HEART DISEASE PREDICTION USING AN ENSEMBLE OF MACHINE LEARNING ALGORITHMS

A PROJECT REPORT

Submitted by

Nithish B	**(185002068)**
Raajalakshumi K	**(185002077)**

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)

JUNE 2022

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project titled **“HEART DISEASE PREDICTION USING AN ENSEMBLE OF MACHINE LEARNING ALGORITHMS”** is the bonafide work of “**Nithish B** (185002068), **Raajalakshumi K** (185002077), who carried out the project 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.

SIGNATURE	**SIGNATURE**
Dr. C. ARAVINDAN	Dr. S. VIDHUSHA,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Head of the Department	Assistant Professor
Department of Information Technology	Department of Information Technology
SSN College of Engineering	SSN College of Engineering
Kalavakkam - 603 110.	Kalavakkam - 603 110.

Submitted for project viva-voce examination held at **Sri Sivasubramaniya Nadar College of Engineering** on 09-06-2022.

EXTERNAL EXAMINER **INTERNAL EXAMINER**

ABSTRACT

Heart disease is one of the leading causes of death, and thus early detection of heart disease is crucial. There are many computerized diagnoses available that lack accuracy from a medical perspective. Consequently, an approach that could improve prediction accuracy is necessary. A large number of data combined with suitable machine learning algorithms can be used to identify a pattern and improve accuracy.

From the literature, it is understood that the accuracy of heart disease prediction through various models is around 85-88%. This project uses an ensemble of machine learning algorithms to build a stronger model to produce improvised results. To achieve better performance than individual models, ensemble algorithms such as K-Nearest Neighbours (KNN), Decision Trees (DT), Naive Bayes (NB), and Logistic Regression (LR) are used as base learners, while Random Forest (RF) is used as the strong learner. With this approach, the prediction of heart disease is possible at an earlier stage.

KEYWORDS: Heart Disease Prediction; Ensemble Learning; Machine Learning; Bagging; Full Stack Ensemble; Hard Voting

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 the founder **PADMA BHUSHAN Dr. SHIV NADAR**, Chairman, SSN Institutions. We also express our appreciation to our Principal, **Dr. V.E. ANNAMALAI**, for all the help he has rendered during this course of study.

Our sincere thanks to **Dr. C. ARAVINDAN**, Professor and Head of the Department of Information Technology, for his words of advice and encouragement. We would like to thank and show our deep sense of gratitude to our guide **Dr. S. VIDHUSHA,** Assistant Professor, Department of Information Technology, for her valuable advice and suggestions as well as her continued guidance, patience and support that helped us to shape and refine our work.

Also, we would like to thank our Project Coordinators **Dr. T. SREE SHARMILA**, Associate Professor, Department of Information Technology and **Dr. N. BHALAJI**, Associate Professor, Department of Information Technology for their valuable suggestions throughout the project. We are grateful to our Project Panel Members for providing us with helpful pointers throughout the course of the project.

We would like to extend our sincere thanks to all the staff of our department who have contributed directly and indirectly during our project work. Finally, we would like to thank our parents and friends for their patience, cooperation and moral support throughout our life.

NITHISH B **RAAJALAKSHUMI K**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

WHO	World Health Organization
ML	Machine Learning
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
MLP	Multilayer Perceptron
ANN	Artificial Neural Network
CART	Classification And Regression Tree
HVC	Hard Voting Classifier
RF	Random Forest
DT	Decision Tree
LDA	Linear Discriminant Analysis
SGD	Stochastic Gradient Descent
OpenCV	Open-Source Computer Vision Library
BSD	Berkeley Source Distribution
MATLAB	Matrix Laboratory
STL	Standard Template Libraries
API	Application Programming Interfaces
BRFSS	Behavioral Risk Factor Surveillance System
CDC	Centers for Disease Control and Prevention
ECG	Electrocardiogram
CHD	Coronary Heart Disease
MI	Myocardial Infarction
BMI	Body Mass Index
TP	True Positive
TN	True Negative

FP	False Positive
FN	False Negative
MCC	Mathews Correlation Coefficient
AUC-ROC	Area under the receiver operating characteristic curve
LR	Logistic Regression

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Heart disease is one of the biggest causes of mortality among the population of the world representing 32% of all deaths. According to World Health Organization (WHO), an estimated amount of 19.2 million people died due to heart diseases in 2019. It is also estimated that about 90% of heart diseases may be preventable at the early stages. Heart disease prediction can play a vital role as it can detect heart diseases so that they can be prevented in the early stages and it might avoid complications as the diagnosis of heart disease is cumbersome that must be performed precisely and efficiently, and moreover, many health institutes have started to find a way to predict heart diseases and many research has been undergone in order to find the influential factor causing heart disease and perfectly predicting the overall risk so that it would be helpful to prevent the disease at the early stages.

There are different types of heart diseases such as stroke, heart failure, hypertensive heart disease, rheumatic heart disease, and so on. There are certain common factors among all these diseases but there is a certain underlying mechanism that may vary for different types. It is estimated that dietary risk factor is associated with 53% of heart disease death, while tobacco accounts for 9%, diabetes 6%, lack of exercise for 6%, and obesity 5%.

In the healthcare sector, there is enormous information about the patients which is impossible for humans to process and predict the outcome perfectly. Machine Learning proves to be effective in this kind of situation since it provides a technique to recognize patterns from the information about the patients and use certain algorithms to predict the outcome of the patients. Even though there are different types of heart disease, there are certain attributes that are similar for all types of heart diseases, and with the help of this, we can predict the outcome of the patient.

1.2 NEED FOR STUDY

Manual diagnosis of heart disease depending upon risk factors is cumbersome, it takes a lot of time for evaluations to be carried out. Many computerized diagnoses are available, but they are inaccurate from a medical standpoint. As a result, a method that can increase forecast accuracy is required. Computational efficiency is reduced. Consequently, an approach that could improve prediction accuracy is necessary. To find a pattern and enhance accuracy, a vast amount of data combined with appropriate machine learning algorithms can be used to identify a pattern and improve accuracy.

1.3 PROBLEM STATEMENT

The project's major goal is to improve performance and accuracy in predicting cardiac disease using ensemble learning approaches. This project employs a collection of machine learning methods to provide a more robust model for improvised results. An ensemble reduces the spread or dispersion of the predictions and model performance. The Ensemble of the same and different classifiers includes Logistic Regression, Decision tree, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naïve Bayes, Extra tree classifier, Adaboost, Random Forest, Xgboost and MLP are employed to obtain greater performance than individual models. This project examines four ensemble methods: Bagging, Hard Voting, Random Forest, and Full stack ensemble.

1.4 OBJECTIVES OF THE STUDY

I. To predict the future likelihood of heart disease at an early stage using ensemble machine learning algorithms with a higher accuracy.

II. To reduce variance and bias so that the model becomes very flexible and tunes itself to the data points of the training set.

III. To build a powerful model that can tune to any large dataset and deliver higher accuracy and performance.

1.5 REPORT ORGANIZATION

The report is organized as follows:

Chapter 1 provides a basic overview of the project, as well as an explanation of the issue statement, and **Chapter 2** is devoted to a literature review relating to our research. The suggested system's overall methodology is discussed in **Chapter 3**. The tools and technologies employed in this research, as well as the dataset, pre-processing, and data balancing procedures used, are discussed in **Chapter 4**. The implementation and source code for the Machine Learning models utilized are found in **Chapter 5**. In **Chapter 6**, multiple model outputs are displayed, and a comparison of all the models surveyed is made, with the optimal model inferred via reasoning. Finally, **Chapter 7** offers the conclusion as well as the work that needs to be done in the future.

CHAPTER 2

LITERATURE SURVEY

[1] Purushottam, et al, (2016) proposed a paper "Efficient Heart Disease Prediction System" using hill climbing and decision tree algorithms. They used the Cleveland dataset and preprocessing of data is performed before using classification algorithms. The Knowledge Extraction is done based on Evolutionary Learning (KEEL), an open-source data mining tool that fills the missing values in the data set. A decision tree follows top-down order. For each actual node selected by the hill-climbing algorithm, a node is selected by a test at each level. The parameters and their values used are confidence. Its minimum confidence value is 0.25. The accuracy of the system is about 86.7%.

[2] Sonam Nikhar et al, (2020) proposed the paper "Prediction of Heart Disease Using Machine Learning Algorithms" their research gives point to point explanation of Naïve Bayes and decision tree classifiers that are used especially in the prediction of Heart Disease. Some analysis has been led to think about the execution of prescient data mining strategy on the same dataset, and the result decided that Decision Tree has the highest accuracy than the Bayesian classifier.

[3] Abhay Kishore et al, (2018) proposed "Heart Attack Prediction Using Deep Learning" which is a heart attack prediction system by uses Deep learning techniques to predict the probable aspects of heart-related infections of the patient Recurrent Neural System is used. This model uses deep learning and data mining to give the best precise model and least blunders. This paper acts as a strong reference model for another type of heart attack prediction model.

[4] Deepika & Seema, (2017) focuses on techniques that can predict chronic disease by mining the data contained in historical health records using Naïve Bayes, Decision tree, Support Vector Machine (SVM) and Artificial Neural Network (ANN). A comparative study is performed on classifiers to measure better performance at an accurate rate. From this experiment, SVM gives the highest accuracy rate, whereas for diabetes Naïve Bayes gives the highest

accuracy.

[5] Chala Beyene, (2018) recommended Prediction and Analysis of the occurrence of Heart Disease Using Data Mining Techniques. The main objective is to predict the occurrence of heart disease for early automatic diagnosis of the disease within result in a short time. The proposed methodology is also critical in a healthcare organization with experts that have no more knowledge and skill. It uses different medical attributes such as blood sugar and heart rate, age, and sex are some of the attributes are included to identify if the person has heart disease or not. Analyses of the data set are computed using WEKA software.

[6] Lakshmana Rao et al, (2019) proposed "Machine Learning Techniques for Heart Disease Prediction" in which the contributing elements for heart disease are more. So, it is difficult to distinguish heart disease. To find the seriousness of heart disease among people different neural systems and data mining techniques are used.

[7] Sai & Reddy, (2017) proposed heart disease prediction using the ANN algorithm in data mining. Due to the increasing expenses of heart disease diagnosis disease, there was a need to develop a new system that can predict heart disease. The prediction model is used to predict the condition of the patient after evaluation on the basis of various parameters like heartbeat rate, blood pressure, cholesterol, etc. The accuracy of the system is proved in java.

[8] Sultana, Haider, & Uddin, (2017) proposed an analysis of cardiovascular disease. This paper proposed data mining techniques to predict the disease. It is intended to provide a survey of Current techniques to extract information from datasets, it will be useful for healthcare practitioners. The performance can be obtained based on the time taken to build the decision tree for the system. The primary objective is to predict the disease with a smaller number of attributes.

CHAPTER 3

SYSTEM DESIGN

In this project, the prediction of heart disease with the help of 18 attributes such as diabetes, BMI, and some personal lifestyle of the person like smoking, drinking, physical activity, sleep time, etc has been performed. Various machine learning algorithms will be used such as Logistic regression, Decision tree, Naive Bayes, KNN and SVM apply them to train and test the data and get the accuracy. The ensemble of the same and different classifiers includes Logistic Regression, Decision tree, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naïve Bayes, Extra tree classifier, Adaboost, RandomForest, Xgboost and MLP were used. Accuracy, f1 score, precision, recall and support were noted. Totally four ensemble methods are analyzed in this project -Bagging, HardVoting, RandomForest and Full Stack Ensemble.

The reason to use the ensemble method is that it’s a meta-approach in machine learning that seeks better predictive performance for the large datasets by combining the predictions from multiple models thus, achieve a better and optimized model for the prediction of heart diseases.

3.1 BASE LEARNERS

A weak classifier is a binary classification model that outperforms random guessing by a small margin. Weak Learners are any models that outperform a naive prediction method by a small margin. In this respect, it's a great tool for thinking about classifier capabilities and ensemble composition.

3.1.1 NAÏVE BAYES ALGORITHM

The Naive Bayes method is a supervised learning method for resolving classification issues that are based on the Bayes theorem. It is one of the simple and most effective Classification algorithms which helps in building fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. It's primarily used for text categorization with a large-scale training dataset.

It's a probabilistic classifier, which means it makes predictions based on an object's likelihood. Spam filtration, Sentimental analysis, and article classification are all common applications of the Naive Bayes Algorithm. It's a classification method based on the Bayes Theorem and the assumption of predictor independence. A Naive Bayes classifier, in simple terms, posits that the existence of one feature in a class is unrelated to the presence of any other feature.

The Naive Bayes model is simple to construct and is especially good for huge data sets. Naive Bayes is renowned to outperform even the most advanced classification systems due to its simplicity.

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

3.1.2 DECISION TREE ALGORITHM

The Decision Tree is a supervised learning technique that may be applied to both classification and regression issues, where It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions however, it is most commonly employed to solve classification problems. Internal nodes represent dataset attributes, branches represent decision rules, and each leaf node provides the conclusion in this tree-structured classifier.

The Decision Node and the Leaf Node are the two nodes of a Decision Tree. Leaf nodes are the output of those decisions and do not contain any more branches, whereas Decision nodes are used to make any decision and have several branches. The decisions or tests are made based on the characteristics of the given dataset. It's a graphical depiction for obtaining all feasible solutions to a problem/decision depending on certain parameters. It's named a Decision Tree because, like a tree, it starts with the root node and grows into a tree-like structure with additional branches. We utilize the CART algorithm, which stands for Classification and Regression Tree algorithm, to form a tree. A Decision Tree simply asks a question and divides the tree into subtrees based on the answer (Yes/No). It can be used to solve a classification as well as a regression problem. The purpose of this technique is to develop a model that predicts the value of a target variable, and the decision tree solves the problem by using the tree representation, where the leaf node corresponds to a class label and characteristics are represented on the internal node of the tree.

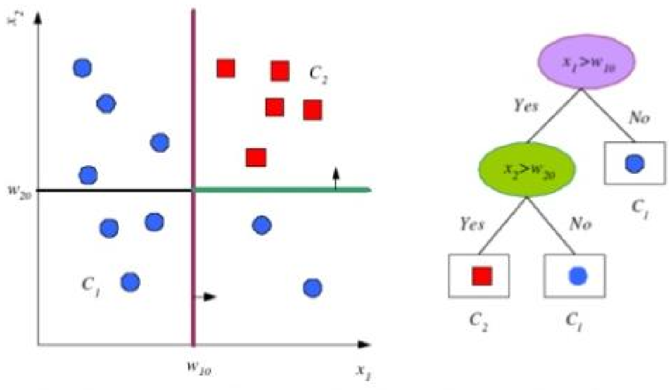


Figure 3.1: Decision Trees - Nearest Neighbors

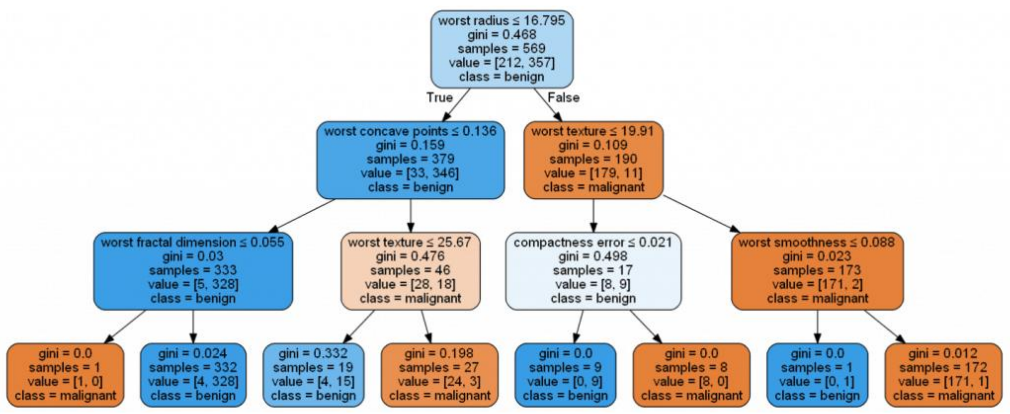


Figure 3.2: Decision Tree example

3.1.3 LOGISTIC REGRESSION ALGORITHM

The Supervised Learning technique includes one of the most common Machine Learning algorithms: logistic regression. It's a method for predicting a categorical dependent variable from a set of independent variables. A categorical dependent variable's output is predicted using logistic regression. As a result, the result must be a discrete or categorical value. It delivers probabilistic values that are somewhere between 0 and 1.

Except for how they are employed, Logistic Regression is very similar to Linear Regression. Regression problems are solved using linear regression, while classification problems are solved using logistic regression. The logistic function's curve reflects the probability of things like whether a person is obese or not based on his/her weight, and so on. Because it can generate probabilities and classify new data using both continuous and discrete datasets, logistic regression is a key machine learning approach. Logistic regression is a statistical analysis technique that uses independent features to try to predict precise probability outcomes. On high-dimensional datasets, this may cause the model to be over-fit on the training set, overstating the accuracy of predictions on the training set, and so preventing the model from accurately predicting results on the test set. This is most common when the model is trained on a little amount of training data with a large number of features. Regularization strategies should be considered on high-dimensional datasets to minimize overfitting (but this makes the model complex). The model may be under-fit on the training data if the regularization variables are too high.

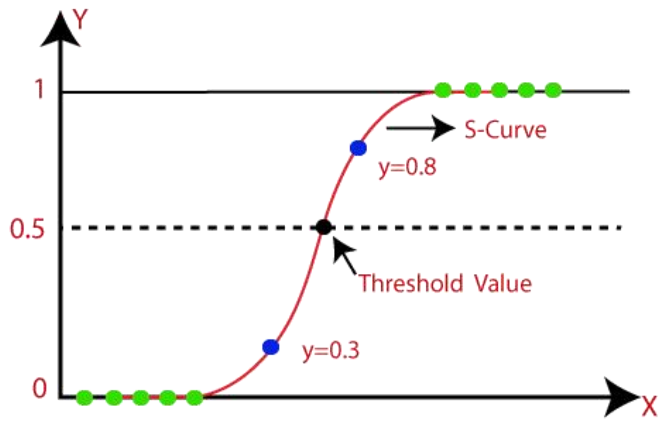


Figure 3.3: Logistic Regression (Sigmoid Function)

3.1.4 KNN ALGORITHM

The K Nearest Neighbor algorithm is a Supervised Learning Algorithm and can be used for classification and regression. It is also a versatile algorithm for imputing missing values and resampling datasets.

KNN is preferred when some of the features are continuous. Identification of the nearest neighbor is used to determine the class of an unknown sample. Because of its high convergence speed and simplicity, KNN classification is preferred over other algorithms. KNN classification has two stages find the k number of instances in the dataset that is closest to instance S and These k number of instances then vote to determine the class of instance S.

KNN Accuracy depends on the distance metric and K value. Various ways of measuring the distance between two instances are cosine and Euclidian distance. Based on the majority vote, KNN computes the K nearest neighbors for the new unknown sample.

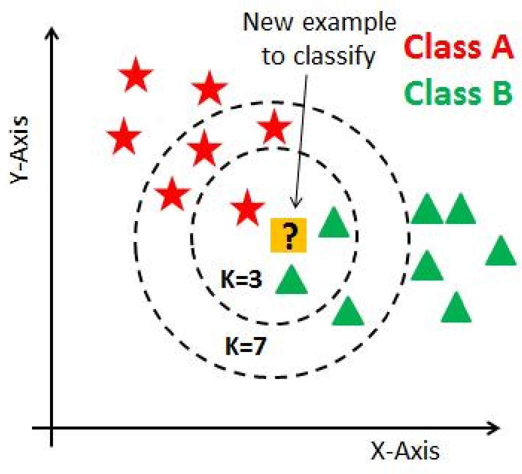


Figure 3.4: KNN Classification

3.1.5 SUPPORT VECTOR MACHINES (SVM)

Support Vector Machine is used 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.

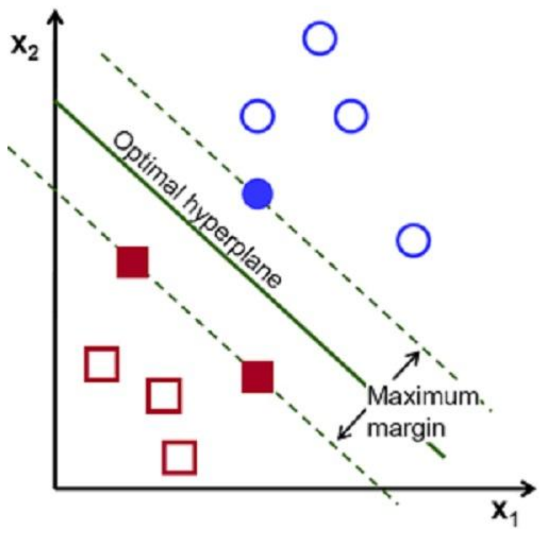


Figure 3.5: SVM Classifier

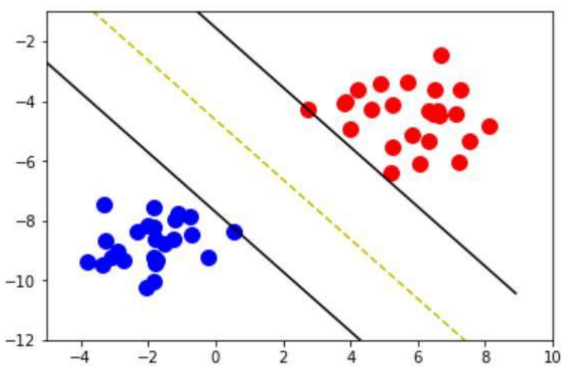


Figure 3.6: SVM Classifier

3.2 ENSEMBLE METHODS

An ensemble model is a machine learning technique that combines numerous base models to build a single best-predicting model. An ensemble technique is a machine learning meta-strategy that tries to improve predictive performance by mixing results from numerous models to produce a better and optimized model for heart disease prediction. An ensemble reduces the spread or dispersion of the predictions and model performance.

In this heart prediction model, ensemble techniques are used in order to produce an optimized model with higher accuracy. Totally four ensemble methods are analyzed in this project- **Bagging, Hard Voting, Random Forest, and Full Stack Ensemble.**

3.2.1 BAGGING

Bootstrap aggregation, or bagging for short, is an ensemble learning method that seeks a diverse group of ensemble members by varying the training data. This typically involves using a single machine learning algorithm, almost always an unpruned decision tree, and training each model on a different sample of the same training dataset. The predictions made by the ensemble members are then combined using simple statistics, such as voting or averaging. In this project KNN and Decision tree were used as base classifiers with n-estimates of 800 and a random state of 8, the Decision tree performed better.

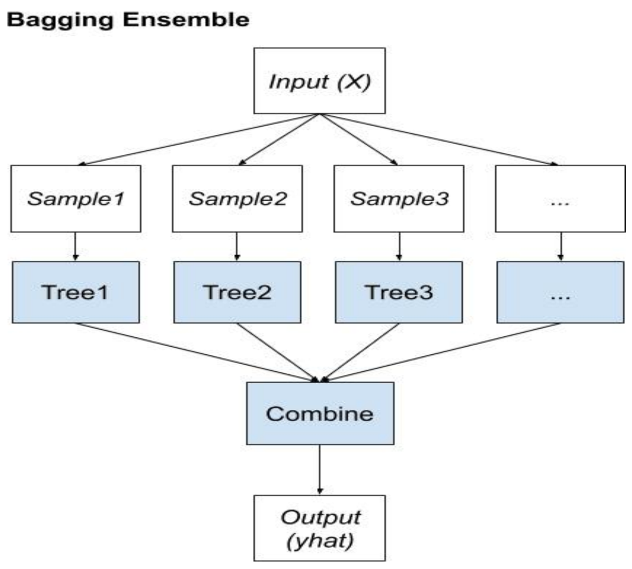


Figure 3.7: Bagging Classifier Architecture

3.2.2 VOTING CLASSIFIER

A Voting Classifier is a machine learning model that learns from an ensemble of models and predicts an output (class) based on the highest probability of the output being the chosen class. It simply adds up the results of each classifier supplied into the Voting Classifier and predicts the output class based on the most votes. Rather than constructing separate specialized models and determining their performance, we create a single model that trains on various models and predicts output based on the cumulative majority of votes for each output class.

3.2.2.1 HARD VOTING

Hard voting ensemble involves summing the votes for crisp class labels from other models and predicting the class with the most votes. Hard voting is appropriate when the models used in the voting ensemble predict crisp class labels. It classifies input data based on the mode of all the predictions made by different classifiers. The majority voting is considered differently when weights associated with the different classifiers are equal or otherwise.

In the HVC, the class with the most votes is the classifier's final prediction. For example, if three classification models projected the output class as (X, X, Y), it can be seen that the majority of classifiers anticipated X as the output, hence X will be the HVC's final output.

The voting classifier in this project uses Logistic Regression, Random Forest, and Decision Tree with the random state set to 1. Hard Voting involves summing the votes for crisp class labels from the three models and predicting the class with the most votes. Then In-Sample performance and Out-Sample performance were noted.

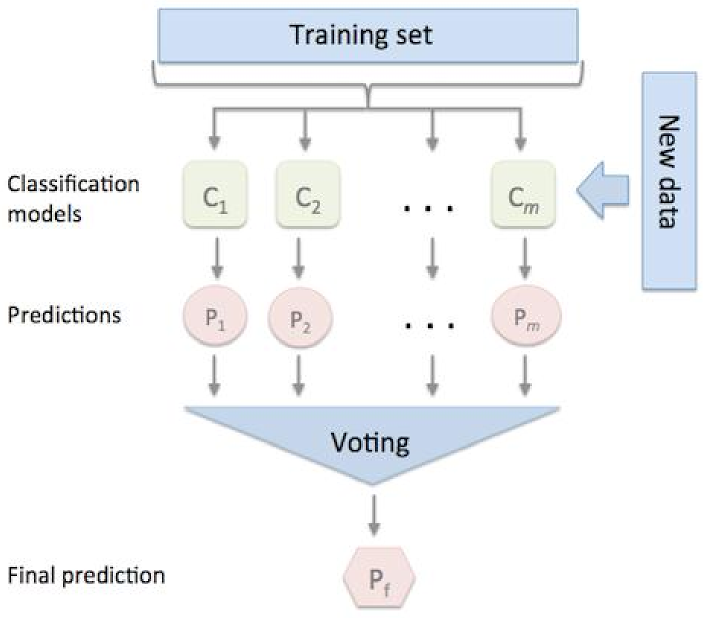


Figure 3.8: Hard Voting Classifier Architecture

3.2.3 FULL STACK ENSEMBLE

Stacking, also called Super Learning or Stacked Regression, is a class of algorithms that involves training a second-level **“meta learner”** to find the optimal combination of the base learners. Unlike bagging and boosting, the goal of stacking is to ensemble strong, diverse sets of learners together. There are some ensemble methods that are broadly labeled as stacking; however, the Super Learner ensemble is distinguished by the use of cross-validation to form what is called the “level-one” data, or the data that the meta-learning or “combiner” algorithm is trained on. Stacked Generalization, or stacking for short, is an ensemble method that seeks a diverse group of members by varying the model types fit on the training data and using a model to combine predictions. Stacking has its own nomenclature where ensemble members are referred to as level-0 models and the model that is used to combine the predictions is referred to as a level-1 model. The two-level hierarchy of models is the most common approach, although more layers of models can be used.

The level-1 model training data is prepared via k-fold cross-validation of the base models, and the training dataset is made up of out-of-fold predictions (actual numbers for regression and class labels for classification). The level-0 models mixture of algorithms or the same algorithm (most often they are diverse). A simple model, such as Linear Regression for regression problems and Logistic Regression for classification problems, is frequently used as the level-1 meta-model.

This is a different strategy in which different models are layered and cross-validated one after the other. All of these predictions are given to the meta classifier (generalizer), which in this case is the MLP classifier. The following models have stacked in this order: RF (criterion-index and n estimator of 100), MLP, RF (criterion-Gini and n estimator of 100), KNN(k=9), Extra tree(n estimators of 500, 100), Xgboost(n estimators of 1000, 100, 500, 2000), SGD(maximum iteration of 1000, tolerance of 1e-4), Adaboost, DT, LDA, Gradient Boosting Classifier(n estimators of 100, maximum features of sqrt) and Extra tree (n estimators of 1000). It was set up in such a way that it provided the best level of accuracy. The highest accuracy algorithm that is Full Stack Ensemble algorithm has been applied to the entire dataset.

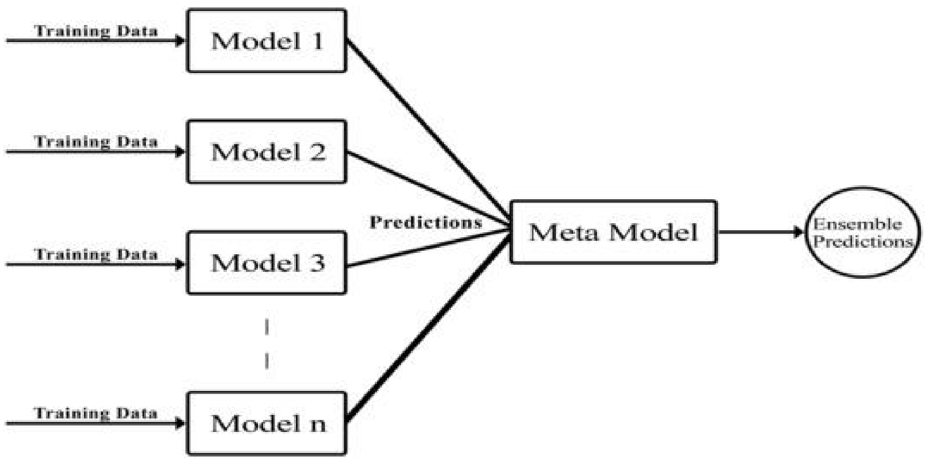


Figure 3.9: Full Stack Ensemble Classifier Architecture

3.2.3.1 XGBOOST ALGORITHM

Gradient Boosted decision trees are implemented using XG-boost. It is a type of software library that was created with the goal of improving model speed and performance. Decision trees are constructed sequentially in this approach. In XG-boost, weights are very significant. All of the independent variables are given weights, which are subsequently fed into the decision tree, which predicts outcomes. The weight of variables that the tree predicted incorrectly is increased, and the variables are then fed into the second decision tree. Individual classifiers/predictors are then combined to form a more robust and precise model. It has the ability to perform regression, classification, ranking, and user-defined prediction.

3.2.3.2 ADABOOST ALGORITHM

Adaboost was the first truly successful boosting algorithm created specifically for binary classification. Adaptive Boosting, often known as Adaboost, is a prominent boosting strategy that merges numerous "weak classifiers" into a single "strong classifier."Adaboost employs a method of gradually learning to boost. As a result, in Adaboost vs Random Forest examples, high-quality data is required. It is also extremely sensitive to outliers and noise in data, necessitating their removal before using the data.

3.2.3.3 EXTRA TREE CLASSIFIER

Extremely Randomized Trees Classifier is a form of ensemble learning technique that outputs a classification result by aggregating the outcomes of numerous de-correlated decision trees collected in a "forest." It is conceptually identical to a Random Forest Classifier, with the exception of how the decision trees in the forest are constructed. The Extra Trees Forest's Decision Trees are all made from the original training sample. Then, at each test node, each tree is given a random sample of k features from the feature set, from which it must choose the best feature to split the data according to certain mathematical criteria (typically the Gini Index). Multiple de-correlated decision trees are created from this random sample of features.

3.2.3.4 LINEAR DISCRIMINANT ANALYSIS (LDA)

A linear decision boundary classifier is created by fitting class conditional densities to data and applying Bayes' rule. This classifier is appealing because it provides closed-form solutions that are simple to compute, are intrinsically multiclass, have been demonstrated to operate in practice, and have no hyperparameters to modify. Each class is given a Gaussian density by the model, which assumes that all classes have the same covariance matrix. Using the transform approach, the fitted model can be utilized to minimize the dimensionality of the input by projecting it to the most discriminative directions.

3.2.3.5 GRADIENT BOOSTING CLASSIFIER

Gradient Boosting Classifier constructs an additive model in a stage-by-stage manner, allowing for the optimization of any differentiable loss function. In each level, n classes_ regression trees are fitted to the loss function's negative gradient, such as binary or multiclass log loss. A specific example of binary classification is when just one regression tree is induced. Both binary and multi-class classification are supported by Gradient Boosting Classifier.

3.2.3.6 SGD CLASSIFIER

This estimator uses stochastic gradient descent (SGD) learning to create regularized linear models: the gradient of the loss is estimated one sample at a time, and the model is updated along the way with a decreasing strength schedule (aka learning rate). The partial fit method in SGD enables minibatch (online/out-of-core) learning. The data should have a zero mean and unit variance for the best results when using the default learning rate schedule. The features are represented as dense or sparse arrays of floating-point values in this approach. The loss parameter can be used to alter the model it fits; by default, it fits a linear support vector machine (SVM).

3.2.3.7 MULTILAYER PERCEPTRON CLASSIFIER

A feedforward artificial neural network model, the multilayer perceptron (MLP), maps input data sets to a collection of acceptable outputs. An MLP is made up of numerous layers, each of which is fully connected to the one before it. Except for the nodes of the input layer, the nodes of the layers are neurons with nonlinear activation functions. One or more nonlinear hidden layers may exist between the input and output layers.

Different models are stacked one after another and cross-validated. All those predictions are sent to the meta classifier(generalizer) here MLP classifier is used as a generalizer.

3.3 STRONG LEARNER

Strong learners are models with a high level of precision. Weak and strong learners are computational learning theory techniques that serve as the foundation for the creation of boosting ensemble approaches. Predictive modeling challenges require strong classifiers. The modeling project's purpose is to create a powerful classifier that produces largely right predictions with high confidence.

3.3.1 RANDOM FOREST

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

It has the ability to be utilized for both classification and regression. It's also the most adaptable and user-friendly algorithm. Trees make up a forest. A forest is thought to be stronger the more trees it has. Random Forests generate Decision Trees from randomly chosen data samples, obtain predictions from each tree, then vote on the best option. It also serves as a strong indicator of the value of the feature.

Random Forests can be used for a range of tasks, including recommendation engines, image categorization, and feature selection. It can be used to identify fraudulent activities, classify loyal loan applicants, and anticipate diseases. The Boruta algorithm, which picks relevant features in a dataset, is built on it. Random Forest is a well-known machine learning algorithm that uses the supervised learning method. In machine learning, it can be utilized for both classification and regression issues. It is based on ensemble learning, which integrates numerous classifiers to solve a complex problem and increase the model's performance.

Because the random forest combines numerous trees to forecast the dataset's class, some decision trees may correctly predict the output while others may not. However, when all of the trees are combined, the proper result is predicted. As a result, two assumptions for a better Random Forest classifier are as follows:

- They feature variable in the dataset should contain some actual values so that the classifier can predict accurate results rather than a guess.
- The predictions from each tree should have very low correlations.

Random Forest can handle both classification and regression problems. It can handle huge datasets with a lot of dimensionalities. It improves the model's accuracy and eliminates the problem of overfitting. Despite the fact that Random Forest may be used for both classification and regression problems, it is not better suited to regression.

RandomForestClassifier and RandomizedSearchCV were imported from sklearn. The dataset was trained and tested using RF with 1000 n estimators and a random state of 42, which resulted in good accuracy.

3.4 WORKFLOW – PROPOSED METHOD

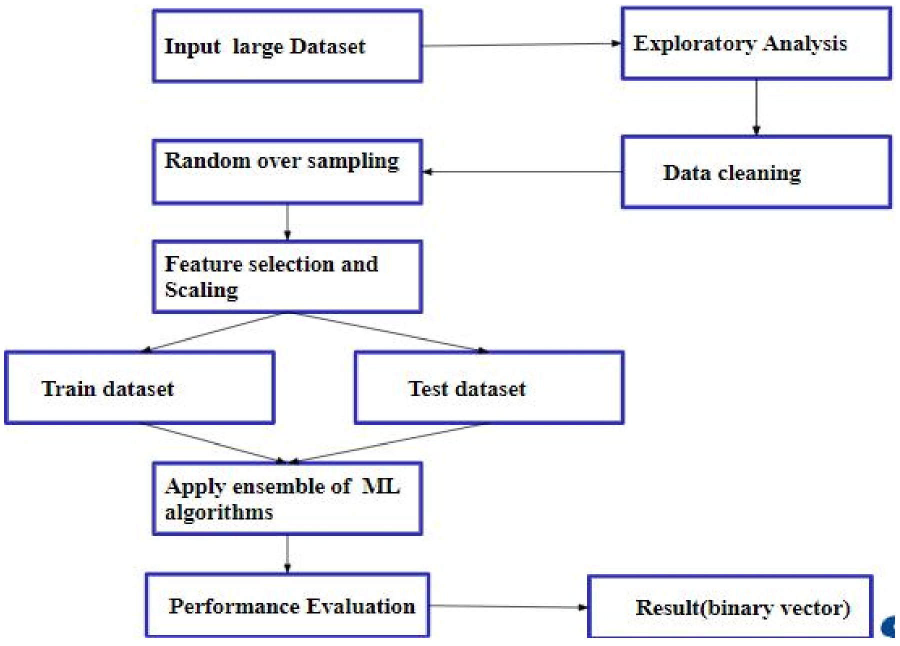


Figure 3.10: Workflow - Proposed Method

3.4.1 PROPOSED METHOD

Dataset was downloaded from Kaggle. Checked for its stability and uniformity. Outliers and duplicate data were been removed. Missing values were filled. Checked for Null values. The ranges of the datasets used are vastly diverse, and they are measured in different units of measurement. In order to normalize the data, Standard Scaler is used, which eliminates the mean from the data and scales it to the unit variance. The selected dataset was so unbalanced and biased, it had only 8% of people having heart disease so, so in order to compensate the Random Over Sampling technique was used to balance the Dataset. Using Chi-square, significant features associated with heart disease prediction were identified. To understand the relationship between the features and the target variable, the data was presented using the bar, count, dist, pie charts, and graphs. The data is divided into two categories: train data and test data, with each receiving 80% and 20% of the total. Individual machine learning models such as KNN, Logistic regression, Decision tree, Naive Bayes and SVM are applied to train and test the data and their corresponded accuracy, and f1 score are noted. The ensemble of the same and different classifiers includes Logistic Regression, Decision tree, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naïve Bayes, Extra tree classifier, Adaboost, Random Forest, Xgboost and MLP were used. Accuracy, f1 score, precision, recall and support were noted. Totally four ensemble methods are analyzed in this project -Bagging, Hard Voting, Random Forest and Full Stack Ensemble.

Bagging: A single machine learning algorithm is used. In this project KNN and Decision tree were used as base classifiers with n estimates of 800 and a random state of 8, the Decision tree performed better.

Hard Voting: The voting classifier in this project uses Logistic Regression, Random Forest, and Decision Tree with the random state set to 1. Hard Voting involves summing the votes for crisp class labels from the three models and predicting the class with the most votes. Then In-Sample performance and Out-Sample performance were noted.

Random Forest: RandomForestClassifier and RandomizedSearchCV were imported from sklearn. The dataset was trained and tested using RF with 1000 n estimators and a random state of 42, which resulted in good accuracy.

Full Stack Ensemble: this is a different approach where different models are stacked one after another and cross-validated. All those predictions are sent to the meta classifier(generalizer) here MLP classifier is used as a generalizer. RF (criterion-index and n estimator of 100), MLP, RF(criterion-Gini and n estimator of 100), KNN(k=9), Extra tree(n estimators of 500, 100), Xgboost(n estimators of 1000, 100, 500, 2000), SGD(maximum iteration of 1000, tolerance of 1e-4), Adaboost, DT, LDA, Gradient Boosting Classifier(n estimators of 100, maximum features of sqrt) and Extra tree (n estimators of 1000) models were stacked in this order. Arranged in such a way that it gave the highest accuracy.
The highest accuracy algorithm that is Full Stack Ensemble algorithm has been applied to the entire dataset. Then a GUI interface will be created and get the user's input and predict whether they have heart disease or not, depending on the values.

3.5 LANGUAGES AND PACKAGES USED

3.5.1 PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

3.5.2 SKLEARN

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It’s built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib. The functionality that scikit-learn provides includes:

- Regression, including Linear and Logistic Regression
- Classification, including K-Nearest Neighbors
- Clustering, including K-Means and K-Means++
- Model selection
- Preprocessing, including Min-Max Normalization.

3.5.3 OPENCV

OpenCV is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

3.5.4 MATPLOTLIB

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open-source alternative to MATLAB. Developers can also use matplotlib’s APIs (Application Programming Interfaces) to embed plots in graphical user interface applications.

3.5.5 NUMPY

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

CHAPTER 4

DATASET

4.1 DATASET USED

Based on the literature survey, the datasets used by various authors are customized datasets with a smaller number of instances. Dataset has been downloaded from Kaggle. which contains large instances of 3 lakhs and 18 attributes.

The data originated with the CDC and is a key component of the Behavioral Risk Factor Surveillance System (BRFSS), which conducts annual telephone surveys to collect information on Americans' health. As the Centers for Disease Control and Prevention (CDC) puts it: "BRFSS began collecting data in 15 states in 1984 and has since expanded to include all 50 states, the District of Columbia, and three US territories. Each year, the BRFSS conducts over 400,000 adult interviews, making it the world's biggest continually conducted health survey system ". Data from 2020 is included in the most recent dataset (as of February 15, 2022). The specialty of the dataset is that it contains attributes that are related to the occurrence of heart disease after covid. These attributes are selected in a way that can predict heart disease without even looking into the ECG reports just by knowing our current health conditions and day-to-day activities.

Another small dataset was downloaded from Kaggle with the instances of 300 and 13 attributes. This dataset was collected by Tanvir Ahmad, Assia Munir, Sajjad Haider Bhatti, Muhammad Aftab, and Muhammad Ali Raza (Government College University, Faisalabad, Pakistan) from BMC Medical Informatics on 2020. By training and testing ensemble methods, the accuracies of large and small datasets is been compared. A Small dataset is used just for comparision but the main focus dataset is the large dataset with 3 Lakhs instances.

4.2 DEMOGRAPHICS

4.2.1 DATASET-1

1. HeartDisease: Respondents that have ever reported having coronary heart disease (CHD) or not.
2. BMI: Body Mass Index (BMI).
3. Smoking: Have you smoked at least 100 cigarettes in your entire life? (The answer Yes or No)
4. AlcoholDrinking: Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week
5. Stroke: (Ever told) (you had) a stroke?
6. PhysicalHealth: Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good? (0-30 days).
7. MentalHealth: Thinking about your mental health, for how many days during the past 30 days was your mental health not good? (0-30 days).
8. DiffWalking: Do you have serious difficulty walking or climbing stairs?
9. Sex: Are you male or female?
10. AgeCategory: Fourteen-level age category.
11. Race: Imputed race/ethnicity value.
12. Diabetic: (Ever told) (you had) diabetes?
13. PhysicalActivity: Adults who reported doing physical activity or exercise during the past 30 days other than their regular job.
14. GenHealth: Would you say that in general, your health is...
15. SleepTime: On average, how many hours of sleep do you get in a 24-hour period?
16. Asthma: (Ever told) (you had) asthma?

17. KidneyDisease: Not including kidney stones, bladder infection, or incontinence, were you ever told you had kidney disease?
18. SkinCancer: (Ever told) (you had) skin cancer?

4.2.2 DATASET-2

1. Age: age of the person
2. Anaemia: Decrease of red blood cells or hemoglobin (boolean)
3. Creatinine phosphokinase: Level of the CPK enzyme in the blood (mcg/L)
4. Diabetes: If the patient has diabetes (boolean)
5. Ejection fraction: Percentage of blood leaving the heart at each contraction (percentage)
6. High blood pressure: If the patient has hypertension (boolean)
7. Platelets: Platelets in the blood (kiloplatelets/mL)
8. Serum creatinine: Level of serum creatinine in the blood (mg/dL)
9. Serum sodium: Level of serum sodium in the blood (mEq/L)
10. Sex: Woman or man (binary)
11. Smoking: If the patient smokes or not (boolean)
12. Time: Follow-up period (days)
13. Death Event: If the patient deceased during the follow-up period (boolean)

4.3 DATA PREPROCESSING

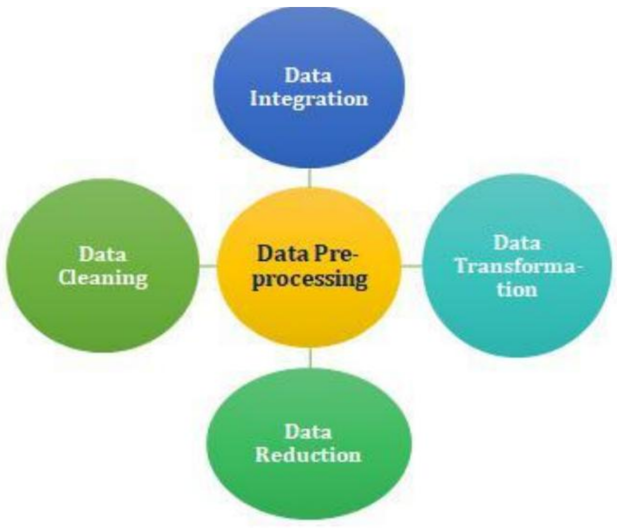


Figure 4.1: Four categories of Data pre-processing

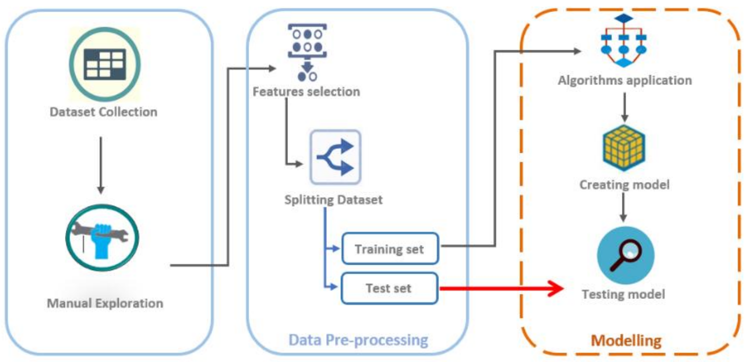
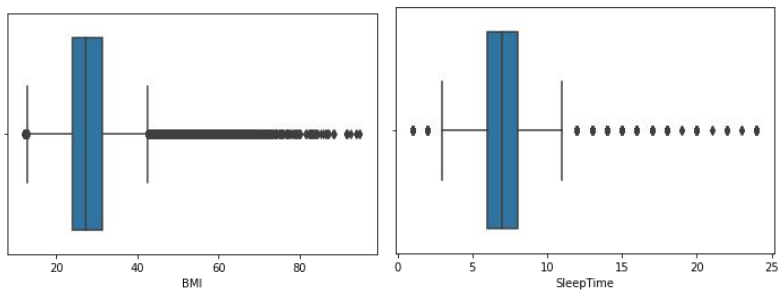


Figure 4.2: Steps involved in Data pre-processing

4.3.1 DATA CLEANING

Duplicate data results in irrelevant observations that actually don't fit the specific problem. Checked for its stability and uniformity. Outliers and duplicate data were been removed. Missing values were filled. Checked for Null values.



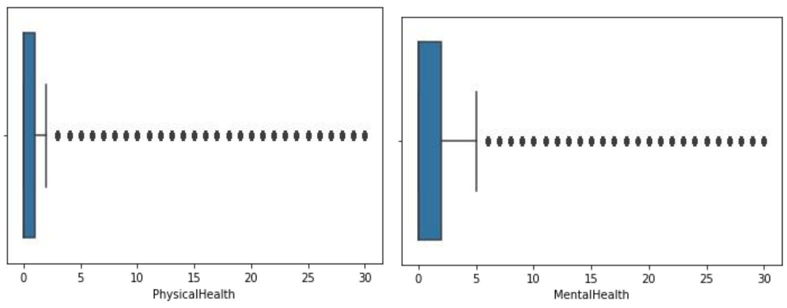


Figure 4.3: Before Removing Outliers

It's seen that there are lots of outliers present in Body Mass Index, Sleep Time, Physical Health and Mental Health features.

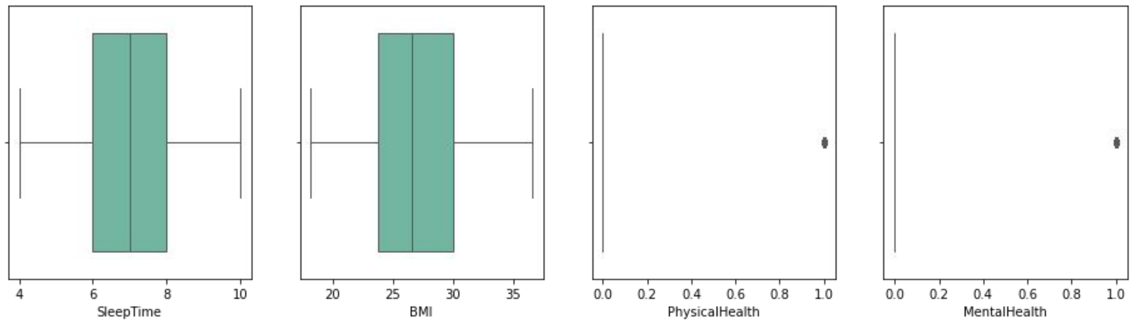


Figure 4.4: After Removing Outliers

Outliers have been removed to a great extent.

4.3.2 DATA SCALING

Feature scaling is essential for machine learning algorithms that calculate distances between data. The scaling is used for making data points generalized so that the distance between them will be lower. Features have been scaled, based on a standard normal distribution.

Data Normalization:

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

4.3.3 DATA ANALYSIS AND PRE - PROCESSING

The column names were changed and separated between categorical columns and numerical columns. Encoded categorical data- changed the column values to binary vectors.

4.3.4 DATA SPLITTING

To train the Machine Learning algorithm, we mention the target column in the data set, then we divide the data set into two small datasets.

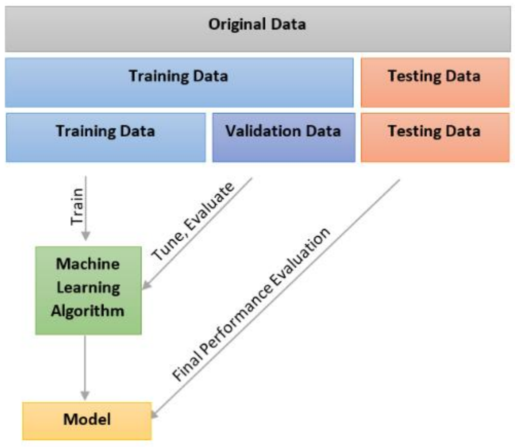


Figure 4.12: Data Splitting Architecture

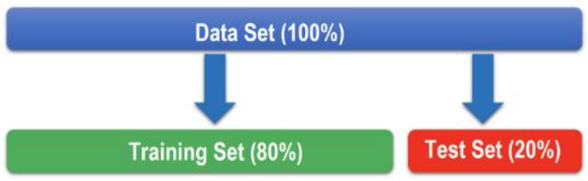


Figure 4.13: Data Splitting of Dataset-1

4.3.5 DATA VISUALIZATION

Data visualization helps to tell stories by curating data into a form easier to understand, highlighting the trends and outliers. A good visualization tells a story, removing the noise from data and highlighting useful information. In the project, the dataset was viewed using count, distance, pie, bar charts, and graphs

.

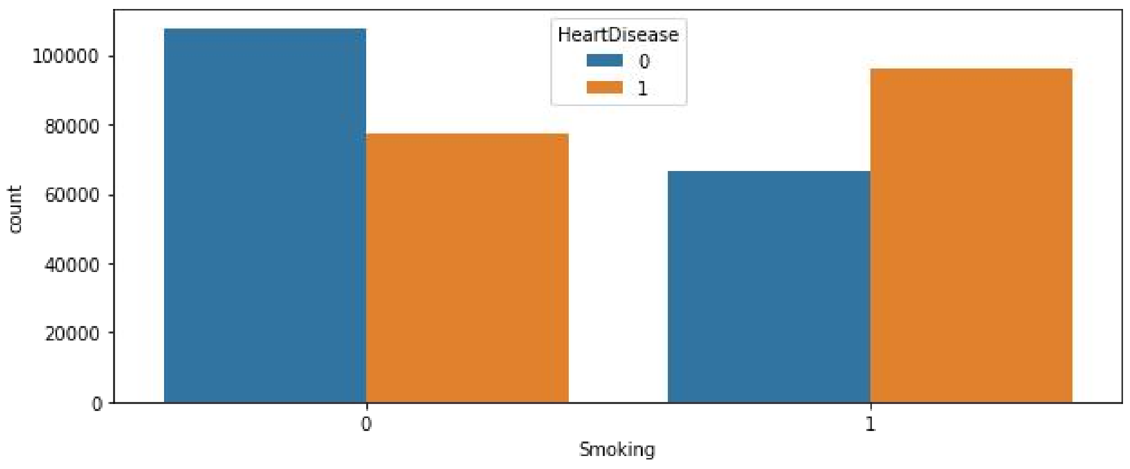


Figure 4.5: Data Visualization - Smoking vs Count

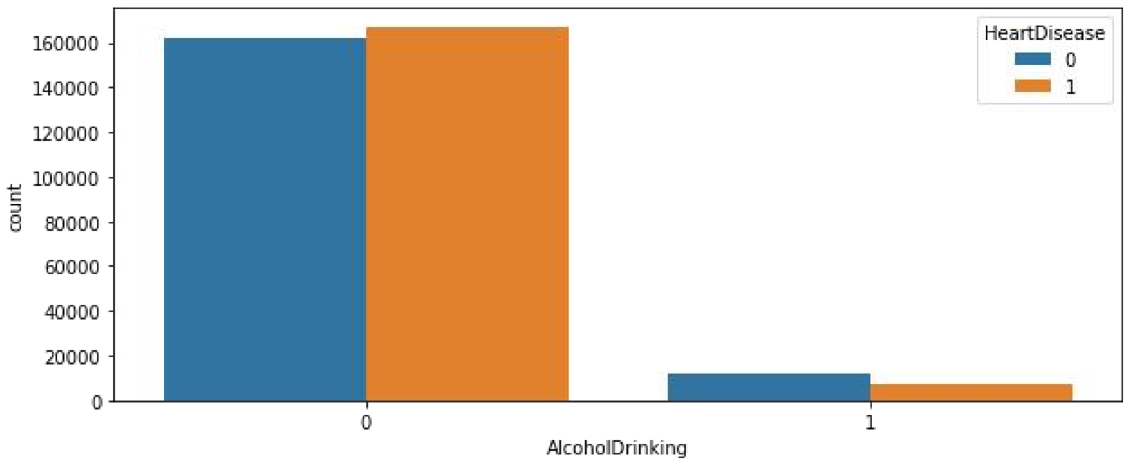


Figure 4.6: Data Visualization - AlcoholDrinking vs Count

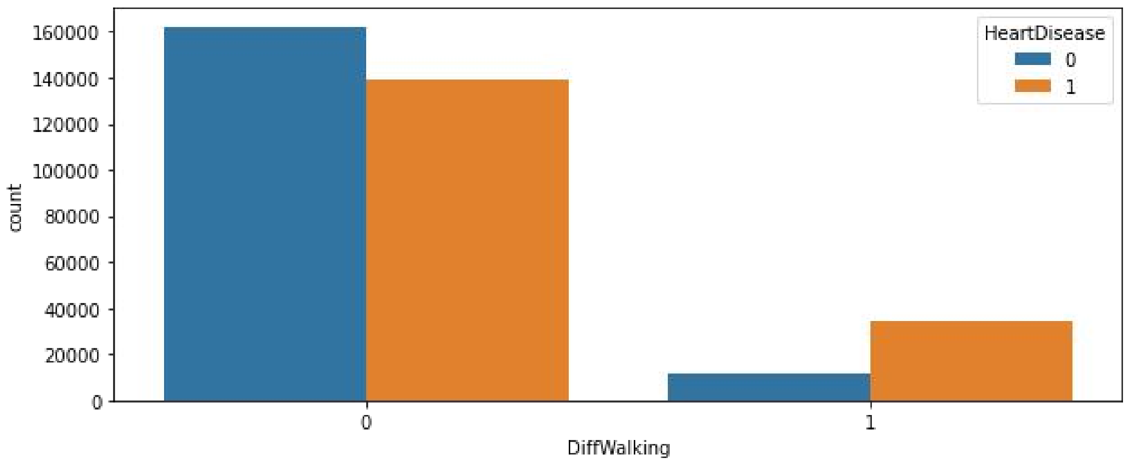


Figure 4.7: Data Visualization - DiffWalking vs Count

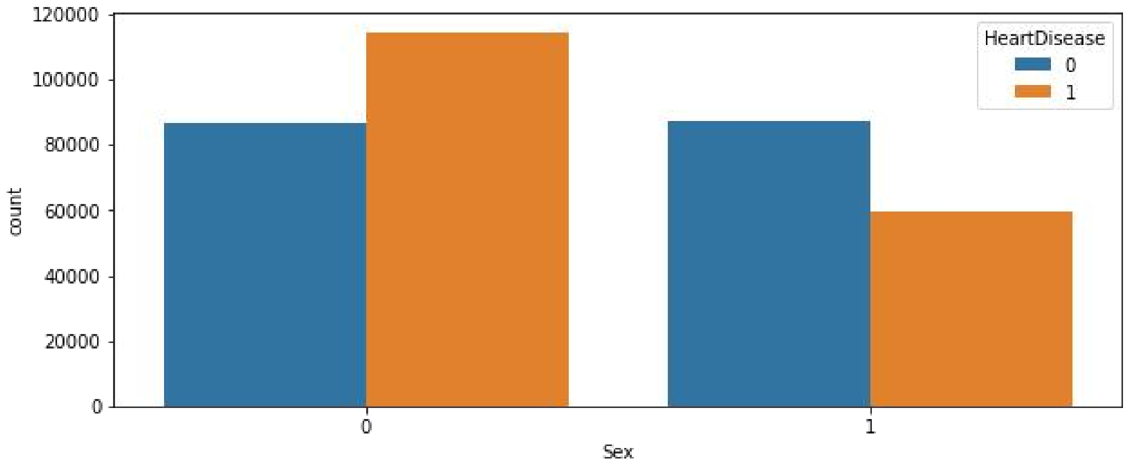


Figure 4.8: Data Visualization - Sex vs Count

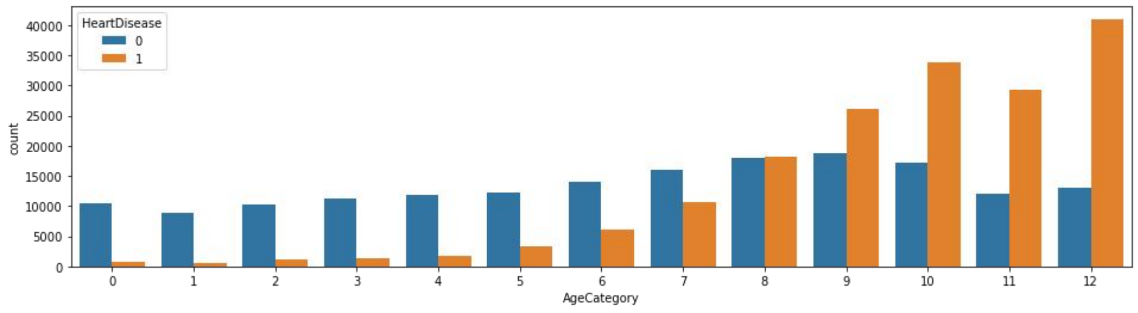


Figure 4.9: Data Visualization - AgeCategory vs Count

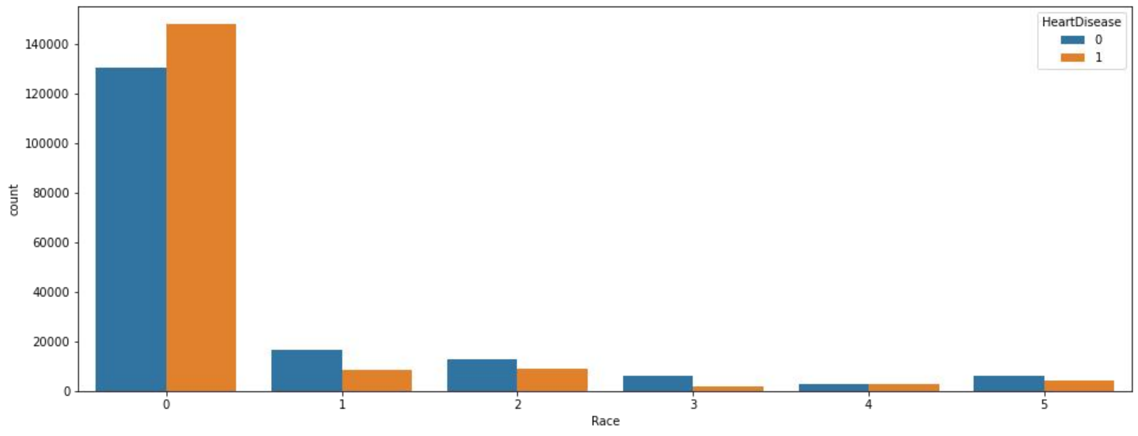


Figure 4.10: Data Visualization - Race vs Count

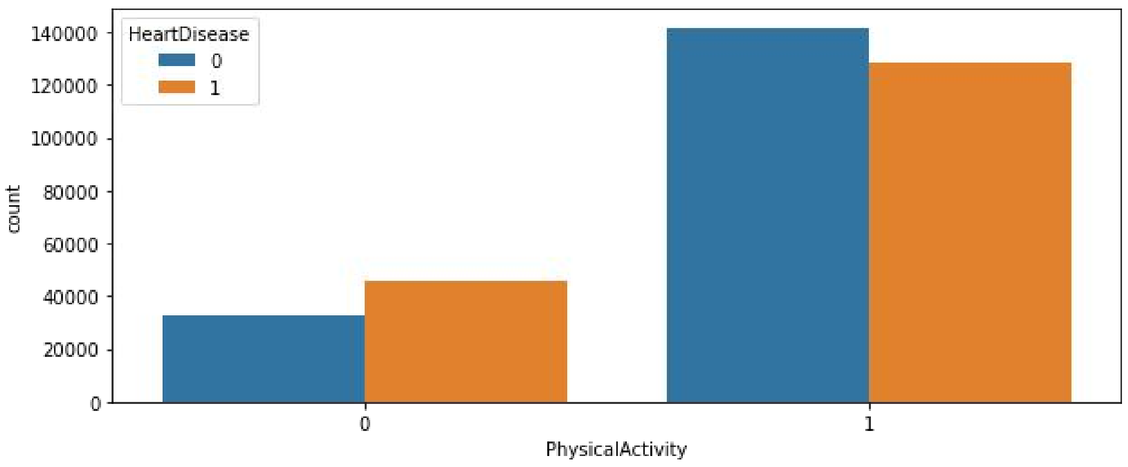


Figure 4.11: Data Visualization - PhysicalActivity vs Count

It has been discovered that smoking increases your risk of heart disease by roughly 50%. Drinking alcohol does not increase your risk of heart disease, but the number of individuals who don't drink and get heart disease is higher than the number of people who don't drink and get heart disease, according to the data. It's connected because persons with heart disease are more likely to have trouble walking or climbing stairs. Males are more likely than females to develop cardiac disease. As the person becomes older, the chances of developing heart disease increase. There are no significant findings from races; whites have a high risk, whereas Asians with heart disease have a big decline. Logically, participating in sports lowers your risk of heart disease, whereas not participating in sports increases your risk of heart disease.

4.3.6 SAMPLING TECHNIQUE

Random sampling is a sampling technique in which each sample has an equal probability of being chosen. A sample chosen randomly is meant to be an unbiased representation of the total population. The selected dataset was so unbalanced and biased, so in order to compensate for it, the **Random Over Sampling technique** was used to balance the Dataset. The class of data is the underrepresented minority class in the data sample, oversampling techniques may be used to duplicate these results for a more balanced number of positive results in training. Random oversampling involves randomly selecting examples from the minority class, with replacements, and adding them to the training dataset. Randomly duplicate examples in the minority class.

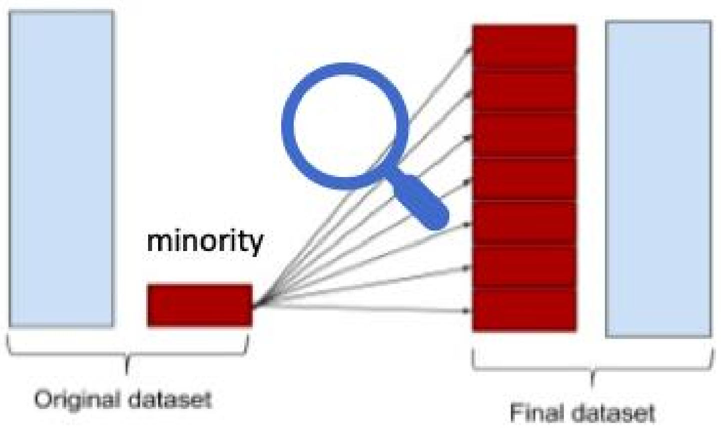


Figure 4.14: Oversampling minority class

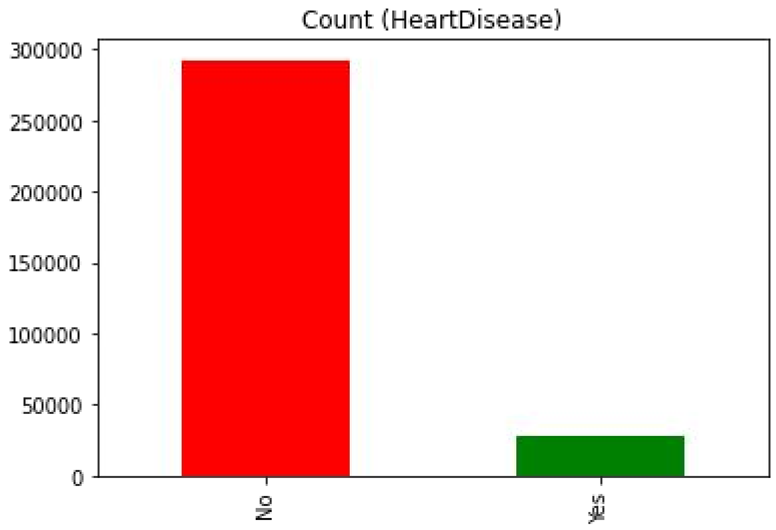


Figure 4.15: Bar Chart representation of Dataset1 - Before Random sampling

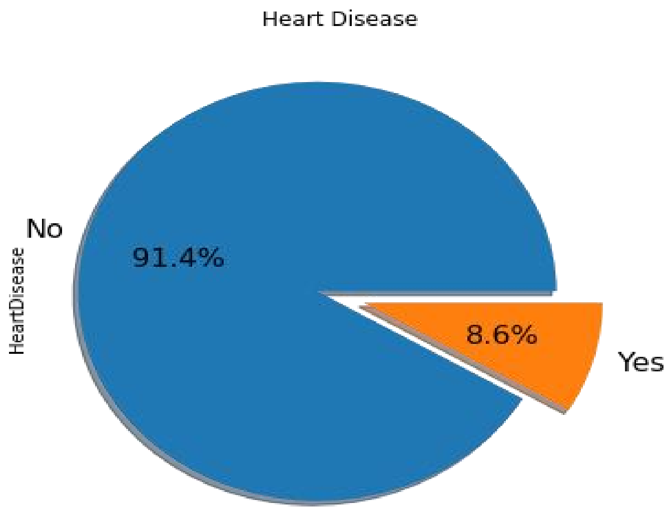


Figure 4.16: Pie-chart representation of dataset-1 before random sampling

It is seen that there are 292422 examples in No class and 27373 in Yes class before oversampling. The random oversample transform is defined to balance the minority class, then fit and applied to the dataset, as stated in sampling strategy='minority'. The class distribution for the new dataset is reported showing that now the Yes class has the same number of examples as the No class.

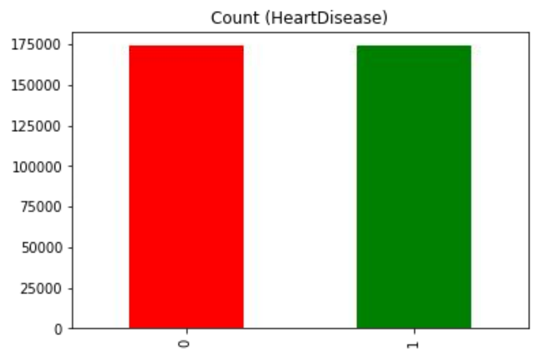


Figure 4.17: Bar-chart representation of dataset-1 after random sampling

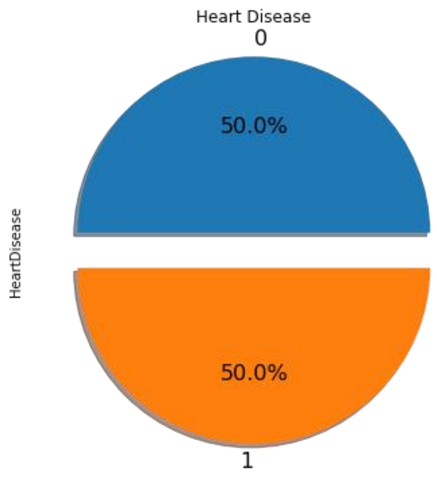


Figure 4.18: Pie-chart representation of dataset-1 after random sampling

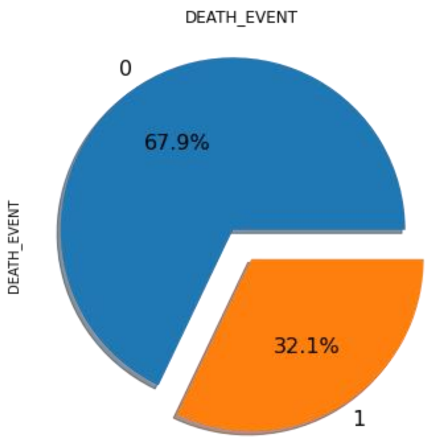


Figure 4.19: Pie-chart representation of dataset-2 before random sampling

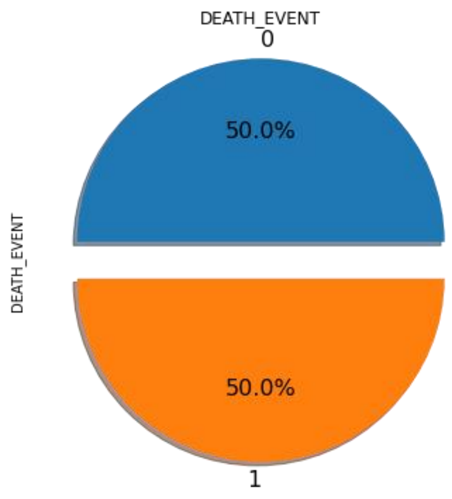
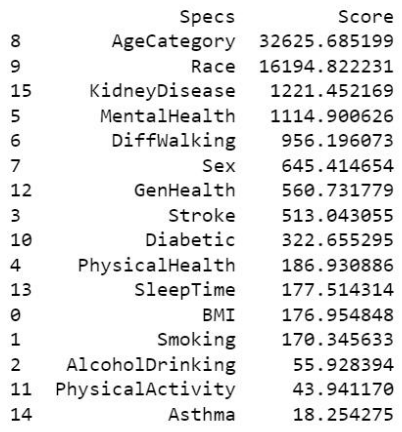


Figure 4.20: Pie-chart representation of dataset-2 after random sampling

4.4 FEATURE SELECTION

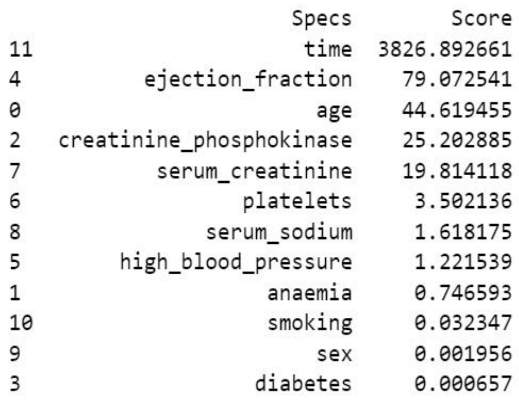
Feature selection isolates the most consistent, non-redundant, and relevant features to use in model construction. Methodically reducing the size of datasets is important as the size and variety of datasets continue to grow. The **Chi-square** (univariate selection) was used to pick certain features that were most closely related to the performance variable. Chi-Square is to be used when the feature is categorical, the target variable in any way that can be thought of as categorical. It measures the degree of association between two categorical variables. In this project, Chi-Square picked the important features which are closely related to Heart Disease Prediction.



```
               Specs         Score
8        AgeCategory  32625.685199
9               Race  16194.822231
15     KidneyDisease   1221.452169
5       MentalHealth   1114.900626
6        DiffWalking    956.196073
7                Sex    645.414654
12         GenHealth    560.731779
3             Stroke    513.043055
10          Diabetic    322.655295
4     PhysicalHealth    186.930886
13         SleepTime    177.514314
0                BMI    176.954848
1            Smoking    170.345633
2    AlcoholDrinking     55.928394
11  PhysicalActivity     43.941170
14            Asthma     18.254275
```

Figure 4.2: Score of Different Attributes in Dataset-1

Age, race, kidney disease, mental health, and walking difficulty have all been highly linked to the development of heart problem. These have higher importance. Whereas, asthma, alcohol drinking, and smoking have less importance.



```
                       Specs        Score
11                      time  3826.892661
4          ejection_fraction    79.072541
0                        age    44.619455
2   creatinine_phosphokinase    25.202885
7           serum_creatinine    19.814118
6                  platelets     3.502136
8               serum_sodium     1.618175
5        high_blood_pressure     1.221539
1                    anaemia     0.746593
10                   smoking     0.032347
9                        sex     0.001956
3                   diabetes     0.000657
```

Figure 4.22: Score of Different Attributes in Dataset-2

Time, ejection fraction, age, and creatinine phosphokinase have all been highly linked to the development of heart problems. These have higher importance. Whereas, sex, smoking, and anemia have less influence on the heart disease

CHAPTER 5

IMPLEMENTATION AND SOURCE CODE

The Jupyter Notebook has become a popular user interface for cloud computing, and major cloud providers have adopted the Jupyter Notebook or derivative tools as a frontend interface for cloud users. Examples include Amazon's SageMaker Notebooks, Google's Colaboratory, and Microsoft's Azure Notebook. Google Colaboratory (also known as Colab) is a free Jupyter notebook environment that runs in the cloud and stores its notebooks on Google Drive. Colab was originally an internal Google project; an attempt was made to open-source all the code and work more directly upstream, leading to the development of the "Open in Colab" Google Chrome extension, but this eventually ended, and Colab development continued internally.

5.1 CODE

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

#Dataset Reading
df = pd.read_csv(r"Dataset\heart_2020_cleaned.csv")
df.head()
df.shape
```

#Random Over Sampling:

```
class_1=class_1.sample(count_class_0,replace=True)
data = pd.concat([class_0, class_1], axis=0)
print(data.HeartDisease.value_counts())
count_class_no, count_class_yes = df['HeartDisease'].value_counts()

# divide the class
df_class_yes = df[df['HeartDisease'] == 1]
df_class_no = df[df['HeartDisease'] == 0]

df_class_yes_over = df_class_yes.sample(count_class_no, replace=True)
df_test_over = pd.concat([df_class_no, df_class_yes_over], axis=0)

print('Random over-sampling')
print(df_test_over['HeartDisease'].value_counts())

df_test_over['HeartDisease'].value_counts().plot(kind='bar',title='Count(HeartDi
sease)', color=['r', 'g'])
plt.figure(figsize=(6,6))

# Pie plot
df_test_over['HeartDisease'].value_counts().plot.pie(explode=[0.1,0.1],
autopct='%1.1f%%', shadow=True, textprops={'fontsize':16}).set_title("Heart
Disease")
```

#Train and Test Splitting

```
train,test=train_test_split(df_test_over,stratify=df_test_over['HeartDisease'],test
_size=.2,random_state=0)
print("train.shape : {}".format(train.shape))
print("test.shape : {}".format(test.shape))
train['Physical_MentalHealth']=train['PhysicalHealth']+train['MentalHealth']
```

```
test['Physical_MentalHealth']=test['PhysicalHealth']+test['MentalHealth']
train['PhysicalHealth_log'] =np.log1p(train['PhysicalHealth'])
train['MentalHealth_log'] = np.log1p(train['MentalHealth'])
test['PhysicalHealth_log'] = np.log1p(test['PhysicalHealth'])
test['MentalHealth_log'] = np.log1p(test['MentalHealth'])

#Feature Scaling

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#Feature Extraction

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data = df_test_over.copy()

X = data.iloc[:,1:17] #independent columns
y = data.iloc[:,-1]    #target column


#apply SelectKBest class to extract the top best features


bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score'] #naming the dataframe columns
print(featureScores.nlargest(18,'Score')) #print best features

#KNN Model

from sklearn.neighbors import KNeighborsClassifier
```

```
   knn = KNeighborsClassifier(n_neighbors=10,
   algorithm='kd_tree', metric='minkowski', p=5)
   knn.fit(x_train, y_train)
   print("KNN Classifier: ")
   print()
   print(classification_report(y_test, model[4].predict(X_test)))
   print(accuracy_score(y_test, model[4].predict(X_test))

#DecisionTree
 from sklearn.tree import DecisionTreeClassifier
 tree_clf = DecisionTreeClassifier(random_state=42)
 tree_clf.fit(X_train, y_train)
 print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
 print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)

 #Naive Bayes
 from sklearn.naive_bayes import GaussianNB
 from sklearn import metrics
 nvc = GaussianNB()
 nvc.fit(X_train, y_train)
 y_pred = nvc.predict(X_test)
 print_score(nvc, X_train, y_train, X_test, y_test, train=True)
 print_score(nvc, X_train, y_train, X_test, y_test, train=False)
```

```
#Logistic Regression
 from sklearn.linear_model import LogisticRegression
    log = LogisticRegression(random_state = 0)
    log.fit(x_train, y_train)
    print(classification_report(y_test, model[0].predict(X_test)))
    print(accuracy_score(y_test, model[0].predict(X_test)))

#Ensemble Methods
 #Bagging
 from sklearn.ensemble import BaggingClassifier
 bag=BaggingClassifier(KNeighborsClassifier(),max_samples=0.5,
 max_features=0.5)
  bag.fit(x_train, y_train)
 print("Bagging Classifier Training Accuracy:", bag.score(X_train,
    y_train))
 bag = BaggingClassifier()
 bag.fit(X_train, y_train)
 y_pred = bag.predict(X_test)
 print(f"The score for the Accuracy score
    is:",accuracy_score(y_test, y_pred))

 #Random Forest
 from sklearn.ensemble import RandomForestClassifier
 from sklearn.model_selection import RandomizedSearchCV
rf_clf=RandomForestClassifier(n_estimators=1000,
random_state=42)
```

```
rf_clf.fit(X_train, y_train)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)


#Hard Voting
clf_hard_voting_estimators =
   [('LR',LogisticRegression(random_state=1)),
                    ('DT', DecisionTreeClassifier(random_state=1)),
                    ('RF',
   RandomForestClassifier(random_state=1))]
clf_hard_voting = VotingClassifier(estimators =
   clf_hard_voting_estimators, voting ='hard')
clf_hard_voting.fit(X_train, y_train)
print('Training set Accuracy
   Score :',clf_hard_voting.score(X_train,y_train))
print('Test set Accuracy
   Score :',clf_hard_voting.score(X_test,y_test))


#Full Stack Ensemble:
models = [
   RandomForestClassifier(criterion='entropy',n_estimators=100),
   MLPClassifier(),
   RandomForestClassifier(criterion='gini',n_estimators=100),
   KNeighborsClassifier(9),
   ExtraTreesClassifier(n_estimators= 500),
   ExtraTreesClassifier(n_estimators= 100),
   xgboost.XGBClassifier(n_estimators= 1000),
    xgboost.XGBClassifier(n_estimators= 100),

    xgboost.XGBClassifier(n_estimators= 500),

    xgboost.XGBClassifier(n_estimators= 2000),

    xgboost.XGBClassifier(),

    SGDClassifier(max_iter=1000, tol=1e-4),

    AdaBoostClassifier(),

    DecisionTreeClassifier(),

    LinearDiscriminantAnalysis(),

GradientBoostingClassifier(n_estimators=100,max_features='sqrt'),

    ExtraTreesClassifier(n_estimators= 1000),

]

S_train, S_test = stacking(models,

                           X_train, y_train, X_test,

                           regression=False,
                           mode='oof_pred_bag',
                           needs_proba=False,
                           save_dir=None,
                           metric=accuracy_score,
                           n_folds=5,
                           stratified=True,
                           shuffle=True,
                           random_state=0,
                           verbose=2)

#Meta Classifier (Generalizer)
model = MLPClassifier()
model = model.fit(S_train, y_train)
y_pred = model.predict(S_test)
y_pred1 = model.predict(S_train)
```

```
 print('Train Score: [%.8f]' % accuracy_score(y_train, y_pred1))
 print('Final prediction score: [%.8f]' % accuracy_score(y_test,
    y_pred))


# Calculate accuracy, precision, recall, f1-score, and kappa score
acc = metrics.accuracy_score(y_test, y_pred)
    prec = metrics.precision_score(y_test, y_pred)
    rec = metrics.recall_score(y_test, y_pred)
    f1 = metrics.f1_score(y_test, y_pred)

    kappa = metrics.cohen_kappa_score(y_test, y_pred)
 # Calculate area under curve (AUC)
    y_pred_proba = model.predict_proba(x_test)[::,1]
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
    auc = metrics.roc_auc_score(y_test, y_pred_proba)
  # Displaying confussion matrix

    cm = metrics.confusion_matrix(y_test, y_pred)

 return {'acc': acc, 'prec': prec, 'rec': rec, 'f1': f1, 'kappa': kappa,

          'fpr': fpr, 'tpr': tpr, 'auc': auc, 'cm': cm}
```

CHAPTER 6

RESULT AND DISCUSSIONS

The training dataset-1 with 278547 instances 80%. The result has been used to test the model on the testing data which has 69637 instances of 20%.

The training dataset-2 with instances 284 80%. The result has been used to test the model on the testing data with has instances 122 of 20%.

6.1 PERFORMANCE METRICS

6.1.1 CONFUSION MATRIX

Confusion Matrix is the most effective tool to analyze cardiac disease prediction in this field of study. The 2x2 Confusion Matrix is used for evaluating our model. The matrix compares the actual target values with those predicted by our model. It is deployed to perceive the behavior of the different classifiers. Such a matrix provides the information about the way the classifier has performed of matching the correctly predicted examples corresponding to the incorrectly predicted examples. Therefore, a confusion matrix is the tabular representation of the model estimated and the actual values of the dataset. It is used by machine learning classification-based problems for measuring the performance of the models. It consists of four separate commixtures of the predicted – actual values namely True Negative, True Positive and False Positive and False Negative as illustrated clearly in.

True Positive: Model does the correct prediction of the positive class (patient has the disease) whereas True Negative: correct prediction of the negative class (patient does not have the disease). False Positive: Incorrect prediction of positive class whereas incorrect prediction of the negative class is coined as False Negative. In medical testing, too many false negatives are very risky, since reports will not show that the person is having heart disease when the person is having heart disease.

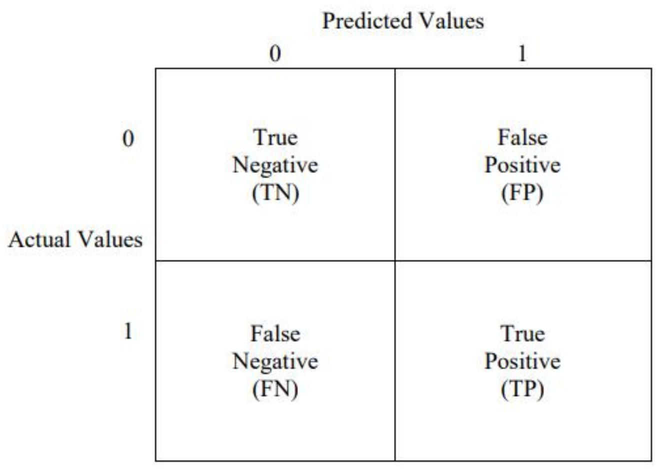


Figure 6.1: Confusion Matrix

6.1.2 SENSITIVITY/ RECALL SENSITIVITY

Sensitivity / Recall Sensitivity indicates the proportions of cardiac patients diagnosed (by the model) with cardiac disease. It is also termed recall.

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

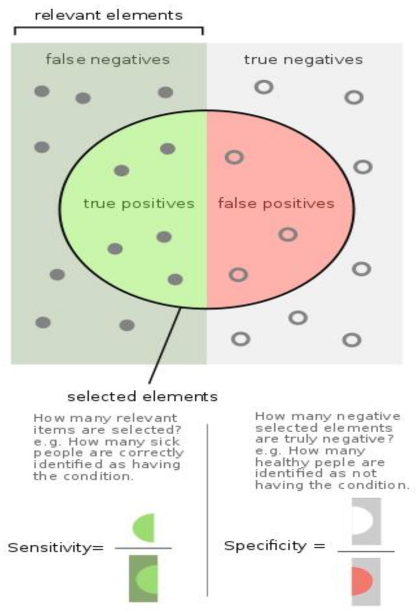

relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many relevant items are selected? e.g. How many sick people are correctly identified as having the condition.
How many negative selected elements are truly negative? e.g. How many healthy peple are identified as not having the condition.
Sensitivity=
Specificity =

Figure 6.2: Sensitivity and Specificity

6.1.3 RECALL

For ailment treatment purposes, it is the ability of the test to correctly detect ill patients. Thus, it is the proportion of those testing positive among those having it. 100% recall identifies all with the disease.

6.1.4 SPECIFICITY

Specificity indicates the proportions of patients not having the disease been forecasted by the model to the category of non-cardiac disease. It is the exact opposite of Sensitivity. Thus, the Specificity for a classification problem is given as:

$$Specificity = \frac{TN}{TN+FP}$$

6.1.5 PRECISION

Precision provides information about the proportion of those classified by the model as with the disease, had heart disease.

Precision = (True Positive) / (True Positive + False Positive)

6.1.6 F1 SCORE

F1 Score is defined as the harmonic mean of sensitivity (or recall) and precision assigning a single number. F1 Score calculation is expressed as mentioned in equation.

F1 Score = (2Precision + Recall) / (Precision + Recall)

6.1.7 ACCURACY

Accuracy is one of the most commonly used measures used to evaluate a classifier's performance. It's calculated as a percentage of correctly identified samples, and it's written as:

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

6.1.8 MATHEWS CORRELATION COEFFICIENT(MCC)

Mathew’s correlation coefficient (MCC) is a correlation coefficient between observed and predicted classes. MCC 1⁄4 + 1 indicates a flawless prediction, MCC 1⁄4 0 indicates no better than the random prediction, and MCC 1⁄4 1 indicates complete disagreement between observed and projected values. Even though the class sizes are substantially different, this statistic is generally considered a balanced measure.

$$MCC = \frac{TP.TN - FP.FN}{\sqrt{(TP+FP).(TP+FN).(TN.+FP).(TN+FN)}}$$

6.1.9 AUC-ROC

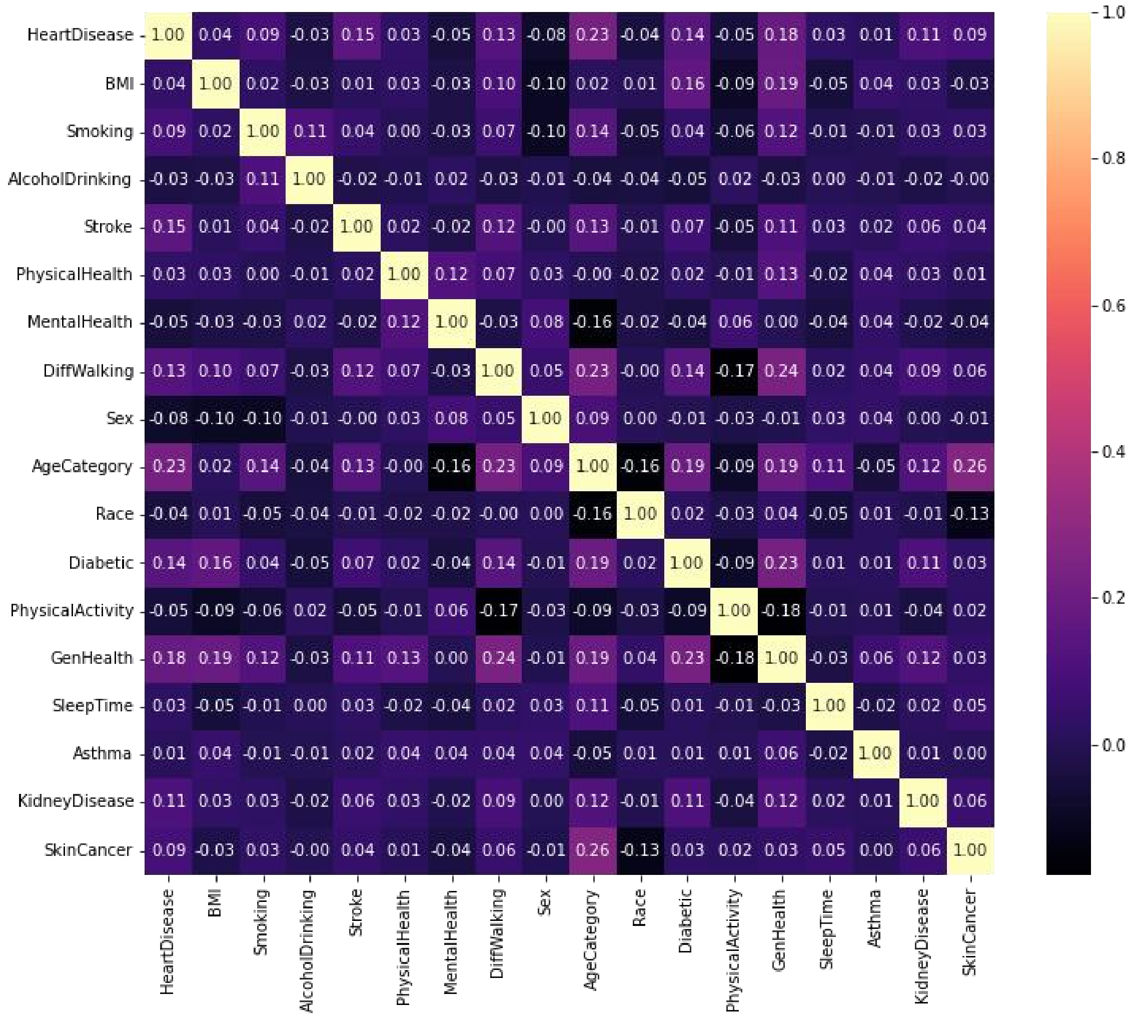
For classification difficulties, it is also a useful and widely used performance metric. It is plotted using TPR vs FPR at various threshold values. Because it evaluates performance across a wide variety of class distributions and error levels, the AU-ROC is an excellent metric for performance comparison.

$$AU - ROC = 1/2\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

6.1.10 LOSS FUNCTION

The performance of a classification model with a prediction input of a probability value between 0 and 1 is measured using logarithmic loss. Our machine learning algorithms aim to reduce this value as much as possible. The log loss in a perfect model would be zero. As the anticipated probability diverges from the actual label, log loss grows. As a result, forecasting a probability of.012 when the actual observation label is 1 is bad and will result in a large log loss.

The graph below shows the range of possible log loss values given a true observation (isDog = 1). As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly.



	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHealth	SleepTime	Asthma	KidneyDisease	SkinCancer
HeartDisease	1.00	0.04	0.09	-0.03	0.15	0.03	-0.05	0.13	-0.08	0.23	-0.04	0.14	-0.05	0.18	0.03	0.01	0.11	0.09
BMI	0.04	1.00	0.02	-0.03	0.01	0.03	-0.03	0.10	-0.10	0.02	0.01	0.16	-0.09	0.19	-0.05	0.04	0.03	-0.03
Smoking	0.09	0.02	1.00	0.11	0.04	0.00	-0.03	0.07	-0.10	0.14	-0.05	0.04	-0.06	0.12	-0.01	-0.01	0.03	0.03
AlcoholDrinking	-0.03	-0.03	0.11	1.00	-0.02	-0.01	0.02	-0.03	-0.01	-0.04	-0.04	-0.05	0.02	-0.03	0.00	-0.01	-0.02	-0.00
Stroke	0.15	0.01	0.04	-0.02	1.00	0.02	-0.02	0.12	-0.00	0.13	-0.01	0.07	-0.05	0.11	0.03	0.02	0.06	0.04
PhysicalHealth	0.03	0.03	0.00	-0.01	0.02	1.00	0.12	0.07	0.03	-0.00	-0.02	0.02	-0.01	0.13	-0.02	0.04	0.03	0.01
MentalHealth	-0.05	-0.03	-0.03	0.02	-0.02	0.12	1.00	-0.03	0.08	-0.16	-0.02	-0.04	0.06	0.00	-0.04	0.04	-0.02	-0.04
DiffWalking	0.13	0.10	0.07	-0.03	0.12	0.07	-0.03	1.00	0.05	0.23	-0.00	0.14	-0.17	0.24	0.02	0.04	0.09	0.06
Sex	-0.08	-0.10	-0.10	-0.01	-0.00	0.03	0.08	0.05	1.00	0.09	0.00	-0.01	-0.03	-0.01	0.03	0.04	0.00	-0.01
AgeCategory	0.23	0.02	0.14	-0.04	0.13	-0.00	-0.16	0.23	0.09	1.00	-0.16	0.19	-0.09	0.19	0.11	-0.05	0.12	0.26
Race	-0.04	0.01	-0.05	-0.04	-0.01	-0.02	-0.02	-0.00	0.00	-0.16	1.00	0.02	-0.03	0.04	-0.05	0.01	-0.01	-0.13
Diabetic	0.14	0.16	0.04	-0.05	0.07	0.02	-0.04	0.14	-0.01	0.19	0.02	1.00	-0.09	0.23	0.01	0.01	0.11	0.03
PhysicalActivity	-0.05	-0.09	-0.06	0.02	-0.05	-0.01	0.06	-0.17	-0.03	-0.09	-0.03	-0.09	1.00	-0.18	-0.01	0.01	-0.04	0.02
GenHealth	0.18	0.19	0.12	-0.03	0.11	0.13	0.00	0.24	-0.01	0.19	0.04	0.23	-0.18	1.00	-0.03	0.06	0.12	0.03
SleepTime	0.03	-0.05	-0.01	0.00	0.03	-0.02	-0.04	0.02	0.03	0.11	-0.05	0.01	-0.01	-0.03	1.00	-0.02	0.02	0.05
Asthma	0.01	0.04	-0.01	-0.01	0.02	0.04	0.04	0.04	0.04	-0.05	0.01	0.01	0.01	0.06	-0.02	1.00	0.01	0.00
KidneyDisease	0.11	0.03	0.03	-0.02	0.06	0.03	-0.02	0.09	0.00	0.12	-0.01	0.11	-0.04	0.12	0.02	0.01	1.00	0.06
SkinCancer	0.09	-0.03	0.03	-0.00	0.04	0.01	-0.04	0.06	-0.01	0.26	-0.13	0.03	0.02	0.03	0.05	0.00	0.06	1.00

Figure 6.3: Correlation matrix between all the attributes in dataset-1

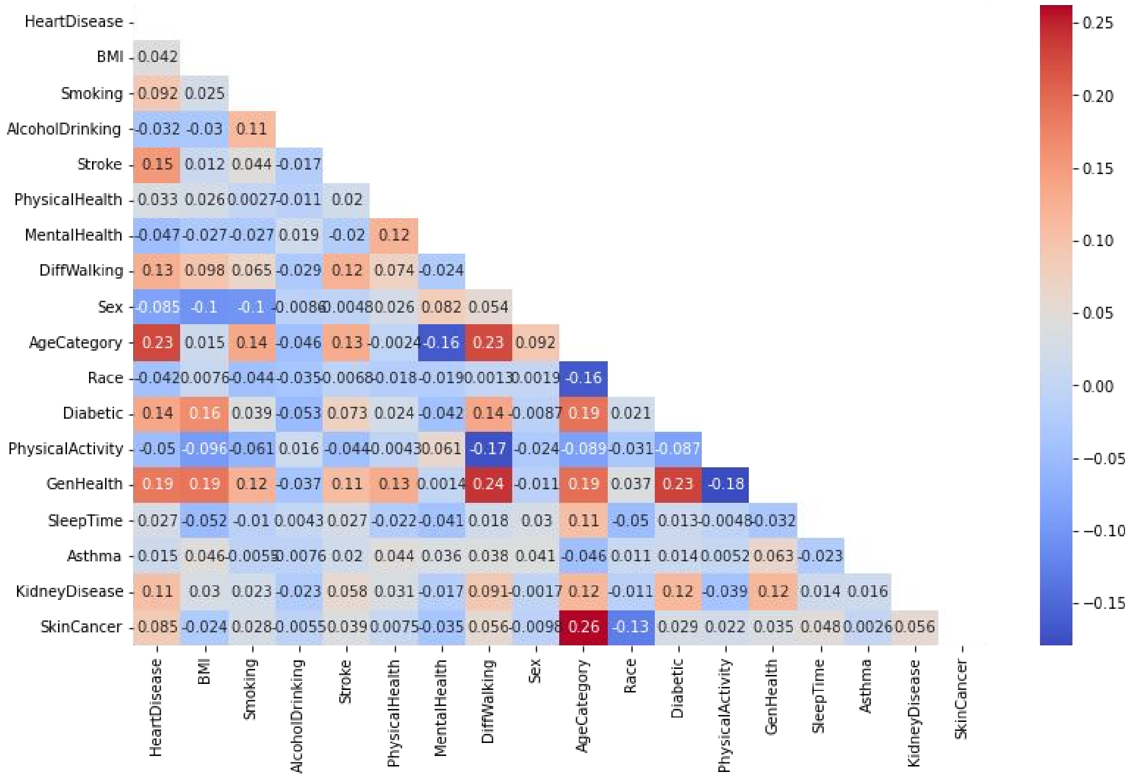


Figure 6.4: Correlation matrix between each attribute in dataset-1

Figure 6.3 and 6.4 shows us the relationship between each attribute with one. This also helps us to understand how each attribute is correlated with one another.

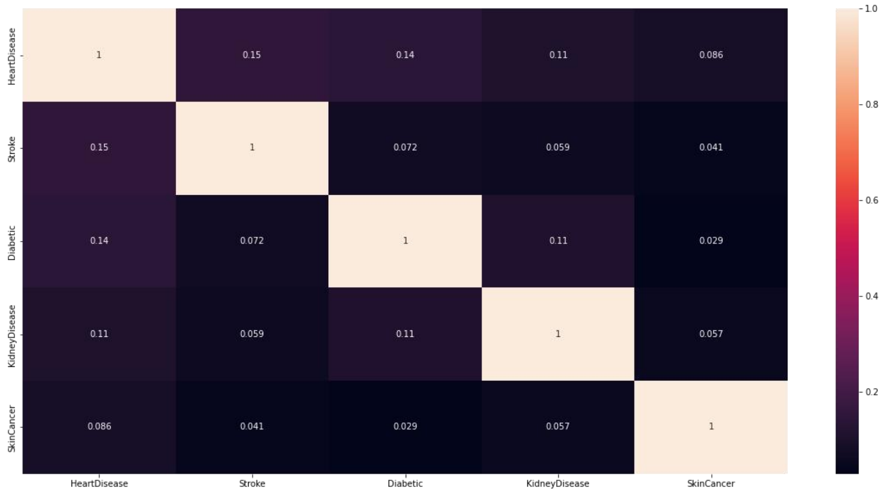


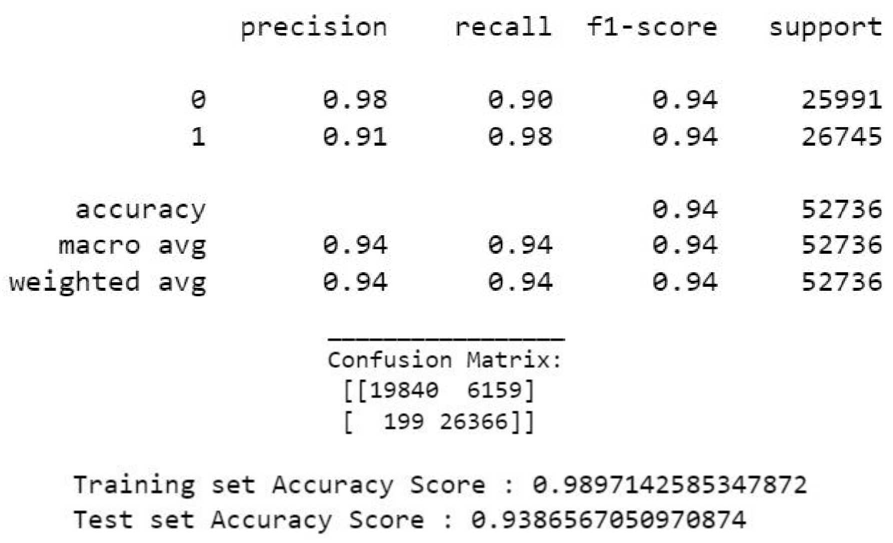
Figure 6.5: Correlation matrix for all the diseases in the attributes

Figure 6.5 shows the relationship between all the diseases in the attributes one another and how much it contributes in the occurrence of heart disease.

1. Heart disease and stroke are correlated
2. Diabetes is correlated with heart disease and kidney disease. Less with Stroke.
3. Skin cancer is not correlated with other 4 diseases.

6.2 DATA SET – 1 RESULTS

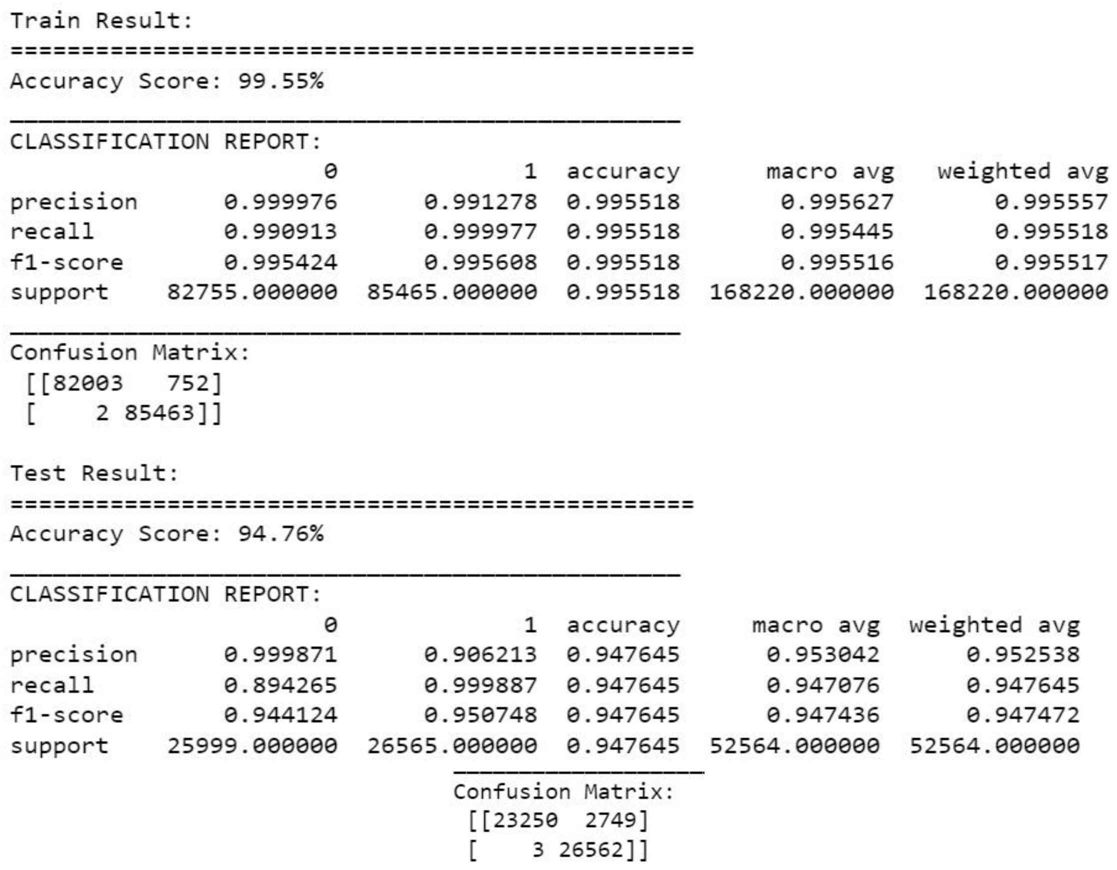
6.2.1 INDIVIDUAL MODELS



```
              precision    recall  f1-score   support

           0       0.98      0.90      0.94     25991
           1       0.91      0.98      0.94     26745

    accuracy                           0.94     52736
   macro avg       0.94      0.94      0.94     52736
weighted avg       0.94      0.94      0.94     52736

                  _______________
                  Confusion Matrix:
                   [[19840  6159]
                   [  199 26366]]

     Training set Accuracy Score : 0.9897142585347872
     Test set Accuracy Score : 0.9386567050970874
```

Figure 6.6: KNN Result

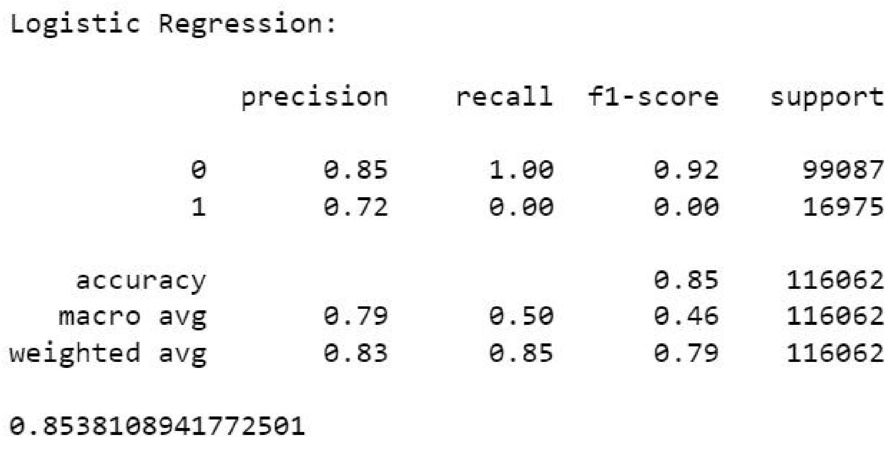
From figure 6.6 it is inferred that a training accuracy of 98.97% and testing accuracy of 93.85% has been achieved when KNN is used as the prediction model.



```
Train Result:
================================================
Accuracy Score: 99.55%
_______________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy      macro avg   weighted avg
precision      0.999976      0.991278  0.995518       0.995627       0.995557
recall         0.990913      0.999977  0.995518       0.995445       0.995518
f1-score       0.995424      0.995608  0.995518       0.995516       0.995517
support    82755.000000  85465.000000  0.995518  168220.000000  168220.000000
_______________________________________________
Confusion Matrix:
 [[82003   752]
 [    2 85463]]

Test Result:
================================================
Accuracy Score: 94.76%
_______________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy     macro avg  weighted avg
precision      0.999871      0.906213  0.947645      0.953042      0.952538
recall         0.894265      0.999887  0.947645      0.947076      0.947645
f1-score       0.944124      0.950748  0.947645      0.947436      0.947472
support    25999.000000  26565.000000  0.947645  52564.000000  52564.000000
                                ________________
                                Confusion Matrix:
                                 [[23250  2749]
                                 [    3 26562]]
```

Figure 6.7: Decision Tree Result

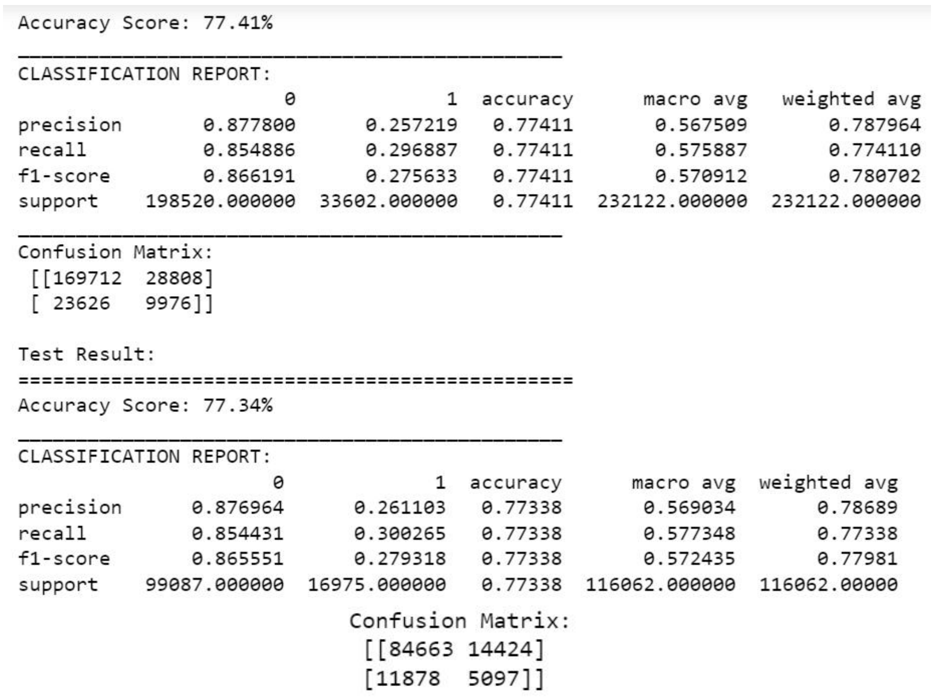
From figure 6.7 it is inferred that a training accuracy of 99.55% and testing accuracy of 94.76% has been achieved when the decision tree is used as the prediction model.



```
Logistic Regression:

              precision    recall  f1-score   support

           0       0.85      1.00      0.92     99087
           1       0.72      0.00      0.00     16975

    accuracy                           0.85    116062
   macro avg       0.79      0.50      0.46    116062
weighted avg       0.83      0.85      0.79    116062

0.8538108941772501
```

Figure 6.8 LR Result

From figure 6.8 it is inferred that a training accuracy of 88.65% and testing accuracy of 85.46% has been achieved when LR is used as the prediction model.

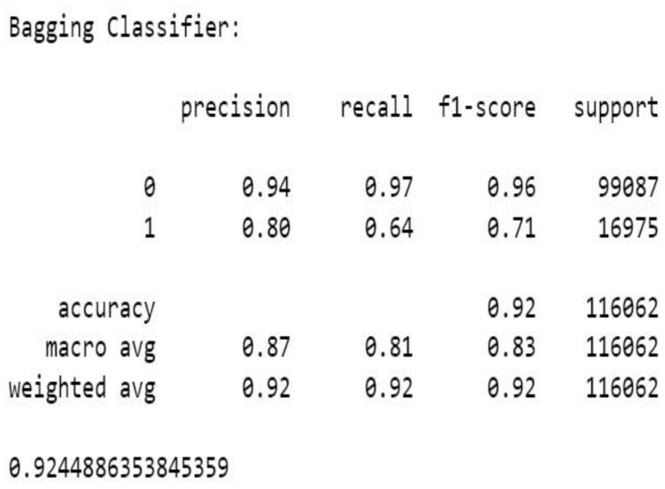


```
Accuracy Score: 77.41%
_______________________________________________
CLASSIFICATION REPORT:
                      0            1  accuracy      macro avg   weighted avg
precision      0.877800     0.257219   0.77411       0.567509       0.787964
recall         0.854886     0.296887   0.77411       0.575887       0.774110
f1-score       0.866191     0.275633   0.77411       0.570912       0.780702
support   198520.000000 33602.000000   0.77411  232122.000000  232122.000000
_______________________________________________
Confusion Matrix:
 [[169712  28808]
 [ 23626   9976]]

Test Result:
================================================
Accuracy Score: 77.34%
_______________________________________________
CLASSIFICATION REPORT:
                     0            1  accuracy      macro avg  weighted avg
precision     0.876964     0.261103   0.77338       0.569034       0.78689
recall        0.854431     0.300265   0.77338       0.577348       0.77338
f1-score      0.865551     0.279318   0.77338       0.572435       0.77981
support   99087.000000 16975.000000   0.77338  116062.000000  116062.00000
                                 Confusion Matrix:
                                  [[84663 14424]
                                  [11878  5097]]
```

Figure 6.9 Naive Bayes Result

From figure 6.9 it is inferred that a training accuracy of 77.41% and testing accuracy of 77.34% has been achieved when Naive Bayes is used as the prediction model.

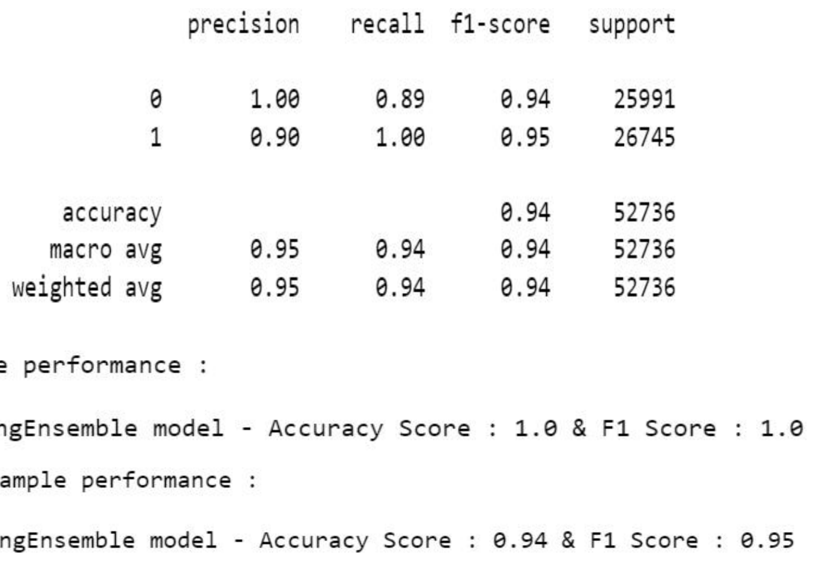
6.2.2 ENSEMBLE MODELS



```
Bagging Classifier:

              precision    recall  f1-score   support

           0       0.94      0.97      0.96     99087
           1       0.80      0.64      0.71     16975

    accuracy                           0.92    116062
   macro avg       0.87      0.81      0.83    116062
weighted avg       0.92      0.92      0.92    116062

0.9244886353845359
```

Figure 6.10 Bagging Classifier Result

From figure 6.10 it is inferred that a training accuracy of 99.1% and testing accuracy of 93.53% has been achieved when Bagging Classifier is used as the prediction model.



```
              precision    recall  f1-score   support

           0       1.00      0.89      0.94     25991
           1       0.90      1.00      0.95     26745

    accuracy                           0.94     52736
   macro avg       0.95      0.94      0.94     52736
weighted avg       0.95      0.94      0.94     52736

In-sample performance :

HardVotingEnsemble model - Accuracy Score : 1.0 & F1 Score : 1.0

Out-of-sample performance :

HardVotingEnsemble model - Accuracy Score : 0.94 & F1 Score : 0.95
```

Figure 6.11 Hard Voting Classifier Result

From figure 6.11 it is inferred that a training accuracy of 100% and testing accuracy of 94% has been achieved when Hard Voting is used as the prediction model.

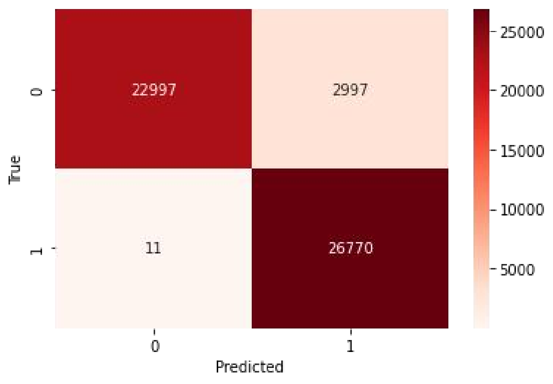


Figure 6.12: Confusion Matrix of Hard Voting Classifier

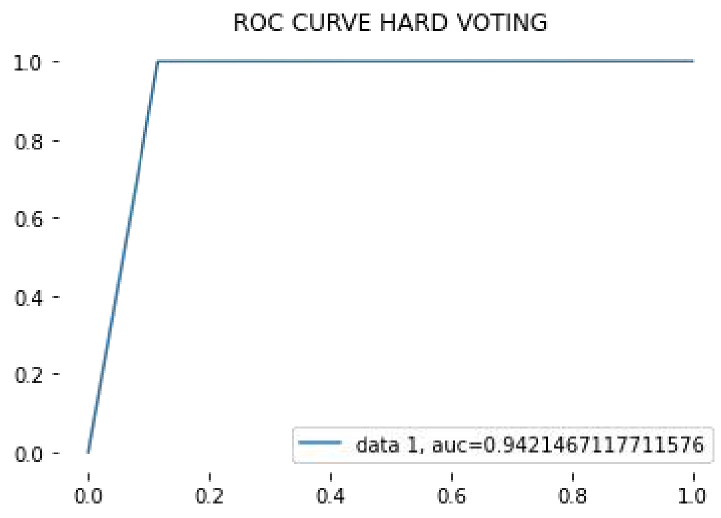
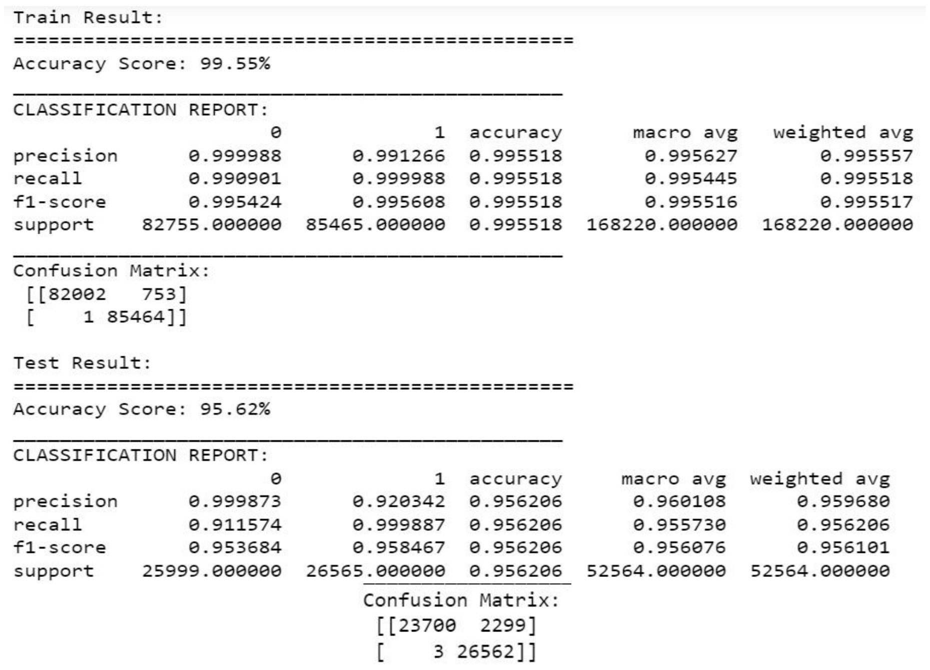


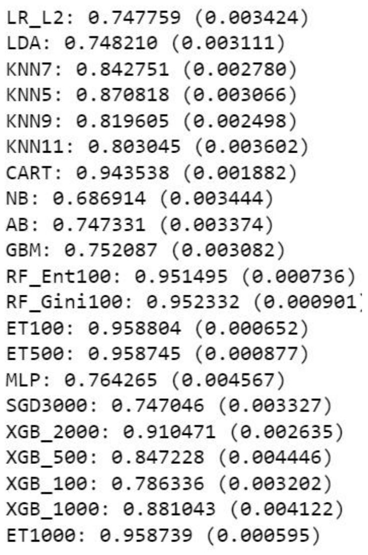
Figure 6.13: ROC and AUC curve of Hard Voting Classifier



```
Train Result:
================================================
Accuracy Score: 99.55%
_______________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy      macro avg   weighted avg
precision      0.999988      0.991266  0.995518       0.995627       0.995557
recall         0.990901      0.999988  0.995518       0.995445       0.995518
f1-score       0.995424      0.995608  0.995518       0.995516       0.995517
support    82755.000000  85465.000000  0.995518  168220.000000  168220.000000
_______________________________________________
Confusion Matrix:
 [[82002   753]
 [    1 85464]]

Test Result:
================================================
Accuracy Score: 95.62%
_______________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy     macro avg  weighted avg
precision      0.999873      0.920342  0.956206      0.960108      0.959680
recall         0.911574      0.999887  0.956206      0.955730      0.956206
f1-score       0.953684      0.958467  0.956206      0.956076      0.956101
support    25999.000000  26565.000000  0.956206  52564.000000  52564.000000
                              Confusion Matrix:
                               [[23700  2299]
                               [    3 26562]]
```

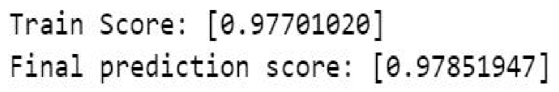
Figure 6.14: RF Result

From figure 6.14 it is inferred that a training accuracy of 99.55% and testing accuracy of 95.62% has been achieved when RF is used for as the prediction model.



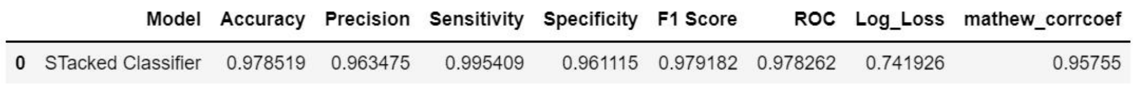
```
LR_L2: 0.747759 (0.003424)
LDA: 0.748210 (0.003111)
KNN7: 0.842751 (0.002780)
KNN5: 0.870818 (0.003066)
KNN9: 0.819605 (0.002498)
KNN11: 0.803045 (0.003602)
CART: 0.943538 (0.001882)
NB: 0.686914 (0.003444)
AB: 0.747331 (0.003374)
GBM: 0.752087 (0.003082)
RF_Ent100: 0.951495 (0.000736)
RF_Gini100: 0.952332 (0.000901)
ET100: 0.958804 (0.000652)
ET500: 0.958745 (0.000877)
MLP: 0.764265 (0.004567)
SGD3000: 0.747046 (0.003327)
XGB_2000: 0.910471 (0.002635)
XGB_500: 0.847228 (0.004446)
XGB_100: 0.786336 (0.003202)
XGB_1000: 0.881043 (0.004122)
ET1000: 0.958739 (0.000595)
```

Figure 6.15: Full Stacking Training Accuracy of the models



```
Train Score: [0.97701020]
Final prediction score: [0.97851947]
```

Figure 6.16: Final Accuracy Scores of Full stack



	Model	Accuracy	Precision	Sensitivity	Specificity	F1 Score	ROC	Log_Loss	mathew_corrcoef
0	STacked Classifier	0.978519	0.963475	0.995409	0.961115	0.979182	0.978262	0.741926	0.95755

Figure 6.17: Final Results of Full stack

From figure 6.16 and figure 6.17 it is inferred that a training accuracy of 97.7% and testing accuracy of 98.01% has been achieved when full-stack is used as the prediction model. When compared to other models and its result the prediction result of the full-stack classifier is higher than the others. From this, we can confer that the full-stack model is the suitable one for heart disease prediction.

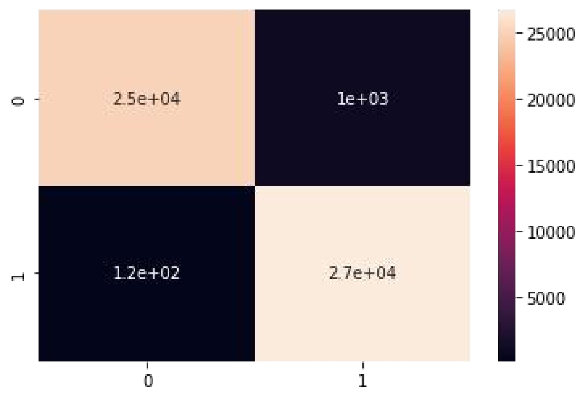
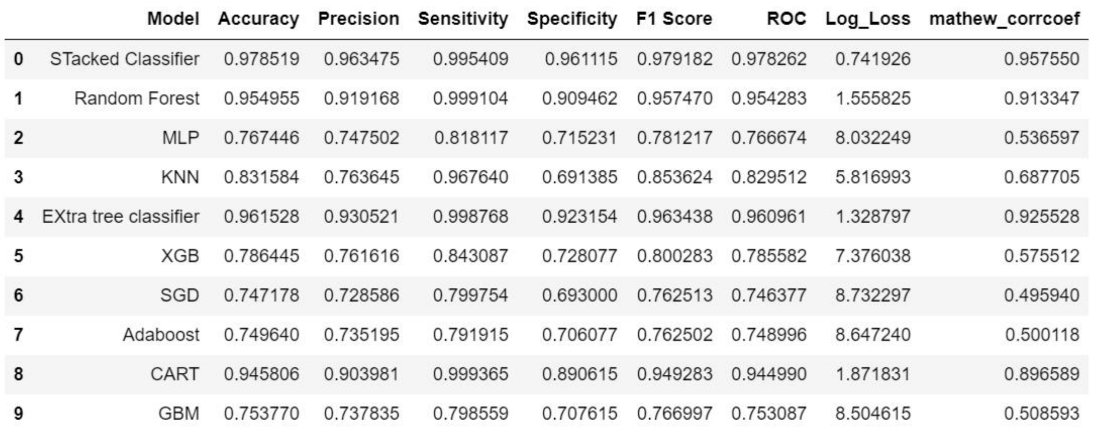


Figure 6.18: Confusion Matrix of Full stack



	Model	Accuracy	Precision	Sensitivity	Specificity	F1 Score	ROC	Log_Loss	mathew_corrcoef
0	STacked Classifier	0.978519	0.963475	0.995409	0.961115	0.979182	0.978262	0.741926	0.957550
1	Random Forest	0.954955	0.919168	0.999104	0.909462	0.957470	0.954283	1.555825	0.913347
2	MLP	0.767446	0.747502	0.818117	0.715231	0.781217	0.766674	8.032249	0.536597
3	KNN	0.831584	0.763645	0.967640	0.691385	0.853624	0.829512	5.816993	0.687705
4	EXtra tree classifier	0.961528	0.930521	0.998768	0.923154	0.963438	0.960961	1.328797	0.925528
5	XGB	0.786445	0.761616	0.843087	0.728077	0.800283	0.785582	7.376038	0.575512
6	SGD	0.747178	0.728586	0.799754	0.693000	0.762513	0.746377	8.732297	0.495940
7	Adaboost	0.749640	0.735195	0.791915	0.706077	0.762502	0.748996	8.647240	0.500118
8	CART	0.945806	0.903981	0.999365	0.890615	0.949283	0.944990	1.871831	0.896589
9	GBM	0.753770	0.737835	0.798559	0.707615	0.766997	0.753087	8.504615	0.508593

Figure 6.19: Accuracy of each model stacked in full-stack

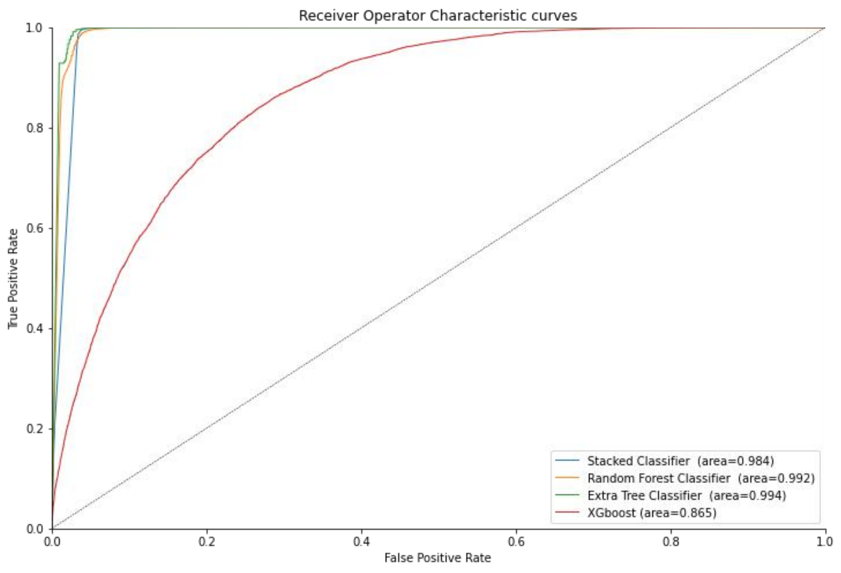


Figure 6.20: ROC and AUC curve of full-stack

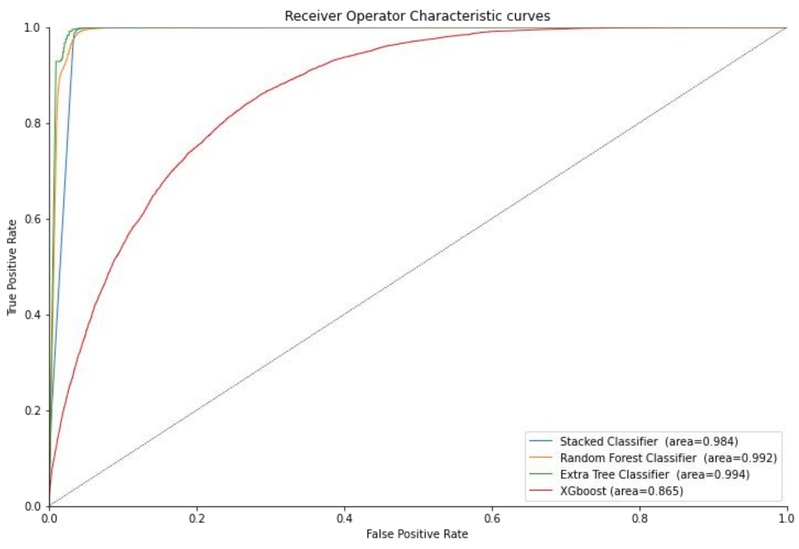


Figure 6.21: Prediction - Recall curve of full-stack

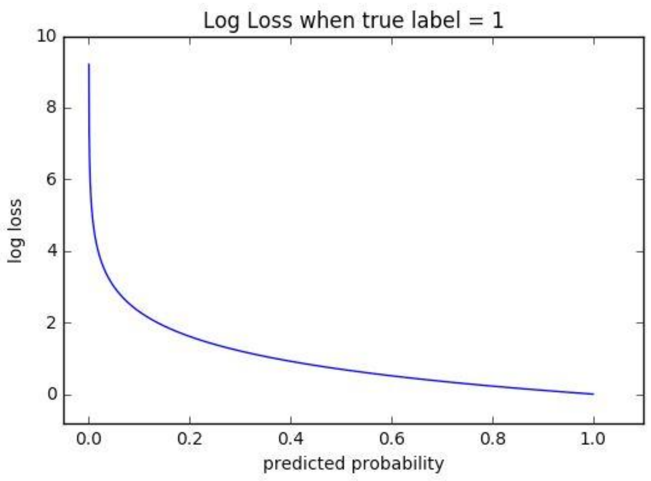
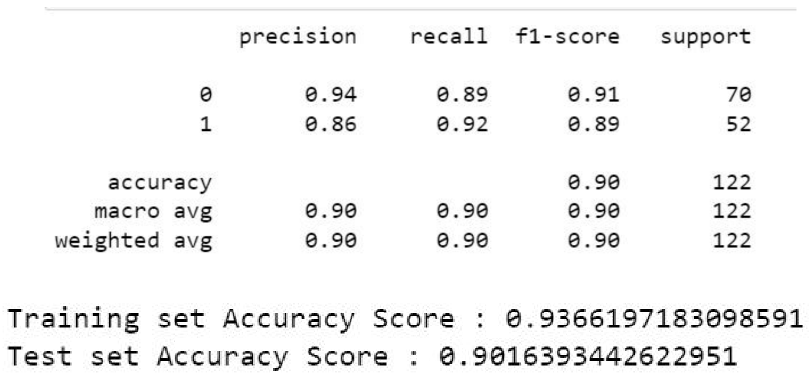


Figure 6.22: Loss function of Full Stack

6.3 DATA SET – 2 RESULTS

6.3.1 ENSEMBLE METHODS



```
              precision    recall  f1-score   support

           0       0.94      0.89      0.91        70
           1       0.86      0.92      0.89        52

    accuracy                           0.90       122
   macro avg       0.90      0.90      0.90       122
weighted avg       0.90      0.90      0.90       122

Training set Accuracy Score : 0.9366197183098591
Test set Accuracy Score : 0.9016393442622951
```

Figure 6.23: Hard voting result

From figure 6.23 it is inferred that a training accuracy of 93.66% and testing accuracy of 90.16% has been achieved when Hard voting is used as the prediction model.

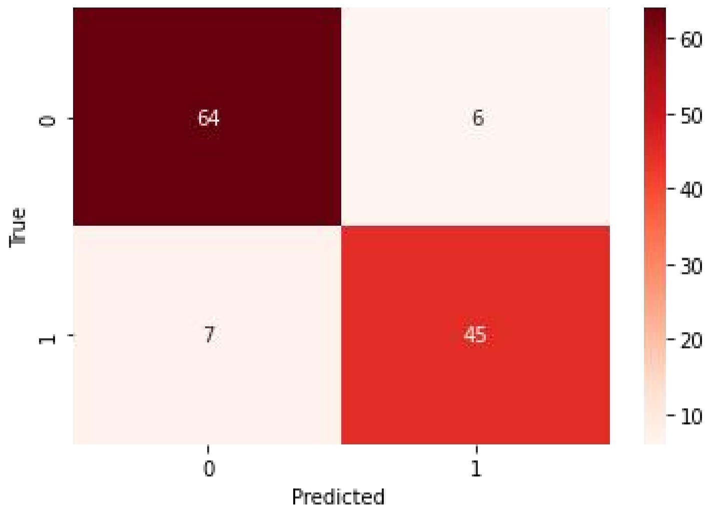


Figure 6.24: Confusion matrix of hard voting classifier

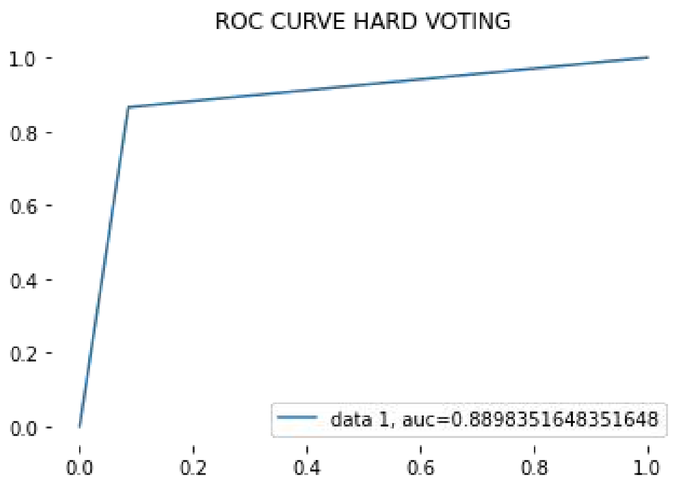
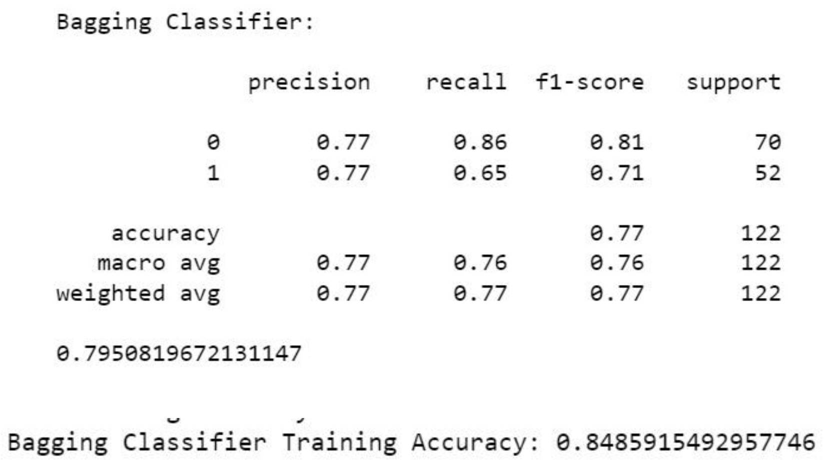


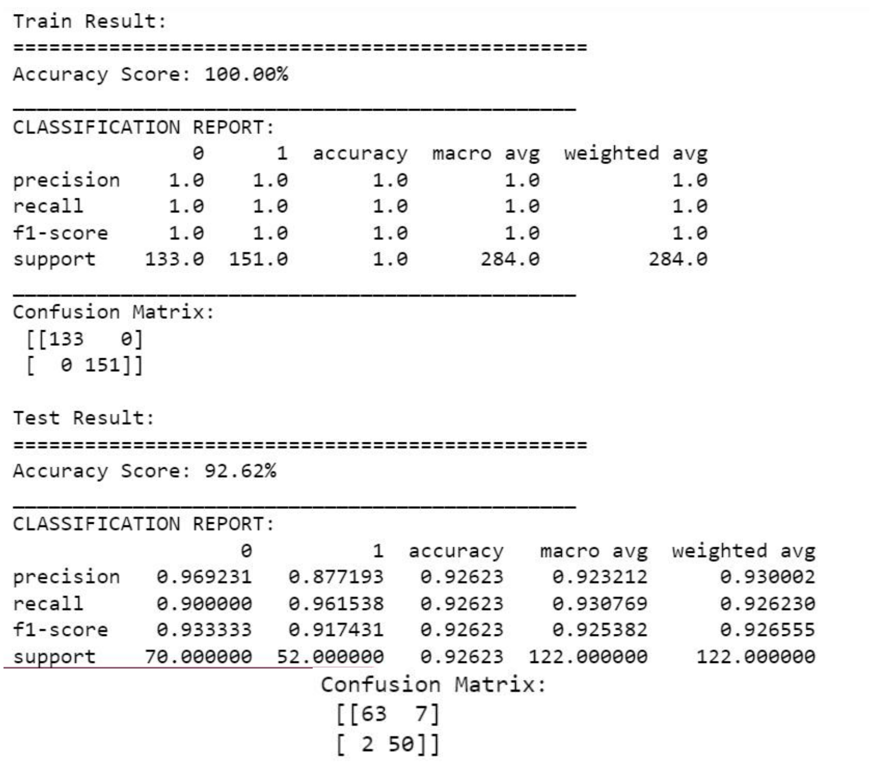
Figure 6.25: ROC and AUC curve of hard voting classifier



```
Bagging Classifier:

              precision    recall  f1-score   support

           0       0.77      0.86      0.81        70
           1       0.77      0.65      0.71        52

    accuracy                           0.77       122
   macro avg       0.77      0.76      0.76       122
weighted avg       0.77      0.77      0.77       122

0.7950819672131147

Bagging Classifier Training Accuracy: 0.8485915492957746
```

Figure 6.26: Bagging classifier accuracy

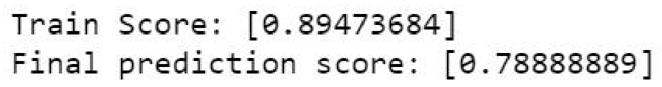
From figure 6.26 it is inferred that a training accuracy of 84% and testing accuracy of 79.5% has been achieved when the Bagging classifier is used as the prediction model.



```
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
               0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0       1.0        1.0           1.0
recall       1.0    1.0       1.0        1.0           1.0
f1-score     1.0    1.0       1.0        1.0           1.0
support    133.0  151.0       1.0      284.0         284.0
_______________________________________________
Confusion Matrix:
 [[133   0]
 [  0 151]]

Test Result:
================================================
Accuracy Score: 92.62%
_______________________________________________
CLASSIFICATION REPORT:
                   0          1  accuracy   macro avg  weighted avg
precision   0.969231   0.877193   0.92623    0.923212      0.930002
recall      0.900000   0.961538   0.92623    0.930769      0.926230
f1-score    0.933333   0.917431   0.92623    0.925382      0.926555
support    70.000000  52.000000   0.92623  122.000000    122.000000
                          Confusion Matrix:
                           [[63  7]
                           [ 2 50]]
```

Figure 6.27: RF Result

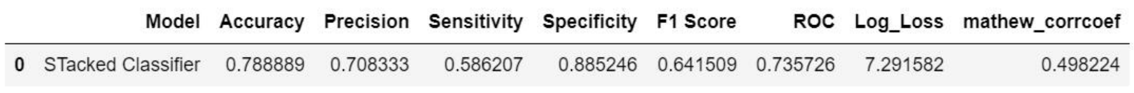
From figure 6.27 it is inferred that a training accuracy of 100% and testing accuracy of 92.62% has been achieved when RF is used as the prediction model.



```
Train Score: [0.89473684]
Final prediction score: [0.78888889]
```

Figure 6.28: Full Stack Result

From figure 6.28 it is inferred that a training accuracy of 89.47% and testing accuracy of 78.88% has been achieved when RF is used as the prediction model.



	Model	Accuracy	Precision	Sensitivity	Specificity	F1 Score	ROC	Log_Loss	mathew_corrcoef
0	STacked Classifier	0.788889	0.708333	0.586207	0.885246	0.641509	0.735726	7.291582	0.498224

Figure 6.29: Accuracy of model stacked in full-stack for dataset-2

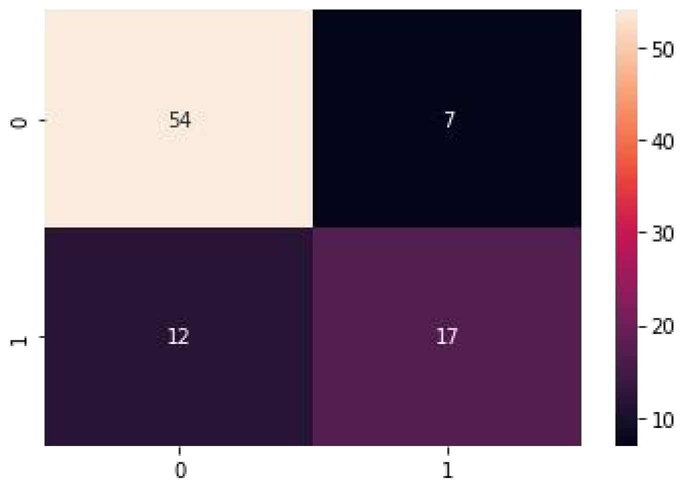


Figure 6.30: Confusion Matrix of Full-stack

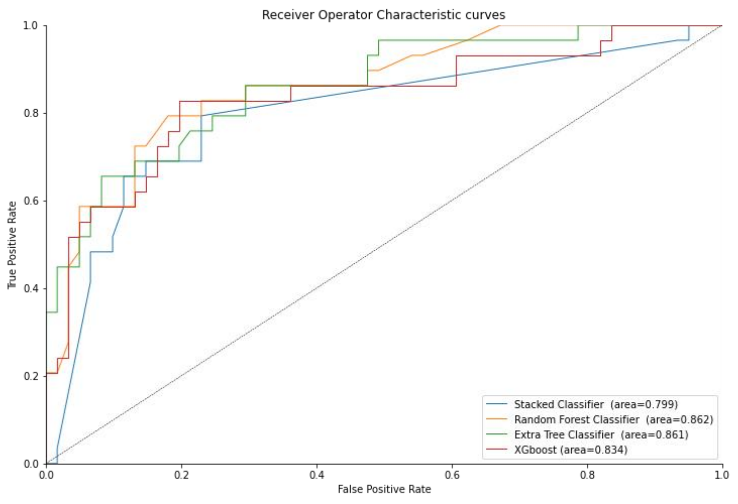


Figure 6.31: ROC and AUC Curve of Full-stack

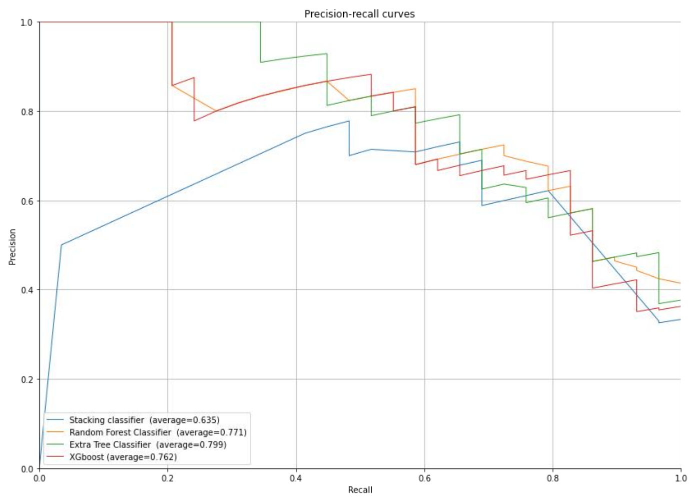


Figure 6.32: Precision-Recall Curve of Full-Stack

6.4 COMPARISON OF ACCURACY

Models	**Train Accuracy**	**Test Accuracy**	**Precision**	**F1 Score**
Bagging	0.991	0.9353	0.94	0.96
Hard Voting	1.00	0.94	0.95	0.95
RF	0.996	95.62	0.95	0.931
Full Stack Ensemble	**0.9770**	**98.01**	**0.963**	**0.9792**

Table 6.1: Performance of Ensemble Methods Comparison Table with Kaggle Dataset-1

Models	**Train Accuracy**	**Test Accuracy**	**F1 Score**	**Precision**
Baaging	0.84	0.80	0.77	0.77
Hard Voting	**0.94**	**0.90**	**0.91**	**0.89**
RF	1.00	0.92	0.933	0.969
Full-stacked ensemble	0.894	0.7888	0.6415	0.7083

Table 6.2: Performance of Ensemble Methods Comparison Table with Kaggle Dataset-2

Models	Train Accuracy	Test Accuracy	Precision	F1 Score
Decision Tree	99.5	94.65	0.9525	0.947
Naive Bayes	68.60	68.68	0.788	0.7804
SVM	78.37	77.80	0.781	0.7772
KNN	98.97	94	0.899	0.877
LR	88.65	85.46	0.82	0.79

Table 6.3: Performance of individual models Comparison Table with Kaggle Dataset-1

Models	Train Accuracy	Test Accuracy	F1 Score	Precision
Decision Tree	100	86.89	0.869	0.8743
Naive Bayes	81.69	78.69	0.7837	0.787
SVM	84.15	76.23	0.8417	0.844
KNN	86.97	78.69	0.786	0.786
LR	77.11	72.95	0.7320	0.7303

Table 6.4: Performance of individual models Comparison Table with Kaggle Dataset-2

Models used	**Accuracy**
Logistics Regression, Decision Tree (After cross validation)	85.50
Logistics Regression, Decision Tree, Naive Bayes and Bagging. (After cross validation)	88.65

Table 6.5: Voting Classifier

Inference:

As it can be seen that Full-Stack ensemble produced higher accuracy than other ensemble methods. In an ensemble of well-performing models, a Full Stack Ensemble model can be used to make predictions for large balanced dataset with greater accuracy than any single model. This is because stacking uses machine learning models to learn to combine the best predictions from contributing ensemble members. Ensemble members in voting are often a diverse collection of model types, such as decision trees, Logistic Regression, SVM, and Random Forest. In the final aggregation in voting, user-specified weights are used to combine the classifiers, but in stacking, a blender/meta classifier is employed to aggregate the data. The goal of Full Stacking is to produce strong models that are less biased than their components (even if variance can also be reduced), whereas bagging is to build an ensemble model with less variance than its components. When compared to small datasets, ensemble approaches produce higher accuracy. As it can be seen in tables 6.1 and 6.2, the ensemble accuracies for small datasets are lower than those for large datasets. This is because single models can produce good accuracy for small datasets, whereas single models cannot produce good accuracy for large datasets, necessitating an ensemble of different Machine Learning models.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this project, a dataset containing large amount data is used which is then random over-sampled to produce a balanced dataset. Many ensemble methods like bagging, Random Forest, stacking and hard voting are used on the dataset to get results of high accuracy. After analyzing all the ensemble methods mentioned above, Full Stack Ensemble has been found to be the most accurate at predicting heart disease with a training accuracy of 97.7% and testing accuracy of 98%.

In an ensemble of well-performing models, a Full Stack Ensemble model can be used to make predictions for large balanced dataset with greater accuracy than any single model. Stacking uses machine learning models to learn to combine the best predictions from contributing ensemble members. An ensemble reduces the spread or dispersion of the predictions and model performance. Ensemble members in voting are often a diverse collection of model types, such as decision trees, Logistic Regression, SVM, and Random Forest. In the final aggregation in voting, user-specified weights are used to combine the classifiers, but in stacking, a blender/meta classifier is employed to aggregate the data. The goal of Full Stacking is to produce strong models that are less biased than their components (even if variance can also be reduced), whereas bagging is to build an ensemble model with less variance than its components.

Boosting ensemble method can be used to train and test the datasets to check whether it enhances the performance and accuracy further. Create a GUI which enables users to input values and predict whether they are likely to have heart disease. This approach of prediction can also be extended to other diseases like diabetics, lung diseases etc.

REFERENCES

[1] Purushottam, Kanak Saxena, Richa Sharma, “Efficient Heart Disease Prediction System”, International Conference on Computing, Communication & Automation (ICCCA), 2016.

[2] Santhana Krishnan J and Geetha S, “Prediction of Heart Disease Using Machine Learning Algorithms”, Conference on Innovations in Information and Communication Technology (ICIICT), 2019.

[3] Abhay Kishore, Ajay Kumar, Karan Singh, Maninder Punia, Yogita Hambir, “Heart Attack Prediction Using Deep Learning”, International Research Journal of Engineering and Technology (IRJET), 2018.

[4] Lakshmana Rao, Sundareswar Pullela and Y Swathi, “Machine Learning Techniques for Heart Disease Prediction”, International Journal of Scientific & Technology Research, 2020.

[5] Sultana, Haider, & Uddin,"Analysis of data mining techniques for heart disease prediction, "International Conference in Electrical Engineering and Information and Communication Technology, 2017.

[6] Mr. Chala Beyene, Prof. Pooja Kamat, “Survey on Prediction and Analysis the Occurrence of Heart Disease Using Data Mining Techniques”, International Journal of Pure and Applied Mathematics, 2018.

[7] Aditi Gavhane, Gouthami Kokkula, Isha Pandya, “Prediction of Heart Disease Using Machine Learning”, International Conference on Electronics, Communication, and Aerospace Technology (ICECA), 2015.

[8] Kumari Deepika & Seema Shedole, “Prediction of chronic disease by mining the data using different Machine Learning Models”, International Conference on Applied and Theoretical Computing and Communication Technology, 2016.