A project on

Heart Disease Prediction System

Submitted by

E. Likhith (AP22110010386)

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Under the guidance of

Mr. PAMULA UDAY RAJU

Assistant Professor – Ad Hoc

Department of CSE

[November, 2024]

SRM University AP, Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

Department of Computer Science and Engineering

SRM University, AP

CERTIFICATE

*This is to certify that the Project report entitled "**Heart Disease Prediction System**" is being submitted by **E. Likhith (AP22110010386)** student of Department of Computer Science and Engineering, SRM University, AP, in partial fulfilment of the requirement for Machine Learning Lab for IIIrd year B. Tech (CSE), Semester V. carried out by him during the academic year 2024-2025.*

Signature of the Lab In Charge

Mr. Pamula Uday Raju

Assistant Professor-Ad Hoc

Department of CSE

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to thank my Professor, Pamula Udaya Raju , for giving us the opportunity to work on this project. The development of the Heart Disease Prediction would not have been possible without the collaborative efforts and commitment of the development team. We extend our sincere gratitude to all team members who contributed their time and expertise to bring this project to completion. This project taught a lot about different core concepts of Machine Learning .I am extremely grateful and express my profound gratitude and indebtedness to my project guide teacher for the kind help and for giving me the necessary guidance and valuable suggestions in completing this project.

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Heart is the next major organ comparing to brain which has more priority in Human Body. It pumps the blood and supplies to all organs of the whole body. Day by Day the cases of heart disease are increasing at a rapid rate & it's very important and concerning to predict any such diseases beforehand. According to World Health Organization (WHO), cardiovascular diseases (CVD) are the major health concern worldwide and a leading cause of death. CVDs were responsible for 32% of all global deaths in 2019, as estimated by World Health Organization. Heart attacks and strokes were responsible for 85% of these deaths. Some of the machine learning are used to predict the heart disease, such as Logistic Regression, Support Vector Machine (SVM), Decision Tree, Random Forest, K- Nearest Neighbor(KNN) for Prediction. The given heart disease prediction system enhances medical care and reduces the cost. This project provides an insight of the existing algorithm, and it gives us significant knowledge that can help us predict the patients with heart disease.

1. INTRODUCTION

Cardiovascular diseases (CVDs) are the leading cause of death in the world. Each year, around 17.9 million people die from cardiovascular diseases, accounting for 32% of all deaths worldwide, according to the World Health Organization (WHO). CVDs are illnesses that affect the heart and blood vessels, including heart attacks and strokes. Smoking, unhealthy diet and lack of exercise increase your risk of heart disease. Prevention is possible through healthy lifestyle choices, regular check-ups and by predicting the risk before the attack.

Machine Learning (ML) is a very vast and diverse field, and its scope and implementation are increasing day by day. It is a method for extracting and analyzing implicit and explicit data. Machine Learning plays a vital role in predicting heart disease by processing vast health data to uncover patterns and risk factors. The prediction of this disease before being infected is part of the prevention methods. These Machine learning techniques use different types of classifiers, including Supervised, Unsupervised, and Reinforcement Learning, to make predictions more precisely and measure the accuracy of a dataset.

This project focuses on the following Algorithms like Logistic regression, SVM, Decision tree, Random Forest and KNN. The algorithm that achieves the highest accuracy among 3 to 4 models will be considered the most effective method for prediction. The Objective of this project is to propose the most efficient algorithm within the existing for prediction of Cardiovascular Disease(CVD) patient based on their medical attributes. Ultimately, by establishing the most accurate predictive algorithm, we aim to enhance patient outcomes and contribute to effective prevention strategies for cardiovascular diseases.

Figure 1 depicts the parts of human heart such as Left atrium, Tricuspid valve, Aortic valve, Mitral valve, Superior vena Right atrium, Right ventricle, Left ventricle, Aorta, cava and Interior vena cava, Pulmonary vein, Pulmonary valve, Pulmonary artery.

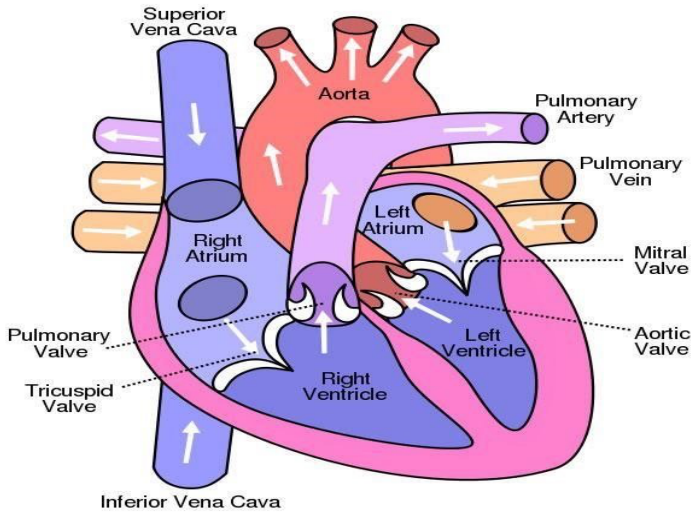


Figure 1: Human Heart

1.1 Brief overview about the Project

This project leverages machine learning to predict the likelihood of heart disease, employing algorithms like Logistic Regression, SVM, Decision Tree, Random Forest, and KNN. Given the heart's vital function in supplying blood throughout the body, early prediction is essential due to the increasing prevalence of heart disease. According to WHO, cardiovascular diseases are a leading global health concern. The goal of this project is to enhance medical care and reduce costs by providing a reliable system for early detection, ultimately contributing to better diagnosis and preventive healthcare.

1.2 Software Requirements

A. **Operating System**:Windows/Linux (with Google Colab Environment)
B. **Software:** Python (with libraries like Pandas, Scikit-Learn, Matplotlib, etc.)
C. **RAM:** 4 GB (minimum recommended)
D. **Data Structure:** Data Frames, Arrays, Trees (for Decision Trees), and Matrices (for SVM, KNN)

2. LITERATURE SURVEY

Other researchers works have been evaluated and it is appreciated that the methodology to predict heart disease from the particulars of the patient provided should be the most efficient. In many works, different cutting-edge technologies have been employed which resulted in varying degrees of success; some achieved high-accuracy results while other methods had shortcomings in their predictive capabilities. Here the strengths and weaknesses of the various methods employed for the prediction of the heart disease are discussed.

Avinash Golande [1] used Machine Learning Techniques to predict Heart Attack effectively. Naive Bayes, decision trees, k-Nearest Neighbours are prevalent machine learning techniques used for classification tasks. However, the precision of each technique may differ based on specific problem and dataset being analysed. In some cases, decision trees may achieve higher accuracy than other algorithms.

Santhana Krishnan [2] used decision trees and Nave Bayes machine learning algorithms, to predict heart attacks. The Decision Tree Model achieved a precision rate of 91% and Nave Bayes' accuracy level was 87%. They presumed that the best algorithm was a decision tree for handling data sets.

Aditi Gavhane et al. [3] worked on machine learning applications by using certain parameters like lifespan, gender, heart rate, etc. to foresee the vulnerability of heart problem. To train and test the dataset, they used neural network supervised algorithms, that is MLP (multilayer perceptron), which gives reliable outcomes from the user's input. Machine learning algorithm using neural network is the most accurate and reliable algorithm.

Rati Goel [4] worked on SVM, LR, RF, Decision tree, Naive Bayes and KNN algorithms of ML and compare their efficiency based on basic parameters like chest pain (CP), gender, blood pressure (BP), and cholesterol to predict heart disease. He concludes that the SVM has better performance in comparison to other machine learning algorithms. Manjula P et al. [5] used various ML techniques like SVM, Decision Tree, Random Forest, Navie Bayes and KNN in their model with Vulnerability factors like age, gender, arterial pressure, cholesterol levels, and family background of heart problem to train and test their models of machine learning. The random forest algorithm achieved exceptional precision in their model.

Pavan Kumar Tadiparthi et al. [6] reviewed several types of machine learning algorithms for predicting heart issues. They determined that the precision of the machine learning algorithms for disease prediction can be enhanced with proper feature selection and ensemble methods. Python environment was used for

experiment. Logistic Regression gives 81.9% accuracy and better performance given by the classification model on the data set, which uses 14 features.

Karthick K. et al. used SVM, Gaussian Naive Bayes (GNB), LR, LightGBM, XGBoost, and RF algorithms to build an ML model for heart disease risk prediction. In this study, the authors applied the Chi-square statistical test to select the best features from the Cleveland heart disease dataset. After feature selection, the RF classifier model obtained the highest classification accuracy rate of 88.5% [7].

Sairabi H.Mujawar et al, [8] used k-means and naïve bayes to predict heart disease. This paper is to build the system using historical heart database that gives diagnosis. 13 attributes have considered for building the system. To extract knowledge from database, data mining techniques such as clustering, classification methods can be used. 13 attributes with total of 300 records were used from the Cleveland Heart Database. This model is to predict whether the patient have heart disease or not based on the values of 13 attributes.

Sharan Monica.L et al [9] proposed an analysis of cardiovascular disease. This paper proposed data mining techniques to predict the disease. It is intended to provide the survey of current techniques to extract information from dataset and it will be useful for healthcare practitioners. The performance can be obtained based on the time taken to build the decision tree for the system. The primary objective is to predict the disease with less number of attributes.

Reldean Williams et al. [10] used a set of eight machine learning methods, including Artificial Neural Networks, RF, LR, SVM , Decision Trees, XG Boost and Naive Bayes for heart problem prediction with elements like Systolic and diastolic pressure, cholesterol level and Chest tightness. UCI data repository is used. They found that out of the machine learning methods utilized, Random Forest demonstrated the highest accuracy for forecasting the incidence of the illness.

Dubey A. K. et al. [11] examined the performance of ML models such as Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), SVM with grid search (SVMG), K-Nearest Neighbor (KNN) and Naïve Bayes (NB) for heart disease classification. Cleveland and Statlog datasets from the UCI Machine Learning repository were used for training and testing. The experimental results show that LR and SVM classifier models perform better on the Cleveland dataset with 89% accuracy, while LR performs better on the Statlog dataset with 93% accuracy.

Veisi H. et al. developed various ML models such as DT, RF, SVM, XGBoost, and Multilayer Perceptron (MLP) using the Cleveland heart disease dataset to predict heart disease. Various preprocessing (outlier detection, normalization, etc.) and feature selection processes were applied to the dataset. Among the ML models evaluated, the highest accuracy of 94.6% was achieved using the MLP [12].

Table 1: Survey of Heart Disease Prediction Models

Reference	Year	Model	Accuracy/Results	Drawbacks
Golande, A. [1]	2018	Naive Bayes, Decision Tree, k-NN	Effective in heart attack prediction	Limited accuracy on small or noisy datasets
Krishnan, S. J. [2]	2020	Decision Tree, Naive Bayes	Decision Tree: 91% precision, Naive Bayes: 87% accuracy	Naive Bayes assumes feature independence, which may reduce accuracy.
Gavhane, A., et al. [3]	2019	MLP Neural Network	Reliable outcomes for heart problem prediction	High computational cost and prone to overfitting.
Goel, R. [4]	2019	SVM, LR, RF, Decision Tree, Naive Bayes, KNN	SVM outperforms other algorithms	SVM is sensitive to kernel & require high memory
Manjula, P., et al. [5]	2021	SVM, Decision Tree, RF, Naive Bayes, KNN	RF achieved Exceptional precision	Random Forest can be slow and complex for large datasets.

Tadiparthi, P. K., et al. [6]	2021	Logistic Regression	LR: 81.9% accuracy	Limited flexibility for complex data patterns.
Karthick. K, et al.[7]	2022	Price	Quantity	Lack of clarity; needs better model specification.
Mujawar, S. H., et al. [8]	2021	k-means, Naive Bayes	Predicts heart disease based on 13 attributes	K-means requires careful tuning of k and can be sensitive to outliers.
Monica, S. L., et al. [9]	2021	Decision Tree	Focus on prediction with fewer attributes	Decision Trees can easily overfit on small datasets.
Williams, R., et al. [10]	2020	ANN, RF, LR, SVM, Decision Tree, XGBoost, Naive Bayes	RF demonstrated highest accuracy	ANN and RF require high computational power and may overfit.
Dubey, A. K., et al. [11]	2020	LR, Decision Tree, RF, SVM, KNN, Naive Bayes	LR: 93% on Statlog, 89% on Cleveland	Logistic Regression (LR) may struggle with non-linear relationships
Veisi, H., et al. [12]	2021	DT, RF, SVM, XGBoost, MLP	MLP: 94.6% accuracy	High risk of overfitting in complex NN

3. PROBLEM STATEMENT

Cardiovascular diseases (CVDs) are among the leading causes of death worldwide, necessitating early detection and effective prevention strategies. With the growing prevalence of heart-related illnesses, there is a pressing need for reliable and efficient prediction models to assist in timely diagnosis and treatment. Existing approaches to predicting heart disease often suffer from varying levels of accuracy, computational complexity, and limitations in handling diverse datasets.

This project aims to evaluate and compare various machine learning algorithms, such as Logistic Regression, Support Vector Machines (SVM), Decision Trees, Random Forest, and K-Nearest Neighbors (KNN), to determine the most efficient and accurate method for predicting heart disease based on medical attributes. By identifying the best-performing algorithm, this study seeks to contribute to improved diagnostic systems, enabling proactive healthcare interventions and reducing mortality rates associated with cardiovascular diseases.

4. PROPOSED MODELS

4.1 DATASET AVAILABILITY

This specific dataset, typically contains 14 key features for prediction, representing important health metrics like age, gender, blood pressure, cholesterol levels, blood sugar levels, and additional diagnostic measures. The target class is included as well, indicating whether or not heart disease is present in each patient. The target feature refers to the presence of heart disease in the subject.

0 = no disease

1 = disease

This simplification facilitates analysis by focusing only on the presence versus absence of heart disease. ***Table 2*** shows the features included in the heart disease dataset.

Table 2: Dataset Description

Order	Features	Description	Feature Value Range
1	Age	Age in years	29 to 77
2	Sex	Gender	Value 1 = male Value 0 = female
3	Cp	Chest pain type	Value 0: typical angina Value 1: atypical angina Value 2: non-anginal pain Value 3: asymptomatic
4	Trest Bps	Resting blood pressure (in mm Hg on admission to the hospital)	94 to 200
5	Chol	Serum cholesterol in mg/dL	126 to 564
6	Fbs	Fasting blood sugar > 120 mg/dL	Value 1 = true Value 0 = false

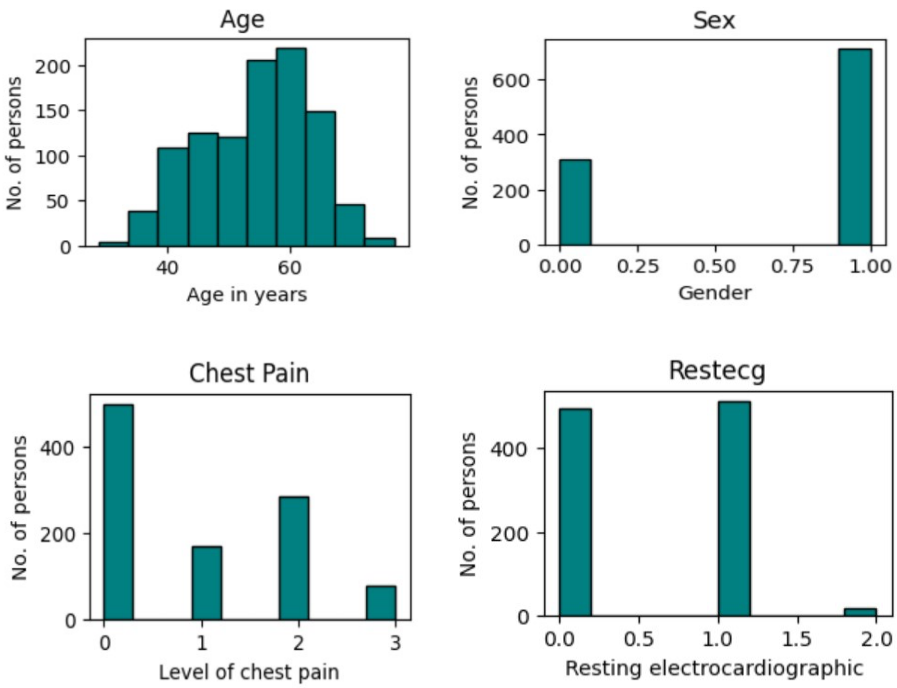
7	Restecg	Resting electrocardiographic results	Value 0: Normal Value1:having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of >0.05 mV) Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8	Thalach	Maximum heart rate achieved	71 to 202
9	Exang	Exercise-induced angina	Value 1 = yes Value 0 = no
10	Oldpeak	Stress test depression induced by exercise relative to rest	0 to 6.2
11	Slope	The slope of the peak exercise ST segment	Value 0: upsloping Value 1: flat Value 2: down sloping
12	Ca	Number of major vessels	Number of major vessels (0–3) coloured by fluoroscopy
13	Thal	Thallium heart rate	Value 0 = normal; Value 1 = fixed defect; Value 2 = reversible defect
14	Target	Diagnosis of heart disease	Value 0 = disease no Value 1 = disease

In this dataset, all samples are complete, with no missing or null values across any features as shown in ***Figure 2***. The dataset contains a total of 1025 samples. Out of these, 499 samples are classified as belonging to the disease class (1), while 526 samples are classified as belonging to the no-disease class (0) ***Figure 3.*** A **correlation matrix** is a table displaying correlation coefficients between pairs of variables, which indicate the strength and direction of their linear relationships. Each cell in the matrix shows the correlation (ranging from -1 to 1) between two variables, helping to identify which variables move together and can aid in feature selection or understanding data dependencies. ***Figure 4.***

Figure 2 : Complete List of Features (No Missing Values)

age	0	exang	0
sex	0	oldpeak	0
cp	0	slope	0
trestbps	0	ca	0
chol	0	thal	0
fbs	0	target	0
restecg	0		
thalach	0		

Figure 3 : Histogram of Elements



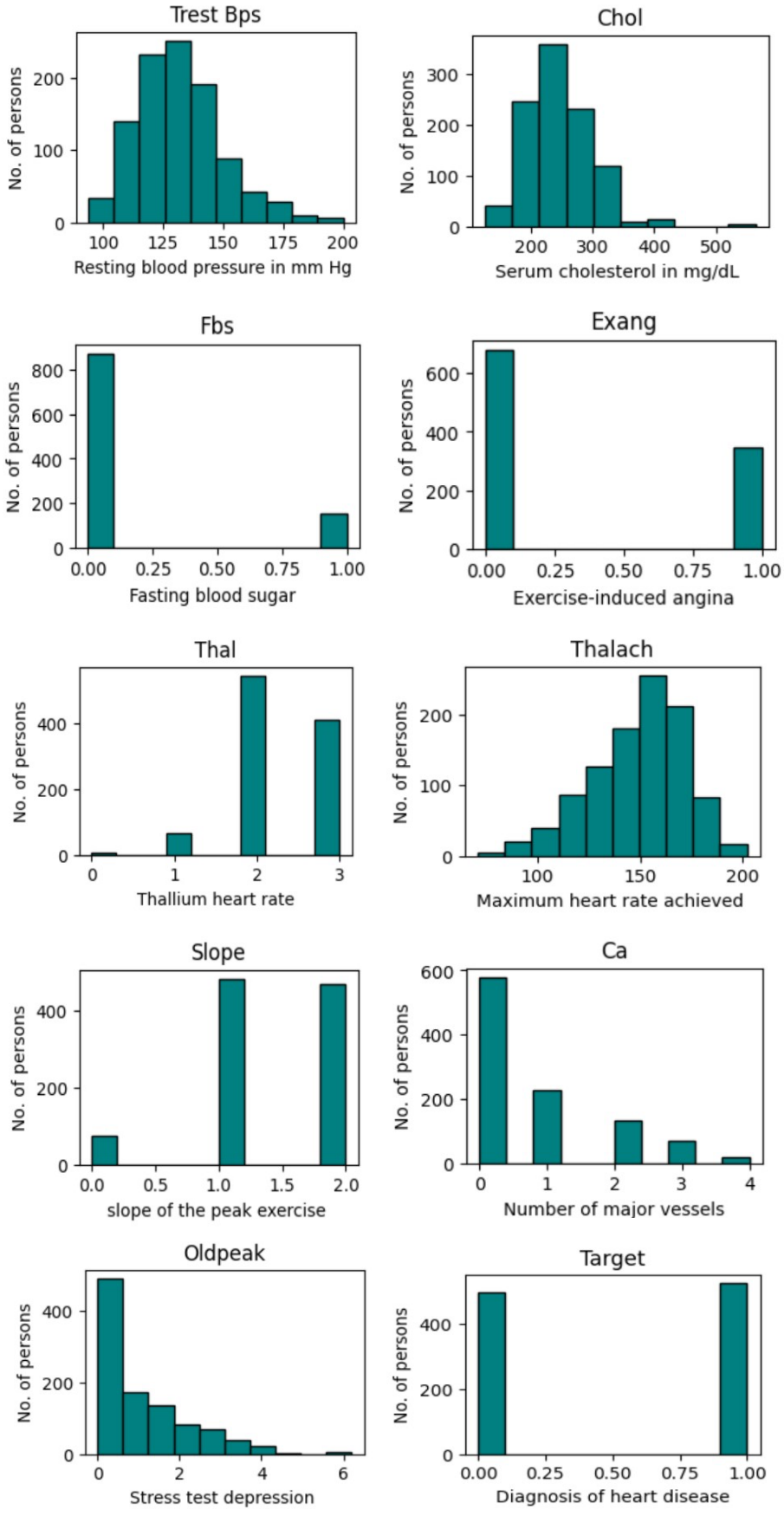

Trest Bps
No. of persons
Resting blood pressure in mm Hg
Chol
No. of persons
Serum cholesterol in mg/dL
Fbs
No. of persons
Fasting blood sugar
Exang
No. of persons
Exercise-induced angina
Thal
No. of persons
Thallium heart rate
Thalach
No. of persons
Maximum heart rate achieved
Slope
No. of persons
slope of the peak exercise
Ca
No. of persons
Number of major vessels
Oldpeak
No. of persons
Stress test depression
Target
No. of persons
Diagnosis of heart disease

Figure 4: Correlation Matrix

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.094962	-0.063107	0.283121	0.207216	0.119492	-0.111590	-0.395235	0.093216	0.206040	-0.164124	0.302261	0.065317	-0.221476
sex	-0.094962	1.000000	-0.051740	-0.057647	-0.195571	0.046022	-0.060351	-0.046439	0.143460	0.098322	-0.032990	0.113060	0.211452	-0.283609
cp	-0.063107	-0.051740	1.000000	0.046486	-0.072682	0.096018	0.041561	0.293367	-0.392937	-0.146692	0.116854	-0.195356	-0.160370	0.432080
trestbps	0.283121	-0.057647	0.046486	1.000000	0.125256	0.178125	-0.115367	-0.048023	0.068526	0.194600	-0.122873	0.099248	0.062870	-0.146269
chol	0.207216	-0.195571	-0.072682	0.125256	1.000000	0.011428	-0.147602	-0.005308	0.064099	0.050086	0.000417	0.086878	0.096810	-0.081437
fbs	0.119492	0.046022	0.096018	0.178125	0.011428	1.000000	-0.083081	-0.007169	0.024729	0.004514	-0.058654	0.144935	-0.032752	-0.026826
restecg	-0.111590	-0.060351	0.041561	-0.115367	-0.147602	-0.083081	1.000000	0.041210	-0.068807	-0.056251	0.090402	-0.083112	-0.010473	0.134874
thalach	-0.395235	-0.046439	0.293367	-0.048023	-0.005308	-0.007169	0.041210	1.000000	-0.377411	-0.342201	0.384754	-0.228311	-0.094910	0.419955
exang	0.093216	0.143460	-0.392937	0.068526	0.064099	0.024729	-0.068807	-0.377411	1.000000	0.286766	-0.256106	0.125377	0.205826	-0.435601
oldpeak	0.206040	0.098322	-0.146692	0.194600	0.050086	0.004514	-0.056251	-0.342201	0.286766	1.000000	-0.576314	0.236560	0.209090	-0.429146
slope	-0.164124	-0.032990	0.116854	-0.122873	0.000417	-0.058654	0.090402	0.384754	-0.256106	-0.576314	1.000000	-0.092236	-0.103314	0.343940
ca	0.302261	0.113060	-0.195356	0.099248	0.086878	0.144935	-0.083112	-0.228311	0.125377	0.236560	-0.092236	1.000000	0.160085	-0.408992
thal	0.065317	0.211452	-0.160370	0.062870	0.096810	-0.032752	-0.010473	-0.094910	0.205826	0.209090	-0.103314	0.160085	1.000000	-0.343101
target	-0.221476	-0.283609	0.432080	-0.146269	-0.081437	-0.026826	0.134874	0.419955	-0.435601	-0.429146	0.343940	-0.408992	-0.343101	1.000000

4.2 FEATURE ENGINEERING

FEATURE SELECTION AND FEATURE EXTRACTION

The performance of ML models depends on the quality of the features used as input. As the number of features in the datasets increases, the prediction performance of the model decreases, and the computational costs increase. By reducing the number of features, the model can obtain more accurate results and work faster and more efficiently. ML models are designed according to the data used in the learning process. Selecting the best features makes the features learned by the model more generalizable. Thus, it makes the model work better with new data. Some features in the datasets are not important to the result and increase the computational complexity of the model. Removing unnecessary features reduces noise and helps the model achieve better results. Also, feature selection is important for understanding the nature of dataset.

Feature Selection using Wrapper Method: Recursive Feature Elimination (RFE)

Feature selection is the process of choosing the most relevant features from the dataset, which reduces the dimensionality and potentially improves the performance of a model. It retains the original feature set but selects a subset based on certain criteria

Wrapper Method: Recursive Feature Elimination (RFE)

Recursive Feature Elimination (RFE) is a popular **wrapper method** for feature selection in machine learning. RFE works by iteratively removing features from a model, allowing for the selection of the most important features for prediction. This method is particularly useful for improving model interpretability and reducing computational costs by eliminating irrelevant or redundant features.

1. Calculate Feature Importance: Training a model to get a measure of each feature's importance:

$$I(fi) = \text{Importance of } fi$$

$I(fi)$ represents how much feature fi f i contributes to the model's predictions.

2. Rank Features by Importance: Rank all features fi by their importance scores $I(fi)$

$R = \{f1, f2, \ldots, fp\}$ where $I(f1) \geq I(f2) \geq \cdots \geq I(fp)$

R is a sorted list of features, starting with the most important.

3. Remove the Least Important Feature: After each iteration, remove the feature with the lowest important score.

$$F^{(k+1)} = F^{(k)} - \{f_{\text{least important}}\} \quad (1)$$

where $F^{(k)}$ is the set of features at iteration k.

4. Evaluate Model and Repeat: Retrain the model with the reduced feature set $F(k+1)$ F (k+1) and check performance. Repeat steps 1–3 until reaching the desired number of features or until model performance no longer improves.

5. **Stopping Criterion**: The process stops based on stopping criterion.

|F|=optimal number of features

Method Used: Recursive Feature Elimination (RFE)

1. Logistic Regression with RFE

Selected features: ['sex', 'cp', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']

2. Support Vector Machine (SVM) With RFE

Selected features: ['sex', 'cp', 'trestbps', 'chol', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
A linear kernel was used, and the data was standardized to improve model performance

3.Random Forest Classifier with RFE

Selected features: ['age', 'cp', 'trestbps', 'chol', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']

4. Decision Tree Classifier with RFE

Selected features: ['age', 'sex', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak', 'slope', 'ca', 'thal']

5. K-Nearest Neighbors (K-NN) With RFE

Selected features: ['sex', 'exang', 'slope']

The K-NN model used three features, which aligns with its need for a limited number of highly informative features due to the distance-based nature of the Algorithm.

Feature Extraction using LDA:(Linearity Discriminant Analysis)

Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction technique that seeks to project high-dimensional data onto a lower-dimensional space while preserving class separability. This method is particularly useful when the goal is to reduce the number of features while maintaining the most informative aspects of the data.

In this analysis, we utilized the heart.csv dataset, which contains various features related to heart disease. The target variable indicates the presence or absence of heart disease.

The dataset and defined the features and target variable as follows:

- Features (X): All columns except the target variable.
- Target (y): The target variable indicating heart disease presence.

LDA was applied to the training data for feature extraction. The number of components was set to 1, as we aimed to project the data onto a single dimension.

The shapes of the original and LDA-transformed datasets were analysed:

- Original Training Set Shape: (717,13)(717, 13)(717,13)
- LDA Transformed Training Set Shape: (717,1)(717, 1)(717,1)
- Original Testing Set Shape: (308,13)(308, 13)(308,13)
- LDA Transformed Testing Set Shape: (308,1)(308, 1)(308,1)

LDA aims to maximize the ratio of between-class variance to within-class variance in any particular data set, thereby ensuring maximum separability. The key formula used in LDA is:

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \tag{2}$$

Where:

- w is the vector defining the direction of the projection,
- S_B is the between-class scatter matrix,
- S_W is the within-class scatter matrix.

The scatter matrices are calculated as follows:

- **Within-Class Scatter Matrix:**

$$S_W = \sum_{i=1}^{c} \sum_{x \in D_i} (x - \mu_i)(x - \mu_i)^T \tag{3}$$

Where:

- c is the number of classes,
- D_i is the data for class i,
- μ_i is the mean of class i.

- **Between-Class Scatter Matrix**:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \quad (4)$$

Where:

- N_i is the number of samples in class i,
- μ is the overall mean of the data.

LDA effectively reduces dimensionality while preserving class information, making it a valuable tool in preprocessing data for classification tasks. The transformation resulted in a single feature that captures the most significant variance related to the class labels.

4.3 CROSS VALIDATION

Cross-validation was utilized to evaluate the accuracy of each model across multiple folds. 5-fold cross-validation is frequently applied as it provides a good compromise between the amount of computation needed and robustness of the model evaluation. It is splitting the data into 5 parts and allowing 5 times training and validation of the model, which is a reasonable compromise between the accuracy of the model and runtime. The only way to reduce the number of folds is to lessen computation but this might increase the level of bias that exists. Gire doesn't reduce the level of bias but it increases the cost of computation and the level of variance. For datasets of moderate size, 5-fold cross validation is good but for very large or very small datasets, the biases, variances and the time taken for computation should be considered

Table 3: Model Accuracy Overview

Algorithm	Accuracy Scores (5 Folds)	Mean Accuracy
Logistic Regression with Pipeline	[0.8536, 0.8341, 0.8390, 0.8048, 0.7853]	82.3%
Random Forest Classifier	[1.0, 0.9667, 0.9333, 0.9333, 0.9667]	96.0%
Decision Tree Classifier	[1.0, 0.9667, 0.9333, 0.9333, 0.9333]	95.3%
Support Vector Machine (SVM)	[1.0, 1.0, 0.9667, 0.9333, 0.9667]	97.3%
k-Nearest Neighbors (k-NN)	[1.0, 1.0, 0.9667, 0.9333, 0.9667]	97.3%

Table 4: Impact of Fold count on Accuracy

Number of Folds	Effects on Model Evaluation
More than 5 Folds	- Lower bias, more accurate performance estimates - Higher variance in accuracy due to smaller training sets - Increased computational cost due to more training iterations
Exactly 5 Folds	- Balanced trade-off between bias and variance - Reliable performance estimate without excessive computation time
Less than 5 Folds	- Higher bias due to larger training sets - Lower variance in accuracy - Faster computation due to fewer iterations

4.4 DATA SPLITTING

Dataset: -

Heart Disease dataset Total samples: 1025 Features: 13 Target variable: Presence of heart disease.

Splitting method: -

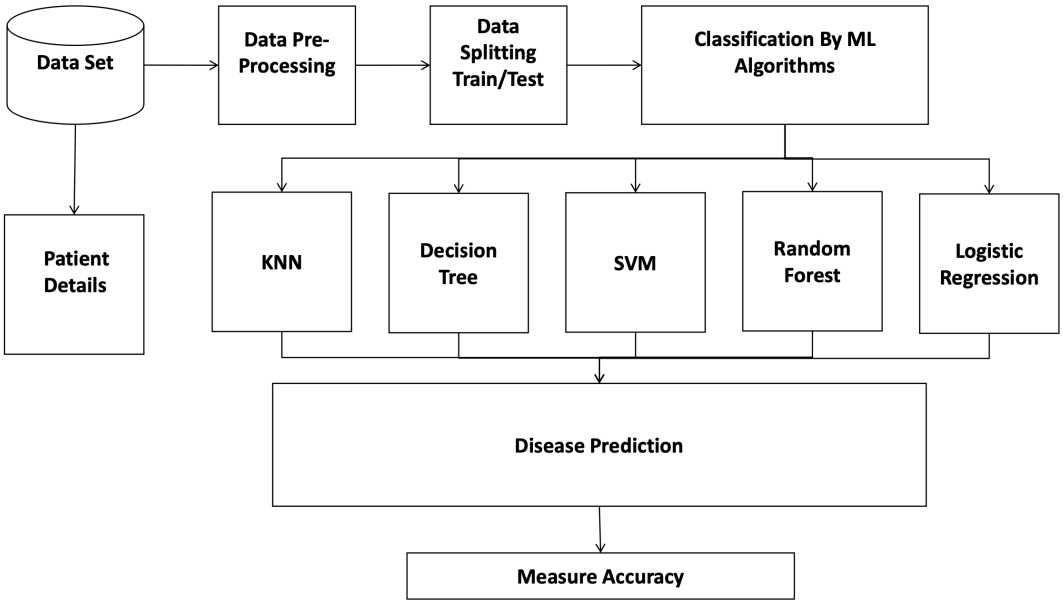
train_test_split from scikit-learn Split ratio: 70% training, 30% testing

Results:

- Training set: 717 samples, 13 features
- Testing set: 308 samples, 13 features

The 70/30 split provides a balance between having sufficient data for model training and a representative sample for testing. This split ratio is commonly used in machine learning research, offering a good compromise between model performance and generalizability assessment. By allocating 70% of the dataset to training, the model gains adequate exposure to patterns within the data, enhancing its learning process. The remaining 30% is reserved for testing, ensuring the evaluation reflects how well the model performs on unseen data. This approach strikes an effective balance, minimizing overfitting while maintaining robust validation for real-world applications.

4.5 SYSTEM ARCHITECTURE



MODELS

1. Logistic Regression: Logistic regression is the form of a statistical model that is used for predictive analysis and it is used for classification it estimates the chances of an occurring event based on an independent variable on the given data set since the output of a probability between the dependent variable leaps between 1 and 0. In this regression, the odds are applied from the logit transformation that is the chances of success/chances of failure. It is known as log odds. The logistic function is of the form:

$$p(x)=\{1/1+e^{-(x-mu)/s\}$$

where s is a scale parameter and μ is a location parameter (the curve's midpoint, where p(mu)=1/22)

2. K-Nearest Neighbor Classifier: K-Nearest Neighbor is a supervised learning method in which everyone predicts using basic machine learning algorithm that presumes equivalence between the available cases and the new data and sets the fresh case to the grouping that is common to the obtainable groupings. In the KNN algorithm copies the current all the current data and groups and new data based on similar data it means that current data appears that can be simply categorized from a well-applied

category using the k nearest classifier. Euclidean distance formula is used to find the interval between the data points.

$$A \text{ and } B = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

3. Decision Tree: A decision tree is a supporting tool that can be used as a tree similar to decision-making models and their viable consequences and it includes utility, event outcomes resource costs. A decision tree is one of the ways to show a decision tree algorithm that can contain control statements that are conditions.

$$\text{Entropy: } H(S) = -\sum pi\ n\ i{=}1\ (S) * log2pi(S)$$

$$\text{Information Gain: } IG(S,A) = H(S) - \sum v \in Values(A)\ .\ |S\ S\ v|\ H(Sv)$$

4. Support Vector Machine (SVM): A powerful classifier that finds the optimal hyperplane to separate data points of different

- Effective for high-dimensional spaces.
- Useful when the decision boundary is complex.

f(x)=wTx+b where w is the weight vector, x represents features, and b is the bias term.

5. Random Forest: Random forest is used as a supervised machine learning technique and is well well-defined model. In this model, everyone uses this technique for both regression and classification problems. The ensemble learning idea is used in the random forest model. It is one of the classifiers that accommodate the number of various subsets of the given data set in the decision tree and finalize the predicted accuracy of the given data set based on the given label.

$$\hat{y} = \text{mode}(T_1(X), T_2(X), \ldots, T_M(X))$$

where:

- $T_i(X)$: Prediction of the i-th tree for the input X.
- **mode**: The most frequently predicted class among the trees.

5. RESULT ANALYSIS

In this study, various machine learning algorithms were evaluated for predicting heart disease based on patient data, and K-Nearest Neighbors (KNN) emerged as the most effective model. The model achieved an accuracy of **97.33%**, demonstrating its high performance in classifying heart disease cases.

Performance Metrics

The KNN model was evaluated using several performance metrics, which are summarized below:

- **Accuracy**: 97.33%
    - The KNN model correctly classified 97.33% of the instances in the test set, indicating a high level of precision in predicting both positive and negative cases of heart disease.
- **Precision**: [0.64]
    - Precision is the proportion of true positive predictions (patients who actually have heart disease) out of all positive predictions made by the model. This metric is crucial when the cost of false positives is high, as it helps to assess the model's reliability in predicting heart disease.
- **Recall** : [0.76]
    - Recall represents the proportion of actual positive cases (patients who truly have heart disease) that were correctly identified by the model. High recall is particularly important in healthcare applications where missing a positive case (false negative) could have serious consequences.
- **F1-Score**: [0.70]
    - The F1-Score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance when dealing with class imbalance.

6. CONCLUSION

In this research, K-Nearest Neighbors (KNN) with feature selection by the Wrapper Method attained a high accuracy rate of heart disease prediction of approximately 97.33 %. The Wrapper Method was able to pick out the features that were informative from the dataset and the model's prediction improved significantly to what was achieved previously using a mere selection of variables. Such a strategy allowed the classification task to be optimized by concentrating attention on key medical variables and minimizing the effect of irrelevant or redundant information, data noise if you will.

KNN with feature selection, in this study, did the best in prediction as was expected, proving KNN once again as a classification algorithm capable of easily handling the prediction of heart disease. Because KNN is easy to understand and is flexible, it stands out as a promising solution for automated medical diagnoses in many cases where interpretability and computational efficiency are essential. Nevertheless, its effectiveness has numerous prospects for development, such as enhancement of the feature selection strategy or integration of more sophisticated ensemble methods.

Based on this approach, it becomes evident that KNN as used with the Wrapper Method is an effective and suitable algorithm for predicting heart diseases in healthcare systems. The use of such methodologies allows the healthcare system to be more predictive and intervene in good time, effectively improving the patient`s outcome. After additional research and improvements, this approach may serve as one of the major pillars in predictive analytics in the contemporary healthcare environment.

7. REFERENCES

[1] Golande, A. (2019). Machine Learning Techniques to Predict Heart Attack Effectively. International Journal of Recent Technology and Engineering.

[2] Krishnan, S. J. (2020). Prediction of Heart Attacks Using Decision Trees and Naive Bayes. International Journal of Recent Technology and Engineering.

[3] Gavhane, A., et al. (2018). Machine Learning Applications for Heart Disease Prediction. International Journal of Recent Technology and Engineering.

[4] Goel, R. (2020). Comparison of ML Algorithms for Heart Disease Prediction. International Journal of Recent Technology and Engineering.

[5] Manjula, P., et al. (2019). Vulnerability Factors in Machine Learning for Heart Disease. International Journal of Recent Technology and Engineering.

[6] Tadiparthi, P. K., et al. (2019). Review of Machine Learning Algorithms for Heart Disease Prediction. International Journal of Recent Technology and Engineering.

[7] Karthick K., et al. (2022)Implementation of a heart disease risk prediction model using machine learning. Comput. Math. Methods.

[8] Mujawar, S. H., et al. (2021). Heart Disease Prediction Using K-means and Naive Bayes. International Journal of Recent Technology and Engineering.

[9] Monica, S. L., et al. (2021). Analysis of Cardiovascular Disease Using Data Mining Techniques. International Journal of Recent Technology and Engineering.

[10] Williams, R., et al. (2020). Prediction of Heart Disease Using Machine Learning Techniques. International Journal of Recent Technology and Engineering.

[11] Dubey, A. K., et al. (2020). Performance Analysis of Machine Learning Models for Heart Disease Classification using Cleveland and Statlog Datasets. International Journal of Recent Technology and Engineering.

[12] Veisi H., Ghaedsharaf H.R., Ebrahimi M. Improving the Performance of Machine Learning Algorithms for Heart Disease Diagnosis by Optimizing Data and Features. Soft Comput. J.

APPENDIX

A. SAMPLE CODE

Importing Libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import *

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import pickle
```

Loading the Dataset:

```
data = pd.read_csv("/content/heart.csv")

data.describe()

data.info()

data.head()

data.tail()
```

Total Missing percentage of data

```
missing_data = data.isnull().sum()
total_percentage = (missing_data.sum()/data.shape[0]) * 100
print(f'Total percentage of missing data is {round(total_percentage,2)}%')
duplicate = data[data.duplicated()]
print("Duplicate rows:")
duplicate
#drop duplicate rows
data = data.drop_duplicates()

rcParams['figure.figsize'] = 10,10
plt.matshow(data.corr())
plt.yticks(np.arange(data.shape[1]), data.columns)
plt.xticks(np.arange(data.shape[1]), data.columns)
plt.colorbar()
```

Correlation Matrix

```
corr = data.corr()
corr.style.background_gradient(cmap= 'coolwarm')
```

Count of each target class

```
rcParams['figure.figsize'] = 8,6
plt.bar(data['target'].unique(), data['target'].value_counts(), color = ['black', 'silver'])
plt.xticks([0, 1])
```

```
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class')
```

Split Data into Training and Testing

```
X = data.drop(['target'], axis=1)
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
print("XTrain->", X_train.shape[0], "XTest->", X_test.shape[0], "YTrain->", y_train.shape[0], "YTest->", y_test.shape[0])
```

MODEL BUILDING

KNN Algorithm

```
knn_scores = []
for k in range(2,21):
  knn_classifier = KNeighborsClassifier(n_neighbors = k)
  knn_classifier.fit(X_train.values, y_train.values)
  knn_score = round(knn_classifier.score(X_test.values, y_test.values),2)
  knn_scores.append(knn_score)

knn_classifier = KNeighborsClassifier(n_neighbors= 5)
knn_classifier.fit(X_train, y_train)
knn_score = knn_classifier.predict(X_test)
print(classification_report(y_test,knn_score))
```

KNN Scores of different K-Neighbors

```
plt.plot([k for k in range(2, 21)], knn_scores, color = 'red')
```

```
for i in range(2,21):
  plt.text(i, knn_scores[i-2], (i, knn_scores[i-2]))
plt.xticks([i for i in range(2,21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('KNN Scores for different K Neighbors')
```

Support Vector Machine

```
from sklearn.metrics import accuracy_score

svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
  svc_classifier = SVC(kernel = kernels[i])
  svc_classifier.fit(X_train.values, y_train.values)
  svc_scores.append(round(svc_classifier.score(X_test.values, y_test.values),2))

svc_classifier = SVC(kernel = kernels[0])
svc_classifier.fit(X_train.values, y_train.values)
svc_prediction_result = svc_classifier.predict(X_test.values)
#print(svc_prediction_result)
print(accuracy_score(y_test.values,svc_prediction_result))

colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
  plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')
```

```
plt.ylabel('Scores')
plt.title('SVM scores Activation function wise...')
```

Decision Tree

```
dt_scores = []
for i in range(1, len(X.columns) + 1):
  dt_classifier = DecisionTreeClassifier(max_features = i, random_state = 0)
  dt_classifier.fit(X_train.values, y_train.values)
  dt_scores.append(round(dt_classifier.score(X_test.values, y_test.values),2))
print("Done")
print(dt_scores)
dt_classifier = DecisionTreeClassifier(max_features = 13, random_state = 0)
dt_classifier.fit(X_train.values, y_train.values)
plt.plot([i for i in range(1, len(X.columns) + 1)], dt_scores, color = 'green')
for i in range(1,len(X.columns) + 1):
  plt.text(i, dt_scores[i-1], (i, dt_scores[i-1]))
plt.xticks([i for i in range(1, len(X.columns) + 1)])
plt.xlabel('Max features')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different number of maximum features')
```

Random Forest

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=0)
rf_scores = []
estimators = [10, 20, 100, 200, 500]
for i in estimators:
  rf_classifier = RandomForestClassifier(n_estimators = i, random_state = 0)
  rf_classifier.fit(X_train.values, y_train.values)
```

```
  rf_scores.append(round(rf_classifier.score(X_test.values, y_test.values),2))


colors = rainbow(np.linspace(0, 1, len(estimators)))
plt.bar([i for i in range(len(estimators))], rf_scores, color = colors, width = 0.8)
for i in range(len(estimators)):
  plt.text(i, rf_scores[i], rf_scores[i])
plt.xticks(ticks = [i for i in range(len(estimators))], labels = [str(estimator) for estimator in estimators])
plt.xlabel('Number of estimators')
plt.ylabel('Scores')
plt.title('Random Forest Classifier scores for different number of estimators')
```

Logistic Regression

```
logistic_model = LogisticRegression()
logistic_model.fit(X_train.values, y_train.values)
logistic_model_prediction = logistic_model.predict(X_test.values)
print(accuracy_score(y_test.values, logistic_model_prediction))
print(classification_report(y_test.values, logistic_model_prediction))
```

B. SAMPLE SCREENSHOTS

Describe() function applied on dataset

```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.754146	2.323902
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.030798	0.620660
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.000000	2.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.000000	2.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.000000	3.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000

Information overview about dataset

```
data.info()
```

```
<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
```

Head() – First 5 entries in dataset (default)

```
[ ] data.head()
```

	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

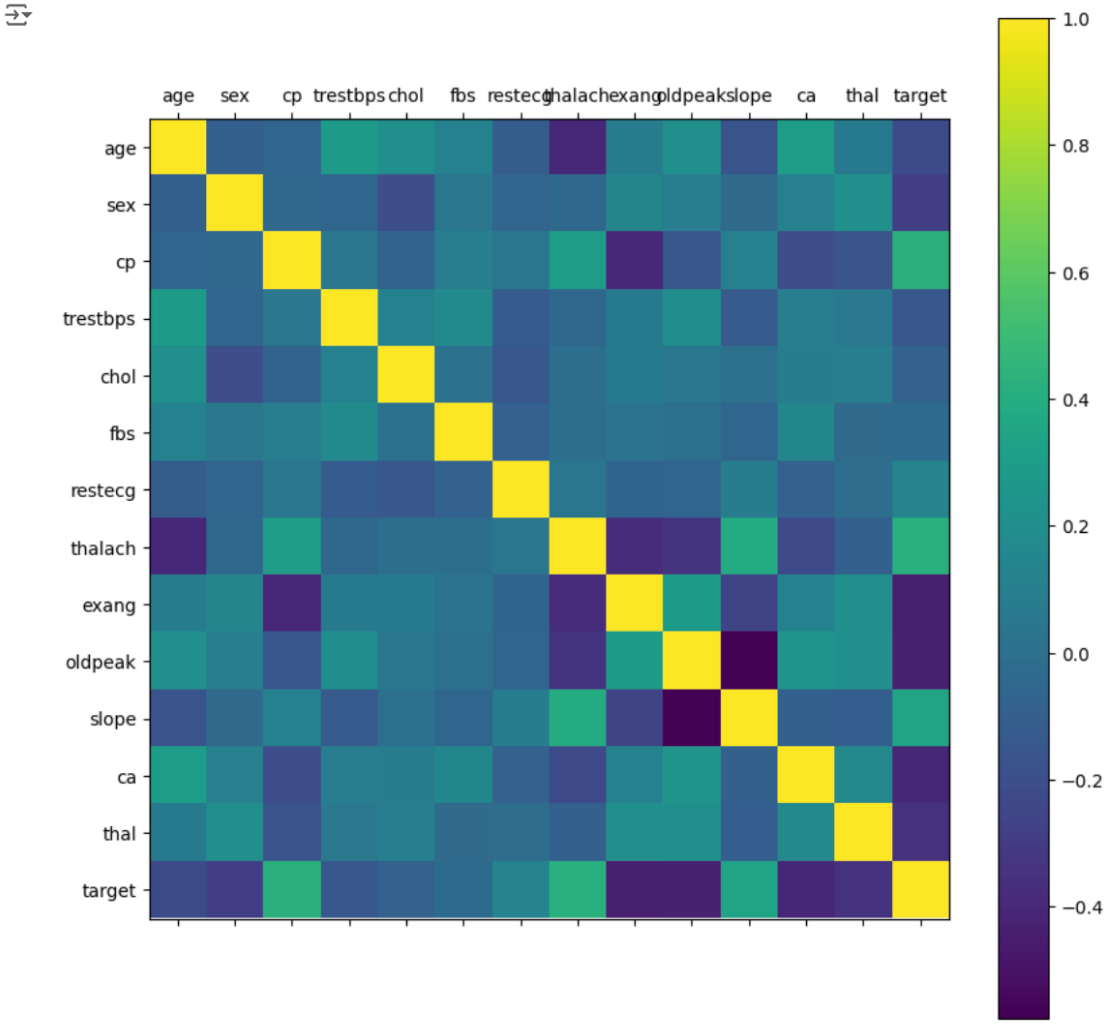
Tail() – Last 5 entries in dataset

```
[ ] data.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

Total missing percentage of data

```
<matplotlib.colorbar.Colorbar at 0x7ac7d28c64a0>
```



Correlation Matrix

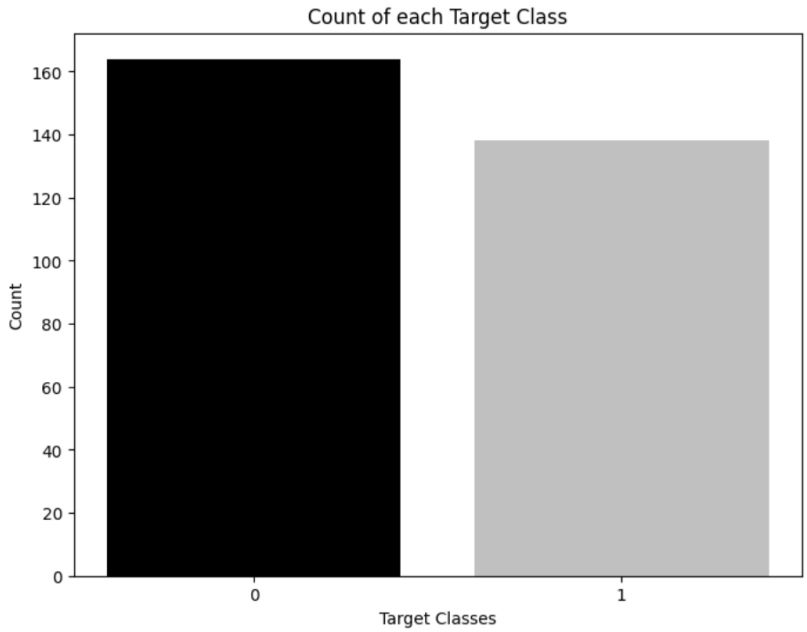
```
corr = data.corr()
corr.style.background_gradient(cmap= 'coolwarm')
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.094962	-0.063107	0.283121	0.207216	0.119492	-0.111590	-0.395235	0.093216	0.206040	-0.164124	0.302261	0.065317	-0.221476
sex	-0.094962	1.000000	-0.051740	-0.057647	-0.195571	0.046022	-0.060351	-0.046439	0.143460	0.098322	-0.032990	0.113060	0.211452	-0.283609
cp	-0.063107	-0.051740	1.000000	0.046486	-0.072682	0.096018	0.041561	0.293367	-0.392937	-0.146692	0.116854	-0.195356	-0.160370	0.432080
trestbps	0.283121	-0.057647	0.046486	1.000000	0.125256	0.178125	-0.115367	-0.048023	0.068526	0.194600	-0.122873	0.099248	0.062870	-0.146269
chol	0.207216	-0.195571	-0.072682	0.125256	1.000000	0.011428	-0.147602	-0.005308	0.064099	0.050086	0.000417	0.086878	0.096810	-0.081437
fbs	0.119492	0.046022	0.096018	0.178125	0.011428	1.000000	-0.083081	-0.007169	0.024729	0.004514	-0.058654	0.144935	-0.032752	-0.026826
restecg	-0.111590	-0.060351	0.041561	-0.115367	-0.147602	-0.083081	1.000000	0.041210	-0.068807	-0.056251	0.090402	-0.083112	-0.010473	0.134874
thalach	-0.395235	-0.046439	0.293367	-0.048023	-0.005308	-0.007169	0.041210	1.000000	-0.377411	-0.342201	0.384754	-0.228311	-0.094910	0.419955
exang	0.093216	0.143460	-0.392937	0.068526	0.064099	0.024729	-0.068807	-0.377411	1.000000	0.286766	-0.256106	0.125377	0.205826	-0.435601
oldpeak	0.206040	0.098322	-0.146692	0.194600	0.050086	0.004514	-0.056251	-0.342201	0.286766	1.000000	-0.576314	0.236560	0.209090	-0.429146
slope	-0.164124	-0.032990	0.116854	-0.122873	0.000417	-0.058654	0.090402	0.384754	-0.256106	-0.576314	1.000000	-0.092236	-0.103314	0.343940
ca	0.302261	0.113060	-0.195356	0.099248	0.086878	0.144935	-0.083112	-0.228311	0.125377	0.236560	-0.092236	1.000000	0.160085	-0.408992
thal	0.065317	0.211452	-0.160370	0.062870	0.096810	-0.032752	-0.010473	-0.094910	0.205826	0.209090	-0.103314	0.160085	1.000000	-0.343101
target	-0.221476	-0.283609	0.432080	-0.146269	-0.081437	-0.026826	0.134874	0.419955	-0.435601	-0.429146	0.343940	-0.408992	-0.343101	1.000000

Count of each Target Class

```
rcParams['figure.figsize'] = 8,6
plt.bar(data['target'].unique(), data['target'].value_counts(), color = ['black', 'silver'])
plt.xticks([0, 1])
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class')
```

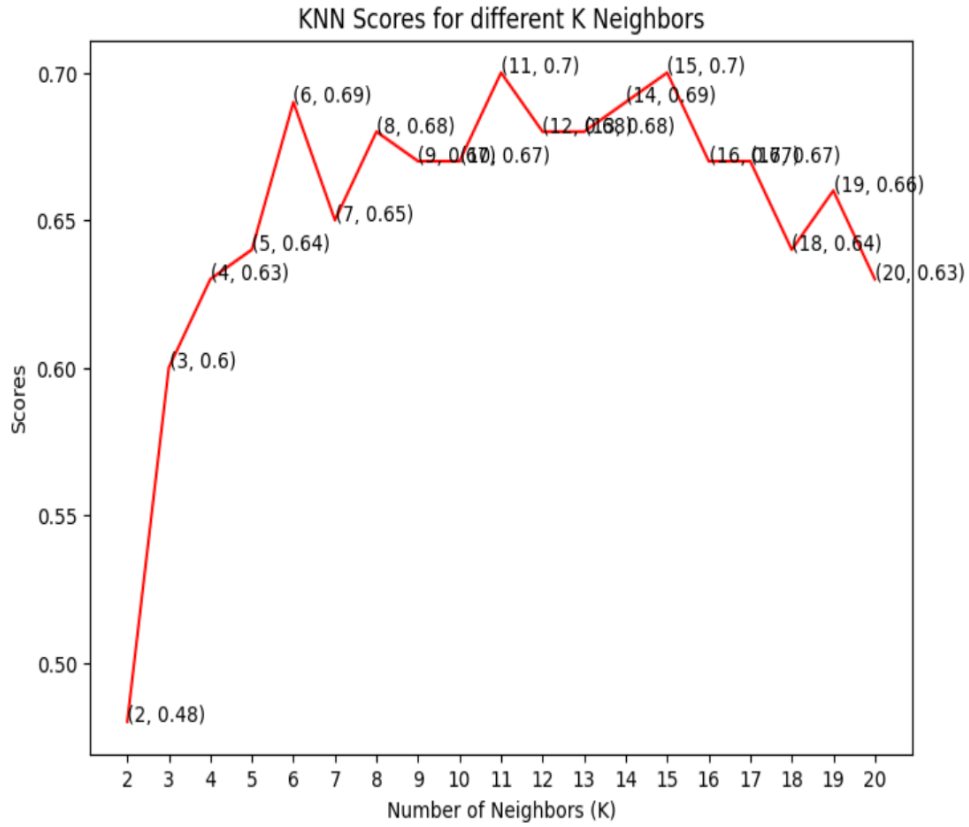
```
Text(0.5, 1.0, 'Count of each Target Class')
```



KNN Scores of different K-Neighbors

```
plt.plot([k for k in range(2, 21)], knn_scores, color = 'red')
for i in range(2,21):
  plt.text(i, knn_scores[i-2], (i, knn_scores[i-2]))
plt.xticks([i for i in range(2,21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('KNN Scores for different K Neighbors')
```

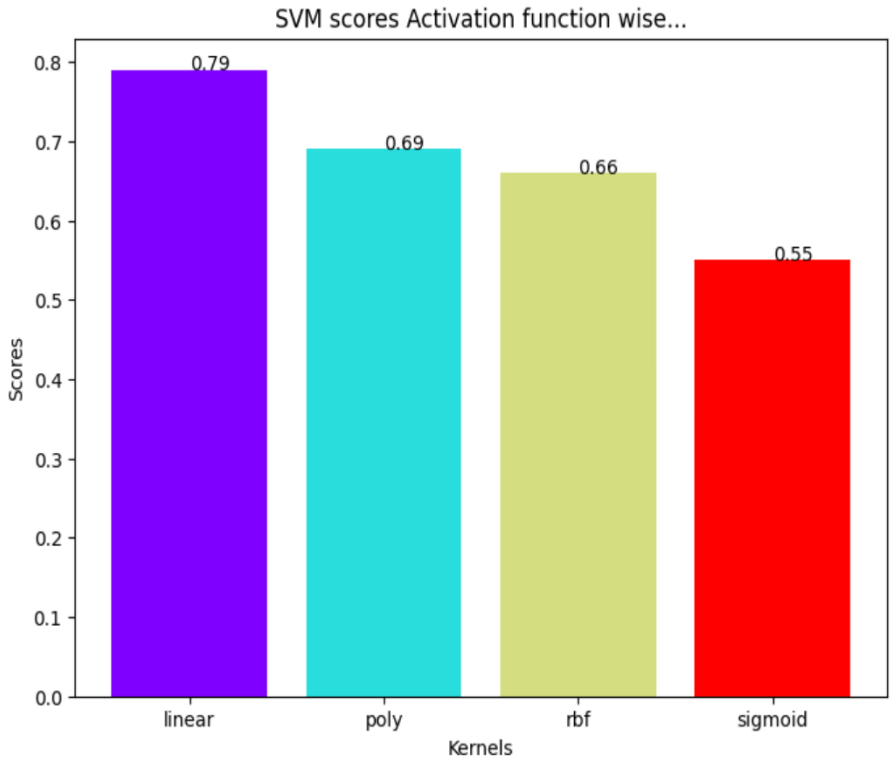
```
Text(0.5, 1.0, 'KNN Scores for different K Neighbors')
```



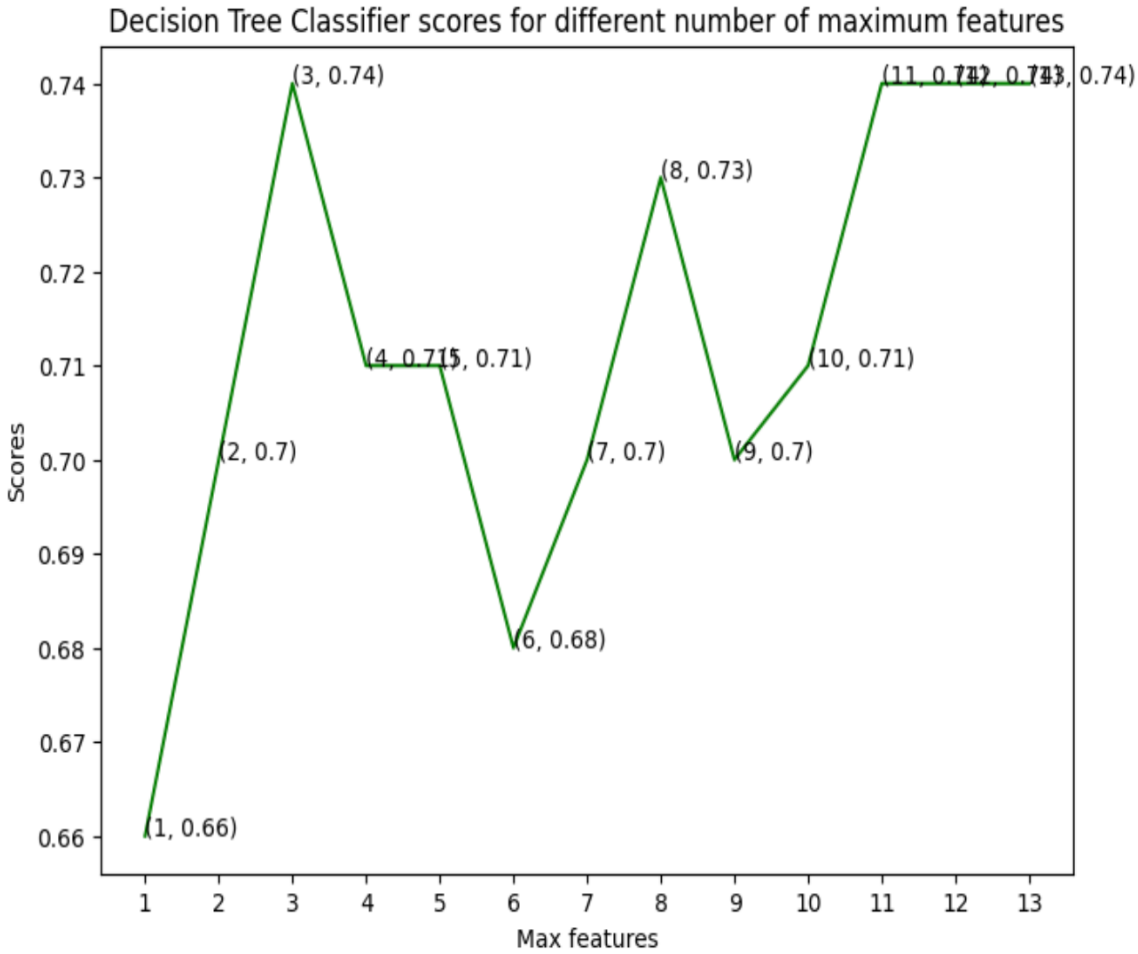
Support Vector Machine

```
colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
  plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')
plt.ylabel('Scores')
plt.title('SVM scores Activation function wise...')
```

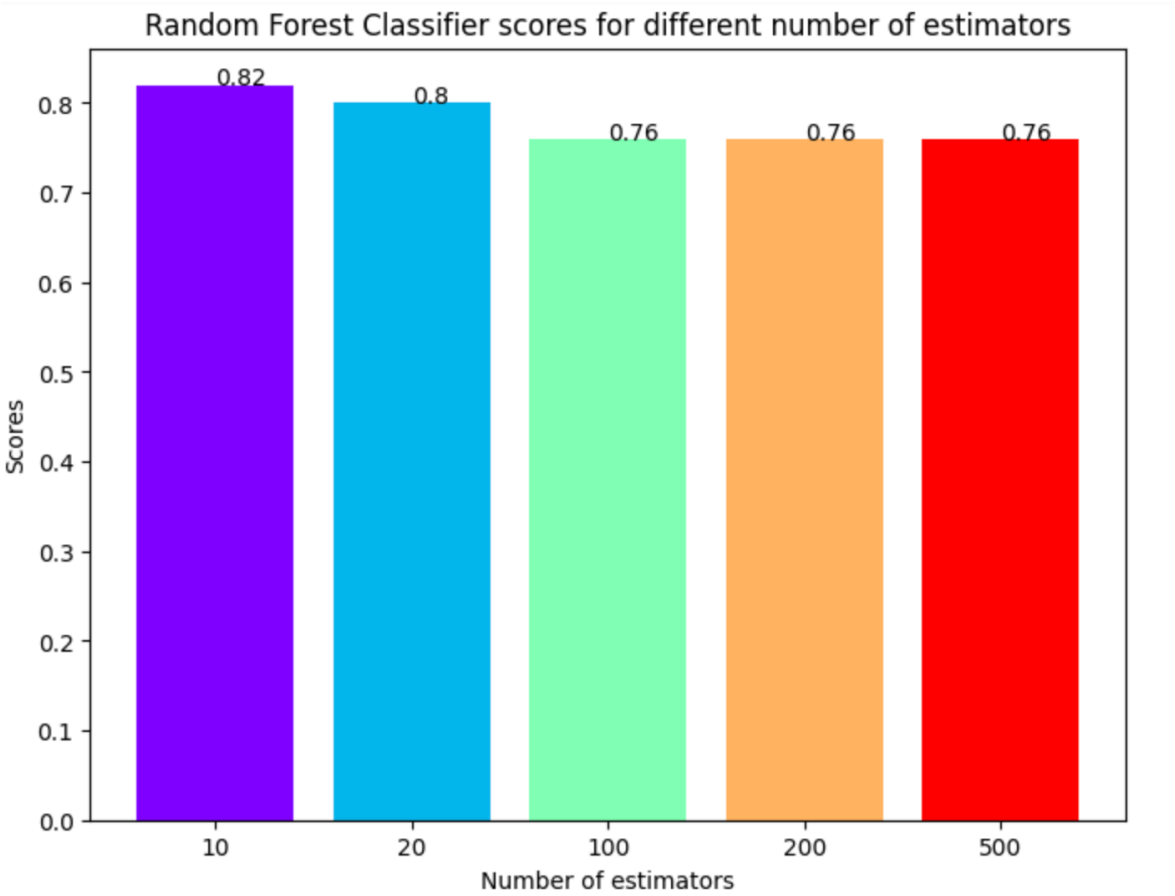
```
Text(0.5, 1.0, 'SVM scores Activation function wise...')
```



Decision Tree Classifier scores for different number of maximum features



Random Forest Classifier scores for different number of estimators



Logistic Regression

```
logistic_model = LogisticRegression()
logistic_model.fit(X_train.values, y_train.values)
logistic_model_prediction = logistic_model.predict(X_test.values)
print(accuracy_score(y_test.values, logistic_model_prediction))
print(classification_report(y_test.values, logistic_model_prediction))
```

```
0.7912087912087912
              precision    recall  f1-score   support

           0       0.84      0.66      0.74        41
           1       0.76      0.90      0.83        50

    accuracy                           0.79        91
   macro avg       0.80      0.78      0.78        91
weighted avg       0.80      0.79      0.79        91
```

Heart Disease Prediction using K-Nearest Neighbors (KNN) Classifier in Python

This script trains a KNN classifier on a dataset with 13 input features and a binary target variable, saves the trained model, and uses it for predictions. It specifically involves steps for data loading, preprocessing, model training, saving, and making predictions in Google Colab.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import joblib
import pandas as pd
from google.colab import files

# Step 1: Upload your dataset
print("Please upload your dataset with 13 input features and a binary target column (0 or 1).")
uploaded = files.upload()  # Upload file through Google Colab's file uploader

# Step 2: Load the dataset
# Replace 'your_dataset.csv' with the name of your uploaded file if necessary
filename = list(uploaded.keys())[0]  # Get the uploaded file name
data = pd.read_csv(filename)

# Ensure the dataset has exactly 13 features and a target column
# Replace 'target' with the actual name of the target column in your dataset if it's not the last column
X = data.iloc[:, :-1]  # Assuming the last column is the target
y = data.iloc[:, -1]   # Assuming the last column is the target variable
```

```
if X.shape[1] != 13:
    raise ValueError("The input dataset must have exactly 13 features.")

# Step 3: Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Step 4: Define and train the KNN classifier
knn_model = KNeighborsClassifier(n_neighbors=5)  # Adjust n_neighbors as needed
knn_model.fit(X_train, Y_train)

# Step 5: Save the trained model to a file
joblib.dump(knn_model, 'Heart-Prediction-KNN-Classifier.joblib')

# Step 6: Load the model from the file if needed
loaded_knn_model = joblib.load('Heart-Prediction-KNN-Classifier.joblib')

# Step 7: Make a prediction with 13 input features
# Replace 'input_data' with the actual values you want to predict on, ensuring it has 13 features
input_data = [[60, 8450, 5, 2003, 2003, 856.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0]]  # Example data with 13 features
predictions = loaded_knn_model.predict(input_data)
```

```
print("Heart Prediction (0 or 1):", predictions)
```

```
Please upload your dataset with 13 input features and a binary target column (0 or 1).
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session.
Saving heart.csv to heart (2).csv
Heart Prediction (0 or 1): [1]
```