EFFECTIVE HEART DISEASE PREDICTION USING HYBRID MACHINE LEARNING TECHNIQUES

Report submitted to SASTRA Deemed to be University

as the requirement for the course

BCSCCS801: MINI PROJECT

Submitted by

Pabba Rohith

(Reg No. 123003172)

Gumpula Hruthik Kumar

(Reg No.123003072)

Marella Vishnu Vardhan Reddy

(Reg No.123003133)

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA-613401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 OF THE UGC ACT, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA-613401

Bonafide certificate

This is to certify that the thesis titled "**Heart Disease Prediction using Hybrid Machine Learning Techniques"** submitted in partial fulfilment of the requirements for the award of the degree of B.Tech Computer Science and Engineering to the SASTRA Deemed to be University, is a bona-fide record of the work done by **Mr. PABBA ROHITH**(Reg.No. 123003172), **Mr. GUMPULA HRUTHIK KUMAR** (Reg.No.123003072), **Mr. MARELLA VISHNU VARDHAN REDDY**(Reg.No. 123003133) during the final semester of the academic year 2021-22, in the **School of Computing**, under my supervision. This thesis has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to any candidate of any University.

Signature of Project Guide with affiliation: 

Date:28-06-2022

Mini Project Viva-voice held on:

Examiner -I **Examiner-II**

SASTRA

ENGINEERING - MANAGEMENT - LAW - SCIENCES - HUMANITIES - EDUCATION

DEEMED TO BE UNIVERSITY

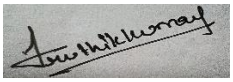
(U/S 3 OF THE UGC ACT, 1956)

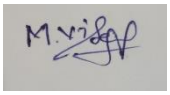
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

DECLARATION

We declare that the thesis titled "**Heart Disease Prediction using Hybrid Machine Learning Techniques"** submitted by us is an original work done by us under the guidance of **Mrs. Sudha N**, **Assistant Professor-III, School of Computing**, **SASTRA Deemed to be University** during the 6th semester of the academic year 2021-2022, in the **School of Computing**. The wok is originaland wherever we have used materials from other sources. We have given due credit and cited them in the text of the thesis. This thesis has not formed the basis for the award of anydegree, diploma, associateship, fellowship or other similar title to any candidate of any University.

Signature of the Candidate(s): 1. 

2. 

3. 

Name of the Candidate(s) : 1. PABBA ROHITH

2.GUMPULA HRUTHIK KUMAR

3. MARELLA VISHNU VARDHAN REDDY

Date : **28-06-2022**

Acknowledgements

We would like to acknowledge each and every one who had a role to play in making our humble efforts an out-to-out success.

We would like to thank **Dr. S. Vaidhyasubramaniam**, **our Honorable Vice - Chancellor** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We find very heartening the encouragement and strategic support offered at every step of our college life by **Dr. A. Umamakeswari, Dean School of Computing, SASTRA Deemed University** and for that we would like to thank them time and again.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram, Associate Dean (Computer Science and Engineering, School of Computing)** and **Dr. Muthaiah. R, Associate Dean (Information Technology, School of Computing)** for the motivation and support offeredby them in materializing this project.

Our **Guide and Mentor Dr. Sudha N, Assistant Professor, School of Computing** was the driving force behind this whole idea from the start.

Last but not the least, I am very much thankful to my parents for guiding me in every step which I took.

List of Figures

List of Tables

Abbreviations

LR - Logistic Regression

NB - Naive Bayes

RF – Random Forest

GBT – Gradient Boosted Trees

DT – Decision Trees

SVM – Support Vector Machine

HRFLM – Hybrid Random Forest using Linear Model

ML – Machine Learning

Abstract

Heart disease has become common cause for death in the modern world. This disease is caused irrespective of age in these days and if not controlled or predicted at the earlier stage then this can cause a great damage and even fatality of the person. There are various kinds of cases that lead to these kinds of diseases. Some of the most commonly caused are high blood pressure where it is the situation in which the blood flows through the heart valves with a great pressure, heart valve disease where it is a situation in which due to some kind of abnormality any one valve out of the four valves couldn't open or close in a right way which leads to blockage of blood or leakage of blood and many other kinds of diseases that leads to heart attack where the heart stops functioning suddenly and the person not even able to breathe which leads to death of the person.

There are many techniques which are used to predict the heart diseases such as Genetic Algorithm, Decision Trees, Naive Bayes and even Neural Networks which gave valuable results and they have also given results of Atrial fibrillation, Normal Sinus Rhythm, Premature Ventricular Contraction which helped us to predict the disease at the earlier stage. Since the disease is related to the heart it is more complex and more sensitive part, so the disease has to be handled very carefully and if the severity is more the treatment has to be done immediately. So, this lead to implement an algorithm that is even more efficient than the above algorithms with such as accuracy etc., parameters.

So, the main objective of the current proposed model is it won't put any restrictions on feature selection. The technique which we applied here is the hybrid model named HRFLM. We have shown the proposed hybrid method showing a stronger capability than the previous ones like the Decision Trees, Naive Bayes etc., in some of the parameters such as accuracy, specificity and in error control.

Keywords: Logistic Regression, Naive Bayes, Decision Tree, Hybrid Random Forest with Linear Model, Specificity.

Table of Contents

1. SUMMARY OF BASE PAPER

Base paper URL: https://ieeexplore.ieee.org/document/8740989

Title of the Base Paper: Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques

Journal: IEEE Access

Page no: 81542-81554

Volume: 7

Year of publication: 2019

METHODOLOGY:

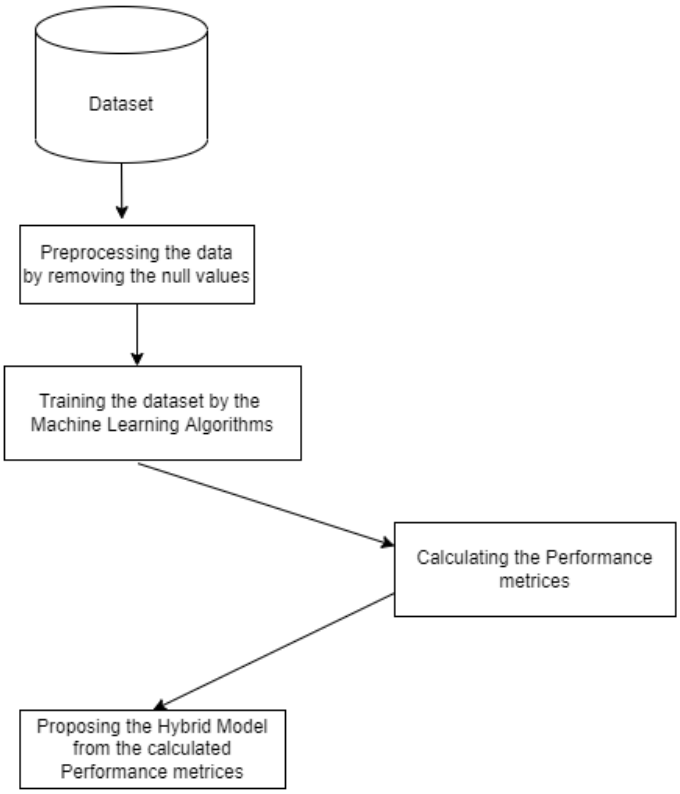


Fig 1.1 Methodology for implementation of the project

The proposed methodology for the Effective Heart Disease Prediction using Hybrid Machine Learning Techniques involves the above given steps. Initially it involves Data collection and preprocessing of the data. We preprocess the collected data by removing the null values or the empty values in the

dataset. After preprocessing the data, we train the machine learning models by using the cleaned dataset.

DATASET COLLECTION AND PREPROCESSING

This dataset is taken from the UCI website. The dataset contained 303 different data out of which 6 data contained the null values. These data are removed from the dataset in the preprocessing stage. To obtain the precise solution, the dataset is made into two data sets namely the Training data set and Testing data set. And we have taken test dataset as 15% from the original dataset.

MACHINE LEARNING TECHNIQUES

After preprocessing the data by applying the above steps, the model is now trained by the following machine learning algorithms with the help of the training dataset and then the trained model is tested using the test data. The ML techniques that are implemented in the current paper are as follows.

1. Logistic Regression

It comes under the category of Supervised learning. In Logistic Regression we use a function called sigmoid function/sigmoid curve. Logistic regression using a set of independent variables finalizes the output of categorical dependent variables. That is, it can be either (yes/no), (0/1), (true or False). It gives the probabilistic values that lie between 0 and 1, instead of giving the exact value as 0 and 1,

2. Naive Bayes

Among Supervised learning algorithms Naïve Bayes classifier is one among them, it works on Bayes theorem which for classification problems solving. It is known as a probabilistic classifier because it finalizes the output based on probability of an object. If the occurrence of a certain feature is independent of the occurrence of other features, then it is called Naive and it is called naive bayes because it is depending on bayes theorem.

3. Random Forest Classifier

Random Forest is one of the Supervised machine learning techniques. Random Forest uses the ensemble technique. It combines multiple models in order to solve complex problems and which in turn increase the performance. In order to increase accuracy for the data set, the classifier takes decision trees of various subsets of the data set. Based on the majority votes of prediction that are taken from each decision tree, it finalizes the result.

4. Support Vector Machine

Among the famous machine learning algorithms support vector machine is machine learning algorithm, it comes under the supervised machine learning technique. In SVM, we use a decision boundary that separates the target variable into two different classes with the best plane. That plane is called Hyperplane. In SVM we use Linear SVM.

5. Decision Tree

From supervised learning techniques Decision tree is one among them,where we use the concept of model trees and we build the trees based on combining two or more features which contribute to the disease prediction and using this we build n number of model trees and predict the result for the given test data from the constructed trees. The depth of the tree can also be mentioned so that we can improve accuracy.

6. Gradient Boosted Trees

In the field of machine learning we all come across errors in predicting the output, the errors may include Bias error or Variance error, so gradient boosting is used to reduce the bias error of the model. Gradient boosting works by constructing simple (weak) predictions models, where each model tries the error left over by the previous model.

7. Voting Classifier

A voting classifier is the one of machine learning algorithm that trains on ensemble of various combination of classifier models and finalizes the output based on the majority of probability. In this it simply collects all the outputs of different classifier models and finalizes the result based on the majority of voting. In this project we have implemented hard voting, in hard voting we will predict the resultant class based on the majority of votes, that means the class which has got more probability of being predicted.

8. Stacking Classifier

Stacking classifier is also a machine learning algorithm that trains on ensemble classification. It is classified as two layers; the 1^{st} layer will consist of all the classification models which are used to finalize the output of given test-data set and 2^{nd} layer is considered as meta classifier takes the outputs of prediction of all the baseline classification as input. Here we give these predicted results to another classifier model so that it generates new prediction.

2. Merits and Demerits of the Base Paper

S. No	Research Paper	Methodology proposed
1.	Using PSO Algorithm for producing the best rules in diagnosis of the heart disease	In this base paper they used Particular Swarm Optimisation method in predicting the heart disease. But the algorithm has low convergence.
2.	Radial Basis function neural network for prediction of cardiac arrhythmias based on heart rate time series	In this base paper they used Radial function in predicting the disease but the classification in this function is low.
3.	Hybrid Intelligent modelling schemes for heart disease classification	In this base paper they used Logistic Regression and ANN but the accuracy is low in ANN.
4.	A computational intelligence method for effective diagnosis of heart disease using genetic algorithm	In this base paper they used Genetic algorithm in the prediction of the heart disease but the efficiency of the algorithm is less.
5.	Diagnosis of heart disease using genetic algorithm based trained recurrent fuzzy neural networks	In this base paper they have used the concept of fuzzy neural networks but they work on inaccurate inputs and the accuracy is too low

The merits that we have achieved from the base paper are we have improved the accuracy of many algorithms namely Naive Bayes, Logistic Regression, Gradient Boosted Trees, Voting classifier and the proposed hybrid model.

We are able to achieve these merits because of the better data pre-processing techniques.

3. SOURCE CODE

PREPROCESSING THE DATA

```
#importing the necessary modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

import os
print(os.listdir())

import warnings
warnings.filterwarnings('ignore')

#removing the null values
heart_data=heart_data.dropna(axis=0)
heart_data["target"]=heart_data["target"].replace([2,3,4],[1,1,1])

#splitting the data into train and test data
from sklearn.model_selection import train_test_split
X = heart_data.drop("target",axis=1)
Y = heart_data["target"]
#The size of the test data is 15% of the original data
X_train,X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=0.15,stratify = Y,random_state=2)
l1=[]
l2=[]
l3=[]
l4=[]
l5=[]
l6=[]
l7=[]
#prints the corresponding size of X_train,Y_train,X_test,Y_test
X_train.shape
Y_train.shape
X_test.shape
Y_test.shape
```

LOGISTIC REGRESSION

```
    #Training the model using the LOGISTIC REGRESSION
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import confusion_matrix
    lr = LogisticRegression()
    lr.fit(X_train,Y_train)
    Y_pred_lr = lr.predict(X_test)
    #printing the confusion matrix
    con_mat=confusion_matrix(Y_test,Y_pred_lr)
    print("Confusion_matrix is\n",con_mat)

#calculating performance metrics namely classification error,
accuracy,precision,recall,f_measure,Sensitivity,Specificity

    classification_error=(con_mat[0,1]+con_mat[1,0])/(sum(sum(con_mat)))
    print("classification_error is",round(classification_error*100,2),"%")

    accuracy=(con_mat[0,0]+con_mat[1,1])/(sum(sum(con_mat)))
    print("accuracy is",round(accuracy*100,2),"%")

    precision=con_mat[1,1]/(con_mat[1,1]+con_mat[0,1])
    print("precision is",round(precision*100,2),"%")

    recall=con_mat[1,1]/(con_mat[1,1]+con_mat[1,0])
    print("recall is",round(recall*100,2),"%")

    f_measure=2*((precision*recall)/(precision+recall))
    print("f_measure is",round(f_measure*100,2),"%")

    Sensitivity=con_mat[0,0]/(con_mat[0,0]+con_mat[0,1])
    print("Sensitivity is",round(Sensitivity*100,2),"%")

    Specificity=con_mat[1,1]/(con_mat[1,0]+con_mat[1,1])
    print("Specificity is",round(Specificity*100,2),"%")
    #appending the values to the corresponding list so that we can build the
    graph
    l2.append(round(accuracy*100,2))
    l3.append(round(precision*100,2))
    l4.append(round(recall*100,2))
    l5.append(round(f_measure*100,2))
    l6.append(round(Sensitivity*100,2))
    l7.append(round(Specificity*100,2))
```

NAIVE BAYES

```
#Training the model using the NAIVE BAYES
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train,Y_train)
Y_pred_nb = nb.predict(X_test)

#printing the confusion matrix

con_mat=confusion_matrix(Y_test,Y_pred_nb)
print("Confusion_matrix is\n",con_mat)

#calculating performance metrics namely classification
error,accuracy,precision,recall,f_measure,Sensitivity,Specificity

classification_error=(con_mat[0,1]+con_mat[1,0])/(sum(sum(con_mat)))
print("classification_error is",round(classification_error*100,2),"%")

accuracy=(con_mat[0,0]+con_mat[1,1])/(sum(sum(con_mat)))
print("accuracy is",round(accuracy*100,2),"%")

precision=con_mat[1,1]/(con_mat[1,1]+con_mat[0,1])
print("precision is",round(precision*100,2),"%")

recall=con_mat[1,1]/(con_mat[1,1]+con_mat[1,0])
print("recall is",round(recall*100,2),"%")

f_measure=2*((precision*recall)/(precision+recall))
print("f_measure is",round(f_measure*100,2),"%")

Sensitivity=con_mat[0,0]/(con_mat[0,0]+con_mat[0,1])
print("Sensitivity is",round(Sensitivity*100,2),"%")

Specificity=con_mat[1,1]/(con_mat[1,0]+con_mat[1,1])
print("Specificity is",round(Specificity*100,2),"%")
l2.append(round(accuracy*100,2))
l3.append(round(precision*100,2))
l4.append(round(recall*100,2))
l5.append(round(f_measure*100,2))
l6.append(round(Sensitivity*100,2))
l7.append(round(Specificity*100,2))
```

RANDOM FOREST

```
#Training the model using RANDOMFOREST
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100,random_state=2) # ,
max_depth=5, random_state=1
model.fit(X_train, Y_train)
Y_pred_rf = model.predict(X_test)

#printing the confusion matrix

con_mat=confusion_matrix(Y_test,Y_pred_rf)
print("Confusion_matrix is\n",con_mat)

#calculating performance metrics namely classification
error,accuracy,precision,recall,f_measure,Sensitivity,Specificity

classification_error=(con_mat[0,1]+con_mat[1,0])/(sum(sum(con_mat)))
print("classification_error is",round(classification_error*100,2),"%")

accuracy=(con_mat[0,0]+con_mat[1,1])/(sum(sum(con_mat)))
print("accuracy is",round(accuracy*100,2),"%")

precision=con_mat[1,1]/(con_mat[1,1]+con_mat[0,1])
print("precision is",round(precision*100,2),"%")

recall=con_mat[1,1]/(con_mat[1,1]+con_mat[1,0])
print("recall is",round(recall*100,2),"%")

f_measure=2*((precision*recall)/(precision+recall))
print("f_measure is",round(f_measure*100,2),"%")

Sensitivity=con_mat[0,0]/(con_mat[0,0]+con_mat[0,1])
print("Sensitivity is",round(Sensitivity*100,2),"%")

Specificity=con_mat[1,1]/(con_mat[1,0]+con_mat[1,1])
print("Specificity is",round(Specificity*100,2),"%")
l2.append(round(accuracy*100,2))
l3.append(round(precision*100,2))
l4.append(round(recall*100,2))
l5.append(round(f_measure*100,2))
l6.append(round(Sensitivity*100,2))
l7.append(round(Specificity*100,2))
```

GRADIENT BOOSTED TREES

```
#Training the model using GRADIENTBOOSTEDTREES
from sklearn.ensemble import GradientBoostingClassifier
gb=GradientBoostingClassifier()
gb.fit(X_train, Y_train)
Y_pred_gb = gb.predict(X_test)

#printing the confusion matrix

con_mat=confusion_matrix(Y_test,Y_pred_gb)
print("Confusion_matrix is\n",con_mat)

#calculating performance metrics namely classification
error,accuracy,precision,recall,f_measure,Sensitivity,Specificity

classification_error=(con_mat[0,1]+con_mat[1,0])/(sum(sum(con_mat)))
print("classification_error is",round(classification_error*100,2),"%")

accuracy=(con_mat[0,0]+con_mat[1,1])/(sum(sum(con_mat)))
print("accuracy is",round(accuracy*100,2),"%")

precision=con_mat[1,1]/(con_mat[1,1]+con_mat[0,1])
print("precision is",round(precision*100,2),"%")

recall=con_mat[1,1]/(con_mat[1,1]+con_mat[1,0])
print("recall is",round(recall*100,2),"%")

f_measure=2*((precision*recall)/(precision+recall))
print("f_measure is",round(f_measure*100,2),"%")

Sensitivity=con_mat[0,0]/(con_mat[0,0]+con_mat[0,1])
print("Sensitivity is",round(Sensitivity*100,2),"%")

Specificity=con_mat[1,1]/(con_mat[1,0]+con_mat[1,1])
print("Specificity is",round(Specificity*100,2),"%")
l2.append(round(accuracy*100,2))
l3.append(round(precision*100,2))
l4.append(round(recall*100,2))
l5.append(round(f_measure*100,2))
l6.append(round(Sensitivity*100,2))
l7.append(round(Specificity*100,2))
```

DECISION TREE

```
#Training the model using DECISION TREE
from sklearn.tree import DecisionTreeClassifier
DTC = DecisionTreeClassifier(criterion =
'entropy',max_depth=4,random_state = 42)
DTC.fit(X_train, Y_train)
Y_pred_DTC = DTC.predict(X_test)

#printing the confusion matrix

con_mat=confusion_matrix(Y_test,Y_pred_DTC)
print("Confusion_matrix is\n",con_mat)

#calculating performance metrics namely classification
error,accuracy,precision,recall,f_measure,Sensitivity,Specificity

classification_error=(con_mat[0,1]+con_mat[1,0])/(sum(sum(con_mat)))
print("classification_error is",round(classification_error*100,2),"%")

accuracy=(con_mat[0,0]+con_mat[1,1])/(sum(sum(con_mat)))
print("accuracy is",round(accuracy*100,2),"%")

precision=con_mat[1,1]/(con_mat[1,1]+con_mat[0,1])
print("precision is",round(precision*100,2),"%")

recall=con_mat[1,1]/(con_mat[1,1]+con_mat[1,0])
print("recall is",round(recall*100,2),"%")

f_measure=2*((precision*recall)/(precision+recall))
print("f_measure is",round(f_measure*100,2),"%")

Sensitivity=con_mat[0,0]/(con_mat[0,0]+con_mat[0,1])
print("Sensitivity is",round(Sensitivity*100,2),"%")

Specificity=con_mat[1,1]/(con_mat[1,0]+con_mat[1,1])
print("Specificity is",round(Specificity*100,2),"%")
l2.append(round(accuracy*100,2))
l3.append(round(precision*100,2))
l4.append(round(recall*100,2))
l5.append(round(f_measure*100,2))
l6.append(round(Sensitivity*100,2))
l7.append(round(Specificity*100,2))
```

SUPPORT VECTOR MACHINE

```
# Training the model using SUPPORT VECTOR MACHINE
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
svc = SVC(kernel ='linear',probability=True,random_state=42)
svc.fit(X_train,Y_train)
Y_pred_svm=svc.predict(X_test)
# printing the confusion matrix

con_mat=confusion_matrix(Y_test,Y_pred_svm)
print("Confusion_matrix is\n",con_mat)

#calculating performance metrics namely classification
error,accuracy,precision,recall,f_measure,Sensitivity,Specificity

classification_error=(con_mat[0,1]+con_mat[1,0])/(sum(sum(con_mat)))
print("classification_error is",round(classification_error*100,2),"%")

accuracy=(con_mat[0,0]+con_mat[1,1])/(sum(sum(con_mat)))
print("accuracy is",round(accuracy*100,2),"%")

precision=con_mat[1,1]/(con_mat[1,1]+con_mat[0,1])
print("precision is",round(precision*100,2),"%")

recall=con_mat[1,1]/(con_mat[1,1]+con_mat[1,0])
print("recall is",round(recall*100,2),"%")

f_measure=2*((precision*recall)/(precision+recall))
print("f_measure is",round(f_measure*100,2),"%")

Sensitivity=con_mat[0,0]/(con_mat[0,0]+con_mat[0,1])
print("Sensitivity is",round(Sensitivity*100,2),"%")

Specificity=con_mat[1,1]/(con_mat[1,0]+con_mat[1,1])
print("Specificity is",round(Specificity*100,2),"%")
l2.append(round(accuracy*100,2))
l3.append(round(precision*100,2))
l4.append(round(recall*100,2))
l5.append(round(f_measure*100,2))
l6.append(round(Sensitivity*100,2))
l7.append(round(Specificity*100,2))
```

VOTING CLASSIFIER

```
# Training the model using VOTE CLASSIFIER
from sklearn.ensemble import VotingClassifier
eclf1 = VotingClassifier(estimators=[('Logistic Regression', lr),('Random
Forest', model),('Decision Tree',DTC)], voting='hard')
eclf1.fit(X_train, Y_train)
Y_pred_v = eclf1.predict(X_test)
#printing the confusion matrix

con_mat=confusion_matrix(Y_test,Y_pred_v)
print("Confusion_matrix is\n",con_mat)

#calculating performance metrics namely classification
error,accuracy,precision,recall,f_measure,Sensitivity,Specificity

classification_error=(con_mat[0,1]+con_mat[1,0])/(sum(sum(con_mat)))
print("classification_error is",round(classification_error*100,2),"%")

accuracy=(con_mat[0,0]+con_mat[1,1])/(sum(sum(con_mat)))
print("accuracy is",round(accuracy*100,2),"%")

precision=con_mat[1,1]/(con_mat[1,1]+con_mat[0,1])
print("precision is",round(precision*100,2),"%")

recall=con_mat[1,1]/(con_mat[1,1]+con_mat[1,0])
print("recall is",round(recall*100,2),"%")

f_measure=2*((precision*recall)/(precision+recall))
print("f_measure is",round(f_measure*100,2),"%")

Sensitivity=con_mat[0,0]/(con_mat[0,0]+con_mat[0,1])
print("Sensitivity is",round(Sensitivity*100,2),"%")

Specificity=con_mat[1,1]/(con_mat[1,0]+con_mat[1,1])
print("Specificity is",round(Specificity*100,2),"%")
```

HRFLM (HYBRID RANDOM FOREST USING LINEAR MODEL)

```
#Training the model using Hybrid Machine Learning Algorithms (HRFLM)

from sklearn.ensemble import StackingClassifier
estimators=[('Logistic Regression', lr),('Random Forest',
model),('Decision Tree',DTC)]
cf=StackingClassifier(estimators=estimators,final_estimator=RandomForestCl
assifier(n_estimators=100,max_depth=4,random_state=2));
cf.fit(X_train,Y_train)
Y_pred_h = cf.predict(X_test)
#printing the confusion matrix

con_mat=confusion_matrix(Y_test,Y_pred_h)
print("Confusion_matrix is\n",con_mat)

#calculating performance metrics namely classification
error,accuracy,precision,recall,f_measure,Sensitivity,Specificity

classification_error=(con_mat[0,1]+con_mat[1,0])/(sum(sum(con_mat)))
print("classification_error is",round(classification_error*100,2),"%")

accuracy=(con_mat[0,0]+con_mat[1,1])/(sum(sum(con_mat)))
print("accuracy is",round(accuracy*100,2),"%")

precision=con_mat[1,1]/(con_mat[1,1]+con_mat[0,1])
print("precision is",round(precision*100,2),"%")

recall=con_mat[1,1]/(con_mat[1,1]+con_mat[1,0])
print("recall is",round(recall*100,2),"%")

f_measure=2*((precision*recall)/(precision+recall))
print("f_measure is",round(f_measure*100,2),"%")

Sensitivity=con_mat[0,0]/(con_mat[0,0]+con_mat[0,1])
print("Sensitivity is",round(Sensitivity*100,2),"%")
Specificity=con_mat[1,1]/(con_mat[1,0]+con_mat[1,1])
print("Specificity is",round(Specificity*100,2),"%")
l2.append(round(accuracy*100,2))
l3.append(round(precision*100,2))
l4.append(round(recall*100,2))
l5.append(round(f_measure*100,2))
l6.append(round(Sensitivity*100,2))
l7.append(round(Specificity*100,2))
```

4. SNAPSHOTS

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0

Fig 4.1 Heart data before pre-processing

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          4
thal        2
target      0
dtype: int64
```

Fig 4.2 Features that containing the null values

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	1
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
297	57	0	4	140	241	0	0	123	1	0.2	2	0.0	7.0	1
298	45	1	1	110	264	0	0	132	0	1.2	2	0.0	7.0	1
299	68	1	4	144	193	1	0	141	0	3.4	2	2.0	7.0	1
300	57	1	4	130	131	0	0	115	1	1.2	2	1.0	7.0	1
301	57	0	2	130	236	0	2	174	0	0.0	2	1.0	3.0	1

297 rows × 14 columns

Fig 4.3 Heart data after the pre-processing

```
Confusion_matrix is
 [[21  3]
 [ 1 20]]
classification_error is 8.89 %
accuracy is 91.11 %
precision is 86.96 %
recall is 95.24 %
f_measure is 90.91 %
Sensitivity is 87.5 %
Specificity is 95.24 %
```

```
Confusion_matrix is
 [[28  4]
 [ 4 24]]
classification_error is 13.33 %
accuracy is 86.67 %
precision is 85.71 %
recall is 85.71 %
f_measure is 85.71 %
Sensitivity is 87.5 %
Specificity is 85.71 %
```

Fig 4.4 Confusion matrix and performance metrices of Logistic Regression and Naive Bayes

```
Confusion_matrix is
 [[20  4]
 [ 1 20]]
classification_error is 11.11 %
accuracy is 88.89 %
precision is 83.33 %
recall is 95.24 %
f_measure is 88.89 %
Sensitivity is 83.33 %
Specificity is 95.24 %
```

```
Confusion_matrix is
 [[19  5]
 [ 2 19]]
classification_error is 15.56 %
accuracy is 84.44 %
precision is 79.17 %
recall is 90.48 %
f_measure is 84.44 %
Sensitivity is 79.17 %
Specificity is 90.48 %
```

Fig 4.5 Confusion matrix and performance metrices of SVM and Random Forest

```
Confusion_matrix is
 [[22  2]
 [ 6 15]]
classification_error is 17.78 %
accuracy is 82.22 %
precision is 88.24 %
recall is 71.43 %
f_measure is 78.95 %
Sensitivity is 91.67 %
Specificity is 71.43 %
```

```
Confusion_matrix is
 [[20  4]
 [ 2 19]]
classification_error is 13.33 %
accuracy is 86.67 %
precision is 82.61 %
recall is 90.48 %
f_measure is 86.36 %
Sensitivity is 83.33 %
Specificity is 90.48 %
```

Fig 4.6 Confusion matrix and performance metrices of DT and GBT

```
Confusion_matrix is
 [[21  3]
 [ 2 19]]
classification_error is 11.11 %
accuracy is 88.89 %
precision is 86.36 %
recall is 90.48 %
f_measure is 88.37 %
Sensitivity is 87.5 %
Specificity is 90.48 %
```

ıce metrices of

```
Confusion_matrix is
 [[22  2]
 [ 1 20]]
classification_error is 6.67 %
accuracy is 93.33 %
precision is 90.91 %
recall is 95.24 %
f_measure is 93.02 %
Sensitivity is 91.67 %
Specificity is 95.24 %
```

ALGORITHM NAME	ACCURACY	PRECISION	RECALL	F-MEASURE	SENSITIVITY	SPECIFICITY
LR	91.11	86.96	95.24	90.91	87.5	95.24
NB	86.67	85.71	85.71	85.71	87.5	85.71
RF	84.44	79.17	90.48	84.44	79.17	90.48
GBT	86.67	82.61	90.48	86.36	83.33	90.48
DT	82.22	88.24	71.43	78.95	91.67	71.43
SVM	88.89	83.33	95.24	88.89	83.33	95.24
VOTE	88.89	86.36	90.48	88.37	87.5	90.48
HRFLM	93.33	90.91	95.24	93.02	91.67	95.24

Fig 4.8 Table representing the performance of machine learning models with the proposed model HRFLM

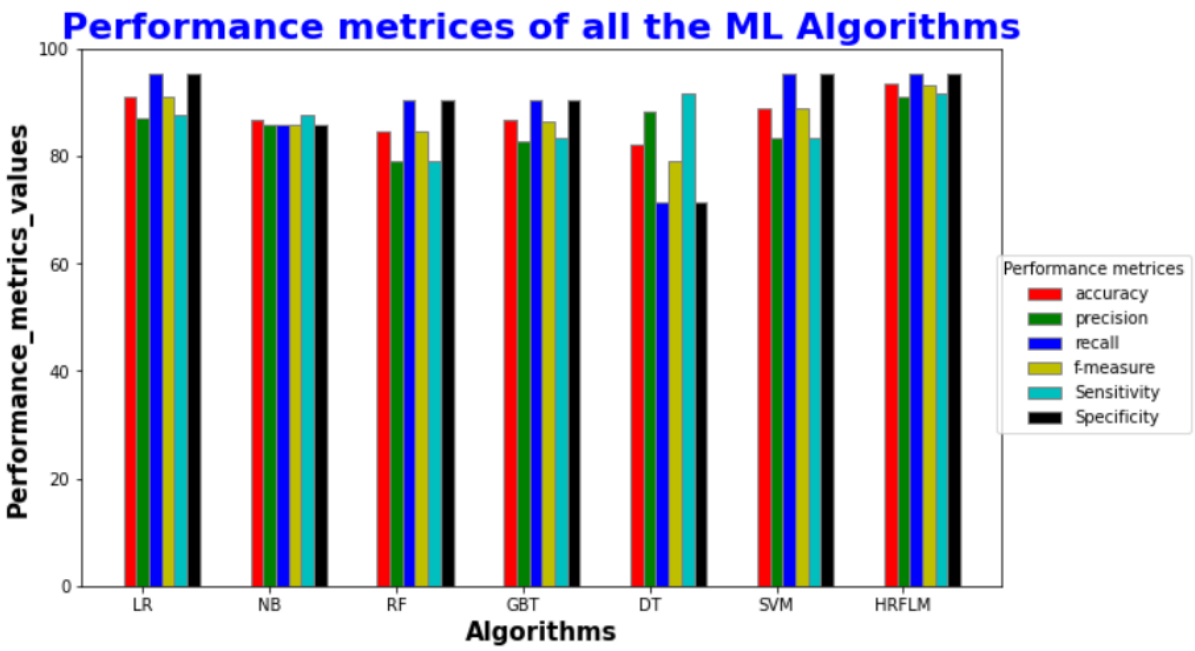


Fig 4.9 Graph Indicating the performance of the machine learning models with the proposed HRFLM

Heart Disease Prediction using HRFLM

Enter the age

63.00 − +

Enter gender

1.00 − +

Enter cp

4.00 − +

Fig 4.10 GUI showing the data to be entered to predict the heart disease using HRFLM

1.40 − +

Enter slope

2.00 − +

Enter ca

1.00 − +

Enter thal

7.00 − +

Heart disease result

heart disease

Fig 4.11 GUI showing the output for the entered data

5. CONCLUSION AND FUTURE PLANS

The proposed framework illustrates about the Heart disease prediction using the Hybrid Machine Learning Techniques. In this framework, initially we pre-processed the data and then we used the ML classifier algorithms and trained the model with this pre-processed dataset. The ML classifier algorithms that we used in this framework are Logistic Regression, Naive Bayes, Random Forest, Gradient boosted trees, Decision Tree, Support Vector Machine and Voting Classifier.

From the above ML classifier algorithms, we calculated the performance metrices of each model and from the results, we proposed a Hybrid model using the Logistic Regression, Random Forest and Decision Tree. After proposing the Hybrid model the calculated experimental results show that the accuracy has improved a lot when compared to that of the normal ML classifier algorithms. The proposed HRFLM framework has achieved an accuracy of 93.33%. We have also used other performance metrics in this framework namely precision, recall, F-measure, Sensitivity and Specificity.

Later, this proposed Hybrid model can be tested on large datasets and we can also combine various other algorithms in this hybrid model like the ConvolutionalNeuralNetworks (CNN) in order to implement an algorithm and to get the model with the less classification error rate and also with increased specificity and sensitivity.

6. REFERENCES

- A. S. Abdullah and R. R. Rajalaxmi, ‘‘A data mining model for predicting the coronary heart disease using random forest classifier,’’ in Proc. Int. Conf. Recent Trends Comput. Methods, Commun. Controls, Apr. 2012, pp. 22–25.
- C. A. Devi, S. P. Rajamhoana, K. Umamaheswari, R. Kiruba, K. Karunya, and R. Deepika, ‘‘Analysis of neural networks-based heart disease prediction system,’’ in Proc. 11th Int. Conf. Hum. Syst. Interact. (HSI), Gdansk, Poland, Jul. 2018, pp. 233–239.
- H. A. Esfahani and M. Ghazanfari, ‘‘Cardiovascular disease detection using a new ensemble classifier,’’ in Proc. IEEE 4th Int. Conf. Knowl.- Based Eng. Innov. (KBEI), Dec. 2017, pp. 1011–1014.
- F. Dammak, L. Baccour, and A. M. Alimi, ‘‘The impact of criterion weights techniques in TOPSIS method of multi-criteria decision making in crisp and intuitionistic fuzzy domains,’’ in Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE), vol. 9, Aug. 2015, pp. 1–8.
- A. Gavhane, G. Kokkula, I. Pandya, and K. Devadkar, ‘‘Prediction of heart disease using machine learning,’’ in Proc. 2nd Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA), Mar. 2018, pp. 1275–1278.