Heart Failure Prediction Using Machine Learning Techniques

BootCamp on Data Science and Tools

Submitted

To

CRC-Training

By

CHARU GUPTA
Roll Number: 2002900100046

ASTHA SHARMA
Roll Number: 2002900100036

Under the guidance of
Mr. Gaurav Kansal / Mr. Gopal Gupta

TABLE OF CONTENTS

CERTIFICATE

This is to certify that Project Report entitled **"Heart Failure Prediction Using Machine Learning Techniques"** which is submitted by **Charu Gupta** and **Astha Sharma** in partial fulfillment of the requirement for the "Bootcamp on Data Science and Tools" in Department of CRC-Training of ABES Institute of Technology, is a record of the candidate own work carried out by him under my/our supervision.

Mr. Gaurav Kansal

Mr. Gopal Gupta

Date:

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the "Bootcamp on Data Science and Tools" *undertaken during B.Tech, 3rd Year. We owe special debt of gratitude to Mr. Gopal Gupta and Mr. Gaurav Kansal for his constant support and guidance throughout the course of our work. His constant motivation has been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.*

We also take the opportunity to acknowledge the contribution of team members of CRC-Training for their full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the motivation of the Department Of Computer Science And Engineering, ABES Institute of Technology to provide us the opportunity to undergo training at CRC-Training.

Signature:

Name : Astha Sharma

Roll No.:2002900100036

Date : 02.01.2023

Signature:

Name : Charu Gupta

Roll No.: 2002900100046

Date : 02.01.2023

ABSTRACT

The modern lifestyle has led to an increase in heart diseases due to people neglecting their health while focusing on work and other activities. Heart disease is a major cause of death, accounting for 31% of global deaths. It has emerged as the top killer that has affected both urban and rural populations. The medical field has a huge amount of data, making it difficult to analyze.

Machine learning techniques can provide an efficient way to predict heart disease. This research aims to predict heart disease using machine learning techniques and analyze the algorithms used for the prediction. Heart disease is becoming more common due to the global pandemic, and predicting it accurately has become a complicated and challenging task.

The paper proposes a robust system that uses a dataset from Kaggle.com with 15 attributes and implements different algorithms, including SVM, Naïve Bayes, Logistic Regression, Decision Tree, Random Forest and KNN. Random Forest produced the best result with an accuracy of 88.7%. The paper also presents a comparative analysis of all the algorithms used. The research uses model validation techniques to design the best suitable model for the current scenario. Early prediction of heart disease can save many lives, and this research aims to design an accurate prediction system that overcomes the limitations of existing systems.

LIST OF TABLES

LIST OF FIGURES

CHAPTER : 1

INTRODUCTION

1.1 Problem Definition: -

- Heart disease is a major cause of death, accounting for 32% of global deaths.
- Therefore to identify people having heart disease we are using data analysis and ML model for prediction.
- We are predicting whether a person has heart disease or not through various ml models.

1.2 Motivation:

- As already stated in our problem statement, this model helps in predicting whether a person has heart disease or not.
- Currently, the world is facing the devastating effects of COVID-19, which has had a significant impact on the global economy, health.
- By using the data analysis and machine learning model we will be predicting whether a person has heart disease or not based on 15 attributes. It helps in early detection of heart disease.

1.3 Objective of the Project: The Project is based on a machine learning model that can predict if a person has a heart disease or not.

1.4 Scope of the Project: The scope of the project is very vast, as it can be used in early detection of heart disease which helps in reducing the death rate due to heart diseases.

1.5 Need of Work: - The objective of this work is early detection of heart disease. It improves accuracy and helps individuals to get personalized treatments. It further reduces healthcare costs.

CHAPTER : 2

RELATED WORK

This section will provide references to papers that were read to gather useful information about the dataset and its attributes for the project. The papers provided valuable insights into the data, including its features and characteristics. Based on these papers,we gained a basic understanding of the dataset and its attributes, which will be helpful in developing the project.

- The paper presents a literature survey on the use of machine learning algorithms for the diagnosis of cardiovascular heart disease.
- Various algorithms such as logistic regression, KNN, and random forest classifier have been used for efficient cardiovascular disease prediction.
- The accuracy of these models varied based on their strengths and the defined objectives.
- Some models incorporated important factors like family history, but their accuracy was found to be less compared to new models that used techniques such as artificial neural networks.
- Studies by McPherson et al. and R. Subramanian et al. introduced the use of neural networks to diagnose and predict heart disease and blood pressure using various attributes.
- The deep neural network built by R. Subramanian et al. was found to be effective in accurately predicting heart disease, with 120 hidden layers and an output perceptron.
- Supervised networks were recommended for heart disease diagnosis.
- Testing of the model was carried out by doctors on unfamiliar data to assess the accuracy of the model.

CHAPTER : 3

PROPOSED METHODOLOGY

4.1 Dataset Description

- The dataset for our Heart Failure Prediction Model was downloaded from KAGGLE.
- The data set has 918 rows.
- This dataset is a result of merging 5 separate heart datasets and every dataset used can be found under Index of heart disease datasets from UCI Machine Learning repository. The five datasets used for its curation are:

  Cleveland : 303 observations
  Hungarian : 294 observations
  Switzerland : 123 observations
  Long Beach VA : 200 observations
  Statlog Data Set : 270 observations

- It contains a total of 15 attributes. They are as follows:

a. **Age: -** This attribute records the patient's age [years].

b. **Sex: -** This attribute takes the gender of the particular individual [M : Male, F : Female].

c. **ChestPainType: -** This attribute tells the type of chest pain the patient has [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic].

d. **RestingBP: -** This attribute takes the resting blood pressure of the patient [mmHg].

e. **Cholesterol: -** This attribute records serum cholesterol of the patient [mm/dl].

f. **FastingBS: -** This attribute takes fasting blood sugar level which is obtained by analysing a blood sample of a patient who has refrained from eating for a minimum of eight hours [1: if FastingBS > 120 mg/dl, 0: otherwise].

g. **RestingECG: -** This attribute takes resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality, LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]

h. **MaxHR: -** This attribute contains the maximum heart rate achieved by the patient.

i. **ExerciseAngina: -** This attribute tells whether an individual experiences chest pain or discomfort during any physical activity or exercise [Y : Yes, N : No].

j. **OldPeak: -** This attribute is used in cardiology to describe a characteristic on an electrocardiogram (ECG) reading.

k. **ST_Slope: -** This attribute is used to refers to the segment of the electrocardiogram (ECG) that connects the end of the S wave to the beginning of the T wave [Flat, Up, Down].

l. **HeartDisease: -** This represents the output class [1 : heart disease, 0 : normal].

3.2 Methods:

After downloading the data set, we begin from analyzing the data set's attributes, then preprocessing it.

With the help of **matplotlib** and **seaborn** we visualize each attribute with a target variable.

Then we applied various Machine Learning Models.

LOGISTIC REGRESSION:

- Logistic regression is a popular algorithm in machine learning used for predicting a categorical dependent variable based on a given set of independent variables.
- It is a form of supervised learning where the outcome is a categorical value such as Yes or No, 0 or 1, or true or false. The algorithm predicts the output as probabilistic values between 0 and 1. Unlike linear regression, which is used for regression problems, logistic regression is used for classification problems.
- This curve indicates the likelihood of a particular event, such as whether a cell is cancerous or not.
- Logistic regression is significant because it can provide probabilities and classify new data using both continuous and discrete datasets. It can also determine the most effective variables for classification.

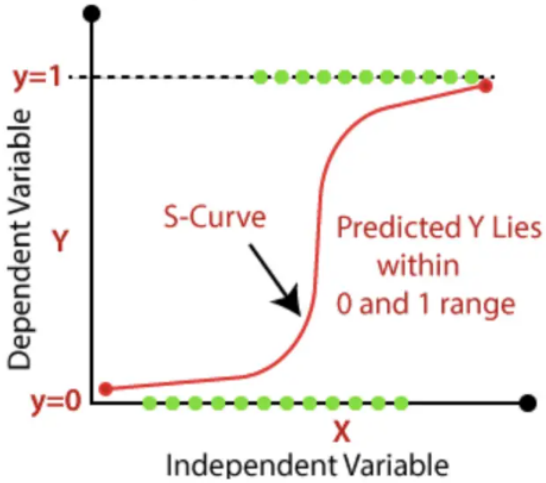

Fig. 3.1: Logistic Regression Graph

Types of Logistic Regression:

- **Binary Logistic Regression:** Binomial logistic regression is a type of logistic regression where the dependent variable can only have two possible outcomes, such as Pass or Fail, Yes or No, or 0 or 1..

- **Multinomial Logistic Regression:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep".
- **Ordinal Logistic Regression:** Ordinal logistic regression is a variation of logistic regression used when the dependent variable has three or more ordered categories, such as "low," "medium," or "high."

Advantages:

- Easy to implement.
- Used widely by data analysts and scientists.

Disadvantages:

- When there are a large number of categorical features or variables, logistic regression may not be suitable for modeling them effectively.
- It constructs linear boundaries.

DECISION TREE:

Decision Tree is a type of supervised learning method that is applicable to both classification and regression problems, although it is commonly used for classification tasks. This approach involves a tree-like structure where the nodes represent the dataset's features, the branches reflect the decision rules, and the leaf nodes indicate the final outcome.

Decision nodes are responsible for making decisions and have multiple branches. In contrast, leaf nodes are the final outputs resulting from those decisions and do not have any additional branches. The decisions made by decision nodes are based on the features present in the given dataset. A decision tree is a visual representation that illustrates all the potential solutions to a problem or decision based on specified conditions. The CART (Classification and Regression Tree) algorithm is utilized to construct a decision tree.

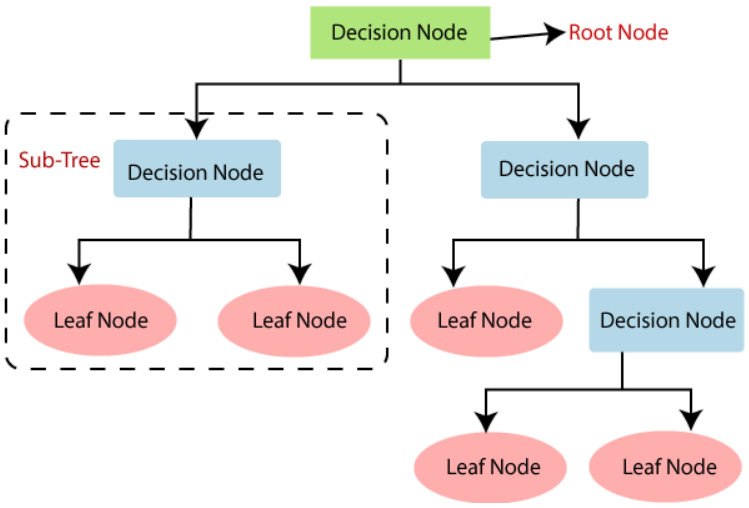

Fig. 3.2: Flow-chart of Decision Tree

How does the Decision Tree algorithm work?

Decision trees utilize several algorithms to determine when to divide a node into multiple sub-nodes. This division of sub-nodes enhances their uniformity concerning the target variable, thereby improving their purity. The decision tree partitions nodes across all accessible variables and chooses the division that generates the most uniform sub-nodes.

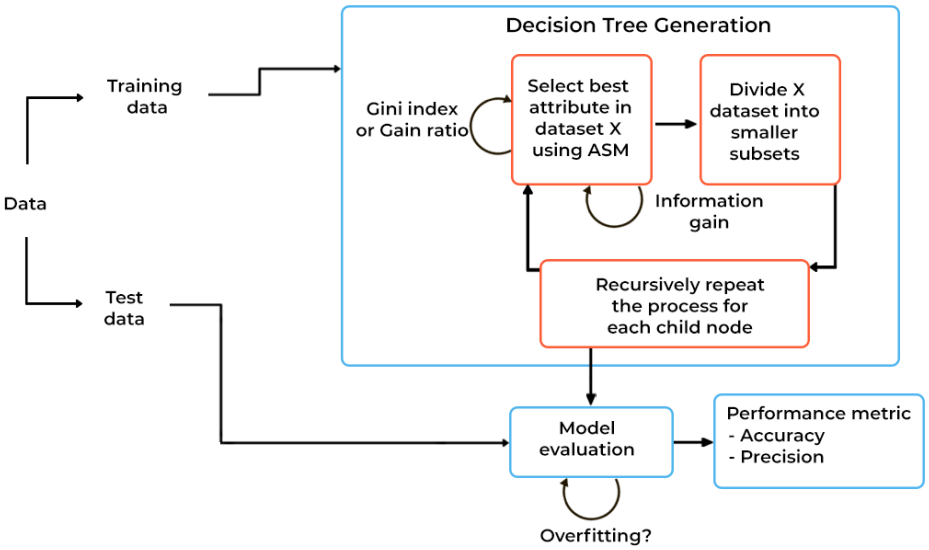

Fig. 3.3: Working of Decision Tree

Advantages:

- Decision trees are simple to comprehend and visualize.
- They can effectively identify non-linear patterns.
- Users don't need to perform as much data pre-processing, such as normalization.
- Decision trees are useful for feature engineering tasks, such as filling in missing values or selecting variables.
- The algorithm's non-parametric nature allows it to avoid making assumptions about data distribution.

Disadvantages:

- Decision trees are prone to overfitting noisy data, making them sensitive to outliers.
- Slight variations in data can lead to different decision trees, but techniques like bagging and boosting can help reduce this issue.
- The algorithm may exhibit bias when dealing with imbalanced datasets, so it's advisable to balance the dataset prior to creating the decision tree.

RANDOM FOREST:

Random Forest is a well-known supervised learning algorithm suitable for solving both Classification and Regression problems in machine learning. The technique is based on ensemble learning, which entails combining multiple classifiers to tackle a complex problem and enhance model performance.

The "Random Forest" classifier derives its name from its composition of various decision trees on different subsets of the input data. The algorithm determines the final output by averaging the predictions of all decision trees, rather than relying on one decision tree. The majority vote of predictions from each tree is used to make the final prediction. Increasing the number of trees in the forest can improve accuracy and prevent overfitting.

Why Random Forest?

Here are some reasons why the Random Forest algorithm is advantageous:

- It requires less training time than other algorithms.
- It is capable of efficiently predicting output with high accuracy, even when working with large datasets.
- It can maintain accuracy even when a large amount of data is missing.

How does the algorithm work?

Random Forest algorithm is a type of ensemble learning algorithm that combines multiple decision trees to make predictions. Here's how the Random Forest algorithm works:

- Data Preparation: The Random Forest algorithm takes a training dataset as input, which is used to train the decision trees.
- Random Sampling: The algorithm selects a random subset of the training dataset for each decision tree. This is done to prevent overfitting, as each tree is trained on a different subset of data.
- Building Decision Trees: The algorithm builds multiple decision trees based on the selected subset of data. The decision trees are built using a recursive process that splits the data into smaller subsets based on different features until a stopping criterion is met.
- Predictions: Once the decision trees are built, the algorithm uses them to make predictions on the test dataset. The prediction is made by traversing each decision tree and taking the majority vote of the predicted values.
- Aggregating Predictions: The final step of the Random Forest algorithm is to aggregate the predictions of all the decision trees. For classification problems, the majority vote is taken as the final prediction, and for regression problems, the average of the predicted values is taken.

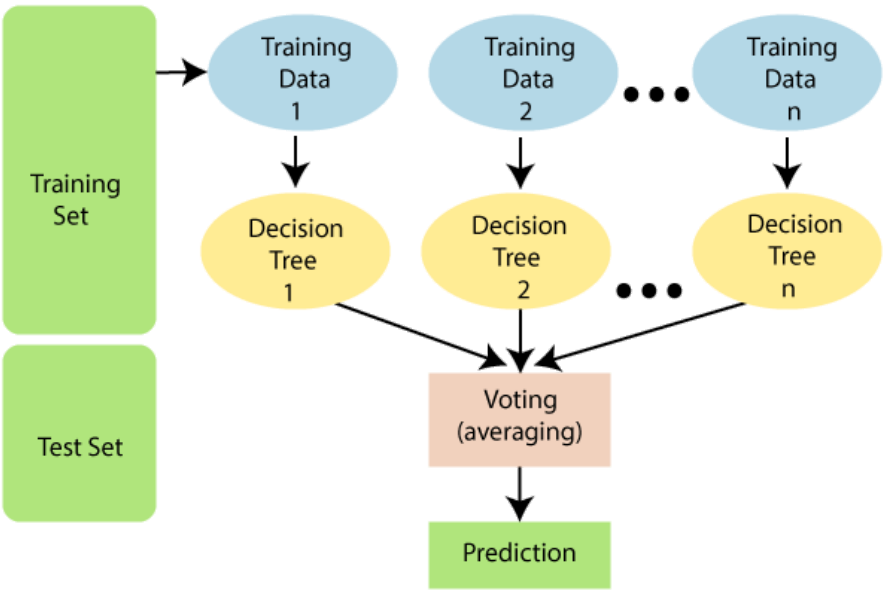

Fig. 3.4: Working of Random Forest

Advantages:

- Random forests are a reliable and precise technique owing to the involvement of multiple decision trees in the process.
- It is not prone to overfitting due to the averaging of all predictions, which negates any biases.
- This approach is applicable to both regression and classification problems.

Disadvantages:

- Random forests are a slow method of generating predictions due to the inclusion of multiple decision trees. Each time a prediction is required, all trees in the forest must make a prediction for the same input, followed by voting. This entire process is time-intensive.

KNN CLASSIFIER :

- K-Nearest Neighbor is a basic supervised learning algorithm that relies on the similarity between new and existing data points. The algorithm stores all available data and classifies new data based on its similarity to the existing data. As a result, K-NN can easily classify new data into appropriate categories. While K-NN can be used for both regression and classification, it is primarily used for classification problems.
- K-NN is a non-parametric algorithm, meaning it does not make any assumptions about the underlying data. Because it does not immediately learn from the training set, K-NN is also known as a lazy learner algorithm. During the training phase, the KNN algorithm simply stores the dataset and then classifies new data based on how similar it is to existing data points.

The K-Nearest Neighbors (KNN) algorithm involves the following fundamental steps:

1. Measure the distance between a given data point and all the other data points in the dataset.
2. Identify the K data points that are closest to the given data point based on the distance metric used.
3. Determine the labels associated with those K data points.
4. Use a voting mechanism to assign a label to the given data point based on the most frequent label among the K neighbors.

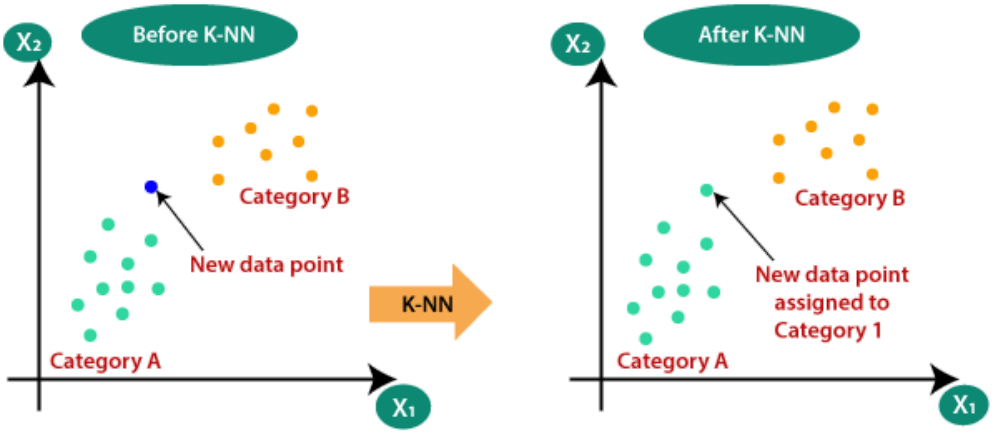

Fig. 3.5: Working of KNN Classifier

SUPPORT VECTOR MACHINE ALGORITHM :

Support Vector Machine (SVM) is a widely used supervised learning algorithm that can handle both classification and regression problems. However, it is mainly employed for classification tasks in the field of Machine Learning.
The primary objective of SVM is to construct the optimal line or decision boundary known as a hyperplane, which can effectively segregate the n-dimensional space into distinct classes. This hyperplane enables the SVM algorithm to assign new data points to the appropriate category with ease.
SVM selects the most crucial points or vectors, which aid in creating the hyperplane. These crucial instances are referred to as support vectors, and the algorithm is named as Support Vector Machine because of their utilization.

Types of SVM :

- **Linear SVM -** Linear Support Vector Machine (SVM) is appropriate for datasets that can be separated into two classes with a straight line. In other words, if the data can be linearly separated, then a Linear SVM classifier is used.

- **Non-Linear SVM** - Non-Linear SVM is suitable for datasets that cannot be classified using a straight line. If the data is non-linearly separable, meaning it cannot be separated by a straight line, then a Non-Linear SVM classifier is utilized.

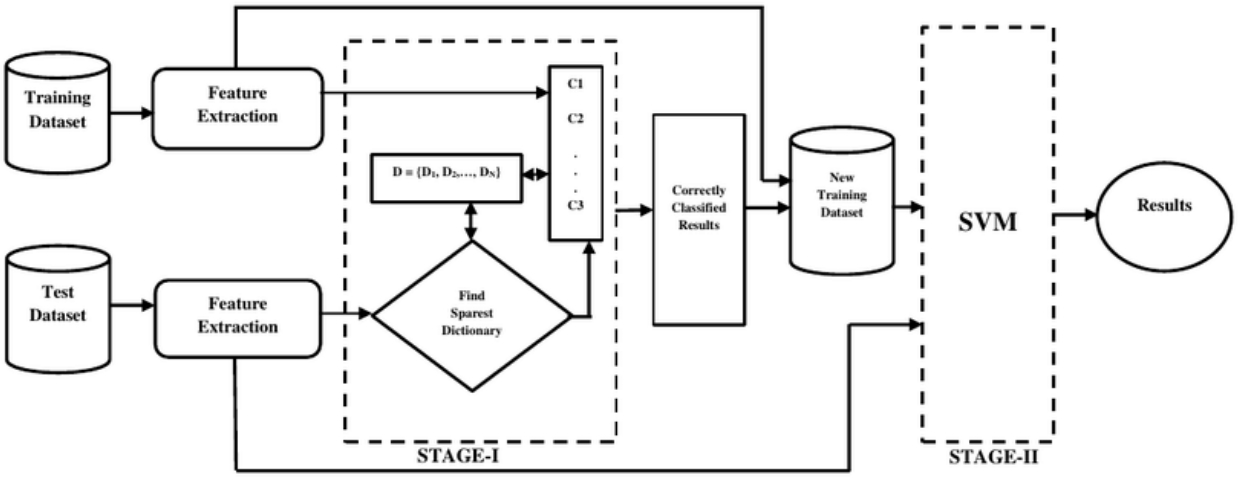

Fig. 3.6: Working of SVM

NAIVE BAYES CLASSIFIER :

The Naïve Bayes algorithm is a supervised learning technique that uses Bayes' theorem for classification problems. It is widely used in text classification problems that involve large, high-dimensional training datasets.The Naïve Bayes Classifier is a straightforward and highly effective classification algorithm that enables the creation of fast machine learning models that can make swift predictions.

Being a probabilistic classifier, Naïve Bayes makes predictions based on the probability of an object. It is commonly used in various applications, such as spam filtering, sentimental analysis, and article classification.

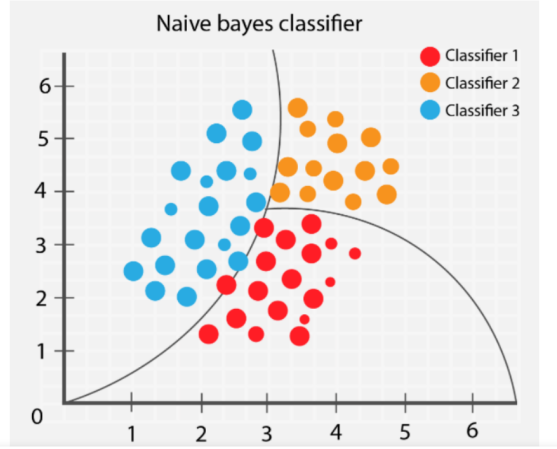

Fig. 3.7: Graph of Naive Bayes Classifier

3.3 Hardware / Software Requirements

Minimums Hardware Requirements:

RAM: 2 GB
Processor: Intel Core 2 Duo
Hard disk: 50GB

Minimums Software Requirements:

Pandas: 0.24.2
Numpy; 1.16.4
Matplotlib: 3.1.0
Seaborn: 0.9.0
Python: 3.7
Scikit-Learn: 0.21.2

3.4 Our Methodology:

- Firstly, we downloaded the data set from UCI machine learning repository.
- Then we loaded the data set into Jupyter Notebook

```
In [10]: import pandas as pd
         import numpy as np
         import seaborn as sns
         import matplotlib.pyplot as plt
```

Fig. 3.8: Importing numpy and pandas

Importing numpy, pandas, seaborn, matplotlib packages.

NumPy is a python library used for working with arrays.

Pandas is a high-level data manipulation tool.

Seaborn is a Python data visualization library based on matplotlib paraphrase.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python

```
In [11]: df=pd.read_csv(r"C:\Users\Dell\OneDrive\Desktop\desk\heart.csv")
```

Fig. 3.9: read_csv()

- **Describing the data:**

```
In [13]: df.describe()
```

Out[13]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Fig. 3.10: describe() function

Using **describe()** function to get the description of the dataframe.

- There are total of 7 int attributes.

```
In [84]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    int64
 2   ChestPainType   918 non-null    int32
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    int32
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    int64
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    int32
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int32(3), int64(8)
memory usage: 75.4 KB
```

Fig. 3.11: info() function

The **info()** function is used to display the concise summary of the dataframe including non-null values, datatype of each column, and the amount of memory used by dataframe

- There are 9 object variable
- There are 6 int64 variable
- There are 32561 rows entries

The total size of data set is 3.7+ MB

```
In [19]: df.head()
```

Out[19]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Fig. 3.12: head() function

Using head() method to get a specified number of rows, string from the top.

CHECKING NULL VALUES FOR EACH COLUMN

Using isnull() method to check for null values.

```
In [18]: df.isnull().sum()

Out[18]: Age              0
         Sex              0
         ChestPainType    0
         RestingBP        0
         Cholesterol      0
         FastingBS        0
         RestingECG       0
         MaxHR            0
         ExerciseAngina   0
         Oldpeak          0
         ST_Slope         0
         HeartDisease     0
         dtype: int64
```

Fig. 3.13: isnull() function

- **CONVERSION INTO CATEGORICAL VARIABLES:**
  Converting categorical variables with numerically coded values

```
In [58]: encode = {"M":0,"F":1}
         df['Sex'] = df['Sex'].replace(encode)
```

```
In [59]: encode = {"N":0,"Y":1}
         df['ExerciseAngina'] = df['ExerciseAngina'].replace(encode)
         df
```

```
In [61]: from sklearn import preprocessing
         label_encoder = preprocessing.LabelEncoder()
         df1['ChestPainType'] = label_encoder.fit_transform(df1['ChestPainType'])
         df1
```

```
In [63]: label_encoder = preprocessing.LabelEncoder()
         df1['RestingECG'] = label_encoder.fit_transform(df1['RestingECG'])
         df1
```

```
In [62]: label_encoder = preprocessing.LabelEncoder()
         df1['ST_Slope'] = label_encoder.fit_transform(df1['ST_Slope'])
         df1
```

Fig. 3.14: Converting into categorical values

Dataset with numerically coded value for each categorical value in columns.

Out[85]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR
0	40	0	1	140	289	0	1	172
1	49	1	2	160	180	0	1	156
2	37	0	1	130	283	0	2	98
3	48	1	0	138	214	0	1	108
4	54	0	2	150	195	0	1	122
...	...	...	...	...	...	...	...	...
913	45	0	3	110	264	0	1	132
914	68	0	0	144	193	1	1	141
915	57	0	0	130	131	0	1	115
916	57	1	1	130	236	0	0	174
917	38	0	2	138	175	0	1	173

918 rows × 12 columns

Fig. 3.15: Attributes after conversion into categorical types

ANALYSING EACH ATTRIBUTE

AGE :

```
In [26]: df.Age.hist()
         plt.show()
```

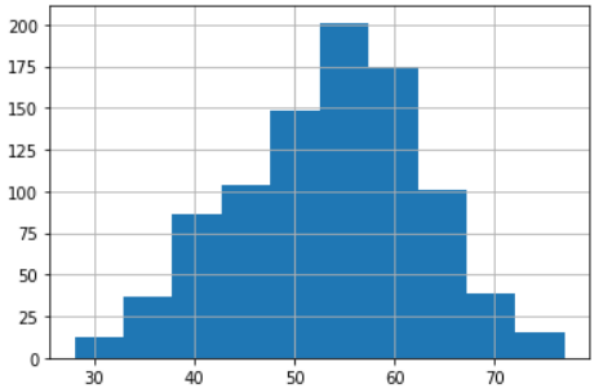

Fig. 3.16: Count of people vs. Age

- The visualization above displays how the ages are distributed among the entries in our dataset.
- The ages range from 25 to 78 years old.
- It has the majority of entries between the ages of 45 and 68 years.

MAXHR :

```
In [27]: df.MaxHR.hist()
Out[27]: <AxesSubplot:>
```

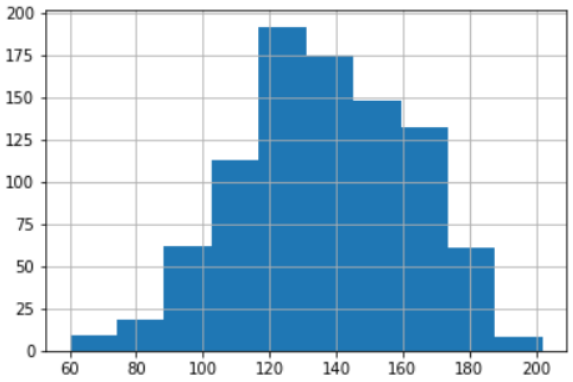

Fig. 3.17: Count of people vs. MaxHR

From the above graph:-

- More than 175 people have a maximum heart rate between 120-140.

CHOLESTEROL :

```
In [28]: df.Cholesterol.hist()
Out[28]: <AxesSubplot:>
```

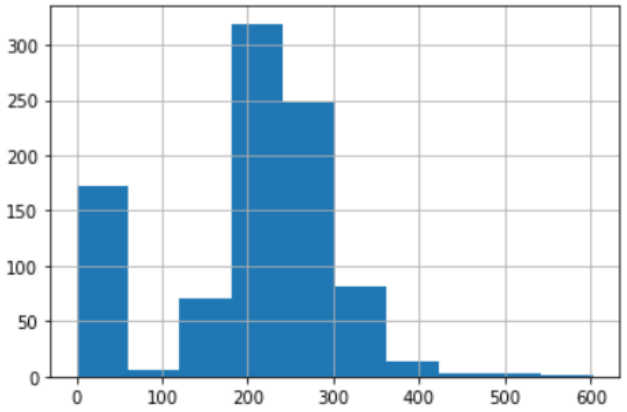

Fig. 3.18: Count of people vs. Cholesterol

From the above graph:-

- More than 300 people have cholesterol between 190-240.

SEX:

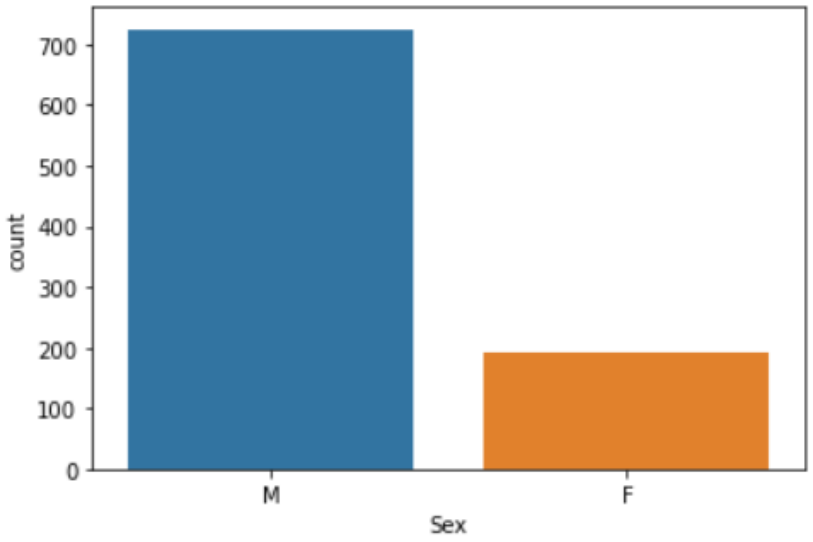

Fig. 3.19: Count of people vs. Sex

From the above graph:-

- Majority of candidates are male in the above dataset.
- There are 700 males and 200 females.

CHEST PAIN TYPE :

```
In [19]: sns.countplot(x="ChestPainType",data=df)
         plt.show()
```

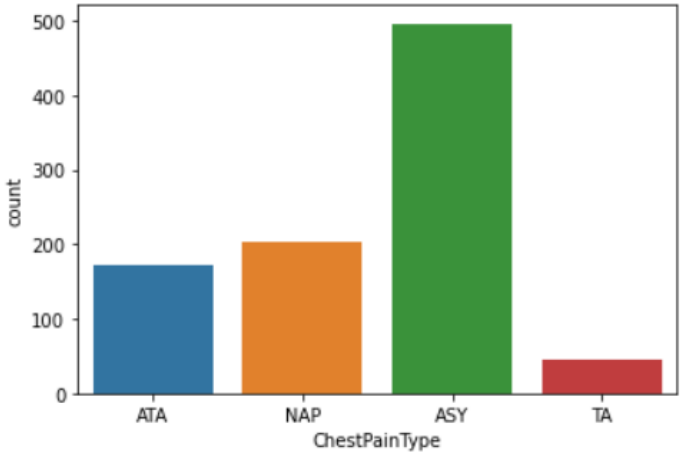

Fig. 3.20: Count of people vs. Chest Pain type

From the above graph:-

- There are 4 types of Chest pain types present in the dataset namely ATA (Atypical angina), ASY(Asymptomatic), NAP(Non anginal pain), TA(Typical angina)
- Majority of the people suffer from ASY.

RESTING ECG :

```
In [211]: plt.figure(figsize=(5,4))
          sns.histplot(data=df, x="RestingECG", hue="RestingECG", multiple="stack", shrink=.9)
          plt.show()
```

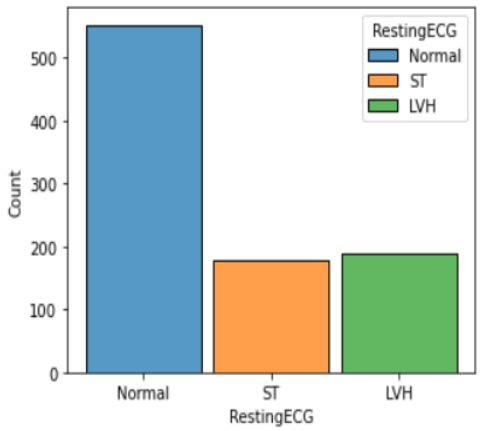

Fig. 3.21: Count of people vs. Resting ECG

From the above graph:-

- We can clearly see that there are a majority of people with a pattern of electrical activity in the heart that is within the normal range.

EXERCISE ANGINA:

```
In [29]: sns.countplot(x="ExerciseAngina", data=df)
         plt.show()
```

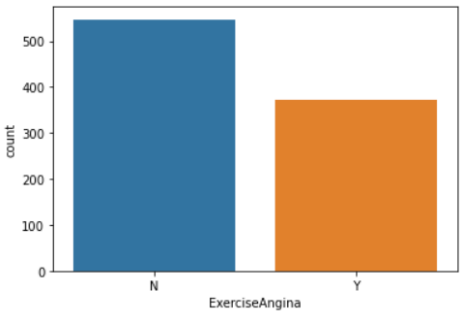

Fig. 3.22: Count of people vs. Exercise Angina

From the above graph:-

- We can see that the majority of the people do not suffer from exercise angina.
- Less than 400 people suffer from exercise angina.

OLD PEAK:

```
In [137]: df.Oldpeak.hist()
Out[137]: <AxesSubplot:>
```

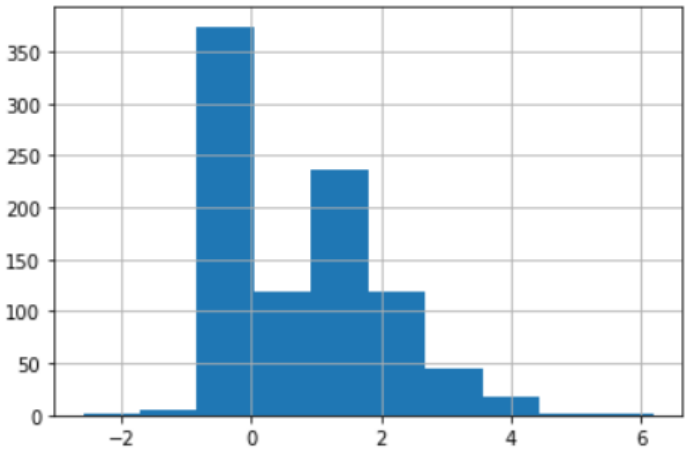

Fig. 3.23: Count of people vs. old peak

From the above graph:-

- We can observe that the majority of people have old peak between -1 and 0.

ST SLOPE:

```
In [209]: plt.figure(figsize=(5,4))
          sns.histplot(data=df, x="ST_Slope", hue="ST_Slope", multiple="stack", shrink=.9)
          plt.show()
```

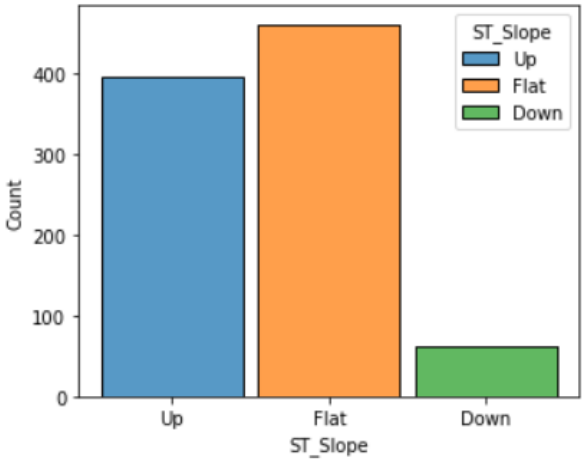

Fig. 3.24: Count of people vs. ST_Slope

From the above graph:-

- We can observe that the majority of people have flat ST segments that are considered abnormal and may indicate myocardial ischemia or injury. .

RESTING BP:

```
In [208]: sns.displot(df, x="RestingBP", bins=15)
          plt.show()
```

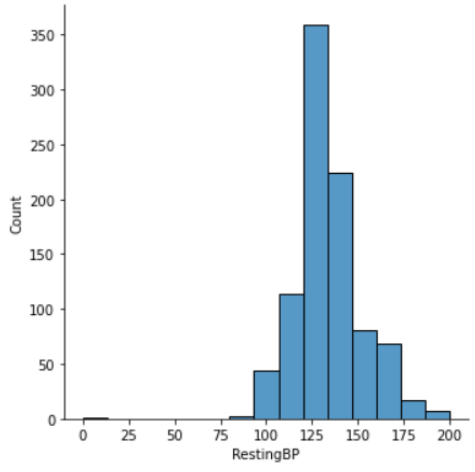

Fig. 3.25: Count of people vs. RestingBP

From the above graph:-

- We can observe that around 300 people have Resting BP ranging from 120 - 135.

FASTING BS:

```
In [27]: plt.figure(figsize=(5,4))
         sns.histplot(data=df, x="FastingBS", hue="FastingBS", multiple="stack", shrink=.9)
         plt.show()
```

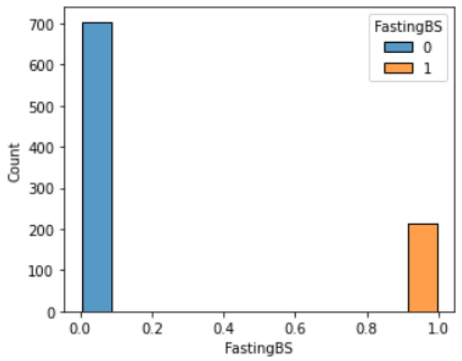

Fig. 3.26: Count of people vs. FastingBS

From the above graph:-

- We can observe that the majority of people have normal fasting blood sugar level (between 70-99 mg/dL).
- 200 people suffer from high blood sugar levels.

HEATMAP :

```
In [184]: plt.figure(figsize=(17,6))
          sns.heatmap(df.corr(),annot=True)
          plt.show()
```

Fig. 3.27: Heatmap code

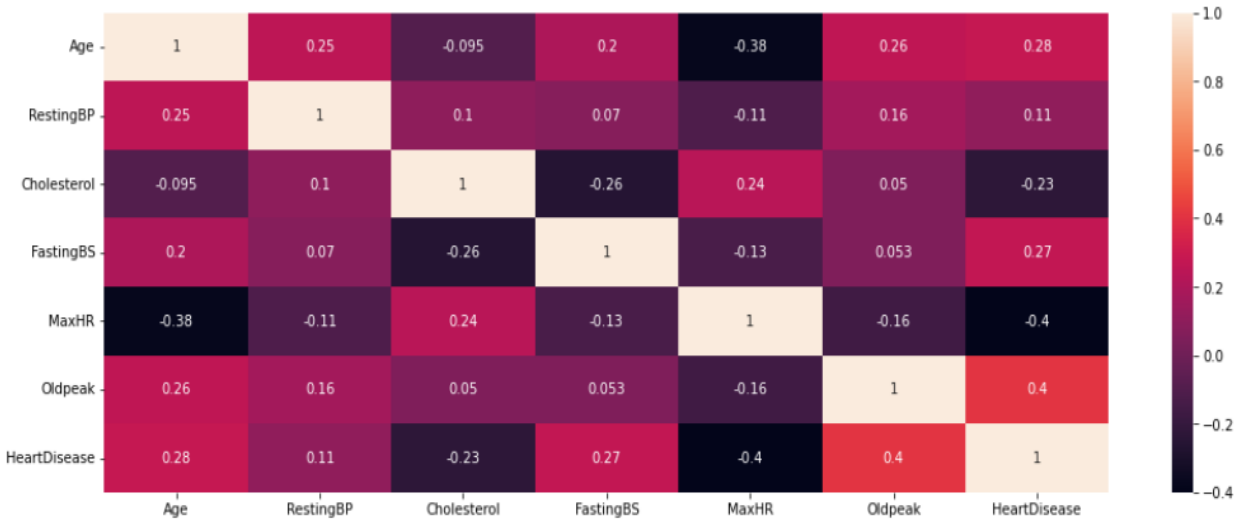

Fig. 3.27.1: HEATMAP

- ✔ This is the correlation matrix of all attributes in the form of Heat map.
- ✔ The darker the color, the lesser correlation it has.
- ✔ Most parts of the heat map are dark.
- ✔ So, the attributes are poorly correlated.

Visualizing relationship between various categorical attributes :

➔ Age group more prone to a heart disease -

```
In [46]: sns.catplot(data=df, x="HeartDisease", y="Age")
         plt.show()
```

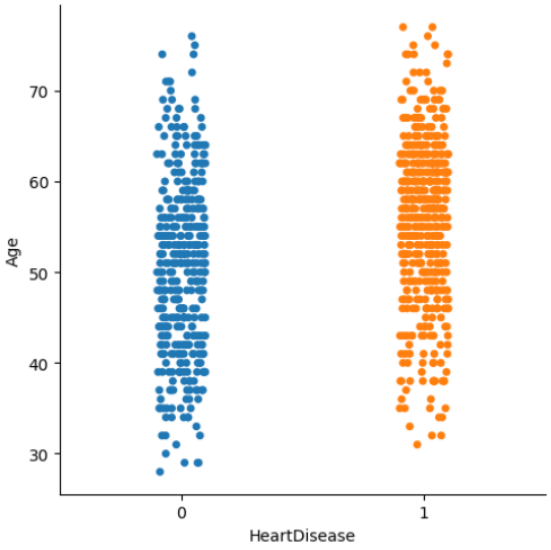

Fig. 3.28: Heart Disease vs. Age

This graph depicts that people between the age group 50-65 have more heart disease cases.

➔ Which chest pain type is more common among male and female -

```
In [87]: sns.catplot(data=df, x="ChestPainType", y="Age", hue="Sex", kind="swarm")
         plt.show()
```

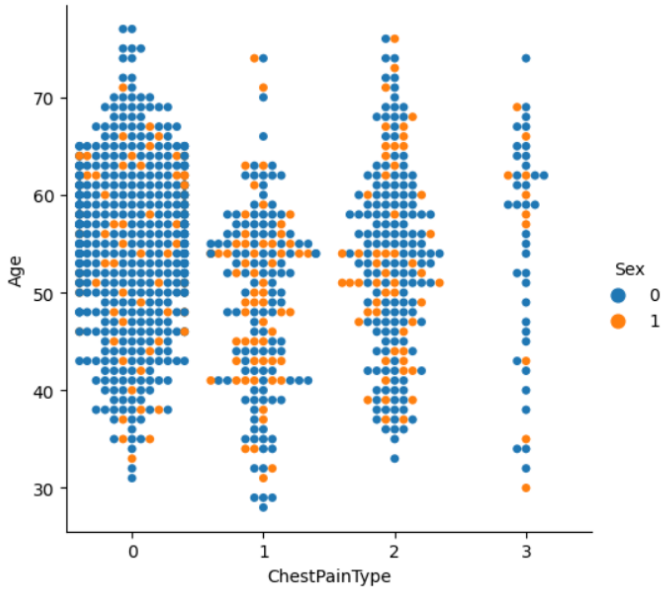

Fig. 3.29: Chest Pain Type vs. Age

It is observed that ASY chest pain is seen more in males whereas ATA chest pain is seen more in females.

➔ Which chest pain type will lead to higher chances of heart disease -

```
In [54]: sns.countplot(x="ChestPainType", hue="HeartDisease",data=df)
         plt.show()
```

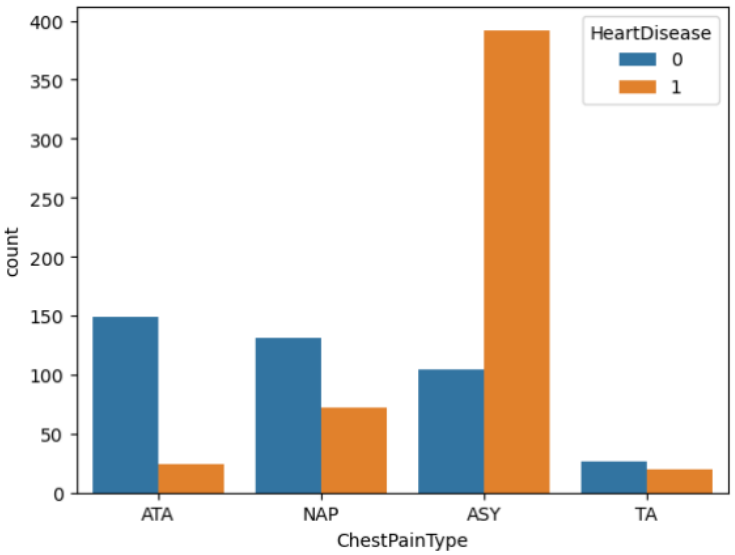

Fig. 3.30: Chest Pain Type vs. Heart Disease

This graph depicts that people with ASY chest pain have higher chances of heart disease.

➔ The relation between ST_slope and Heart disease -

```
In [55]: sns.countplot(x="ST_Slope", hue="HeartDisease",data=df)
         plt.show()
```

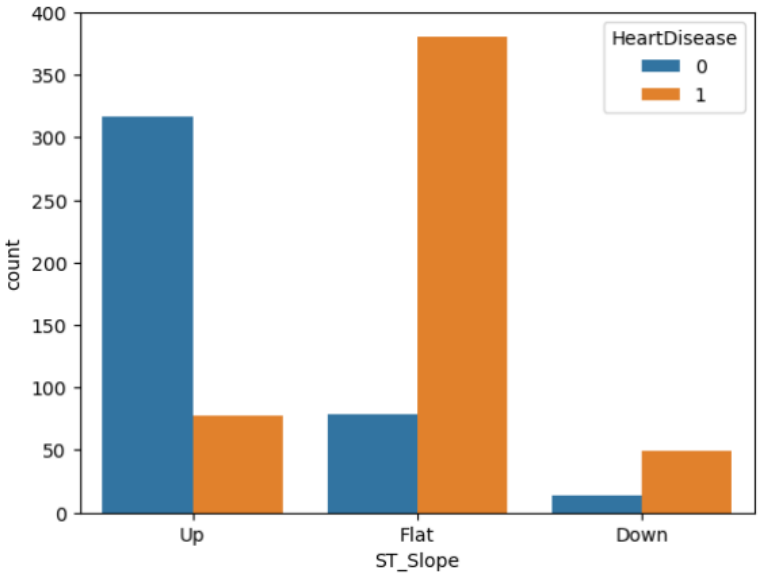

Fig. 3.31: Sex vs. ST_Slope

This graph depicts that people having flat ST_Slope are more likely to have heart disease.

➔ The relationship between Exercise angina and gender -

```
In [33]: sns.catplot(
             data=df, y="Sex", hue="ExerciseAngina", kind="count",
             palette="pastel", edgecolor=".0",
         )
         plt.show()
```

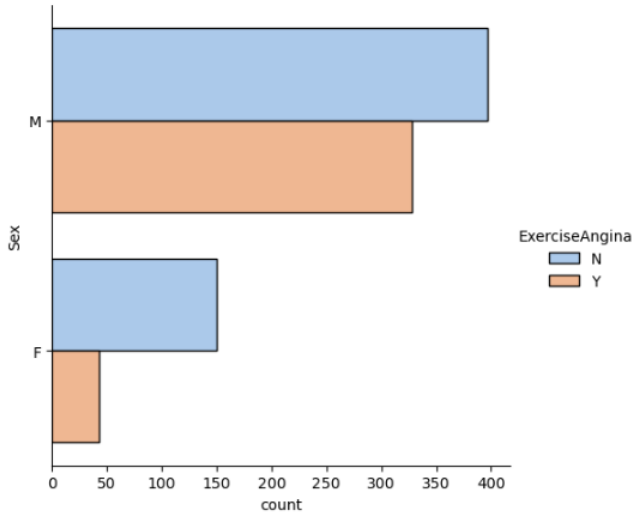

Fig. 3.32: Sex vs. Exercise Angina

This graph depicts that males are more prone to exercise angina.

➔ Least observed chest pain type among people -

```
In [85]: sns.catplot(data=df, x="ChestPainType", y="Age")
         plt.show()
```

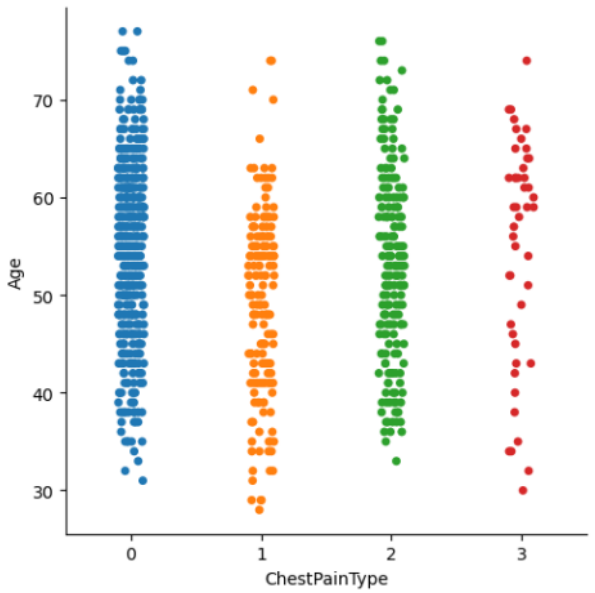

Fig. 3.33: Age vs. Chest Pain Type

This graph depicts that TA (Chest pain type) is least observed among people .

➔ Relation between Exercise angina and Chest pain type -

```
In [34]: sns.catplot(
             data=df, y="ExerciseAngina", hue="ChestPainType", kind="count",
             palette="pastel", edgecolor=".3",
         )
         plt.show()
```

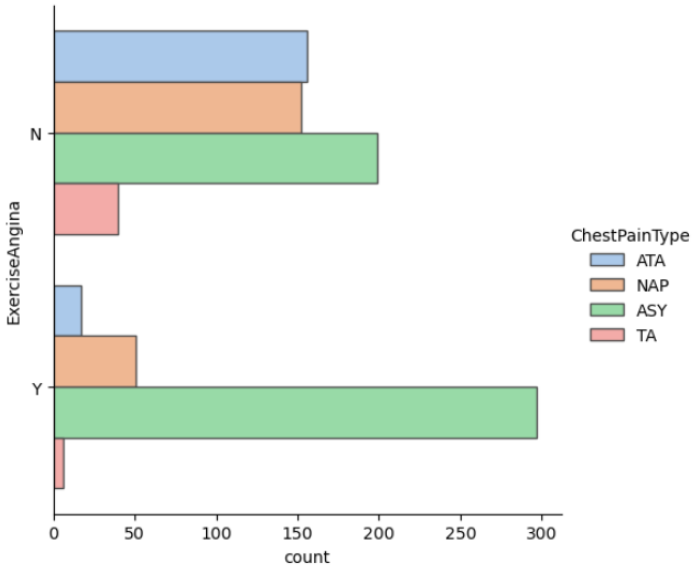

Fig. 3.34: Exercise Angina vs. Chest Pain Type

This graph shows that if a person has exercise angina then he/she is suffering from ASY.

➔ The relationship between Exercise angina and ST_Slope -

```
In [35]: sns.catplot(
             data=df, y="ExerciseAngina", hue="ST_Slope", kind="count",
             palette="pastel", edgecolor=".1",
         )
         plt.show()
```

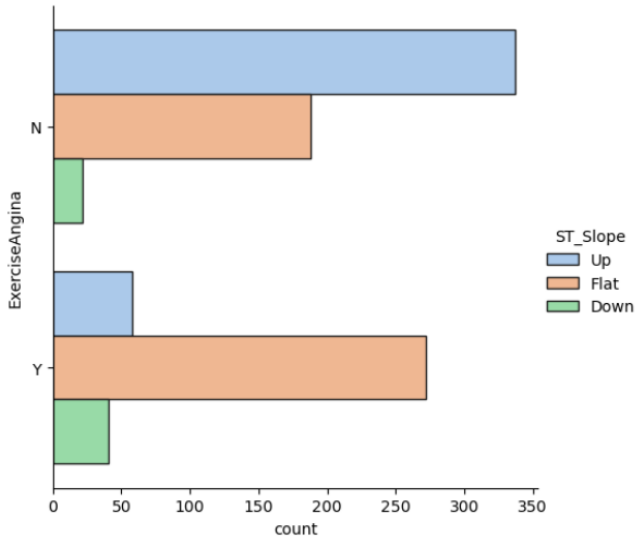

Fig. 3.35: Exercise Angina vs. St_Slope

This graph shows that if you don't have exercise angina then you are suffering from Up ST_Slope.

APPLYING VARIOUS MACHINE LEARNING ALGORITHMS :

Then we implement 4 Machine Learning Models on the data set :-

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. KNN Classifier
5. Support Vector Machine
6. Naive Bayes Classifier

Then After Comparison of Accuracy Score, One model is finalised for prediction.

Testing and **Training** Sets for both **Dependent** and **Independent** variable

By importing above module and using

train_test_split() function.

```
In [37]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 3)
```

Fig. 3.36: train_test_split() function

CHAPTER 4

EXPERIMENT AND RESULT ANALYSIS

We experimented with 6 Machine Learning Algorithms and according to their accuracy result, we selected the Random Forest Classifier.

LOGISTIC REGRESSION :
Logistic regression is a statistical technique that is commonly used for predicting binary outcomes.

It employs the use of the Sigmoid function to generate probability estimates for different classes.

```
In [31]: from sklearn.linear_model import LogisticRegression
```

Fig. 4.1: Importing Logistic Regression

After importing the above module

```
In [32]: log_reg=LogisticRegression()
         log_reg.fit(x_train,y_train)
         y_pred=log_reg.predict(x_test)
         print(y_pred)
```

Fig. 4.2: Training Model

Running the above code.

Checking accuracy of our ML Model :

```
In [37]: score = metrics.accuracy_score(y_test,y_pred)
         score
Out[37]: 0.8695652173913043
```

Fig. 4.3: Accuracy Percentage

This comes to be **86.9 %**

DECISION TREE :

Represents data in a tree-like structure, similar to a flowchart, and they make decisions by recursively splitting the dataset based on the attribute value that maximizes information gain. This algorithm continues to split the data into smaller and smaller groups, it learns how to partition the data effectively to make accurate predictions.

```
In [39]: from sklearn import tree
```

Fig. 4.4: Importing DecisionTreeClassifier

After importing the above module

```
In [40]: model=tree.DecisionTreeClassifier()
         model.fit(x_train,y_train)
```

Fig. 4.5: Training the model

Running the above code.

Checking accuracy of our ML Model :

```
In [41]: model.score(x_test,y_test)
Out[41]: 0.8347826086956521
```

Fig. 4.6: Accuracy Percentage

This comes to be **83.4 %**

RANDOM FOREST :

Random Forest is an ensemble learning method that creates multiple decision trees on randomly selected subsets of the data. Then it combines the predictions of each tree using a voting method to select the best possible solution. By using random sampling and combining the results of multiple trees, random forests can improve the accuracy and reduce overfitting in prediction tasks.

```
In [50]: from sklearn.ensemble import RandomForestClassifier
```

Fig. 4.7: Importing RandomForestClassifier

After importing the above module

```
In [56]: regressor = RandomForestClassifier(n_estimators=1000, max_features=9)
         regressor.fit(x_train,y_train)
         y_pred=regressor.predict(x_test)
```

Fig. 4.8: Training the model

Running the above code.

Checking accuracy of our ML Model :

```
In [55]: regressor.score(x_test,y_test)
Out[55]: 0.8876811594202898
```

Fig. 4.9: Accuracy Percentage

This comes to be **88.7 %**

KNN CLASSIFIER:

This algorithm works by examining the observations that are closest to the data point it is trying to classify or predict. Based on the majority of those nearest observations, the algorithm assigns a class to the data point in question. The number of nearest neighbors, represented by the value k, is a critical factor in the decision-making process of the algorithm, as it determines how many observations are taken into account when making a classification decision.

```
In [58]: from sklearn.neighbors import KNeighborsClassifier
```

Fig. 4.10: Importing KNeighboursClassifier

After importing the above module

```
In [60]: knc = KNeighborsClassifier(n_neighbors=10)
         knc=knc.fit(x_train,y_train)
         y_knc=knc.predict(x_test)
```

Fig. 4.11: Training the model

Running the above code.

Checking the accuracy of our ML Model

```
In [63]: knc.score(x_test,y_test)
Out[63]: 0.7318840579710145
```

Fig. 4.12: Accuracy Percentage

This comes to be **73.1 %**

SUPPORT VECTOR MACHINE :

SVM tries to find the best hyperplane that separates the different classes of data points in a given dataset. The hyperplane is chosen to maximize the margin or the distance between the closest data points of each class.

```
In [64]: from sklearn.svm import SVC
```

Fig. 4.13: Importing SVM

After importing the above module

```
In [66]: svm_model = SVC(kernel='linear')
         svm_model.fit(x_train, y_train)
         y_pred = svm_model.predict(x_test)
```

Fig. 4.14: Training the model

Running the above code.

Checking the accuracy of our ML Model

```
In [69]: svm_model.score(x_test,y_test)
Out[69]: 0.8768115942028986
```

Fig. 4.15: Accuracy Percentage

This comes to be **87.6 %**

NAIVE BAYES :

The algorithm calculates the probability of each class given the input features and selects the class with the highest probability as the predicted class. Naive Bayes is computationally efficient and requires less training data compared to other algorithms.

```
In [70]: from sklearn.naive_bayes import GaussianNB
```

Fig. 4.16: Importing Naive BayesClassifier

After importing the above module

```
In [72]: nb_model = GaussianNB()
         nb_model.fit(x_train, y_train)
         y_pred = nb_model.predict(x_test)
```

Fig. 4.17: Training the model

Running the above code.

Checking the accuracy of our ML Model

```
In [73]: nb_model.score(x_test,y_test)
Out[73]: 0.8768115942028986
```

Fig. 4.18: Accuracy Percentage

This comes to be **87.6 %**

Table 4.19: Accuracy Percentage of different algorithms

MACHINE LEARNING ALGORITHM	ACCURACY PERCENTAGE
Logistic Regression	86.9
Decision Tree	83.4
Random Forest	88.7
KNN Classifier	73.1
Support Vector Machine	87.6
Naive Bayes	87.6

After applying various machine learning algorithms to the dataset and analyzing the results, it has been determined that the Random Forest Classifier is the most effective algorithm for the given dataset.

CHAPTER 5
CONCLUSION

5.1 Discussion

After completing my first project on Data Analysis and Machine Learning, we concluded that our data set gave us best accuracy of 86.9% even when the Correlation matrix and heat map weren't so correlated. Therefore, we think that we should do some more research to make the data set more correlated and also find more complex as well as flexible dataset to increase our Prediction Accuracy.

5.2 Future Work

Future work on our project are as follows:

- Increase the accuracy.
- Make the machine learning model much better.
- Make a web version of the project and easy to use.

REFERENCES

Downloading data set:

https://archive.ics.uci.edu/ml/machine-learning-databases/adult/

http://abhigyan.dataritz.com/mod/folder/view.php?id=602

Loading and analyzing

https://pandas.pydata.org/

https://numpy.org/

Pre-processing

https://towardsdatascience.com/data-preprocessing-in-python-b52b652e37d5

https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/

https://scikit-learn.org/stable/modules/preprocessing.html

https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_data_preprocessing_analysis_visualization.htm

https://www.datacamp.com/courses/preprocessing-for-machine-learning-in-python

Data Visualization

https://matplotlib.org/tutorials/index.html

https://python-graph-gallery.com/seaborn/

Machine Learning Algorithm

https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

https://www.datacamp.com/community/tutorials/random-forests-classifier-python

https://www.datacamp.com/community/tutorials/decision-tree-classification-python

https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn