HEART DISEASE RISK LEVEL PREDICTION USING MACHINE LEARNING

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the
requirement for the award of the
Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

by

A. GUNA NAGA SAHITH NAIDU (20BCB7014)

Under the Guidance of

Dr. NAGENDRA PANINI CHALLA

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2023

CERTIFICATE

This is to certify that the Capstone Project work titled "**HEART DISEASE RISK LEVEL PREDICTION USING MACHINE LEARNING**" that is being submitted by **A.GUNA NAGA SAHITH NAIDU (20BCB7014)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. NAGENDRA PANINI CHALLA
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner 1
Dr. RAJESH DUVVURU

Internal Examiner 2
Dr. SHAIK KAREEMULLA

Approved by
Dr. ANIL V TURUMAKANE
HoD, Networking and Security.
School of Computer Science and Engineering

ABSTRACT

Nearly every day, terabytes of data are generated, as we are living in the "digital era." Day in day out, the healthcare sector produces a massive amount of data about patients and diseases. Researchers and practitioners, on the other hand, do not make good use of this data. Heart disease is widely regarded as one of the world's leading causes of death. If present trends are left unabated, heart disease will claim the lives of 23.6 million people by 2030. As a response, credible, viable, and accurate diagnosis of such diseases sufficient time for proper treatment is needed.

Today's healthcare sector is "data-rich" but "knowledge-poor." There is indeed a scarcity of efficient data analysis methods for uncovering concealed patterns and relationships. So, the main objectives of this research paper are predicting and making an analytic conclusion regarding Heart Diseases with comparative results by using data analysis and machine learning.

TABLE OF CONTENTS

LIST OF TABLES:

LIST OF FIGURES:

CHAPTER-1

1.INTRODUCTION

Health is an important part of everyone's life. However, due to external factors such as unhealthy living conditions, work stress, psychological stress, pollution, hazardous work environment and inadequate health care, millions of people worldwide suffer from cardiovascular diseases (CVD) and other chronic diseases affecting the heart and blood vessels resulting in death or disability.

In recent years, CVD has been recognized as the cause of most deaths. Associated conditions include hypertension, thromboembolism, hyperlipidemia, and heart failure ending in heart failure. High blood pressure is the leading cause of CVD. In 2012, 7.4 million deaths from heart disease and 6.7 million deaths from stroke were reported. The World Health Organization estimates that there are approximately 17 million deaths from CVD each year, accounting for approximately 31% of global deaths.

Early detection of CVD can cure patients and save countless lives. Diagnosing and treating patients in the early stages by cardiologists is challenging. Any traditional CVD risk-assessment model implicitly treats each risk factor associated with CVD outcome in a linear fashion.

Such models tend to oversimplify strong relationships, with several risks associated with nonlinear interactions. Multiple risk factors need to be effectively included, and smaller factors that are more closely associated with risk factors and outcomes need to be identified. To date, there is no large-scale study using conventional clinical data and machine learning (ML) in predictive CVD research. The aim of this study was to determine whether ML can significantly increase the accuracy of cardiovascular risk prediction in population primary care and to identify ML programs for which results were relatively brief.

In recent years, several ML-based CVD detection models have been proposed. A review of previous research is presented to identify research problems and objectives of each study. ML helps the cardiologist to predict diseases early and treat the patient accordingly. There are many ML techniques such as Random Forest, artificial neural networks, decision trees, K-nearest neighbors (K-NN) etc.

Each one has strengths and weaknesses. These methods liver, human heart and have been applied in many fields such as the prediction of skin diseases. The results of each method vary due to several limitations.

A review of related studies shows that there is much scope for the use of new ML models that improve performance for automatic CVD detection This review involves an in-depth statistical analysis of the input data to understand the impact of available data on CVD prediction. This includes correlational studies of patient aspects occurring in different groups.

In addition, data visualization and scatter plots were obtained for pairs of significant factors to understand the relationships between significant factors, and these are discussed and analyzed in the results section.

1.1. OBJECTIVES

The main objective of heart disease risk level prediction using machine learning is to accurately assess an individual's likelihood of developing heart disease based on relevant data, enabling early intervention and personalized healthcare strategies to prevent or manage cardiovascular conditions.

- Risk Assessment Improvement
- Early Detection and Prevention
- Personalized Healthcare
- Informing Patients and Professionals
- Validation and Collaboration

1.2. MOTIVATION

A suitable motto for heart disease risk prediction using machine learning could be: "Predicting Hearts, Saving Lives: Harnessing the Power of AI for Healthier Tomorrows."

The project helps to develop a machine learning model that can predict the risk level of heart disease in individuals based on their medical and lifestyle data. This prediction can help healthcare professionals identify high-risk patients and the project involves gathering relevant medical and lifestyle data, such as age, gender, blood pressure, cholesterol levels, smoking habits, and exercise frequency. The dataset is split into training and testing sets to train the model and assess its performance.

1.3. LITERATURE SURVEY

1) Title: A Comprehensive Review on Heart Disease Prediction Using Data Mining and Machine Learning Techniques.

Contribution: This paper investigates the state of the art of various clinical decision support systems for heart disease prediction, proposed by various researchers using data mining and machine learning techniques.

Authors: Lamido Yahaya, N. Oye, E. J. Garba

Year Published: 2020

2) Title: Machine learning-based heart disease diagnosis

Contribution: Researchers primarily focused on models' performance while disregarding issues like interpretability and explain ability. In the diagnosis of heart disease, machine learning approaches help to improve data-driven decision-making.

Authors: Md Manjurul Ahsan, Zahed Siddique

Year Published: 2022

3) Title: A Systematic Literature Review on Heart Disease Prediction Using Blockchain and Machine Learning Techniques

Contribution: Emerging technologies like Machine Learning (ML) and blockchain are revolutionizing the existing healthcare infrastructure, which is a difficult task to securely and accurately forecast heart disease. This study proves that SVM works well compared to all other processes and achieves a maximum accuracy of 98.2%.

Authors: Aleeza Nouman, Salman Muneer

Year Published: 2022

CHAPTER-2

2. HEART DISEASE RISK LEVEL PREDICTION USING MACHINE LEARNING

Here we will know about the purpose and the algorithm used for this **HEART DISEASE RISK LEVEL PREDICTION USING MACHINE LEARNING.**

2.1. PURPOSE

- Early Detection: Identify individuals at high risk of heart disease. before symptoms manifest, allowing for early intervention and prevention.
- Personalized Healthcare: Provide personalized recommendations and interventions based on individual risk profiles to improve overall heart health.
- Reduce Healthcare Costs: By preventing heart disease through risk prediction and management, we can reduce the economic burden of treating advanced cardiac conditions.
- Public Health Impact: Contribute to public health efforts by providing data-driven insights for policymakers and healthcare providers to target high-risk populations.

2.2. ARCHITECTURE DIAGRAM

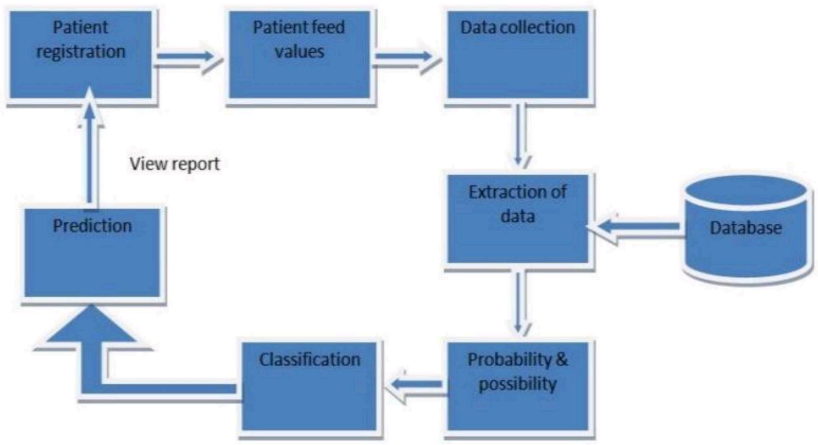


Fig - 1

2.3. ALGORITHM

Stochastic Gradient Descent Model:

```
model_sgd = 'Stochastic Gradient Descent'
sgdc = SGDClassifier(max_iter=5000, random_state=0)
sgdc.fit(X_train, y_train)
sgdc_predicted = sgdc.predict(X_test)
sgdc_conf_matrix = confusion_matrix(y_test, sgdc_predicted)
sgdc_acc_score = accuracy_score(y_test, sgdc_predicted)
print("Confussion Matrix")
print(sgdc_conf_matrix)
print("------------------------------------------------------------------")
print("------------------------------------------------------------------")
print("Accuracy of : Stochastic Gradient Descent",sgdc_acc_score*100)
print("------------------------------------------------------------------")
print("------------------------------------------------------------------")
print(classification_report(y_test,sgdc_predicted))
```

K-NeighborsClassifier Model:

```
model_knn = 'K-NeighborsClassifier'
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)
knn_predicted = knn.predict(X_test)
knn_conf_matrix = confusion_matrix(y_test, knn_predicted)
knn_acc_score = accuracy_score(y_test, knn_predicted)
print("Confussion matrix")
print(knn_conf_matrix)
print("----------------------------------------------------------------------")
print("----------------------------------------------------------------------")
print("Accuracy of K-NeighborsClassifier:",knn_acc_score*100)
print("----------------------------------------------------------------------")
print("----------------------------------------------------------------------")
print(classification_report(y_test,knn_predicted))
```

Extreme Gradient Boost:

```
model_egb = 'Extreme Gradient Boost'
xgb = XGBClassifier(learning_rate=0.01, n_estimators=25,
max_depth=15,gamma=0.6, subsample=0.52,colsample_bytree=0.6,seed=27,
                    reg_lambda=2, booster='dart', colsample_bylevel=0.6,
colsample_bynode=0.5)
xgb.fit(X_train, y_train)
xgb_predicted = xgb.predict(X_test)
xgb_conf_matrix = confusion_matrix(y_test, xgb_predicted)
xgb_acc_score = accuracy_score(y_test, xgb_predicted)
print("Confussion matrix")
print(xgb_conf_matrix)
print("----------------------------------------------------------------------")
print("----------------------------------------------------------------------")
print("Accuracy of Extreme Gradient Boost:",xgb_acc_score*100)
print("----------------------------------------------------------------------")
print("----------------------------------------------------------------------")
print(classification_report(y_test,xgb_predicted))
```

DecisionTreeClassifier Model:

```
model_dtc = 'DecisionTreeClassifier'
dt = DecisionTreeClassifier(criterion = 'entropy',random_state=0,max_depth
= 6)
dt.fit(X_train, y_train)
dt_predicted = dt.predict(X_test)
dt_conf_matrix = confusion_matrix(y_test, dt_predicted)
dt_acc_score = accuracy_score(y_test, dt_predicted)
print("Confussion matrix")
print(dt_conf_matrix)
print("----------------------------------------------------------------------")
print("----------------------------------------------------------------------")
print("Accuracy of DecisionTreeClassifier:",dt_acc_score*100)
print("----------------------------------------------------------------------")
print("----------------------------------------------------------------------")
print(classification_report(y_test,dt_predicted))
```

Random Forest Classfier Model:

```
model_rfc = 'Random Forest Classfier'
rf = RandomForestClassifier(n_estimators=20, random_state=12,max_depth=5)
rf.fit(X_train,y_train)
rf_predicted = rf.predict(X_test)
rf_conf_matrix = confusion_matrix(y_test, rf_predicted)
rf_acc_score = accuracy_score(y_test, rf_predicted)
print("13onfusion matrix")
print(rf_conf_matrix)
print("----------------------------------------------------------------------")
print("----------------------------------------------------------------------")
print("Accuracy of Random Forest:",rf_acc_score*100)
print("----------------------------------------------------------------------")
print("----------------------------------------------------------------------")
print(classification_report(y_test,rf_predicted))
```

CHAPTER-3

3.METHODOLOGY

Heart disease risk level prediction using machine learning involves several steps.

3.1. Data Collection

Gather relevant data such as medical history, lifestyle factors, and diagnostic test results from individuals.

3.2. Data Preprocessing

Clean and preprocess the data to handle missing values, standardize formats, and ensure consistency.

3.3. Feature Selection

Identify key features (variables) that contribute significantly to heart disease risk. This can include factors like age, blood pressure, cholesterol levels, and lifestyle habits.

3.4. Model Training

Utilize machine learning algorithms (e.g., logistic regression, decision trees, or neural networks) to train a predictive model based on the selected features and historical data.

3.5. Validation

Assess the model's performance using validation datasets to ensure it generalizes well to new, unseen data.

3.6. Testing and Evaluation

Test the model on independent datasets to evaluate its accuracy, sensitivity, specificity, and other relevant metrics.

By iteratively refining the model and incorporating new information, machine learning enables a dynamic and effective approach to heart disease risk prediction.

CHAPTER-4

4.RESULTS AND DISCUSSION

EXPLORATORY DATA ANALYSIS

- Understand everything about the data, the patterns etc.
- This is done by adopting statistics and data visualization.

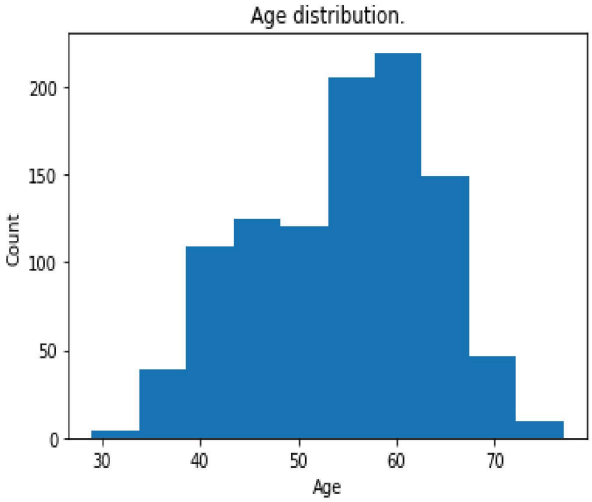


Fig – 2 Age Distribution

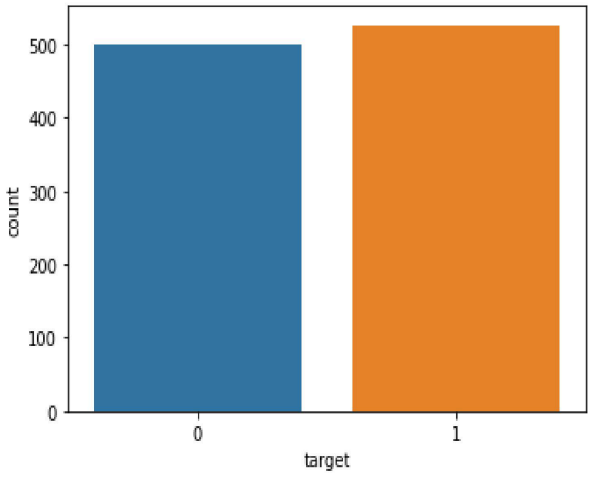


Fig – 3 Case Count

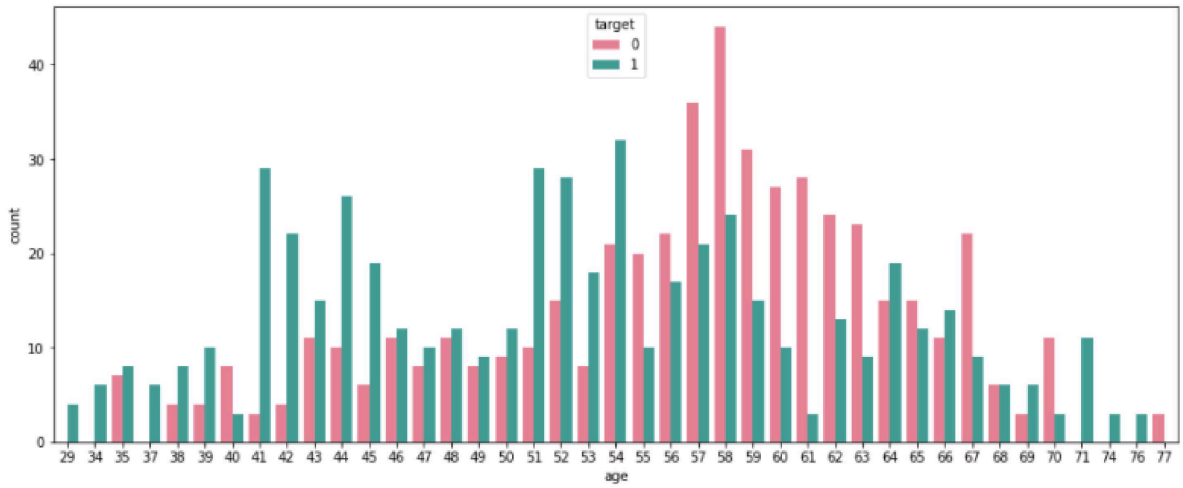


Fig – 4 Exploratory Data Analysis

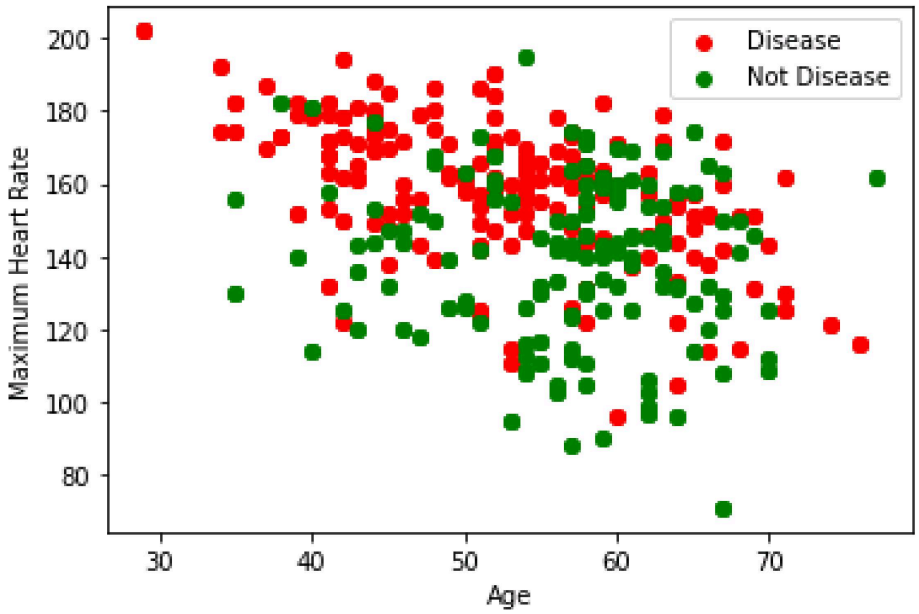


Fig – 5 Scatter plot for Heart Rate vs Age

MODELLING

Models used for the prediction:

1. Stochastic Gradient Descent: Stochastic gradient descent is an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs.
2. K-Nearest Neighbour: K Nearest Neighbour is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classify a data point based on how its neighbours are classified.
3. Decision Tree: A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems.
4. Random Forest: Random Forest is a type of algorithm which combines the output of multiple decision trees to reach a single result.
5. Extreme Gradient Boost: Extreme Gradient Boosting is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems.

S. No	Model	Accuracy	Absence of heart disease	Presence of heart disease
1.	Stochastic Gradient Descent	80.19	0.76	0.85
2.	K-Nearest Neighbour	89.61	0.88	0.91
3.	Extreme Gradient Boost	89.93	0.89	0.91
4.	Decision tree	91.88	0.88	0.95
5.	Random Forest	93.51	0.96	0.91

Table – 1 Accuracy table

- Random Forest was chosen because it has the highest accuracy and precision in predicting heart diseases.

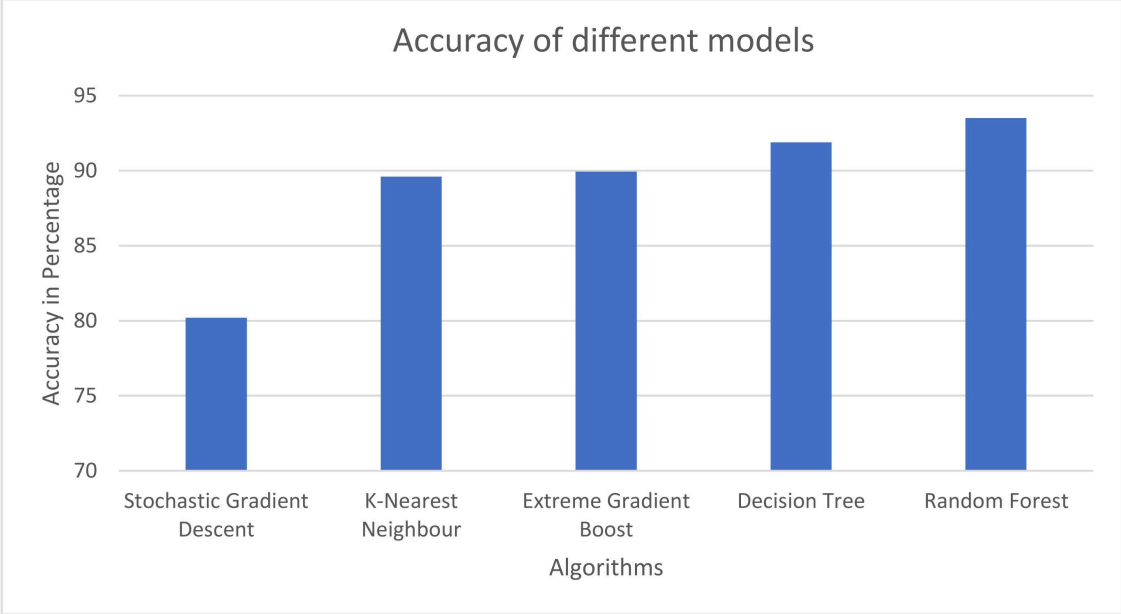


Fig – 6 Accuracy of Different Models

CHAPTER-5

5.CONCLUSION AND FUTURE SCOPE

5.1. CONCLUSION:

In conclusion, heart disease risk level prediction using machine learning presents a promising avenue for advancing personalized healthcare. By harnessing sophisticated algorithms to analyze diverse data sets, these models offer the potential to enhance the accuracy of risk assessments and enable early interventions. The continuous learning capability of machine learning further ensures adaptability to evolving health trends, contributing to more effective preventive strategies and improved patient outcomes in the realm of cardiovascular health.

5.2. FUTURE SCOPE:

The future where machine learning (ML) will be used to quantify cardiovascular risk is promising. Advances in ML algorithms, and the availability of large-scale health data, can improve the accuracy of cardiovascular disease prediction and prevention. Integrating wearable devices for real-time monitoring and personal risk assessment could be a breakthrough. Furthermore, ethical considerations, data privacy, and simple integration of health care protocols play an important role in successfully using ML to predict cardiovascular disease risk.

CHAPTER-6

6.APPENDIX

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

import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import tree

import os

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_classif
from sklearn.model_selection import KFold
from sklearn.feature_selection import SelectFromModel
from sklearn.svm import LinearSVC

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict

from sklearn.preprocessing import normalize
from sklearn.preprocessing import StandardScaler, LabelEncoder,  OneHotEncoder
from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

from xgboost import XGBClassifier
from xgboost import XGBClassifier, XGBRFClassifier
from xgboost import plot_tree, plot_importance

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE

import warnings
warnings.filterwarnings("ignore")


url="Heart_kaggle1.xlsx"
df =pd.read_excel(url)


print("Rows: ",len(df))
print("Column: ",df.shape[1])

     Rows:  1025
     Column:  14


df.head(5)

        age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal  target
```

0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0				
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0				
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0				
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0				
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0				

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.

```
df.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

```
missing_data = df.isnull().sum()
total_percentage = (missing_data.sum()/df.shape[0]) * 100
print(f'The total percentage of missing data is {round(total_percentage,2)}%')
```

```
The total percentage of missing data is 0.0%
```

```
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)
```

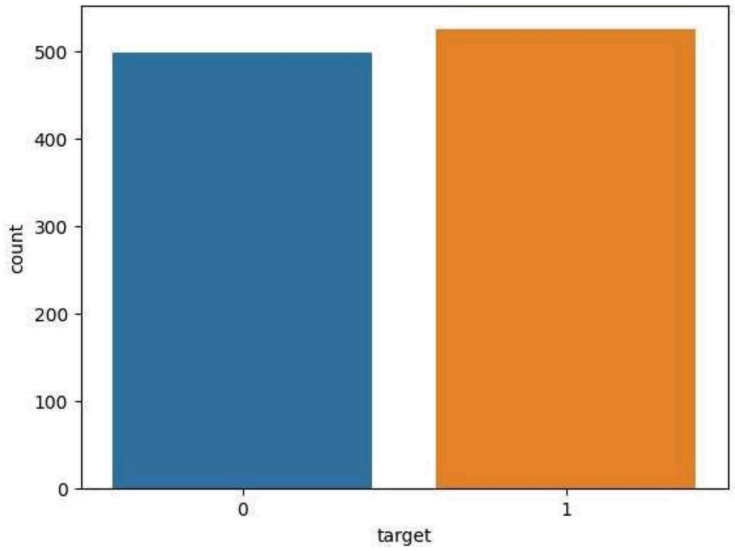
```
==============================
age : [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 63 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
==============================
sex : [1 0]
==============================
cp : [0 1 2 3]
==============================
trestbps : [125 140 145 148 138 100 114 160 120 122 112 132 118 128 124 106 104 135
 130 136 180 129 150 178 146 117 152 154 170 134 174 144 108 123 110 142
 126 192 115  94 200 165 102 105 155 172 164 156 101]
==============================
chol : [212 203 174 294 248 318 289 249 286 149 341 210 298 204 308 266 244 211
 185 223 208 252 209 307 233 319 256 327 169 131 269 196 231 213 271 263
 229 360 258 330 342 226 228 278 230 283 241 175 188 217 193 245 232 299
 288 197 315 215 164 326 207 177 257 255 187 201 220 268 267 236 303 282
 126 309 186 275 281 206 335 218 254 295 417 260 240 302 192 225 325 235
 274 234 182 167 172 321 300 199 564 157 304 222 184 354 160 247 239 246
 409 293 180 250 221 200 227 243 311 261 242 205 306 219 353 198 394 183
 237 224 265 313 340 259 270 216 264 276 322 214 273 253 176 284 305 168
 407 290 277 262 195 166 178 141]
==============================
fbs : [0 1]
==============================
restecg : [1 0 2]
==============================
thalach : [168 155 125 161 106 122 140 145 144 116 136 192 156 142 109 162 165 148
 172 173 146 179 152 117 115 112 163 147 182 105 150 151 169 166 178 132
 160 123 139 111 180 164 202 157 159 170 138 175 158 126 143 141 167  95
 190 118 103 181 108 177 134 120 171 149 154 153  88 174 114 195 133  96
 124 131 185 194 128 127 186 184 188 130  71 137  99 121 187  97  90 129
 113]
==============================
exang : [0 1]
==============================
```

```
oldpeak : [1.  3.1 2.6 0.  1.9 4.4 0.8 3.2 1.6 3.  0.7 4.2 1.5 2.2 1.1 0.3 0.4 0.6
 3.4 2.8 1.2 2.9 3.6 1.4 0.2 2.  5.6 0.9 1.8 6.2 4.  2.5 0.5 0.1 2.1 2.4
 3.8 2.3 1.3 3.5]
==============================
slope : [2 0 1]
==============================
ca : [2 0 1 3 4]
==============================
thal : [3 2 1 0]
==============================
target : [0 1]
```

```
categorical_val
```

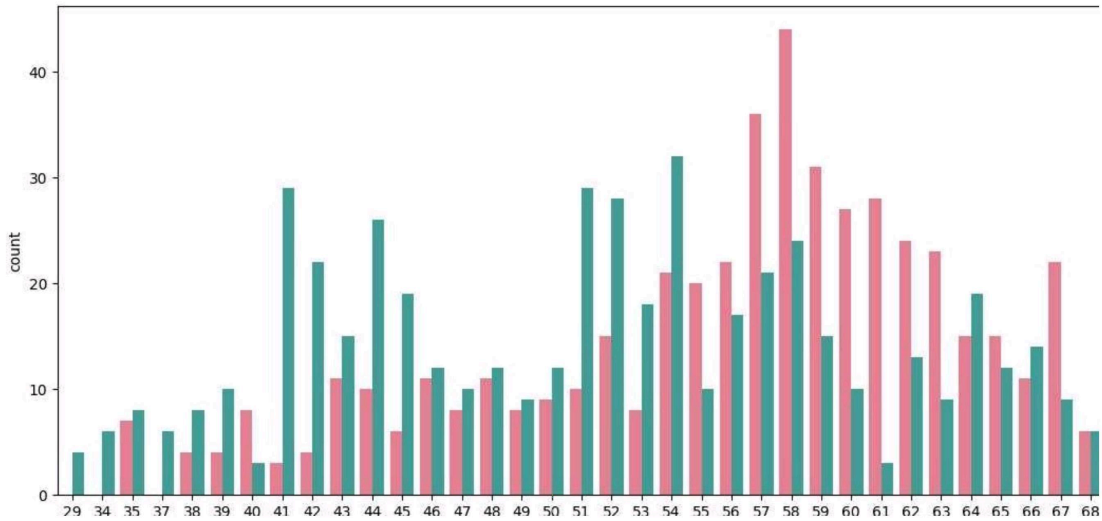
```
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
```

```
sns.countplot(x='target',data=df)
plt.show()
cases = df.target.value_counts()
print(f"There are {cases[0]} patients without heart disease and {cases[1]} patients with the disease")
```



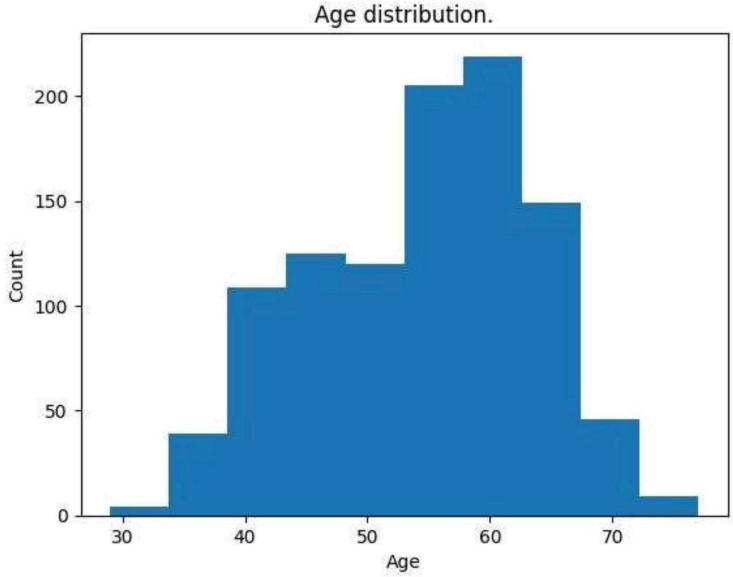

```
There are 499 patients without heart disease and  526 patients with the disease
```

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data = df,  hue = 'target',palette='husl')
plt.show()
```



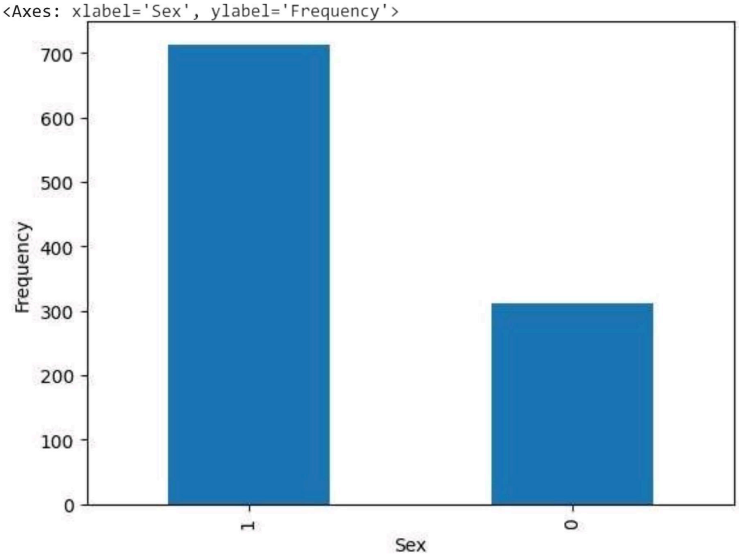
```
plt.hist(df['age'])
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age distribution.')
```

```
Text(0.5, 1.0, 'Age distribution.')
```




```
plt.xlabel('Sex')
plt.ylabel('Frequency')
df['sex'].value_counts().plot.bar()
```

```
<Axes: xlabel='Sex', ylabel='Frequency'>
```




```
minAge=min(df.age)
maxAge=max(df.age)
meanAge=df.age.mean()
print('Min Age :',minAge)
print('Max Age :',maxAge)
print('Mean Age :',meanAge)
```

```
Min Age : 29
Max Age : 77
Mean Age : 54.43414634146342
```

```
presence = df[df['target']==1]
absence = df[df['target']==0]
```

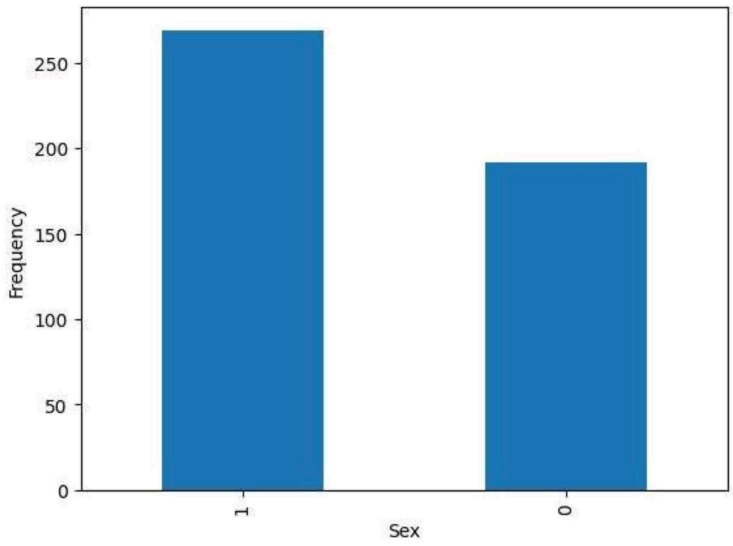
```
check = presence[presence['age']>40]
```

```
check = check[check['age']<70]
check
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	1
16	51	0	2	140	308	0	0	142	0	1.5	2	1	2	1
18	50	0	1	120	244	0	1	162	0	1.1	2	0	2	1
19	58	1	2	140	211	1	0	165	0	0.0	2	0	2	1
21	67	0	0	106	223	0	1	142	0	0.3	2	2	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1011	45	1	1	128	308	0	0	170	0	0.0	2	0	2	1
1014	44	0	2	108	141	0	1	175	0	0.6	1	0	2	1
1019	47	1	0	112	204	0	1	143	0	0.1	2	0	2	1
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1

461 rows × 14 columns

```
male = check[check['sex']==1]
female = check[check['sex']==0]


plt.xlabel('Sex')
plt.ylabel('Frequency')
check['sex'].value_counts().plot.bar()
plt.show()
```




```
check.shape
```

```
(461, 14)
```

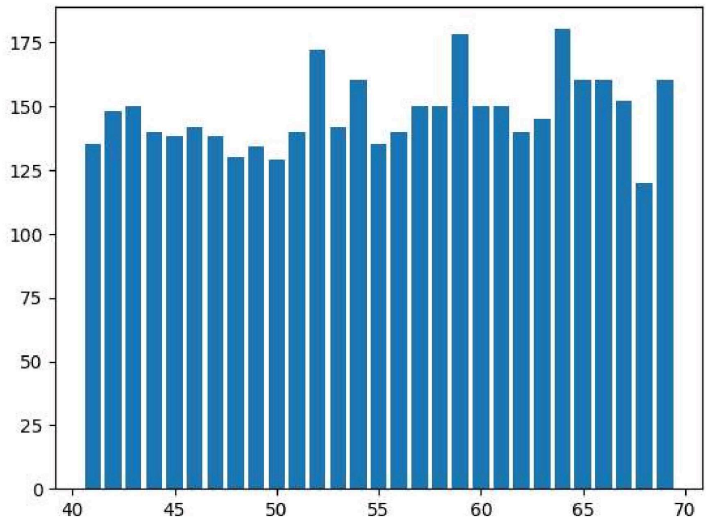
```
print(f"GENDER")
cases = check.sex.value_counts()
print(f"Gender")
print(f"Female = {cases[0]}\nMale = {cases[1]}\n")

print(f"CHEST PAIN")
cases = check.cp.value_counts()
print(f"Chest Pain")
print(f"Typical = {cases[0]}\nAtypical = {cases[1]}\nNon-anginal = {cases[2]}\nasymptomatic = {cases[3]}\n")

print(f"FASTING BLOOD SUGAR")
cases = check.fbs.value_counts()
print(f"Fasting Blood Sugar >120 mg/dL")
print(f"False = {cases[0]}\nTrue = {cases[1]}\n")
```

```
print(f"EXERCISE INDUCED ANGINA")
cases = check.exang.value_counts()
print(f"Yes = {cases[0]}\nNo = {cases[1]}")
```

```
GENDER
Gender
Female = 192
Male = 269

CHEST PAIN
Chest Pain
Typical = 114
Atypical = 114
Non-anginal = 188
asymptomatic = 45

FASTING BLOOD SUGAR
Fasting Blood Sugar >120 mg/dL
False = 394
True = 67

EXERCISE INDUCED ANGINA
Yes = 396
No = 65
```

```
x=check['age']
y=check['trestbps']
plt.bar(x,y)
plt.show()
```

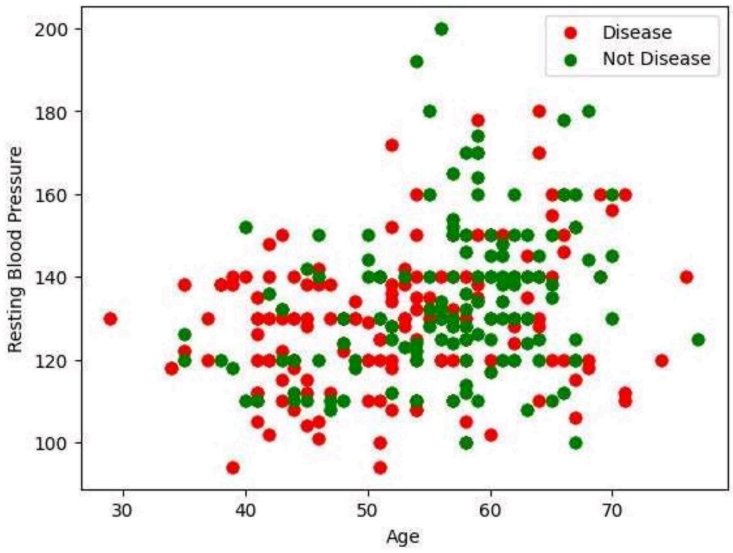
```
mincp=min(check.trestbps)
maxcp=max(check.trestbps)
print('Minimum Resting Blood Pressure:',mincp)
print('Maximum Resting Blood Pressure:',maxcp)
cases = check.trestbps.value_counts()
count = 0
for i in check['trestbps'] :
    if i > 120 :
        count = count+1
print(f"High blood pressure count = ",count)

count = 0
for i in check['trestbps'] :
    if i <80 :
        count = count+1
print(f"Low blood pressure count = ",count)
```



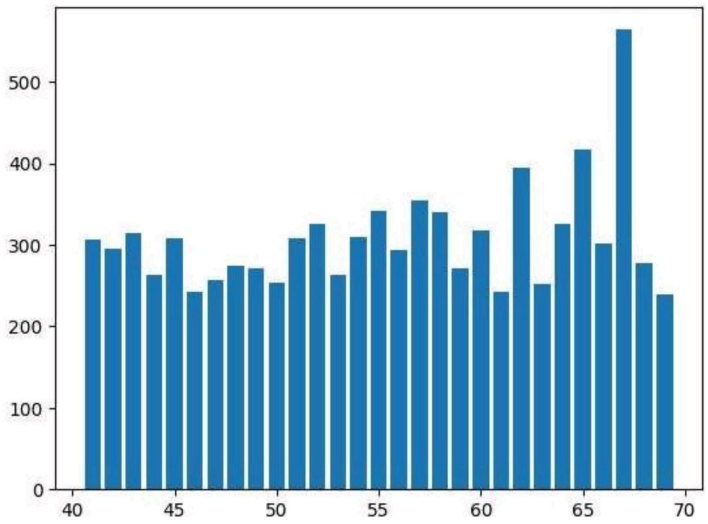

```
Minimum Resting Blood Pressure: 94
Maximum Resting Blood Pressure: 180
High blood pressure count =  293
Low blood pressure count =  0
```

```
plt.scatter(x=df.age[df.target==1], y=df.trestbps[(df.target==1)],  c="red")
plt.scatter(x=df.age[df.target==0], y=df.trestbps[(df.target==0)],  c="green")
plt.legend(["Disease", "Not Disease"])
```

```
plt.xlabel("Age")
plt.ylabel("Resting Blood Pressure")
plt.show()
```

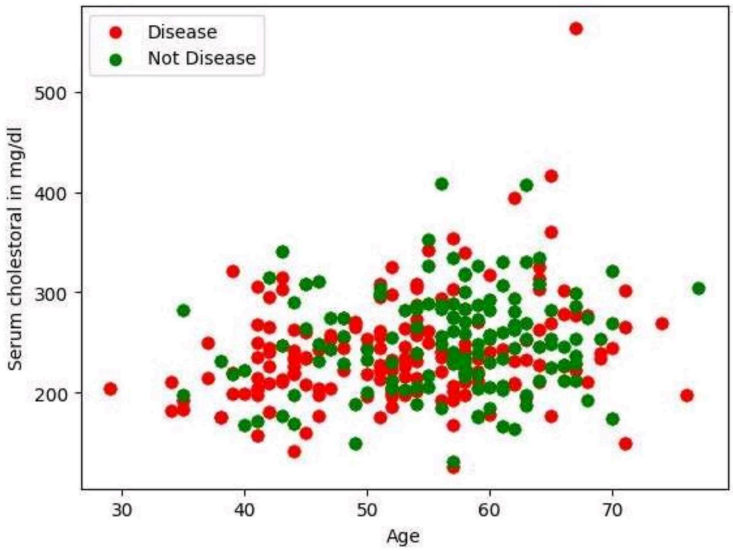



```
x=check['age']
y=check['chol']
plt.bar(x,y)
plt.show()

mincp=min(check.chol)
maxcp=max(check.chol)
print('Minimum Serum cholestoral:',mincp)
print('Maximum Serum cholestoral:',maxcp)

count = 0
for i in check['chol'] :
    if i > 200 :
        count = count+1
print(f"Higher Serum Cholestoral count = ",count)
```

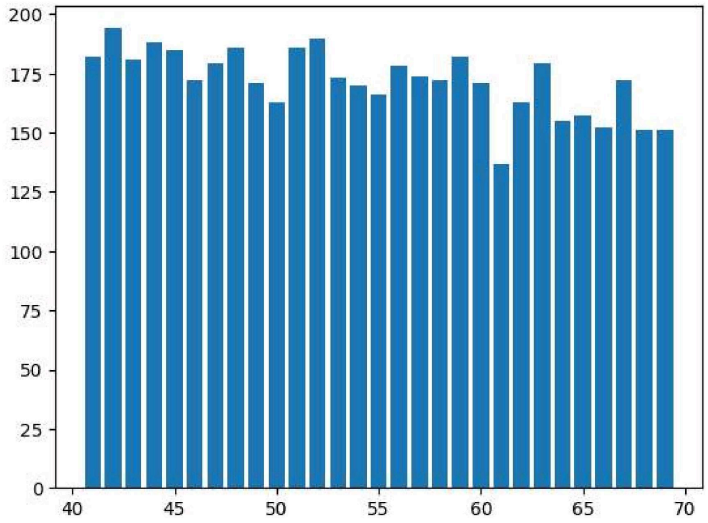



```
Minimum Serum cholestoral: 126
Maximum Serum cholestoral: 564
Higher Serum Cholestoral count =  394
```

```
plt.scatter(x=df.age[df.target==1], y=df.chol[(df.target==1)], c="red")
plt.scatter(x=df.age[df.target==0], y=df.chol[(df.target==0)], c="green")
plt.legend(["Disease", "Not Disease"])
plt.xlabel("Age")
plt.ylabel("Serum cholestoral in mg/dl")
plt.show()
```

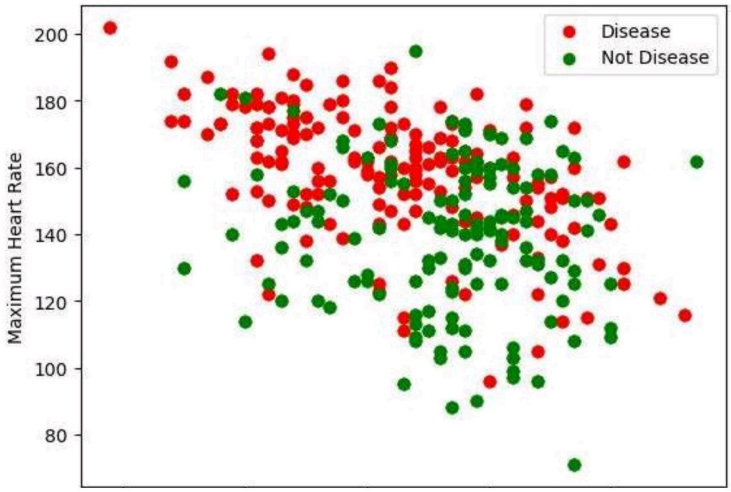



```
x=check['age']
y=check['thalach']
plt.bar(x,y)
plt.show()

mincp=min(check.thalach)
maxcp=max(check.thalach)
print('Minimum heart rate:',mincp)
print('Maximum heat rate:',maxcp)

count = 0
for i in check['thalach'] :
    if i > 80:
        count = count+1
print(f"Higher heart rate count = ",count)
```

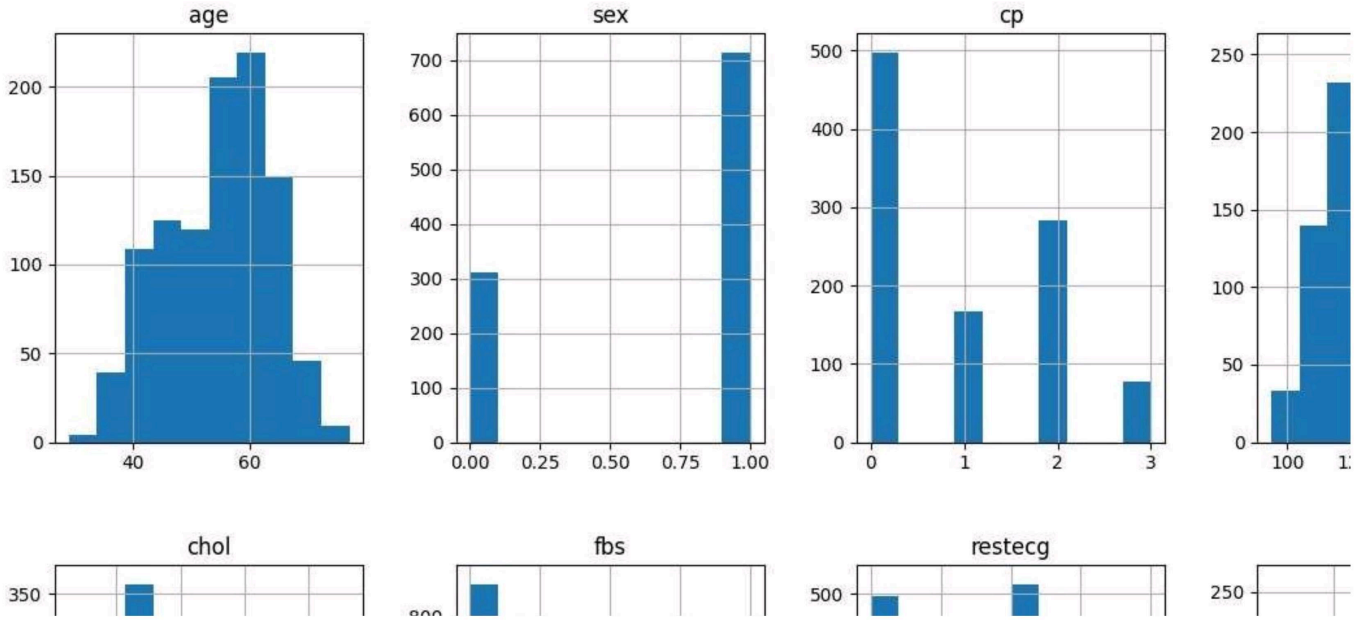



```
Minimum heart rate: 96
Maximum heat rate: 194
Higher heart rate count =  461
```

```
plt.scatter(x=df.age[df.target==1], y=df.thalach[(df.target==1)], c="red")
plt.scatter(x=df.age[df.target==0], y=df.thalach[(df.target==0)], c="green")
plt.legend(["Disease", "Not Disease"])
plt.xlabel("Age")
plt.ylabel("Maximum Heart Rate")
plt.show()
```




```
fig = plt.figure(figsize = (15,20))
ax = fig.gca()
df.hist(ax = ax)
plt.show()
```




```
y = df["target"]
X = df.drop('target',axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state = 0)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
model_sgd = 'Stochastic Gradient Descent'
sgdc = SGDClassifier(max_iter=5000, random_state=0)
sgdc.fit(X_train, y_train)
sgdc_predicted = sgdc.predict(X_test)
sgdc_conf_matrix = confusion_matrix(y_test, sgdc_predicted)
sgdc_acc_score = accuracy_score(y_test, sgdc_predicted)
print("Confussion Matrix")
print(sgdc_conf_matrix)
print("------------------------------------------------------------------")
print("------------------------------------------------------------------")
print("Accuracy of : Stochastic Gradient Descent",sgdc_acc_score*100)
print("------------------------------------------------------------------")
print("------------------------------------------------------------------")
print(classification_report(y_test,sgdc_predicted))
```

```
Confussion Matrix
[[124  21]
 [ 40 123]]
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
Accuracy of : Stochastic Gradient Descent 80.1948051948052
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.76      0.86      0.80       145
           1       0.85      0.75      0.80       163

    accuracy                           0.80       308
   macro avg       0.81      0.80      0.80       308
weighted avg       0.81      0.80      0.80       308
```

```
model_knn = 'K-NeighborsClassifier'
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)
knn_predicted = knn.predict(X_test)
knn_conf_matrix = confusion_matrix(y_test, knn_predicted)
knn_acc_score = accuracy_score(y_test, knn_predicted)
print("Confussion matrix")
print(knn_conf_matrix)
print("------------------------------------------------------------------")
```

```
print("-------------------------------------------------------------------------------------------")
print("Accuracy of K-NeighborsClassifier:",knn_acc_score*100)
print("-------------------------------------------------------------------------------------------")
print("-------------------------------------------------------------------------------------------")
print(classification_report(y_test,knn_predicted))
```

```
Confussion matrix
[[130  15]
 [ 17 146]]
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
Accuracy of K-NeighborsClassifier: 89.6103896103896
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       145
           1       0.91      0.90      0.90       163

    accuracy                           0.90       308
   macro avg       0.90      0.90      0.90       308
weighted avg       0.90      0.90      0.90       308
```

```
model_egb = 'Extreme Gradient Boost'
xgb = XGBClassifier(learning_rate=0.01, n_estimators=25, max_depth=15,gamma=0.6, subsample=0.52,colsample_bytree=0.6,seed=27,
                    reg_lambda=2, booster='dart', colsample_bylevel=0.6, colsample_bynode=0.5)
xgb.fit(X_train, y_train)
xgb_predicted = xgb.predict(X_test)
xgb_conf_matrix = confusion_matrix(y_test, xgb_predicted)
xgb_acc_score = accuracy_score(y_test, xgb_predicted)
print("Confussion matrix")
print(xgb_conf_matrix)
print("-------------------------------------------------------------------------------------------")
print("-------------------------------------------------------------------------------------------")
print("Accuracy of Extreme Gradient Boost:",xgb_acc_score*100)
print("-------------------------------------------------------------------------------------------")
print("-------------------------------------------------------------------------------------------")
print(classification_report(y_test,xgb_predicted))
```

```
Confussion matrix
[[127  18]
 [ 12 151]]
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
Accuracy of Extreme Gradient Boost: 90.25974025974025
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.91      0.88      0.89       145
           1       0.89      0.93      0.91       163

    accuracy                           0.90       308
   macro avg       0.90      0.90      0.90       308
weighted avg       0.90      0.90      0.90       308
```

```
model_dtc = 'DecisionTreeClassifier'
dt = DecisionTreeClassifier(criterion = 'entropy',random_state=0,max_depth = 6)
dt.fit(X_train, y_train)
dt_predicted = dt.predict(X_test)
dt_conf_matrix = confusion_matrix(y_test, dt_predicted)
dt_acc_score = accuracy_score(y_test, dt_predicted)
print("Confussion matrix")
print(dt_conf_matrix)
print("-------------------------------------------------------------------------------------------")
print("-------------------------------------------------------------------------------------------")
print("Accuracy of DecisionTreeClassifier:",dt_acc_score*100)
print("-------------------------------------------------------------------------------------------")
print("-------------------------------------------------------------------------------------------")
print(classification_report(y_test,dt_predicted))
```

```
Confussion matrix
[[138   7]
 [ 18 145]]
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
Accuracy of DecisionTreeClassifier: 91.88311688311688
```

```
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.88      0.95      0.92       145
           1       0.95      0.89      0.92       163

    accuracy                           0.92       308
   macro avg       0.92      0.92      0.92       308
weighted avg       0.92      0.92      0.92       308
```

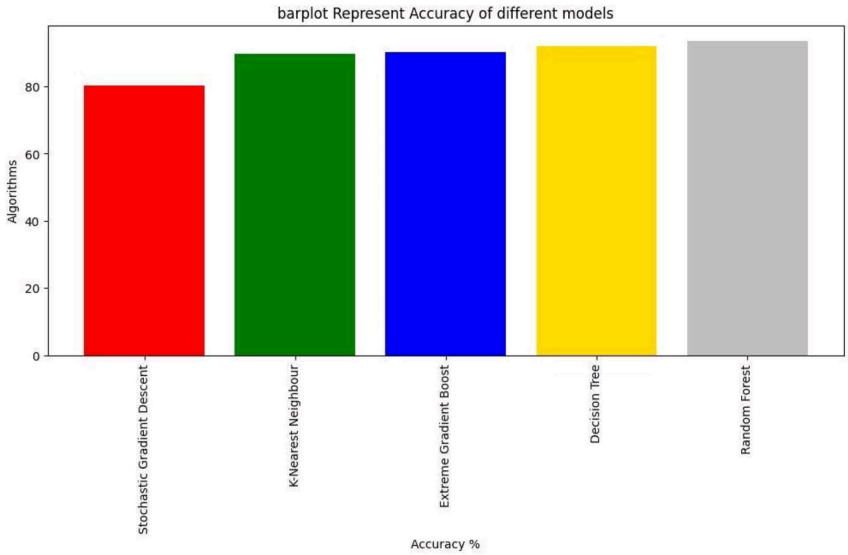
```
model_rfc = 'Random Forest Classfier'
rf = RandomForestClassifier(n_estimators=20, random_state=12,max_depth=5)
rf.fit(X_train,y_train)
rf_predicted = rf.predict(X_test)
rf_conf_matrix = confusion_matrix(y_test, rf_predicted)
rf_acc_score = accuracy_score(y_test, rf_predicted)
print("Confussion matrix")
print(rf_conf_matrix)
print("------------------------------------------------------------------")
print("------------------------------------------------------------------")
print("Accuracy of Random Forest:",rf_acc_score*100)
print("------------------------------------------------------------------")
print("------------------------------------------------------------------")
print(classification_report(y_test,rf_predicted))
```

```
Confussion matrix
[[130  15]
 [  5 158]]
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
Accuracy of Random Forest: 93.5064935064935
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.96      0.90      0.93       145
           1       0.91      0.97      0.94       163

    accuracy                           0.94       308
   macro avg       0.94      0.93      0.93       308
weighted avg       0.94      0.94      0.93       308
```

```
model_ev = pd.DataFrame({'Model': ['Stochastic Gradient Descent','K-Nearest Neighbour','Extreme Gradient Boost','Decision Tree','Random Fores
                        'Accuracy':[ sgdc_acc_score*100, knn_acc_score*100,xgb_acc_score*100,dt_acc_score*100, rf_acc_score*100 ]})
model_ev
```

	Model	Accuracy
0	Stochastic Gradient Descent	80.194805
1	K-Nearest Neighbour	89.610390
2	Extreme Gradient Boost	90.259740
3	Decision Tree	91.883117
4	Random Forest	93.506494

```
colors = ['red','green','blue','gold','silver','yellow','orange','magenta', 'cyan']
plt.figure(figsize=(12,5))
plt.title("barplot Represent Accuracy of different models")
plt.xlabel("Accuracy %")
plt.xticks(rotation=90)
plt.ylabel("Algorithms")
plt.bar(model_ev['Model'],model_ev['Accuracy'],color = colors)
plt.show()
```


barplot Represent Accuracy of different models
Algorithms
0
20
40
60
80
Stochastic Gradient Descent
K-Nearest Neighbour
Extreme Gradient Boost
Decision Tree
Random Forest
Accuracy %

CHAPTER-7

7.REFERRENCES

1. Adil Hussain Seh, Dr. Pawan Kumar Chaurasia"A Review on Heart Disease Prediction Using Machine Learning Techniques", International Journal of Mangement, IT & Engineering, Volume 9, Issue 4, April 2019
2. Abhijeet Jagtap , Priya Malewadkar , Omkar Baswat , Harshali Rambade, "Heart Disease Prediction using Machine Learning ", International Journal of Research in Engineering, Science and Management Volume-2, Issue-2, February-2019, ISSN (Online): 2581-5792
3. Animesh Hazra, Subrata Kumar Mandal, Amit Gupta, Arkomita Mukherjee and Asmita Mukherjee,"Heart Disease Diagnosis and Prediction Using Machine Learning and Data Mining Techniques: A Review", Research India Publications, Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 10, Number 7 (2017) pp. 2137-2159.
4. Jaymin Patel, Prof.TejalUpadhyay, Dr. Samir Patel, "Heart Disease Prediction Using Machine learning and Data Mining Technique", IJCSC Volume 7, Number 1 Sept 2015 – March 2016, pp. 129-137
5. G. Parthiban, S.K.Srivatsa, "Applying Machine Learning Methods in Diagnosing Heart Disease for Diabetic Patients", International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 3– No.7, August 2012
6. Sonam Nikhar , A.M. Karandikar, "Prediction of Heart Disease Using Machine Learning Algorithms", International Journal of Advanced Engineering, Management and Science (IJAEMS), [Vol-2, Issue-6, June-2016], ISSN : 2454-1311

7. Amruta Powar and Prof. Dr. Vijay Ghorpade, “Heart Disease Prediction System Using Naïve Bayes Data Mining Technique”, wjert, 2018, Vol. 4, Issue 4, 313-318, ISSN 2454-695
8. M. Marimuthu, M. Abinaya, K. S. Hariesh, K. Madhankumar, K. Madhankumar, “A Review on Heart Disease Prediction using Machine Learning and Data Analytics Approach”, International Journal of Computer Applications (0975 – 8887) Volume 181 – No. 18, September 2018
9. Mr. Amol A. Wghmode, Mr. Darpan Sawant, Prof. Deven D. Ketkar, “Heart Disease Prediction Using Data Mining Techniques”, Heart Disease Prediction Using Data Mining Techniques, ISSN 2394 – 3386 Volume 4, Issue 10 October
10. Ajad Patel, Sonali Gandhi, Swetha Shetty,Prof. Bhanu Tekwan, “Heart Disease Prediction Using Data Mining”, International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 01 | Jan -2017, e-ISSN: 2395 -0056, p-ISSN: 2395-0072
11. Prof. Mamta Sharma , Farheen Khan , Vishnupriya Ravichandran, Comparing Data Mining Techniques Used For Heart Disease Prediction, International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 06 | June -2017, e-ISSN: 2395 -0056 , p-ISSN: 2395-0072
12. Devansh Shah1 · Samir Patel1 · Santosh Kumar Bharti, “Heart Disease Prediction using Machine Learning Techniques”, Springer Nature Singapore Pte Ltd 2020, 16 October 2020
13. Smriti Mukesh Singh, Dr. Dinesh B. Hanchate, “Improving Disease Prediction by Machine Learning”, International Research Journal of Engineering and Technology (IRJET), Volume: 05 Issue: 06 | June-2018, e-ISSN: 2395-0056, p-ISSN: 2395-0072