

# HEART DISEASE PREDICTION USING MACHINE LEARNING

A Major Project Report

*Submitted to*



**Jawaharlal Nehru Technological University Hyderabad**

*In partial fulfillment of the requirements for the*

*award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**ELECTRONICS & COMMUNICATION ENGINEERING**

By

**MADIREDDY RISHIKA (19VE1A04F3)**

**KANDANURU SRIVATSA (19VE1A04E8)**

**BEEJANI SHIVA KUMAR (19VE1A04C7)**

**Under the Guidance of**

**Mrs. Ashwini Kumari**

**ASSISTANT PROFESSOR**



**SREYAS**  
INSTITUTE OF ENGINEERING AND TECHNOLOGY  
AUTONOMOUS

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

Approved by AICTE, New Delhi | Affiliated to JNTUH, Hyderabad | Accredited by NAAC "A" Grade & NBA  
Hyderabad | PIN: 500068

(2019 – 2023)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

Approved by AICTE, New Delhi | Affiliated to JNTUH, Hyderabad | Accredited by NAAC "A" Grade & NBA |  
Hyderabad | PIN: 500068

*Certificate*

This is to certify that the Major Project Report on “**HEART DISEASE PREDICTION USING MACHINE LEARNING**” submitted by **MADIREDDY RISHIKA, KANDANURU SRIVATSA, BEEJANI SHIVA KUMAR** bearing Hall Ticket No's. **19VE1A04F3, 19VE1A04E8, 19VE1A04C7** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Electronics & Communication Engineering** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2022- 23 is a record of bonafide work carried out by them under our guidance and Supervision.

**Guide**  
**Mrs. T. Ashwini Kumari**

**Head of the Department**

**Project Coordinator**

**Signature of the External Examiner**



**SREYAS**  
INSTITUTE OF ENGINEERING AND TECHNOLOGY  
AUTONOMOUS

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

Approved by AICTE, New Delhi | Affiliated to JNTUH, Hyderabad | Accredited by NAAC "A" Grade & NBA |  
Hyderabad | PIN: 500068

**DECLARATION**

We, **MADIREDDY RISHIKA, KANDANURU SRIVATSA, BEEJANI SHIVA KUMAR** bearing Roll No's **19VE1A04F3, 19VE1A04E8, 19VE1A04C7** hereby declare that the Major Project titled "**HEART DISEASE PREDICTION USING MACHINE LEARNING**" done by me under the guidance of **Mrs. T. Ashwini Kumari**, which is submitted in the partial fulfillment of the requirement for the award of the B.Tech degree in **Electronics & Communication Engineering** at **Sreyas Institute of Engineering & Technology** for Jawaharlal Nehru Technological University, Hyderabad is my original work.

**MADIREDDY RISHIKA(19VE1A04F3)**

**KANDANURU SRIVATSA(19VE1A04E8)**

**BEEJANI SHIVA KUMAR(19VE1A04C7)**

## ***ACKNOWLEDGEMENT***

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

We take this opportunity to acknowledge with thanks and deep sense of gratitude to **Mrs. T. Ashwini Kumari, Assistant Professor, Department of ECE** for her constant encouragement and valuable guidance during the Project work.

A Special note of Thanks to **Prof. Chinnam S V Maruti rao, Associate Professor and Head of the Department**, who has been a source of Continuous motivation and support in the completion of this project. He had taken time and effort to guide and correct us all through the span of this work.

We owe very much to the **Department Faculty, Principal and the Management** who made our term at Sreyas a Stepping stone for my career. We treasure very moment we had spent in the college.

Last but not least, our heartiest gratitude to our parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

**MADIREDDY RISHIKA(19VE1A04F3)**  
**KANDANURU SRIVATSA(19VE1A04E8)**  
**BEEJANI SHIVA KUMAR(19VE1A04C7)**

# CONTENTS

<b>ABSTRACT.....</b>	<b>i</b>
<b>LIST OF FIGURES.....</b>	<b>ii</b>
<b>LIST OF TABLES.....</b>	<b>iii</b>
<b>Chapter 1: INTRODUCTION.....</b>	<b>1</b>
1.1 Problem Statement.....	3
1.2 Existing System.....	3
1.3 Proposed System.....	4
1.4 Project Pipeline.....	4
1.5 Diabetes Prediction using Medical Data.....	5
<b>Chapter 2: LITERATURE SURVEY.....</b>	<b>7</b>
<b>Chapter 3: EXPERIMENTAL SETUP AND PROCEDURE.....</b>	<b>11</b>
<b>Chapter 4: METHODOLOGY.....</b>	<b>16</b>
4.1 Data Constrains.....	16
4.2 Data Preprocessing.....	17
4.2.1 Scaling.....	17
4.2.2 Normalization.....	19
4.2.3 Binarization.....	22
4.3 Training and Testing.....	25
<b>Chapter 5: Algorithms.....</b>	<b>34</b>
5.1 Random Forest Classifier.....	34
5.2 Decision Tree.....	38
5.3 Logistic Regression.....	41
<b>Chapter 6: Results.....</b>	<b>51</b>
<b>Chapter 7: Advantages.....</b>	<b>58</b>
<b>Chapter 8: Future Scope.....</b>	<b>60</b>
<b>Chapter 9: Applications.....</b>	<b>62</b>
<b>Chapter 10: Code.....</b>	<b>63</b>
<b>Chapter 11: Conclusion.....</b>	<b>73</b>
<b>REFERENCES.....</b>	<b>74</b>

<i>

## **ABSTRACT**

Day by day the cases of heart diseases are increasing at a rapid rate and it's very Important and concerning to predict any such diseases beforehand. This diagnosis is a difficult task i.e. it should be performed precisely and efficiently. We prepared a heart disease prediction system to predict whether the patient is likely to be diagnosed with a heart disease or not using the medical history of the patient. We used different algorithms of machine learning such as logistic regression, Random forest classifier and KNN to predict and classify the patient with heart disease. A quite helpful approach was used to regulate how the model can be used to improve the accuracy of prediction of Heart Attack in any individual. The strength of the proposed model was quiet satisfying and was able to predict evidence of having a heart disease in a particular individual by using Random Forest Classifier which showed a good accuracy. The Given heart disease prediction system enhances medical care and reduces the cost. This project gives us significant knowledge that can help us predict the patients with heart disease. It is implemented on the .pynb format.

## **LIST OF FIGURES**

<b>FIGURE NUMBER</b>	<b>FIGURE NAME</b>	<b>PAGE NUMBER</b>
FIG 1	METHODOLOGY	16
FIG 2	FACTORS AFFECTING TRAINING DATA QUALITY	42
FIG 3	WORKING OF RANDOM FOREST ALGORITHM	45
FIG 4	GENERAL STRUCTURE OF A DECISION TREE	47
FIG 5	GRADIENT BOOSTED TREES FOR REGRESSION	59
FIG 6	DETERMINING WHETHER A PERSON IS HAVING HEART DISEASE	61
FIG 7	RELATION BETWEEN FACTORS AND TARGET	62
FIG 8	HEAT MAP	63

**LIST OF TABLES**

<b>TABLE NUMBER</b>	<b>TABLE NAME</b>	<b>PAGE NUMBER</b>
Table 1	VARIOUS ATTRIBUTES	21



## **Chapter - 1**

### **INTRODUCTION**

Heart is an important organ of the human body. It pumps blood to every part of our anatomy. If it fails to function correctly, then the brain and various other organs will stop working, and within few minutes, the person will die. Change in lifestyle, work related stress and bad food habits contribute to the increase in the rate of several heart-related diseases. Heart diseases have emerged as one of the most prominent causes of death all around the world. According to World Health Organisation, heart related diseases are responsible for taking 17.7 million lives every year, 31% of all global deaths. In India too, heart-related diseases have become the leading cause of mortality. Heart diseases have killed 1.7 million Indians in 2016, according to the 2016 Global Burden of Disease Report, released on September 15, 2017. Heart-related diseases increase the spending on health care and also reduce the productivity of an individual. Estimates made by the World Health Organisation (WHO), suggest that India has lost up to \$237 billion, from 2005- 2015, due to heart-related or Cardio-vascular diseases. Thus, feasible and accurate prediction of heart-related diseases is very important. Medical organizations, all around the world, collect data on various health-related issues. These data can be exploited using various machine learning techniques to gain useful insights. But the data collected is very massive and, many times, this data can be very noisy. These datasets, which are too overwhelming for human minds to comprehend, can be easily explored using various machine learning techniques. Thus, these algorithms have become very useful, in recent times, to predict the presence or absence of heart-related diseases accurately. The usage of information technology in the health care industry is increasing day by day to aid doctors in decision making activities. It helps doctors and physicians in disease management, medications, and discovery of patterns and relationships among diagnosis data. Current approaches to predict cardiovascular risk fail to identify many people who would benefit from preventive treatment, while others receive unnecessary intervention. Machine-learning offers an opportunity to improve accuracy by exploiting complex interactions between risk factors. We assessed whether machine-learning can improve cardiovascular risk prediction.

### **1.1 PROBLEM STATEMENT:**

In this project we want to identify a patient has heart disease or not. Our objective is to build a Heart disease prediction system using Machine learning techniques. In the past, such systems were rule-based. Machine learning offers powerful new ways.

In this project, we will analyse dataset taken from Kaggle Website. This dataset dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease. dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V.

### **1.2 EXISTING SYSTEM:**

In existing system methods such as Decision tree, SVM, KNN, Logistic Regression, Naïve Bayes model are used to find out the heart disease. The methods used in the existing system are based on supervised learning and the accuracy obtained by these methods is mostly between 80% and 90%.

### **1.3 PROPOSED SYSTEM:**

The proposed system overcomes the above mentioned issue in an efficient way. It aims at analyzing the number of heart patients that are present in the dataset.

In proposed System, we use Random forest, Decision tree and Logistic Regression to classify the heart disease dataset.

Random Forest is an algorithm for classification and regression.

The dataset is classified into trained and test dataset where the data can be trained individually, these algorithms are very easy to implement as well as very efficient in producing better results and can able to process large amount of data.

Even for large dataset these algorithms are extremely fast and can able to give accuracy of about over 100%.

## **1.4 PROJECT PIPELINE:**

There are several approaches you could take to build a machine learning model for predicting heart disease using a dataset. Here is one possible architecture that you could use:

### **1.41 Data Understanding:**

Here, we need to load the data and understand the features present in it. This would help us choose the features that we will need for your final model. This step involves cleaning and preparing the dataset for modelling. This may include tasks such as handling missing values, encoding categorical variables, and scaling numeric features. In the data pre-processing step of building a machine learning model for predicting heart disease using a dataset, you would typically perform a number of tasks to clean and prepare the data for modelling. Some common data pre-processing tasks include:

- Handling missing values: If the dataset contains missing values, you will need to decide how to handle them. You could try imputing the missing values using techniques such as mean or median imputation, or you could simply drop rows or columns with missing values.
- Encoding categorical variables: If the dataset contains categorical variables, you will need to encode them as numeric values so that they can be used as inputs to a machine learning model. There are several ways to encode categorical variables, such as one-hot encoding or ordinal encoding.
- Scaling numeric features: It is often a good idea to scale numeric features so that they are on the same scale. This can help the model to converge faster and improve its performance. You can scale the features using techniques such as standardization or normalization.
- Splitting the dataset into training and test sets: Before you can train a machine learning model, you will need to split the dataset into a training set

and a test set. The training set will be used to fit the model, while the test set will be used to evaluate its performance.

- **Balancing the dataset:** If the dataset is imbalanced (i.e., there are significantly more examples of one class than the other), it can be beneficial to balance the dataset by oversampling the minority class or under sampling the majority class. This can help the model to learn from a more balanced set of examples.

These are just a few of the data pre-processing tasks that you might need to perform when building a machine learning model for predicting heart disease using a dataset. The specific tasks you need to perform will depend on the characteristics of the dataset and the requirements of your application.

#### **1.42 Exploratory data analytics (EDA):**

Normally, in this step, we need to perform univariate and bivariate analyses of the data, followed by feature transformations, if necessary. For the current data set, we have done oversampling as the classes in the target variable were not balanced. We have also done the scaling for transforming our data so that it fits within a specific scale '-2-2'. In this step, you would select the features from the dataset that are most relevant for predicting heart disease. This could be done using techniques such as correlation analysis or feature importance scores. Feature selection is the process of selecting a subset of the available features in a dataset to use as inputs to a machine learning model. In the context of predicting heart disease using a dataset, feature selection can be an important step in building a high-performing model. Here are a few approaches you could take to select features for your model:

- **Correlation analysis:** One way to identify potentially useful features is to calculate the correlation between each feature and the target variable (i.e., whether or not an individual has heart disease). Features that are strongly correlated with the target variable may be more informative and useful for prediction.

- Feature importance scores: Another approach is to use a machine learning algorithm to compute feature importance scores, which measure the impact of each feature on the model's predictions. Features with high importance scores may be more useful for prediction and could be selected for the model.
- Wrapper methods: Wrapper methods involve training a machine learning model using a subset of the features and evaluating its performance. The subset of features that leads to the best model performance is then selected for use in the final model.
- Filter methods: Filter methods involve evaluating the relevance of each feature independently, without considering the relationships between features. Features that are highly relevant to the target variable can be selected for use in the model.

No matter which approach you take, it is important to carefully evaluate the performance of the model with different sets of features to ensure that you are selecting the most useful ones for prediction.

### **1.43 Train/Test Split:**

Now we are familiar with the train/test split, which we can perform in order to check the performance of our models with unseen data. Next, you would train a machine learning model using the selected features and evaluate its performance using a suitable evaluation metric, such as accuracy or F1 score. You may want to try out a few different models and select the one that performs the best on the evaluation metric. Once you have selected the features for your machine learning model for predicting heart disease using a dataset, the next step is to train the model and evaluate its performance. Here is a general outline of the process:

- Split the dataset into a training set and a test set: Before you can train a model, you will need to split the dataset into a training set and a test set. The training set will be used to fit the model, while the test set will be used to evaluate its performance.

- **Select a model:** There are many different types of machine learning models that you could use for predicting heart disease, such as logistic regression, decision trees, or support vector machines. You may want to try out a few different models and select the one that performs the best on the evaluation metric.
- **Train the model:** Once you have selected a model, you can fit it to the training set using the selected features. This involves adjusting the model's parameters to minimize the error between the model's predictions and the true labels in the training set.
- **Evaluate the model:** After the model has been trained, you can evaluate its performance on the test set using a suitable evaluation metric, such as accuracy, precision, or F1 score. This will give you an idea of how well the model is able to generalize to new data.
- **Fine-tune the model:** If the model's performance is not satisfactory, you can try to fine-tune it by adjusting the hyper parameters or by adding or removing features. You may want to use techniques such as grid search or Bayesian optimization to find the best combination of hyper parameters.
- **Evaluate the fine-tuned model:** After you have fine-tuned the model, you should evaluate its performance on the test set again to see if the changes have improved the model's performance. This is a general outline of the process for training and evaluating a machine learning model for predicting heart disease using a dataset. The specific steps you take will depend on the characteristics of the dataset and the requirements of your application.

#### **1.44 Model-Building/Hyper parameter Tuning:**

This is the final step at which we can try different models until we get the desired level of performance on the given dataset. We should try and see if we get a better model by the various sampling techniques. Once you have selected a model, you can try to fine-tune its hyper parameters to further improve its performance. This could involve techniques such as grid search or Bayesian

optimization. Model fine-tuning is the process of adjusting the hyper parameters of a machine learning model in order to improve its performance. In the context of predicting heart disease using a dataset, model fine-tuning could involve adjusting parameters such as the learning rate, regularization strength, or the number of hidden units in a neural network. There are several approaches you could take to fine-tune a machine learning model for predicting heart disease. Here are a few examples:

- **Grid search:** Grid search involves specifying a grid of hyper parameter values to search over, and training and evaluating a model for each combination of values. The combination of values that leads to the best performance on the evaluation metric is then selected as the final model.
- **Bayesian optimization:** Bayesian optimization is a method for finding the optimal set of hyper parameters for a machine learning model by constructing a probabilistic model of the performance of different hyper parameter combinations. This method can be more efficient than grid search, as it uses a smaller number of model evaluations to find the optimal hyper parameters.
- **Early stopping:** Early stopping is a technique for fine-tuning the number of epochs (i.e., iterations over the training dataset) to train a model. The model is trained for a fixed number of epochs and then evaluated on a validation set. If the model's performance on the validation set does not improve after a certain number of epochs, training is stopped to prevent overfitting.
- **Learning rate scheduling:** Learning rate scheduling involves adjusting the learning rate of a model during training based on a pre-defined schedule. This can help the model to converge faster and improve its performance.

These are just a few examples of techniques that you could use to fine-tune a machine learning model for predicting heart disease using a dataset. The specific techniques you use will depend on the characteristics of the dataset and the requirements of your application.

### **1.45 Model Evaluation:**

We need to evaluate the models using appropriate evaluation metrics. We need to choose an appropriate evaluation metric which reflects this business goal. Finally, you can deploy the trained model in a production environment, where it can be used to make predictions on new data. You may also want to monitor the model's performance over time to ensure that it continues to make accurate predictions. Once you have trained and fine-tuned a machine learning model for predicting heart disease using a dataset, the next step is to deploy the model in a production environment, where it can be used to make predictions on new data. Here are a few considerations for deploying your model:

- **Model serving:** There are several ways to serve a machine learning model for prediction, such as using a serverless function or deploying the model to a cloud platform. You will need to choose the approach that is most suitable for your application.
- **Model monitoring:** It is important to monitor the performance of the deployed model over time to ensure that it continues to make accurate predictions. This could involve monitoring the model's accuracy or other evaluation metrics, as well as tracking the input data and the model's predictions.
- **Model updates:** You may need to update the deployed model periodically, either to incorporate new data or to improve its performance. You will need to have a plan in place for how to update the model and test the changes before deploying them to production.
- **Model governance:** To ensure that the deployed model is being used responsibly, you may need to implement policies and procedures for managing the model. This could include things like setting rules for how the model is used, defining roles and responsibilities for model stakeholders, and establishing processes for audit and compliance.

By carefully planning and executing the model deployment process, you can ensure that your machine learning model for predicting heart disease is



deployed successfully and used effectively to make predictions on new data. This is just one possible architecture for building a machine learning model for predicting heart disease using a dataset. There are many other approaches you could take, depending on the specific requirements of your application and the characteristics of the dataset you are working with.

## **CHAPTER 2**

### **LITERATURE SURVEY**

A quiet significant amount of work related to the diagnosis of Cardiovascular Heart disease using Machine Learning algorithms has motivated this work. This paper contains a brief literature survey. An efficient Cardiovascular disease prediction has been made by using various algorithms some of them include Logistic Regression, KNN, Random Forest Classifier Etc. It can be seen in Results that each algorithm has its strength to register the defined objectives.

The model incorporating IHDPS had the ability to calculate the decision boundary using the previous and new model of machine learning and deep learning. It facilitated the important and the most basic factors/knowledge such as family history connected with any heart disease. But the accuracy that was obtained in such IHDPS model was far more less than the new upcoming model such as detecting coronary heart disease using artificial neural network and other algorithms of machine and deep learning. The risk factors of coronary Heart disease or atherosclerosis is identified using the inbuilt implementation algorithm using uses some techniques of Neural Network and were just accurately able to predict whether the test patient is suffering from the given disease or not.

A deep Neural Network was built incorporating the given attributes related to the disease which were able to produce a output which was carried out by the output perceptron and almost included 120 hidden layers which is the basic and most relevant technique of ensuring a accurate result of having heart disease if we use the model for Test Dataset. The supervised network has been advised for diagnosis of heart diseases. When the testing of the model was done by a doctor using an unfamiliar data, the model used and trained from the previous learned data and predicted the result thereby calculating the accuracy of the given model. A literature survey is a comprehensive review of the existing research on a particular topic. Conducting a literature survey can help you identify trends and gaps in the current research and can inform your own research or project. In the context of predicting heart disease using machine learning, there are a number of different approaches that have been proposed and evaluated in the literature.

One common approach is to use various machine learning algorithms to predict the likelihood of an individual developing heart disease based on various risk factors, such as age, blood pressure, cholesterol levels, and other medical history. Some studies have used traditional machine learning algorithms, such as decision trees, logistic regression, and support vector machines, while others have used more modern techniques, such as neural networks and deep learning.

Other studies have focused on using machine learning to improve the accuracy of heart disease diagnosis by analyzing various types of data, such as electrocardiogram (ECG) signals, imaging data, and physiological measurements. Some of these studies have used supervised learning techniques, in which the algorithm is trained on labeled data, while others have used unsupervised learning techniques, in which the algorithm is not given any labeled data and must learn to identify patterns in the data on its own. There are also a number of studies that have focused on developing machine learning-based models for predicting the outcomes of various treatments for heart disease, such as the likelihood of success or the potential for adverse side effects. Overall, the literature on heart disease prediction using machine learning is quite extensive, and there are many different approaches that have been proposed and evaluated. It is important to carefully review the existing research in order to understand the strengths and limitations of different approaches and to identify opportunities for future research.

## **CHAPTER 3**

### **DATA SOURCE**

An Organized Dataset of individuals had been selected keeping in mind their history of heart problems and in accordance with other medical conditions. Heart diseases are the diverse conditions by which the heart is affected. According to World Health Organization (WHO), the greatest number of deaths in middle aged people are due to Cardiovascular diseases. We take a data source which is comprised of medical history of 304 different patients of different age groups. This dataset gives us the much-needed information i.e. the medical attributes such as age, resting blood pressure, fasting sugar level etc. of the patient that helps us in detecting the patient that is diagnosed with any heart disease or not. This dataset contains 13 medical attributes of 304 patients that helps us detecting if the patient is at risk of getting a heart disease or not and it helps us classify patients that are at risk of having a heart disease and that who are not at risk. This Heart Disease dataset is taken from the UCI repository. According to this dataset, the pattern which leads to the detection of patient prone to getting a heart disease is extracted. These records are split into two parts: Training and Testing. This dataset contains 303 rows and 14 columns, where each row corresponds to a single record. All attributes are listed in 'Table 1'.

S. No	Observation	Description	Values
1.	Age	Age in Years	Continuous
2.	Sex	Sex of Subject	Male/Female
3.	CP	Chest Pain	Four Types
4.	Trestbps	Resting Blood Pressure	Continuous
5.	Chol	Serum Cholesterol	Continuous
6.	FBS	Fasting Blood Sugar	<,or> 120 mg/dl
7.	Restecg	Resting Electrocardiograph	Five Values
8.	Thalach	Maximum Heart Rate Achieved	Continuous
9.	Exang	Exercise Induced Angina	Yes/No
10.	Oldpeak	ST Depression when Workout compared to the Amount of Rest Taken	Continuous
11.	Slope	Slope of Peak Exercise ST segment	up/ Flat /Down
12.	Ca	Gives the number of Major Vessels Coloured by Fluoroscopy	0-3
13.	Thal	Defect Type	Reversible/Fixed/Normal
14.	Num(Disorder)	Heart Disease	Not Present /Present in the Four Major types.

Table no 3.1: Various attributes

## **CHAPTER 4**

### **METHODOLOGY**

Heart disease is a significant health concern worldwide, with millions of people suffering from this condition. Accurate diagnosis of heart disease is crucial for timely intervention and treatment. In recent years, machine learning algorithms have emerged as promising tools for predicting and diagnosing heart disease. In this paper, we present an analysis of various machine learning algorithms, including K nearest neighbors (KNN), Logistic Regression, and Random Forest Classifiers, which can be helpful for practitioners or medical analysts in accurately diagnosing heart disease.

The methodology of this study provides a framework for the proposed model. The methodology is a systematic process that includes several steps to transform the given data into recognized data patterns for the knowledge of users. The first step in the proposed methodology is the collection of data related to cardiovascular disease from various sources, including journals, published papers, and recent data sets. The second step involves extracting significant values from the collected data, which are relevant for predicting heart disease. These values may include medical parameters such as chest pain, fasting sugar, blood pressure, cholesterol, age, sex, and others.

The third step in the methodology is data preprocessing, which deals with handling missing values, cleaning the data, and normalizing the data depending on the algorithms used. Missing values in the data set can adversely affect the accuracy of the prediction models, and hence, appropriate techniques such as imputation or deletion need to be employed to handle them effectively. Data cleaning involves removing any inconsistencies or errors in the data set, such as duplicate entries, outliers, or inconsistent values. The choice of normalization technique may vary depending on the characteristics of the data and the algorithms used.

After data preprocessing, the next step in the proposed methodology is the use of classifiers to classify the pre-processed data. Classifiers are machine learning algorithms that are trained on a labeled data set to learn patterns and relationships between input features and output labels. The classifiers used in the proposed model include KNN, Logistic Regression, and Random Forest Classifier. KNN is a simple and intuitive algorithm that classifies data points based on the majority of their k-nearest neighbors. Logistic Regression is a widely used algorithm for binary classification that models the probability of an input belonging to a particular class. Random Forest Classifier is an ensemble learning algorithm that combines multiple decision trees to make predictions.

Finally, the proposed model is undertaken, where we evaluate the performance of the model on the basis of accuracy and other performance metrics. Accuracy is a commonly used metric to measure the performance of a classification model, which indicates the percentage of correctly predicted instances out of the total instances. Other performance metrics such as precision, recall, F1-score, and area under the Receiver Operating Characteristic (ROC) curve may also be used to assess the performance of the model. Precision measures the percentage of true positive predictions out of the total positive predictions, while recall measures the percentage of true positive predictions out of the actual positive instances. F1-score is the harmonic mean of precision and recall, and it provides a balanced measure of the model's performance. The ROC curve plots the true positive rate against the false positive rate, and the area under the curve indicates the performance of the model in distinguishing between positive and negative instances.

In this proposed model, an effective Heart Disease Prediction System (EHDPS) has been developed using different classifiers, including KNN, Logistic Regression, and Random Forest Classifier. These parameters are considered relevant in predicting heart disease based on previous research and medical knowledge. The model is trained on a labeled data set that includes instances with known heart disease.

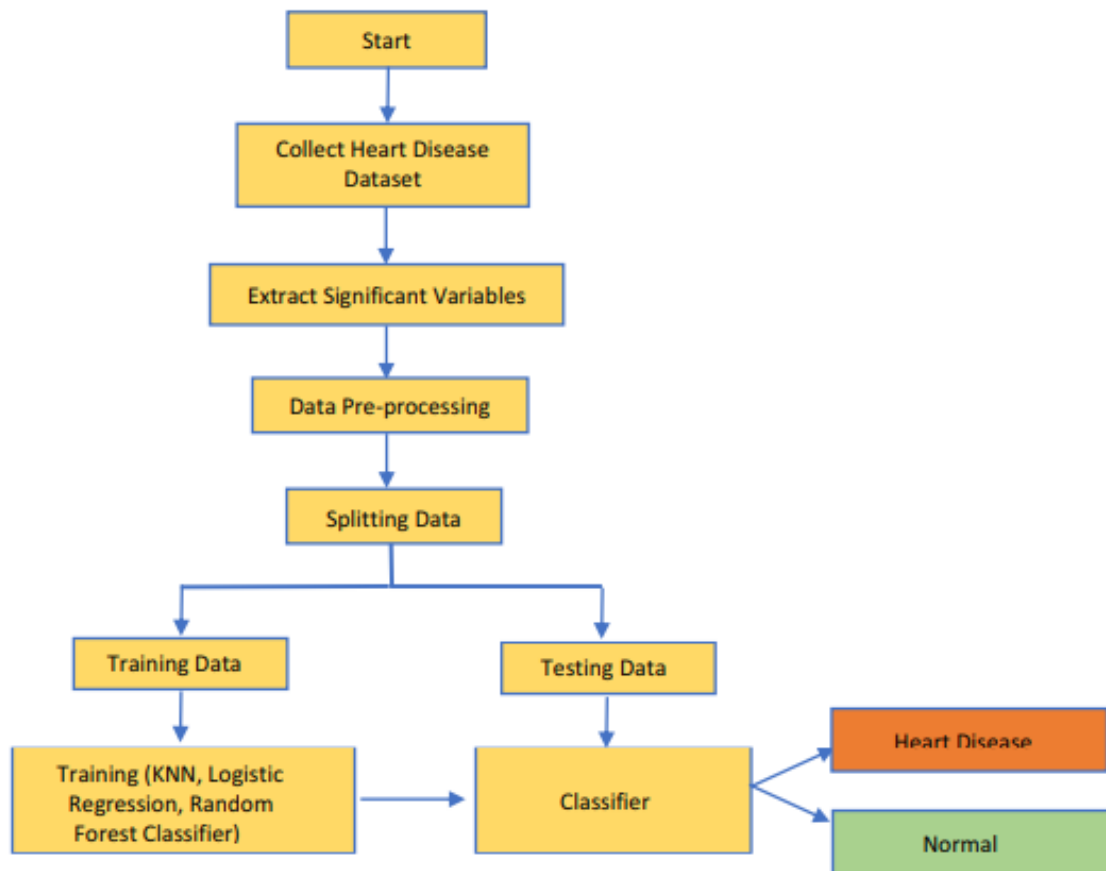


Fig no 4.1 Methodology

The present dataset is collected from Kaggle.



## 4.1 Data Preprocessing:

After selecting the raw data for ML training, the most important task is data pre-processing. In broad sense, data preprocessing will convert the selected data into a form we can work with or can feed to ML algorithms. We always need to preprocess our data so that it can be as per the expectation of machine learning algorithm.

### Data Pre-processing Techniques

We have the following data preprocessing techniques that can be applied on data set to produce data for ML algorithms –

#### 4.1.1 Scaling

Most probably our dataset comprises of the attributes with varying scale, but we cannot provide such data to ML algorithm hence it requires rescaling. Data rescaling makes sure that attributes are at same scale. Generally, attributes are rescaled into the range of 0 and 1. ML algorithms like gradient descent and k-Nearest Neighbors requires scaled data. We can rescale the data with the help of MinMaxScaler class of scikit-learn Python library.

#### Example

In this example we will rescale the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded (as done in the previous chapters) and then with the help of MinMaxScaler class, it will be rescaled in the range of 0 and 1.

The first few lines of the following script are same as we have

written in previous chapters while loading CSV data.

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn import preprocessing
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use MinMaxScaler class to rescale the data in the range of 0 and 1.

```
data_scaler=
preprocessing.MinMaxScaler(feature_range=(0,1))
data_rescaled = data_scaler.fit_transform(array)
```

We can also summarize the data for output as per our choice. Here, we are setting the precision to 1 and showing the first 10 rows in the output.

```
set_printoptions(precision=1)
print ("\nScaled data:\n", data_rescaled[0:10])
```

## Output

Scaled data:

```
[
[0.4 0.7 0.6 0.4 0. 0.5 0.2 0.5 1. ]
[0.1 0.4 0.5 0.3 0. 0.4 0.1 0.2 0. ]
[0.5 0.9 0.5 0. 0. 0.3 0.3 0.2 1. ]
[0.1 0.4 0.5 0.2 0.1 0.4 0. 0. 0. ]
```

```
[0. 0.7 0.3 0.4 0.2 0.6 0.9 0.2 1. ]
[0.3 0.6 0.6 0. 0. 0.4 0.1 0.2 0. ]
[0.2 0.4 0.4 0.3 0.1 0.5 0.1 0.1 1. ]
[0.6 0.6 0. 0. 0. 0.5 0. 0.1 0. ]
[0.1 1. 0.6 0.5 0.6 0.5 0. 0.5 1. ]
[0.5 0.6 0.8 0. 0. 0. 0.1 0.6 1. ]
]
```

From the above output, all the data got rescaled into the range of 0 and 1.

#### **4.1.2 Normalization**

Another useful data preprocessing technique is Normalization. This is used to rescale each row of data to have a length of 1. It is mainly useful in Sparse dataset where we have lots of zeros. We can rescale the data with the help of Normalizer class of scikit-learn Python library.

#### **Types of Normalization**

In machine learning, there are two types of normalization preprocessing techniques as follows –

##### **L1 Normalization**

It may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the absolute values will always be up to 1. It is also called Least Absolute Deviations.

### Example

In this example, we use L1 Normalize technique to normalize the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded and then with the help of Normalizer class it will be normalized.

The first few lines of following script are same as we have written in previous chapters while loading CSV data.

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn.preprocessing import Normalizer
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']
dataframe = read_csv (path, names=names)
array = dataframe.values
```

Now, we can use Normalizer class with L1 to normalize the data.

```
Data_normalizer = Normalizer(norm='l1').fit(array)
Data_normalized = Data_normalizer.transform(array)
```

We can also summarize the data for output as per our choice. Here, we are setting the precision to 2 and showing the first 3 rows in the output.

```
set_printoptions(precision=2)
print ("\nNormalized data:\n", Data_normalized [0:3])
```

**Output**

Normalized data:

```
[  
[0.02 0.43 0.21 0.1 0. 0.1 0. 0.14 0.]  
[0. 0.36 0.28 0.12 0. 0.11 0. 0.13 0.]  
[0.03 0.59 0.21 0. 0. 0.07 0. 0.1 0.]  
]
```

## L2 Normalization

It may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the squares will always be up to 1. It is also called least squares.

### Example

In this example, we use L2 Normalization technique to normalize the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded (as done in previous chapters) and then with the help of Normalizer class it will be normalized.

The first few lines of following script are same as we have written in previous chapters while loading CSV data.

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn.preprocessing import Normalizer
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use Normalizer class with L1 to normalize the data.

```
Data_normalizer = Normalizer(norm='l2').fit(array)
Data_normalized = Data_normalizer.transform(array)
```

We can also summarize the data for output as per our choice.

Here, we are setting the precision to 2 and showing the first 3 rows in the output.

```
set_printoptions(precision=2)
print ("\nNormalized data:\n", Data_normalized [0:3])
```

### Output

```
Normalized data:
[
[0.03 0.83 0.4  0.2  0. 0.19 0. 0.28 0.01]
[0.01 0.72 0.56 0.24 0. 0.22 0. 0.26 0. ]
[0.04 0.92 0.32 0.  0. 0.12 0. 0.16 0.01]
]
```

#### 4.1.3 Binarization

As the name suggests, this is the technique with the help of which we can make our data binary. We can use a binary threshold for making our data binary. The values above that threshold value will be converted to 1 and below that threshold will be converted to 0. For example, if we choose threshold value = 0.5, then the dataset value above it will become 1 and below this will become 0. That is why we can call it **binarizing** the data or **thresholding** the data. This technique is useful when we have probabilities in our dataset and want to convert them into crisp values.

We can binarize the data with the help of Binarizer class of scikit-learn Python library.

### Example

In this example, we will rescale the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded and then with the help of Binarizer class it will

be converted into binary values i.e. 0 and 1 depending upon the threshold value. We are taking 0.5 as threshold value.

The first few lines of following script are same as we have written in previous chapters while loading CSV data.

```
from pandas import read_csv
from sklearn.preprocessing import Binarizer
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use Binarize class to convert the data into binary values.

```
binarizer = Binarizer(threshold=0.5).fit(array)
Data_binarized = binarizer.transform(array)
```

Here, we are showing the first 5 rows in the output.

```
print ("\nBinary data:\n", Data_binarized [0:5])
```

### Output

```
Binary data:
[
[1. 1. 1. 1. 0. 1. 1. 1. 1.]
[1. 1. 1. 1. 0. 1. 0. 1. 0.]
[1. 1. 1. 0. 0. 1. 1. 1. 1.]
[1. 1. 1. 1. 1. 1. 0. 1. 0.]
[0. 1. 1. 1. 1. 1. 1. 1. 1.]
]
```



#### 4.1.4 Standardization

Another useful data preprocessing technique which is basically used to transform the data attributes with a Gaussian distribution. It differs the mean and SD (Standard Deviation) to a standard Gaussian distribution with a mean of 0 and a SD of 1. This technique is useful in ML algorithms like linear regression, logistic regression that assumes a Gaussian distribution in input dataset and produce better results with rescaled data. We can standardize the data (mean = 0 and SD = 1) with the help of StandardScaler class of scikit-learn Python library.

#### Example

In this example, we will rescale the data of Pima Indians Diabetes dataset which we used earlier. First, the CSV data will be loaded and then with the help of StandardScaler class it will be converted into Gaussian Distribution with mean = 0 and SD = 1.

The first few lines of following script are same as we have written in previous chapters while loading CSV data.

```
from sklearn.preprocessing import StandardScaler
from pandas import read_csv
from numpy import set_printoptions
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

Now, we can use StandardScaler class to rescale the data.

```
data_scaler = StandardScaler().fit(array)
data_rescaled = data_scaler.transform(array)
```

We can also summarize the data for output as per our choice. Here, we are setting the precision to 2 and showing the first 5 rows in the output.

```
set_printoptions(precision=2)
print ("\nRescaled data:\n", data_rescaled [0:5])
```

### Output

```
Rescaled data:
[
[ 0.64  0.85  0.15  0.91 -0.69  0.2   0.47  1.43  1.37]
[-0.84 -1.12 -0.16  0.53 -0.69 -0.68 -0.37 -0.19 -0.73]
[ 1.23  1.94 -0.26 -1.29 -0.69 -1.1   0.6  -0.11  1.37]
[-0.84 -1.   -0.16  0.15  0.12 -0.49 -0.92 -1.04 -0.73]
[-1.14  0.5  -1.5   0.91  0.77  1.41  5.48 -0.02  1.37]
]
```

## 4.2 TRAINING AND TESTING:

The construction of ML algorithms is dependent on how data is collected. Moreover, three categories are most frequently used to categorize the information gathered.

The machine learning method creates algorithms using three data sets:

- Training data
- Validation data
- Test data

### What is Training Data and its Use in Machine Learning?

Training data is used to train a model to predict an expected outcome. So, the algorithm's design focuses on creating the expected or predicted result.

Training data teaches an algorithm to extract relevant aspects of the outcome. It is usually the initial dataset used to program an algorithm on how to use various features, characteristics, and technologies to achieve the desired result.

Learning is a process, not a one-time effort. Humans are natural learners as they learn more from real-life examples. To make a machine understand and learn, they require patterns in data. It is easy for us humans to understand from one or maybe two examples, but the computer needs a lot of models since they work differently than we do. Machines have their language, so to train a machine, it must be done in a way a machine understands, for example, by using structured programming languages.

Take an example, teaching a child to identify a car is fairly easy by showing them an image of a car. In the case of machines, you'll need to show thousands of images of different cars for them to identify a car accurately.

Training and test data are essential components of machine learning. Machine learning algorithms learn from data, so having the correct data is critical to building successful models. Training and test data sets help us evaluate our models' performance and provide insights into how they work.

Let's discuss testing data sets, another concept you should understand when addressing how to train ML models. Training data and test data sets are two distinct yet critical elements of machine learning. While training data is required to educate an ML algorithm, testing data allows you to assess the progress of the algorithm's training and change or optimize it for better results.

In other words, when gathering data to train your algorithm, remember that some data will be needed to evaluate how well the training is going. This means that your data will be split into two parts: training and testing.

The most typical technique for data splitting is classifying 80% of the data as the training data set, and the remaining 20% will make up the testing data set. This is known as Pareto Principle or the "80/20 rule".

### Possible Issue While Training the Datasets

Now that we are done with two training and test datasets, a question arises. How do we deal with the errors made by our algorithm while trying to improve during the training process?

A good training session requires passing the data through our algorithm multiple times, meaning we will get the same patterns each time. To avoid this, we use another dataset to see different patterns in the underprocess data. This only calls for one action to be done. The apparent solution is to split your data set again. This is done only to the training data while designating a part of it for validation. This validation will assist your model in filling in the voids that it might have missed earlier and improve faster.

Always ensure that your test dataset fits the following requirements:

- It is large enough to produce significant statistical findings.
- It represents the entire data set. Thus, avoid selecting a test set with characteristics different from those in the training set.

### How Is Training Data Used in Machine Learning?

Machine Learning techniques allow machines to forecast and solve issues based on previous observations or experiences. These are the experiences or observations an algorithm can derive from the training data provided to it. Moreover, ML algorithms can learn and develop independently over time since they're trained with the appropriate training data.

Once the model has been sufficiently trained with the required training data, it is tested using the test data. The entire training and testing procedure can be broken down into three steps, which are as follows:

- Feed: First, we must train the model by giving it training data.
- Define: In Supervised Learning, training data is now tagged with the corresponding outputs, and the model transforms the training data into text vectors or various data features.
- Test: In the final phase, we put the model to the test by feeding it

test data or an unfamiliar dataset. This stage ensures that the model is efficiently trained and widely applicable.

Traits of Quality Training Data. It is critical to train the model using high-quality data since an ML model's forecast ability greatly depends on how it has been taught. ML also works with the principle of "Garbage In, Garbage Out." The model will make predictions based on whatever data we feed it. The following factors should be considered for high-quality training data:

### **Relevance**

The primary quality of training data should be relevant to the problem being solved. Any data you use should be relevant to the current challenge. For example, if you are developing a model to assess social media data, data should be gathered from various social media platforms such as Twitter, Facebook, and Instagram.

### **Standardization**

The features of a dataset should always be consistent. That means that all data for a specific problem should come from the same source and have the same qualities.

### **Uniformity**

To guarantee uniformity in the dataset, similar characteristics must always correspond to a similar label.

### **Comprehensive**

The training data must be large enough to reflect all the features required to train the model more effectively. The model will be able to learn all of the edge cases with a large dataset.

## How Much Training Data Is Required?

In machine learning, it's safe to say that the more data, the better the model. This is because the more you train your model, the smarter it'll become. However, if the data is properly prepared, follows a basic data prep checklist, and is ready for machine learning, you will still get accurate results.

Some machines require at least 1,000 records to construct a model. Yet, the accuracy of the data is critical. The industry follows an unwritten rule to construct a dependable model: use 1,000 poor data plus X number of good ones. For example, 1,000 non-performing loans plus X successfully repaid debts. Nevertheless, this is only an estimate. The number of documents required for the specific scenario can only be established by evaluating various possibilities. According to some experiences, developing a solid model with only 100 records is feasible, while in certain circumstances, over 30,000 records are required.

### **Factors affecting the quality of training data:**

To put it simply, the labeled data will influence how intelligent your model can become. It's similar to how a human only exposed to adolescent-level reading would fail to understand complex, university-level literature. But, there are three additional aspects to consider while training your machine learning models:

- **People:** The humans who train the model substantially impact its accuracy or performance. If they are biased, this will naturally affect how they label data and, as a result, how the ML model performs.
- **Processes:** Strict quality control checks must be in place during the data labeling process. This will greatly improve the quality of training data.
- **Tools:** Incompatible or out-of-date tools can degrade data quality. Utilizing powerful data labeling software will help you save money and time.

**Final Steps: Testing**

When you've trained your machine learning model, it's time to test it with the 20% of your training data set that you saved. Now is the time to fine-tune the model and ensure it functions properly. Do not be disheartened if it does not: we have yet to find a situation with no problems during the earliest stages of building the model.

Moreover, be prepared to tweak and develop the machine-learning model long after it goes live. Each algorithm is like a living thing that needs to be fed and cared for to function effectively and produce the greatest outcomes. Handle it well, and it will respond by providing you with accurate results.



## CHAPTER 5

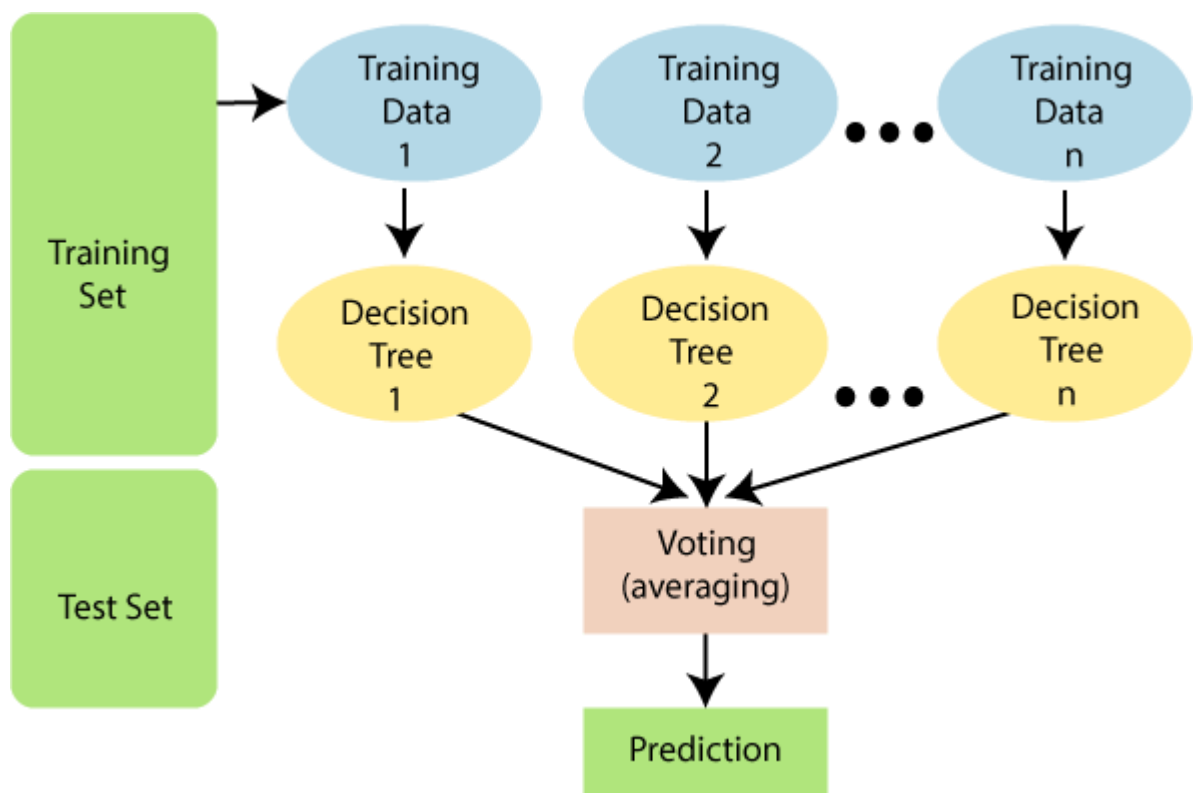
### ALGORITHMS

#### 5.1 THE RANDOM FOREST CLASSIFIER:

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

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- o There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- o The predictions from each tree must have very low correlations.



**Figure 6 :** Working of the Random Forest algorithm

Below are some points that explain why we should use the Random Forest algorithm:

<="" li="">

- o It takes less training time as compared to other algorithms.
- o It predicts output with high accuracy, even for the large dataset it runs efficiently.

- o It can also maintain accuracy when a large proportion of data is missing.

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase. The Working process can be explained in the below steps:

- o **Step-1:** Select random K data points from the training set.
- o **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- o **Step-3:** Choose the number N for decision trees that you want to build.
- o **Step-4:** Repeat Step 1 & 2.
- o **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

### **APPLICATIONS FOR RANDOM FOREST:**

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

**ADVANTAGES:**

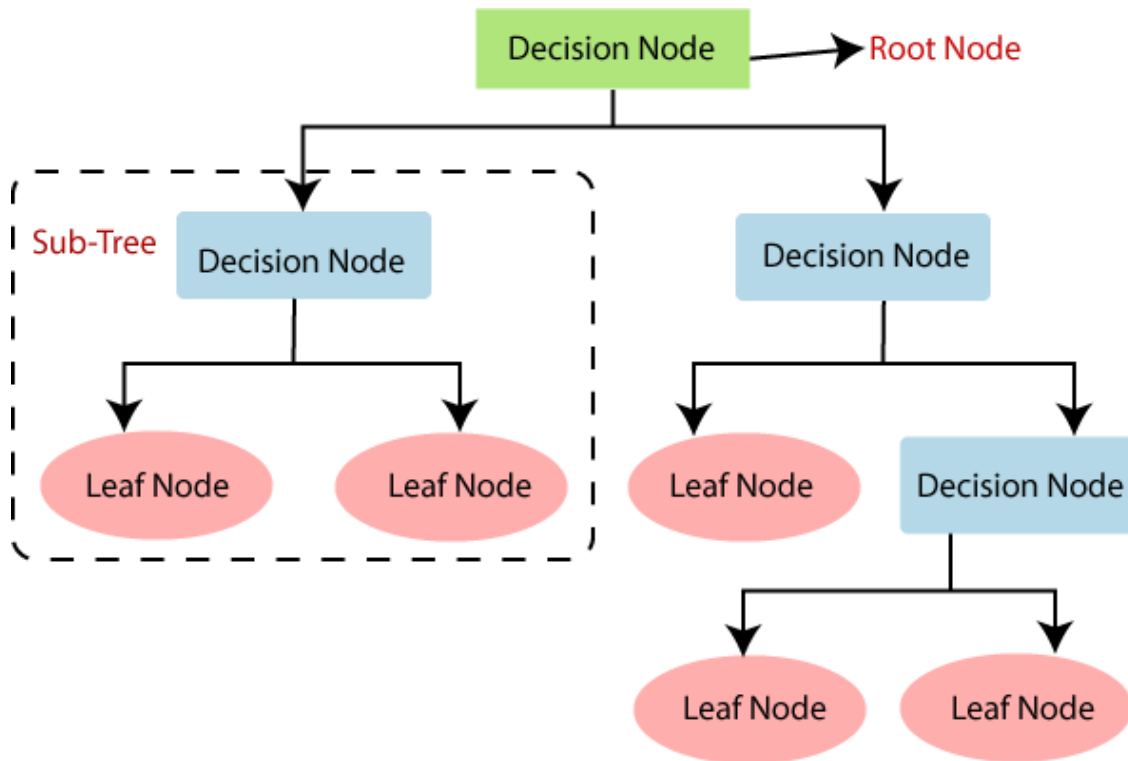
- o Random Forest is capable of performing both Classification and Regression tasks.
- o It is capable of handling large datasets with high dimensionality.
- o It enhances the accuracy of the model and prevents the overfitting issue.

**DISADVANTAGES:**

- o Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

## **5.2 DECISION TREE:**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.



**Figure 7 :** General structure of a decision tree

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree. Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand. The logic behind the decision tree can be easily understood because it shows a tree-like structure.

### **DECISION TREE TERMINOLOGIES:**

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given

conditions.

- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.

**Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM)**.
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

**ADVANTAGES:**

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

**DISADVANTAGES:**

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

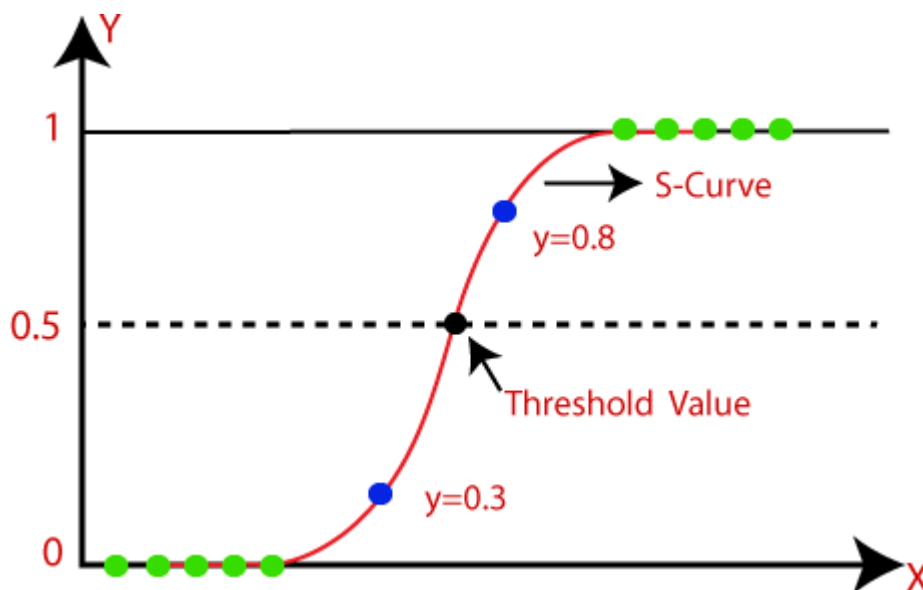
**5.3 LOGISTIC REGRESSION:**

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).



The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function.

**LOGISTIC FUNCTION:** The sigmoid function is a mathematical function used to map the predicted values to probabilities. It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form.



The S-form curve is called the Sigmoid function or the logistic function. In logistic

regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

### **TYPES OF LOGISTIC REGRESSION:**

- 6 Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

**Example:** There is a dataset given which contains the information of various users obtained from the social networking sites. There is a car making company that has recently launched a new SUV car. So the company wanted to check how many users from the dataset, wants to purchase the car.

For this problem, we will build a Machine Learning model using the Logistic regression algorithm. The dataset is shown in the below image. In this problem, we will predict the **purchased variable (Dependent Variable)** by using **age and salary (Independent variables)**.

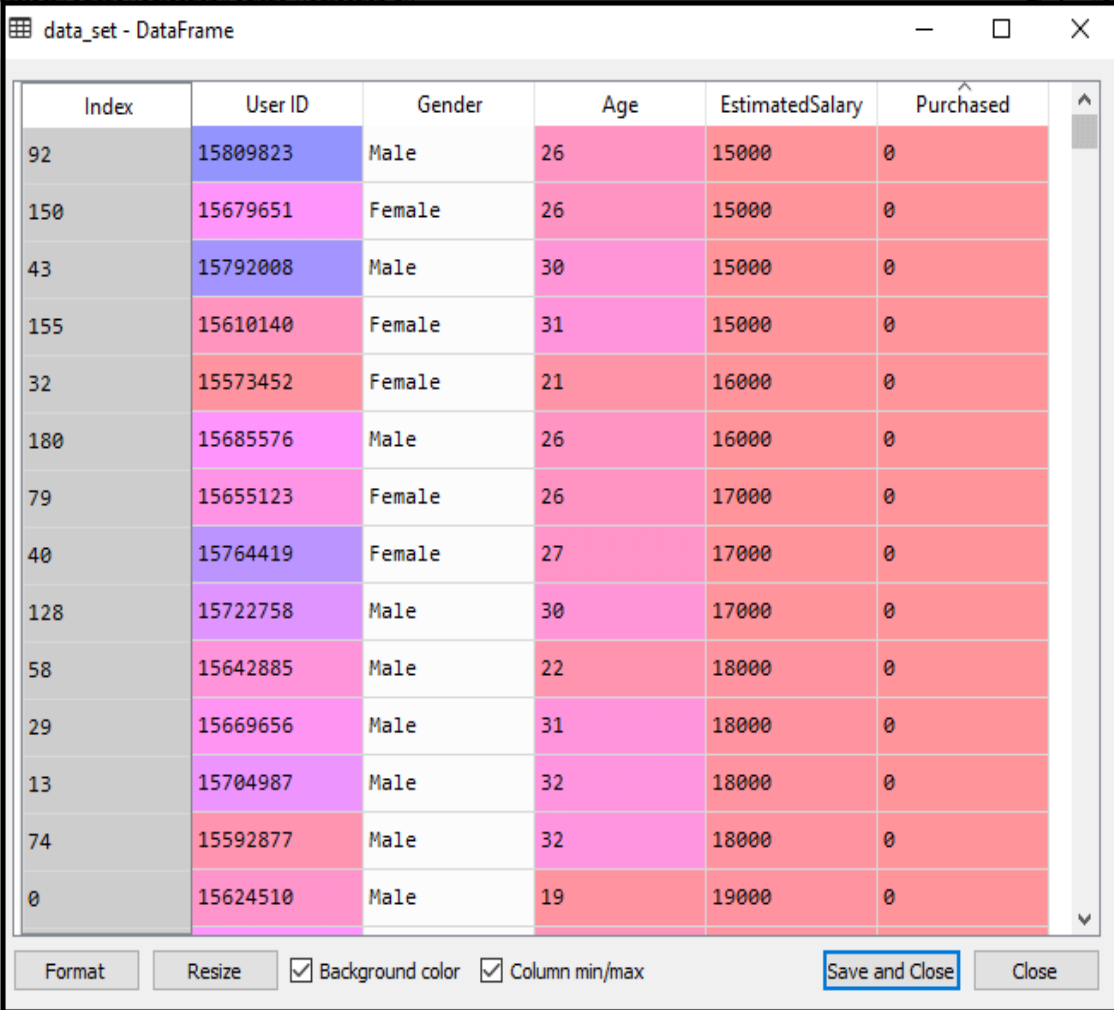
User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

### Steps in Logistic Regression:

To implement the Logistic Regression using Python, we will use the same steps as we have done in previous topics of Regression. Below are the steps:

- 7 Data Pre-processing step
- 8 Fitting Logistic Regression to the Training set
- 9 Predicting the test result
- 10 Test accuracy of the result(Creation of Confusion matrix)
- 11 Visualizing the test set result.

**Data Pre-processing step:** In this step, we will pre- process/prepare the data so that we can use it in our code efficiently. It will be the same as we have done in Data pre- processing topic.



Index	User ID	Gender	Age	EstimatedSalary	Purchased
92	15809823	Male	26	15000	0
150	15679651	Female	26	15000	0
43	15792008	Male	30	15000	0
155	15610140	Female	31	15000	0
32	15573452	Female	21	16000	0
180	15685576	Male	26	16000	0
79	15655123	Female	26	17000	0
40	15764419	Female	27	17000	0
128	15722758	Male	30	17000	0
58	15642885	Male	22	18000	0
29	15669656	Male	31	18000	0
13	15704987	Male	32	18000	0
74	15592877	Male	32	18000	0
0	15624510	Male	19	19000	0

### Fitting Logistic Regression to the Training set:

We have well prepared our dataset, and now we will train the dataset using the training set. For providing training or fitting the model to the training set, we will import the **Logistic Regression** class of the **sklearn** library.

After importing the class, we will create a classifier object and use it to fit the model to the logistic regression.

### Predicting the Test Result:

Our model is well trained on the training set, so we will now predict the result by using test set data.

### Test Accuracy of the result:

Now we will create the confusion matrix here to check the accuracy of the classification. To create it, we need to import the **confusion matrix** function of the sklearn library. After importing the function, we will call it using a new variable **cm**. The function takes two parameters, mainly **y true** (the actual values) and **y\_pred** (the targeted value return by the classifier).

### Visualizing the training set result

Finally, we will visualize the training set result. To visualize the result, we will use **Listed Colormap** class of matplotlib library.



**12 Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep".

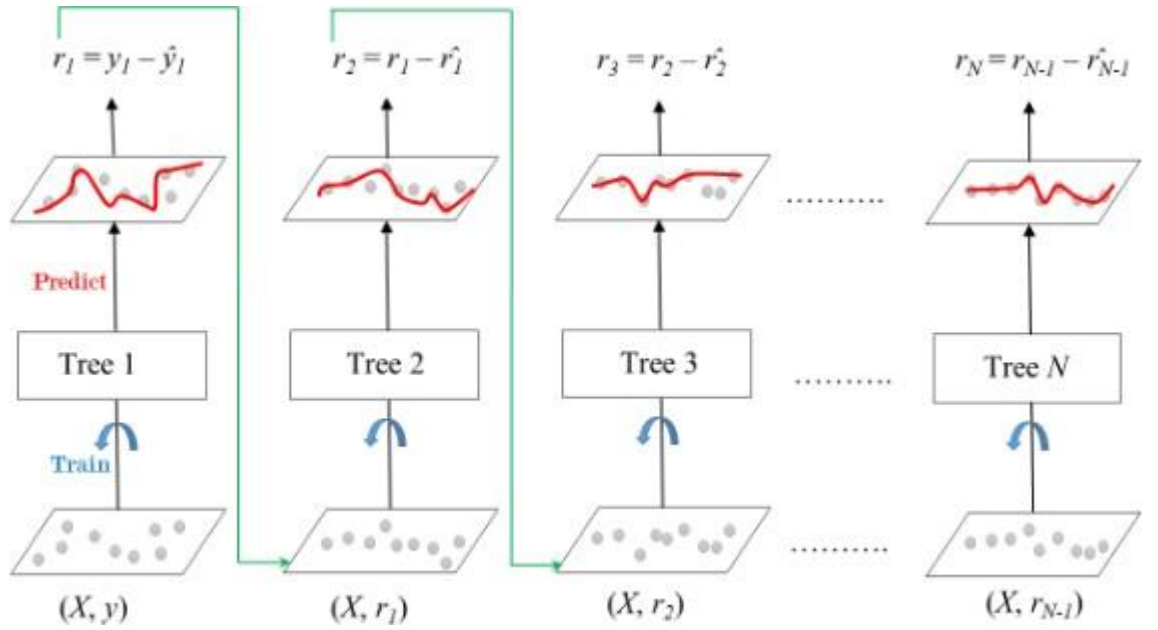
**13 Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low",

"Medium", or "High".

#### **5.4 Gradient Boosting Classifier:**

Gradient boosting is a powerful ensemble machine learning algorithm. It's popular for structured predictive modeling problems, such as classification and regression on tabular data, and is often the main algorithm or one of the main algorithms used in winning solutions to machine learning competitions, like those on Kaggle. There are many implementations of gradient boosting available, including standard implementations in SciPy and efficient third-party libraries. Each uses a different interface and even different names for the algorithm. In this tutorial, you will discover how to use gradient boosting models for classification and regression in Python. Standardized code examples are provided for the four major implementations of gradient boosting in Python, ready for you to copy-paste and use in your own predictive modeling project. Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

In contrast to AdaBoost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of the predecessor as labels. There is a technique called the **Gradient Boosted Trees** whose base learner is CART (Classification and Regression Trees). The below diagram explains how gradient-boosted trees are trained for regression problems.



**FIGURE 8:** Gradient Boosted Trees for Regression

The ensemble consists of  $M$  trees. Tree1 is trained using the feature matrix  $X$  and the labels  $y$ . The predictions labeled  $\hat{y}_1$  are used to determine the training set residual errors  $r_1$ . Tree2 is then trained using the feature matrix  $X$  and the residual errors  $r_1$  of Tree1 as labels. The predicted results  $\hat{r}_1$  are then used to determine the residual  $r_2$ . The process is repeated until all the  $M$  trees forming the ensemble are trained. There is an important parameter used in this technique known

as **Shrinkage**. **Shrinkage** refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate ( $\eta$ ) which ranges between 0 to 1. There is a trade-off between  $\eta$  and the number of estimators, decreasing learning rate needs to be compensated with increasing estimators in order to reach certain model performance. Since all trees are trained now, predictions can be made. Each tree predicts a label and the final prediction is given by the formula, Gradient Boosting updates the weights by computing the negative gradient of the loss function with respect to the predicted output. Gradient Boosting can use a wide range of base learners, such as decision trees, and linear models. Gradient Boosting is generally more robust, as it updates the weights based on the gradients, which are less sensitive to outliers.

### **GRADIENT BOOSTING ALGORITHM:**

Example: 1 Classification

Steps:

- Import the necessary libraries
- Setting SEED for reproducibility
- Load the digit dataset and split it into train and test.
- Instantiate Gradient Boosting classifier and fit the model.
- Predict the test set and compute the accuracy score.

Example: 2 Regression

Steps:

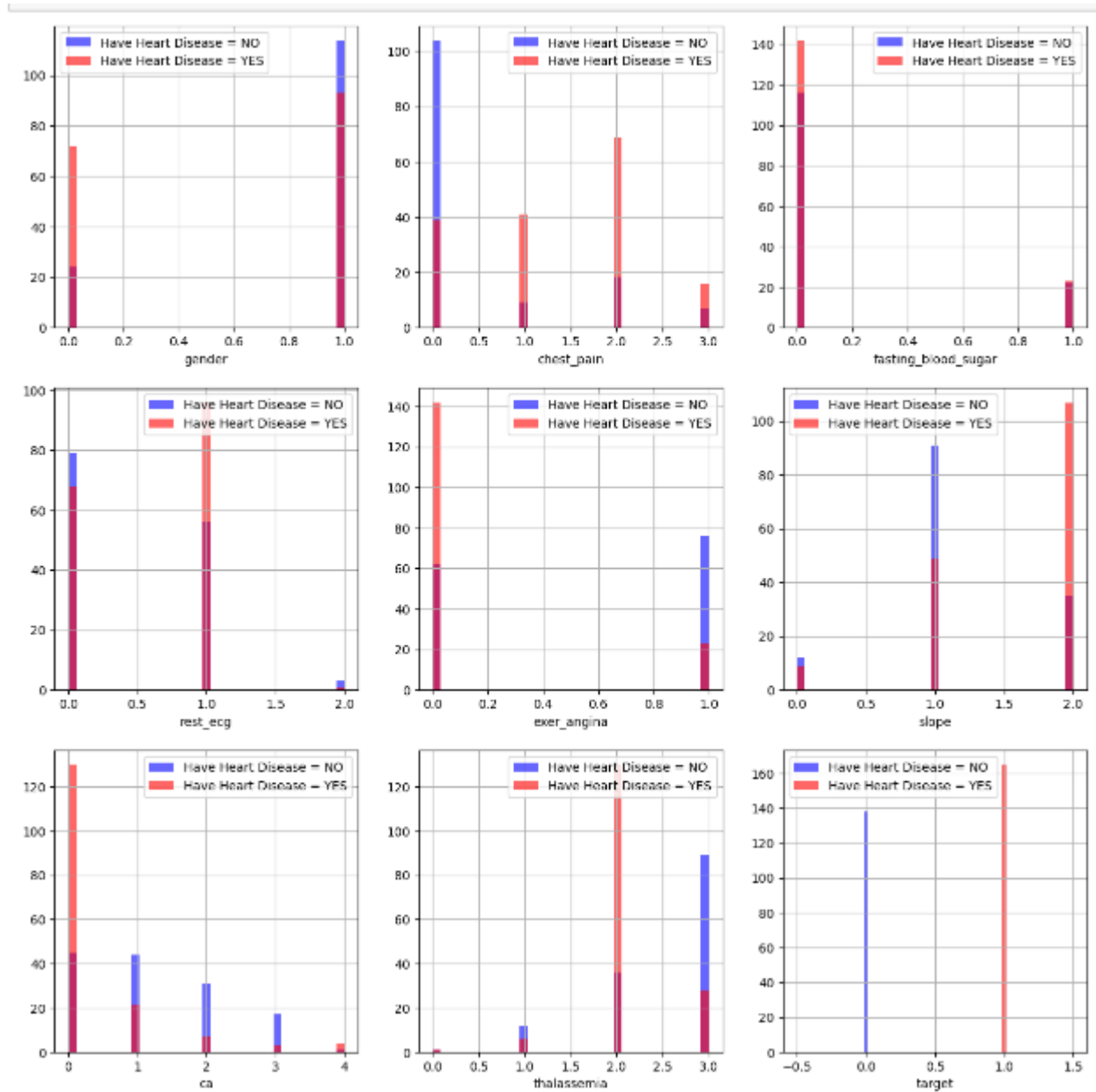
- Import the necessary libraries
- Setting SEED for reproducibility
- Load the diabetes dataset and split it into train and test.
- Instantiate Gradient Boosting Regressor and fit the model.
- Predict on the test set and compute RMSE.

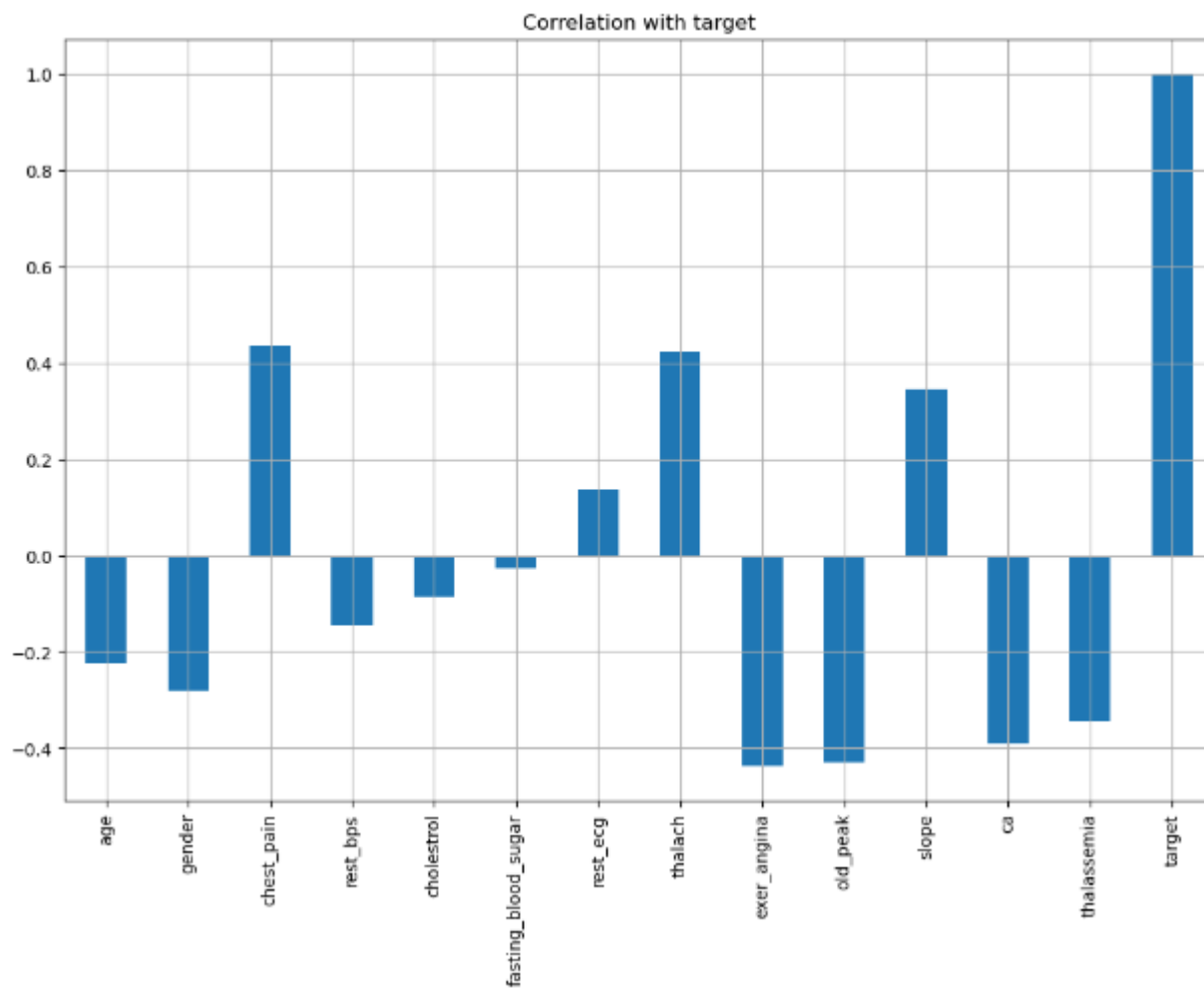


## CHAPTER 6

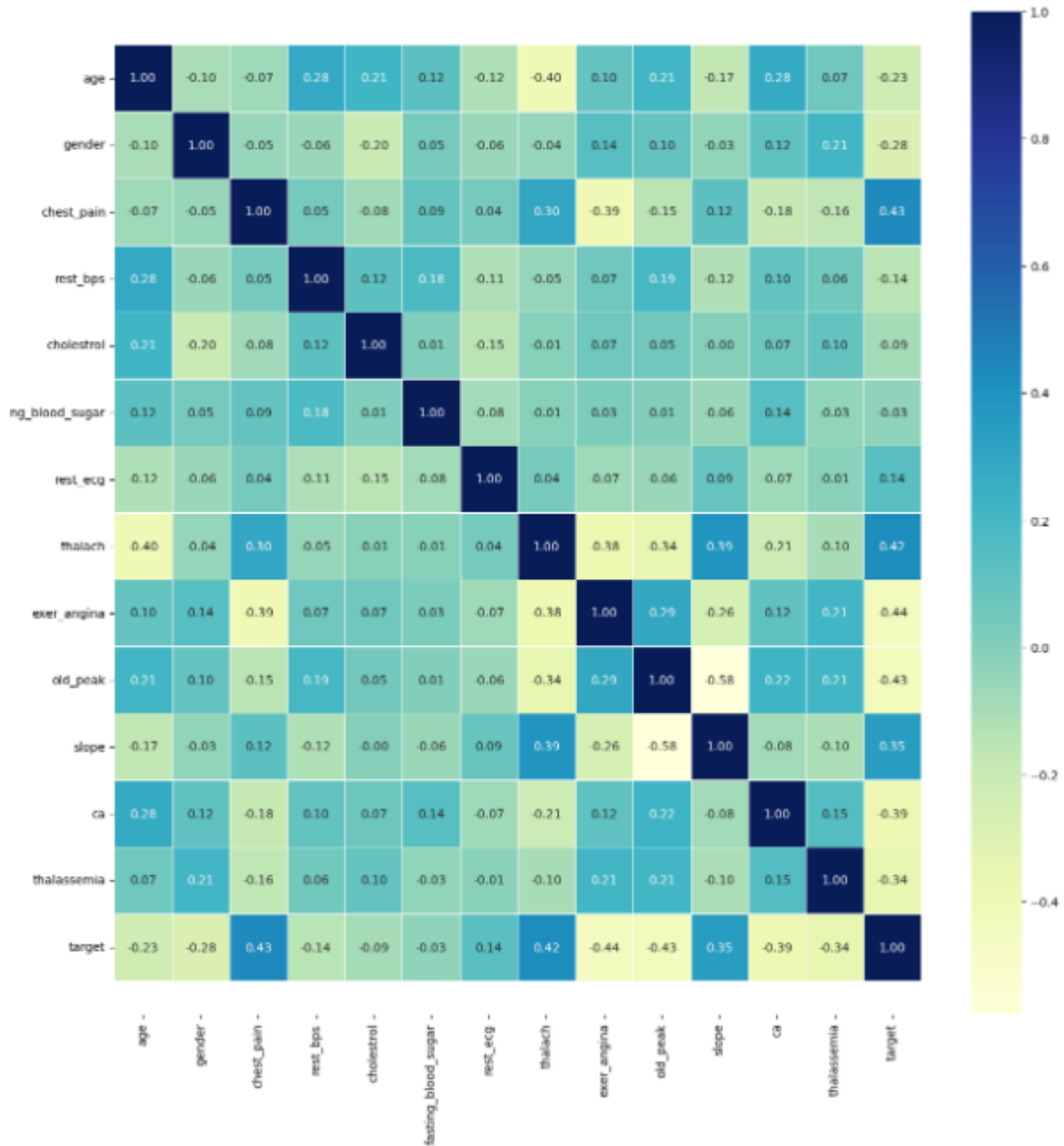
### RESULTS

**DETERMINING WHETHER A PERSON IS HAVING HEART DISEASE, BASED ON THE FACTORS:**



**Relation between factors and target:**

## Correlation Diagram



## Algorithms Accuracy:

### Logistic Regression

```
In [32]: lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train, y_train)

print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
```

Train Result:

=====

Accuracy Score: 87.68%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.88	0.87	0.88	0.88	0.88
recall	0.83	0.91	0.88	0.87	0.88
f1-score	0.86	0.89	0.88	0.87	0.88
support	109.00	133.00	0.88	242.00	242.00

Confusion Matrix:

```
[[ 91 18]
 [ 12 121]]
```

Test Result:

=====

Accuracy Score: 88.52%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.87	0.90	0.89	0.88	0.89
recall	0.90	0.88	0.89	0.89	0.89
f1-score	0.88	0.89	0.89	0.89	0.89
support	29.00	32.00	0.89	61.00	61.00

Confusion Matrix:

```
[[26  3]
 [ 4 28]]
```

### Support Vector Machine ¶

```
In [33]: svc_clf = SVC()
svc_clf.fit(X_train, y_train)

print_score(svc_clf, X_train, y_train, X_test, y_test, train=True)
print_score(svc_clf, X_train, y_train, X_test, y_test, train=False)
```

Train Result:

=====

Accuracy Score: 92.15%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.92	0.92	0.92	0.92	0.92
recall	0.90	0.94	0.92	0.92	0.92
f1-score	0.91	0.93	0.92	0.92	0.92
support	109.00	133.00	0.92	242.00	242.00

Confusion Matrix:

```
[[ 98 11]
 [  8 125]]
```

Test Result:

=====

Accuracy Score: 90.16%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.87	0.93	0.90	0.90	0.90
recall	0.93	0.88	0.90	0.90	0.90
f1-score	0.90	0.90	0.90	0.90	0.90
support	29.00	32.00	0.90	61.00	61.00

Confusion Matrix:

```
[[27  2]
 [ 4 28]]
```

## Naive Bayes

```
In [34]: GaussianNB_clf = GaussianNB()
GaussianNB_clf.fit(X_train, y_train)

print_score(GaussianNB_clf, X_train, y_train, X_test, y_test, train=True)
print_score(GaussianNB_clf, X_train, y_train, X_test, y_test, train=False)
```

Train Result:

Accuracy Score: 54.96%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.50	1.00	0.55	0.75	0.77
recall	1.00	0.18	0.55	0.59	0.55
f1-score	0.67	0.11	0.55	0.49	0.47
support	109.00	133.00	0.55	242.00	242.00

Confusion Matrix:

```
[[109  0]
 [  0 24]]
```

Test Result:

Accuracy Score: 49.18%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.48	0.60	0.49	0.54	0.54
recall	0.93	0.09	0.49	0.51	0.49
f1-score	0.64	0.16	0.49	0.48	0.39
support	29.00	32.00	0.49	61.00	61.00

Confusion Matrix:

```
[[27  2]
 [29  3]]
```

## Decision Tree

```
In [35]: dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)

print_score(dt_clf, X_train, y_train, X_test, y_test, train=True)
print_score(dt_clf, X_train, y_train, X_test, y_test, train=False)
```

Train Result:

Accuracy Score: 100.00%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	1.00	1.00	1.00	1.00	1.00
recall	1.00	1.00	1.00	1.00	1.00
f1-score	1.00	1.00	1.00	1.00	1.00
support	109.00	133.00	1.00	242.00	242.00

Confusion Matrix:

```
[[109  0]
 [  0 133]]
```

Test Result:

Accuracy Score: 86.89%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.84	0.90	0.87	0.87	0.87
recall	0.90	0.84	0.87	0.87	0.87
f1-score	0.87	0.87	0.87	0.87	0.87
support	29.00	32.00	0.87	61.00	61.00

Confusion Matrix:

```
[[26  3]
 [ 5 27]]
```

From the above results it is clear that Support Vector Machine is the most suitable or preferred classifier as the train accuracy is 92.5% and the test accuracy is 90.16% which is better when compared with other classifiers or algorithms.

### Prediction results:

```
In [38]: new = X_train.iloc[78]
a = np.asarray(new)
a = a.reshape(1,-1)
p = svc_clf.predict(a)
```

```
In [39]: p[0]
```

```
Out[39]: 0
```

```
In [40]: if (p[0] == 1):
          print("Person has heart disease")
        else:
          print("Great! the results are normal")
```

```
Great! the results are normal
```

Here the variable `p` is considered as the final prediction variable and the total prediction message will only depend on the value. Here an if else statement is created where the prediction variable `p` is considered as the main parameter. The Parameter is compared with the binary digits 0 and 1. If the value of `p` is 1 then the message that will appear is “Person has Diabetes and is at risk of dying”. If the value of `p` is 0 the message that will appear will be “Great! The result is negative and you don’t have to worry”.

## Chapter 7

### Advantages

Heart disease is a serious and common medical condition that affects millions of people worldwide. Machine learning has the potential to aid in the prediction of heart disease by identifying patterns in large amounts of data. Here are the general steps for developing an application for heart disease prediction using machine learning:

**Data Collection:** Collect a large amount of relevant data from sources such as medical records, wearable devices, or surveys.

**Data Preprocessing:** Clean the collected data, remove missing values, and convert the data into a suitable format for machine learning algorithms.

**Feature Selection:** Identify the most important features (e.g. age, blood pressure, cholesterol levels, family history) that contribute to the development of heart disease.

**Algorithm Selection:** Choose a suitable machine learning algorithm(s) for the prediction of heart disease based on the available data and problem statement. Some popular algorithms for heart disease prediction include logistic regression, decision trees, random forests, and neural networks.

**Model Training:** Split the data into training and testing sets, and use the training set to train the selected machine learning algorithm.

**Model Evaluation:** Evaluate the performance of the trained model using the testing set, and tune the model parameters as needed to improve its accuracy.

**Deployment:** Deploy the trained model in a user-friendly interface, such as a web application or mobile app, to allow users to input their data and obtain a prediction of their risk for heart disease.

It is important to note that the development of a heart disease prediction application using machine learning requires expertise in data science, machine learning, and software engineering. Collaboration between medical professionals and data scientists is also necessary to ensure the accuracy and reliability of the application. Additionally, the application must comply with regulations related to medical data privacy and security.



## **Chapter 8**

### **Future Scope**

Heart disease is a leading cause of mortality worldwide, and predicting the likelihood of its occurrence can play a significant role in its prevention and management. Machine learning techniques have shown promise in predicting the risk of heart disease based on various factors such as age, gender, medical history, lifestyle habits, and genetic predisposition. Here are some potential future directions for heart disease prediction using machine learning:

**Integration of multimodal data:** In addition to traditional risk factors, machine learning models can incorporate data from wearable devices, electronic health records, and imaging tests to improve the accuracy of heart disease prediction.

**Personalized risk assessment:** Machine learning models can be trained to provide personalized risk assessments for patients based on their unique characteristics and medical history. This approach can help healthcare providers identify patients who are at the highest risk of heart disease and tailor their treatment accordingly.

**Early detection of heart disease:** Machine learning models can be used to detect early signs of heart disease using non-invasive imaging techniques such as electrocardiography (ECG), echocardiography, and magnetic resonance imaging (MRI).

**Real-time monitoring:** Machine learning algorithms can be integrated with wearable devices and mobile applications to provide real-time monitoring of heart health, enabling early intervention and prevention of heart disease.

Integration with clinical decision support systems: Machine learning models can be integrated decision support systems to provide healthcare providers with real-time recommendations for the prevention, diagnosis, and treatment of heart disease.

Overall, the future scope for heart disease prediction using machine learning is vast, and it has the potential to significantly improve the accuracy of risk assessment, early detection, and prevention of heart disease.

## Chapter 9

### Applications

There are several applications of diabetes prediction using machine learning, including:

- **Early Detection:** Machine learning models can be trained on historical patient data to predict the risk of developing diabetes at an early stage. Early detection can lead to earlier interventions and lifestyle changes that can help prevent the onset of the disease or mitigate its severity.
- **Personalized Medicine:** Machine learning models can be used to predict the most effective treatment plan for individual patients. This can help doctors make more informed decisions about medication, dosage, and frequency of treatment.
- **Remote Monitoring:** Machine learning models can be used to monitor patients remotely, allowing healthcare professionals to track glucose levels and other vital signs in real-time. This can help detect fluctuations in glucose levels that may indicate the need for intervention or medication adjustments.
- **Public Health:** Machine learning models can be used to predict the incidence and prevalence of diabetes in specific populations. This can help public health officials allocate resources and design interventions to prevent and manage diabetes at the population level.
- **Data Analytics:** Machine learning models can be used to analyze large datasets of patient data, helping researchers identify risk factors and patterns that can lead to new insights and treatment options. Overall, the use of machine learning for diabetes prediction can lead to more personalized and effective care for patients, as well as better public health outcomes.

## Chapter 10

### Code

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

import sklearn
from sklearn import tree
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler

from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from matplotlib import rcParams
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier

df = pd.read_csv('HeartDisease.csv')
df.info()
df.head(80)
df['target'].value_counts()
pd.set_option("display.float", "{:.2f}".format)
df.describe()
df.target.value_counts().plot(kind="bar", color=["salmon", "lightblue"])
```

```

plt.xlabel('Patient has heart disease')
plt.ylabel('counts')
plt.title('Histogram of Patient has heart disease')
df.isna().sum()
categorical_val = []
continous_val = []
for column in df.columns:
    print('=====')
    print(f'{column} : {df[column].unique()}')
    if len(df[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
plt.figure(figsize=(15, 15))

for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease
= NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease
= YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
annot=True,
linewidths=0.5,
fmt=".2f",
cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)

```

```

f.corrwith(df.target).plot(kind='bar', grid=True, figsize=(12, 8),
                                title="Correlation with target")

num_val = df[['age', 'rest_bps', 'cholesterol', 'thalach', 'old_peak']]
sns.pairplot(num_val)
target_var = df['target']
independent_features = df.drop(columns = ['target'])
df = pd.get_dummies(independent_features, columns = ['gender', 'chest_pain',
'fasting_blood_sugar', 'rest_ecg', 'exer_angina', 'slope', 'ca', 'thalassemia'])
df.head()
sc = StandardScaler()
col_to_scale = ['age', 'rest_bps', 'cholesterol', 'thalach', 'old_peak']
df[col_to_scale] = sc.fit_transform(df[col_to_scale])
df.head()
X = df
y = target_var

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred,
output_dict=True))
        print("Train
Result:\n=====")
        print(f'Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%')
        print("_____")
        print(f'CLASSIFICATION REPORT:\n{clf_report}')
        print("_____")
        print(f'Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

```

```

elif train==False:
    pred = clf.predict(X_test)
clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
    print("Test
Result:\n=====")
    print(f'Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%')
    print("_____")
    print(f'CLASSIFICATION REPORT:\n{clf_report}')
    print("_____")
    print(f'Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n')
lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train, y_train)

print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
svc_clf = SVC()
svc_clf.fit(X_train, y_train)

print_score(svc_clf, X_train, y_train, X_test, y_test, train=True)
print_score(svc_clf, X_train, y_train, X_test, y_test, train=False)
GaussianNB_clf = GaussianNB()
GaussianNB_clf.fit(X_train, y_train)

print_score(GaussianNB_clf, X_train, y_train, X_test, y_test, train=True)
print_score(GaussianNB_clf, X_train, y_train, X_test, y_test, train=False)
dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)

print_score(dt_clf, X_train, y_train, X_test, y_test, train=True)
print_score(dt_clf, X_train, y_train, X_test, y_test, train=False)

```

```
aa=( 0.29046364, 0.47839125, -0.10172985, -1.16528085, -0.7243226 ,
      1.      , 0.      , 1.      , 0.      , 0.      ,
      0.      , 1.      , 0.      , 0.      , 1.      ,
      0.      , 0.      , 1.      , 0.      , 1.      ,
      0.      , 1.      , 0.      , 0.      , 0.      ,
      0.      , 0.      , 0.      , 0.      , 1. )
```

```
ab=(0.95, 0.76, -0.26, 0.02,  1.09,  0.00,  1.00,  0.00,
     0.00,  0.00,  1.00, 0.00, 1.00,  1.00,  0.00, 0.00,
     1.00, 0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
     0.00, 0.00,  1.00, 0.00, 0.00)
```

```
a = np.asarray(aa)
a = a.reshape(1,-1)
p = svc_clf.predict(a)
print(p)
X_train.iloc[19]
new = X_train.iloc[78]
a = np.asarray(new)
a = a.reshape(1,-1)
p = svc_clf.predict(a)
p[0]
if (p[0] == 1):
    print("Person has heart disease")
else:
    print("Great! the results are normal")
```



## **Chapter 11**

### **Conclusion**

In conclusion, we have developed a heart disease prediction system using machine learning algorithms, including logistic regression, Random Forest Classifier, and KNN. The aim of this system is to accurately and efficiently predict the likelihood of a patient being diagnosed with a heart disease based on their medical history.

Through our experimentation and analysis, we have found that the Random Forest Classifier algorithm yielded the best results in terms of accuracy. This algorithm effectively classified patients with heart disease, providing a satisfying level of prediction strength. By utilizing this model, we can improve the accuracy of diagnosing heart attacks in individuals.

The implementation of our heart disease prediction system has the potential to greatly enhance medical care while also reducing costs. By accurately identifying individuals at risk of heart disease, healthcare providers can intervene early, providing necessary treatment and preventive measures. This proactive approach can significantly reduce the occurrence and severity of heart-related complications.

This project has provided us with valuable insights into predicting heart disease in patients. The knowledge gained can assist healthcare professionals in making informed decisions and prioritizing patients who require further examination or intervention. Furthermore, the system's format, in .pynb, makes it accessible and easily reproducible for medical practitioners and researchers.

Overall, our heart disease prediction system demonstrates the potential of machine learning algorithms in healthcare and signifies a step forward in preventive medicine. As technology continues to advance, we anticipate further improvements in accuracy and efficiency, ultimately leading to better patient outcomes and a healthier population.

## References

- [1] Soni J, Ansari U, Sharma D & Soni S (2011). Predictive data mining for medical diagnosis: an overview of heart disease prediction. *International Journal of Computer Applications*, 17(8), 43-8.
- [2] Dangare C S & Apte S S (2012). Improved study of heart disease prediction system using data mining classification techniques. *International Journal of Computer Applications*, 47(10), 44-8.
- [3] Ordonez C (2006). Association rule discovery with the train and test approach for heart disease prediction. *IEEE Transactions on Information Technology in Biomedicine*, 10(2), 334-43.
- [4] Shinde R, Arjun S, Patil P & Waghmare J (2015). An intelligent heart disease prediction system using k-means clustering and Naïve Bayes algorithm. *International Journal of Computer Science and Information Technologies*, 6(1), 637.
- [5] Bashir S, Qamar U & Javed M Y (2014, November). An ensemble-based decision support framework for intelligent heart disease diagnosis. In *International Conference on Information Society (i-Society 2014)* (pp. 259-64). IEEE.
- [6] Jee S H, Jang Y, Oh D J, Oh B H, Lee S H, Park S W & Yun Y D (2014). A coronary heart disease prediction model: the Korean Heart Study. *BMJ open*, 4(5), e005025.
- [7] Ganna A, Magnusson P K, Pedersen N L, de Faire U, Reilly M, Ärnlöv J & Ingelsson E (2013). Multilocus genetic risk scores for coronary heart disease prediction. *Arteriosclerosis, thrombosis, and vascular biology*, 33(9), 2267-72.
- [8] Jabbar M A, Deekshatulu B L & Chandra P (2013, March). Heart disease prediction using lazy associative classification. In *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)* (pp. 40- 6). IEEE.
- [9] Dangare Chaitrali S and Sulabha S Apte. "Improved study of heart disease prediction system using data mining classification techniques." *International Journal of Computer Applications* 47.10 (2012): 44-8.

- [10] Soni Jyoti. "Predictive data mining for medical diagnosis: An overview of heart disease prediction." *International Journal of Computer Applications* 17.8 (2011): 43-8.
- [11] Chen A H, Huang S Y, Hong P S, Cheng C H & Lin E J (2011, September). HDPS: Heart disease prediction system. In *2011 Computing in Cardiology* (pp. 557-60). IEEE.
- [12] Parthiban, Latha and R Subramanian. "Intelligent heart disease prediction system using CANFIS and genetic algorithm." *International Journal of Biological, Biomedical and Medical Sciences* 3.3 (2008).
- [13] Wolgast G, Ehrenborg C, Israelsson A, Helander J, Johansson E & Manefjord H (2016). Wireless body area network for heart attack detection [Education Corner]. *IEEE antennas and propagation magazine*, 58(5), 84-92.
- [14] Patel S & Chauhan Y (2014). Heart attack detection and medical attention using motion sensing device -kinect. *International Journal of Scientific and Research Publications*, 4(1), 1-4.
- [15] Zhang Y, Fogoros R, Thompson J, Kenknight B H, Pederson M J, Patangay A & Mazar S T (2011). U.S. Patent No. 8,014,863. Washington, DC: U.S. Patent and Trademark Office.