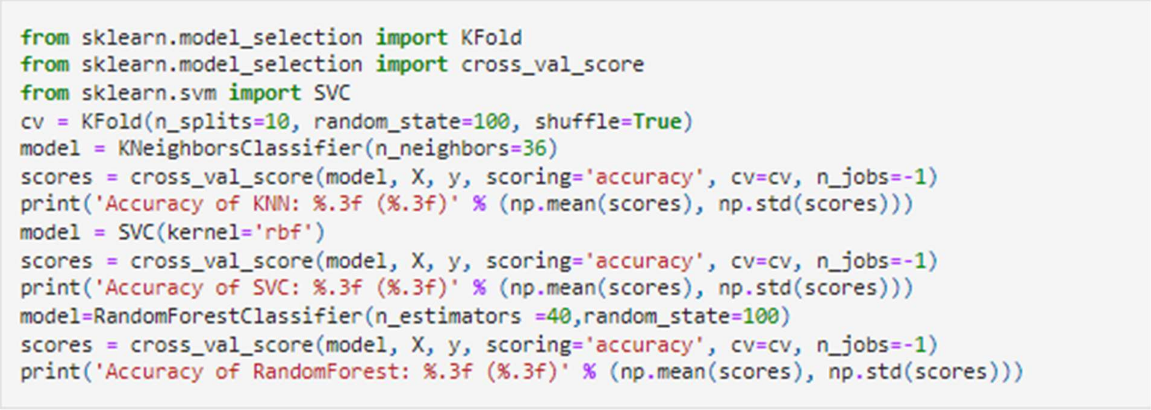
CROSS VALIDATION & MODEL BUILDING

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold crossvalidation.

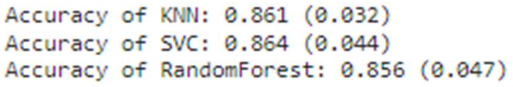
Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

In this step, we will build different baseline models and perform 10-fold cross validation to filter top performing baseline models to be used in level 0 of stacked ensemble method.



```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
cv = KFold(n_splits=10, random_state=100, shuffle=True)
model = KNeighborsClassifier(n_neighbors=36)
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
print('Accuracy of KNN: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
model = SVC(kernel='rbf')
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
print('Accuracy of SVC: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
model=RandomForestClassifier(n_estimators =40,random_state=100)
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
print('Accuracy of RandomForest: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

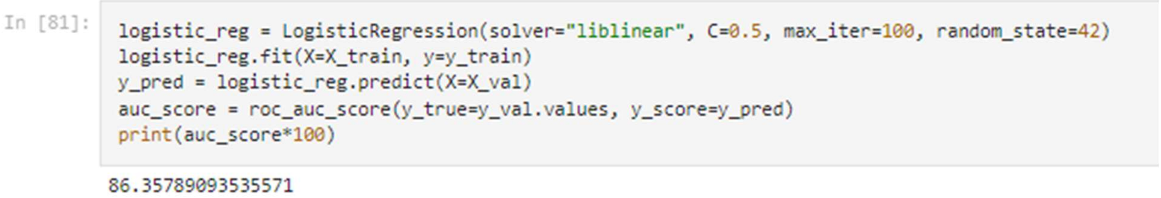
Figure 7.1. Cross Validation of all models



```
Accuracy of KNN: 0.861 (0.032)
Accuracy of SVC: 0.864 (0.044)
Accuracy of RandomForest: 0.856 (0.047)
```

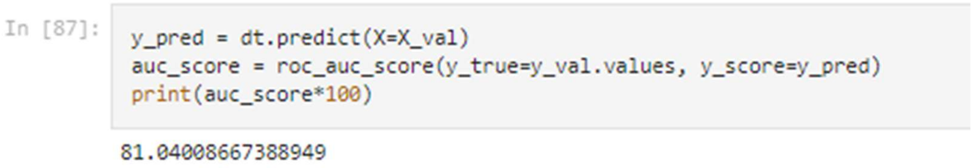
Figure 7.2. Performance of all the baseline models

As we can see the performance of the baseline models, Random forest is the highest performer among all the models.



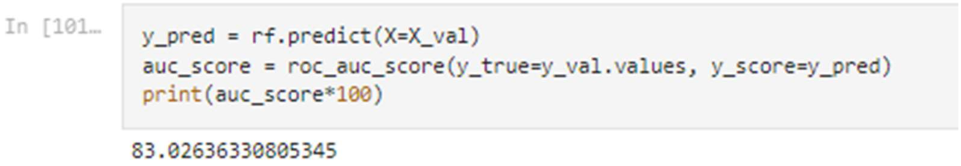
```
In [81]:  logistic_reg = LogisticRegression(solver="liblinear", C=0.5, max_iter=100, random_state=42)
          logistic_reg.fit(X=X_train, y=y_train)
          y_pred = logistic_reg.predict(X=X_val)
          auc_score = roc_auc_score(y_true=y_val.values, y_score=y_pred)
          print(auc_score*100)

          86.35789093535571
```

Figure 7.3.: Model building of Logistic Regression Classifier



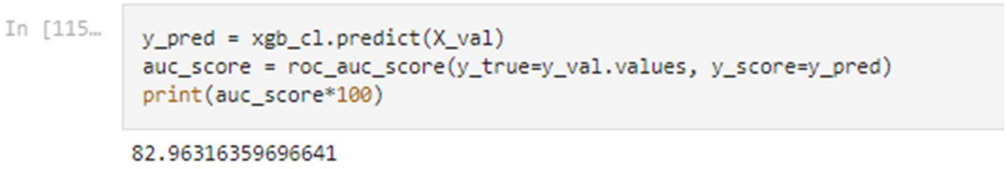
```
In [87]:
y_pred = dt.predict(X=X_val)
auc_score = roc_auc_score(y_true=y_val.values, y_score=y_pred)
print(auc_score*100)

81.04008667388949
```

Figure 7.4.: Model building of Decision Tree Classifier



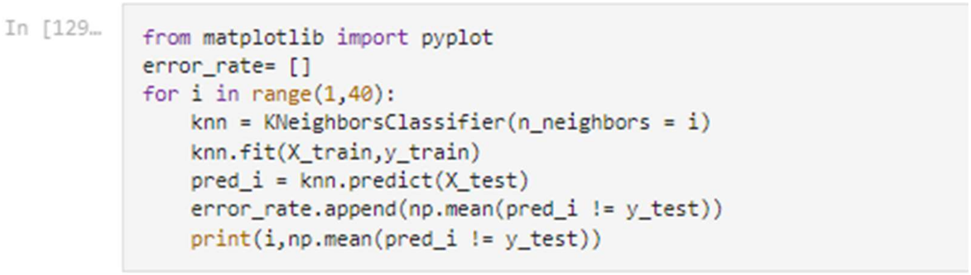
```
In [101…
y_pred = rf.predict(X=X_val)
auc_score = roc_auc_score(y_true=y_val.values, y_score=y_pred)
print(auc_score*100)

83.02636330805345
```

Figure 7.5.: Model building of Random Forest Classifier



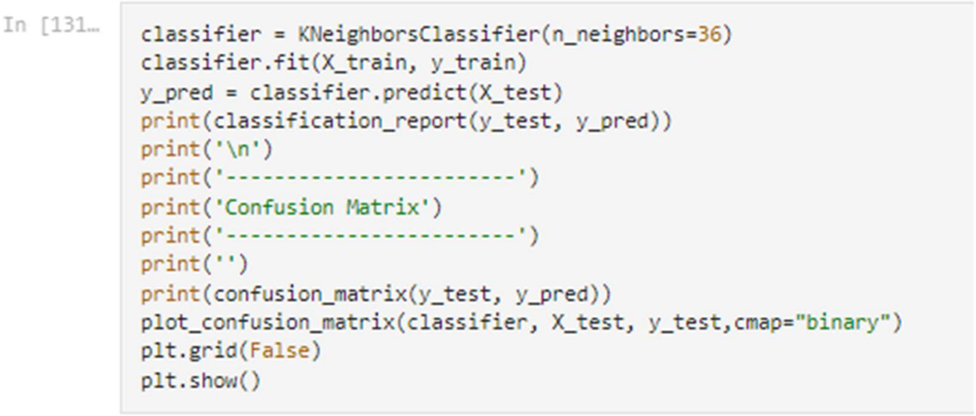
```
In [115…
y_pred = xgb_cl.predict(X_val)
auc_score = roc_auc_score(y_true=y_val.values, y_score=y_pred)
print(auc_score*100)

82.96316359696641
```

Figure 7.6.: Model building of XGB Classifier



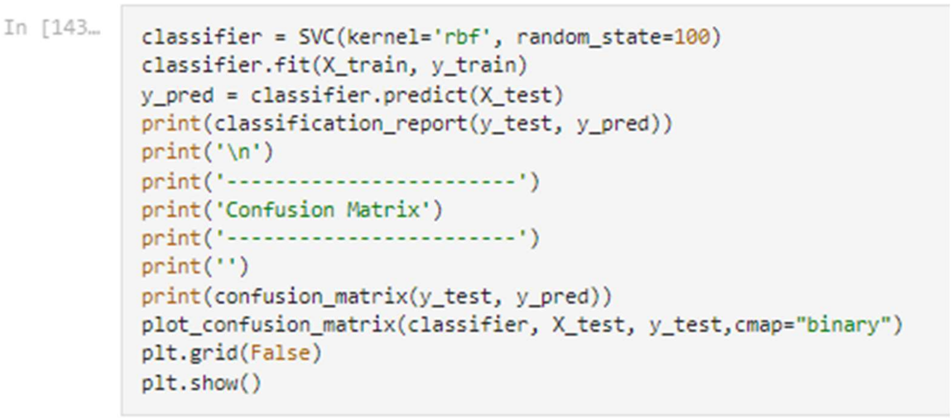
```
In [129…
from matplotlib import pyplot
error_rate= []
for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
    print(i,np.mean(pred_i != y_test))
```

Figure 7.7.: Model building of K Value Estimation



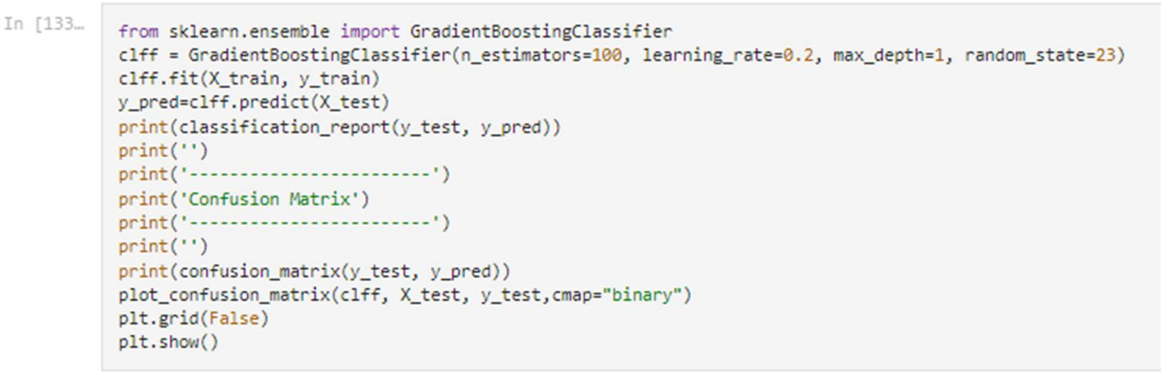
```
In [131…
classifier = KNeighborsClassifier(n_neighbors=36)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print(classification_report(y_test, y_pred))
print('\n')
print('------------------------')
print('Confusion Matrix')
print('------------------------')
print('')
print(confusion_matrix(y_test, y_pred))
plot_confusion_matrix(classifier, X_test, y_test,cmap="binary")
plt.grid(False)
plt.show()
```

Figure 7.8.: Model building of KNN



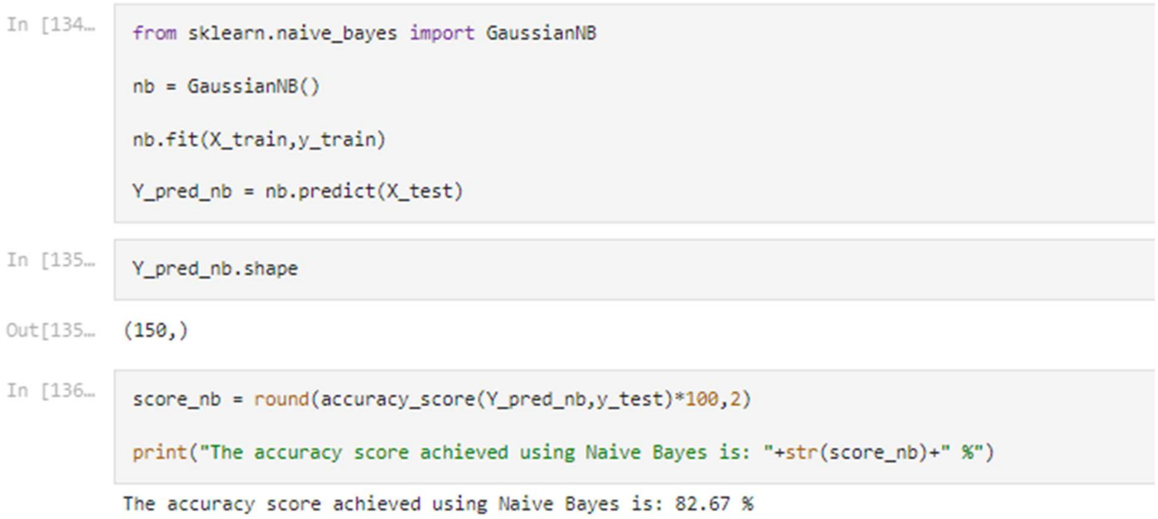
```
In [143…  classifier = SVC(kernel='rbf', random_state=100)
          classifier.fit(X_train, y_train)
          y_pred = classifier.predict(X_test)
          print(classification_report(y_test, y_pred))
          print('\n')
          print('------------------------')
          print('Confusion Matrix')
          print('------------------------')
          print('')
          print(confusion_matrix(y_test, y_pred))
          plot_confusion_matrix(classifier, X_test, y_test,cmap="binary")
          plt.grid(False)
          plt.show()
```

Figure 7.4.: Model building of Support Vector Classifier



```
In [133…  from sklearn.ensemble import GradientBoostingClassifier
          clff = GradientBoostingClassifier(n_estimators=100, learning_rate=0.2, max_depth=1, random_state=23)
          clff.fit(X_train, y_train)
          y_pred=clff.predict(X_test)
          print(classification_report(y_test, y_pred))
          print('')
          print('-----------------------')
          print('Confusion Matrix')
          print('-----------------------')
          print('')
          print(confusion_matrix(y_test, y_pred))
          plot_confusion_matrix(clff, X_test, y_test,cmap="binary")
          plt.grid(False)
          plt.show()
```

Figure 7.4.: Model building of Gradient Boosting Classifier



```
In [134…  from sklearn.naive_bayes import GaussianNB

          nb = GaussianNB()

          nb.fit(X_train,y_train)

          Y_pred_nb = nb.predict(X_test)

In [135…  Y_pred_nb.shape

Out[135…  (150,)

In [136…  score_nb = round(accuracy_score(Y_pred_nb,y_test)*100,2)

          print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")

          The accuracy score achieved using Naive Bayes is: 82.67 %
```

Figure 7.4.: Model building of Naïve Bayes Classifier

CHAPTER 8

MODEL EVALUATION

8.1. Accuracy:

Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

It works well only if there are equal number of samples belonging to each class.

8.2. Precision:

Precision is the ratio between the True Positives and all the Positives.

$$\textbf{Precision} = \frac{\textbf{True Positives}}{\textbf{True Positives + False Positives}}$$

8.3. Recall:

The recall is the measure of our model correctly identifying True Positives.

$$\text{Recall} = \frac{\mathit{True\ Positive}}{\mathit{True\ Positive} + \mathit{False\ Negative}}$$

$$= \frac{\mathit{True\ Positive}}{\mathit{Total\ Actual\ Positive}}$$

8.4. Sensitivity:

sensitivity is the metric that evaluates a model’s ability to predict true positives of each available category.

8.5. Specificity:

Specificity is the metric that evaluates a model’s ability to predict true negatives of each available category.

$$\textbf{Sensitivity} = \frac{\textbf{True Positives}}{\textbf{True Positives} + \textbf{False Negatives}}$$

$$\textbf{Specificity} = \frac{\textbf{True Negatives}}{\textbf{True Negatives} + \textbf{False Positives}}$$

8.6. F1 Score:

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost.

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

8.7. Log Loss:

Logarithmic loss measures the performance of a classification model where the prediction input is a probability value between 0 and 1. A perfect model would have a log loss of 0. Log loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high log loss.

$$LogLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} * \ log(p_{ij})$$

CHAPTER 9

FEATURE SCALING AND SELECTION

9.1. Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.

The simplest method is rescaling the range of features to scale the range in [0, 1] or [−1, 1]. The general formula is given as:

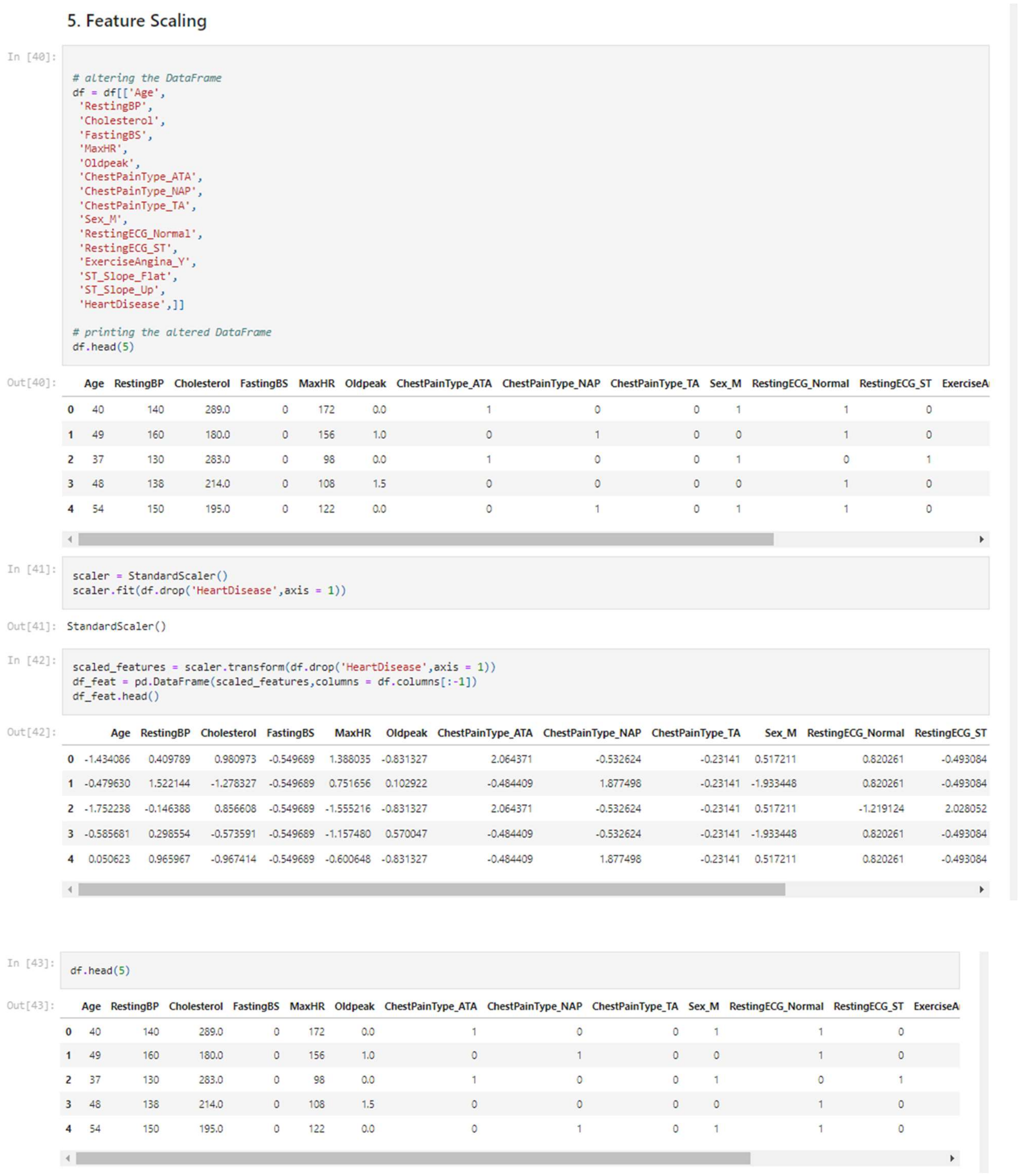
X' = (X — Xmin)/ (Xmax — Xmin)

where, X' is the value we want to rescale, X is the given value, Xmax is the largest value of X and Xmin the smallest.

Let us consider, old weights = [115,140,175] and we are going to scale for the value 140.

X' = (140–115)/ (175–115) = 0.41666

Therefore, the range is, [0,0.41666,1].



5. Feature Scaling

In [40]:

```
# altering the DataFrame
df = df[['Age',
 'RestingBP',
 'Cholesterol',
 'FastingBS',
 'MaxHR',
 'Oldpeak',
 'ChestPainType_ATA',
 'ChestPainType_NAP',
 'ChestPainType_TA',
 'Sex_M',
 'RestingECG_Normal',
 'RestingECG_ST',
 'ExerciseAngina_Y',
 'ST_Slope_Flat',
 'ST_Slope_Up',
 'HeartDisease',]]

# printing the altered DataFrame
df.head(5)
```

Out[40]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	Sex_M	RestingECG_Normal	RestingECG_ST	ExerciseA
0	40	140	289.0	0	172	0.0	1	0	0	1	1	0	
1	49	160	180.0	0	156	1.0	0	1	0	0	1	0	
2	37	130	283.0	0	98	0.0	1	0	0	1	0	1	
3	48	138	214.0	0	108	1.5	0	0	0	0	1	0	
4	54	150	195.0	0	122	0.0	0	1	0	1	1	0	

In [41]:

```
scaler = StandardScaler()
scaler.fit(df.drop('HeartDisease',axis = 1))
```

Out[41]: StandardScaler()

In [42]:

```
scaled_features = scaler.transform(df.drop('HeartDisease',axis = 1))
df_feat = pd.DataFrame(scaled_features,columns = df.columns[:-1])
df_feat.head()
```

Out[42]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	Sex_M	RestingECG_Normal	RestingECG_ST
0	-1.434086	0.409789	0.980973	-0.549689	1.388035	-0.831327	2.064371	-0.532624	-0.23141	0.517211	0.820261	-0.493084
1	-0.479630	1.522144	-1.278327	-0.549689	0.751656	0.102922	-0.484409	1.877498	-0.23141	-1.933448	0.820261	-0.493084
2	-1.752238	-0.146388	0.856608	-0.549689	-1.555216	-0.831327	2.064371	-0.532624	-0.23141	0.517211	-1.219124	2.028052
3	-0.585681	0.298554	-0.573591	-0.549689	-1.157480	0.570047	-0.484409	-0.532624	-0.23141	-1.933448	0.820261	-0.493084
4	0.050623	0.965967	-0.967414	-0.549689	-0.600648	-0.831327	-0.484409	1.877498	-0.23141	0.517211	0.820261	-0.493084

In [43]:

```
df.head(5)
```

Out[43]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	Sex_M	RestingECG_Normal	RestingECG_ST	ExerciseA
0	40	140	289.0	0	172	0.0	1	0	0	1	1	0	
1	49	160	180.0	0	156	1.0	0	1	0	0	1	0	
2	37	130	283.0	0	98	0.0	1	0	0	1	0	1	
3	48	138	214.0	0	108	1.5	0	0	0	0	1	0	
4	54	150	195.0	0	122	0.0	0	1	0	1	1	0	

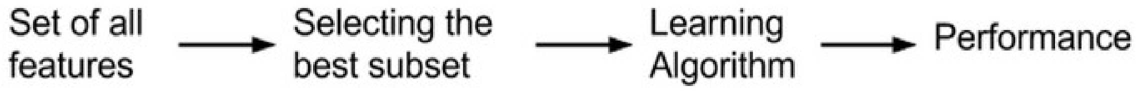
9.2. Feature Selection

Why do we have to perform feature selection?

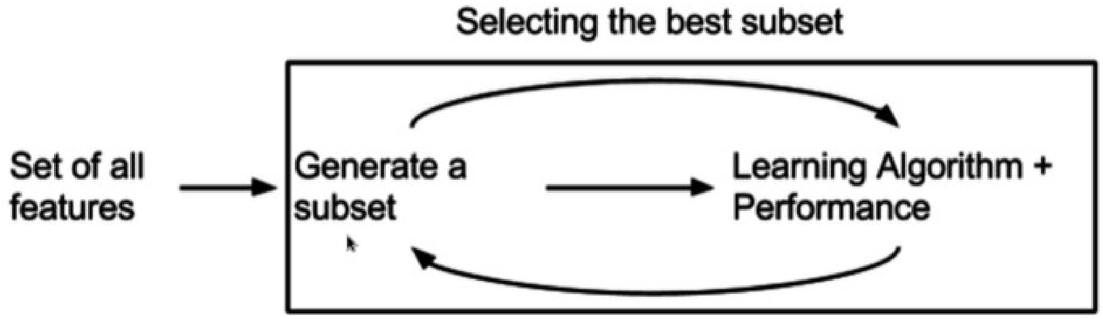
- Knowledge discovery, Interpretability and to gain some insights.
- Curse of dimensionality.

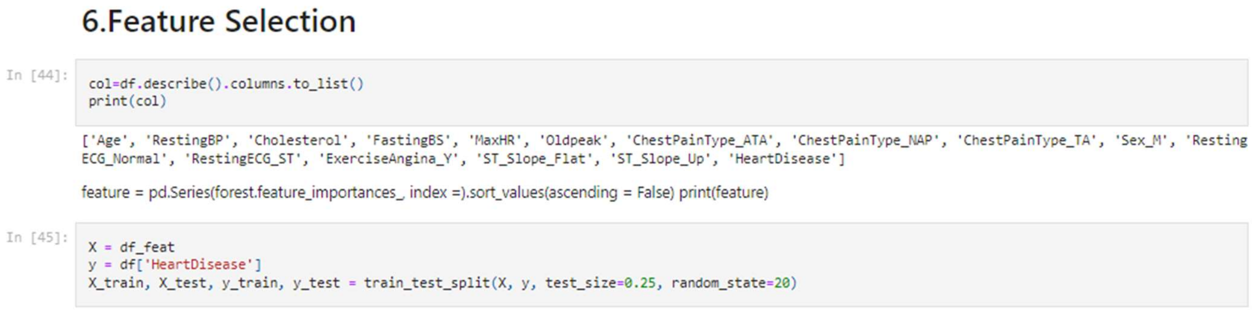
There are two methods of Feature Selection :

- Filtering — Filter type methods select variables regardless of the model. They are based only on general features like the correlation with the variable to predict. Filter methods suppress the least interesting variables. They are mainly used as a pre-process method.



- Wrapping — Wrapper methods evaluate subsets of variables which allows, unlike filter approaches, to detect the possible interactions between variables.





6.Feature Selection

In [44]:
```
col=df.describe().columns.to_list()
print(col)
```

```
['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak', 'ChestPainType_ATA', 'ChestPainType_NAP', 'ChestPainType_TA', 'Sex_M', 'Resting
ECG_Normal', 'RestingECG_ST', 'ExerciseAngina_Y', 'ST_Slope_Flat', 'ST_Slope_Up', 'HeartDisease']
```

feature = pd.Series(forest.feature_importances_, index =).sort_values(ascending = False) print(feature)

In [45]:
```
X = df_feat
y = df['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=20)
```

CHAPTER 10

CONCLUSION

Heart disease is the most common disease in India. Early detection of heart diseases will increase the survival rate hence this research work is intended to predict the whether the patient has heart disease or not with the help of clinical data which will assist the diagnosis process.

- Our machine learning algorithm can now classify patients with heart disease. Now we can properly diagnose patients, & get them the help they need to recover. Ny diagnosis detecting these features early, we may prevent worse symptoms from arising later.

- Our Random Forest Algorithm yields the highest accuracy, 90%. Any accuracy above 70% is considered good.

- The top 5 most contributoi0pn features are:

    - Max heart Rate achieved
    - Cholesterol
    - St_depression
    - Age
    - Exercise_induced_angima

CHAPTER 11

REFERENCES

[1] Soni J, Ansari U, Sharma D & Soni S (2011). Predictive data mining for medical diagnosis: an overview of heart disease prediction. International Journal of Computer Applications, 17(8), 43-8

[2] Dangare C S & Apte S S (2012). Improved study of heart disease prediction system using data mining classification techniques. International Journal of Computer Applications, 47(10), 44-8.

[3] Ordonez C (2006). Association rule discovery with the train and test approach for heart disease prediction. IEEE Transactions on Information Technology in Biomedicine, 10(2), 334-43.

[4] Shinde R, Arjun S, Patil P & Waghmare J (2015). An intelligent heart disease prediction system using k-means clustering and Naïve Bayes algorithm. International Journal of Computer Science and Information Technologies, 6(1), 637-9.

[5] Bashir S, Qamar U & Javed M Y (2014, November). An ensemble-based decision support framework for intelligent heart disease diagnosis. In International Conference on Information Society (i-Society 2014) (pp. 259-64). IEEE. ICCRDA 2020 IOP Conf. Series: Materials Science and Engineering 1022 (2021) 012072 IOP Publishing doi:10.1088/1757-899X/1022/1/012072 9

[6] Jee S H, Jang Y, Oh D J, Oh B H, Lee S H, Park S W & Yun Y D (2014). A coronary heart disease prediction model: the Korean Heart Study. BMJ open, 4(5), e005025.

[7] Ganna A, Magnusson P K, Pedersen N L, de Faire U, Reilly M, Ärnlöv J & Ingelsson E (2013). Multilocus genetic risk scores for coronary heart disease prediction. Arteriosclerosis, thrombosis, and vascular biology, 33(9), 2267-72.

[8] Jabbar M A, Deekshatulu B L & Chandra P (2013, March). Heart disease prediction using lazy associative classification. In 2013 International MutliConference on Automation, Computing,Communication, Control and Compressed Sensing (iMac4s) (pp. 40- 6). IEEE.

[9] Brown N, Young T, Gray D, Skene A M & Hampton J R (1997). Inpatient deaths from acute myocardial infarction, 1982-92: analysis of data in the Nottingham heart attack register. BMJ, 315(7101), 159-64.

[10] Folsom A R, Prineas R J, Kaye S A & Soler J T (1989). Body fat distribution and self-reported prevalence of hypertension, heart attack, and other heart disease in older women. International journal of epidemiologyy, 18(2), 361-7.

[11] Chen A H, Huang S Y, Hong P S, Cheng C H & Lin E J (2011, September). HDPS: Heart disease prediction system. In 2011 Computing in Cardiology (pp. 557- 60). IEEE.

[12] Parthiban, Latha and R Subramanian. "Intelligent heart disease prediction system using CANFIS and genetic algorithm." International Journal of Biological, Biomedical and Medical Sciences 3.3 (2008).

[13] Wolgast G, Ehrenborg C, Israelsson A, Helander J, Johansson E & Manefjord H (2016). Wireless body area network for heart attack detection [Education Corner]. IEEE antennas and

propagation magazine, 58(5), 84-92.

[14] Patel S & Chauhan Y (2014). Heart attack detection and medical attention using motion sensing device -kinect. International Journal of Scientific and Research Publications, 4(1), 1-4.

[15] Piller L B, Davis B R, Cutler J A, Cushman W C, Wright J T, Williamson J D & Haywood L J (2002). Validation of heart failure events in the Antihypertensive and Lipid Lowering Treatment to Prevent Heart Attack Trial (ALLHAT) participants assigned to doxazosin and chlorthalidone. Current controlled trials in cardiovascular medicine

[16] Raihan M, Mondal S, More A, Sagor M O F, Sikder G, Majumder M A & Ghosh K (2016, December). Smartphone based ischemic heart disease (heart attack) risk prediction using clinical data and data mining approaches, a prototype design. In 2016 19th International Conference on Computer and Information Technology (ICCIT) (pp. 299-303). IEEE.

[17] A. Aldallal and A. A. A. Al-Moosa, "Using Data Mining Techniques to Predict Diabetes and Heart Diseases", 2018 4th International Conference on Frontiers of Signal Processing (ICFSP), pp. 150-154, 2018, September.

[18] Takci H (2018). Improvement of heart attack prediction by the feature selection methods. Turkish Journal of Electrical Engineering & Computer Sciences, 26(1), 1-10.

[19] Ankita Dewan and Meghna Sharma, "Prediction of heart disease using a hybrid technique in data mining classification", 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom).

[20] Aditya Methaila, Prince Kansal, Himanshu Arya and Pankaj Kumar, "Early heart disease prediction using data mining techniques", Computer Science & Information Technology Journal, pp. 53-59, 2014.