A COMPARATIVE ANALYSIS OF MACHINE LEARNING ALGORITHMS FOR HEART DISEASE PREDICTION

A project report submitted in partial fulfillment of the requirements for the award of the degree

Bachelor of Technology

In

Computer Science & Engineering

Submitted by:

ASHUTOSH JENA

Univ. Reg. No-1901301035

Under the Guidance of

Prof. SATYANANDA SWAIN

CSE, VITAM, Berhampur

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VIGNAN INSTITUTE OF TECHNOLOGY AND MANAGEMENT, BERHAMPUR

VITAM
wisdom counts...

VIGNAN

INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(Approved by AICTE and Affiliated to BPUT, Rourkela, Odisha)

ACCREDITED BY IAO

CERTIFICATE FOR APPROVAL

This is to certify that the project entitled "A COMPARATIVE ANALYSIS OF MACHINE LEARNING APPROACHES FOR HEART DISEASE PREDICTION", being submitted by Ashutosh Jena Bearing Univ.Reg. No-1901301035 for the award of degree of B.Tech. in Computer Science & Engineering is a record of bonafide project work carried out by him/her under my supervision. The results embodied in this project have not been submitted for the award of any other degree anywhere. In my opinion the project fulfils the requirement for the award of the degree.

Date:

Prof. Satyananda Swain

Dept. of CSE

VITAM
wisdom counts...

VIGNAN
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(Approved by AICTE and Affiliated to BPUT, Rourkela, Odisha)

ACCREDITED BY
IAO

CERTIFICATE

This is to certify that the project entitled "A COMPARATIVE ANALYSIS OF MACHINE LEARNING APPROACHES FOR HEART DISEASE PREDICTION', being submitted by Ashutosh Jena Bearing Univ.Reg. No- 1901301035 for the award of degree of B.Tech. in Computer Science & Engineering is a record of bonafide project work carried out by him/her under my supervision. The results embodied in this project have not been submitted for the award of any other degree anywhere. In our opinion the project fulfils the requirement for the award of the degree.

Date:

HOD, CSE ***Internal***

ACKNOWLEDGEMENTS

My sincere thanks to ***Prof. Satyananda Swain (Project Co-ordinator cum Project Guide)****, Department of Computer Science and Engineering, Vignan Institute of Technology and Management, Berhampur for their encouragement and valuable suggestions during the period of my project work.*

*No words would suffice to express my regard and gratitude to****, Prof. B. Srinivas Rao,*** *Head of Department of Computer Science and Engineering for his inspiring guidance and constant encouragement, immense support and help during the project work.*

Date:

NAME: Ashutosh Jena

REGD NO: 1901301035

DECLARATION

I hereby declared that the matter embodied in this project report is original and has not been submitted for the award of any other degree.

NAME: Ashutosh Jena

REGD NO: 1901301035

CHAPTER 1

INTRODUCTION

In day-to-day life, many factors affect the human heart. The term heart disease includes many diseases that are diverse and specifically affect the heart and the arteries of a human being. Even young aged people around 20-30 years of lifespan are getting affected by heart diseases. The increase in the possibility of heart disease among young may be due to bad eating habits, lack of sleep, restless nature, depression and numerous other factors such as obesity, poor diet, family history, high blood pressure, high blood cholesterol, idle behaviour, family history, smoking and hypertension.

The diagnosis of heart disease is very important and is itself the most complicated task in the medical field. All the mentioned factors are taken into consideration when analyzing and understanding the patients by the doctor through manual check-ups at regular intervals of time. The symptoms of heart disease greatly depend upon which of the discomfort felt by an individual. Some symptoms are not usually identified by the common people. However, common symptoms include chest pain, breathlessness, and heart palpitations. The chest pain common to many types of heart disease is known as angina, or angina pectoris, and occurs when a part of the heart does not receive enough oxygen. Recently, the healthcare industry has been generating huge amounts of data about patients and their disease diagnosis reports are being especially taken for the prediction of heart attacks worldwide.

When the data about heart disease is huge, the machine learning techniques can be implemented for the analysis. Data Mining is a task of extracting the vital decision-making information from a collective of past records for future analysis or prediction. The information may be hidden and is not identifiable without the use of data mining. The classification is one data mining technique through which the future outcome or predictions can be made based on the historical data that is available. The medical data mining made a possible solution to integrate the classification techniques and provide computerized training on the dataset that further leads to exploring the hidden patterns in the medical data sets which is used for the prediction of the patient's future state. Thus, by using medical data mining it is possible to provide insights on a patient's history and is able to provide clinical support through the analysis. For clinical analysis of the patients, these patterns are very much essential. In simple English, the medical data mining uses

classification algorithms that is a vital part for identifying the possibility of heart attack before the occurrence. The classification algorithms can be trained and tested to make the predictions that determine the person's nature of being affected by heart disease.

In this work, the supervised machine learning concept is utilized for making the predictions. A comparative analysis of the three data mining classification algorithms namely Random Forest, Support Vector Machines and Naïve Bayes are used to make predictions. The analysis is done at several levels of cross validation and several percentages of percentage split evaluation methods respectively.

The Stat Log dataset from UCI machine learning repository is utilized for making heart disease predictions in this research work. The predictions are made using the classification model that is built from the classification algorithms when the heart disease dataset is used for training. This final model can be used for prediction of any types of heart diseases. Health-care is a field of the most needed service and an economically 2ndlargest industry in 21stcentury. While we talk about the affordability and quality assurance in health-care industry, several statistical analyses are carried on making health solutions more precise and flawless in this current era of increasing health problems and chronic diseases. Advancements on data driven intelligent technologies is disease diagnosis and detection, treatment and research are remarkable.

Medical image analysis, symptom-based disease prediction is the part where the most sought-after brains are working. In this paper we aim to present our proposed model on the prediction on diagnosis of cardio vascular disease with ECG analysis and symptom-based detection. The model aims to be researched and advance in further to become robust and end to end reliable research tool. We will discuss about the classical methods and algorithms implemented on CVD prediction, gradual advancements, draw comparison of performance among the existing systems and propose an enhanced multi-module system performing better in terms of accuracy and feasibility. Implementation, training and testing of the modules have been done on datasets obtained from UCI and Physio net data repositories. Data format have been modified in case of the ECG report data for betterment of action by the convolutional neural network used in our research and in the risk prediction module we have chosen attributes for training and implementing the multi-layered neural network developed by us. The further research and advancement possibilities are also mentioned.

1.1 Motivation

The main motivation for doing this research is to present a heart disease prediction model for the prediction of the occurrence of heart disease. Further, this research work is aimed at identifying the best classification algorithm for identifying the possibility of heart disease in a patient. This work is justified by performing a comparative study and analysis using three classification algorithms namely Naïve Bayes, Logistic Regression, SVM, K-NN a,nd Random Forest are used at different levels of evaluations. Although these are commonly used machine learning algorithms, the heart disease prediction is a vital task involving highest possible accuracy. Hence, the three algorithms are evaluated at numerous levels and types of evaluation strategies. This will provide researchers and medical practitioners to establish a better.

1.2 Problem Statement

The major challenge in heart disease is its detection. There are instruments available which can predict heart disease but either it are expensive or are not efficient to calculate chance of heart disease in human. Early detection of cardiac diseases can decrease the mortality rate and 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 analyse the data for hidden patterns. The hidden patterns can be used for health diagnosis in medicinal data.

1.3 Project Goals

As we seen in now a days major cause of Death is due to the Cardiovascular Disease and many of the patients claimed that they not knew about their condition earlier , and if they knew they get a chance to survive among us , now by seeing this situation we came to an idea to build a system which predicts the condition of their heart by measuring some the key aspects of their health, this system was HDFS " Heart Failure Predicting System" this helps millions of the people by knowing their heart condition earlier and can have a treatment at earlier stages which helps them to not to undergo any critical conditions . We as a human's pay more importance to our life than any other things as it is the one which won't come back after time, this is a small effort of us to predict heart failure, if this works properly, we are trying to implement other diseases also.

1.4 Classification Basics

Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and can use it to learn by themselves. Supervised learning is the Data mining task of inferring a function from labelled training data. Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically, supervised learning is learning in which we teach or train the machine using data that is well labelled which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples (data) so that the supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labelled data. A Classification is a division or category in a system that divides things into groups or types. It is the act or process of classifying or arranging groups or categories according to established class criteria, and categories. Here are some of the most commonly used classification algorithms: "Logistic Regression", "Naïve Bayes", "Support Vector Machine", and "K-Nearest Neighbour". In our case we have used naive Bayes, logistic regression, and K-Nearest Neighbour support vector classifier.

1.5 Overview of the Proposed System

A set of ML algorithms such as Support Vector Machines (SVM), K-Nearest Neighbour (KNN), NAVIE BAYES (NB), RANDOM FOREST, and LOGISTIC REGRESSION will be deployed for the prediction of corona virus disease followed by performance evaluation and analysis of the chosen algorithms to identify the optimal technique with the highest accuracy. The purpose of this project is to find out the most likely chance of Heart Failure for a person so that he/she can have a chance to save their life by taking early treatment.

1.6 Organization of Report

At the outset, this report states a brief introduction. Followed by related work, that has been followed while carrying out our research work. The next chapter gives a detailed description of the methodologies used. The next chapter provides a detailed demonstration of results, analysis of results, and visualizations. The next two chapters deal with the conclusion, future enhancements, and references considered for the project. The last chapter presents the complete code for heart disease prediction.

CHAPTER 2

LITERATURE REVIEW

Rishabh Magar et al. [1] proposed a web application based predictive model trained on the UCI dataset with a 75-25 training and testing division of the dataset. Logistic Regression based predictive models were found to be the most accurate with 82.89% accuracy, followed by SVM at 81.57% and Naive Bayes and Decision Tree at 80.43% each. The web application can be used by the end user as a preliminary test for checking their heart condition and seeking medical advice if needed.

Apurb Rajdhan et al. [2] proposed a system where four classification algorithms such as Random Forest, Decision Tree, Logistic Regression and Naive Bayes are used to predict the patient's condition. Data is split into 80% training data and 20% testing data. A confusion matrix depicting true and false positives as well as true and false negatives was created. Maximum accuracy obtained was 90.16% using Random Forest classification.

Devansh Shah et al. [3] proposed a system of models using supervised learning methods through the WEKA tool. Four individual classification techniques including NB, KNN, RF, DT were used to predict the chances of having a heart disease. The dataset was
initially cleaned, transformed by smoothening, normalisation, and aggregation, integrated and reduced. The maximum accuracy obtained was through KNN method.

Harshit Jindal et al. [4] implemented a system that uses three different classification algorithms, KNN, RF, LR and results in an accuracy of 87.5%. In this EHDPS i.e, effective heart disease prediction system, Logistic Regression and KNN outperform RF with
KNN providing an accuracy of 88.52% which is highest amongst the three techniques used.

Aadar Pandita et al. [5] proposed a predictive model that implements 5 machine learning algorithms and uses the technique with the
highest accuracy to build a web application that takes in patient's medical details and predicts if they have a heart disease or not.

The web application is built using HTML/CSS and Flask based framework. The maximum accuracy obtained was obtained using KNN, i.e., 89.06% while Logistic Regression contributed with least accuracy of 84.38%.

N. Saranya et al. [6] proposed a time and money efficient model of predicting heart disease using a web application. The model works on two different methods : Random Forest and KNN. The dataset has been taken from one of Coimbatore's hospitals which
produces an accuracy of 100% using Random Forest and 91.36% using KNN after cleaning and pre-processing of the dataset. An ensemble model with and without Logistic Regression is also used to predict the chances with an accuracy of 98.77% and 95.06% respectively.

Aravind Akella et al. [7] applied six predictive models on the UCI dataset and achieved a maximum accuracy of 93.03% with Neural Networks with a recall of 93.8 indicating low chances of false negatives and therefore extremely precise result, while the
other five had an accuracy of almost 80% and more.

Ravindhar NV et al. [8] implemented five algorithms: Logistic Regression, Naive Bayes, Fuzzy KNN, K-Means Clustering and back propagation Neural-Network. A 10-fold cross validation method is used in the experimental analysis of heart conditions. The
maximum accuracy was gathered using back propagation Neural Network with 98.2% accuracy and 87.64% recall and 89.65% precision.

Dr. Poonam Ghuli et al. [9] proposed work predicts the chance of Heart Disease and classifiers patient's risk level by implementing different data mining techniques such as NB, DT, LR, RF. Thus this paper presents a comparative study by analysing the performance machine learning algorithms using UCI ML repository dataset. Result of this study indicates that the random forest algorithm is the most efficient algorithm with accuracy score of 90.16% for prediction of heart disease. It help in providing better result and help health professionals.
Harshit Jindal et al. [10] developed a model that classifies patients are at risk of getting a heart disease or not and also this method is totally cost efficient. They summarised that with improved

accuracy due to the increased medical attributes used from the dataset. KNN outperforms other classifiers with 87.5%.

Abhijeet Jagtap et al. [11] classified a prediction system for efficient and effective heart disease detection out of 727 people dataset using algorithm. SVM, LR, NV got accuracy as 64.4%, 61.45% and 60% respectively. SVM was selected as the most efficient algorithm.

Md Mamun Ali et al. [12] found that using a Heart disease dataset collected from Kaggle three/classification based on KNN ,Decision Tree and Random Forest , Where RF Algorithm achieved 95% accuracy along with 90% Sensitivity and Specificity. They found that relatively simple supervised ML Algorithms can be used to make Heart Disease Predictions with a very high accuracy and excellent potential utility.

Pooja Rani et al. [13] found that the system has given the most accurate results with random forest classifier. It has achieved an accuracy of 86.60%, which is superior to some of the existing HDP systems found in the literature.

Muhammad Adnan et al. [14] proposed a SVM based architecture for heart disease prediction, empowered with a fuzzy based decision level fusion, is presented. The SVM architecture has improved the accuracy significantly as compared to existing solutions where 96.23% accuracy has been achieved.

Apurv Garg et al. [15] Classified a supervised learning-based machine learning algorithm which can make diagnoses of Cardio Vascular Diseases (CVDs) easy. Two supervised ML algorithms are used in this paper which are, KNN and Random Forest. The prediction accuracy obtained by KNN is 86.885% and 81.967% by Random Forest.

Rizwan Ali et al. [16] Combined cloud computing and ML algorithm using SVM, simulating results have shown that the proposed intelligent cloud-based Heart Disease Prediction system empowered with a SVM based model which gives 93.33% accuracy, which is better than previously published approaches.

R.Jane Preetha Princy et al. [17] implemented a supervised machine learning algorithm system using Cardiovascular dataset is classified precisely used for Disease Prediction. The results indicate that Decision Tree classification model predicted the Cardiovascular Diseases better then Naïve Bayes, Logistic Regression, Support Vector Machine and K-Nearest Neighbour based approaches. Space the Decision Tree outperforms the other algorithms by achieving the accuracy of 73%.

Yuan Zhao MPH et al. [18] This study reviews how social determinants of health care are being included in ML algorithms to inform best Practices. Logistic Regression (LR) achieved 75% accuracy.

Deepak Kumar Chohan et al. [19] in this work, they used the LR, Decision Tree, SVM, Naïve Bayes, KNN and Random Forest algorithms to predict Heart Disease with the purpose of determining which method is the most reliable. In this case, Decision Tree came out at the top with 98.53% accuracy.

Kummita Sravan Kumar Reddy et al. [20] dynamic KNN is a simple algorithm used in this paper. Heart Disease dataset is used for Disease Prediction, accuracy of Dynamic KNN is 84.44% and of SVM is 67.21%. KNN appears to perform significantly better then SVM (Support Vector Machine) for Novel Heart Disease prediction.

Year	Author	Paper Name	Algorithm Used	Accuracy Obtained
2020	Rishabh Magar	Heart disease prediction using machine learning	Logistic Regression SVM Naive Bayes Decision Tree	82.89% 81.57% 80.43% 80.43%
2020	Apurb Rajdhan	Heart disease prediction using machine learning	Logistic Regression Decision Tree Random Forest Naïve Bayes	85.25% 81.97% 90.16% 85.25%
2020	Devansh Shah	Heart disease prediction using machine learning	Naive Bayes KNN Random Forest Decision Tree	88.157% 90.789% 86.84% 80.263%
2021	Harshit Jindal	Heart disease prediction using machine learning algorithms	KNN Logistic Regression KNN & LR based model	88.25% 88.5% 87.5%
2021	Aadar Pandita	Prediction of Heart Disease using Machine learning Algorihtm	Logistic Regression	84.38%
2021	Pooja Rani	A decision support system for heart disease prediction based upon machine learning	Random Forest	86.60%
2021	Muhammad Adnan	Fusion-Based Machine Learning Architecture for Heart Disease Prediction	SVM	96.23%
2020	Apurv Garg	Heart disease prediction using machine learning techniques	K-Nearest Neighbor Random Forest	86.885% 81.967%
2020	Rizwan Ali	Intelligent Cloud Based Heart Disease Prediction System Empowered with Supervised Machine Learning	SVM	93.33%
2020	R. Jane Preetha Princy	Prediction of Cardiac Disease using Supervised Machine Learning Algorithms	Decision tree	73%
2021	Yuan Zhao MPH	Social Determinants in Machine Learning Cardiovascular Disease Prediction Models: A Systematic Review	Logistic Regression	75%
2022	Deepak Kumar Chohan	A Comparison Based Study of Supervised Machine Learning Algorithms for Prediction of Heart Disease	Decision Tree	98.53%
2022	Kummita Sravan Kumar Reddy	Novel Intelligent Model for Heart Disease Prediction using Dynamic KNN (DKNN) with improved accuracy over SVM	KNN	84.44%

2020	N. Saranya	Heart Disease prediction using Machine Learning	Random Forest	100%
2021	Aravind Akella	Machine learning algorithms for predicting coronary artery disease: effort towards an open source solution	Neual Network	93.03%
2019	Ravindhar NV	Intelligent Diagnosis of Cardiac Disease Predicttion using Machine Learning	BP-Neural Network	98.20%
2020	Dr. Poonam Ghuli	IJERT Heart Disease Prediction using Machine Learning	Decision Tree Logistic Regression Random Forest Naive Bayes	81.97% 85.25% 90.16% 85.25%
2021	Harshit Jindal	Heart disease prediction using machine learning algorithms ; ICCRDA 2020 IOP Conf. Series: Materials Science and Engineering	KNN	87.50%
2019	Abhijeet Jagtap	Heart Disease Prediction using Machine Learning	SVM Logistic Regression Naive Bayes	64.4% 61.45% 60%
2021	Md Mamun Ali	Heart disease prediction using supervised machine learning algorithms: Performance analysis and comparison	Random Forest	95%

Table 2.1: Literature survey details

CHAPTER 3

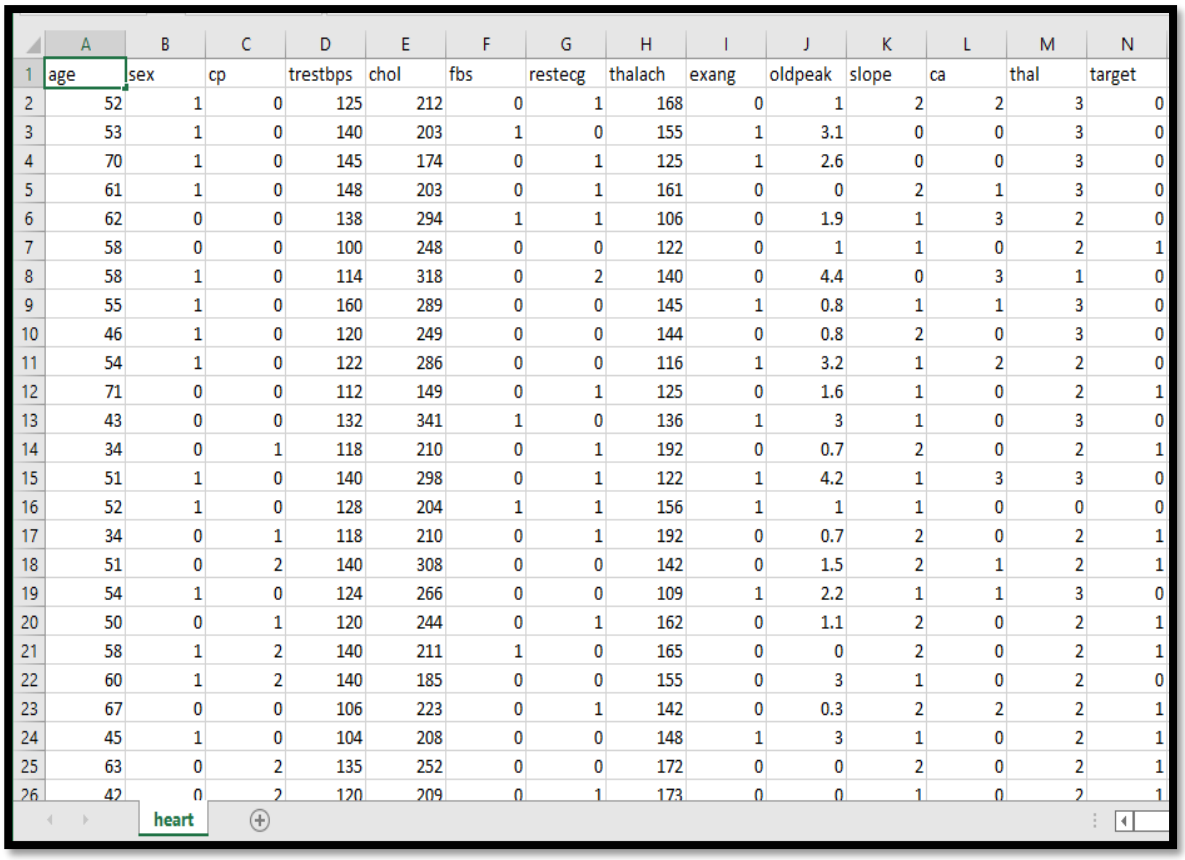
DATA DETAILS AND EXPERIMENT DETAILS

3.1 Data Source

This data comes from the University of California Irvine's Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/Heart+Disease.

Figure 3.1: Dataset Representation for Heart disease prediction

3.2 Dataset Details



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
2	52	1	0	125	212	0	1	168	0	1	2	2	3	0
3	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
4	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
5	61	1	0	148	203	0	1	161	0	0	2	1	3	0
6	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
7	58	0	0	100	248	0	0	122	0	1	1	0	2	1
8	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
9	55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
10	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
11	54	1	0	122	286	0	0	116	1	3.2	1	2	2	0
12	71	0	0	112	149	0	1	125	0	1.6	1	0	2	1
13	43	0	0	132	341	1	0	136	1	3	1	0	3	0
14	34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
15	51	1	0	140	298	0	1	122	1	4.2	1	3	3	0
16	52	1	0	128	204	1	1	156	1	1	1	0	0	0
17	34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
18	51	0	2	140	308	0	0	142	0	1.5	2	1	2	1
19	54	1	0	124	266	0	0	109	1	2.2	1	1	3	0
20	50	0	1	120	244	0	1	162	0	1.1	2	0	2	1
21	58	1	2	140	211	1	0	165	0	0	2	0	2	1
22	60	1	2	140	185	0	0	155	0	3	1	0	2	0
23	67	0	0	106	223	0	1	142	0	0.3	2	2	2	1
24	45	1	0	104	208	0	0	148	1	3	1	0	2	1
25	63	0	2	135	252	0	0	172	0	0	2	0	2	1
26	42	0	2	120	209	0	1	173	0	0	1	0	2	1

heart

Figure 3.2: Dataset Representation in Excel file for features of heart disease prediction dataset

3.3 Experimental Setup

Hardware Requirement

1	Processor	Dual core or core i3/i5/i7
2	Hard disk	500 GB or above
3	RAM	4GB or above

Table 3.1: Hardware Requirement Details

Software Requirement

1	Operating system	Windows or Linux
2	Python IDE	Python 2.7.x and above
3	Pycharm IDE Required	Jupiter notebook
4	Language	Python Scripting

Table 3.2: Software Requirement Details

3.4 Python and its Packages

Python is an object-oriented programming language that provides rapid application development. Python is an interpreted, high-level, general purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Pre-defined packages:

Sklearn:

Sklearn or scikit learn is a free machine learning library of python. It features various algorithms like support vector machine, random forest and k-neighbors. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib.

Matplotlib:

Matplotlib is a plotting library for the python programming language and its numerical mathematics extension numpy. Matplotlib. Pyplot is a collection of command style functions that make matplotlib work like MATLAB.

Seaborn:

Seaborn is a library for making statistical in graphics Python. It is built on top of matplotlib and closely integrated with pandas data structure. seaborn: statistical data visualization. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Confusion matrix:

Confusion matrix is a table that is often used to describe the performance of a classification model or classifier on a set of test data for which true values are known. Performance measure are computed from the confusion matrix. It allows the visualization of performance of an algorithm.

NumPy:

It is a numeric python module that provides fast math functions for calculations.

CHAPTER 4

METHODOLOGY AND PROPOSED WORK

4.1 Approaches

4.1.1 Logistic Regression

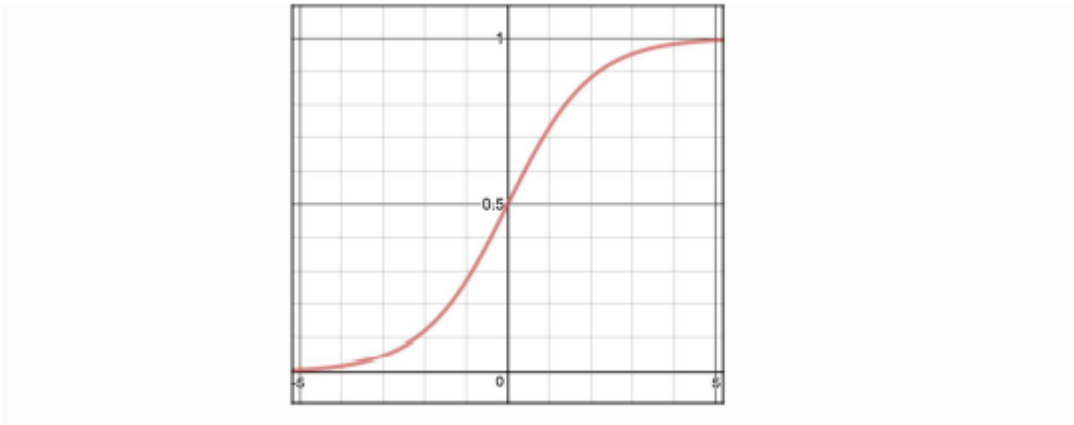
Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

Types of logical regression:

- Binary (Pass/Fail)
- Multi (Cats, Dogs, Sheep)

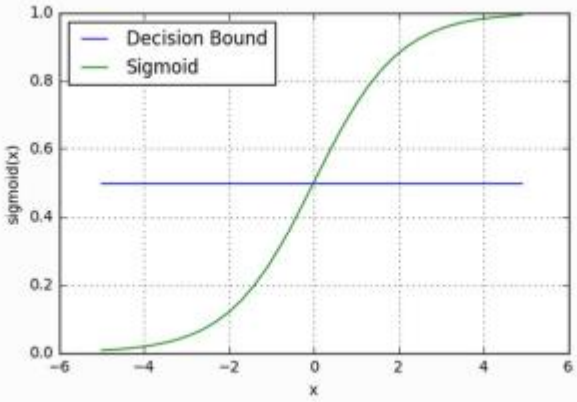
Sigmoid function

S(z)=1/1+e^−z



Decision Boundary

p≥0.5, class=1 p<0.5, class=0



Cost Funcction

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if } y = 1$$
$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if } y = 0$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))]$$

Vectorised Cost Function

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1 - y)^T \log(1 - h)\right)$$

For Multiclass - Instead of y=0,1 we will expand our definition so that y=0,1...n. Basically we re-run binary classification multiple times, once for each class.

Procedure –

1. Divide the problem into n+1 binary classification problem (+1 because the index starts at 0).

2. For each class…

3. Predict the probability the observations are in that single class.

4. prediction = max (probability of the classes)

4.1.2 Random Forest

Random Forest is a supervised learning algorithm. Random forest can be used for both classification and regression problems, by using random forest regressor we can use random forest on regression problems. But we have used random forest on classification in this project so we will only consider the classification part.

Random Forest pseudocode

1. Randomly select “k” features from total “m” features. Where k << m
2. Among the “k” features, calculate the node “d” using the best split point.
3. Split the node into daughter nodes using the best split.
4. Repeat 1 to 3 steps until “l” number of nodes has been reached.
5. Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

4.1.3 Naïve Bayes

Bayes’ Theorem is stated as:

$P(h|d) = (P(d|h) * P(h)) / P(d)$

- P(h|d) is the probability of hypothesis h given the data d. This is called the posterior probability.
- P(d|h) is the probability of data d given that the hypothesis h was true.
- P(h) is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- P(d) is the probability of the data (regardless of the hypothesis).

we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h). After calculating the posterior probability for a number of different hypotheses, we will select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the (MAP) hypothesis.

This can be written as: **MAP(h) = max(P(h|d))**

Or **MAP(h) = max((P(d|h) * P(h)) / P(d))**

Or **MAP(h) = max(P(d|h) * P(h))**

The P(d) is a normalizing term that allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize. Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. P(h)) will be equal. Again, this would be a constant term in our equation, and we could drop it so that we end up with:

$$\mathbf{MAP(h) = max(P(d|h))}$$

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P (d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) * P(d2|H) and so on. This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

$$\mathbf{MAP(h) = max(P(d|h) * P(h))}$$

4.1.4 K-Nearest Neighbour

We can implement a KNN model by following the below steps:

1. Load the data

2. Initialize the value of k

3. For getting the predicted class, iterate from 1 to the total number of training data points

- Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.

- Sort the calculated distances in ascending order based on distance values

- Get the top k rows from the sorted array

- Get the most frequent class of these rows

- Return the predicted class

4.1.5 Support Vector Machines

The objective of the support vector machine algorithm is to find a hyperplane in N- dimensional space (N-the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. The numeric input variables (x) in your data (the columns) form an n-dimensional space. For example, if you had two input variables, this would form a two-dimensional space. A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1. In two-dimensions you can visualize this as a line and let's assume that all of our input points can be completely separated by this line. For example: B0 + (B1 * X1) + (B2 * X2) = 0

Where the coefficients (B1 and B2) that determine the slope of the line and the intercept (B0) are found by the learning algorithm, and X1 and X2 are the two input variables. You can make classifications using this line. By plugging in input values into the line equation, you can calculate whether a new point is above or below the line.

• Above the line, the equation returns a value greater than 0 and the point belongs to the first class (class 0).

• Below the line, the equation returns a value less than 0 and the point belongs to the second class (class 1).

• A value close to the line returns a value close to zero and the point may be difficult to classify.

4.2 Architecture

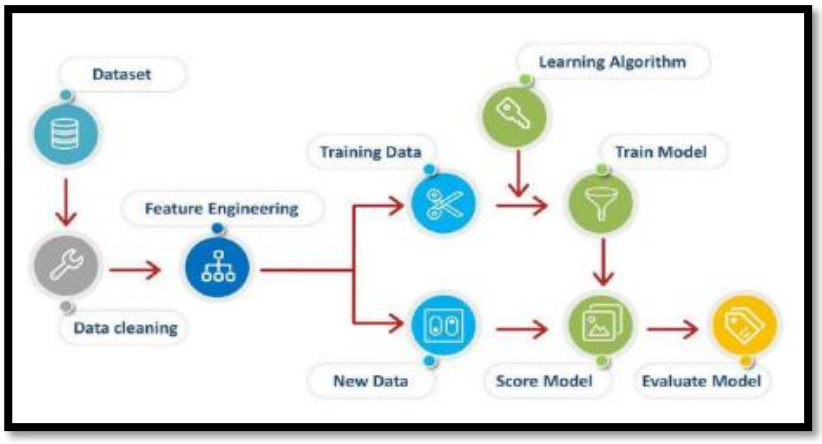


Figure 4.1 : Architecture of SVM Workflow.

4.3 Performance Evaluation

4.3.1 Confusion Matrix

Confusion matrix is a table that is often used to describe the performance of a classification model or classifier on a set of test data for which the true values are known. Performance measure is computed from the confusion matrix. It allows the visualization of performance of an algorithm.

True positive (TP): Predictive values correctly predicted as actual positive.

False positive (FP): Predicted values incorrectly predicted an actual positive. i.e., negative values predicted as positive.

False Negative (FN): Positive values predicted as negative.

True Negative (TN): Predicted values correctly predicted as an actual negative.

4.3.2 Precision:

In pattern recognition, classification precision called positive predictive is the fraction of relevant instance among the retrieved instances.

Precision = TP/(TP+FP)

4.3.3 Recall:

Recall is the fraction of the total amount of relevant instances that were actually retrieved.

Recall = TP/(TP+FN)

4.3.4 F1-measure:

F1-measure is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negative into account

F1-measure = 2* Recall * Precision / Recall + Precision

4.4 Outline of Experiment

Step 1:

Importing a package make all the objects defined in the packages available to the importer so that the python is able to run all the codes present in that file. The predefined packages and class are required to be imported to make python program run smoothly as well as make the code easier for the user.

Step 2:

Reading dataset refers to loading or importing the dataset which we have downloaded from the KAGGLE Repository website into our code. Data collection is a very basic module and the initial step towards the project. It generally deals with the collection of the right dataset.

Step 3:

The dataset that is to be used in the binary classification has to be used to be filtered based on various aspects. Data pre-processing involves transforming raw data into a more coherent format. The data pre-processing involves nan-removal.

Step 4:

Dataset selection refers to dividing the dataset into sections to store in variables to find out the status for the biomedical voice measurements.

Step 5:

Training the machine is similar to feeding the data to the algorithm to touch up the test data. The training of the model includes cross-validation where we get a well-grounded approximate performance of the model using the training data. We have divided the dataset in the ratio of 45:55 where 45% is for the training set and the rest 55% for the testing set of the data. The test sets are untouched, as a model should not be judged based on unseen data.

Step 6:

The data we use is usually split into training data and test data. In our case it divides into 45:55 where 55% is for testing set of the data. The training set contains a known output and the model learns based on this data in order to be generalized to other data later on. We have the test dataset in order to test our models work on this subset.

Step 7:

Here we had implemented four models / algorithm to classify the raw data based on result of the given training. Confusion matrix is often used to describe the performance of a classification model or classifier on a set of test data for which the true values are known.

Step 8:

Classification accuracy is the ratio of correct predictions to total prediction made. It is often presented as a percentage by multiplying the result by 100. We have found out the accuracy for all four models using confusion matrix. Then we have compared the accuracy of all four models and have visualized the comparison using the visualization tools.

Step 9:

After evaluation of accuracy and visualization of each model, we have provided new sample data as input and evaluate its correct inclusion in a respective status. Based on the accuracy of classification models, we determine the optimal model structure that define intuitive classification of biological voice measurements into status out of the considered four machine learning models.

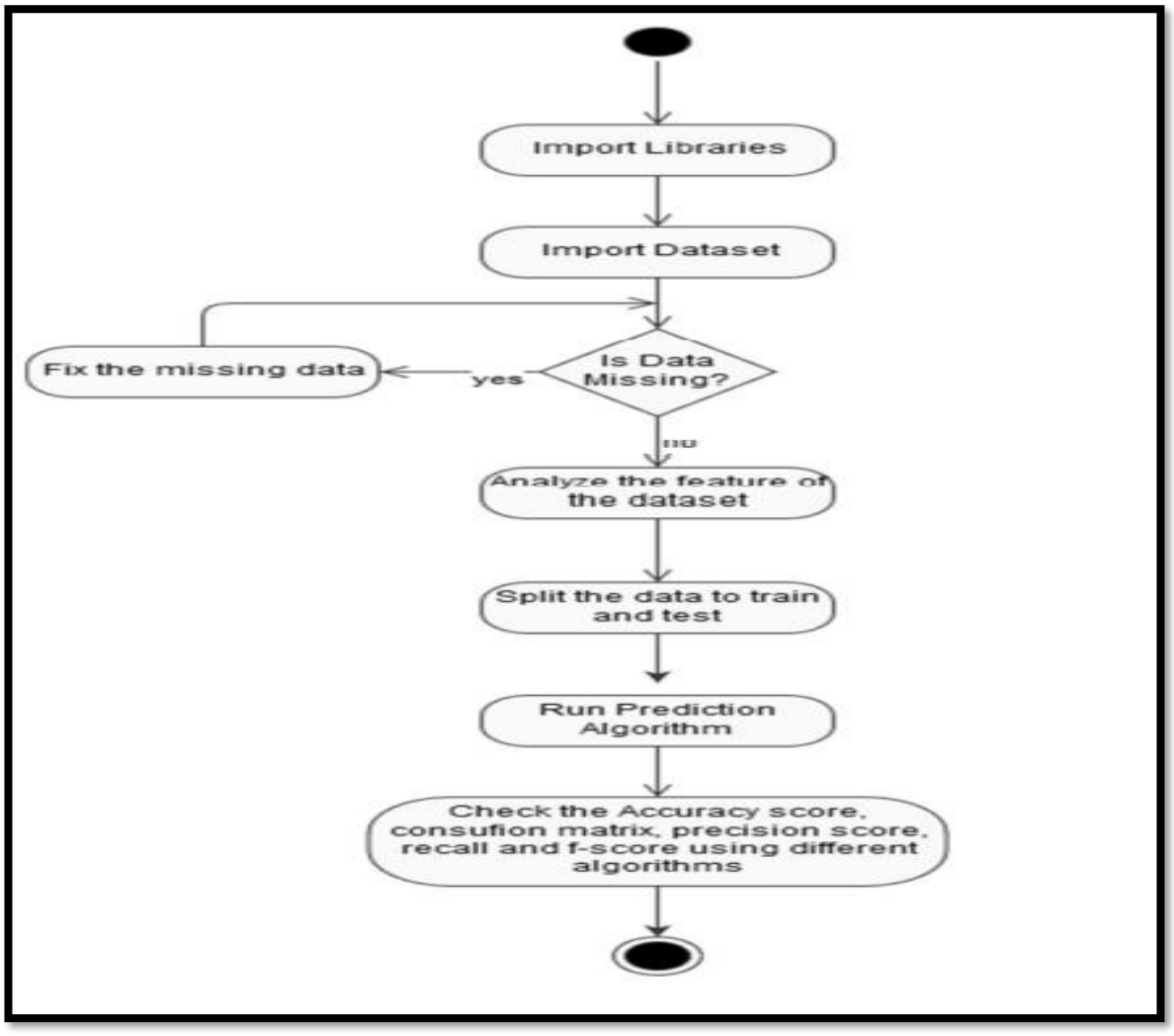


Figure 4.2: Heart Disease Prediction Workflow Diagram.

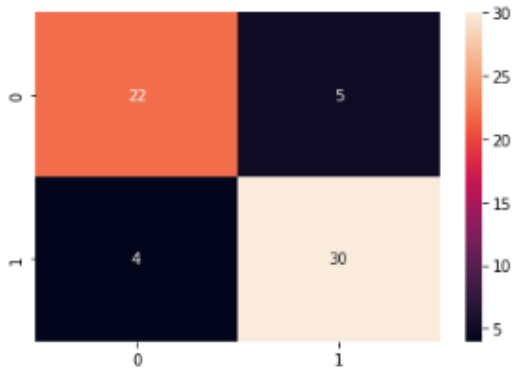
CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Results for Logistic Regression

Confusion Matrix

Out[34]: <AxesSubplot:>



Precision Score

```
Precision:  0.8571428571428571
```

Recall

```
Recall is:  0.8823529411764706
```

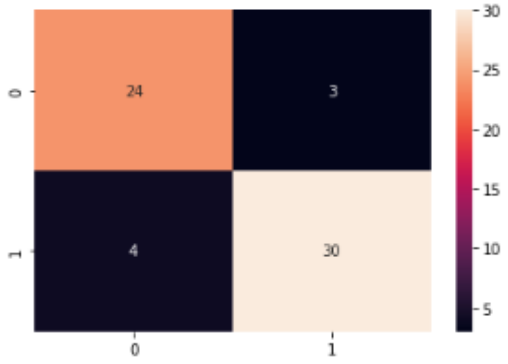
F-Score

```
0.8695652173913043
```

5.2 Results for Random Forest

Confusion Matrix

```
Out[53]: <AxesSubplot:>
```



Precision Score

```
Precision:  0.9090909090909091
```

Recall

```
Recall is:  0.8823529411764706
```

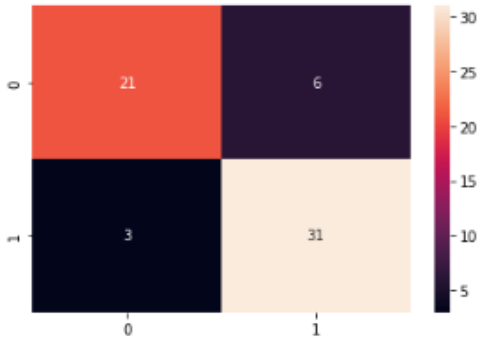
F-Score

```
Out[69]: 11.764705882352942
```

5.3 Results for Naïve Bayes

Confusion Matrix

```
Out[76]: <AxesSubplot:>
```



Precision Score

```
Precision:  0.8378378378378378
```

Recall

```
Recall is:  0.9117647058823529
```

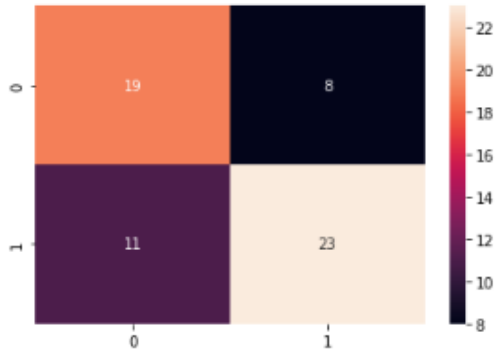
F-Score

```
Out[86]: 8.823529411764707
```

5.4 Results for K-Nearest Neighbours

Confusion Matrix

```
Out[93]: <AxesSubplot:>
```



Precision Score

```
Precision:  0.7419354838709677
```

Recall

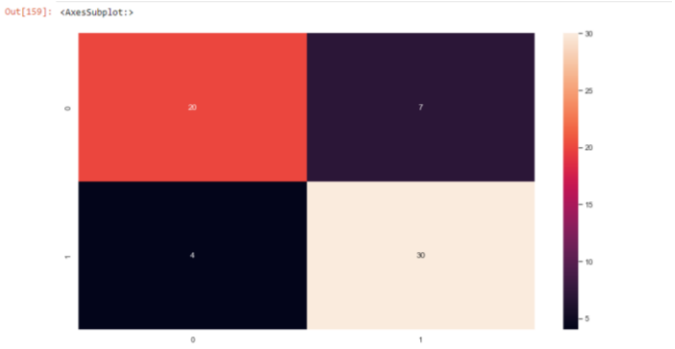
```
Recall is:  0.6764705882352942
```

F-Score

```
Out[105]: 32.35294117647059
```

5.5 Results for Support Vector Machines

Confusion Matrix



Precision Score

```
Precision:  0.8108108108108109
```

Recall

```
Recall is:  0.8823529411764706
```

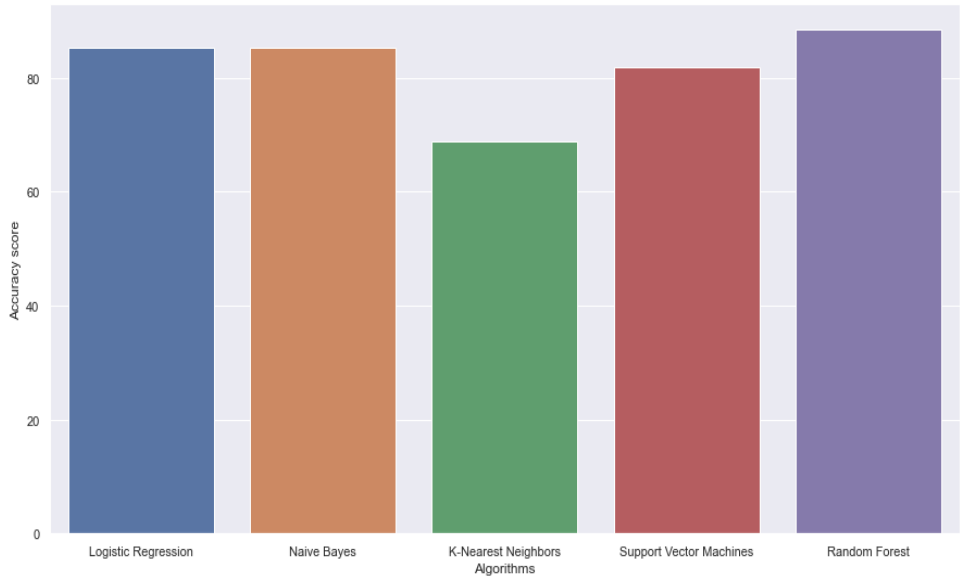
F-Score

```
Out[169]: 11.764705882352942
```

5.6 Comparative Analysis Result of Machine Learning Approaches

As per the results obtained, for the considered problem random forest algorithm provide the optimal result with an accuracy of value 0.88.

Out[172]: <AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>



Out[171]:

	accuracy
KNN	0.688525
SVM	0.459016
Logistic Regression	0.852459
Naive Bayes	0.852459
Random Forests	0.885246

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Inference

The overall objective of our project is to predict accurately with less number of tests and attributes the presence of heart disease. In this project, fourteen attributes are considered which form the primary basis for tests and give accurate results more or less. Many more input attributes can be taken but our goal is to predict with less number of attributes and faster efficiency to predict the risk of having heart disease at a particular age span. Five data mining classification techniques were applied namely K-Nearest Neighbor, Naive Bayes, Support Vector Machines, Random Forest & Logistic Regression. It is shown that Random Forest has better accuracy than the other techniques.

This is the most effective model to predict patients with heart disease. This project could answer complex queries, each with its own strength with respect to ease of model interpretation, access to detailed information and accuracy.

This project can be further enhanced and expanded. For example, it can incorporate other medical attributes besides the 14 attributes we used. It can also incorporate other data mining techniques, e.g., Time Series, Clustering and Association Rules. Continuous data can also be used instead of just categorical data. Another area is to use Text Mining to mine the vast amount of unstructured data available.

This project is presented using data mining techniques. From logistic regression, KNN, Naive Bayes, Support Vector Machines, Random Forest are used to develop the system. Random Forest proves the better results and assists the domain experts and even the person related to the medical field to plan for a better and early diagnosis for the patient. This system performs realistically well even without retraining.

6.2 Drawbacks

The Algorithms used in our project does not give a 100% accuracy, so the prediction is not 100% feasible. Clinical diagnosis and diagnosis using our project may differ slightly because the prediction is not 100% accurate. Medical diagnosis is considered as a significant yet intricate task that needs to be carried out precisely and efficiently. The automation of the same would be highly beneficial. Clinical decisions are often made based on doctor's intuition and experience rather than on the knowledge rich data collected from the dataset.

6.3 Future Scope

We are planning to introduce an efficient disease prediction system to predict the heart disease with better accuracy using Logistic Regression. Our project aims to provide a web platform to predict the occurrences of disease based on various symptoms. The user can select various symptoms and can find the diseases with their probabilistic figures. Our project can be improved by implementing medicine suggestion to the patient along with the results. We can implement a feedback from the experienced doctors who can give their views and opinions about certain medicines /practices done by the doctor on the patient.

We can implement a live chat option where the patient can chat with a doctor available regarding medication for the respective result for their symptoms. Our project could be used as a training tool for Nurses and Doctors who are freshly introduced in the field related to heart diseases. The patient can have a choice in choosing the medicines he/she should take in order to have a healthier life. Moreover, if implemented on a large scale it can be used in medical facilities like hospital, clinics where a patient wouldn't have to wait in long queues for treatment if he is feeling symptoms related to heart disease.

CHAPTER 7

REFERENCES

1. "HEART DISEASE PREDICTION USING MACHINE LEARNING", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.7, Issue 6, page no.2081-2085, June-2020, Available :http://www.jetir.org/papers/JETIR2006301.pdF
2. Apurb Rajdhan , Avi Agarwal , Milan Sai , Dundigalla Ravi, Dr. Poonam Ghuli, 2020, Heart Disease Prediction using Machine Learning, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 04 (April 2020)
3. Shah, D., Patel, S. & Bharti, S.K. Heart Disease Prediction using Machine Learning Techniques. SN COMPUT. SCI. 1, 345 (2020). https://doi.org/10.1007/s42979-020-00365
4. Harshit Jindal et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1022 012072
5. Aadar Pandita, Siddharth Vashisht, Aryan Tyagi, Prof. Sarita Yadav."Prediction of Heart Disease using Machine Learning Algorithms", Volume 9, Issue V, International Journal for Research in Applied Science and Engineering Technology (IJRASET) Page No: 2422-2429, ISSN : 2321-9653, www.ijraset.com
6. N. Saranya, P. Kaviyarasu, A. Keerthana, C. Oveya. Heart Disease Prediction using Machine Learning International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-9 Issue-1, May 2020, Page No: 700-70
7. Akella, Aravind and Akella, Sudheer. Machine learning algorithms for predicting coronary artery disease: efforts toward an open source solution. Future Science OA Volume 7, Number 6, Pages FSO698, 2021, https://doi.org/10.2144/fsoa-2020-0206
8. Ravindhar NV, Anand, Hariharan Shanmugasundaram, Ragavendran, Godfrey Winster. Intelligent Diagnosis of Cardiac Disease Prediction using Machine Learning. Volume-8 Issue-11, September 2019, ISSN: 2278-3075 (Online). Page No: 1417-1421. DOI: 10.35940/ijitee.J9765.0981119
9. International Journal of Engineering Research & Technology (IJERT)Heart Disease Prediction using Machine Learning ISSN: 2278-0181 Vol. 9 Issue 04, April-2020
10. ICCRDA(2020) volume1022 Heart disease prediction using machine learning algorithms Harshit Jindal et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1022 012072

11. International Journal of Research in Engineering, Science and Management Abhijeet Jagtap Volume-2, Issue-2, February-2019 Heart Disease Prediction using Machine Learning ISSN (Online): 2581-5792
12. Heart disease prediction using supervised machine learning algorithms: Performance analysis and comparison ; Department of Software Engineering (SWE), Daffodil International University (DIU), Sukrabad, Dhaka, 1207, Bangladesh
13. Rani, P., Kumar, R., Ahmed, N.M.O.S. et al. A decision support system for heart disease prediction based upon machine learning. J Reliable Intell Environ 7, 263–275 (2021)
14. Fusion-Based Machine Learning Architecture for Heart Disease Prediction ; CMC-COMPUTERS MATERIALS & CONTINUA, v.67, no.2, pp.2481 - 2496 (2021)
15. Apurv Garg et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1022 012046
16. Khan, Muhammad Adnan et al 2021 ; CMC-COMPUTERS MATERIALS & CONTINUA, v.65, no.1, pp.139 - 151 ; ISSN 1546-2218
17. R. J. P. Princy, S. Parthasarathy, P. S. Hency Jose, A. Raj Lakshminarayanan and S. Jeganathan, "Prediction of Cardiac Disease using Supervised Machine Learning Algorithms," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 570-575, doi: 10.1109/ICICCS48265.2020.9121169.
18. Yuan Zhao, Erica P. Wood, Nicholas Mirin, Stephanie H. Cook, Rumi Chunara, Social Determinants in Machine Learning Cardiovascular Disease Prediction Models: A Systematic Review, American Journal of Preventive Medicine, Volume 61, Issue 4, 2021, Pages 596-605, ISSN 0749-3797
19. D. K. Chohan and D. C. Dobhal, "A Comparison Based Study of Supervised Machine Learning Algorithms for Prediction of Heart Disease," 2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), 2022, pp. 372-375, doi: 10.1109/CISES54857.2022.9844328.
20. K. S. K. Reddy and K. V. Kanimozhi, "Novel Intelligent Model for Heart Disease Prediction using Dynamic KNN (DKNN) with improved accuracy over SVM," 2022 International Conference on Business Analytics for Technology and Security (ICBATS), 2022, pp. 1-5, doi: 10.1109/ICBATS54253.2022.9758996.

CHAPTER 8

ANNEXURE

8.1 Complete Program for Heart Disease Prediction

```
# # A comparative analysis of machine learning algorithms for heart disease prediction

# # 1. Importing essential libraries

# In[2]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

get_ipython().run_line_magic('matplotlib', 'inline')

import os

print(os.listdir())

import warnings

warnings.filterwarnings('ignore')

# # 2. Importing and undeerstanding our dataset

# In[3]:

data = pd.read_csv("heart_disease_data.csv")

# In[4]:

type(data)

# In[8]:

data.shape
```

```
# In[9]:

data.head()

# In[10]:

data.describe()

# In[11]:

data.info()

# # Dataset Information(14 attributes)

# 1. age: The person's age in years

# 2. sex: The person's sex (1 = male, 0 = female)

# 3. cp: The chest pain experienced (Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic)

# 4. trestbps: The person's resting blood pressure (mm Hg on admission to the hospital)

# 5. chol: The person's cholesterol measurement in mg/dl

# 6. fbs: The person's fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)

# 7. restecg: Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)

# 8. thalach: The person's maximum heart rate achieved

# 9. exang: Exercise induced angina (1 = yes; 0 = no)

# 10. oldpeak: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot. See more here)

# 11. slope: the slope of the peak exercise ST segment (Value 1: upsloping, Value 2: flat, Value 3: downsloping)

# 12. ca: The number of major vessels (0-3)
```

13. thal: A blood disorder called thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)

14. target: Heart disease (0 = no, 1 = yes)

Heart disease risk factors to the following: high cholesterol, high blood pressure, diabetes, weight, family history and smoking .

According to another source , the major factors that can't be changed are: increasing age, male gender and heredity.

Note that thalassemia, one of the variables in this dataset, is heredity.

Major factors that can be modified are: Smoking, high cholesterol, high blood pressure, physical inactivity, and being overweight and having diabetes.

Other factors include stress, alcohol and poor diet/nutrition.

In[12]:

data.sample(69)

In[5]:

data.isnull().sum()

In[14]:

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

0 indicates no missing values

In[15]:

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

3. Exploratory Data Analysis(EDA)

In[6]:

y = data["target"]

ax = sns.countplot(data["target"])

```
target_temp = data.target.value_counts()

print(target_temp)

# In[18]:

print("Percentage of patients without heart problems:
"+str(round(target_temp[0]*100/303,2)))

print("Percentage of patients with heart problems:
"+str(round(target_temp[1]*100/303,2)))

# In[7]:

data["sex"].unique()

# In[8]:

countFemale = len(data[data.sex == 0])

countMale = len(data[data.sex == 1])

print("Percentage of Female
Patinets:{:.2f}%".format((countFemale)/(len(data.sex))*100))

print("Percentage of Male
Patients:{:.2f}%".format((countMale)/(len(data.sex))*100))

# In[10]:

sns.barplot(data["sex"],y)

# In[13]:

pd.crosstab(data.age,data.target).plot(kind="bar",figsize=(20,6))

plt.title('Heart Disease Frequency for Ages')

plt.xlabel('Age')

plt.ylabel('frequency')

plt.savefig('heartDiseaseAndAges.png')
```

```
plt.show()

# In[ ]:

pd.crosstab(data.sex,data.target).plot(kind="bar",fightsize=(20,10),color=['blue','
#AA1111'])

plt.title('Heart Disease Frequency for Sex')

plt.xlabel('Sex(0 = Female, 1 = Male)')

plt.xticks(rotation = 0)

plt.legend(["Don't have Disease","Have Disease"])

plt.ylabel('Frequency')

plt.show()

# In[14]:

data.coloumns =
['age','sex','chest_pain_type','resting_blood_pressure','cholestrol',

            'fasting_blood_sugar','rest_ecg','max_heart_rate_achieved',


'exercise_induced_angina','st_depression','st_slope','num_major_vessels',

            'thalassenia','target']

# In[17]:

data["cp"].unique()

# In[18]:

plt.figure(figsize=(26,10))

sns.barplot(data["cp"],y)

# In[19]:

data["trestbps"].unique()
```

```
# In[20]:

plt.figure(figsize=(26, 10))

sns.barplot(data["trestbps"],y)

# In[22]:

data["restecg"].unique()

# In[24]:

plt.figure(figsize=(26, 10))

sns.barplot(data["restecg"],y)

# In[25]:

data["exang"].unique()

# In[26]:

plt.figure(figsize=(26, 10))

sns.barplot(data["exang"],y)

# In[27]:

data["slope"].unique()

# In[28]:

plt.figure(figsize=(25, 10))

sns.barplot(data["slope"],y)

# # 4. Train Test dataset split

# In[41]:

from sklearn.model_selection import train_test_split

predictors =data.drop("target",axis=1)

target =data["target"]
```

```
X_train,X_test,Y_train,Y_test =
train_test_split(predictors,target,test_size=0.20,random_state=0)

print("Training features have {0} records and Testing features have {1}
records.".     format(X_train.shape[0], X_test.shape[0]))

# In[42]:

X_train.shape

# In[43]:

X_test.shape

# In[44]:

Y_train.shape

# In[23]:

Y_test.shape

# In[45]:

#Importing Accuracy Score

from sklearn.metrics import accuracy_score

# # 5. Model Fitting

# # # Modelling and predicting with Machine Learning

# The main goal of the entire project is to predict heart disease occurrence with the highest accuracy. In order to achieve this, we will test several classification algorithms. This section includes all results obtained from the study and introduces the best performer according to accuracy metric. I have chosen several algorithms typical for solving supervised learning problems throughout classification methods.

# First of all, let's equip ourselves with a handy tool that benefits from the cohesion of SciKit Learn library and formulate a general function for training our models. The reason for displaying accuracy on both, train and test sets, is to allow
```

us to evaluate whether the model overfits or underfits the data (so-called bias/variance tradeoff).

In[25]:

#model : Algorithm performance testing using Confusion matrix, Precision Score, Recall, F-Score

In[26]:

#Models: Support Vector Machine(SVM), Logistic Regression(LR), K-Nearest Neighbors, naive Bayes, Random Forest

In[46]:

```
def train_model(X_train, y_train, X_test, y_test, classifier, **kwargs):
    """
    Fit the chosen model and print out the score.
    """
    # instantiate model
    model = classifier(**kwargs)
    # train model
    model.fit(X_train,y_train)
    # check accuracy and print out the results
    fit_accuracy = model.score(X_train, y_train)
    test_accuracy = model.score(X_test, y_test)
    print(f"Train accuracy: {fit_accuracy:0.2%}")
    print(f"Test accuracy: {test_accuracy:0.2%}")
    return model
```

1. LOGISTIC REGRESSION

```
# In[47]:

from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(X_train, Y_train

y_pred_lr = logreg.predict(X_test)

print(y_pred_lr)

# In[48]:

score_lr = round(accuracy_score(y_pred_lr,Y_test)*100,2)

print("The accuracy score achieved using Logistic Regression is:
"+str(score_lr)+" %")

# In[49]:

from sklearn.linear_model import LogisticRegression

model = train_model(X_train, Y_train, X_test, Y_test, LogisticRegression)

# In[31]:

#Logistic Regression supports only solvers in ['liblinear', 'newton-cg'<-93.44,
'lbfgs'<-91.8, 'sag'<-72.13, 'saga'<-72.13]

clf = LogisticRegression(random_state=0, solver='newton-
cg',multi_class='multinomial').fit(X_test, Y_test)

#The solver for weight optimization,'lbfgs' is an optimizer in the family of quasi-
Newton methods.

clf.score(X_test, Y_test)

# CONFUSION MATRIX

# In[32]:

from sklearn.metrics import confusion_matrix
```

```
# In[33]:

matrix= confusion_matrix(Y_test, y_pred_lr)

# In[34]:

sns.heatmap(matrix,annot = True, fmt = "d")

#FMT = D IS FORMAT = DEFAULT

# PRECISION SCORE

# In[35]:

from sklearn.metrics import precision_score

# In[36]:

precision = precision_score(Y_test, y_pred_lr)

# In[37]:

print("Precision: ",precision)

# RECALL

# In[38]:

from sklearn.metrics import recall_score

# In[39]:

recall = recall_score(Y_test, y_pred_lr)

# In[40]:

print("Recall is: ",recall)

# F-SCORE

# In[41]:

#balance of precision and recall score

print((2*precision*recall)/(precision+recall))
```

```
# In[42]:

#cm using bad style

CM =pd.crosstab(Y_test, y_pred_lr)

CM

# In[43]:

TN=CM.iloc[0,0]

FP=CM.iloc[0,1]

FN=CM.iloc[1,0]

TP=CM.iloc[1,1]

# In[44]:

#False Negative rate of the model

fnr=FN*100/(FN+TP)

fnr

# # 2. RANDOM FOREST

# In[45]:

from sklearn.ensemble import RandomForestClassifier

randfor = RandomForestClassifier(n_estimators=100, random_state=0)

randfor.fit(X_train, Y_train)

y_pred_rf = randfor.predict(X_test)

print(y_pred_rf)

# Learning curve for Training score & cross validation score

# In[46]:

from sklearn.model_selection import learning_curve
```

```
# Create CV training and test scores for various training set sizes
train_sizes, train_scores, test_scores =
learning_curve(RandomForestClassifier(),
                                X_train,
                                Y_train,
                                # Number of folds in cross-validation
                                cv=10,
                                # Evaluation metric
                                scoring='accuracy',
                                # Use all computer cores
                                n_jobs=-1,
                                # 50 different sizes of the training set
                                train_sizes=np.linspace(0.01, 1.0, 50))
# Create means and standard deviations of training set scores
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
# Create means and standard deviations of test set scores
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)
# Draw lines
plt.plot(train_sizes, train_mean, '--', color="#111111",  label="Training score")
plt.plot(train_sizes, test_mean, color="#111111", label="Cross-validation
score")
```

```
# Draw bands

plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="#DDDDDD")

plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, color="#DDDDDD")

# Create plot

plt.title("Learning Curve")

plt.xlabel("Training Set Size"), plt.ylabel("Accuracy Score"), plt.legend(loc="best")

plt.tight_layout()

plt.show()

# In[47]:

score_rf = round(accuracy_score(y_pred_rf,Y_test)*100,2)

print("The accuracy score achieved using Random Forest is: "+str(score_rf)+" %")

# In[48]:

#Random forest with 100 trees

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, random_state=0)

rf.fit(X_train, Y_train)

print("Accuracy on training set: {:.3f}".format(rf.score(X_train, Y_train)))

print("Accuracy on test set: {:.3f}".format(rf.score(X_test, Y_test)))

# In[49]:

#Pruning ddepth of the trees and checking accuracy
```

```
rf1 = RandomForestClassifier(max_depth=3, n_estimators=100,
random_state=0)

rf1.fit(X_train, Y_train)

print("Accuracy on training set: {:.3f}".format(rf1.score(X_train, Y_train)))

print("Accuracy on test set: {:.3f}".format(rf1.score(X_test, Y_test)))

# performance metrics

# -Accuracy: is the ratio between the number of correct predictions and total
number of predications.

# $acc = \frac{TP + TN}{TP + TN + FP + FN}$

# -Precision: is the ratio between the number of correct positives and the number
of true positives plus the number of false positives.

# $Precision (p) = \frac{TP}{TP + FP}$

# -Recall: is the ratio between the number of correct positives and the number of
true positives plus the number of false negatives.

# $recall = \frac{TP}{TP + FN}$

# -F-score: is known as the harmonic mean of precision and recall.

# $acc = \frac{1}{\frac{1}{2}(\frac{1}{p}+\frac{1}{r})} = \frac{2pr}{p+r}$

# -Problem characteristics in context of our case study:

# TP = True positive (has heart disease). TN = True negative (has no heart
disease). FP = False positive (has no heart disease) FN = False negative (has
heart disease)

# CONFUSION MATRIX

# In[50]:

from sklearn.metrics import confusion_matrix

# In[52]:
```

```
matrix= confusion_matrix(Y_test, y_pred_rf)
# In[53]:
sns.heatmap(matrix,annot = True, fmt = "d")
# PRECISION SCORE
# In[55]:
from sklearn.metrics import precision_score
# In[56]:
precision = precision_score(Y_test, y_pred_rf)
# In[57]:
print("Precision: ",precision)
# RECALL
# In[62]:
from sklearn.metrics import recall_score
# In[63]:
recall = recall_score(Y_test, y_pred_rf)
# In[65]:
print("Recall is: ",recall)
# F SCORE
# In[66]:
print((2*precision*recall)/(precision+recall))
# In[67]:
#CM USING BAD STYLE
CM =pd.crosstab(Y_test, y_pred_rf)
```

```
CM
# In[68]:
TN=CM.iloc[0,0]
FP=CM.iloc[0,1]
FN=CM.iloc[1,0]
TP=CM.iloc[1,1]
# In[69]:
#FALSE NEGATIVE RATE
fnr=FN*100/(FN+TP)
fnr
# # 3. NAIVE BAYES
# In[70]:
from sklearn.naive_bayes import GaussianNB
nb = train_model(X_train, Y_train, X_test, Y_test, GaussianNB)
nb.fit(X_train, Y_train)
y_pred_nb = nb.predict(X_test)
print(y_pred_nb)
# In[71]:
score_nb = round(accuracy_score(y_pred_nb,Y_test)*100,2)
print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")
# In[72]:
#Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
```

```
model = train_model(X_train, Y_train, X_test, Y_test, GaussianNB)
# CONFUSION MATRIX
# In[74]:
from sklearn.metrics import confusion_matrix
# In[75]:
matrix= confusion_matrix(Y_test, y_pred_nb)
# In[76]:
sns.heatmap(matrix,annot = True, fmt = "d")
# PRECISION SCORE
# In[77]:
from sklearn.metrics import precision_score
# In[78]:
precision = precision_score(Y_test, y_pred_nb)
# In[79]:
print("Precision: ",precision)
# RECALL
from sklearn.metrics import recall_score
# In[81]:
recall = recall_score(Y_test, y_pred_nb)
# In[82]:
print("Recall is: ",recall)
# F_SCORE
# In[83]:
```

```
print((2*precision*recall)/(precision+recall))

# In[84]:

#BAD CM STYLE

CM = pd.crosstab(Y_test, y_pred_nb)

CM

# In[85]:

TN=CM.iloc[0,0]

FP=CM.iloc[0,1]

FN=CM.iloc[1,0]

TP=CM.iloc[1,1]

# In[86]:

#FALSE NEGATIVE RATE OF THE MODEL

fnr = FN*100/(FN+TP)

fnr

# # 4. KNN(K NEAREST NEIGHBORS)

# In[87]:

from sklearn.neighbors import KNeighborsClassifier

knn = train_model(X_train, Y_train, X_test, Y_test, KNeighborsClassifier,
n_neighbors=8)

knn.fit(X_train, Y_train)

y_pred_knn = knn.predict(X_test)

print(y_pred_knn)

# In[88]:
```

```
score_knn = round(accuracy_score(y_pred_knn,Y_test)*100,2)

print("The accuracy score achieved using KNN is: "+str(score_knn)+" %")

# In[89]:

# KNN

from sklearn.neighbors import KNeighborsClassifier

model = train_model(X_train, Y_train, X_test, Y_test, KNeighborsClassifier)

# Let's see if KNN can perform even better by trying different 'n_neighbours' inputs.

# In[90]:

# Seek optimal 'n_neighbours' parameter

for i in range(1,10):

    print("n_neigbors = "+str(i))

 train_model(X_train, Y_train, X_test, Y_test, KNeighborsClassifier, n_neighbors=i)

# It turns out that value of n_neighbours (8) is optimal.

# CONFUSION MATRIX

# In[91]:

from sklearn.metrics import confusion_matrix

# In[92]:

matrix= confusion_matrix(Y_test, y_pred_knn)

# In[93]:

sns.heatmap(matrix,annot = True, fmt = "d"

# PRECISION SCORE
```

```
# In[94]:

from sklearn.metrics import precision_score

# In[95]:

precision = precision_score(Y_test, y_pred_knn)

# In[96]:

print("Precision: ",precision)

# RECALL

# In[98]:

from sklearn.metrics import recall_score

# In[99]:

recall = recall_score(Y_test, y_pred_knn)

# In[100]:

print("Recall is: ",recall)

# F_SCORE

# In[101]:

print((2*precision*recall)/(precision+recall))

# In[102]:

#BAD CM STYLE

CM = pd.crosstab(Y_test, y_pred_knn)

CM

# In[104]:

TN=CM.iloc[0,0]

FP=CM.iloc[0,1]
```

```
FN=CM.iloc[1,0]

TP=CM.iloc[1,1]

# In[105]:

#FALSE NEGATIVE RATE OF THE MODEL

fnr = FN*100/(FN+TP)

fnr

# # 5. SVM(SUPPORT VECTOR MACHINE)

# In[155]:

from sklearn import svm

from sklearn.svm import LinearSVC

sv = svm.SVC(kernel='linear')

sv.fit(X_train, Y_train)

Y_pred_svm = sv.predict(X_test)

print(Y_pred_svm)

# In[156]:

score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)

print("The accuracy score achieved using Linear SVM is: "+str(score_svm)+"
%")

# In[157]:

from sklearn.metrics import confusion_matrix

# In[158]:

matrix= confusion_matrix(Y_test, Y_pred_svm)

# In[159]:
```

```
sns.heatmap(matrix,annot = True, fmt = "d")
# PRECISION SCORE
# In[160]:
from sklearn.metrics import precision_score
# In[161]:
precision = precision_score(Y_test, Y_pred_svm)
# In[162]:
print("Precision: ",precision)
# RECALL
# In[163]:
from sklearn.metrics import recall_score
# In[164]:
recall = recall_score(Y_test, Y_pred_svm)
# In[165]:
print("Recall is: ",recall)
# F_SCORE
# In[166]:
print((2*precision*recall)/(precision+recall))
# In[167]:
#BAD CM STYLE
CM = pd.crosstab(Y_test, Y_pred_svm)
CM
# In[168]:
```

```
TN=CM.iloc[0,0]

FP=CM.iloc[0,1]

FN=CM.iloc[1,0]

TP=CM.iloc[1,1]

# FALSE NEGATIVE RATE

# In[169]:

fnr = FN*100/(FN+TP)

fnr

# F1 score

# LOGISTIC REGRESSION: 0.87

# RANDOM FOREST: 0.89

# NAIVE BAYES: 0.87

# KNN: 0.707

# SVM: 0.84

# # FINAL SCORE

# In[170]:

# initialize an empty list

accuracy = []

# list of algorithms names

classifiers = ['KNN', 'SVM', 'Logistic Regression', 'Naive Bayes', 'Random
Forests']

# list of algorithms with parameters
```

```
models = [KNeighborsClassifier(n_neighbors=8), LinearSVC(random_state=0, tol=1e-05), LogisticRegression(),
      GaussianNB(), RandomForestClassifier(n_estimators=100, random_state=0)]
# loop through algorithms and append the score into the list
for i in models:
   model = i
   model.fit(X_train, Y_train)
   score = model.score(X_test, Y_test)
   accuracy.append(score)
# In[171]:
# create a dataframe from accuracy results
summary = pd.DataFrame({'accuracy':accuracy}, index=classifiers)
summary
# In[172]:
scores = [score_lr,score_nb,score_knn,score_svm,score_rf]
algorithms = ["Logistic Regression","Naive Bayes","K-Nearest Neighbors","Support Vector Machines","Random Forest"]
sns.set(rc={'figure.figsize':(15,8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")
sns.barplot(algorithms,scores).
```