A Project Report
on

PREDICTION ANALYSIS USING SUPPORT VECTOR MACHINE IN CARDIOVASCULAR AILMENTS

Submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SICENCE AND ENGINEERING

by

17WH1A0518	Ms. O.VEDA VARSHITHA
17WH1A0523	Ms. G.NAGALAXMI PRASANNA
18WH5A0504	Ms. P.ANJALI

Under the esteemed guidance of

Ms. M SHANMUGA SUNDARI
Assistant Professor

Department of Computer Science and Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

June, 2021

DECLARATION

We hereby declare that the work presented in this project entitled **“PREDICTION ANALYSIS USING SUPPORT VECTOR MACHINE IN CARDIOVASCULAR AILMENTS”** submitted towards completion of Project Work in IV year of B.Tech., CSE at ‘BVRIT HYDERABAD College of Engineering For Women’, Hyderabad is an authentic record of our original work carried out under the guidance of Ms. M. Shanmuga sundari, Assistant Professor, Department of CSE.

Sign. with date:
Ms. O.VEDA VARSHITHA
(17WH1A0518)

Sign. with date:
Ms. G.NAGALAXMI PRASANNA
(17WH1A0523)

Sign. with date:
Ms. P.ANJALI
(18WH5A0504)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**PREDICTION ANALYSIS USING SUPPORT VECTOR MACHINE IN CARDIOVASCULAR AILMENTS**" is a bonafide work carried out by Ms. O.VEDA VARSHITHA (17WH1A0518); Ms. G.NAGALAXMI PRASANNA(17WH1A0523); Ms. P.ANJALI (18WH5A0504) in the partial fulfillment for the award of B.Tech. Degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr. K. Srinivasa Reddy	**Ms. M.Shanmuga Sundari**
Professor and HoD,	**Assistant Professor**
Department of CSE	

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K.Srinivasa Reddy, Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. M.Shanmuga Sundari, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. O.VEDA VARSHITHA

(17WH1A0518)

Ms. G.NAGALAXMI PRASANNA

(17WH1A0523)

Ms. P.ANJALI

(18WH5A0504)

Contents

ABSTRACT

Cardiovascular diseases are the most common leading cause of death for both men and women over the last few years. There are several factors for causing heart disease such as unhealthy diet, lack of physical exercise, smoking and so on. Some of the risk factors that cannot be determined such as family background and age factor. Deaths due to heart disease have become very common.

In order to lower the number of deaths from heart diseases, there has to be a fast and efficient detection technique and medical professionals working in the field of heart disease have their limitations, they cannot predict the chance of getting heart disease up to high accuracy. So, the objective of this project is to improve Heart Disease prediction accuracy and predict the 10 years risk of coronary heart disease, based on the data available. The main purpose is to develop a heart disease prediction system that predicts whether a patient will suffer from heart disease or not.

Machine Learning models Support Vector Machines (SVM), stochastic gradient descent and Logistic Regression algorithms are used for Heart disease predictions and comparative analysis will be done to get the model with the best accuracy.

LIST OF FIGURES

1. INTRODUCTION

1.1 Objective

The problem is to predict whether patients have heart disease by giving some features of users. This is important to medical fields. Human heart is the principal part of the human body. Basically, it regulates blood flow throughout our body. Any irregularity to the heart can cause distress in other parts of the body. Any sort of disturbance to normal functioning of the heart can be classified as a Heart disease. When a patient without a heart disease is diagnosed with heart disease, he will fall into unnecessary panic and when a patient with heart disease is not diagnosed with heart disease, he will miss the best chance to cure his disease. Such wrong diagnosis is painful to both patients and hospitals.

With accurate predictions, we can solve the unnecessary trouble. Besides, if we can apply machine learning tools into medical prediction; this will save human resources because complicated diagnosis processes are unnecessary in hospitals. (Though it is a very long way to go.).

Machine Learning (ML) which is a subfield of data mining handles large scale well-formatted datasets efficiently. In the medical field, machine learning can be used for diagnosis, detection and prediction of various diseases. The main goal of this project is to provide a tool for doctors to detect heart disease at an early stage. This in turn will help to provide effective treatment to patients and avoid severe consequences. ML plays a very important role to detect the hidden discrete patterns and thereby analyze the given data. After analysis of data ML techniques help in heart disease prediction and early diagnosis. The input to the algorithm is 16 features with number values. Logistic Regression, SVM, Stochastic gradient descent algorithms are to output a binary number 1 or 0. 1 indicates the patient has heart disease and vice versa.

1.2 Methodology

1.2.1 Proposed System:

The proposed work makes an attempt to detect these heart diseases at an early stage to avoid disastrous consequences. Records of large sets of medical data created by medical experts are available for analyzing and extracting valuable knowledge from it. Data mining techniques are the means of extracting valuable and hidden information from the large amount of data available. Mostly the medical database consists of discrete information. Hence, decision making using discrete data becomes a complex and tough task.

1.2.2 Organization of Project:

The technique which is developed is taking input as the details of the patient and compares the details from the dataset using classification algorithms. After the prediction of input details, it gives the output as to whether a patient has heart disease or not.

It has two modules in the project.

- Gathering the details of the patient.
- Heart Disease prediction.

1.2.3 Dataset:

The dataset for the Prediction is downloaded from kaggle website, and it is from an ongoing cardiovascular research which is taking place on the residents of Framingham city in Massachusetts, USA. The goal is to predict the 10-yrs risk of coronary heart disease, based on the data we have. The Framingham dataset provides us with the patient’s information, with 4240 records and 16 attributes/columns. The data set is in CSV (Comma Separated Value) format which is further prepared to data frame as supported by pandas library in python.

The information provided by the dataset is:

Demographic:

• Sex: male or female (Nominal)

• Age: Age of the patient ;(Continuous - Although the recorded ages have been truncated to whole numbers, the concept of age is continuous)

Behavioral:

• Current Smoker: whether or not the patient is a current smoker (Nominal)

• Cigs Per Day: the number of cigarettes that the person smoked on average in one day.(can be considered continuous as one can have any number of cigarettes, even half a cigarette.)

Medical (History):

• BP Meds: whether or not the patient was on blood pressure medication (Nominal)

• Prevalent Stroke: whether or not the patient had previously had a stroke (Nominal)

• Prevalent Hyp: whether or not the patient was hypertensive (Nominal)

• Diabetes: whether or not the patient had diabetes (Nominal)

Medical (current):

• TotChol: total cholesterol level (Continuous)

• SysBP: systolic blood pressure (Continuous)

• DiaBP: diastolic blood pressure (Continuous)

• BMI: Body Mass Index (Continuous)

• Heart Rate: heart rate (Continuous - In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of a large number of possible values.)

• Glucose: glucose level (Continuous)

Predict variable (desired target):

• 10 year risk of coronary heart disease CHD (binary: “1”, means “Yes”, “0” means “No”)

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirements Gathering:

2.1.1 Software Requirements:

Programming Language	:	Python3.6
User Interface	:	Tkinter
Packages	:	Numpy, Panda, Matplotlib, Scikit-learn
IDE	:	PyCharm
Tool	:	Google Colab

2.1.2 Hardware Requirements:

Operating system	:	Windows 10
Processor	:	Intel Core i5
Memory (RAM)	:	8GB
Storage	:	1TB

2.1.3 Technologies Description:

PYTHON:

Python is an interpreted high level programming language for general purposes programming. Created by Guido van Rossum and first released in 1991, python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of the other languages can pick up basic python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time and several of them have a later been patched and updated by people with no Python Background- without breaking.

Modules Used in Python:

1. Numpy:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

2. Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data mugging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

3. Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts; scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

4. Scikit – learn:

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis
- Extensions or modules for SciPy care conventionally named SciKits. As such, the module

5. Training, Validation, and Test Sets:

- Splitting your dataset is essential for an unbiased evaluation of prediction performance. In most cases, it’s enough to split your dataset randomly into three subsets:
- The training set is applied to train, or fit, your model. For example, you use the training set to find the optimal weights, or coefficients, for linear regression, logistic regression, or neural networks.
- The validation set is used for unbiased model evaluation during hyper parameter tuning. For example, when you want to find the optimal number of neurons in a neural network or the best kernel for a support vector machine, you experiment with different values. For each considered setting of hyper parameters, you fit the model with the training set and assess its performance with the validation set.
- The test set is needed for an unbiased evaluation of the final model. You shouldn’t use it for fitting or validation.

2.1.4 TOOLS:

1. PyCharm:

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License,and there is also Professional Edition with extra features – released under a proprietary license.

PyCharm provides an API so that developers can write their own plugins to extend PyCharm features. Several plugins from other JetBrains IDE also work with PyCharm. There are more than 1000 plugins which are compatible with PyCharm.

Features:

- Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes.
- Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages.
- Python refactoring: includes rename, extract method, introduce variable, introduce constant, pull up, push down and others.
- Support for web frameworks: Django, web2py and Flask [professional edition only.
- Integrated Python debugger.
- Integrated unit testing, with line-by-line code coverage.
- Google App Engine Python development.
- Version control integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with change lists and merge

- Support for scientific tools like matplotlib, numpy and scipy professional edition only.

It competes mainly with a number of other Python-oriented IDEs, including Eclipse's PyDev, and the more broadly focused Komodo IDE.

2. Google Colab:

Google Colab is the best project from Google Research. It is an open-source, Jupyter based environment. It helps us write and execute Python based code, other Python-based third-party tools and machine learning frameworks such as Python, PyTorch, Tensorflow, Keras, OpenCV and many others. It runs on the web-browser.

Google Colab is an open-source platform where we can write and Python execute code. It requires any source on the internet. We can also mount it into Google Drive. Google Colab requires no configuration to get started and provides free access to GPUs. It facilitates us to share live code, mathematical equations, data visualization, data cleaning and transformation, machine learning models, numerical simulations, and many others.

Advantages of Using Google Colab

- Google Colab provides many features. These features are given below.
- It comes with pre-installed packages.
- We don't need to install Python or its package.
- It runs on the web-browser.
- It allows us to work with web-browser Jupyter notebooks.
- It is an open-source Google platform, which means anyone can use it free of cost.
- It offers GPU and TPU power.
- It supports both Python versions 2 and 3.

- It provides two hardware accelerations - GPU (Graphical Processing Unit) and TPU (Tensor Processing Unit).
- It provides a free Jupyter notebook environment.

3. DESIGN

3.1 Introduction:

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

This structure shows how the prediction mechanism occurs in the model. First, the data set is considered. The dataset is checked to see if the null value remains in it. If the attribute value is omitted it can cause trouble (errors and incorrect accuracy). These can be removed using pre-processing, where the void is filled by the entire dataset of the attribute value. Then the method we want to try is used on it. We use the necessary methods and functions to keep the model adequate. Then when we use the dataset for training, this process is called classification. The model then gains knowledge through test data. Using this we estimate the output.

3.2 Architecture Diagram:

This structure shows how the prediction mechanism occurs in the model. First, the data set is considered. The dataset is checked to see if the null value remains in it. If the attribute value is omitted it can cause trouble (errors and incorrect accuracy). These can be removed using pre-processing, where the void is filled by the entire dataset of the attribute value. Then the method we want to try is used on it. We use the necessary methods and functions to keep the model adequate. Then when we use the dataset for training, this process is called classification. The model then gains knowledge through test data. Using this we estimate the output.

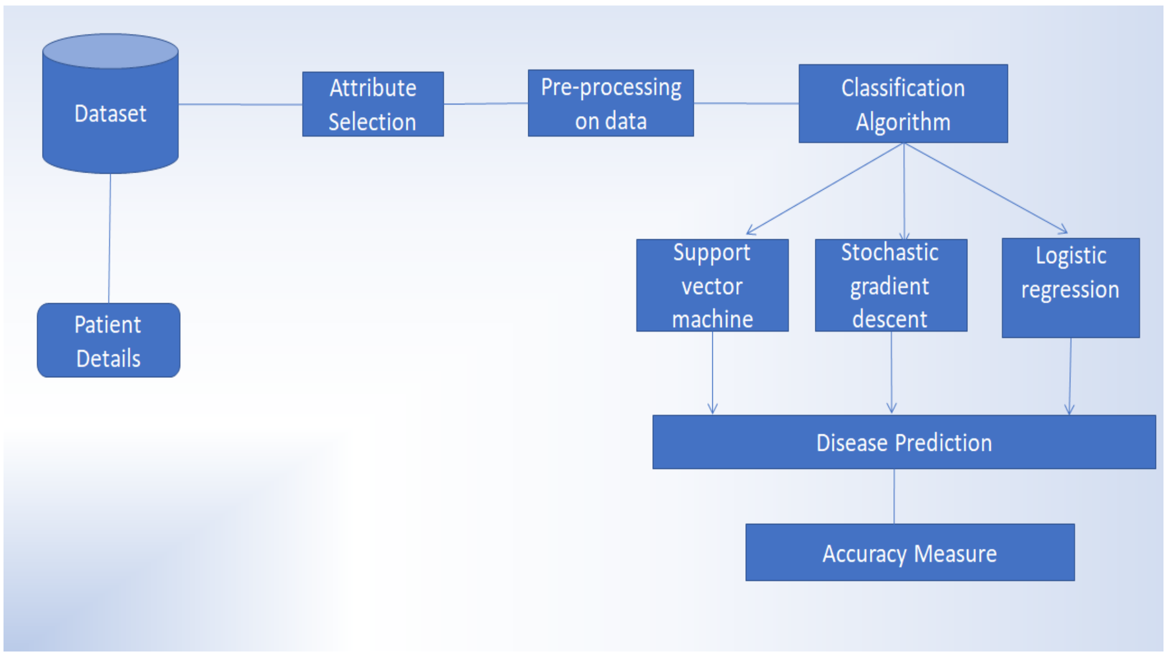


Figure 3.2 Architecture Diagram

3.3 UML Diagrams:

3.3.1 Use Case Diagram:

A use case is a technique used for the definition, description and arrangement of program parameters in program research. This consists of such a collection of potential series of experiences in such a specific context among systems as well as users which is linked to a specific target. The approach produces a guide detailing all of the actions a person needs to accomplish a task. These are usually developed by market experts and could be seen during many phases of product creation, including designing program specifications, reaffirming configuration, reviewing applications, as well as making a model for online support as well as users manuals.

One of the basic methods to show how the system works is to use the UML diagram. The client interacts with the system, they supply the queries required by the system and then send the output of the generated system. The client here is the patient. The questions asked are about the patient’s health statement, which can be useful if the patient does not have heart disease or does have heart disease.

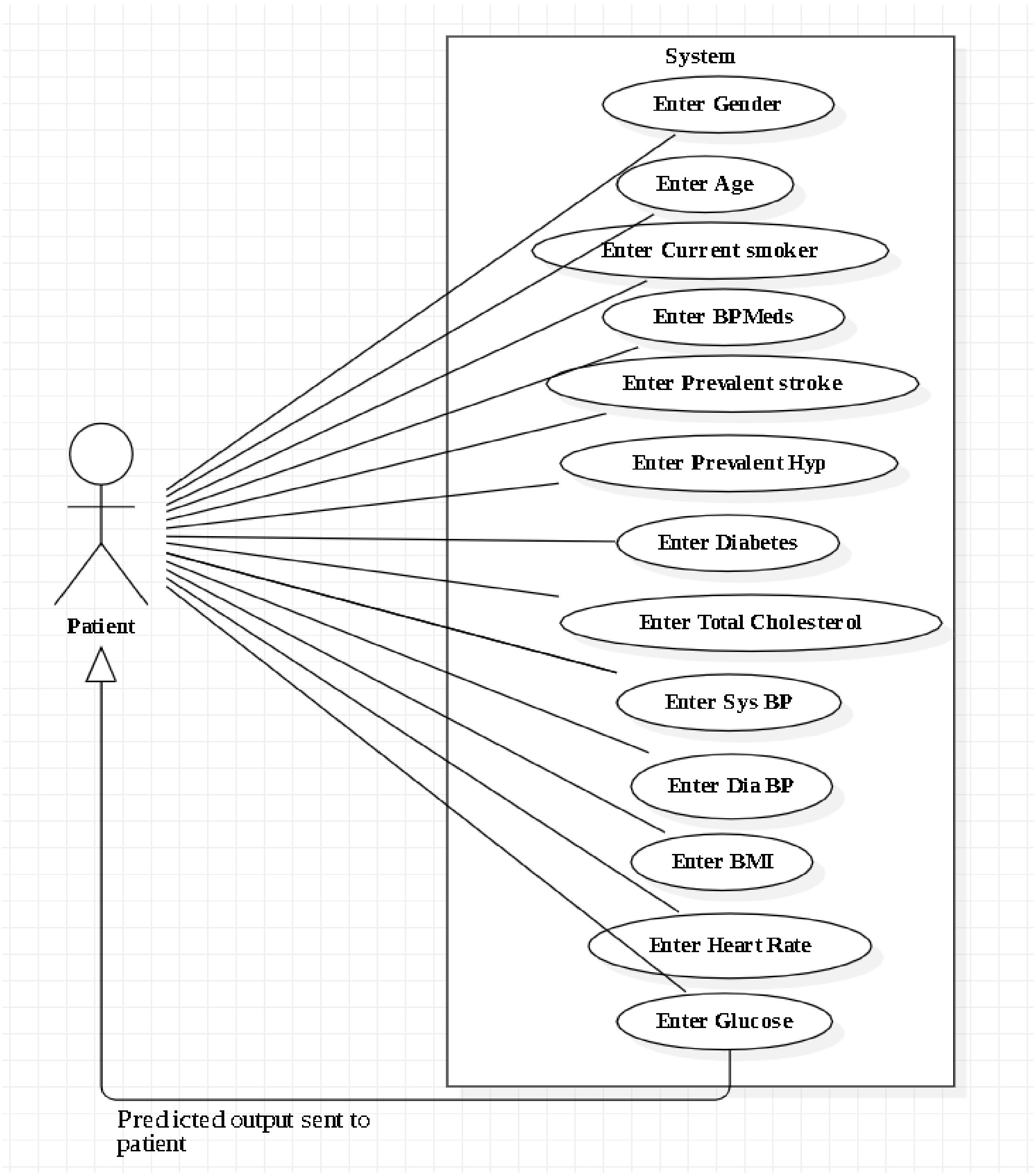


Figure 3.3.1 Use Case Diagram

3.3.2. Sequence Diagram:

Sequence Diagrams are graphs of relationships which describe how activities are done. Sequence Diagrams provide time-based and visually display the sequence of a relationship by utilizing the diagram’s vertical axis that depicts when messages are received and how often.

Here, the system interacts with the patient. It takes the necessary data from the user/patient, which supplies the system the necessary queries and then sends the output of the generated system. Questions about the patient’s health statement, which then predicts the patient has or does not have heart disease.

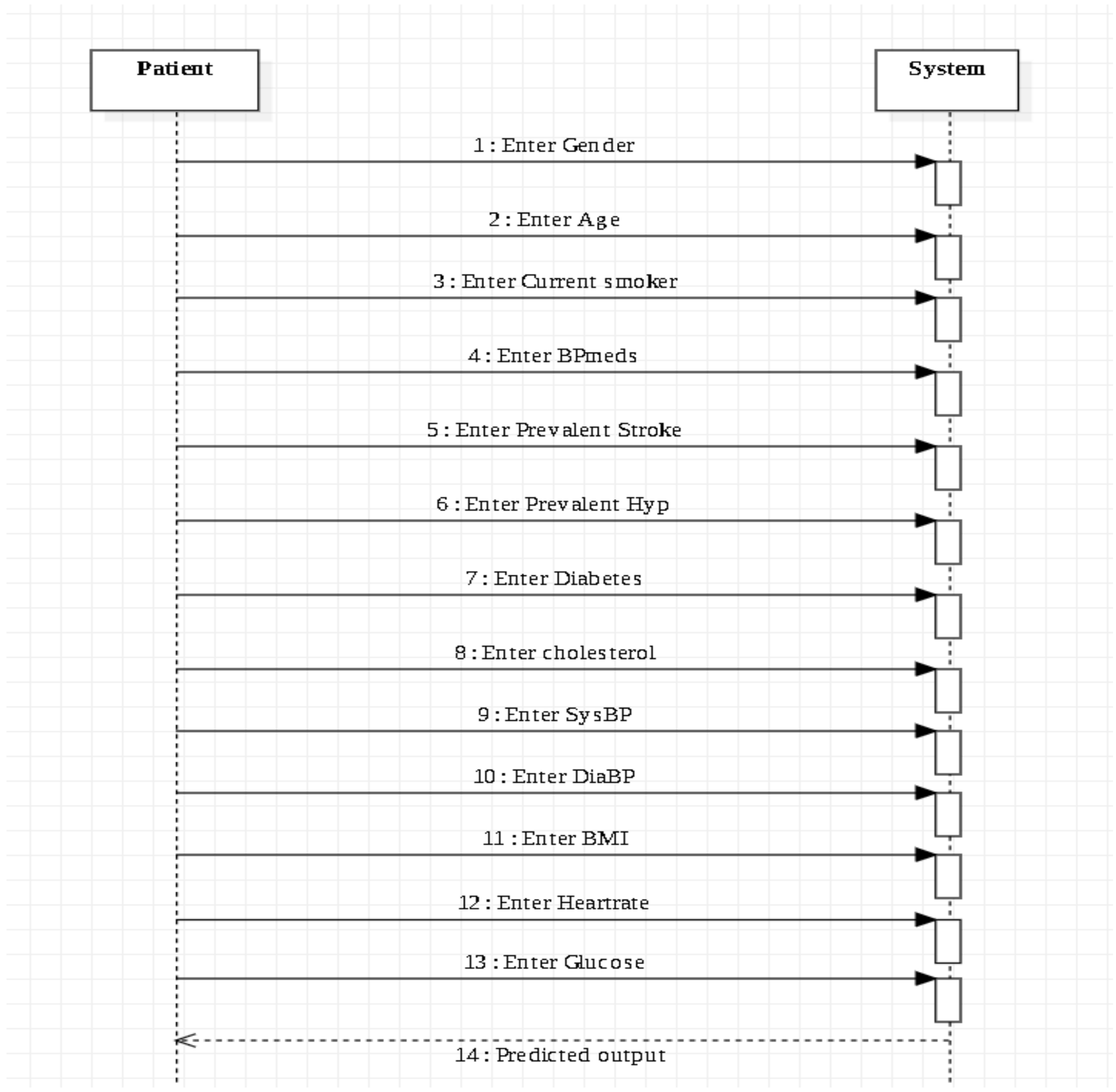


Figure 3.3.2 Sequence Diagram

3.3.3 Activity Diagram:

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

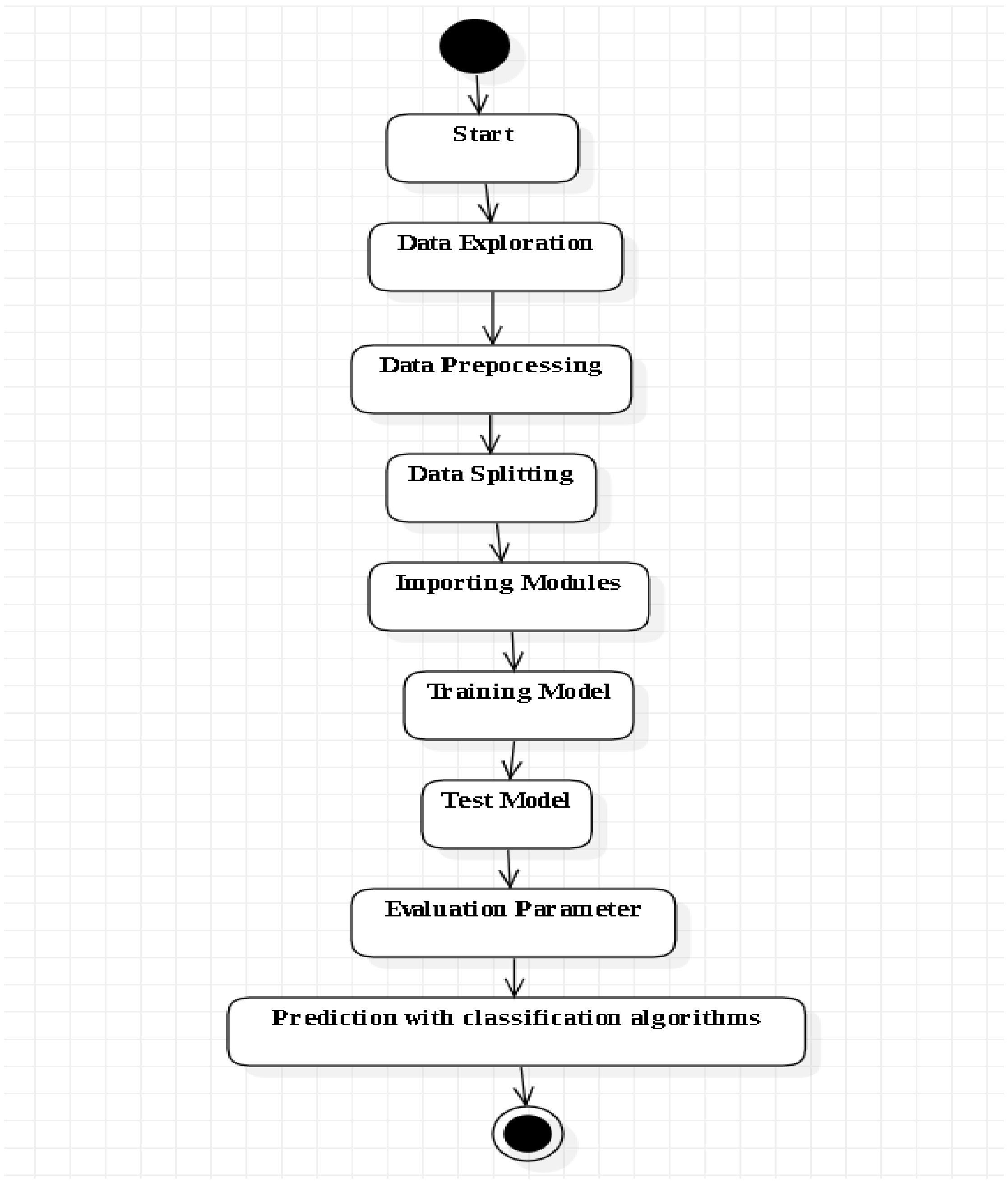


Figure 3.3.3 Activity Diagram

3.3.4 Class Diagram:

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

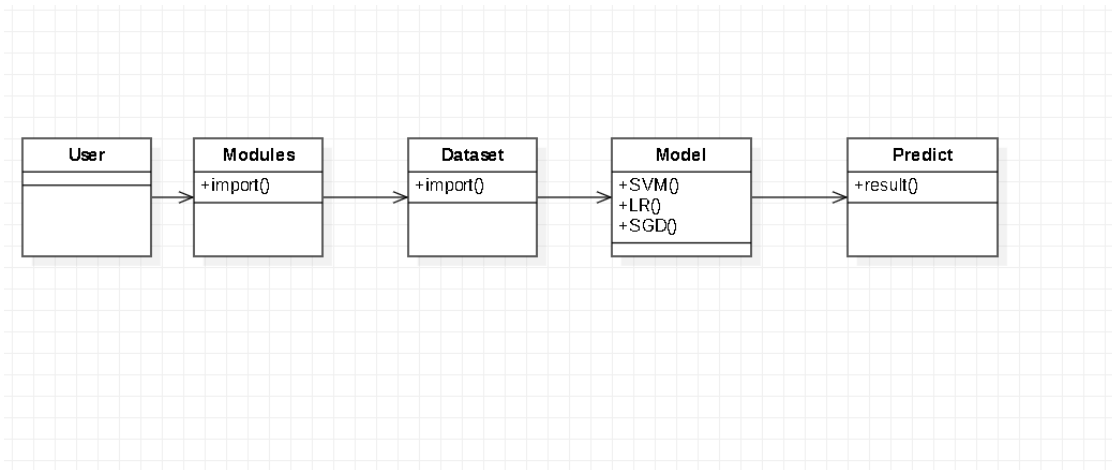
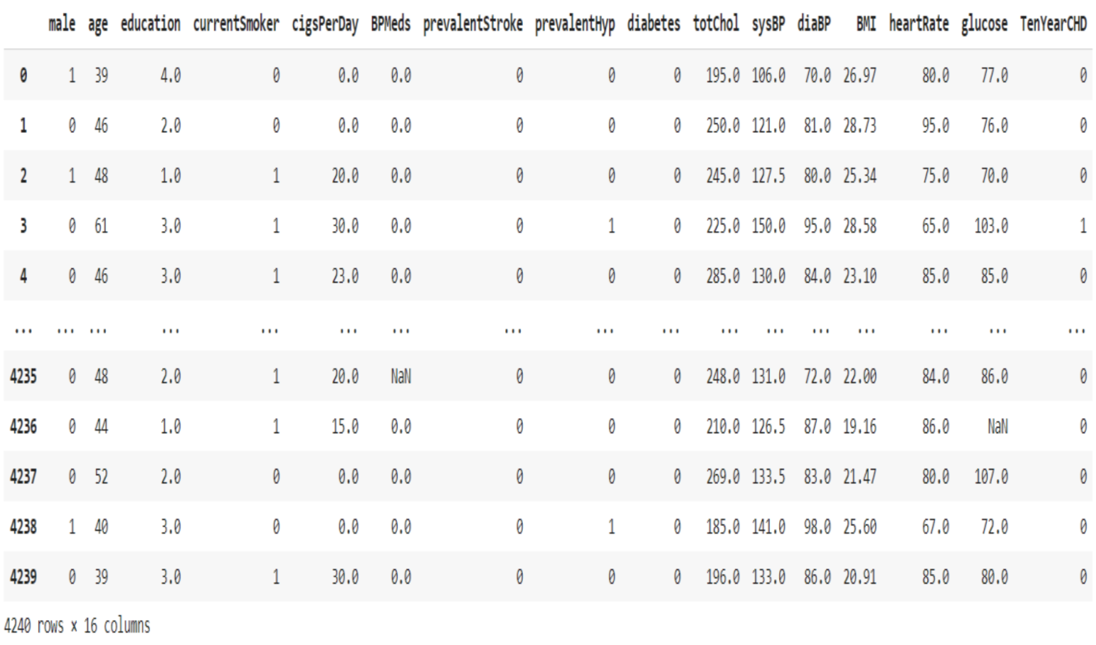


Figure 3.3.4 Class Diagram

4. IMPLEMENTATION

4.1 Coding:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.svm import SVC

from google.colab import drive
drive.mount('/content/drive')

data=pd.read_csv('/content/drive/MyDrive/framingham.csv')
data
```

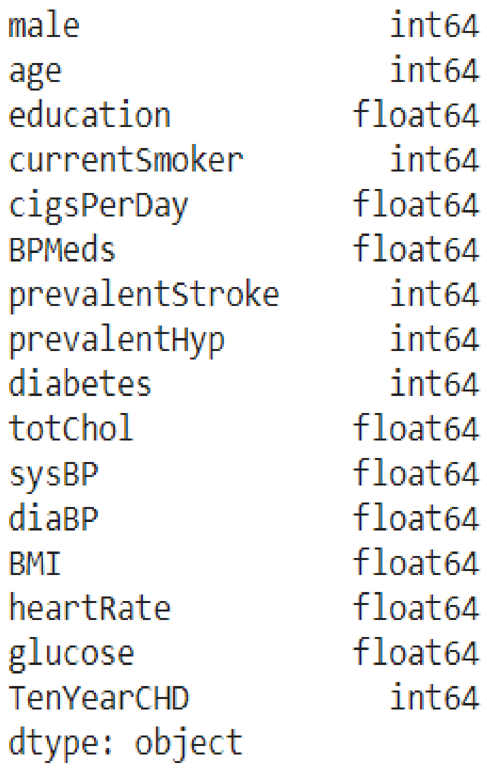


	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4235	0	48	2.0	1	20.0	NaN	0	0	0	248.0	131.0	72.0	22.00	84.0	86.0	0
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	86.0	NaN	0
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	80.0	107.0	0
4238	1	40	3.0	0	0.0	0.0	0	1	0	185.0	141.0	98.0	25.60	67.0	72.0	0
4239	0	39	3.0	1	30.0	0.0	0	0	0	196.0	133.0	86.0	20.91	85.0	80.0	0

4240 rows x 16 columns

Figure 4.1.1: Dataset

data.dtypes

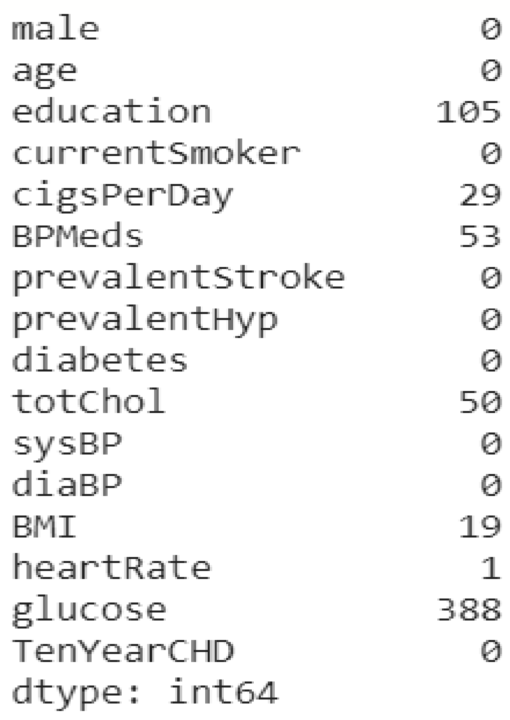


```
male                 int64
age                  int64
education          float64
currentSmoker        int64
cigsPerDay         float64
BPMeds             float64
prevalentStroke      int64
prevalentHyp         int64
diabetes             int64
totChol            float64
sysBP              float64
diaBP              float64
BMI                float64
heartRate          float64
glucose            float64
TenYearCHD           int64
dtype: object
```

Figure 4.1.2: Attributes Data type

#number of missing values per attribute in data

data.isnull().sum()



```
male                 0
age                  0
education          105
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
dtype: int64
```

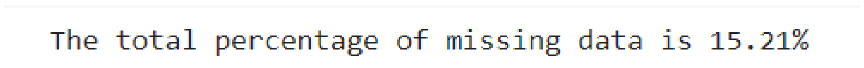
Figure 4.1.3: Missing values in data

#total percentage of missing data

missing_data = data.isnull().sum()

total_percentage = (missing_data.sum()/data.shape[0]) * 100

print(f'The total percentage of missing data is {round(total_percentage,2)}%')



```
The total percentage of missing data is 15.21%
```

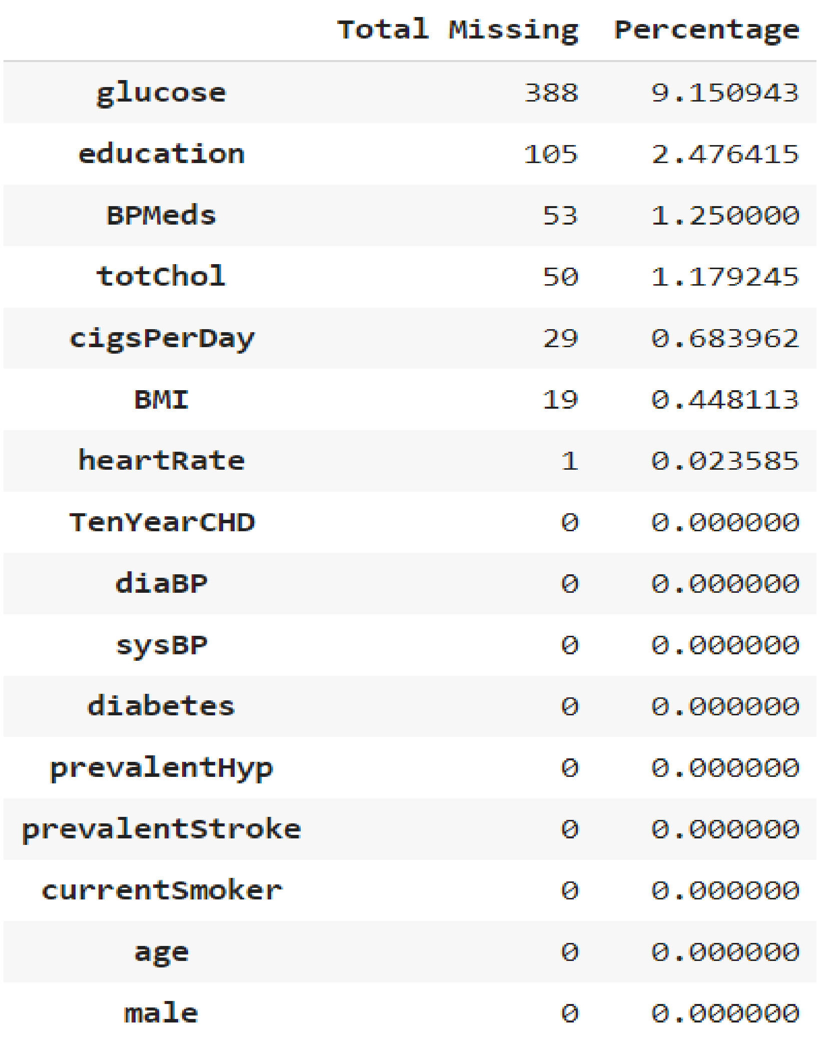
Figure 4.1.4: Total percentage of missing data

total = missing_data.sort_values(ascending=False)

percent_total = (missing_data/data.isnull().count()).sort_values(ascending=False)*100

missing = pd.concat([total, percent_total], axis=1, keys=["Total Missing", "Percentae]

)missing



	Total Missing	Percentage
glucose	388	9.150943
education	105	2.476415
BPMeds	53	1.250000
totChol	50	1.179245
cigsPerDay	29	0.683962
BMI	19	0.448113
heartRate	1	0.023585
TenYearCHD	0	0.000000
diaBP	0	0.000000
sysBP	0	0.000000
diabetes	0	0.000000
prevalentHyp	0	0.000000
prevalentStroke	0	0.000000
currentSmoker	0	0.000000
age	0	0.000000
male	0	0.000000

Figure 4.1.5: Missing data in descending order

```
plt.figure(figsize=(20,10))
sns.barplot(x=missing.index, y=missing['Percentage'], data = missing)
plt.title('Percentage of missing data by feature', fontsize=16)
plt.xlabel('Features', fontsize=16)
plt.ylabel('Percentage', fontsize=16)
plt.show()
```

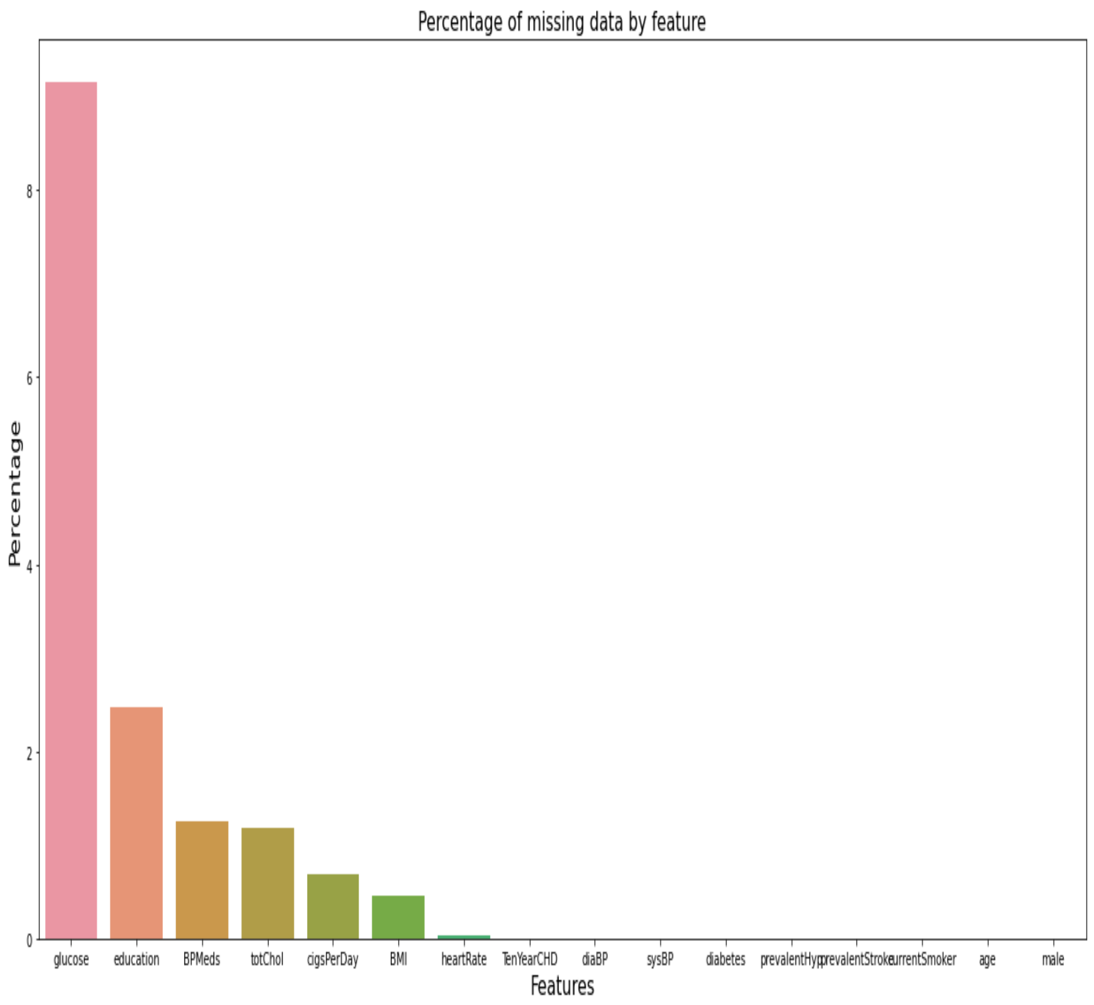


Figure 4.1.6: Missing data percentage in Graph

```
correlation_mat = data.corr()
plt.subplots(figsize=(20,15))
sns.heatmap(correlation_mat, annot = True, linewidths=1)

plt.show()
```

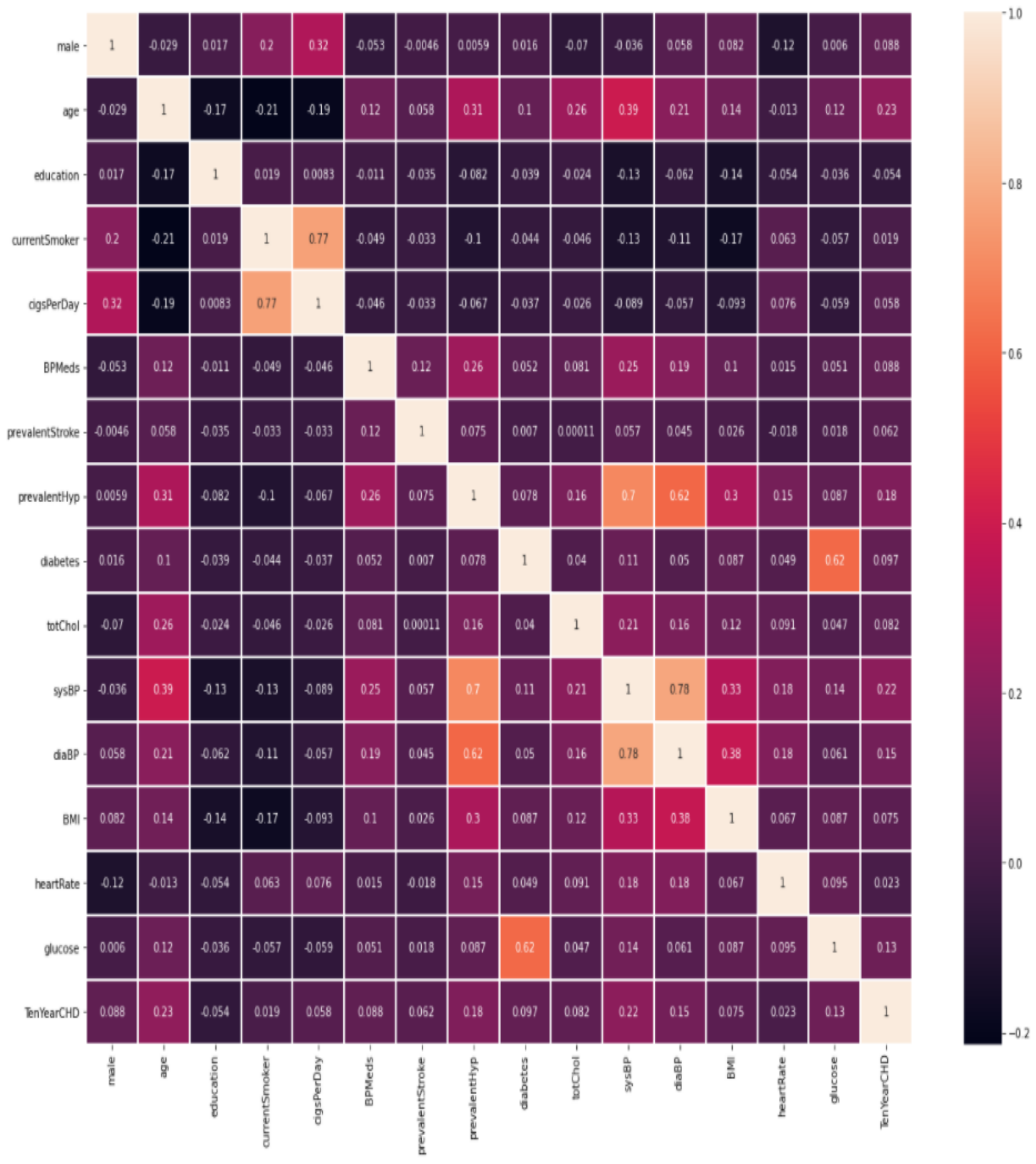


Figure 4.1.7 Correlation matrix between attributes

```
# education cannot have relation with CHD.
data.drop(['education'], axis=1, inplace=True)

for value in ['cigsPerDay','BPMeds', 'totChol','BMI','glucose','heartRate']:
 data[value].fillna(round(data[value].mean()), inplace= True)

# To check for outliers, we are plotting a box-whisker plot.
plt.figure(figsize=(20,10), facecolor='w')
sns.boxplot(data=data)
plt.show()
```

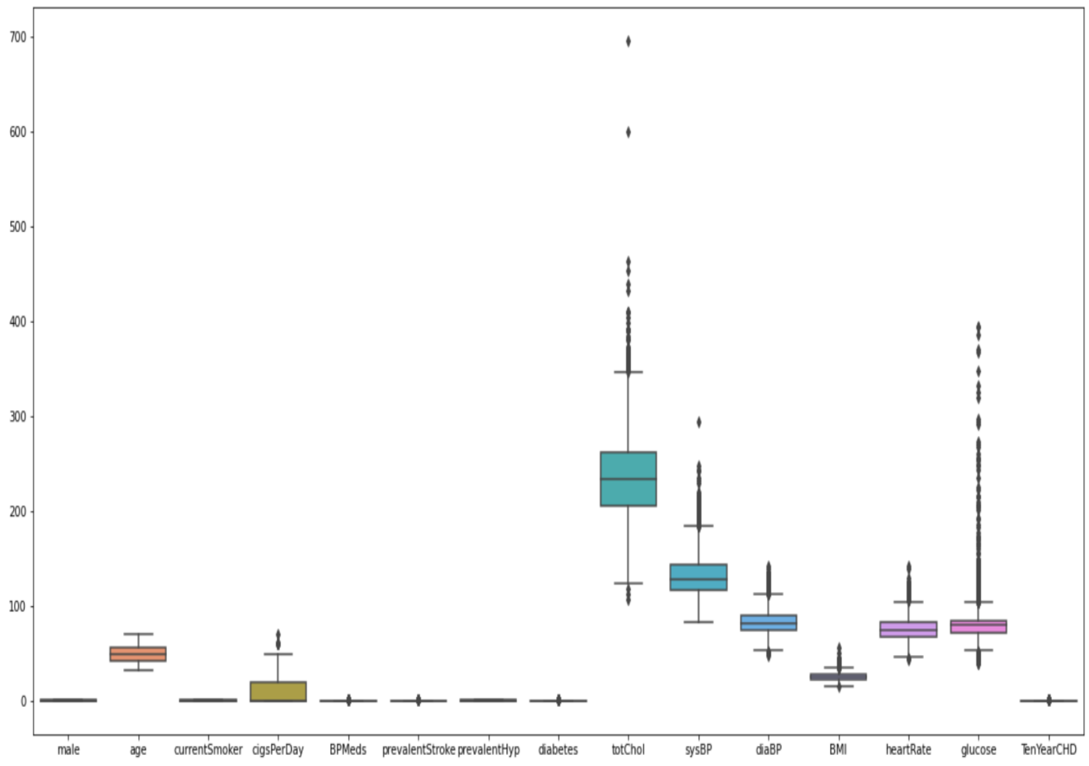


Figure 4.1.8 Box plot to check the outliers

```
data= data[data['totChol']<600.0]
data = data[data['sysBP']<295.0]
data.shape
```

```
(4237, 15)
```

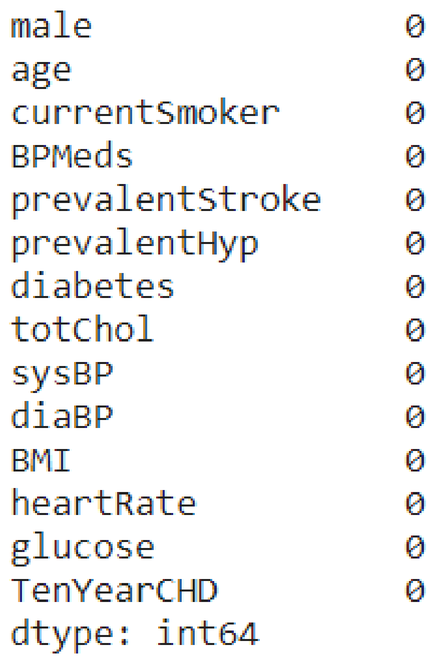
Figure 4.1.9: Size of the dataset

```
data=data.drop('cigsPerDay',axis=1)
data.shape
```

```
(4237, 14)
```

Figure 4.1.10: Size of the dataset after dropping

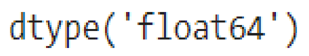
```
data.isnull().sum()
```



```
male               0
age                0
currentSmoker      0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

Figure 4.1.11: Missing values in attributes

```
X = data.values[:,0:-1]
Y = data.values[:,-1]

# Scaling The Data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
Y.dtype
```



```
dtype('float64')
```

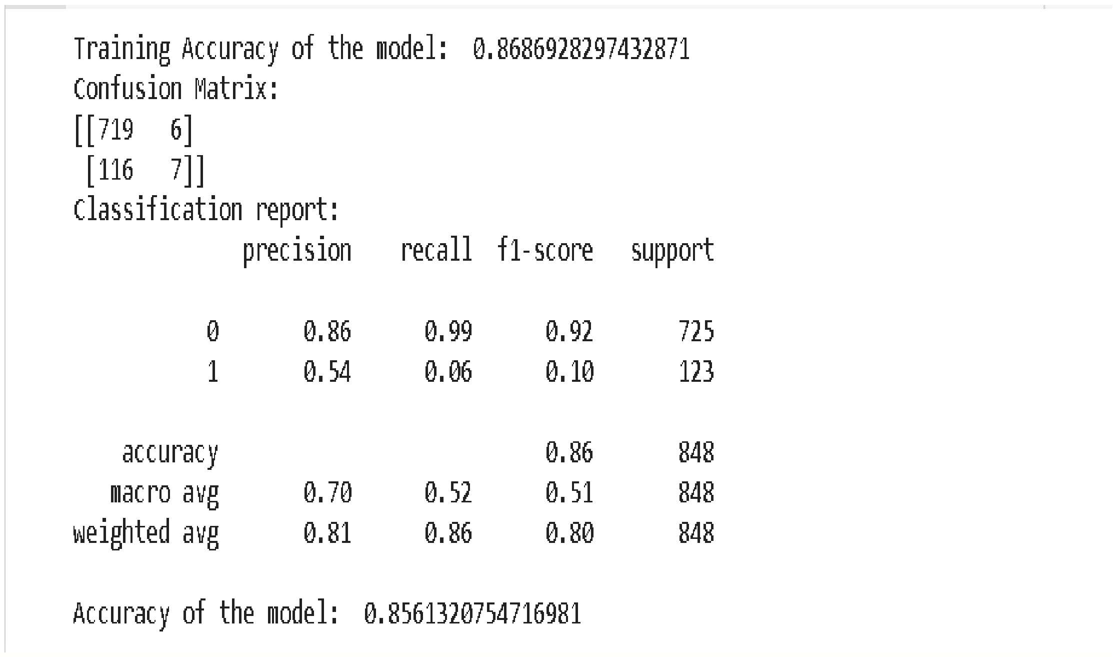
Figure 4.1.12: Data type of variable Y

```
Y= Y.astype(int)
```

```
# Splitting data to train and test
from sklearn.model_selection import train_test_split
#Split the data into test and train
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20,
random_state=27)
```

SVM Code:

```
from sklearn.svm import SVC

svc_model= SVC(kernel="rbf",gamma=0.01,C= 80)
svc_model.fit(X_train,Y_train)
Y_pred= svc_model.predict(X_test)

acc_SVr= svc_model.score(X_train,Y_train)
print("Training Accuracy of the model: ",acc_SVr)

#acc_SVr= accuracy_score(Y_test, Y_pred)
#print("Testing Accuracy of the model: ",acc_SVr)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

cfm_Svm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix: ")
print(cfm_Svm)


print("Classification report: ")

print(classification_report(Y_test,Y_pred))

acc_Svm=accuracy_score(Y_test, Y_pred)
print("Accuracy of the model: ",acc_Svm)
```



```
Training Accuracy of the model:  0.8686928297432871
Confusion Matrix:
[[719   6]
 [116   7]]
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       725
           1       0.54      0.06      0.10       123

    accuracy                           0.86       848
   macro avg       0.70      0.52      0.51       848
weighted avg       0.81      0.86      0.80       848

Accuracy of the model:  0.8561320754716981
```

Figure 4.1.13: Output of Support Vector Machine

Stochastic Gradient Descent:

from sklearn.linear_model import SGDClassifier

#create a model

classifier=SGDClassifier(loss="log",random_state=27,eta0=0.1,alpha=0.01,max_iter=1000)

#fitting training data to the model

classifier.fit(X_train,Y_train)

Y_pred=classifier.predict(X_test)

#Generating a confusion matrix, classification report and accuracy score to check agai n.

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

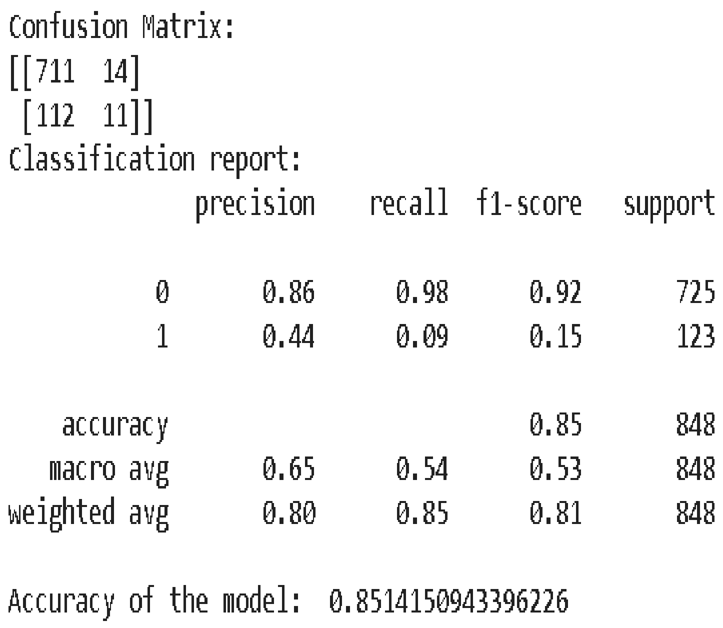
cfm_SGD=confusion_matrix(Y_test,Y_pred)

print("Confusion Matrix: ")

print(cfm_SGD)

print("Classification report: ")

```
print(classification_report(Y_test,Y_pred))

acc_SGD=accuracy_score(Y_test, Y_pred)
print("Accuracy of the model: ",acc_SGD)
```



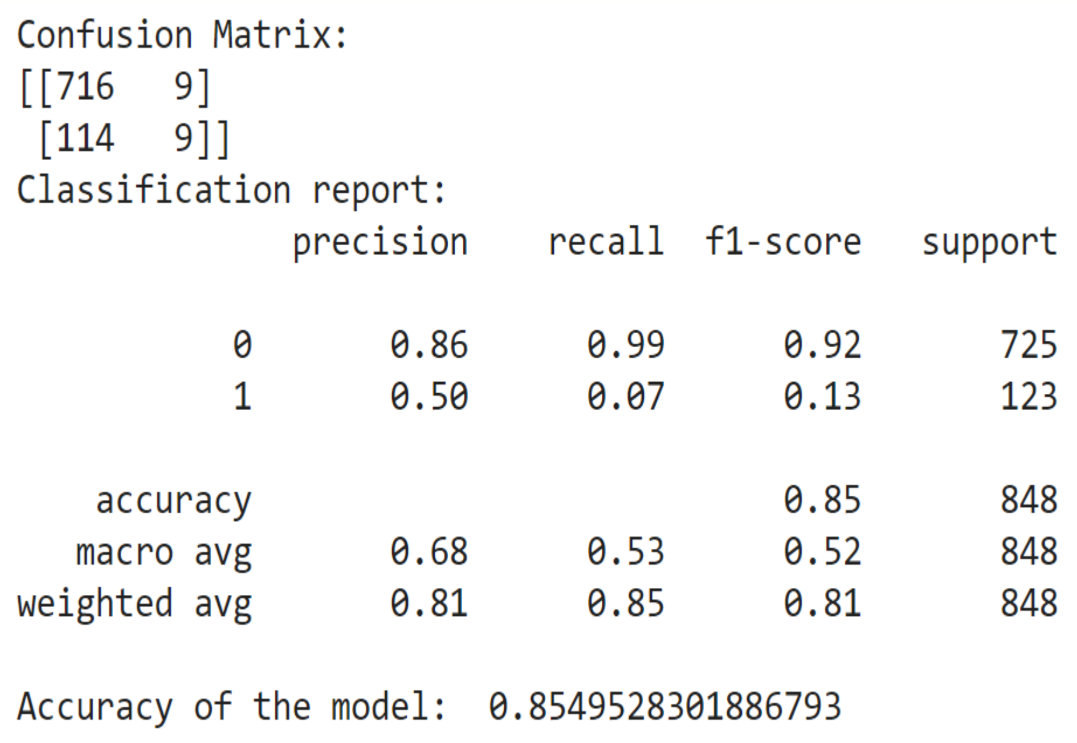
```
Confusion Matrix:
[[711  14]
 [112  11]]
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.98      0.92       725
           1       0.44      0.09      0.15       123

    accuracy                           0.85       848
   macro avg       0.65      0.54      0.53       848
weighted avg       0.80      0.85      0.81       848

Accuracy of the model:  0.8514150943396226
```

Figure 4.1.14: Output of Stochastic Gradient Descent

Logistic regression Code:

```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()

classifier.fit(X_train,Y_train)

Y_pred=classifier.predict(X_test)

#generating confusion matrix, accuracy score and classification report to check score of our prediction.

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

cfm_LR=confusion_matrix(Y_test,Y_pred)
```

```
print("Confusion Matrix:")
print(cfm_LR)

print("Classification report: ")

print(classification_report(Y_test,Y_pred))

acc_LR=accuracy_score(Y_test, Y_pred)
print("Accuracy of the model: ",acc_LR)
```



```
Confusion Matrix:
[[716   9]
 [114   9]]
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       725
           1       0.50      0.07      0.13       123

    accuracy                           0.85       848
   macro avg       0.68      0.53      0.52       848
weighted avg       0.81      0.85      0.81       848

Accuracy of the model:  0.8549528301886793
```

Figure 4.1.15: Output of Logistic Regression

4.2 Testing:

Testing is the process where the test data is prepared and it is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that is passed through all possible condition. The following is the description of the testing strategies, which were carried out during the testing period.

4.2.1 Testing Strategies:

System Testing:

Testing has become an integral part of my system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical, when the software is developed before it is given to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

Module testing:

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the System is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

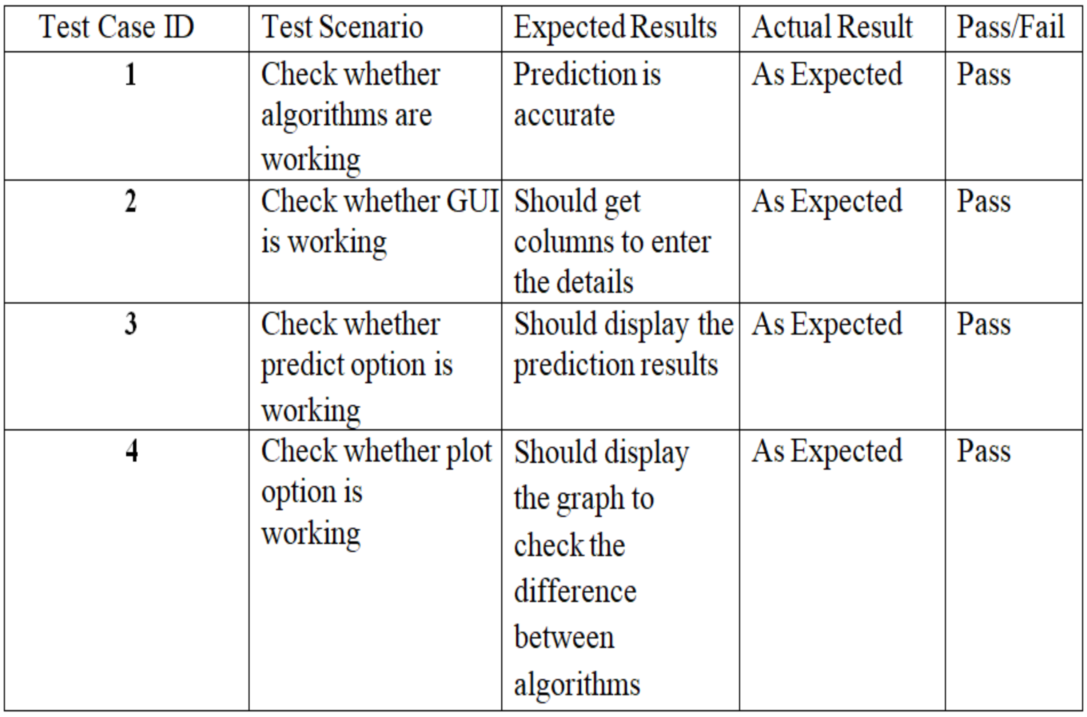
Integrated testing:

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

Acceptance testing:

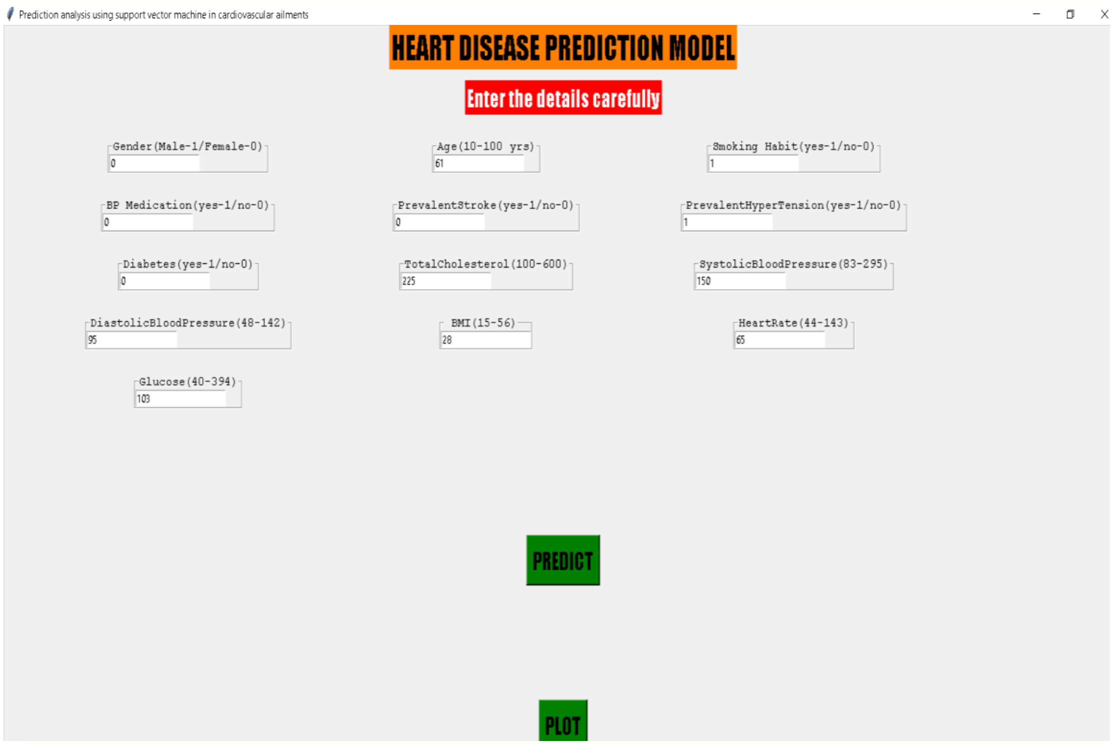
When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

4.3 Test Cases:



Test Case ID	Test Scenario	Expected Results	Actual Result	Pass/Fail
1	Check whether algorithms are working	Prediction is accurate	As Expected	Pass
2	Check whether GUI is working	Should get columns to enter the details	As Expected	Pass
3	Check whether predict option is working	Should display the prediction results	As Expected	Pass
4	Check whether plot option is working	Should display the graph to check the difference between algorithms	As Expected	Pass

4.4 Input Screenshots:

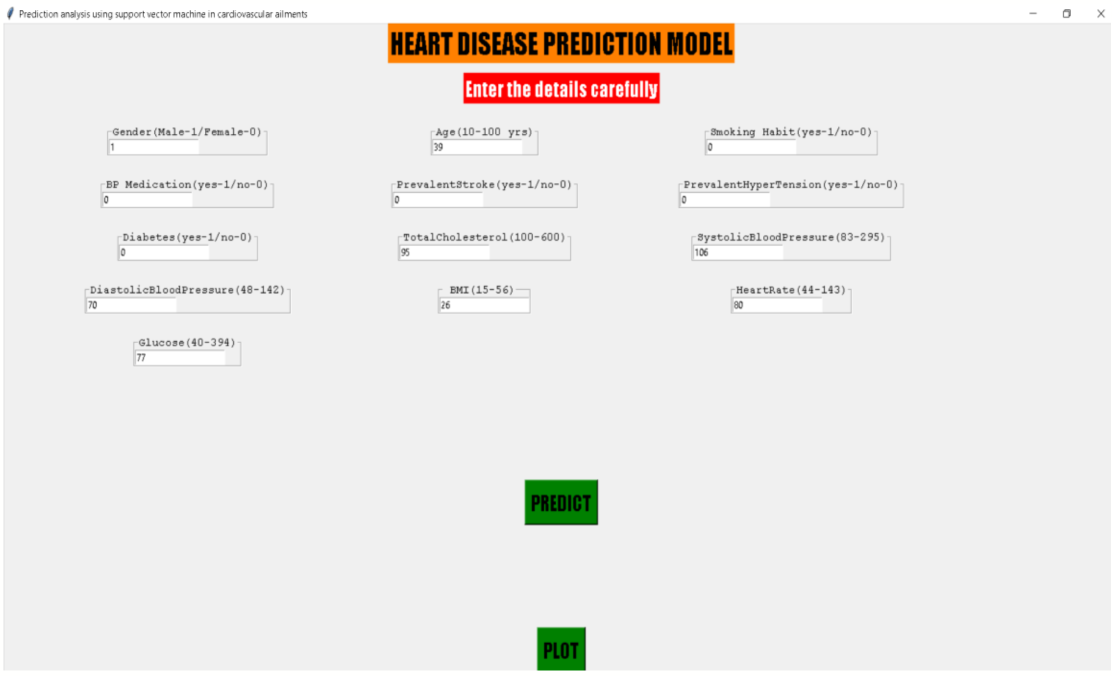

Prediction analysis using support vector machine in cardiovascular ailments

HEART DISEASE PREDICTION MODEL

Enter the details carefully

Gender(Male-1/Female-0): 0
Age(10-100 yrs): 61
Smoking Habit(yes-1/no-0): 1
BP Medication(yes-1/no-0): 0
PrevalentStroke(yes-1/no-0): 0
PrevalentHyperTension(yes-1/no-0): 1
Diabetes(yes-1/no-0): 0
TotalCholesterol(100-600): 225
SystolicBloodPressure(83-295): 150
DiastolicBloodPressure(48-142): 95
BMI(15-56): 28
HeartRate(44-143): 65
Glucose(40-394): 103

Figure 4.4.1: Input 1 for prediction of heart disease


Prediction analysis using support vector machine in cardiovascular ailments

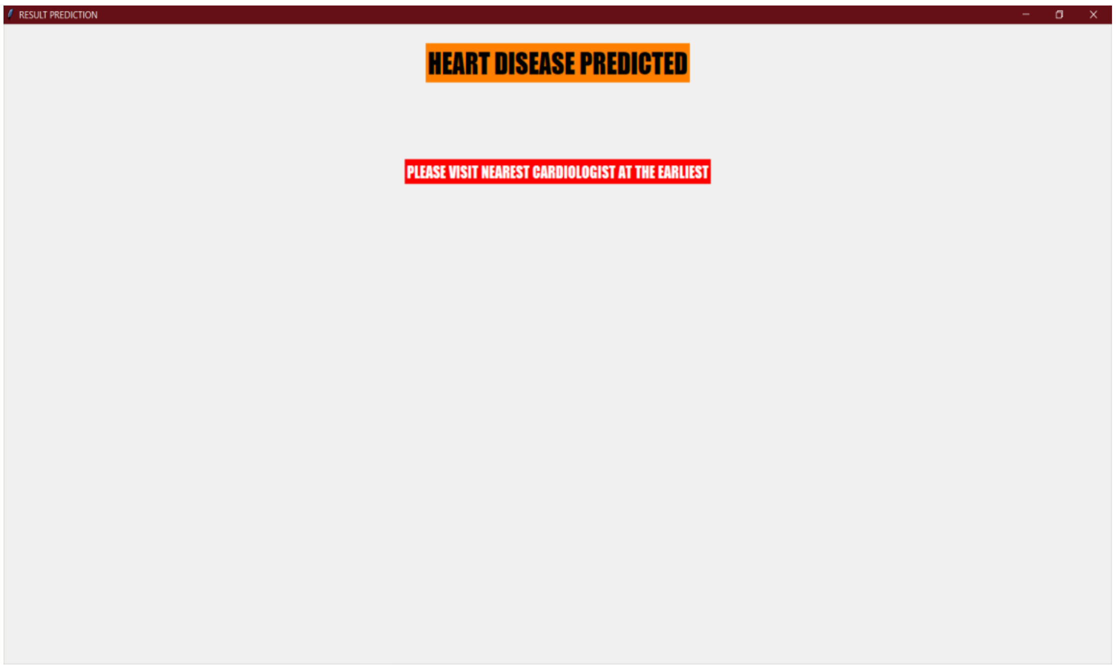
HEART DISEASE PREDICTION MODEL

Enter the details carefully

Gender(Male-1/Female-0): 1
Age(10-100 yrs): 39
Smoking Habit(yes-1/no-0): 0
BP Medication(yes-1/no-0): 0
PrevalentStroke(yes-1/no-0): 0
PrevalentHyperTension(yes-1/no-0): 0
Diabetes(yes-1/no-0): 0
TotalCholesterol(100-600): 95
SystolicBloodPressure(83-295): 106
DiastolicBloodPressure(48-142): 70
BMI(15-56): 26
HeartRate(44-143): 80
Glucose(40-394): 77

Figure 4.4.2: Input 2 for prediction of heart disease

4.5 Output Screenshots:

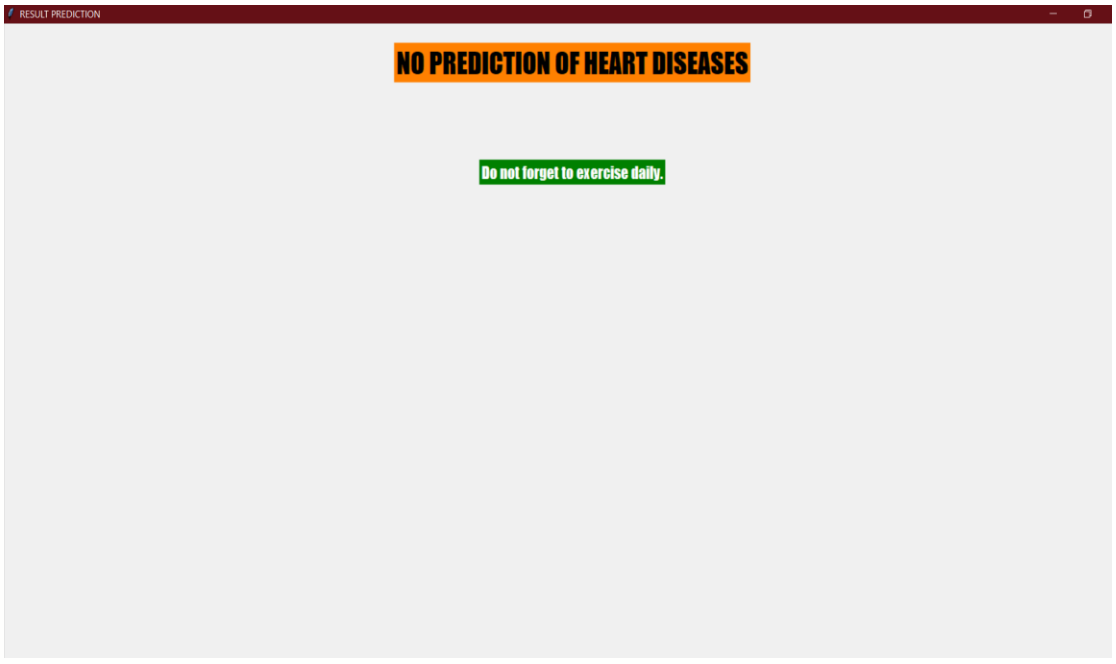


RESULT PREDICTION

HEART DISEASE PREDICTED

PLEASE VISIT NEAREST CARDIOLOGIST AT THE EARLIEST

Fig4.5.1: Output 1 for Prediction of Heart disease



RESULT PREDICTION

NO PREDICTION OF HEART DISEASES

Do not forget to exercise daily.

Figure 4.5.2: Output 2 for Prediction of Heart disease

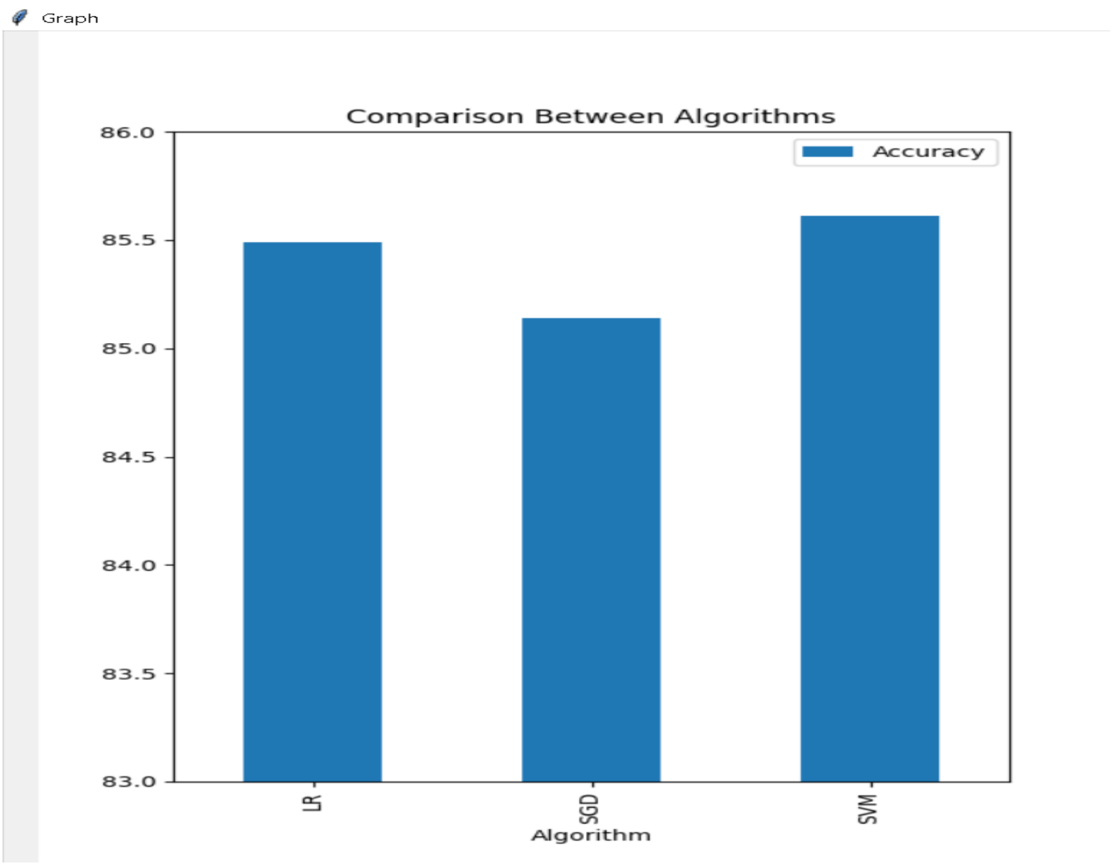


Figure 4.5.3: Output (comparison between algorithms)

4.6 Result:

- Among the three algorithms used, highest accuracy was achieved using Support vector machine, a supervised learning algorithm. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. Using this algorithm, an accuracy of 85.61% was achieved.

- Stochastic Gradient Descent is an optimization algorithm. Starting from an initial value, Gradient Descent runs iteratively to find the optimal values of the parameters to find the minimum possible value of the given cost function. The word ‘stochastic‘means a system or a process that is linked with a random probability. Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. The accuracy of 85.14% is achieved.

- Logistic Regression is a supervised classification algorithm. It is a predictive analysis algorithm based on the concept of probability. It measures the relationship between the dependent variable (TenyearCHD) and the one or more independent variables (risk factors) by estimating probabilities using the underlying logistic function (sigmoid function). Using this algorithm, an accuracy of 85.49% was achieved.

5. CONCLUSION AND FUTURE SCOPE

The prediction model for Cardio vascular diseases is a GUI based user friendly system. The proposed working model can also help in reducing treatment costs by providing initial diagnostics in time. The model can also serve the purpose of a training tool for medical students and will be a soft diagnostic tool available for physicians and cardiologists.

There are many possible improvements that could be explored to improve the scalability and accuracy of this prediction system. As a generalized system is developed, in future this system can be used for the analysis of different data sets. The performance of the health's diagnosis can be improved significantly by handling numerous class labels in the prediction process. The model can be further extended with a different dataset to be able to distinguish between different types of cardiovascular ailments.

6. REFERENCES

1. M. Saw, T. Saxena, S. Kaithwas, R. Yadav and N. Lal, "Estimation of Prediction for Getting Heart Disease Using Logistic Regression Model of Machine Learning," *2020 International Conference on Computer Communication and Informatics (ICCCI)*, 2020, pp. 1-6

2. Dhai Eddine Salhi, Abdelkamel Tari, M-Tahar Kechadi, International Conference on Computing Systems and Applications, CSA 2020: Advances in Computing Systems and Applications pp 70-81.

3. Galla Siva Sai Bindhika, Munaga Meghana(2020), "Heart Disease Prediction Using Machine Learning Techniques", (IRJET) , Volume: 07 Issue: 04.

4. " HEART DISEASE PREDICTION USING MACHINE LEARNING", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.7, Issue 6, page no.2081-2085, June-2020.

5. Kiranjit Kaur, Munish Saini, "HEART DISEASE PREDICTION USING MACHINE LEARNING TECHNIQUES: A SYSTEMATIC REVIEW", Vol – 15 No -5, May 2020

6. A. Lakshmanarao, Y.Swathi, P.Sri Sai Sundareswar, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 8, ISSUE 11, NOVEMBER 2019, Machine Learning Techniques For Heart Disease Prediction.

7. Sonam Nikhar, A.M. Karandikar “Prediction of Heart Disease Using Machine Learning Algorithms” in International Journal of Advanced Engineering, Management and Science (IJAEMS) June-2016 vol-2

8. T.Mythili, Dev Mukherji, Nikita Padaila and Abhiram Naidu, “A Heart Disease Prediction Model using SVM-Decision Trees- Logistic Regression (SDL)”, International Journal of Computer Applications, vol. 68, 16 April 2013

9. A. Singh and R. Kumar, "Heart Disease Prediction Using Machine Learning Algorithms," *2020 International Conference on Electrical and Electronics Engineering (ICE3)*, 2020, pp. 452-457

10. Jagtap, Abhijeet, Priya Malewadkar, Omkar Baswat, and Harshali Rambade. "Heart Disease Prediction using Machine Learning." *International Journal of Research in Engineering, Science and Management* **2**, no. 2 (2019): p1-5