

PROJECT CONTENTS

INDEX

CERTIFICATE DECLARATION ACKNOWLEDGEMENT

ABSTRACT 01

CHAPTER 1 INTRODUCTION 02-03

- 1.1 BRIEF INFORMATION
- 1.2 MOTIVATION
- 1.3 OBJECTIVE
- 1.4 PROBLEM STATEMENTS

CHAPTER 2 SYSTEM ANALYSIS 04-08

- 2.1 EXISTING SYSTEM
- 2.2 PROPOSED SYSTEM
- 2.3 FEASIBILITY STUDY
- 2.4 SYSTEM REQUIREMENT SPECIFICATION
- 2.5 HARDWARE REQUIREMENTS
- 2.6 SOFTWARE REQUIREMENTS

CHAPTER 3 SYSTEM DESIGN 09-19

- 3.1 SYSTEM ARCHITECTURE
- 3.2 MODULES
- 3.3 UML DIAGRAMS
- 3.4 DATABASE DESIGN

CHAPTER 4 SYSTEM IMPLEMENTATION 20-64

- 4.1 FRONT END IMPLEMENTATION
- 4.2 BACK END IMPLEMENTATION
- 4.3 SOURCE CODE
- 4.4 OUTPUT SCREENS

CHAPTER 5 SYSTEM TESTING 65-68

- 5.1 TESTING CONCEPTS
- 5.2 TESTING STRATEGIES
- 5.3 TEST CASES

CHAPTER 6 - 7 CONCLUSIONS & REFERENCES 69-71

Efficient Heart Disease Prediction System using Several Machine Learning classification Algorithms

ABSTRACT

Heart disease is one of the most significant reasons of mortality in the current days. Now a days it is very complicated task to predict the cardiovascular disease in the area of clinical data analysis. All the prediction is done by using manual approach which is becoming time complexity for the end users to find out the abnormalities. Hence this motivated me to design the proposed application in order to efficient heart disease prediction using machine learning (ML) algorithm. In current days ML has been shown to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry. This is mainly because of its usage in different areas especially in the medical field. In this proposed work, we propose a novel model which is specifically that aims at finding significant features by applying machine learning techniques resulting in improving the accuracy in the prediction of cardiovascular disease. This proposed application is trained by using several ML algorithms and then check the following factors such as accuracy, precision, recall and F1-Score. By conducting various experiments on several ML Algorithms by taking UCI dataset, we finally check which algorithm fits best for efficient heart disease prediction.

1. INTRODUCTION

1.1 BREIF INFORMATION

The principle goal of our paper is to become familiar with the various strategies of information mining utilized in forecast of coronary illness by utilizing various information mining instruments. Life is reliant on proficient working of heart since heart is basic piece of our body. On the off chance that activity of heart isn't appropriate, it will influence the other body portions of human, for example, cerebrum, kidney and so forth. Coronary illness is a sickness that effects on the activity of heart. There are number of elements which expands danger of Heart illness. Presently a days, on the planet Heart sickness is the significant reason for passings. The World Health Organization (WHO) has evaluated that 12 million passings happen around the world, consistently because of the Heart maladies.

In 2008, 17.3 million individuals kicked the bucket because of Heart Disease. Over 80% of passings in world are a result of Heart illness. WHO assessed by 2030, practically 23.6 million individuals will kick the bucket because of Heart malady as written in [10]. Expectation by utilizing information mining methods gives us precise consequence of ailment. IHDPS (shrewd coronary illness expectation framework) can find and concentrate shrouded information related with coronary illness from a chronicled coronary illness database. It can answer complex questions for diagnosing coronary illness and in this manner help medicinal services experts and specialists to settle on clever clinical choices which customary choice emotionally supportive networks can't. In this paper investigation of different information mining methods given in tables which were utilized and supportive for clinical experts or specialists for precise coronary illness conclusion.

AI calculations can possibly be put profoundly in all fields of medication, from tranquilize disclosure to clinical dynamic, essentially modifying the manner in which medication is rehearsed. The achievement of AI calculations at PC vision assignments lately comes at an advantageous time when clinical records are progressively digitalized. The utilization of electronic wellbeing records (EHR) quadrupled from 11.8% to 39.6% among office-based doctors in the US from 2007 to 2012. Clinical pictures are a vital piece of a patient's EHR and are as of now examined by human radiologists, who are restricted by speed, exhaustion, and

experience. It takes years and incredible budgetary expense to prepare a certified radiologist, and some social insurance frameworks re-appropriate radiology answering to bring down cost nations, for example, India by means of tele-radiology. A postponed or wrong conclusion cause's damage to the patient. Accordingly, it is perfect for clinical picture examination to be done by a computerized, precise and productive AI calculation.

1.2 MOTIAVTION

The huge amounts of data generated for prediction of heart disease are too complex and voluminous to be processed and analyzed by traditional methods. Data mining provides the methodology and technology to transform these mounds of data into useful information for decision making. By using data mining techniques it takes less time for the prediction of the disease with more accuracy. In this paper we survey different papers in which one or more algorithms of data mining used for the prediction of heart disease. Result from using neural networks is nearly 100% in one paper [10] and in [6]. So that the prediction by using data mining algorithm given efficient results. Applying data mining techniques to heart disease treatment data can provide as reliable performance as that achieved in diagnosing heart disease.

1.3 OBJECTIVE

In this paper we survey different papers in which one or more algorithms of data mining used for the prediction of heart disease. Result from using neural networks is nearly 100% in one paper [10] and in [6]. So that the prediction by using data mining algorithm given efficient results. Applying data mining techniques to heart disease treatment data can provide as reliable performance as that achieved in diagnosing heart disease.

1.4 PROBLEM STATEMENTS

In this system we try to use several ML algorithms to find out the heart disease prediction from set of sample images. Here we use several ML algorithms and check which one suits best for the finding of heart disease prediction.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In the existing system there was no proper method to identify the heart disease prediction using data mining algorithms. The following are the main limitations in the existing system.

LIMITATION OF EXISTING SYSTEM

1. More Time Delay in finding the route cause of heart diseases
2. There is no prevention technique due to late prediction.
3. There is no early prediction of heart disease.
4. There is no method to identify the heart diseases based on ML and DM Methods

2.2 PROPOSED SYSTEM

The huge amounts of data generated for prediction of heart disease are too complex and voluminous to be processed and analyzed by traditional methods. Data mining provides the methodology and technology to transform these mounds of data into useful information for decision making. By using data mining techniques it takes less time for the prediction of the disease with more accuracy. In this paper we survey different papers in which one or more algorithms of data mining used for the prediction of heart disease. Result from using neural networks is nearly 100% in one paper [10] and in [6]. So that the prediction by using data mining algorithm given efficient results. Applying data mining techniques to heart disease treatment data can provide as reliable performance as that achieved in diagnosing heart disease.

ADVANTAGES OF PROPOSED SYSTEM

- 1) By using data mining techniques it takes less time for the prediction of the disease with more accuracy.
- 2) In this paper we survey different papers in which one or more algorithms of data mining used for the prediction of heart disease.

3) Result from using neural networks is nearly 100% in one paper [10] and in [6]. So that the prediction by using data mining algorithm given efficient results.

4) Applying data mining techniques to heart disease treatment data can provide as reliable performance as that achieved in diagnosing heart disease.

2.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

2.3.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.3.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.3.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and

to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.4 SYSTEM REQUIREMENT SPECIFICATION

2.4.1 Functional Requirements

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define *what* a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Generally, functional requirements are expressed in the form “system shall do <requirement>”. The plan for implementing functional requirements is detailed in the system *design*. In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is: user/stakeholder request -> feature -> use case -> business rule.

Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. Functional requirements may be technical details, data manipulation and other specific functionality of the project is to provide the information to the user. The following are the Functional requirements of our system:

- We are providing all the information related to heart disease.
- The user can give his own values as input to the system and can predict if there is heart disease found.
- We are having multiple ML algorithms for classification.
- The proposed application can accurately identify the heart disease prediction from the best classification algorithm which is found after comparison.

2.4.2 Non Functional Requirements

Non-functional requirements define the overall qualities or attributes of the resulting System. Non-functional requirements place restrictions on the product being developed, the development process, and specify external constraints that the product must meet. Examples of NFR include safety, security, usability, reliability and performance Requirements. Project management issues (costs, time, and schedule) are often considered as non-functional requirements.

Performance requirements

Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance. In this project, the user will try to gather all the information from KAGGLE website and then try to train the system with that dataset.

Modifiability

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

Portability

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

Reliability

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

Security

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time) to break into the system. Do not discuss solutions (e.g. passwords) in a requirements document.

Usability

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

Legal

There may be legal issues involving privacy of information, intellectual property rights, export of restricted technologies, etc.

2.5 HARDWARE REQUIREMENTS

- Processor : Core I3
- RAM : 4 GB(min)
- Hard Disk : 100 GB

2.6 SOFTWARE REQUIREMENTS

- Operating system : Windows7 (Min).
- Coding Language : Python
- Front-End : Google Collab
- Dataset : ML and Deep Learning Model using <https://www.kaggle.com/alxmamaev/hearts-recognition> Dataset

3. SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

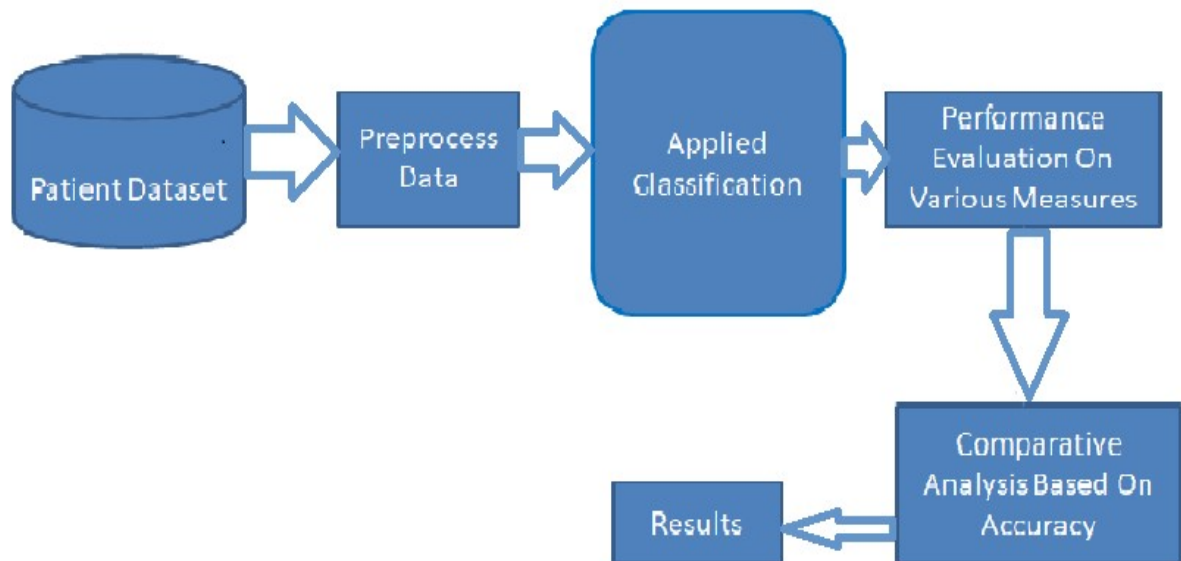


Fig. 3.1 System Architecture

3.2 MODULES

Implementation is the stage where the theoretical design is converted into programmatically manner. In this stage we will divide the application into a number of modules and then coded for deployment. The front end of the application takes Google Collaboratory and as a Back-End Data base we took UCI Heart Patients Records as dataset. Here we are using Python as Programming Language to Implement the current application. The application is divided mainly into following 5 modules. They are as follows:

1. Import Necessary Libraries
2. Load Dataset Module
3. Data Pre-Processing

4. Train the Model Using Several ML Algorithms

5. Find the Performance of ML Algorithms

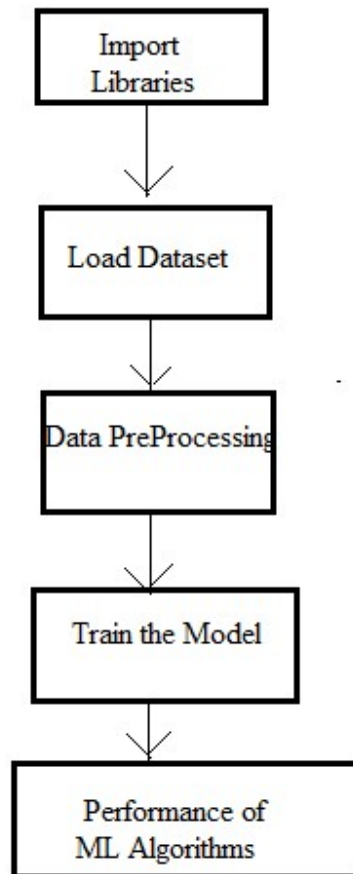


Fig. 3.2 Flowchart of the Technique

3.2.1 IMPORT NECESSARY LIBRARIES

In this module initially we need to import all the necessary libraries which are required for building the model. Here we try to use all the libraries which are used to convert the data into meaningful manner. Here the data is divided into numerical values which are easily identified by the system, hence we try to import numpy module and for plotting the data in graphs and charts we used matplotlib library.

I. Importing essential libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

import os
print(os.listdir())

import warnings
warnings.filterwarnings('ignore')
```

```
['.ipynb_checkpoints', 'heart.csv', 'Heart_disease_prediction.ipynb', 'README.md']
```

3.2.2 LOAD DATASET MODULE

In this module the we try to load the dataset which is downloaded or collected from UCI repository. Here we store the dataset names as 'Heart.csv' file and this dataset contains the following information such as :

```
Data columns (total 14 columns):
age          303 non-null int64
sex          303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Each and every attribute contains some information which are tested and collected based on individual patient id.

3.2.3 DATA PRE-PROCESSING MODULE

Here in this section we try to pre-process the input dataset and find out if there are any missing values or in-complete data present in the dataset. If there is any such data present in the dataset, the application will ignore those values and load only valid rows which have all the valid inputs.

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

3.2.4 TRAIN THE MODEL USING SEVERAL ML ALGORITHMS

Here we try to train the current model on given dataset using several ML classification algorithms and then try to find out which algorithms suits best in order to identify and classify the input dataset accurately and efficiently. Here we try to use following algorithms on input dataset such as:

1. Logistic Regression
2. Naïve Bayes
3. Support Vector Machine
4. K-Nearest Neighbors
5. Decision Tree
6. Random Forest
7. XGBoost
8. Neural Networks

3.2.5 PERFORMANCE ANALYSIS MODULE

Here in this module we try to compare each and every classification algorithm on given input dataset and then try to find out which one suits best for finding the accurate results. Finally

we will identify the best algorithm which give accurate results in very less time. Here we can see **Random Forest** gives more accurate result compared with other ML Algorithms.

3.3 UML Diagrams

The underlying premise of UML is that no one diagram can capture the different elements of a System in its entirety. Hence, UML is made up of nine diagrams that can be used to model a System at different points of time in the software life cycle of a system.

A software system can be said to have two distinct characteristics: a structural, "static" part and a behavioral, "dynamic" part. In addition to these two characteristics, an additional characteristic that a software system possesses is related to implementation. Before we categorize UML diagrams into each of these three characteristics, let us take a quick look at exactly what these characteristics are.

- Use case diagram
- Class diagram
- Object diagram
- State diagram
- Activity diagram
- Sequence diagram
- Collaboration diagram
- Component diagram
- Deployment diagram

3.3.1 Use case Diagram

The use case diagram is used to identify the primary elements and processes that form the System. The primary elements are termed as "actors" and the processes are called "use cases." The use case diagram shows which actors interact with each use case.

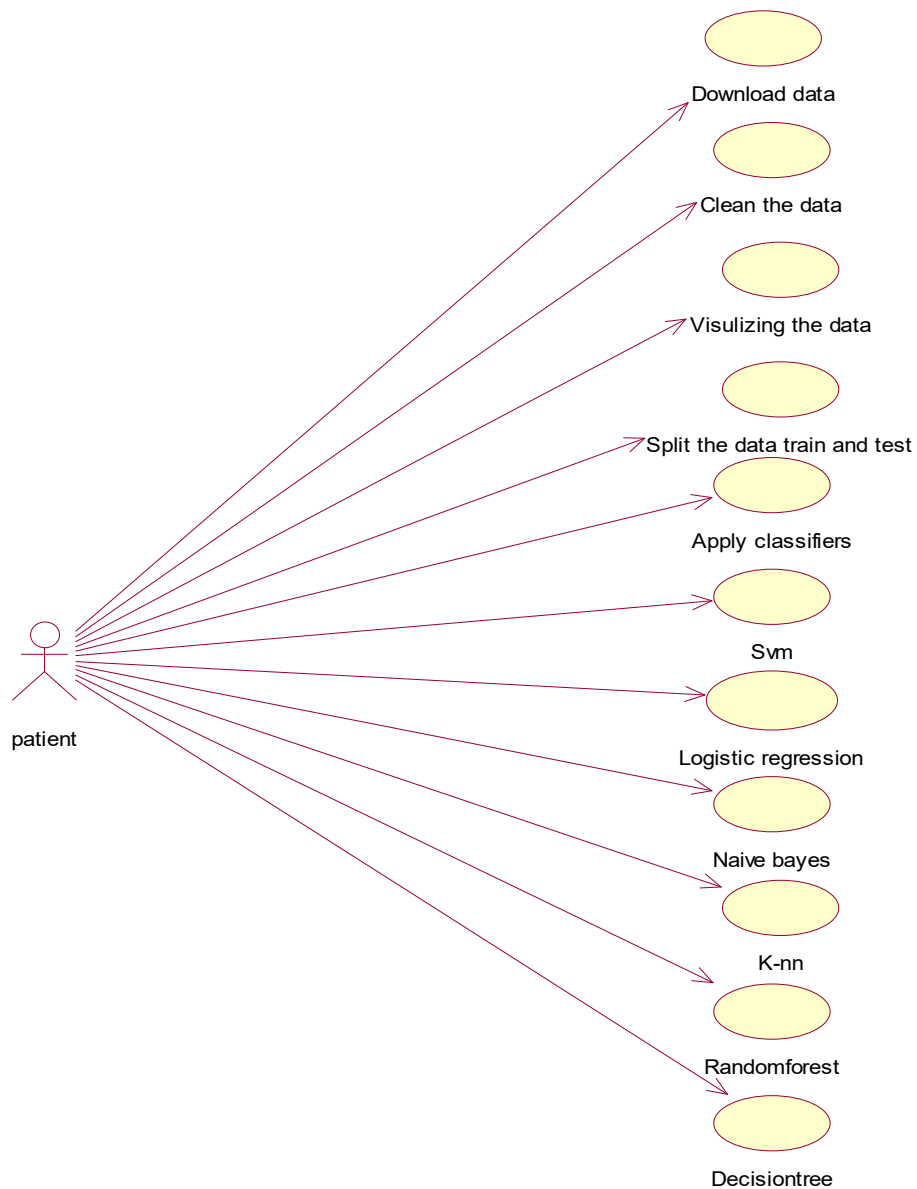


Fig. 3.3 Usecase Diagram Admin/User

Use cases A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

3.3.2 Class Diagram

The class diagram is used to refine the use case diagram and define a detailed design of the System. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" Or "has-a" relationship.

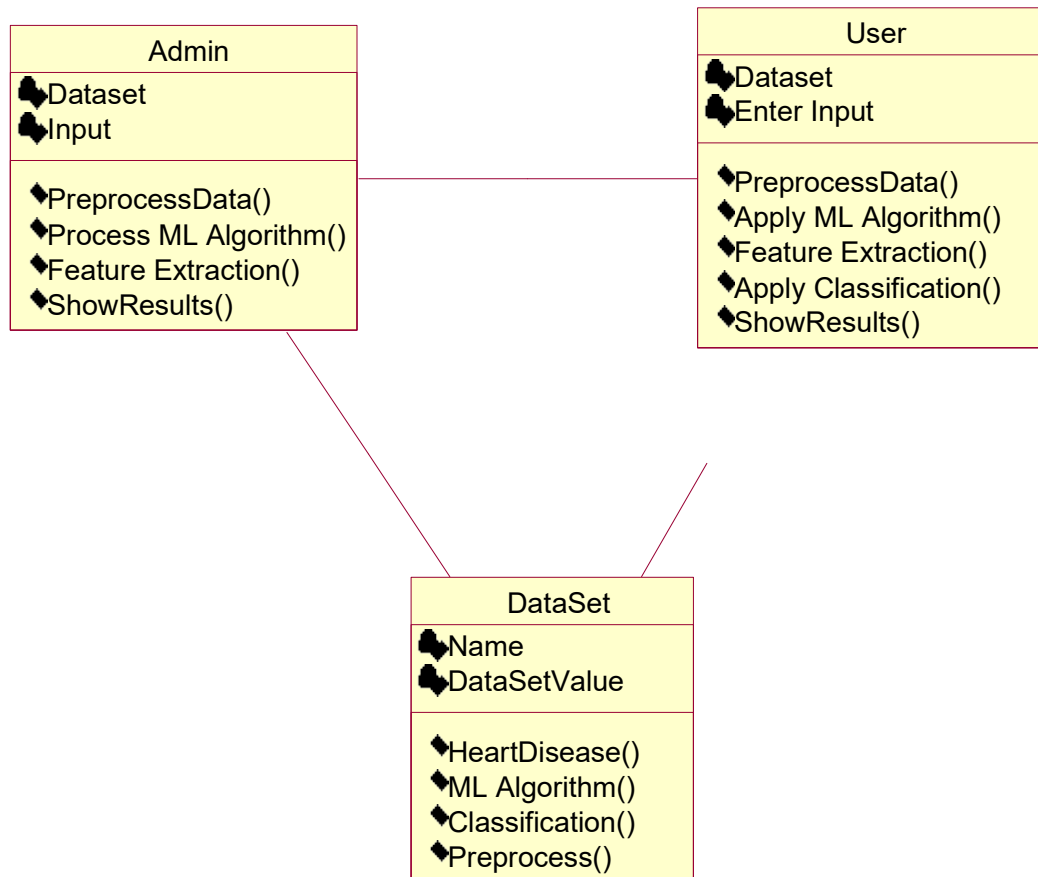


Fig. 3.4 Class Diagram

3.3.3 Sequence Diagram

A sequence diagram represents the interaction between different objects in the system. The Important aspect of a sequence diagram is that it is time-ordered. Different objects In the sequence diagram interact with each other by passing "messages".

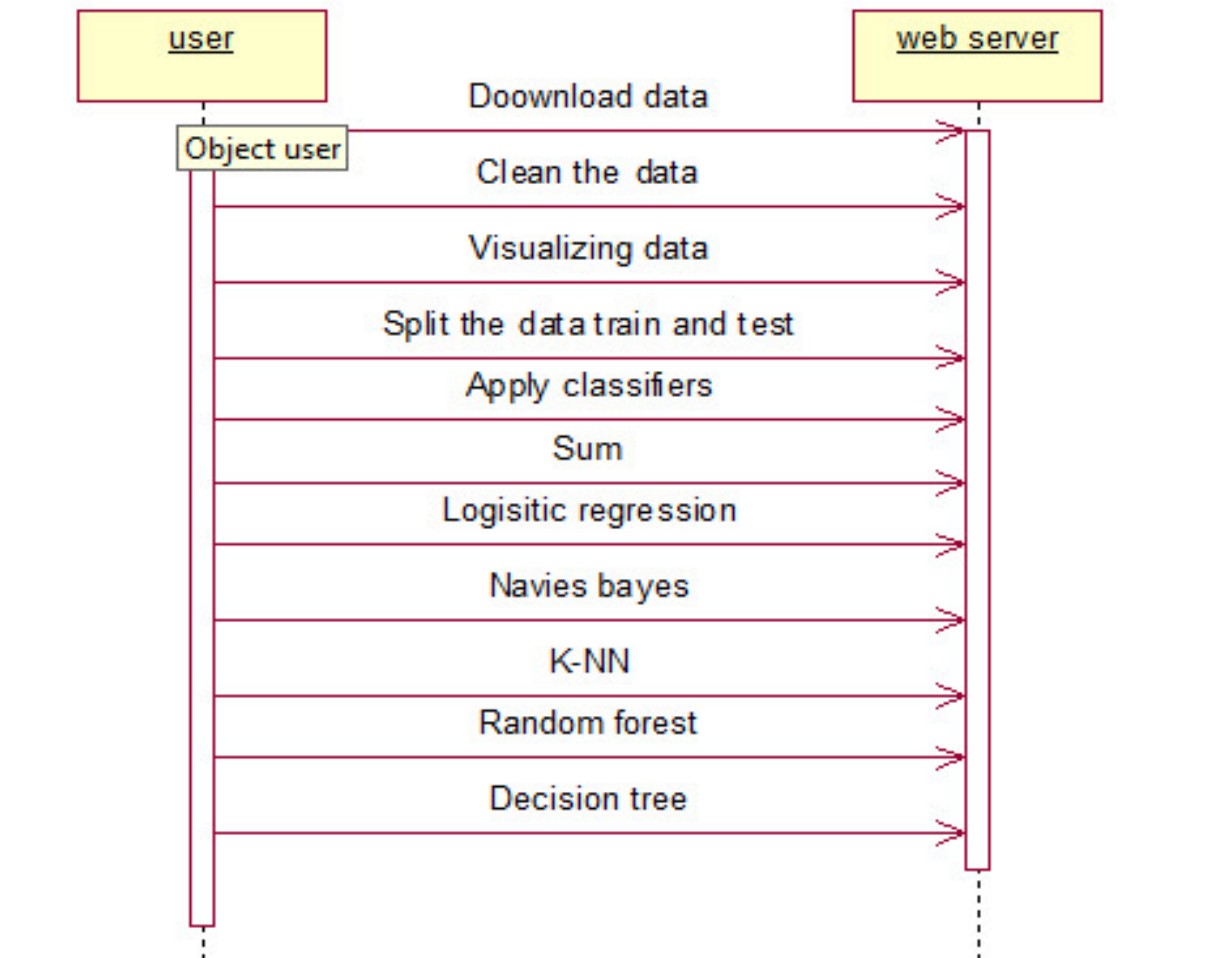


Fig. 3.5 Sequence Diagram

3.3.4 Activity Diagram

The process flows in the system are captured in the activity diagram. Similar to a state Diagram, an activity diagram also consists of activities, actions, transitions, initial and final States, and guard conditions.

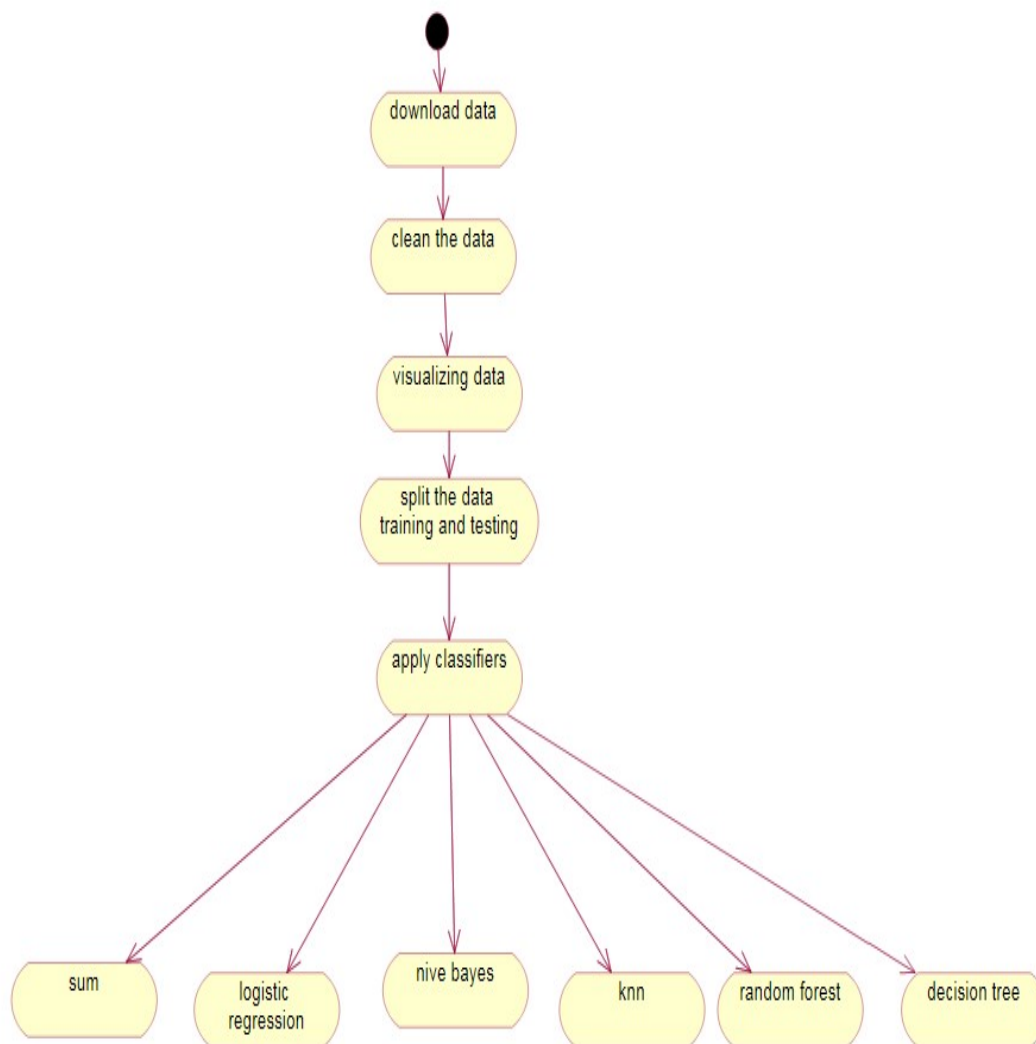


Fig. 3.6 Activity Diagram

3.3.5 Component Diagram

The process of this diagram shows the organizations and dependencies among a set of components. It represents the static implementation view of a system.

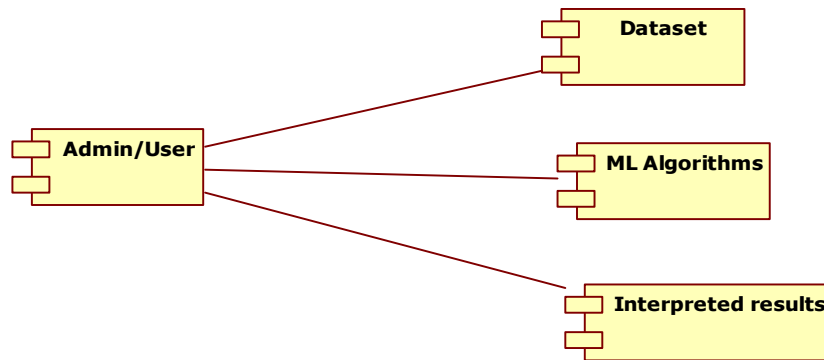


Fig. 3.7 Component Diagram

3.3.6 Deployment Diagram

The deployment diagram captures the configuration of the runtime elements of the Application. This diagram is by far most useful when a system is built and ready to be Deployed. The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related. The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

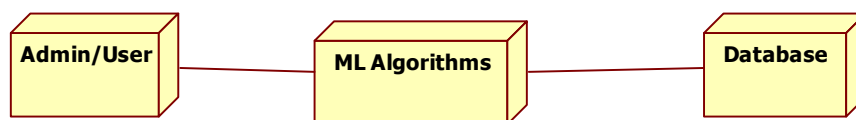


Fig. 3.8 Deployment Diagram

3.4 DATABASE DESIGN

The data pertaining to proposed system is voluminous that a careful design of the database must proceed before storing the data in the database. A database management system

Provides flexibility in the storage and retrieval of data and production of information. The DBMS is a bridge between the application programs, which determines what data are needed and how they are processed, and the operating system of the computer, which is responsible for placing data on the magnetic storage devices.

3.4.1 Normalization

Normalization theory is built around the concept of normal forms. A relation is said to be in particular normal form if it satisfies a certain specified set of constraints.

First Normal form

A relation R is in first normal form if and only if all underlying domains contained atomic values only

Second Normal form

A relation R is said to be in second normal form if and only if it is in first normal form and every non-key attribute is fully dependent on the primary key.

Third Normal form

A relation R is said to be in third normal form if and only if it is in second normal form and every non key attribute is non transitively depend on the primary key.

Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form
- For each functional dependency ($X \rightarrow Y$), X should be a super Key.

Fourth Normal Form (4NF)

Fourth Normal Form comes into picture when Multi-valued Dependency occurs in any relation.

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

- It should be in the Boyce-Codd Normal Form.
- The table should not have any Multi-valued Dependency.

4. SYSTEM IMPLEMENTATION

4.1 FRONT END IMPLEMENTATION

Python Introduction

Python is a general purpose, dynamic, high level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. It supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles. Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. We don't need to use data types to declare variable because it is dynamically typed so we can write `a=10` to assign an integer value in an integer variable. It makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development. Here, we are specifying applications areas where python can be applied.

- **Web Applications:** We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, BeautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc. The useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering.
- **Desktop GUI Applications:** Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

- **Software Development:** Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.
- **Scientific and Numeric:** Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.
- **Business Application:** Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.
- **Console Based Application:** We can use Python to develop console based applications. For example: IPython.
- **Audio or Video based Applications:** Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.\
- **3D CAD Applications:** To create CAD application Fandango is a real application which provides full features of CAD.
- **Enterprise Applications:** Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.
- **Applications for Images:** Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. It supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles.

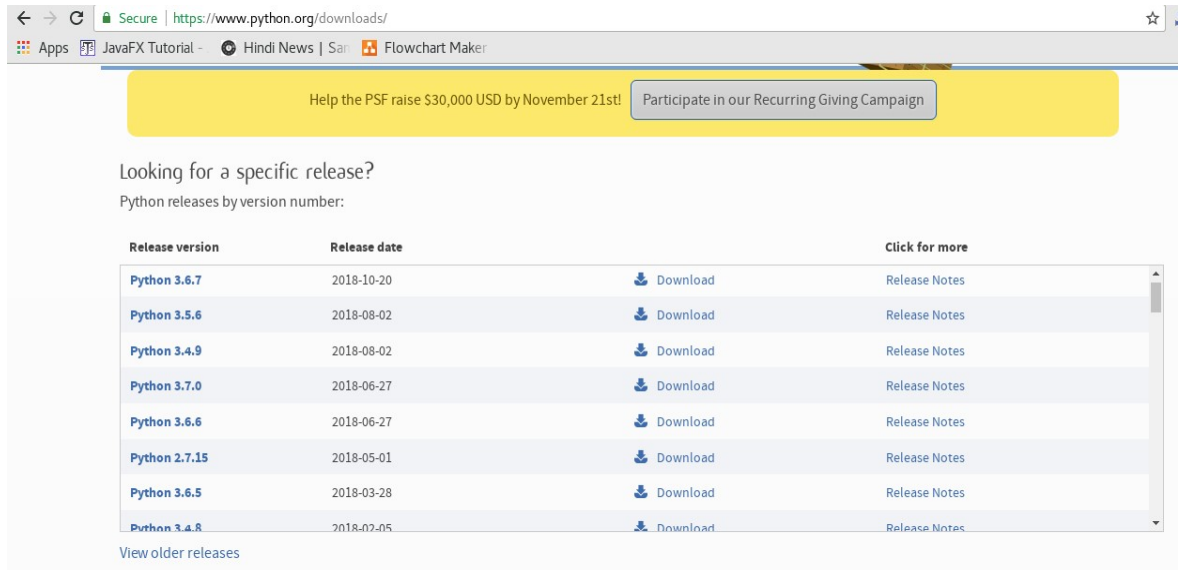
There are several such applications which can be developed using Python

How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of python on various operating systems.

Installation on Windows:

Visit the link <https://www.python.org/downloads/> to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.



Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.



Fig. 4.1 Python Installation

The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.

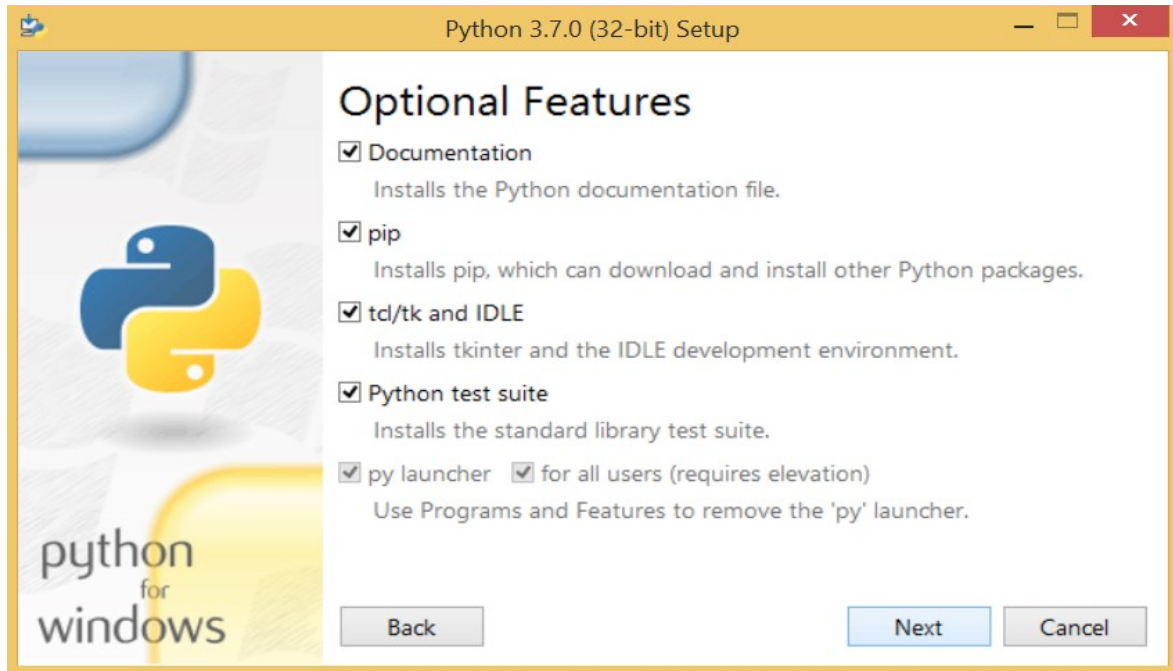
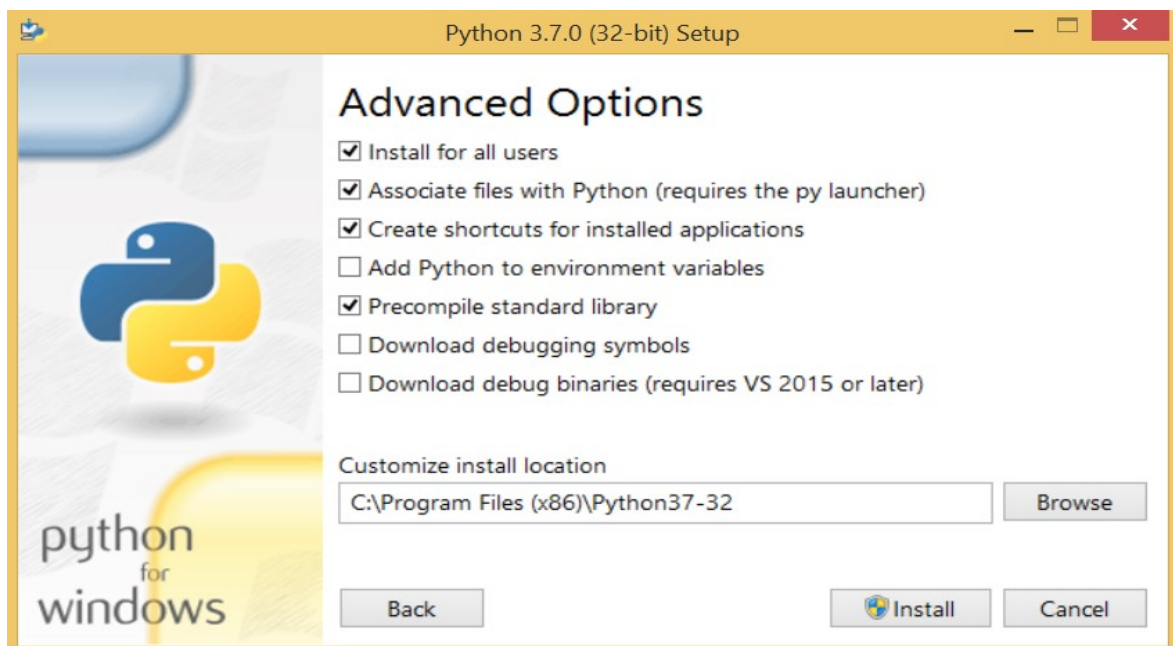


Fig. 4.2 Python Setup

The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



Now, we are ready to install python-3.6.7. Let's install it.

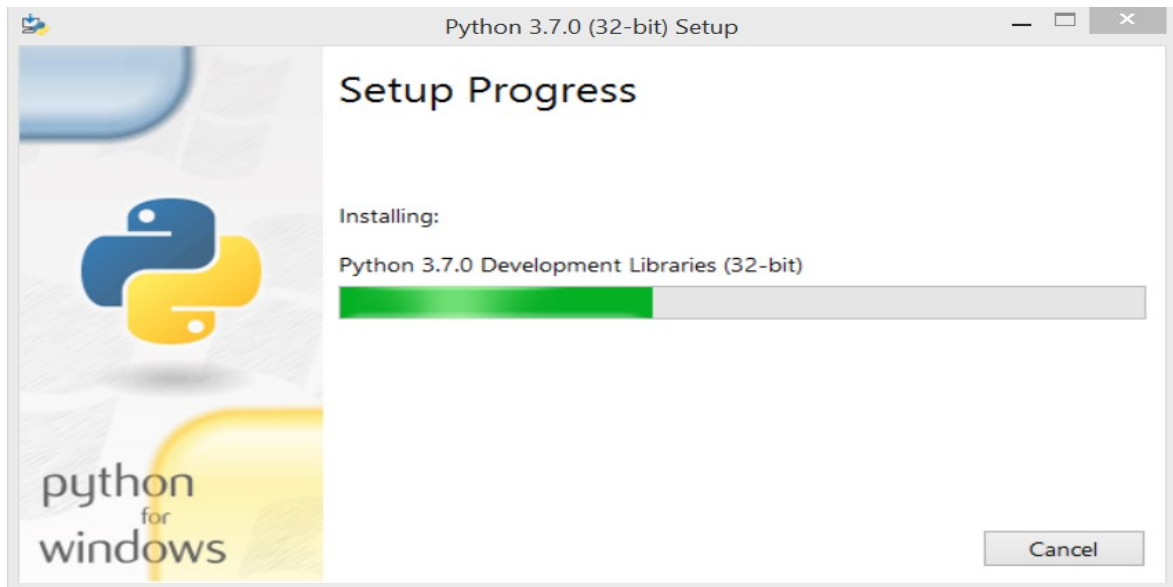
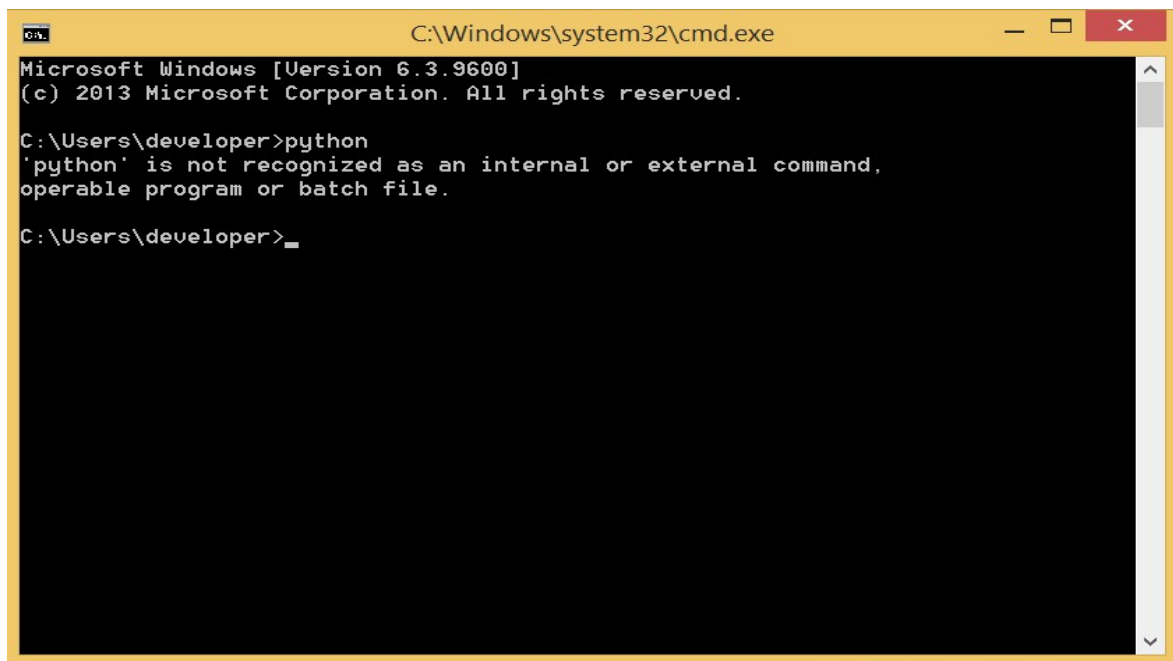
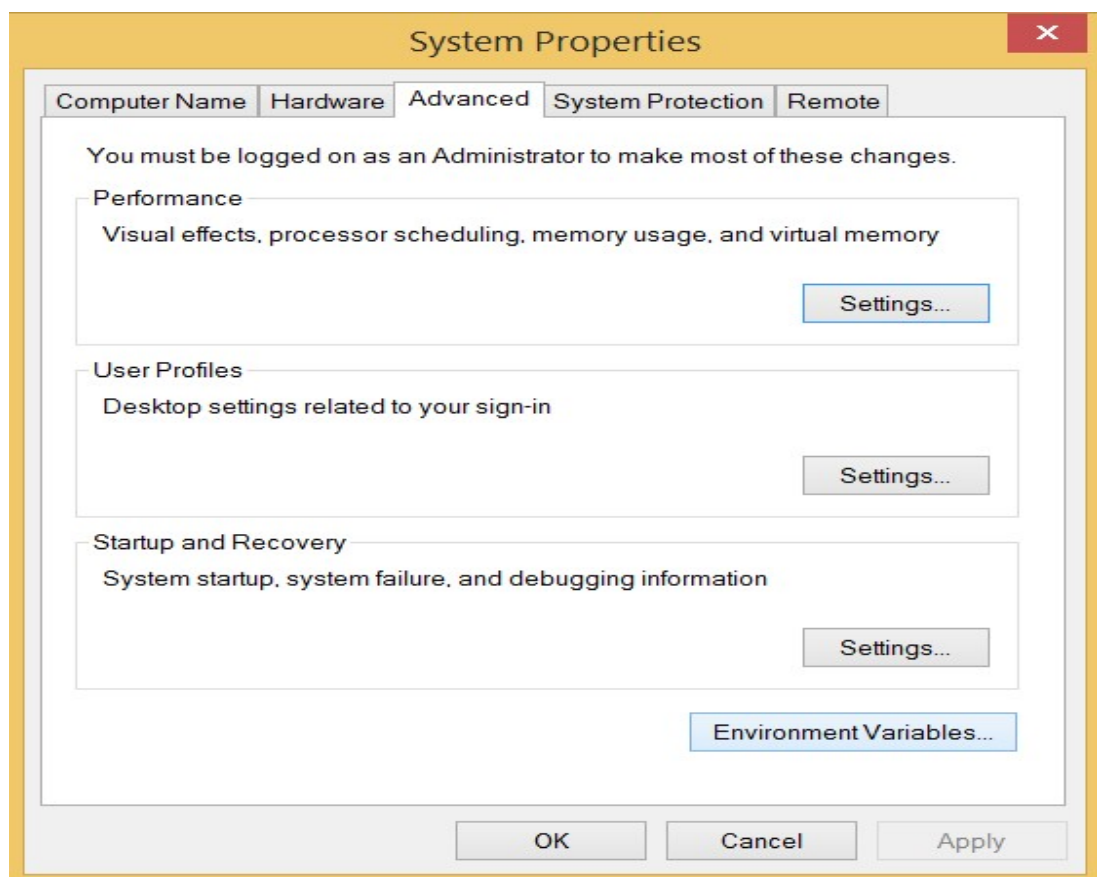
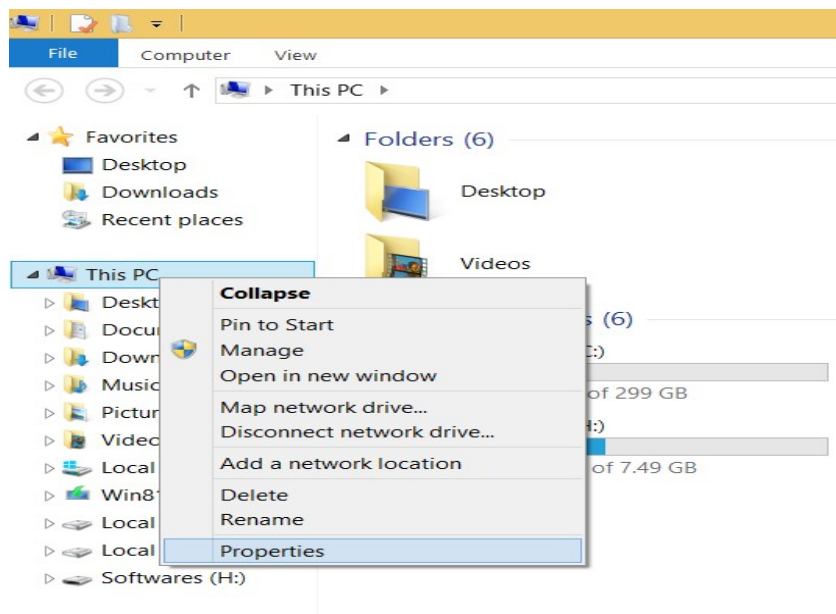


Fig. 4.3 Python Setup Progress

Now, try to run python on the command prompt. Type the command **python** in case of python2 or python3 in case of **python3**. It will show an error as given in the below image. It is because we haven't set the path.



To set the path of python, we need to right click on "my computer" and go to Properties
→ Advanced → Environment Variables.



Add the new path variable in the user variable section.

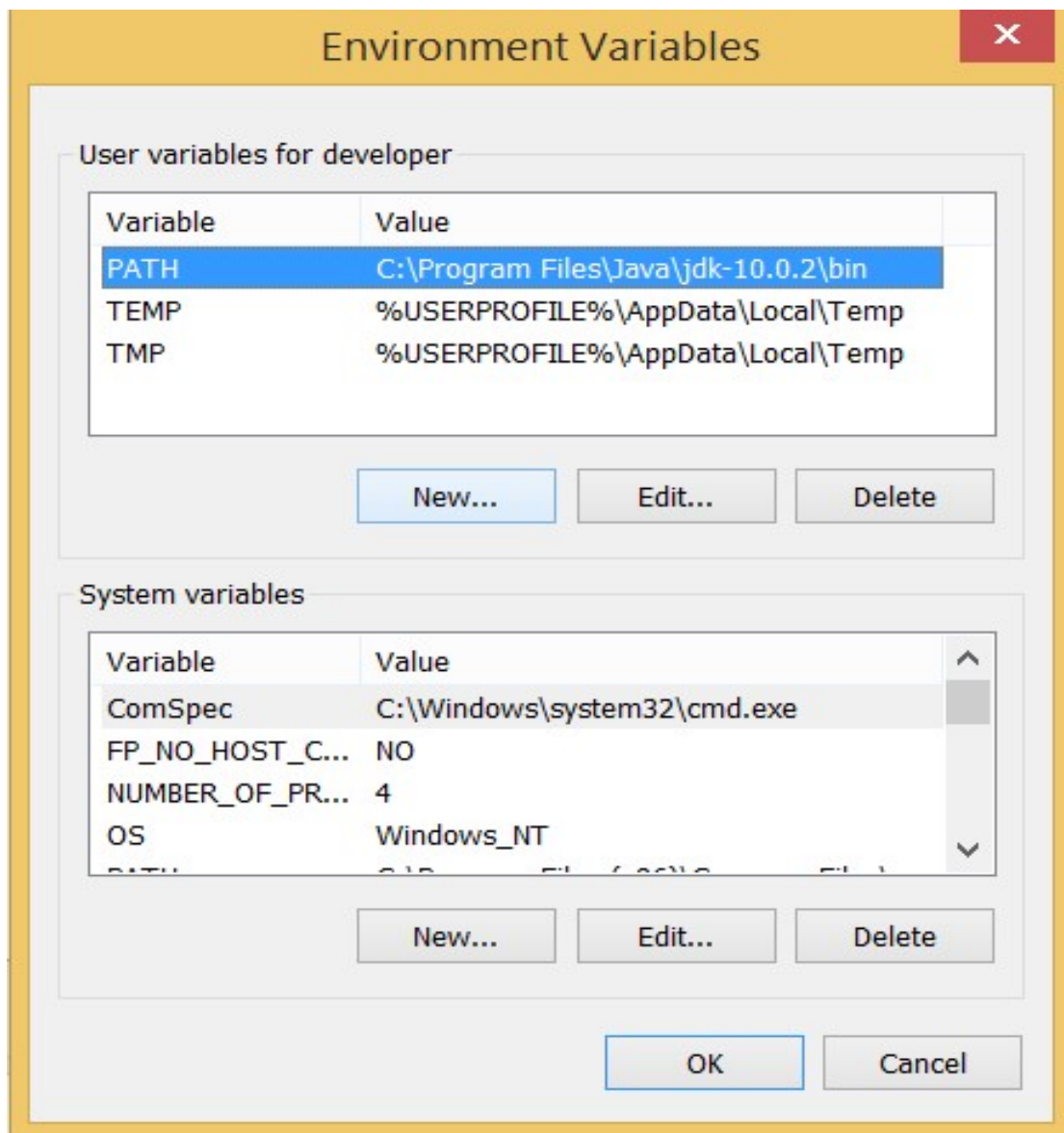
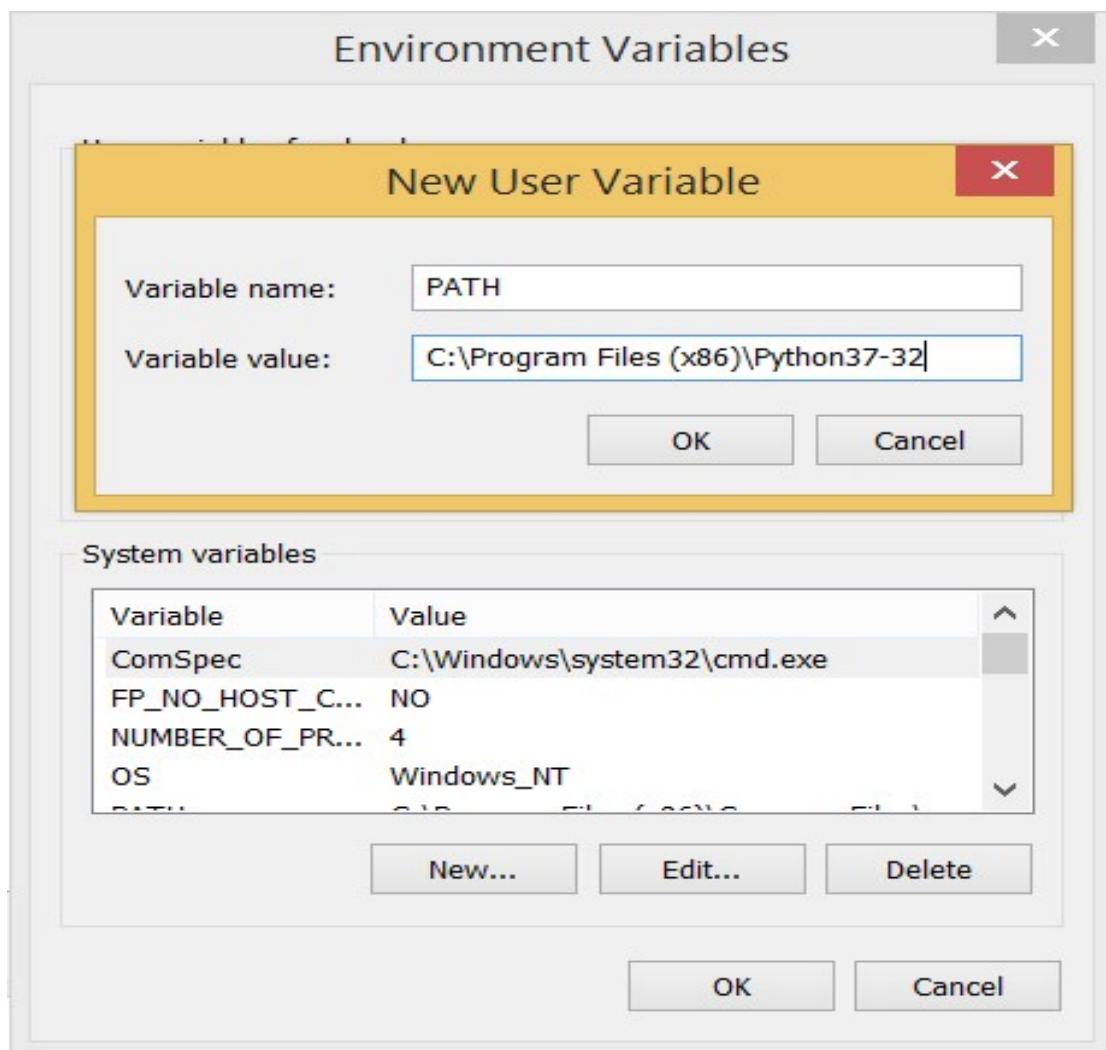


Fig. 4.4 Python Path Setup

Type **PATH** as the variable name and set the path to the installation directory of the python shown in the below image.



Now, the path is set, we are ready to run python on our local system. Restart CMD, and type **python** again. It will open the python interpreter shell where we can execute the python statements.

First Python Program

In this Section, we will discuss the basic syntax of python by using which, we will run a simple program to print hello world on the console.

Python provides us the two ways to run a program:

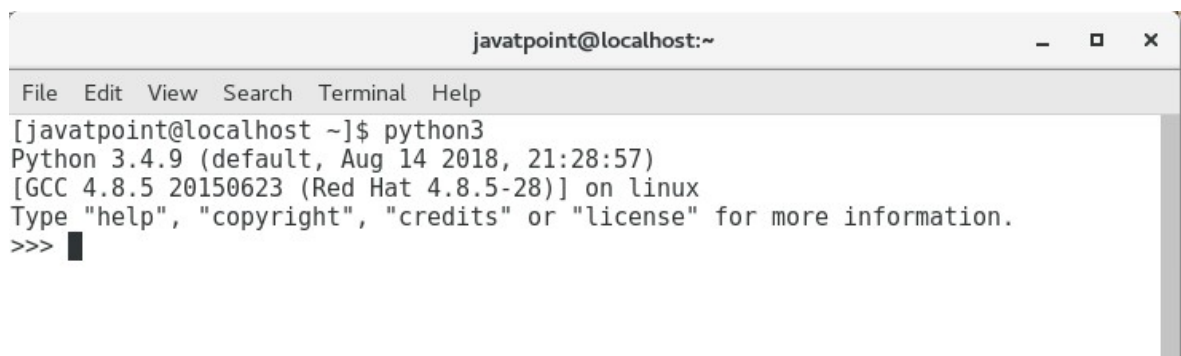
- Using Interactive interpreter prompt
- Using a script file

Let's discuss each one of them in detail.

Interactive interpreter prompt

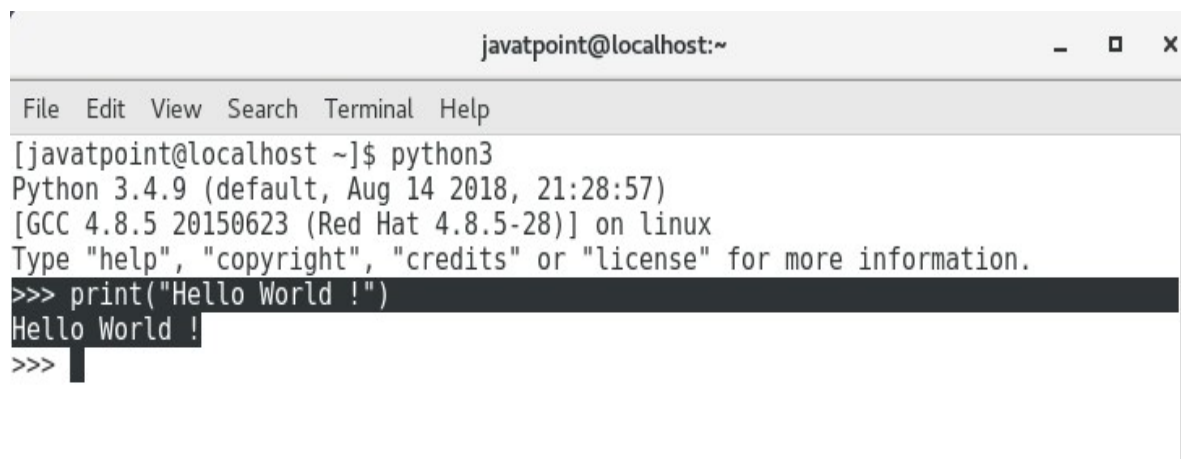
Python provides us the feature to execute the python statement one by one at the interactive prompt. It is preferable in the case where we are concerned about the output of each line of our python program. To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have python2 and python3 both installed on your system).

It will open the following prompt where we can execute the python statement and check their impact on the console.

A terminal window titled 'javatpoint@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'python3' being executed, followed by the Python version and GCC information. The prompt '>>>' is visible at the bottom.

```
javatpoint@localhost:~  
File Edit View Search Terminal Help  
[javatpoint@localhost ~]$ python3  
Python 3.4.9 (default, Aug 14 2018, 21:28:57)  
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> █
```

Let's run a python statement to print the traditional hello world on the console. Python3 provides print() function to print some message on the console. We can pass the message as a string into this function. Consider the following image.

A terminal window titled 'javatpoint@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the same initial output as the previous image, but now includes the command 'print("Hello World !")' and its output 'Hello World !'. The prompt '>>>' is visible at the bottom.

```
javatpoint@localhost:~  
File Edit View Search Terminal Help  
[javatpoint@localhost ~]$ python3  
Python 3.4.9 (default, Aug 14 2018, 21:28:57)  
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("Hello World !")  
Hello World !  
>>> █
```

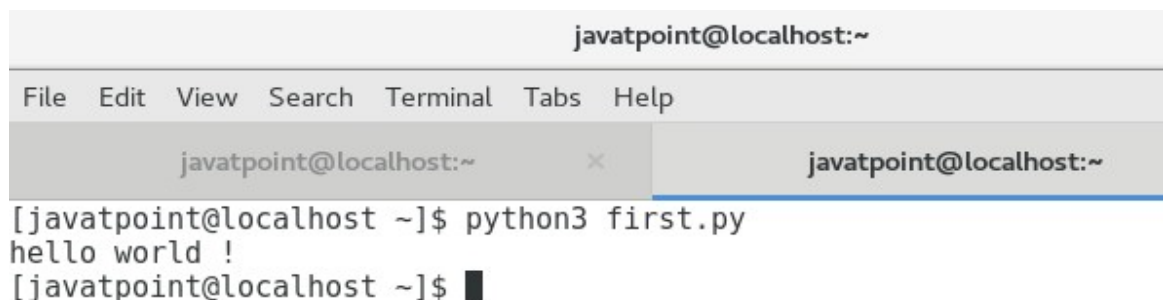
Here, we get the message "**Hello World !**" printed on the console.

Using a script file

Interpreter prompt is good to run the individual statements of the code. However, we can not write the code every-time on the terminal. We need to write our code into a file which can be executed later. For this purpose, open an editor like notepad, create a file named first.py (python used .py extension) and write the following code in it.

- Print ("hello world"); #here, we have used print() function to print the message on the console. To run this file named as first.py, we need to run the following command on the terminal.

\$ python3 first.py

A screenshot of a terminal window titled 'javatpoint@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', 'Tabs', and 'Help'. Below the menu bar, there are two tabs, both labeled 'javatpoint@localhost:~'. The active tab shows the command '[javatpoint@localhost ~]\$ python3 first.py' and its output 'hello world !'. The prompt '[javatpoint@localhost ~]\$' is followed by a cursor.

Hence, we get our output as the message **Hello World !** is printed on the console.

AlexNet

AlexNet as described in [3] is shown in figure 3 is one of the two models that we are using for classification. For training AlexNet, we have used NVIDIA DIGITS framework [4] with the BVLC Caffe [5] as backend. The training was performed on an NVIDIA TitanX GPU with the CUDA toolkit. For our experiments, we use pre-trained model of AlexNet trained on ILSVRC2012 dataset which has earlier been helpful in transfer learning problems. The AlexNet has . The network has a softmax layer at its output which showcases the likelihood of a particular class during inference.

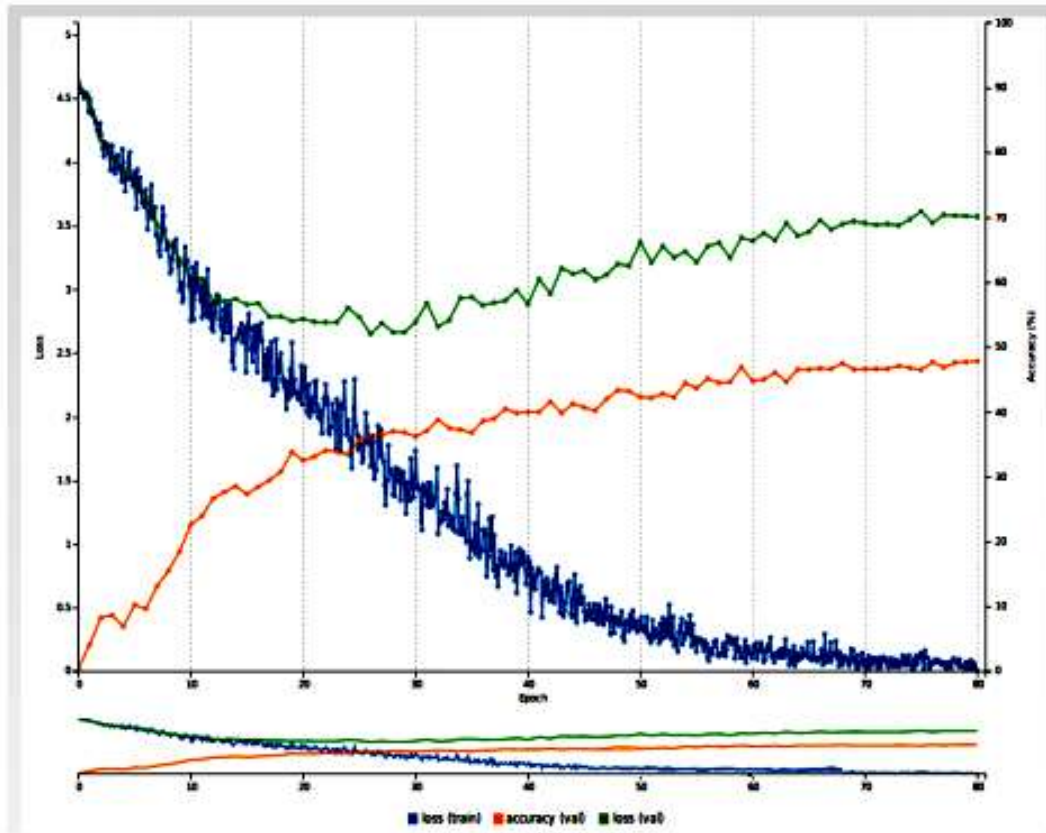


Figure 5: AlexNet Training Curve

For optimization, we used the stochastic gradient descent algorithm which computes the loss function in batches. The base learning rate used was the 0.01 with linear learning rate decay. We trained the network for 100 epochs. A dropout layer with a dropout ratio of 0.5 was also implemented to introduce regularization. At the end, we had a softmax layer to give out scores of each respective class. The network took 15 minutes 9 seconds to finish training. We observed the results on the test set and found the top-1 accuracy to be 43.39% and top-5 accuracy to be 68.68%. The training curve of the AlexNet can be found in figure 5.

4.2 Back End Implementation

Machine Learning

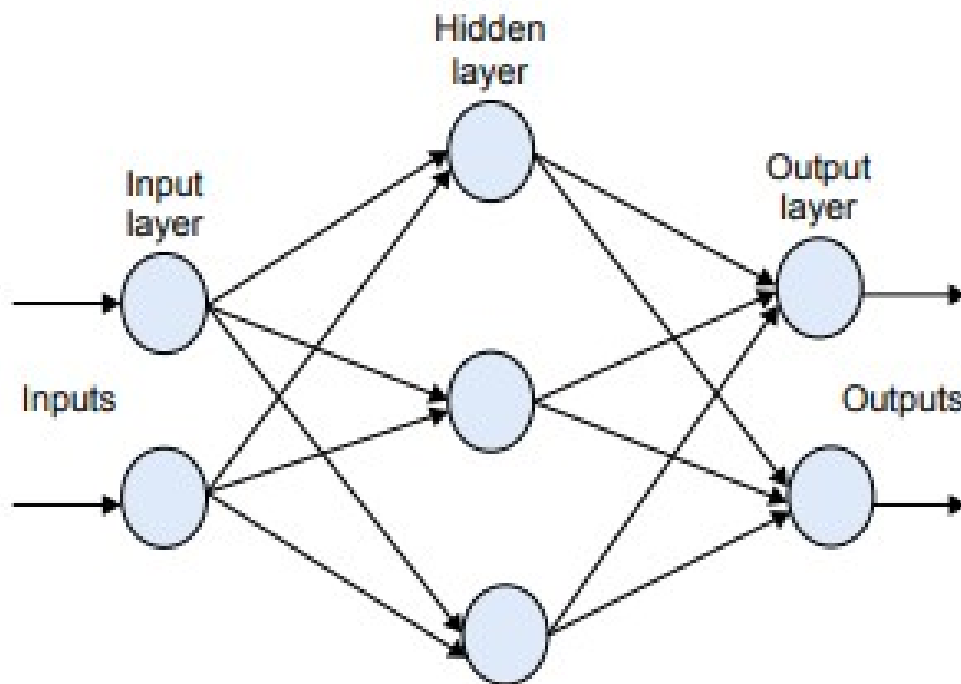
Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range.

Machine learning is the scientific field dealing with the ways in which machines learn from experience. For many scientists, the term “machine learning” is identical to the term “artificial intelligence”, given that the possibility of learning is the main characteristic of an entity called intelligent in the broadest sense of the word. The purpose of machine learning is the construction of computer systems that can adapt and learn from their experience. There are two types

- **Supervised Learning** In supervised learning, the system must “learn” inductively a function called target function, which is an expression of a model describing the data. The objective function is used to predict the value of a variable, called dependent variable or output variable, from a set of variables, called independent variables or input variables or characteristics or features. The set of possible input values of the function, i.e. its domain, are called instances. Each case is described by a set of characteristics (attributes or features). A subset of all cases, for which the output variable value is known, is called training data or examples. In order to infer the best target function, the learning system, given a training set, takes into consideration alternative functions, called hypothesis and denoted by h . In supervised learning, there are two kinds of learning tasks: classification and regression.
- **Unsupervised Learning** In unsupervised learning, the system tries to discover the hidden structure of data or associations between variables. In that case, training data consists of instances without any corresponding labels. Association Rule Mining appeared much later than machine learning and is subject to greater influence from the research area of databases.

CNN MODEL

The experiment was carried out on a publicly available database for heart disease. The dataset contains a total of 303 records that were divided into two sets, training set



Multilayer perceptron neural network

(40%) and testing set (60%). A data mining tool named Weka 3.6.11 was used for the experiment. Additionally, multilayer perceptron neural network (MLPNN) with back propagation (BP) was used as the training algorithm.

MLPNN

MLPNN is one of the most significant models in artificial neural network. The MLPNN consists of one input layer, one or more hidden layers and one output layer.³ In MLPNN, the input nodes pass values to the first hidden layer, and then nodes of first hidden layer pass values to the second and so on till producing outputs as shown in Figure.

BP network

The BP algorithm has served as a useful methodology to train multilayer perceptron for a wide range of applications.⁴ The BP network calculates the difference between real and predicted values, which is circulated from output nodes backwards to nodes in previous layer. The BP learning algorithm can be divided into two phases, propagation and weight update.⁴ First, this learning algorithm provides training data to the network and compares the actual and desired outputs. Then, it calculates the error in each neuron. Based on this, the algorithm calculates what output should be for each neuron and how much higher or lower output must be adjusted for desired output and finally adjusts the weights. The overall process is done to improve weights during processing.

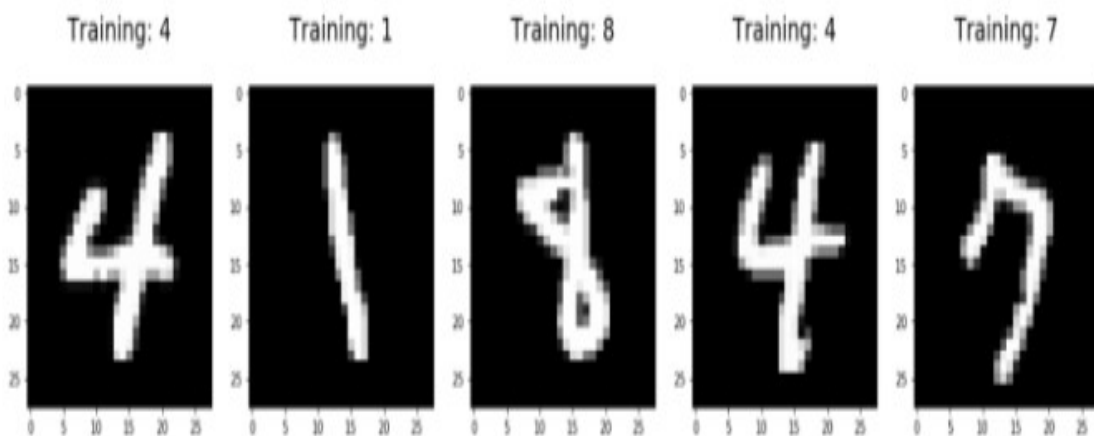
The risk factor for heart disease

1. **Family history of heart disease:** - most people know that the heart disease can run in families. That if anybody has a family history of heart disease, he/she may be at greater risk for heart attack, stroke and other heart diseases. Smoking: - smoking is major cause of heart attack, stroke and other peripheral arterial disease. Nearly 40% of all people who die from smoking tobacco do so due of heart and blood vessel diseases. A smoker's risk of heart attack reduces rapidly after only one year of not smoking.
2. **Cholesterol:** - abnormal levels of lipids (fats) in the blood are risk factor of heart diseases. Cholesterol is a soft, waxy substance found among the lipids in the bloodstream and in all the body's cells. High level of triglyceride (most common type of fat in body) combined with high levels of LDL (low density lipoprotein) cholesterol speed up atherosclerosis increasing the risk of heart diseases.
3. **High blood pressure:** - High blood pressure also known as HBP or hypertension is a widely misunderstood medical condition. High blood

pressure increase the risk of the walls of our blood vessels walls becoming overstretched and injured. Also increase the risk of having heart attack or stroke and of developing heart failure, kidney failure and peripheral vascular disease.

4. **Obesity:-**the term obesity is used to describe the health condition of anyone significantly above his or her ideal healthy weight. Being obese puts anybody at a higher risk for health problem such as heart disease, stroke, high blood pressure, diabetes and more.
5. **Lack of physical exercise:** -lack of exercise is a risk factor for developing coronary artery disease (CAD). Lack of physical exercise increases the risk of CAD, because it also increases the risk for diabetes and high blood pressure

Logistic Regression using Python (scikit-learn)



Visualizing the Images and Labels in the MNIST Dataset

One of the most amazing things about Python's scikit-learn library is that it has a 4-step modeling pattern that makes it easy to code a machine learning classifier. While this tutorial uses a classifier called Logistic Regression, the coding process in this tutorial applies to other

classifiers in sklearn (Decision Tree, K-Nearest Neighbors etc). In this tutorial, we use Logistic Regression to predict digit labels based on images. The image above shows a bunch of training digits (observations) from the MNIST dataset whose category membership is known (labels 0–9). After training a model with logistic regression, it can be used to predict an image label (labels 0–9) given an image.

a toy dataset (digits dataset) to show quickly illustrate scikit-learn's 4 step modeling pattern and show the behavior of the logistic regression algorithm. The second part of the tutorial goes over a more realistic dataset (MNIST dataset) to briefly show how changing a model's default parameters can effect performance (both in timing and accuracy of the model).

Naive Bayes Classification using Scikit-learn

Suppose you are a product manager, you want to classify customer reviews in positive and negative classes. Or As a loan manager, you want to identify which loan applicants are safe or risky? As a healthcare analyst, you want to predict which patients can suffer from diabetes disease. All the examples have the same kind of problem to classify reviews, loan applicants, and patients.

Naive Bayes is the most straightforward and fast classification algorithm, which is suitable for a large chunk of data. Naive Bayes classifier is successfully used in various applications such as spam filtering, text classification, sentiment analysis, and recommender systems. It uses Bayes theorem of probability for prediction of unknown class.

Classification Workflow

Whenever you perform classification, the first step is to understand the problem and identify potential features and label. Features are those characteristics or attributes which affect the results of the label. For example, in the case of a loan distribution, bank manager's identify customer's occupation, income, age, location, previous loan history, transaction history, and credit score. These characteristics are known as features which help the model classify customers.

The classification has two phases, a learning phase, and the evaluation phase. In the learning phase, classifier trains its model on a given dataset and in the evaluation phase, it tests the classifier performance. Performance is evaluated on the basis of various parameters such as accuracy, error, precision, and recall.

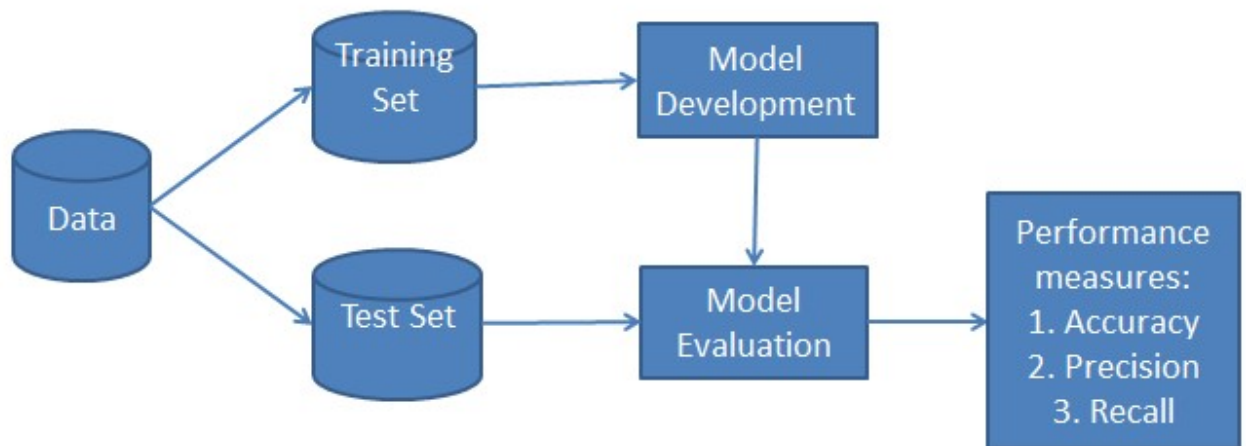


Fig 6: Data Model

What is Naive Bayes Classifier?

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

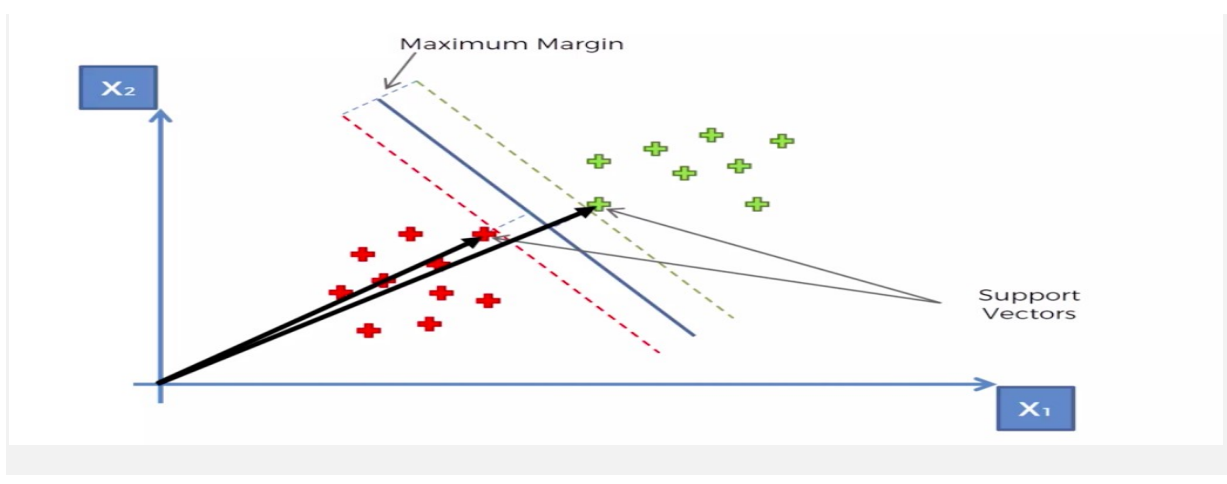
Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$: the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h .
- $P(D)$: the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- $P(h|D)$: the probability of hypothesis h given the data D . This is known as posterior probability.
- $P(D|h)$: the probability of data d given that the hypothesis h was true. This is known as posterior probability.

Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm capable of performing classification, regression and even outlier detection. The linear SVM classifier works by drawing a straight line between two classes. All the data points that fall on one side of the line will be labeled as one class and all the points that fall on the other side will be labeled as the second. Sounds simple enough, but there's an infinite amount of lines to choose from. How do we know which line will do the best job of classifying the data? This is where the LSVM algorithm comes in to play. The LSVM algorithm will select a line that not only separates the two classes but stays as far away from the closest samples as possible. In fact, the “support vector” in “support vector machine” refers to two position vectors drawn from the origin to the points which dictate the decision boundary.



K Nearest Neighbor Algorithm

K-Nearest Neighbors, or KNN for short, is one of the simplest machine learning algorithms and is used in a wide array of institutions. KNN is a **non-parametric, lazy** learning algorithm. When we say a technique is non-parametric, it means that it does not make any assumptions about the underlying data. In other words, it makes its selection based off of the proximity to other data points regardless of what feature the numerical values represent. Being a lazy learning algorithm implies that there is little to no training phase. Therefore, we can immediately classify new data points as they present themselves.

Some pros and cons of KNN

Pros:

- No assumptions about data
- Simple algorithm — easy to understand
- Can be used for classification and regression

Cons:

- High memory requirement — All of the training data must be present in memory in order to calculate the closest K neighbors
- Sensitive to irrelevant features
- Sensitive to the scale of the data since we're computing the distance to the closest K points

Decision Tree Classification

Decision Tree Classification, attribute selection measures, and how to build and optimize Decision Tree Classifier using Python Scikit-learn package. As a marketing manager, you want a set of customers who are most likely to purchase your product. This is how you can save your marketing budget by finding your audience. As a loan manager, you need to identify risky loan applications to achieve a lower loan default rate. This process of classifying customers into a group of potential and non-potential customers or safe or risky loan applications is known as a classification problem. Classification is a two-step process, learning step and prediction step. In the learning step, the model is developed based on given training data. In the prediction step, the model is used to predict the response for given data. Decision Tree is one of the easiest and popular classification algorithms to understand and interpret. It can be utilized for both classification and regression kind of problem.

Random Forest Algorithm

Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks

XGBoost Algorithm:-

XGBoost is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

4.3 SOURCE CODE

Main.py

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
import datetime
import numpy as np
from tkinter.filedialog import askopenfilename
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KDTree
import skimage.io as io
import time
import os
import matplotlib.patches as patches
from PIL import Image, ImageTk
main = tkinter.Tk()
main.title("Bird Species Identification using Deep Learning")
main.geometry("1200x1200")
global filename
LR = LogisticRegression(solver='lbfgs', multi_class='multinomial', max_iter=100)
pretrain_model = 'inception_v3_iNat_299'
dataset = 'cub_200'
load_dir = os.path.join('./feature', pretrain_model)
features_train = np.load(os.path.join(load_dir, dataset + '_feature_train.npy'))
labels_train = np.load(os.path.join(load_dir, dataset + '_label_train.npy'))
features_val = np.load(os.path.join(load_dir, dataset + '_feature_val.npy'))
labels_val = np.load(os.path.join(load_dir, dataset + '_label_val.npy'))
print(features_train.shape)
```

```

print(labels_train.shape)
print(features_val.shape)
print(labels_val.shape)
tic = time.time()
LR.fit(features_train, labels_train)
labels_pred = LR.predict(features_val)
num_class = len(np.unique(labels_train))
acc = np.zeros((num_class, num_class), dtype=np.float32)
for i in range(len(labels_val)):
    acc[int(labels_val[i]), int(labels_pred[i])] += 1.0
fig, ax = plt.subplots(figsize=(6,6))
plt.imshow(acc)
cbar = plt.colorbar()
cbar.ax.tick_params(labelsize=12)
print('Accuracy: %f % (sum([acc[i,i] for i in range(num_class)]) / len(labels_val)))
print('Elapsed Time: %f s' % (time.time() - tic))
data_dir = './data'
train_list = []
val_list = []
bounding_box = []
for line in open(os.path.join(data_dir, dataset, 'train.txt'), 'r'):
    train_list.append(
        (os.path.join(data_dir, dataset, line.strip().split(':')[0]),
         int(line.strip().split(':')[1])))
for line in open(os.path.join(data_dir, dataset, 'val.txt'), 'r'):
    val_list.append(
        (os.path.join(data_dir, dataset, line.strip().split(':')[0]),
         int(line.strip().split(':')[1])))
for line in open(os.path.join(data_dir, dataset, 'bounding_boxes.txt'), 'r'):
    bounding_box.append(line.strip())
def upload():

```

```

    global filename
    filename = askopenfilename(initialdir = "images")
    pathlabel.config(text=filename)
def DCNN():
    name = os.path.basename(filename)
    arr = name.split(".")
    kdt = KDTree(features_train, leaf_size=30, metric='euclidean')
    K = 5
    q_ind = int(arr[0])
    box = bounding_box[val_list[q_ind][1]+1].split(" ")
    print(features_val[q_ind:q_ind+1])
    dist, ind = kdt.query(features_val[q_ind:q_ind+1], k=K)
    print('Query image from validation set:')
    I = io.imread(filename)
    fig,ax = plt.subplots(1)
    plt.axis('off')
    plt.imshow(I)
    plt.suptitle("query image : "+os.path.basename(filename), fontsize=10)
    #rect =
    patches.Rectangle((float(box[1]),float(box[2])),float(box[3]),float(box[4]),linewidth=1,edgecolor
    ='r',facecolor='none')
    #ax.add_patch(rect)
    plt.show()
    for i in range(K):
    plt.figure(figsize=(30,30))
    plt.subplot(1, K, i+1)
        I = io.imread(train_list[ind[0,i]][0])
        box = bounding_box[train_list[ind[0,i]][1]+1].split(" ")
    fig,ax = plt.subplots(1)
    plt.axis('off')
    plt.imshow(I)

```

```

plt.suptitle(os.path.basename(train_list[ind[0,i]][0]), fontsize=10)
rect =
patches.Rectangle((float(box[1]),float(box[2])),float(box[3]),float(box[4]),linewidth=1,edgecolor
='r',facecolor='none')
ax.add_patch(rect)
plt.show()
img =Image.open("274.jpg")
photo=ImageTk.PhotoImage(img)
lab=Label(image=photo).place(x=0,y=0)
font = ('times', 20, 'bold')
title = Label(main, text='Bird Species Identification using Deep Learning')
title.config(bg='black', fg='white')
title.config(font=font)
title.config(height=2, width=80)
title.place(x=5,y=5)
font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload an Image", command=upload)
upload.place(x=70,y=100)
upload.config(fg='blue')
upload.config(font=font1)
depthbutton = Button(main, text="Run to Identify Bird Species", command=DCNN)
depthbutton.place(x=280,y=100)
depthbutton.config(fg='red')
depthbutton.config(font=font1)
pathlabel = Label(main)
pathlabel.config(bg='#b0b0b0',fg='black')
pathlabel.config(font=font1)
pathlabel.place(x=70,y=140)
font = ('times', 20, 'bold')
title = Label(main, text="DNR COLLEGE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING")

```

```
title.config(bg='black', fg='white')
title.config(font=font)
title.config(height=2, width=80)
title.place(x=5,y=620)
main.config(bg='#b0b0b0')
main.mainloop()
```

4.4 OUTPUT SCREENS

In order to predict the probability of patients having heart disease, a confusion matrix (Table 1) was created, where A denotes patients with heart disease, and B denotes patients with no heart disease.

Table 1 A confusion matrix

	A (patients with heart disease)	B (patients with no heart disease)
A (patients with heart disease)	TP	FN
B (patients with no heart disease)	FP	TN

Abbreviations: TP, true positive; FN, false negative; FP, false positive; TN, true negative.

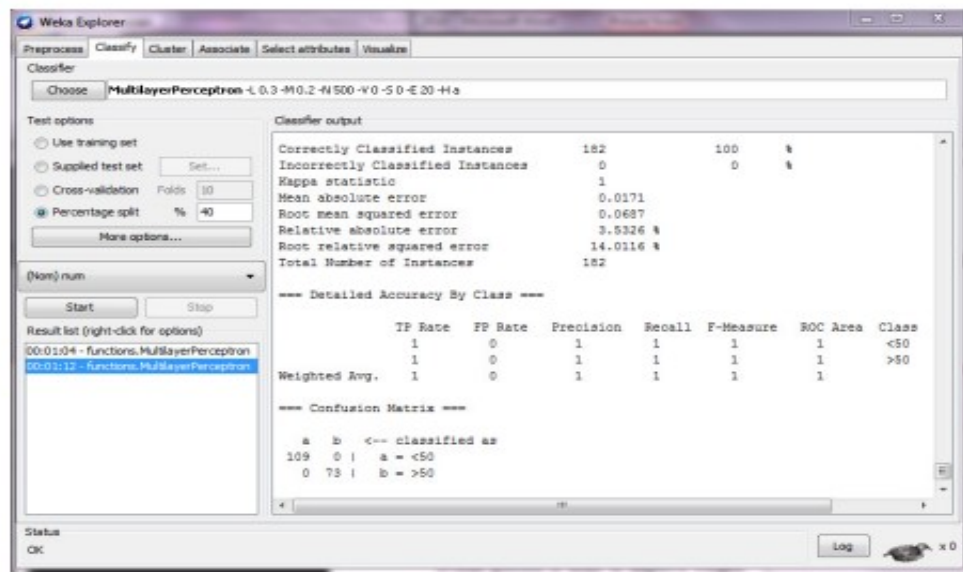
Table 2 Description of 15 used parameters

S no	Parameters	Parameter description	Values
1	age	Age in years	Continuous
2	sex	Male or female	1= male 0= female
3	threstbps	Resting blood pressure	Continuous value in mmHg
4	cp	Chest pain type	1= typical type I 2= typical type angina 3= non-angina pain 4= asymptomatic
5	chol	Serum cholesterol	Continuous value in mm/dL
6	fbx	Fasting blood sugar	1 \geq 120 mg/dL 0 \leq 120 mg/dL
7	restecg	Resting electrographic results	0= normal 1= having ST-T wave abnormal 2= left ventricular hypertrophy
8	thalach	Maximum heart rate achieved	Continuous value
9	old peak	ST depression induced by exercise relative to rest	Continuous value
10	exang	Exercise induced angina	0= no 1= yes
11	ca	Number of major vessels colored by fluoroscopy	0–3 value
12	slope	Slope of the peak exercise ST segment	1= unsloping 2= flat 3= downsloping
13	thal	Defect type	3= normal 6= fixed 7= reversible defect
14	obes	Obesity	1= yes 0= no
15	num	Diagnosis of heart disease	0% \leq 50% 1% > 50%

Table 3 Results for neural network showing 100% accuracy

	A (patients with heart disease)	B (patients with no heart disease)
A (patients with heart disease)	109 (TP)	0 (FN)
B (patients with no heart disease)	0 (FP)	73 (TN)

Abbreviations: TP, true positive; FN, false negative; FP, false positive; TN, true negative.



Results showing accuracy for 15 parameters using Weka tool.

A confusion matrix contains information about real and predicted classifications done by a classification system. The data in the matrix are evaluated to know the performance of such systems. The confusion matrix contains the following four entries:

*TP (true positive): The number of records classified as true while they were actually true.

*FP (false positive): The number of records classified as true while they were actually false.

*FN (false negative): The number of records classified as false while they were actually true.

*TN (true negative): The number of records classified as false while they were actually false.

The overall process of effective heart disease prediction system (EHDPS) is based on the following three steps:

1. Data collection
2. Data pre-processing and
3. The classification of data.

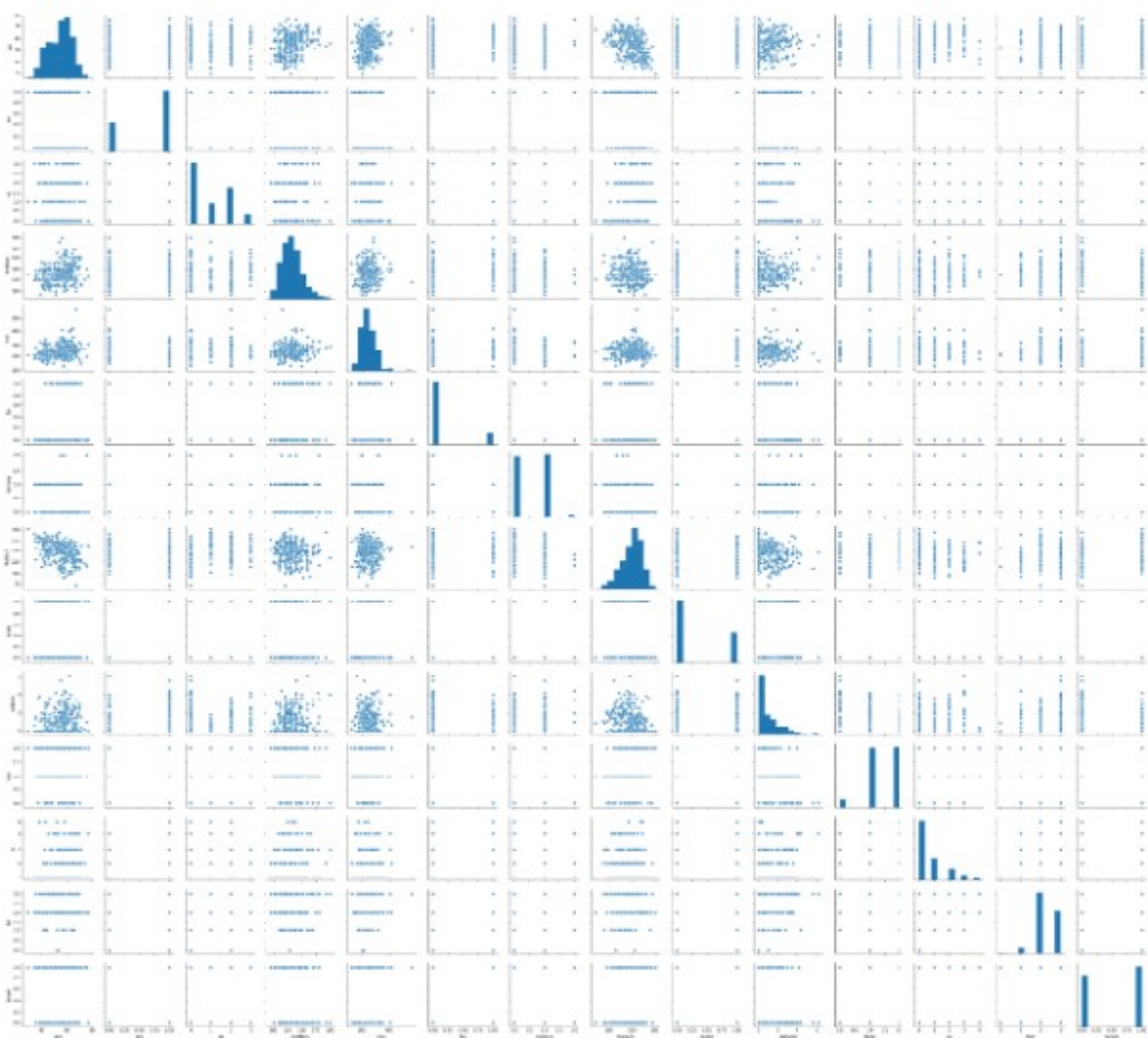
The data are collected from a standard dataset that contains 303 records. The 15 parameters, such as age, sex, chest pain type (CP), and cholesterol (chol), with some domain values associated with them, considered to predict the probability of heart disease are shown in Table 2. The collected data were used to create a structured database system. The pre-processing was done by identifying the associated fields and removing all the duplications. After that, all the missing values were filled, and the data were coded according to the domain value. After applying neural networks on training dataset, the results show that there are zero FN or FP entries (Table 3), suggesting that the system predicts heart disease with 100% accuracy. Figure 2 shows the actual work done by Weka 3.6.11 tool.

Exploratory Data Analysis (EDA)

First, analysing the target variable:

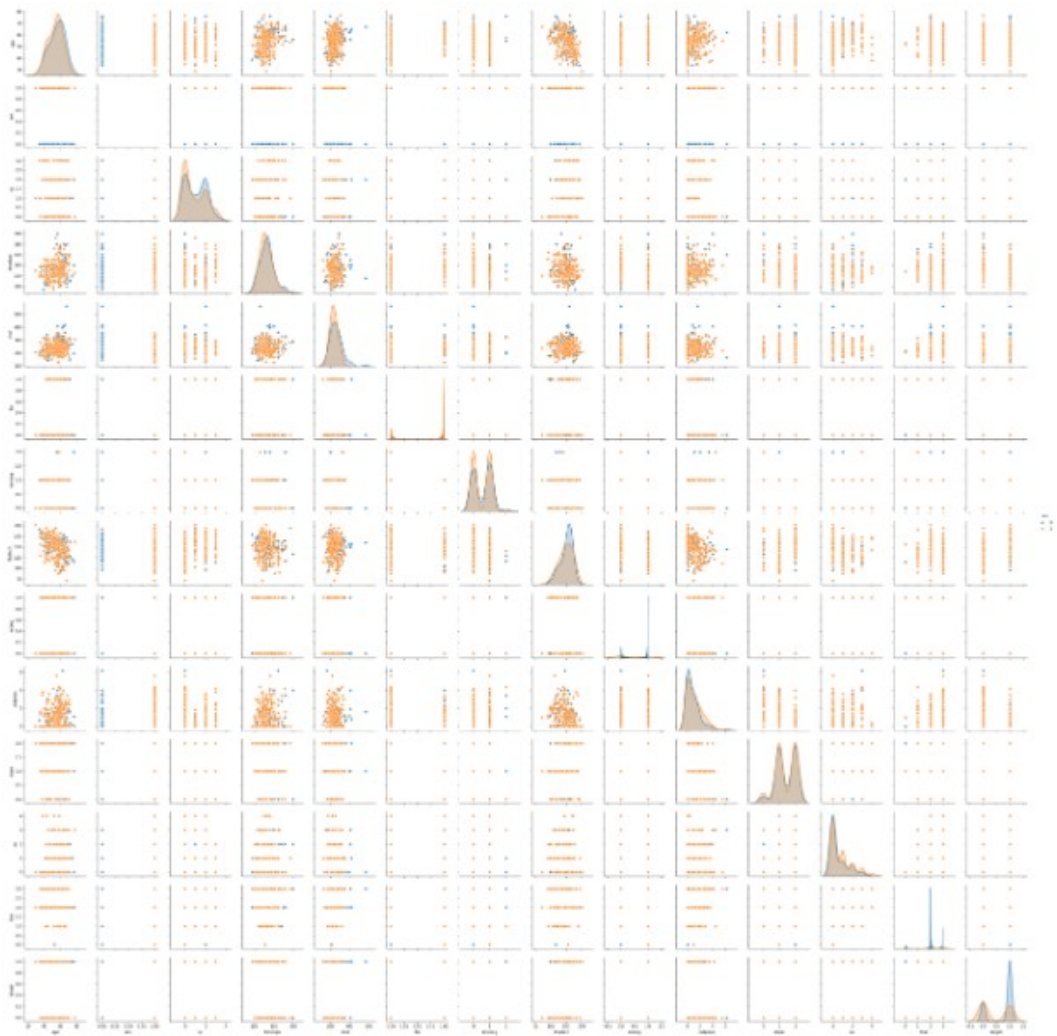
```
sns.pairplot(dataset)
```

<seaborn.axisgrid.PairGrid at 0x22f765f0108>



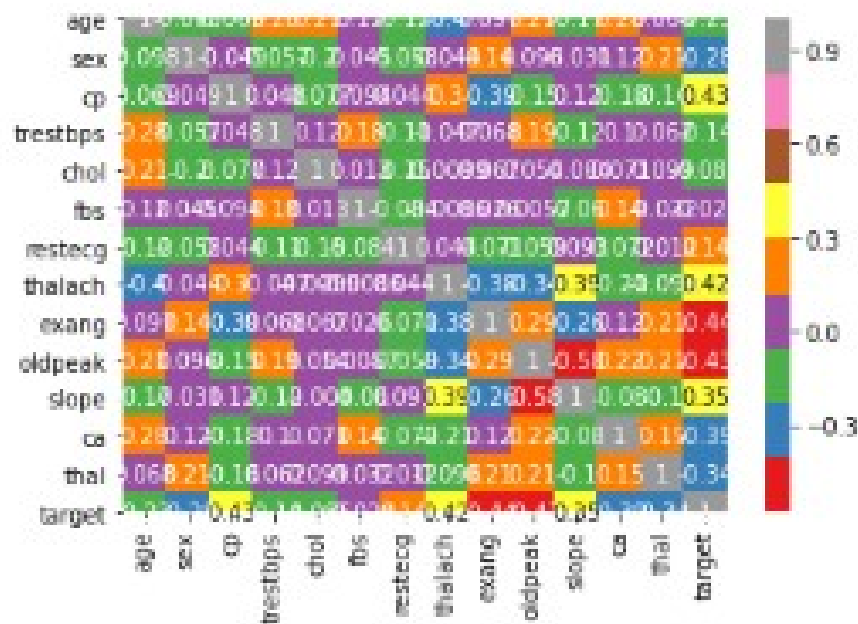
In: - `sns.pairplot(dataset, hue='sex')`

Out: - <seaborn.axisgrid.PairGrid at 0x22f7f416148>



In: `sns.heatmap(dataset.corr(),cmap='Set1',annot=True)`

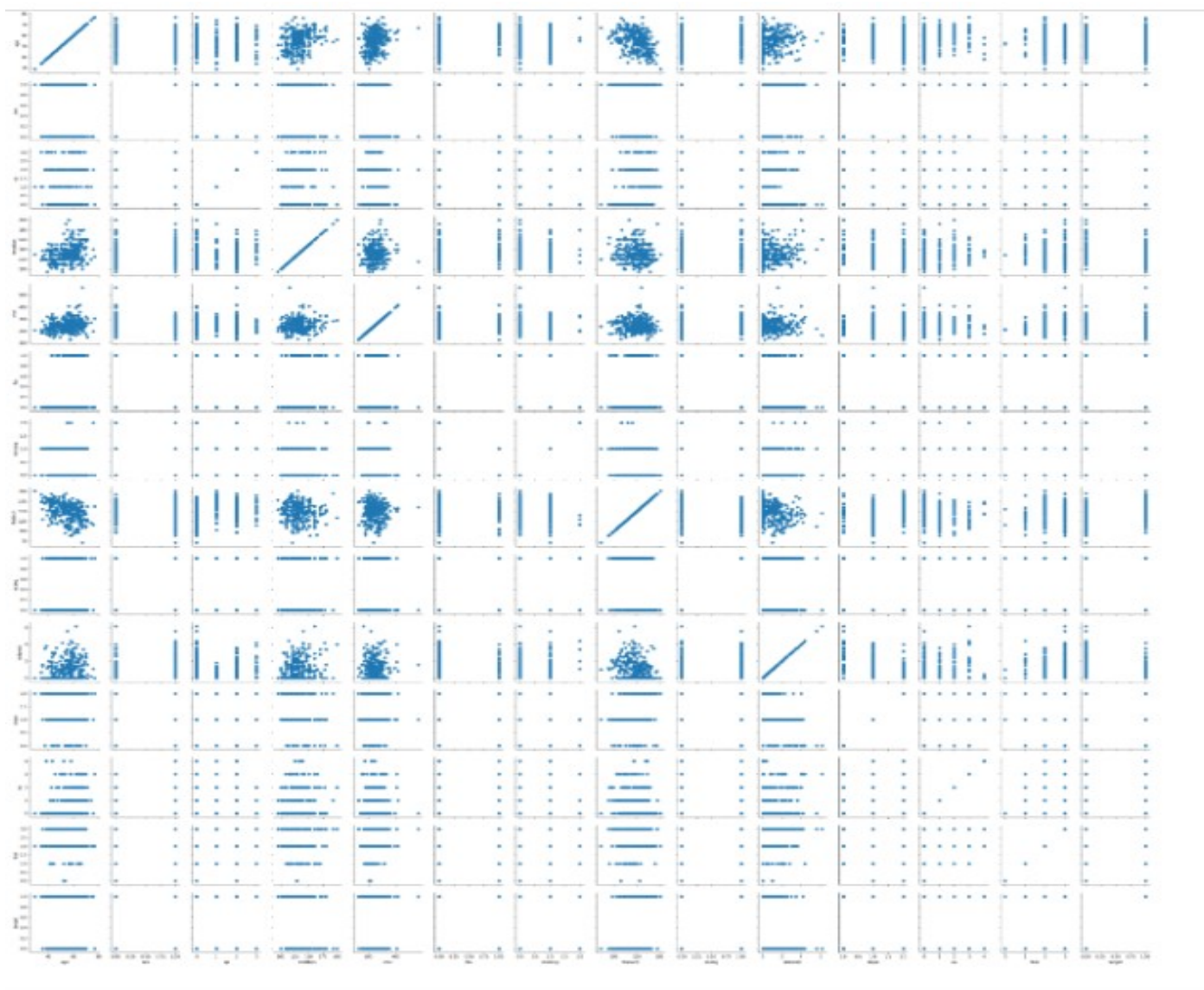
Out:- `<matplotlib.axes._subplots.AxesSubplot at 0x22f09ffce88>`



In:- g = sns.PairGrid(dataset)

g.map(plt.scatter)

out:- <seaborn.axisgrid.PairGrid at 0x1c8aac15f88>

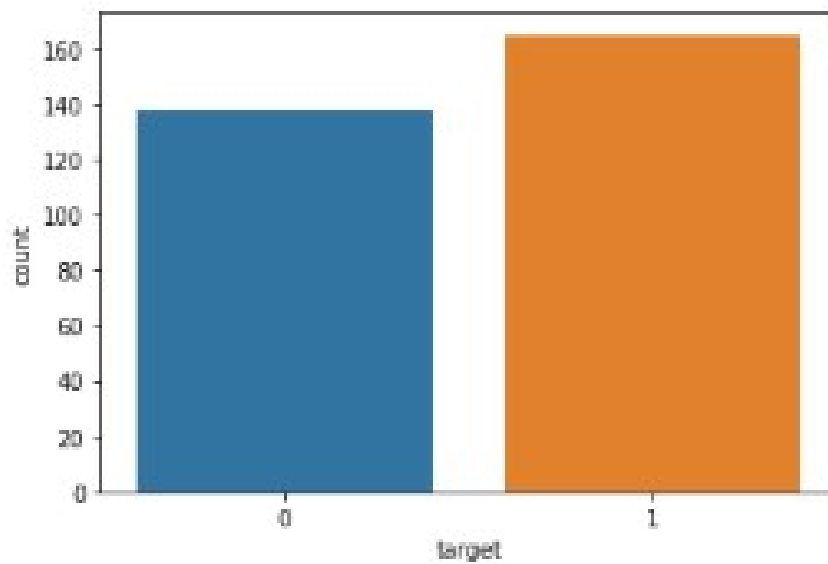


```
In:- y = dataset["target"]  
sns.countplot(y)  
target_temp = dataset.target.value_counts()  
print(target_temp)
```

```
1    165
```

```
0    138
```

```
Name: target, dtype: int64
```



We'll analyse 'sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca' and 'thal' features

Analysing the 'Sex' feature

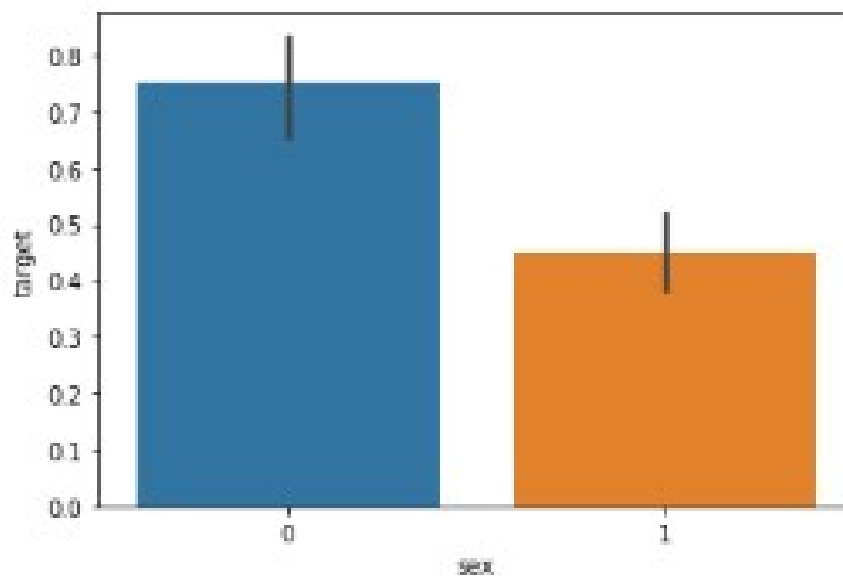
In: `dataset["sex"].unique()`

Out: `array([1, 0], dtype=int64)`

Note: the 'sex' feature has 2 unique features

In: `sns.barplot(dataset["sex"],y)`

Out: `<matplotlib.axes._subplots.AxesSubplot at 0x22f0a34df08>`



Analysing the 'Chest Pain Type' feature

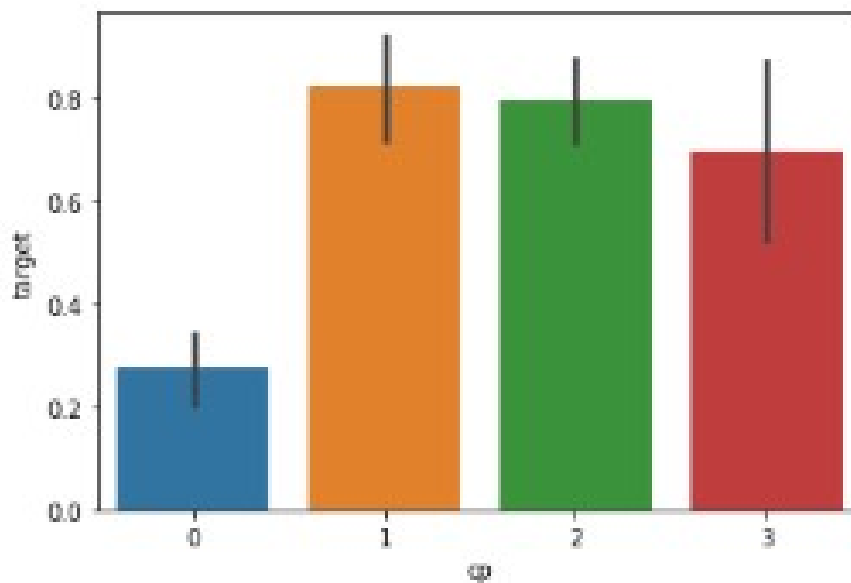
In: dataset["cp"].unique()

Out: array([3, 2, 1, 0], dtype=int64)

As expected, the CP feature has values from 0 to 3

In: sns.barplot(dataset["cp"],y)

Out: <matplotlib.axes._subplots.AxesSubplot at 0x22f0a34dc48>



Analysing the FBS feature

In: dataset["fbs"].describe()

Out:

```
count    303.000000
mean      0.148515
std       0.356198
min       0.000000
25%      0.000000
50%      0.000000
75%      0.000000
max       1.000000
```

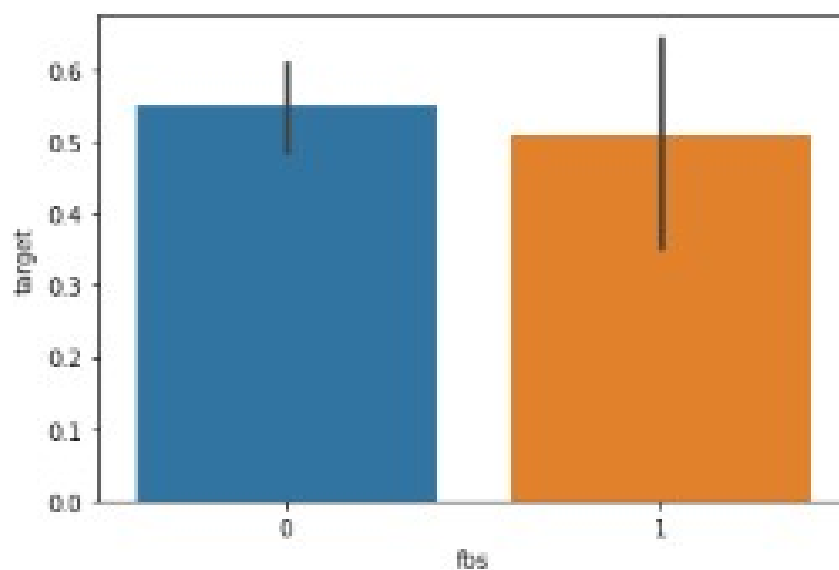
Name: fbs, dtype: float64

In: dataset["fbs"].unique()

Out: array([1, 0], dtype=int64)

In: sns.barplot(dataset["fbs"],y)

Out: <matplotlib.axes._subplots.AxesSubplot at 0x22f0a52c7c8>



We realize that people with restecg '1' and '0' are much more likely to have a heart disease than with restecg '2'

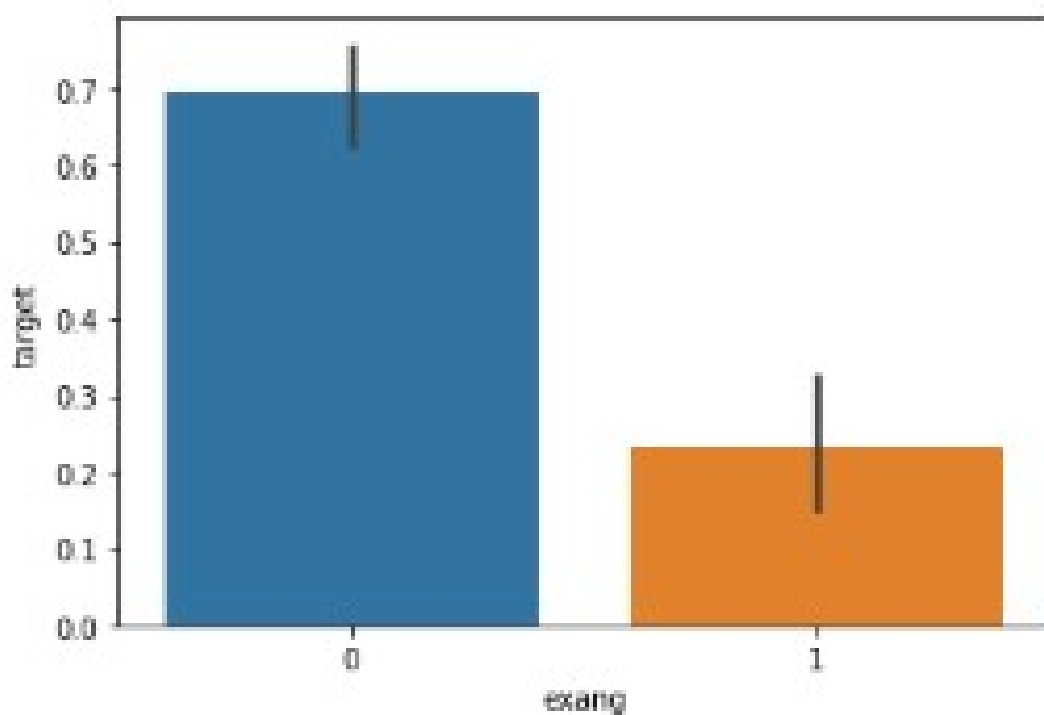
Analysing the 'exang' feature

```
In: dataset["exang"].unique()
```

```
Out: array([0, 1], dtype=int64)
```

```
In: sns.barplot(dataset["exang"],y)
```

```
Out: <matplotlib.axes._subplots.AxesSubplot at 0x1c8b5d52d88>
```



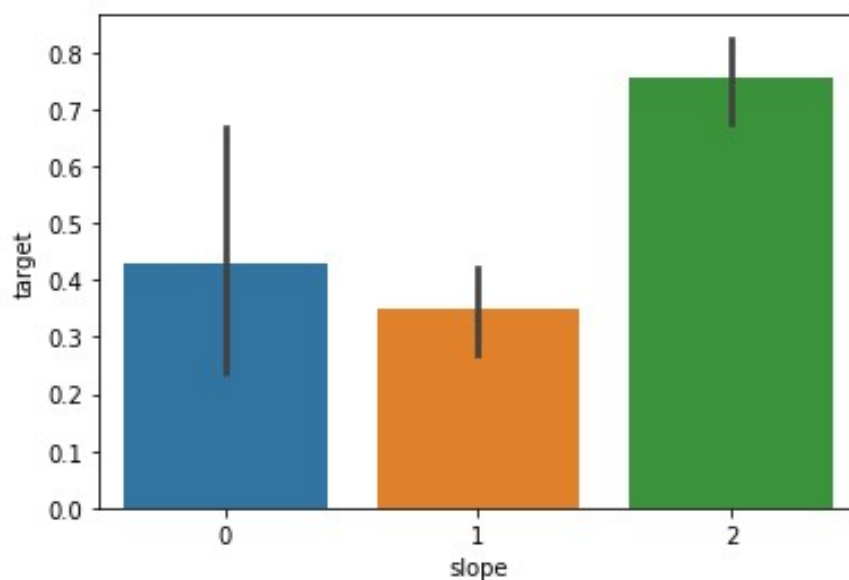
Analysing the Slope feature

In: dataset["slope"].unique()

Out: array([0, 2, 1], dtype=int64)

In: sns.barplot(dataset["slope"],y)

Out : <matplotlib.axes._subplots.AxesSubplot at 0x1c8b6d85a88>



We observe, that Slope '2' causes heart pain much more than Slope '0' and '1'

Analysing the 'ca' feature

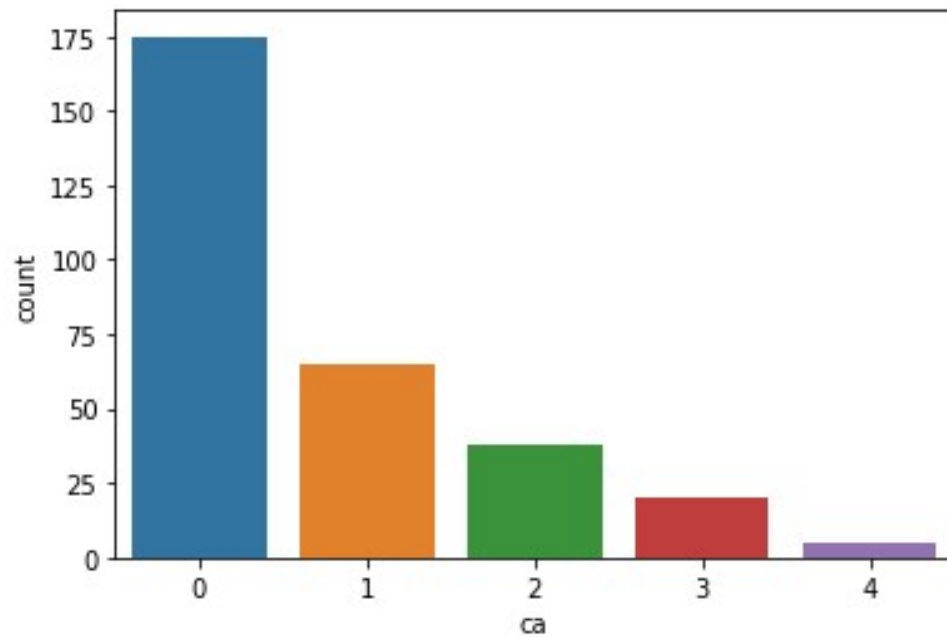
#number of major vessels (0-3) colored by flourosopy

In: dataset["ca"].unique()

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

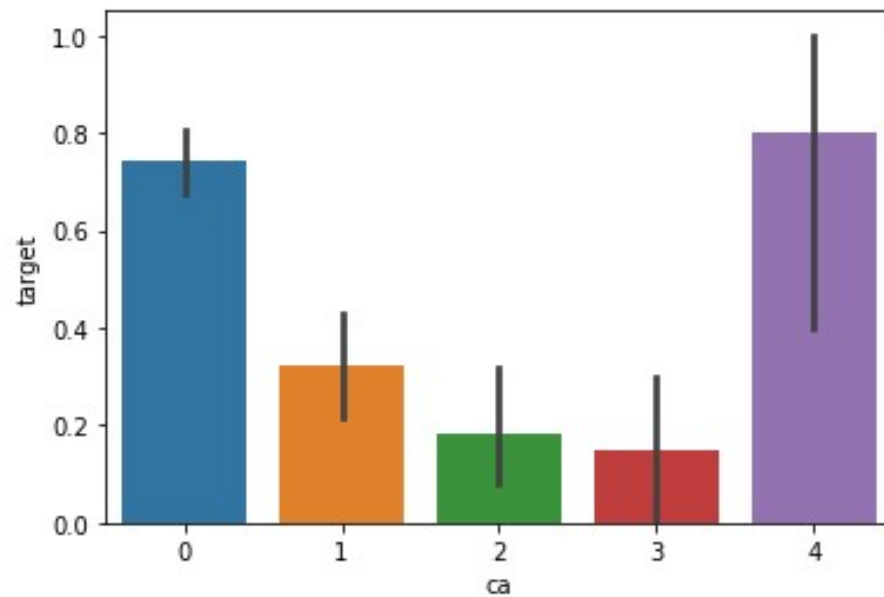
In: sns.countplot(dataset["ca"])

Out: <matplotlib.axes._subplots.AxesSubplot at 0x1c8b6dd14c8>



In: sns.barplot(dataset["ca"],y)

Out: <matplotlib.axes._subplots.AxesSubplot at 0x1c8b5d56588>



ca=4 has astonishingly large number of heart patients

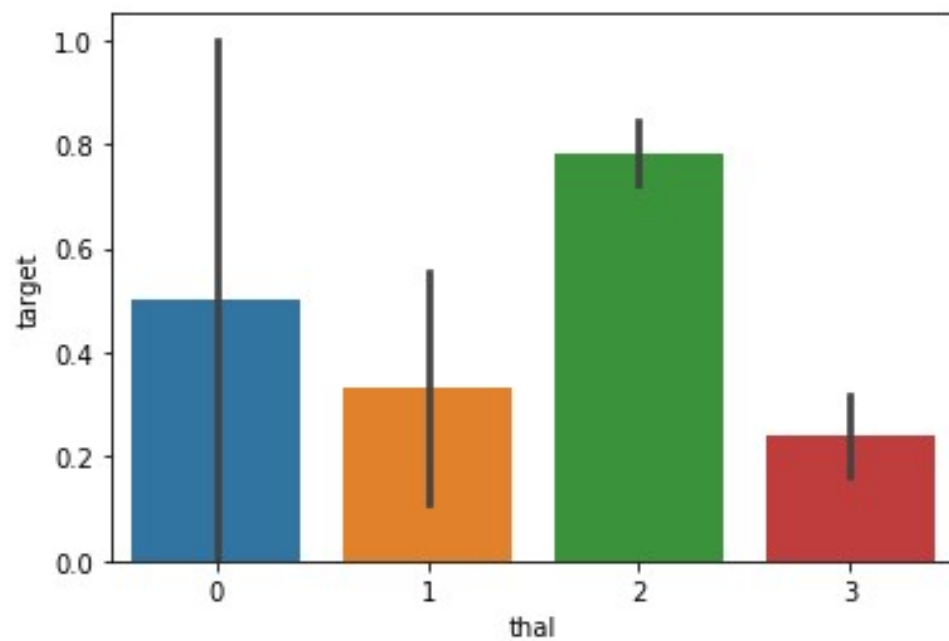
In: ### Analysing the 'thal' feature

In: dataset["thal"].unique()

Out: array([1, 2, 3, 0], dtype=int64)

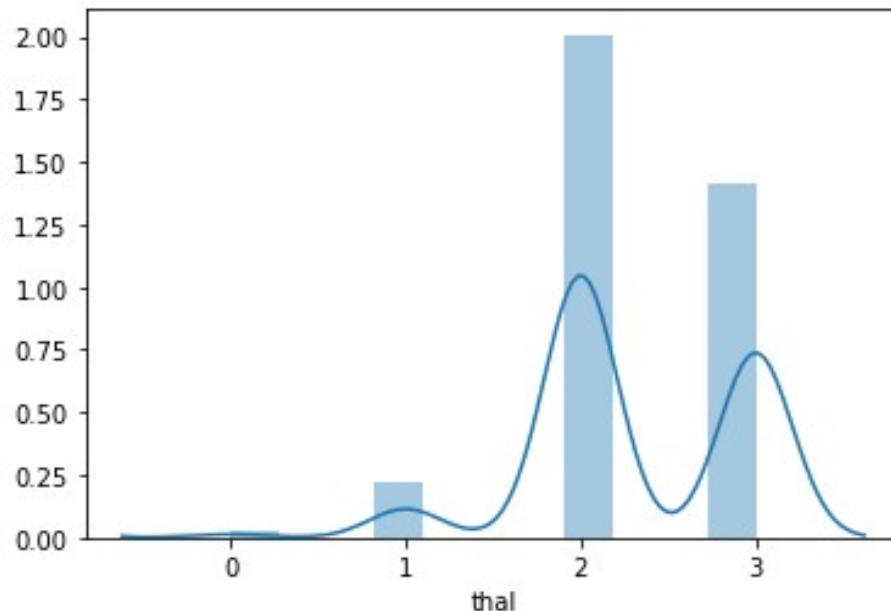
In: sns.barplot(dataset["thal"],y)

Out: <matplotlib.axes._subplots.AxesSubplot at 0x1c8b6ed7788>



In: sns.distplot(dataset["thal"])

Out : <matplotlib.axes._subplots.AxesSubplot at 0x1c8b6f36908>



Output final score

```
In:                                     scores                                     =  
[score_lr,score_nb,score_svm,score_knn,score_dt,score_rf,score_xgb,score_nn]  
algorithms = ["Logistic Regression","Naive Bayes","Support Vector  
Machine","K-Nearest Neighbors","Decision Tree","Random  
Forest","XGBoost","Neural Network"]  
for i in range(len(algorithms)):  
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+"  
    %")
```

The accuracy score achieved using Logistic Regression is: 85.25 %

The accuracy score achieved using Naive Bayes is: 85.25 %

The accuracy score achieved using Support Vector Machine is: 81.97 %

The accuracy score achieved using K-Nearest Neighbors is: 67.21 %

The accuracy score achieved using Decision Tree is: 81.97 %

The accuracy score achieved using Random Forest is: 95.08 %

The accuracy score achieved using XGBoost is: 85.25 %

The accuracy score achieved using Neural Network is: 83.61 %

In:

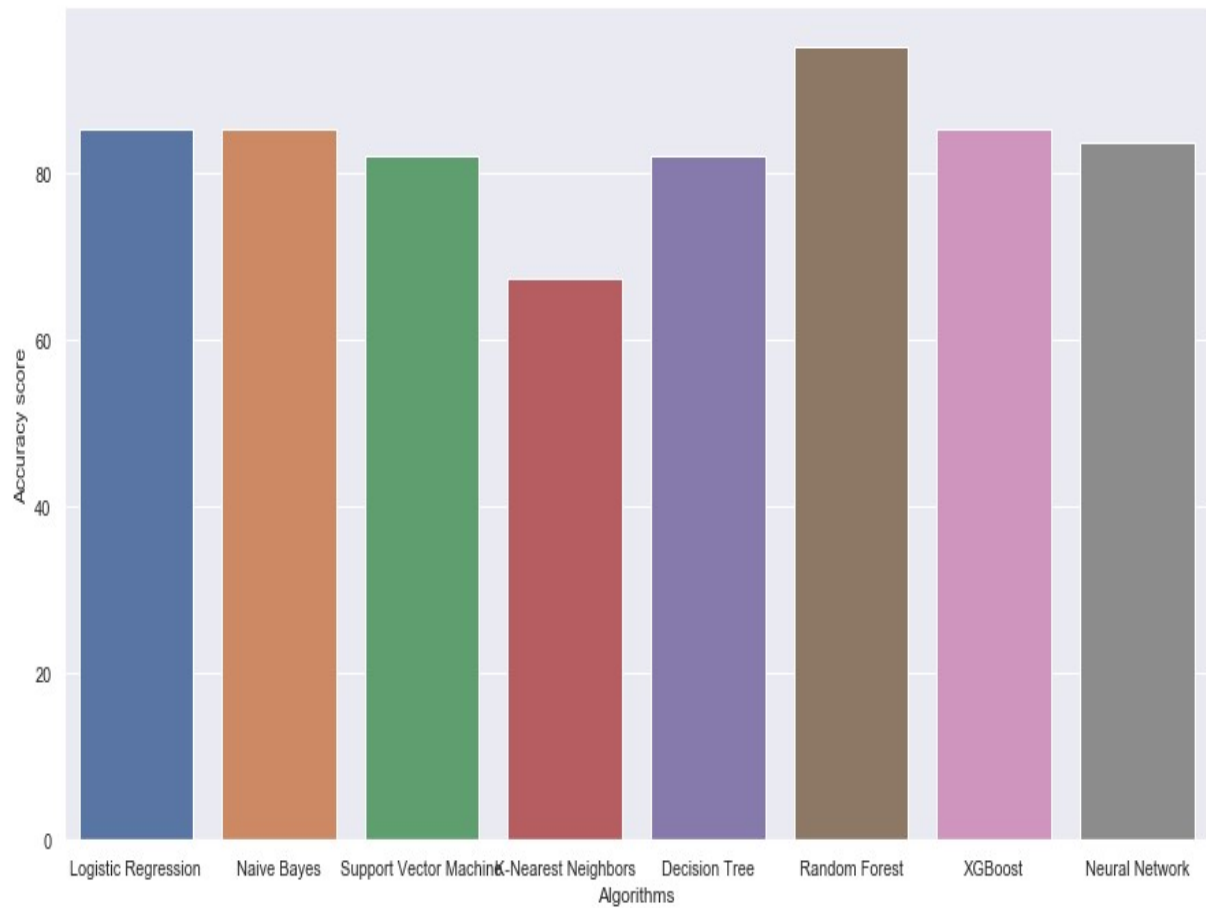
```
sns.set(rc={'figure.figsize':(15,8)})  
plt.xlabel("Algorithms")  
plt.ylabel("Accuracy score")
```

```
sns.barplot(algorithms,scores)
```

Out:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c8beff5188>
```


Comparison of All ML Classification Algorithms



5. SYSTEM TESTING

5.1 TESTING CONCEPTS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.2 TESTING STRATEGIES

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

5.3 TEST CASES

Negative Test Cases

Test Case 1: Dataset Selection	Priority(H.L):High
Test Objective: to check Dataset Selection success or fail	
Test Description: In this HOME screen, when the user selects the heart disease dataset as input	
Requirement Verified: Yes	
Test Environment: System connected with the dataset.	
Actions	Expected Results
<ul style="list-style-type: none"> when the user selects the dataset by clicking upload button. 	<ul style="list-style-type: none"> Uploading dataset fail Select valid Dataset
Pass: no Condition Pass: No Fail: Yes	
Problems/Issues: Yes	
Notes: Selection is fail	

Test Case 2: Execute ML Algorithms	Priority(H.L):High
Test Objective: has to check whether algorithms are working	
Test Description: In this home page user dataset and after that he will check best ML classification algorithm which can find the heart disease	
Requirement Verified: No	
Test Environment: System connected with the dataset.	
Actions	Expected Results
<ul style="list-style-type: none"> Run ML algorithm by clicking Run button 	<ul style="list-style-type: none"> Comparing fail Select valid dataset and algorithms
Pass: no Condition Pass: No Fail: Yes	
Problems/Issues: Yes	

Notes: Algorithm fail

Positive Test Cases

Test Case 1: Dataset Selection	Priority(H.L):High
Test Objective: to check dataset Selection success or fail	
22252Test Description: In this HOME screen, when the user selects the input dataset it display path label	
Requirement Verified: Yes	
Test Environment: System connected with the dataset.	
Actions	Expected Results
<ul style="list-style-type: none">when the user selects the input by clicking upload button.	<ul style="list-style-type: none">Dataset path Label can be shown
Pass: yes Condition Pass: No Fail: No	
Problems/Issues: No	
Notes: Dataset Selection successfully completed	

Test Case 2: Running ML Algorithm	Priority(H.L):High
Test Objective: has to check ML algorithms related to trained Dataset	
22252Test Description: In this home page user will choose best ML algorithms for classification	
Requirement Verified: No	
Test Environment: System connected with the dataset.	
Actions	Expected Results
<ul style="list-style-type: none">Run ML algorithm by clicking Run button	<ul style="list-style-type: none">Display Features of Heart Patients who are relatively matched
Pass: yes Condition Pass: No Fail: Yes	
Problems/Issues: No	
Notes: ML Classification Algorithm successful.	

6. CONCLUSION

The prediction of heart disease by using different data mining tools. Life is dependent on efficient working of heart because heart is essential part of our body. If operation of heart is not proper, it will affect the other body parts of human such as brain, kidney etc. Heart disease is a disease that affects on the operation of heart. we survey different papers in which one or more algorithms of data mining used for the prediction of heart disease. Result from using neural networks is nearly 100% in heart disease. So that the prediction by using data mining algorithm given efficient results. Applying data mining techniques to heart disease treatment data can provide as reliable performance as that achieved in diagnosing heart disease.

7. REFERENCES

- [1] Mohammad Taha Khan, Dr. Shamimul Qamar and Laurent F. Massin, A Prototype of Cancer/Heart Disease Prediction Model Using Data Mining, International Journal of Applied Engineering Research, 2012.
- [2] Ma.jabbar, Dr.priti Chandra, B.L.Deekshatulu, cluster based association rule mining for heart attack prediction, Journal of Theoretical and Applied Information Technology, 2011.
- [3] Ms. Ishtake S.H ,Prof. Sanap S.A., “Intelligent Heart Disease Prediction System Using Data Mining Techniques”, International J. of Healthcare & Biomedical Research, 2013.
- [4] Dr. K. Usha Rani, analysis of heart diseases dataset using neural network approach, International Journal of Data Mining & Knowledge Management Process, 2011.
- [5] Carlos Ordonez, Edward Omiecinski, Mining Constrained Association Rules to Predict Heart Disease, IEEE. Published in International Conference on Data Mining (ICDM), p. 433-440, 2001.
- [6] Nidhi Bhatla Kiran Jyoti, An Analysis of Heart Disease Prediction using Different Data Mining Techniques, International Journal of Engineering Research & Technology (IJERT), 2012.
- [7] Shantakumar B.Patil, Dr.Y.S. Kumaraswamy, Extraction of Significant Patterns from Heart Disease Warehouses for Heart Attack Prediction, (IJCSNS) International Journal of Computer Science and Network 228 Security ,2009.
- [8] Abhishek taneja, Heart Disease Prediction System Using Data Mining Techniques, Oriental Scientific Publishing Co., India, 2013.

- [9] M. Anbarasi, E. Anupriya, N.ch.s.n.Iyengar, Enhanced Prediction of Heart Disease with Feature Subset Selection using Genetic Algorithm, International Journal of Engineering Science and Technology, 2010.
- [10] Miss. Chaitrali S. Dangare, Dr. Mrs. Sulabha S. Apte, A data mining approach for prediction of heart disease using neural networks, international journal of computer engineering and technology, 2012.
- [11] N. Aditya Sundar, P. Pushpa Latha, M. Rama Chandra, performance analysis of classification data mining techniques over heart diseases data base, international journal of engineering science and advanced technology, 2012.
- [12] Shadab Adam Pattekari and Asma Parveen, prediction system for heart disease using naïve bayes, International Journal of Advanced Computer and Mathematical Sciences, 2012.
- [13] Latha Parthiban and R.Subramanian, Intelligent Heart Disease Prediction System using CANFIS and Genetic Algorithm, International Journal of Biological and Medical Sciences, 2008.
- [14] Jesmin Nahar, Tasadduq Imama, Kevin S. Tickle, Yi-Ping Phoebe Chen, Association rule mining to detect factors which contribute to heart disease in males and females, Elsevier, 2013.



INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | ISSN: 2320 - 2882

An International Open Access, Peer-reviewed, Refereed Journal

The Board of
International Journal of Creative Research Thoughts
Is hereby awarding this certificate to

SANDHYA GANDI

In recognition of the publication of the paper entitled
**COMPARISON OF SEVERAL ML ALGORITHMS FOR EFFECTIVE HEART
DISEASE PREDICTION**

Published In IJCRT (www.ijert.org) & 7.97 Impact Factor by Google Scholar

Volume 10 Issue 8 August 2022 , Date of Publication: 17-August-2022

UGC Approved Journal No: 49025 (18)

PAPER ID : IJCRT2208291

Registration ID : 224468

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly Journal

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT

An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal

Website: www.ijert.org | Email id: editor@ijert.org | ESTD: 2013



EDITOR IN CHIEF



INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | ISSN: 2320 - 2882

An International Open Access, Peer-reviewed, Refereed Journal

The Board of
International Journal of Creative Research Thoughts
Is hereby awarding this certificate to

VADAMODULA VIJAY KUMAR

In recognition of the publication of the paper entitled
**COMPARISON OF SEVERAL ML ALGORITHMS FOR EFFECTIVE HEART
DISEASE PREDICTION**

Published In IJCRT (www.ijert.org) & 7.97 Impact Factor by Google Scholar

Volume 10 Issue 8 August 2022 , Date of Publication: 17-August-2022

UGC Approved Journal No: 49025 (18)

PAPER ID : IJCRT2208291

Registration ID : 224468

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly Journal

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT

An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal

Website: www.ijert.org | Email id: editor@ijert.org | ESTD: 2013



EDITOR IN CHIEF