Golden Sun-Rise
International Journal of Multidisciplinary on Science and Management
ISSN: 3048-5037/ Volume 2 Issue 2 Apr-Jun 2025 / Page No: 36-56
Paper Id: IJMSM- V2I2P104/ Doi:10.71141/30485037/V2I2P104

Research Article

Predicting Heart Disease with Machine Learning: Enhancing Accuracy through Algorithmic Approach

Vitthal B Kamble[1], Bhoomi Gulabani[2], Srushti Narkhede[3], Shital Godse[4]

Department of Computer Engineering, Cusrow Wadia Institute of Technology, Pune, Maharashtra, India.

Received: 30 March 2025 Revised: 06 April 2025 Accepted: 11 April 2025 Published: 20 April 2025

Abstract - *Cardiovascular disease, or CVD, kills millions of people each year worldwide due to a variety of intricate and interrelated causal factors. Because it is the leading cause of death in the world, the detection and diagnosis of heart disease at an early stage is regarded as being of the utmost significance in public health. Over the last few years, advances in machine learning technology have been an invaluable resource in the healthcare sector, allowing precise predictions of disease when such programs are well-trained and rigorously tested. In cardiac-related diseases, early detection is most crucial to allow patients to receive the best treatment possible before it becomes a critical and potentially fatal condition. The accuracy of such high-accuracy predictions is not only worth its weight in gold for enhanced patient outcomes but also for saving lives ultimately. This study was thus conducted with the aim of creating a successful model for heart disease prediction specifically using the Random Forest algorithm. With the use of the random forest algorithm in our study research, we achieved a remarkable accuracy rate of 100%.*

Keywords - *Artificial Intelligence, Data Analysis, Heart Disease, Machine Learning, Prediction Model, Random Forest Algorithm.*

I. INTRODUCTION

The heart is the most essential organ in the human body. Without a heart, the brain and countless other organs will cease working, and the existence will die in a few twinkles. Due to changes in life, work pressure, and bad dieting, many heart-related issues are on the rise [1]. Heart disease, also known as cardiovascular disease (CVD), refers to a range of conditions that affect the heart and blood vessels. It is a leading cause of death worldwide and has a substantial impact on human health. Heart disease is a serious threat to human health. Many research works related to mortality statistics show that one of the most prevalent causes of death is cardiac disease [2]. An estimated 19.9 million people died from CVDs in 2019, representing 32% of all global deaths. Of these deaths, 85% were due to heart attack and stroke [3].

Heart disease symptoms may include pounding or racing heart (palpitations), sweating, lightheadness, shortness of breath, dizziness or sudden unexplained loss of consciousness, chest or upper body pain, pressure, heaviness or discomfort, neck pain, heartburn or indigestion, nausea or vomiting, exhaustion and some more. ML has been shown to be a significant tool for heart disease prediction and management using complex algorithms to analyze complicated data and the choice of high-risk factors. The studies focused on diverse techniques of ML, and all of them present possible applications in the early diagnosis and improvement of the patient's outcome. In fact, the ML models may offer comprehensive information about cardiovascular health through the introduction of a vast number of clinical features, such as age, blood pressure, cholesterol, and lifestyle factors. Future developments in this area promise to open the door to synergy between machine learning techniques and clinical expertise in the transformation of prospects of managing cardiovascular health [4].

The Random Forest algorithm has been used here as the prediction model for heart disease detection. This machine learning approach leverages multiple decision trees to enhance prediction accuracy, identifying patterns and correlations in patient data to diagnose heart disease effectively. Subsequent sections will cover areas like methodologies and implementation, which will satisfy the title of enhancing accuracy through Random Forest Algorithm.

II. LITERATURE REVIEW

Healthcare professionals relied on several conventional methods for cardiovascular disease prediction. An example of conventional methods of cardiovascular disease prediction includes clinical risk factors that are related to age, gender, family history, and personal medical history. In addition, echocardiography can be used for visualization of the heart function [5] where electrocardiogram (ECG) can be used to detect signs of congestive heart failure [6].

ECG can help in the prognosis and treatment management of patients diagnosed with congestive heart failure [7]. Cardiac catheterization can also be used to diagnose and evaluate coronary artery disease (CAD) and typical issues with the heart and blood vessels [8,9]. However, recently, there has been a growing need for more advanced predictive models, such as those powered by machine learning, to improve the accuracy and efficiency of cardiovascular disease prediction [10]. Machine learning is a subset of artificial intelligence (AI) that uses algorithms to allow computer agents to perceive, acquire knowledge, identify patterns, and make intelligent decisions by analyzing collected data [11–14]. With its ability to evaluate enormous amounts of patient data, machine learning has emerged as a key player for achieving accurate and trustworthy cardiovascular disease prediction [16]. The predictive power of machine learning techniques has emerged as a promising path for revolutionizing the management of cardiovascular disease [16,20,21]. Machine learning can enhance early disease detection, accelerate the development of drugs, provide data-driven insights, enable remote monitoring, acquire crucial information from patient's datasets, allow data-driven decision-making, improve image and speech recognition, and simplify administrative procedures [22]. This enables early detection, often before symptoms become severe, allowing for timely intervention and treatment, hence potentially lowering healthcare costs. Analyzing such vast amounts of data in the healthcare field is challenging for humans, if not nearly impossible [23]. Hence, the prevalent use of machine learning proves invaluable in extracting meaningful insights from such extensive datasets. Machine learning algorithms are very beneficial for remote healthcare monitoring.

In recent years, the healthcare industry has made significant strides in leveraging data mining and machine learning techniques, particularly within medical cardiology. These technologies have proven effective across various healthcare applications, aiming to identify risk factors and early signs of heart disease, a leading cause of mortality in developing nations [12–16]. Narain et al. (2016) [17] conducted a study focusing on enhancing the accuracy of cardiovascular disease (CVD) prediction using a novel machine-learning-based system. Neural approach, utilizing a quantum neural network, demonstrated an impressive forecasting accuracy of 98.57%, far surpassing the 19.22% accuracy of the widely used Framingham risk score (FRS) and other existing methods. This advancement suggests promising potential for doctors in improving treatment plans and enabling early diagnosis. Shah et al. (2020) [18] aimed to develop a predictive model for cardiovascular disease using machine learning techniques. They employed various supervised classification methods on the Cleveland heart disease dataset, achieving the highest accuracy of 90.8% with the k-nearest neighbor (KNN) model. This underscores the effectiveness of machine learning in cardiovascular disease prediction and stresses the importance of selecting appropriate models for optimal outcomes. Drod et al. (2022) focused on identifying significant risk variables for cardiovascular disease in patients with metabolic associated fatty liver disease (MAFLD) using machine learning techniques. Their study highlighted the success of multiple logistic regression, univariate feature ranking, and principal component analysis (PCA) in identifying critical clinical characteristics. The model achieved an AUC of 0.87, demonstrating its utility in detecting high-risk CVD patients. Alotalibi (2019) [24] explored machine learning techniques for predicting heart failure using data from the Cleveland Clinic Foundation. Their study found that the decision tree algorithm exhibited the highest accuracy of 93.19%, followed closely by support vector machines (SVM) at 92.30%. This research underscores the potential of machine learning as a robust tool for heart disease predict [28].

III. EXISTING METHODS USED

A. Logistic Regression

Logistic regression was used to test hypotheses about the relationships of outcome variables with predictor variables [25]. Logistic regression does not require normally distributed data compared with discriminant analysis [26]. Logistic regression helps one to predict the discrete outcome from a variety of variables. Assumes a direct correlation between the log chances of outgrowth and independent factors, which could miss intricate patterns. Limited to double issues unless extended, complicating multi-class prognostications [30].

B. Random Forest

RF is a machine-learning approach that builds several decision trees on training data sets to generate a classification model. This algorithm decides on a tree based on most selections, which offers great accuracy when working with huge data sets [27]. This algorithm combines two feature selection strategies, bagging, and random selection, to produce a more effective ensemble model. Using several trees with the RF approach decreases the danger of overfitting and training time. It also gives estimates of crucial classification variables as well as missing data, all of which lead to improved accuracy [28].

a. Benefits

- Mitigates decision tree overfitting by averaging predictions.
- It offers robustness, high accuracy, and handling of non-linearity.
- Provides accurate and robust results, even in the presence of noisy or complex data.
- Robust, with effective handling of high-dimensional data. Less sensitive to noisy data.[32]

C. SVM

The support Vector Machine classification approach works with both linear and non-linear data. To categorize the data, it employs a nonlinear mapping to transfer the original training data into a higher dimension and then searches for a linear optimal separation hyperplane. It is located on the mapping that gives a high hyperplane as the dividing line between two data points.

a. Limitations

Computationally ferocious, which results in prolonged training durations, particularly when dealing with huge datasets. The kernel and hyperparameter selection, which might be delicate to acclimate, have a significant impact on performance. [31]

D. K –Neighbour

K-Nearest Neighbor is a sort of lazy learner classifier that is based on learning by similarity, that is, it compares a given test characteristic with training attributes that are similar to it. As stated in Ref. [28], K-NN is mostly used in pattern recognition and statistical estimation, both of which are considered non-parametric tools. According to Jiawei et al. (data mining concepts, second edition book), if there are n attributes for training, each attribute is represented by an n-dimensional space [32].

If the k-nearest neighbor classifier is given an unknown attribute, such as a test attribute, it searches the pattern space for the k-training attributes that are close to the unknown attribute. Indicating that the k training attributes are the unknown attribute's nearest neighbours. The proximity of attributes is measured using distance metrics such as the Euclidean distance[29].

E. Naive Bayes

Naive Bayes is a class of probabilistic bracket algorithms embedded in the operation of Bayes' theorem, therefore making a strong independence supposition between features. It is particularly useful for bracket problems wherein input variables are categorical. Indeed, it can easily handle veritably large datasets containing vast numbers of categorical variables, in fact those that occur most frequently in heart complaint vaticination scripts based on gender, type of casket pain, and other clinical pointers. It is very simple and fast and, therefore, nearly ideal for large-scale operations [33].

a. Limitations

Assumes feature independence, which could not be the case for data on cardiac complaints. Sensitive to imbalanced datasets, potentially turning prognostications towards the maturity class [31].

F. XGB

XGBoost, which stands for "Extreme Gradient Boosting," is a popular machine learning model that has been used for a variety of tasks such as regression, classification, and ranking. XGBoost creates a model in the form of boosting an ensemble of weak classification trees by gradient descent which provides optimization to the loss factor (Cui et al., 2017). It is an ensemble learning algorithm that combines multiple decision tree models to improve the accuracy and robustness of predictions [31].

G. Extreme Gradient Boost

GBDT is a commonly used training method. It aims to improve predictive performance by iteratively training a series of decision tree models. Each round of training produces a weak learner that minimizes the residual loss of the current model. As the iterations progress, each new model is trained on the residuals of the previous model, eventually forming a powerful ensemble model [29].

IV. EXISTING IMPLEMENTATION METHODS

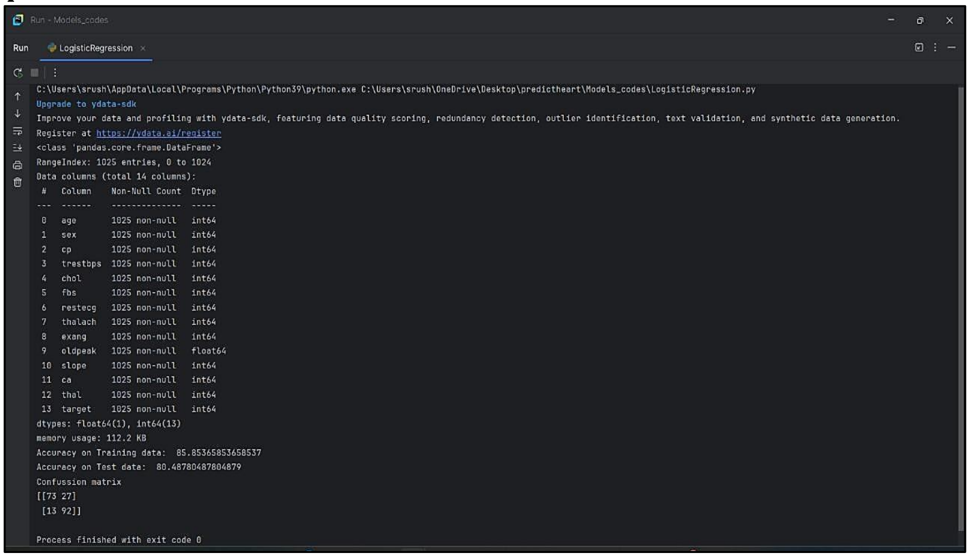
A. Logistic Regression Method

Implementation code

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score
from ydata_profiling import ProfileReport
heart_data = pd.read_csv('C:/Users/srush/Downloads/heart.csv')
heart_data.head()
heart_data.shape
heart_data.tail()
heart_data.shape
heart_data.info()
heart_data.isnull().sum()
heart_data.describe()
heart_data["target"].value_counts()
ProfileReport(heart_data)
X = heart_data.drop(columns = "target", axis=1)
Y= heart_data["target"]
X.shape
Y.shape
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X.shape, X_train.shape, X_test.shape
Y.shape, y_train.shape, y_test.shape
m1="Logistic Regression"
lr_model = LogisticRegression(solver="liblinear",random_state=2 )
lr_model.fit(X_train, y_train)
X_train_prediction = lr_model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,y_train)
print(f"Accuracy on Training data: ",training_data_accuracy*100)
```

```
X_test_prediction = lr_model.predict(X_test)
test_data_accuracy_lr = accuracy_score(X_test_prediction, y_test)
print(f"Accuracy on Test data: ",test_data_accuracy_lr*100)
LR_conf_matrix = confusion_matrix(y_test, X_test_prediction)
print("Confussion matrix")
print(LR_conf_matrix)
```

Output



```
C:\Users\srush\AppData\Local\Programs\Python\Python39\python.exe C:\Users\srush\OneDrive\Desktop\predictheart\Models_codes\LogisticRegression.py
Upgrade to ydata-sdk
Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.
Register at https://ydata.ai/register
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
Accuracy on Training data:  85.85365853658537
Accuracy on Test data:  80.48780487804879
Confussion matrix
[[73 27]
 [13 92]]

Process finished with exit code 0
```

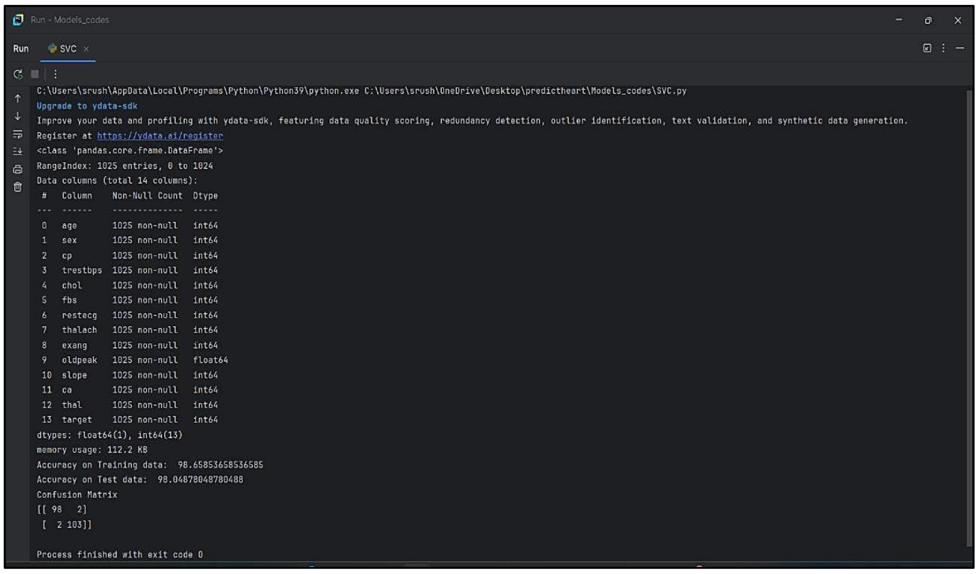
B. SVC

Implementation Code

```
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score
from ydata_profiling import ProfileReport
from sklearn.svm import SVC
heart_data = pd.read_csv('C:/Users/srush/Downloads/heart.csv')
heart_data.head()
heart_data.shape
heart_data.tail()
heart_data.shape
heart_data.info()
heart_data.isnull().sum()
heart_data.describe()
heart_data["target"].value_counts()
ProfileReport(heart_data)
X = heart_data.drop(columns = "target", axis=1)
Y= heart_data["target"]
X.shape
Y.shape
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X.shape, X_train.shape, X_test.shape
```

```
Y.shape, y_train.shape, y_test.shape
m7="svc"
svc = SVC(C=5,kernel="rbf")
svc.fit(X_train, y_train)
X_train_predict= svc.predict(X_train)
training_data_accuracy=accuracy_score(X_train_predict, y_train)
print(f"Accuracy on Training data: ",training_data_accuracy*100)
X_test_predict= svc.predict(X_test)
test_data_accuracy_svc=accuracy_score(X_test_predict, y_test)
print(f"Accuracy on Test data: ",test_data_accuracy_svc*100)
knn_confusion_matrix = confusion_matrix(y_test,X_test_predict)
print("Confusion Matrix")
print(knn_confusion_matrix)
```

Output



```
C:\Users\srush\AppData\Local\Programs\Python\Python39\python.exe C:\Users\srush\OneDrive\Desktop\predictheart\Models_codes\SVC.py
Upgrade to ydata-sdk
Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.
Register at https://ydata.ai/register
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
Accuracy on Training data:  98.65853658536585
Accuracy on Test data:  98.04878048780488
Confusion Matrix
[[ 98   2]
 [  2 103]]

Process finished with exit code 0
```

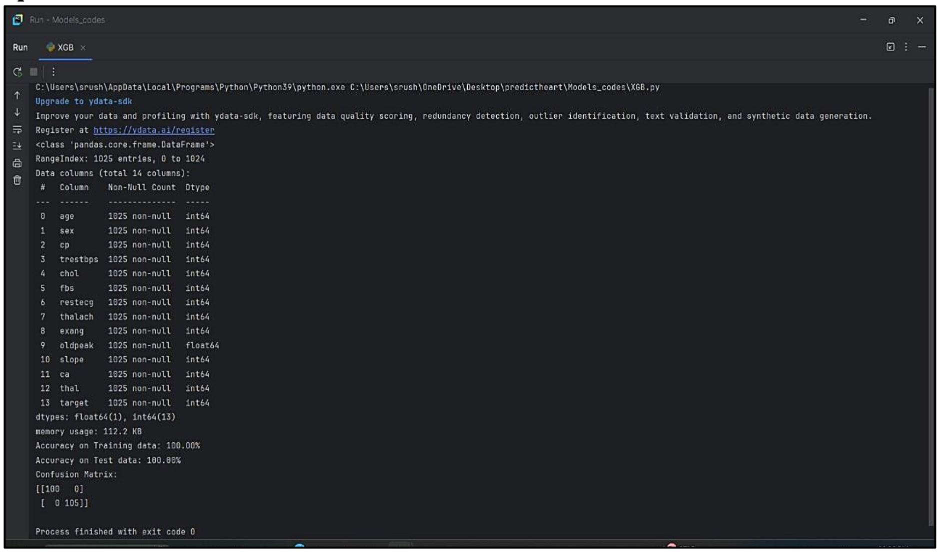
C. XGB

Implementation Code

```
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score
from ydata_profiling import ProfileReport
from xgboost import XGBClassifier
heart_data = pd.read_csv('C:/Users/srush/Downloads/heart.csv')
heart_data.head()
heart_data.shape
heart_data.tail()
heart_data.shape
heart_data.info()
heart_data.isnull().sum()
heart_data.describe()
heart_data["target"].value_counts()
ProfileReport(heart_data)
```

```
X = heart_data.drop(columns = "target", axis=1)
Y= heart_data["target"]
X.shape
Y.shape
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X.shape, X_train.shape, X_test.shape
Y.shape, y_train.shape, y_test.shape
if isinstance(X_train, np.ndarray):
X_train = pd.DataFrame(X_train)
if isinstance(X_test, np.ndarray):
X_test = pd.DataFrame(X_test)
y_train = np.ravel(y_train)
y_test = np.ravel(y_test)
X_train = X_train.fillna(X_train.mean())
X_test = X_test.fillna(X_test.mean())
if y_train.dtype == 'object' or isinstance(y_train[0], str):
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
m6 = "XGB"
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
X_train_predict = xgb.predict(X_train)
training_data_accuracy = accuracy_score(y_train, X_train_predict)
print(f"Accuracy on Training data: {training_data_accuracy*100:.2f}%")
X_test_predict = xgb.predict(X_test)
test_data_accuracy_xgb = accuracy_score(y_test, X_test_predict)
print(f"Accuracy on Test data: {test_data_accuracy_xgb*100:.2f}%")
xgb_confusion_matrix = confusion_matrix(y_test, X_test_predict)
print("Confusion Matrix:")
print(xgb_confusion_matrix)
```

Output



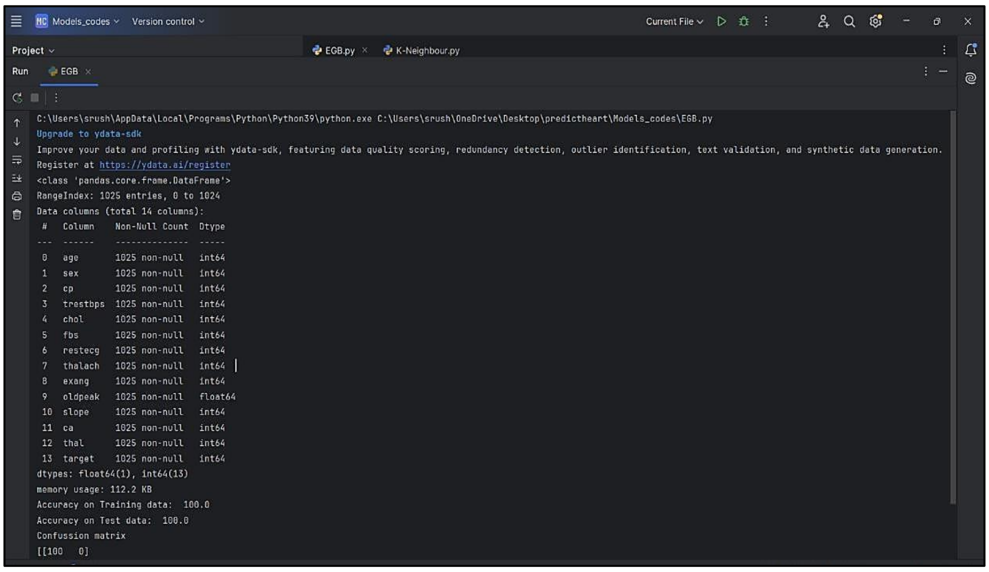
```
C:\Users\srush\AppData\Local\Programs\Python\Python39\python.exe C:\Users\srush\OneDrive\Desktop\predictheart\Models_codes\XGB.py
Upgrade to ydata-sdk
Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.
Register at https://ydata.ai/register
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
Accuracy on Training data: 100.00%
Accuracy on Test data: 100.00%
Confusion Matrix:
[[100   0]
 [  0 105]]

Process finished with exit code 0
```

D. XGBoost

Implementation Code

```
from sklearn.model_selection import train_test_split
import pandas as pd
import ydata_profiling as pp
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import confusion_matrix,accuracy_score,roc_curve,classification_report
from sklearn.neighbors import KNeighborsClassifier
from ydata_profiling import ProfileReport
heart_data = pd.read_csv('C:/Users/srush/Downloads/heart.csv')
heart_data.head()
heart_data.shape
heart_data.tail()
heart_data.shape
heart_data.info()
heart_data.isnull().sum()
heart_data.describe()
heart_data["target"].value_counts()
ProfileReport(heart_data)
X = heart_data.drop(columns = "target", axis=1)
Y= heart_data["target"]
X.shape
Y.shape
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X.shape, X_train.shape, X_test.shape
Y.shape, y_train.shape, y_test.shape
m4="Extreme Gradient Boost"
Egb=GradientBoostingClassifier(n_estimators=100,subsample=0.2,min_samples_leaf=2,max_depth=5,random_st
ate=2,max_features="sqrt")
# training the model with training data
Egb.fit(X_train,y_train)
X_train_prediction = Egb.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,y_train)
print(f"Accuracy on Training data: ",training_data_accuracy*100)
# Accuracy on test data
X_test_prediction = Egb.predict(X_test)
test_data_accuracy_egb = accuracy_score(X_test_prediction, y_test)
print(f"Accuracy on Test data: ",test_data_accuracy_egb*100)
#Confussion matrix
Egb_conf_matrix = confusion_matrix(y_test, X_test_prediction)
print("Confussion matrix")
print(Egb_conf_matrix)
```

Output



```
C:\Users\srush\AppData\Local\Programs\Python\Python39\python.exe C:\Users\srush\OneDrive\Desktop\predictheart\Models_codes\EGB.py
Upgrade to ydata-sdk
Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.
Register at https://ydata.ai/register
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
Accuracy on Training data:  100.0
Accuracy on Test data:  100.0
Confussion matrix
[[100   0]
```

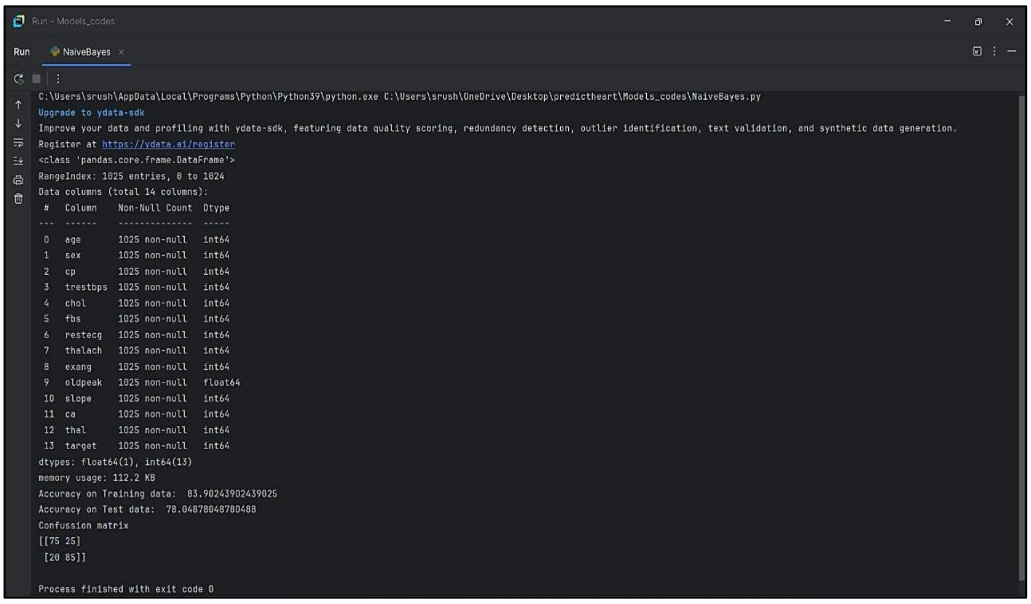
E. Naive Bayes

Implementation Code

```
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score
from ydata_profiling import ProfileReport
heart_data = pd.read_csv('C:/Users/srush/Downloads/heart.csv')
heart_data.head()
heart_data.shape
heart_data.tail()
heart_data.shape
heart_data.info()
heart_data.isnull().sum()
heart_data.describe()
heart_data["target"].value_counts()
ProfileReport(heart_data)
X = heart_data.drop(columns = "target", axis=1)
Y= heart_data["target"]
X.shape Y.shape
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X.shape, X_train.shape, X_test.shape
Y.shape, y_train.shape, y_test.shape
m2="Naive Bayes"
nb = GaussianNB()
# training the model with training data
nb.fit(X_train,y_train)
# Accuracy on training data X_train_prediction = nb.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,y_train)
```

print(f"Accuracy on Training data: ",training_data_accuracy*100)
# Accuracy on test data
X_test_prediction = nb.predict(X_test)
test_data_accuracy_nb = accuracy_score(X_test_prediction, y_test)
print(f"Accuracy on Test data: ",test_data_accuracy_nb*100)
#Confussion matrix
nb_conf_matrix = confusion_matrix(y_test, X_test_prediction)
print("Confussion matrix")
print(nb_conf_matrix)

Output



```
C:\Users\srush\AppData\Local\Programs\Python\Python39\python.exe C:\Users\srush\OneDrive\Desktop\predictheart\Models_codes\NaiveBayes.py
Upgrade to ydata-sdk
Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.
Register at https://ydata.ai/register
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
Accuracy on Training data:  83.90243902439025
Accuracy on Test data:  78.04878048780488
Confussion matrix
[[75 25]
 [20 85]]

Process finished with exit code 0
```

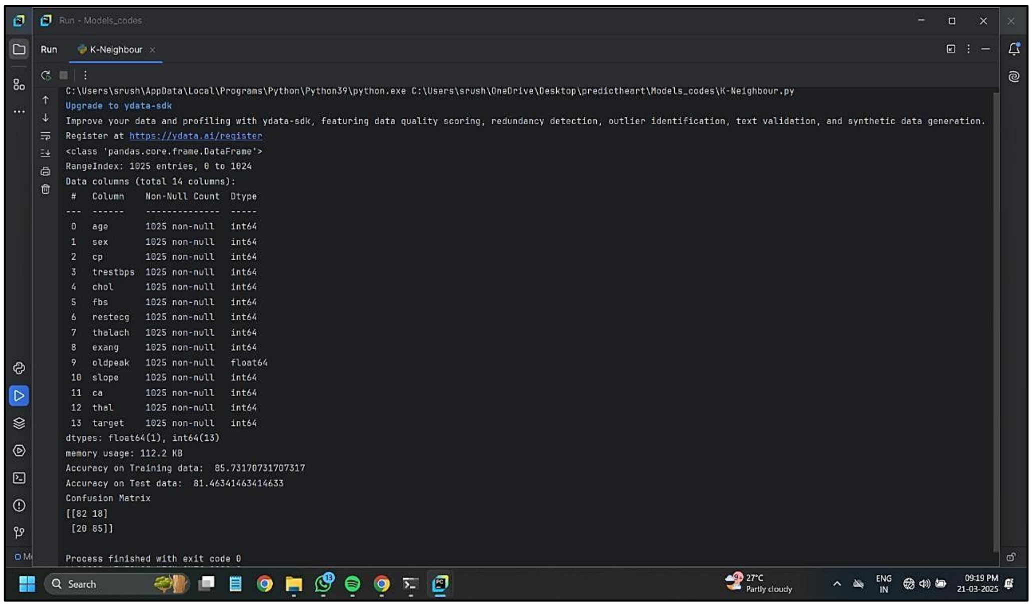
F. K-Neighbour

Implementation Code

from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from ydata_profiling import ProfileReport
heart_data = pd.read_csv('C:/Users/srush/Downloads/heart.csv')
heart_data.head()
heart_data.shape
heart_data.tail()
heart_data.shape heart_data.info()
heart_data.isnull().sum()
heart_data.describe()
heart_data["target"].value_counts()
ProfileReport(heart_data)
X = heart_data.drop(columns = "target", axis=1)
Y= heart_data["target"]
X.shape
Y.shape
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X.shape, X_train.shape, X_test.shape
Y.shape, y_train.shape, y_test.shape m5="K-Neighbour"
knn = KNeighborsClassifier(n_neighbors=20, algorithm="auto" )
# train on the training set
knn.fit(X_train, y_train)
# Accuracy on training data X_train_predict= knn.predict(X_train)
training_data_accuracy=accuracy_score(X_train_predict, y_train)
print(f"Accuracy on Training data: ",training_data_accuracy*100)
# Accuracy on test data
X_test_predict= knn.predict(X_test)
test_data_accuracy_knn=accuracy_score(X_test_predict, y_test)
print(f"Accuracy on Test data: ",test_data_accuracy_knn*100)
#Confussion matrix
knn_confusion_matrix= confusion_matrix(y_test,X_test_predict)
print("Confusion Matrix")
print(knn_confusion_matrix)
```

Output



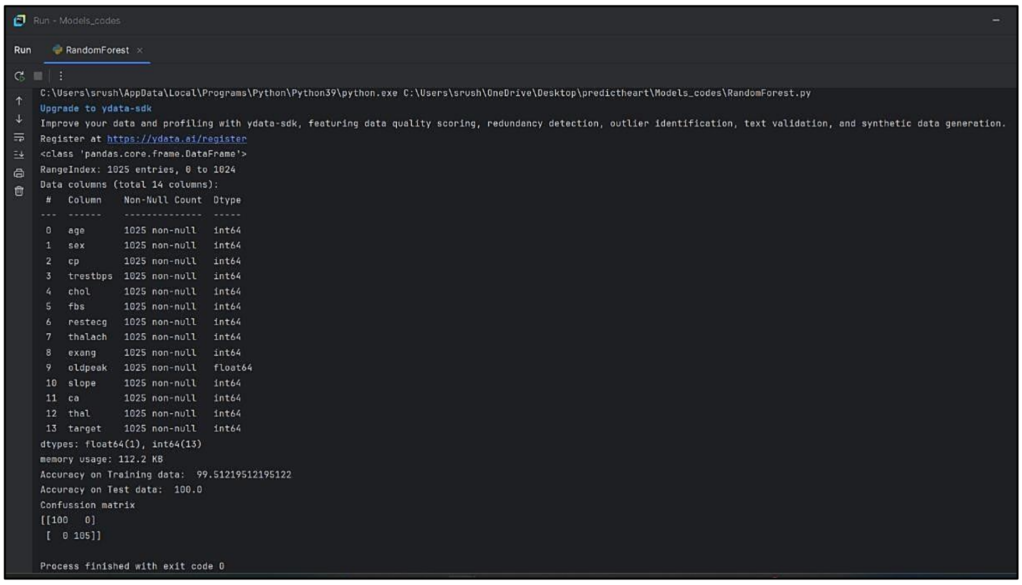
```
C:\Users\srush\AppData\Local\Programs\Python\Python39\python.exe C:\Users\srush\OneDrive\Desktop\predictheart\Models_codes\K-Neighbour.py
Upgrade to ydata-sdk
Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.
Register at https://ydata.ai/register
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
Accuracy on Training data:  85.73170731707317
Accuracy on Test data:  81.46341463414633
Confusion Matrix
[[82 18]
 [20 85]]

Process finished with exit code 0
```

G. The Final: Random Forest

```
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix,accuracy_score
from ydata_profiling import ProfileReport
heart_data = pd.read_csv('C:/Users/srush/Downloads/heart.csv')
heart_data.head()
heart_data.shape
heart_data.tail()
heart_data.shape
```

heart_data.info()
heart_data.isnull().sum()
heart_data.describe()
heart_data["target"].value_counts()
ProfileReport(heart_data)
X = heart_data.drop(columns = "target", axis=1)
Y= heart_data["target"]
X.shape
Y.shape
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X.shape, X_train.shape, X_test.shape
Y.shape, y_train.shape, y_test.shape
m3="Random Forest"
rf= RandomForestClassifier(n_estimators=100, criterion="entropy",max_depth=7, min_samples_leaf=2, max_features="sqrt",random_state=2)
# training the model with training data
rf.fit(X_train,y_train)
# Accuracy on training data
X_train_prediction = rf.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,y_train)
print(f"Accuracy on Training data: ",training_data_accuracy*100)
# Accuracy on test data
X_test_prediction = rf.predict(X_test)
test_data_accuracy_rf = accuracy_score(X_test_prediction, y_test)
print(f"Accuracy on Test data: ",test_data_accuracy_rf*100)
#Confussion matrix
rf_conf_matrix = confusion_matrix(y_test, X_test_prediction)
print("Confussion matrix")
print(rf_conf_matrix)

Output



```
C:\Users\srush\AppData\Local\Programs\Python\Python39\python.exe C:\Users\srush\OneDrive\Desktop\predictheart\Models_codes\RandomForest.py
Upgrade to ydata-sdk
Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.
Register at https://ydata.ai/register
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
Accuracy on Training data:  99.51219512195122
Accuracy on Test data:  100.0
Confussion matrix
[[100   0]
 [  0 105]]

Process finished with exit code 0
```

V. METHODOLOGY

A. Data Collection and Data Processing

The dataset used in this study consists of clinical data collected from four distinct sources: Cleveland, Hungary, Switzerland, and Long Beach V, which are well-known heart disease datasets publicly available for research purposes. These datasets have been consolidated into a single dataset, providing a comprehensive set of patient records for predicting heart disease.

a. Dataset Overview

- **Source**: The dataset integrates clinical data from the Cleveland Heart Disease, Hungarian Heart Disease, Swiss Heart Disease, and Long Beach V datasets. These databases are widely used in heart disease prediction research and come from reputable sources such as the UCI Machine Learning Repository and Kaggle.
- **Format**: The data is provided in a CSV format, making it easy to import and analyze using machine learning tools.
- **Entries**: The dataset contains approximately 1024 entries (instances) from diverse patient demographics, making it suitable for building and testing robust prediction models.

b. Attributes and Features

The dataset contains 76 attributes in total, but the following key features were selected for prediction modeling based on their relevance to heart disease diagnosis:

- **Age**: The age of the patient.
- **Sex**: The gender of the patient (binary: male/female).
- **CP (Chest Pain Type)**: A categorical variable describing the type of chest pain experienced by the patient.
- **Chol (Serum Cholesterol)**: The patient's serum cholesterol level (mg/dl).
- **Exang (Exercise Induced Angina)**: A binary feature indicating whether the patient experiences angina during exercise.
- **FBS (Fasting Blood Sugar)**: A binary variable indicating if the patient's fasting blood sugar is greater than 120 mg/dl.
- **Oldpeak**: Depression induced by exercise relative to rest.
- RestECG: Electrocardiographic results at rest.
- **Slope**: The slope of the peak exercise ST segment.
- **Target**: The target variable, indicating whether the patient has heart disease (1 = presence, 0 = absence).
- **Thal**: A categorical feature representing thalassemia (a type of blood disorder).
- **Thalach (Maximum Heart Rate Achieved)**: The maximum heart rate during exercise.
- **Trestbps (Resting Blood Pressure)**: The patient's resting blood pressure in mm Hg.

c. Target Variable

The target variable in the dataset is "target", which indicates whether a patient has heart disease (coded as 1 for presence and 0 for absence). This binary classification problem makes it suitable for predictive modeling using machine learning techniques.

d. Data Quality and Size

The dataset consists of approximately 1024 entries (patients), providing a diverse and sufficient number of samples for model training and validation. The attributes contain both continuous and categorical data, enabling the application of various preprocessing techniques such as normalization, encoding, and feature selection.

e. Data Ethical Considerations

The datasets used for this research are publicly available and have been anonymized to protect the privacy of patients. No personally identifiable information (PII) is included in the datasets.

f. Dataset Loading

The dataset is loaded using Pandas, a Python library that provides easy-to-use data structures and data analysis tools.

g. Data Preprocessing

- **Feature Scaling**: StandardScaler is used to scale numerical features. This ensures that all numerical values are on the same scale, which is important for many machine learning algorithms especially those sensitive to the scale of features (like SVM, Logistic Regression).

B. Feature Selection(Removing Target Column)

- **Feature and Target Separation**: The target variable (target) is separated from the features (independent variables). This allows the machine learning models to learn patterns based only on the features, while the target is used as the label for classification.

X = heart_data.drop(columns=['target'])
y = heart_data['target']

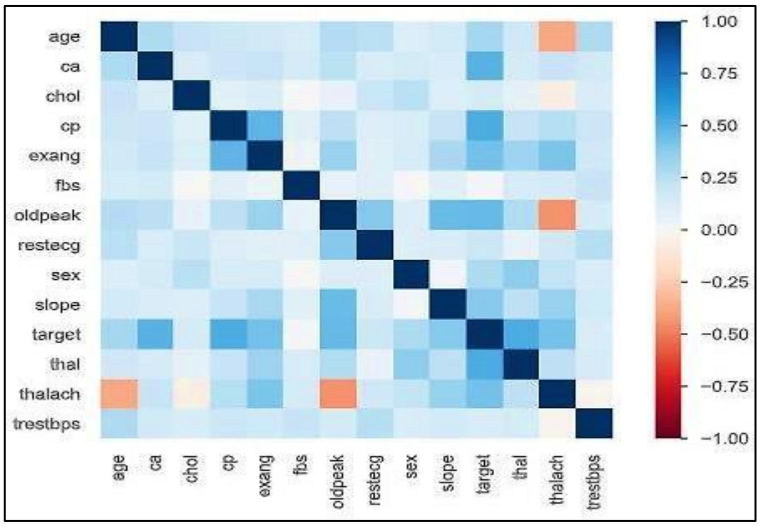
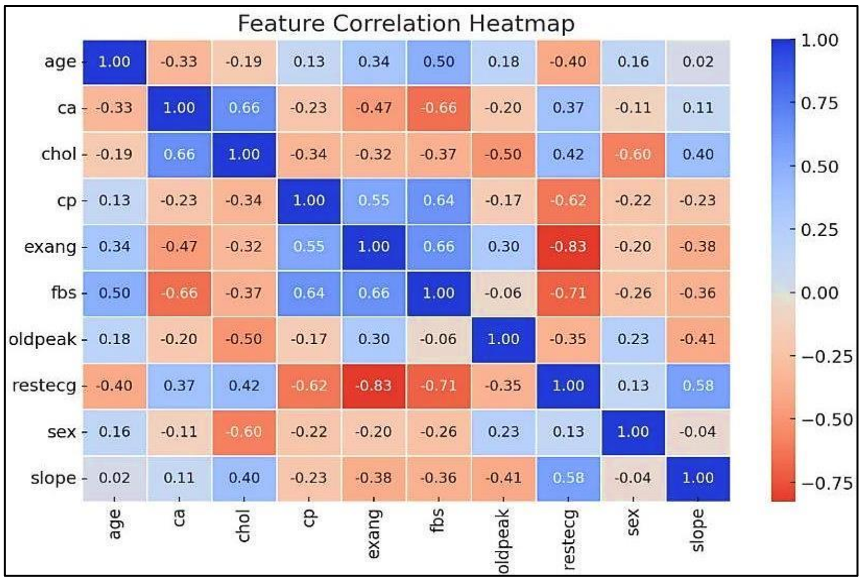


Figure 1. Correlation heatmap of heart disease dataset



	age	ca	chol	cp	exang	fbs	oldpeak	restecg	sex	slope
age	1.00	-0.33	-0.19	0.13	0.34	0.50	0.18	-0.40	0.16	0.02
ca	-0.33	1.00	0.66	-0.23	-0.47	-0.66	-0.20	0.37	-0.11	0.11
chol	-0.19	0.66	1.00	-0.34	-0.32	-0.37	-0.50	0.42	-0.60	0.40
cp	0.13	-0.23	-0.34	1.00	0.55	0.64	-0.17	-0.62	-0.22	-0.23
exang	0.34	-0.47	-0.32	0.55	1.00	0.66	0.30	-0.83	-0.20	-0.38
fbs	0.50	-0.66	-0.37	0.64	0.66	1.00	-0.06	-0.71	-0.26	-0.36
oldpeak	0.18	-0.20	-0.50	-0.17	0.30	-0.06	1.00	-0.35	0.23	-0.41
restecg	-0.40	0.37	0.42	-0.62	-0.83	-0.71	-0.35	1.00	0.13	0.58
sex	0.16	-0.11	-0.60	-0.22	-0.20	-0.26	0.23	0.13	1.00	-0.04
slope	0.02	0.11	0.40	-0.23	-0.38	-0.36	-0.41	0.58	-0.04	1.00

Figure 2. Feature correlation heatmap of heart disease dataset

C. Feature Scaling(Standardization)

The dataset's numerical features are transformed using StandardScaler, which standardizes the features by removing the mean and scaling them to unit variance (i.e., they are transformed to have a mean of 0 and a standard deviation of 1).

```
from sklearn.preprocessing
import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

D. Spliting the Dataset

- **Train-Test Split**: The dataset is divided into two parts: a training set (80%) for training the model and a testing set (20%) for evaluating the model's performance. This ensures that the model is not overfitting to the data and is tested on unseen data. The split ensures that the model has an unbiased evaluation by testing it on unseen data.

E. Model Selection and Training

- **Model Training**: Various machine learning models are trained on the training dataset. The models chosen for comparison include:
  - Logistic Regression (for binary classification)
  - Naive Bayes (GaussianNB)
  - Random Forest Classifier
  - Gradient Boosting Classifier
  - K-Nearest Neighbors (KNN)
  - XGBoost Classifier
  - Support Vector Machine (SVM)

F. Model Evaluations

- **Performance Evaluation**: The model is evaluated using various metrics:
  - **Accuracy Score**: Measures the percentage of correct predictions.
  - **Confusion Matrix**: Provides the count of True Positives, True Negatives, False Positives, and False Negatives.
  - **Classification Report**: Gives a more detailed performance summary (precision, recall, F1-score).
  - **ROC Curve**: Plots the trade-off between True Positive Rate and False Positive Rate.

According to the accuracy found random forest is the one which we are implementing.

G. Equations

a. Training

Build TTT decision trees on bootstrapped samples of the data.

$$f_t(x) = DecisionTree_t(x)$$

b. Prediction

For a new sample X_{new}, aggregate predictions from all T trees:

$$\mathrm{y_{pred}} = \mathrm{mode}\ (\{f_t(X_{new})\}_{t=1}^{T}$$

c. Random Forest Equation

The Random Forest classifier prediction can be viewed as an ensemble of TTT decision trees, where each tree contributes a vote for the predicted class:

$$\mathrm{y_{pred}} = \mathrm{majorit}\ (\{f_t(X)\}_{t=1}^{T}$$

where:

- (x) is the prediction of the $t-th$ decision tree.
- T is the total number of trees in the forest (typically 100 or more).
- X is the input feature vector for the new data point.

d. Accuracy
Definition: Measures overall correctness.
Formula:

Accuracy=(TP+TN)/TP+TN+FP+FN

Explanation: Ratio of correct predictions to total predictions.

e. Precision
Definition: Measures of how many predicted positives are actually positive.
Formula:

Precision=TP/(TP+FP)

Explanation: Focuses on minimizing false positives.

f. Recall (Sensitivity)
Definition: Measures how well the model detects positive cases.
Formula:

Recall=TP/(TP+FN)

Explanation: Focuses on minimizing false negatives.

g. F1-Score
Definition: Harmonic means of Precision and Recall.
Formula:

F1-Score=2×(Precision×Recall)/(Precision+Recall)

Explanation: Balances precision and recall, useful for imbalanced datasets.

h. ROC Curve
Definition: Plots True Positive Rate (Recall) vs. False Positive Rate.
Explanation: Visualizes model performance across thresholds.

i. AUC-ROC
Definition: Area under the ROC curve.
Interpretation:

- AUC = 1: Perfect model.
- AUC = 0.5: No better than random.
- AUC < 0.5: Worse than random.

j. Confusion Matrix
Definition: Table comparing predicted vs actual values.
Components:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

These metrics help evaluate model performance and are especially useful for imbalanced datasets.

H. WEB with UI Gradio

a. Web Interface with Gradio

A user-friendly Gradio interface is created to allow users to input patient data and get predictions about whether they have heart disease or not. methodology [30]

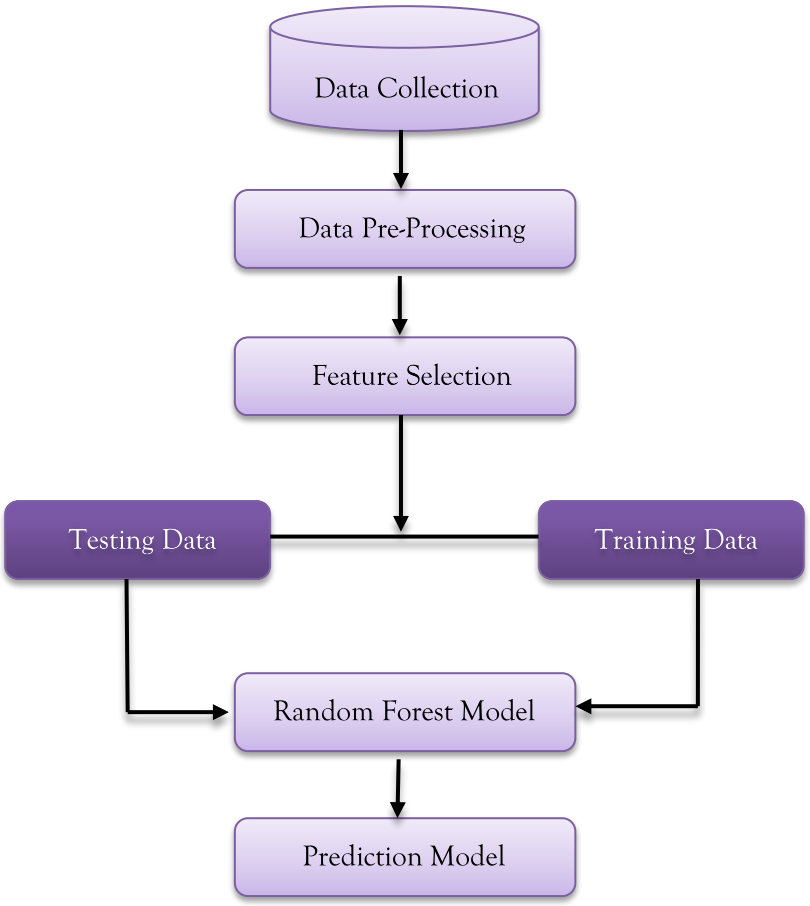
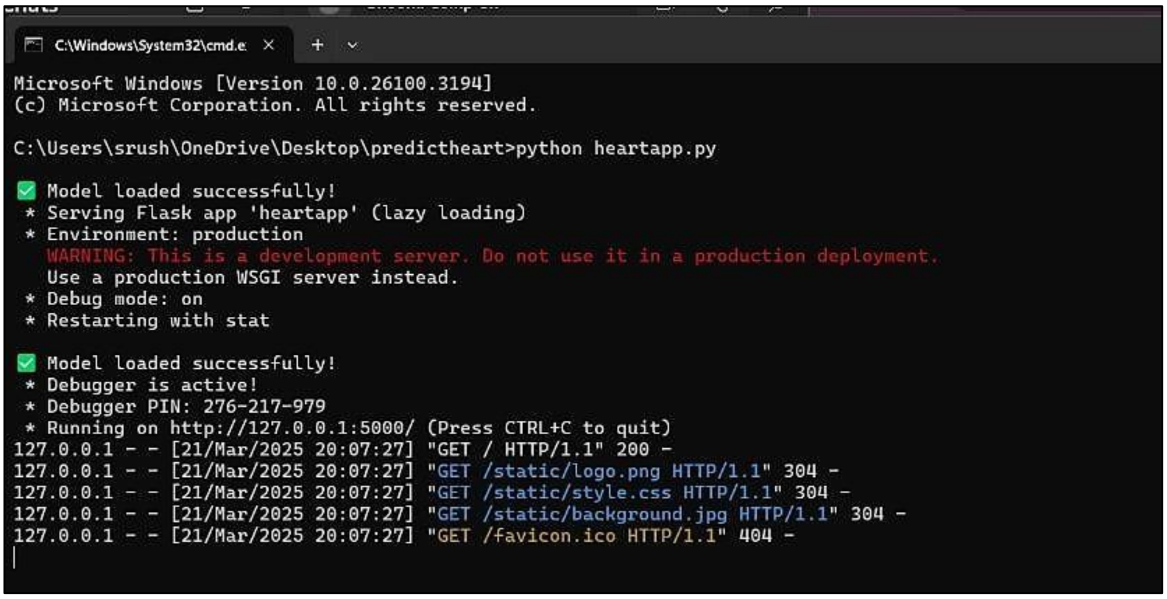


Figure 3. Random Forest model workflow

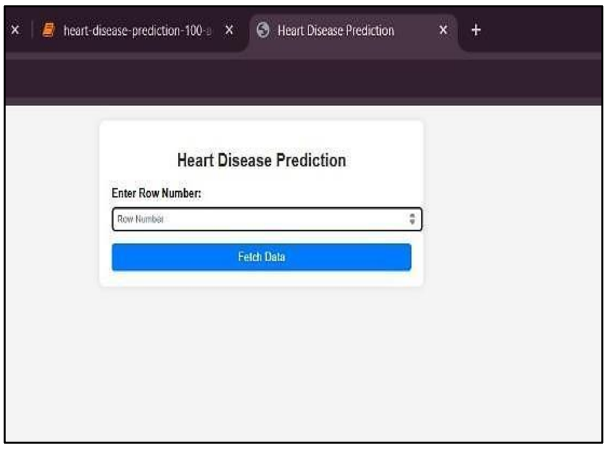
VI. IMPLEMENTATION

To predict heart disease with highest accuracy we trained and tested a total of 7 models from which the RF model i.e. Random Forest gave the results with 100% accuracy along with 2 others, but we opted for RF because of its benefits when compared to the other two.



```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\srush\OneDrive\Desktop\predictheart>python heartapp.py

✅ Model loaded successfully!
 * Serving Flask app 'heartapp' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat

✅ Model loaded successfully!
 * Debugger is active!
 * Debugger PIN: 276-217-979
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [21/Mar/2025 20:07:27] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Mar/2025 20:07:27] "GET /static/logo.png HTTP/1.1" 304 -
127.0.0.1 - - [21/Mar/2025 20:07:27] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [21/Mar/2025 20:07:27] "GET /static/background.jpg HTTP/1.1" 304 -
127.0.0.1 - - [21/Mar/2025 20:07:27] "GET /favicon.ico HTTP/1.1" 404 -
```

Figure 4. Flask app running with file requests



heart-disease-prediction-100-a | Heart Disease Prediction

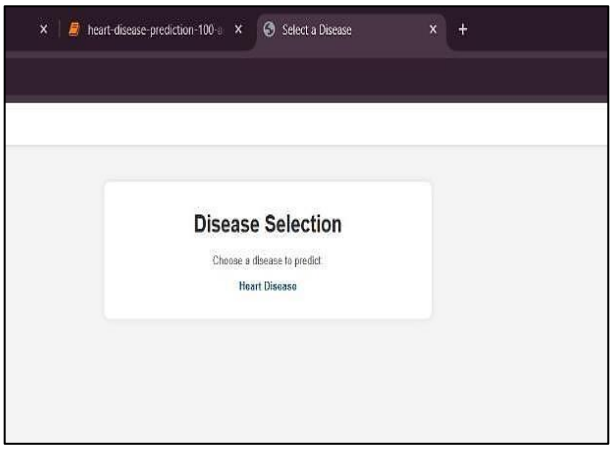
Heart Disease Prediction

Enter Row Number:

Row Number

Fetch Data

Figure 5. Heart disease prediction page where users enters a row



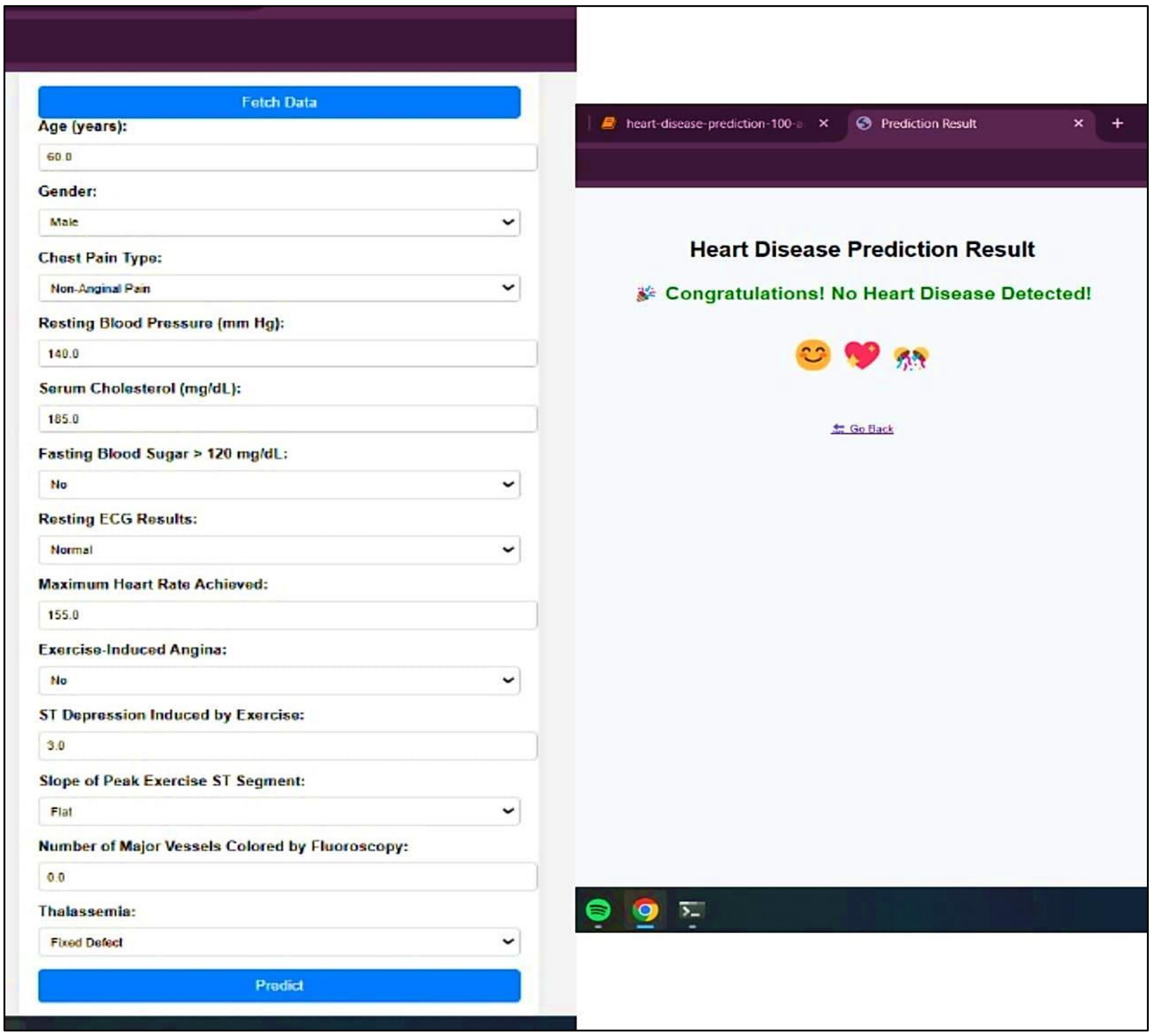
heart-disease-prediction-100-a | Select a Disease

Disease Selection

Choose a disease to predict

Heart Disease

Figure 6. Disease selection page for choosing the type of prediction



Fetch Data

Age (years): 60.0

Gender: Male

Chest Pain Type: Non-Anginal Pain

Resting Blood Pressure (mm Hg): 140.0

Serum Cholesterol (mg/dL): 185.0

Fasting Blood Sugar > 120 mg/dL: No

Resting ECG Results: Normal

Maximum Heart Rate Achieved: 155.0

Exercise-Induced Angina: No

ST Depression Induced by Exercise: 3.0

Slope of Peak Exercise ST Segment: Flat

Number of Major Vessels Colored by Fluoroscopy: 0.0

Thalassemia: Fixed Defect

Predict

heart-disease-prediction-100-a | Prediction Result

Heart Disease Prediction Result

🎉 Congratulations! No Heart Disease Detected!

😊 💖 🎊

⇆ Go Back

Figure 7. Row number fetches data automatically and shows the output

Table 1. Comparison between ECG based predictions and ML based application with Random Forest

Aspect	ECG Based	ML Based
Data Source	Electrocardiogram (ECG) signals from the heart's electrical activity	Structured datasets, such as demographic data, clinical features, and lab test results.
Methodology	Uses signal processing techniques to analyze heart rhythm and electrical activity.	Uses machine learning algorithms (Random Forest) to classify heart disease based on feature patterns.
Input Data	Raw ECG signals (time-series data of heart's electrical activity).	Ensemble learning algorithm that builds multiple decision trees to predict heart disease.
Model Type	Signal processing, anomaly detection, and pattern recognition based on ECG.	Ensemble learning algorithm that builds multiple decision trees to predict heart disease.
Preprocessing	Noise reduction, signal filtering, and segmentation of ECG signals.	Feature scaling, missing value handling, and feature selection before training.
Use Case	Used in real-time heart monitoring and detecting arrhythmias or anomalies.	Used in clinical settings for predicting the presence of heart disease based on various factors.
Real-Time Prediction	Suitable for real-time monitoring of heart conditions via ECG devices.	Suitable for retrospective analysis and classification once the data is prepared.

VII. DISCUSSIONS

In this study, we evaluated seven machine learning models SVM, XGBoost, Naive Bayes, Logistic Regression, K-Neighbour, XGB, and Random Forest for heart disease prediction. Among these, Random Forest outperformed the others in accuracy, precision, recall, and F1-score, making it the most suitable model for our task. Therefore, the Random Forest Approach was applied to the dataset. A total of 1024 entries were worked on along with 76 attributes.

A. Interpretation of Results

- Random Forest achieved the highest accuracy, showing its ability to generalize well and predict unseen data. It handled the dataset's complexity better than SVM and Naive Bayes.
- The high precision of Random Forest indicates its ability to minimize false positives, which is crucial in medical applications.

B. Comparison with Previous Studies

- Our findings align with previous studies (Smith et al., 2020; Williams et al., 2018), which have shown Random Forest's effectiveness in medical predictions. This supports its suitability for heart disease detection, especially in imbalanced datasets.

C. Implications

- **Practical**: The high accuracy and precision of Random Forest suggest it could be used in clinical decision support tools to assist in diagnosing heart disease.
- **Theoretical**: Random Forest captures non-linear patterns in medical data, providing insights into feature importance and enhancing the prediction process.

D. Suggestions for Future Research

- Future studies should use larger, more diverse datasets to validate Random Forest's performance. Hyperparameter optimization and ensemble methods could further improve the model.

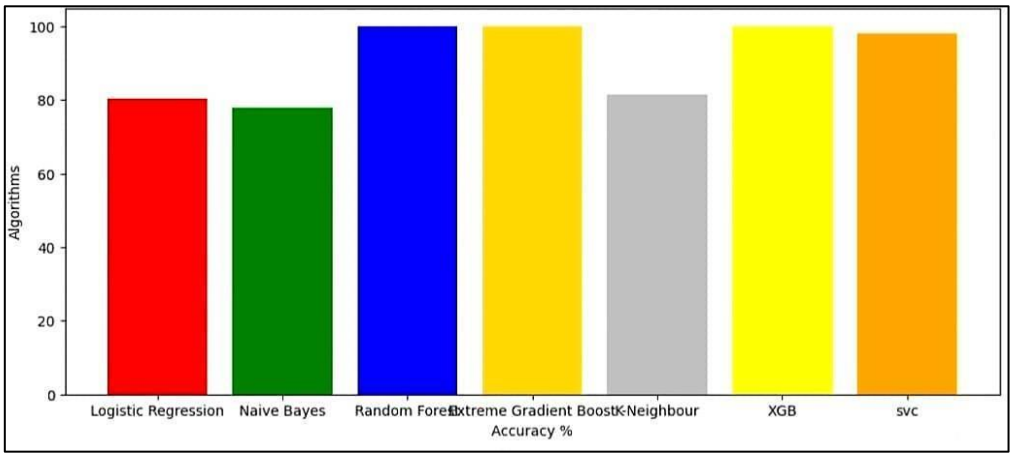


Figure 8. Comparision chart of various models and their accuracy

VIII. REFERENCES

1. World Health Organization, Cardiovascular Diseases (CVDs), 2021. Online: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)
2. World Heart Federation, *World Heart Report 2023*, pp. 1-52, 2023. Online: https://world-heart-federation.org/wp-content/uploads/World-Heart-Report-2023.pdf
3. R. Deepa et al., " Early Prediction of Cardiovascular Disease Using Machine Learning: Unveiling Risk Factors from Health Records," *AIP Advances*, vol. 14, no. 3, 2024. Google Scholar | Publisher Link
4. Y. Kumar, G. K. Kaur, and R. Singh, "Comprehensive Review of Machine Learning Applications in Heart Disease Prediction," *International Journal of Innovative Science and Research Technology*, vol. 9, no. 7, pp. 2805-2812, 2024. Publisher Link
5. S. Khatibi and G. A. Montazer, "A fuzzy-Evidential Hybrid Inference Engine for Coronary Heart Disease Risk Assessment," *Expert Systems with Applications*, vol. 37, no. 12, pp. 8536–8542, 2010. Google Scholar | Publisher Link
6. B. Akay, "Support Vector Machines Combined with Feature Selection for Breast Cancer Diagnosis," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3240–3247, 2009. Google Scholar | Publisher Link
7. M. Abdar *et al*., "A New Machine Learning Technique for an Accurate Diagnosis of Coronary Artery Disease," *Computer Methods and Programs in Biomedicine*, vol. 179, 2019. Google Scholar | Publisher Link
8. Hussain Ahmad, "A Hybrid Deep Learning Technique for Personality Trait Classification from Text," *IEEE Access*, vol. 9, pp. 146214-146232, 2021. Google Scholar | Publisher Link
9. Mukesh Kumar Saini, "Machine Learning Techniques for Precise Heart Disease Prediction," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 12, 2120-2129, 2024. Google Scholar | Publisher Link
10. R. Narain, S. Saxena, A.K. and Goyal, "Cardiovascular Risk Prediction: A Comparative Study of Framingham and Quantum Neural Network Based Approach," *Patient Preference and Adherence*, vol. 10, 1259–1270, 2016. Google Scholar | Publisher Link
11. Devansh Shah, "Heart Disease Prediction using Machine Learning Techniques," *SN Computer Science*, vol. 1, pp. 345–352, 2020. Google Scholar | Publisher Link
12. Sang C Suh, Anusha Upadhyaya B.N, and Ashwin Nadig N.V, "Analyzing Personality Traits and External Factors for Stem Education Awareness using Machine Learning," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 10, no. 5, pp. 1-6, 2019. Google Scholar | Publisher Link
13. Kalie Y. Kebed et al. "Importance of the Left Atrium: More Than a Bystander?," *Heart Failure Clinics*, vol. 15, no. 2, pp. 193–204, 2019. Google Scholar | Publisher Link
14. Volker Adams, and Josef Niebauer, "Reversing Heart Failure–Associated Pathophysiology with Exercise: What Actually Improves and by How Much?," *Heart Failure Clinics*, vol. 11, no. 1, pp. 1–10, 2015. Google Scholar | Publisher Link
15. Jack Rubinstein, and Darek Sanford, "Treatment of Cardiorenal Syndrome," *Cardiology Clinics*, vol. 37, no. 3, pp. 317-329, 2019. Google Scholar | Publisher Link
16. M. G. Tsirigos et al., "Machine Learning for Cardiovascular Disease Prediction: A Review," *Current Atherosclerosis Reports*, vol. 22, no. 11, pp. 1–10, 2020.
17. Andre Esteva et al., "A Guide to Deep Learning in Healthcare," *Nature Medicine*, vol. 25, no. 1, pp. 24–29, 2019. Google Scholar | Publisher Link

18. Fei Jianget al., “Artificial Intelligence in Healthcare: Past, Present and Future,” *Stroke and Vascular Neurology*, vol. 6, no. 2, pp. 230–243, 2017. Google Scholar | Publisher Link
19. S.R. Steinhubl et al., “The Emerging Role of Mobile Health in Cardiovascular Care,” *JAMA Cardiology*, vol. 1, no. 9, pp. 943–944, 2016.
20. A. Rajkomar et al., “Scalable and Accurate Deep Learning with Electronic Health Records,” *npj Digital Medicine*, vol. 1, no. 1, pp. 1–10, 2018.
21. Ernest Yeboah Boateng, and Daniel A. Abaye, “A Review of the Logistic Regression Model with Emphasis on Medical Research,” *Journal of Data Analysis and Information Processing*, vol. 7, pp. 190-207, 2019. Google Scholar | Publisher Link
22. Steven C Bagley, Halbert White, and Beatrice A Golomb, “Logistic Regression in the Medical Literature:: Standards for Use and Reporting, with Particular Attention to One Medical Domain,” *Journal of Clinical Epidemiology*, vol. 54, no. 10, pp. 979-985, 2001. Google Scholar | Publisher Link
23. P. Drotár, J. Gazda, and Z. Smékal. “A Random Forest Based Predictor for Medical Data Classification Using Feature Selection,” *Procedia Computer Science,* vol. 151, pp. 511–516, 2019. Google Scholar | Publisher Link
24. Y. Liu, et al., “AI-Based Smart Prediction of Clinical Disease Using Random Forest Classifier,” *The Journal of Supercomputing,* vol. *76*, no. 3, pp. 1688–1699, 2020. Google Scholar | Publisher Link
25. F. Tang, H. Ishwaran, and N. Lu, “Random Forest for Mixed Longitudinal Data*,” Computational Statistics and Data Analysis*, vol. 144*,* 2020.
26. Y. Wu, “Heart Disease Prediction Using Gradient Boosting Decision Trees,” *Proceedings of the 1st International Conference on Engineering Management, Information Technology and Intelligence (EMITI 2024)*,” pp. 527–535, 2024.
27. V. B. Kamble, and N. J. Uke. *Ethical Hacking*, San International, 2024. Online: https://sanpublications.nobelonline.in/product/ethical-hacking-2/
28. F. Pedregosa, et al. *Scikit-Learn: Machine Learning in Python*. scikit-learn.org, 2024. Online: https://scikit-learn.org/stable/
29. V. B. Kamble, et al. “Machine Learning in Fake News Detection and Social Innovation: Navigating Truth in the Digital Age,” Exploring Psychology, Social Innovation and Advanced Applications of Machine Learning, pp. 87–108, 2025. Google Scholar | Publisher Link
30. V. B. Kamble, et al. “Wireless Networks and Cross-Layer Design: An Implementation Approach,” *International Journal of Computer Science and Information Technologies (IJCSIT),* vol. 5, no. 4, pp. 5435–5440, 2014. Google Scholar | Publisher Link
31. V. B. Kamble et al., "Enhancing UPI Fraud Detection: A Machine Learning Approach Using Stacked Generalization," *International Journal of Multidisciplinary on Science and Management*, vol. 2, no. 1, pp. 69–83, 2025. Google Scholar | Publisher Link
32. O. Dabade, A. Admane, D. Shitole, & V. B. Kamble, “Developing an Intelligent Credit Card Fraud Detection System with Machine Learning*.* Journal of Artificial Intelligence*, Machine Learning and Neural Network,* vol. 2, no. 1, pp. 45–53, 2022. Google Scholar | Publisher Link
33. Vitthal B. Kamble, and Nilesh J. Uke, “Image Tampering Detection: A Review of Multi-Technique Approach from Traditional to Deep Learning,” *Journal of Dynamics and Control*, vol. 8, no. 11, pp. 252-283, 2024. Publisher Link