A project report on

Heart Disease Prediction Using Machine Learning

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology (IT)

Under the supervision of

Er. Varun Goel
Assistant professor

Submitted by

Nishika Mittal
01014803119

DEPARTMENT OF INFORMATION TECHNOLOGY MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY SECTOR-22, ROHINI, DELHI –110086

December 2022

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide Er. **Varun Goel** Assistant Professor, Department of Information Technology, MAIT, for his guidance with unsurpassed knowledge and immense encouragement. We are grateful to

DR. M.L. SHARMA, Head of the Department, Information Technology, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, MAIT, DELHI,** for their encouragement and cooperation to carry out this work.

We express our thanks to Project Coordinator **Dr. Bhoomi Gupta**, for her continuous support and encouragement. We thank all **teaching faculty** of Department of IT, whose suggestions during reviews helped us in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this project report titled, "HEART DISEASE PREDICTION USING MACHINE LEARNING" submitted by me in the partial fulfillment of the requirement of the award of the degree of Bachelor of Technology (B.Tech.) Submitted in the Department of Information Technology, Maharaja Agrasen Institute of Technology is an authentic record of my project work carried out under the guidance of **Er. Varun Goel**, Assistant Professor.

The matter presented in this project report has not been submitted either in part or full to any university or Institute for award of any degree.

Date: December 12, 2022 Nishika Mittal

Place: Delhi Roll No.:01014803119

SUPERVISOR'S CERTIFICATE

It is to certify that the Project entitled HEART DISEASE PREDICTION USING MACHINE LEARNING " which is being submitted by Ms. Nishika Mittal to the Maharaja Agrasen Institute of Technology, Rohini in the fulfillment of the requirement for the award of the degree of Bachelor of Technology (B.Tech.), is a record of bonafide project work carried out by her under my guidance and supervision.

Dr. VARUN GOEL
Assistant Professor
Department of Information Technology

Prof. (Dr.) M.L. Sharma
H.O.D.
Department of Information Technology

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Cardiovascular diseases are the most common cause of death worldwide over the last few decades in developed as well as underdeveloped and developing countries. Early detection of cardiac diseases and continuous supervision of clinicians can reduce the mortality rate. 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. In this project, we have developed and researched models for heart disease prediction through the various heart attributes of patients and detect impending heart disease using Machine learning techniques like Logistic Regression, K-Nearest neighbors Classifier, Random Forest Classifier and AdaBoost Classifier on the dataset available publicly in Kaggle Website, further evaluating the results using confusion matrix and cross-validation. The early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes in high risk patients and in turn reduce complications, which can be a great milestone in the field of medicine.

***Keywords**: Machine Learning, Logistic regression, Cross-Validation, K-nearest neighbors*

CHAPTER 1: INTRODUCTION

*According to the World Health Organization, every year **17.9 million deaths** occur worldwide due to Heart Disease. The load of cardiovascular diseases is rapidly increasing all over the world in the past few years. Various unhealthy activities are the reason for the increase in the risk of heart disease like high cholesterol, obesity, increase in triglycerides levels, hypertension, etc. Many types of research have been conducted in an attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk. Heart Disease is even highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn, reduces complications.*

We can use different machine models to diagnose the disease and classify or predict the results. A complete genomic data analysis can easily be done using machine learning models. Models can be trained for knowledge pandemic predictions and also medical records can be transformed and analyzed more deeply for better predictions. This project aims to predict future Heart Disease by analyzing data of patients which classifies whether they have heart disease or not using machine-learning algorithms. The input to our algorithm is 14 features with number values. We use several algorithms such as Logistic Regressions, Random Forest, K-nearest neighbors to output a binary number 1 or 0. 1 indicates the patient has heart disease and vice versa.

1.1 Problem Definition

Heart disease is very fatal and it should not be taken lightly. Heart disease happens more in males than females, which can be read further from Harvard Health Publishing. Researchers found that, throughout life, men were about twice as likely as women to have a heart attack. That higher risk persisted even after they accounted for traditional risk factors of heart disease, including high cholesterol, high blood pressure, diabetes, body mass index, and physical activity. The major challenge in heart disease is its detection. There are instruments available that can predict heart disease but either they are expensive or are not efficient to calculate the chance of heart disease in humans. Early detection of cardiac diseases can decrease the mortality rate and overall 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 analyze the data for hidden patterns. The hidden patterns can be used for health diagnosis in medicinal data.

1.2 Scope

Machine learning techniques have been around us and have been compared and used for analysis for many kinds of data science applications. The major motivation behind this research-based project was to explore the feature selection methods, data preparation, and processing behind the training models in machine learning. With first-hand models and libraries, the challenge we face today is data were beside their abundance, and in our cooked models, the accuracy we see during training, testing, and actual validation has a higher variance. Hence this project is carried out with the motivation to explore behind the models, and further implement Logistic Regression model to train the obtained data. Furthermore, as the whole machine learning is motivated to develop an appropriate computer-based system and decision support that can aid in the early detection of heart disease, in this project we have developed a model which classifies if a patient will have heart disease in ten years or not based on various features (i.e. potential risk factors that can cause heart disease) using logistic regression. Hence, the early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes in high-risk patients and in turn reduce complications, which can be a great milestone in the field of medicine.

1.3 Objectives

The main objectives of developing this project are:

- To develop a machine learning model to predict the future possibility of heart disease by implementing Logistic Regression.
- To determine significant risk factors based on a medical dataset which may lead to heart disease.
- To analyze feature selection methods and understand their working principle.

CHAPTER 2: LITERATURE REVIEW

With growing development in the field of medical science alongside machine learning various experiments and researches has been carried out in these recent years releasing the relevant significant papers.

[1] Senthilkumar Mohan, Chandrasegar Thiurumalai, and Gautam Srivastava "**Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques"**: The prediction model is introduced with different combinations of features and several known classification techniques. It produced an enhanced performance level with an accuracy level of 88.7% through the prediction model for heart disease with the hybrid random forest with a linear model (HRFLM).

[2] Rohit Bharti, Aditya Khamparia, Mohammad Shabaz, Gaurav Dhiman, Sagar Pande, and Parneet Singh **"Prediction of Heart Disease Using a Combination of Machine Learning and Deep Learning"** :The dataset consists of 14 main attributes used for performing the analysis. Various promising results are achieved and are validated using accuracy and confusion matrix. Using deep learning approach, 88.2% accuracy was obtained. It was also found out that the statistical analysis is also important when a dataset is analyzed and it should have a Gaussian distribution, and then the outlier's detection is also important and a technique known as Isolation Forest is used for handling this. The difficulty which came here is that the sample size of the dataset is not large. If a large dataset is present, the results can increase very much in deep learning and ML as well. The algorithm applied by us in ANN architecture increased the accuracy which we compared with the different researchers.

[3] Harshit Jindal "**Heart disease prediction using machine learning algorithms**": A quite helpful approach was used to regulate how the model can be used to improve the accuracy of prediction of Heart Attack in any individual. The strength of the proposed model was quiet satisfying and was able to predict evidence of having a heart disease in a particular individual by using KNN and Logistic Regression which showed a good accuracy in comparison to the previously used classifier such as naive bays etc. Most efficient of these algorithms is KNN which gives us the accuracy of 88.52%. And, finally we classify patients that are at risk of getting a heart disease or not and also this method is totally cost efficient.

[4] Lakshmana Rao et al, proposed "**Machine Learning Techniques for Heart Disease Prediction"** in which the contributing elements for heart disease are more. So, it is difficult to distinguish heart disease. To find the seriousness of the heart disease among people different neural systems and data mining techniques are used.

[5] Using the similar dataset of Framingham, Massachusetts, the experiments were carried out using 4 models and were trained and tested with maximum accuracy K-Neighbors Classifier: 87%, Support Vector Classifier: 83%, Decision Tree Classifier: 79% ,Random Forest Classifier: 84%.

[6] Anjan N. Repaka et al, proposed a model stated the performance of prediction for two classification models, which is analyzed and compared to previous work. The experimental results show that accuracy is improved in finding the percentage of risk prediction of our proposed method in comparison with other models.

[7] Avinash Golande et al, proposed "**Heart Disease Prediction Using Effective Machine Learning Techniques**" in which few data mining techniques are used that support the doctors to differentiate the heart disease. Usually utilized methodologies are k-nearest neighbor, Decision tree and Naïve Bayes. Other unique characterization-based strategies utilized are packing calculation, Part thickness, consecutive negligible streamlining and neural systems, straight Kernel self-arranging guide and SVM (Support Vector Machine).

CHAPTER 3: IMPLEMENTATION OF PROPOSED SYSTEM

Heart disease is even being highlighted as a silent killer which leads to the death of a person without obvious symptoms. The nature of the disease is the cause of growing anxiety about the disease & its consequences. Hence continued efforts are being done to predict the possibility of this deadly disease in prior. So that various tools & techniques are regularly being experimented with to suit the present-day health needs. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can conclude. This technique can be very well adapted to the do the prediction of heart disease. As the well-known quote says "Prevention is better than cure", early prediction & its control can be helpful to prevent & decrease the death rates due to heart disease.

3.1 System Requirements

3.1.1 Hardware requirements:

Processer	:	Any Update Processer
Ram	:	Min 4GB
Hard Disk	:	100GB Free space

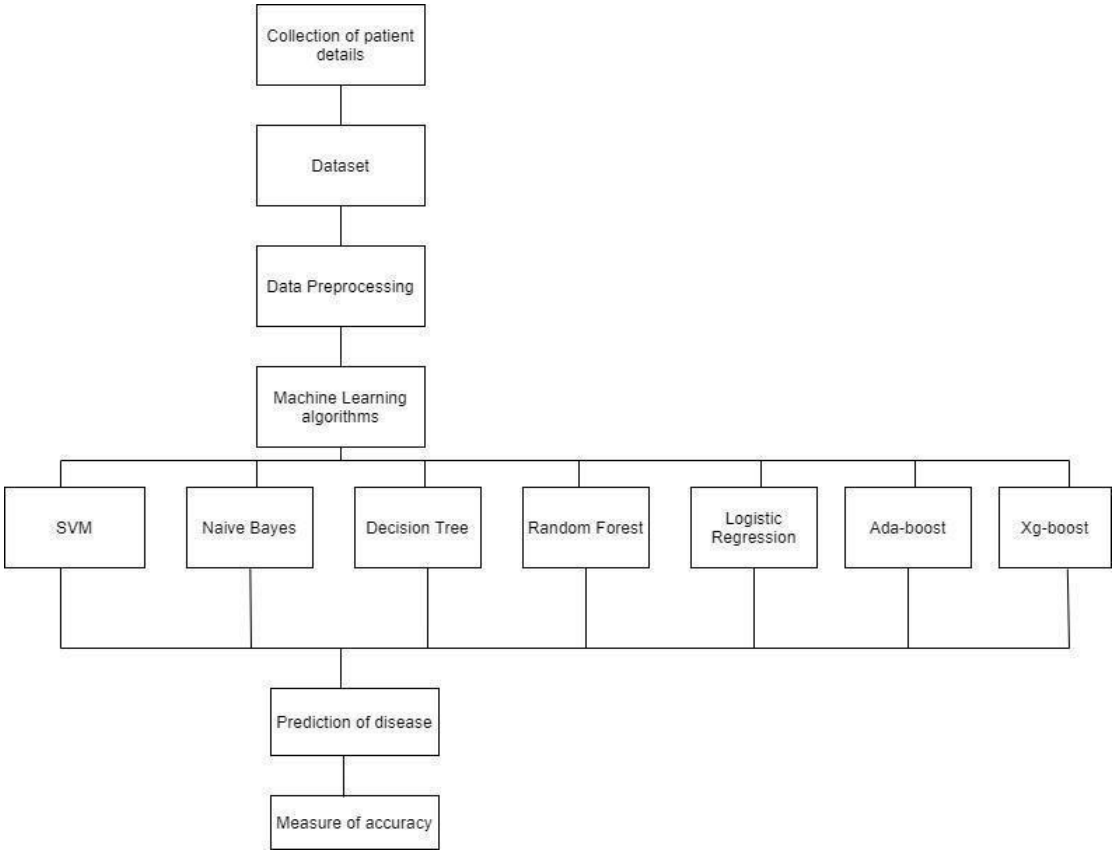
3.1.2 Software requirements:

Operating System	:	Windows Family
Technology	:	Python3.7
IDE	:	Jupyter notebook

3.2 Working System

Dataset collection is collecting data which contains patient details. Attributes selection process selects the useful attributes for the prediction of heart disease. After identifying the available data resources, they are further selected, cleaned, made into the desired form. Different classification techniques as stated will be applied on preprocessed data to predict the accuracy of heart disease. Accuracy measure compares the accuracy of different classifiers.

The working of this system is described as follows:



3.3 Proposed System

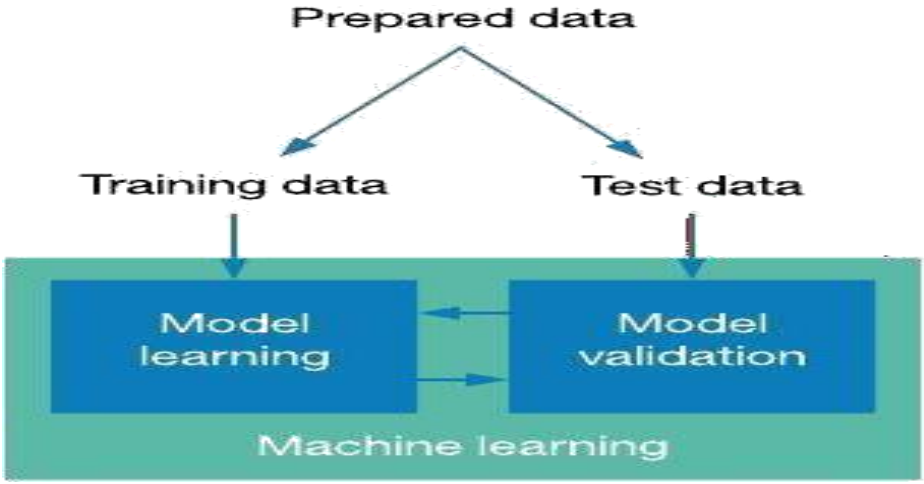
The working of the system starts with the collection of data and selecting the important attributes. Then the required data is preprocessed into the required format. The data is then divided into two parts training and testing data. The algorithms are applied and the model is trained using the training data. The accuracy of the system is obtained by testing the system using the testing data. This system is implemented using the following modules.

1.) Collection of Dataset
2.) Selection of attributes
3.) Data Pre-Processing
4.) Balancing of Data
5.) Disease Prediction

3.3.1 Collection of dataset

Initially, we collect a dataset for our heart disease prediction system. After the collection of the dataset, we split the dataset into training data and testing data. The

training dataset is used for prediction model learning and testing data is used for evaluating the prediction model. For this project, 70% of training data is used and 30% of data is used for testing. The dataset used for this project is Heart Disease UCI. The dataset consists of 76 attributes; out of which, 14 attributes are used for the system.



3.3.2 Selection of attributes

Attribute or Feature selection includes the selection of appropriate attributes for the prediction system. This is used to increase the efficiency of the system. Various attributes of the patient like gender, chest pain type, fasting blood pressure, serum cholesterol, etc. are selected for the prediction. The Correlation matrix is used for attribute selection for this model.

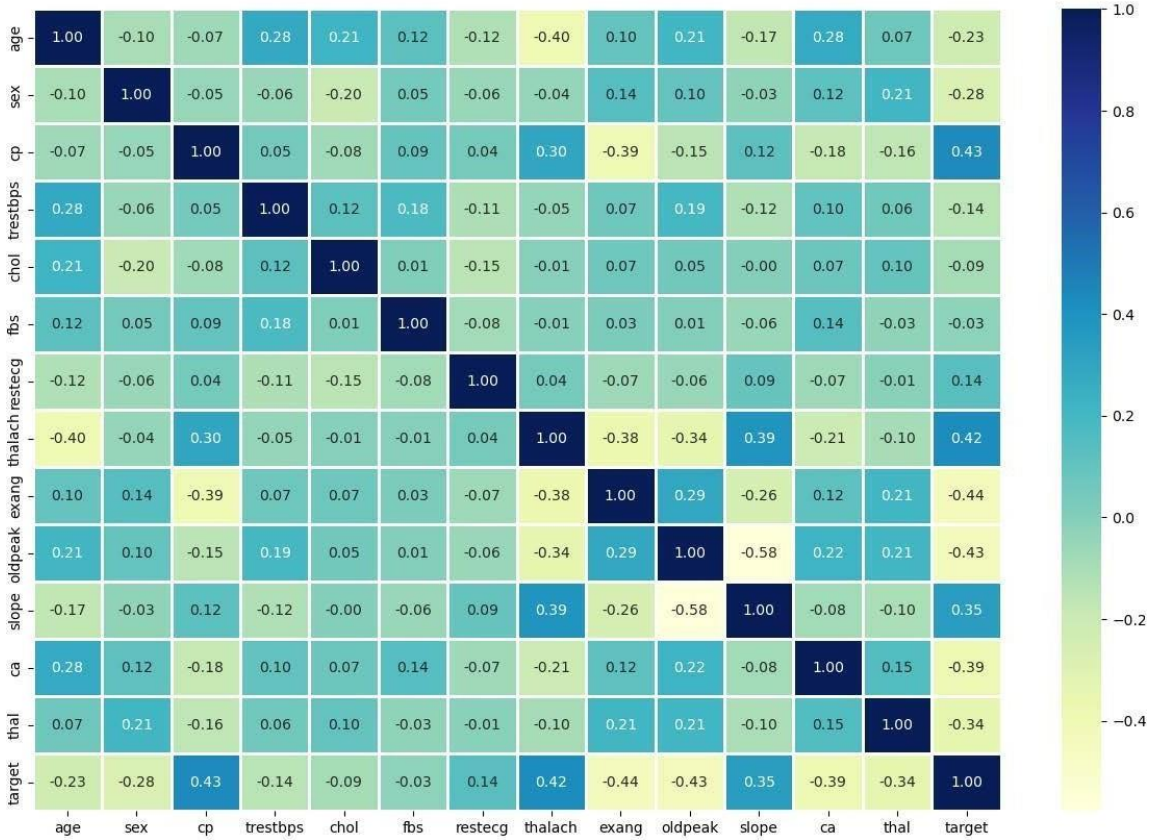


Figure: Correlation matrix

3.3.3 Pre-processing of Data

Data pre-processing is an important step for the creation of a machine learning model. Initially, data may not be clean or in the required format for the model which can cause misleading outcomes. In pre-processing of data, we transform data into our required format. It is used to deal with noises, duplicates, and missing values of the dataset. Data pre-processing has the activities like importing datasets, splitting datasets, attribute scaling, etc. Preprocessing of data is required for improving the

3.3.4 Balancing of Data

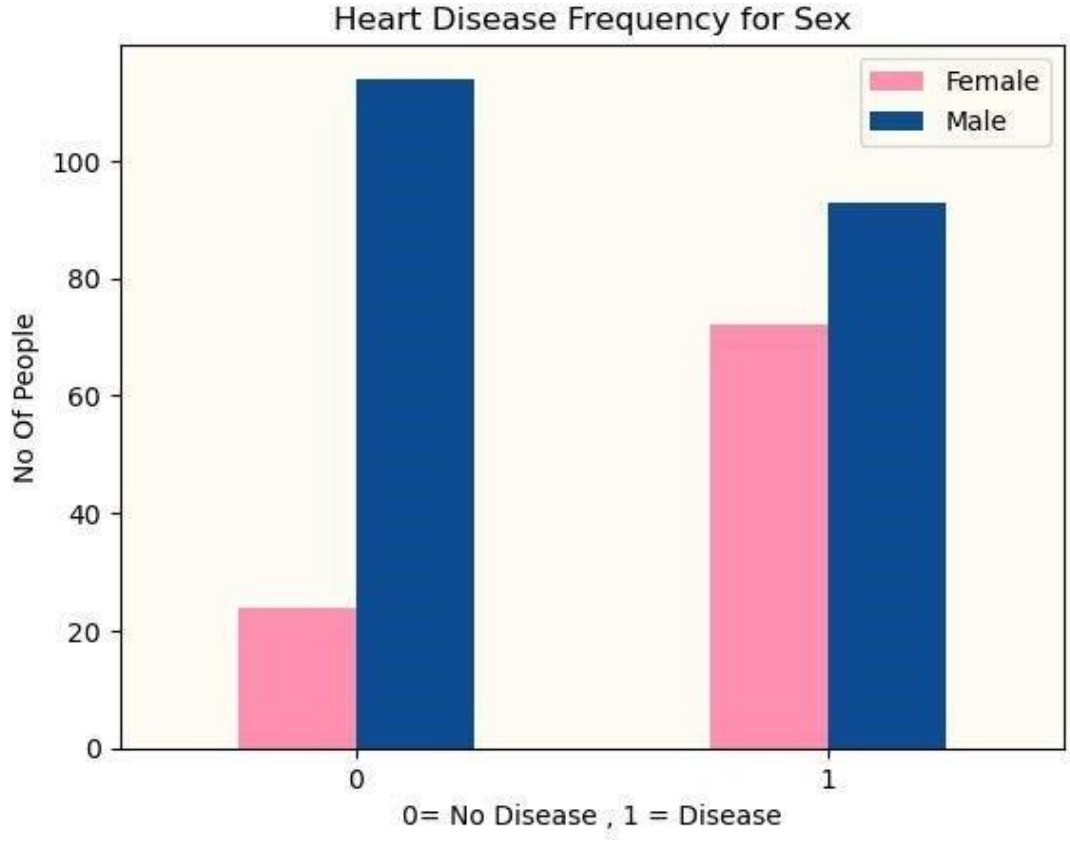
Imbalanced datasets can be balanced in two ways. They are Under Sampling and Over Sampling

(a) Under Sampling:

In Under Sampling, dataset balance is done by the reduction of the size of the ample class. This process is considered when the amount of data is adequate.

(b) Over Sampling:

In Over Sampling, dataset balance is done by increasing the size of the scarce samples. This process is considered when the amount of data is inadequate.



3.3.5 Prediction of Disease

Various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Tree, Logistic Regression, K-nearest neighbor (KNN) are used for classification. Comparative analysis is performed among algorithms and the algorithm that gives the highest accuracy is used for heart disease prediction.

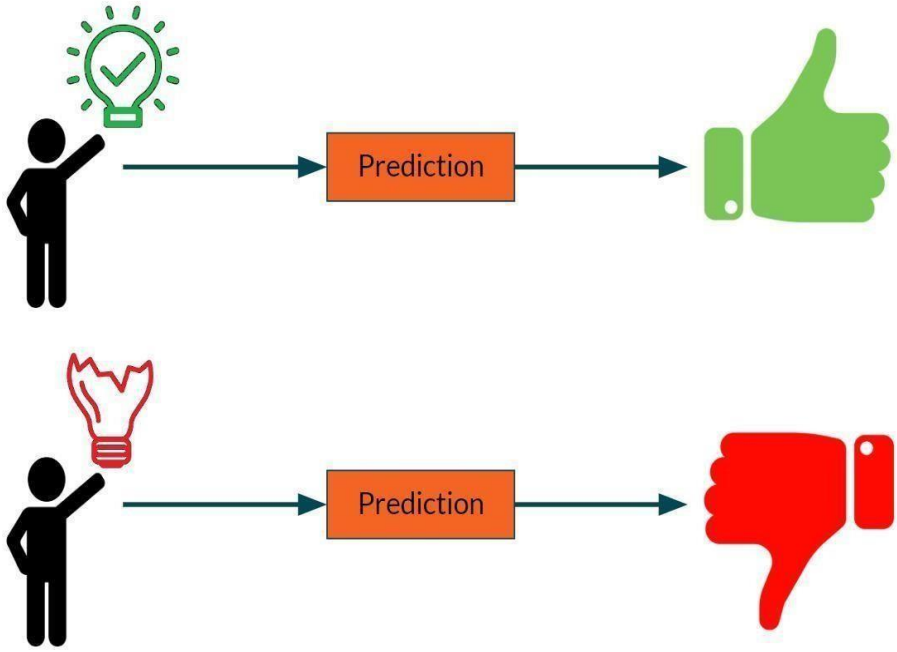


Figure: Prediction of Disease

CHAPTER 4: EXPERIMENTAL ANALYSIS

4.1 MACHINE LEARNING

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data.

- **Supervised Learning**

Supervised learning is the type of machine learning in which machines are trained using well "labeled" training data, and on the basis of that data, machines predict the output. The labeled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

- **Unsupervised learning**

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar to how a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

- **Reinforcement learning**

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

4.2 ALGORITHMS

4.2.1 SUPPORT VECTOR MACHINE (SVM):

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyper plane. SVM chooses the extreme points/vectors that help in creating the hyper plane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In the 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

The followings are important concepts in SVM -

Support Vectors - Data Points that are closest to the hyper plane are called support vectors. Separating line will be defined with the help of these data points.

Hyper plane - As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

Margin - It may be defined as the gap between two lines on the closest data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

Types of SVM:

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data,

which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N - the number of features) that distinctly classifies the data points.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where the number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.

SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

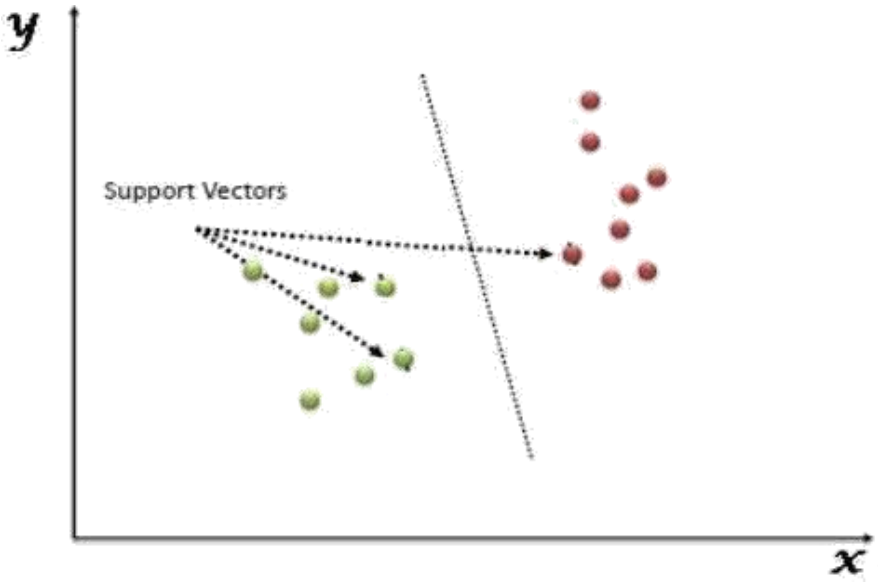


Figure: Support Vector Machine

4.2.2 NAIVE BAYES ALGORITHM:

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset.

Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

The Naive Bayes algorithm is comprised of two words Naive and Bayes, Which can be described as:

- **Naive:** It is called Naive because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the basis of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes:** It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Types of Naive Bayes model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian:** The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.

- **Multinomial:** The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.
- **Bernoulli:** The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

4.2.3 DECISION TREE ALGORITHM

Decision Tree is a supervised learning technique that can be used for both classification and regression problems, but mostly it is preferred for solving classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision Tree, there are two nodes, which are the Decision Node and Leaf Node.

Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a Decision Tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A Decision Tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub trees.

The Decision Tree Algorithm belongs to the family of supervised machine learning algorithms. It can be used for both a classification problem as well as for a regression problem.

The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem

in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision Tree:

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

In Decision Tree the major challenge is to identify the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain:

When we use a node in a Decision Tree to partition the training instances into smaller subsets, the entropy changes. Information gain is a measure of this change in entropy.

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples.

The higher the entropy the more the information content.

2. Gini Index:

Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini index should be preferred. Sklearn supports “Gini” criteria for Gini Index and by default, it takes “gini” value.

The most notable types of Decision Tree algorithms are:-

1. **IDichotomiser 3 (ID3):**

   This algorithm uses Information Gain to decide which attribute is to be used to classify the current subset of the data. For each level of the tree, information

gain is calculated for the remaining data recursively.

2. **C4.5:** This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.
3. **Classification and Regression Tree (CART):** It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

Working:

In a Decision Tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure

(ASM).

- Step-3: Divide the S into subsets that contains possible values for the best attributes.
- Step-4: Generate the Decision Tree node, which contains the best

attribute.

- Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you

cannot further classify the nodes and call the final node as a leaf node.

4.2.4 RANDOM FOREST ALGORITHM

Random Forest is a supervised learning algorithm. It is an extension of machine learning classifiers which include the bagging to improve the performance of Decision Tree. It combines tree predictors, and trees are dependent on a random vector which is independently sampled. The distribution of all trees are the same. Random Forests splits nodes using the best among of a predictor subset that are randomly chosen from the node itself, instead of splitting nodes based on the variables. The time complexity of the worst case of learning with Random Forests is O(M(dnlogn)) , where M is the number of growing trees, n is the number of instances, and d is the data dimension.

It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest consists of trees. It is said that the more trees it has, the more robust a forest is. Random Forests create Decision Trees on randomly selected data samples, get predictions from each tree and select the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random Forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

Assumptions:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Advantages:

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages:

Although Random Forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

4.2.5 LOGISTIC REGRESSION ALGORITHM

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S"shaped logistic function, which predicts two maximum values (0 or 1).

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Advantages:

Logistic Regression is one of the simplest machine learning algorithms and is easy to implement yet provides great training efficiency in some cases. Also due to these reasons, training a model with this algorithm doesn't require high computation power.

The predicted parameters (trained weights) give inference about the importance of each feature. The direction of association i.e. positive or negative is also given. So we can use Logistic Regression to find out the relationship between the features.
This algorithm allows models to be updated easily to reflect new data, unlike Decision Tree or Support Vector Machine. The update can be done using stochastic gradient descent.

Logistic Regression outputs well-calibrated probabilities along with classification results. This is an advantage over models that only give the final classification as results. If a training example has a 95% probability for a class, and another has a 55% probability for the same class, we get an inference about which training examples are more accurate for the formulated problem.

Disadvantages:

Logistic Regression is a statistical analysis model that attempts to predict precise probabilistic outcomes based on independent features. On high dimensional datasets, this may lead to the model being over-fit on the training set, which means overstating the accuracy of predictions on the training set and thus the model may not

be able to predict accurate results on the test set. This usually happens in the case when the model is trained on little training data with lots of features. So on high dimensional datasets, Regularization techniques should be considered to avoid over- fitting (but this makes the model complex). Very high regularization factors may even lead to the model being under-fit on the training data.

Non linear problems can't be solved with logistic regression since it has a linear decision surface. Linearly separable data is rarely found in real world scenarios. So the transformation of non linear features is required which can be done by increasing the number of features such that the data becomes linearly separable in higher dimensions.

Non-Linearly Separable Data:

It is difficult to capture complex relationships using logistic regression. More powerful and complex algorithms such as Neural Networks can easily outperform this algorithm

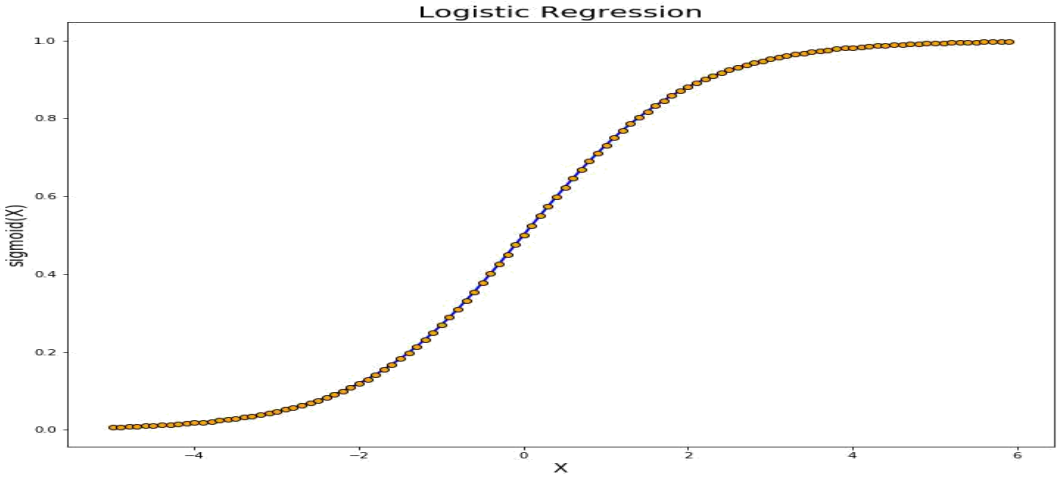


Figure: Logistic Regression

4.2.6 K-NEAREST NEIGHBORS (KNN)

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Regression problems use a similar concept as classification problem, but in this case, the average the k nearest neighbors is taken to make a prediction about a classification.

The main distinction here is that classification is used for discrete values, whereas regression is used with continuous ones. However, before a classification can be made, the distance must be defined. Euclidean distance is most commonly used, which we'll delve into more below.

It's also worth noting that the KNN algorithm is also part of a family of "lazy learning" models, meaning that it only stores a training dataset versus undergoing a training stage. This also means that all the computation occurs when a classification or prediction is being made. Since it heavily relies on memory to store all its training data, it is also referred to as an instance-based or memory-based learning method.

Compute KNN: defining k

The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point. For example, if k=1, the instance will be assigned to the same class as its single nearest neighbor. Defining k can be a balancing act as different values can lead to overfitting or underfitting. Lower values of k can have high variance, but low bias, and larger values of k may lead to high bias and lower variance. The choice of k will largely depend on the input data as data with more outliers or noise will likely perform better with higher values of k. Overall, it is recommended to have an odd number for k to avoid ties in classification, and cross-validation tactics can help you choose the optimal k for your dataset.

Advantages

- **Easy to implement**: Given the algorithm's simplicity and accuracy, it is one of the first classifiers that a new data scientist will learn.

- **Adapts easily**: As new training samples are added, the algorithm adjusts to account for any new data since all training data is stored into memory.

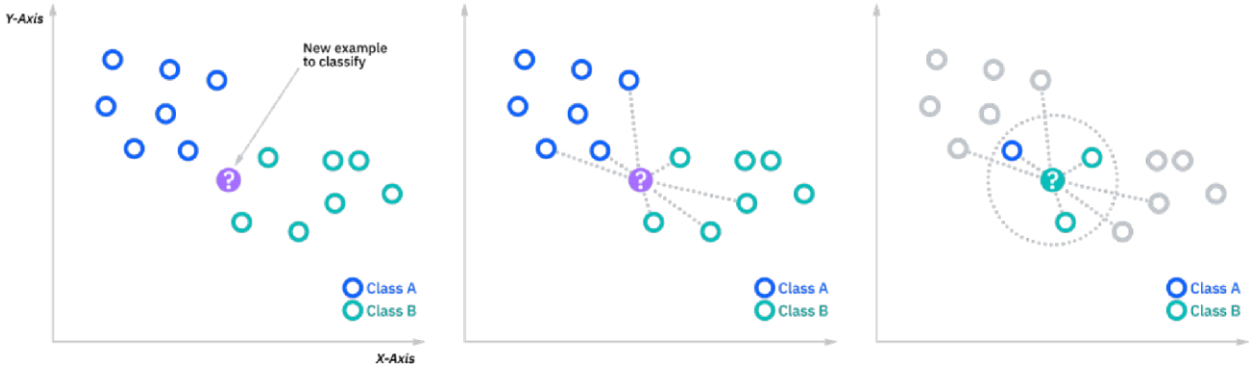
- **Few hyper parameters**: KNN only requires a k value and a distance metric, which is low when compared to other machine learning algorithms.

Disadvantages

- **Does not scale well**: Since KNN is a lazy algorithm, it takes up more memory and data storage compared to other classifiers. This can be costly from both a time and money perspective. More memory and storage will drive up business expenses and more data can take longer to compute. While different data structures, such as Ball-Tree, have been created to address the computational inefficiencies, a different classifier may be ideal depending on the business problem.

- **Curse of dimensionality**: The KNN algorithm tends to fall victim to the curse of dimensionality, which means that it doesn't perform well with high-dimensional data inputs. This is sometimes also referred to as peaking phenomenon, where after the algorithm attains the optimal number of features, additional features increases the amount of classification errors, especially when the sample size is smaller.

Prone to overfitting: Due to the "curse of dimensionality", KNN is also more prone to overfitting. While feature selection and dimensionality reduction techniques are leveraged to preventthis from occurring, the value of k can also impact the model's behavior. Lower values of k can overfit whereas higher values of k tend to "smooth out" the prediction values since it is averaging the values over agreater area, or neighborhood. However, if the value of k is too high, then it can underfit the data.



4.3 DATASET DETAILS

- Of the 76 attributes available in the dataset,14 attributes are considered for theprediction of the output.
- Heart Disease UCI : https://raw.githubusercontent.com/mrdbourke/zero-to-mastery-ml/master/data/heart-disease.csv

A	B	C	D	E	F	G	H	I	J	K	L	M	N
age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	
37	1	2	130	250	0	1	187	0	3.5	0	0	2	
41	0	1	130	204	0	0	172	0	1.4	2	0	2	
56	1	1	120	236	0	1	178	0	0.8	2	0	2	
57	0	0	120	354	0	1	163	1	0.6	2	0	2	
57	1	0	140	192	0	1	148	0	0.4	1	0	1	
56	0	1	140	294	0	0	153	0	1.3	1	0	2	
44	1	1	120	263	0	1	173	0	0	2	0	3	
52	1	2	172	199	1	1	162	0	0.5	2	0	3	
57	1	2	150	168	0	1	174	0	1.6	2	0	2	
54	1	0	140	239	0	1	160	0	1.2	2	0	2	
48	0	2	130	275	0	1	139	0	0.2	2	0	2	
49	1	1	130	266	0	1	171	0	0.6	2	0	2	
64	1	3	110	211	0	0	144	1	1.8	1	0	2	
58	0	3	150	283	1	0	162	0	1	2	0	2	
50	0	2	120	219	0	1	158	0	1.6	1	0	2	
58	0	2	120	340	0	1	172	0	0	2	0	2	
66	0	3	150	226	0	1	114	0	2.6	0	0	2	
43	1	0	150	247	0	1	171	0	1.5	2	0	2	
69	0	3	140	239	0	1	151	0	1.8	2	2	2	
59	1	0	135	234	0	1	161	0	0.5	1	0	3	
44	1	2	130	233	0	1	179	1	0.4	2	0	2	
42	1	0	140	226	0	1	178	0	0	2	0	2	
61	1	2	150	243	1	1	137	1	1	1	0	2	
40	1	3	140	199	0	1	178	1	1.4	2	0	3	
71	0	1	160	302	0	1	162	0	0.4	2	2	2	
59	1	2	150	212	1	1	157	0	1.6	2	0	2	
51	1	2	110	175	0	1	123	0	0.6	2	0	2	
65	0	2	140	417	1	0	157	0	0.8	2	1	2	
53	1	2	130	197	1	0	152	0	1.2	0	0	2	
41	0	1	105	198	0	1	168	0	0	2	1	2	
65	1	0	120	177	0	1	140	0	0.4	2	0	3	
44	1	1	130	219	0	0	188	0	0	2	0	2	
54	1	2	125	273	0	0	152	0	0.5	0	1	2	
51	1	3	125	213	0	0	125	1	1.4	2	1	2	

Input dataset attributes

S. No.	Attribute	Description	Type
1	Age	Patient's age (29 to 77)	Numerical
2	Sex	Gender of patient(male-0 female-1)	Nominal
3	Cp	Chest pain type	Nominal
4	Trestbps	Resting blood pressure(in mm Hg on admission to hospital ,values from 94 to 200)	Numerical
5	Chol	Serum cholesterol in mg/dl, values from 126 to 564)	Numerical
6	Fbs	Fasting blood sugar>120 mg/dl, true-1 false-0)	Nominal
7	Restecg	Resting electrocardiographics result	Nominal
		(0 to 1)	
8	Thalach	Maximum heart rate achieved(71 to 202)	Numerical
9	Exang	Exercise included agina(1-yes 0-no)	Nominal
10	Oldpeak	ST depression introduced by exercise relative to rest (0 to .2)	Numerical
11	Slope	The slop of the peak exercise ST segment (0 to 1)	Nominal
12	Ca	Number of major vessels (0-3)	Numerical
13	Thal	3-normal	Nominal
14	Targets	1 or 0	Nominal

TABLE2: Attributes of the dataset

4.4 PERFORMANCE ANALYSIS

In this project, various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Forest, Logistic Regression, K-nearest neighbors are used to predict heart disease. Heart Disease UCI dataset, has a total of 76 attributes, out of those only 14 attributes are considered for the prediction of heart disease. Various attributes of the patient like gender, chest pain type, fasting blood pressure, serum cholesterol, exang, etc are considered for this project. The accuracy for individual algorithms has to measure and whichever algorithm is giving the best accuracy, that is considered for the heart disease prediction. For evaluating the experiment, various evaluation metrics like accuracy, confusion matrix, precision, recall, and f1-score are considered.

Accuracy- Accuracy is the ratio of the number of correct predictions to the total number of inputs in the dataset. It is expressed as:

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

Confusion Matrix- It gives us a matrix as output and gives the total performance of the system.

Figure: Confusion Matrix

Where

TP: True positive

FP: False Positive

FN: False Negative

TN: True Negative

Correlation Matrix: The correlation matrix in machine learning is used for feature selection. It represents dependency between various attributes.

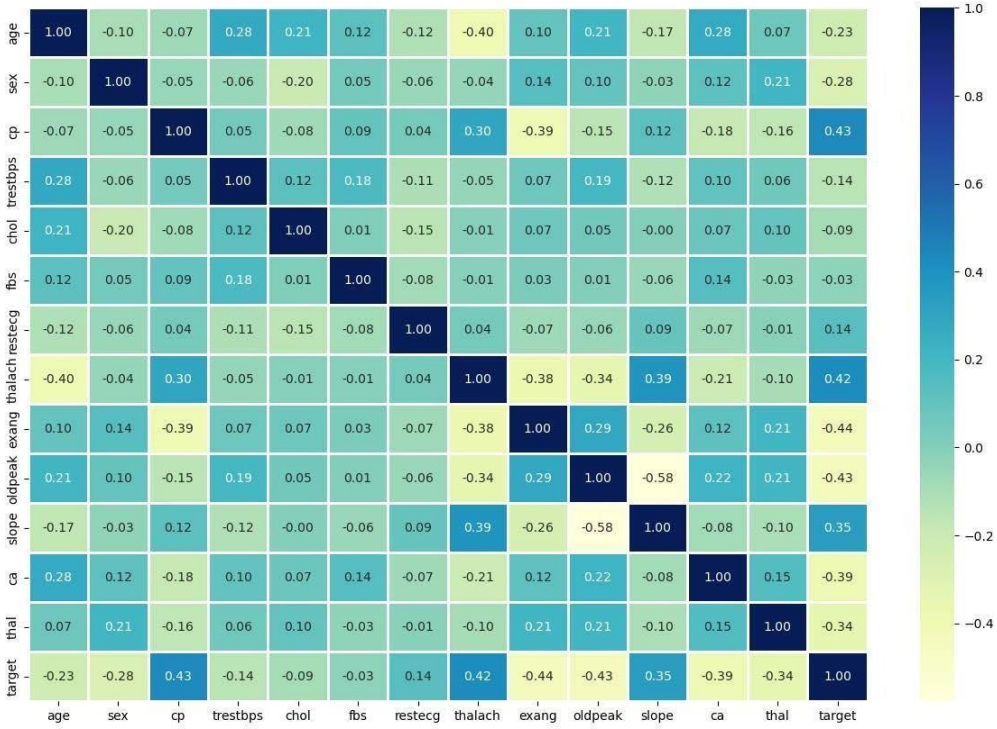


Fig: Correlation matrix

- Precision- It is the ratio of correct positive results to the total number of positive results predicted by the system.
  It is expressed as: TP/(TP+FP)
- Recall-It is the ratio of correct positive results to the total number of positive results predicted by the system.
  It is expressed as: TP/(TP+FN)
- F1 Score-It is the harmonic mean of Precision and Recall. It measures the test accuracy. The range of this metric is 0 to 1.

4.5 PERFORMANCE MEASURES

• The highest accuracy is given by Ada-Boost Classifier.

ACCURACY OF LOGISTIC REGRESSION	88.52%
ACCURACY OF RANDOM FOREST	85.25%
ACCURACY OF K-NEAREST NEIGHBOR	68.85%
ACCURACY OF ADA-BOOST CLASSIFIER	90.16%

TABLE: Accuracy comparison of algorithms Algorithm Accuracy

4.6 RESULT

After performing the machine learning approach for training and testing we find that accuracy of the XG-boost is better compared to other algorithms. Accuracy is calculated with the support of the confusion matrix of each algorithm, here the number count of TP, TN, FP, FN is given and using the equation of accuracy, value has been calculated and it is concluded that extreme gradient boosting is best with 81% accuracy and the comparison is shown below.

Algorithm	**Accuracy**
Logistic Regression	88.52 %
Random Forest	85.25 %
K-Nearest Neighbors	68.85 %
Ada-Boost Classifier	90.16%

TABLE 2: Accuracy Table

CHAPTER 5: CONCLUSION AND FUTURE SCOPE

Heart diseases are a major killer in India and throughout the world, application of promising technology like machine learning to the initial prediction of heart diseases will have a profound impact on society. The early prognosis of heart disease can aid in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications, which can be a great milestone in the field of medicine. The number of people facing heart diseases is on a raise each year. This prompts for its early diagnosis and treatment. The utilization of suitable technology support in this regard can prove to be highly beneficial to the medical fraternity and patients. In this project, the four different machine learning algorithms used to measure the performance are K-nearest neighbor, Random Forest, Logistic Regression, and Adaptive Boosting applied on the dataset.

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 taken into the 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 four 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 four the Adaptive Boosting Classifier gives the highest accuracy of 90.16%.

ANNEXURE I : CODE

In [1]:
```
# Import all the tools we need

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

# we want our plots ton appear inside the notebook
%matplotlib inline

#Models from sklearn
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier



#Model Evaluation
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, r2_score,f1_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import plot_roc_curve
```

Load Data

In [2]:
```
df = pd.read_csv("data/heart-disease.csv")
df
```

In [3]:
```
df.shape
# (row,col)
```

Out[3]: (303, 14)

In [4]:
```
df.head()
```

In [5]:
```
df.tail()
```

In [6]:
```
# Let's find out how many of each type are there
df["target"].value_counts()
```

In [7]:
```
# plt.figure( facecolor="#FFFBEB")
plt.rcParams['axes.facecolor'] = '#FEFCF3'
df["target"].value_counts().plot(kind="bar",color=["#D61C4E","#5FD068"]  )


# ax = plt.axes()
# ax.set_facecolor("#FFFBEB")
plt.ylabel("No Of People ->",fontweight="bold")
plt.title("(Heart Attack) vs (No of People)")

plt.xticks(rotation=0)

plt.show()
```

In [8]:
```
df.info()
```

In [9]:
```
df.describe()
```

Heart Disease Frequenct according to Sex

In [10]:
```
df.sex.value_counts()
```

In [11]:
```
crosstab_TS = pd.crosstab(df.target,df.sex)
crosstab_TS
```

In [12]:
```
# # HeatMap
# DvG=sns.heatmap(crosstab_TS,xticklabels="FM",yticklabels="NY",cmap="YlGnBu" )
# DvG.set(xlabel='Gender ----->',ylabel='Heart Disease ----->')
```

In [13]:
```
legend_label={'0':'#FF8FB1','1':'#0D4C92'}
crosstab_TS.plot(kind="bar",color=[ "#FF8FB1","#0D4C92"],label=legend_label)

plt.title("Heart Disease Frequency for Sex")
plt.xlabel("0= No Disease , 1 = Disease")
plt.ylabel("No Of People")

plt.legend(["Female","Male"])
plt.xticks(rotation=0);
```

Age vs. Max Heart rate for heart Disease ¶

In [14]:
```
# Create another figure
plt.figure(figsize=(10,6))
plt.rcParams['axes.facecolor'] = '#FEFCF3'
# Scatter with positiv examples
AvsM_Y = df.age[df.target==1]
plt.scatter(AvsM_Y,df.thalach[df.target==1],c="#D2001A")

AvsM_N = df.age[df.target==0]
plt.scatter(AvsM_N,df.thalach[df.target==0],c="#5FD068")

plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Heart Rate");


plt.legend(["Disease","No Disease"]);
```

In [15]:
```
df.age[df.target==1]
```

In [16]:
```
# Check the distriubution of age with the histogram
plt.rcParams['axes.facecolor'] = '#FFE6F7'
df.age.plot.hist(edgecolor='k',alpha=0.5,color='#937DC2')

plt.title("Histogram of Age")
plt.xlabel("Age  ---------->")
plt.ylabel("No of People   --------->")
```

Heart Disease Frequency per Chest Pain

3 cp-chest pain type

0: Typical angina: chest pain related decrease blood supply to the heart

1: Atypical angina: chest pain not related to heart

2: Non-anginal pain: typically esophageal spasms (non heart related)

3: Asymptomatic: chest pain not showing signs of disease

In [17]:
```
CPvT = pd.crosstab(df.cp,df.target)
CPvT
```

In [18]:
```
legend_label={'0':'#FF8FB1','1':'#0D4C92'}
plt.rcParams['axes.facecolor'] = '#FEFCF3'
CPvT.plot(kind="bar",color=[ "#68B984","#E0144C"],label=legend_label)

plt.title("Heart Disease Frequency per Chest Pain")
plt.xlabel("CP ( Chest Pain)")
plt.ylabel("No Of People")

plt.legend(["No Disease","Disease"])

plt.xticks(rotation=0);
```

In [19]:
```
correlation_metrics = df.corr()
correlation_metrics
```

Below shows the relation between metrics

> **Positive value > 0 shows some relation between metrics**

> **Negative value <0 shows no relation between metrics**

In [20]:
```
fig,ax = plt.subplots(figsize=(15,10))
ax = sns.heatmap(correlation_metrics,annot=True,linewidths=1,fmt=".2f",cmap="YlGnBu")
```

Modelling

In [21]:
```
df.head()
```

Splitting Of Data

In [22]:
```
X = df.drop("target",axis=1)
Y = df["target"]
```

In [23]:
```
X
```

In [24]:
```
Y
```

In [25]:
```
np.random.seed(42)
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2)
```

3 Different type of Machine Learning Model

- Logisic Regression
- K-Neaest Classifier
- Random Forest Classifier

Create a Function to fit and score model

In [26]:
```
models = {"KNN":KNeighborsClassifier(),
          "AdaBoostClassifier":AdaBoostClassifier(),
          "Logistic Regression":LogisticRegression(),
          "Random Forest": RandomForestClassifier(),

         }

def fit_and_score(models,X_train,X_test,Y_train,Y_test):
    """Fits and Evaluate given Machine Learning Models
    models: A dict of scikit learning Models
    X_test: testing data(no labels)
    X_train: training data(no labels)
    Y_train:training target data corresponding to training data
    Y_test: testing data correspoding to testing data
    """

    #set random seed
    np.random.seed(42)
    # Make a model to keep model score
    model_score={}
    # Loop through Models

    for name,model in models.items():
        # Fit the model to the data
        model.fit(X_train,Y_train)

        # Score of model trained on train data
        score = model.score(X_test,Y_test)
        model_score[name] = score
    return model_score
```

In [27]:
```
model_scores = fit_and_score(models,X_train,X_test,Y_train,Y_test)
model_scores
```

In [28]:
```
plt.rcParams['axes.facecolor'] = '#EFFFFD'
ax = pd.DataFrame(model_scores,index=["Accuracy"]).T.plot(kind="bar",color="#42C2FF",figsize=(10,6))
```

Hyperparameter Tuning

KNN

In [29]:
```
train_scores=[]
test_scores=[]

# Create a list of different values for n_neighbors
neighbors = range(1,20)
# neighbors 0 - 20
knn=KNeighborsClassifier()

# Loop through different n_neighbors

for i in neighbors:
    knn.set_params(n_neighbors=i)

    # Fit the Algorithm with training data
    knn.fit(X_train,Y_train)

    # Update the training score list
    train_scores.append(knn.score(X_train,Y_train))

    # Update the test score list
    test_scores.append(knn.score(X_test,Y_test))
```

In [30]:
```
plt.rcParams['axes.facecolor'] = '#EFFFFD'

plt.plot(neighbors,train_scores,color="#344D67"  )
plt.plot(neighbors,test_scores,color="#7FE9DE"  )


plt.title("KNN Model's Accuracy on varying n_neighbors")
plt.xlabel("n_neighbors ---------->")
plt.ylabel("Accuracy ---------->")

plt.legend(["Train Data","Test Data"])


print(f"Maximum KNN score on the test_data is {max(test_scores)*100:.2f}  %")
print(f"Minimum KNN score on the test_data is {min(test_scores)*100:.2f}  %")

print()

print(f"Maximun KNN score on train data is {max(train_scores)*100:.2f} %")
print(f"Minimum KNN score on train data is {min(train_scores)*100:.2f} %")
```

```
Maximum KNN score on the test_data is 75.41  %
Minimum KNN score on the test_data is 62.30  %

Maximun KNN score on train data is 100.00 %
Minimum KNN score on train data is 66.53 %
```

AdaBoost

In [31]:
```
train_scores=[]
test_scores=[]

n_estimators = range(1,40)
ada_boost =AdaBoostClassifier()
for i in n_estimators:
    ada_boost.set_params(n_estimators=i)
    ada_boost.fit(X_train,Y_train)

    train_scores.append(ada_boost.score(X_train,Y_train))
    test_scores.append(ada_boost.score(X_test,Y_test))
```

In [32]:
```
plt.rcParams['axes.facecolor'] = '#EFFFFD'

plt.plot(n_estimators,train_scores,color="#344D67" )
plt.plot(n_estimators  , test_scores,color="#7FE9DE" )


plt.title("ADABoost Model's Accuracy on varying n_estimator")
plt.xlabel("n_neighbors ---------->")
plt.ylabel("Accuracy ---------->")

plt.legend(["Train Data","Test Data"])


print(f"Maximum ADA score on the test_data is {max(test_scores)*100:.2f}  %")
print(f"Minimum ADAscore on the test_data is {min(test_scores)*100:.2f}  %")

print()

print(f"Maximun ADA score on train data is {max(train_scores)*100:.2f} %")
print(f"Minimum ADA score on train data is {min(train_scores)*100:.2f} %")
```

```
Maximum ADA score on the test_data is 90.16  %
Minimum ADAscore on the test_data is 72.13  %
```

Logistic Regression

In [33]:
```
train_scores=[]
test_scores=[]

# Create a list of different values for n_neighbors
C = np.arange(0.01,1,0.02)
# C 0 - 0.4 (jump of 0.2)
logistic=LogisticRegression(max_iter=1000)

# Loop through different n_neighbors

for i in C:
    logistic.set_params(C=i)

    # Fit the Algorithm with training data
    logistic.fit(X_train,Y_train)

    # Update the training score list
    train_scores.append(logistic.score(X_train,Y_train))

    # Update the test score list
    test_scores.append(logistic.score(X_test,Y_test))
```

In [34]:
```
plt.rcParams['axes.facecolor'] = '#EFFFFD'

plt.plot(C,train_scores,color="#344D67" )
plt.plot(C  , test_scores,color="#7FE9DE" )


plt.title("Logistic Model's Accuracy on varying C")
plt.xlabel("C ---------->")
plt.ylabel("Accuracy ---------->")

plt.legend(["Train Data","Test Data"])


print(f"Maximum Logistic score on the test_data is {max(test_scores)*100:.2f}  %")
print(f"Minimum Logistic on the test_data is {min(test_scores)*100:.2f}  %")

print()

print(f"Maximun Logistic score on train data is {max(train_scores)*100:.2f} %")
print(f"Minimum Logistic score on train data is {min(train_scores)*100:.2f} %")
```

```
Maximum Logistic score on the test_data is 88.52  %
Minimum Logistic on the test_data is 86.89  %

Maximun Logistic score on train data is 87.19 %
Minimum Logistic score on train data is 76.45 %
```

Random Forest

In [35]:
```
train_scores=[]
test_scores=[]

# Create a list of different values for n_neighbors
n_estimators = range(1,100,1)
# C 0 - 0.4 (jump of 0.2)
random=RandomForestClassifier()

# Loop through different n_neighbors

for i in n_estimators:
    random.set_params(n_estimators=i)

    # Fit the Algorithm with training data
    random.fit(X_train,Y_train)

    # Update the training score list
    train_scores.append(random.score(X_train,Y_train))

    # Update the test score list
    test_scores.append(random.score(X_test,Y_test))
```

In [36]:
```
plt.rcParams['axes.facecolor'] = '#EFFFFD'

plt.plot(n_estimators ,train_scores,color="#344D67" )
plt.plot(n_estimators , test_scores,color="#7FE9DE" )


plt.title("RandomForest Model's Accuracy on varying n_estimator")
plt.xlabel("n_estimator ---------->")
plt.ylabel("Accuracy ---------->")

plt.legend(["Train Data","Test Data"])


print(f"Maximum RandomForest score on the test_data is {max(test_scores)*100:.2f}  %")
print(f"Minimum RandomForest on the test_data is {min(test_scores)*100:.2f}  %")

print()

print(f"Maximun RandomForest score on train data is {max(train_scores)*100:.2f} %")
print(f"Minimum RandomForest score on train data is {min(train_scores)*100:.2f} %")
```

```
Maximum RandomForest score on the test_data is 88.52  %
Minimum RandomForest on the test_data is 75.41  %
```

Logistic Regression ¶

- Baseline Model

In [37]:
```
np.random.seed(42)
L_clf = LogisticRegression( max_iter=1000)
L_clf.fit(X_train,Y_train)

L_Baseline = L_clf.score(X_test,Y_test)
L_Baseline
```

Out[37]:
```
0.8852459016393442
```

- clf_2

In [38]:
```
# np.random.seed(42)
L_clf_2 = LogisticRegression(C=0.23357214690901212,

                             max_iter=1000)
# 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'
L_clf_2.fit(X_train,Y_train)

L_clf2 = L_clf_2.score(X_test,Y_test)
L_clf2
```

Out[38]:
```
0.8852459016393442
```

In [39]:
```
# Create a hyperparamter grid for LogisticRegression
log_reg_grid={"C":np.logspace(-4,4,20),
              "solver":['newton-cg', 'liblinear'   ],
              }

# Create a hyper paramete grid for RandomForestClassifier
rf_grid = {"n_estimators":np.arange(10,1000,50),
           "max_depth":[None,3,5,10],
           "min_samples_split":np.arange(2,20,2),
           "min_sample_lead":np.arange(1,20,2)}
```

Now we have hyperparameter grid for our models, let's tune them using `RandomizedSearchCV`

In [40]:
```
# Tune Logistic Regression
np.random.seed(42)

# Setup random hyperparameter search for LogisticRegression
RS_log_reg = RandomizedSearchCV(LogisticRegression(),
                                param_distributions=log_reg_grid
                                ,cv=10,
                               n_iter=40,verbose=True)

# Fit random hypermete search model for Logistic Regression
RS_log_reg.fit(X_train,Y_train);
```

```
Fitting 10 folds for each of 40 candidates, totalling 400 fits
```

In [41]:
```
RS_Logistic = RS_log_reg.score(X_test,Y_test)
RS_Logistic
```

Out[41]:
```
0.8524590163934426
```

In [42]:
```
RS_log_reg.best_params_
```

Hyperparameter Tuning with GridSearchCV

Since our LogisticRegression Model provides the best scores so far, we'll try and improve them again using GridSearchCV.....

In [43]:
```
# Create a hyperparamter grid for LogisticRegression


GS_log_reg = GridSearchCV(LogisticRegression(),
                          param_grid=log_reg_grid,
                         cv=5,verbose=True)

GS_log_reg.fit(X_train,Y_train)
print(GS_log_reg.score(X_test,Y_test));
```

```
Fitting 5 folds for each of 40 candidates, totalling 200 fits
0.8852459016393442
```

In [44]:
```
GS_Logistic = GS_log_reg.score(X_test,Y_test)
GS_Logistic
```

Out[44]:
```
0.8852459016393442
```

In [45]:
```
GS_log_reg.best_params_
```

Out[45]:
```
{'C': 0.23357214690901212, 'solver': 'liblinear'}
```

In [46]:
```
HPT_scores={"GS_Logistic":GS_Logistic,"LogisticRegression_Baseline":L_Baseline,
            "RF_SelfTune":L_clf2,
            "RS_Logistic":RS_Logistic
            }

model_s = pd.DataFrame(HPT_scores,index=["Accuracy"])
model_s
```

Out[46]:

	GS_Logistic	LogisticRegression_Baseline	RF_SelfTune	RS_Logistic
Accuracy	0.885246	0.885246	0.885246	0.852459

In [47]:
```
# code for annotated barplot
plt.figure(figsize=(8, 6))
splot = sns.barplot( data=model_s, palette='rocket')
plt.rcParams['axes.facecolor'] = '#FAF8F1'
plt.ylabel("Accuracy --------->", size=14)

plt.title("Accuracy after Tuning LogisticRegression", size=18)

for p in splot.patches:
    splot.annotate(round(p.get_height(), 4)*100,
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha='center', va='center',color='white',
                   size=14,
                   xytext=(0, -70),
                   rotation=0,
                   textcoords='offset points')
    plt.xticks(rotation=90)
```

In [48]:
```
Y_pred = GS_log_reg.predict(X_test)
Y_pred
```

Out[48]:
```
array([0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0], dtype=int64)
```

In [49]:
```
from sklearn.metrics import RocCurveDisplay
c_matrix = confusion_matrix(Y_test,Y_pred)

plot_roc_curve(GS_log_reg,X_test,Y_test);
```

In [50]:
```
confusion_matrix(Y_test,Y_pred)
```

Out[50]:
```
array([[25,  4],
       [ 3, 29]], dtype=int64)
```

In [51]:
```
sns.set(font_scale=2)
sns.heatmap(confusion_matrix(Y_test,Y_pred),annot=True)
plt.xlabel("True Label")
plt.ylabel("Predicted Label");
```

In [52]:
```
print(classification_report(Y_test,Y_pred))
```

Calculating evaluation metrics using cross-validation

We're going to calculate precision, recall and f1-score of our model using cross-validation and to do so we'll be using `cross_val_score()`

In [53]:
```
# Check best hyperparameter
GS_log_reg.best_params_
```

Out[53]:
```
{'C': 0.23357214690901212, 'solver': 'liblinear'}
```

In [54]:
```
clf = LogisticRegression(C=0.23357214690901212,solver='liblinear')
clf.fit(X_train,Y_train)
```

Out[54]:
```
LogisticRegression(C=0.23357214690901212, solver='liblinear')
```

In [55]:
```
def fn_cv_metric(model,X,Y,Y_true,Y_pred):
    # Cross validated accuracy
    cv_acc = cross_val_score(clf,X,Y,cv=5,scoring="accuracy")
    cv_acc=cv_acc.mean()

    # Cross validated precison
    cv_prec = cross_val_score(clf,X,Y,cv=5,scoring="precision")
    cv_prec=cv_prec.mean()

    # Cross validated recall
    cv_recall = cross_val_score(clf,X,Y,cv=5,scoring="recall")
    cv_recall=cv_recall.mean()

    # Cross validated f2-score
    cv_f1 = cross_val_score(clf,X,Y,cv=5,scoring="f1")
    cv_f1=cv_f1.mean()

    acc = accuracy_score(Y_true,Y_pred)
    prec =precision_score(Y_true,Y_pred)
    recall = recall_score(Y_true,Y_pred)
    f1 = f1_score(Y_true,Y_pred)

    # Visulize our cross validated matrix
    cv_metrics = pd.DataFrame({"Accuracy":[cv_acc,acc],
                           "Precision":[cv_prec,prec],
                           "Recall":[cv_recall,recall],
                           "F1":[cv_f1,f1]},index=["0","1"])

    return cv_metrics
```

In [56]:
```
y_pred = clf.predict(X_test )
y_pred
```

In [56]:
```
y_pred = clf.predict(X_test )
y_pred
```

In [57]:
```
cv_metrics = fn_cv_metric(clf,X,Y,Y_test,y_pred)
cv_metrics
```

In [58]:
```
cv_metrics.T.plot(kind="bar",color=["#439A97","#97DECE"],
                  legend=False,
                  title="Classification Metric( CV vs Without CV) [LogisticRegression]",figsize=(30,15))

plt.legend(["With CV","Without CV"])
plt.rcParams['axes.facecolor'] = '#EFFFFD'
plt.xticks(rotation=0);
```

Feature Importance

Features importance is another as asking , which features contributed most to the outcomes of the model and hw did they contribute.

Finding feature importance is differnet is different for each Machine Learning Model . One way to find feature importance is to search for (Model name) feature importance

Let's find the feature importance for our Logisticregression Model

In [59]:
```
# Fit an instance of Logistic Regression
GS_log_reg.best_params_

clf = LogisticRegression(C=0.23357214690901212,solver='liblinear')
clf.fit(X_train,Y_train);
```

In [60]:
```
clf.score(X_test,Y_test)
```

Out[60]:
```
0.8852459016393442
```

In [61]:
```
clf.coef_
```

In [62]:
```
# Match coef of feature to columns\
feature_dict = dict(zip(df.columns,list(clf.coef_[0])))
feature_dict
```

In [63]:
```
pd.DataFrame(feature_dict,index=[0]).T.plot(kind="bar",color="#42C2FF",
                  legend=False,
                  title="Feature Importance Of LogisticRegression",figsize=(10,10))

plt.rcParams['axes.facecolor'] = '#EFFFFD'
# plt.grid(True,color='#42C2FF')
```

AdaBoost Classifier

- Baseline

In [64]:
```
np.random.seed(42)
Ada_clf = AdaBoostClassifier()
Ada_clf.fit(X_train,Y_train)

Ada_Baseline = Ada_clf.score(X_test,Y_test)
Ada_Baseline
```

Out[64]:
```
0.8032786885245902
```

- Self-Tuning

In [65]:
```
# np.random.seed(42)
Ada_Self = AdaBoostClassifier(n_estimators=3)
# 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'
Ada_Self.fit(X_train,Y_train)

Ada_Self_tune = Ada_Self.score(X_test,Y_test)
Ada_Self_tune
```

Out[65]:
```
0.9016393442622951
```

Hyperparameter tuning with RandomSearchCV

In [66]:
```
np.random.seed(42)
from sklearn.tree import DecisionTreeClassifier

def getBaseEstimator(end):
    base_estimators=[]

    for d in range(1,end+1,1):
        base = DecisionTreeClassifier(max_depth=d)
        base_estimators.append(base)

    return base_estimators
```

In [67]:
```
# Create a hyperparamter grid for LogisticRegression

base_estimators = getBaseEstimator(10)
Ada_R_grid={"n_estimators":range(1,10,1),
            "base_estimator":base_estimators,
            }
```

In [68]:
```
# Tune Logistic Regression
np.random.seed(42)

# Setup random hyperparameter search for LogisticRegression
RS_Ada_clf = RandomizedSearchCV(AdaBoostClassifier(),
                                param_distributions=Ada_R_grid
                                ,cv=5,
                               n_iter=100,verbose=True)

# Fit random hypermete search model for Logistic Regression
RS_Ada_clf.fit(X_train,Y_train)
RS_Ada = RS_Ada_clf.score(X_train,Y_train);
```

In [69]:
```
RS_Ada_clf.score(X_test,Y_test)
```

Out[69]:
```
0.819672131147541
```

In [70]:
```
RS_Ada_clf.best_params_
```

Out[70]:
```
{'n_estimators': 8, 'base_estimator': DecisionTreeClassifier(max_depth=1)}
```

Hyperparameter Tuning with GridSearchCV

Since our LogisticRegression Model provides the best scores so far, we'll try and improve them again using GridSearchCV.....

In [71]:
```
base_estimators = getBaseEstimator(5)
Ada_R_grid={"n_estimators":[2,3,4,5],
            "base_estimator":base_estimators,
            }
```

In [72]:
```
# Create a hyperparamter grid for LogisticRegression


GS_Ada_clf = GridSearchCV(AdaBoostClassifier(),
                          param_grid=Ada_R_grid,
                         cv=5,verbose=True)

GS_Ada_clf.fit(X_train,Y_train)
GS_Ada= GS_Ada_clf.score(X_test,Y_test) ;
```

In [73]:
```
GS_Ada
```

Out[73]:
```
0.8360655737704918
```

In [74]:
```
HPT_scores={"GS_AdaBoost":GS_Ada,"AdaBoost_Baseline":Ada_Baseline,
           "RF_SelfTune":Ada_Self_tune,
           "RS_AdaBoost":RS_Ada
           }

model_s = pd.DataFrame(HPT_scores,index=["Accuracy"])
model_s
```

Out[74]:

	GS_AdaBoost	AdaBoost_Baseline	RF_SelfTune	RS_AdaBoost
Accuracy	0.836066	0.803279	0.901639	0.871901

In [75]:
```
# code for annotated barplot
plt.figure(figsize=(11,7))
splot = sns.barplot( data=model_s, palette='rocket' )
plt.rcParams['axes.facecolor'] = '#FAF8F1'
plt.ylabel("Accuracy --------->", size=14)

plt.title("Accuracy after Tuning AdaBoostClassifier", size=18)

for p in splot.patches:
    splot.annotate(round(p.get_height(), 4)*100,
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha='center', va='center',
                   size=20,color='white',
                   xytext=(0, -70),
                   rotation=0,
                   textcoords='offset points')
plt.xticks(rotation=90)
```

In [76]:
```
# Match coef of feature to columns\
feature_dict = dict(zip(df.columns,list(Ada_Self.feature_importances_)))
feature_dict
```

In [77]:
```
Ada_Self.feature_importances_
```

In [78]:
```
pd.DataFrame(feature_dict,index=[0]).T.plot(kind="bar",color="#42C2FF",
                  legend=False,
                  title="Feature Importance Of AdaBoostClassifier",figsize=(10,10))

plt.rcParams['axes.facecolor'] = '#EFFFFD'
# plt.grid(True,color='#42C2FF')
```

In [79]:
```
y_pred = Ada_Self.predict(X_test)
cv_metrics = fn_cv_metric(Ada_Self,X,Y,Y_test,y_pred)
cv_metrics
```

Out[79]:

	Accuracy	Precision	Recall	F1
0	0.847978	0.821587	0.927273	0.870540
1	0.901639	0.882353	0.937500	0.909091

In [80]:
```
sns.set(font_scale=2)
sns.heatmap(confusion_matrix(Y_test,y_pred),annot=True)
plt.xlabel("True Label")
plt.ylabel("Predicted Label");
```

In [81]:
```
cv_metrics.T.plot(kind="bar",color=["#439A97","#97DECE"],
                  legend=False,
                  title="Classification Metric( CV vs Without CV)",figsize=(30,15))

plt.legend(["With CV","Without CV"])
plt.rcParams['axes.facecolor'] = '#EFFFFD'
plt.xticks(rotation=0);
```

RandomForest Classifier

- Baseline

In [82]:
```
np.random.seed(42)
RF_clf_B = RandomForestClassifier()
RF_clf_B.fit(X_train,Y_train)

R_Baseline = RF_clf_B.score(X_test,Y_test)
R_Baseline
```

Out[82]:
```
0.8360655737704918
```

- Self Tuning

In [83]:
```
np.random.seed(42)
RF_clf_ST = RandomForestClassifier(n_estimators=42)
# 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'
RF_clf_ST.fit(X_train,Y_train)

RF_ST = RF_clf_ST.score(X_test,Y_test)
RF_ST
```

Out[83]:
```
0.8524590163934426
```

Hyperparameter tuning with RandomSearchCV

In [84]:
```
np.random.seed(42)
RS_RF_Grid={"n_estimators":[16,39,78,12,42,1200],
       "max_depth":[None,2,5,10,20,30],
       "max_features":["auto","sqrt"],
      "min_samples_split":[2,4,6],
      "min_samples_leaf":[1,2,4]}

clf = RandomForestClassifier(n_jobs=-1)
```

In [85]:
```
# Setup RandomizesSEarchCV

RF_RS_clf = RandomizedSearchCV(estimator=clf,
                           param_distributions=RS_RF_Grid,
                           n_iter=20,
                           cv=5,
                           verbose=2)

# fit the randomSearchCV version of CLF
RF_RS_clf.fit(X_train,Y_train)
```

In [86]:
```
RF_RS=RF_RS_clf.score(X_test,Y_test)
```

In [87]:
```
RF_RS_clf.best_params_
```

Out[87]:
```
{'n_estimators': 12,
 'min_samples_split': 4,
 'min_samples_leaf': 2,
 'max_features': 'sqrt',
 'max_depth': None}
```

Hyperparamete tuning with GridSearchCV

In [88]:
```
GS_RF_Grid={"n_estimators":[12,16,39,42,50,78  ],
      "max_depth":[None, 2,10 ,30],
      "max_features":["auto","sqrt"],
     "min_samples_split":[2,4 ],
     "min_samples_leaf":[1,2 ]}
```

In [89]:
```
clf = RandomForestClassifier(n_jobs=-1)
RF_GS_clf = GridSearchCV(estimator=clf,param_grid=GS_RF_Grid,cv=5,verbose=2)
```

In [90]:
```
RF_GS_clf.fit(X_train,Y_train)
```

In [91]:
```
RF_GS=RF_GS_clf.score(X_test,Y_test)
```

In [92]:
```
RF_GS_clf.best_params_
```

Out[92]:
```
{'max_depth': None,
 'max_features': 'sqrt',
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 12}
```

In [93]:
```
HPT_scores={"GS_RandomForest":RF_GS,"RandomForest_Baseling":R_Baseline,
            "RF_SelfTune":RF_ST,
            "RS_RandomForest":RF_RS
            }

model_s = pd.DataFrame(HPT_scores,index=["Accuracy"])
model_s
```

Out[93]:

	GS_RandomForest	RandomForest_Baseling	RF_SelfTune	RS_RandomForest
Accuracy	0.836066	0.836066	0.852459	0.819672

In [94]:
```
# code for annotated barplot
plt.figure(figsize=(8, 6))
splot = sns.barplot( data=model_s, palette='rocket')
plt.rcParams['axes.facecolor'] = '#FAF8F1'
plt.ylabel("Accuracy --------->", size=14)

plt.title("Accuracy after Tuning RandomForest", size=18)

for p in splot.patches:
    splot.annotate(round(p.get_height(), 4)*100,
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha='center', va='center',color='white',
                   size=14,
                   xytext=(0, -70),
                   rotation=0,
                   textcoords='offset points')
    plt.xticks(rotation=90)
```

In [95]:
```
RF_clf_ST.feature_importances_
```

In [96]:
```
# Match coef of feature to columns\
feature_dict = dict(zip(df.columns,list(RF_clf_ST.feature_importances_)))
feature_dict
```

In [97]:
```
pd.DataFrame(feature_dict,index=[0]).T.plot(kind="bar",color="#42C2FF",
                  legend=False,
                  title="Feature Importance RandomForestClassifier",figsize=(10,10))

plt.rcParams['axes.facecolor'] = '#EFFFFD'
# plt.grid(True,color='#42C2FF')
```

In [100]:
```
y_pred = RF_clf_ST.predict(X_test)
cv_metrics = fn_cv_metric(RF_clf_ST,X,Y,Y_test,y_pred)
cv_metrics
```

In [101]:
```
sns.set(font_scale=2)
sns.heatmap(confusion_matrix(Y_test,y_pred),annot=True)
plt.title("")
plt.xlabel("True Label")
plt.ylabel("Predicted Label");
```

In [102]:
```
cv_metrics.T.plot(kind="bar",color=["#439A97","#97DECE"],
                  legend=False,
                  title="Classification Metric( CV vs Without CV)",figsize=(30,15))

plt.legend(["With CV","Without CV"])
plt.rcParams['axes.facecolor'] = '#EFFFFD'
plt.xticks(rotation=0);
```

In [103]:
```
import pickle

pickle.dump(Ada_Self,open("AdaBoostClassifier_90.pkl","wb"))
```

Heart Disease Prediction Using Machine Learning

Nishika Mittal
Information Technology
Maharaja Agrasen Institute of Technology Delhi, India
nishikamittal0905@gmail.com

Nikhil Verma
Information Technology
Maharaja Agrasen Institute of Technology Delhi, India
nikhil.kops@gmail.com

Abstract— Heart disease is considered as one of the major causes of death throughout the world. It cannot be easily predicted by the medical practitioners as it is a difficult task which demands expertise and higher knowledge for prediction. The research paper mainly intends to predict the occurrence of a disease based on data gathered from Kaggle and Cleveland foundation medical research particularly in Heart Disease.. We prepared a heart disease prediction system to predict whether the patient is likely to be diagnosed with a heart disease or not using the medical history of the patient. We used different algorithms of machine learning such as logistic regression, Random Forest Classifier, AdaBoost Classifier and KNN to predict and classify the patient with heart disease. The strength of the proposed model was quiet satisfying and was able to predict evidence of having a heart disease in a particular individual by using AdaBoost Classifier which showed a good accuracy in comparison to the previously used classifier such as KNN, Random forest, etc.

Keywords— supervised; unsupervised; reinforced; linear regression; AdaBoostClassifier decision tree; python programming; jupyter Notebook; confusion matrix;

I. INTRODUCTION

The highest mortality of both India and abroad is mainly because of heart disease. According to World Health Organization (WHO), heart related diseases are responsible for the taking 17.7 million lives every year, 31% of all global deaths. Hence, this is vital time to check this death rate by identifying the disease correctly in the initial stage. We can use data mining technologies to discover knowledge from the datasets. The discovered knowledge can be used by the healthcare administrators to improve the quality of service. Anticipating patient's future behavior on the given history is one of the important applications of data mining techniques that can be used in healthcare management.

Machine Learning is one of the efficient technologies for the testing, which is based on training and testing. It is the branch of Artificial Intelligence (AI) which is one of broad area of learning where machines emulating human abilities, machine learning is a specific branch of AI. On the other hand machines learning systems are trained to learn how to process and make use of data hence the combination of both technologies is also called as Machine Intelligence.

It recognizes who all are having any symptoms of heart disease such as chest pain or high blood pressure and on the basis of these; comparison is done in the terms of accuracy of algorithms such as in this project we have used four algorithms which are Random forest, Logistic regression, K-neighbour, Adaboost Classifier.

In this paper, we calculate the accuracy of four different machine learning approaches and on the basis of calculation,, we conclude that which one is best among them. A dataset is selected from the UCI repository with patient's medical history and attributes. By using this dataset, we predict whether the patient can have a heart disease or not. To predict this, we use 14 medical attributes of a patient and classify them if the patient is likely to have a heart disease. These medical attributes are trained under three algorithms: Logistic regression, KNN and Random Forest Classifier. Most efficient of these algorithms is AdaBoostClassifier which gives us the accuracy of **90.16%**. And, finally we classify patients that are at risk of getting a heart disease or not.

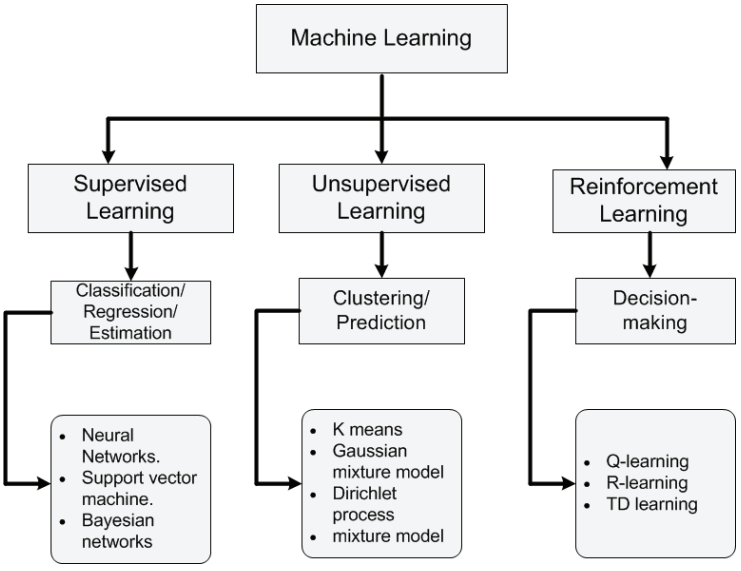
The further paper is divided into sections, Section 1 of this paper consist the introduction about the machine learning and heart diseases. Section II described, the machine learning classification. Section III illustrated the related work of researchers. Section IV is about the methodology used for this prediction system. Section V is about the algorithms used in this project. Section VI briefly describes the dataset and their analysis with the result of this project. And the last Section VII concludes the summary of this paper with slight view about future scope of this paper.

II. MACHINE LEARNING

Machine Learning is one of efficient technology which is based on two terms namely testing and training i.e. system take training directly from data and experience and based on this training test should be applied on different type of need as per the algorithm required.

Types of Machine Learning are:

i. Supervised Learning
ii. Unsupervised Learning
iii. Reinforcement Learning



A.*Supervised Learning*

Supervised learning can be define as learning with the properguide or you can say that learning in the present of teacher .we have a training dataset which act as the teacher for prediction on the given dataset that is for testing a data there are always a training dataset. Supervised learning is based on "train me" concept. Supervised learning have following processes:

- Classification
- Random Forest
- Decision tree
- Regression

To recognize patterns and measures probability of uninterruptable outcomes, is phenomenon of regression. System have ability to identify numbers, their values and grouping sense of numbers which means width and height, etc. There are following supervised machine learning algorithms:

- Linear Regression
- Logistical Regression
- Support Vector Machines (SVM)
- Neural Networks
- Random Forest
- Gradient Boosted Trees
- Decision Trees
- Naive Bayes

B.*Unsupervised Learning*

Unsupervised learning can be define as the learning without aguidance which in Unsupervised learning there are no teacher are guiding. In Unsupervised learning when a dataset is given it automatically work on the dataset and find the pattern and relationship between them and according to the created relationships, when new data is given it classify them and store in one of them relation . Unsupervised learning is based on "self-sufficient" concept.

For example suppose there are combination fruits mango, banana and apple and when Unsupervised learning is applied it classify them in three different clusters on the basis if there relation with each other and when a new data is given it automatically send it to one of the cluster.

Supervisor learning say there are mango, banana and apple but unsupervised learning said it as there are three different clusters. Unsupervised algorithms have following process:

- Dimensionality
- Clustering

There are following unsupervised machine learning algorithms:

- t-SNE
- k-means clustering
- PCA

C. Reinforcement

Reinforced learning is the agent ability to interact with the environment and find out the outcome. It is based on "hit and trial" concept. In reinforced learning each agent is awarded with positive and negative points and on the basis of positive points reinforced learning give the dataset output that is on the basis of positive awards it trained and on the basis of this training perform the testing on datasets

III. *RELATED WORK*

Heart is one of the core organ of human body, it play crucial role on blood pumping in human body which is as essential as the oxygen for human body so there is always need of protection of it, this is one of the big reasons for the researchers to work on this. So there are number of researchers working on it .There is always need of analysis of heart related things either diagnosis or prediction or you can say that protection of heart disease .There are various fields like artificial intelligence, machine learning, data mining that contributed on this work .

[1] Senthilkumar Mohan, Chandrasegar Thiurumalai, and Gautam Srivastava "Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques": The prediction model is introduced with different combinations of features and several known classification techniques. It produced an enhanced performance level with an accuracy level of 88.7% through the prediction model for heart disease with the hybrid random forest with a linear model (HRFLM)

[2] Rohit Bharti, Aditya Khamparia, Mohammad Shabaz, Gaurav Dhiman, Sagar Pande, and Parneet Singh "Prediction of Heart Disease Using a Combination of Machine Learning and Deep Learning" :The dataset consists of 14 main attributes used for performing the analysis. Various promising results are achieved and are validated using accuracy and confusion matrix. Using deep learning approach, 88.2% accuracy was obtained. It was also found out that the statistical analysis is also important when a dataset is analyzed and it should have a Gaussian distribution, and then the outlier's detection is also important and a technique known as Isolation Forest is used for handling this. The difficulty which came here is that the sample size of the dataset is not large. If a large dataset is present, the results can increase very much in deep learning and ML as well. The algorithm applied by us in ANN architecture increased the accuracy which we compared with the different researchers

[3] Harshit Jindal "Heart disease prediction using machine learning algorithms": A quite helpful approach was used to regulate how the model can be used to improve the accuracy of prediction of Heart Attack in any individual. The strength of the proposed model was quiet satisfying and was able to predict evidence of having a heart disease in a particular individual by using KNN and Logistic Regression which showed a good accuracy in comparison to the previously used classifier such as naive bays etc. Most efficient of these algorithms is KNN which gives us the accuracy of 88.52%. And, finally we classify patients that are at risk of getting a heart disease or not and also this method is totally cost efficient.

[4] Lakshmana Rao et al, proposed "Machine Learning Techniques for Heart Disease Prediction" in which the contributing elements for heart disease are more. So, it is difficult to distinguish heart disease. To find the seriousness of the heart disease among people different neural systems and data mining techniques are used.

[5] Using the similar dataset of Framingham, Massachusetts, the experiments were carried out using 4 models and were trained and tested with maximum accuracy K-Neighbors Classifier: 87%, Support Vector Classifier: 83%, Decision Tree Classifier: 79% and Random Forest Classifier: 84%.

[6] Anjan N. Repaka et al, proposed a model stated the performance of prediction for two classification models, which is analyzed and compared to previous work. The experimental results show that accuracy is improved in finding the percentage of risk prediction of our proposed method in comparison with other models.

[7] Avinash Golande et al, proposed "Heart Disease Prediction Using Effective Machine Learning Techniques" in which few data mining techniques are used that support the doctors to differentiate the heart disease.

IV METHODOLOGY

The dataset used for this research purpose was the Public Health Dataset and it is dating from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer-valued 0 = no disease and 1 = disease. The first four rows and all the dataset features are shown in Table 1 without any preprocessing. Now the attributes which are used in this research purpose are described as follows and for what they are used or resemble:

S. No.	Attribute	Description	Type
1	Age	Patient's age (29 to 77)	Numerical
2	Sex	Gender of patient(male-1 female-0)	Nominal
3	Cp	Chest pain type	Nominal
4	Trestbps	Resting blood pressure(in mm Hg on admission to hospital ,values from 94 to 200)	Numerical
5	Chol	Serum cholesterol in mg/dl, values from 126 to 564)	Numerical
6	Fbs	Fasting blood sugar>120 mg/dl, true-1 false-0)	Nominal
7	restecg	Resting electrocardiographic result (0 to 1)	Nominal
8	thalach	Maximum heart rate achieved(71 to 202)	Numerical
9	exang	Exercise included agina(1-yes 0-no)	Nominal
10	Oldpeak	ST depression introduced by exercise relative to rest (0 to .2)	Numerical
11	Slope	The slop of the peak exercise ST segment (0 to 1)	Nominal
12	Ca	Number of major vessels (0-3)	Numerical
13	thal	3-normal	Nominal
14	Target	1 or 0	Nominal

The working of the system starts with the collection of data and selecting the important attributes. Then the required data is preprocessed into the required format. The data is then divided into two parts training and testing data. The algorithms are applied and the model is trained using the training data. The accuracy of the system is obtained by
testing the system using the testing data.

A. *Data Collection*

First step for predication system is data collection and deciding about the training and testing dataset. In this project we have used 73% training dataset and 37% dataset used as testing dataset the system.

B. *Attribute Selection*

Attribute of dataset are property of dataset which are used for system and for heart many attributes are like heart bit rate of person, gender of the person, age of the person and many more shown in TABLE.1 for predication system.

C. *Preprocessing of data*

Preprocessing needed for achieving prestigious result from the machine learning algorithms. For example Random forest algorithm does not support null values dataset and for this we have to manage null values from original raw data. For our project we have to convert some categorized value by dummy value means in the form of "0"and "1" .

D. *Data Balancing*

Data balancing is essential for accurate result because by data balancing graph we can see that both the target classes are equal. Fig.3 represents the target classes where "0" represents with heart diseases patient and "1" represents no heart diseases patients.

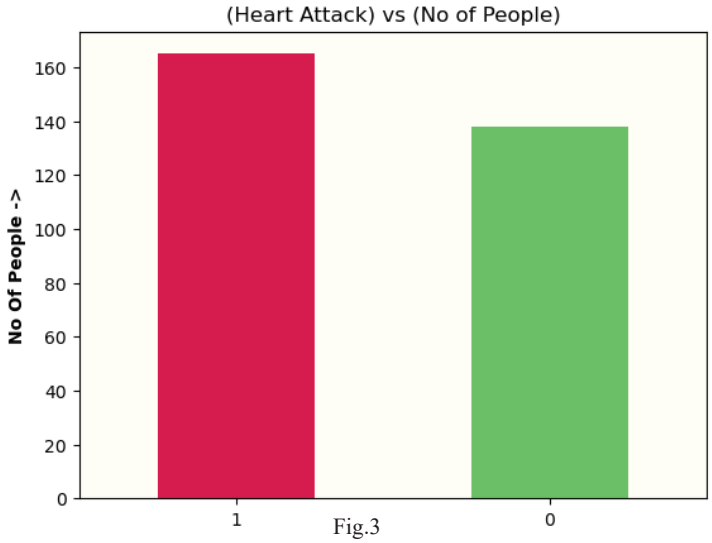


Fig.3

E. *Prediction of Disease*

Various machine learning algorithms like SVM, Naive Bayes, Decision Tree, Random Tree, Logistic Regression, K-nearest neighbor (KNN) are used for classification. Comparative analysis is performed among algorithms and the algorithm that gives the highest accuracy is used for heart disease prediction.

V. MACHINE LEARNING ALGORITHMS

Machine Learning relies on different algorithms to solve data problems. Data scientists like to point out that there's no single one-size-fits-all type of algorithm that is best to solve a problem. The kind of algorithm employed depends on the kind of problem you wish to solve, the number of variables, the kind of model that would suit it
best and so on.

A. *Logistic Regression*

Logistic regression is also a supervised learning classification algorithm that is used to solve both classification and regression problems. In classification problems, the target variable may be in a binary or discrete format either 0 or 1. Logistic regression algorithm works on the sigmoid function, so the categorical variable results as 0 or 1, Yes or No, True or False, etc. It is a predictive analysis algorithm that works on mathematical functions. Logistic regression uses a sigmoid function or logistic function which is a complex cost function. The sigmoid functions return the value between 0 and 1. If the value less than 0.5 then it is considered as 0 and greater than 0.5 it is considered as 1. Thus to build a model using logistic regression sigmoid function is required.

1) Binomial: The target variable can have only 2 possibilities either “0” or “1” which may represent “win” or “loss”, “pass” or “fail”, “true” or “false”, etc.

2) Multinomial: Here, the target variable can have 3 or more possibilities that are not ordered which means it has no measure in quantity like “disease A” or “disease B” or “disease C”.

3) Ordinal: In this case, the target variables deal with ordered categories. For example, a test score can be categorized as: “poor”, “average”, “good”, and “excellent”. Here, each category can be given a score like 0, 1, 2, and 3.

$$f(x) = \frac{1}{(1+e^{X})}$$

f(x) = Output between the 0 and 1 value
e = base of the natural logarithm
x = input to the function.

The value of the logistic regression must range from 0 to 1, does not go beyond this limit, so the only possible curve formed is S-shaped. The S-form curve formed is known as the sigmoid function or the logistic function. In logistic regression, the threshold value plays an important role, which defines the probability of either 0 or 1. The values above the threshold value reach 1, and a value below the threshold value reaches 0.

B. *Random Forest*

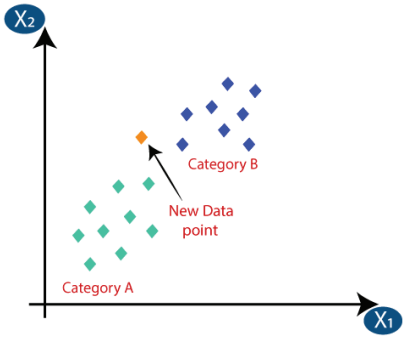
Random Forest classifier is a supervised learning technique in machine learning. It can be used to solve both Classification and Regression problems in machine learning. It is based on the process of combining multiple classifiers to solve a complex problem and to improve the performance of the model, which is known as ensemble learning. Random Forest consists of several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Rather than relying on a single decision tree, the random forest acquires the prediction from each tree, and based on the majority of votes for predictions, it predicts the final output. The higher number of trees in the forest leads to better accuracy and also prevents the problem of over fitting. The final output is taken by using the majority voting classifier for a classification problem while in the case of a regression problem the final output is the mean of all the outputs.

C. *Ada-boost*

AdaBoost is short for Adaptive Boosting and is a widely accepted boosting technique that combines multiple weak classifiers to build a strong classifier. It is done by building a model using a series of weak models. AdaBoost was developed for binary classification. It selects a training subset randomly and builds a model. It then iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training. It assigns the higher weight to the erroneously classified observations so that in the next iteration these observations would have a higher prospect for classification. This is done to correct the errors present in the first model. It also assigns weight to the trained classifier in every iteration according to the accuracy of the classifier. The more accurate classifier will get high weight. This process iterates until the entire training data fits without any error or until it reaches the specified maximum number of models are added.

D. *K-nearest Neighbor*

It work on the basis of distance between the location of data and on the basis of this distinct data are classified with each other. All the other group of data are called neighbor of each other and number of neighbor are decided by the user which play very crucial role in analysis of the dataset.



In the above Fig. k=3 shows that there are three neighbor that means three different type of data are there. Each cluster represented in two dimensional space whose coordinates are represented as (Xi,Yi) where Xi is the x-axis, Y represent y- axis and i= 1,2,3,….n.

V. PERFORMANCE ANALYSIS

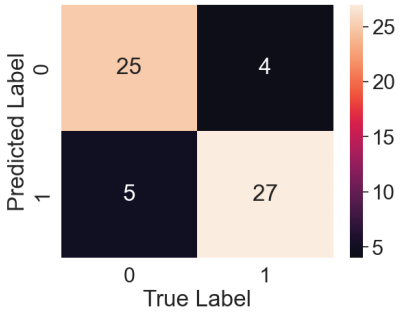
In this project, various machine learning algorithms like Random Forest, Logistic Regression,Adaboost, KNN are used to predict heart disease. Heart Disease UCI dataset, has a total of 76 attributes, out of those only 14 attributes are considered for the prediction of heart disease.

Various attributes of the patient like gender, chest pain type, fasting blood pressure, serum cholesterol, exang, etc. are considered for this project. The accuracy for individual algorithms has to measure and whichever algorithm is giving the best accuracy, that is considered for the heart disease prediction. For evaluating the experiment, various evaluation metrics like accuracy, confusion matrix, precision, recall, and f1-score are considered.

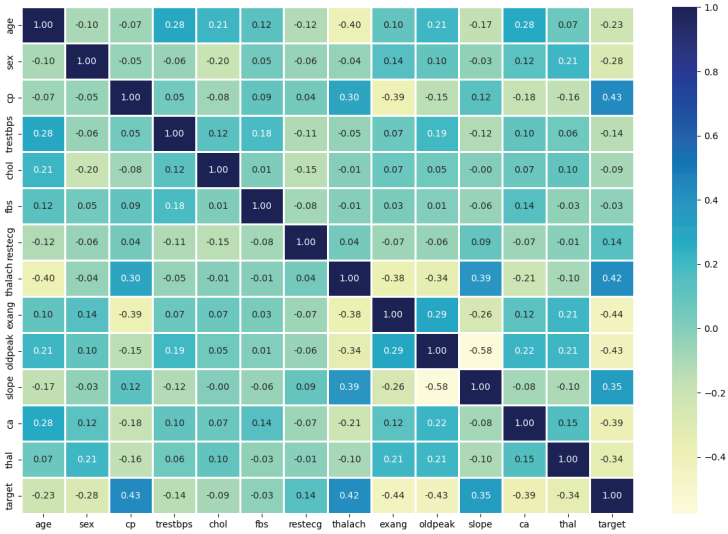
Accuracy- Accuracy is the ratio of the number of correct predictions to the total number of inputs in the dataset.

It is expressed as: $$\text{Accuracy} = \frac{(TP+TN)}{(TP+FP+FN+TN)}$$

Confusion Matrix- It gives us a matrix as output and gives the total performance of the system.



Correlation Matrix: The correlation matrix in machine learning is used For feature selection. It represents dependency between various attributes.



Precision- It is the ratio of correct positive results to the total number of positive results predicted by the system.

It is expressed as:

$$\text{Precision(P)} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

Recall- It is the ratio of correct positive results to the total number of positive results predicted by the system.

It is expressed as:

$$\text{Recall(R)} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

F1 score- It is the harmonic mean of Precision and Recall. It measures the test accuracy. The range of this metric is 0 to 1.

$$\text{F1 score} = 2 * \frac{1}{\left(\frac{1}{\text{Precision}}\right)+\left(\frac{1}{\text{Recall}}\right)} = \frac{2PR}{(P+R)}$$

VI. RESULT

After performing the machine learning approach for training and testing we find that accuracy of the adaboost is better compared to other algorithms. Accuracy is calculated with the support of the confusion matrix of each algorithm, here the number count of TP, TN, FP, FN is given and using the equation of accuracy, value has been calculated and it is concluded that extreme gradient boosting is best with 90.16% accuracy and the comparison is shown below.

Algorithm	Accuracy
KNN	68.85 %
Random Forest	85.25 %
Logistic Regression	88.52 %
AdaBoostClassifier	**90.16 %**

Table: Accuracy comparison of algorithm

The Highest accuracy is given by **AdaBoostAlgorithm**

VII. CONCLUSION AND FUTURE SCOPE

Heart diseases are a major killer in India and throughout the world, application of promising technology like machine learning to the initial prediction of heart diseases will have a profound impact on society. The early prognosis of heart disease can aid in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications, which can be a great milestone in the field of medicine. The number of people facing heart diseases is on a raise each year. This prompts for its early diagnosis and treatment. The utilization of suitable technology support in this regard can prove to be highly beneficial to the medical fraternity and patients. In this paper, the four different machine learning algorithms used to measure the performance are Random Forest, Logistic Regression, Adaptive Boosting, and K-Nearest Neighbor applied on the dataset. 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 taken into the consideration then the efficiency of the system the author gets is less. To increase efficiency, attribute selection is done. In this 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 four 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 four the adaptive boosting classifier gives the highest accuracy of 90%.

References

[1] Santhana Krishnan J and Geetha S, “Prediction of Heart Disease using Machine Learning Algorithms” ICIICT, 2019.

[2] Aditi Gavhane, Gouthami Kokkula, Isha Panday, Prof. Kailash Devadkar, “Prediction of Heart Disease using Machine Learning”, Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology(ICECA), 2018.

[3] Senthil kumar mohan, chandrasegar thirumalai and Gautam Srivastva, “Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques” IEEE Access 2019.

[4] Himanshu Sharma and M A Rizvi, “Prediction of Heart Disease using Machine Learning Algorithms: A Survey” International Journal on Recent and Innovation Trends in Computing and Communication Volume: 5 Issue: 8 , IJRITCC August 2017.

[5] M. Nikhil Kumar, K. V. S. Koushik, K. Deepak, “Prediction of Heart Diseases Using Data Mining and Machine Learning Algorithms and Tools” International Journal of Scientific Research in Computer Science, Engineering and Information Technology ,IJSRCSEIT 2019.

[6] Amandeep Kaur and Jyoti Arora,“Heart Diseases Prediction using Data Mining Techniques: A survey” International Journal of Advanced Research in Computer Science , IJARCS 2015-2019.

[7] Pahulpreet Singh Kohli and Shriya Arora, “Application of Machine Learning in Diseases Prediction”, 4th International Conference on Computing Communication And Automation(ICCCA), 2018.

[8] M. Akhil, B. L. Deekshatulu, and P. Chandra, “Classification of Heart Disease Using K- Nearest Neighbor and Genetic Algorithm,” Procedia Technol., vol. 10, pp. 85–94, 2013.

[9] S. Kumra, R. Saxena, and S. Mehta, “An Extensive Review on Swarm Robotics,” pp. 140–145, 2009.

[10] Hazra, A., Mandal, S., Gupta, A. and Mukherjee, “ A Heart Disease Diagnosis and Prediction Using Machine Learning and Data Mining Techniques: A Review” Advances in Computational Sciences and Technology , 2017.

REFERENCES

[1] Soni J, Ansari U, Sharma D & Soni S (2011). Predictive data mining for medical diagnosis: an overview of heart disease prediction. International Journalof Computer Applications, 17(8), 43-8

[2] Dangare C S & Apte S S (2012). Improved study of heart disease predictionsystem using data mining classification techniques. International Journal of Computer Applications, 47(10), 44-8.

[3] Ordonez C (2006). Association rule discovery with the train and test approach for heart disease prediction. IEEE Transactions on InformationTechnology in Biomedicine, 10(2), 334-43.

[4] Shinde R, Arjun S, Patil P & Waghmare J (2015). An intelligent heart disease prediction system using k-means clustering and Naïve Bayes algorithm. International Journal of Computer Science and Information Technologies, 6(1), 637-9.

[5] Bashir S, Qamar U & Javed M Y (2014, November). An ensemble-based decision support framework for intelligent heart disease diagnosis. In International Conference on Information Society (i-Society 2014) (pp. 259-64). IEEE. ICCRDA 2020 IOP Conf. Series: Materials Science and Engineering 1022(2021) 012072 IOP Publishing doi:10.1088/1757-899X/1022/1/012072 9

[6] Jee S H, Jang Y, Oh D J, Oh B H, Lee S H, Park S W & Yun Y D (2014). Acoronary heart disease prediction model: the Korean Heart Study. BMJ open, 4(5), e005025.

[7] Ganna A, Magnusson P K, Pedersen N L, de Faire U, Reilly M, Ärnlöv J &Ingelsson E (2013). Multilocus genetic risk scores for coronary heart disease prediction. Arteriosclerosis, thrombosis, and vascular biology, 33(9), 2267-72.

[8] Jabbar M A, Deekshatulu B L & Chandra P (2013, March). Heart diseaseprediction using lazy associative classification. In 2013 International MutliConference on Automation, Computing,Communication, Control and Compressed Sensing (iMac4s) (pp. 40- 6). IEEE.

[9] Brown N, Young T, Gray D, Skene A M & Hampton J R (1997). Inpatientdeaths from acute myocardial infarction, 1982-92: analysis of data in the Nottingham heart attack register. BMJ, 315(7101), 159-64.

[10] Folsom A R, Prineas R J, Kaye S A & Soler J T (1989). Body fat distribution and self-reported prevalence of hypertension, heart attack, and other heart disease in older women. International journal of epidemiologyy, 18(2),361-7.

[11] Chen A H, Huang S Y, Hong P S, Cheng C H & Lin E J (2011, September).HDPS: Heart disease prediction system. In 2011 Computing in Cardiology (pp. 557- 60). IEEE.

[12] Parthiban, Latha and R Subramanian. "Intelligent heart disease predictionsystem using CANFIS and genetic algorithm." International Journal of Biological, Biomedical and Medical Sciences 3.3 (2008).

[13] Wolgast G, Ehrenborg C, Israelsson A, Helander J, Johansson E & Manefjord H (2016). Wireless body area network for heart attack detection [Education Corner]. IEEE antennas and propagation magazine, 58(5), 84-92.

[14] Patel S & Chauhan Y (2014). Heart attack detection and medical attentionusing motion sensing device -kinect. International Journal of Scientific and Research Publications, 4(1), 1-4.

[15] Piller L B, Davis B R, Cutler J A, Cushman W C, Wright J T, Williamson JD & Haywood L J (2002). Validation of heart failure events in the Antihypertensive and Lipid Lowering Treatment to Prevent Heart Attack Trial (ALLHAT) participants assigned to doxazosin and chlorthalidone.
Current controlled trials in cardiovascular medicine

[16] Raihan M, Mondal S, More A, Sagor M O F, Sikder G, Majumder M A &Ghosh K (2016, December). Smartphone based ischemic heart disease (heart attack) risk prediction using clinical data and data mining approaches, a prototype design. In 2016 19th International Conference on Computer and Information Technology (ICCIT) (pp. 299-303). IEEE.

[17] A. Aldallal and A. A. A. Al-Moosa, "Using Data Mining Techniques toPredict Diabetes and Heart Diseases", 2018 4th International Conference on Frontiers of Signal Processing (ICFSP), pp. 150-154, 2018, September.

[18] Takci H (2018). Improvement of heart attack prediction by the featureselection methods. Turkish Journal of Electrical Engineering & Computer Sciences, 26(1), 1-10.

[19] Ankita Dewan and Meghna Sharma, "Prediction of heart disease using ahybrid technique in data mining classification", 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)

[20] Aditya Methaila, Prince Kansal, Himanshu Arya and Pankaj Kumar, "Earlyheart disease prediction using data mining techniques", Computer Science & Information Technology Journal, pp. 53-59, 2014.