

Industry Oriented Mini Project Report

On

Heart Disease Prediction using Machine Learning Techniques

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

of

Bachelor of Technology

In

Computer Science and Engineering

By

PANJALA THILAK (17241A0539)

Under the Guidance of

Dr. B. SANKARA BABU

Professor



Department of Computer Science and Engineering

**GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY**

(Autonomous)

Bachupally, Kukatpally, Hyderabad-500090.

2019-2020



**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING
AND TECHNOLOGY**

(Autonomous)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the mini project dissertation entitled “**Heart Disease Prediction using Machine Learning Techniques**” that is being submitted by

PANJALA THILAK (17241A0539)

PARELLY RAKESH KUMAR (17241A0541)

P. PAVAN VARMA (16241A05R7)

CH. SIDHARTH (16241A05P0)

in partial fulfillment of the requirement for the award of the degree in **Bachelor of Technology** in Computer Science and Engineering during the academic year 2019-2020.

Dr.B. Sankara Babu
Guide

Dr.K. Madhavi
Head of Department

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

There are many people who helped me directly and indirectly to complete my project successfully. We would like to take this opportunity to thank one and all.

First of all, we would like to express my deep gratitude towards my internal guide, **Dr. B. Sankara Babu, Professor**, for his for their help and constructive criticism in the completion of my dissertation.

We thank our mini Project coordinator **Dr. B. Sankara Babu** for his extensive Support. We would like to thank our faculty Coordinators **Ashlin Deepa R.N** and **G. Padmaja** for giving consistent suggestions and inputs during the project period.

We bestow our sincere thanks to our HOD, **Dr.K. Madhavi, Professor** for providing ample environment and exceptional encouragement to complete this project. We thank our Principal **Dr. J. Praveen** for providing the internal resources and facilities to complete the dissertation.

Finally, we very much indebted to our parent for their moral support and encouragement to achieve goals.

PANJALA THILAK

(17241A0539)

PARELLY RAKESH KUMAR

(17241A0541)

P. PAVAN VARMA

(16241A05R7)

CH. SIDHARTH

(16241A05P0)



DECLARATION

We hereby declare that the project work titled “ **Heart Disease Prediction using Machine Learning Techniques**” is the work done during the period from **09-12-2019 to 04-04-2020** and is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** from **Gokaraju Rangaraju Institute of Engineering and Technology** . The result embodied in this project have not been submitted to any other university or Institute for the award any degree or diploma.

PANJALA THILAK

(17241A0539)

PARELLY RAKESH KUMAR

(17241A0541)

P. PAVAN VARMA

(16241A05R7)

CH. SIDHARTH

(16241A05P0)

ABSTRACT

Cardiovascular diseases are a major cause of death globally. Application of Computer data processing in the Medical field has been witnessing several significant revolutions in recent days. Hospitals collect a huge amount of data on patients treated by heart diseases. But the data collected is not being used properly. This data collected can be used to predict the chance of patients being attacked by heart disease. Data Mining is the process used to extract the data. The data extracted is used to train the computer about the patterns of heart disease occurrence inpatient, and then used to predict the chance of a heart disease based on the lifestyle of a person.

This project proposes a prediction model to predict whether a people have a heart disease or not and to provide an awareness or diagnosis on that. This is done by comparing the accuracies of applying rules to the individual results of Support Vector Machine, Decision Trees, Random Forest, Naive Bayes classifier and KNN classifier on the dataset taken. By this project, we can build an accurate model of predicting cardiovascular disease

Keywords: Classification, Training, Testing.

CHAPTER	TITLE	PAGE No.
	Certificate	i-ii
	Acknowledgement	iii
	Declaration	iv
	Abstract	v
1	INTRODUCTION	1
	1.1 Introduction	1
	1.2 Domain Description	1
	1.3 Application	3
2	SYSTEM ANALYSIS	6
	2.1 Objective of the project	6
	2.2 Problem statement	6
	2.3 Feasibility Study	7
	2.4 Software/ Hardware Requirements	7
	2.5 Schematic Diagrams (Block diagrams and flowcharts)	8
3	IMPLEMENTATION	9
	3.1 Architecture	9
	3.2 Modules description	9
	3.3 Technology	12
	3.4 Algorithms	17
	3.5 Input and output analysis	29
4	CONCLUSION AND FUTURE ENHANCEMENT	30
	4.1 Conclusion	30
	4.2 Future enhancement	30

REFERENCES	31
APPENDICES	32
I.Plagiarism Report	32
II. Sample code	33
III. Screen shots	59

CHAPTER

1.INTRODUCTION

1.1 INTRODUCTION

The heart is a crucial organ of the physical body. It pumps blood to each a part of our anatomy. If it fails to function correctly, then the brain and various other organs will stop working, and within a few minutes, the person will die. Change in lifestyle, work related stress and bad food habits contribute to the increase in the rate of several heart-related diseases.

Heart diseases have emerged as one of the most prominent causes of death all round the world. According to the World Health Organization, heart-related diseases are responsible for taking 17.7 million lives every year, 31% of all global deaths. In India too, heart-related diseases became the leading explanation for mortality. Heart diseases have killed 1.7 million Indians in 2016, according to the 2016 Global Burden of Disease Report, released on September 15, 2017. Heart-related diseases increase the spending on health care and also reduce the productivity of a private. Estimates made by the World Health Organization (WHO), suggest that India have lost up to \$237 billion, from 2005-2015, due to heart related or cardiovascular diseases. Thus, a feasible and accurate prediction of heart-related diseases is very important.

Medical organizations, all around the world, collect data on various health-related issues. These data are often exploited using various machine learning techniques to realize useful insights. But the data collected is very massive and, many times, this data can be very noisy. These datasets, which are too overwhelming for human minds to grasp, are often easily explored using various machine learning techniques. Thus, these algorithms became very useful, in recent times, to predict the presence or absence of heart-related diseases accurately.

1.2 DOMAIN DESCRIPTION

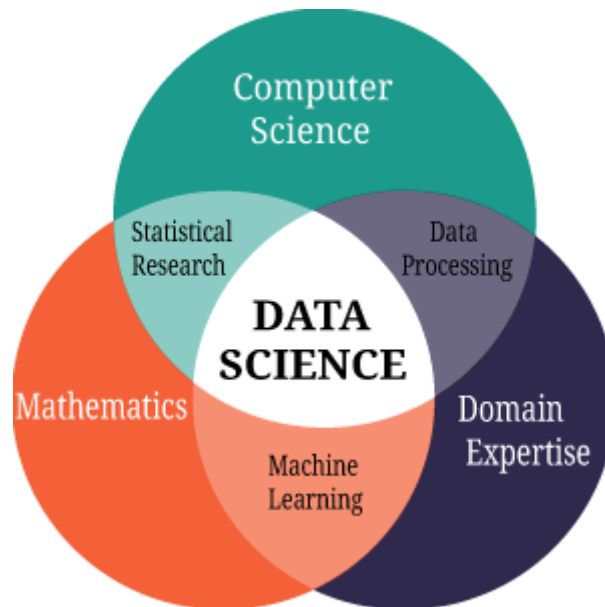
DATA SCIENCE

Data science could also be an idea to statistics, data analysis, machine learning and their related methods so on know and analyze actual phenomena with data. It employs techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, and knowledge science.



DATA SCIENCE

In other words, data science stands for an all-inclusive term that consists of aspects of ML for functionality. Interestingly, ML is additionally a component of AI , where a various set of purpose is achieved on a whole new level. ML and AI are parts of data science[1].



MACHINE LEARNING

The area of Machine Learning deals with the planning of programs which will learn rules from data, adapt to changes, and improve performance with experience. In addition to being one among the initial dreams of computer science, Machine Learning has become crucial as computers are expected to resolve increasingly complex problems and become more integrated into our daily lives. Writing a computer program is a bit like writing down instructions for an extremely literal child who just happens to be millions of times faster than you. Yet many of the problems we now want computers to solve are no longer tasks we know how to explicitly tell a computer how to do. These include identifying faces in images, autonomous driving in the desert, finding relevant documents in a database (or throwing out irrelevant ones, such as spam email), finding patterns in large volumes of scientific data, and adjusting internal parameters of systems to optimize performance. That is, we may ourselves be good at identifying people in photographs, but we do not know how to directly tell a computer how to do it. Instead, methods that take labeled training data (images labeled by who is in them, or email messages labeled by whether or not they are spam) and then learn appropriate rules from the data, seem to be the best approaches to solving these problems. Furthermore, we need systems that can adapt to changing conditions, that can be user-friendly by adapting to needs of their individual users, and that can improve performance over time.

It is very difficult to cater to all or any the choices supported all possible inputs. To tackle this problem, algorithms are developed. These algorithms build knowledge from specific data and past experience with the principles of statistics, applied mathematics , logic, combinatorial optimization, search, reinforcement learning, and control theory.

The developed algorithms form the idea of varied applications such as:

- Vision processing
- Language processing
- Forecasting (e.g., stock market trends)
- Pattern recognition
- Games
- Data mining
- Expert systems
- Robotics



1.3 APPLICATIONS

Machine learning Steps Machine Learning Applications

Let us discuss all the real-world Machine Learning Applications one by one.

Image Recognition
One of the most common uses of machine learning is image recognition. There are many situations where you'll classify the thing as a digital image. For digital images, the measurements describe the outputs of every pixel within the image. In the case of a black and white image, the intensity of every pixel is one measurement. So, if a black and white image has $N \times N$ pixels, the overall number of pixels and hence measurement is N^2 . In the colored image, each pixel is considered as providing 3 measurements to the intensities of three main color components i.e. RGB. So, $N \times N$ colored image there are three N^2 measurements.

- for face detection – The categories might be face versus no face present. There might be a separate category for each person in a data base of several individuals.
- For character recognition – We can segment a piece of writing into smaller images, each containing a single character. The categories might contain the 26 letters of English alphabet, the ten digits, and a few special characters.

Speech Recognition
Speech recognition (SR) is that the translation of spoken words into text. It is also referred to as “automatic speech recognition” (ASR), “computer speech recognition”, or “speech to text” (STT).

In speech recognition, a software application recognizes spoken words by the user. The measurements during this application could be a collection of numbers that represent the speech signal. We can segment the signal into parts that contain distinct words or phonemes. In each segment, we will represent the speech signal by the intensities or energy in several time-frequency bands.

Although the small print of signal representation are outside the scope of this program, we will represent the signal by a group of real values. Speech recognition applications include voice user interfaces. Voice user interfaces are such as voice dialing, call routing, domestic appliance control. It also can use as simple data entry, preparation of structured documents, speech-to-text processing, and more. Classification A Classification may be a process of placing each individual from the population under study in many classes. This is identified as independent variables. Classification helps analysts to use measurements of an object to spot the category to which that object belongs. To establish an efficient rule, analysts use data. Data consists of the many samples of objects with their correct classification.

For example, before a bank decides to disburse a loan, it assesses customers on their ability to repay the loan. By considering factors like customer's earning, age, savings and financial history we will roll in the hay. This information taken from the past data of the loan. Hence, Seeker uses to make relationship between customer attributes and related risks.

PREDICTION

Consider the instance of a bank computing the probability of any of loan applicants faulting the loan repayment. To compute the probability of the fault, the system will first got to classify the available data in certain groups. It is described by a set of rules prescribed by the analysts.

Once we do the classification, as per need we will compute the probability. These probability computations can compute across all sectors for varied purpose. Current prediction is one among the most well liked machine learning algorithms. Information Extraction (IE) is another application of machine learning. It is the process of extracting structured information from unstructured data. For example, web pages, articles, blogs, business reports, and e-mails. The electronic database maintains the output produced by the knowledge extraction.

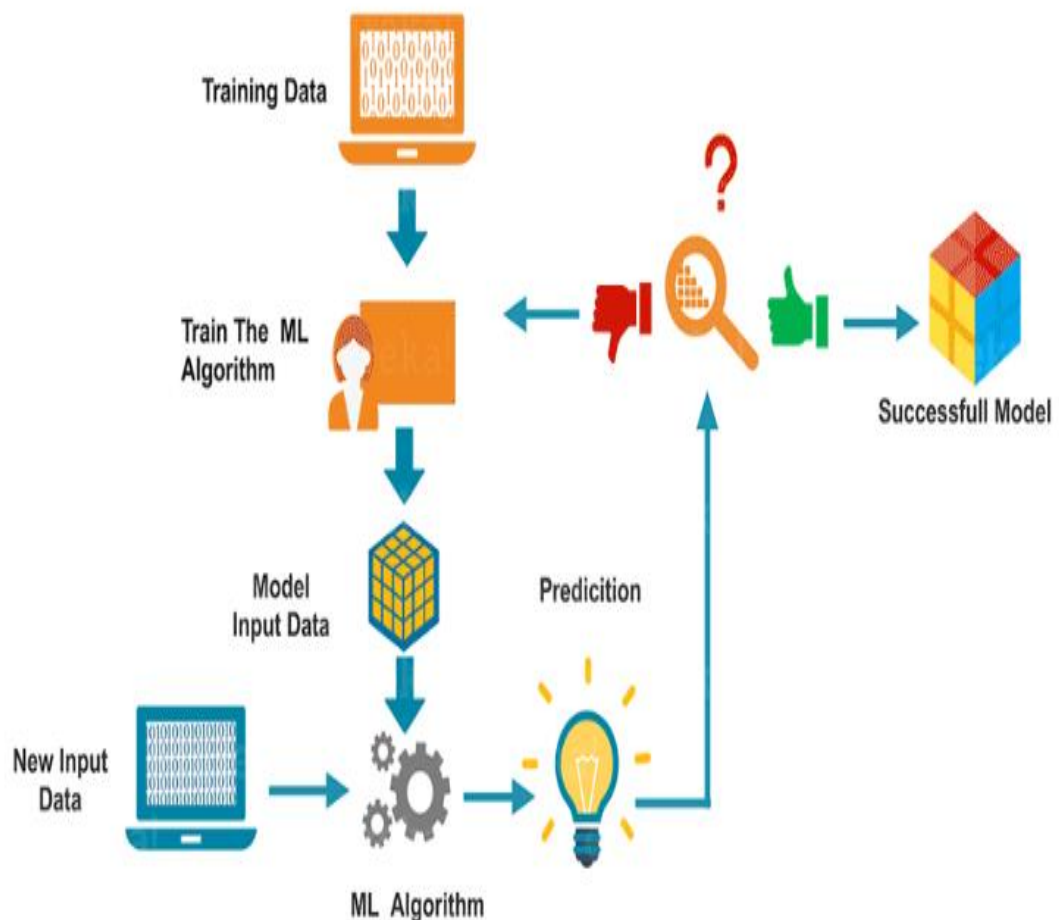
The process of extraction takes input as a group of documents and produces a structured data. This output is in summarized form such as excel sheet and table in a relational database. Now-a-days extraction is becoming a key in big data industry.

As we know that huge volume of data is getting generated out of which most of the data is unstructured. The first key challenge is handling of unstructured data. Now conversion of unstructured data to structured form supported some pattern in order that an equivalent can stored in RDBMS. Regression we will apply Machine learning to regression also .

Assume that $x = x_1, x_2, x_3, \dots, x_n$ are the input variables and y is that the outcome variable. In this case, we can use machine learning technology to produce the output (y) on the basis of the input variables (x). You can use a model to precise the connection between various parameters as below: $Y = g(x)$ where g may be a function that depends on specific characteristics of the model.

In regression, we will use the principle of machine learning to optimize the parameters. To cut the approximation error and we will calculate the closest possible outcome. We can also use ML for function optimization. We can choose to alter the inputs to get a better model. This gives a replacement and improved model to figure with. This is known as response surface design.

How Machine Learning Works?



CHAPTER 2

SYSTEM ANALYSIS

2.1 OBJECTIVE OF THE PROJECT

The main objective of this project is to develop a heart disease prediction system that can be used by the doctors in the medical field to predict the heart disease of a patient. Today, heart diseases are one of the leading causes of deaths and the best prevention for the disease is to have an early system that can predict the early symptoms which can save more lives. The use of data mining techniques helps the people in the medical field to predict the probability of getting heart disease among susceptible patients. The system can discover and extract hidden knowledge related to diseases from a historical heart data set heart condition prediction system that aims to take advantage of data processing techniques on medical data set to help within the prediction of the heart diseases. This provides a new approach to the concealed patterns in the data. It also helps in avoiding human biases. This project also reduces the cost of medical tests. It will implement the machine learning algorithm that classifies the state of the heart as per the input of the user.

2.2 PROBLEM STATEMENT

Heart diseases are often managed effectively with a mixture of lifestyle changes, medicine and, in some cases, surgery. With the proper treatment, the symptoms of the heart condition are often reduced and therefore the functioning of the heart improved. The anticipated results are often wont to prevent and thus reduce the cost of surgery and other expenses. The general objective of my work is going to be to predict accurately with a few tests and attributes the presence of heart condition. Attributes considered form the first basis for tests and provide accurate results more or less. More input attributes are often taken but our goal is to predict with few attributes and faster efficiency the danger of getting heart condition. Decisions are often made supported doctors' intuition and knowledge instead of on the knowledge-rich data hidden within the data set and databases. This practice results in unwanted biases, errors and excessive medical costs which affect the standard of service provided to patients.

2.3 FEASIBILITY STUDY:

This assessment involves a cost analysis of the project, helping organizations determine the cost, viability, and benefits related to a project before financial resources are allocated. It also acts as an independent project assessment and it enhances the credibility of the project – helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide. Our project is economically feasible because we are using “PYTHON” and some external modules such as Numpy, matplotlib, scikit-learn, etc. which are all available as an open source.

2.4 SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

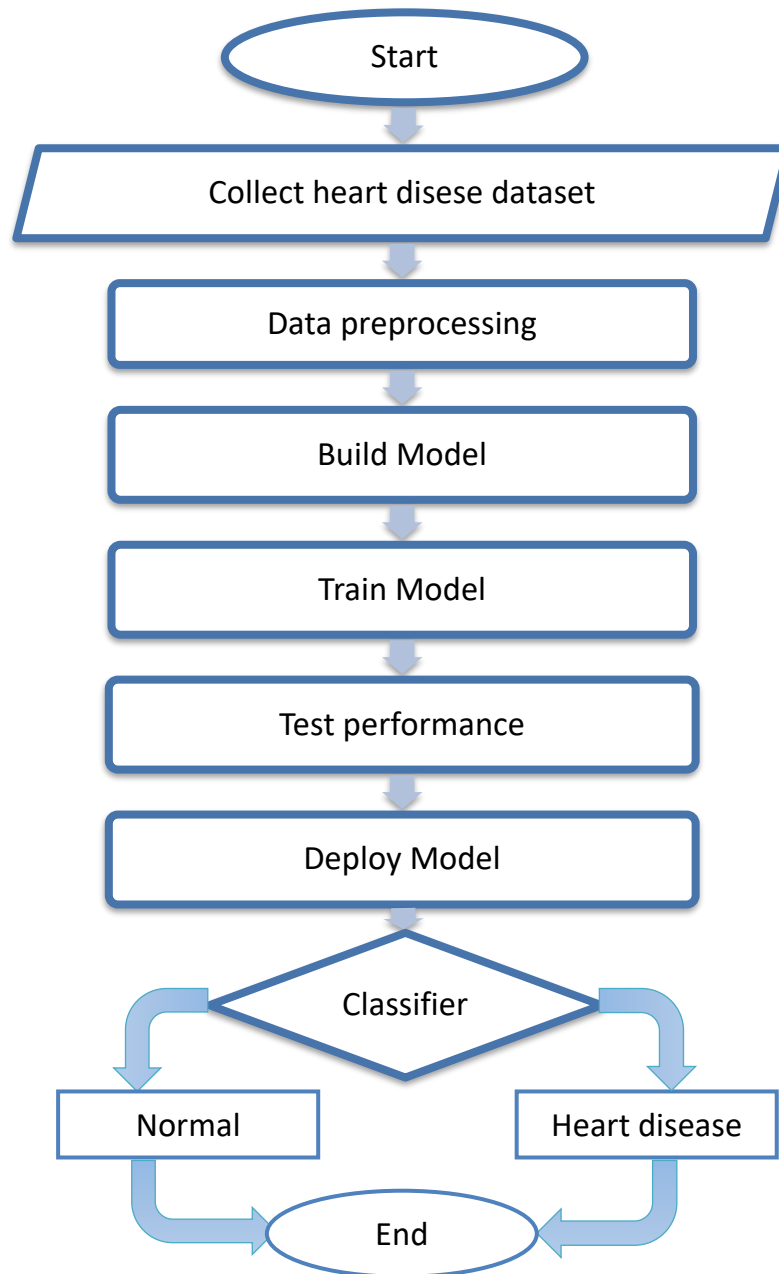
OPERATING SYSTEM	: Windows 7 or higher or Linux or Mac
PROGRAMMING LANGUAGE	: Python
TEXT EDITOR	: Anaconda
EXTERNAL MODULES	: numpy, scikit-learn, pandas, flask

HARDWARE REQUIREMENTS

PROCESSOR	: Intel core i3 processor or above
MEMORY	: 4 GB RAM
STORAGE	: 500 GB or above Hard disk drive

2.5 SCHEMATIC DIAGRAMS (BLOCK DIAGRAMS AND FLOW CHARTS)

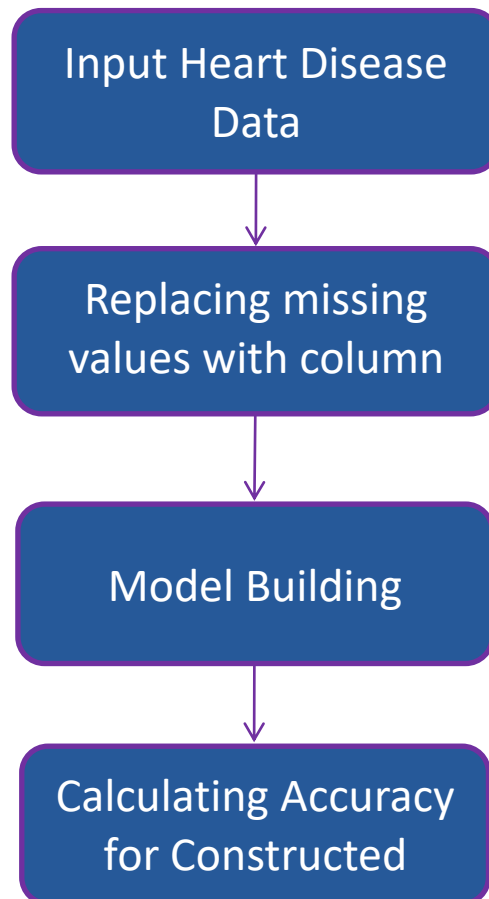
FLOW CHART



CHAPTER 3

IMPLEMENTATION

3.1 ARCHITECTURE



3.2 MODULES DESCRIPTION

DATA PREPARATION:

Data preparation is the process of loading our data into a suitable place and preparing it for use in our Machine learning training. We'll also need to split the data into two parts. The first part will be used in training our model. It will be the majority of the dataset. The second part is used for evaluating our trained model's performance. We don't want to use the same data that the model was trained on for evaluation, since it could then just memorize. Sometimes the data needs other forms of adjusting and manipulation, which can be done by the process of Data cleaning. Data cleaning is the task of removing errors and anomalies or replacing observed values with true values from data to get more value in analytics. There are the normal sorts of data cleaning like imputing missing data and data transformations and

there also more complex data unification problems like deduplication and repairing integrity constraint violations. Record linkage is where multiple mentions of an equivalent real-world entity appear across the data. Missing data refers to values which are missing from a dataset. Missing value imputation is the process of replacing missing data with substituted values. In practice, the matter is more complicated because missing data isn't represented by Nulls but instead by garbage. Integrity constraints make sure that data follow functional dependencies and business rules that dictate what values can legally exist together. The main aim of Data Cleaning is to spot and take away errors & duplicate data, to make a reliable dataset. This improves the quality of the training data for analytics and enables accurate decision-making. These would all happen in the data preparation stage and now the data is ready for training.

TRAINING:

Training a model simply means learning and determining good values for all the weights and the bias from labeled examples. The process of training a Machine Learning model involves providing a Machine Learning algorithm with training data to learn from. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to seek out a model that minimizes loss; this process is named empirical risk minimization. During training, data is evaluated by the ML algorithm, which analyzes the distribution and type of data, trying to find rules and patterns that are used for later prediction. The term Machine Learning model refers to the model artifact that is created by the training process. The training data must contain the right answer, which is understood as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target, and it outputs a Machine Learning model that captures these patterns. You can use the Machine Learning model to urge predictions on new data that you are doing not know the target.

TESTING:

Once training is complete, it's time to see if the model is any good, using Testing. This is where that dataset that we put aside earlier comes into play. A good training-testing split will be somewhere on the order of 80/20 or 70/30 Testing allows us to test our model against testing dataset. This metric allows us to ascertain how the model might perform against data that it's not yet seen. This is meant to be representative of how the model might perform within the real world.

CALCULATING THE ACCURACY:

Evaluating your machine learning algorithm is an important part of any project. Accuracy is one metric for evaluating classification models. Informally, accuracy is defined as the fraction of predictions our model got right. Formally, accuracy has the following definition: $\text{Accuracy} = \frac{\text{Number of correct predictions (True)}}{\text{Total number of predictions}}$

Your model may offer you satisfying results when evaluated employing a metric say accuracy score but may give poor results when evaluated against other metrics like logarithmic loss or any other such metric. Most of the time, we use classification accuracy to measure the performance of the model.

Classification Accuracy:

Classification Accuracy is what we usually mean, when using the term accuracy. It is the ratio of the number of correct predictions to the total number of input samples. It works well only if there are an equal number of samples belonging to every class. For example, consider that there are 98% of samples belonging to class A and 2% of samples of class B in our training set. Then our model can easily get 98% training accuracy by predicting every training sample as class A. When the same model is tested on a test set with 60% samples of class A and 40% samples of class B, then the test accuracy would drop to 60%. Classification Accuracy is great, but it gives us a false sense of achieving high accuracy.

The real problem arises, when the cost of misclassification of the minor class samples is very high. If we deal with a rare but fatal disease, the cost of failing to diagnose the disease of a diseased person is far above the cost of sending a healthy person to more tests.

Confusion Matrix:

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and counteracted by each class. This is the key to the confusion matrix. The confusion matrix shows the ways during which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the kinds of errors that are being made.

Here,

- Class 1 : Positive
- Class 2 : Negative

Definition of the Terms:

- Positive (P) : Observation is positive.
- Negative (N) : Observation is not positive.
- True Positive (TP) : Observation is positive, and the prediction made is positive.
- False Negative (FN) : Observation is positive, but the prediction made is negative.
- True Negative (TN) : Observation is negative, and the prediction made is negative.
- False Positive (FP) : Observation is negative, but the prediction made is positive.

Classification Rate/Accuracy: Classification Accuracy is given by the relation.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

However, there are problems with accuracy. It assumes equal costs for both sorts of errors. A 99% accuracy can be good, mediocre, excellent, poor or terrible depending upon the problem.

Recall: Recall is defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates that the class is recognized correctly (a small number of FN). Recall is given by the relation:

$$Recall = \frac{TP}{TP + FN}$$

Precision: Precision can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of predicted positive examples. High Precision indicates that an example labeled as positive is actually positive (small number of FP). Precision is given by the relation:

$$Precision = \frac{TP}{TP + FP}$$

High recall, low precision: This means that majority of the positive examples are correctly recognized (low FN) but there are a lot of false positives.

Low recall, high precision: This shows that we miss many positive examples (high FN) but those we predict as positive are indeed positive (low FP).

3.3.TECHNOLOGY

3.3.1 PYTHON (Programming language)

Python is a high-level, interactive, interpreted and object-oriented scripting language and is designed to be highly readable. It uses English keywords often whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted, which means it is processed at runtime by the interpreter. You don't need to compile your program before you execute it. This is similar to PERL and PHP.
- Python is Interactive – You will be able to actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented – Python supports the Object-Oriented style of programming that encapsulates code within objects.
- Python is a Beginner's Language – It is a great language for novice programmers and can be used for the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed within the late eighties and early nineties by Guido van Rossum at the National Research Institute for Mathematics and Computer Science which is located in the Netherlands. It is derived from many other languages, including the languages like ABC, Modula-3, C, C++, Algol-68, SmallTalk, and UNIX shell and other scripting languages.

Python is copyrighted. The source code of Python is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds an important role in directing its progress.

Python Features

- Easy-to-learn – Python has fewer keywords, simple structure, and a clearly defined syntax. This allows the student to progress with the language quickly.
- Easy-to-read – Python code is more clearly defined and visual to the eyes.
- Easy-to-maintain – Python's source code is very easy-to-maintain.
- A broad standard library – Python's bulk of the library is extremely portable and cross-platform compatible with UNIX, Windows, and Macintosh.
- Interactive Mode – Python has support for an interactive mode that allows interactive testing and debugging of snippets of code.
- Portable – Python can run on an extensive range of hardware platforms and has an equivalent interface on all platforms.
- Extendable – you'll be able to add low-level modules to the Python interpreter. These modules enable programmers to feature to or customize their tools to be more efficient.
- Databases – Python provides interfaces to all significant commercial databases.
- GUI Programming – Python has support for Graphical-User-Interface applications that can be created and ported to many libraries, system calls and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- Scalable – Python provides a far better structure and support for big programs than shell scripting.

Apart from the above-mentioned features, Python has a considerable list of good features, few of them are listed below –

- It supports functional and structured programming methods as well as Object-Oriented Programming.
- It can also be used as a scripting language or can be compiled to byte-code to build large applications.
- It provides very high-level dynamic data types and also supports dynamic type checking.
- It also supports automatic garbage collection.
- It is easy to integrate it with C, C++, COM, ActiveX, CORBA, and Java.

PANDAS

Pandas is an open-source, BSD-licensed Python library providing high performance, easy-to-use data structures and data analysis tools for the Python. Python with Pandas is employed in a very wide selection of fields including academic and commercial domains including finance, economics, statistics, analytics, etc. In this tutorial, we are going to learn the various features of Python Pandas and the way to use them in practice.

Audience

This tutorial has been prepared for those who seek to study the fundamentals and various functions of Pandas. It will be specifically useful for people who are working with data cleansing and analysis. After completing this tutorial, you'll end up at a moderate level of expertise from where you'll take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of programming terminologies. A basic understanding of any of the programming languages could be a plus. Pandas library utilizes most of the functionalities of NumPy. It is suggested that you simply undergo our tutorial on NumPy before proceeding with this tutorial. You can access it from – NumPy Tutorial.

Pandas provides high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas came from the word Panel Data – an Econometrics from Multidimensional data.

Developer Wes McKinney started developing pandas in 2008, when in need of high performance, flexible tool for analysis of data.

Before the Pandas, Python was mostly used for data munging and preparation and had very little contribution to data analysis. Pandas solved this problem. Using Pandas, we are able to accomplish five typical steps within the processing and analysis of data, no matter the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is employed in a wide selection of fields including academic and commercial domains including finance, economics, statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, indexing and subsetting of large data sets.

- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High-performance merging and joining of data.
- Time Series functionality.

Matplot library

Matplotlib is a plotting library for Python which is used for plotting points on different graphs and its numerical mathematics extension NumPy. It provides an object-oriented application program Interface(API) for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is a procedural "pylab" interface formulated on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

Matplotlib is used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing features to control line styles, font properties, formatting axes, etc. It supports a very wide variety of graphs and plots such as histogram, bar charts, power spectra, error charts etc.

Matplotlib was originally written by John D. Hunter, has an operative development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in 2012.

Seaborn

Seaborn is a library for creating statistical graphics in Python. It is built on top of matplotlib and is closely integrated with pandas data structures.

Here are some of the functionalities that seaborn offers:

- A dataset-oriented API for inspecting relationships between multiple variables.
- Specialized support for using categorical variables to indicate observations or aggregate statistics.
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data.

- Automatic estimation and plotting of linear regression models for various kinds dependent variables.
- Convenient views on the overall structure of complex datasets.
- High-level abstractions for structuring multi-plot grids that allow you to easily build complex visualizations.
- Concise control over matplotlib figure styling with numerous built-in themes.
- Tools for selecting color palettes that faithfully reveal patterns in your data

Scikit-learn

Scikit-learn provides a variety of supervised and unsupervised learning algorithms via a uniform interface in Python.

It is licensed under a permissive simplified BSD license and it is distributed under many Linux distributions, encouraging academic and commercial use.

The library is made upon the SciPy (Scientific Python) that has to be installed before you can use scikit-learn. This stack that includes:

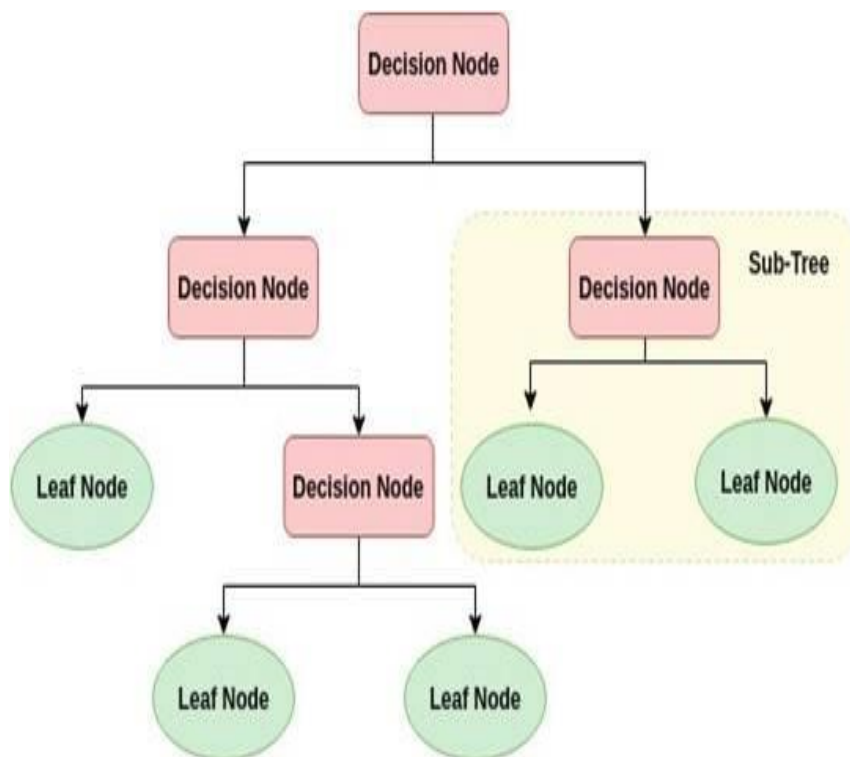
- NumPy: Base n-dimensional array package
- SciPy: Fundamental library for scientific computing
- Matplotlib: Comprehensive 2D/3D plotting
- IPython: Enhanced interactive console
- SymPy: Symbolic mathematics
- Pandas: Data structures and analysis

Extensions for SciPy are conventionally named SciKits. As such, the module provides learning algorithms and is known as scikit-learn. The vision for the library is a level of robustness and support required to be used in production systems. This means a deep focus on concerns like simple use, code quality, collaboration, documentation and performance. Although the interface is Python, c-libraries are leverage for performance like NumPy for arrays and matrix operations, LAPACK, LibSVM and therefore the careful use of Cython.

3.4 ALGORITHMS

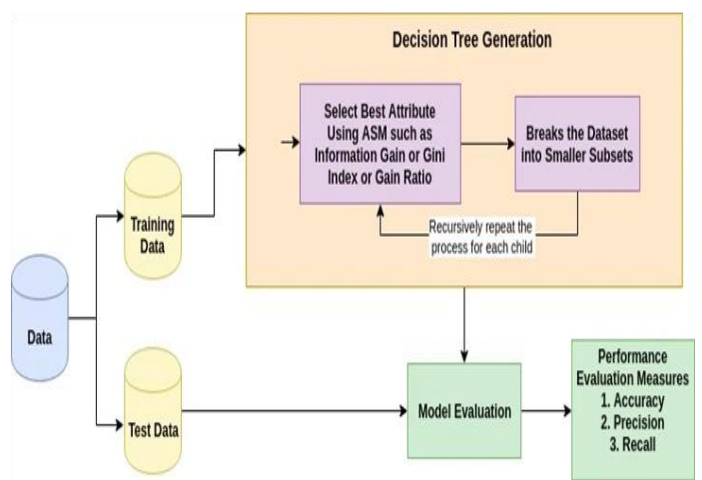
DECISION TREES

A decision tree is a flowchart which is in tree structure where an interior node represents feature (or attribute), the branch represents a decision rule, and every leaf node represents the end result. The topmost node during a decision tree is thought because of the root node. It learns to partition on the idea of the attribute value. It partitions the tree in a recursive manner called recursive partitioning. Its training time is quicker compared to the neural network algorithm. The decision does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with best accuracy[1].



The basic idea is as follows:

- Select the best attribute using Attribute Selection Measures (ASM) for splitting.
- Make that attribute as a decision node and breaks the dataset into smaller subsets.
- Start the tree building by repeating this process recursively for each child until one of the following condition will match:



- All the records belong to the same attribute value.
- There are no more remaining attributes.
- There are no more instances.

Attribute Selection Measures

Attribute selection measure is a heuristic for choosing the splitting criterion that partition data into the most effective possible manner. It is also referred to as splitting rules because it helps us to see breakpoints for tuples on a given node. ASM provides a rank to every feature (or attribute) by explaining the given dataset. Best score attribute will be selected as a splitting attribute. In the case of a continuous-valued attribute, split points for branches also need to define. Most popular selection measures are Information Gain, Gini Index, and Gain Ratio [1].

a) Information Gain:

- Information gain computes the difference between average entropy after split of the dataset based on given attribute values and entropy before split.

$$\text{Info}(D) = - \sum_{i=1}^m p_i \log_2 p_i$$

Where, P_i is that the probability that an arbitrary tuple in D belongs to class C_i .

$$\text{Info}_A(D) = \sum_{j=1}^V \frac{|D_j|}{|D|} \times \text{Info}(D_j)$$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

Where,

- $\text{Info}(D)$ is that the average amount of information needed to identify the category label of a tuple in D .
- $|D_j|/|D|$ acts as a weight of the j th partition.
- $\text{Info}_A(D)$ is that the expected information required to classify a tuple from D based on the partitioning by A .

The attribute A with the best information gain, $\text{Gain}(A)$, is chosen as the splitting attribute at node $N()$.

Iterative Dichotomiser (ID3) decision tree algorithm uses the information gain.

b) Gain Ratio:

- Information gain is biased for the attribute(column) with many outcomes. It means it prefers the attribute with an outsized number of distinct values.
- C4.5, an improvement of ID3, uses an extension to information gain called as the gain ratio. Gain ratio handles the problem of bias by normalizing the information gain using Split Info.

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

Where,

- $|D_j|/|D|$ acts as a weight of the jth partition.
- v is the number of discrete(different) values in attribute A.

The gain ratio can be defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

The attribute with the best gain ratio is chosen as the splitting attribute.

c) Gini index

Another decision tree algorithm CART (Classification and Regression Tree) uses the Gini method to make split points.

$$Gini(D) = 1 - \sum_{i=1}^m p_i^2$$

Where, p_i is that the probability that a tuple in D belongs to class C_i .

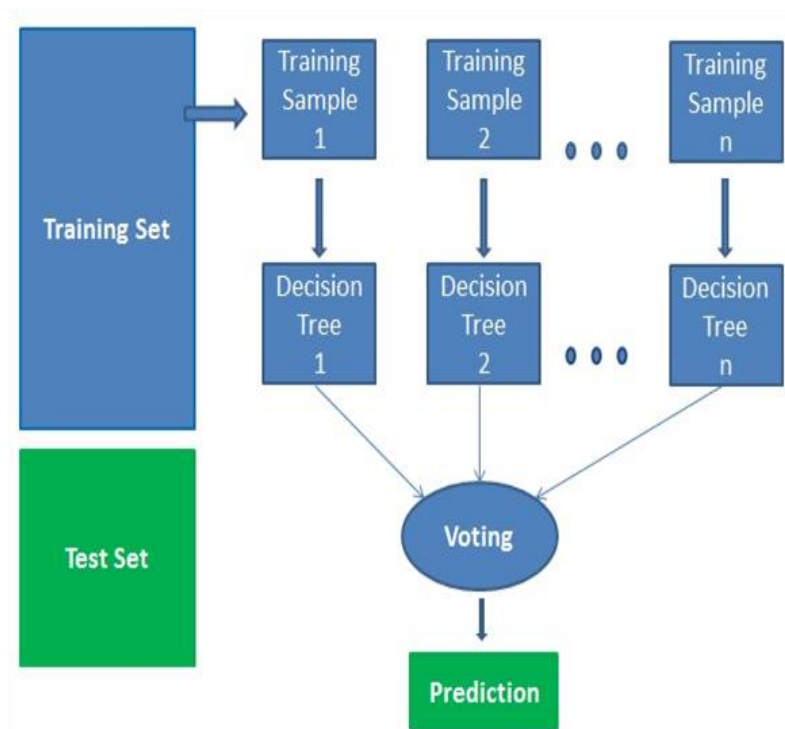
The Gini Index considers a binary split for every attribute. You can compute a weighted sum of the impurity of every partition. If a binary split on attribute A divides the data D into D1 and D2, the Gini index of D is:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

The attribute with the least Gini index is chosen as the splitting attribute.

RANDOM FOREST

Random forests is a supervised learning algorithm. It will be used both for classification and regression. A forest is comprised of trees. it's said that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, get a prediction from each tree and selects the most effective solution by means of voting. It also provides a fairly good indicator of the feature importance. It technically is an ensemble method of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is additionally referred to as the forest. Random forests include a variety of applications, like recommendation engines, image classification, and feature selection. It is used to identify fraudulent activity and predict diseases.



- Random forests is taken into account as a highly accurate and robust method due to the quantity of decision trees participating within the process.

It works in four steps:

- Select random samples from a given dataset.
- Construct a decision tree for every sample and obtain a prediction result from each decision tree.
- Perform a vote for every predicted result.
- Select the prediction result with the foremost votes as the final prediction.

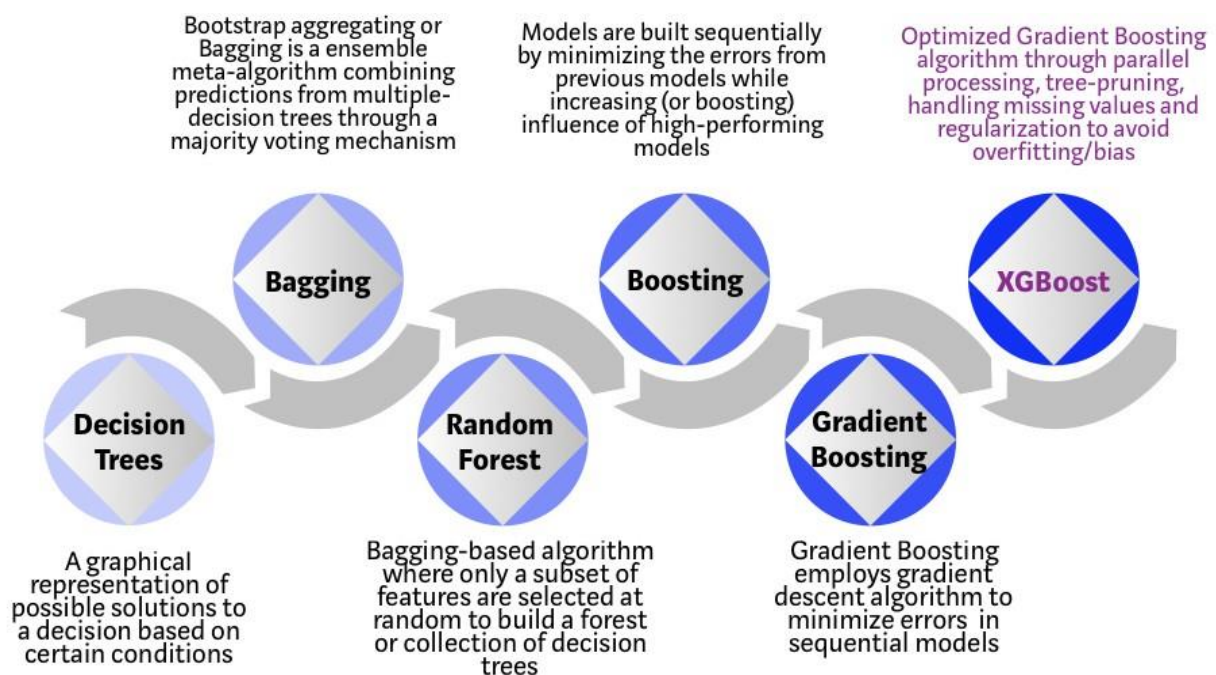
A very sizable amount of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is that the key. similar to how investments with low correlations (like stocks and bonds) close to make a portfolio that's greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the other individual predictions. the reason for this excellent effect is that the trees protect one another from their individual errors (as long as they don't constantly all err within the same direction). While some trees could also be wrong, many other trees are going to be right, so as a group the trees are ready to move within the correct direction. therefore the prerequisites for random forest to perform well are:

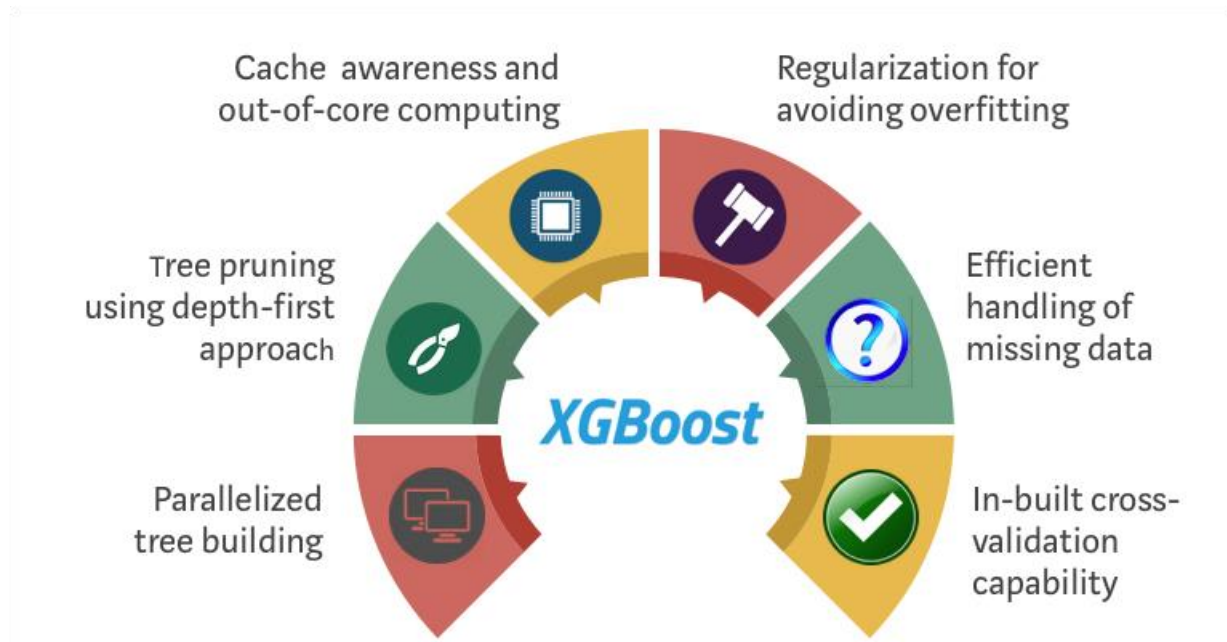
1. There must be some actual signal in our features in order that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with one another .

XGBoost

XGBoost is a ML algorithm which is decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks(ANN) tend to outperform all other algorithms or frameworks. However, when it involves small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class immediately .

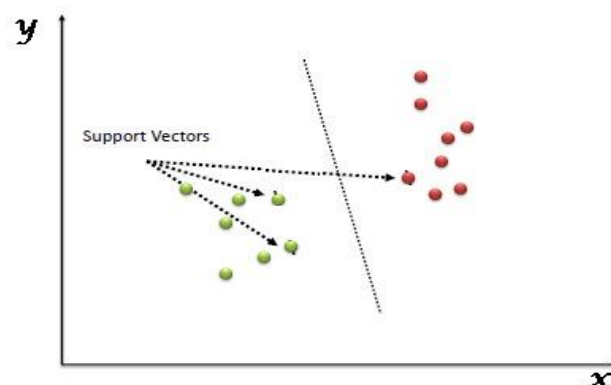


XGBoost and Gradient Boosting Machines both are the ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base Gradient Boosting Machines framework through systems optimization and algorithmic enhancements.



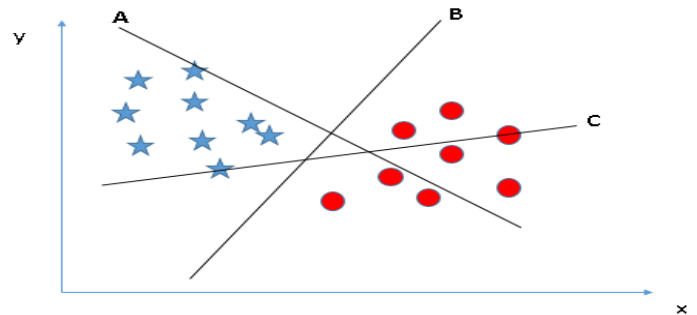
SUPPORT VECTOR MACHINE

Support Vector Machine is a supervised machine learning algorithm that may be used for both classification or regression challenges. However, it's mostly utilized in classification problems. In this algorithm, we plot each data item as a point in n -dimensional space (where n is number of features you have) with the worth of every feature being the worth of a specific coordinate. Then, we perform classification by finding the hyper-plane that differentiates or divides the 2 classes.



Support Vectors are the co-ordinates of individual observation. Support Vector Machine may be a frontier which best segregates the 2 classes (hyper-plane/ line).

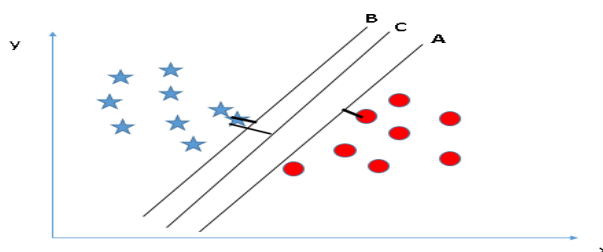
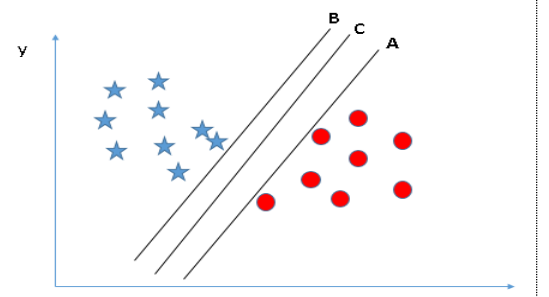
- **Identify the proper hyper-plane (Scenario-1):** Here, we've three hyper-planes (A, B and C). Now, identify the proper hyper-plane to classify star and circle.



You need to recollect a thumb rule to spot the proper hyper-plane: “Select the hyper-plane which segregates the 2 classes better”. during this scenario, hyper-plane “B” has excellently performed this job.

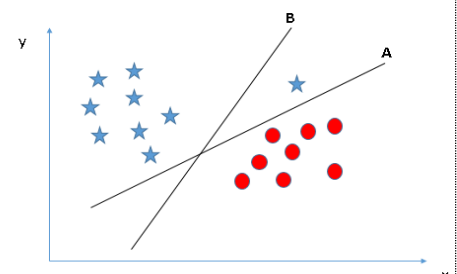
- **Identify the proper hyper-plane (Scenario-2):** Here, we've three hyper-planes (A, B and C) and everyone is segregating the classes well. Now, How can we identify the proper hyper-plane?

Here, maximizing the distances between the nearest datum (either class) and hyper-plane will help us to make a decision on the proper hyper-plane. This distance is named as Margin. Let's check out the below snapshot: Above, you'll see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the proper hyper-plane as C. Another



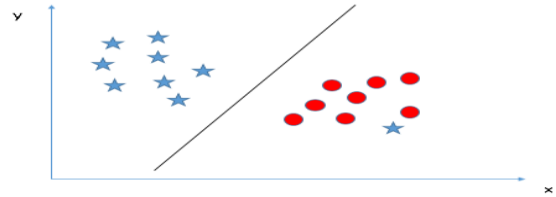
lightning reason for choosing the hyper-plane with higher margin is robustness. If we select a hyper-plane having a low margin then there's a high chance of miss-classification.

- **Identify the right hyper-plane (Scenario-3):** Hint: Use the principles as discussed in the previous section to spot the proper hyper-plane
Some of you'll have selected the hyper-plane B because it has a higher margin compared to A. But, here is that the catch, SVM selects the hyper-plane which classifies the classes accurately before maximizing margin. Here, hyper-plane B features a classification error and A has classified all correctly. Therefore, the proper hyper-plane may be A.



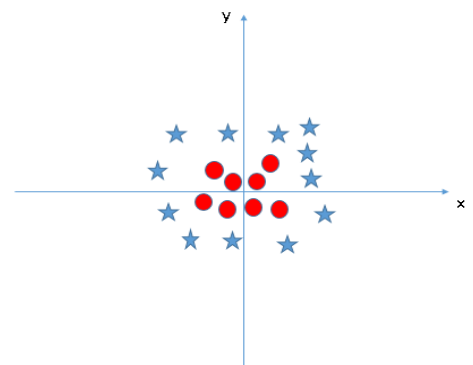
- **Can we classify two classes (Scenario-4)?:** Below, I'm unable to segregate the 2 classes employing a line, together of star lies within the territory of other(circle) class as an outlier.

As I even have already mentioned, one star at the other end is like an outlier for star class. SVM features a feature to ignore outliers and find the hyper-plane that has a maximum margin. Hence, we will say, SVM is strong to outliers



- **Find the hyper-plane to segregate to classes (Scenario-5):** within the scenario below, we can't have linear hyper-plane between the 2 classes, so how does SVM classify these two classes? Till now, we've only checked out the linear hyper-plane.

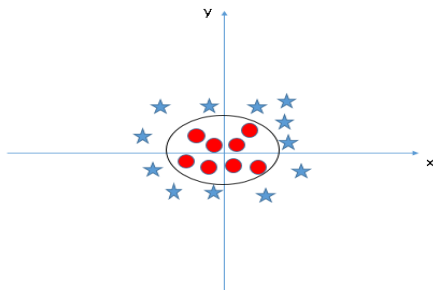
SVM can solve this problem. Easily! It solves this problem by introducing additional features. Here, we'll add a replacement feature $z = x^2 + y^2$. Now, let's plot the info points on axis x and z:



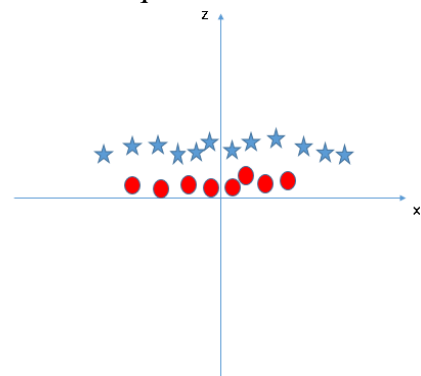
In the above plot, points to think about are:

o All values for z would be positive always because z is that the squared sum of both x and y

o In the first plot, red circles appear on the brink of the origin of x and y axes, resulting in a lower value of z and star relatively far away from the original result to the higher value of z .

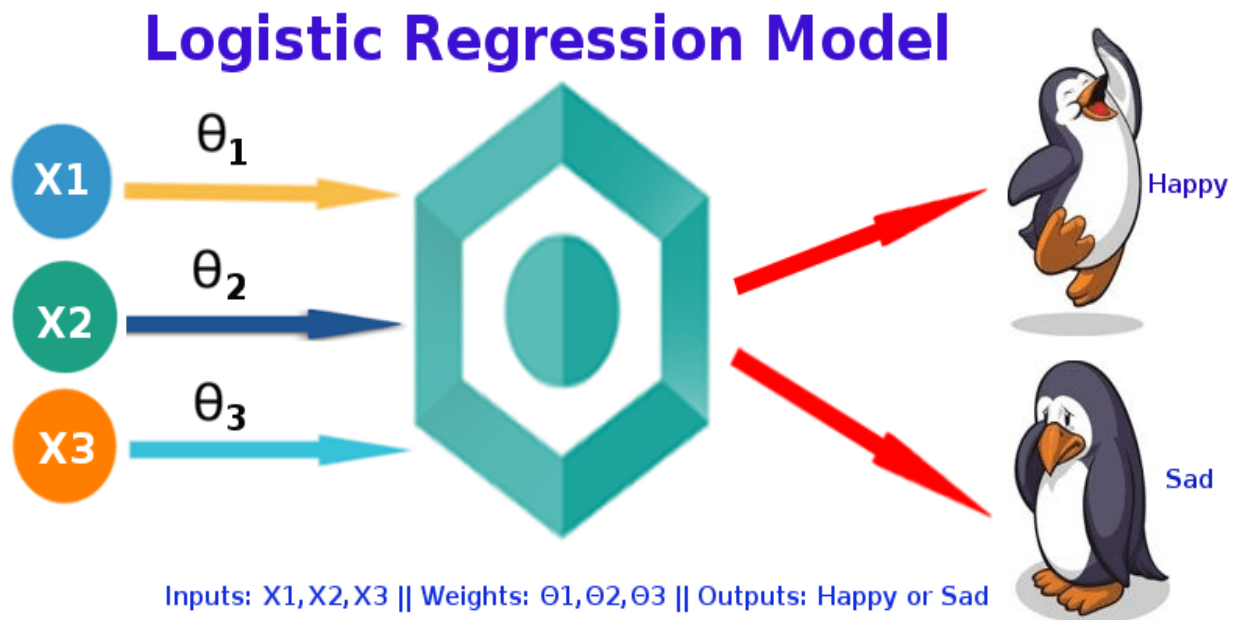


- In SVM, it's easy to possess a linear hyper-plane between these two classes. But, another burning question which arises is, should we'd like to feature this feature manually to possess a hyper-plane. No, SVM features a technique called the kernel trick. These are functions which take low dimensional input space and transform it into a better dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. it's mostly useful in non-linear separation problems. Simply put, it does some extremely complex data transformations, then determine the method to separate the info supported the labels or outputs you've defined. once we check out the hyper-plane in original input space it's sort of a circle:



LOGISTIC REGRESSION

Logistic Regression was a Machine Learning algorithm used in the biological sciences in the early twentieth century. It was then used in many social science applications also. Logistic Regression is used when the dependent target variable is categorical.



For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Consider a scenario where we want to classify whether an email is spam or not. If we use simple linear regression for this problem, there's a requirement for fitting a threshold based on which classification will be done. Say if the particular class is malignant, predicted continuous value 0.4 and also the threshold value is 0.5, the info point are going to be classified as not malignant which may result in serious consequence in real time.

From this instance , it will be inferred that simple regression isn't suitable for classification problem. Linear regression is unbounded, and this brings the logistic regression into the frame. Their value strictly ranges from 0 to 1. Simple Logistic Regression

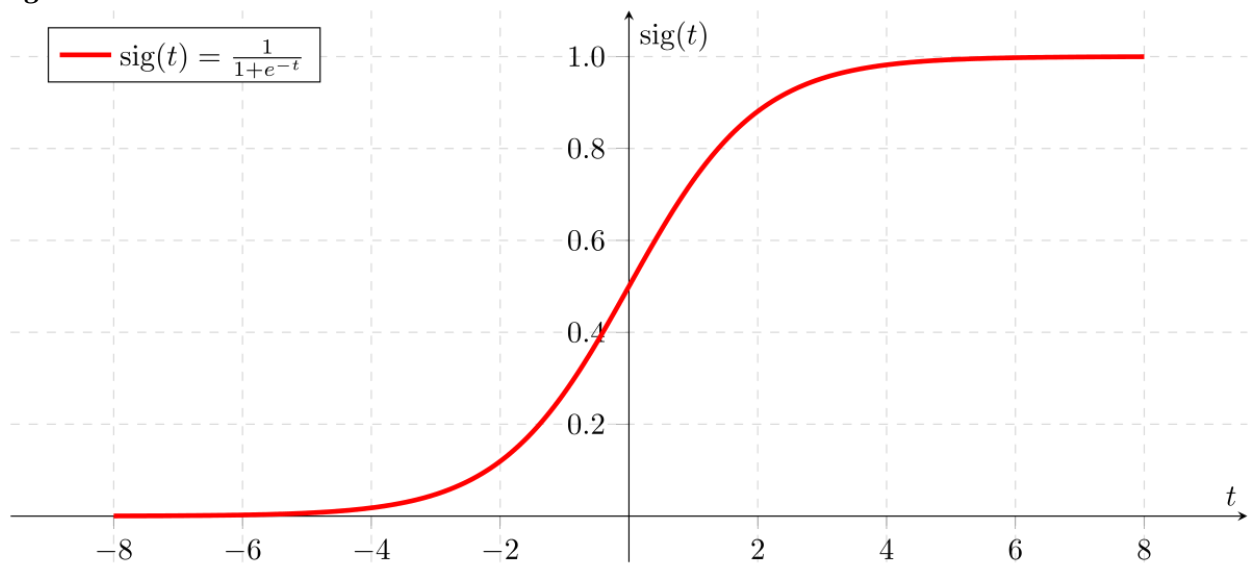
Model

Output = 0 or 1

Hypothesis $\Rightarrow Z = WX + B$

$h\Theta(x) = \text{sigmoid}(Z)$

Sigmoid Function



Sigmoid Activation Function

If 'Z' goes to infinity, Y(predicted) will become 1 and if 'Z' goes to negative infinity, Y(predicted) will become 0.

NAVIE BAYES

Navie Bayes is a classification technique based on Bayes Theorem which is used for the probability with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a specific feature in a class is unrelated to the presence of the other feature.

Naive Bayes algorithms are mostly utilized in

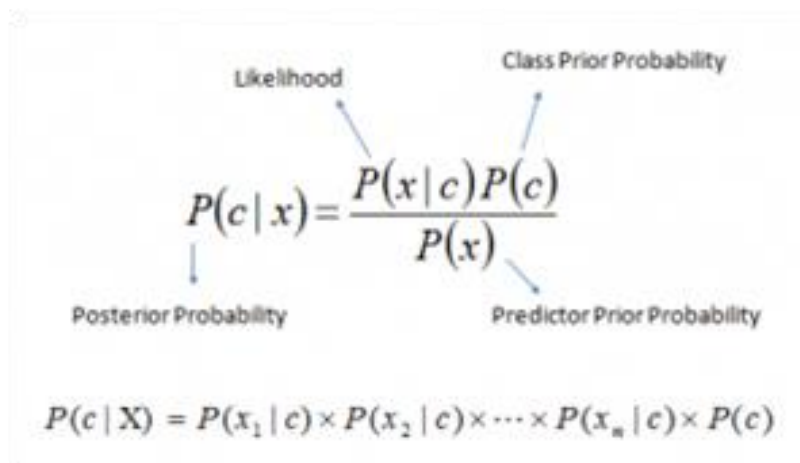
- sentiment analysis,
- spam filtering,

- recommendation systems etc.

For example, a fruit is also considered to be a kiwi if it's red, round, and about 3 inches in diameter. even though these features depend upon one another or upon the existence of the opposite features, all of those properties independently contribute to the probability that this fruit is a kiwi which is why it's called Naive.

Naive Bayes model is simple to make and particularly useful for very large data sets.

Bayes theorem provides the simplest way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. check out the equation below:



The diagram shows the Bayes' Theorem formula with labels for its components:

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

Labels and arrows:

- Likelihood** points to $P(x | c)$.
- Class Prior Probability** points to $P(c)$.
- Posterior Probability** points to $P(c | x)$.
- Predictor Prior Probability** points to $P(x)$.

Below the formula, the joint probability formula is shown:

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

- Navie Bayes is a classification technique based on Bayes Theorem which is used for the probability with an assumption of independence among predictors.
- In simple terms, a Naive Bayes classifier assumes that the presence of a specific feature during a class is unrelated to the presence of the other feature.
- Naive Bayes model is simple to make and particularly useful for very large data sets.

Naive Bayes classifier calculates the probability of an occurrence within the following steps:

- Step 1: Calculate the prior probability for given class labels
- Step 2: Find Likelihood probability with each attribute for every class
- Step 3: Put these value in Bayes Formula and calculate posterior probability of the given value.
- Step 4: See which class contains a higher probability, given the input belongs to the upper probability class.

KNN (K-Nearest Neighbors)

KNN(K-Nearest Neighbors) is a non-parametric and lazy learning algorithm. Non-parametric means there's no assumption for underlying data distribution. In other words, the model structure is determined from the dataset.

This will be very helpful in practice where most of the important world datasets don't follow mathematical theoretical assumptions. Lazy algorithm means it doesn't need any training data points for model generation.

All training data utilized in the testing phase. This would make training faster and testing phase slower and costlier. The costly testing phase means time and memory. within the worst case, KNN needs longer to scan all data points and scanning all data points would require more memory for storing training data.

Suppose P1 is that the point, that label must predict. First, you discover the k closest point to P1 then classify points by majority vote of its k neighbors. Each object votes for their class and therefore the class with the foremost votes is taken as the prediction. for locating closest similar points, you discover the distance between points using distance measures like Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance.

Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

- All 3 distance measures are only valid for continuous variables.

In the instance of categorical variables, the Hamming distance must be used to calculate the distance.

KNN has the subsequent basic steps:

1. Calculate distance
2. Find closest neighbors
3. Vote for labels

We can implement a KNN(K-Nearest Neighbors) model by following the below steps:

1. Load the info
2. Initialise the value of k
3. For getting the expected class, iterate from 1 to total number of training data points
 1. Calculate the distance between test data and every row of training data. Here we will use Euclidean distance as our distance metric since it's the foremost popular method.
 2. Sort the calculated distances in low to high(ascending) order based on distance values
 3. Get top k rows from the sorted array of the values
 4. Get the foremost frequent class of those rows
 5. Return the predicted class

3.5 INPUT AND OUTPUT ANALYSIS

INPUT

- In this project, we will take input as the dataset which contains large amount of records of the patients.
- We have to download the dataset from internet.
- Then, we will split the dataset into both training and testing parts.
- We will take the input as training set.
- We will build a model on the training data.
- We built a interactive web page their we will provide the values of the attributes we have taken to build the model.

OUTPUT

- The output of this project is to check whether the person has a heart disease or not.
- This output can be determined by our model which we have chosen.
- We have to choose the model with best accuracy.
- Then, we check the input taken on that model.
- Finally, we show the patient has a heart disease or not.
- The output can used for the further medication for the patient.

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION:

This project entitled “Heart disease prediction using Machine Learning Techniques” has presented an approach to predict the occurrence of heart disease by using different types of data mining and machine learning techniques. In this project, many machine learning algorithms have been used on the dataset and the performance of each algorithm is determined. Among all the algorithms used we got the highest accuracy with the Random Forest algorithm, which gave an accuracy of 95%. This project is useful for people in the medical field in treating people with heart disease, there are several treatment methods for the patient if they once diagnosed with a particular form of heart disease.

4.2 FUTURE SCOPE:

This project can be enhanced further with the use of combinational and more complex models to increase the accuracy of predicting the early onset of heart disease. With more data being fed into the database the system is going to be very intelligent. There are many possible improvements that could be explored to enhance the scalability and accuracy of this prediction system.

REFERENCES

1. Dats Science wikipedia https://en.wikipedia.org/wiki/Data_science
2. <https://www.datacamp.com/community/tutorials/decision-tree-classification-python>
3. M.A. Jabbar ,2016;Heart disease prediction system based on hidden naviebayes classifier;2018 International conference on circuits , controls, communications and computing(14C)
4. Rifki Wijaya,2013.Preliminary design of estimation heart disease by using machine learning ANN within on year; communication technology andelectricvehicle technology , bandung-bali, Indonesia.
5. Sana Bharti,2015 .Analytical study of heart disease prediction comparing with different algorithms; International conference on computing, communication andautomation(ICC2015).
6. Monika Gandhi,2015. Prediction in heart disease using techniques of data mining, International conference on futuristic trend in computational analysis and knowledge management(ABLAZE- 2015)
7. Sarath Babu,2017.Heart disease diagnosis using data mining technique,international conference on electronics,communication and aerospace technology ICECA2017
8. A H Chen, 2011. HDPS: heart disease prediction system; 2011 computing in cardiology
9. Dinesh Kumar G, 2018 . Prediction of cardiovascular disease using machine learning algorithms, proceedingof 2018 IEEE International Conference on Current Trends toward Converging Technologies, Coimbatore, India.
10. Purushottam ,2015. Efficient heart disease prediction system using decision tree;International conference on computing, communication &automation
11. Nikhil Gawande ,2017. Heart diseases classification using convolutional neural network; 2012 2ndInternational conference on communication and electronics systems(ICCES)
12. AnkitaDewan, 2015. Prediction of heart disease using a hybrid technique in data mining classification 20152nd.
- 13.Heart Desease Prediction System By Kennedy Ngure Ngare Bsc/Lmr/5551/17
- 14.Data Mining: Concepts and Techniques,3rdEdition Jiawei han| Micheline Kamber
- 15.Introduction to Data Science, <https://www.coursera.org/specializations/introduction-data-science>

APPENDICES

I.PLAGARISM REPORT

II. SAMPLE CODE

Heart Disease Prediction

Importing the libraries

```
#importing the libraries requiried
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler
```

Loading the Data

```
#Reading the data with th help of pandas
dataset = pd.read_csv("heart.csv",sep=',')
```

```
print(dataset)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang
oldpeak \									
0	63	1	3	145	233	1	0	150	0
2.3									
1	37	1	2	130	250	0	1	187	0
3.5									
2	41	0	1	130	204	0	0	172	0
1.4									
3	56	1	1	120	236	0	1	178	0
0.8									
4	57	0	0	120	354	0	1	163	1
0.6									
5	57	1	0	140	192	0	1	148	0
0.4									
6	56	0	1	140	294	0	0	153	0
1.3									
7	44	1	1	120	263	0	1	173	0
0.0									
8	52	1	2	172	199	1	1	162	0
0.5									
9	57	1	2	150	168	0	1	174	0
1.6									
10	54	1	0	140	239	0	1	160	0
1.2									

Heart Disease Prediction

11 0.2	48	0	2	130	275	0	1	139	0
12 0.6	49	1	1	130	266	0	1	171	0
13 1.8	64	1	3	110	211	0	0	144	1
14 1.0	58	0	3	150	283	1	0	162	0
15 1.6	50	0	2	120	219	0	1	158	0
16 0.0	58	0	2	120	340	0	1	172	0
17 2.6	66	0	3	150	226	0	1	114	0
18 1.5	43	1	0	150	247	0	1	171	0
19 1.8	69	0	3	140	239	0	1	151	0
20 0.5	59	1	0	135	234	0	1	161	0
21 0.4	44	1	2	130	233	0	1	179	1
22 0.0	42	1	0	140	226	0	1	178	0
23 1.0	61	1	2	150	243	1	1	137	1
24 1.4	40	1	3	140	199	0	1	178	1
25 0.4	71	0	1	160	302	0	1	162	0
26 1.6	59	1	2	150	212	1	1	157	0
27 0.6	51	1	2	110	175	0	1	123	0
28 0.8	65	0	2	140	417	1	0	157	0
29 1.2	53	1	2	130	197	1	0	152	0
...
...
273 0.1	58	1	0	100	234	0	1	156	0
274 1.0	47	1	0	110	275	0	0	118	1
275 1.0	52	1	0	125	212	0	1	168	0
276 2.0	58	1	0	146	218	0	1	105	0
277	57	1	1	124	261	0	1	141	0

0.3									
278	58	0	1	136	319	1	0	152	0
0.0									
279	61	1	0	138	166	0	0	125	1
3.6									
280	42	1	0	136	315	0	1	125	1
1.8									
281	52	1	0	128	204	1	1	156	1
1.0									
282	59	1	2	126	218	1	1	134	0
2.2									
283	40	1	0	152	223	0	1	181	0
0.0									
284	61	1	0	140	207	0	0	138	1
1.9									
285	46	1	0	140	311	0	1	120	1
1.8									
286	59	1	3	134	204	0	1	162	0
0.8									
287	57	1	1	154	232	0	0	164	0
0.0									
288	57	1	0	110	335	0	1	143	1
3.0									
289	55	0	0	128	205	0	2	130	1
2.0									
290	61	1	0	148	203	0	1	161	0
0.0									
291	58	1	0	114	318	0	2	140	0
4.4									
292	58	0	0	170	225	1	0	146	1
2.8									
293	67	1	2	152	212	0	0	150	0
0.8									
294	44	1	0	120	169	0	1	144	1
2.8									
295	63	1	0	140	187	0	0	144	1
4.0									
296	63	0	0	124	197	0	1	136	1
0.0									
297	59	1	0	164	176	1	0	90	0
1.0									
298	57	0	0	140	241	0	1	123	1
0.2									
299	45	1	3	110	264	0	1	132	0
1.2									
300	68	1	0	144	193	1	1	141	0
3.4									
301	57	1	0	130	131	0	1	115	1
1.2									

302 57 0 1 130 236 0 0 174 0
0.0

	slope	ca	thal	target
0	0	0	1	1
1	0	0	2	1
2	2	0	2	1
3	2	0	2	1
4	2	0	2	1
5	1	0	1	1
6	1	0	2	1
7	2	0	3	1
8	2	0	3	1
9	2	0	2	1
10	2	0	2	1
11	2	0	2	1
12	2	0	2	1
13	1	0	2	1
14	2	0	2	1
15	1	0	2	1
16	2	0	2	1
17	0	0	2	1
18	2	0	2	1
19	2	2	2	1
20	1	0	3	1
21	2	0	2	1
22	2	0	2	1
23	1	0	2	1
24	2	0	3	1
25	2	2	2	1
26	2	0	2	1
27	2	0	2	1
28	2	1	2	1
29	0	0	2	1
..
273	2	1	3	0
274	1	1	2	0
275	2	2	3	0
276	1	1	3	0
277	2	0	3	0
278	2	2	2	0
279	1	1	2	0
280	1	0	1	0
281	1	0	0	0
282	1	1	1	0
283	2	0	3	0
284	2	1	3	0
285	1	2	3	0
286	2	2	2	0

287	2	1	2	0
288	1	1	3	0
289	1	1	3	0
290	2	1	3	0
291	0	3	1	0
292	1	2	1	0
293	1	0	3	0
294	0	0	1	0
295	2	2	3	0
296	1	0	2	0
297	1	2	1	0
298	1	0	3	0
299	1	0	3	0
300	1	2	3	0
301	1	1	3	0
302	1	1	2	0

[303 rows x 14 columns]

Dataset Information

dataset.info()

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 303 entries, 0 to 302
```

```
Data columns (total 14 columns):
```

```
age          303 non-null int64
```

```
sex          303 non-null int64
```

```
cp           303 non-null int64
```

```
trestbps     303 non-null int64
```

```
chol         303 non-null int64
```

```
fbs          303 non-null int64
```

```
restecg      303 non-null int64
```

```
thalach      303 non-null int64
```

```
exang        303 non-null int64
```

```
oldpeak      303 non-null float64
```

```
slope        303 non-null int64
```

```
ca           303 non-null int64
```

```
thal         303 non-null int64
```

```
target       303 non-null int64
```

```
dtypes: float64(1), int64(13)
```

```
memory usage: 33.2 KB
```

dataset.hist()

```
array([[<matplotlib.axes._subplots.AxesSubplot object at
0x00000279BD7CC748>,
```

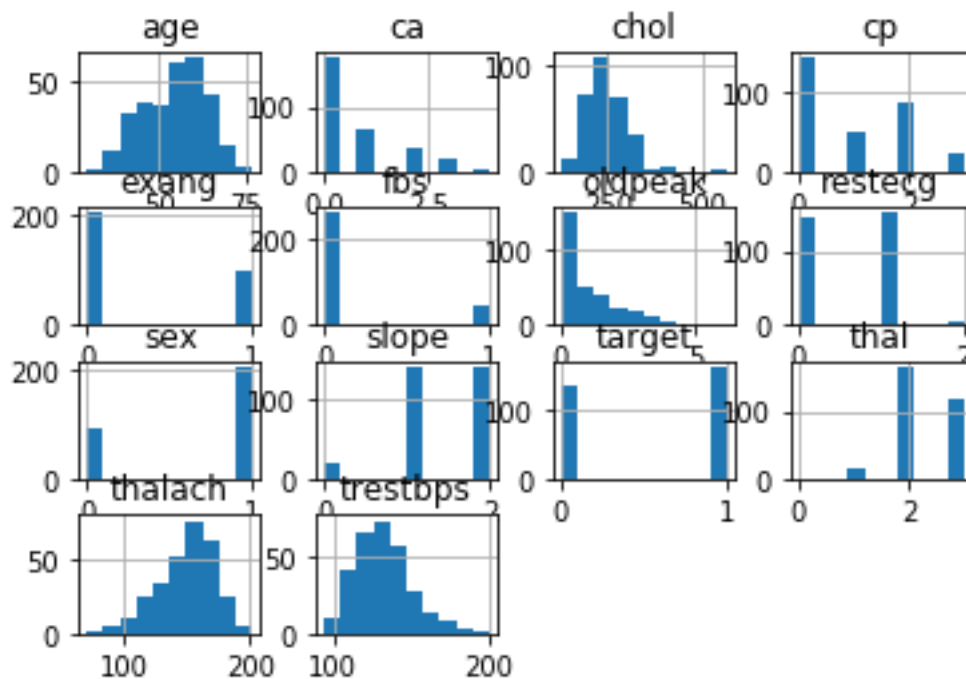
```
      <matplotlib.axes._subplots.AxesSubplot object at
0x00000279BEA8FE80>,
```

```
      <matplotlib.axes._subplots.AxesSubplot object at
0x00000279BEABF128>],
```

```

    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEAE390>],
    [<matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEB0E908>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEB36E80>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEB68438>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEB8D9E8>],
    [<matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEB8DA20>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEBE84E0>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEC0EA58>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEC36FD0>],
    [<matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEC66588>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BEC8DB00>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BECC10B8>,
    <matplotlib.axes._subplots.AxesSubplot object at
    0x00000279BECE5630>]],
    dtype=object)

```



Description

```
dataset.describe()
```

```
age
```

```
sex
```

```
cp
```

```
trestbps
```

```
chol
```

```
fbs
```

```
restecg
```

```
thalach
```

```
exang
```

```
oldpeak
```

```
slope
```

```
ca
```

```
thal
```

```
target
```

```
count
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
303.000000
```

```
mean
```

```
54.366337
```

```
0.683168
```

```
0.966997
```

```
131.623762
```

```
246.264026
```

```
0.148515
```

```
0.528053
```

```
149.646865
```

```
0.326733
```

1.039604
1.399340
0.729373
2.313531
0.544554
std
9.082101
0.466011
1.032052
17.538143
51.830751
0.356198
0.525860
22.905161
0.469794
1.161075
0.616226
1.022606
0.612277
0.498835
min
29.000000
0.000000
0.000000
94.000000
126.000000
0.000000
0.000000
71.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
25%
47.500000
0.000000
0.000000
120.000000
211.000000
0.000000

0.000000
133.500000
0.000000
0.000000
1.000000
0.000000
2.000000
0.000000
50%
55.000000
1.000000
1.000000
130.000000
240.000000
0.000000
1.000000
153.000000
0.000000
0.800000
1.000000
0.000000
2.000000
1.000000
75%
61.000000
1.000000
2.000000
140.000000
274.500000
0.000000
1.000000
166.000000
1.000000
1.600000
2.000000
1.000000
3.000000
1.000000
max
77.000000
1.000000
3.000000


```

200.000000
564.000000
1.000000
2.000000
202.000000
1.000000
6.200000
2.000000
4.000000
3.000000
1.000000

```

Correlation between columns

```
print(dataset.corr()["target"].abs().sort_values(ascending=False))
```

```

target      1.000000
exang       0.436757
cp          0.433798
oldpeak     0.430696
thalach     0.421741
ca          0.391724
slope       0.345877
thal        0.344029
sex         0.280937
age         0.225439
trestbps    0.144931
restecg     0.137230
chol        0.085239
fbs         0.028046
Name: target, dtype: float64

```

preprocessing the data

```

y = dataset['target']
X = dataset.drop(['target'], axis = 1)
print(X)
print(y)

```

```

      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak \
0      63   1   3      145    233   1         0      150      0
2.3
1      37   1   2      130    250   0         1      187      0
3.5
2      41   0   1      130    204   0         0      172      0
1.4
3      56   1   1      120    236   0         1      178      0
0.8

```

4	57	0	0	120	354	0	1	163	1
0.6									
5	57	1	0	140	192	0	1	148	0
0.4									
6	56	0	1	140	294	0	0	153	0
1.3									
7	44	1	1	120	263	0	1	173	0
0.0									
8	52	1	2	172	199	1	1	162	0
0.5									
9	57	1	2	150	168	0	1	174	0
1.6									
10	54	1	0	140	239	0	1	160	0
1.2									
11	48	0	2	130	275	0	1	139	0
0.2									
12	49	1	1	130	266	0	1	171	0
0.6									
13	64	1	3	110	211	0	0	144	1
1.8									
14	58	0	3	150	283	1	0	162	0
1.0									
15	50	0	2	120	219	0	1	158	0
1.6									
16	58	0	2	120	340	0	1	172	0
0.0									
17	66	0	3	150	226	0	1	114	0
2.6									
18	43	1	0	150	247	0	1	171	0
1.5									
19	69	0	3	140	239	0	1	151	0
1.8									
20	59	1	0	135	234	0	1	161	0
0.5									
21	44	1	2	130	233	0	1	179	1
0.4									
22	42	1	0	140	226	0	1	178	0
0.0									
23	61	1	2	150	243	1	1	137	1
1.0									
24	40	1	3	140	199	0	1	178	1
1.4									
25	71	0	1	160	302	0	1	162	0
0.4									
26	59	1	2	150	212	1	1	157	0
1.6									
27	51	1	2	110	175	0	1	123	0
0.6									
28	65	0	2	140	417	1	0	157	0

0.8									
29	53	1	2	130	197	1	0	152	0
1.2									
...
...									
273	58	1	0	100	234	0	1	156	0
0.1									
274	47	1	0	110	275	0	0	118	1
1.0									
275	52	1	0	125	212	0	1	168	0
1.0									
276	58	1	0	146	218	0	1	105	0
2.0									
277	57	1	1	124	261	0	1	141	0
0.3									
278	58	0	1	136	319	1	0	152	0
0.0									
279	61	1	0	138	166	0	0	125	1
3.6									
280	42	1	0	136	315	0	1	125	1
1.8									
281	52	1	0	128	204	1	1	156	1
1.0									
282	59	1	2	126	218	1	1	134	0
2.2									
283	40	1	0	152	223	0	1	181	0
0.0									
284	61	1	0	140	207	0	0	138	1
1.9									
285	46	1	0	140	311	0	1	120	1
1.8									
286	59	1	3	134	204	0	1	162	0
0.8									
287	57	1	1	154	232	0	0	164	0
0.0									
288	57	1	0	110	335	0	1	143	1
3.0									
289	55	0	0	128	205	0	2	130	1
2.0									
290	61	1	0	148	203	0	1	161	0
0.0									
291	58	1	0	114	318	0	2	140	0
4.4									
292	58	0	0	170	225	1	0	146	1
2.8									
293	67	1	2	152	212	0	0	150	0
0.8									
294	44	1	0	120	169	0	1	144	1
2.8									

295	63	1	0	140	187	0	0	144	1
4.0									
296	63	0	0	124	197	0	1	136	1
0.0									
297	59	1	0	164	176	1	0	90	0
1.0									
298	57	0	0	140	241	0	1	123	1
0.2									
299	45	1	3	110	264	0	1	132	0
1.2									
300	68	1	0	144	193	1	1	141	0
3.4									
301	57	1	0	130	131	0	1	115	1
1.2									
302	57	0	1	130	236	0	0	174	0
0.0									

	slope	ca	thal
0	0	0	1
1	0	0	2
2	2	0	2
3	2	0	2
4	2	0	2
5	1	0	1
6	1	0	2
7	2	0	3
8	2	0	3
9	2	0	2
10	2	0	2
11	2	0	2
12	2	0	2
13	1	0	2
14	2	0	2
15	1	0	2
16	2	0	2
17	0	0	2
18	2	0	2
19	2	2	2
20	1	0	3
21	2	0	2
22	2	0	2
23	1	0	2
24	2	0	3
25	2	2	2
26	2	0	2
27	2	0	2
28	2	1	2
29	0	0	2
..

273	2	1	3
274	1	1	2
275	2	2	3
276	1	1	3
277	2	0	3
278	2	2	2
279	1	1	2
280	1	0	1
281	1	0	0
282	1	1	1
283	2	0	3
284	2	1	3
285	1	2	3
286	2	2	2
287	2	1	2
288	1	1	3
289	1	1	3
290	2	1	3
291	0	3	1
292	1	2	1
293	1	0	3
294	0	0	1
295	2	2	3
296	1	0	2
297	1	2	1
298	1	0	3
299	1	0	3
300	1	2	3
301	1	1	3
302	1	1	2

[303 rows x 13 columns]

0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1

```
17      1
18      1
19      1
20      1
21      1
22      1
23      1
24      1
25      1
26      1
27      1
28      1
29      1
      ..
273     0
274     0
275     0
276     0
277     0
278     0
279     0
280     0
281     0
282     0
283     0
284     0
285     0
286     0
287     0
288     0
289     0
290     0
291     0
292     0
293     0
294     0
295     0
296     0
297     0
298     0
299     0
300     0
301     0
302     0
Name: target, Length: 303, dtype: int64
```

Splitting the data into training and testing

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20,random_state=0)
```

RandomForest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
max_accuracy=0
for x in range(2000):
    randomforest=RandomForestClassifier(random_state=x)
    randomforest.fit(X_train,y_train)
    y_pred=randomforest.predict(X_test)
    accuracy=round(accuracy_score(y_pred,y_test)*100,2)
    if accuracy>max_accuracy:
        max_accuracy=accuracy
        best_x=x
```

```
randomforest=RandomForestClassifier(random_state=best_x)
randomforest.fit(X_train,y_train)
y_pred=randomforest.predict(X_test)
print(y_pred)
```

```
[0 1 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1
1 0 0
1 0 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1]
```

Evaluating the Random Forest Algorithm

```
print("Evaluating the Random Forest Algorithm")
from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
print(confusion_matrix(y_pred, y_test))
print(classification_report(y_pred, y_test))
print("Accuracy:(in%)",round(accuracy_score(y_pred,y_test)*100,2))
accuracy_rf=round(accuracy_score(y_pred,y_test)*100,2)
```

Evaluating the Random Forest Algorithm

```
[[27  3]
 [ 0 31]]
```

	precision	recall	f1-score	support
0	1.00	0.90	0.95	30
1	0.91	1.00	0.95	31
micro avg	0.95	0.95	0.95	61
macro avg	0.96	0.95	0.95	61
weighted avg	0.96	0.95	0.95	61

Accuracy:(in%) 95.08

The accuracy achieved using Random Forest is: 95.08 %

Decision Tree

Creating the classifier object using Gini

```
from sklearn.tree import DecisionTreeClassifier
clf_gini = DecisionTreeClassifier(criterion = "gini",random_state =
100,max_depth=3, min_samples_leaf=5)
clf_gini.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=3,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=5, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False,
random_state=100,
                        splitter='best')
```

predicting on the testing set

```
y_pred=clf_gini.predict(X_test)
print(y_pred)

[0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1 1
1 1 0
 1 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1]
```

Evaluating the Decision Tree Algorithm

```
from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
print("Evaluating the Decision Tree by Gini")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:(in%)",round(accuracy_score(y_test,y_pred)*100,2))
accuracy_dtg=round(accuracy_score(y_test,y_pred)*100,2)
```

Evaluating the Decision Tree by Gini

```
[[23  4]
 [ 7 27]]
```

	precision	recall	f1-score	support
0	0.77	0.85	0.81	27
1	0.87	0.79	0.83	34
micro avg	0.82	0.82	0.82	61
macro avg	0.82	0.82	0.82	61

weighted avg	0.82	0.82	0.82	61
--------------	------	------	------	----

Accuracy:(in%) 81.97

Creating the classifier object using Entropy

```
clf_entropy = DecisionTreeClassifier(
    criterion = "entropy", random_state = 100,
    max_depth = 3, min_samples_leaf = 5)
clf_entropy.fit(X_train, y_train)

DecisionTreeClassifier(class_weight=None, criterion='entropy',
max_depth=3,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=5, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False,
random_state=100,
                        splitter='best')
```

predicting on the testing set

```
y_pred =clf_entropy.predict(X_test)
print(y_pred)

[0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1 1
 1 1 0
 1 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1]
```

Evaluating the Decision Tree Algorithm

```
from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
print("Evaluating the Decision Tree by Entropy")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:(in%)",round(accuracy_score(y_test,y_pred)*100,2))
accuracy_dte=round(accuracy_score(y_test,y_pred)*100,2)
```

Evaluating the Decision Tree by Entropy

```
[[23  4]
 [ 7 27]]
```

	precision	recall	f1-score	support
0	0.77	0.85	0.81	27
1	0.87	0.79	0.83	34
micro avg	0.82	0.82	0.82	61
macro avg	0.82	0.82	0.82	61
weighted avg	0.82	0.82	0.82	61

Accuracy:(in%) 81.97

```
accuracy_dt= accuracy_dte
print(accuracy_dt)
```

81.97

The accuracy achieved using Decision Tree is: 81.97 %

K-Nearest Neighbors

Training and Predicting using K-NN

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=23)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[0 1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 1 0 1 0
1 0 0
1 0 1 0 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 0 1 1 1 0]
```

Evaluating the KNN Algorithm

```
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score
print("Evaluating the KNN ")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:(in%)", round(accuracy_score(y_test,y_pred)*100,2))
accuracy_knn=round(accuracy_score(y_test,y_pred)*100,2)
```

Evaluating the KNN

```
[[19  8]
 [11 23]]
```

	precision	recall	f1-score	support
0	0.63	0.70	0.67	27
1	0.74	0.68	0.71	34
micro avg	0.69	0.69	0.69	61
macro avg	0.69	0.69	0.69	61
weighted avg	0.69	0.69	0.69	61

Accuracy:(in%) 68.85

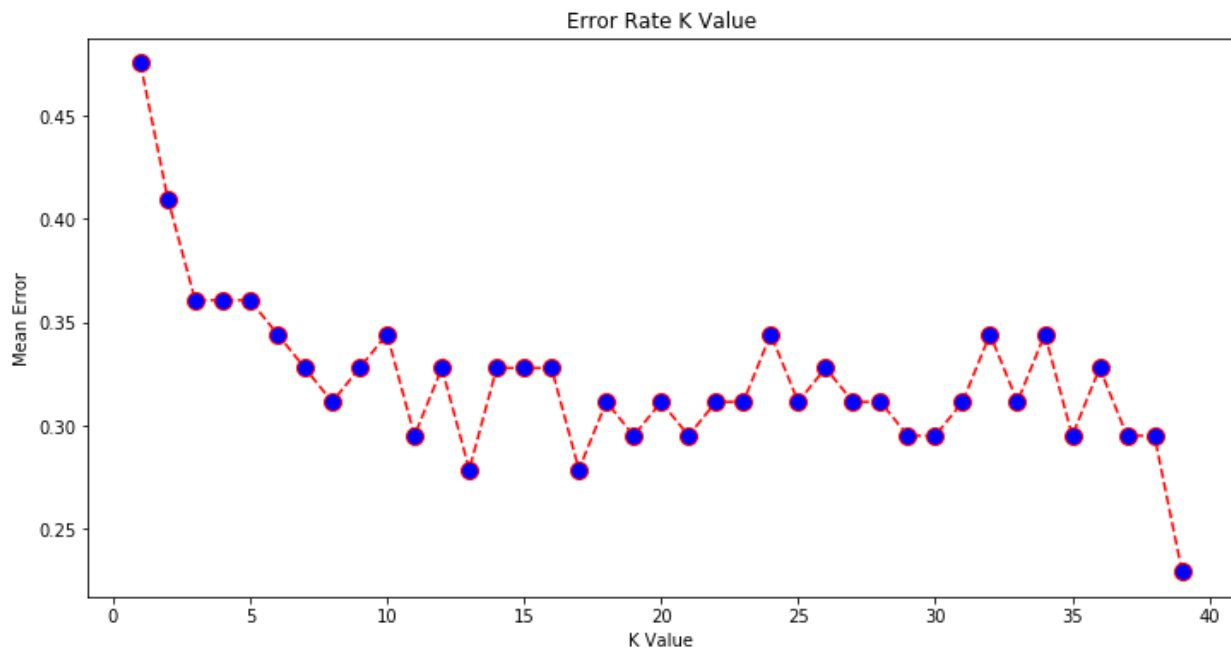
Calculating error for K values between 1 and 40

```
error=[]
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
```

```

    error.append(np.mean(pred_i != y_test))
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed',
marker='o',
        markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
error=[]

```



The accuracy achieved using KNN is: 68.85%

Support Vector Machine

Training and Predicting using SVM

```

from sklearn import svm
s=svm.SVC(kernel='linear')
s.fit(X_train,y_train)
y_pred=s.predict(X_test)
print(y_pred)

[0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 1
 1 1 0
 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1]

```

Evaluating the SVM Algorithm

```

from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
print("Evaluating the SVM ")
print(confusion_matrix(y_test, y_pred))

```

```
print(classification_report(y_test, y_pred))
print("Accuracy:(in%)",round(accuracy_score(y_test,y_pred)*100,2))
accuracy_svm=round(accuracy_score(y_test,y_pred)*100,2)
```

Evaluating the SVM

```
[[20  7]
 [ 4 30]]
```

	precision	recall	f1-score	support
0	0.83	0.74	0.78	27
1	0.81	0.88	0.85	34
micro avg	0.82	0.82	0.82	61
macro avg	0.82	0.81	0.81	61
weighted avg	0.82	0.82	0.82	61

Accuracy:(in%) 81.97

The accuracy achieved using SVM is: 81.97%

Navie Bayes

Training and Predicting using Navie Bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
y_pred = gnb.predict(X_test)
print(y_pred)
```

```
[0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 0 1 0 0 1 1 1 0 0 1 1
 1 0 0
 1 0 0 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

Evaluating the Navie bayes Algorithm

```
from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
print("Evaluating the NavieBayes ")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:(in%)",round(accuracy_score(y_test,y_pred)*100,2))
accuracy_nb=round(accuracy_score(y_test,y_pred)*100,2)
```

Evaluating the NavieBayes

```
[[21  6]
 [ 3 31]]
```

	precision	recall	f1-score	support
0	0.88	0.78	0.82	27

1	0.84	0.91	0.87	34
micro avg	0.85	0.85	0.85	61
macro avg	0.86	0.84	0.85	61
weighted avg	0.85	0.85	0.85	61

Accuracy:(in%) 85.25

The accuracy achieved using Navie Bayes is: 85.25%

XGBoost

Training and Predicting using XGBoost

```
import xgboost as xb
xgb_model=xb.XGBClassifier(objective="binary:logistic",random_state=
42)
xgb_model.fit(X_train, y_train)

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
               colsample_bynode=1, colsample_bytree=1, gamma=0,
               learning_rate=0.1,
               max_delta_step=0, max_depth=3, min_child_weight=1,
               missing=None,
               n_estimators=100, n_jobs=1, nthread=None,
               objective='binary:logistic', random_state=42, reg_alpha=0,
               reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
               subsample=1, verbosity=1)

y_pred = xgb_model.predict(X_test)
print(y_pred)

[0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 1
 1 0 0
 1 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

Evaluating the xgboost Algorithm

```
from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
print("Evaluating the xgboost ")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:(in%)",round(accuracy_score(y_test,y_pred)*100,2))
accuracy_xb=round(accuracy_score(y_test,y_pred)*100,2)
```

Evaluating the xgboost

```
[[22  5]
 [ 4 30]]
```

	precision	recall	f1-score	support
0	0.85	0.81	0.83	27

	1	0.86	0.88	0.87	34
micro avg		0.85	0.85	0.85	61
macro avg		0.85	0.85	0.85	61
weighted avg		0.85	0.85	0.85	61

Accuracy:(in%) 85.25

The accuracy achieved using XGBoost is: 85.25%

Logistic Regression

Training and predicting the data using LogisticRegression

```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print(y_pred)

[0 1 1 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 1 0 1 1
 1 1 0
 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

Evaluating the Logistic Regression Algorithm

```
from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score
print("Evaluating the Logistic Regression ")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:(in%)",round(accuracy_score(y_test,y_pred)*100,2))
accuracy_lr=round(accuracy_score(y_test,y_pred)*100,2)
```

Evaluating the Logistic Regression

```
[[22  5]
 [ 4 30]]
```

	precision	recall	f1-score	support
0	0.85	0.81	0.83	27
1	0.86	0.88	0.87	34
micro avg	0.85	0.85	0.85	61
macro avg	0.85	0.85	0.85	61
weighted avg	0.85	0.85	0.85	61

Accuracy:(in%) 85.25

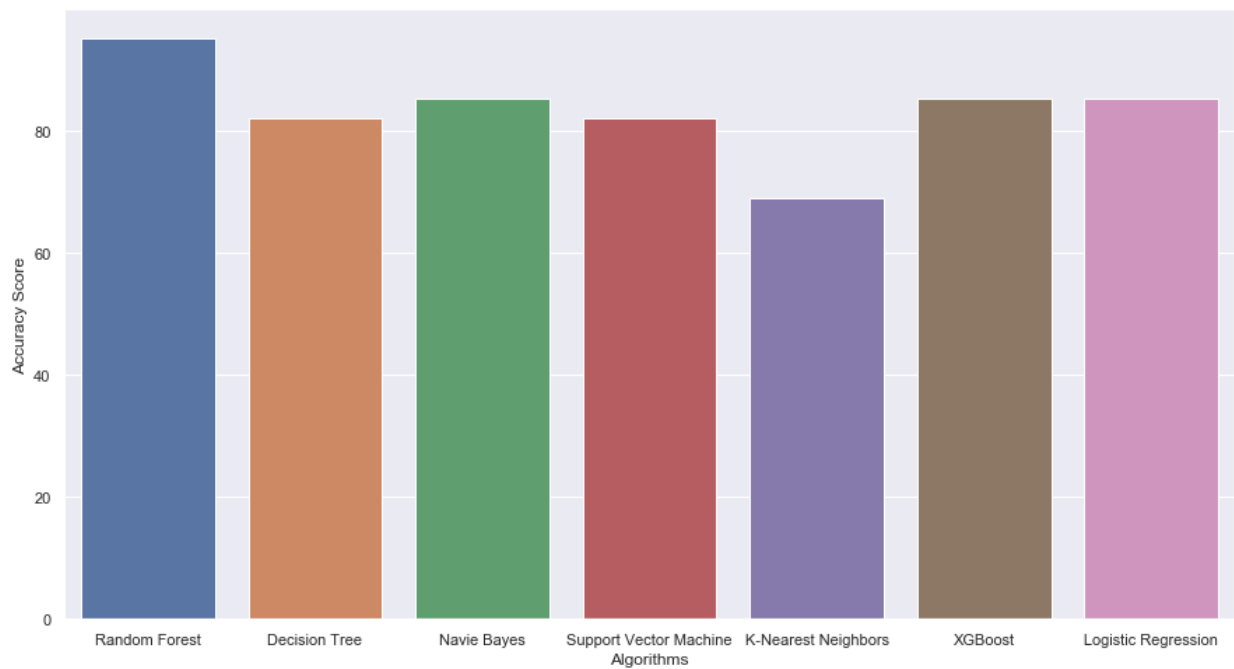
The accuracy achieved using Logistic Regression is: 85.25%

```
scores=[accuracy_rf,accuracy_dt,accuracy_nb,accuracy_svm,accuracy_knn,accuracy_xb,accuracy_lr]
```

```
algorithms=["Random Forest","Decision Tree","Navie Bayes","Support  
Vector Machine","K-Nearest Neighbors","XGBoost","Logistic  
Regression"]
```

Comparsion Between the Accuracies of Different Algorithms

```
import seaborn as sns  
sns.set(rc={'figure.figsize':(15,8)})  
plt.xlabel("Algorithms")  
plt.ylabel("Accuracy Score")  
sns.barplot(algorithms,scores)  
  
<matplotlib.axes._subplots.AxesSubplot at 0x279bf7f2b00>
```



Random Forest has the Best Accuracy than the other Algorithm

Flask Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import warnings
import os
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler

from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/')
def student():
    return render_template('student.html')

@app.route('/result', methods = ['POST', 'GET'])
def result():
    if request.method == 'POST':
        result = request.form
        dataset = pd.read_csv("heart.csv", sep=',')
        y = dataset['target']
        X = dataset.drop(['target'], axis = 1)
        from sklearn.model_selection import train_test_split
        X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y,
test_size=0.20)
        from sklearn.ensemble import RandomForestClassifier
        randomforest=RandomForestClassifier(random_state=1818)
        randomforest.fit(X_train1,y_train1)
        age = int(request.form['age'])
        sex = int(request.form['sex'])
        trestbps = float(request.form['trestbps'])
        chol = float(request.form['chol'])
        restecg = float(request.form['restecg'])
        thalach = float(request.form['thalach'])
        exang = int(request.form['exang'])
        cp = int(request.form['cp'])
        fbs = float(request.form['fbs'])
        slope=int(request.form['slope'])
        ca=int(request.form['ca'])
        oldpeak=float(request.form['oldpeak'])
```



```
thal=int(request.form['thal'])

d={'age':[age], 'sex':[sex], 'trestbps':[trestbps], 'chol':[chol], 'restecg':[
restecg], 'thalach':[thalach], 'exang':[exang], 'cp':[cp], 'fbs':[fbs], 'slope'
:[slope], 'ca':[ca], 'oldpeak':[oldpeak], 'thal':[thal]}
df = pd.DataFrame.from_dict(d)
df.to_csv('pred.csv')
pre=pd.read_csv('pred.csv', sep=',', index_col=0)
sol=randomforest.predict(pre)
if sol ==1 :
    os.remove('pred.csv')
    return render_template('failure.html')
else:
    os.remove('pred.csv')
    return render_template('healthy.html')
if __name__ == '__main__':
    app.run(debug = True)
```

III. SCREEN SHOTS

The screenshot shows a web browser window with the title 'Heart Disease Diagnosis'. The form contains the following fields:

- Enter your age: Your current age in years
- Enter your Gender: Male
- Resting blood pressure (in mm Hg on admission to the hospital): Your Blood Pressure
- Serum Cholesterol in mg/dl: Cholesterol
- Fasting blood sugar > 120mg/dl: Yes
- Rest ECG results: Normal
- Maximum heart rate achieved during ecg: Max heart rate
- Chest pain during exercise?: Yes
- Chest pain type?: No chest pain
- Slope: Upsloping



Your Heart is perfectly alright



Please Consult a Doctor