EFFECTIVE CARDIOVASCULAR DISEASE PREDICTION USING MACHINE LEARNING ALGORITHMS

A PROJECT REPORT

Submitted by

DINESH K (195002032)

HARI HARA SAKTHIVEL K (195002041)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

MAY 2023

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project titled **"EFFECTIVE CARDIOVASCULAR DISEASE PREDICTION USING MACHINE LEARNING ALGORITHMS"** is the bonafide work of **DINESH K** (195002032) **and HARI HARA SAKTHIVEL K** (195002041) who carried out the work under my supervision.

Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. Chandrabose Aravindan	Dr. I. Joe Louis Paul
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Head of the Department	Associate Professor
Department of Information Technology	Department of Information Technology
SSN College of Engineering	SSN College of Engineering
Kalavakkam – 603110	Kalavakkam – 603110

Submitted for the project viva-voice Examination held on ……

EXTERNAL EXAMINER **INTERNAL EXAMINER**

ABSTRACT

In the present scenario, cardiovascular disease impacts human life very inadequately and raises the cause of death in the world. To prevent heart failure, an early precise and on time diagnosis is very significant.

Through the conventional medical record, heart disease diagnosis has not been considered reliable in many aspects. In this regard, the authors developed a novel medical diagnosis system using machine learning (ML) algorithms.

The optimal set of features is considered to enrich the proposed heart disease multi-class classification superiority by utilizing the different feature selection methods. The utilized machine learning algorithms are Random Forest (RF), Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), Extreme Gradient (XGBoost), and ensemble learning classifiers (Stacking) to perform the multi-class classification.

To determine and investigate the findings realized by ML algorithms, performance measures such as accuracy, precision, f-score, and recall are considered and observed. The results reveal that the Stacking ensemble classifier accomplished exceptional accuracy of 73.21%.

ACKNOWLEDGEMENT

We are honored to acknowledge the below mentioned people who have been instrumental in the completion of this project. We express our deep respect to our principal **Dr. V. E. Annamalai**, and to the Head of Department, **Dr. C. Aravindan**, who gave us this opportunity to do this project on the topic, **EFFECTIVE CARDIOVASCULAR DISEASE PREDICTION USING MACHINE LEARNING ALGORITHMS** as our final year project, which was a great learning experience and helped us in doing a lotof research.

Next, it is a matter of pleasure of acknowledgement my indebtedness to our project guide, **Dr. I. Joe Louis Paul**, for his needy half in the completion of the project and able guidance.

We sincerely thank our project panel members **Dr. A. Shahina, Dr. K.R. Uthayan,** and **Dr. E.M. Malathy** in the Department of Information Technology for providing us with helpful pointers throughoutthe course of the project.

Dinesh K
Hari Hara Sakthivel K

TABLE OF CONTENTS

CHAPTER NO.	TITLE			PAGE NO.
	ABSTRACT			iii
	LIST OF TABLES			vii
	LIST OF FIGURES			viii
	LIST OF SYMBOLS, ABBREVIATIONS			ix
1.	INTRODUCTION			1
	1.1	GENERAL		1
	1.2	NEED FOR THE STUDY		2
	1.3	OBJECTIVE OF THE STUDY		3
2.	LITERATURE SURVEY			4
3.	METHODOLOGY			7
	3.1	PROPOSED WORK		7
		3.1.1	DATASET	7
		3.1.2	DATAPREPROCESSING	8
		3.1.3	FEATURE SELECTION	11
		3.1.4	DATA BALANCING	11
		3.1.5	DATA PORTIONING	13
		3.1.6	CLASSIFICATION	13
	3.2	DEVELOPMENT TOOLS		14

		3.2.1	GOOGLE COLABORATORY	14
	3.3	LANGUAGES AND LIBRARIES		15
		3.3.1	MATPLOTLIB	15
		3.3.2	PANDAS	16
		3.3.3	NUMPY	16
		3.3.4	SKLEARN	17
		3.3.5	PYTHON	18
4.	MACHINE LEARNING			19
		ALGORITHMS		21
		4.1.1	RANDOM FOREST	21
		4.1.2	SUPPORT VECTOR MACHINE	24
		4.1.3	XG BOOST	27
		4.1.4	K-NEAREST NEIGHBOUR	29
		4.1.5	STACKING	32
5.	IMPLEMENTATION			35
	5.1	SAMPLE CODE		35
6.	RESULTS			47
7.	CONCLUSION AND FUTURE WORK			49
	REFERENCES			50

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
3.1.1	Dataset attributes and description	8
6.1	Various models with accuracy	47
6.2	Various models with accuracy, precision, recall and f1-score	48

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Working design	7
3.1.2	Correlation between the features	9
3.1.2	Distribution of cholesterol feature before removing outliers	10
3.1.2	Distribution of cholesterol feature after removing outliers	10
3.1.4	Distribution of outcomes before sampling	12
3.1.4	Distribution of outcomes after sampling	12
3.1.6	Formulae for various performance metrics	14
4.1.1	Working of Random Forest	23
4.1.2	Support Vector Machines	27
4.1.3	Working of XG Boost	29
4.1.4	Before and After K-NN	31
4.1.5	Working of Stacking	34
6.3	Various models with accuracy	48

LIST OF SYMBOLS, ABBREVIATIONS

ML	Machine Learning
RF	Random Forest
SVM	Support Vector Machine
KNN	K-Nearest Neighbour
XGB	XG Boost

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The heart is an important organ of the human body. It pumps blood to every 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 around 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 have become the leading cause of 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 reduce the productivity of an individual. Estimates made by the World Health Organization (WHO), suggest that India has lost up to $237 billion, from 2005- 2015, due to heart-related or cardiovascular diseases. Thus, feasible and accurate prediction of heart-related diseases is very important. Medical organizations, all around the world, collect data on various health-related issues. This data can be exploited using various machine learning techniques to gain 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 comprehend, can be easily explored using various machine learning techniques. Thus, these algorithms have become very useful, in

recent times, to predict the presence or absence of heart-related diseases accurately the usage of information technology in the health care industry is increasing day by day to aid doctors in decisionmaking activities. It helps doctors and physicians in disease management, medications, and discovery of patterns and relationships among diagnosis data. Current approaches to predict cardiovascular risk fail to identify many people who would benefit from preventive treatment, while others receive unnecessary intervention. Machine-learning offers an opportunity to improve accuracy by exploiting complex interactions between risk factors. We assessed whether machine-learning can improve cardiovascular risk prediction.

1.2 NEED FOR THE STUDY

The main motivation of doing this research is to present a heart disease prediction model for the prediction of occurrence of heart disease. Further, this research work is aimed towards 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 K-Nearest Neighbour, Support Vector Classifier, Random Forest, and Stacking are used at different levels of evaluations. Although these are commonly used machine learning algorithms, 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.

1.3 OBJECTIVE OF THE STUDY

- To develop a prediction model that detects and identifies cardiovascular disease based on patients' data using Machine Learning Algorithms.
- To evaluate for ascertaining its predictive performance using several performance metrics like accuracy, precision score, and f1 score.

CHAPTER 2

LITERATURE SURVEY

Prediction of Heart Diseases using Random Forest (2021)

The authors Madhumita Pal, Smita Parija proposed to predict the occurrence of heart disease of a patient using random forest algorithm.
Their proposed system of random forest algorithm with 86.9% accuracy was obtained. Many trees can make the algorithm too slow and ineffective. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained.

Intelligent Cardiovascular Disease Risk Estimation Prediction System (2020)

The authors Froyston Mendonca, Riyazuddin Manihar, Ashishkumar Pal, Sapna U Prabu proposed a method which uses K Nearest Neighbour algorithm. Additionally, data reduction techniques have been utilised to reduce the number of attributes. They got an accuracy of 92.30%. The drawback of the model was that accuracy remains the same even after reducing the number of attributes.

Cardiovascular Disease Prediction Using Machine Learning Models (2021)

The authors Atharv Nikam, Sanket Bhandari, Aditya Mhaske, Shamla Mantri proposed machine learning technique to predict cardiovascular disease with the help of Body Mass Index (BMI) feature using various algorithms. The accuracy of various models is 72% in decision tree, 71.85% in ANN, 71.65% in Logistic Regression, 71.60% in KNN and 70.26% in Naïve Bayesian. The drawback was prediction is not accurate using limited feature.

Multi-model Ensemble Based Approach for Heart Disease Diagnosis (2022)

Areel Ashfaq, Azhar Imran, Inam Ullah, Abdulkareem Alzahrani, Khattab M. Ali Alheeti proposed a model which has techniques based on ensemble learning methods. Ensemble methods involve voting classifiers, stacking, bagging, and boosting models. The accuracy of models is 87% in combination of RF GNB, and KNC, 88% in bagging, 86% in soft voting classifier, 86% in Adaboost. The drawback is significant overhead.

Heart Disease Prediction using a Stacked Ensemble of Supervised Machine Learning Classifiers (2022)

Nazim Nisar Itoo, Vijay Kumar Garg proposed stacked ensemble classifier is made using Logistic Regression, K Nearest Neighbors, and Naïve Bayes classifiers as base learners proves to be highly efficient and can, therefore, be used to assist health care practitioners. The accuracy of the models is 86.67% in LR, 88.33% in KNN, 86.67% in Naïve Bayesian, 90% in Stacking Classifier. The drawback is when volume of data increases, computation time will also increase.

CHAPTER 3

METHODOLOGY

3.1 PROPOSED WORK

In heart disease based medical diagnosis system, there are input and output variables that are disease risk aspects and categories such as heart disease absence and heart disease presence. The heart disease presence includes mild heart disease, moderate heart disease, severe heart disease, and very severe heart disease. The proposed medical diagnosis system steps for multi-class heart disease classification are shown in Fig. 1. Important concepts such as dataset detail and source, data preprocessing, feature engineering, classification modelling is described following.

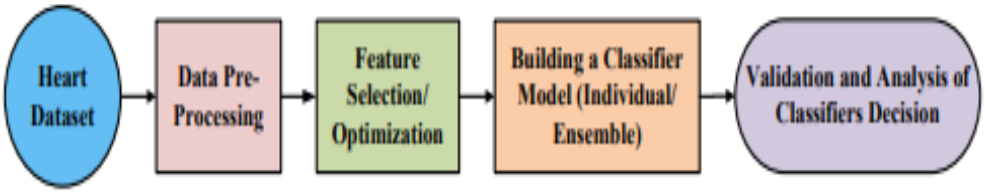

Figure 3.1: *Working Design*

3.1.1 Dataset

The University of California Irvine (UCI) ML repository is utilized, from which the Cleveland dataset is selected for this work. The dataset contains 303 events of patient data, and 76 features, however only 14 of them are revealed by available studies. The “objective” area has diverging values from 0 to 4 indicates if heart disease is absence or presence in the patient.

Table 3.1.1: *Dataset attributes and description*

Clinical Attributes	Description
Age	Person age
Sex	Gender
Ca	Major vessels colored by flourosopy (0-3)
Chol(mg/dl)	Cholesterol fluid
Cp	Types of chest pain
Exang	Exercise caused angina
Fbs	Fasting blood sugar
Oldpeak	ST depression
Restecg	Resting electrocardiographic
Slope	ST segment slope
Thal	Thallium scan types
Thalach	Extreme heart rate attained
Trestbps(mmHg)	Resting blood pressure
Outcomes	Heart Disease Absence: 0: No Heart Disease Heart Disease Presence: 1: Mild Heart Disease 2: Moderate Heart Disease 3: Severe Heart Disease 4: Very Severe Heart Disease

3.1.2 Data Preprocessing

The data are preprocessed after the gathering. There are 6 entries with missing estimates in the dataset. All the missing estimates are excluded from the dataset, thus reducing the total entries from 303 to 297. Subsequently, the exploratory data analysis (EDA) has been performed to analyze the dataset. The categorical variables are examined by using the bar graphs, while the continuous variables are distributed into three categories including the

univariate, bivariate, and multivariate analyses followed by the KDE and box plots. The categorical data is encoded into the numeric values as required by the machine learning models. In the next step, outliers are detected and removed within the dataset which increases the variability in the data and reduces the accuracy and performance of the model by using the Inter-Quartile Range (IQR) method. Likewise, from all features, the outliers are detected and removed.

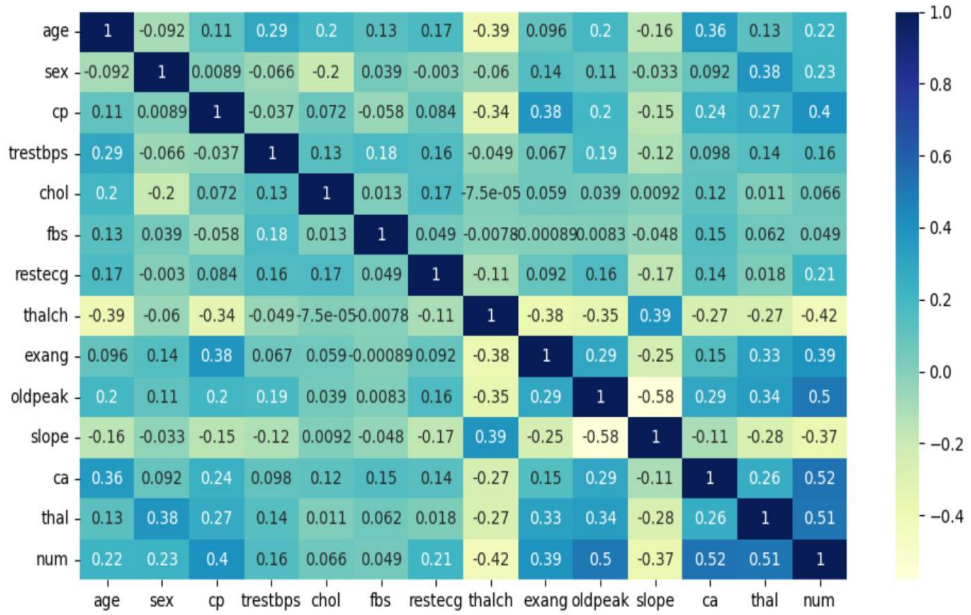

Figure 3.1.2: *Correlation between the features.*

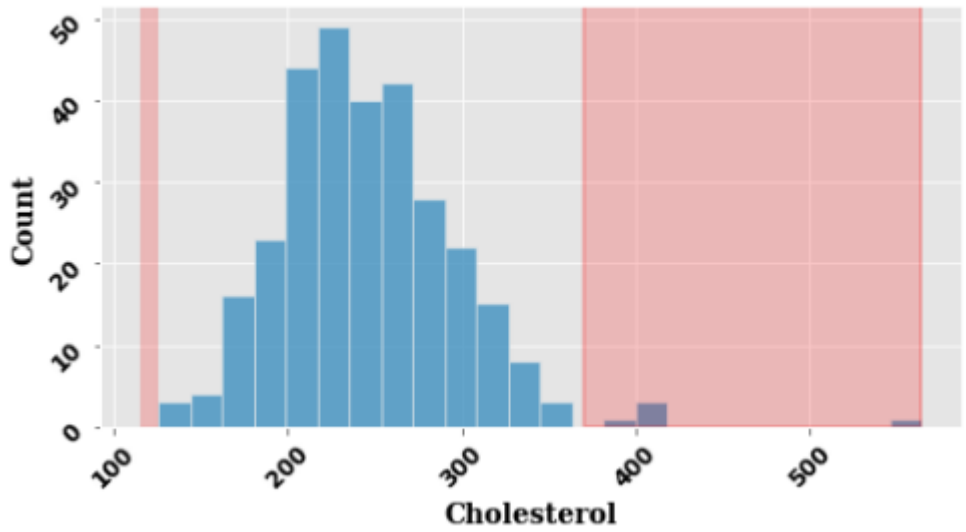

Figure 3.1.2: *Distribution of Cholesterol feature before removing outliers.*

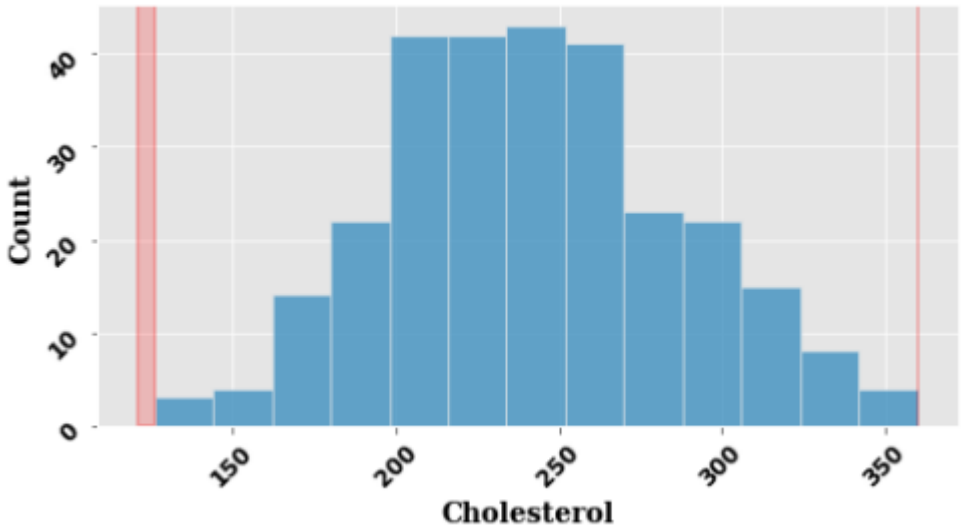

Fig 3.1.2: *Distribution of Cholesterol feature after removing outliers.*

3.1.3 Feature Selection

The irrelevant features in the dataset frequently impact the training model, and a few noise variables drive the model to diverge from the precise way. Feature selection illustrates the input data and guarantees good prediction outcomes by selecting the subset of variables. In heart disease prediction, from the original dataset of 76 features, only 14 are selected in which 'age' and 'sex' features describe the individual knowledge of the patient, the resting 12 are the clinical attributes obtained from the several medical assessments. Different feature selection methods comprise Pearson Correlation Coefficient, Linear SVC, Lasso, Chi-2, Recursive Feature Elimination (RFE) and Linear Regression, RFE and Random Forest Classifier, and Variance Threshold are utilized to determine the optimum batch of features, and later take the intersection of features generated by the different methods. The objective of using multiple methods for the selection of features is to distinguish the features in a training dataset that are significant in predicting the outcomes. Through feature selection, the authors decrease the extent of features and choose the best dependable feature subset.

3.1.4 Data Balancing

To balance out the outcomes, a Synthetic Minority Over-Sampling Technique (SMOTE) method is employed. In this method, the majority outcomes are under-samples, while the minority outcomes are over-samples, besides introducing a few instances of the minority to fill certain features space for the outcomes rather than just oversample with an alternative or creating multiple copies of occurrences.

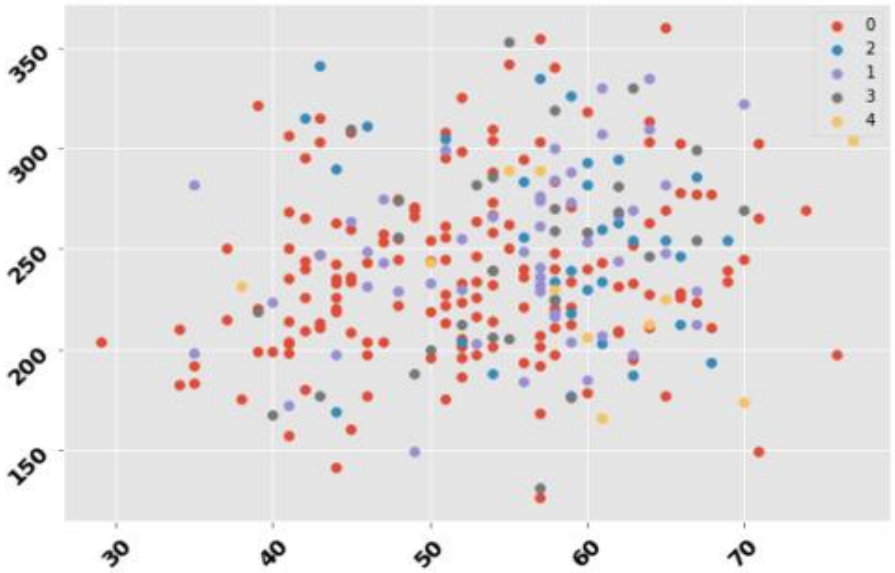

Figure 3.1.4: *Distribution of outcomes before sampling*

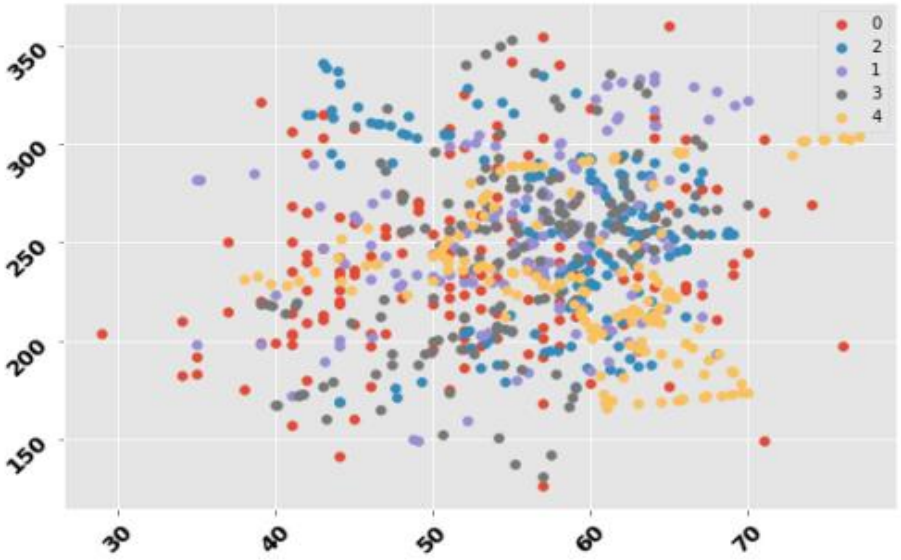

Figure 3.1.4: *Distribution of outcomes after sampling*

3.1.5 Data Portioning Technique

A cross-validation method is accomplished on the data to get certain assessments, where data is split into 80% and 20% for training and testing, respectively. Data portioning has been performed due to the constraints in the amount of data by utilizing the k-fold cross-validation method. This method has lower variance in contrast with other estimators such as the single hold-out method. Besides, large tests on several various datasets with different methods indicated that the value of k equal to 10 is the correct number of the fold to acquire the best estimate of error.

3.1.6 Classification

In this work, six classifiers namely SVC, RF, XGB, KNN, and a combination of these classifiers utilizing ensemble learning techniques such as Stacking are discussed. The proposed ensemble classifiers employed RF, XGB, KNN, and SVC as a base learner to be extremely competent. In each method, the execution is examined by utilizing standard metrics namely accuracy, recall, precision, and f-score.

Accuracy verifies how accurately the outcomes are predicted. Precision defines how many predictions are accurate. Recall reveals several accurate outcomes were obtained. F-score uses both precision and recall computing a score that can be construed as an averaging of both terms. The subsequent equations demonstrate how to compute these values, where the terminologies TN, FN, TP, and FP describe true negative, true positive, false negative, and false positive, respectively.

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

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

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

$$F - Score = \frac{2 \times Precision \times Recall}{Precision \times Recall}$$

Figure 3.1.6: *Formulae for various performance metrics.*

3.2 DEVELOPMENT ENVIRONMENT AND TOOLS

3.2.1 Google Colaboratory

Google Colaboratory or Colab is a product of Google which is well suited for machine learning applications. It offers free resources like GPU which helps users to execute codes that require huge computational power with ease, and it requires no setup or any complicated procedures. However, runtimes expire if you go an extended period without using the application, which means that all data you uploaded and any results you got are lost. The use of colab has various benefits. First, it is free even though there are paid plans that offer superior resources (albeit the free plan itself already comes enough resources). Second, not a lot of a computer's storage or processing power is lost. This is because libraries need not be downloaded and made available on your computer rather it can be imported in colab itself. It comes with a lot of pre-

installed libraries like numpy, scipy, tensorflow, pandas, pytorch etc. Colab is easy to use, collaborate and share. The codes executed can be downloaded as .ipynb file or a .py file. It is possible to mount google drive onto colab. Thus, datasets can be uploaded to drive as zip file which later can be downloaded and unzipped. This process is much faster, simplified, easier and less time-consuming. Libraries like matplotlib allow us to create plots and other visual representations which helps in understanding what's really happening in our code. Another remarkable feature of colab is that it can contain both text and code in the same document. Users can create headings or give descriptions of what is really going on in the cell below or above. Colab notebooks can be shared via Google drive, GitHub, or other online platforms. Single notebooks can be used by multiple users at the same time.

3.3 LANGUAGES AND LIBRARIES

3.3.1 Matplotlib

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in Python. It provides a wide variety of customizable 2D and 3D plots and charts, including scatter plots, line plots, bar charts, histograms, and more Matplotlib has a wide range of uses, including data exploration and scientific visualization as well as web applications and graphical user interfaces Some of the key features of Matplotlib include:

- Easy to use and learn: It provides a straightforward and intuitive API for creating plots and charts.

- Customizable: Lets you personalize practically every aspect of your

plot, including colors, line styles, typefaces, axes, and more.

- Multiple output formats: Can generate plots in a variety of output formats, including PNG, PDF, SVG, and more.
- Integration with other Python libraries: Can be used with other prominent Python libraries, such as NumPy, Pandas, and SciPy.

Matplotlib is widely used in the scientific community and has a large and active user community. It is available for free and can be installed using Python's pip package manager.

3.3.2 Pandas

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

3.3.3 NumPy

NumPy is a popular Python library for computing that provides support for large, multi-dimensional arrays and matrices, as well as a wide range of mathematical functions to operate on these arrays. It is widely used in data analysis, machine learning, image processing, and other scientific computing applications. Some of the key features of NumPy include:

- Multi-dimensional arrays: Provides support for creating and manipulating multi-dimensional arrays with many elements. These

arrays can be easily reshaped, sliced, and indexed.

- Mathematical functions: Provides a wide range of mathematical functions, such as trigonometric, logarithmic, and exponential functions, as well as linear algebra and statistical functions.

- Broadcasting: Provides a powerful feature called broadcasting, which allows mathematical operations to be performed on arrays of different shapes and sizes.

- Efficient computation: Written in C and provides fast and efficient computation of arrays and mathematical functions.

NumPy is open-source and freely available for use in commercial and non-commercial applications. It can be installed using Python's pip package manager and is often used in conjunction with other Python libraries, such as Matplotlib and Pandas.

3.3.4 Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

3.3.5 Python

Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant White space. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

CHAPTER 4

MACHINE LEARNING

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

1.Supervised Learning

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

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

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

2.Unsupervised learning

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

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

3.Reinforcement learning

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

4.1 ALGORITHMS

4.1.1 RANDOM FOREST ALGORITHM

Random Forest is a supervised learning algorithm. It is an extension of machine learning classifiers which include the bagging to improve the performance of Decision Tree. It combines tree predictors, and trees are dependent on a random vector which is independently sampled. The distribution of all trees is the same. Random Forests splits nodes using the best among of a predictor subset that are randomly chosen from the node itself, instead of splitting nodes based on the variables. The time complexity of the worst case of learning with Random Forest is O(M(dnlogn)), where M is the number of growing trees, n is the number of instances, and d is the data dimension. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest consists of trees. It is said that the more trees it has, the more robust a forest is. Random Forests create Decision Trees on randomly selected data samples, get predictions from each tree and select the best solution by means of voting. It also provides a pretty good indicator of the feature importance. Random Forests have a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity, and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset. Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the

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

Assumptions:

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

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

Algorithm Steps:

It works in four steps:

- Select random samples from a given dataset.
- Construct a Decision Tree for each sample and get a prediction result from each Decision Tree.
- Perform a vote for each predicted result.
- Select the prediction result with the most votes as the final

prediction.

Advantages:

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

Disadvantages:

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

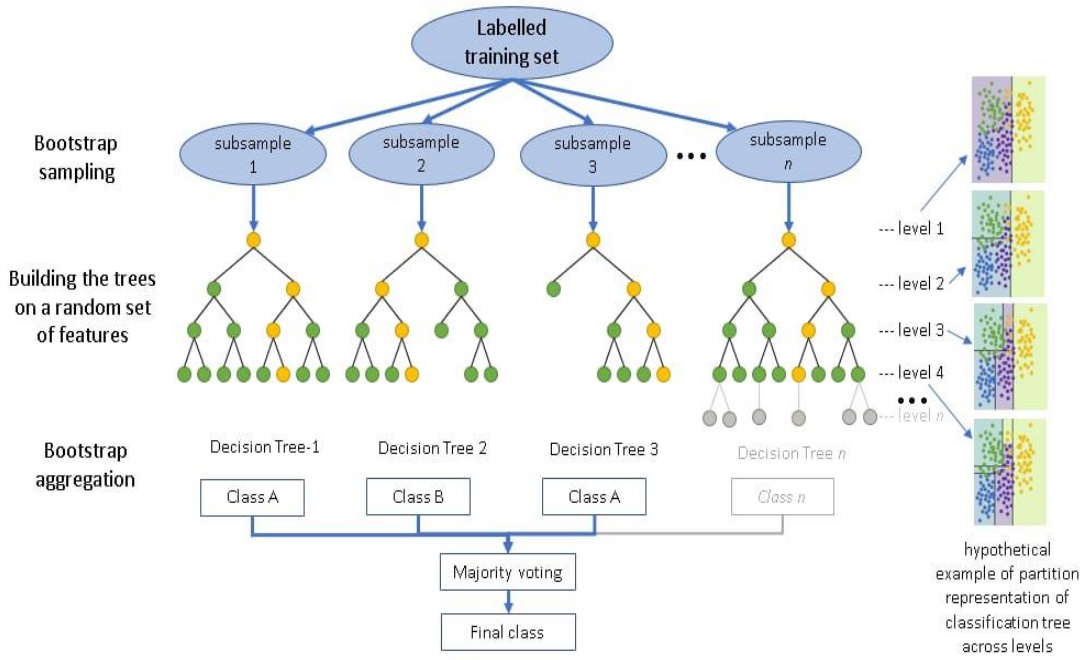

Figure 4.1.1: *Working of Random Forest*

4.1.2 SUPPORT VECTOR CLASSIFIER

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

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

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In the 1960s, SVMs were first introduced but later they got refined in 1990.

SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

The following are important concepts in SVM:

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

- Hyperplane - As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- Margin - It may be defined as the gap between two lines on the closest data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. The large margin is considered a good margin and the small margin is considered a bad margin.

Types of SVM:

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

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

Advantages:

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

Disadvantages:

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

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

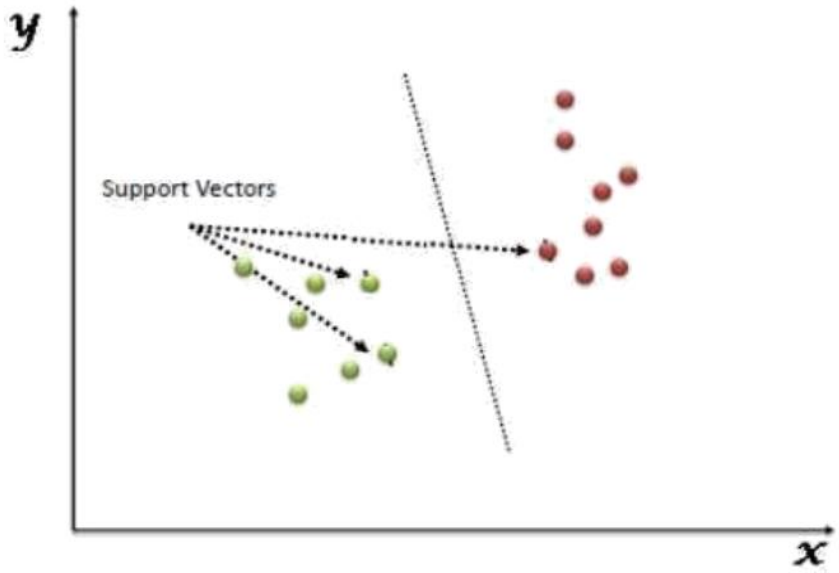

Figure 4.1.2: *Support Vector Machines*

4.1.3 XG BOOST

XG-boost is an implementation of Gradient Boosted decision trees. It is a type of Software library that was designed basically to improve speed and model performance. In this algorithm, decision trees are created in sequential form. Weight plays an important role in XG-boost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then assemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined predictions.

1. Regularization:

   XG-boost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why XG-boost is also called regularized form of GBM (Gradient Boosting Machine). While using Scikit Learn library, we pass two hyper-parameters (alpha and lambda) to XG-boost related to regularization. Alpha is used for L1 regularization and lambda is used for L2 regularization.

2. Parallel Processing:

   XG-boost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model. nthread represents number of CPU cores to be used. If you want to use all the available cores, don't mention any value for nthread and the algorithm will detect automatically.

3. Handling Missing Values:

   XG-boost has an in-built capability to handle missing values. When XG-boost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

4. Cross Validation:

   XG-boost allows the user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum

number of boosting iterations in a single run. This is unlike GBM where we must do grid-search and only a limited values can be tested.

5. Effective Tree Pruning:

   A GBM would stop splitting a node when it encounters a negative loss in the split. Thus, it is more of a greedy algorithm. XG-boost on the other hand makes splits up to the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

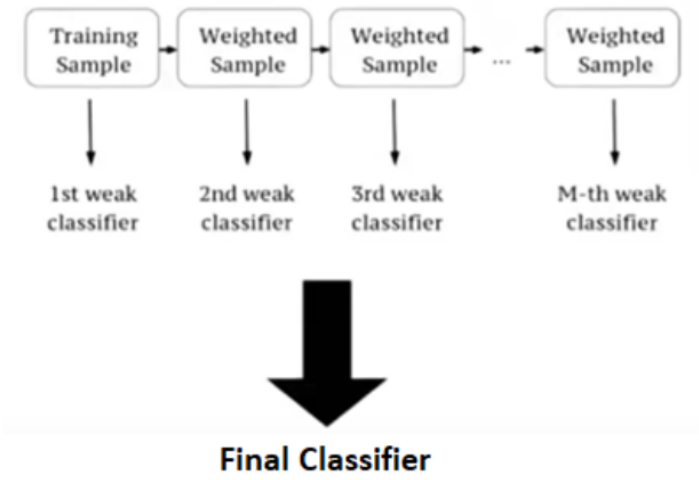

Figure 4.1.3: *Working of XG Boost*

4.1.4 K-NEAREST NEIGHBOUR

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most like the available categories. K-NN algorithm

stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much like the new data.

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.

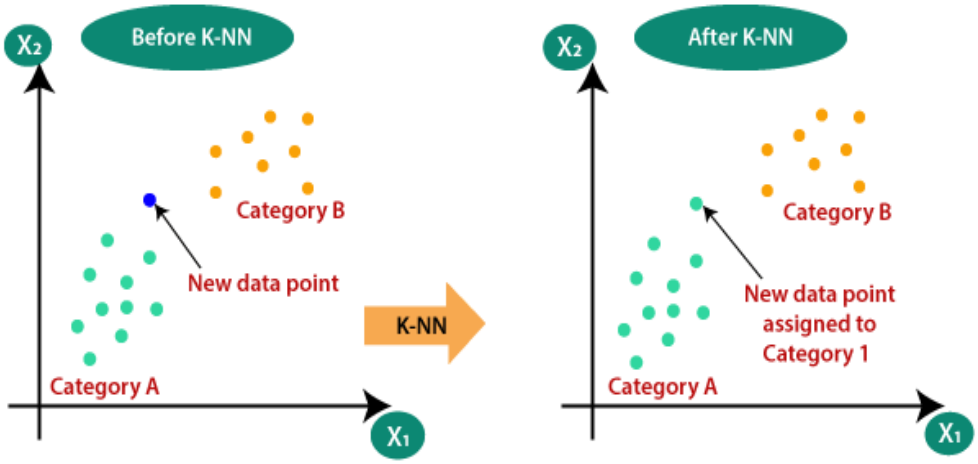

Figure 4.1.4: *Before and After K-NN*

The K-NN working can be explained based on the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Advantages:

- It is simple to implement.
- It is robust to the noisy training data.
- It can be more effective if the training data is large.

Disadvantages:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

4.1.5 STACKING

Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance. Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance. This ensemble technique works by applying input of combined multiple weak learners' predictions and Meta learners so that a better output prediction model can be achieved.

In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction.

So, the stacking ensemble method includes **original (training) data, primary level models, primary level prediction, secondary level model,**

and final prediction. The basic architecture of stacking can be represented as shown below the image.

- Original data: This data is divided into n-folds and is also considered test data or training data.
- Base models: These models are also referred to as level-0 models. These models use training data and provide compiled predictions (level 0) as an output.
- Level-0 Predictions: Each base model is triggered on some training data and provides different predictions, which are known as level-0 predictions.
- Meta Model: The architecture of the stacking model consists of one meta-model, which helps to best combine the predictions of the base models. The meta-model is also known as the level-1 model.
- Level-1 Prediction: The meta-model learns how to best combine the predictions of the base models and is trained on different predictions made by individual base models, i.e., data not used to train the base models are fed to the meta-model, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

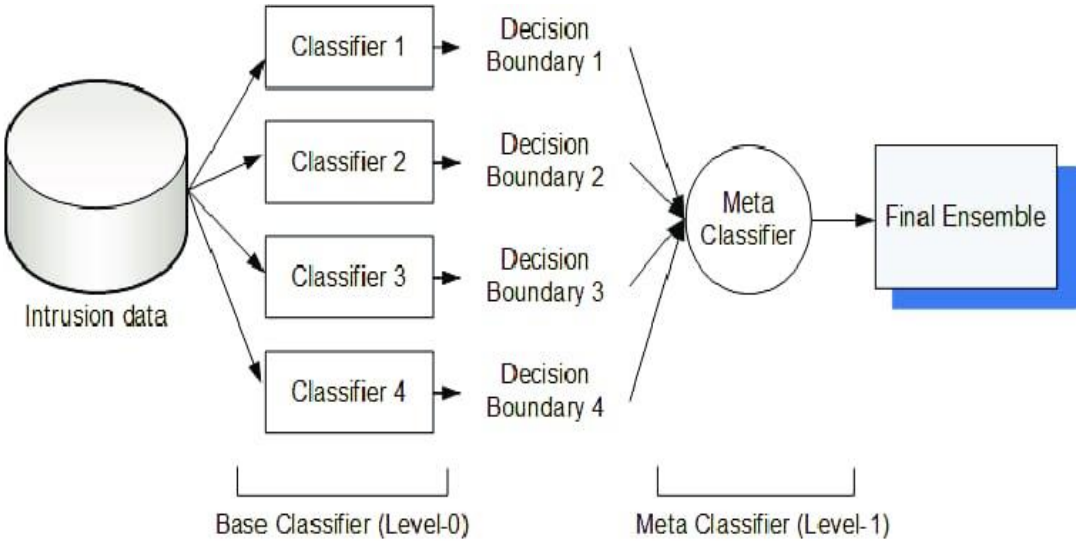

Figure 4.1.5: *Working of Stacking Model*

CHAPTER 5

IMPLEMENTATION

5.1 SAMPLE CODE

```
import pandas as pd
from pandas import DataFrame
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import six
import sys
sys.modules['sklearn.externals.six'] = six
import seaborn as sns
from sklearn.metrics import log_loss,roc_auc_score,precision_score,f1
_score,recall_score,roc_curve,auc
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,ac
curacy_score,fbeta_score,matthews_corrcoef
from sklearn import metrics
from sklearn.metrics import log_loss
from imblearn.metrics import geometric_mean_score
import warnings
import re
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
```

```
warnings.filterwarnings('ignore')
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,VotingClassifie
r
from sklearn.calibration import CalibratedClassifierCV
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import SGDClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
import xgboost as xgb
from scipy import stats
from sklearn import model_selection
#from sklearn.ensemble import StackingClassifier
from mlxtend.classifier import StackingClassifier
import os
filename = "Multi-class dataset.csv"
```

```
headers = ["age","sex","cp","trestbps","chol","fbs","restecg","thalch","
exang","oldpeak","slope","ca","thal","num"]
df = pd.read_csv(filename, names = headers)
# To see the data set looks like, we'll use the head() method.
df.head()
df['sex'] = df['sex'].map({'Male': 1, 'Female': 0})
df['cp'] = df['cp'].map({'typical angina': 1, 'atypical angina': 2, 'non-
anginal': 3, 'asymptomatic':4})
df['fbs'] = df['fbs'].map({True: 1, False: 0})
df['restecg'] = df['restecg'].map({'normal': 0, 'lv hypertrophy': 1, 'st-
t abnormality': 2})
df['exang'] = df['exang'].map({True: 1, False: 0})
df['slope'] = df['slope'].map({'downsloping': 3, 'flat': 2, 'upsloping': 1})
df['thal'] = df['thal'].map({'normal': 3, 'fixed defect': 6, 'reversable defec
t': 7})
# To see what the data, we'll use the head() method.
df.head()
from matplotlib import pyplot as plt
plt.rcParams["figure.figsize"] = [10, 5]
plt.rcParams["figure.autolayout"] = True
fig, axes = plt.subplots(5, 3)
df.hist('age', ax=axes[0][0])
df.hist('sex', ax=axes[0][1])
df.hist('cp', ax=axes[0][2])
df.hist('trestbps', ax=axes[1][0])
df.hist('chol', ax=axes[1][1])
```

```
df.hist('fbs', ax=axes[1][2])
df.hist('restecg', ax=axes[2][0])
df.hist('thalch', ax=axes[2][1])
df.hist('exang', ax=axes[2][2])
df.hist('oldpeak', ax=axes[3][0])
df.hist('slope', ax=axes[3][1])
df.hist('ca', ax=axes[3][2])
df.hist('thal', ax=axes[4][0])
df.hist('num', ax=axes[4][1])
plt.show()
df.shape
data = df.values
# split into inputs and outputs
X = df.iloc[:,0:12].values
y =df.iloc[:,13].values
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
ndom_state=5)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
def models(X_train, X_test, y_train, y_test):
  knn = KNeighborsClassifier(9)
  knn.fit(X_train,y_train)
  y_pred_knn = knn.predict(X_test)
  acc_knn = accuracy_score(y_test, y_pred_knn)
  accuracy=[]
  accuracy.append(acc_knn)
```

```
  print('Accuracy Score of KNN:', acc_knn * 100)
  rf_ent = RandomForestClassifier(criterion='entropy',n_estimators=10
0)
  rf_ent.fit(X_train, y_train)
  y_pred_rfe = rf_ent.predict(X_test)
  acc_rfe = accuracy_score(y_test, y_pred_rfe)
  accuracy.append(acc_rfe)
  print('Accuracy Score of RF:', acc_rfe * 100)
  xgb = XGBClassifier()
  xgb.fit(X_train,y_train)
  y_pred_xgb = xgb.predict(X_test)
  acc_xgb = accuracy_score(y_test, y_pred_xgb)
  accuracy.append(acc_xgb)
  print('Accuracy Score of XGB:', acc_xgb * 100)
  svc = SVC(kernel='linear',gamma='auto',probability=True)
  svc.fit(X_train,y_train)
  y_pred_svc = svc.predict(X_test)
  acc_svc = accuracy_score(y_test, y_pred_svc)
  accuracy.append(acc_svc)
  print('Accuracy Score of SVC:', acc_svc * 100)
  stack = StackingClassifier(classifiers=[knn,rf_ent,xgb],meta_classifie
r=svc)
  stack.fit(X_train, y_train)
  y_pred_stack = stack.predict(X_test)
  acc_stack = accuracy_score(y_test, y_pred_stack)
  accuracy.append(acc_stack)
```

```
  print('Accuracy Score of Stacked model:', acc_stack * 100)
models(X_train, X_test, y_train, y_test)
# import modules
import matplotlib.pyplot as mp
import pandas as pd
import seaborn as sb
# plotting correlation heatmap
dataplot=sb.heatmap(df.corr(), cmap="YlGnBu", annot=True)
# displaying heatmap
mp.show()
df[["age","sex","cp","trestbps","chol","fbs","restecg","thalch","exang",
"oldpeak","slope","ca","thal","num"]].corr()['num'][:]
def plot_boxplot(df,ft):
  df.boxplot(column=[ft])
  plt.grid(False)
  plt.show()
plot_boxplot(df,"chol")
plot_boxplot(df,"trestbps")
plot_boxplot(df,"thalch")
plot_boxplot(df,"oldpeak")
def outliers(df,ft):
  Q1=df[ft].quantile(0.25)
  Q3=df[ft].quantile(0.75)
  IQR=Q3-Q1
  lower_bound=Q1-1.5*IQR
  upper_bound=Q3+1.5*IQR
```

```
  ls=df.index[(df[ft]<lower_bound)|(df[ft]>upper_bound)]
  return ls
index_list=[]
for feature in ['chol','trestbps','thalch','oldpeak']:
  index_list.extend(outliers(df,feature))
index_list
def remove(df,ls):
  ls=sorted(set(ls))
  df=df.drop(ls)
  return df
df_cleaned=remove(df,index_list)
df_cleaned.shape
data = df_cleaned.values
# split into inputs and outputs
X = df_cleaned.iloc[:,0:12].values
y =df_cleaned.iloc[:,13].values
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
ndom_state=5)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
models(X_train, X_test, y_train, y_test)
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest, SelectPercentile
from sklearn.metrics import accuracy_score
df_cleaned.drop(labels = ['age', 'sex'], axis = 1, inplace = True)
df_cleaned.isnull().sum()
X=df_cleaned[['cp','trestbps','chol','fbs','restecg','thalch','exang','oldpeak
','slope','ca','thal']]
y=df_cleaned['num']
#Chi-2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, r
andom_state = 0)
f_score = chi2(X_train, y_train)
f_score
p_values = pd.Series(f_score[1], index = X_train.columns)
p_values.sort_values(ascending = True, inplace = True)
p_values
p_values.plot.bar()
X_train_2=X_train[['thalch', 'ca','oldpeak','chol','thal','exang','cp']]
X_test_2=X_test[['thalch', 'ca','oldpeak','chol','thal','exang','cp']]

models(X_train_2, X_test_2, y_train, y_test)
X_train_4 = X_train[['thalch', 'ca','oldpeak','chol','thal','exang','cp','trest
bps']]
```

```
X_test_4 = X_test[['thalch', 'ca','oldpeak','chol','thal','exang','cp','trestbps']]
models(X_train_4, X_test_4, y_train, y_test)
X_train_3 = X_train[['thalch', 'ca','oldpeak','chol','thal','exang','cp','trestbps','slope']]
X_test_3 = X_test[['thalch', 'ca','oldpeak','chol','thal','exang','cp','trestbps','slope']]
models(X_train_3, X_test_3, y_train, y_test)
X_train_3.shape,X_test_3.shape,y_train.shape,y_test.shape
def models_final(X_train, X_test, y_train, y_test):
  knn = KNeighborsClassifier(9)
  knn.fit(X_train,y_train)
  y_pred_knn = knn.predict(X_test)
  acc_knn = accuracy_score(y_test, y_pred_knn)
  accuracy=[]
  accuracy.append(acc_knn)
  print('Accuracy Score of KNN:', acc_knn * 100)
  print('Precision Score of KNN:', precision_score(y_test, y_pred_knn,average='weighted')*100)
  print('Recall Score of KNN:', recall_score(y_test, y_pred_knn,average='weighted')*100)
  print('F1-
Score of KNN:', f1_score(y_test, y_pred_knn,average='weighted')*100)
  rf_ent = RandomForestClassifier(criterion='entropy',n_estimators=100)
```

```
  rf_ent.fit(X_train, y_train)
  y_pred_rfe = rf_ent.predict(X_test)
  acc_rfe = accuracy_score(y_test, y_pred_rfe)
  accuracy.append(acc_rfe)
  print('\nAccuracy Score of RF:', acc_rfe * 100)
  print('Precision Score of RF:', precision_score(y_test, y_pred_rfe,aver
age='weighted')*100)
  print('Recall Score of RF:', recall_score(y_test, y_pred_rfe,average='
weighted')*100)
  print('F1-
Score of RF:', f1_score(y_test, y_pred_rfe,average='weighted')*100)
  xgb = XGBClassifier()
  xgb.fit(X_train,y_train)
  y_pred_xgb = xgb.predict(X_test)
  acc_xgb = accuracy_score(y_test, y_pred_xgb)
  accuracy.append(acc_xgb)
  print('\nAccuracy Score of XGB:', acc_xgb * 100)
  print('Precision Score of XGB:', precision_score(y_test, y_pred_xgb,a
verage='weighted')*100)
  print('Recall Score of XGB:', recall_score(y_test, y_pred_xgb,average
='weighted')*100)
  print('F1-
Score of XGB:', f1_score(y_test, y_pred_xgb,average='weighted')*100
)
  svc = SVC(kernel='linear',gamma='auto',probability=True)
  svc.fit(X_train,y_train)
```

```
  y_pred_svc = svc.predict(X_test)
  acc_svc = accuracy_score(y_test, y_pred_svc)
  accuracy.append(acc_svc)
  print('\nAccuracy Score of SVC:', acc_svc * 100)
  print('Precision Score of SVC:', precision_score(y_test, y_pred_svc,av
erage='weighted')*100)
  print('Recall Score of SVC:', recall_score(y_test, y_pred_svc,average
='weighted')*100)
  print('F1-
Score of SVC:', f1_score(y_test, y_pred_svc,average='weighted')*100)
  stack = StackingClassifier(classifiers=[knn,rf_ent,xgb],meta_classifie
r=svc)
  stack.fit(X_train, y_train)
  y_pred_stack = stack.predict(X_test)
  acc_stack = accuracy_score(y_test, y_pred_stack)
  accuracy.append(acc_stack)
  print('\nAccuracy Score of Stacking model:', acc_stack * 100)
  print('Precision Score of Stacking model:', precision_score(y_test, y_p
red_stack,average='weighted')*100)
  print('Recall Score of Stacking model:', recall_score(y_test, y_pred_st
ack,average='weighted')*100)
  print('F1-
Score of Stacking model:', f1_score(y_test, y_pred_stack,average='wei
ghted')*100)
models_final(X_train_3, X_test_3, y_train, y_test)
import matplotlib.pyplot as plt
```

```
models=['KNN','RF','XGB','SVC','Stack']
accuracy=[67.14,69.64,71.42,71.42,73.21]
plt.bar(models,accuracy,width=0.1)
plt.title("Various Models with its accuracy")
plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.rcParams["figure.figsize"] = [25, 15]
plt.show()
```

CHAPTER 6

RESULT

The outcomes of employed five above-mentioned techniques (SVC, RF, XGB, and KNN separately and in combination using ensemble learning) utilizing 10-fold cross-validation are displayed below.

Table 6.1: *Various models with accuracy*

Applied Methods (in Percentage)	With all features	Removing outliers	With selected number of features (thalch, ca, oldpeak, chol, thal, exang, cp)	With selected number of features (thalch, ca, oldpeak, chol, thal, exang, cp, trestbps)	With selected number of features (thalch, ca, oldpeak, chol, thal, exang, cp, trestbps, slope)
Stacking	63.33	64.28	65.28	66.07	73.21
RF	60.00	60.71	66.07	67.00	69.64
SVC	61.66	64.25	65.08	65.64	71.42
XGB	61.66	62.50	64.77	64.28	71.42
KNN	56.66	50.00	57.85	59.14	67.14

The table below shows the assessment of accuracy, precision, recall, and f-score for heart disease with individual and ensemble classifiers. It can be noticed that the Stacking ensemble classifier obtained 73.21% higher accuracy, with 66.36% higher precision, 73.21% higher recall, and 69.32% higher f-score as compared to Voting, SVC, RF, XGB, and KNN classifiers as observed in the results. The comparison of the results demonstrates that the Stacking ensemble classifier was superior to state of art classification methods. The proposed system could assist in the development of an intelligent medical diagnosis system for heart disease prediction.

Table 6.2: *Various models with accuracy, precision, recall, f-score*

Applied Methods (in Percentage)	Accuracy	Precision	Recall	F-score
Stacking	73.21%	66.36%	73.21%	69.32%
RF	69.64%	62.80%	73.21%	66.96%
SVC	71.42%	63.43%	71.42%	66.55%
XGB	71.42%	62.94%	71.42%	66.60%
KNN	67.14%	51.99%	57.14%	51.16%

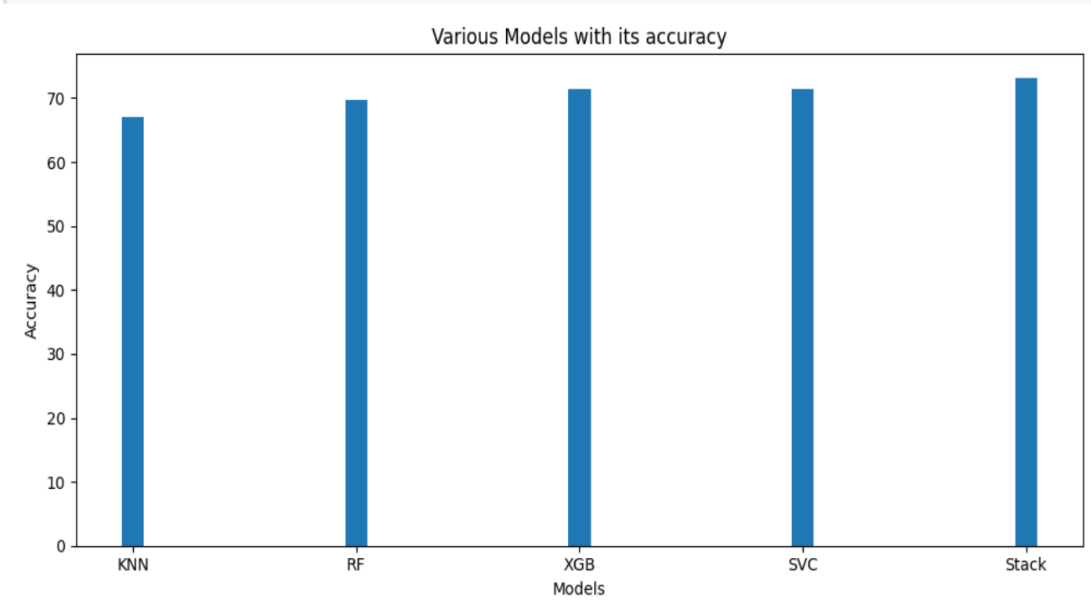

Figure 6.3: *Various models with accuracy*

CHAPTER 7
CONCLUSION AND FUTURE WORK

CONCLUSION:

In this study, we have proposed a novel web-based medical diagnosis system for heart disease prediction that gives a more precise diagnosis. The authors have realized the multi-class classification based on several machine learning algorithms such as RF, DT, XGBoost, SVC, KNN, and Stacking ensemble classifiers. The Stacking classifier achieved 73.21% higher accuracy, with 66.36% higher precision, 73.21% higher recall, 69.36% higher f-score compared to SVC, RF, XGB, and KNN classifiers as noted in the results. The proposed system accomplished the highest accuracy, as well as having a multi-classification characteristic that was found deficient in the existing systems. Hospitals/doctors may utilize the proposed medical diagnosis system for diagnostic recommendations.

FUTURE SCOPE:

Further, as an extension to this work, a more real-time and bigger dataset is required to obtain a better training model. Also, an emphasis on refining the preprocessing further will give veracious outcomes.

REFERENCES

[1] Madhumita Pal and Smita Parija, “Prediction of Heart Diseases using Random Forest”, 2021, DOI: 10.1088/1742-6596/1817/1/012009

[2] Froyston Mendonca, Riyazuddin Manihar, Ashishkumar Pal, Sapna U Prabu, “Intelligent Cardiovascular Disease Risk Estimation Prediction System”, 2020, DOI: 10.1109/ICAC347590.2019.9036738.

[3] Atharv Nikam, Sanket Bhandari, Aditya Mhaske, Shamla Mantri, “Cardiovascular Disease Prediction Using Machine Learning Models”, 2021, DOI: 10.1109/PuneCon50868.2020.9362367

[4] Areel Ashfaq, Azhar Imran, Inam Ullah, Abdulkareem Alzahrani, Khattab M. Ali Alheeti, “Multi-model Ensemble Based Approach for Heart Disease Diagnosis”,2022, DOI: 10.1109/RAEECS56511.2022.9954490

[5] Nazim Nisar Itoo, Vijay Kumar Garg, “Heart Disease Prediction using a Stacked Ensemble of Supervised Machine Learning Classifiers”, 2022, DOI: 10.1109/MECON53876.2022.9751883

[6] K. Rehman, F. Fatima, I. Waheed, “Prevalence of exposure of heavy metals and their impact on health consequences,”Journal of Cellular Biochemistry, vol. 119, pp. 157–184, 2018.

[7] L. Ding, Y. Liang, E.C.K. Tan, “Smoking, heavy drinking, physical inactivity, and obesity among middle-aged and older adults in China: cross-sectional findings from the baseline survey of CHARLS 2011– 2012,” BMC Public Health, vol. 20, 2020.

[8] MJ. Pencina, AM. Navar, D. Wojdyla, RJ. Sanchez I. Khan, J. Elassal, “Quantifying importance of major risk factors for coronary heart disease,”

Stroke, vol. 139, no. 13, pp. 1603–1611, 2019.

[9] E.E. Tripoliti, G.P. Papadopoulos, S.K. Georgia, K.N. Katerina,,“Heart Failure: Diagnosis, Severity Estimation and Prediction of Adverse Events Through Machine Learning Techniques, Computational and Structural Biotechnology Journal,” vol. 15, pp. 26-47, 2017.

[10] H. Sharma, M. Rizvi, “Prediction of Heart Disease using Machine Learning Algorithms: A Survey,” International Journal on Recent and Innovation Trends in Computing and Comm., vol. 5, no. 8, 2017.

[11] A. Saboor, M. Usman, S. Ali, A. Samad, “A Method for Improving Prediction of Human Heart Disease Using Machine Learning Algorithms,” Hindawi, vol. 2022, 2021.

[12] S. Kaur, “Medical Diagnostic Systems Using Artificial Intelligence (AI) Algorithms: Principles and Perspectives,” IEEE Access, vol. 8, pp. 228049-228069, 2020.

[13] S.F. Waris and S. Koteeswaran, “Heart disease early prediction using a novel machine learning method called improved k-means neighbor classifier in python,” Materials Today Proceedings, 2021.

[14] A. Newaz, N. Ahmed, F. S. Haq, “Survival prediction of heart failure patients using machine learning techniques,” Informatics in Medicine Unlocked, vol. 26, 2021.

[15] I. D. Mienye, Y. Sun, and Z.Wang, “An improved ensemble learning approach for the prediction of heart disease risk,” Informatics in Medicine Unlocked, vol. 20, 2020.