Banaras Hindu University,Varanasi

HEART DISEASE PREDICTION USING MACHINE LEARNING

Submitted by
Rahul Kumar Vishwakarma

Under the guidance of
Dr.Manjari Gupta

In partial fulfillment of the award of Masters in Science
(Mathematics and Computing)

BHU
Banaras Hindu University

Centre for Interdisciplinary Mathematical Sciences
Banaras Hindu University,
Varanasi (Uttar Pradesh)
2022-2023

DECLARATION

I hereby declare that the thesis entitled, **"HEART DISEASE PREDICTION USING MACHINE LEARNING"** has been completed and written by me.

To best of my knowledge and belief, the work embodied in this thesis has not formed earlier the basis for the award of any Degree or similar title of this any other University or examining body.

I give an undertaking that the material included in the thesis from other sources is duly acknowledged. I have incorporated all the changes as suggested by the Pre-submission presentation Committee, if any.

Mr. Rahul Kumar Vishwakarma
Banaras Hindu University,Varanasi
21419MAC033

Date: 31/05 /2023
Place: Varanasi

CERTIFICATE

This is to certify that the dissertation entitled **"HEART DISEASE PREDICTION USING MACHINE LEARNING"**, submitted by **Rahul Kumar Vishwakarma** is the bonafide work completed under my supervision and guidance in partial fulfillment for the award of Masters in Science (Mathematics and Computing) of Banaras Hindu University, Varanasi (U.P).

Place:
Date: .....

Signature
Dr. Manjari Gupta(Associate Professor)
(Project Guide)
.....

ACKNOWLEDGEMENTS

The portion of success is brewed by the efforts put in by many individuals. It is constant support provided by people who give you the initiative, and who inspire you at each step of your endeavor that eventually helps you in your goal.

I wish to express my deep gratitude and hearty appreciation for the invaluable guidance of our professors throughout the span of this dissertation. We are indebted to our college **Co-Ordinator Dr.Manjari Gupta**

I am also thankful for our **Course Coordinator Dr. Raghvendra Chaubey,** and my project guide **Dr. Manjari Gupta** for his invaluable and elaborate suggestions. Their excellent guidance made me complete this task successfully.

Rahul Kumar Vishwakarma

ROLL NO:21419MAC033

DIVISION:

ABSTRACT

As we are living in the era of the most connective era, from mobile phones to utensils, from cars to cycles, from every aspect of life data is generated from everywhere. The medical domain is no exception to this revolution. Every sector is taking a decision based on some statistical tools and sophisticated algorithms. In medical patients, records are stored not for commercial benefit but to get insights about a disease so that certain patterns may be established. Heart Disease is something that can be early detected. In India, the leading cause of death is still Heart attacks, nowadays healthy people are getting heart attacks. But the diagnosis of heart disease requires a lot of money and headache as well. No one likes to be in doctors'. But in India, the majority can not afford a medical diagnosis. But the boom in Artificial Intelligence after the introduction of deep learning. Again artificial intelligence has gained a response in academia as well as in the industry. Now everybody is talking about artificial intelligence and machine learning. In the medical domain, artificial intelligence and machine learning are used extensively for studying the behavior of diseases and gaining insights from them. In this dissertation, I have also worked on a healthcare-related problem. Here I have to find out the impact of different factors on getting heart disease. The data set I have used is taken from Kaggle a popular website for machine learning enthusiasts where data sets are available for free and the integrity of the data is very well. In the problem, I have used different tools like Scikit-learn, Pandas, TensorFlow, Numpy, Matplotlib, and Seaborn to solve the problem. In the Scikit-learn library, there are so many machine-learning algorithms available. In this particular problem, we are using a supervised machine-learning algorithm to train our model. Some of the algorithms I used here are **DT**,**NB**,**KNN**, and tried to come up with a model which performs well on both training and test data. For dimensionality reduction, I used one of the popular techniques **PCA**. Finally, I evaluated models based on cross-validation. The most accurate model is adapted.

KEYWORDS

DT:Decision Tree
CNN: Convolutional Neural Network
ANN: Artificial Neural Network
KNN:K-Nearest Neighbours
NB:Naive Bayes
SVM:Support Vector Machine
XGB:Extreme Gradient Boosting
Principle Component Analysis
CV: Cross Validation

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

Table 1: Abbreviations

CNN	Convolutional Neural network
ReLU	Rectified Linear Unit
Ex	Example
RGB	Red Green Blue

NOTATION

English Symbols

R_E	Radius of the earth
R_u	Universal Gas Constant

CHAPTER 1

INTRODUCTION

1.1 Introduction

1.2 Need of project

cardiovascular diseases contributed 28·1% (95% UI 26·5–29·1) of the total deaths and 14·1% (12·9–15·3) of the total DALYs in India in 2016, compared with 15·2% (13·7–16·2) and 6·9% (6·3–7·4), respectively, in 1990. In 2016, there was a nine-times difference between states in the DALY rate for ischaemic heart disease, a six-times difference for stroke, and a four-times difference for rheumatic heart disease. 23·8 million (95% UI 22·6–25·0) prevalent cases of ischaemic heart disease were estimated in India in 2016, and 6·5 million (6·3–6·8) prevalent cases of stroke, a 2·3 times increase in both disorders from 1990. The age-standardized prevalence of both ischaemic heart disease and stroke increased in all ETL state groups between 1990 and 2016, whereas that of rheumatic heart disease decreased; the increase for ischaemic heart disease was highest in the low ETL state group. 53·4% (95% UI 52·6–54·6) of crude deaths due to cardiovascular diseases in India in 2016 were among people younger than 70 years, with a higher proportion in the low ETL state group. The leading overlapping risk factors for cardiovascular diseases in 2016 included dietary risks (56·4% [95% CI 48·5–63·9]

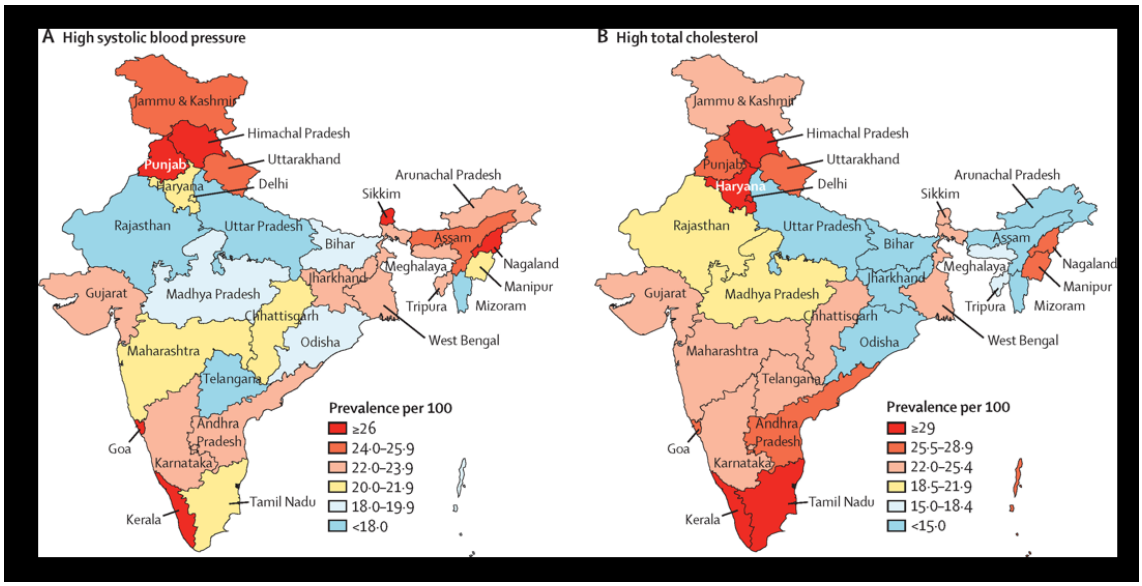

Fig. 1.1: heart disease prediction

of cardiovascular disease DALYs), high systolic blood pressure (54·6% [49·0–59·8]), air pollution (31·1% [29·0–33·4]), high total cholesterol (29·4% [24·3–34·8]), tobacco use (18·9% [16·6–21·3]), high fasting plasma glucose (16·7% [11·4–23·5]), and high body-mass index (14·7% [8·3–22·0]). The prevalence of high systolic blood pressure, high total cholesterol, and high fasting plasma glucose increased generally across all ETL state groups from 1990 to 2016, but this increase was variable across the states; the prevalence of smoking decreased during this period in all ETL state groups. [1]

1.3 Objective of Project

In the rural area where the diagnosis is easily and cheaply accessible providing a little bit of a hint about their heart's health at their fingertips may be save countless lives.

- Exploring the Dataset
- Data Cleaning
- Building Model and Performance Evaluation

The above-mentioned objectives in the section 1.3 is accomplished by the architecture diagram given in the figure 1.2.

From the architecture diagram, the process to be followed is clear. Still, let me discuss briefly first

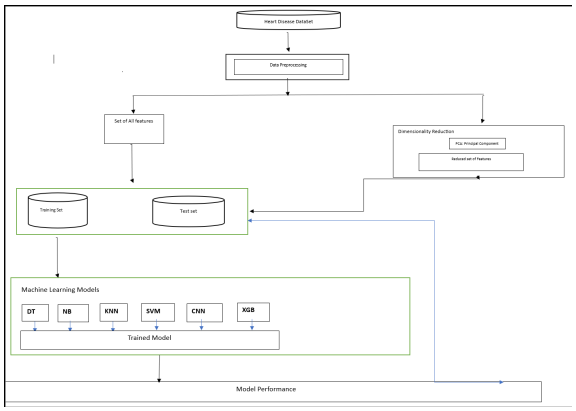

Fig. 1.2: Architecture Diagram

1.3.1 Understanding Model Architecture

- DATASET: Acquiring data is the most important part of any research or project. Since I have very less time while conducting this research so I used a dataset from the website Kaggle.[2]

- Data cleaning: Data cleaning is a process of handling all the possible data points which are irrelevant or not present.

- Feature Selection: A dataset has different attributes or features and each feature has an impact on the outcome of the result. There are several ways to select features of a dataset but I will discuss two of them here
  i)Collinearity: This is often calculated as the correlation between different features of a dataset, this may be done with the help of software.

  ii)Dimensionality Reduction: In higher dimension, there is a term called the curse of dimensionality, as we increase the number of attributes the complexity of the algorithm start increasing. So imagine the higher dimensional data in lower dimension by using mathematical tools like projection is termed dimensionality reduction. Examples include PCA(Principle component Analysis), SVD(singular value decomposition),LDA(Linear Discriminant Analysis) etc.

- Preparation of Data for Model Training: When done with the above-mentioned steps we come to the model training stage. First, we divide the data into two datasets as training and test data. Usually, training data is chosen from 80% of the complete dataset.

- Model Selection: After preparation of the data we come to select a model for training our dataset, here we are using models Decision Tree Classifier, Naive Bayes Classifier, KNN, Support Vector Machine, CNN, and XGB.

- Performance Evaluation: When done with the model we evaluated some important metrics like accuracy score, confusion matrix, precision, and recall.

CHAPTER 2

Literature Review

The literature available on the project idea is found to be very less. There are several authors have done projects in this subject but their idea is different there are several articles which proposed different approach solution to the implementation of the algorithm.

2.1 Literature Description

In this journal, Authors, B. Jin et al. proposed a network architecture based on moving data, the data obtained here is directly from EHG, the information is in the form of a signal, there is a time series type of object is formed and autoregression analysis is done. [3].

While in the abstract of cited article[4].Day by day the cases of heart disease are increasing at a rapid rate and it's very important and concerning to predict any such diseases beforehand. This diagnosis is a difficult task i.e. it should be performed precisely and efficiently. The research paper mainly focuses on which patient is more likely to have a heart disease based on various medical attributes. We prepared a heart disease prediction system to predict whether the patient is likely to be diagnosed with heart

disease or not using the medical history of the patient. We used different algorithms of machine learning such as logistic regression and KNN to predict and classify the patient with heart disease. A quite Helpful approach was used to regulate how the model can be used to improve the accuracy of prediction of Heart attacks in any individual. The strength of the proposed model was quite satisfying and was able to predict evidence of having heart disease in a particular individual by using KNN and Logistic Regression which showed good accuracy in comparison to the previously used classifier such as naive Bayes etc. So a quite significant amount of pressure has been lifted off by using the given model in finding the probability of the classifier to correctly and accurately identify the heart disease. The Given heart disease prediction system enhances medical care and reduces the cost. This project gives us significant knowledge that can help us predict the patients with heart disease It is implemented in the.pynb format.

There are also some articles available that laid emphasis on ensemble learning, boosting methods such as Gradient Boosting and Extreme gradient boosting also had satisfactory results in terms of improving the machine learning model. There are several authors who claimed that the combination of the decision tree, naive Bayes, ANN, and SVM has an accuracy score of 98% which is quite interesting and a little less to believe. There are several papers that proposed solely the use of Deep learning algorithms only there are several tunning hyperparameters available for this. The use of XGB(Xtreme Gradient Boosting) is very rare. Although in many Kaggle competitions, XGB is the most preferable machine learning algorithm for heart disease prediction most authors believed in using ensemble learning and deep learning methods. One of the benefits of using Deep learning is that it may be customized and tuned in higher dimensions also. The computation in GPU makes the data like ECG signals, MRI scans, and other formats of data could be easily computed without much loss of the data. In other simple algorithms, it is very hard to compute data given in large volumes.

There are several authors who tried to use XGB algorithm for the computation of higher dimension data and obtained a very signficant improvement over other machine algorithms like decision tree classifier and KNN algorithms. The parallel computation of XGB gives computations much more faster than the algorithms available witout parallel compuations.

CHAPTER 3

Methodology

The complete methodology of this research is given in fig **??**. We will start describing each and every term here individually here.

3.1 Datasets

First of all, very good quality data whose integrity can not be defied must be chosen. There are several ways of acquiring data, a few of them are as follows

3.1.1 Census

Census is a survey of the entire population. There are certain questions that are asked by government agents who conduct the census. The whole data is processed by the statistics office of that particular country. In India, National Statistics Office (NSO) oversees these affairs, and surveys are conducted in every 10 years. But these data mainly involve social questions and health-related questions are not dealt also we are conducting research on a particular niche area, so acquiring data from the census will be not helpful

as well as useful.

3.1.2 Survey

When a particular kind of data is required then the survey is often designed and random sample surveys are done. Random Sample Surveys means they are representative of the whole population but as we know health is a very personal subject and it falls under the section of privacy so a survey is not also possible for acquiring datasets.

3.1.3 Records

Records are the best way to acquire data since every medical center keeps data on its patients, past disease history, symptoms, growth of the symptoms, and other medically relevant factors. So we can use records specifically medical records for acquiring datasets.

3.1.4 Online Open Sourced Data

There are several websites available that provide highly trustworthy and free data for research purposes. Examples CERN Open Machine Learning Datasets, UCI Machine Learning Repository, Data.gov, FBI Crime Data Explorer, etc.

3.2 Exploring Data Set

In this particular experiment, I used data set from the website Kaggle[2]. The dataset contains 918 instances of data with 12 features. All features are shown in the figures.

3.2.1 Statistical Summary of all features individually

A statistical summary is a report about the behavior of the individual attribute and the mutual relation between them.

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
count	918.000000	918	918	918.000000	918.000000	918.000000	918	918.000000	918	918.000000	918	918.000000
unique	NaN	2	4	NaN	NaN	NaN	3	NaN	2	NaN	3	NaN
top	NaN	M	ASY	NaN	NaN	NaN	Normal	NaN	N	NaN	Flat	NaN
freq	NaN	725	496	NaN	NaN	NaN	552	NaN	547	NaN	460	NaN
mean	53.510893	NaN	NaN	132.396514	198.799564	0.233115	NaN	136.809368	NaN	0.887364	NaN	0.553377
std	9.432617	NaN	NaN	18.514154	109.384145	0.423046	NaN	25.460334	NaN	1.066570	NaN	0.497414
min	28.000000	NaN	NaN	0.000000	0.000000	0.000000	NaN	60.000000	NaN	-2.600000	NaN	0.000000
25%	47.000000	NaN	NaN	120.000000	173.250000	0.000000	NaN	120.000000	NaN	0.000000	NaN	0.000000
50%	54.000000	NaN	NaN	130.000000	223.000000	0.000000	NaN	138.000000	NaN	0.600000	NaN	1.000000
75%	60.000000	NaN	NaN	140.000000	267.000000	0.000000	NaN	156.000000	NaN	1.500000	NaN	1.000000
max	77.000000	NaN	NaN	200.000000	603.000000	1.000000	NaN	202.000000	NaN	6.200000	NaN	1.000000

Fig. 3.1: Snippet of Dataset

3.2.2 Removing redundant Features

Since our data is primarily medical data so every detail is carefully filled in so we do not have to worry about the outlier or other detections because when any record for a person is prepared everything feature is taken care of.

In the medical domain, there is very less possibility of outliers, because the readings for let's say blood pressure is taken, the data can't turn into an outlier until the equipment measuring blood pressure becomes faulty or gives wrong results.

3.2.3 Outlier Detection Method

But after taking care of everything if there is some chance of having outliers then it is taken care of by the IQR method.
IQR=Inter Quartile Range
IQR=q3-q1
where q3= Third Quartile
q1= First Quartile
If we say the upper ceiling is something the tolerance level of the upper bound and the lower ceiling is something the tolerance level of the lower bound then
upper ceiling = 1.5*IQR+q3
lower ceiling =q1-1.5*IQR

3.2.4 Outliers by Visualization

Outliers can be detected by plotting individual boxplots for each feature and you will find out outlier points will be out from the IQR region in the box plot. In this example of heart disease prediction boxplots of some attributes are given by the figures. when done with outlier detection our data cleaning is complete with this cleansed data we sent it to the next level.

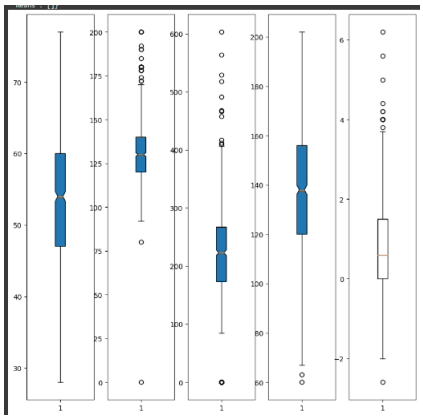

Fig. 3.2: Caption

3.3 Feature Engineering

Feature engineering is a machine learning technique that transforms available datasets into sets of figures essential for a specific task which involves the following processes

- Performing data analysis and correcting inconsistencies (like incomplete, incorrect data or anomalies).
- Deleting variables that do not influence model behavior.
- Dismissing duplicates and correlating records and, sometimes, carrying out data normalization.

3.3.1 Feature Engineering Steps involved

- Data Preparation
  To start the feature engineering process, you first need to convert raw data collected from various sources into a format that the ML model can use. For this, you perform data cleansing, fusion, ingestion, loading, and other operations. Now you're ready for feature engineering
- Exploratory data analysis
  It consists in performing descriptive statistics on datasets and creating visualizations to explore the nature of your data. Next, we should look for correlated variables and their properties in the dataset columns and clean them, if necessary.
- Feature improvement
  This step involves the modification of data records by adding missing values, transforming, normalizing, or scaling data, as well as adding dummy variables. We'll explain all these methods in detail in the next section.
- Feature construction
  You can construct features automatically and manually. In the first case, algorithms like PCA, tSNE, or MDS (linear and nonlinear) will be helpful. When it comes to manual feature construction, options are virtually endless. The choice of the method depends on the problem to be solved. One of the most well-known solutions

is convolution matrices. For example, they have been widely used to create new features while working on computer vision problems.

- Feature selection
  Feature selection, also known as variable selection or attribute selection, is a process of reducing the number of input variables (feature columns) by selecting the most important ones that correlate best with the variable you're trying to predict while eliminating unnecessary information.

  There are many techniques you can use for feature selection:

  filter-based, where you filter out the irrelevant features; wrapper-based, where you train ML models with different combinations of features; hybrid, which implements both of the techniques above. In the case of filter-based methods, statistical tests are used to determine the strength of the correlation of the feature with the target variable. The choice of the test depends on the data type of both input and output variables (i.e. whether they are categorical or numerical.). You can see the most popular tests in the table below.[5]

3.4 Model Selection

Machine Learning: Process of learning of a machine such that the learning is based on complete data. This is an intersection of Data Science and Artificial Intelligence. There are various primary fours types of machine learning algorithms
i) Supervised Machine Learning: In this type of machine learning algorithm we have to train our model according to the data present already. In supervised machine learning past results are already given one has to come up with a model that is more efficient. Examples of supervised machine learning algorithms are decision trees both regression and classification, Naive Bayes algorithm both regression and classification, KNN both regression and classification, CNN(Convolutional Neural Network), and boosting algorithms.
ii) Unsupervised Machine Learning: In this type of machine learning algorithm we have to train our model not to know about the 'labels' of the data. It does contain data points only and as a machine learning person, you must handle unsupervised data and try evaluating the model. Examples of unsupervised learning are K-mean clustering, etc.
iii)Reinforcement Learning: In this particular type of learning machine learns by reward and prize method when something has to be done by the machine right then it is rewarded and when the machine learns wrong then it imposes a penalty and learning happens. E.g. Self-driving cars.

iv) Semi-supervised Learning: It is a combination of supervised learning and unsupervised learning where some data are labeled while some of them are not we have to create an algorithm such that it learns from a supervised method and tries classifying data.

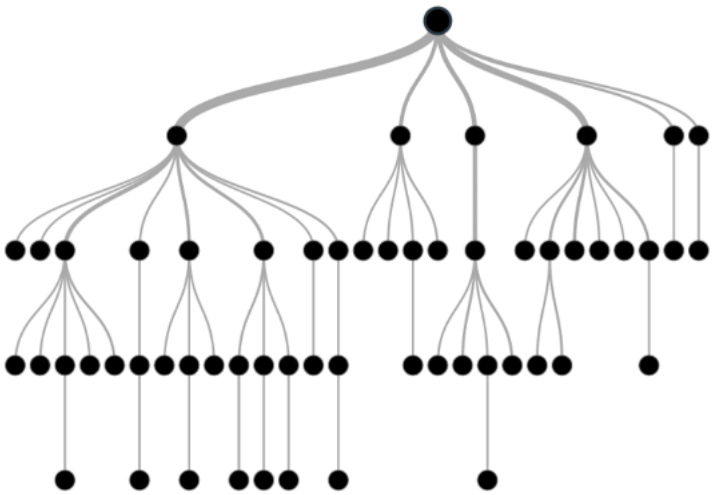

Fig. 3.3: Decision Tree

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

Fig. 3.4: Entropy

Supervised Learning
Here in this particular problem, we have a case of supervised learning, every data point is labeled and we try to build a model there are learning from the given data labeled dataset the following algorithms are used here for achieving this

3.4.1 Decision Tree Classifier

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

It is a tool that has applications spanning several different areas. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.

- **Root Node-**It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.

- **Decision Nodes –** the nodes we get after splitting the root nodes are called Decision Node.

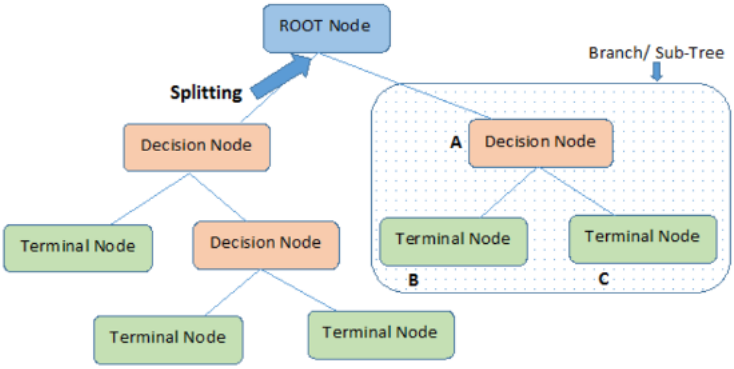

Fig. 3.5: Decision Tree

- **Leaf Nodes –** The nodes where further splitting is not possible are called leaf nodes or terminal nodes.

- **Sub-tree –**ust like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.

- **Pruning –** is nothing but cutting down some nodes to stop overfitting. [6]

3.4.2 Naive Bayes Classifier

Naive Bayes classifiers is a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Naive-Bayes Classifier works on the principle of Bayes theorem which can be formulated as follows

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = (P(B|A) * P(A))/P(B)$$

where A and B are events and $(P(B)! = 0$.

- Basically, we are trying to find the probability of event A, given that event B is true. Event B is also termed evidence.

- $P(A)$ is the priority of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- $P(A|B)$ is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regard to our dataset, we can apply Bayes' theorem in the following way:

$$P(y|X) = (P(X|y) * P(y))/P(X)$$

where y is the class variable and X is a dependent feature vector (of size n)
where:

$$X = (x_1, x_2, x_3, x_4, ...x_n)$$

Naive assumption
Now, it's time to put a naive assumption to Bayes' theorem, which is, independence among the features. So now, we split evidence into the independent parts.

Now, if any two events A and B are independent, then,

$$P(A, B) = P(A) * P(B)$$

hence we reach the result

$$P(y|(x_1, x_2, x_3, x_4, ...x_n) = (P(X_1|y)*P(X_2|y)*P(X_3|y)*P(X_4|y)*....*P(X_n|y)*P(y))/P(x_1)P(x_2)$$

which can be expressed as

$$P(y|(x_1, x_2, x_3, x_4, ...x_n) = (P(y) * \Pi[P(x_i|y))/\Pi(P(x_i))$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|(x_1, x_2, x_3, x_4, ...x_n)\alpha((P(y) * \Pi[P(x_i|y))$$

Now, we need to create a classifier model. For this, we find the probability of a given set of inputs for all possible values of the class variable y and pick up the output with

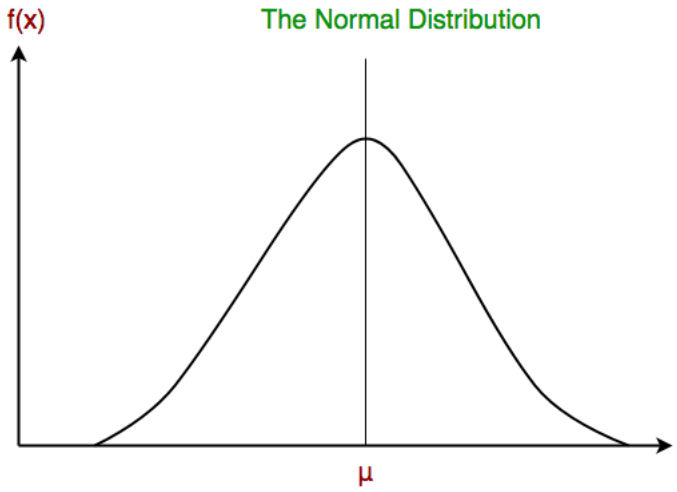

Fig. 3.6: Normal Distribution

maximum probability. This can be expressed mathematically as:

$$y = argmax_y P(y)\Pi(P(X_i|y)$$

So, finally, we are left with the task of calculating $P(y)$ and $P(x_i|y)$.

Please note that P(y) is also called class probability and $P(x_i|y)$ is called conditional probability.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$.
The method that we discussed above is applicable to discrete data. In the case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$.

Now, we discuss one such classifier here.

Gaussian Naive Bayes classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell-shaped curve that is symmetric about the mean of the feature values as shown in figure 3.6:

The updated table of prior probabilities for the outlook feature is as follows:

The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

$$P(x_i|y) = (1/\sqrt{2\pi\sigma_y^2}) \exp -(x_i - \mu_y)^2/(2\sigma_y^2)$$

Other Popular Naive Bayes Algorithms are

- **Multinomial Naive Bayes:** Feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. This is the event model typically used for document classification.
- **Bernoulli Naive Bayes:** In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used rather than term frequencies(i.e. frequency of a word in the document).[7]

3.4.3 KNN Classifier

Nearest neighbor classification, also known as K-nearest neighbors (KNN), is based on the idea that the nearest patterns to a target pattern x, for which we seek the label, deliver useful label information. KNN assigns the class label of the majority of the K-nearest patterns in data space. For this sake, we have to be able to define a similarity measure in data space. In R^q, it is reasonable to employ the Minkowski metric $p - norm$

$$||x_i - y_i||^p = (\Sigma|x_i - y_i|^p)^{\frac{1}{p}}$$

- For $p = 2$ This becomes a special case of the Minkowski metric, termed Euclidean metric, and often in algorithms Euclidean metrics are used as per default.

$$||x_i - y_i||^2 = (\Sigma|x_i - y_i|^2)^{\frac{1}{2}}$$

- For $p = 1$ This becomes another distance metric termed the Hamming distance metric which are also used frequently in heuristics.

$$||x_i - y_i|| = (\Sigma|x_i - y_i|)$$

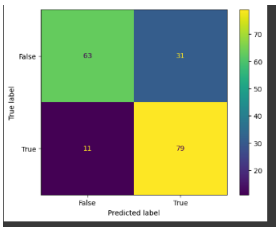

Fig. 3.7: confusion matrix

Algorithm for KNN classifier

Step-1: Select the number K of the neighbors.

Step-2: Calculate the Euclidean distance of K number of neighbors.

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

3.5 Performance Evaluation

When a model is created then well this model works on certain parameters to check the accuracy, precision, etc. we evaluate our model's performance there are certain terms discussed below mentioned

True Positive(TP): The total number of classifications that are actually true and also classified true by our model.

False Positive(FP): The total number of classifications that are actually false but classified as true in the model is termed false positive also called Type-2 error in statistical language.

True Negative(TN): The total number of classifications that are actually false and also classified false by our model is termed as true negative.

False Negative(FN): The total number of classifications that are actually false but classified true by our model is termed as false negative also referred to as type-1 error. In the medical domain for a test to be effective the type-1 error must be low.

Confusion Matrix: The matrix formed by the elements TP, FP, TN, and FN is termed a confusion matrix in multiclass classification the dimension of the matrix changes but in binary classification, it is $2 * 2$ matrix. Which may be seen in the figure below

3.5.1 Precision

Out of all positives, how well your model is predicting true positives.

$$Precision = TruePositive/(TruePositive + FalsePositive)$$

The value of precision always lies between 0 to 1.

3.5.2 Recall

Out of all true positives and false negatives how well your model is predicting true positives. [

$$Recall = TruePositive/(TruePositive + FalseNegative)$$

The value of recall also lies between 0 to 1.

3.5.3 F1-measure

F1-measure is defined as the harmonic mean of precision and recall.

$$F1 = (2 * precision * recall)/(precision + recall)$$

3.5.4 Accuracy Score

The accuracy score is defined as the actual true classifications out of all classifications.

$$AccuracyScore = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$$

3.6 Validation

There will be cases the model accuracy will be very high while training the model but while generalization the accuracy will drop significantly also when we most of the time train our model on a static dataset there may be instances that may behave differently and

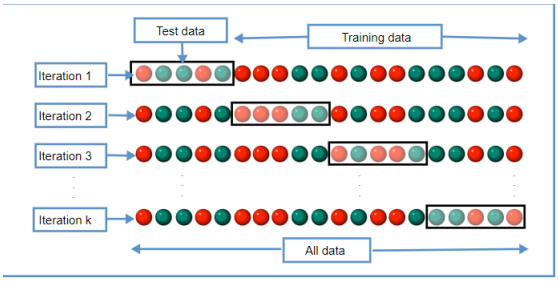

Fig. 3.8: cross validation

significantly drop the model's generalization to check whether the model is performing well in every segment of data we use **validation**. There are several methods to validate our model

3.6.1 Cross Validation

Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

3.6.2 Holdout Method

3.7 Hyperparameter Tuning

When model accuracy is very low or high and not generalizing well to the test set then there is a certain parameter that needs to be fixed in every model. For example, in a decision tree classifier, we can limit the depth of the tree, measure to split the tree also may be chosen there are various ways to split, in a KNN classifier the distance metric may be tuned according to the requirement.

3.8 Deployment

This is the last part of this research methodology when done with all of the above-mentioned steps the built model may be deployed to cloud servers like AWS, Microsoft Azure, etc.

3.9 Summary of chapter

This chapter highlighted the research methodology used for the heart disease prediction problem, and the complete lifecycle of the model is discussed one by one. The first dataset is understood statistically, its features are analyzed, there was outlier detection was done by the inter-quartile range method. When done with the feature engineering model the model is selected here the model I am using is Decision Tree Classifier, Naive Bayes Classifier, and KNN Classifier. When the model is created the performance of the model is evaluated using precision, recall, f1-measure, and accuracy score. Checked for overfitting by cross-validation, if the model is overfitting then hyperparameters of the model are tuned accordingly, and again performance calculated and when model is performing well on both training and test data then the model is deployed to cloud servers or SQL servers.

CHAPTER 4

Implementation Using Python

In this chapter, I am going to implement the project using Python programming language.

4.1 Implementation

The architecture for the implementation is given by figure 1.2

```
import pandas as pd
import numpy as np
from google.colab import drive
drive.mount('/content/drive')
#Reading file from drive
df=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/heart.csv")

df.describe(include="all") #Statistical Summary of the pandas data frame
df.isnull().sum()

#plotting the features individually to see their behaviour
```

```
fig, axs = plt.subplots(3,4, figsize=(10, 10), sharey=True)
axs[0][0].hist(df['Age'],color='g')
axs[0][1].hist(df['Sex'],color='b')
axs[0][2].hist(df['ChestPainType'],color='r')
axs[0][3].hist(df['RestingBP'],color='m')
axs[1][0].hist(df['Cholesterol'],color='pink')
axs[1][1].hist(df['FastingBS'],color='yellow')
axs[1][2].hist(df['RestingECG'],color='black')
axs[1][3].hist(df['MaxHR'],color='purple')
axs[2][0].hist(df['ExerciseAngina'],color='blue')
axs[2][1].hist(df['Oldpeak'])
axs[2][2].hist(df['ST_Slope'])
axs[2][3].hist(df['HeartDisease'])

sns.pairplot(df,hue=None,height=2.5)
from scipy import stats
# For outlier Detection
### Plotting of the variable has been done
fig,axs=plt.subplots(1,5,figsize=(10,10))
axs[0].boxplot(df['Age'],patch_artist=True,notch=True)
axs[1].boxplot(df['RestingBP'],patch_artist=True,notch=True)
axs[2].boxplot(df['Cholesterol'],patch_artist=True,notch=True)
axs[3].boxplot(df['MaxHR'],patch_artist=True,notch=True)
axs[4].boxplot(df['Oldpeak'])

#IQR implementation
age=list(df['Age'])
age.sort()
a1=len(age)/4
q1=age[int(a1)-1]
q1 #quantiles
q2=age[len(age)//2-1]
q2 #quantiles
q3=age[3*len(age)//4-1]
q3 #quantiles
q4=age[len(age)-1]
q4 #quantiles
IQR=q3-q1
IQR
upper_ceiling=1.5*IQR+q3
lower_ceiling=q1-1.5*IQR
tol_level=(upper_ceiling,lower_ceiling)
tol_level

# Accessing Dataframe to filter out the outliers
df_1=pd.DataFrame(age)
df_1.rename(columns={0:'Age'},inplace=True)
```

```
## Outliers for the data is filtered out
df_1=(df_1[df_1['Age']>27.5] & df_1[df_1['Age']<79.5])

resting_bp=list(df['RestingBP'])
resting_bp.sort()


resting_bp_=resting_bp[len(resting_bp)//4-1]
q2_resting_bp=resting_bp[len(resting_bp)//2-1]
q2_resting_bp

q3_resting_bp=resting_bp[3*len(resting_bp)//4-1]
q3_resting_bp

q4_resting_bp=resting_bp[len(resting_bp)-1]
q4_resting_bp

IQR_=q3_resting_bp-resting_bp_
IQR_
upper_ceiling_=1.5*IQR_+q3_resting_bp
lower_ceiling_=-(1.5*IQR_ -resting_bp_)
tol_level_=(upper_ceiling_,lower_ceiling_)
tol_level_
df2=pd.DataFrame(resting_bp)
df2=df2.rename(columns={0:'RestingBP'})

df.info()

li=list(df.columns)
lis_=[]
s=set()
for items in li:
  lis_.append(df[items])

  one_hot_encoded_data = pd.get_dummies(df, columns = ['Sex', 'ChestPainType','Rest
one_hot_encoded_data=one_hot_encoded_data.drop(columns=['Sex_F'],axis=1)
one_hot_encoded_data=one_hot_encoded_data.rename(columns={'Sex_M':'sex'})
df1=one_hot_encoded_data
Using Scaling for the given data
We can clearly observe the most of the data are having range from 0 to 1 but featur

'''There are two types of method by which data can be scaled
1.Normalization
2. Standardization
Normalization: Rescaling of data in range from 0 to 1.
There are various methods for the normalization process
```

```
MinMaxScaling:
Xsc=  X  XminXmaxXmin

Standardization:Making Gaussian Distributed data such that the mean is 0 and varian
x_std=(x-mu)/sigma'''

age=list(one_hot_encoded_data['Age'])

def mean(a):

  sum=0
  count=0
  for i in range(len(a)):
    sum=sum+a[i]
    count=count+1

  mean=sum/count
  return mean

def variance(a):
  sum=0
  count=0
  for i in range(len(a)):
   sum+=(a[i]-mean(a))**2
   count=count+1
  variance=sum/(count-1)
  return variance

def sqrt(x):
  return x**1/2
#Standard Scaler applied to Age
X_dy=(one_hot_encoded_data['Age']-mean(age))/sqrt(variance(age))
one_hot_encoded_data['Age']=X_dy
# Standard scaler to RestingBP

restingBP=list(one_hot_encoded_data['RestingBP'])
Y_dy=(one_hot_encoded_data['RestingBP']-mean(restingBP))/sqrt(variance(restingBP))
one_hot_encoded_data['RestingBP']=Y_dy

#Standard Scaling to Cholesterol data
cholesterol=list(one_hot_encoded_data['Cholesterol'])
Z_dy=(one_hot_encoded_data['Cholesterol']-mean(cholesterol))/sqrt(variance(choleste
one_hot_encoded_data['Cholesterol']=Z_dy

#Standard Scaling to MaxHR
maxhr=list(one_hot_encoded_data['MaxHR'])
ZZ=(one_hot_encoded_data['MaxHR']-mean(maxhr))/sqrt(variance(maxhr))
```

```
one_hot_encoded_data['MaxHR']=ZZ
#ZZ=(one_hot_encoded_data['MaxHR']-mean(maxhr))/sqrt(variance(maxhr))
#one_hot_encoded_data['Cholesterol']=ZZ

one_hot_encoded_data

sns.heatmap(one_hot_encoded_data)

from sklearn.model_selection import train_test_split
X_data=one_hot_encoded_data.drop('HeartDisease',axis=1)
Y_data=one_hot_encoded_data['HeartDisease']

X_train,X_test,y_train,y_test=train_test_split(X_data,Y_data,test_size=0.2,random_s

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

dt=DecisionTreeClassifier()
dt.fit(X_train,y_train)
arr=dt.predict(X_test)
arr=arr.reshape(-1,1)



array1=dt.predict(X_train)
accuracy_score(y_train,array1)*100

from sklearn.tree import plot_tree, export_text
text_representation = tree.export_text(dt)
print(text_representation)

plt.figure(figsize =(80,20))

plot_tree(dt, feature_names=X_train.columns, filled=True);

from matplotlib import pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn import metrics




print(accuracy_score(arr,y_test))
model1=confusion_matrix(arr,y_test)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = model1, display_labe
cm_display.plot()
```

```
plt.show()

#Finding Precision,recall and F-measure for the given model
tp=model1[1,1]
tn=model1[0,0]
fp=model1[0,1]
fn=model1[1,0]
precision=tp/(tp+fp)
recall=tp/(tp+fn)

F1=(1/precision)+(1/recall)
accuracy=tp+tn/(tp+tn+fp+fn)

print("The performance of the model is given by precision,recall,F1,accuracy",preci

from sklearn.model_selection import cross_val_score
score=cross_val_score(dt,X_data,Y_data,cv=96)
score2=cross_val_score(dt,X_data,Y_data,cv=97)
score3=cross_val_score(dt,X_data,Y_data,cv=98)
score4=cross_val_score(dt,X_data,Y_data,cv=99)
score5=cross_val_score(dt,X_data,Y_data,cv=100)
score=score*100
score2=score2*100
score3=score3*100
score4=score4*100
score5=score5*100
plt.plot(range(1,97),score)
plt.plot(range(1,98),score2)
plt.plot(range(1,99),score3)
plt.plot(range(1,100),score4)
plt.plot(range(1,101),score5)
plt.xlabel("Iteration")
plt.ylabel("Accuracy")
plt.legend(labels=['score','score2','score3','score4','score5'])

cv_iterations=[score,score2,score3,score4,score5]
for item in cv_iterations:
  print('The performance of the model at ',len(item),'iterations is given as ',mean
#without using Standard scaler let us check the performance of the data
df1
X1=df1.drop('HeartDisease',axis=1)
y1=df1['HeartDisease']

X1_train,X1_test,y1_train,y1_test=train_test_split(X1,y1,test_size=0.2,random_state
dt.fit(X1_train,y1_train)
array3=dt.predict(X1_train)
print(accuracy_score(array3,y1_train))
```

```
array2=dt.predict(X1_test)
accuracy_score(array2,y1_test)

score11=cross_val_score(dt,X1,y1,cv=96)
score12=cross_val_score(dt,X1,y1,cv=97)
score13=cross_val_score(dt,X1,y1,cv=98)
score14=cross_val_score(dt,X1,y1,cv=99)
score15=cross_val_score(dt,X1,y1,cv=100)
score11=score11*100
score12=score12*100
score13=score13*100
score14=score14*100
score15=score15*100
plt.plot(range(1,97),score11)
plt.plot(range(1,98),score12)
plt.plot(range(1,99),score13)
plt.plot(range(1,100),score14)
plt.plot(range(1,101),score15)
plt.xlabel("Iteration")
plt.ylabel("Accuracy")
plt.legend(labels=['score11','score12','score13','score14','score15'])

cv_iterations2=[score11,score12,score13,score14,score15]
for item in cv_iterations2:
  print('The performance of the model at ',len(item),'iterations is given as ',mean

'''  Tuning the parameter of the decision tree to check whether model performance i
There are so many parameters in Decision Tree Classifier lets Tune them and try to
One can se very clearly that the model performance on training data is 100 percent
The various parameter of decision tree classifier is given as

Max_depth: The depth of decision tree may be set accordingly to avoid overfitting
min samples split : the minimum number of samples a node must have before it can be
min samples leaf : the minimum number of samples a leaf node must have
max leaf nodes : maximum number of leaf nodes
max features : maximum number of features that are evaluated for splitting at each

### Tunning Hyperparameters of the decision tree
%matplotlib inline
var1=[]
for i in range(3,100):

  new_model=DecisionTreeClassifier(max_depth=i)
#### Creating model upto three depth
  new_model.fit(X1_train,y1_train)
  array_new=new_model.predict(X1_test)
  var1.append(accuracy_score(array_new,y1_test))
```

```
plt.figure()
plt.plot(range(1,len(var1)+1),var1,linestyle='dashed')
plt.ylabel("Model Accuracy")
plt.xlabel("Depth of the Tree")
plt.show()
'''Effect on Tunning the parameters
When Depth of the tree is increased, the model's performance is improved but after
#We may say three depth is performing very well
var2=[]
for i in range(3,100):
 new_model1=DecisionTreeClassifier(max_depth=3,min_samples_split=i)
 new_model1.fit(X1_train,y1_train)
 array33=new_model1.predict(X1_test)
 var2.append(accuracy_score(array33,y1_test)*100)

plt.figure(figsize=(5,5),facecolor='green',edgecolor='blue')
plt.plot(range(1,len(var2)+1),var2,'r+')
plt.xlabel("No of min sample splits",labelpad=1)
plt.ylabel("Accuracy Score")
plt.show()

#Tunning min sample leaf lets find out what happens
var3=[]
for i in range(3,1000):
  new_model2=DecisionTreeClassifier(max_depth=3,min_samples_split=3,min_samples_lea
  new_model2.fit(X1_train,y1_train)
  array333=new_model2.predict(X1_test)
  var3.append(accuracy_score(array333,y1_test)*100)
  print(i,'|',accuracy_score(array333,y1_test))

plt.figure(figsize=(5,5),facecolor='orange',edgecolor='blue')
plt.plot(range(1,len(var3)+1),var3,'r+')
plt.xlabel("No of min sample leaf",labelpad=1)
plt.ylabel("Accuracy Score")
plt.show()

var4=[]
for i in range(2,1000):
  new_model3=DecisionTreeClassifier(max_depth=3,min_samples_split=3,min_samples_lea
  new_model3.fit(X1_train,y1_train)
  array3333=new_model3.predict(X1_test)
  var4.append(accuracy_score(array3333,y1_test)*100)
  print(i,'|',accuracy_score(array3333,y1_test))

plt.figure(figsize=(5,5),facecolor='orange',edgecolor='blue')
plt.plot(range(1,len(var4)+1),var4,'r+')
plt.xlabel("No of max attribute",labelpad=1)
```

```
plt.ylabel("Accuracy Score")
plt.show()

#Lets check validation score for this tuned model
new_model5=DecisionTreeClassifier(max_depth=3,min_samples_split=3,min_samples_leaf=
new_model5.fit(X1_train,y1_train)
y_pred=new_model5.predict(X1_test)
print(accuracy_score(y_pred,y1_test))
score000=cross_val_score(new_model5,X1,y1,cv=5)
score001=cross_val_score(new_model5,X1,y1,cv=6)
score111=cross_val_score(new_model5,X1,y1,cv=96)
score112=cross_val_score(new_model5,X1,y1,cv=97)
score113=cross_val_score(new_model5,X1,y1,cv=98)
score114=cross_val_score(new_model5,X1,y1,cv=99)
score115=cross_val_score(new_model5,X1,y1,cv=100)

score00=score000*100
score001=score001*100
score111=score111*100
score112=score112*100
score113=score113*100
score114=score114*100
score115=score115*100
plt.figure()
plt.title("Fully Tunned Decision Tree performance vs Cross Validation")
plt.plot(range(1,97),score111)
plt.plot(range(1,98),score112)
plt.plot(range(1,99),score113)
plt.plot(range(1,100),score114)
plt.plot(range(1,101),score115)
plt.xlabel("Iteration")
plt.ylabel("Accuracy")
plt.legend(labels=['score111','score112','score113','score114','score115'])

cv_iterations22=[score000,score001,score111,score112,score113,score114,score115]
for item in cv_iterations22:
  print('The performance of the model at ',len(item),'iterations is given as ',mean


plt.figure(figsize =(80,20))

plot_tree(new_model5, feature_names=X1_train.columns, filled=True);

full_tuned_model=confusion_matrix(y1_test,y_pred)
cm_display_3 = metrics.ConfusionMatrixDisplay(confusion_matrix =full_tuned_model, d
cm_display_3.plot()
plt.show()
```

```
'''Let us start classifying using Naive Bayes Classifier
Naive Bayes Classifier
It is a very old algorithm for finding conditional probabilites based on class cond
P(A|B)=(P(B|A)*P(A))/P(B)

In generalized Notion
P(A| X  )=P( X  |A)*P(A)
In actuality it can be written based on conditional probabilites rules but in sciki

from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(X1_train,y1_train)

#Lets start predicting the values using our test set
array222=gnb.predict(X1_test)
accuracy_score(y1_test,array222)

score_NB=cross_val_score(gnb,X1,y1,cv=96)
score2_NB=cross_val_score(gnb,X1,y1,cv=97)
score3_NB=cross_val_score(gnb,X1,y1,cv=98)
score4_NB=cross_val_score(gnb,X1,y1,cv=99)
score5_NB=cross_val_score(gnb,X1,y1,cv=100)
score_NB=score_NB*100
score2_NB=score2_NB*100
score3_NB=score3_NB*100
score4_NB=score4_NB*100
score5_NB=score5_NB*100
plt.figure()
plt.title("Naive␣Bayes␣classifier␣vs␣Cross␣Validation")
plt.plot(range(1,97),score_NB)
plt.plot(range(1,98),score2_NB)
plt.plot(range(1,99),score3_NB)
plt.plot(range(1,100),score4_NB)
plt.plot(range(1,101),score5_NB)
plt.xlabel("Iteration")
plt.ylabel("Accuracy")
plt.legend(labels=['score_NB','score2_NB','score3_NB','score4_NB','score5_NB'])
plt.show()

cv_iterations2_NB=[score_NB,score2_NB,score3_NB,score4_NB,score5_NB]
for item in cv_iterations2_NB:
  print('The␣performance␣of␣the␣model␣at␣',len(item),'iterations␣is␣given␣as␣',mean

  '''KNN
KNN:K-Nearest Neighbour
This algorithm also called as lazy algorithm, first we have given an instance to cl
```

```
from sklearn.neighbors import KNeighborsClassifier
#First trying with default tuned model after evaluation of this model we will tune
our model accordingly
neigh=KNeighborsClassifier()
neigh.fit(X1_train,y1_train)
neigh.fit(X_train,y1_train)
array_neigh=neigh.predict(X1_test)
array_neigh1=neigh.predict(X_test)
array_neigh=np.array(array_neigh)
array_neigh11=array_neigh.reshape(-1,1)
array_neigh111=array_neigh1.reshape(-1,1)

y1_test=np.array(y1_test)
y1_test=y1_test.reshape(-1,1)
knn_=confusion_matrix(y1_test,array_neigh1)
knn__=confusion_matrix(y_test,array_neigh11)

knn_,knn__

accuracy_score(y1_test,array_neigh1),accuracy_score(y_test,array_neigh11)


cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = knn_, display_labels
cm_display_2=metrics.ConfusionMatrixDisplay(confusion_matrix=knn__,display_labels=[
cm_display.plot()
cm_display_2.plot()
plt.show()

'''Some Observations could be directly drawn here
The KNN model of dataset without standard scaler are having less Type-1 error
The KNN model of dataset with standard scaler are having large Type-1 error
The model accuracy can also be drawn in the similar way'''
#Lets check by the cross validation score and its plot
score_knn=cross_val_score(neigh,X1,y1,cv=96)
score2_knn=cross_val_score(neigh,X1,y1,cv=97)
score3_knn=cross_val_score(neigh,X1,y1,cv=98)
score4_knn=cross_val_score(neigh,X1,y1,cv=99)
score5_knn=cross_val_score(neigh,X1,y1,cv=100)
score_knn=score_knn*100
score2_knn=score2_knn*100
score3_knn=score3_knn*100
score4_knn=score4_knn*100
score5_knn=score5_knn*100
plt.figure()
plt.title("Naive␣Bayes␣classifier␣vs␣Cross␣Validation")
plt.plot(range(1,97),score_knn)
plt.plot(range(1,98),score2_knn)
```

```
plt.plot(range(1,99),score3_knn)
plt.plot(range(1,100),score4_knn)
plt.plot(range(1,101),score5_knn)
plt.xlabel("Iteration")
plt.ylabel("Accuracy")
plt.legend(labels=['score_knn','score2_knn','score3_knn','score4_knn','score5_knn']
plt.show()

cv_iterations2_knn=[score_knn,score2_knn,score3_knn,score4_knn,score5_knn]
for item in cv_iterations2_knn:
  print('The performance of the model at ',len(item),'iterations is given as ',mean

 ''' Start Tuning Parameter in the KNN algorithm
Parmeters in KNN algorithms are the followings:
Total Number of neighbors to be taken.
Distance metric: Usually the value of p is 2 but may be further tuned'''
new_list=list(one_hot_encoded_data.columns)
print(len(new_list))
val_knn=[]
for i in range(3,len(X1_train)-1,2):
  knn_model=KNeighborsClassifier(n_neighbors=i)
  knn_model.fit(X1_train,y1_train)
  arr_knn=knn_model.predict(X1_test)
  acc_knn=accuracy_score(y1_test,arr_knn)
  val_knn.append(acc_knn)
  print(i,'|',acc_knn)

  plt.figure(facecolor='red')
plt.title("The accuracy graph for different values K")
plt.plot(range(3,len(X1_train)-1,2),val_knn)
plt.xlabel("The values of K")
plt.ylabel("Model Accuracy")

val_knn1=[]
for i in range(1,len(new_list)-1):
  knn_model1=KNeighborsClassifier(n_neighbors=55,p=i)
  knn_model1.fit(X1_train,y1_train)
  arr_knn1=knn_model1.predict(X1_test)
  acc_knn1=accuracy_score(y1_test,arr_knn1)
  val_knn1.append(acc_knn1)
  print(i,'|',acc_knn1)

  plt.figure(facecolor='red')
plt.title("The accuracy graph for different values p")
plt.plot(range(1,len(new_list)-1),val_knn1)
plt.xlabel("The values of p")
plt.ylabel("Model Accuracy")
```

Table 4.1: Specifications of Software

Language	Python
Version	Latest
Libraries	Scikit-learn, Pandas, Numpy, Seaborn and Matplotlib

```
#From above notion it can be figured out that model is best for p-value of 2 i.e fo
#Performing cross validation and its scores.
score_knn_=cross_val_score(knn_model1,X1,y1,cv=96)
score2_knn_=cross_val_score(knn_model1,X1,y1,cv=97)
score3_knn_=cross_val_score(knn_model1,X1,y1,cv=98)
score4_knn_=cross_val_score(knn_model1,X1,y1,cv=99)
score5_knn_=cross_val_score(knn_model1,X1,y1,cv=100)
score_knn_=score_knn_*100
score2_knn_=score2_knn_*100
score3_knn_=score3_knn_*100
score4_knn_=score4_knn_*100
score5_knn_=score5_knn_*100
plt.figure()
plt.title("KNN␣classifier␣vs␣Cross␣Validation")
plt.plot(range(1,97),score_knn_)
plt.plot(range(1,98),score2_knn_)
plt.plot(range(1,99),score3_knn_)
plt.plot(range(1,100),score4_knn_)
plt.plot(range(1,101),score5_knn_)
plt.xlabel("Iteration")
plt.ylabel("Accuracy")
plt.legend(labels=['score_knn_','score2_knn_','score3_knn_','score4_knn_','score5_k
plt.show()

cv_iterations2_knn_=[score_knn_,score2_knn_,score3_knn_,score4_knn_,score5_knn_]
for item in cv_iterations2_knn_:
  print('The␣performance␣of␣the␣model␣at␣',len(item),'iterations␣is␣given␣as␣',mean
```

4.2 Summary

The complete implementation is done in googlecolab notebook. Firstly dataset is loaded into colab notebook then a statistical summary is obtained, detection of outliers has been done there using boxplots after that categorical data is encoded using pandas. Models are applied checked and validated and again accordingly parameters are tuned until a optimized model performance is not obtained.

CHAPTER 5

Experimental Results

5.1 Statistical Summary of the dataset

5.2 Evaluation of the model for Decision Tree

5.2.1 Decision Tree with default parameters

Different performance measures of the model are given in the figure :

5.2.2 Cross Validation Score

Cross-validation scores are given in the figure as follows:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
count	918.000000	918	918	918.000000	918.000000	918.000000	918	918.000000	918	918.000000	918	918.000000
unique	NaN	2	4	NaN	NaN	NaN	3	NaN	2	NaN	3	NaN
top	NaN	M	ASY	NaN	NaN	NaN	Normal	NaN	N	NaN	Flat	NaN
freq	NaN	725	496	NaN	NaN	NaN	552	NaN	547	NaN	460	NaN
mean	53.510893	NaN	NaN	132.396514	198.799564	0.233115	NaN	136.809368	NaN	0.887364	NaN	0.553377
std	9.432617	NaN	NaN	18.514154	109.384145	0.423046	NaN	25.460334	NaN	1.066570	NaN	0.497414
min	28.000000	NaN	NaN	0.000000	0.000000	0.000000	NaN	60.000000	NaN	-2.600000	NaN	0.000000
25%	47.000000	NaN	NaN	120.000000	173.250000	0.000000	NaN	120.000000	NaN	0.000000	NaN	0.000000
50%	54.000000	NaN	NaN	130.000000	223.000000	0.000000	NaN	138.000000	NaN	0.600000	NaN	1.000000
75%	60.000000	NaN	NaN	140.000000	267.000000	0.000000	NaN	156.000000	NaN	1.500000	NaN	1.000000
max	77.000000	NaN	NaN	200.000000	603.000000	1.000000	NaN	202.000000	NaN	6.200000	NaN	1.000000

Fig. 5.1: Summary

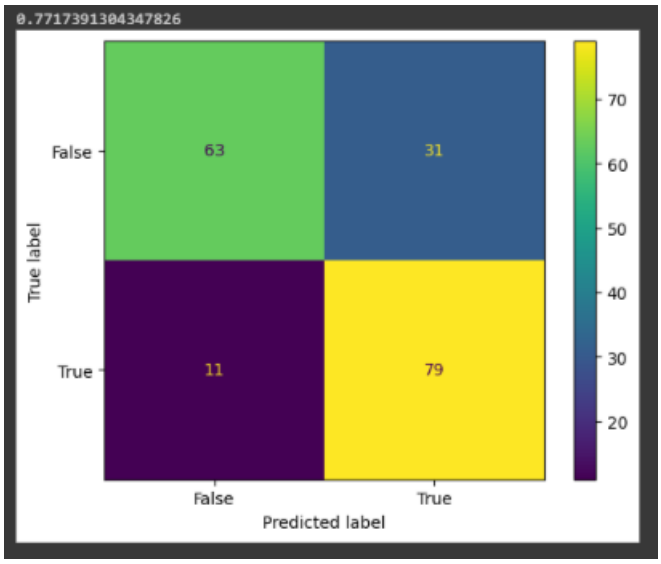

Fig. 5.2: confusion matrix decision tree

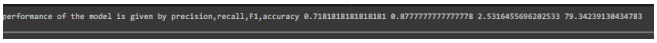

Fig. 5.3: scores

5.2.3 After Parameter Tuning scores of the decision tree

5.3 Naive Bayes Classifier

5.4 KNN Classifier

5.4.1 Before parameter tunning

5.4.2 After Tuning Accuracy

Fig. 5.4: Cross Validation Score

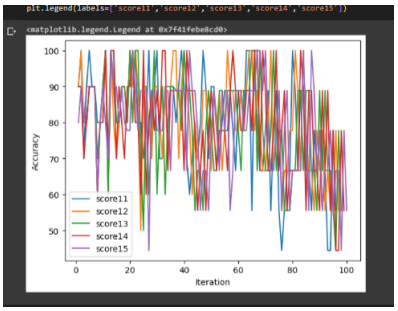

Fig. 5.5: cross val performance after parameter tuning

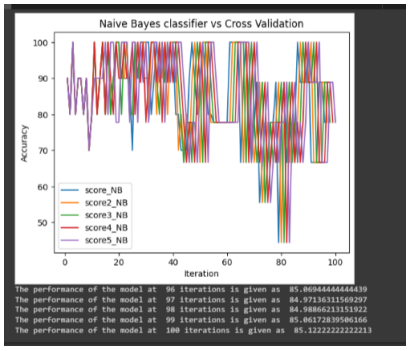

Fig. 5.6: Performance naive bayes

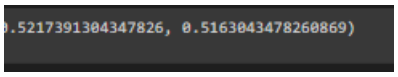

Fig. 5.7: before tuning accuracy

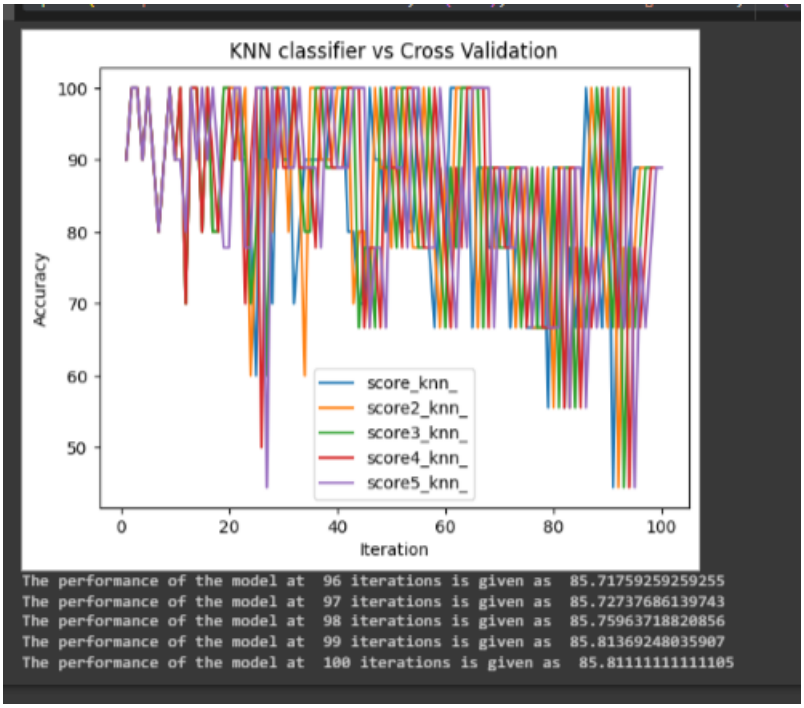

Fig. 5.8: after tuning parameter

CHAPTER 6

Conclusion and Future Scope

6.1 Conclusion

As it is clear from the results all the simple model's accuracy is varying from 75-85% and after tunning two parameters in KNN algorithm gave us an accuracy of 85% percent which is the highest amongst all the models. But these models are very simple and in higher dimensions not perform up to the mark.

The expected attributes leading to heart disease in patients are available in the dataset which contains 76 features and 14 important features that are useful to evaluate the system are selected among them. If all the features are taken into consideration then the efficiency of the system the author gets is less. To increase efficiency, attribute selection is done. In this n features have to be selected for evaluating the model which gives more accuracy. The correlation of some features in the dataset is almost equal and so they are removed. If all the attributes present in the dataset are taken into account then the efficiency decreases considerably. All the seven machine learning methods' accuracies are compared based on which one prediction model is generated. Hence, the aim is to use various evaluation metrics like confusion matrix, accuracy, precision, recall, and f1-score which predicts the

disease efficiently. Comparing all three the KNN classifier gives the highest accuracy of 84

6.2 Future Scope

There is always a chance for improvement in these technologies as the world is changing so does we so does our habits may whatever parameters are taken into consideration, for now, will not be even counted for the next generation. The sophisticated algorithms may change the scope we are looking into things. As the coming of CHATGPT already impacted us so the accuracy of the models may be improved further using other algorithms and their ensembles. So at best there are a range of possibilities.

REFERENCES

[1] D. Prabhakaran, P. Jeemon, M. Sharma, G. A. Roth, C. Johnson, S. Harikrishnan, R. Gupta, J. D. Pandian, N. Naik, A. Roy, *et al.*, "The changing patterns of cardiovascular diseases and their risk factors in the states of india: the global burden of disease study 1990–2016," *The Lancet Global Health*, vol. 6, no. 12, pp. e1339–e1351, 2018.

[2] O. sourced, "The heart disease dataset used here." `https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/`.

[3] B. Jin, C. Che, Z. Liu, S. Zhang, X. Yin, and X. Wei, "Predicting the risk of heart failure with ehr sequential data modeling," *IEEE Access*, vol. 6, pp. 9256–9261, 2018.

[4] H. Jindal, S. Agrawal, R. Khera, R. Jain, and P. Nagrath, "Heart disease prediction using machine learning algorithms," *IOP Conference Series: Materials Science and Engineering*, vol. 1022, p. 012072, jan 2021.

[5] "Feature engineering article." `https://serokell.io/blog/feature-engineering-for-machine-learning`.

[6] "Decision tree." `https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/`.

[7] N. Kumar, "Naive bayes classifier." `https://www.geeksforgeeks.org/naive-bayes-classifiers/`.