Heart Disease Prediction

Chao Cao

12/19/2022

Introduction

As known to all, heart disease is becoming more common in the present world. People often feel depressed and anxious because they are afraid of catching heart disease without being aware of it. Heart disease can be caused by many different variables. So is there any way for people to detect it simply? Thanks to current advanced machine learning techniques and better databases, we can build some models/classifiers via training enough data to improve accuracy and performance and then make predictions based on limited information. In this project, I aim to explore a heart disease dataset, build various machine learning models that can predict whether patients have heart disease or not, and perform predictions on new example data given.

Background

Raw Data

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is an integer value from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (value 1,2,3,4) from absence (value 0). One file has been "processed", that one containing the Cleveland database. All four unprocessed files also exist in this directory. Though I combined all four datasets at first, this project still focuses on the Cleveland database.

Data Description

There are 14 variables in the dataset, of which 9 categorical variables and 5 continuous variables.

Here is the basic structure of the dataset:

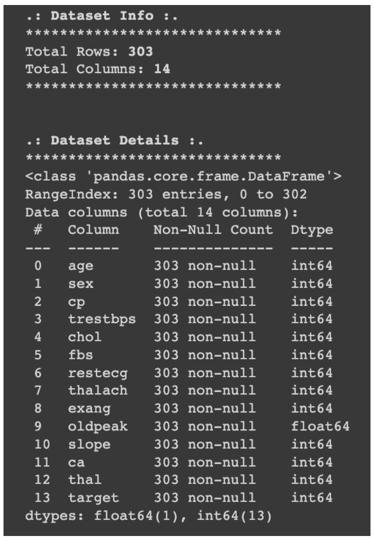
Variable Name	**Description**	**Sample Data**
Age	Patient Age (in years)	63; 37; ...
Sex	Gender of patient (0 = male; 1 = female)	1; 0; ...
cp	Chest pain type (4 values: 0, 1, 2, 3)	3; 1; 2; ...
trestbps	resting blood pressure (in mm Hg)	145; 130; ...

chol	Serum cholestoral (in mg/dl)	233; 250; ...
fbs	Fasting blood sugar > 120 mg/dl (1 = true; 0 = false)	1; 0; ...
restecg	Resting electrocardiographic results (values 0, 1, 2)	0; 1; ...
thalach	Maximum heart rate achieved	150; 187; ...
exang	Exercise induced angina (1 = yes; 0 = no)	1; 0; ...
oldpeak	ST depression induced by exercise relative to rest	2.3; 3.5; ...
slope	The slope of the peak exercise ST segment (values 0, 1, 2)	0; 2; ...
ca	number of major vessels (0-4) colored by flourosopy	0; 3; ...
thal	(3 = normal; 6 = fixed defect; 7 = reversable defect)	1; 3; ...
target	Target column (1 = Yes; 0 = No)	1; 0; ...

Data Processing

1. Data Overview

In the dataset, there are 14 columns with 303 observations. Also, there are no null values in this dataset.


```
.: Dataset Info :.
******************************
Total Rows: 303
Total Columns: 14
******************************


.: Dataset Details :.
******************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
```

2. Fix datatypes

The data types for some columns are not matched so I fixed the data types for these columns by converting them from (‘sex’, ‘cp’, ‘fbs’, ‘exang’, ‘slope’, ‘ca’, ‘thal’)int to object.

3. One-hot encoding

I use the one-hot encoding technique to transform categorical variables.

Categorical data are variables that contain label values rather than numeric values. The number of possible values is often limited to a fixed set. Categorical variables are often called nominal. Though some algorithms can work with categorical data directly, many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric. In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves. This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back into a categorical form in order to present them or use them in some application.

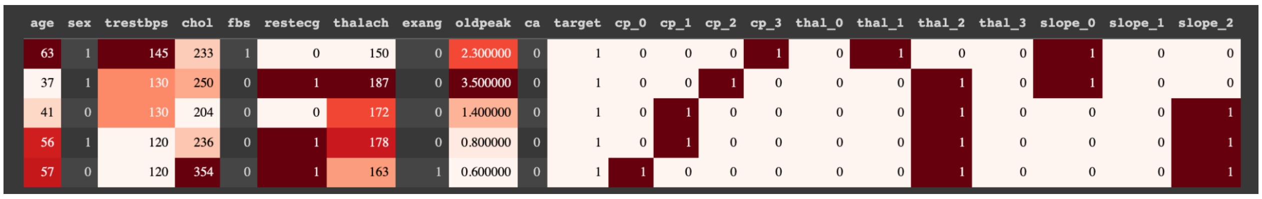
The integer encoding is not enough for categorical variables where no such ordinal relationship exists. In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value. The binary variables are often called “dummy variables” in other fields, such as statistics.

After creating dummy variables for ‘cp’, ‘thal’, and ‘slope’, we can merge dummy variables to the main data frame and drop unnecessary variables, which are the original ‘cp’, ‘thal’, and ‘slope’ columns.

4. Processed Data

The processed data is saved as ‘processed_data.csv’.



age	sex	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	ca	target	cp_0	cp_1	cp_2	cp_3	thal_0	thal_1	thal_2	thal_3	slope_0	slope_1	slope_2
63	1	145	233	1	0	150	0	2.300000	0	1	0	0	0	1	0	1	0	0	1	0	0
37	1	130	250	0	1	187	0	3.500000	0	1	0	0	1	0	0	0	1	0	1	0	0
41	0	130	204	0	0	172	0	1.400000	0	1	0	1	0	0	0	0	1	0	0	0	1
56	1	120	236	0	1	178	0	0.800000	0	1	0	1	0	0	0	0	1	0	0	0	1
57	0	120	354	0	1	163	1	0.600000	0	1	1	0	0	0	0	0	1	0	0	0	1

Model Implementation

Data pre-processing

1. Feature Separating

The 'target' column means the presence of the patients. I separate it as the target, and other columns as variables.

2. Data Normalization

I used the Min-max method to normalize the data. Min-max normalization (usually called feature scaling) performs a linear transformation on the original data. This technique gets all the scaled data in the range (0, 1). Min-max normalization can preserve the relationships among the original data values. Though it can't handle and is sensitive to outliers, it works well on this dataset.

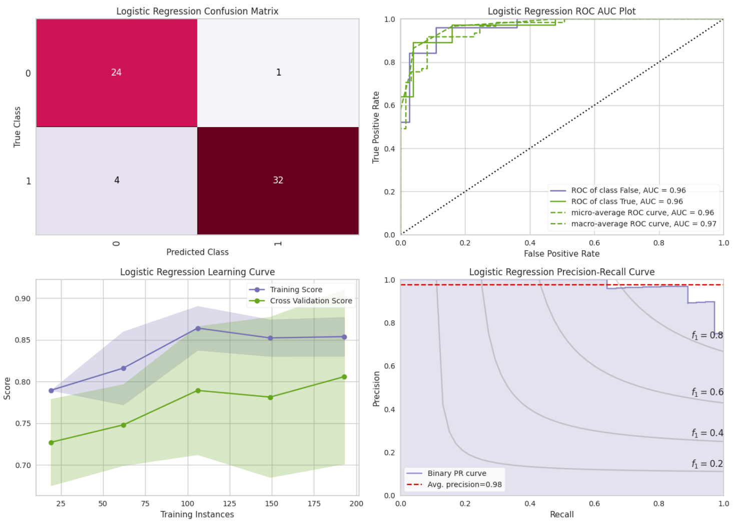
3. Splitting Dataset,

As usual, I split the dataset into 80% for training dataset and 20% for testing dataset.

Models

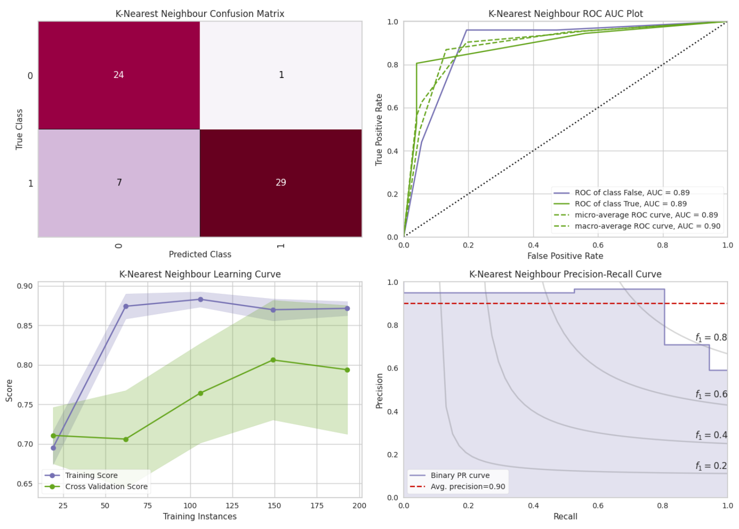
1. Logistic Regression (LR)

LRclassifier = LogisticRegression(max_iter=1000, random_state=1, solver='liblinear', penalty='l1')



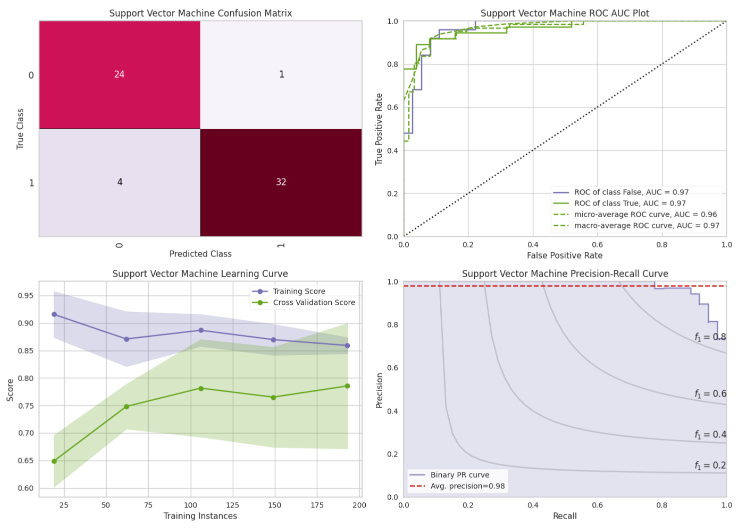
2. K-Nearest Neighbour (KNN)

KNNClassifier = KNeighborsClassifier(n_neighbors=3)



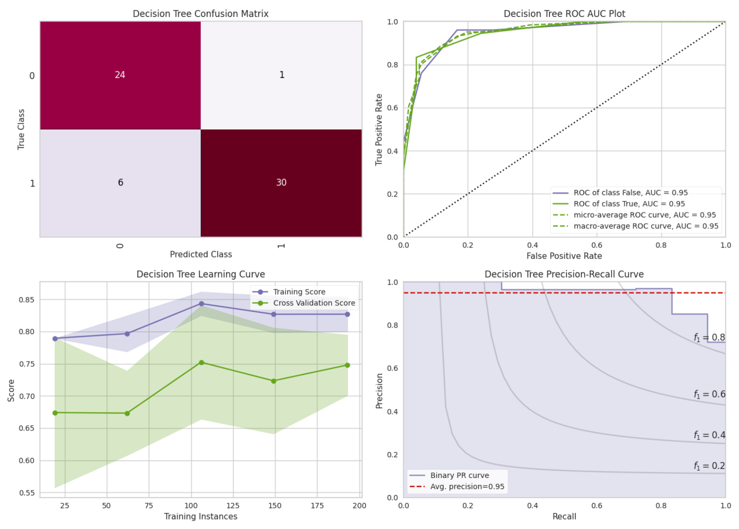
3. Support Vector Machine (SVM)

SVMclassifier = SVC(kernel='linear', max_iter=1000, C=10, probability=True)



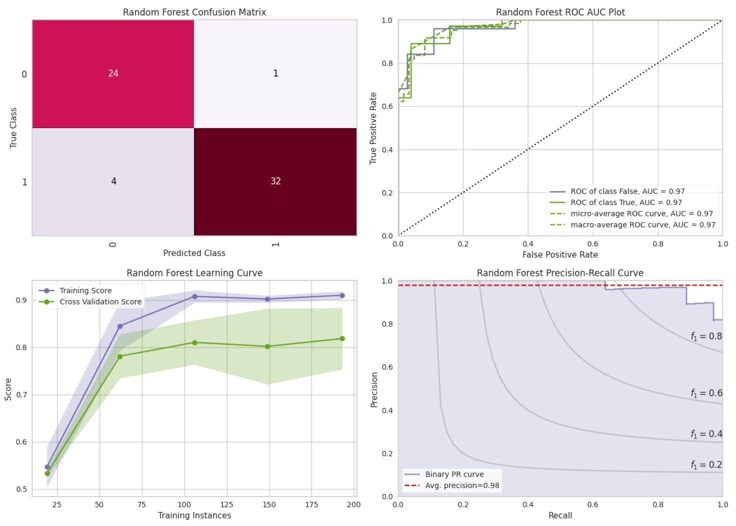
4. Decision Tree (DT)

DTCclassifier = DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, criterion='entropy', min_samples_split=5, splitter='random', random_state=1)



5. Random Forest (RF)

RFclassifier = RandomForestClassifier(n_estimators=1000, random_state=1, max_leaf_nodes=20, min_samples_split=15)



Evaluations

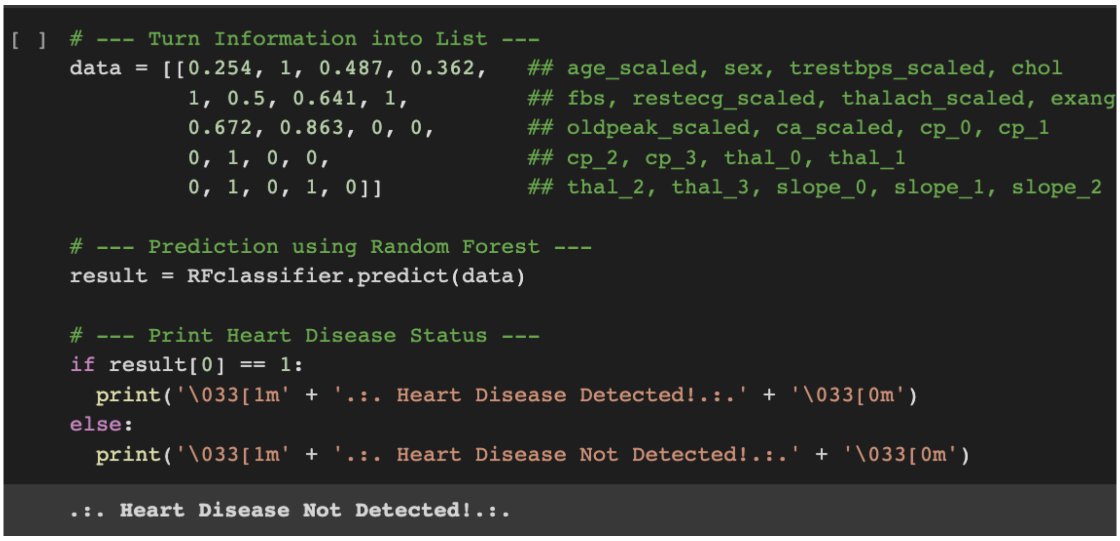
Model	**Accuracy**
Logistic Regression	91.803279
Support Vector Machine	91.803279
Random Forest	91.803279
Decision Tree	88.524590
K-Nearest Neighbour	86.885246

Testing

Here we set a new virtual patient with below information

age (scaled)	: 0.254
sex	: Female (1)
trestbps (scaled)	: 0.487
chol (scaled)	: 0.362
fbs	: > 120 mg/dl (1)
restecg (scaled)	: 0.5
thalach (scaled)	: 0.641
exang	: 1
oldpeak (scaled)	: 0.672
ca (scaled)	: 0.863
cp	: 3
thal	: 3
slope	: 1

And I get the prediction result that he doesn't get heart disease.



```
[ ] # --- Turn Information into List ---
    data = [[0.254, 1, 0.487, 0.362,    ## age_scaled, sex, trestbps_scaled, chol
             1, 0.5, 0.641, 1,          ## fbs, restecg_scaled, thalach_scaled, exang
             0.672, 0.863, 0, 0,        ## oldpeak_scaled, ca_scaled, cp_0, cp_1
             0, 1, 0, 0,                ## cp_2, cp_3, thal_0, thal_1
             0, 1, 0, 1, 0]]            ## thal_2, thal_3, slope_0, slope_1, slope_2

    # --- Prediction using Random Forest ---
    result = RFclassifier.predict(data)

    # --- Print Heart Disease Status ---
    if result[0] == 1:
      print('\033[1m' + '.:. Heart Disease Detected!.:.' + '\033[0m')
    else:
      print('\033[1m' + '.:. Heart Disease Not Detected!.:.' + '\033[0m')

    .:. Heart Disease Not Detected!.:.
```

Insights

In this project, I used five machine learning models to predict whether someone has got heart disease. Five models all perform well. Among them, Logistic Regression, Support Vector Machine, and Random Forest even achieve the same accuracy 91.803279%. The test has proved they are valid.

Future & Challenges

I think that the accuracy can be improved in the future.

1. There are more models and methods, like AdaBoost, and Gradient Boosting. They may be more suitable to the data and perform better.
2. The data is old but valuable anyway. The shortcoming of this project is that there are only 303 rows of patient records. Though they are accurate, which is most important, it is not ideal for training models. It will be better if we can make use of a better dataset.
3. The raw dataset has missing data, including Nan and 0, which is hard to handle. As this is a medical prediction, I have no choice but to drop all rows with missing data. In the future, I can have a better way to deal with missing data.

Appendix

1. Heart Disease Data Set

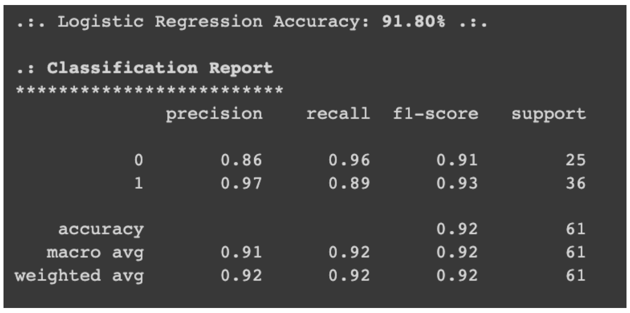
https://archive.ics.uci.edu/ml/datasets/heart+Disease

2. Data Attributes

Only 14 attributes used:
1. #3 (age)
2. #4 (sex)
3. #9 (cp)
4. #10 (trestbps)
5. #12 (chol)
6. #16 (fbs)
7. #19 (restecg)
8. #32 (thalach)
9. #38 (exang)
10. #40 (oldpeak)
11. #41 (slope)
12. #44 (ca)
13. #51 (thal)
14. #58 (num) (the predicted attribute)

3. Model Performance

1. Logistic Regression (LR)



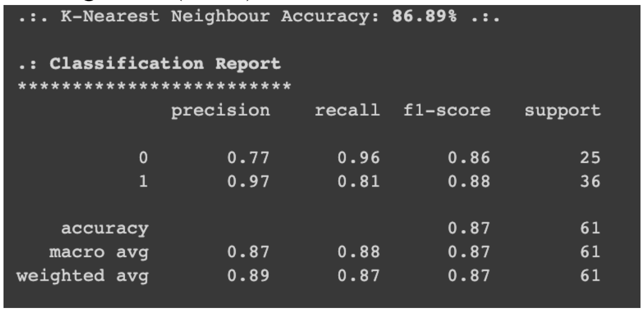
```
.:. Logistic Regression Accuracy: 91.80% .:.

.: Classification Report
*************************
              precision    recall  f1-score   support

           0       0.86      0.96      0.91        25
           1       0.97      0.89      0.93        36

    accuracy                           0.92        61
   macro avg       0.91      0.92      0.92        61
weighted avg       0.92      0.92      0.92        61
```

2. K-Nearest Neighbour (KNN)



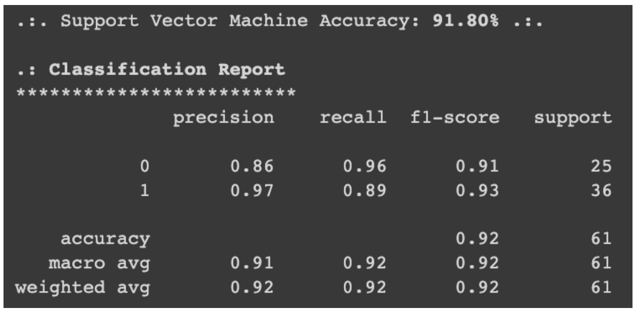
```
.:. K-Nearest Neighbour Accuracy: 86.89% .:.

.: Classification Report
*************************
              precision    recall  f1-score   support

           0       0.77      0.96      0.86        25
           1       0.97      0.81      0.88        36

    accuracy                           0.87        61
   macro avg       0.87      0.88      0.87        61
weighted avg       0.89      0.87      0.87        61
```

3. Support Vector Machine (SVM)



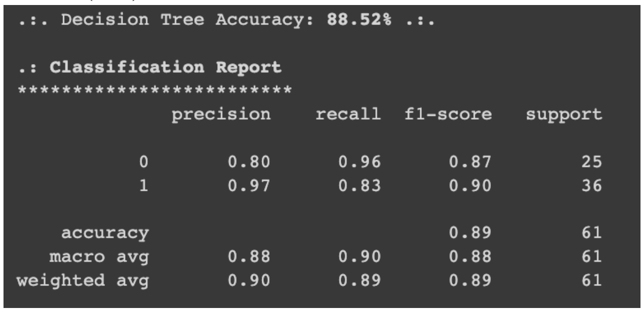
```
.:. Support Vector Machine Accuracy: 91.80% .:.

.: Classification Report
*************************
              precision    recall  f1-score   support

           0       0.86      0.96      0.91        25
           1       0.97      0.89      0.93        36

    accuracy                           0.92        61
   macro avg       0.91      0.92      0.92        61
weighted avg       0.92      0.92      0.92        61
```

4. Decision Tree (DT)



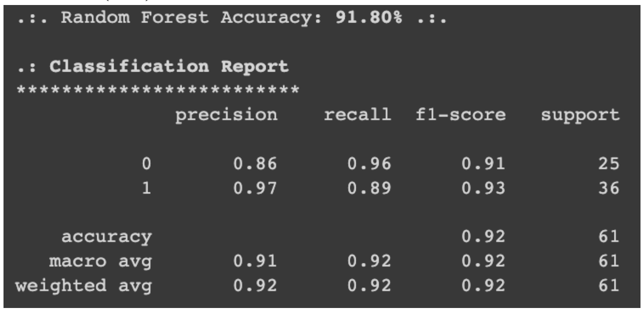
```
.:. Decision Tree Accuracy: 88.52% .:.

.: Classification Report
*************************
              precision    recall  f1-score   support

           0       0.80      0.96      0.87        25
           1       0.97      0.83      0.90        36

    accuracy                           0.89        61
   macro avg       0.88      0.90      0.88        61
weighted avg       0.90      0.89      0.89        61
```

5. Random Forest (RF)



```
.:. Random Forest Accuracy: 91.80% .:.

.: Classification Report
*************************
              precision    recall  f1-score   support

           0       0.86      0.96      0.91        25
           1       0.97      0.89      0.93        36

    accuracy                           0.92        61
   macro avg       0.91      0.92      0.92        61
weighted avg       0.92      0.92      0.92        61
```

4. Code Repo

https://github.com/HenryVarro666/Heart_Disease_Prediction