# Heart Disease and Diabetes Diagnosis Model for Smart Health Care System

*Submitted in partial fulfilment of the requirements for the degree of*

**Bachelor of Technology**

in

**Computer Science and Engineering**

*By*

**Robin Sah (18BCE2462)**
**Sunil Kumar Sah (18BCE2470)**
**Kismat Khatri (18BCE2473)**

**Under the Guidance of**
**Prof/Dr. Vishnu Srinivasa Murthy Y**
**SCOPE**
**VIT, Vellore**


**May, 2022**

# DECLARATION

We hereby declare that the thesis entitled "**Heart Disease and Diabetes Diagnosis Model for Smart Healthcare Systems**" submitted by us, for the award of the degree of Bachelor of Technology in Computer Science and Engineering to VIT is a record of bonfide work carried out by us under the supervision of Guide **Dr. Vishnu Srinivasa Murthy Y**.

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or credential from this or any other college or university.

Place: Vellore

Date: 31st May 2022

**Signature of the Candidate**

Robin Sah (18BCE2462)
Robin.sah2018@vitstudent.ac.in

Sunil Kumar Sah (18BCE2470)
Sunilkumar.sah2018@vitstudent.ac.in

Kismat Khatri (18BCE2473)
Kismat.khatri2018@vitstudent.ac.in

# CERTIFICATE

This is to certify that the thesis entitled "**Heart Disease and Diabetes Diagnosis Model for Smart Healthcare Systems**" submitted by Sunil Kumar Sah (18BCE2470), Robin Sah (18BCE2462) & Kismat Khatri (18BCE2473), SCOPE, VIT University, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by them under my supervision during the period, 01. 12. 2021 to 30.05.2022, as per the VIT code of academic and research ethics. This report's contents have not been and will not be submitted for the granting of any other degree or diploma in this institute or any other institute or university, in part or in whole. The thesis complies with the University's rules and regulations, and in my opinion, meets the necessary standards for submission.

Place: Vellore

Date: 31st May 2022

**Signature of the Guide**

Prof./Dr. Vishnu Srinivasa Murthy Y.
Vishnu.murthy@vit.ac.in

**Internal Examiner** **External Examiner**

**Head of the Department**

**Computer Science and Engineering**

# ACKNOWLEDGEMENTS

We would like to express our special thank of gratitude to our Faculty Dr. Vishnu Srinivasa Murthy Y and our HOD who gave us the golden opportunity to do this project on this topic "**Heart Disease and Diabetes Diagnosis Model for Smart Healthcare Systems**". It helped us in doing a lot of research and we came to know about a lot of things related to this topic.

Finally, we would also like to thank my Proctor Dr. Rajarajan G and friends who helped me a lot in finalizing this project within the limited time frame and for successfully competition.

Robin Sah 18BCE2462

Sunil Kumar Sah 18BCE2470

Kismat Khatri 18BCE2473

# Executive Summary

Modern health care events are considered to include cellular internet, microelectronics, 5G internet system, big data analytics, cloud based computing and Artificial Intelligence (AI), and smart biotechnology. At all levels of modern health care, these procedures are applied. Patients can check their health using wearable or portable devices as needed. They can get health care advice with practical help and run their own homes using remote resources.

In addition, easy-to-use, efficient, and versatile software is considered part of advanced healthcare. Clinical model adjustments (from disease support to patient-based care), changes in cognitive development (from clinical data to regional clinical data), extensions to clinical management (from general management to personal management), and adjustments to prevention and treatment with only a few changes (focus shifts) from disease treatment to the preventive medical system).

As a result, the following reforms focus on addressing people's basic needs in order to improve healthcare capacity, which expands health care knowledge and paves the way for the distribution of intelligent medicine in the future. Advanced medical services are made up of doctors, patients, medical and research institutions, and other stakeholders. All aspects of disease prevention and monitoring, prognosis and treatment, clinical care, health decision-making, and clinical studies should be considered.

**CONTENTS** **Page No.**

## List of Figures



# 1. INTRODUCTION

### 1.1. OBJECTIVE

For an intelligent health care system, design and develop an AI and IoT-based diagnostic model. Suggest a CSO-CLSTM model for the diagnosis of diabetes and heart disease. Improve category performance by integrating an outlier identification process based on forestry expertise. Using the CSO algorithm, tune the LSTM model parameters. Verify the performance of the CSO-LSTM model using two benchmark datasets.

### 1.2 MOTIVATION

In light of this, our research introduces a new diagnostic model based on AI and IoT for intelligent health care systems. The mail goal of our project is to use AI and IoT integration techniques to develop a diagnostic model for heart disease and diabetes. Data collection, pre-processing, classification, and parameter tuning are all phases of the defined model. IoT devices, such as wearable devices and sensors, allow seamless data collection, while AI methods use information to diagnose diseases.

### 1.3 BACKGROUND

The healthcare industry has begun to use information technology to create new applications and improve diagnostic and therapeutic procedures in recent years. Advanced techniques and scientific theories are the main sources of large amounts of digital data. Then there are advanced clinical applications, which are the result of recent advances in information technology.

In addition, easy-to-use, efficient, and versatile software is considered part of advanced healthcare. Clinical model adjustments (from disease support to patient-based care), changes

in cognitive development (from clinical data to regional clinical data), extensions to clinical management (from general management to personal management), and adjustments to prevention and treatment with only a few changes (focus shifts) from disease treatment to the preventive medical system).

As a result, the following reforms focus on addressing people's basic needs in order to improve healthcare capacity, which expands health care knowledge and paves the way for the distribution of intelligent medicine in the future. Advanced medical services are made up of doctors, patients, medical and research institutions, and other stakeholders. All aspects of disease prevention and monitoring, prognosis and treatment, clinical care, health decision-making, and clinical studies should be considered.

Modern health care events are considered to include cellular internet, microelectronics, 5G internet system, big data analytics, cloud based computing and Artificial Intelligence (AI), and smart biotechnology. At all levels of modern health care, these procedures are applied. Patients can check their health using wearable or portable devices as needed. They can get health care advice with practical help and run their own homes using remote resources.

## 2. PROJECT DESCRIPTIONS AND GOALS

The results were compared with existing classification models such as k-Nearest Neighbours and classification or regression to determine their accuracy. Alternatively, the study used several models similar to those used by Azmi et al. confusing separation in ECG data. IBM has devised a Monitor-analyse-Plan-Execute plus Knowledge (MAPE-K) method, separate from the Hierarchical Computing Architecture for Healthcare, to streamline the process between three stages called fog, fog, and clouds. An CNN-based automatic detection model for ENG is presented in the literature. Use 1D and 2D convolutions to separate temporary and spatial information.

Advanced healthcare is thought to have applications that are easy, elegant, and multi-tasking. The extensions in clinical model which includes the priority from disease-based to patient-based care, changes in the development of information where normal medical data is made available as regional medical data, extensions in the management of the clinic promoting general management to personal management. Also, modifications in prevention and treatment shifting of focus from treatment of the disease to preventive medical system all are accompanied in these changes.

## 3. TECHNICAL SPECIFECATIONS

### 3.1 System Requirements

#### 3.1.1 Hardware Requirements

The presented model will be implemented in a PC with specifications such as

Motherboard - NB. Q3L11.001,

Processor – i7-8750H,

Graphics Card – Nvidia GTX GeForce 1050Ti 4GB,

RAM - 8GB and File Storage - 1TB HDD.

#### 3.2.1 Software Requirements

Operating System : Windows 10.

Platform : Anaconda Navigator

Language : Python 3.6

IDE : Spyder

## 4. DESIGN APPORACH AND DETAILS

### 4.1.1 Framework, Architecture or Module for the Proposed System (with explanation)

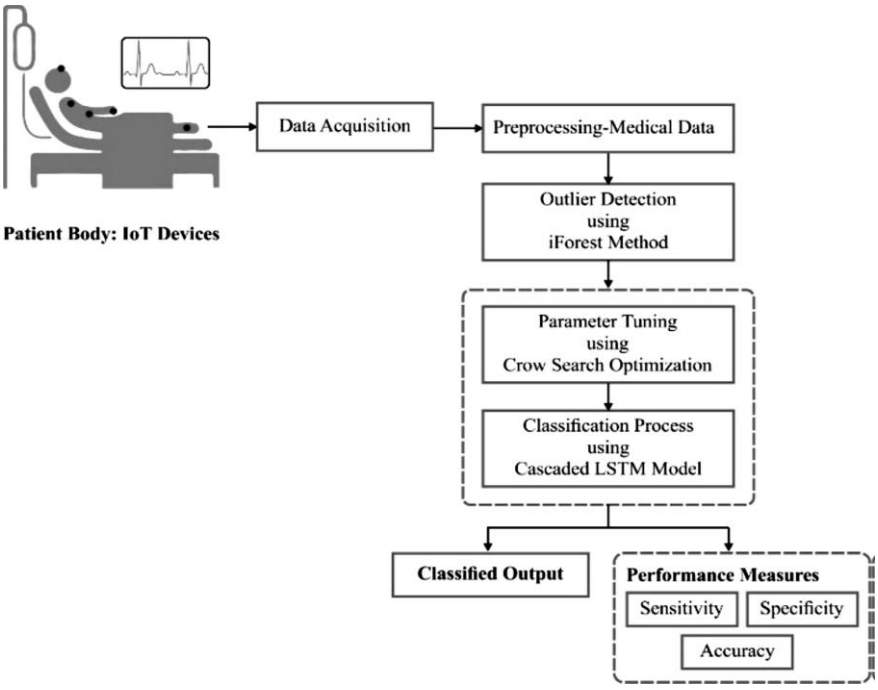



*Figure 1.1: The purposed system's architecture*

To know and differentiate between normal and abnormal rhythms of the heart of a patient, the sensors are used which, in turn, perform computationally difficult calculations. The patients can carry the smart devices such as smart watches, mobile phones etc. in their pocket. In addition, the data of different characteristics of the heart of a patient can be collected from extremely approved embedded electrocardiogram also known as ECG and sensors to determine temperature. This data is also useful in determining patient's lifestyle. Any smart devices like smart watches or mobile phones using their less powered Bluetooth integration process data received and label it either unhealthy or healthy. The Android platform can predict diabetes and maintain a healthy heart rate. The data of the patients are collected by IoT devices and convert the data into a structure after pre-processing that may be used by other devices. Data translation, format conversions, and class labelling are some of the processes of pre-processing. The dataset after pre-processing is visualized carefully so that the outliers can be recognized. The outliers are removed using iForest technique which uses anomaly score to determine whether a row in the dataset in outlier or inlier. After the removal of the outlier, a model is built using KNN along with CSO-CLSTM algorithm to increase the accuracy of the heart disease and diabetes prediction.

4.1.2 Proposed System Model (ER Diagram/Mathematical Modelling)

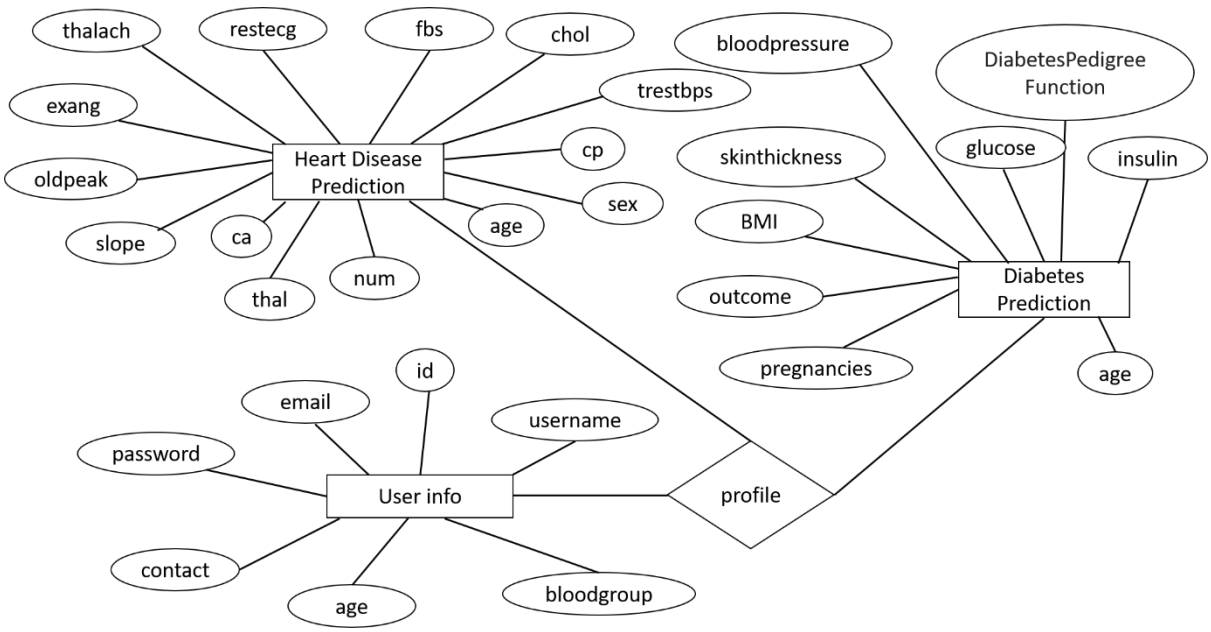



*Figure- 1.2 ER Diagram Mathematical Modelling*

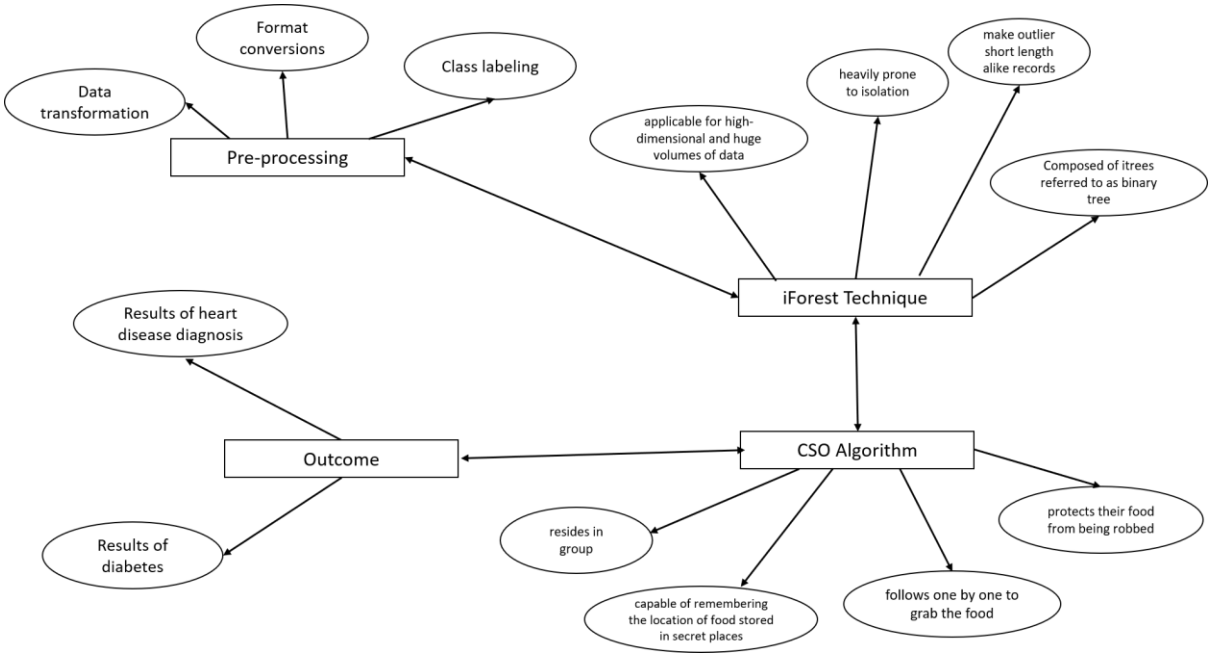



*Figure- 1.3 ER Diagram Mathematical Modelling*

Most of the researchers and analysts use 14 attributes in the dataset although there are 76 attributes available because they are often used in medical process to determine the heart disease. The researchers mostly have used the Cleveland database in their study to make prediction but we have used all the four datasets namely from Cleveland, Hungary, and two more. The main goal is to determine and find the accuracy whether the patient has any heart disease or not where 0 indicates of no presence of any disease and 4 indicates “presence”. The presence values of 1, 2, 3, and 4 and the absence values of number 0 were the attempt to distinguish in the Cleveland dataset.

In case of the diabetes dataset, several medical predictor attributes are used along with one aim variable, the Outcome. The BMI, age, the total number of pregnancies the patient had, the level of insulin and various other factors are included for predictor variables in the diabetes dataset.

4.1.3 Sequence Diagram

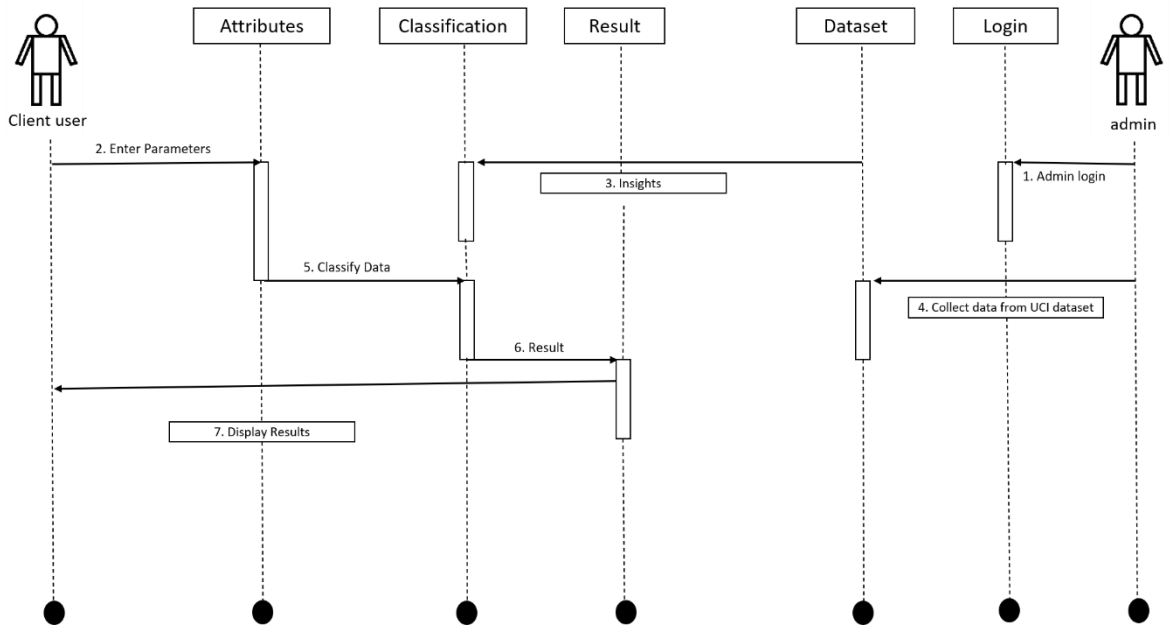



*Figure – 1.4 Sequence Diagram*

## 4.2 Requirement Analysis

### 4.2.1 Functional Requirements

#### 4.2.1.1 Product Perspective

This self-contained smart Health Care System controls hospital activities such as bed assignment, operations scheduling, personnel management, and administrative difficulties. The health service involves a number of parties.

#### 4.2.1.2 Product features

Only approved users with certain responsibilities will be able to access the system (Administrator, Operator). Depending on the user's job, he or she will only be able to access certain system modules.

A login system that allows only authorized users to access the system. When a patient is admitted, the front-desk personnel checks to determine if the patient is already a patient of the hospital. If he is, the computer records his or her name. Otherwise, this patient is assigned a new Patient ID.

#### 4.2.1.3 User characteristics

The system will be implemented in a healthcare setting. The key users will be administrators, doctors, nurses, and front-desk workers. Given the fact that not every user is computer literate. Some users may need to be taught how to utilize the system.

The system is also intended to be simple to use It employs a graphical user interface (GUI).

#### 4.2.1.4 Assumption & Dependencies

Before the system is deployed and tested, it is believed that compatible PCs will be available. It is believed that the healthcare system will have sufficient skilled personnel to manage the system.

#### 4.2.1.5 Domain Requirements

Because the system is intended to be used in the medical domain, the physical effects of various diseases on the hearts and its diagnosis must be considered prior to its development.

#### 4.2.1.6 User Requirements

The user is required to have professional medical knowledge in the area of embedded ECG, temperature sensors, heart disease like diabetes in order to use the system and understand the diagnosis made by it.

## 4.3 Non-Functional Requirements

### 4.3.1 Product Requirements

#### 4.3.1.1 Efficiency (in terms of Time and Space)

The system is currently able to diagnose heart diseases and diabetes with an accuracy of 96.16 percent and 97.26%.

#### 4.3.2.1 Reliability

The medical specialist examines the heart disease and diabetes in the current practice. By a wide margin, this system is more reliable than the existing method.

Availability: The system is always available.

#### 4.3.2.2 Usability

The technology is easy to use, as it takes an average of seconds for a medical specialist to identify a heart disease using it. It has a user interface that is simple to understand and utilize. The user's workload is quite low, making it very convenient to use.

## 4.3 Codes and Standards

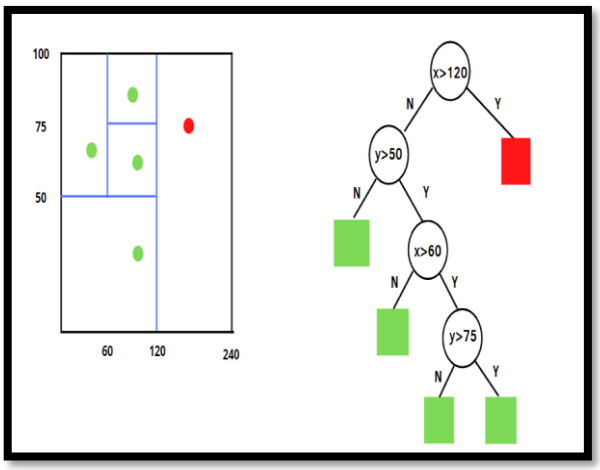
4.3.1 Proposed Diagnosis Model

IoT devices collect patient data and pre-process it to convert it into a format that may be used by other devices. Data translation, format conversions, and class labelling are some of the processes of pre-processing. The iForest approach is then used to remove any outliers found in the patient data. The CSO-CLSTM model is then used to classify the data into illness existence and non-existence.

Isolation Forests, like Random Forests, are most effective for removing an outlier. They are employed by making use of a sequence of decision trees. Then, they are used for a particular dataset. There are some distinctions, however. Short average path lengths play important role in isolation forest. They detect anomalies from observing on isolation trees with those average lengths. For each isolation tree, certain conditions can be used like:

i. Pick two features at random.

ii. The minimum as well as maximum characteristics are first made clear. A random value is chosen between them and the data points are split accordingly.




The process of splitting the observation is continued till all the observations are isolated.

### A. Anomaly Score

The isolation forest uses an Anomaly Score to measure how unusual a data piece is. It has a range of values from 0 to 1. The following formula is used to determine the anomalous score:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

In the above formula, E(h(x)) can be calculated as the average of h(x). Here, the external node is x. The distance from the root node to x is known as the length of the path and c(n) is calculated as the average of h(x) where n should be known and then h(x) can be normalized. The 3 scenarios that could happen are:

- We have an anomaly when there is short path length and easy to know the data point where the score of the observation is near to 1.

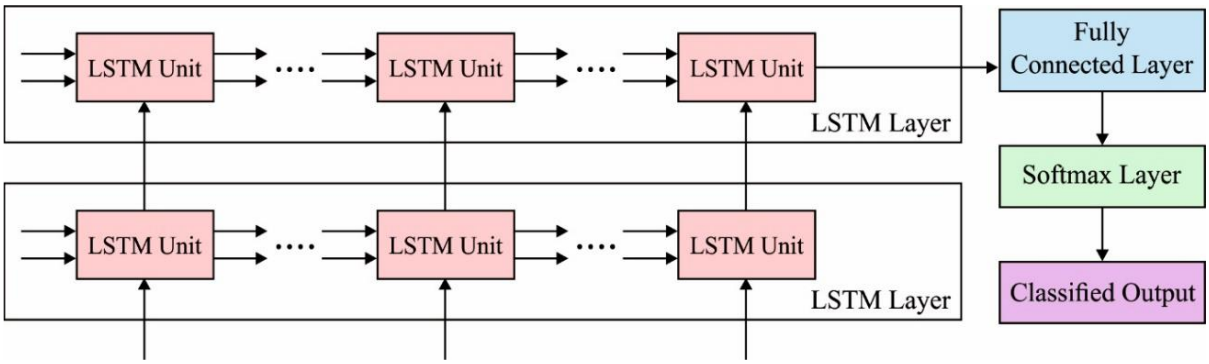
- The second scenario is when the path length is long and normal data point with the score of the observation being less than 0.5.

- When we have an anomaly score for all the observations of approximately near to 0.5, then the whole sample is anomaly-free.

For each tree, the isolation forest calculates the anomaly score making it efficient in isolating anomalies and average across the itrees. This, in fact, is achieved in less steps than any other normal observations. Also, the shorter routes are especially the characteristics of higher scoring outliers.

**Model for disease diagnosis based on the CS0-LSTM algorithm**

After outliers in healthcare data have been removed, the classification is done by making use of the model mentioned above. RNNs come under a type of Artificial Neural Network (ANN) that can be used to generate long-term structural variable time series. A significant characteristic of RNNs includes a unit known as time delay unit. The feedback connection is also an important characteristic of the RNNS where in the following phase, the previous data is put into action. The input layer also called as the sequence layer is the foundation of RNN.




Our technique uses a combination of two RNNs, a kind of ANN along with LSTM units. The core network uses elements from the mRMR model to classify data into four categories :

- NI-REM,
- W,
- N2, and
- N3)

A combination of REM epochs and N1 forms a single class. The PCA-estimated input properties are used in the alternate network. The RNN along with the NI-REM epochs are tgeb split into two categories which are N1 and REM. The SoftMax layer is considered a target class option. The capacity to generate the output probability range in a short amount of time is one of the most significant advantages of employing the SoftMax activation function. The numerical expression is shown in the following table.

4.3.2. Implementation

**A. Dataset**

Context

The dataset used in this project comes way back from 1988 which includes the four following databases:

a. Hungary,
b. Cleveland,
c. Long Beach V, and
d. Switzerland.

Most of the researchers and analysts use 14 attributes in the dataset although there are 76 attributes available because they are often used in medical process to determine the heart disease. If a patient has heart illness, the "target" field indicates that. It has an integer value of 0 to 1, with 0 denoting no disease and 1 denoting disease.

Content

Attribute Information:

1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholesterol in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina
10. old peak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) coloured by fluoroscopy

thal: 0 = normal; 1 = fixed defect; 2 = reversable defect

## B. Exploratory Data Analysis with Code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import hvplot.pandas
from scipy import stats

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
data = pd.read_csv("heart.csv")
data.head()
data.info()
data.shape
pd.set_option("display.float", "{:.2f}".format)
data.describe()
data.target.value_counts()
data.target.value_counts().hvplot.bar(
    title="Heart Disease Count", xlabel='Heart Disease', ylabel='Count',
    width=500, height=350
)
# Checking for messing values
data.isna().sum()
categorical_val = []
continous_val = []
for column in data.columns:
    if len(data[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
categorical_val
have_disease = data.loc[data['target']==1,
'sex'].value_counts().hvplot.bar(alpha=0.4)
no_disease = data.loc[data['target']==0,
'sex'].value_counts().hvplot.bar(alpha=0.4)

(no_disease * have_disease).opts(
    title="Heart Disease by Sex", xlabel='Sex', ylabel='Count',
    width=500, height=450, legend_cols=2, legend_position='top_right'
)
```

```
have_disease = data.loc[data['target']==1,
'cp'].value_counts().hvplot.bar(alpha=0.4)
no_disease = data.loc[data['target']==0,
'cp'].value_counts().hvplot.bar(alpha=0.4)

(no_disease * have_disease).opts(
    title="Heart Disease by Chest Pain Type", xlabel='Chest Pain Type',
ylabel='Count',
    width=500, height=450, legend_cols=2, legend_position='top_right'
)
have_disease = data.loc[data['target']==1,
'fbs'].value_counts().hvplot.bar(alpha=0.4)
no_disease = data.loc[data['target']==0,
'fbs'].value_counts().hvplot.bar(alpha=0.4)

(no_disease * have_disease).opts(
    title="Heart Disease by fasting blood sugar", xlabel='fasting blood
sugar > 120 mg/dl (1 = true; 0 = false)',
    ylabel='Count', width=500, height=450, legend_cols=2,
legend_position='top_right'
)
have_disease = data.loc[data['target']==1,
'restecg'].value_counts().hvplot.bar(alpha=0.4)
no_disease = data.loc[data['target']==0,
'restecg'].value_counts().hvplot.bar(alpha=0.4)

(no_disease * have_disease).opts(
    title="Heart Disease by resting electrocardiographic results",
xlabel='resting electrocardiographic results',
    ylabel='Count', width=500, height=450, legend_cols=2,
legend_position='top_right'
)
plt.figure(figsize=(15, 15))

for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
    data[data["target"] == 0][column].hist(bins=35, color='blue',
label='Have Heart Disease = NO', alpha=0.6)
    data[data["target"] == 1][column].hist(bins=35, color='red',
label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(continous_val, 1):
    plt.subplot(3, 2, i)
    data[data["target"] == 0][column].hist(bins=35, color='blue',
label='Have Heart Disease = NO', alpha=0.6)
    data[data["target"] == 1][column].hist(bins=35, color='red',
label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
# Create another figure
plt.figure(figsize=(9, 7))

# Scatter with postivie examples
plt.scatter(data.age[data.target==1],
            data.thalach[data.target==1],
            c="salmon")

# Scatter with negative examples
plt.scatter(data.age[data.target==0],
            data.thalach[data.target==0],
            c="lightblue")

# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);
# Let's make our correlation matrix a little prettier
corr_matrix = data.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
data.drop('target', axis=1).corrwith(data.target).hvplot.barh(
    width=600, height=400,
    title="Correlation between Heart Disease and Numeric Features",
    ylabel='Correlation', xlabel='Numerical Features',
```

```
)
categorical_val.remove('target')
dataset = pd.get_dummies(data, columns = categorical_val)
dataset.head()
print(data.columns)
print(dataset.columns)
from sklearn.preprocessing import StandardScaler


s_sc = StandardScaler()
col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])
```

### Models Building

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
```

In [27]:

```
from sklearn.model_selection import train_test_split

X = dataset.drop('target', axis=1)
y = dataset.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

## 1. Logistic Regression

8

```
from sklearn.linear_model import LogisticRegression

lr_clf = LogisticRegression(solver='liblinear')
lr_clf.fit(X_train, y_train)

print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 86.79%

_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision    0.88   0.86      0.87       0.87          0.87
recall       0.82   0.90      0.87       0.86          0.87
f1-score     0.85   0.88      0.87       0.87          0.87
support     97.00 115.00      0.87     212.00        212.00

_______________________________________________
Confusion Matrix:
 [[ 80  17]
 [ 11 104]]

Test Result:
================================================
Accuracy Score: 86.81%

_______________________________________________
CLASSIFICATION REPORT:
                0     1  accuracy  macro avg  weighted avg
precision    0.87  0.87      0.87       0.87          0.87
recall       0.83  0.90      0.87       0.86          0.87
f1-score     0.85  0.88      0.87       0.87          0.87
support     41.00 50.00      0.87      91.00         91.00

_______________________________________________
Confusion Matrix:
 [[34  7]
 [ 5 45]]
```

In [29]:

```
test_score = accuracy_score(y_test, lr_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, lr_clf.predict(X_train)) * 100

results_df = pd.DataFrame(data=[["Logistic Regression", train_score, test_score]],
                          columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df
```

Out[29]:

# 2. K-nearest neighbors

0

```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)

print_score(knn_clf, X_train, y_train, X_test, y_test, train=True)
print_score(knn_clf, X_train, y_train, X_test, y_test, train=False)
```

| | Model | Training Accuracy % | Testing Accuracy % |
|---|---|---|---|
| 0 | Logistic Regression | 86.79 | 86.81 |

```
Train Result:
================================================
Accuracy Score: 86.79%

_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision    0.86   0.87      0.87       0.87          0.87
recall       0.85   0.89      0.87       0.87          0.87
f1-score     0.85   0.88      0.87       0.87          0.87
support     97.00 115.00      0.87     212.00        212.00
_______________________________________________
Confusion Matrix:
 [[ 82  15]
 [ 13 102]]

Test Result:
================================================
Accuracy Score: 86.81%

_______________________________________________
CLASSIFICATION REPORT:
                0     1  accuracy  macro avg  weighted avg
precision    0.85  0.88      0.87       0.87          0.87
recall       0.85  0.88      0.87       0.87          0.87
f1-score     0.85  0.88      0.87       0.87          0.87
support     41.00 50.00      0.87      91.00         91.00
_______________________________________________
Confusion Matrix:
 [[35  6]
 [ 6 44]]
```

In [31]:

```
test_score = accuracy_score(y_test, knn_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, knn_clf.predict(X_train)) * 100

results_df_2 = pd.DataFrame(data=[["K-nearest neighbors", train_score, test_score]],
                            columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[31]:

| | Model | Training Accuracy % | Testing Accuracy % |
|---|---|---|---|
| **0** | Logistic Regression | 86.79 | 86.81 |
| **1** | K-nearest neighbors | 86.79 | 86.81 |

## 3. Support Vector machine

In [32]:

```
from sklearn.svm import SVC


svm_clf = SVC(kernel='rbf', gamma=0.1, C=1.0)
svm_clf.fit(X_train, y_train)

print_score(svm_clf, X_train, y_train, X_test, y_test, train=True)
print_score(svm_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 93.40%

_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision    0.94   0.93      0.93       0.93          0.93
recall       0.92   0.95      0.93       0.93          0.93
f1-score     0.93   0.94      0.93       0.93          0.93
support     97.00 115.00      0.93     212.00        212.00
_______________________________________________
Confusion Matrix:
 [[ 89   8]
 [  6 109]]

Test Result:
================================================
Accuracy Score: 87.91%

_______________________________________________
CLASSIFICATION REPORT:
                0     1  accuracy  macro avg  weighted avg
precision    0.86  0.90      0.88       0.88          0.88
recall       0.88  0.88      0.88       0.88          0.88
f1-score     0.87  0.89      0.88       0.88          0.88
support     41.00 50.00      0.88      91.00         91.00
_______________________________________________
Confusion Matrix:
 [[36  5]
 [ 6 44]]
```

3

```
test_score = accuracy_score(y_test, svm_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, svm_clf.predict(X_train)) * 100

results_df_2 = pd.DataFrame(data=[["Support Vector Machine", train_score, test_score]],
                            columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[33]:

| | Model | Training Accuracy % | Testing Accuracy % |
|---|---|---|---|
| 0 | Logistic Regression | 86.79 | 86.81 |
| 1 | K-nearest neighbors | 86.79 | 86.81 |
| 2 | Support Vector Machine | 93.40 | 87.91 |

# 4. Decision Tree Classifier

4

```
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)
```

```
Train Result:
================================================
Accuracy Score: 100.00%

_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision    1.00   1.00      1.00       1.00          1.00
recall       1.00   1.00      1.00       1.00          1.00
f1-score     1.00   1.00      1.00       1.00          1.00
support     97.00 115.00      1.00     212.00        212.00
_______________________________________________
Confusion Matrix:
 [[ 97   0]
 [  0 115]]

Test Result:
================================================
Accuracy Score: 78.02%

_______________________________________________
CLASSIFICATION REPORT:
                0     1  accuracy  macro avg  weighted avg
precision    0.72  0.84      0.78       0.78          0.79
recall       0.83  0.74      0.78       0.78          0.78
f1-score     0.77  0.79      0.78       0.78          0.78
support     41.00 50.00      0.78      91.00         91.00
_______________________________________________
Confusion Matrix:
 [[34  7]
 [13 37]]
```

5

```
test_score = accuracy_score(y_test, tree_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, tree_clf.predict(X_train)) * 100

results_df_2 = pd.DataFrame(data=[["Decision Tree Classifier", train_score, test_score]],
                            columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[35]:

| | Model | Training Accuracy % | Testing Accuracy % |
|---|---|---|---|
| 0 | Logistic Regression | 86.79 | 86.81 |
| 1 | K-nearest neighbors | 86.79 | 86.81 |
| 2 | Support Vector Machine | 93.40 | 87.91 |
| 3 | Decision Tree Classifier | 100.00 | 78.02 |

# 5. Random Forest

6

```
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)
```

```
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision    1.00   1.00      1.00       1.00          1.00
recall       1.00   1.00      1.00       1.00          1.00
f1-score     1.00   1.00      1.00       1.00          1.00
support     97.00 115.00      1.00     212.00        212.00
_______________________________________________
Confusion Matrix:
 [[ 97   0]
 [  0 115]]

Test Result:
================================================
Accuracy Score: 82.42%
_______________________________________________
CLASSIFICATION REPORT:
                0     1  accuracy  macro avg  weighted avg
precision    0.80  0.84      0.82       0.82          0.82
recall       0.80  0.84      0.82       0.82          0.82
f1-score     0.80  0.84      0.82       0.82          0.82
support     41.00 50.00      0.82      91.00         91.00
_______________________________________________
Confusion Matrix:
 [[33  8]
 [ 8 42]]
```

7

```
test_score = accuracy_score(y_test, rf_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, rf_clf.predict(X_train)) * 100

results_df_2 = pd.DataFrame(data=[["Random Forest Classifier", train_score, test_score]],
                            columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[37]:

| | Model | Training Accuracy % | Testing Accuracy % |
|---|---|---|---|
| 0 | Logistic Regression | 86.79 | 86.81 |
| 1 | K-nearest neighbors | 86.79 | 86.81 |
| 2 | Support Vector Machine | 93.40 | 87.91 |
| 3 | Decision Tree Classifier | 100.00 | 78.02 |
| 4 | Random Forest Classifier | 100.00 | 82.42 |

# 6. XGBoost Classifer

]:

```
from xgboost import XGBClassifier

xgb_clf = XGBClassifier(use_label_encoder=False)
xgb_clf.fit(X_train, y_train)

print_score(xgb_clf, X_train, y_train, X_test, y_test, train=True)
print_score(xgb_clf, X_train, y_train, X_test, y_test, train=False)
```

```
[14:09:33] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the d
efault evaluation metric used with the objective 'binary:logistic' was chang
ed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to re
store the old behavior.
Train Result:
================================================
Accuracy Score: 100.00%
________________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision    1.00   1.00      1.00       1.00          1.00
recall       1.00   1.00      1.00       1.00          1.00
f1-score     1.00   1.00      1.00       1.00          1.00
support     97.00 115.00      1.00     212.00        212.00
________________________________________________
Confusion Matrix:
 [[ 97   0]
 [  0 115]]

Test Result:
================================================
Accuracy Score: 82.42%
________________________________________________
CLASSIFICATION REPORT:
                0     1  accuracy  macro avg  weighted avg
precision    0.80  0.84      0.82       0.82          0.82
recall       0.80  0.84      0.82       0.82          0.82
f1-score     0.80  0.84      0.82       0.82          0.82
support     41.00 50.00      0.82      91.00         91.00
________________________________________________
Confusion Matrix:
 [[33  8]
 [ 8 42]]
```

In [3 ]:

]:

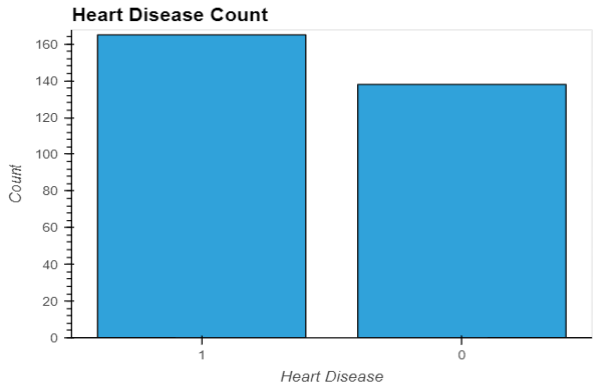
```
test_score = accuracy_score(y_test, xgb_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, xgb_clf.predict(X_train)) * 100

results_df_2 = pd.DataFrame(data=[["XGBoost Classifier", train_score, test_score]],
                            columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[43]:

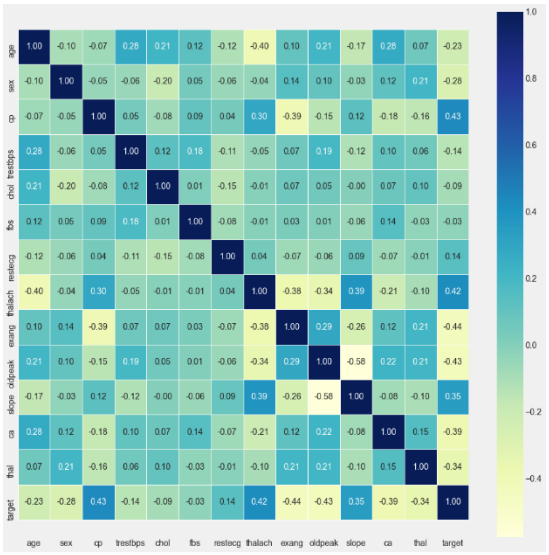
| | Model | Training Accuracy % | Testing Accuracy % |
|---|---|---|---|
| 0 | Logistic Regression | 86.79 | 86.81 |
| 1 | K-nearest neighbors | 86.79 | 86.81 |
| 2 | Support Vector Machine | 93.40 | 87.91 |
| 3 | Decision Tree Classifier | 100.00 | 78.02 |
| 4 | Random Forest Classifier | 100.00 | 82.42 |
| 5 | XGBoost Classifier | 100.00 | 82.42 |
| 6 | XGBoost Classifier | 100.00 | 82.42 |

I. The number of persons who have heart disease vs. the number of people who do not have heart disease:

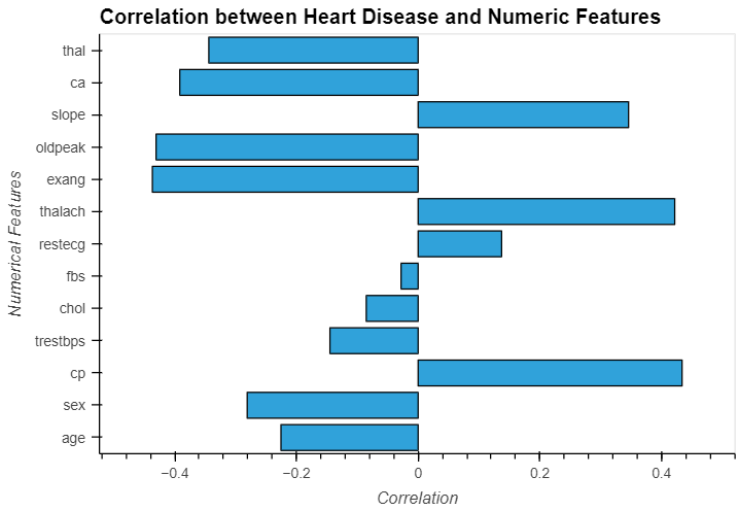



- 165 people → heart disease and
- 138 persons → without heart disease.

II. Heatmap of Corelation Matrix




I. Corelation with Target

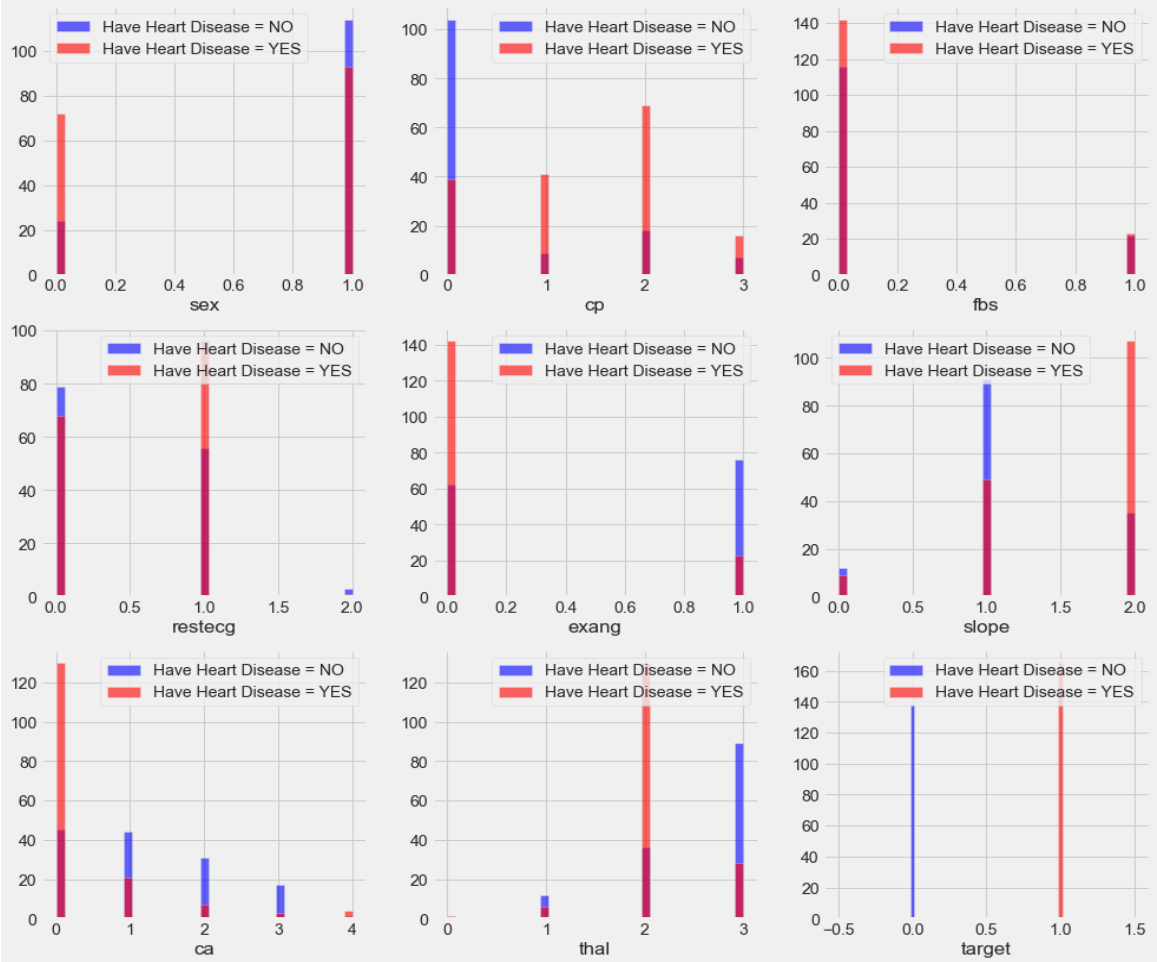



Correlational observations:

• The least correlated variables with the target variable are fbs and chol.

• The target variable with different attributes exhibits an important association.

III. Histogram of Categorical Values




The above graph gives the following observations:

• Chest Pain: cp→1,2,3 = have high risk of heart disease and other condition

cp → 0 = less risk

• resting EKG results: People with a value of 1 (abnormal heart rhythm ranging from mild symptoms to severe difficulties) are more likely to develop heart disease.

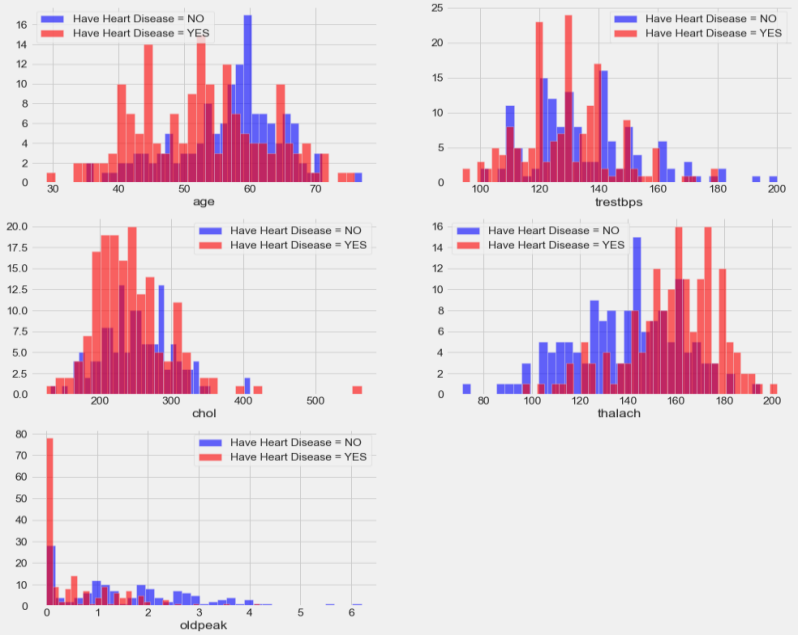
• exang exercise-induced angina: People with a value of 0 (No — exercise-induced angina) have greater heart disease than people with a value of 1 (Yes — exercise-induced angina).

People with a slope value of 2 (Down sloping: symptoms of an unhealthy heart) are more likely to suffer heart disease than people with a slope value of 2 slope is 0 (Upsloping: best heart rate with exercise) or 1 (Flat sloping: minimum change (typical healthy heart)).

• ca number of main vessels stained by fluoroscopy (0–3): Because more blood movement is beneficial, those with ca equal to 0 are more likely to have heart disease.

- Thal(thallium stress) result: thal value = 2
  (Defect corrected: used to flaw, but it is no longer) are at high risk to develop heart disease.

IV. Continuous Values Histogram




The above graph gives the following observations:

- trestbps: a relaxing BP of more than 120 or 140 is a cause for alarm.
- Cholesterol: anything over 200 is a cause for trouble.
- thalach: maximal heart rate > 140 → have a higher risk of heart disease.

the previous high point of exercise-induced ST Heart stress during exercise is examined in depression vs. rest. A heart that isn't in good shape will be more stressed.

## C. Outlier Removal

In [1]:
```
import pandas as pd
import plotly.express as px
from sklearn.ensemble import IsolationForest
```

In [2]:
```
data = pd.read_csv('heart1.csv')
X,y = data,data.target
data.head()
```

Out[2]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 2 | 2 | 3 | 0 |
| **1** | 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | 0 | 3 | 0 |
| **2** | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0 | 3 | 0 |
| **3** | 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 2 | 1 | 3 | 0 |
| **4** | 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | 3 | 2 | 0 |

In [3]:
```
iforest = IsolationForest(n_estimators=100, max_samples='auto',
                          contamination=0.05, max_features=13,
                          bootstrap=False, n_jobs=-1, random_state=50)
```

In [4]:
```
pred= iforest.fit_predict(X)
data['scores']=iforest.decision_function(X)
data['anomaly_label']=pred
```

In [12]:
```
data[data.anomaly_label==-1].head(10)
```

Out[12]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target | scores | anomaly_label | anomaly |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | 3 | 2 | 0 | -0.006082 | -1 | outlier |
| **6** | 58 | 1 | 0 | 114 | 318 | 0 | 2 | 140 | 0 | 4.4 | 0 | 3 | 1 | 0 | -0.057239 | -1 | outlier |
| **11** | 43 | 0 | 0 | 132 | 341 | 1 | 0 | 136 | 1 | 3.0 | 1 | 0 | 3 | 0 | -0.001886 | -1 | outlier |
| **29** | 55 | 0 | 0 | 180 | 327 | 0 | 2 | 117 | 1 | 3.4 | 1 | 0 | 2 | 0 | -0.028872 | -1 | outlier |
| **47** | 66 | 0 | 0 | 178 | 228 | 1 | 1 | 165 | 1 | 1.0 | 1 | 2 | 3 | 0 | -0.007784 | -1 | outlier |
| **69** | 62 | 0 | 0 | 160 | 164 | 0 | 0 | 145 | 0 | 6.2 | 0 | 3 | 3 | 0 | -0.053038 | -1 | outlier |
| **135** | 58 | 0 | 0 | 170 | 225 | 1 | 0 | 146 | 1 | 2.8 | 1 | 2 | 1 | 0 | -0.012022 | -1 | outlier |
| **150** | 58 | 1 | 0 | 114 | 318 | 0 | 2 | 140 | 0 | 4.4 | 0 | 3 | 1 | 0 | -0.057239 | -1 | outlier |
| **158** | 67 | 0 | 2 | 115 | 564 | 0 | 0 | 160 | 0 | 1.6 | 1 | 0 | 3 | 1 | -0.002503 | -1 | outlier |
| **175** | 56 | 0 | 0 | 200 | 288 | 1 | 0 | 133 | 1 | 4.0 | 0 | 2 | 3 | 0 | -0.036521 | -1 | outlier |

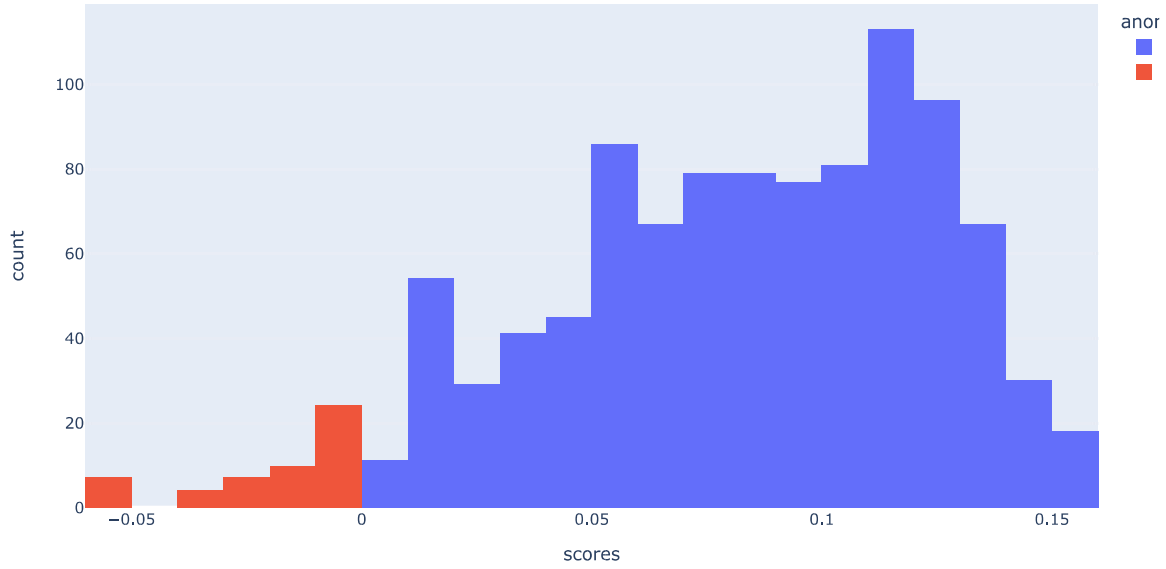
In [6]:
```
data[data.anomaly_label==1]
```

Out[6]:

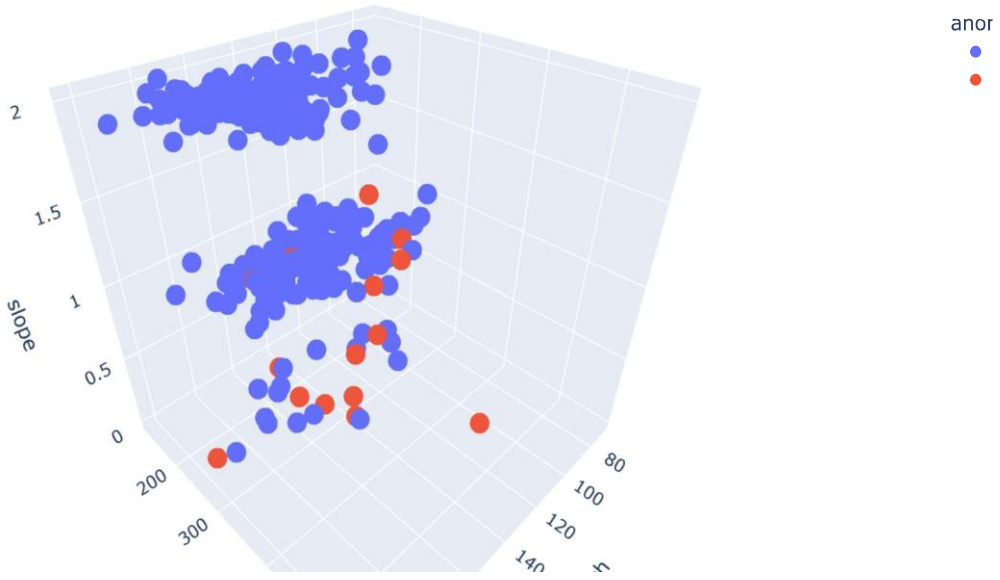
| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target | scores | anomaly_label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 2 | 2 | 3 | 0 | 0.103554 | 1 |
| **1** | 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | 0 | 3 | 0 | 0.032815 | 1 |
| **2** | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0 | 3 | 0 | 0.039500 | 1 |
| **3** | 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 2 | 1 | 3 | 0 | 0.109699 | 1 |
| **5** | 58 | 0 | 0 | 100 | 248 | 0 | 0 | 122 | 0 | 1.0 | 1 | 0 | 2 | 1 | 0.077271 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1020** | 59 | 1 | 1 | 140 | 221 | 0 | 1 | 164 | 1 | 0.0 | 2 | 0 | 2 | 1 | 0.104645 | 1 |
| **1021** | 60 | 1 | 0 | 125 | 258 | 0 | 0 | 141 | 1 | 2.8 | 1 | 1 | 3 | 0 | 0.145382 | 1 |
| **1022** | 47 | 1 | 0 | 110 | 275 | 0 | 0 | 118 | 1 | 1.0 | 1 | 1 | 2 | 0 | 0.122524 | 1 |
| **1023** | 50 | 0 | 0 | 110 | 254 | 0 | 0 | 159 | 0 | 0.0 | 2 | 0 | 2 | 1 | 0.118418 | 1 |
| **1024** | 54 | 1 | 0 | 120 | 188 | 0 | 1 | 113 | 0 | 1.4 | 1 | 1 | 3 | 0 | 0.119426 | 1 |

973 rows × 16 columns

In [13]:
```
data['anomaly']=data['anomaly_label'].apply(lambda x: 'outlier' if x==-1  else 'inlier')
fig=px.histogram(data,x='scores',color='anomaly')
fig.show()
```




In [8]:
```
fig = px.scatter_3d(data, x='thalach',
                    y='chol',
                    z='slope',
                    color='anomaly')
fig.show()
```



In [9]:
```
from sklearn.tree import export_graphviz
from subprocess import call
from IPython.display import Image
estimator = iforest.estimators_[5]
print(estimator)
export_graphviz(estimator,out_file='tree.dot',max_depth=5,
                feature_names = ['age','sex','cp','trestbps','chol','fbs',
                                 'restecg','thalach','exang','oldpeak','slope',
                                 'ca','thal'],
                special_characters=True,rounded=True,precision=2)
#call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])
#Image(filename = 'tree.png')
```

```
ExtraTreeRegressor(max_depth=8, max_features=1, random_state=1622078562)
```

In [10]:
```
data[(data['age']<=45)|(data['age']>=65)]
```

Out[10]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target | scores | anomaly_label | anomaly |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2** | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0 | 3 | 0 | 0.039500 | 1 | inlier |
| **10** | 71 | 0 | 0 | 112 | 149 | 0 | 1 | 125 | 0 | 1.6 | 1 | 0 | 2 | 1 | 0.057564 | 1 | inlier |
| **11** | 43 | 0 | 0 | 132 | 341 | 1 | 0 | 136 | 1 | 3.0 | 1 | 0 | 3 | 0 | -0.001886 | -1 | outlier |
| **12** | 34 | 0 | 1 | 118 | 210 | 0 | 1 | 192 | 0 | 0.7 | 2 | 0 | 2 | 1 | 0.076397 | 1 | inlier |
| **15** | 34 | 0 | 1 | 118 | 210 | 0 | 1 | 192 | 0 | 0.7 | 2 | 0 | 2 | 1 | 0.076397 | 1 | inlier |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1009** | 40 | 1 | 0 | 152 | 223 | 0 | 1 | 181 | 0 | 0.0 | 2 | 0 | 3 | 0 | 0.067003 | 1 | inlier |
| **1011** | 45 | 1 | 1 | 128 | 308 | 0 | 0 | 170 | 0 | 0.0 | 2 | 0 | 2 | 1 | 0.136064 | 1 | inlier |
| **1014** | 44 | 0 | 2 | 108 | 141 | 0 | 1 | 175 | 0 | 0.6 | 1 | 0 | 2 | 1 | 0.088219 | 1 | inlier |
| **1016** | 65 | 1 | 3 | 138 | 282 | 1 | 0 | 174 | 0 | 1.4 | 1 | 1 | 2 | 0 | 0.032919 | 1 | inlier |
| **1018** | 41 | 1 | 0 | 110 | 172 | 0 | 0 | 158 | 0 | 0.0 | 2 | 0 | 3 | 0 | 0.076183 | 1 | inlier |

351 rows × 17 columns

## Removal of Outlier from the dataset

In [1]:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno # To visualize missing value
import plotly.graph_objects as go # To Generate Graphs
import plotly.express as px # To Generate box plot for statistical representation
%matplotlib inline
```

In [2]:

```
df = pd.read_csv('heart1.csv')
df.head(5)
```

Out[2]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 2 | 2 | 3 | 0 |
| **1** | 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | 0 | 3 | 0 |
| **2** | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0 | 3 | 0 |
| **3** | 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 2 | 1 | 3 | 0 |
| **4** | 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | 3 | 2 | 0 |

In [3]:

```
df.shape
```

Out[3]:

```
(1025, 14)
```

In [4]:

```
df.columns
```

Out[4]:

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

[5]:

```
<bound method DataFrame.info of       age  sex  cp  trestbps  chol  fbs  res
tecg  thalach  exang  oldpeak  \
0        52    1   0       125   212    0        1      168      0      1.0
1        53    1   0       140   203    1        0      155      1      3.1
2        70    1   0       145   174    0        1      125      1      2.6
3        61    1   0       148   203    0        1      161      0      0.0
4        62    0   0       138   294    1        1      106      0      1.9
...     ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
1020     59    1   1       140   221    0        1      164      1      0.0
1021     60    1   0       125   258    0        0      141      1      2.8
1022     47    1   0       110   275    0        0      118      1      1.0
1023     50    0   0       110   254    0        0      159      0      0.0
1024     54    1   0       120   188    0        1      113      0      1.4

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
...     ...  ..   ...     ...
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]>
```

In [6]:

```
df.nunique()
```

Out[6]:

```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

[7]:

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [8]:

```
# Check for the data characters mistakes
df['ca'].unique()
```

Out[8]:

```
array([2, 0, 1, 3, 4], dtype=int64)
```

In [9]:

```
df.ca.value_counts()
```

Out[9]:

```
0    578
1    226
2    134
3     69
4     18
Name: ca, dtype: int64
```

```
[df['ca']==4]
```

[10]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | targ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **52** | 38 | 1 | 2 | 138 | 175 | 0 | 1 | 173 | 0 | 0.0 | 2 | 4 | 2 | |
| **83** | 38 | 1 | 2 | 138 | 175 | 0 | 1 | 173 | 0 | 0.0 | 2 | 4 | 2 | |
| **128** | 52 | 1 | 2 | 138 | 223 | 0 | 1 | 169 | 0 | 0.0 | 2 | 4 | 2 | |
| **208** | 38 | 1 | 2 | 138 | 175 | 0 | 1 | 173 | 0 | 0.0 | 2 | 4 | 2 | |
| **242** | 38 | 1 | 2 | 138 | 175 | 0 | 1 | 173 | 0 | 0.0 | 2 | 4 | 2 | |
| **290** | 52 | 1 | 2 | 138 | 223 | 0 | 1 | 169 | 0 | 0.0 | 2 | 4 | 2 | |
| **340** | 38 | 1 | 2 | 138 | 175 | 0 | 1 | 173 | 0 | 0.0 | 2 | 4 | 2 | |
| **348** | 43 | 1 | 0 | 132 | 247 | 1 | 0 | 143 | 1 | 0.1 | 1 | 4 | 3 | |
| **417** | 52 | 1 | 2 | 138 | 223 | 0 | 1 | 169 | 0 | 0.0 | 2 | 4 | 2 | |
| **428** | 43 | 1 | 0 | 132 | 247 | 1 | 0 | 143 | 1 | 0.1 | 1 | 4 | 3 | |
| **465** | 38 | 1 | 2 | 138 | 175 | 0 | 1 | 173 | 0 | 0.0 | 2 | 4 | 2 | |
| **521** | 58 | 1 | 1 | 125 | 220 | 0 | 1 | 144 | 0 | 0.4 | 1 | 4 | 3 | |
| **597** | 38 | 1 | 2 | 138 | 175 | 0 | 1 | 173 | 0 | 0.0 | 2 | 4 | 2 | |
| **743** | 58 | 1 | 1 | 125 | 220 | 0 | 1 | 144 | 0 | 0.4 | 1 | 4 | 3 | |
| **749** | 58 | 1 | 1 | 125 | 220 | 0 | 1 | 144 | 0 | 0.4 | 1 | 4 | 3 | |
| **831** | 58 | 1 | 1 | 125 | 220 | 0 | 1 | 144 | 0 | 0.4 | 1 | 4 | 3 | |
| **970** | 38 | 1 | 2 | 138 | 175 | 0 | 1 | 173 | 0 | 0.0 | 2 | 4 | 2 | |
| **993** | 43 | 1 | 0 | 132 | 247 | 1 | 0 | 143 | 1 | 0.1 | 1 | 4 | 3 | |

In [11]:

```
df.loc[df['ca']==4, 'ca'] = np.NaN
```

In [12]:

```
df['ca'].unique()
```

Out[12]:

```
array([ 2.,  0.,  1.,  3., nan])
```

```
    3

.thal.value_counts()
```

[13]:

```
2    544
3    410
1     64
0      7
Name: thal, dtype: int64
```

In [14]:

```
df.loc[df['thal']==0, 'thal'] = np.NaN
```

In [15]:

```
df[df['thal']==0]
```

Out[15]:

| age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

In [16]:

```
df['thal'].unique()
```

Out[16]:

```
array([ 3.,  2.,  1., nan])
```

In [17]:

```
#to check missing values
df.isnull().sum()
```

Out[17]:

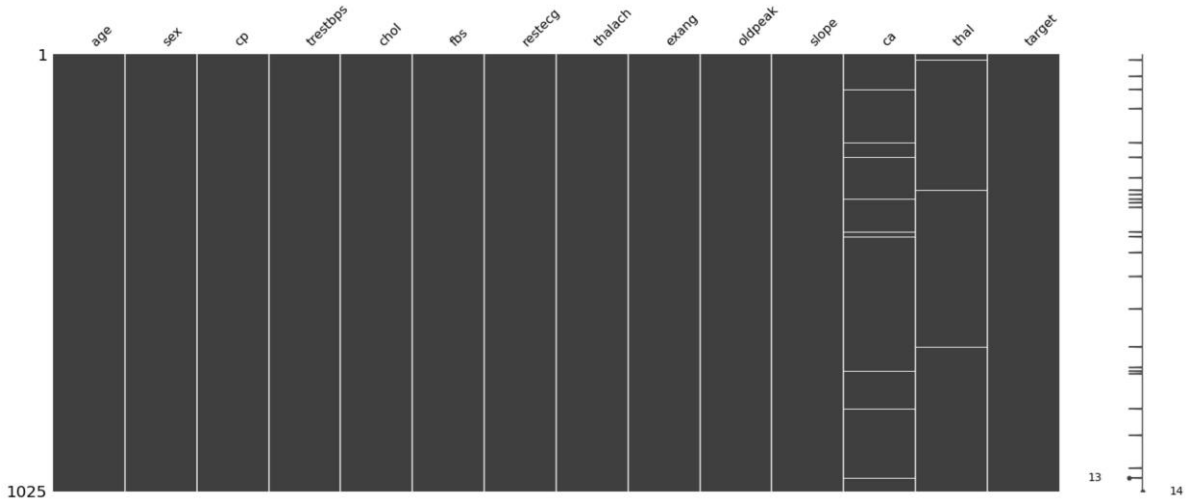
```
age          0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca          18
thal         7
target       0
dtype: int64
```

8

```
#visualizing using Missingo Library
msno.matrix(df)
```

Out[18]:

<AxesSubplot:>




In [19]:

```
#Replace the NaN with median.

df = df.fillna(df.median())
df.isnull().sum()
```

Out[19]:

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

0

```
#Check for duplicate rows
duplicated = df.duplicated().sum()
if duplicated:
    print('Duplicates Rows in Dataset are : {}'.format(duplicated))
else:
    print('Dataset contains no Duplicate Values')
```

```
Duplicates Rows in Dataset are : 723
```

In [21]:

```
duplicated = df[df.duplicated(keep=False)]
duplicated.head()
```

Out[21]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | targe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 2 | 2.0 | 3.0 | ( |
| **1** | 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | 0.0 | 3.0 | ( |
| **2** | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0.0 | 3.0 | ( |
| **3** | 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 2 | 1.0 | 3.0 | ( |
| **4** | 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | 3.0 | 2.0 | ( |

In [22]:

```
#Statistics Summary to know the basis stats
df.describe()
```

Out[22]:

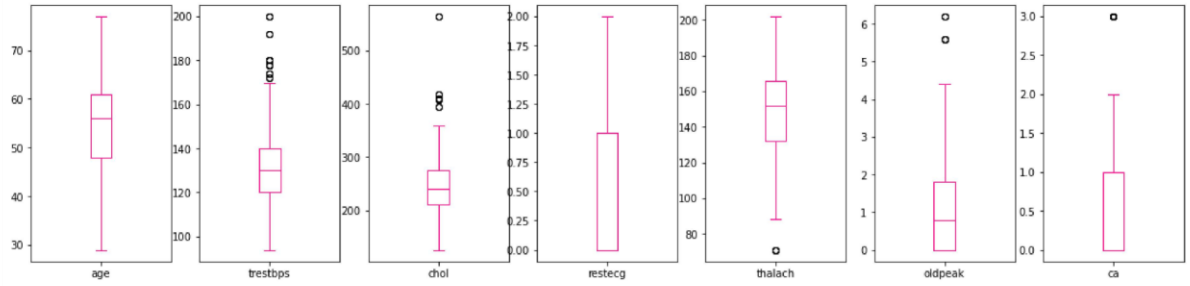
| | age | sex | cp | trestbps | chol | fbs | reste |
|---|---|---|---|---|---|---|---|
| **count** | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.00000 | 1025.000000 | 1025.000( |
| **mean** | 54.434146 | 0.695610 | 0.942439 | 131.611707 | 246.00000 | 0.149268 | 0.5297 |
| **std** | 9.072290 | 0.460373 | 1.029641 | 17.516718 | 51.59251 | 0.356527 | 0.5278 |
| **min** | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.00000 | 0.000000 | 0.000( |
| **25%** | 48.000000 | 0.000000 | 0.000000 | 120.000000 | 211.00000 | 0.000000 | 0.000( |
| **50%** | 56.000000 | 1.000000 | 1.000000 | 130.000000 | 240.00000 | 0.000000 | 1.000( |
| **75%** | 61.000000 | 1.000000 | 2.000000 | 140.000000 | 275.00000 | 0.000000 | 1.000( |
| **max** | 77.000000 | 1.000000 | 3.000000 | 200.000000 | 564.00000 | 1.000000 | 2.000( |

3

```
#Before we plot the outliers, let's change the labeling for better visualization and interp
df['target'] = df.target.replace({1: "Disease", 0: "No_disease"})
df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
df['cp'] = df.cp.replace({0: "typical_angina",
                          1: "atypical_angina",
                          2:"non-anginal pain",
                          3: "asymtomatic"})
df['exang'] = df.exang.replace({1: "Yes", 0: "No"})
df['fbs'] = df.fbs.replace({1: "True", 0: "False"})
df['slope'] = df.slope.replace({0: "upsloping", 1: "flat",2:"downsloping"})
df['thal'] = df.thal.replace({1: "fixed_defect", 2: "reversable_defect", 3:"normal"})
```

In [24]:

```
#Outliers and how to remove them
df.plot(kind='box', subplots=True, layout=(2,7),
sharex=False,sharey=False, figsize=(20, 10),
color='deeppink');
```




In [25]:

```
# define continuous variable & plot
continous_features = ['age','trestbps','chol','thalach','oldpeak']
def outliers(df_out, drop = False):
    for each_feature in df_out.columns:
        feature_data = df_out[each_feature]
        Q1 = np.percentile(feature_data, 25.) # 25th percentile of the data of the given fe
        Q3 = np.percentile(feature_data, 75.) # 75th percentile of the data of the given fe
        IQR = Q3-Q1 #Interquartile Range
        outlier_step = IQR * 1.5 #That's we were talking about above
        outliers = feature_data[~((feature_data >= Q1 - outlier_step) & (feature_data <= Q3
        if not drop:
            print('For the feature {}, No of Outliers is {}'.format(each_feature, len(outli
        if drop:
            df.drop(outliers, inplace = True, errors = 'ignore')
            print('Outliers from {} feature removed'.format(each_feature))

outliers(df[continous_features])
```

```
For the feature age, No of Outliers is 0
For the feature trestbps, No of Outliers is 30
For the feature chol, No of Outliers is 16
For the feature thalach, No of Outliers is 4
For the feature oldpeak, No of Outliers is 7
```

6

```
outliers(df[continous_features], drop=True)
```

```
Outliers from age feature removed
Outliers from trestbps feature removed
Outliers from chol feature removed
Outliers from thalach feature removed
Outliers from oldpeak feature removed
```

In [27]:

```
df.shape
```

Out[27]:

```
(968, 14)
```

### D. RNN along with CSO-LSTM algorithm

31

```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
#from keras.optimizers import Adam
from keras.layers import Dropout
from keras import regularizers
from keras.layers import LSTM


# define a function to build the keras model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(16, input_dim=13, kernel_initializer='normal', kernel_regularizer=regul
    model.add(Dropout(0.25))
    model.add(Dense(8, kernel_initializer='normal', kernel_regularizer=regularizers.l2(0.00
    model.add(Dropout(0.25))
    model.add(Dense(2, activation='softmax'))

    # compile model
    #opt = keras.optimizers.Adam(learning_rate=0.01)
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'
    return model

model = create_model()

print(model.summary())
```

```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_48 (Dense)            (None, 16)                224

 dropout_32 (Dropout)        (None, 16)                0

 dense_49 (Dense)            (None, 8)                 136

 dropout_33 (Dropout)        (None, 8)                 0

 dense_50 (Dense)            (None, 2)                 18

=================================================================
Total params: 378
Trainable params: 378
Non-trainable params: 0
_________________________________________________________________
None
```

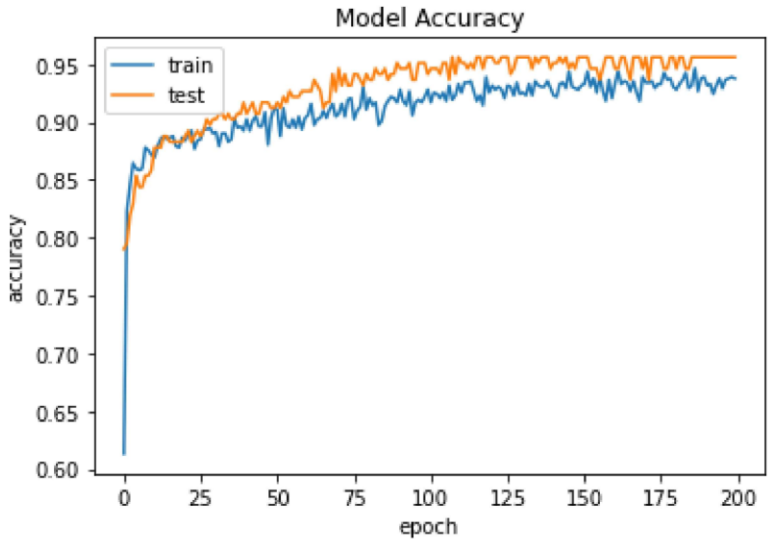
18

```
history=model.fit(X_train, Y_train, validation_data=(X_test, Y_test),epochs=200, batch_size
```

```
Epoch 1/200
WARNING:tensorflow:AutoGraph could not transform <function Model.make_trai
n_function.<locals>.train_function at 0x00000269FF7D0C10> and will run it
as-is.
Please report this to the TensorFlow team. When filing the bug, set the ve
rbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the f
ull output.
Cause: closure mismatch, requested ('self', 'step_function'), but source f
unction had ()
To silence this warning, decorate the function with @tf.autograph.experime
ntal.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_train_functio
n.<locals>.train_function at 0x00000269FF7D0C10> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the ve
rbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the f
ull output.
Cause: closure mismatch, requested ('self', 'step_function'), but source f
unction had ()
To silence this warning, decorate the function with @tf.autograph.experime
```

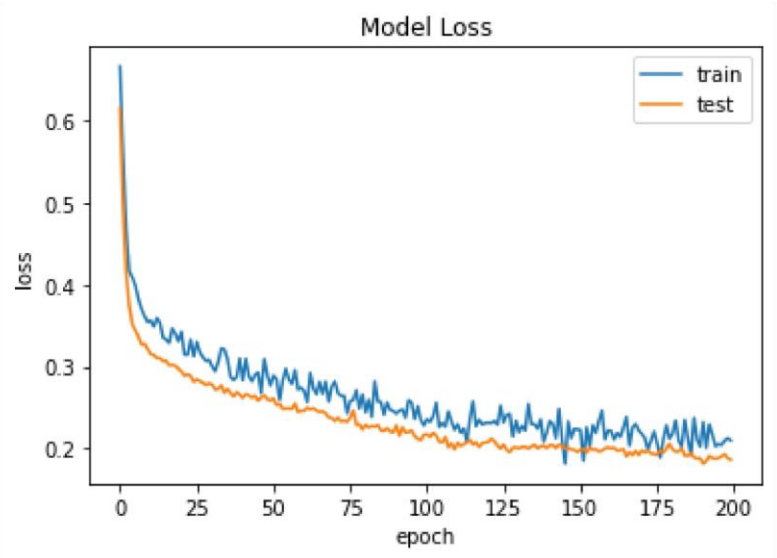
In [119]:

```
import matplotlib.pyplot as plt
%matplotlib inline
# Model accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```



0

```
# Model Losss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```




In [121]:

```
Y_train_binary = y_train.copy()
Y_test_binary = y_test.copy()

Y_train_binary[Y_train_binary > 0] = 1
Y_test_binary[Y_test_binary > 0] = 1

print(Y_train_binary[:20])
```

```
[0 0 0 1 1 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0]
```

2

```
# define a new keras model for binary classification
def create_binary_model():
    # create model
    model = Sequential()
    model.add(Dense(16, input_dim=13, kernel_initializer='normal',  kernel_regularizer=regu
    model.add(Dropout(0.25))
    model.add(Dense(8, kernel_initializer='normal',  kernel_regularizer=regularizers.l2(0.0
    model.add(Dropout(0.25))
    model.add(Dense(1, activation='sigmoid'))

    # Compile model
    #adam = Adam(lr=0.001)
    model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
    return model

binary_model = create_binary_model()

print(binary_model.summary())
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_36 (Dense)            (None, 16)                224

 dropout_24 (Dropout)        (None, 16)                0

 dense_37 (Dense)            (None, 8)                 136

 dropout_25 (Dropout)        (None, 8)                 0

 dense_38 (Dense)            (None, 1)                 9

=================================================================
Total params: 369
Trainable params: 369
Non-trainable params: 0
_________________________________________________________________
None
```

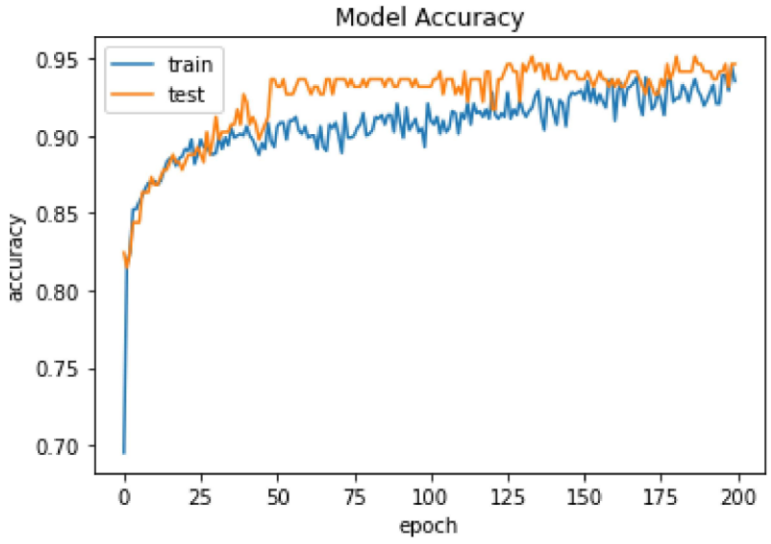
3

```
history=binary_model.fit(X_train, Y_train_binary, validation_data=(X_test, Y_test_binary),
```

```
Epoch 1/200
WARNING:tensorflow:AutoGraph could not transform <function Model.make_trai
n_function.<locals>.train_function at 0x00000269FF1CF280> and will run it
as-is.
Please report this to the TensorFlow team. When filing the bug, set the ve
rbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the f
ull output.
Cause: closure mismatch, requested ('self', 'step_function'), but source f
unction had ()
To silence this warning, decorate the function with @tf.autograph.experime
ntal.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_train_functio
n.<locals>.train_function at 0x00000269FF1CF280> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the ve
rbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the f
ull output.
Cause: closure mismatch, requested ('self', 'step_function'), but source f
unction had ()
To silence this warning, decorate the function with @tf.autograph.experime
```

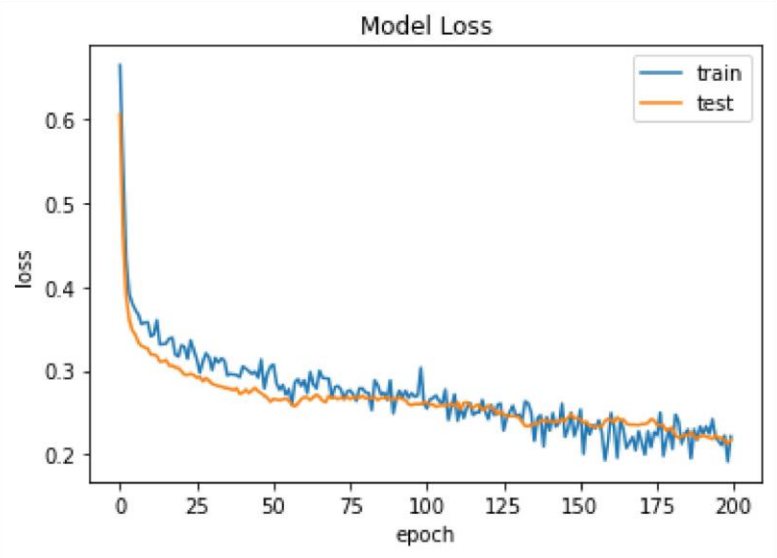
In [124]:

```
import matplotlib.pyplot as plt
%matplotlib inline
# Model accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```



5

```
# Model Losss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```



6

```
# generate classification report using predictions for categorical model
from sklearn.metrics import classification_report, accuracy_score

categorical_pred = np.argmax(model.predict(X_test), axis=1)

print('Results for Categorical Model')
print(accuracy_score(y_test, categorical_pred))
print(classification_report(y_test, categorical_pred))
```

```
WARNING:tensorflow:AutoGraph could not transform <function Model.make_predic
t_function.<locals>.predict_function at 0x00000269F5DFFD30> and will run it
as-is.
Please report this to the TensorFlow team. When filing the bug, set the verb
osity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full
output.
Cause: closure mismatch, requested ('self', 'step_function'), but source fun
ction had ()
To silence this warning, decorate the function with @tf.autograph.experiment
al.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_predict_functio
n.<locals>.predict_function at 0x00000269F5DFFD30> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verb
osity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full
output.
Cause: closure mismatch, requested ('self', 'step_function'), but source fun
ction had ()
To silence this warning, decorate the function with @tf.autograph.experiment
al.do_not_convert
Results for Categorical Model
0.9560975609756097
              precision    recall  f1-score   support

           0       0.95      0.96      0.96       100
           1       0.96      0.95      0.96       105

    accuracy                           0.96       205
   macro avg       0.96      0.96      0.96       205
weighted avg       0.96      0.96      0.96       205
```

7

```
# generate classification report using predictions for binary model
from sklearn.metrics import classification_report, accuracy_score
# generate classification report using predictions for binary model
binary_pred = np.round(binary_model.predict(X_test)).astype(int)

print('Results for Binary Model')
print(accuracy_score(Y_test_binary, binary_pred))
print(classification_report(Y_test_binary, binary_pred))
```

```
WARNING:tensorflow:AutoGraph could not transform <function Model.make_predic
t_function.<locals>.predict_function at 0x00000269FA93D430> and will run it
as-is.
Please report this to the TensorFlow team. When filing the bug, set the verb
osity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full
output.
Cause: closure mismatch, requested ('self', 'step_function'), but source fun
ction had ()
To silence this warning, decorate the function with @tf.autograph.experiment
al.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_predict_functio
n.<locals>.predict_function at 0x00000269FA93D430> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verb
osity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full
output.
Cause: closure mismatch, requested ('self', 'step_function'), but source fun
ction had ()
To silence this warning, decorate the function with @tf.autograph.experiment
al.do_not_convert
Results for Binary Model
0.9463414634146341
              precision    recall  f1-score   support

           0       0.94      0.95      0.95       100
           1       0.95      0.94      0.95       105

    accuracy                           0.95       205
   macro avg       0.95      0.95      0.95       205
weighted avg       0.95      0.95      0.95       205
```

## 5.1 RESULT AND DISSCUSSION

Our proposed method first visualizes each data in the dataset. Since there are many duplicate and null values in the dataset, they all are then rectified. After that, iForest techniques is used to remove the outliers in the dataset by initializing anomaly score to each data. Basically,

Isolation Forests, like Random Forests, are most effective for removing an outlier. They are employed by making use of a sequence of decision trees. A CSO-CLSTM model is then built which includes RNN and back propagation neural network methods to achieve high accuracy and F1-score. This study also uses the isolation Forest (iForest) technique to reduce outliers. The CLSTM model's diagnostic outcomes are then greatly improved when CSO is used.

**Table No. – 1 Model Accuracy**

| | Model | Training Accuracy % | Testing Accuracy % |
|---|---|---|---|
| 1 | Logistic Regression | 86.79 | 86.81 |
| 2 | K-nearest neighbours | 86.79 | 86.81 |
| 3 | Support Vector Machine | 93.40 | 87.91 |
| 4 | Decision Tree Classifier | 100.00 | 78.02 |
| 5 | Random Forest Classifier | 100.00 | 82.42 |
| 6 | XGBoost Classifier | 100.00 | 82.42 |
| 7 | CSO-LSTM | 96.00 | 94.956 |

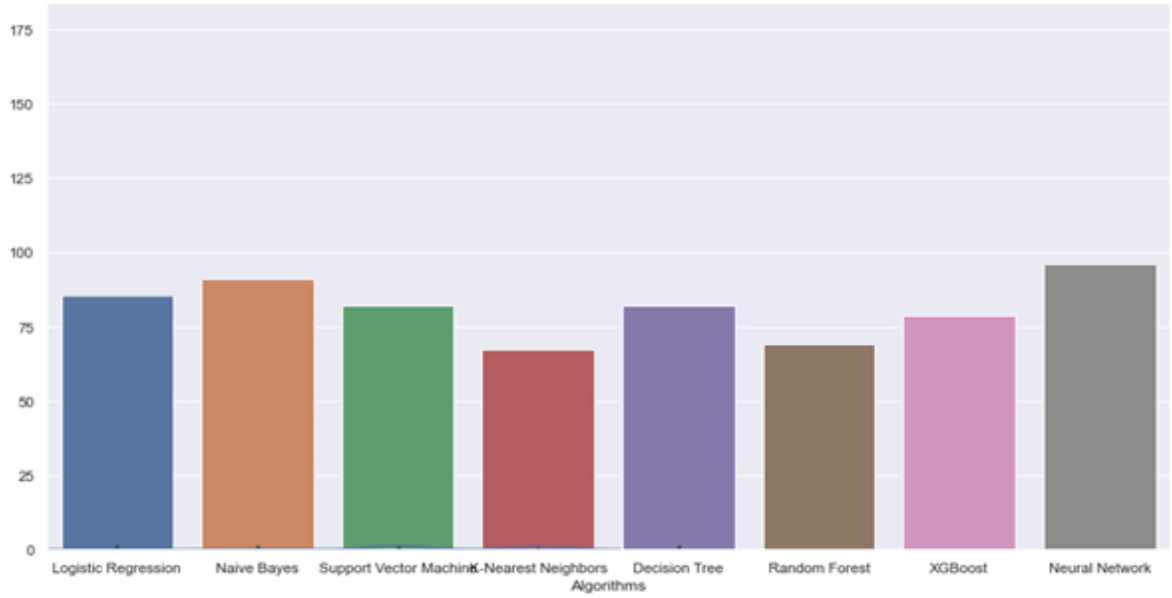


*Figure – 3.1 Model Accuracy for heart disease*

**Table No. – 2 Model Accuracy for Diabetes**

| | Model | Average Accuracy | Test Accuracy |
|---|---|---|---|
| 1 | Logistic Regression | 0.7502 | 0.8667 |
| 2 | Support Vector Machine | 0.7387 | 0.84 |
| 3 | Decision Tree Classifier | 0.6623 | 0.74 |
| 3 | Random Forest Classifier | 0.7468 | 0.8733 |

Data collecting, pre-processing, classification, and parameter tweaking are all stages of the described model. IoT based devices like smart wearables, sensors with the help of approaches from AI collect data to detect and help in the diagnosis of the diseases. Models like KNN, Logistic Regression, SVM, and Random Forest achieved low accuracy scores of 86.81 percent, 86.81 percent, 87.91 percent, and 82.42 percent, respectively, under 2000 instances. After that, from the patient data, the outliers are removed using iForest approach. The CSO-CLSTM along with RNN and other neural networks is used to increase the accuracy and determine whether or not the sickness occurs. In addition, CSO will be used to optimize the CLSTM model's weights along with the bias parameters. The use of Crow Search aids in the enhancement of the CLSTM model's diagnostic outcome. The performance of the CSO LSTM model was validated using healthcare data. These steps should result in a high level of accuracy and sensitivity of more than 90%.

# Summary.

Several trials have been undertaken in the past to build systems that detect significant adverse events and accidents by identifying physical activity flexibility and health markers. Mustlag first employed the WBSN to track a user's heart rate and movement whenever they required it, even if they were a long distance away. When big changes (early fall prediction, tachycardia, or bradycardia) occur in this study, the Edge node is connected to the Internet and sends notifications (through mobile phones) to family members. In line with this, Villarrubia has proposed using basic electrocardiogram (ECG) information to remotely monitor a patient's heart rate. In the literature, Emotion aware Connected Healthcare big data was converted to 5G and the Emotion aware Connected Healthcare model was built using an efficient emotion detection engine. A variety of IoT devices were used in this study to collect audio and visual inputs from patients in smart homes.

The Kaur and Jasuja developers investigated how to leverage the Bluemix cloud technology to record physiological data and allow clinicians to view it remotely. The simulation results are inspected and analysed in the IBM Watson IoT environment. Alwan and Rao conducted a case study for fever analysis using an integrated system that monitors patient health data on a regular basis. Satija et al. proposed IoT-based ECG telemetry in real time Researchers have demonstrated the efficiency of a model based on multiple activities of this task. Static monitoring eliminates the need to use domain sensors to collect contextual data and run multimodal processes. Phametal. Next, we developed the concept of combining ecological sensors, Opti-Track cameras, and smartwatch-based sensors with specific wearables to collect video, image, and audio data for physiological data collection. An innovative smart healthcare model has been proposed in the literature. This included pathological diagnosis techniques based on deep learning. Patient EEG measurements can be used to identify pathogens in this sense, a smart Electroencephalogram headset collects EEG signals and transmits them to a computer server on the mobile edge. The signal is pre-processed by the server before it is sent to the cloud sever.

# References


**Journals**

[1] A. A. Abdellatif, A. Mohamed, C. F. Chiasserini, M. Tlili, and A. Erbad,`` Edge computing for smart health: Context-aware approaches, opportunities, and challenges," IEEE Netw., vol.33, no. 3, pp. 196_203, May 2019.

[2] F. F. Gong, X. Z. Sun, J. Lin, and X. D. Gu, ``Primary exploration in establishment of China's intelligent medical treatment," (in Chinese), Mod. Hos Manag., vol. 11, no. 2, pp. 28_29, 2013.

[3] A. A. Mutlag, M. K. Abd Ghani, N. Arunkumar, M. A. Mohammed, and O. Mohd, ``Enabling technologies for fog computing in healthcare IoT systems," Future Genre. Compute. Syst., vol. 90, pp. 62_78, Jan. 2019.

[4] K. H. Abdulkareem, M. A. Mohammed, A. Salim, M. Arif, O. Geman, D. Gupta, and A. Khanna, ``Realizing an effective COVID-19 diagnosis system based on machine learning and IOT in smart hospital environment," IEEE Internet Things J., early access, Jan. 11, 2021, doi: 10.1109/JIOT.2021.3050775.

[5] A. A. Mutlag, M. K. A. Ghani, M. A. Mohammed, M. S. Maashi, O. Mohd, S. A. Mostafa, K. H. Abdulkareem, G. Marques, and I. de la Torre Díez, ``MAFC: Multi-agent fog computing model for healthcare critical tasks management," Sensors, vol. 20, no. 7, p. 1853, Mar. 2020.

**Web Links**

[1] *Heart Disease Data Set*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/heart+disease

[2] *Pima Indians Diabetes Database*. [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database