

## **MSc Data Science Project**

**7PAM2002-0509-2022**

Department of Physics, Astronomy and Mathematics

### **Data Science FINAL PROJECT REPORT**

#### **Project Title:**

**PREVENTION OF HEART DISEASE THROUGH EARLY DETECTION**

#### **Student Name and SRN:**

Omesha Prashanthika Samarakoon  
21088775

Supervisor: DR. Mykola Gordovskyy

Date Submitted: 31 August 2023

Word Count: 9662

## DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.


I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Omesha Prashanthika Samarakoon

Student Name signature:  .....

Student SRN number: 21088775

## **ABSTRACT**

Heart disease is one of the major reasons for deaths worldwide. According to the World Health Organization, an average of 17.9 million people lost their lives in 2016, and the number of deaths shows an increasing trend. Therefore, it is important to discover accurate and more feasible methods to detect and predict heart diseases early to start treatments as early as possible. As a solution, experiments have been conducted with machine learning and deep learning classifiers to predict diseases using patients' past records. Supervised and unsupervised learning algorithms are able to find hidden patterns in complex datasets. In previous research, commonly used algorithms include Decision Tree, Logistic Regression, Support Vector Machine, and Artificial Neural Network, which perform well with high accuracy levels. Moreover, I used Logistic Regression, Decision Tree, and Artificial Neural Network classifiers to detect and predict the probability of risk presence in heart disease early, in a time and cost-effective manner. Through my project, I have achieved the highest accuracy of 85% and the lowest False Negative rate of 13% using the Decision Tree algorithm with four important independent variables: ST Slope, Chest Pain Type, Sex, and Exercise Angina. Therefore, this research would be helpful for future medical and artificial intelligence combinations to heal the world.

# CONTENTS

DECLARATION STATEMENT .....	2
ABSTRACT .....	3
LIST OF TABLES.....	6
LIST OF FIGURES.....	9
CHAPTER ONE: INTRODUCTION AND RELATED WORKS FOR HEARTS DISEASE PREDICTION .....	9
1.1. BACKGROUND FOR STUDY .....	10
1.2. PROBLEM STATEMENT.....	10
1.3. JUSTIFICATION OF THE STUDY .....	11
1.4. RESEARCH QUESTION .....	12
1.5. AIMS AND OBJECTIVES .....	13
1.6. RELATED WORKS HEART DISEASE PREDICTIONS THROUGH MACHINE LEARNING APPROACHES .....	13
CHAPTER TWO: METHODOLOGY FOR HEART DISEASE PREDICTION.....	15
2.1 DESCRIPTION OF THE DATASET.....	16
2.2 DATA PREPROCESSING .....	20
2.3 FEATURE SELECTION.....	21
CHAPTER THREE: TRAINED ALGORITHMS TO ANALYSIS THE PERFORMANCE .....	26
3.1 SELECTED ALGORITHMS TO TRAIN THE DATA .....	27
3.2 LOGISTIC REGRESSION .....	27
3.3 DECISION TREE .....	28
3.4 ARTIFICIAL NEURAL NETWORK .....	29
3.5 EVALUATION PROCESS USED .....	30
CHAPTER FOUR: ANALYSIS OF RESULT REACHED THROUGH TRAINED MODELS .....	31
4.1 FEATURES SECTION USING IMPORTANCE SCORE.....	32
4.2 EARLY HEART DISEASE PREDICTION THROUGH LOGISTIC REGRESSION.....	33
4.3 HEART DISEASE EARLY PREDICTION THROUGH DECISION TREE ALGORITHM .....	37
4.3.1 APPROACH 01: TRAIN A DECISION TREE WITH NUMERICAL INDEPENDENT VARIABLE .....	37
4.3.2 APPROACH 02: TRAIN A DECISION TREE WITH CATEGORICAL INDEPENDENT VARIABLES.....	39
4.3.3 APPROACH 03: TRAIN A DECISION TREE WITH CATEGORICAL INDEPENDENT VARIABLE.....	40

4.3.4	DECISION TREE MODEL OPTIMIZATION .....	42
4.4	EARLY PREDICTION OF HEART DISEASE THROUGH ANN.....	46
4.5	EVALUATE THE MODEL USING CONFUSION METRIC .....	50
4.5.1	LOGISTIC REGRESSION .....	50
4.5.2	DECISION TREE.....	51
4.5.3	ARTIFICIAL NEURAL NETWORK .....	52
CHAPTER FIVE: CONCLUSION AND FUTURE WORK RELATED WITH RESEARCH .....		53
5.1	CONCLUSION.....	54
5.2	LIMITATION AND FUTURE WORKS .....	54
REFERANCE.....		55
APPENDIX.....		57

## LIST OF TABLES

<b>Table 1.0</b>	<i>The feature importance score according to the logistic regression coefficient analysis</i>	25
<b>Table 2.0</b>	<i>The accuracy score of Decision Tree (3<sup>rd</sup> approach) with different tree depths</i>	42

## LIST OF FIGURES

<b>Figure 1.0</b>	<i>The Visualization of the distribution of continuous variables</i>	18
<b>Figure 2.0</b>	<i>Visualize the numerical continuous data to identify the outliers properly</i>	19
<b>Figure 3.0</b>	<i>the visualization for identifying the relationship between a target variable and numerical continuous independent variables, data plotted using a violin plot</i>	22
<b>Figure 4.0</b>	<i>the visualization for identifying the relationship between a dependent variable and categorical variables, data plotted using a counter plot</i>	23
<b>Figure 5.0</b>	<i>The Correlation between independent variables and dependent variable</i>	24
<b>Figure 6.0</b>	<i>The correlation among Resting ECG and Heart Disease with respect to each category</i>	25
<b>Figure 7.0</b>	<i>The figure illustrates how the ANN work internally, what is the process working in a neuron, as it shows the input data process with weights and calculation happens in a first step of inside neuron and then the result goes through the activation function and give the output as probability('Data Flair,' (no date)).</i>	29
<b>Figure 8.0</b>	<i>The importance score visualization for each independent variable according to the Logistic Regression Coefficient</i>	32
<b>Figure 9.0</b>	<i>The output of selecting best parameters to train the Logistic Regression model using GridSearchCV, the default parameters selected as good ones.</i>	34
<b>Figure 10.0</b>	<i>Show the logistic curve with respect to each independent variable and prediction probability</i>	35
<b>Figure 11.0</b>	<i>The above figure shows how the model performs with real life data, the set of data selected from testing data set and feed it as input data for the model and check the probability the positive probability was 11% then patient at low risk, the actual output is also '0' as in original data</i>	36
<b>Figure 12.0</b>	<i>The validation curve with respect to the maximum depth parameter of decision tree to identify whether the model is overfitting.</i>	38
<b>Figure 13.0</b>	<i>The validation curve visualization according to the Decision Tree approach</i>	40
<b>Figure 14.0</b>	<i>The Decision Tree model visualization how it trains with given data</i>	41
<b>Figure 15.0</b>	<i>The visualization of validation curve after define the max_depth to 3</i>	42

<b>Figure 16.0</b>	<i>The visualization of trimmed Decision Tree</i>	43
<b>Figure 17.0</b>	<i>The relationship between ST Slope and Chest Pain Type with respect to the occurrence of Heart Disease</i>	44
<b>Figure 18.0</b>	<i>Visualize how the decision tree model predict the Heart Disease with real life data</i>	45
<b>Figure 19.0</b>	<i>The Visualization of structure of train ANN</i>	46
<b>Figure 20.0</b>	<i>Visualization of Validation curve of ANN model according to the change of number of Epochs</i>	47
<b>Figure 21.0</b>	<i>Figure 21.0 Output result of hyper parameter Tuning through GridSearchCV in ANN</i>	48
<b>Figure 22.0</b>	<i>The ANN model gives the prediction probability of risk to having heart disease with real life data</i>	49
<b>Figure 23.0</b>	<i>The visualization of Logistic Regression model evaluation</i>	50
<b>Figure 24.0</b>	<i>The visualization of Decision Tree model evaluation</i>	51
<b>Figure 25.0</b>	<i>The visualization of ANN model evaluation</i>	52



# **CHAPTER ONE**

## **INTRODUCTION AND RELATED WORKS FOR HEART DISEASE PREDICTION**

# INTRODUCTION

## 1.1 BACKGROUND FOR THE STUDY

The Largest number of people are effected some kind of heart disease annually and heart disease is the one of the major cause for the global deaths of both men and women. The World Health Organization (WHO) examined that in almost each 34 seconds the coronary disease kills one human being in the world (Kumar and Koushik and Deepak, 2018).

In living organisms Heart acts an important role because it supplies oxygen and nutrients to the body throughout the day. Therefore, early diagnosis and prediction of heart related diseases critically important and should have more accuracy and reliability because a slight fault may reason to lifelong damage or lost the valuable life of a human being, there are number of deaths associated with heart and their counting is climbing up daily in exponential manner (Singh and Kumar, 2020).

According to the estimation report of the World Health Organization (WHO), an average of 17.90 million people lost their lives from cardiovascular disease in 2016. Nearly it is 30% of all worldwide deaths. As investigations of (WHO), the total number of people who is victimizing from heart disease speedily growing up (WHO, 2023).

According to the report was generated by European Society of Cardiology (ESC), 26.5 million adults were already diagnosed as positive heart patient and annually identify 3.8 million were affected by heart disease. But unfortunately 50-55% of heart disease patients lost their lives in initial 1-3 years (Muhammad et al. 2020). So if it is possible to develop and find a method to identify the risk level of heart disease early with high accuracy it can save large number of lives and also a considerable amount of cost as it estimated 4% of the overall healthcare annual expenses for heart disease.

## 1.2 PROBLEM STATEMENT

Early-stage identification of Heart disease is a significant method to overcome this dangerous situation. Most of old investigation methods which are used to investigate heart disease were found complex and expense high cost and take more time due to human errors and more time to assess (Muhammad et al. 2020). If unable to find heart disease early, it may cause too lost the lives.

For this effort machine learning is one of the suitable approaches (Ali, Paul and Ahmed, 2021). Because Machine learning is fine to appropriately handle the complex data which comes through different data sources and the massive range of variables and the amount of data concerned, where Machine Learning succeeds in on increasing datasets. When supplying additional data into a machine learning algorithm it can be trained and give attention on the importance of the superior value of insights. At the permission from the

limitations of the study, Machine learning is ingenious to catch and show the patterns hidden in the data (Osisanwo et al. 2017).

And Machine learning delivers impressive contribution in predicting various incident which is training through natural things. And machines learning algorithms are trained to learn how to work and make use of data and the mixture of both technologies (Singh and Kumar, 2020).

In general machine learning learns from the natural things, it Incorporates on different algorithms of Supervised and Unsupervised Learning which are able to predict and discover the reliability and accuracy of the considerable dataset (Agrawal and Jindal, 2021). And also we can identify projects which analyzing biological factors as testing data such as cholesterol, Blood pressure, sex, age, etc. (Singh and Kumar, 2020).

To early detection and prediction of heart diseases we can consider about the different attributes according to the person. Such as sex, age, pulse rate etc. and can find the relationship of each or multiple attributes to the target variable. I'm going to calculate the co-relation according to each metric in my dataset and analyze the co-relation which are more affected with heart diseases and use selected factors to develop a model to detect and predict the probability of risk to affect from heart diseases early using machine learning. In different researchers calculate The percentage of risk, of the person with coronary diseases is classified by work with different data mining techniques such as Naïve Bayes, KNN, Decision Tree Algorithm, Neural Network, etc. and found the Accuracy of the prediction is high when have more number of attributes (Thomas and Princy, 2016). In this research I'm analyzing a way to identify the risk level with high accuracy with specific attributes which is cost and time effective.

In this project I have chosen Heart Failure Prediction Dataset from Kaggle which includes 11 attributes Age, Sex, Chest Pain Type, Resting BP, Cholesterol, Fasting Blood Sugar, Resting ECG, maximum heart rate, exercise-induced angina, the old-peak, the slope of the peak exercise ST segment, and target value as heart disease. And initially I'm going to analyze the correlation among independent variables vs. dependent variable, and use factors with high correlation to develop a model to predict probability of risk to presence or absence from heart diseases early using machine learning algorithms such as logistic regression, Decision Tree and Neural Network and evaluate the model performance using confusion-matrices through accuracy, sensitivity, precision and false negative rate.\_

The Link for the Dataset:

<https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction/code>.

### **1.3 JUSTIFICATION OF THE STUDY**

Coronary heart disease is the one of the main source of death in the developed countries and is single of the top reason of disease burden in unindustrialized countries. Due to heart disease, there were 7.3 million deaths and make 58 million disability-adjusted life years in 2001 globally. 75 percent of total deaths and 82 percent of the disability-adjusted people comes by heart diseases happened in the developing countries (Gaziano et al. 2010).

Machine learning algorithms are able of analyzing large amount of data from different sectors, Machine learning algorithms give predicted and factual outcomes by minimizing the errors, because it can routine prediction and designing approach to improve an understanding of compound and non-linear interactions among different features. (Shah, Patel and Bharti, 2020). So it is much suitable to use in medical diagnostics.

As a part of data science data mining can discover large datasets to extract unseen important decision-making facts from a collection of a past repository for future investigation. The medical sector contains tremendous data of patients. Healthcare authorities do analysis of data to realize effective analytical decision by using machine learning algorithms. It uses the classification algorithms to predict cardiovascular disease in patients (Shah, Patel and Bharti, 2020).

Mainly, I can identify algorithms used mostly in heart disease prediction such as decision tree, Random forest, Naïve Bayes, K-nearest neighbor algorithms and Logistic Regression.

## **1.4 RESEARCH QUESTIONS**

The research aims to answer the following questions:

- Investigate and identify the metrics with the highest level of correlation when it comes to predicting the early onset of heart disease.
- Find machine learning algorithms best suited to predict coronary heart disease along with probability of risk to effect of from heart disease early of patients with the highest possible accuracy utilizing the metrics the most appropriate metrics that have been deduced (above).

## **1.5 AIMS OF THE RESEARCH AND OBJECTIVES**

The motivation of this research is the prevention of heart disease through early detection here I try to predict the probability of risk to presence or absence heart disease through machine learning algorithms and related past facts. The main objectives of this research are as follows:

- Calculate the correlation between predictive variables vs. target variables and analyze the relationship between these in order to identify the variables that provide the strongest indicators in relation to predicting the increase in the risk of having heart disease.
- Implement a Machine Learning (ML) model using the most appropriate ML algorithms to help predict with an effective accuracy, the possibility of contracting heart disease.
- Compare the relative performance of the machine learning model developed through project with existing machine learning models used to achieve the same goal of predicting heart disease early.

## **1.6 RELATED WORKS OF HEART DISEASE PREDICTIONS THROUGH MACHINE LEARNING APPROACHES.**

The human heart is considered as the most important working muscle of the body in a person. To provide oxygen and nutrients to the body cells to function properly, the heart averagely beats 100,000 times per day and also remove byproducts from the body, as example carbon dioxide from the lungs. So the proper functioning of the heart is a critical fact to live livelier (Kumar, Koushik and Deepak, 2018).

Heart disease is one of the critical reason of deaths in developed and underdeveloped nations, Heart disease caused the Three-fourths of worldwide deaths and 82% of disability-adjusted life years of developing countries in the year 2001(Gaziano et al. 2010).

As a solution for this health care sector needs to recognize the heart illnesses early with high accuracy and cost effective manner for that purpose machine learning algorithms and data mining technologies are better approaches as they have ability to analyze raw data, identify the correlations and use them for solve problems and do the predictions to take actions before danger (Kumar, Koushik and Deepak, 2018).

As Singh and Kumar, (2020) argue they used k-nearest neighbor, decision tree, linear regression and support vector machine algorithms to predicting heart disease, using the UCI repository dataset for implementations. Furthermore, calculate an accuracy score to compare the suitability of algorithms to do the predictions and achieve accuracy scores as 87%, 79%, 78% and 83% respectively. I can identify in this implementation they achieve a high accuracy score in k-nearest neighbor algorithm and also they haven't mentioned about the data normalization throughout the research to overcome the dimensionality reduction while training

And in the Jindal et al. (2021), have trained UCI repository dataset with 14 medical attributes and choose Logistic regression, Random Forest and k-nearest Neighbour (KNN) classification algorithms. And identify that the KNN algorithm gives high accuracy by comparing to the other two algorithms and it achieves 88.52% accuracy score and using the develop models their diagnosis that the patients are at risk of presence heart diseases.

As identified in the journal of Ali et al. (2021) they used six machine learning algorithms to develop a model based on a heart disease dataset from kaggle which has 14 attributes. The implemented algorithms were K-nearest neighbor (KNN), Random forest (RF), Decision tree (DT), AdaboostM1 (ABM1), Logistic regression (LR), Multilayer perceptron (MLP) and compared their testing predictions using accuracy, recall, specificity, kappa statistics, precision and ROC and Precision-Recall curves. Through the evaluation identified the KNN, DT and RF gave the best score on performance with 100% accuracy level by showing the most suitability of machine learning classifiers to predict the heart disease in most effective and efficient manner.

In a research Bharti et al. (2021) has been used Random Forest (RF), Logistic Regression, K Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree (DT), XGBoost and deep learning to predict the heart disease. I have identified that they have used three different approaches. As first approach, uncleaned dataset which was chosen is used for classification without doing any preprocessing, then as second approach, the feature selection was done on data but no process on outliers detection, as the third method the dataset was properly preprocessed normalized the data and done both feature selection and outliers detection using Isolation Forest. According to the accuracy they achieved best result in the third approach consequently the accuracy of the RF is 80.3%, LR is 83.31%, KNN is 84.86%, SVM is 83.29%, DT is 82.33%, and XGBoost is 71.4%. And the KNeighbors achieved high precision of 77.7% and a specificity of 80% as well., and in this approach deep learning achieved 94.2% accuracy as the maximum. When comparing the accuracy of all algorithms, I can identify that deep learning algorithm gave the highest accuracy on this dataset.

# **CHAPTER TWO**

## **METHODOLOGY FOR HEART DISEASE PREDICTION**

# METHODOLOGY FOR HEART DISEASE PREDICTION

## 2.1 DESCRIPTION OF THE DATASET

The dataset for this research was chosen by Kaggle website which was named as Heart Failure Prediction Dataset was created by using five different datasets are,

- Cleveland Dataset: 303 observations
- Hungarian Dataset: 294 observations
- Switzerland Dataset: 123 observations
- Long Beach Dataset: 200 observations
- Stalog (Heart) Dataset: 270 observations

By removing the duplicate values, the dataset includes 918 records, when download it from the kaggle. And this dataset doesn't include any person name or personal identification details so there is no ethical issue with using this data set for research purpose. The dataset contains 12 attributes are,

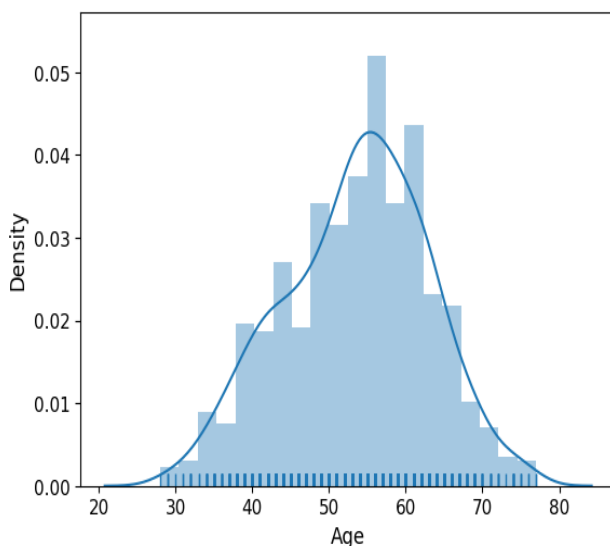
- **Age:** age of the patient in years, age is a continuous numerical variable.
- **Sex:** whether the patient is male or female, 'M' represents Male and 'F' represents Female, Sex is Nominal categorical variable.
- **ChestPainType:** the data set contains four different of chest pains such as TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain and ASY: Asymptomatic and chest pain type consider as ordinal categorical variable as it has natural order (Jarar Zaidi,2020).
- **RestingBP:** In the chosen dataset resting blood pressure measured by [mm Hg] and it contains numerical continuous values.
- **Cholesterol:** The cholesterol column includes serum cholesterol [mm/dl] (covers all level of cholesterol) level as numerical continuous values.
- **FastingBS:** fasting blood sugar level, [1: if FastingBS level > 120 mg/dl, or else:0] it represents with numerical value under two categories.
- **RestingECG:** represents the resting electrocardiogram outcomes [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria], and it identified as nominal categorical variable.
- **MaxHR:** represents achieved maximum heart rate of a patient [Numeric value laid between 60 and 202] as a numerical continuous variable.
- **ExerciseAngina:** exercise-induced angina [if patient have pain: Y, otherwise: No], A pain or an uncomfortable sense when blood flow to the heart is decrease. The variable values included as nominal categorical variable



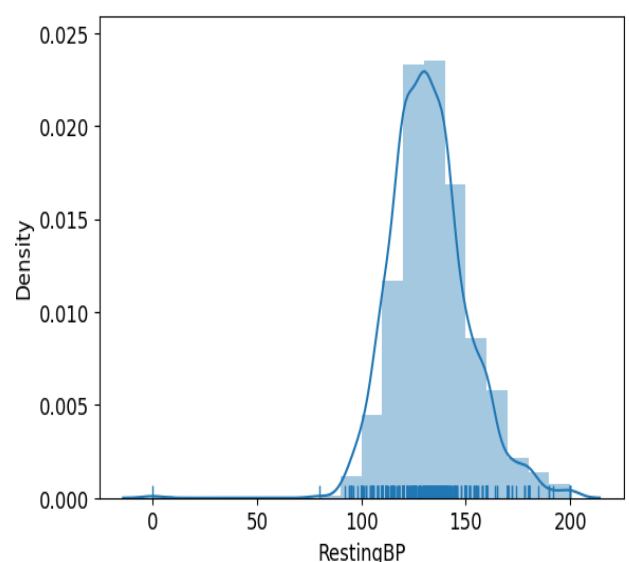
- **Oldpeak:** the ST depression included by exercise relative to rest it's a numerical measured value represented as numerical continuous variable (The ST-elevation myocardial infarction arises from blocking of one or more of coronary arteries which source the heart with blood)
- **ST\_Slope:** the slant of the increment which is related to exercise-included heart rate exercise [Up: upsloping, Flat: flat, Down: downsloping] and this is ordinal categorical variable
- **HeartDisease:** The dependent variable is Heart Disease, which includes details whether, the patient presence heart disease or absence heart disease include categorical data under two categories [1: heart disease, 0: Normal]

When considering the balance of dataset according to the dependent variable of the dataset it includes 508 heart disease presence cases and 410 heart disease absence cases, then it was considered as an approximately balanced dataset. In this dataset was identified 5 categorical independent variables and six numerical variables and dependent class with binary output values.

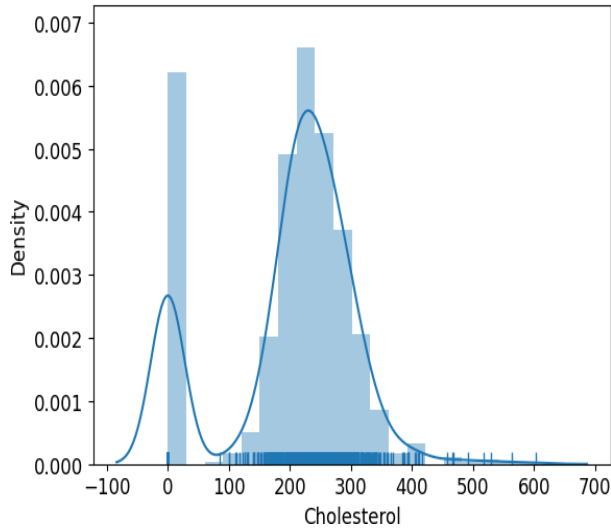
As a first step of the research to identify the distribution of the data, the data was visualized using matplotlib library, numerical variables were displayed in displot as in figure 1.0, in here identified that the fasting blood sugar contained values which data type was int64, but it includes only two categories because of that to identify the distribution of fasting blood sugar and other categorical variables were plotted data in pie charts.



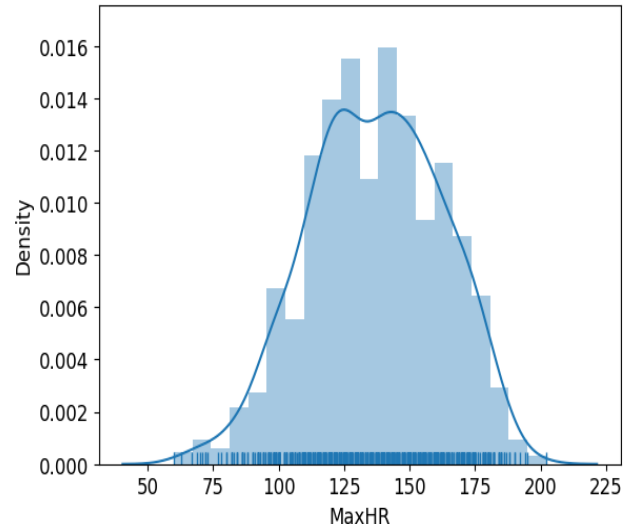
**Figure 1.1** Age vs. density of distribution



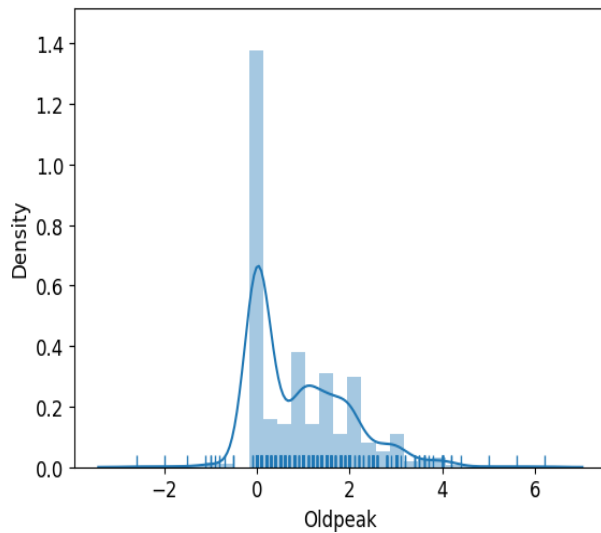
**Figure 1.2** Resting Blood Pressure vs. density of distribution



**Figure 1.3** Cholesterol vs. Density of distribution



**Figure 1.4** Maximum Heart Rate vs. Density of distribution

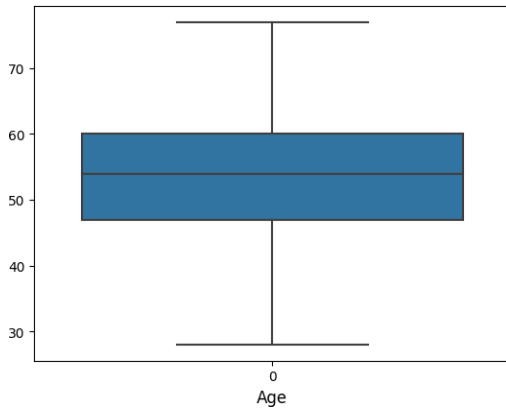


**Figure 1.5** Oldpeak vs. Density of distribution

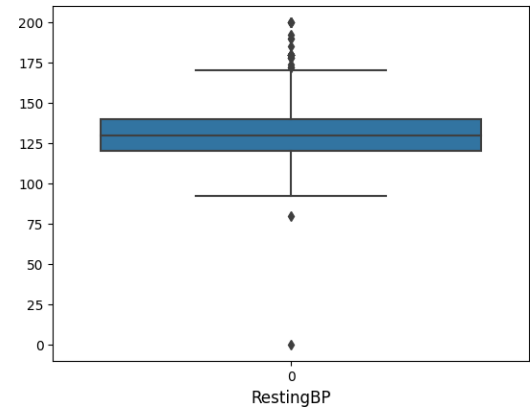
**Figure 1.0** The Visualization of the distribution of continuous variables.

The given figures describe that the distribution of the continuous numerical variables. According to the plots can be identified, the dataset contains data with human errors (includes cholesterol level as '0' which cannot be happening) and some range of misleading data points. Therefore, it requires extensive cleaning and preprocessing to ensure the usability and the reliability of dataset. And there, when consider the probability density function curves the continuous numerical variables show normal distribution curves approximately after removing the outliers.

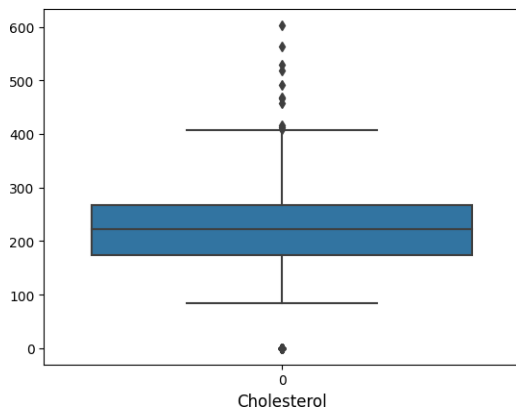
After identifying the data distribution as a next step, data is checked for detecting the outliers, for that purpose the boxplots are plotted as below figures (2.0).



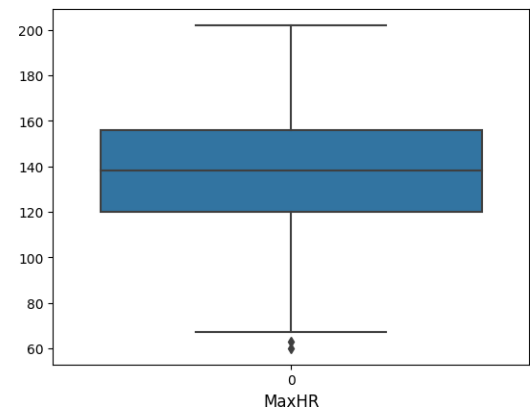
**Figure 2.1** Boxplot visualization for Age variable



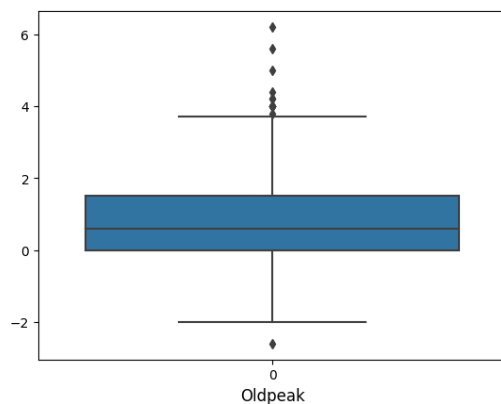
**Figure 2.2** Boxplot visualization for RestingBP variable



**Figure 2.3** Boxplot visualization for Cholesterol variable



**Figure 2.4** Boxplot visualization for Cholesterol variable



**Figure 2.5** Boxplot visualization for Oldpeak variable

And I identified that, data includes some outliers and need to do the preprocessing to clean the data to overcome the misleading problems while classification process.

**Figure 2.0** Visualize the numerical continuous data to identify the outliers properly

## 2.2 DATA PREPROCESSING

After visualizing the data in histograms and box plots, was identified that the outliers lay on the variables. Therefore, before implementing the models data need to be clean, for that purpose check for the null values and duplicate values. Furthermore, to scale the data use Min-Max scalar as the scientific report by Muhammad et al. (2020) Which achieved better accuracy levels in model predictions. Then for removing the outliers used percentile method.

- **Detect and remove outliers using percentile**

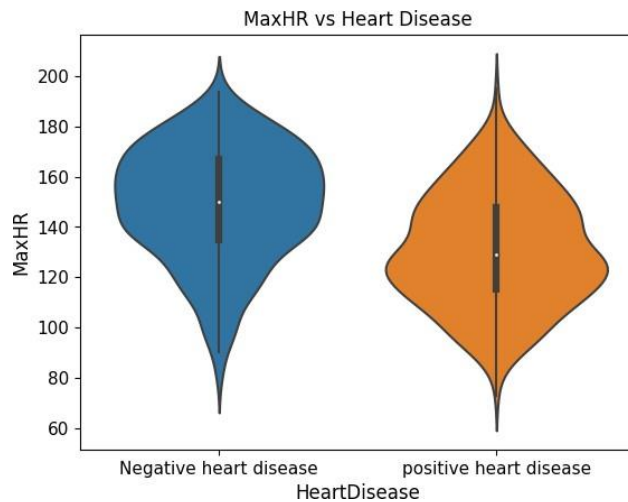
To detect the outliers, analyze on the distribution plots and gains idea about the ranges that the data laid on and referred National Health Service (NHS) website, to find the optimal levels of health indicators to set the minimum and maximum threshold values, to identify the outliers in a reasonable manner. From that method, the far end outliers are mainly detected and removed. And got the dataset with 717 data entries which is reasonable to the model implementations.

As a next step, the numerical continuous data was normalized using Min-Max scalar in sklearn library, Normalization is the procedure of transforming values of a number of variables into a similar range of scaling the variables, when scaling variable average is 0 and the variance is 1 then the value range in between 0 to 1. normalization is one of the most significant step in the machine learning model development process as it increases the performance and training stability while avoiding dimensionality reduction of models (Alshangiti et al. 2019).

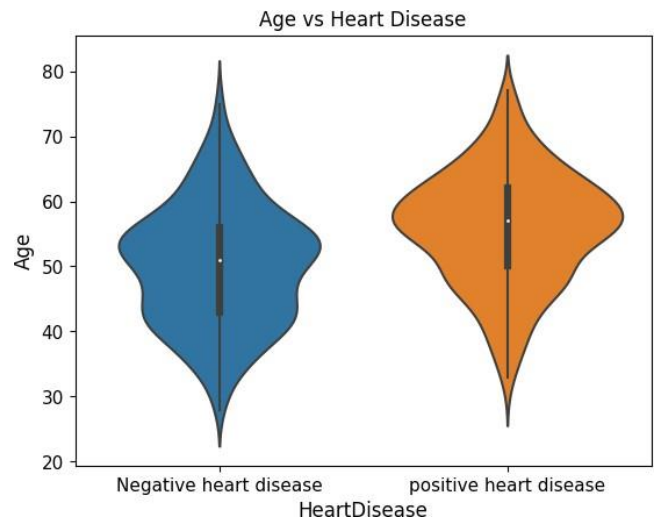
As a next step, the relationship between independent variables and dependent variable need to be identified. It is important step, because occasionally, the classification model performance reduces due to unrelated variables in the dataset. The feature selection method expands the performance of classification metrics, and furthermore it helps to decrease the execution time of the algorithms. Muhammad et al. (2020) for select the most important features of the heart disease used, Data Visualization, co-relation analysis and logistic regression coefficient (Feature importance score) analysis. To implement the object variables with these analyses I labeled data into numerical data using encoding techniques. To convert ordinal categorical variables to numerical, the pandas dummy variable method have been used and to convert nominal categorical variables to numerical, sklearn library label encoding method has been used.

## 2.3 FEATURE SELECTION

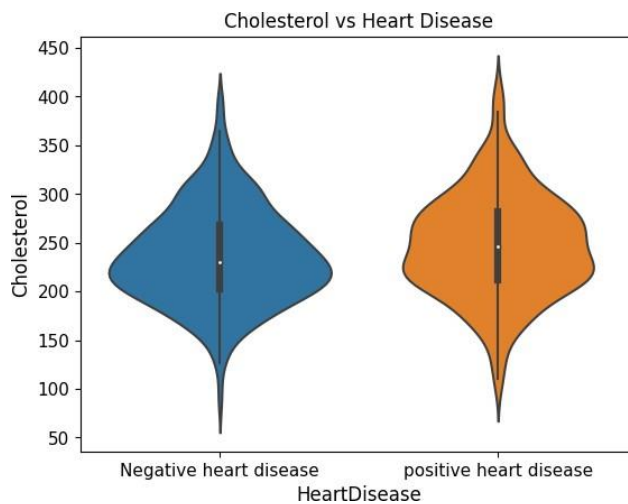
To get the initial idea of the relationship between dependent and independent variables data was visualized using violin plots



**Figure 3.1** Comparison of the density of negative and positive heart cases accorded with the levels of Maximum Heart Rate



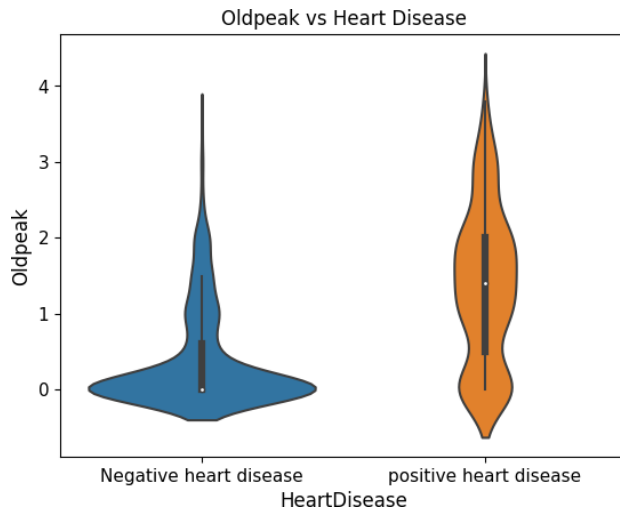
**Figure 3.2** Comparison of the density of negative and positive heart cases according to the different Age



**Figure 3.3** comparison of the density of negative and positive heart cases according to the levels Cholesterol



**Figure 3.4** comparison of the density of negative And positive heart cases according to the levels RestingBP

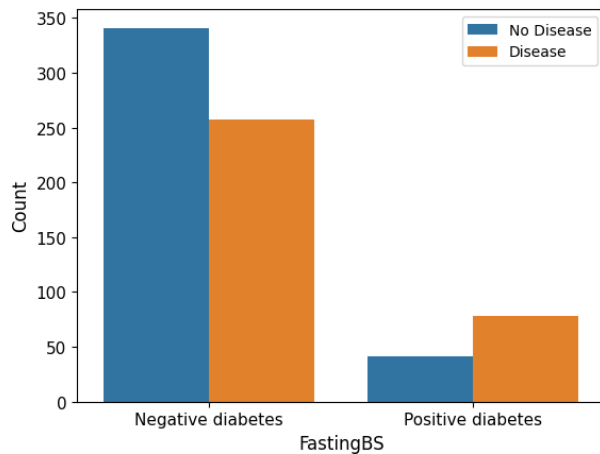


**Figure 3.0** The visualization for identifying the relationship between a target variable and numerical continuous independent variables, data plotted using a violin plot

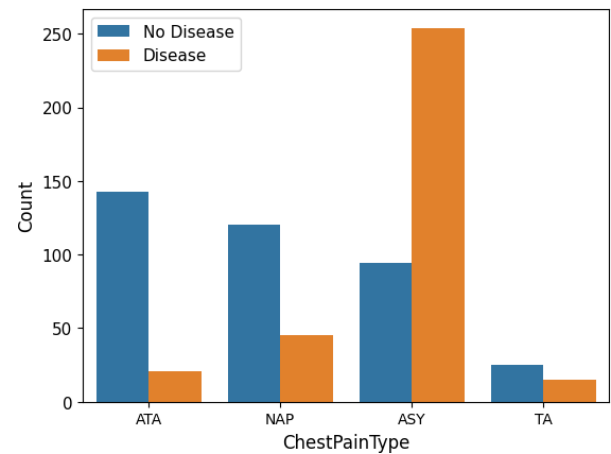
**Figure 3.5** Comparison of the density of negative and positive heart cases according to the ranges of Oldpeak

**Analysis of numerical Data - Age:** Having heart disease risk worsens with age and shows positive correlation and positive mean around 55-60 and Negative mean around 45-50, **Cholesterol:** Cannot identify clear correlation on violin plot, **Max Heart Rate:** It shows a negative correlation with heart disease when maximum heart rate is low and below 120, the risk of being positive in heart diseases getting high, **Old Peak:** Old Peak is around '0' there is no risk of having heart disease and it is near to '2' the positivity of getting heart disease is high, and it shows the positive correlation with heart disease.

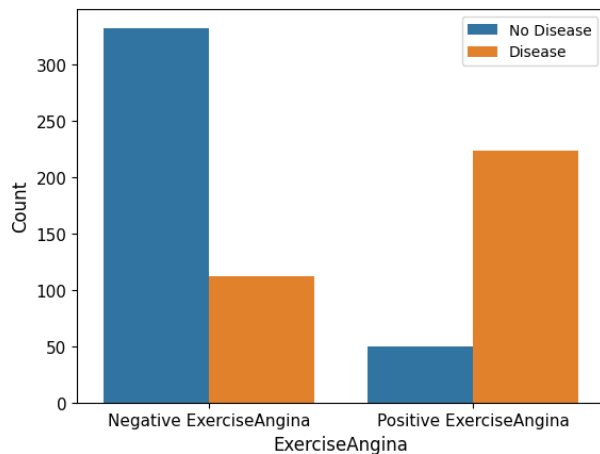
**Analysis of Categorical Data - Chest Pain type:** we can identify the 'ASY'- Asymptomatic chest pain type is more affected type in positive heart disease if patient has Asymptomatic chest pain he has high risk than others, **Exercise Angina:** If the patient has exercise angina the probability of having heart disease is high, **Blood Sugar:** If patient has high blood sugar the probability of having heart disease is higher than non-diabetics patient, **Resting ECG:** If patient has ST or LHV, ECG report he has high probability of having heart diseases than patient who has Normal ECG report, **Sex:** Males have high probability of having heart diseases then female, **ST Slope:** If patient has Flat ST Slope the chance of positive heart Disease patient is higher than Up and Down as in figure 4



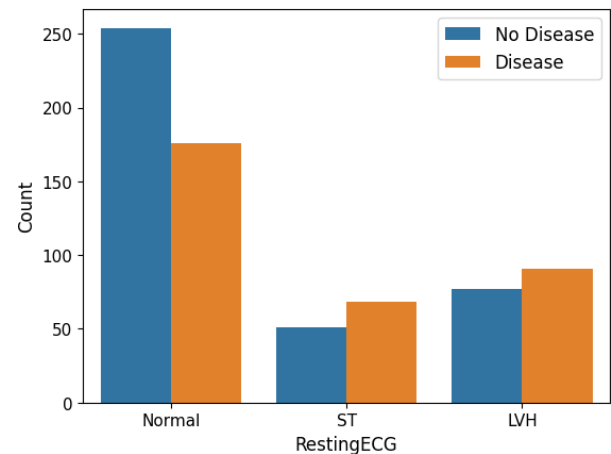
**Figure 4.1** Comparison between negative and positive heart cases according to the fasting blood sugar



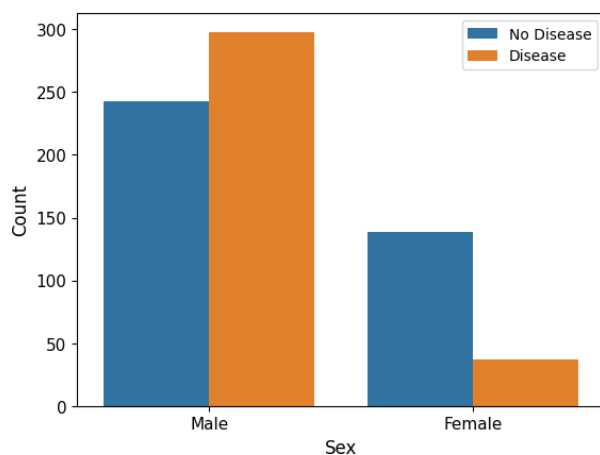
**Figure 4.2** Comparison between negative and positive heart cases according to the Chest Pain Type



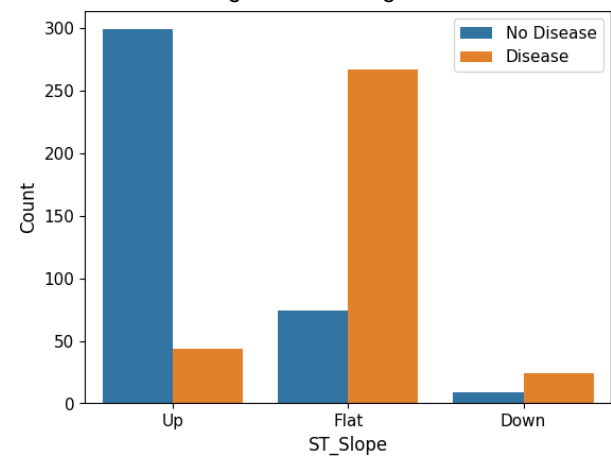
**Figure 4.3** Comparison between negative and positive heart cases according to the Exercise Angina



**Figure 4.4** Comparison between negative and positive heart cases according to the Resting ECG



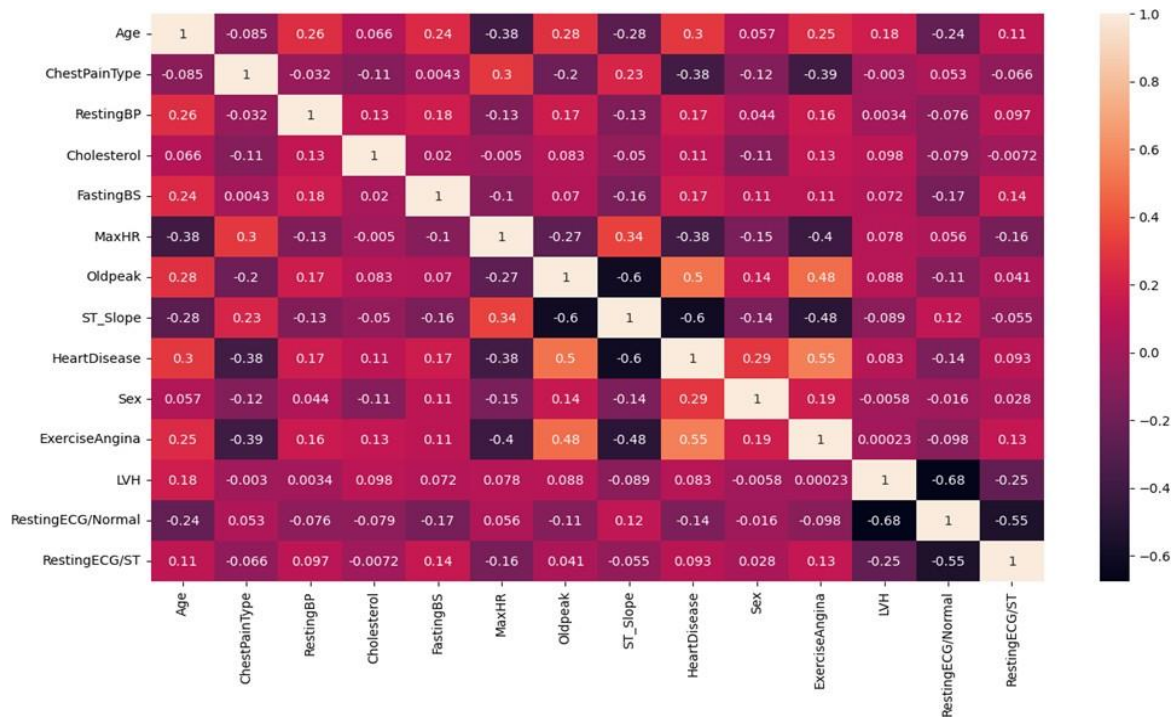
**Figure (4.5)** Comparison between negative and positive heart cases according to the Sex



**Figure (4.6)** Comparison between negative and positive heart cases according to the ST\_Slope

**Figure 4.0** The visualization for identifying the relationship between a dependent variable and categorical variables, data plotted using a counter plot

From visualization technique I was unable to identify the clear relationship among independent continuous numerical variables and dependent variables, therefore, the correlation analysis was done and plotted as in Figure 5.0.



**Figure 5.0** The Correlation between independent variables and dependent variable

The approach of feature selection, the co-relation analysis was done but unable to detect a clear liner relationship among most independent and dependent variables, the relationship magnitude and the sign (+ or -) not match with the domain knowledge as example Cholesterol vs. heart disease show weak negative relationship. In my dataset the target variable contains the binary data. As the result of that it was difficult to identify the linear correlation among data and maybe the dataset was small and the data related to the patient may not influence by health indicators maybe they are smokers or high alcohol addicted persons.

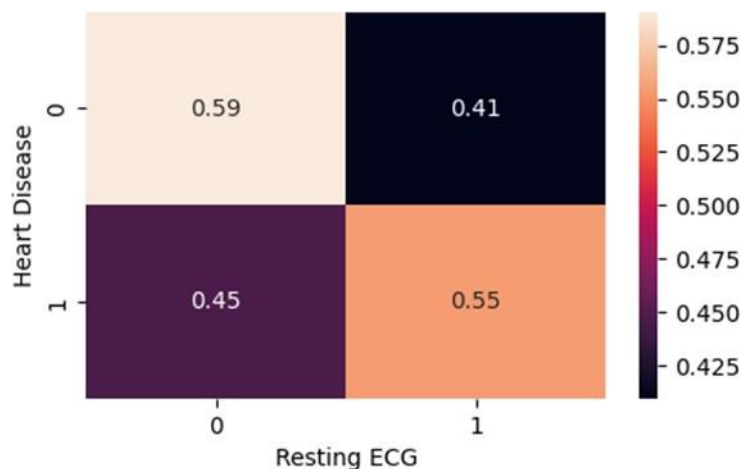
Moreover, analysis the logistic regression coefficient to Improve model performance in heart disease prediction by doing more reasonable feature selection using logistic regression (Bashir et al. 2019). When using logistic regression for feature selection we got coefficient scores as in Table 1.0.



No of the feature	Name of the feature	Importance score
01	ST_Slope	1.982892
02	Sex	1.552141
03	ExerciseAngina	1.218809
04	Age	1.129196
05	RestingBP	1.070920
06	Cholesterol	1.043643
07	Oldpeak	1.004430
08	ChestPainType	0.686745
09	FastingBS	0.390128
10	MaxHR	0.328329
11	RestingECG	0.259101

**Table 1.0** The feature importance score according to the logistic regression coefficient analysis

In logistic regression feature importance score shows the high coefficient score in ST Slope, Sex etc. which is matching with our general knowledge on fact related to heart disease. But the Resting ECG still shows the low relationship with dependent variable for further analysis I grouped the Resting ECG data into two main categories as abnormal and normal to simplify the process to identify the relationship and check the correlation again, but it doesn't show the clear correlation with dependent classes as in figure 6.0.



From the result analysis I have assumed that the records of the patients not related with resting ECG may be patients caused to affect heart disease due to other medical conditions, lifestyle factors, or genetic influences

**Figure 6.0** The correlation among Resting ECG and Heart Disease with respect to each category

# **CHAPTER THREE**

## **TRAINED ALGORITHMS TO ANALYSIS THE PERFORMANCE**

# TRAINED ALGORITHMS TO ANALYSIS THE PERFORMANCE

## 3.1 SELECTED ALGORITHMS TO TRAIN THE DATA

The main aim of this research is early detect and predict the heart disease through machine learning and help to save millions of lives. The early identification, helps to treat the patient in the early stages and support to change their lifestyles in a healthy manner. For this analysis, I used heart diseases dataset with binary target variable, so here I used binary classification algorithms, Logistic Regression and Decision Tree as supervised machine learning algorithms and Artificial Neural Network as deep learning algorithm.

## 3.2 LOGISTIC REGRESSION

Logistic regression models consider as statistical models in which an assessment is made of the relationship based on logistic function in the other word sigmoid function, the formula for the sigmoid function as below,

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

$$p(x) = \sigma(t) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

In logistic regression algorithm that models the probability of an event occurring with the linear relationship, with one or more independent variables by having log-odds, it calculates the coefficient in the linear combinations ('Logistic Regression', 2019). And the function need to predict probability so by applying the exponent on log-odds the sigmoid function was derived as it is used to predict the probability of classification problems (Saini, 2021).

The applications and combination of logistic regression and machine learning is famous today in different fields specially in medical fields, economic fields and social sciences. As instance, the total score of Trauma and Injury Severity is one of the most applied application to predict deaths in injured patients. Moreover, the use of logistic regression can be identified in predicting the risk of developing a given disease such as diabetes and coronary heart disease ('Logistic Regression', 2019). furthermore, when the target variable contains qualitative data such as spam or not spam, it is not applicable for the algorithm linear regression, and simple or multinomial logistic regression is suitable as applicable to the number of classes is included in the target variable (Domínguez-Almendros, Benítez-Parejo and Gonzalez-Ramirez, 2011).

In this research, I used most influencing 8 features to train the logistic regression model which is detected through logistic regression coefficient analyzing and split the data for training purpose 70% and 30% of data for testing purpose.

### 3.3 DECISION TREE

Decision Tree algorithm define as a simple classification algorithm which is commonly used in health sector to handle datasets as it able to implement with both numerical and categorical data. Mostly, three types of nodes can be identified in this tree-shape graph such as Root node, Interior node and a leaf node. The decision tree algorithm divides the data into two or more comparable parts based on the most significant indicators. The entropy of individual attribute is calculated and then the data are split, with indicators having highest information gain or lowest entropy (Shah, Patel, and Bharti, 2020). The formulas for entropy and information gain as follows,

$$\text{Entropy}(S) = -\sum_{i=1}^c P_i \log_2 P_i,$$

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

In sometimes decision tree shows lower accuracy level compared to the regression models as the number of nodes are imbalanced and it caused to overfitting with predictions (Singh and Kumar, 2020).

In this analysis, I used three different decision tree approaches to train and test the data and to generate the predictions:

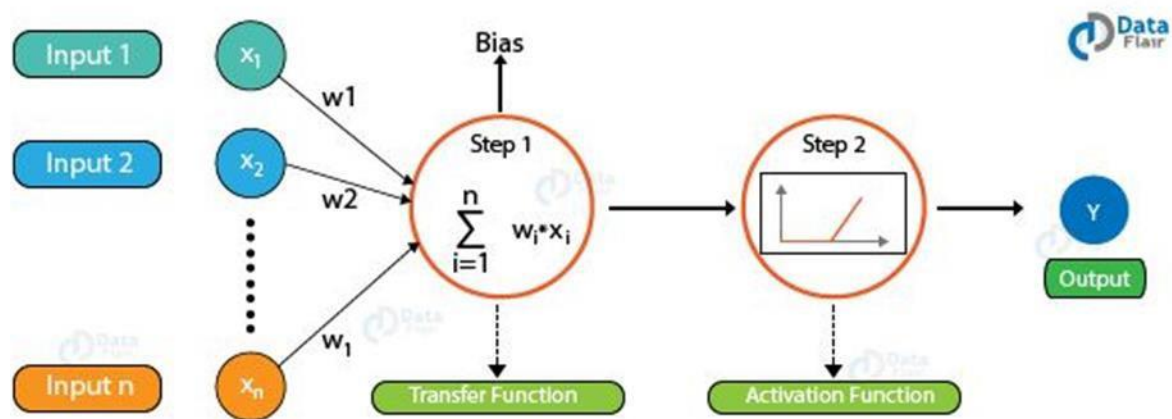
- a) Train a decision tree model using numerical independent variables.
- b) Train a decision tree algorithm using categorical variables by setting ranges using sklearn KBinsDiscretizer.
- c) Train a decision tree algorithm using categorical variables by setting ranges (discretizes) manually using domain knowledge which found by google.

### 3.4 ARTIFICIAL NEURAL NETWORK

Artificial Neural Networks (ANN) are a combination of algorithms which are capable of identify the patterns. Mainly, the structure contains three types of layers, first one is the input layer, the place which provides the training features to the network, the hidden layer, the neural network can have structured with more than one hidden layer the actual processing is going on that layer with the support of weighted connection, every hidden layer examines the output from the earlier layer, processes it more, and passes it on to the subsequent layer. The final layer is named as output layer which is connected to the hidden layer, the main aim of a deep learning model was to generate the hypothesis which is able to generate by reducing the error in training occasions. The hypothesis is expressed using forward propagation. The input is specified to the neurons which executes some operation to produce the output this procedure is called an activation function. The activation function defines the output of a node, mainly the sigmoid activation function present at the output layer (Ramprakash et al. 2020). The Formula of sigmoid function as below,

$$f(x) = \frac{1}{1 + e^{-x}}$$

(Sigmoid Function which is called as activation function of output layer in binary classification)



**Figure 7.0** The figure illustrates how the ANN work internally, what is the process working in a neuron, as it shows the input data process with weights and calculation happens in a first step of inside neuron and then the result goes though the activation function and give the output as probability('Data Flair,' (no date)).

Furthermore, Artificial neural networks are capable of detect and display the nonlinear statistical complex relationship between input and output data and notice new patterns in a data set to give the higher accuracy in predictions in different tasks like image and speech recognition and medical diagnosis ('Data Flair,' (no date)).

We develop a deep learning approach to predict the probability of risk of a patient to presence or absent the heart disease, in here we develop a simple artificial neural network with input layer which includes 128 neurons, and two hidden layer with 64 and 32 neurons and output layer with 1 neuron and with sigmoid activation function. Moreover, dropout layers are included to overcome the overfitting problem.

### 3.5 EVALUATION PROCESSED USED

For the model evaluation we used confusion matrix which includes important statistical computational result such as accuracy score, precision, recall and F1 score, as we know there are four major parameters can be identified,

- True Positive (TP)
- False Positive (FP)
- False Negative (FN)
- True Negative (TN)

Accuracy score is used for assess the model performance, which can be defined as True Positive count plus, True Negative count divided by the true positive count plus, true negative count plus, false positive and false negative count the formula is,

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

Moreover, sensitivity/recall is used for evaluate the model, which the proportion of true positive cases got predicted as true positive. it can be explained as number of unhealthy persons got predicted as unhealthy the formula is,

$$\text{sensitivity} = \frac{TP}{TP + FN}.$$

And the precision is the consideration of positive instances, it's the ratio of true positive cases towards the all positive occurrences which predicted as positive the formula is,

$$\text{Precision} = \frac{TP}{TP + FP}$$

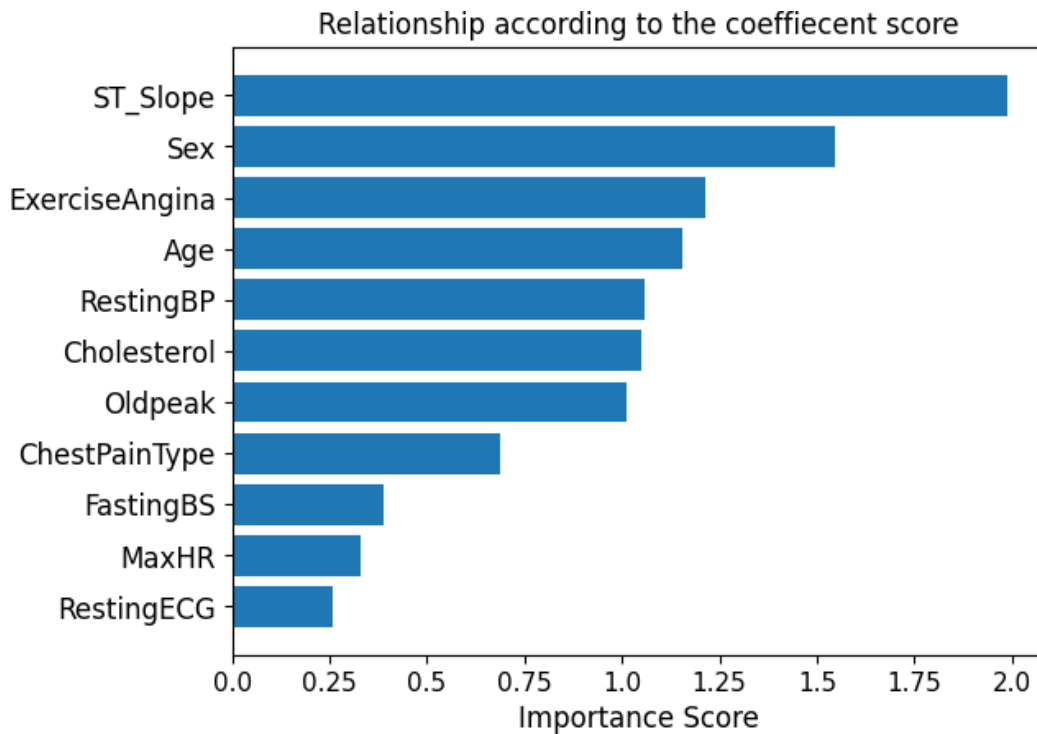
# **CHAPTER FOUR**

## **ANALYSIS OF RESULT REACHED THROUGH TRAINED MODELS**

## RESULTS AND ANALYSIS

### 4.1 FEATURE SELECTION USING IMPORTANCE SCORE

Machine learning is the most common approach in detecting diseases of patients in the health care sector. But in many researches I have identified that they use over 11 features to develop the models, when the number of indicators is high, the cost and spending time is also high as a solution for this in here I tried to recognize the most effective indicators to occur the coronary diseases. After that train the model using selected minimum features to achieve better accuracy, for that purpose I used visualization techniques, co-relation analysis using sklearn library and logistic regression coefficient analysis and reached the best reasonable output from logistic regression importance score analysis as below figure 8.0.



**Figure 8.0** The importance score visualization for each independent variable according to the Logistic Regression Coefficient



## 4.2 EARLY HEART DISEASE PREDICTION THROUGH LOGISTIC REGRESSION

In the dataset which I used to train the models, I have binary values as target outcomes, Therefore, I have used classification algorithms, to develop the model. Here, I used binary classification method, as my dependent variable includes 0 and 1 binary values. In related works such as Bharti et al. (2021) and Deshmukh. (2020), have achieved high accuracy in predictions, as 83% and 87% respectively by using logistic regression as the training algorithm, Specially, I can identify the feature of the logistic regression algorithm which gives the probability score of a case, so it helps to predict the probability of risk to presence from heart disease of a patient early and able to take the necessary actions to save lives. By considering all factors, I also choose the logistic regression with my implementation.

For identify the most effective and efficient method I train the algorithm under three different approaches with slight modifications as follows,

- **Train the Model without consider on Outliers**

When train the model without considering on the distribution of outlier, I got training accuracy as 84% and testing accuracy as 87%. But usually training accuracy should be higher than testing accuracy, so I can identify the model is not performing properly with the outliers. (The Zero values of cholesterol level were removed as considering the human error).

- **Train the Model after removing Outliers, with all independent variables**

As the second approach, the outliers were detected using the percentile method and remove the far end outliers and train the model with 717 data entries. And achieve training accuracy as 86% and testing accuracy as 84% by dodging the previous performance error. According to the behavior of models I can identify that, outlier detection and removal is a significant step in machine learning algorithm training process.

- **Train Model after removing Outliers and Feature Selection**

As third approach and most effective approach, the algorithm was tested with eight features with highest coefficient, against to the target variable which was identified through logistic regression feature selection. And I got training accuracy as 85% and testing accuracy as 84%.

And to choose the best parameters to implement the Logistic Regression model, I did the hyper-parameter tuning using GridSearchCV on sklearn library and acknowledge that the best parameters for this dataset training is default parameters in the logistic regression algorithm. As shown in the below figure 9.0,

```
param_grid = {'penalty' : ['l1','l2','None'],  
              'C' : [0.1,1,10],  
              'max_iter' : [100,1000,10000],  
              'solver':['lbfgs','libliner','newton_cholesky']}
```

```
grid_search.best_params_
```

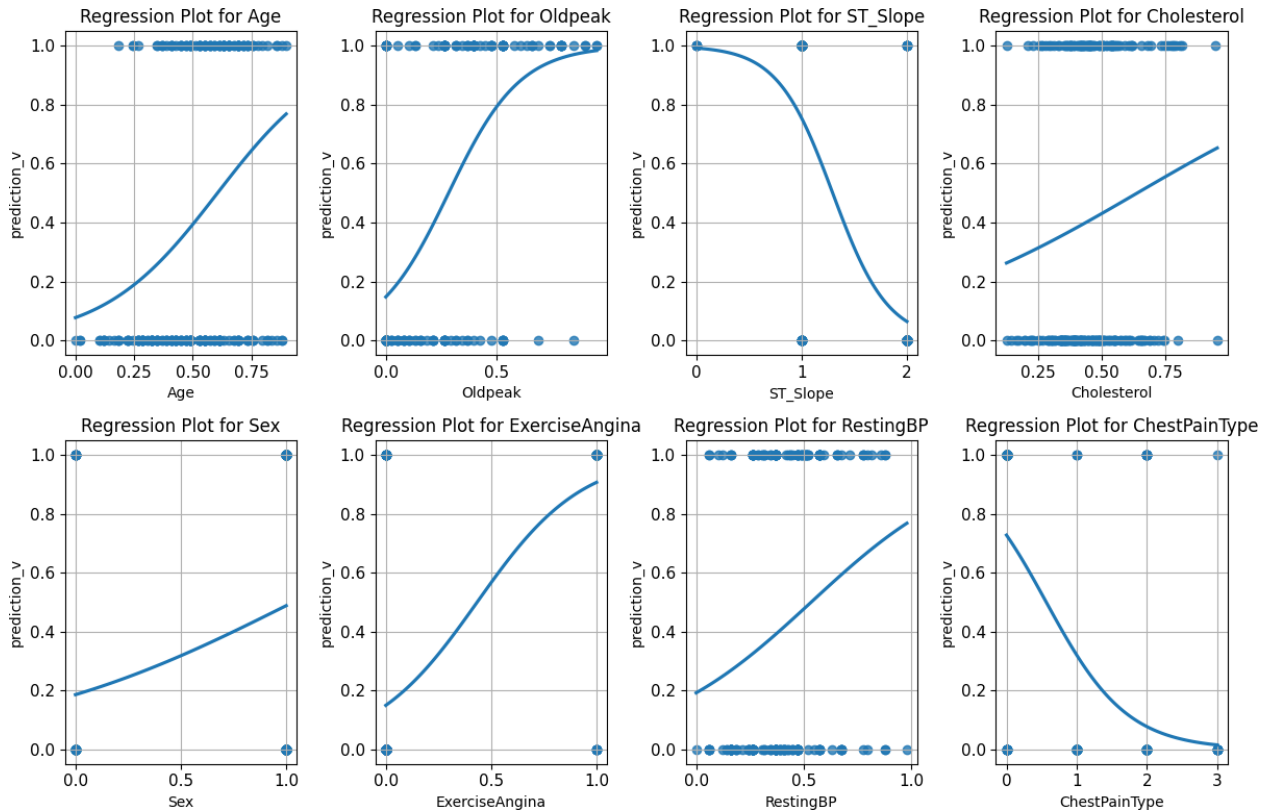
```
{'C': 1, 'max_iter': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
```

**Figure 9.0** The output of selecting best parameters to train the Logistic Regression model using GridSearchCV, the default parameters selected as good ones.

When comparing my approaches, which was developed on logistic regression that I'm able to acknowledge the third approach is performed well than 1st and 2nd approaches as it gives accuracy level, 84% with only eight features. My 2nd approach was also given the accuracy as 84% with 11 attributes. So my third approach is more cost and time effective model and performed well.

Moreover, I can identify that our logistic regression model was performed well by comparing to the previous researches in Bharti, (2021), they achieve 83% accuracy with 13 attributes and my model already shows higher accuracy than above model with 8 attributes. So my model is performing well with 8 attributes accountably as few independent variables means few examinations and it helps to save cost and time.

After identifying the best model of logistic regression I plotted logistic curves, between predicted probability and independent variable to evaluate model performance and to check the co-relation.



**Figure 10.0** Show the logistic curve with respect to each independent variable and prediction probability

As we all know the sigmoid function which was used in logistic regression gives the output between 0 and 1. From above illustration I tried to identify the relationship of independent variables and prediction probabilities, and I was able to identify that some variables, as example cholesterol don't show the good correlation with the target variable in the testing dataset as well.

For more understanding when consider Old peak variable in testing set, according to the domain knowledge, if the Old peak is higher than 0.15 it means the result of old peak is in high risk level, so when I check the values of actual and predicted which are respective to the value greater than 0.15(0.3 is the equal normalized value to the 0.14 actual value) of patients, I have identified that major of them got heart disease, as 59 of them got heart disease from 73 of patients set. In the old peak plot which represent the logistic curve according to the independent variable and dependent variable, when set threshold value as 0.5, I can acknowledge that patients who had old peak over 0.15(0.3) have the high probability of the presence of heart disease. So the I can deducible that the model is perform properly.

From this model, doctors are able recognize the risk level of the patient, who affect from heart disease before the critical level. when give the examination results according to the input parameters the model will give the probability of presence heart disease, as example if some patient got probability as 0.6 then he is in a risk to having heart disease than patients who has probability of model result 0.3, but the risk level is not much higher it means doctors may able to save the lives by taking the necessary actions. And also if the patient gets 0.9 as probability he is at high risk and have to give priority to treat the patient to save the live.

In figure (11.0), you can see how the model is performed with real life data,

	Age	ST_Slope	Sex	ExerciseAngina	RestingBP	Cholesterol	Oldpeak	ChestPainType	HeartDisease
0	0.244898	2	1	0	0.469388	0.629630	0.000000	1	0

```
[ ] input_data = (0.244898, 2, 1, 0, 0.469388, 0.629630, 0.000000, 1)
    input_data_as_numpy_array = np.asarray(input_data)
    input_data_resaped = input_data_as_numpy_array.reshape(1,-1)

prediction_prob = model_L.predict_proba(input_data_resaped)

print("Positive Probabilities Precentage:", positive_proba_p)
print("Negative Probabilities precentage:", negative_proba_p)

Positive Probabilities Precentage: [11.8957526]
Negative Probabilities precentage: [88.1042474]
```

**Figure 11.0** The above figure shows how the model performs with real life data, the set of data selected from testing data set and feed it as input data for the model and check the probability the positive probability was 11% then patient at low risk, the actual output is also '0' as in original data.

### **4.3 Heart disease early prediction through Decision Tree algorithm**

Decision tree algorithm is the simple and famously used machine learning algorithm. which is established on taking decisions based on gini index or entropy and evaluate and matches results of classification as graphical representation (Ali et al. 2021).

When analyzing the related work, I have acknowledged that simple machine learning algorithms are also successful in medical diagnoses as in the research, Ali et al. (2021) done the comparison on different algorithms and achieve 100% accuracy, sensitivity and specificity using decision tree algorithm. Furthermore, (Singh and Kumar, 2020) in their research, reach 79% accuracy based on decision tree algorithm.

The main target of this project is to, predict the heart disease early in cost and time effective manner, successful simple machine learning algorithms are interesting to analyze. Therefore, I used decision tree algorithm to implement and analyze the results on my project as well.

#### **4.3.1 Approach 01: Train a decision tree with numerical independent Variables**

- **Train the Model not consider on Outliers**

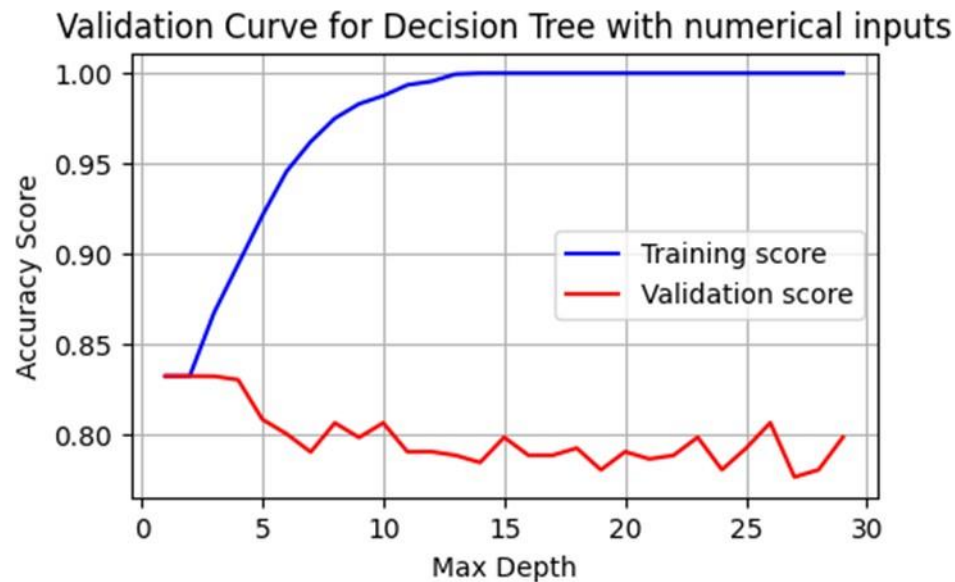
As a first step of a first approach on decision tree algorithm implementation I tried to train model with all numerical continuous data and categorical data as input data. without consider about the outliers. As a result, model have been reached 100% training accuracy and 75% testing accuracy. As indicated in result I can identified that the model is overfitting as it shows low bias and high variance and doesn't perform well, with comparing to the related work as the data is not preprocess properly.

- **Train the Model after removing Outliers with all independent variables**

To overcome the limitations in above method I tried to train the model after apply the preprocessing methods and removing the far end outliers using percentile outlier handling method. From this perspective I got training and testing accuracy scores as 100% and 78% respectively. The method is quite better than previous method as it gives testing accuracy as 78% however I can see this method is also facing overfitting problem as the complexity of model. When the model is much complex the decision tree models facing to the overfitting problem. So I can deduce that both outliers and the complexity of the input data may cause to the overfitting of the model.

- **Train Model after removing Outliers and Feature Selection**

To decrease the complexity of data which is used as input of the model, I tried to check the model performance using selected most important features, according to the logistic regression feature importance score and used 8 selected features. From this perspective the model gives 100% accuracy in training and 81% accuracy in testing set.



**Figure 12.0** The validation curve with respect to the maximum depth parameter of decision tree to identify whether the model is overfitting.

As represent in above figure (12.0) I can recognize that the decision tree model is overfitting, even the model with higher testing accuracy. So the decision tree algorithm training with continues numerical data, is not perform well in heart disease prediction with respect to the related works which was done before.

As a solution for this I tried to use categories data to train the decision tree model by reducing the complexity if input data. As an example here I tried to categories cholesterol level into three categories such as low, medium and high risk level which is simpler than making decision on large range of numerical values which was laid between 0 and 1 as they have been normalized.

### **4.3.2 Approach 02: Train a Decision Tree with Categorical Independent Variables (Data Discretize by Scikit-Learn, Kbinsdiscretize)**

The library scikit-learn, have special classes to import, which is discretize continuous numerical data into categories with equal wide intervals. In this research, as a one approach I use that class to set the data categories which introduce as KBinsDiscretize. Here, I set three bins according to the data by considering on risk as low, Medium and high (In ordinal manner 0,1 and 2) respectively.

- **Train the Model not consider on Outliers**

I have trained the model with all features without the consider on outliers, and all features in a categorical format in this approach I got the output scores as 100% in training and 81% in testing. So I identify that the model is still overfitting whether the continuous data, convert to the simple categories. Furthermore, data includes impurity data points.

- **Train the Model after removing Outliers with all independent variables**

Then I remove the outliers as mentioned previously using percentile method. Then train the model with categorical data, and reach 99% of training accuracy and 77% of testing accuracy in this approach also model is overfitting and the accuracy level is also decrease whether I remove the far end outliers. So I can deduce that the model is still too complex as decision tree algorithm gives the output with the effect of overfitting.

- **Train the Model after removing Outliers with all independent variables**

Then I tried to train the model using the most effecting features to the dependent variable. as the number of features, have been studiously when looking for the best split. In this examination I reach training and testing accuracy as 95% and 80% in respectively. In this approach I acknowledged that the difference between training accuracy and testing accuracy was reduced. But the model is performing with the overfitting.

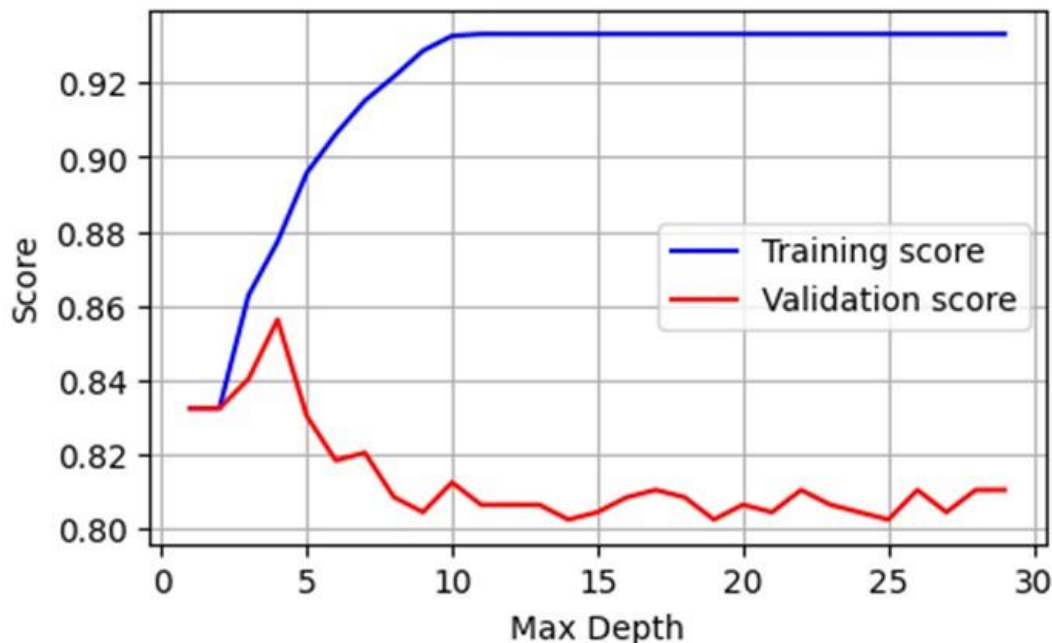
For reach the high accuracy level as related researches and to overcome the overfitting problem I tried to train the model with categorical independent variables which are discretized using domain knowledge. From above main two approaches I identified that the most appropriate way to train the algorithm is to train it with selected features and dataset with proper preprocess. Because of that I continue the third approach with only the selected method.

### 4.3.3 Approach 03: Train A Decision Tree with Categorical Independent Variables (Data Discretize Using Domain Knowledge)

- **Train Model after removing Outliers and Feature Selection**

In this approach, I got the output accuracy scores as 92% for training and 81% for testing when compare with my previous models, this model gives the high testing accuracy while reducing the gap between training and testing score. However, the model shows the overfitting problem with this approach as in the figure 13.0.

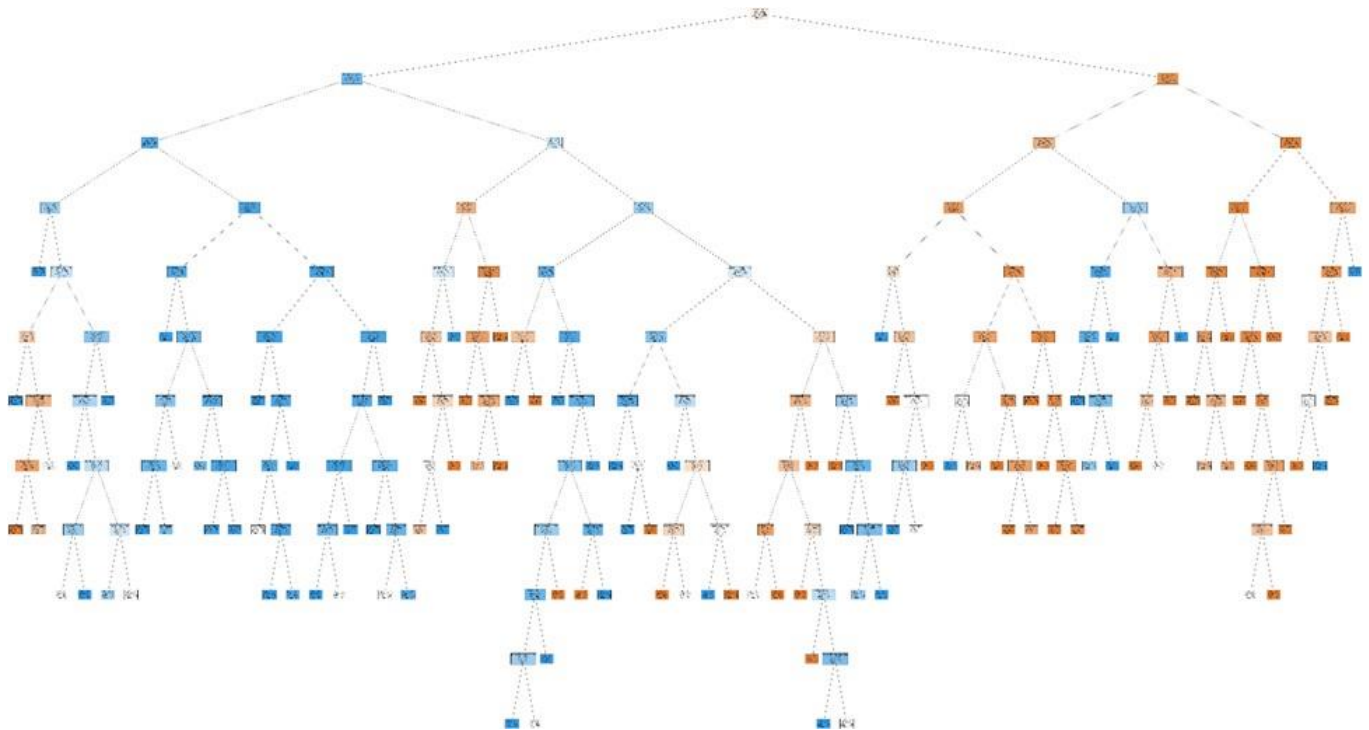
Validation Curve for Decision Tree with customized categories



**Figure 13.0** The validation curve visualization according to the Decision Tree approach 03

In figure 13.0, shows that the testing accuracy is also increase, up to some point with training accuracy and then start to show the decreasing trend. According to the Pawel, (2019), maximum depth is important factor which is effects to the overfitting of a decision tree as they mention if there is no mentioned any specific value for max\_depth parameter, the decision tree will carry on to split until it comes to the number of examinations in a given node equal to the min\_samples\_split parameter as 2 which is set by default, in complex sets they contain large number variables, because the detail of the model rises significantly. As a consequence, the model is almost perfectly matched to the training set on which it was created, and thus it is not able to generalize and use it on independent test sets. This phenomenon is called overfitting. To examine the problem, we can visualize the decision tree as in figure 14.0.





**Figure 14.0** *The Decision Tree model visualization how it trains with given data*

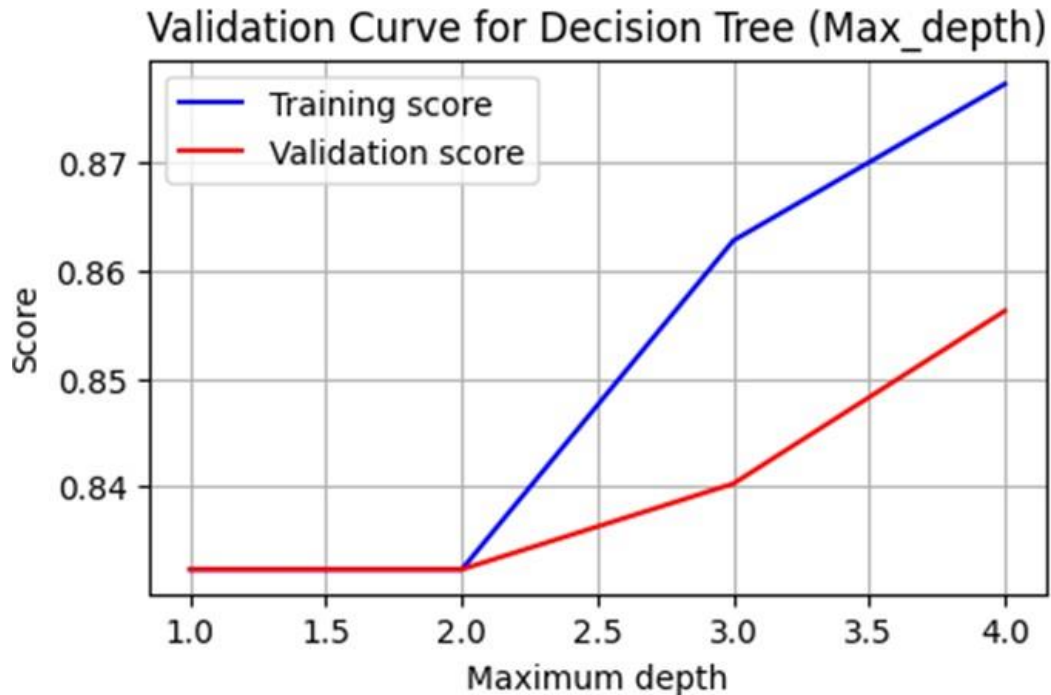
In above figure I have recognized that the tree is more complex and difficult to generalize and there are maybe some observations where the branch of a tree ends with one leaf which is useless. And the tree is unbalanced. In here I have tried to improve a model to predict the coronary disease early with new data. So I tried trimmed the tree in suitable length. For that purpose, I check the decision tree algorithm with range of values for max\_depth parameter as in table 2.0.

#### 4.3.4 Decision Tree Model Optimization

Depth of a tree = 1	Accuracy: 0.8009
Depth of a tree = 2	Accuracy: 0.8009
Depth of a tree = 3	Accuracy: 0.8519
Depth of a tree = 4	Accuracy: 0.8194
Depth of a tree = 5	Accuracy: 0.8287
Depth of a tree = 6	Accuracy: 0.8333
Depth of a tree = 7	Accuracy: 0.8148
Depth of a tree = 8	Accuracy: 0.8102
Depth of a tree = 9	Accuracy: 0.7963
Depth of a tree = 10	Accuracy: 0.8056
Depth of a tree = 11	Accuracy: 0.8009
Depth of a tree = 12	Accuracy: 0.8009
Depth of a tree = 13	Accuracy: 0.8009
Depth of a tree = 14	Accuracy: 0.8009
Depth of a tree = 15	Accuracy: 0.8009
Depth of a tree = 16	Accuracy: 0.8009
Depth of a tree = 17	Accuracy: 0.8009
Depth of a tree = 18	Accuracy: 0.8009
Depth of a tree = 19	Accuracy: 0.8009

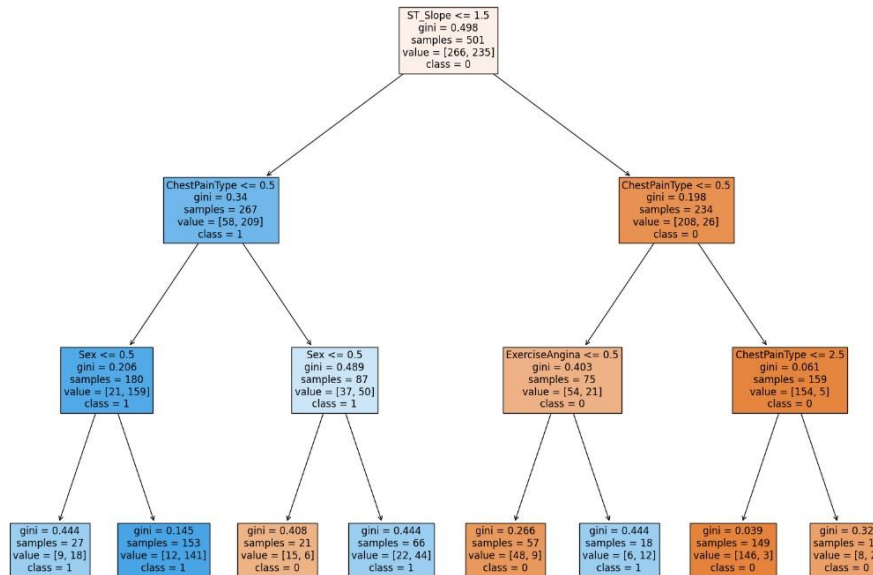
According to the table 2.0 it clearly shows after max\_depth of tree at 10 the accuracy of tree is not increase and after depth level 3 the accuracy shows up and downs and at length 9 it reached the minimum accuracy. At the max\_depth 3 the model extended the highest accuracy as 85%. Moreover I have trained the model with max\_depth parameter with 3 I got the training accuracy as 86% and testing accuracy as 85% and when visualized the validation curve, it shows that the overfitting error of the model is almost overcome from this modification as in figure 15.0.

**Table 2.0** The accuracy score of Decision Tree (3<sup>rd</sup> approach) with different tree depths



**Figure 15.0** The visualization of validation curve after define the max\_depth to 3

Furthermore, to have a clear idea I have illustrated the trimmed tree with limited the max\_depth parameter to 3, I can recognize simple balanced tree model using four independent variables as ST\_Slope, ChestPainType, Sex and ExerciseAngina for training and reach high training accuracy level as 85% by comparing to the previous decision tree models which were trained in this research.



**Figure 16.0** *The visualization of trimmed Decision Tree*

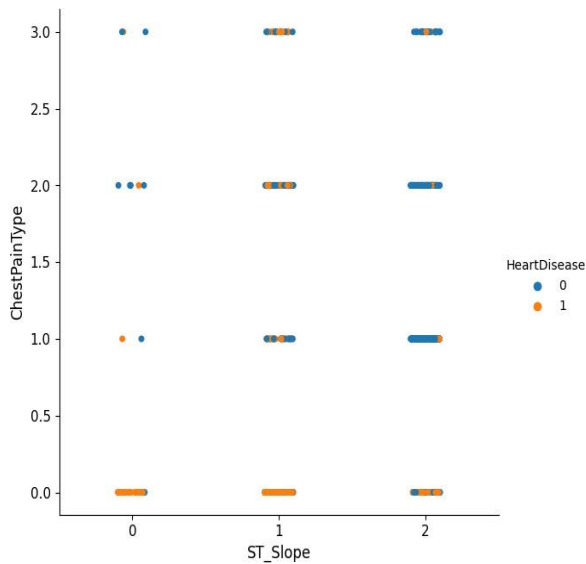
As I mention earlie decision tree model used four features to train until max\_depth 3,

- ST\_Slope
- ChestPainType
- Sex
- ExerciseAngina

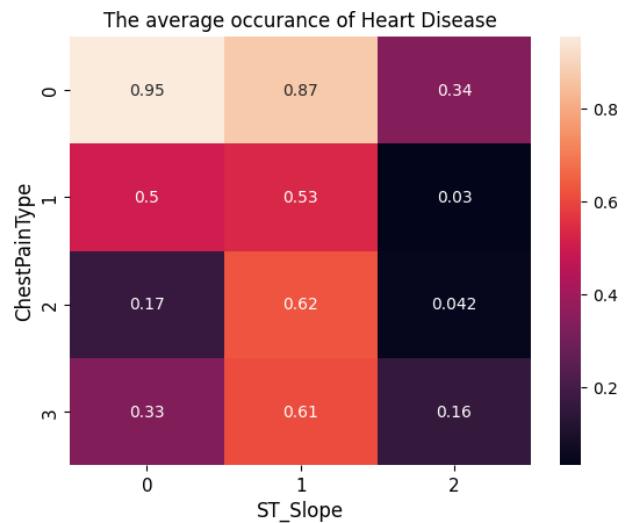
When I compared with the most effecting features according to the Logistic regression model

- ST\_Slope
- Sex
- ExerciseAngina
- Age

In logistic regression ChestPainType shows quite low correlation, but Decision Tree choose it as second splitting node to observe the matter, I plotted the ChestPainType vs ST\_Slope as in figure 17.0.



**Figure 17.1** The distribution of positive and negative heart disease with respect to the Chest pain type and ST Slope

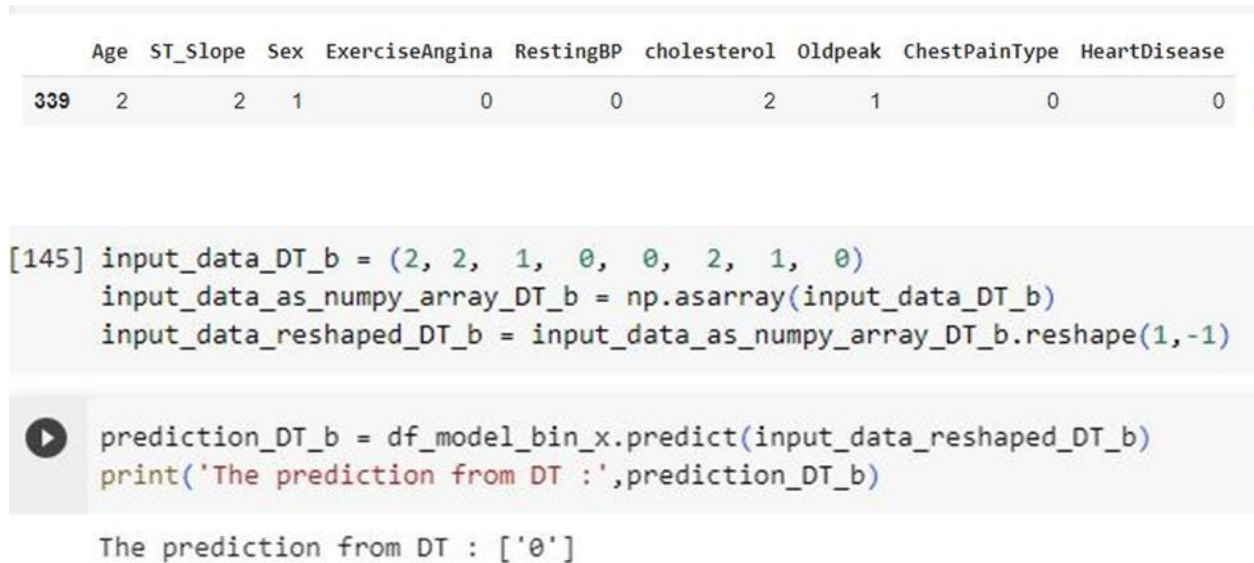


**Figure 17.2** The average of heart disease occurrence with respect to the Chest Pain Type and ST Slope

**Figure 17.0** The relationship between ST Slope and Chest Pain Type with respect to the occurrence of Heart Disease

According to the above figures I can identify that, when split data from ST\_Slope ( $\leq 1.5$ ) the majority of patients shows absence of heart disease with respect to the different ChestPainType in class 2 (ST\_Slope). So after split data using ST\_Slope ChestPainType shows good relationship to set it as a second splitting node. But in logistic regression when using all independent variables once, it may not show very good co-relation.

The finalist decision tree model is also gives better predictions with high accuracy with real life data as below figure 18.0.



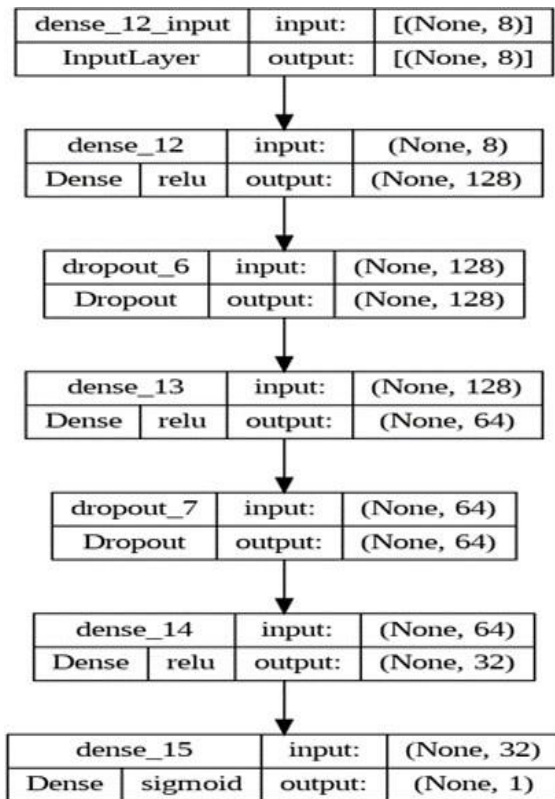
**Figure 18.0** Visualize how the decision tree model predict the Heart Disease with real life data

When comparing with, Singh,(2020) research which gain 79% testing accuracy my decision tree model has reached higher accuracy than the related research in medical predictions accuracy is one of the most important factor so my decision tree model is perform well than the mentioned related research, in that paper they are not mention that they have use method to normalize data and steps to overcome the overfitting so I think it may cause to lower their accuracy score with related to my trained model.

However, my model reached low accuracy with respect to the research Ali.(2021), when analyze that journal I identified that they trained the model with 1025 patient records and they used different algorithms to select the best features. they select best features by comparing feature importance calculations with different algorithms but in my implementation I used only logistic regression model to calculate the feature importance so I deduce that they are more accurate in feature selection with comparing with my feature selection approach and they have more data to train as they have more records so it may cause to they achieve high accuracy than my trained decision tree model.

## 4.4 EARLY PREDICTION OF HEART DISEASE THROUGH ARTIFICIAL NEURAL NETWORK (ANN)

In today health care sector not only machine learning but also deep learning done a huge service with medical predictions and detections. In the research article Bharti et al. (2021), have reached 94% of accuracy by using deep learning approach and which was highest accuracy they have achieve through different machine learning and deep learning models. Because of that I also interested to train my data with artificial neural network and to analyze the results.



For train the ANN to predict the probability of risk to occur the heart disease early, I have model a simple ANN with 4 layers, input layer with 128 nodes and two hidden dense layers with 64 and 32 neurons respectively with relu activation function, to handle the vanishing gradient problem while training process. and output layer with 1 neuron and set the activation function as sigmoid as we expect to take an output probability between 0 and 1 as shown in figure (19.0).

**Figure 19.0** The Visualization of structure of train ANN

- **Train the Model without consider on Outliers**

As the first approach of the ANN model training, I have trained the model with all recodes without removing the outliers. In this approach I have reached 90% and 86% of training and testing accuracy consequently. With this output result I deduce that ANN are able to identify hidden patterns in the complex and impurity datasets.



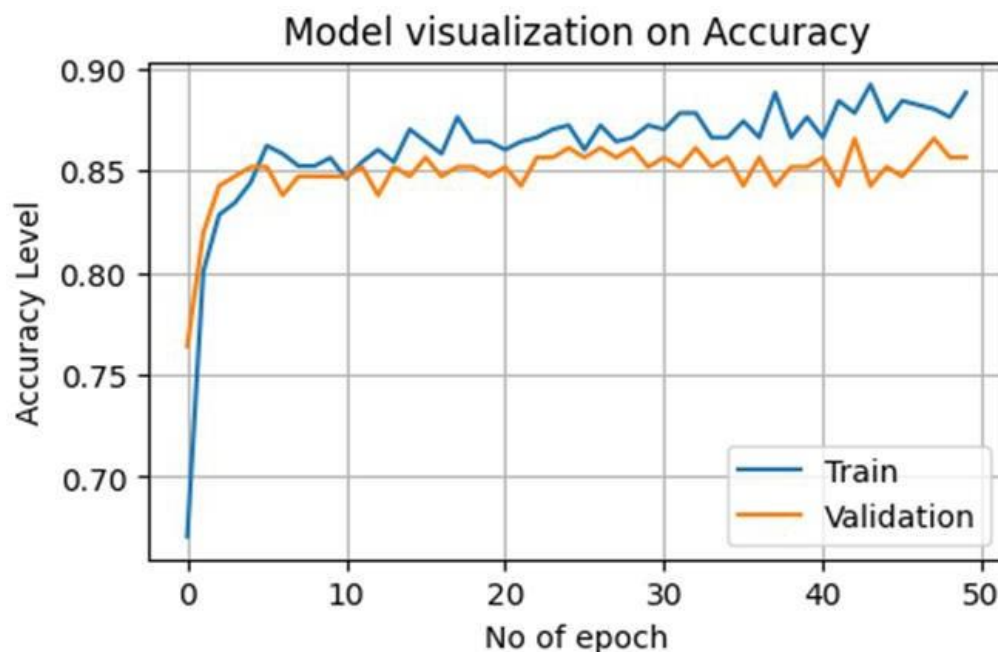
- **Train the Model after removing Outliers with all independent variables**

To analyze whether the model perform better without outliers, I have trained the ANN model after removing the outliers because if that the number of recodes are decrease than previous model training and I have reached training accuracy as 91% and testing accuracy as 81%. This output is slightly difference with respect to the logistic regression and Decision tree model the accuracy was decrease with the removal of outliers as I assume that in this approach the main reason to decrease the accuracy is the decline of records which is using with training and testing process. As we know one of the drawback of the deep leaning approaches are they need large amount of data to perform more accurately.

- **Train Model after removing Outliers and Feature Selection**

To recognized the effect of feature selection, I have trained the ANN network with selected features of preprocessed dataset and achieve 87% training accuracy and 86% of testing accuracy. so I can acknowledge that ANN performs better with small datasets with most effective features as slightly similar to the performance with large datasets.

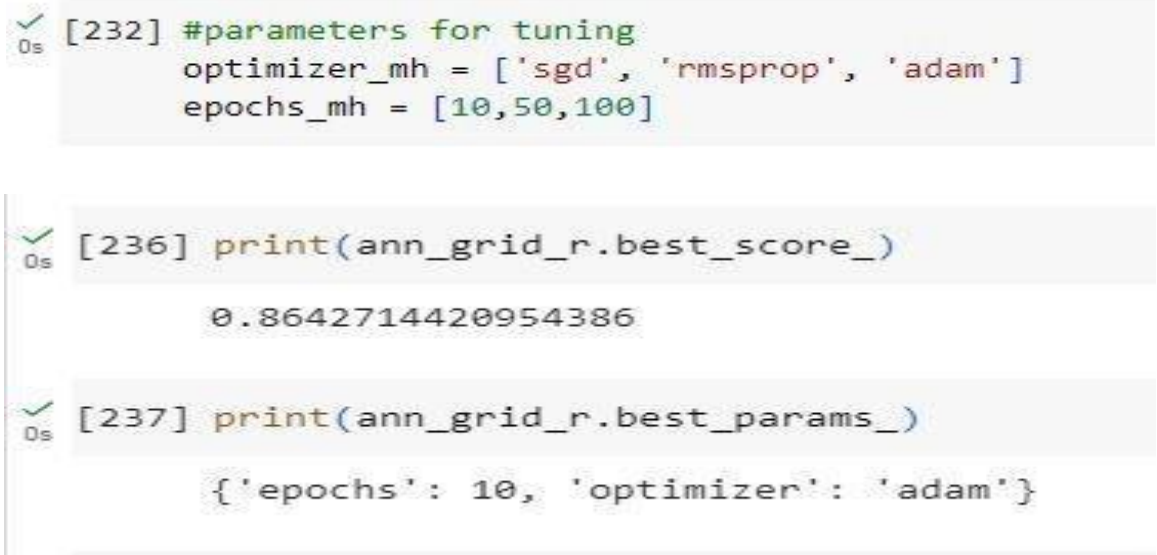
In my ANN model I have used dropout layers to overcome the overfitting problem and when visualized the validation accuracy and training accuracy can be identified that the model is train properly without showing overfitting or underfitting problem as figure (20.0).



**Figure 20.0** Visualization of Validation curve of ANN model according to the change of number of Epochs

- **Hyper-parameter Tuning for ANN model**

To find the best parameters for the ANN model I have tried to do the hyper-parameter tuning through changing the Optimizer and the number of Epochs, and use GridSearchCV method to find the best parameters with tensor flow, and I got the accuracy score and best parameters as below figure 21.0



```
[232] #parameters for tuning
      optimizer_mh = ['sgd', 'rmsprop', 'adam']
      epochs_mh = [10,50,100]

[236] print(ann_grid_r.best_score_)
      0.8642714420954386

[237] print(ann_grid_r.best_params_)
      {'epochs': 10, 'optimizer': 'adam'}
```

**Figure 21.0** Output result of hyperparameter Tuning through GridSearchCV in ANN

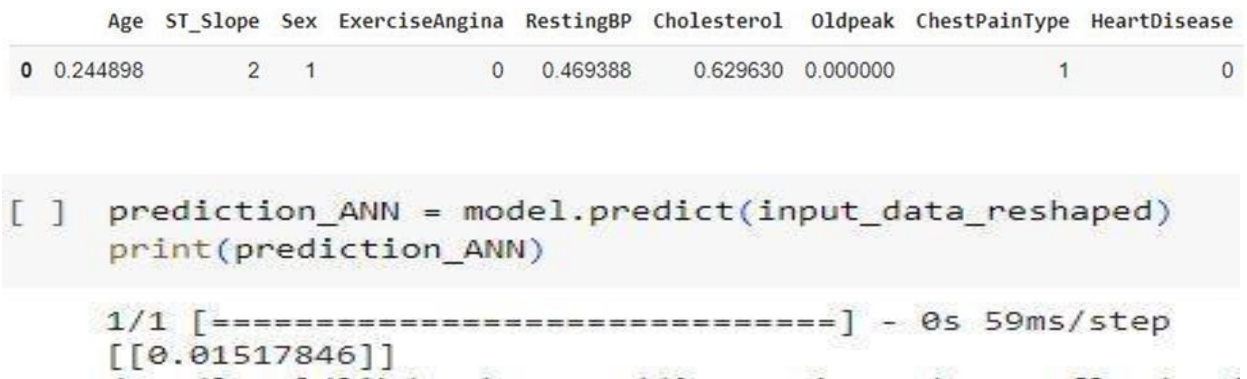
Through the best parameters, which was choose from the GridSearchCV gives the nearly same accuracy level to the model which I train as initial ANN model.

The above research paper I have referred, achieve 94% of accuracy with training the model using ANN and I achieve 86% accuracy with eight selected features. With comparing with Bharti et al. (2021) my model is performing quite low accuracy. When analyzing the methods, which they have used to implement their model, I recognized that they used isolation forest method to detect and remove the outliers and they used LASSO algorithm to perform feature selection as they mention in the paper LASSO algorithm shows better predictive accuracy than other methods and give fine feature subsets for the used algorithm. So I deduce that this LASSO algorithm and the isolation forest method perform well with their dataset than the methods I have used to feature selection and outlier removing.

From this model as similar to the logistic regression I can predict the probability of risk level to positive in coronary disease early, with feeding the selected input variable values to the model and it may help to protect millions of lives which is unable to assess the worth.



In below figure (19.0) can be identified that how the ANN model gives the prediction probability of risk to having heart disease

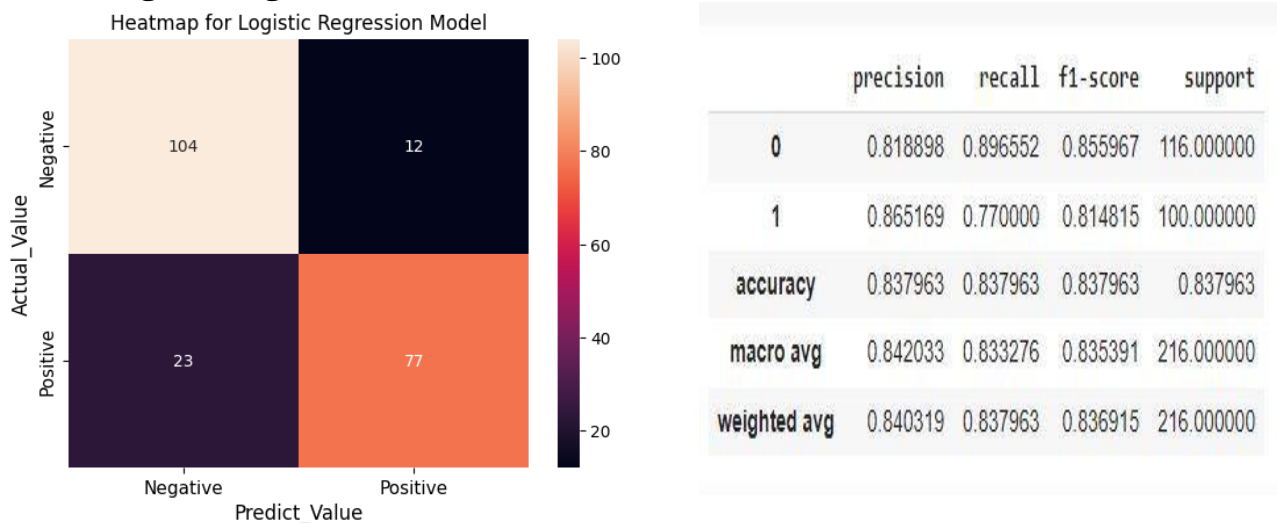


**Figure 22.0** The ANN model gives the prediction probability of risk to having heart disease with real life data

## 4.5 EVALUATE THE MODELS USING CONFUSION METRICS

In medical evaluations, the accuracy is not the only metrics to assess the model performance because accuracy, sensitivity and false-negative rate is also the critical factor to pay attention. As if model predict high ratio of false negative instances, the model is not performing well as it predicts high risk level patients as low risk, therefor I used confusion matrix to assess the model performance, as in research articles Singh and Kumar. (2020) and Basha and Venkatesh. (2019).

### 4.5.1 Logistic Regression



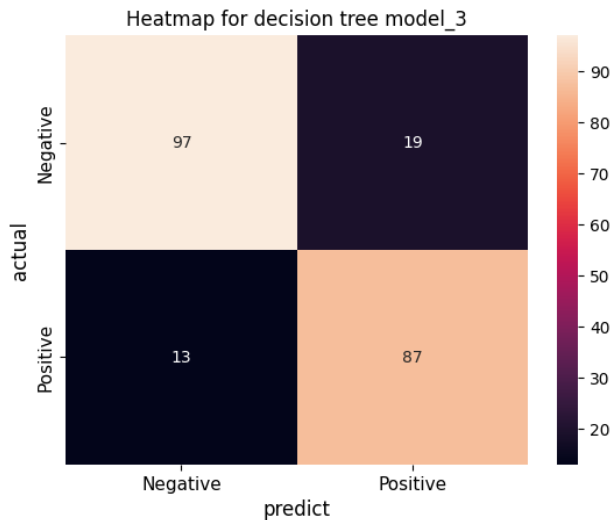
```
FN= 23
TP = 77
false_negative_rate_lo = (FN / (FN + TP)) * 100
print(f"False Negative Rate of Logistic Regression : {false_negative_rate_lo:.2f}%")

False Negative Rate of Logistic Regression : 23.00%
```

**Figure 23.0** The visualization of Logistic Regression model evaluation

My logistic regression model has reached 84% of testing accuracy level, it means the model predict 84% of instances (True Positive and True Negative) correctly, 84% of precision (correctly predicted true positive ratio), 83% of sensitivity and the False Negative rate is 23% which is quite high it means when model check 100 people who has affected with heart disease the model predicts 23 of them are in low risk whether they are in the high risk. If it is possible to gather more accurate dataset with more patients records this model may able to give high accuracy level and low false negative rate than this model performance.

## 4.5.2 Decision Tree



	precision	recall	f1-score	support
0	0.881818	0.836207	0.858407	116.000000
1	0.820755	0.870000	0.844660	100.000000
accuracy	0.851852	0.851852	0.851852	0.851852
macro avg	0.851286	0.853103	0.851534	216.000000
weighted avg	0.853548	0.851852	0.852043	216.000000

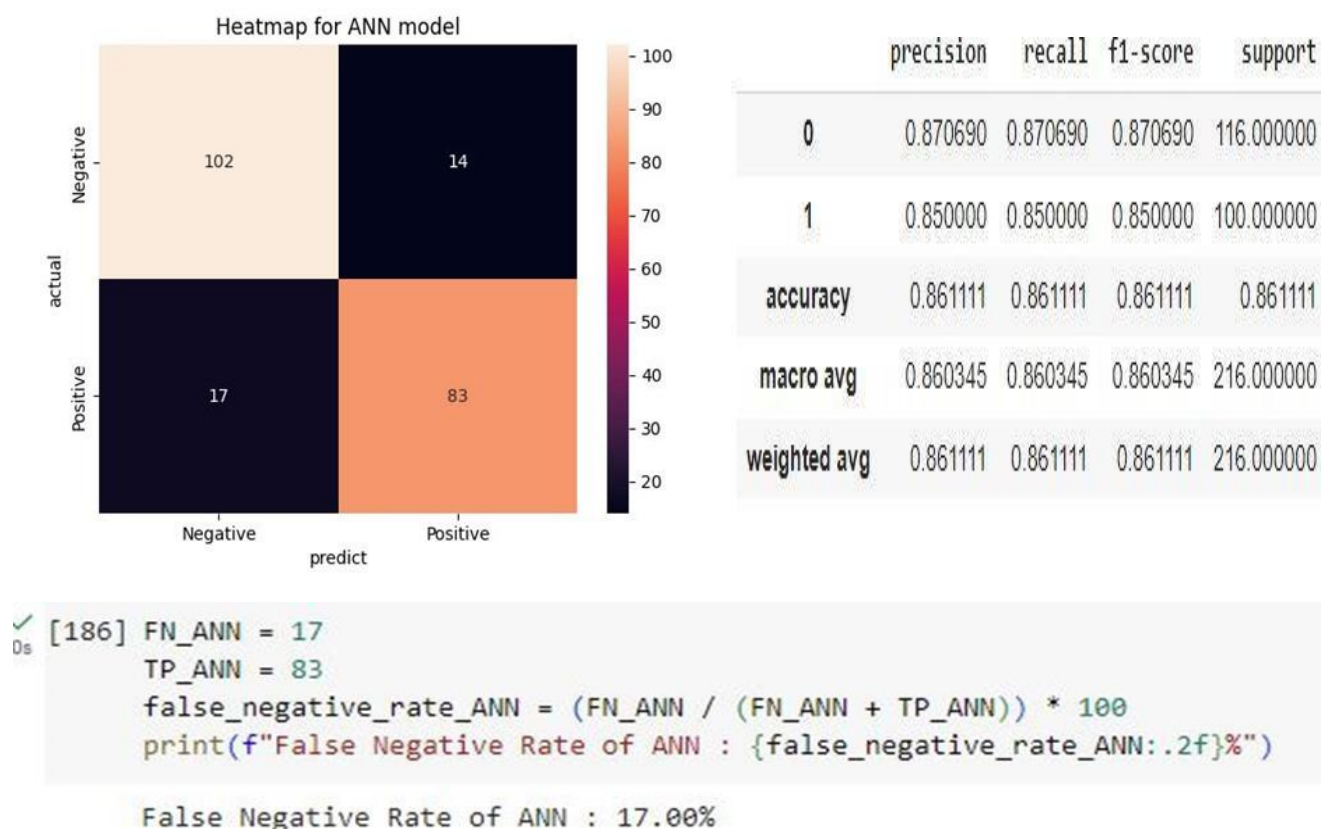
```
[158] FN_DT = 13
      TP_DT = 87
      false_negative_rate_DT = (FN_DT / (FN_DT + TP_DT)) * 100
      print(f"False Negative Rate of Decision Tree : {false_negative_rate_DT:.2f}%")
```

False Negative Rate of Decision Tree : 13.00%

**Figure 24.0** The visualization of Decision Tree model evaluation

My Decision tree 3rd approach has reached 85% of accuracy level, 85% of precision (correctly predicted true positive ratio), 85% of sensitivity and the False Negative rate is 13% with only using four independent variables, decision tree model achieves quite high accuracy than logistic regression model and the false negative ration is significantly lower than logistic regression model.

### 4.5.3 Artificial Neural Network



**Figure 25.0** The visualization of ANN model evaluation

My ANN model has reached 86% of accuracy level, 86% of precision (correctly predicted true positive ratio), 86% of sensitivity and the False Negative rate is 17%.

When comparing the three models which I have trained using past dataset of heart patients I reached high accuracy, precision and sensitivity from ANN model was 86% as it has high ability to identify the hidden patterns in dataset than supervised machine learning algorithms but the false negative rate is higher than decision tree algorithm (3rd approach). My customized decision tree model also reached 85% of accuracy, precision and sensitivity which was lower than 1% to ANN model. but it achieves 4% of lower false negative ration with comparing with ANN. So from the result analyzing which we found from this research I can deduce that Decision Tree with customized discretizes (3rd Approach) is the best algorithm to predict the heart disease early. But It would not be able to predict the risk level as ANN or Logistic Regression. so I have to imagine that when the Decision tree gives the 0 the patient is in low risk and 1 patient is in high risk. To predict the probability of risk to having heart disease early the ANN model is performing better than Logistic Regression model.

# **CHAPTER FIVE**

## **CONCLUSION AND FUTURE WORK**

## **CONCLUSION AND FUTURE WORK**

### **5.1 CONCLUSION**

The main aim of this research to predict the probability of risk to affect from heart disease early using machine learning algorithms and save lives, for that purpose I have analysis 3 main algorithms. Trained supervised machine learning algorithms as Logistic Regression and Decision Tree, Decision Tree algorithms trained with three different modifications and trained a deep learning algorithm which is call as Neural Network. From these models I have identified the Decision Tree (3rd Approach) is the best with the dataset I have used, based on accuracy, recall and False negative rate.

### **5.2 LIMITATIONS AND FUTURE WORKS**

The main limitation of this research is data, in here I used dataset with 717 records of different patients and some of independent variables didn't show good co-relationship with target as it unable to find the patterns due to lack of data, moreover this dataset only includes medical conditions as data. But family history, life style is also reason for increased the probability of heart disease. so I would like to purpose for the hospitals to gather those data of the patients as well and it may help to perform models more accurately.

Furthermore, Decision Tree model is not gives probability of risk level as output therefore I have assumed that if patient gets 0 as output he is in low risk and if gets 1 he is in high risk, but ANN and logistic regression models give probability level of risk as output.

And also from the knowledge I gather through my research, deep learning models are usually performing well in medical predations, so I would like to purpose to do more hyper-parameter Tuning with ANN model to check whether it gives more accurate prediction levels than I achieved.

## REFERENCE

Ali, M.M., Paul, B.K., Ahmed, K., Bui, F.M., Quinn, J.M. and Moni, M.A., 2021. Heart disease prediction using supervised machine learning algorithms: Performance analysis and comparison. *Computers in Biology and Medicine*, 136, p.104672.

Alshangiti, M., Sapkota, H., Murukannaiah, P.K., Liu, X. and Yu, Q., 2019, September. Why is developing machine learning applications challenging? a study on stack overflow posts. In *2019 acm/ieee international symposium on empirical software engineering and measurement (esem)* (pp. 1-11). IEEE

Basha, N., PS, A.K. and Venkatesh, P., 2019, December. Early detection of heart syndrome using machine learning technique. In *2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)* (pp. 387-391). IEEE.

Bashir, S., Khan, Z.S., Khan, F.H., Anjum, A. and Bashir, K., 2019, January. Improving heart disease prediction using feature selection approaches. In *2019 16th international bhurban conference on applied sciences and technology (IBCAST)* (pp. 619-623). IEEE

Bharti, R., Khamparia, A., Shabaz, M., Dhiman, G., Pande, S. and Singh, P., 2021. Prediction of heart disease using a combination of machine learning and deep learning. *Computational intelligence and neuroscience*, 2021

'Data Flair', (no date). *Artificial Neural Networks for Machine Learning – Every aspect you need to know about*. Available at: <https://data-flair.training/blogs/artificial-neural-networks-for-machine-learning/>.

Deshmukh, H. (2020). *Heart Disease UCI-Diagnosis & Prediction*. Available at : <https://towardsdatascience.com/heart-disease-uci-diagnosis-prediction-b1943ee835a7#70f9>

Domínguez-Almendros, S., Benítez-Parejo, N. and Gonzalez-Ramirez, A.R., 2011. Logistic regression models. *Allergologia et immunopathologia*, 39(5), pp.295-305

Gaziano, T.A., Bitton, A., Anand, S., Abrahams-Gessel, S. and Murphy, A., 2010. Growing epidemic of coronary heart disease in low-and middle-income countries. *Current problems in cardiology*, 35(2), pp.72-115.

Jarar, Z (2020). *Project: Predicting Heart Disease with Classification Machine Learning Algorithms*. Available at: [https://towardsdatascience.com/project-predicting-heart-disease-with-classification-machine-learning-algorithms-fd69e6fdc9d6#:~:text=Cp%20\(chest%20pain\)%2C%20is,pain%20%2C%20Value%204%3A%20asymptomatic](https://towardsdatascience.com/project-predicting-heart-disease-with-classification-machine-learning-algorithms-fd69e6fdc9d6#:~:text=Cp%20(chest%20pain)%2C%20is,pain%20%2C%20Value%204%3A%20asymptomatic)

Jindal, H., Agrawal, S., Khera, R., Jain, R. and Nagrath, P., 2021. Heart disease prediction using machine learning algorithms. In *IOP conference series: materials science and engineering* (Vol. 1022, No. 1, p. 012072). IOP Publishing.

Kumar, M.N., Koushik, K.V.S. and Deepak, K., 2018. Prediction of heart diseases using data mining and machine learning algorithms and tools. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(3), pp.887-898.

'Logistic Regression' (2019) *Wikipedia* Available at: [https://en.wikipedia.org/wiki/Logistic\\_regression](https://en.wikipedia.org/wiki/Logistic_regression) (Accessed: 27 August 2023)

Muhammad, Y., Tahir, M., Hayat, M. and Chong, K.T., 2020. Early and accurate detection and diagnosis of heart disease using intelligent computational model. *Scientific reports*, 10(1), p.19747.

Osisanwo, F.Y., Akinsola, J.E.T., Awodele, O., Hinmikaiye, J.O., Olakanmi, O. and Akinjobi, J., 2017. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, 48(3), pp.128-138.

Pawel, S. (2019). *Decision tree optimization*. Available at: <https://analizadanychwpilce.com/2019/10/11/optymalizacja-drzewa-decyzyjnego/>

Ponikowski, P., Anker, S.D., AlHabib, K.F., Cowie, M.R., Force, T.L., Hu, S., Jaarsma, T., Krum, H., Rastogi, V., Rohde, L.E. and Samal, U.C., 2014. Heart failure: preventing disease and death worldwide. *ESC heart failure*, 1(1), pp.4-25.

Ramprakash, P., Sarumathi, R., Mowriya, R. and Nithyavishnupriya, S., 2020, February. Heart disease prediction using deep neural network. In *2020 International Conference on Inventive Computation Technologies (ICICT)* (pp. 666-670). IEEE

Saini, A. (2021). *Conceptual understanding of Logistic Regression for Data Science Beginners*. Available at: <https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/> (Accessed : 28 July 2023)

Shah, D., Patel, S. and Bharti, S.K., 2020. Heart disease prediction using machine learning techniques. *SN Computer Science*, 1, pp.1-6.

Singh, A. and Kumar, R., 2020, February. Heart disease prediction using machine learning algorithms. In *2020 international conference on electrical and electronics engineering (ICE3)* (pp. 452-457). IEEE

Thomas, J. and Princy, R.T., 2016, March. Human heart disease prediction system using data mining techniques. In *2016 international conference on circuit, power and computing technologies (ICCPCT)* (pp. 1-5). IEEE



## APPENDIX

Link for the Colab Notebook:

<https://colab.research.google.com/drive/1TK-IIMVfn-FOdhjn8prChQHaf-1zSKV?usp=sharing>

### Import Libraries And Mount In Google Colab

```
#import the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder

#mount with the google drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

#import the dataset from drive
Heart_Df = pd.read_csv('/content/drive/MyDrive/heart_data.csv')
Heart_Df.head()

#checking the number of rows and columns of dataset
Heart_Df.shape

#check for the null entries and data types
Heart_Df.info()
Heart_Df.duplicated().sum()

#check the balance of target variable
Heart_Df.HeartDisease.value_counts()
# 0= Normal/1=Heart Disease
sns.countplot(x='HeartDisease', data = Heart_Df)
#get the statistical overall idea
Heart_Df.describe()
```

### Data Visualization

```
#Age is continous variable so we can use distpot to find the distribution
sns.distplot(Heart_Df['Age'], bins=20, kde=True, hist=True, rug=True)
plt.xlabel('Age', fontsize=12)
```

```
plt.ylabel('Density', fontsize=12)
plt.xticks(fontsize=11)
plt.yticks(fontsize=11)
plt.show()
```

```
#RestingBP is continous variable so we can use distpot to find the distribution
sns.distplot(Heart_Df['RestingBP'], bins = 20, kde=True, hist=True, rug=True)
plt.xlabel('RestingBP', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```

```
#Cholesterol is continous variable so we can use distpot to find the distribution
sns.distplot(Heart_Df['Cholesterol'], bins = 20, kde=True, hist=True, rug=True)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel('Cholesterol', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.show()
```

```
#MaxHR is continous variable so we can use distpot to find the distribution
sns.distplot(Heart_Df['MaxHR'], bins = 20, kde=True, hist=True, rug=True)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel('MaxHR', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.show()
```

```
#oldpeak is continous variable so we can use distpot to find the distribution
sns.distplot(Heart_Df['Oldpeak'], kde=True, hist=True, rug=True)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel('Oldpeak', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.show()
```

```
#Sex is categorical variable so use pie plot to get idea on distribution
Heart_Df.groupby('Sex').size().plot(kind='pie', autopct='%1f', title='Sex Distribution',
fontsize=12)
#Chest pain type is categorical variable so use pie plot
Heart_Df.groupby('ChestPainType').size().plot(kind='pie', autopct='%1f', title='Chest pain
Distribution')
#Fasting BS type is categorical variable so use pie plot
labels = 'Diabetics Negative', 'Diabetics Positive'
Heart_Df.groupby('FastingBS').size().plot(kind='pie', autopct='%1f', title='Fasting Blood
sugur Distribution', labels=labels)
```

```
#labels = 'Diabetics Negative', 'Diabetics Positive'
Heart_Df.groupby('RestingECG').size().plot(kind='pie', autopct='%1f', title='Resting ECG
Distribution')
#Exercise Angina type is categorical variable so use pie plot
Heart_Df.groupby('ExerciseAngina').size().plot(kind='pie', autopct='%1f', title='Exercise
Angina Distribution')
#ST_Slope type is categorical variable so use pie plot
Heart_Df.groupby('ST_Slope').size().plot(kind='pie', autopct='%1f', title='ST_Slope
Distribution')
```

### **Identify The Outliers Using Boxplot**

```
#plot boxplots to visualize the outliers properly for numerical continous variables
num_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
for i in num_cols:
    sns.boxplot(data = Heart_Df[i])
plt.xlabel(i, fontsize=12)
plt.show()
```

### **Remove Outliears**

```
#Remove the outliers of cholesterol find the max threslod using quantile
max_thresold = Heart_Df['Cholesterol'].quantile(0.99)
max_thresold
Heart_Df[Heart_Df['Cholesterol']>max_thresold].shape
#To remove outliers of cholesterol find the min threslod using quantile
min_thresold = Heart_Df['Cholesterol'].quantile(0.01)
min_thresold
Heart_Df[Heart_Df['Cholesterol']<=min_thresold].shape
#Remove the outliers of cholesterol
Heart_Df_C =
Heart_Df[(Heart_Df['Cholesterol']<max_thresold)&(Heart_Df['Cholesterol']>min_thresold)]

#To Remove the outliers from Oldpeak find the max threslod using quantile
max_thresold_o = Heart_Df_C['Oldpeak'].quantile(0.99)
max_thresold_o
Heart_Df_C[Heart_Df_C['Oldpeak']>max_thresold_o].shape
#To Remove the outliers from Oldpeak find the min threslod using quantile
min_thresold_o = Heart_Df_C['Oldpeak'].quantile(0.001)
min_thresold_o
Heart_Df_C[Heart_Df_C['Oldpeak']<min_thresold_o].shape

#Remove the outliers from Oldpeak
Heart_Df_O =
```

```
Heart_Df_C[(Heart_Df_C['Oldpeak']<max_thresold_o)&(Heart_Df_C['Oldpeak']>min_thresold_o)]
```

```
#To Remove the outliers from RestingBP find the max threslod using quantile
```

```
max_thresold_r = Heart_Df_O['RestingBP'].quantile(0.999)
```

```
max_thresold_r
```

```
Heart_Df_O[Heart_Df_O['RestingBP']>=max_thresold_r].shape
```

```
#To Remove the outliers from RestingBP find the min threslod using quantile
```

```
min_thresold_r = Heart_Df_O['RestingBP'].quantile(0.001)
```

```
min_thresold_r
```

```
Heart_Df_O[Heart_Df_O['RestingBP']<min_thresold_r].shape
```

```
#To Remove the outliers from RestingBP
```

```
Heart_Df_R =
```

```
Heart_Df_O[(Heart_Df_O['RestingBP']<max_thresold_r)&(Heart_Df_O['RestingBP']>min_thresold_r)]
```

```
#To Remove the outliers from MaxHR find the max threslod using quantile
```

```
max_thresold_m = Heart_Df_R['MaxHR'].quantile(0.999)
```

```
max_thresold_m
```

```
Heart_Df_R[Heart_Df_R['MaxHR']>=max_thresold_m].shape
```

```
#To Remove the outliers from MaxHR find the min threslod using quantile
```

```
min_thresold_m = Heart_Df_R['MaxHR'].quantile(0.001)
```

```
min_thresold_m
```

```
Heart_Df_R[Heart_Df_R['MaxHR']<min_thresold_m].shape
```

```
#To Remove the outliers from MaxHR
```

```
Heart_Df_Q =
```

```
Heart_Df_R[(Heart_Df_R['MaxHR']<max_thresold_m)&(Heart_Df_R['MaxHR']>min_thresold_m)]
```

```
Heart_Df_Q.shape
```

### **Visualize After Removing The Outliers**

```
num_q_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
```

```
for j in num_q_cols:
```

```
    sns.boxplot(data = Heart_Df_Q[j])
```

```
    plt.xlabel(j, fontsize=12)
```

```
    plt.show()
```

```
Heart_Df_Q.describe()
```

```
#visualize the Age vs heart disease using violine plot to identiy the relationship
```

```
fig = sns.violinplot(y = Heart_Df_Q['Age'], x = Heart_Df_Q['HeartDisease'])
```

```
fig.set_xticklabels(labels=["Negative heart disease", 'positive heart disease'], rotation=0, fontsize=11)
```

```
plt.gca().tick_params(axis='y', labels=11)
```

```
plt.xlabel('HeartDisease', fontsize =12)
plt.ylabel('Age', fontsize =12)
plt.title('Age vs Heart Disease', fontsize=12)
plt.show()
```

```
#visualize the RestingBP vs heart disease using violine plot to identiy the relationship
fig = sns.violinplot(y = Heart_Df_Q['RestingBP'], x = Heart_Df_Q['HeartDisease'])
fig.set_xticklabels(labels=["Negative heart disease", 'positive heart disease'], rotation=0,
fontsize=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('HeartDisease', fontsize =12)
plt.ylabel('RestingBP', fontsize =12)
plt.title('RestingBP vs Heart Disease', fontsize=12)
plt.show()
```

```
#visualize the Cholesterol vs heart disease using violine plot to identiy the relationship
fig = sns.violinplot(y = Heart_Df_Q['Cholesterol'], x = Heart_Df_Q['HeartDisease'])
fig.set_xticklabels(labels=["Negative heart disease", 'positive heart disease'], rotation=0,
fontsize=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('HeartDisease', fontsize =12)
plt.ylabel('Cholesterol', fontsize =12)
plt.title('Cholesterol vs Heart Disease', fontsize=12)
plt.show()
```

```
#visualize the MaxHR vs heart disease using violine plot to identiy the relationship
fig = sns.violinplot(y = Heart_Df_Q['MaxHR'], x = Heart_Df_Q['HeartDisease'])
fig.set_xticklabels(labels=["Negative heart disease", 'positive heart disease'], rotation=0,
fontsize=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('HeartDisease', fontsize =12)
plt.ylabel('MaxHR', fontsize =12)
plt.title('MaxHR vs Heart Disease', fontsize=12)
plt.show()
```

```
#visualize the Oldpeak vs heart disease using violine plot to identiy the relationship
fig = sns.violinplot(y = Heart_Df_Q['Oldpeak'], x = Heart_Df_Q['HeartDisease'])
fig.set_xticklabels(labels=["Negative heart disease", 'positive heart disease'], rotation=0,
fontsize=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('HeartDisease', fontsize =12)
plt.ylabel('Oldpeak', fontsize =12)
plt.title('Oldpeak vs Heart Disease', fontsize=12)
plt.show()
```

```
#sex vs heart disease
x_labels = ['Male', 'Female']
sns.countplot(x='Sex', hue='HeartDisease', data= Heart_Df_Q)
```

```
plt.legend(labels=['No Disease', 'Disease'])
plt.xticks(range(len(x_labels)), x_labels, fontsize=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('Sex', fontsize =12)
plt.ylabel('Count', fontsize =12)
plt.show()
```

```
#chest pain vs heart disease
sns.countplot(x='ChestPainType', hue='HeartDisease', data= Heart_Df_Q)
plt.legend(labels=['No Disease', 'Disease'], fontsize=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('ChestPainType', fontsize =12)
plt.ylabel('Count', fontsize =12)
plt.show()
```

```
#Blood Sugar vs Heart Disease
x_labels = ['Negative diabetes', 'Positive diabetes']
sns.countplot(x='FastingBS', hue='HeartDisease', data= Heart_Df_Q)
plt.xticks(range(len(x_labels)), x_labels, fontsize=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('FastingBS', fontsize =12)
plt.ylabel('Count', fontsize =12)
plt.legend(labels=['No Disease', 'Disease'])
plt.show()
```

```
#Resting ECG vs Heart Disease
sns.countplot(x='RestingECG', hue='HeartDisease', data= Heart_Df_Q)
plt.gca().tick_params(axis='x', labels=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('RestingECG', fontsize =12)
plt.ylabel('Count', fontsize =12)
plt.legend(labels=['No Disease', 'Disease'], fontsize=12)
plt.show()
```

```
#ExerciseAngina vs Heart Disease
x_labels = ['Negative ExerciseAngina', 'Positive ExerciseAngina']
sns.countplot(x='ExerciseAngina', hue='HeartDisease', data= Heart_Df_Q)
plt.xticks(range(len(x_labels)), x_labels, fontsize=11)
plt.gca().tick_params(axis='y', labels=11)
plt.xlabel('ExerciseAngina', fontsize =12)
plt.ylabel('Count', fontsize =12)
plt.legend(labels=['No Disease', 'Disease'])
plt.show()
```

```
#ST_Slope vs Heart Disease
sns.countplot(x='ST_Slope', hue='HeartDisease', data= Heart_Df_Q)
plt.gca().tick_params(axis='y', labels=11)
```

```
plt.gca().tick_params(axis='x', labelsiz=11)
plt.xlabel('ST_Slope', fontsize =12)
plt.ylabel('Count', fontsize =12)
plt.legend(labels=['No Disease', 'Disease'], fontsize=11)
plt.show()
```

## **Normalize The Data**

```
#normalize the data to minimise the error in classificaton process
cols_to_scale = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
scaler = MinMaxScaler()
Heart_Df_Q[cols_to_scale] = scaler.fit_transform(Heart_Df_Q[cols_to_scale])
Heart_Df_Q.head()
Heart_Df_Q.describe()
```

## **Encoding Variables**

```
#Sex include as a categorical variable with two categoies used pandas dummy variable
encoding
sex_dummy = pd.get_dummies(Heart_Df_Q.Sex)
sex_dummy.head()
```

```
#merged encoded dataframe and original dataframe
sex_merged =
pd.concat([Heart_Df_Q.reset_index(drop=True),sex_dummy.reset_index(drop=True)],
axis='columns')
sex_merged.head()
```

```
#Remove the sex column with categorical variable and one encoded column as it has only
two classes
sex_merged_f = sex_merged.drop(['Sex', 'F'], axis = 'columns')
sex_merged_f.head()
```

```
# in heart_merged_f dataframe in sex column 1=male and 0=female / and rename the
column
sex_merged_f = sex_merged_f.rename(columns={'M':'Sex'})
sex_merged_f.head()
```

```
ex_angina_dummy = pd.get_dummies(sex_merged_f.ExerciseAngina)
ex_angina_dummy.head()
ex_angina_merged =
pd.concat([sex_merged_f.reset_index(drop=True),ex_angina_dummy.reset_index(drop=Tr
ue)], axis='columns')
ex_angina_merged.head()
ex_angina_merged_f = ex_angina_merged.drop(['ExerciseAngina', 'N'], axis = 'columns')
ex_angina_merged_f.head()
#in exerciseAngina column 0= no exerciseAngina and 1=yes exerciseAngina
```

```
ex_angina_merged_f = ex_angina_merged_f.rename(columns={'Y': 'ExerciseAngina'})
ex_angina_merged_f.head()
```

#Going to map resting ECG to two categories as it doesn't show the relationship with heart disease when check it three classes (investigate)

```
ex_angina_merged_f.RestingECG.value_counts()
# Define the mapping dictionary for RestingECG categories
resting_ECG_mapping = {'Normal': 'Normal', 'ST': 'Abnormal', 'LVH': 'Abnormal'}
# Map the categories using the dictionary and create the 'resting_ECG_category' column
ex_angina_merged_f['resting_ECG_category'] =
ex_angina_merged_f['RestingECG'].map(resting_ECG_mapping)
# Drop the original 'RestingECG' column
ex_angina_merged_f.drop('RestingECG', axis=1, inplace=True)
# Print the updated dataset
ex_angina_merged_f.head()
```

#check the value count after mapping to two classes

```
ex_angina_merged_f.resting_ECG_category.value_counts()
RestingECG_merged =
ex_angina_merged_f.rename(columns={'resting_ECG_category': 'RestingECG'})
RestingECG_merged.head()
RestingECG_merged.RestingECG.value_counts()
RestingECG_dummy = pd.get_dummies(RestingECG_merged.RestingECG)
RestingECG_dummy.head()
```

```
RestingECG_merged_2 =
pd.concat([RestingECG_merged.reset_index(drop=True), RestingECG_dummy.reset_index
(drop=True)], axis='columns')
RestingECG_merged_2.head()
```

#Normal=0/Abnormal=1

```
RestingECG_merged_f = RestingECG_merged_2.drop(['RestingECG', 'Normal'], axis =
'columns')
RestingECG_merged_f.head()
```

#Normal=0/Abnormal=1

```
RestingECG_merged_f =
RestingECG_merged_f.rename(columns={'Abnormal': 'RestingECG'})
RestingECG_merged_f.head()
```

#encoding ordinal categorical variables using LabelEncoder() function

```
columns_encode = ['ChestPainType', 'ST_Slope']
label_encoder = LabelEncoder()
RestingECG_merged_f[columns_encode] =
RestingECG_merged_f[columns_encode].apply(label_encoder.fit_transform)
RestingECG_merged_f.head()
df_encoded = RestingECG_merged_f
```



```

df_encoded.head()

#check for the data type after encoding
df_encoded.info()

#find the correlation using sklearn .corr() function
df_en_corr = df_encoded.corr()
#plot heatmap to identify the correlation
plt.figure(figsize=(15, 8))
sns.heatmap(df_en_corr, annot=True)

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import statsmodels.api as sm

#define the X variables and y variables from dataset to use the logistic regression for feature
selection
X = df_encoded.drop('HeartDisease', axis=1)
y = df_encoded['HeartDisease']

#split data into training and testing set testing dataset is set as 20% from data set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

#Train the dataset using logistic regression
model = LogisticRegression(max_iter=100)
model.fit(X_train, y_train)

#find the coefficient against each independent variable
coef_df = pd.DataFrame({'Variable': X.columns, 'Coefficient': model.coef_[0]})

for i, row in coef_df.iterrows():
    print(f"Variable: {row['Variable']}")
    print(f"Coefficient: {row['Coefficient']}\n")

#find the absolute score of coefficient
importance_scores = abs(model.coef_[0])
importance_scores

#visualize in a dataframe
feature_importance_LR = pd.DataFrame({'Feature': X.columns, 'Importance':
importance_scores})
feature_importance_LR = feature_importance_LR.sort_values(by='Importance',
ascending=False )
print(feature_importance_LR)

#visualize scores in the bar plot

```

```

fig, axis = plt.subplots()
axis.barh(feature_importance_LR['Feature'], feature_importance_LR['Importance'],
align='center')
axis.invert_yaxis()
axis.set_xlabel('Importance Score', fontsize=12)
axis.set_title('Relationship according to the coefficient score')
axis.set_yticklabels(feature_importance_LR['Feature'], fontsize=12)
axis.set_xticklabels(axis.get_xticks(), fontsize=11)
plt.show()

```

```

# Create the countplot
sns.countplot(x='RestingECG', hue='HeartDisease', data=df_encoded)
plt.xlabel('Resting ECG')
plt.ylabel('Count')
plt.title('Relationship between Resting ECG and Heart Disease')
plt.show()
cor_ecg = df_encoded[['RestingECG', 'HeartDisease']]
cor_ecg.head()
cor_ecg.info()

```

```

#use pandas crosstab to get the count for each category of variable and normalize = index
to set percentage
cross_tab_ecg = pd.crosstab(index=cor_ecg['RestingECG'],
columns=cor_ecg['HeartDisease'], normalize='index')
cross_tab_ecg.head()
# visualize the heatmap
plt.figure(figsize=(5, 3))
sns.heatmap(cross_tab_ecg, annot=True)
plt.xlabel('Resting ECG', fontsize=12)
plt.ylabel('Heart Disease', fontsize=12)
plt.xticks(fontsize=11)
plt.yticks(fontsize=11)
plt.show()

```

### **Training A Model Using Logistic Regression To Predict The Probability Of Risk To Presence Heart Diseases**

```

#define a new dataframe to model development by selecting the variables with highest
confident values

```

```

df_model = df_encoded[['Age', 'ST_Slope', 'Sex', 'ExerciseAngina', 'RestingBP',
'Cholesterol', 'Oldpeak', 'ChestPainType', 'HeartDisease']]

```

```

df_model.head()

```

```

# Identify the relationship between chest pain type, ST slope and target occurrence

```

```

#define the independent variable and dependent variable
X1 = df_model.drop(columns='HeartDisease', axis=1)
Y1 = df_model['HeartDisease']

#split the data into training and testing set , testing size define as 30% from dataset to
overcome the overfitting problems
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, Y1, test_size=0.3,
random_state=0)
y1_test

#Train the logistic regression model using new dataframe with selected features
model_L = LogisticRegression()
model_L.fit(X1_train ,y1_train)

#checking the prediction values according to the training set
model_L.predict(X1_train)
#do the prediction according to the test set
pred = model_L.predict(X1_test)
pred
#check the accuracy score of training set
model_L.score(X1_train, y1_train)
#check the accuracy score of the model according to the testing set
model_L.score(X1_test, y1_test)
from sklearn.metrics import accuracy_score
accuracy_score(y1_test,pred)

```

### **Visualize The Relationship Between Variables With Model Output**

```

#define a dataframe with new name to plot the selected features from LR in regression
plots / only testing data set
inde_vari = X1_test[['Age', 'Oldpeak', 'ST_Slope','Cholesterol', 'Sex', 'ExerciseAngina',
'RestingBP', 'ChestPainType']]
inde_vari.head(1)

#select the dependent variable values of testing set
acc_v = y1_test
acc_v.head()

#select the prediction values of logistic regression model
columns = ['prediction_v']
df_checkv = pd.DataFrame(pred, columns=columns)

# Check the shape and content of the DataFrame
print(df_checkv.shape)
print(df_checkv.head())

```

```
acc_v.shape
inde_vari.shape
df_checkv.shape
```

```
#drop the index column from each dataframe
inde_vari_reset = inde_vari.reset_index(drop=True)
acc_v_reset = acc_v.reset_index(drop=True)
df_checkv_reset = df_checkv.reset_index(drop=True)
```

```
#merge the data in three different places for testing dataset
logi_curx = pd.concat([inde_vari_reset, acc_v_reset, df_checkv_reset], axis=1)
logi_curx.head()
```

```
#code to check the visualization of plots are correct
old_peak_risk = logi_curx.loc[logi_curx['Oldpeak'] > 0.3]
old_peak_risk.HeartDisease.value_counts()
old_peak_risk.prediction_v.value_counts()
```

```
#plot eight features in regplots to identify the trend of prediction probability vs independent variable
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(12, 8))
axes = axes.flatten()
```

```
for i, ax in enumerate(axes):
    xx_lo = logi_curx[inde_vari.columns[i]]
    yy_lo = logi_curx['prediction_v']

    sns.regplot(x=xx_lo, y=yy_lo, data=logi_curx, logistic=True, ci=None, ax=ax)
    ax.set_title(f'Regression Plot for {inde_vari.columns[i]}')
    ax.tick_params(axis='x', labelsize=11)
    ax.tick_params(axis='y', labelsize=11)
    ax.grid()
# Adjust layout to prevent overlap of titles and labels
plt.tight_layout()
plt.show()
```

```
#Hyperparameter Tunning
from sklearn.model_selection import GridSearchCV
#select set of parameters to check the best parameters to train the model
para_tune = {'penalty' : ['l1', 'l2', 'None'],
             'C' : [0.1, 1, 10],
             'max_iter' : [100, 1000, 10000],
             'solver':['lbfgs', 'liblinear', 'newton_cholesky']}
```

```
grid_search_lr = GridSearchCV(estimator=model_L, param_grid= para_tune, cv=5)
```

```

grid_search_lr.fit(X1_train, y1_train)

#define a set of outputs as best parameters
grid_search_lr.best_params_
grid_search_lr.score(X1_train, y1_train)
grid_search_lr.score(X1_test, y1_test)

#evaluate the model performance by passing selected values of input variables to the
implemented model
input_data = (0.244898, 2, 1, 0, 0.469388, 0.629630, 0.000000, 1)
input_data_np_array = np.asarray(input_data)
reshaped_data = input_data_np_array.reshape(1,-1)
#check with selected real life data
prediction = model_L.predict(reshaped_data)
print(prediction)
#check the prediction probability
prediction_prob = model_L.predict_proba(reshaped_data)
print(prediction_prob)
positive_proba = prediction_prob[:, 1]
negative_proba = prediction_prob[:, 0]
print("Positive Probabilities:", positive_proba)
print("Negative Probabilities:", negative_proba)
#find the precentage of probabiltiy of heart disease to predict the precentage of risk level
positive_proba_p = prediction_prob[:, 1] * 100
negative_proba_p = prediction_prob[:, 0] * 100
print("Positive Probabilities Precentage:", positive_proba_p)
print("Negative Probabilities precentage:", negative_proba_p)

```

### **Model Evaluation Using Confusion Matrix**

```

#convert y1_test dataframe data ta a numpy array to evaluate the model with confusion
matrix
y_test_array = y1_test.values
y_test_array

#Logistic Regression performance analysis using confusion matrix
actual_values = y_test_array
prediction_values = grid_search_lr.predict(X1_test)

#check the accuracy score
from sklearn.metrics import accuracy_score
accuracy_score(actual_values, prediction_values)

#define a confusion metrix to identiy the status of predictions (correct/wrong counts)
from sklearn.metrics import confusion_matrix
confusion_m = confusion_matrix(actual_values, prediction_values)
confusion_m

```

```

#generate the classification report according to the LR model
from sklearn.metrics import classification_report
report = pd.DataFrame(classification_report(actual_values, prediction_values,
output_dict=True))
report.transpose()

heatmap_lr = sns.heatmap(confusion_m, annot=True, fmt = 'd')
heatmap_lr.set_xticklabels(['Negative', 'Positive'], fontsize=11)
heatmap_lr.set_yticklabels(['Negative', 'Positive'], fontsize=11)
plt.xlabel('Predict_Value', fontsize=12)
plt.ylabel('Actual_Value', fontsize=12)
plt.title('Heatmap for Logistic Regression Model')
plt.show()

#calculate the FN rate using confusion matrix outputs
FN= 23
TP = 77
false_negative_rate_lo = (FN / (FN + TP)) * 100
print(f"False Negative Rate of Logistic Regression : {false_negative_rate_lo:.2f}%")

```

### **Training Using Decision Tree Algorithm**

```

#import decision tree classifier
from sklearn.tree import DecisionTreeClassifier
#define an object to train a decision tree algorithm
model_d = DecisionTreeClassifier()

#train the model using training data
model_d.fit(X1_train, y1_train)

#check the predictions on testing data
pred_d = model_d.predict(X1_test)
pred_d
#test the traing accuracy score
model_d.score(X1_train, y1_train)
#test the testing accuracy score
model_d.score(X1_test, y1_test)
#test the accuracy score using accuracy_score
accuracy_score(pred_d, y1_test)

# Define the range of max_depth values to plot the validation curve
para_range = np.arange(1, 30)

from sklearn.model_selection import validation_curve, train_test_split

# Calculate the train and validation score using cross validation
train_scores, valid_scores = validation_curve(

```

```

    model_d, X1_train, y1_train, param_name="max_depth", param_range=para_range,
    cv=5, scoring="accuracy", n_jobs=-1
)
#take the mean value of train and valid score
train_mean_d = np.mean(train_scores, axis=1)
valid_mean_d = np.mean(valid_scores, axis=1)

#plot to show how the model accuracy change with max depth
plt.style.use('classic')
plt.figure(figsize=(5, 3))
plt.plot(para_range, train_mean_d, label="Training score", color="blue")
plt.plot(para_range, valid_mean_d, label="Validation score", color="red")
plt.xlabel("Max Depth", fontsize=12)
plt.ylabel("Score", fontsize=12)
plt.title("Validation Curve for Decision Tree with numeric inputs")
plt.legend(loc="best")
plt.grid(True)
plt.gca().tick_params(axis='x', labelsize=11)
plt.gca().tick_params(axis='y', labelsize=11)
plt.show()

```

### **Performance Evaluation Using Confusion Metrics For Decision Tree Algorithm With Numerical Variables**

```

#pass y1_test dataframe values to numpy array
y_test_array = y1_test.values
#define variables for actual target values and predicted target values
actual_values_d = y_test_array
prediction_values_d = model_d.predict(X1_test)
accuracy_score(actual_values_d, prediction_values_d)
confusion_m_d = confusion_matrix(actual_values_d, prediction_values_d)
confusion_m_d

heatmap_d = sns.heatmap(confusion_m_d, annot=True, fmt = 'd')
heatmap_d.set_xticklabels(['Negative', 'Positive'])
heatmap_d.set_yticklabels(['Negative', 'Positive'])
plt.title('Heatmap for decision tree_model_1')
plt.xlabel('predict')
plt.ylabel('actual')
plt.show()

report_d = pd.DataFrame(classification_report(actual_values_d, prediction_values_d,
output_dict=True))
report_d.transpose()

```

### **Convert Numerical Variables To Categorical Variables Using Sklearn Library**

```

from sklearn.preprocessing import KBinsDiscretizer

# Define the number of bins and the strategy for binning
# Number of bins for low, medium, and high
n_bins = 3

# Binning strategy
strategy = 'uniform'

# Specify the columns to be binned
numerical_columns = ['Age', 'RestingBP', 'Cholesterol', 'Oldpeak']

# Initialize the KBinsDiscretizer with the specified parameters
discretizer = KBinsDiscretizer(n_bins=n_bins, encode='ordinal', strategy=strategy)
# Fit and transform the selected columns using the discretizer
binned_data = discretizer.fit_transform(df_model[numerical_columns])
# Create a new DataFrame with the binned data
binned_df = pd.DataFrame(binned_data, columns=numerical_columns)
binned_df.head()
binned_df.isna().sum()
binned_df.shape
df_model.head()
df_model.isna().sum()

#drop original numerical continous columns from the dataframe
df_drop = df_model.drop(['Age', 'Cholesterol', 'Oldpeak', 'RestingBP'], axis = 1)
df_drop.head()
df_drop.shape
df_drop.isna().sum()

#concatinate original dataframe with binned encoded dataframe
df_model_sk = pd.concat([df_drop.reset_index(drop=True),
binned_df.reset_index(drop=True)], axis=1)
df_model_sk.head()

#check for null values
df_model_sk[df_model_sk.isna().any(axis=1)]
df_model_sk.head(1)
df_model_sk.info()

#convert data types in to int as they are caterogies not floats
convert_vari_dic= {'Age': int,
                    'Cholesterol': int,
                    'Oldpeak' : int,
                    'RestingBP' : int
                    }

```



```

df_model_sk = df_model_sk.astype(convert_vari_dic)
df_model_sk.info()

#define independent and dependent variable using new encoded dataframe
X3 = df_model_sk.drop(columns='HeartDisease', axis=1)
Y3 = df_model_sk['HeartDisease']

X3_train, X3_test, y3_train, y3_test = train_test_split(X3, Y3, test_size=0.3,
random_state=0)

#define object to train with only categorical independent variable
df_model_sk_d = DecisionTreeClassifier()

#train the model using training data
df_model_sk_d.fit(X3_train, y3_train)

# Calculate training and testing score according to the second DT approach
train_scores_2, valid_scores_2 = validation_curve(
    df_model_sk_d, X3_train, y3_train, param_name="max_depth",
    param_range=para_range,
    cv=5, scoring="accuracy", n_jobs=-1
)
train_mean_2 = np.mean(train_scores_2, axis=1)
valid_mean_2 = np.mean(valid_scores_2, axis=1)

plt.figure(figsize=(5, 3))
plt.plot(para_range, train_mean_2, label="Training score", color="blue")
plt.plot(para_range, valid_mean_2, label="Validation score", color="red")
plt.xlabel("Max Depth")
plt.ylabel("Score")
plt.title("Validation Curve for Decision Tree_model_2")
plt.legend(loc="best")
plt.show()

df_model_sk_d.score(X3_train, y3_train)
df_model_sk_d.predict(X3_test)
df_model_sk_d.score(X3_test, y3_test)

```

### **Performance Evaluation Of Decision Tree With Categorical Class Using Sklearn Kbinsdiscretizer**

```

y_test_array_ds = y3_test.values
actual_values_ds = y_test_array_ds
prediction_values_ds = df_model_sk_d.predict(X3_test)

#find accuray score accoeding to the DT 2nd approach

```

```
accuracy_score(actual_values_ds, prediction_values_ds)
confusion_m_ds = confusion_matrix(actual_values_ds, prediction_values_ds)
confusion_m_ds
```

```
#visualize the confusion_matrix
heatmap_ds = sns.heatmap(confusion_m_ds, annot=True, fmt = 'd')
heatmap_ds.set_xticklabels(['Negative', 'Positive'])
heatmap_ds.set_yticklabels(['Negative', 'Positive'])
plt.xlabel('predict')
plt.ylabel('actual')
plt.title('Heatmap for decision tree_model_2')
plt.show()
```

```
report_ds = pd.DataFrame(classification_report(actual_values_ds, prediction_values_ds,
output_dict=True))
report_ds.transpose()
```

### **Decision Tree Model With Customize Bins**

#to define the own ranges for bins find the normalized values respect to the actual values

# normalized neumerical continous data (between 0 and 1)

```
normalized_data = [0.0, 0.1, 0.2, 0.3,0.4,0.5, 0.6,0.7, 0.8, 0.9, 1.0]
```

# Original min-max scaling range of cholesterol data

```
min_value_c = 85
```

```
max_value_c = 409
```

#Original min-max scaling range of age data

```
min_value_a = 28
```

```
max_value_a = 77
```

#Original min-max scaling range of RestingBP data

```
min_value_r = 94
```

```
max_value_r = 192
```

#Original min-max scaling range of Oldpeak data

```
min_value_o = 0
```

```
max_value_o = 3.8
```

```
#actual_value = (normalized_value * (max_value - min_value)) + min_value
```

# Reverse the normalization to get the actual cholesterol values

```
actual_values_c = [(n * (max_value_c - min_value_c)) + min_value_c for n in
normalized_data]
```

# Reverse the normalization to get the actual age values

```
actual_values_a = [(n * (max_value_a - min_value_a)) + min_value_a for n in
normalized_data]
```

# Reverse the normalization to get the actual RestingBP values

```
actual_values_r = [(n * (max_value_r - min_value_r)) + min_value_r for n in
normalized_data]
```

```

# Reverse the normalization to get the actual Oldpeak values
actual_values_o = [(n * (max_value_o - min_value_o)) + min_value_o for n in
normalized_data]

# Create a DataFrame with the actual values and normalized values
df = pd.DataFrame({'Normalized_Value': normalized_data, 'Cholesterol': actual_values_c,
'Age' : actual_values_a, 'RestingBP' : actual_values_r, 'Oldpeak' : actual_values_o })
print(df)
df_model.head()
df_model_remake = df_model.copy()
df_model_remake.head()

# Define custom bin edges and labels for variable levels
# Define the value ranges for low, medium, and high
bin_edges_cho = [0.0, 0.2, 0.5, 1.0]
# Labels for the corresponding bins
labels_cho = ['low', 'medium', 'high']
bin_edges_age = [0.0, 0.2, 0.6, 1.0]
labels_age = ['low', 'medium', 'high']
bin_edges_rbp = [0.0, 0.5, 1.0]
labels_rbp = ['low', 'high']
bin_edges_old = [0.0, 0.4, 0.7, 1.0]
labels_old = ['low', 'medium', 'high']

# Use pd.cut() to create categorical bins based on the value ranges and labels
df_model_remake['cholesterol_category'] = pd.cut(df_model_remake['Cholesterol'],
bins=bin_edges_cho, labels=labels_cho)
df_model_remake['Age_category'] = pd.cut(df_model_remake['Age'], bins=bin_edges_age,
labels=labels_age)
df_model_remake['RestingBP_category'] = pd.cut(df_model_remake['RestingBP'],
bins=bin_edges_rbp, labels=labels_rbp)
df_model_remake['Oldpeak_category'] = pd.cut(df_model_remake['Oldpeak'],
bins=bin_edges_old, labels=labels_old)
df_model_remake['Oldpeak_category'].fillna('low', inplace=True)
df_model_remake.head()

#check for null values in encoded dataframe
num_nan_values = df_model_remake['cholesterol_category'].isna().sum()
print("Number of NaN values in 'cholesterol_category':", num_nan_values)
num_nan_values = df_model_remake['Age_category'].isna().sum()
print("Number of NaN values in '      Age_category':", num_nan_values)

num_nan_values = df_model_remake['RestingBP_category'].isna().sum()
print("Number of NaN values in '      RestingBP_category':", num_nan_values)
num_nan_values = df_model_remake['Oldpeak_category'].isna().sum()

```

```

print("Number of NaN values in '      Oldpeak_category':", num_nan_values)

#find the specific place the null values laid on
df_model_remake[df_model_remake.isna().any(axis=1)]
#after identified the correct class of NaN vables fill the NaN with correct label
df_model_remake['cholesterol_category'].fillna('low', inplace=True)
df_model_remake['Age_category'].fillna('low', inplace=True)
df_model_remake['RestingBP_category'].fillna('low', inplace=True)
df_model_remake.isna().sum()

#drop the columns which encoded from dataframe
df_model_no_null = df_model_remake.drop(['Age', 'RestingBP', 'Cholesterol', 'Oldpeak'],
axis=1)
df_model_no_null.head()

df_model_no_null.cholesterol_category.value_counts()
df_model_no_null.Age_category.value_counts()
df_model_no_null.RestingBP_category.value_counts()
df_model_no_null.Oldpeak_category.value_counts()

label_encoder = LabelEncoder()
# Fit and transform the binned variable columns using the LabelEncoder
df_model_no_null['RestingBP_encoded'] =
label_encoder.fit_transform(df_model_no_null['RestingBP_category'])
df_model_no_null['Oldpeak_encoded'] =
label_encoder.fit_transform(df_model_no_null['Oldpeak_category'])
df_model_no_null['Age_encoded'] =
label_encoder.fit_transform(df_model_no_null['Age_category'])
df_model_no_null['cholesterol_encoded'] =
label_encoder.fit_transform(df_model_no_null['cholesterol_category'])

# Map the encoded values back to 'low' and 'high'
encoded_map_rbp = {0: 'low', 1: 'high'}
encoded_map_old = {0: 'low', 1: 'medium', 2: 'high'}
encoded_map_age = {0: 'low', 1: 'medium', 2: 'high'}
encoded_map_cho = {0: 'low', 1: 'medium', 2: 'high'}

df_model_no_null['RestingBP_category'] =
df_model_no_null['RestingBP_encoded'].map(encoded_map_rbp)
df_model_no_null['Oldpeak_category'] =
df_model_no_null['Oldpeak_encoded'].map(encoded_map_old)
df_model_no_null['Age_category'] =
df_model_no_null['Age_encoded'].map(encoded_map_age)

```

```

df_model_no_null['cholesterol_category'] =
df_model_no_null['cholesterol_encoded'].map(encoded_map_cho)

# Drop the intermediate encoded columns if from dataframe
df_model_no_null.drop(columns=['RestingBP_category'], inplace=True)
df_model_no_null.drop(columns=['Oldpeak_category'], inplace=True)
df_model_no_null.drop(columns=['Age_category'], inplace=True)
df_model_no_null.drop(columns=['cholesterol_category'], inplace=True)
df_model_no_null.head(1)

#rename the modified column names with actual dataframe column names
df_model_no_null =
df_model_no_null.rename(columns={'RestingBP_encoded': 'RestingBP',
'Oldpeak_encoded': 'Oldpeak', 'Age_encoded': 'Age', 'cholesterol_encoded':
'cholesterol'})
df_model_no_null.head()

#define dataframe with the column order in training dataset
df_model_no_null = df_model_no_null[['Age', 'ST_Slope', 'Sex', 'ExerciseAngina',
'RestingBP', 'cholesterol', 'Oldpeak', 'ChestPainType', 'HeartDisease']]
df_model_no_null.head(1)

#choose row from testing dataset for evaluate the model after training the model
row_index = 339
selected_row = df_model_no_null.iloc[row_index:row_index+1]
selected_row
df_model_no_null.info()

#define independent and dependent variable respect to the new dataframe with customized
bins
X2 = df_model_no_null.drop(columns='HeartDisease', axis=1)
Y2 = df_model_no_null['HeartDisease']

X2_train, X2_test, y2_train, y2_test = train_test_split(X2, Y2, test_size=0.3,
random_state=0)
X2_test.head(1)
X2_train.head(1)
y2_test.head()

#define object for train DT with cusomized bin dataframe
df_model_bin = DecisionTreeClassifier()
df_model_bin.fit(X2_train, y2_train)

# Visualized validation curve

```

```

train_scores_3, valid_scores_3 = validation_curve(
    df_model_bin, X2_train, y2_train, param_name="max_depth",
    param_range=para_range,
    cv=5, scoring="accuracy", n_jobs=-1
)
train_mean_3 = np.mean(train_scores_3, axis=1)
valid_mean_3 = np.mean(valid_scores_3, axis=1)

plt.figure(figsize=(5, 3))
plt.plot(para_range, train_mean_3, label="Training score", color="blue")
plt.plot(para_range, valid_mean_3, label="Validation score", color="red")
plt.xlabel("Max Depth", fontsize=12)
plt.ylabel("Score", fontsize=12)
plt.title("Validation Curve for Decision Tree with customized categories")
plt.gca().tick_params(axis='x', labelsize=11)
plt.gca().tick_params(axis='y', labelsize=11)
plt.legend(loc="best")
plt.grid()
plt.show()

df_model_bin.score(X2_train, y2_train)
pred_dt = df_model_bin.predict(X2_test)
df_model_bin.score(X2_test, y2_test)

input_data_DT = (2, 2, 1, 0, 0, 2, 1, 0)
input_data_as_numpy_array_DT = np.asarray(input_data_DT)
input_data_resaped_DT = input_data_as_numpy_array_DT.reshape(1,-1)
prediction_DT = df_model_bin.predict(input_data_resaped_DT)
print('The prediction from DT :', prediction_DT)

#import functions to visualized decision tree model of trained
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

#define feature names and target name from dataframe
feature_n_dt = df_model_no_null.drop('HeartDisease', axis=1).columns.tolist()
target_n_dt = df_model_no_null['HeartDisease'].unique().tolist()
df_model_no_null.info()
target_names_str = [str(class_name) for class_name in target_n_dt]

fig = plt.figure(figsize=(140, 80))
_ = tree.plot_tree(df_model_bin,
    feature_names=feature_n_dt,
    class_names=target_names_str,
    filled=True,
    fontsize=10)

```

```

plt.show()
fig.savefig('decision_tree_2.png')
max_depth_values = range(1, 20)
for max_depth in max_depth_values:
    df_model_bin = DecisionTreeClassifier(criterion="gini", max_depth=max_depth,
    random_state=0)
    df_model_bin.fit(X2_train, y2_train)
    y_pred = df_model_bin.predict(X2_test)

    accuracy = df_model_bin.score(X2_test, y2_test)
    print("Depth of a tree =", max_depth, "Accuracy:", round(accuracy, 4))

# Varying max_depth values
param_range = np.arange(1, 5)

# Calculate validation curve
train_scores, valid_scores = validation_curve(
    df_model_bin, X2_train, y2_train, param_name="max_depth",
    param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=-1
)
# Calculate mean training and validation scores
train_mean = np.mean(train_scores, axis=1)
valid_mean = np.mean(valid_scores, axis=1)

# Plot the validation curve
plt.figure(figsize=(5, 3))
plt.plot(param_range, train_mean, label="Training score", color="blue")
plt.plot(param_range, valid_mean, label="Validation score", color="red")
plt.xlabel("Maximum depth", fontsize=12)
plt.ylabel("Score", fontsize=12)
plt.title("Validation Curve for Decision Tree (Max_depth)")
plt.gca().tick_params(axis='x', labelsize=11)
plt.gca().tick_params(axis='y', labelsize=11)
plt.legend(loc="best")
plt.grid()
plt.show()

```

## **Optimize The DT Algorithm**

```

#Train model with max_depth = 3
df_model_bin_x = DecisionTreeClassifier(criterion = "gini", max_depth = 3, random_state=
0)
df_model_bin_x.fit(X2_train, y2_train)
y_pred = df_model_bin_x.predict(X2_test)
accuracy_train = df_model_bin_x.score(X2_train, y2_train)
#df_model_bin.score(X2_train, y2_train)

```

```

accuracy_test = accuracy_score(y2_test, y_pred)
print("Accuracy_train:", round(accuracy_train, 4))
print("Accuracy_test:", round(accuracy_test, 4))

```

```

fig = plt.figure(figsize=(20, 15))
_ = tree.plot_tree(df_model_bin_x,
                  feature_names=feature_names,
                  class_names=target_names_str,
                  filled=True,
                  fontsize=12)
plt.show()

```

```

# Identify the relationship between chest pain type, ST slope and target occurrence
df_agg = pd.crosstab([df_model.ChestPainType], [df_model.ST_Slope], values=
df_model.HeartDisease, aggfunc=np.average)
df_agg

```

```

plt.figure()
sns.heatmap(df_agg, annot=True)
plt.title('The average occurrence of Heart Disease')
plt.xlabel('ST_Slope', fontsize=12)
plt.ylabel('ChestPainType', fontsize=12)
plt.xticks(fontsize=11)
plt.yticks(fontsize=11)
plt.show()

```

```

#plot scatter plot to identify the trends between ST_Slope and ChestPainType
sns.catplot(data=df_model, x="ST_Slope", y="ChestPainType", kind="strip", jitter=True,
height=6, hue='HeartDisease')
plt.xlabel('ST_Slope', fontsize=12)
plt.ylabel('ChestPainType', fontsize=12)
plt.xticks(fontsize=11)
plt.yticks(fontsize=11)
plt.show()

```

```

input_data_DT_b = (2, 2, 1, 0, 0, 2, 1, 0)
input_data_as_numpy_array_DT_b = np.asarray(input_data_DT_b)
input_data_resaped_DT_b = input_data_as_numpy_array_DT_b.reshape(1,-1)
prediction_DT_b = df_model_bin_x.predict(input_data_resaped_DT_b)
print('The prediction from DT :',prediction_DT_b)

```

## **Performance Evaluation**

```

y_test_array_db = y2_test.values
actual_values_db = y_test_array_db
prediction_values_db = df_model_bin_x.predict(X2_test)
accuracy_score(actual_values_db, prediction_values_db)

```



```
confusion_m_db = confusion_matrix(actual_values_db, prediction_values_db)
confusion_m_db
```

```
heatmap_db = sns.heatmap(confusion_m_db, annot=True, fmt = 'd')
heatmap_db.set_xticklabels(['Negative', 'Positive'], fontsize=11)
heatmap_db.set_yticklabels(['Negative', 'Positive'], fontsize=11)
plt.xlabel('predict', fontsize=12)
plt.ylabel('actual', fontsize=12)
plt.title('Heatmap for decision tree model_3')
plt.show()
```

```
FN_DT = 13
TP_DT = 87
false_negative_rate_DT = (FN_DT / (FN_DT + TP_DT)) * 100
print(f"False Negative Rate of Decision Tree : {false_negative_rate_DT:.2f}%")
```

```
report_db = pd.DataFrame(classification_report(actual_values_db, prediction_values_db,
output_dict=True))
report_db.transpose()
```

## **Develop The Model Using ANN**

```
#import libararies for train ANN
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, BatchNormalization, Dropout, LSTM
```

```
len(X1_train.columns)
```

```
#Develop the simple ANN model
model = Sequential([
    keras.layers.Dense(128, input_shape=(8,), activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.1),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid'),
])
```

```
from tensorflow.keras.utils import plot_model
# Visualize the ANN model
plot_model(model, to_file='model_NN.png', show_shapes=True, show_layer_names=True,
```

```

rankdir='TB',
    dpi=96, layer_range=None, show_layer_activations=True)
tf.config.run_functions_eagerly(True)
#Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
#Train the model
#model.fit(X1_train,y1_train,epochs=50)
history = model.fit(X1_train, y1_train, epochs=50, validation_data=(X1_test, y1_test))

model.evaluate(X1_train, y1_train)
model.evaluate(X1_test,y1_test)

plt.figure(figsize=(5,3))
plt.plot(history.history['accuracy'], label='Train')
plt.plot(history.history['val_accuracy'], label = 'Validation')
plt.xlabel('No of epoch')
plt.ylabel('Accuracy Level')
plt.title('Model visualization on Accuracy')
plt.legend()
plt.grid()
plt.show()

```

## **Hyper-Parameter Tuning**

```

#from scikeras.wrappers import KerasClassifier
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier

def create_model(optimizer='adam'):
    model_h = Sequential()
    model_h.add(Dense(128, input_shape=(8,), activation='relu'))
    model_h.add(Dropout(0.2))
    model_h.add(Dense(64, activation='relu'))
    model_h.add(Dropout(0.1))
    model_h.add(Dense(32, activation='relu'))
    model_h.add(Dense(1, activation='sigmoid'))

    model_h.compile(optimizer=optimizer,
                    loss='binary_crossentropy',
                    metrics=['accuracy'])
    return model_h

#create the model with KerasClassifier
model_h = KerasClassifier(build_fn=create_model, verbose=0)

#parameters for tuning

```

```

optimizer_mh = ['adam', 'rmsprop', 'sgd']
epochs_mh = [10,50,100]
para_grid_ann = dict(optimizer=optimizer_mh, epochs=epochs_mh)
grid_ann = GridSearchCV(estimator=model_h, param_grid=para_grid_ann, n_jobs=-1,
cv=3)
ann_grid_r = grid_ann.fit(X1_train, y1_train)
print(ann_grid_r.best_score_)
print(ann_grid_r.best_params_)
ann_grid_r.score(X1_test,y1_test)

```

## **Model evaluation**

```

yp = model.predict(X1_test)
yp[:5]
y_test[:5]

```

```

prediction_ANN = model.predict(reshaped_data)
print(prediction_ANN)

```

```

#X_test prediction values convert into the 1D array
y_pred = []
for element in yp:
    if element > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)

```

```

y_test_array_ann = y1_test.values
actual_values_ann = y_test_array_ann
prediction_values_ann = y_pred
accuracy_score(actual_values_ann, prediction_values_ann)

```

```

confusion_m_ann = confusion_matrix(actual_values_ann, prediction_values_ann)
confusion_m_ann

```

```

heatmap_ann = sns.heatmap(confusion_m_ann, annot=True, fmt = 'd')
heatmap_ann.set_xticklabels(['Negative', 'Positive'], fontsize=11)
heatmap_ann.set_yticklabels(['Negative', 'Positive'], fontsize=11)
plt.xlabel('predict', fontsize=12)
plt.ylabel('actual', fontsize=12)
plt.title('Heatmap for ANN model')
plt.show()

```

```

FN_ANN = 17
TP_ANN = 83
false_negative_rate_ANN = (FN_ANN / (FN_ANN + TP_ANN)) * 100
print(f"False Negative Rate of ANN : {false_negative_rate_ANN:.2f}%")

```

```
report_ann = pd.DataFrame(classification_report(actual_values_ann,  
prediction_values_ann, output_dict=True))  
report_ann.transpose()
```