CHAPTER 1

1. INTRODUCTION

1.1 . Overview:

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are Supervised Learning, Unsupervised Learning and Reinforcement Learning.

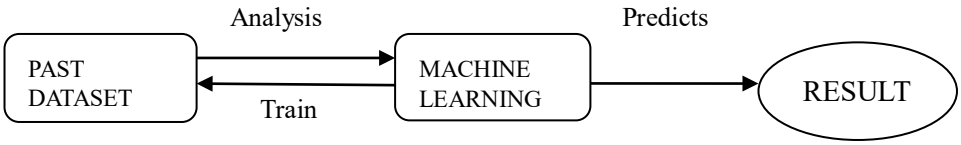


Fig 1.1 Process of Machine Learning

This project based on Supervised Learning. Supervised Learning is defined as when a model gets trained on a "Labelled Dataset". Labelled datasets have both input and output parameters. In Supervised Learning algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labelled.

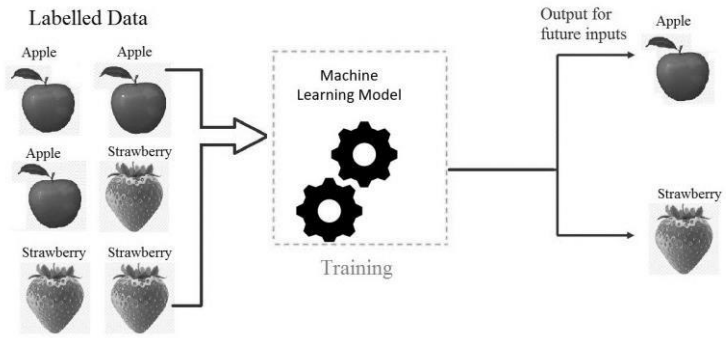


Fig 1.2 Process of Supervised Learning

1.2 . Project Description:

The major challenge in heart disease is its detection. There are instruments available which can predict heart disease but either they are expensive or are not efficient to calculate chance of heart disease in human. Early detection of cardiac diseases can decrease the mortality rate and over all complications. However, it is not possible to monitor patients every day in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more sapience, time and expertise. Since we have a good amount of data in today's world, we can use various machine learning algorithms to analyse the data for hidden patterns. The hidden patterns can be used for health diagnosis in medicinal data.

1.3 . Goal Of The Project:

Create accurate models: The main goal is to create machine learning models that can accurately forecast the risk of heart disease. Early detection: Prevention of heart disease depends on early detection. Patients who are at a high risk of developing heart disease should be able to be identified by the machine learning models, who should then be able to provide them the proper interventions. The ultimate goal is to improve healthcare outcomes by identifying individuals who are at high risk for heart disease and giving them the proper data to prevent or manage the condition.

1.4 . Objectives:

i. The main objective of developing this project are:
ii. To develop machine learning model to predict future possibility of heart disease by implementing Random Forest.
iii. To determine significant risk factors based on medical dataset which may lead to heart disease.
iv. To analyze feature selection methods and understand their working principle.

CHAPTER 2

2. LITERATURE REVIEW:

2.1. Predicting Heart Disease at Early Stages using Machine Learning:

Predicting and detection of heart disease has always been a critical and challenging task for healthcare practitioners. Hospitals and other clinics are offering expensive therapies and operations to treat heart diseases. So, predicting heart disease at the early stages will be useful to the people around the world so that they will take necessary actions before getting severe. Heart disease is a significant problem in recent times; the main reason for this disease is the intake of alcohol, tobacco, and lack of physical exercise. Over the years, machine learning shows effective results in making decisions and predictions from the broad set of data produced by the health care industry. Some of the supervised machine learning techniques used in this prediction of heart disease are artificial neural network (ANN), decision tree (DT), random forest (RF), support vector machine (SVM), naïve Bayes) (NB) and k-nearest neighbour algorithm. Furthermore, the performances of these algorithms are summarized.

2.2. Data Science And Its Application In Heart Disease Prediction:

This paper is the attempt to make the heart disease prediction at very early stage. As it is well known fact that most of the death causing disease all over the world is heart disease then comes cancer which is also very chronic and dangerous disease which has haunted the human being all over the globe. This disease and problem do not occur all of a sudden. Scientist and Doctors reveals that it is a continues process and is the result of being on a particular lifestyle for long time and also results after giving some basic and common symptoms being occurring all of a sudden. Eventually what does happen in the heart attacks is, the heart is not able to pump the required amount of blood to the parts of the body and more over it itself also does not get enough blood supply due to blocked arteries in the heart chambers there, for it results in heart failure and deaths. This paper brings the concepts of data science and its algorithms to make a hybrid model which can predict the disease of heart in the patient in comfortable ample of time advance. Moreover, the system must suggest useful and precautionary steps to the patient which are of globally accepted standards well in advance. The hybrid model is to predict and suggest the heart patient with world class heart solutions made with the help of data science algorithms

namely Naïve Bayes, ANN, SVM and Hybrid Naïve Bayes, SVM, and ANN. The Accuracy, specificity, and sensitivity of Naïve Bayes, ANN, SVM and Hybrid Naïve Bayes, SVM, and ANN has been measured. The results nearly 2% increase in the accuracy in predicting the heart disease in hybrid model. Hybrid model also shows higher specificity and sensitivity by having 82.11% and 91.47 % respectively. This paper also considers different attributes and has shown positivity in the connections with the heart disease like genetics, physical activity, total fat consumption, stress and working conditions. This research shows new direction to the research area where these concepts can be applied to the smart devices, data science to revolutionize the heart disease diagnosis and cures. This project can be effective to the make awareness of the complexities of the heart disease.

2.3. A New Automatic Identification Method of Heart Failure Using Improved Support Vector Machine Based on Duality Optimization Technique:

Currently, Heart failure disease is considered a multifaceted clinical disease affecting millions of people worldwide. Hospitals and cardiac centers rely heavily on ECG as a regular tool for evaluating and diagnosing Heart failure disease as an initial stage. The process of Heart failure disease identification from the ECG signal aims to reduce the time of the diagnostic process for patients with heart failure and to improve the outcomes of the detection process applied to these patients. The information acquired from the ECG signal simplifies the laboratory evaluation and other traditional diagnosis methods to evaluate and diagnose Heart failure disease. Unluckily, the problem of segmentation of the ECG signal is complicated because of similarities in time interval and amplitude between several ECG signal as well as the presence of noise in ECG signals. Most cardiologists use the ECG signal to identify Heart failure disease and their decision depends on the identification process, to determine whether surgery or medical treatment is required. This paper offers a new identification technique to overcome the current problems, such as overlapping in heart rate duration from the time interval from one PQRST wave to the next. In this study, the aim is to develop a new automatic method using improved support vector machine to diagnosis HFD from ECG signals. This is particularly relevant to ECG signals for the diagnosis of HFD as the first step to treatment and care of patients in general and specifically those with early heart disease to increase their overall survival. This paper outlines a hybrid approach of dual SVM and nonparametric algorithm to spot HFD in ECG signals leading to increase reliability and accuracy of identification and

diagnosis of heart failure classes in the early stages using the proposed algorithm. The nonparametric algorithm is used to train SVM and its dual to get two models of SVM. The dual problem gives a different view that is better and sometimes simpler than the original problem. This feature is used to detect Heart failure disease in ECG signals by comparing the outputs of model and those of dual SVM model. Experiments show that the hybrid approach produces good results, is more efficient and increases accuracy of Heart failure disease detection with an acceptable accuracy of 94.97% when compared with other algorithms to which the paper refers to. This is especially noted in patients with multiple diseases who were not initially identified as heart failure.

2.4. HDPM: An Effective Heart Disease Prediction Model for a Clinical Decision Support System:

Heart disease, one of the major causes of mortality worldwide, can be mitigated by early heart disease diagnosis. A clinical decision support system (CDSS) can be used to diagnose the subjects' heart disease status earlier. This study proposes an effective heart disease prediction model (HDPM) for a CDSS which consists of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to detect and eliminate the outliers, a hybrid Synthetic Minority Over-sampling Technique-Edited Nearest Neighbor (SMOTE-ENN) to balance the training data distribution and XGBoost to predict heart disease. Two publicly available datasets (Statlog and Cleveland) were used to build the model and compare the results with those of other models (naive bayes (NB), logistic regression (LR), multilayer perceptron (MLP), support vector machine (SVM), decision tree (DT), and random forest (RF)) and of previous study results. The results revealed that the proposed model outperformed other models and previous study results by achieving accuracies of 95.90% and 98.40% for Stat log and Cleveland datasets, respectively. In addition, we designed and developed the prototype of the Heart Disease CDSS (HDCDSS) to help doctors/clinicians diagnose the patients'/subjects' heart disease status based on their current condition. Therefore, early treatment could be conducted to prevent the deaths caused by late heart disease diagnosis.

2.5. Heart Diseases Prediction based on Stacking Classifiers Model:

Cardiovascular Diseases (CVDs), or heart diseases, are one of the top-ranking causes of death worldwide. About 1 in every 4 deaths are related to heart diseases, which are broadly classified as various types of abnormal heart conditions. However, diagnosis of CVDs is a time-consuming process in which data obtained from various clinical tests are manually analyzed. Therefore, new approaches for automating the detection of such irregularities in human heart conditions should be developed to provide medical practitioners with faster analysis via reducing the time of obtaining a diagnosis and enhancing results. Electronic Health Records (EHRs) are often utilized to discover useful data patterns that help improve the prediction of machine learning algorithms. Specifically, Machine Learning contributes significantly to solving issues like predictions in various domains, such as healthcare. Considering the abundance of available clinical data, there is a need to leverage such information for the betterment of humankind. In this work, a predictive model is proposed for heart disease prediction based on the stacking of various classifiers in two levels (Base level and Meta level). Various heterogeneous learners are combined to produce the strong model outcome. The model obtained 92% accuracy in prediction with a precision score of 92.6%, sensitivity of 92.6%, and specificity of 91%. The performance of the model was evaluated using various metrics, including accuracy, precision, recall, F1-scores, and area under the ROC curve values.

CHAPTER 3

3. PROBLEM DEFINITION:

3.1. Existing System:

In this system, the input details are obtained from the patient. Then from the user inputs, using ML techniques heart disease is analyzed. Now, the obtained results are compared with the results of existing models within the same domain and found to be improved. The data of heart disease patients collected from the UCI laboratory is used to discover patterns with NN, DT, Support Vector machines SVM, and Naive Bayes. The results are compared for performance and accuracy with these algorithms. The proposed hybrid method returns results of 87% for F-measure, competing with the other existing methods.

3.1.1. Drawback:

i. Prediction of cardiovascular disease results is not accurate.

ii. Data mining techniques does not help to provide effective decision making.

iii. Cannot handle enormous datasets for patient records. iv. The system is not fully automated, it needs data from user for full diagnosis.

3.2. Proposed System:

After evaluating the results from the existing methodologies, Machine Learning Algorithm is used to train and test the machine to predict the data. In this Project used Machine Learning algorithm for classification of dataset. This Machine Learning algorithm uses the Python language. Initially 33% of the data was used to predict heart disease. Then 63% of the data from the dataset was used to predict heart disease. Let's use 65% of the data to predict heart disease from the dataset.

It provides an easy-to-use visual representation of the dataset, working environment and building the predictive analytics. Machine Learning process starts from a pre-processing data phase followed by feature selection based on data Analysing, classification of modelling performance evaluation. Some technique are used for in this project such as K-Nearest

Neighbor (KNN), Support Vector Machine (SVM), Decision Tree (DT) and Random forest technique is used to improve the accuracy of the result.

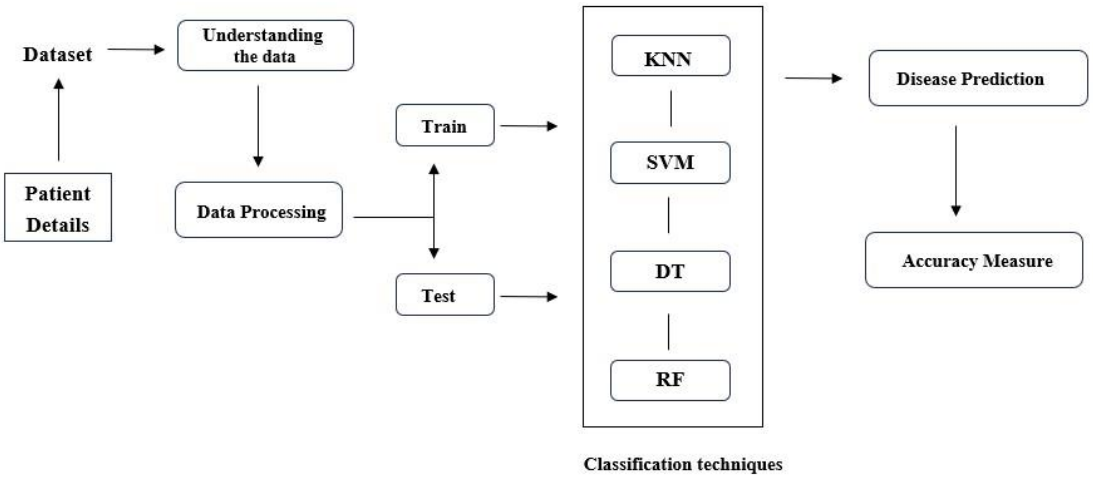


Fig: 3.1 Architecture Diagram

3.2.1. Advantage:

i. Cost effective for patients. ii. It is an accurate possibility. iii. Reduce the time complexity of doctors. iv. Increased accuracy for effective heart disease diagnosis.

v. Handles roughest(enormous) amount of data using random forest algorithm and feature selection.

CHAPTER 4

4. SYSTEM IMPLEMENTATION:

4.1. System Requirement:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sklearn, pandas, NumPy, mat plot lib and sea born.

4.1.1. Software Requirements:

i. Operating System: Windows

Windows is an operating system designed by Microsoft. The operating system is what allows you to use a computer. Windows comes preloaded on most new personal computers (PCs), which helps to make it the most popular operating system in the world.

ii. Tool: Colaboratory

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

4.1.2. Hardware Requirements:

i. Processor : Pentium IV/III

ii. Hard disk : Minimum80 GB

iii. RAM : Minimum 2GB

CHAPTER 5

5. MODULE:

5.1. K-Nearest Neighbors:

KNN is a simple, Supervised Machine Learning (ML) algorithm that can be used for classification or regression tasks - and is also frequently used in missing value imputation. It is based on the idea that the observations closest to a given data point are the most "similar" observations in a data set, and we can therefore classify unforeseen points based on the values of the closest existing points. By choosing *K*, the user can select the number of nearby observations to use in the algorithm.

5.1.1 Euclidean Distance:

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2]}$$

5.1.2 Manhattan Distance:

Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

$$d\left(x, y\right) = \sum_{i=1}^{n} \left|x_i - y_i\right|$$

5.1.3 Minkowski Distance:

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

$$d\left(x, y\right) = \left(\sum_{i=1}^{n} \left(x_i - y_i\right)^p\right)^{\frac{1}{p}}$$

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

5.2. Support Vector Machine:

A Support Vector Machine (SVM) algorithm is a non-probabilistic classifier aiming to generate hyperplanes that divide the data points of two classes in the vector space. For N number of features and M targets, SVM creates M-1 N-dimensional hyperplanes that separate data points of different classes from each other. The image below shows how "support" vectors are calculated such that the margin (or distance) between the vectors of two classes is the most. SVM optimizes this margin metric to find the best hyperplane for all the categories.

Thus, SVMs are popular for disease prediction since they can effectively categorize tabular data into different categories.

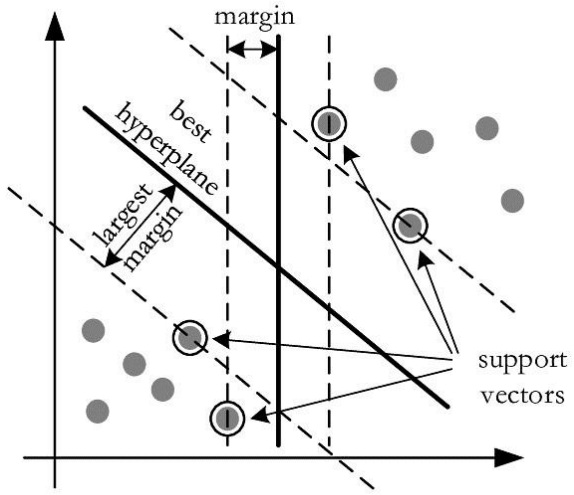


Fig: 5.1 Support Vector Machine

5.3. Decision tree:

A decision tree is a non-parametric supervised learning algorithm for classification and regression tasks**.** It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. Decision trees are used for classification and regression tasks, providing easy-to-understand models. In a decision tree, the output is mostly “yes” or “no”

The formula:

$$E(S) = -p_{(+)}\log p_{(+)} - p_{(-)}\log p_{(-)}$$

Here,

- p_+ is the probability of positive class.
- P_- is the probability of negative class.
- S is the subset of the training example.

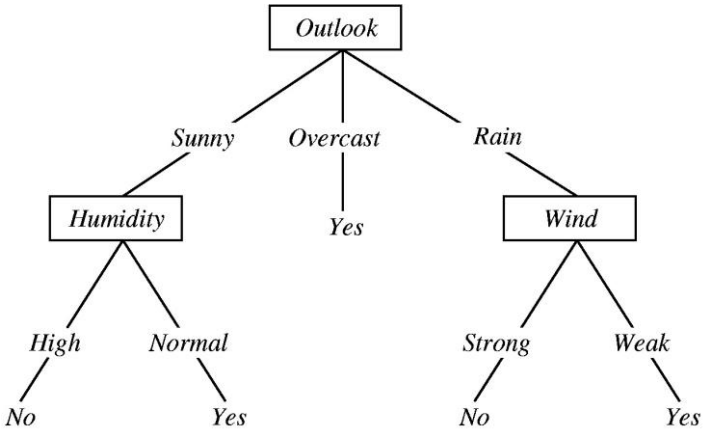


Fig 5.2 Decision Tree

5.4. Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision trees, resulting in *a* forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks. The following are the basic steps involved in performing the random forest.

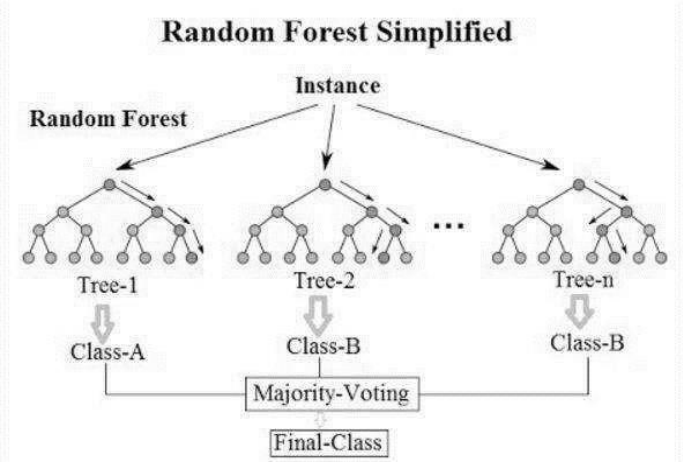


Fig 5.3 Random Forest

This ensemble classifier builds several decision trees and incorporates them to get the best result. For tree learning, it mainly applies bootstrap aggregating or bagging. For a given data, X = {x1, x2, x3, ..., xn} with responses Y = {x1, x2, x3, ..., xn} which repeats the bagging from b = 1 to B.

CHAPTER 6

6. SYSTEM TESTING:

6.1. White Box Testing:

It is testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers. It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing.

The term "White Box" was used because of the see-through box concept. The clear box or White Box name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

6.2. Unit Testing:

It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.

6.3. Testing for Memory Leaks:

Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application. Apart from above, a few testing types are part of both black box and white box testing. They are listed as below.

CHAPTER 7

RESULT

7.1. Import libraries:

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

7.2. Import dataset:

```
data = pd.read_csv("heart.csv")
print(data)
```

Output:

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      52    1   0       125   212    0        1      168      0      1.0
1      53    1   0       140   203    1        0      155      1      3.1
2      70    1   0       145   174    0        1      125      1      2.6
3      61    1   0       148   203    0        1      161      0      0.0
4      62    0   0       138   294    1        1      106      0      1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
1020   59    1   1       140   221    0        1      164      1      0.0
1021   60    1   0       125   258    0        0      141      1      2.8
1022   47    1   0       110   275    0        0      118      1      1.0
1023   50    0   0       110   254    0        0      159      0      0.0
1024   54    1   0       120   188    0        1      113      0      1.4

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
...     ...  ..   ...     ...
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]
```

7.3. Understanding the data:

```
data.head()
```

Output:

	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

```
data.isnull()
```

```
data.info()
```

Output:

```
<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
```

```
data.columns
```

Output:

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
data.shape
```

Output:

```
(1025, 14)
```

```
data.describe()
```

Output:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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	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	0.513171
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	0.500070
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	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	0.000000
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	1.000000
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	1.000000
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	1.000000

Output:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1021	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1022	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1023	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1024	False	False	False	False	False	False	False	False	False	False	False	False	False	False

1025 rows × 14 columns

```
data.isnull().sum()

data.nunique(axis=0)
```

Output:

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

```
age         41
sex          2
cp           4
trestbps    49
chol       152
fbs          2
restecg      3
thalach     91
exang        2
oldpeak     40
slope        3
ca           5
thal         4
target       2
dtype: int64
```

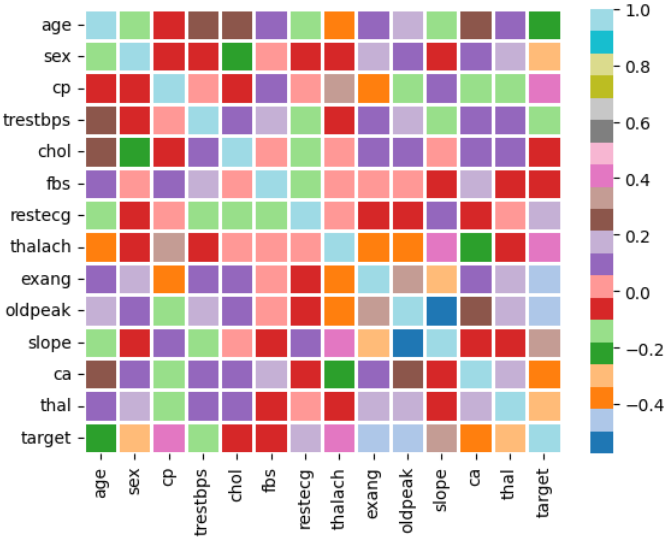
```
print(data.corr())
plt.show()
```

Output:

```
                age       sex        cp  trestbps      chol       fbs
age        1.000000 -0.103240 -0.071966  0.271121  0.219823  0.121243
sex       -0.103240  1.000000 -0.041119 -0.078974 -0.198258  0.027200
cp        -0.071966 -0.041119  1.000000  0.038177 -0.081641  0.079294
trestbps   0.271121 -0.078974  0.038177  1.000000  0.127977  0.181767
chol       0.219823 -0.198258 -0.081641  0.127977  1.000000  0.026917
fbs        0.121243  0.027200  0.079294  0.181767  0.026917  1.000000
restecg   -0.132696 -0.055117  0.043581 -0.123794 -0.147410 -0.104051
thalach   -0.390227 -0.380281  0.306839 -0.039264 -0.021772 -0.008866
exang      0.088163  0.139157 -0.401513  0.061197  0.067382  0.049261
oldpeak    0.208137  0.084687 -0.174733  0.187434  0.064880  0.010859
slope     -0.169105 -0.026666  0.131633 -0.120445 -0.014248 -0.061902
ca         0.271551  0.111729 -0.176206  0.104554  0.074259  0.137156
thal       0.072297  0.198424 -0.163341  0.059276  0.100244 -0.042177
target    -0.229324 -0.279501  0.434854 -0.138772 -0.099966 -0.041164

             restecg   thalach     exang   oldpeak     slope        ca
age        -0.132696 -0.390227  0.088163  0.208137 -0.169105  0.271551
sex        -0.055117 -0.049365  0.139157  0.084687 -0.026666  0.111729
cp          0.043581  0.306839 -0.401513 -0.174733  0.131633 -0.176206
trestbps   -0.123794 -0.039264  0.061197  0.187434 -0.120445  0.104554
chol       -0.147410 -0.021772  0.067382  0.064880 -0.014248  0.074259
fbs        -0.104051 -0.008866  0.049261  0.010859 -0.061902  0.137156
restecg     1.000000  0.048411 -0.065606 -0.050114  0.086086 -0.078072
thalach     0.048411  1.000000 -0.380281 -0.349796  0.395308 -0.207888
exang      -0.065606 -0.380281  1.000000  0.310844 -0.267335  0.107849
oldpeak    -0.050114 -0.349796  0.310844  1.000000 -0.575189  0.221816
slope       0.086086  0.395308 -0.267335 -0.575189  1.000000 -0.073440
ca         -0.078072 -0.207888  0.107849  0.221816 -0.073440  1.000000
thal       -0.020584 -0.098068  0.197201  0.202672 -0.094090  0.149014
target      0.134468  0.422895 -0.438029 -0.438441  0.345512 -0.382085

              thal    target
age       0.072297 -0.229324
sex       0.198424 -0.279501
cp       -0.163341  0.434854
trestbps  0.059276 -0.138772
chol      0.100244 -0.099966
fbs      -0.042177 -0.041164
restecg  -0.020584  0.134468
thalach  -0.098068  0.422895
exang     0.197201 -0.438029
oldpeak   0.202672 -0.438441
slope    -0.094090  0.345512
ca        0.149014 -0.382085
thal      1.000000 -0.337838
target   -0.337838  1.000000
```

```
sns.heatmap(data.corr(),annot=False,cbar=True,cmap
= "tab20",linewidths = 1,
linecolor = "white")
plt.show()
```

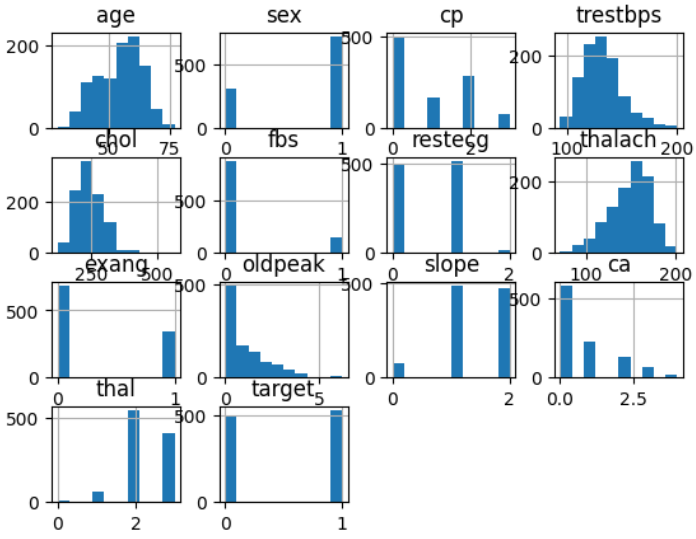
Output:




```
data.hist()
```

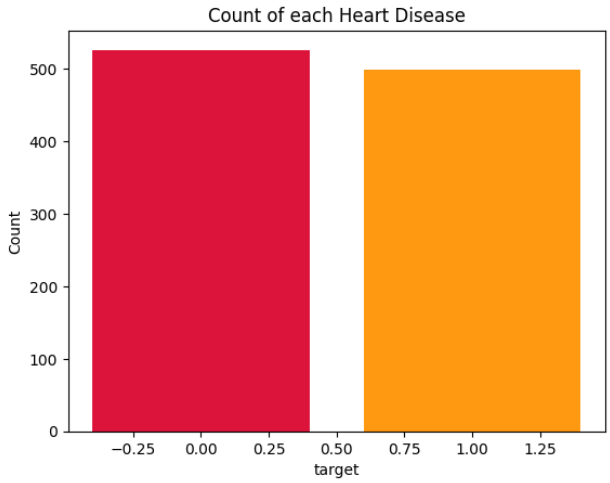
Output:

```
array([[<Axes: title={'center': 'age'}>, <Axes: title={'center': 'sex'}>,
        <Axes: title={'center': 'cp'}>,
        <Axes: title={'center': 'trestbps'}>],
       [<Axes: title={'center': 'chol'}>,
        <Axes: title={'center': 'fbs'}>,
        <Axes: title={'center': 'restecg'}>,
        <Axes: title={'center': 'thalach'}>],
       [<Axes: title={'center': 'exang'}>,
        <Axes: title={'center': 'oldpeak'}>,
        <Axes: title={'center': 'slope'}>,
        <Axes: title={'center': 'ca'}>],
       [<Axes: title={'center': 'thal'}>,
        <Axes: title={'center': 'target'}>, <Axes: >, <Axes: >]],
      dtype=object)
```



```
plt.bar(data['target'].unique(),
data['target'].value_counts(), color = ['#DC143C',
'#FF9912'])
plt.xlabel('target')
plt.ylabel('Count')
plt.title('Count of each Heart Disease')
```

Output:



7.4 Data Processing:

```
data = pd.get_dummies(data, columns =
['sex','cp','fbs','restecg','exang','slope','ca','thal'])


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach',
'oldpeak']
data[columns_to_scale] =
standardScaler.fit_transform(data[columns_to_scale])
```

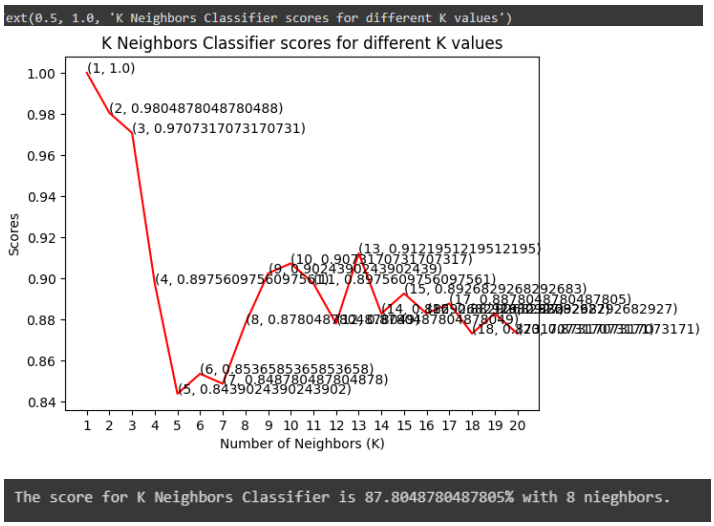
7.5. Test and Train:

```
y = data['target']
X = data.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size = 0.2,
random_state = 0)
```

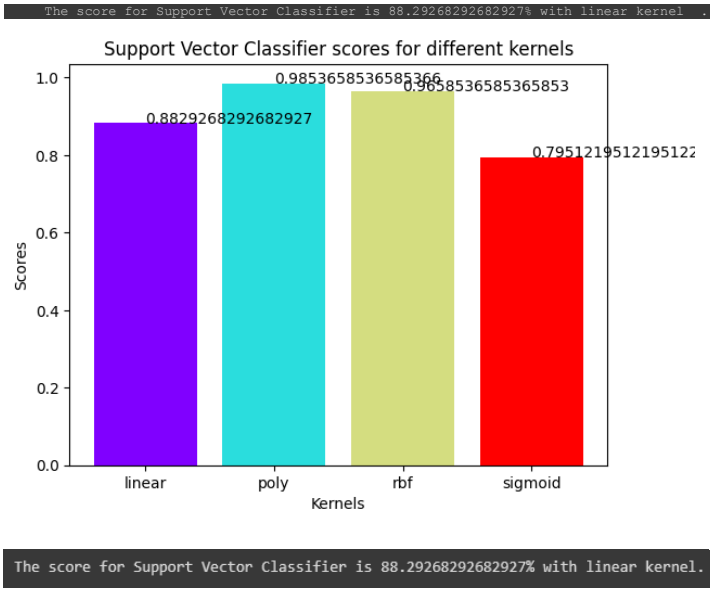
7.6. K-Nearest Neighbors:

```
from sklearn.neighbors import KNeighborsClassifier
knn_scores = []
for k in range(1,21):
knn_classifier = KNeighborsClassifier(n_neighbors =
k)
knn_classifier.fit(X_train, y_train)
knn_scores.append(knn_classifier.score(X_test,
y_test))
plt.plot([k for k in range(1, 21)], knn_scores, color =
'red')
for i in range(1,21):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K
values')
print("The score for K Neighbors Classifier is {}% with
{} nieghbors.".format(knn_scores[7]*100, 8))
```

Output:



Output:



7.8. Support Vector Machine:

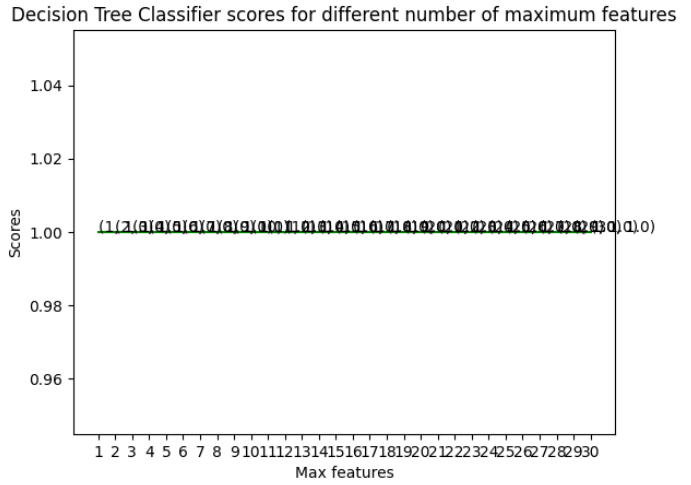
```
from sklearn.svm import SVC
svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    svc_classifier = SVC(kernel = kernels[i])
    svc_classifier.fit(X_train, y_train)
    svc_scores.append(svc_classifier.score(X_test, y_test))
from matplotlib.cm import rainbow
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('Support Vector Classifier scores for different kernels')
print("The score for Support Vector Classifier is {}% with {} kernel.".format(svc_scores[0]*100, 'linear'))
```

7.9. Decision tree:

```
from sklearn.tree import DecisionTreeClassifier
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, y_train)
    dt_scores.append(dt_classifier.score(X_test, y_test))
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')
print("The score for Decision Tree Classifier is {}% with {} maximum features.".format(dt_scores[17]*100, [2,4,18]))
```

Output:

```
Text(0.5, 1.0, 'Decision Tree Classifier scores for different number of maximum features')
```




```
The score for Decision Tree Classifier is 100.0% with [2, 4, 18] maximum features.
```

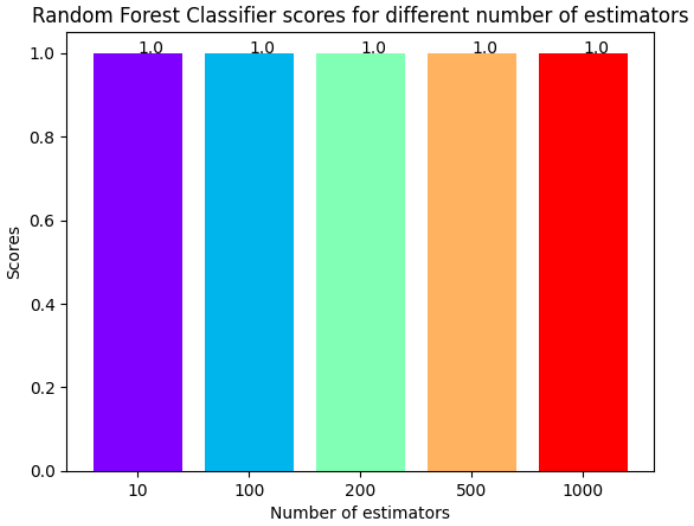
7.10. Random Forest:

```
from sklearn.ensemble import RandomForestClassifier

rf_scores = []

estimators = [10, 100, 200, 500, 1000]

for i in estimators:

    rf_classifier = RandomForestClassifier(n_estimators = i, random_state = 0)

    rf_classifier.fit(X_train, y_train)

    rf_scores.append(rf_classifier.score(X_test, y_test))

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')
```

```
print("The score for Random Forest Classifier is {}% with {} estimators.".format(rf_scores[1]*100, [100, 500]))
```

Output:

```
Text(0.5, 1.0, 'Random Forest Classifier scores for different number of estimators')
```




```
The score for Random Forest Classifier is 100.0% with [100, 500] estimators.
```

CHAPTER 8

8. CONCLUSION:

8.1. Project Conclusion:

In this paper, we proposed a method for heart disease prediction using machine learning techniques, these results showed a great accuracy standard for producing a better estimation result. By introducing new proposed Random forest classification, we find the problem of prediction rate without equipment and propose an approach to estimate the heart rate and condition. Sample results of heartrate are to be taken at different stages of the same subjects, we find the information from the above input via ML Techniques. Firstly, we introduced a support vector classifier based on datasets.

8.2. Future Enhancement:

In this project we made a model which gives us the best accuracy among all the machine learning algorithms and by using using that model we have made a web service 50 using stream lit which will connect the back end of ml model to the front end that we designed it, with of help of this many people can get their health condition priorly and might taken care accordingly. We have tried it on our local host. Weare going to implement by adding some features to our deployed model like adding all algorithms and by showing the ROC curve to the people who are using it to predict and make some changes and then we can implement it as an online website. This can also be used for medical professionals as a second reference. Phish Haven can be further enhanced by incorporating unsupervised learning, i.e., deep learning models

CHAPTER 9

9. REFERENCE:

- A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque, "Artificial intelligence driven mechanism
- for edge computing based industrial applications," IEEE Transactions on Industrial Informatics, 2019.

- A. H. Sodhro, Z. Luo, G. H. Sodhro, M. Muzamal, J. J. Rodrigues, and V. H. C. de Albuquerque, "Artificial intelligence based qos optimization for multimedia communication in iov systems," Future Generation Computer Systems, vol. 95, pp. 667–680, 2019.

- L. University, "Refit smart home dataset," https://repository.lboro.ac.uk/articles/REFIT Smart Home dataset/2070091,2019(accessedApril26, 2019).

- R, "Rstudio," 2019 (accessed October 23, 2019)

- H. A. Esfahani and M. Ghazanfari, ''Cardiovascular disease detection using a new ensemble classifier,'' in Proc. IEEE 4th Int. Conf. Knowl.Based Eng. Innov. (KBEI), Dec. 2017, pp. 1011–1014.

- F. Dammak, L. Baccour, and A. M. Alimi, ''The impact ofcriterion weights techniques in TOPSIS method of multi-criteria decision making in crisp and intuitionistic fuzzy domains,'' in Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE), vol. 9, Aug. 2015, pp. 1–8.

- R. Das, I. Turkoglu, and A. Sengur, ''Effective diagnosis of heart disease through neural networks ensembles,'' Expert Syst. Appl., vol. 36, no. 4, pp. 7675–7680, May 2009. doi: 10.1016/j.eswa.2008.09.013