DATA SCIENCE

(Prediction of heart disease occurrence)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science Engineering

By

Kannekanti Neha

221710313020

Department of Computer Science and Engineering

GITAM School of Technology

GITAM(Deemed to be University)

Hyderabad-502329

July 2020

DECLARATION

I submit this industrial training work entitled "**PREDICTION OF HEART DISEASE OCCURRENCE**" to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology**" in "**Computer Science and Engineering**". I declare that it was carried out independently by me under the guidance of **Mr. ______**, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.
The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place : HYDERABAD **Kannekanti Neha**
Date: **221710313020**

Certificate

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected Dr. N. Siva Prasad, Pro Vice Chancellor, GITAM Hyderabad and Dr. CH. Sanjay, Principal, GITAM Hyderabad.

I would like to thank respected Prof. S. Phani Kumar, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present the internship report. It helped me alot to realize what we study for.

I would like to thank the respected faculties ________ who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Kannekanti Neha
221710313020

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on heart data set My perception of understanding the given data set has been in the view of undertaking a client's requirement of overcoming the predictions on whether a person is suffering from Heart Disease or not.

Health care field has a vast amount of data, for processing those data certain techniques are used. Data science is one of the techniques often used. Heart disease is the Leading cause of death worldwide. This System predicts the arising possibilities of Heart Disease. The outcomes of this system provide the chances of occurring heart disease in terms of percentage. The datasets used are classified in terms of medical parameters. This system evaluates those parameters using data science classification technique. The datasets are processed in python programming using four main Machine Learning Algorithms namely logistic regression , knn Algorithm and Naive Bayes Algorithm and svc which shows the best algorithm among these four in terms of accuracy level of heart disease.

TABLE OF CONTENTS

CHAPTER 5: DATA PREPROCESSING

CHAPTER 6: FEATURES SELECTION

CHAPTER 7: MODEL BUILDING AND EVALUATION

CHAPTER 8: TUNING THE HYPER PARAMETERS

LIST OF FIGURES:

CHAPTER 1

INFORMATION ABOUT DATA SCIENCE

1.1 WHAT IS DATA SCIENCE

Data Science is a blend of various tools, algorithms, and machine learning principles with the goal to discover hidden patterns from the raw data. How is this different from what statisticians have been doing for years?

The answer lies in the difference between explaining and predicting.

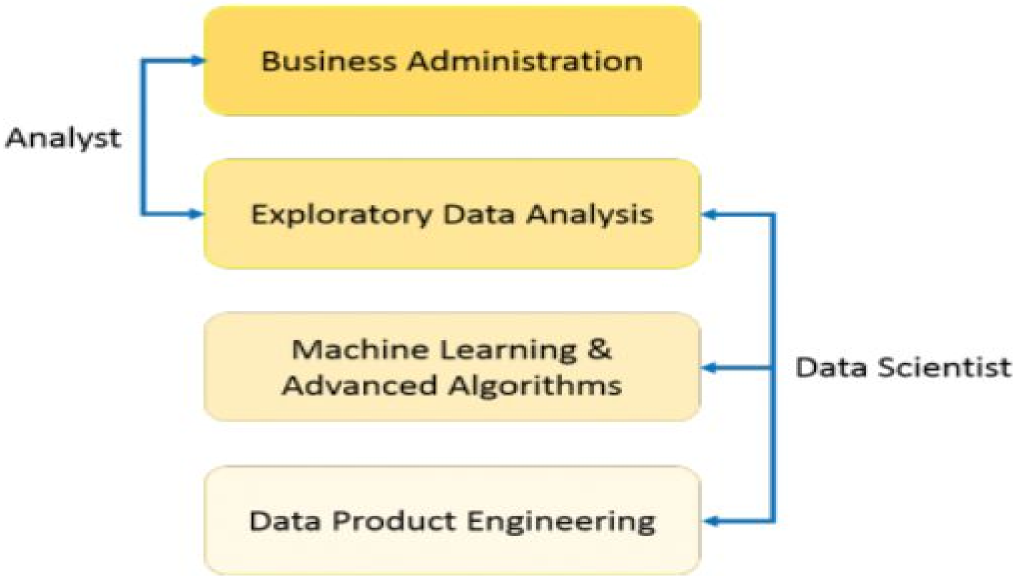


Fig.1.1.1: Difference between explaining and predicting

As you can see from the above image, a Data Analyst usually explains what is going on by processing the history of the data. On the other hand, Data Scientist not only does exploratory analysis to discover insights from it, but also uses various advanced machine learning algorithms to identify the occurrence of a particular event in the future. A Data Scientist will look at the data from many angles, sometimes angles not known earlier.

1.2 NEED OF DATA SCIENCE

Data Science is primarily used to make decisions and predictions making use of predictive causal analytics, prescriptive analytics (predictive plus decision science) and machine learning.

- Predictive causal analytics – If you want a model which can predict the possibilities of a particular event in the future, you need to apply predictive causal analytics. Say, if you are providing money on credit, then the probability of customers making future credit payments on time is a matter of concern for you. Here, you can build a model which can perform predictive analytics on the payment history of the customer to predict if the future payments will be on time or not.

- Prescriptive analytics: If you want a model which has the intelligence of taking its own decisions and the ability to modify it with dynamic parameters, you certainly need prescriptive analytics for it. This relatively new field is all about providing advice. In other terms, it not only predicts but suggests a range of prescribed actions and associated outcomes.

  The best example for this is Google's self-driving car which I had discussed earlier too. The data gathered by vehicles can be used to train self-driving cars. You can run algorithms on this data to bring intelligence to it. This will enable your car to take decisions like when to turn, which path to take, when to slow down or speed up.

- Machine learning for making predictions — If you have transactional data of a finance company and need to build a model to determine the future trend, then machine learning algorithms are the best bet. This falls under the paradigm of supervised learning. It is called supervised because you already have the data based on which you can train your machines. For example, a fraud detection model can be trained using a historical record of fraudulent purchases.

- Machine learning for pattern discovery — If you don't have the parameters based on which you can make predictions, then you need to find out the hidden patterns within the dataset to be able to make meaningful predictions. This is nothing but the unsupervised model as you don't have any predefined labels for grouping. The most common algorithm used for pattern discovery is Clustering.

Let's say you are working in a telephone company and you need to establish a network by putting towers in a region. Then, you can use the clustering technique to find those tower locations which will ensure that all the users receive optimum signal strength.

Let's see how the proportion of above-described approaches differ for Data Analysis as well as Data Science. As you can see in the image below, Data Analysis includes descriptive analytics and prediction to a certain extent. On the other hand, Data Science is more about Predictive Causal Analytics and Machine Learning.

1.3 USES OF DATA SCIENCE

1. Medical Image Analysis:

Procedures such as detecting tumors, artery stenosis, organ delineation employ various different methods and frameworks like MapReduce to find optimal parameters for tasks like lung texture classification. It applies machine learning methods, support vector machines (SVM), content-based medical image indexing, and wavelet analysis for solid texture classification.

2. Genetics & Genomics:

Data Science applications also enable an advanced level of treatment personalization through research in genetics and genomics. The goal is to understand the impact of the DNA on our health and find individual biological connections between genetics, diseases, and drug response. Data science techniques allow integration of different kinds of data with genomic data in the disease research, which provides a deeper understanding of genetic issues in reactions to particular drugs and diseases. As soon as we acquire reliable personal genome data, we will achieve a deeper understanding of the human DNA. The advanced genetic risk prediction will be a major step towards more individual care.

3. Drug Development:

The drug discovery process is highly complicated and involves many disciplines. The greatest ideas are often bounded by billions of testing, huge financial and time expenditure. On average, it takes twelve years to make an official submission.

Data science applications and machine learning algorithms simplify and shorten this process, adding a perspective to each step from the initial screening of drug compounds to the prediction of the success rate based on the biological factors. Such algorithms can forecast how the compound will act in the body using advanced mathematical modeling and simulations instead of the “lab experiments”. The idea behind the computational drug discovery is to create computer model simulations as a biologically relevant network simplifying the prediction of future outcomes with high accuracy.

4. Virtual assistance for patients and customer support:

Optimization of the clinical process builds upon the concept that for many cases it is not actually necessary for patients to visit doctors in person. A mobile application can give a more effective solution by bringing the doctor to the patient instead.The AI-powered mobile apps can provide basic healthcare support, usually as chatbots. You simply describe your symptoms, or ask questions, and then receive key information about your medical condition derived from a wide network linking symptoms to causes.This approach promotes a healthy lifestyle by encouraging patients to make healthy decisions, saves their time waiting in line for an appointment, and allows doctors to focus on more critical cases.

5.Internet Search:

Now, this is probably the first thing that strikes your mind when you think Data Science Applications.When we speak of search, we think 'Google'. Right? But there are many other search engines like Yahoo, Bing, Ask, AOL, and so on. All these search engines (including Google) make use of data science algorithms to deliver the best result for our searched query in a fraction of seconds. Considering the fact that, Google processes more than 20 petabytes of data every day.

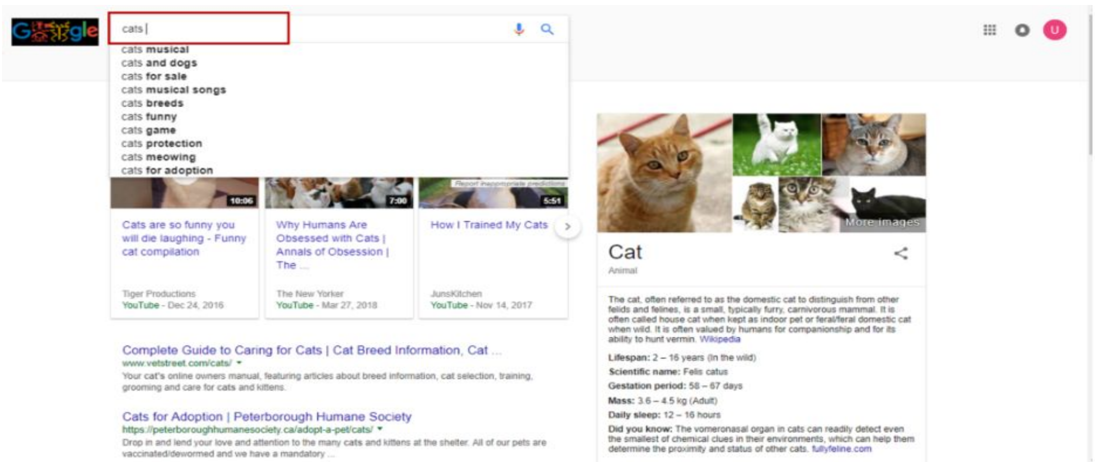


Fig.1.3.1: Internet search

6.Targeted Advertising:

If you thought Search would have been the biggest of all data science applications, here is a challenger – the entire digital marketing spectrum. Starting from the display banners on various websites to the digital billboards at the airports – almost all of them are decided by using data science algorithms.

This is the reason why digital ads have been able to get a lot higher CTR (Call-Through Rate) than traditional advertisements. They can be targeted based on a user's past behavior.

This is the reason why you might see ads of Data Science Training Programs while I see an ad of apparels in the same place at the same time.

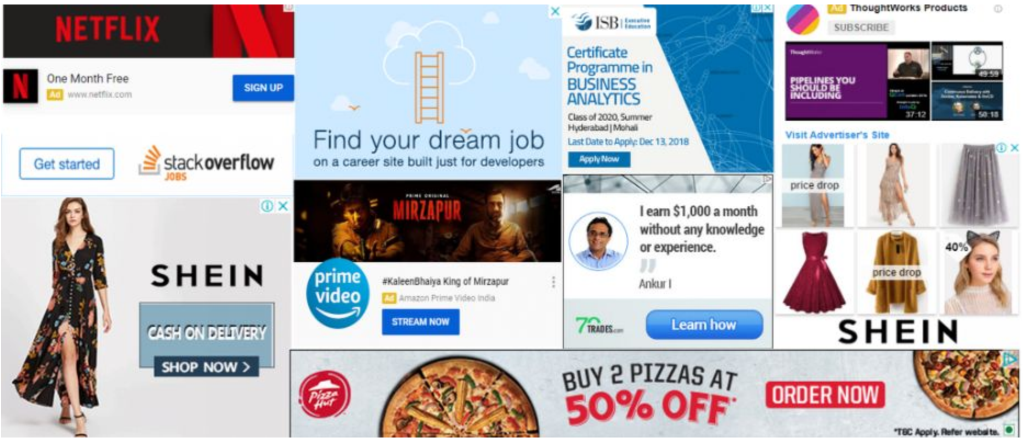


Fig 1.3.2: Targeted advertising

CHAPTER 2

INFORMATION ABOUT MACHINE LEARNING

2.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

2.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works:

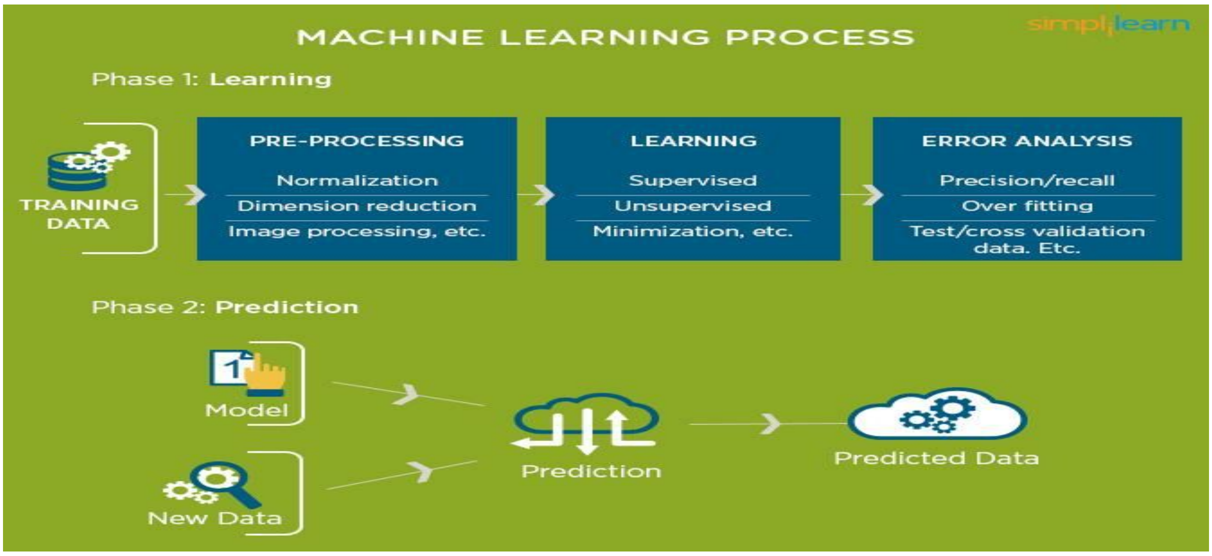


Fig 2.1.1 : The Process Flow

2.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data Traditionally, data analysis was always characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

2.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

2.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to “learn” how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

2.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

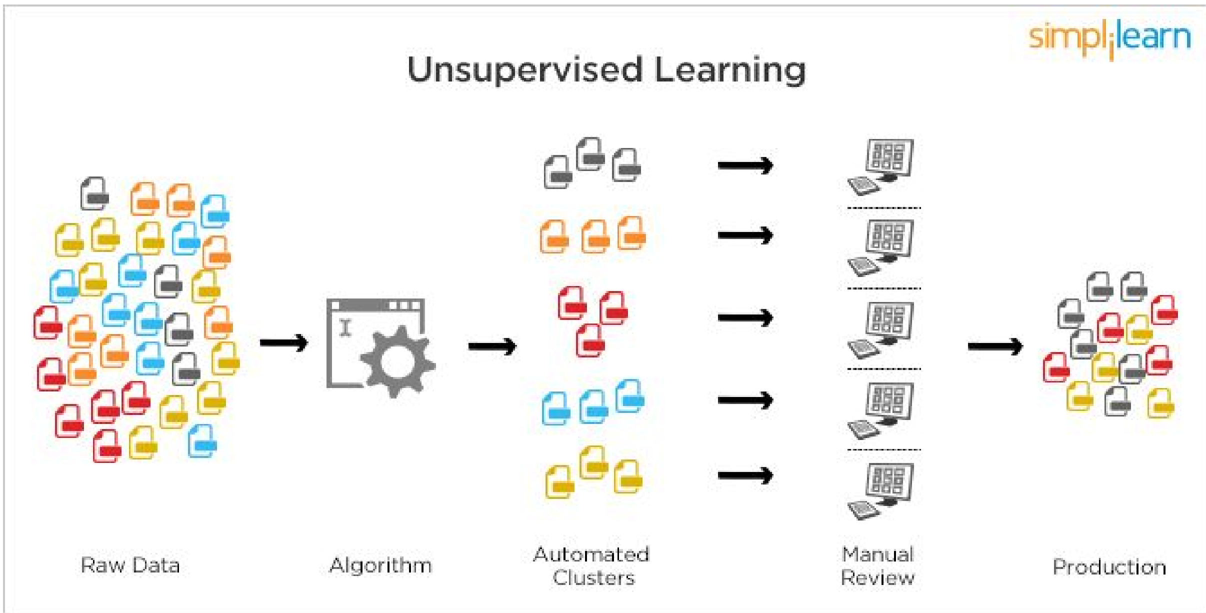


Fig 2.4.2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor maps, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

2.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

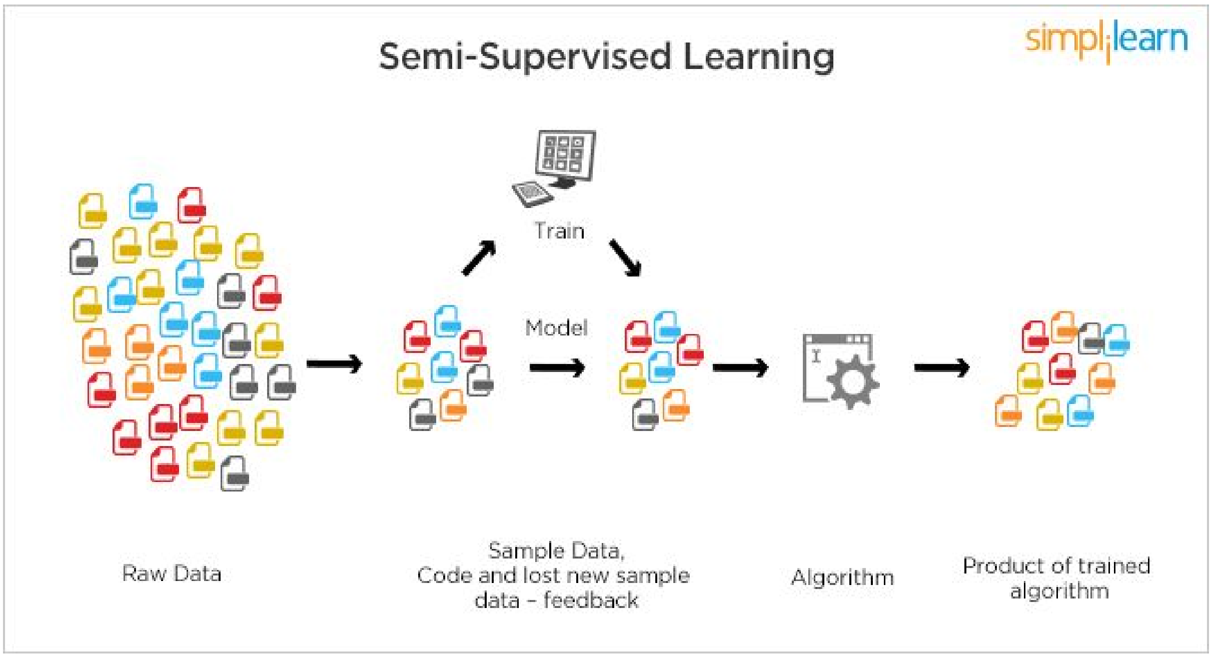


Fig 2.4.3 : Semi Supervised Learning

2.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovered previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 3

INFORMATION ABOUT PYTHON

Basic programming language used for machine learning is : PYTHON

3.1 INTRODUCTION TO PYTHON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles.
- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- **Python is Interactive:** You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's.
- Its latest version is 3.7 , it is generally called as python.

3.2 FEATURES OF PYTHON:

● **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.

● **Easy-to-read:** Python code is more clearly defined and visible to the eyes.

● **Easy-to-maintain**: Python's source code is fairly easy-to-maintaining.

● **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

● **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

● **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

● **Databases:** Python provides interfaces to all major commercial databases.

● **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

3.3 HOW TO SETUP PYTHON:

● Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

● The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

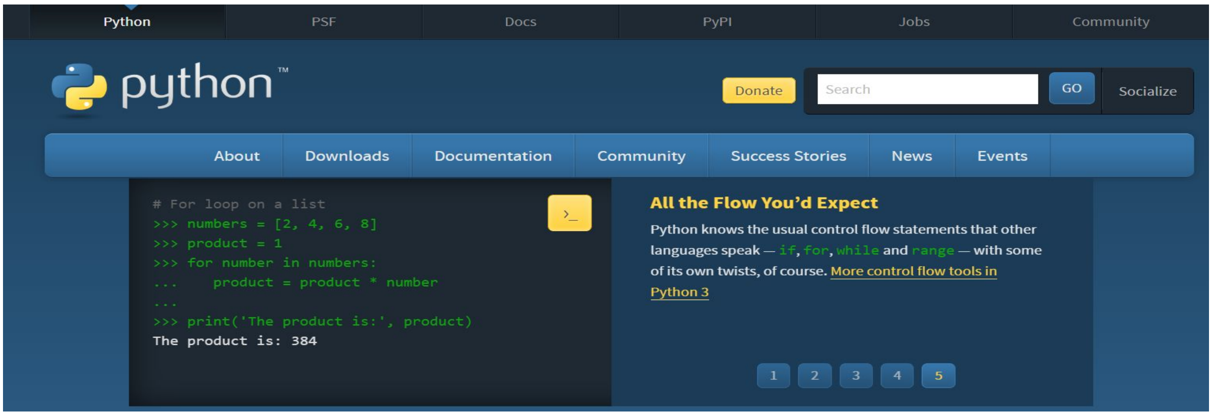


Fig 3.1.1 : Python download

Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager that quickly installs and manages packages.

In WINDOWS:

In windows,

- Step 1: Open Anaconda.com/downloads in a web browser.
- Step 2: Download python 3.4 version for (32-bits graphic/64 -bit graphic installer)
- Step 3: select installation type(all users)
- Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser).

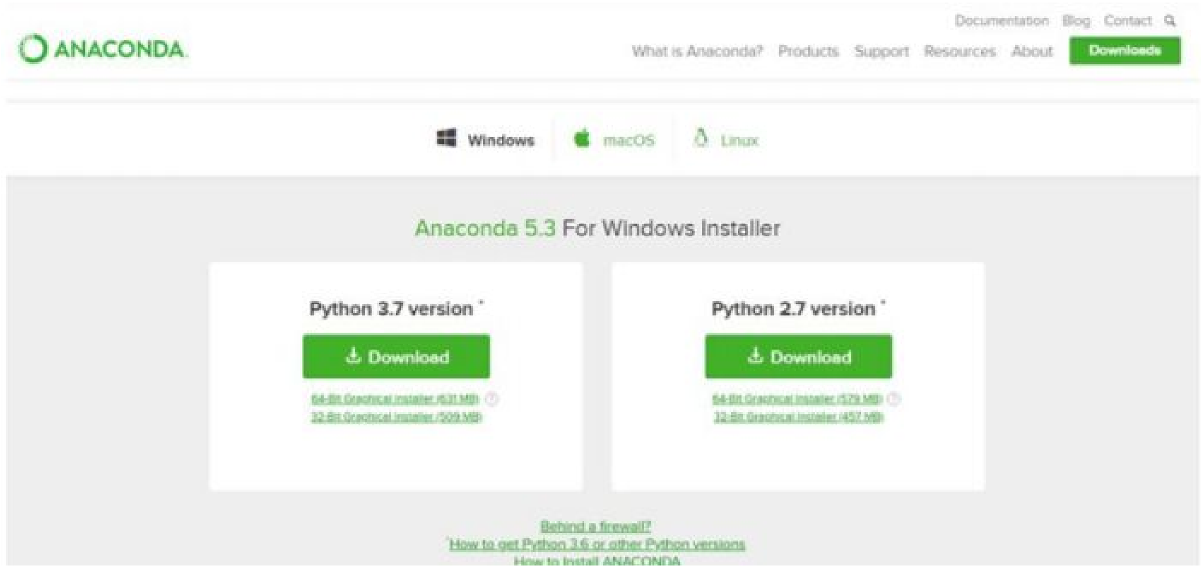


Fig 3.2.1 : Anaconda download

Fig 3.3.1 : Jupyter notebook

3.4 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

● Variables are nothing but reserved memory locations to store values.

● Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

● Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

● Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

● Python has five standard data types –

  o Numbers
  o Strings
  o Lists
  o Tuples
  o Dictionary

Python Numbers:

● Number data types store numeric values. Number objects are created when you assign a value to them.

● Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

Python Strings:

● Strings in Python are identified as a contiguous set of characters represented in the quotation marks.

● Python allows for either pairs of single or double quotes.

● Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

● The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

Python Lists:

● Lists are the most versatile of Python's compound data types

.● A list contains items separated by commas and enclosed within square brackets.

● To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data types.

● The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.

● The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples:

● A tuple is another sequence data type that is similar to the list.

● A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

● The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.

● Tuples can be thought of as read-only lists.

● For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't remove elements from a tuple. Tuples have no remove or pop method.

Python Dictionary:

● Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

● Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

● You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

● What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.5 PYTHON FUNCTION:

Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

3.6 PYTHON USING OOP's CONCEPTS

Class:

- **Class:** A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

- **Class variable:** A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

- **Data member**: A class variable or instance variable that holds data associated with a class and its objects.

- **Instance variable:** A variable that is defined inside a method and belongs only to the current instance of a class.

- **Defining a Class:**
  - o We define a class in a very similar way how we define a function.
  - o Just like a function ,we use parentheses and a colon after the class name when we define a class. Similarly, the body of our class is indented like a the function body is.

```python
def my_function():
    # the details of the
    # function go here
```

```python
class MyClass():
    # the details of the
    # class go here
```

Fig 3.4.1 : Defining a Class

__init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

- The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 4

PROJECT NAME (INFORMATION ABOUT THE PROJECT)

4.1 PROJECT REQUIREMENTS

4.1.1 Packages Used:

- pandas
- numpy
- seaborn
- matplotlib

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     %matplotlib inline
```

Fig 4.1.1: Importing packages

4.1.2 Versions of the Packages:

- pandas version 1.0.5
- numpy version 1.18.5
- seaborn version 0.10.1

```
print(pd.__version__)
print(np.__version__)
print(sns.__version__)

1.0.5
1.18.5
0.10.1
```

Fig 4.1.2 Versions

4.1.3 ALGORITHM USED

- Naive Bayes Classifier
- Logistic Regression
- K Neighbors Classifier
- SVC

4.2 PROBLEM STATEMENT

The problem we are going to solve here is the predictions on whether a person is suffering from Heart Disease or not. , taking into consideration the features of the dataset heart.csv

4.3 DATASET DESCRIPTION

The dataset consists of the following features:

There are 14 columns in the dataset, which are described below:

1. Age: displays the age of the individual.
2. Sex: displays the gender of the individual using the following format :

1 = male

0 = female

3. Chest-pain type: displays the type of chest-pain experienced by the individual using the following format :

    1 = typical angina

    2 = atypical angina

    3 = non — anginal pain

    4 = asymptotic

4. Resting Blood Pressure: displays the resting blood pressure value of an individual in mmHg (unit)
5. Serum Cholesterol: displays the serum cholesterol in mg/dl (unit)
6. Fasting Blood Sugar: compares the fasting blood sugar value of an individual with 120mg/dl.

    If fasting blood sugar > 120mg/dl then : 1 (true) else : 0 (false)

7. Resting ECG : displays resting electrocardiographic results

    0 = normal

    1 = having ST-T wave abnormality

    2 = left ventricular hypertrophy

8. Max heart rate achieved : displays the max heart rate achieved by an individual.
9. Exercise induced angina :

    1 = yes

    0 = no

10. ST depression induced by exercise relative to rest: displays the value which is an integer or float.
11. Peak exercise ST segment :

    1 = upsloping

    2 = flat

    3 = down sloping

12. Number of major vessels (0–3) colored by fluoroscopy : displays the value as integer or float.
13. Thal : displays the thalassemia :

    0 = normal

    1 = fixed defect

    2 = reversible defect

14. Diagnosis of heart disease : Displays whether the individual is suffering from heart disease or not :

    0 = absence

    1, 2, 3, 4 = present.

4.4 OBJECTIVE OF CASE STUDY

To get a better understanding of whether a predictions on whether a person is suffering from Heart Disease or not by considering the features of the data and provide the client with desired results

CHAPTER 5

DATA PREPROCESSING

5.1 READING THE DATASET

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a DataFrame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be accessed using the dataframe.

```
heart = pd.read_csv("/content/drive/My Drive/summer internship 1/heart.csv",encoding="unicode_escape")
heart
```

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

1025 rows × 14 columns

Fig 5.1.1: Reading the dataset

5.2 HANDLING MISSING VALUES AND DUPLICATE VALUES

We can find the number of missing values and duplicate values in each column using isnull() and duplicated() functions respectively. For our dataset no missing values or duplicated values were found.

```
heart.isnull()
```

	age	sex	chestpain	restingbloodpressure	Chlorestrol	Fasting Bloodsugar	Resting ECG	Max Heart Rate Achieved	Exercise induced angina	Old Peak	Slope	Major vessels	Thalassemia	target
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
723	False	False	False	False	False	False	False	False	False	False	False	False	False	False
733	False	False	False	False	False	False	False	False	False	False	False	False	False	False
739	False	False	False	False	False	False	False	False	False	False	False	False	False	False
843	False	False	False	False	False	False	False	False	False	False	False	False	False	False
878	False	False	False	False	False	False	False	False	False	False	False	False	False	False

302 rows × 14 columns

fig: 5.2.1 Checking missing values.

From above we can observe that we have used isnull() in order to check whether there are any missing values present in the given dataset **.**

```
heart.isnull().sum() # no null values in the dataset

age                        0
sex                        0
chestpain                  0
restingbloodpressure       0
Chlorestrol                0
Fasting Bloodsugar         0
Resting ECG                0
Max Heart Rate Achieved    0
Exercise induced angina    0
Old Peak                   0
Slope                      0
Major vessels              0
Thalassemia                0
target                     0
dtype: int64
```

Fig 5.2.2 : Total number of missing values in each column.

From the above output we can observe that the given dataset does not contain any missing Values.

```
[ ] print('duplicated entries: {}'.format(heart.duplicated().sum()))

duplicated entries: 723

[ ] heart.drop_duplicates(inplace = True)
    heart.shape

(302, 14)

[ ] print('duplicate entries: {}'.format(heart.duplicated().sum())) #no duplicates values in the dataset

duplicate entries: 0
```

Fig 5.2.3 duplicate values are null

```
sns.heatmap(heart.isnull())

<matplotlib.axes._subplots.AxesSubplot at 0x7f8d910d0908>
```

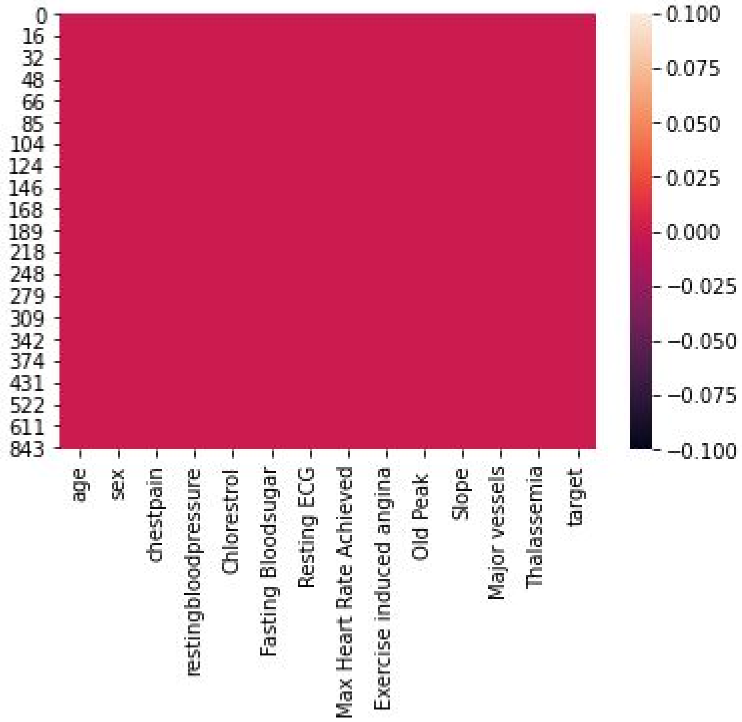


Fig 5.2.4 visualization through heatmap

5.3 CATEGORICAL DATA:

```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

Fig 5.3.1 : Description about the type of each feature in the dataset.(Categorical or Numerical).

From the above figure it looks like it returns a series with the datatype of each column.

5.4 Statistical Analysis

```
#calculatng the mean
heart.mean()
```

```
age                         54.420530
sex                          0.682119
chestpain                    0.963576
restingbloodpressure       131.602649
Chlorestrol                246.500000
Fasting Bloodsugar           0.149007
Resting ECG                  0.526490
Max Heart Rate Achieved    149.569536
Exercise induced angina      0.327815
Old Peak                     1.043046
Slope                        1.397351
Major vessels                0.718543
Thalassemia                  2.314570
target                       0.543046
dtype: float64
```

Fig 5.4.1: Mean of the given data set

From the above figure it shows the mean of the given dataset,, we usually do this by dividing the sum of given numbers with the count of the number present.

```
#calculatng the median
heart.median()
```

```
age                         55.5
sex                          1.0
chestpain                    1.0
restingbloodpressure       130.0
Chlorestrol                240.5
Fasting Bloodsugar           0.0
Resting ECG                  1.0
Max Heart Rate Achieved    152.5
Exercise induced angina      0.0
Old Peak                     0.8
Slope                        1.0
Major vessels                0.0
Thalassemia                  2.0
target                       1.0
dtype: float64
```

Fig 5.4.2 median of the given dataset

5.5 Generating Plots

5.5.1 Visualization of the confirmed cases of one country

```
#visualizing the heart using heatmap
sns.heatmap(heart)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d94caea58>
```

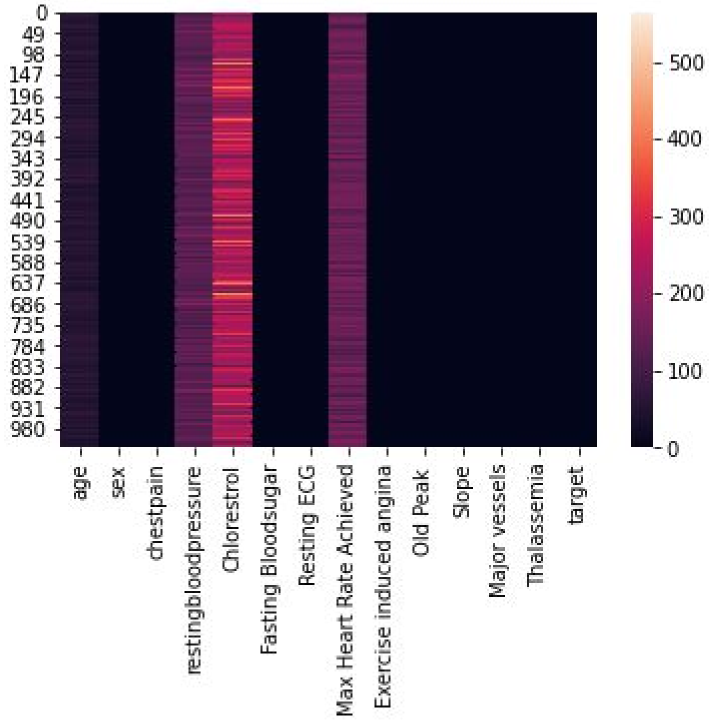


Fig 5.5.1: visualizing the data using heatmap

The above figure shows the data visualization by using a heat map which displays numeric tabular data where the cells are colored depending upon the contained value.

```
[ ] sns.countplot(x="target", data=heart, palette="bwr")
    plt.show()
```

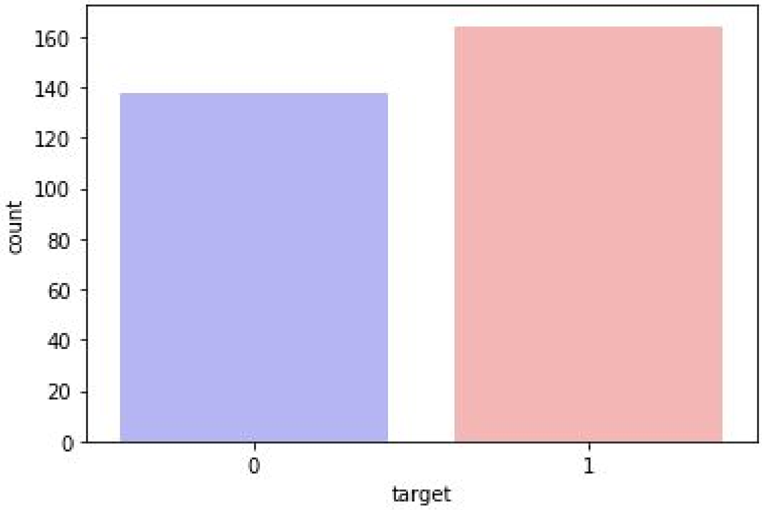


Fig 5.5.2: From the plot, we can see that the classes are almost balanced and we are good to proceed with data processing.

```
pd.crosstab(heart.age,heart.target).plot(kind="bar",figsize=(20,6))
plt.title('Heart Disease Frequency for Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('heartDiseaseAndAges.png')
plt.show()
```

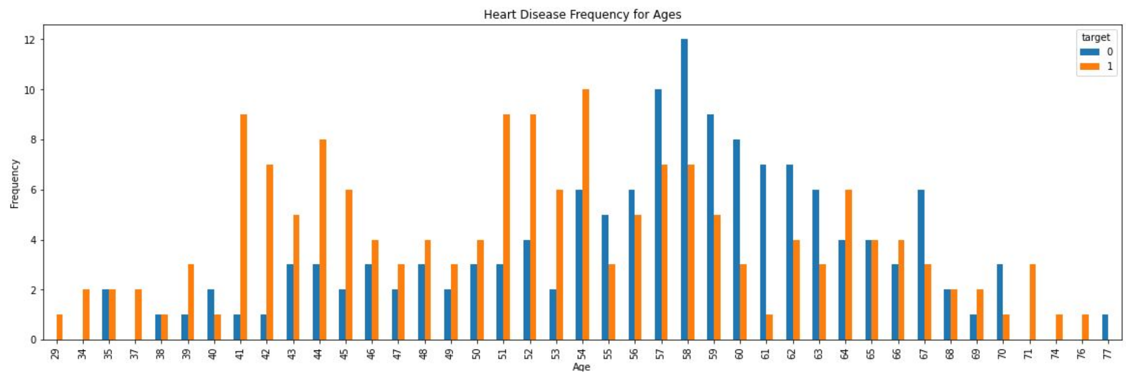


Fig 5.5.3 We see that most people who are suffering are of the age of 58, followed by 57. Majorly, people belonging to the age group 50+ are suffering from the disease.

```
pd.crosstab(heart.sex,heart.target).plot(kind="bar",figsize=(15,6),color=['#1CA53B','#AA1111' ])
plt.title('Heart Disease Frequency for Sex')
plt.xlabel('Sex (0 = Female, 1 = Male)')
plt.xticks(rotation=0)
plt.legend(["Haven't Disease", "Have Disease"])
plt.ylabel('Frequency')
plt.show()
```

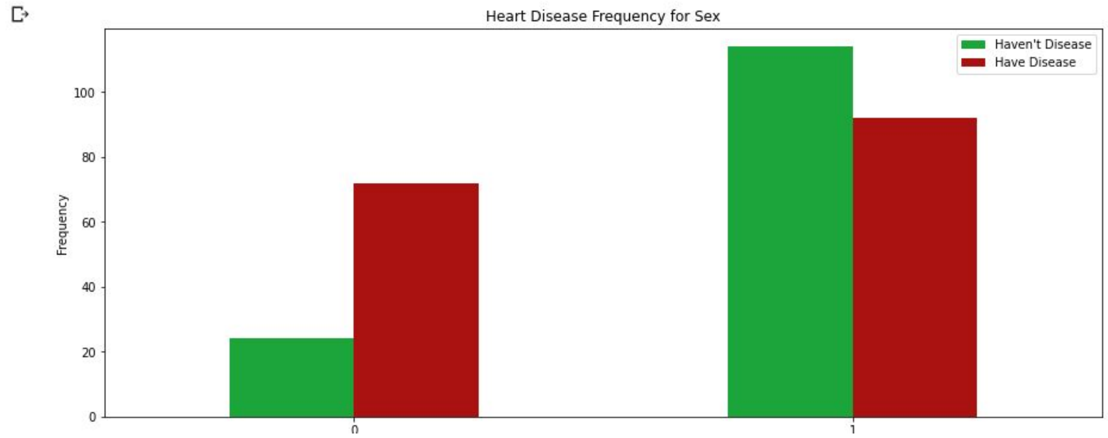
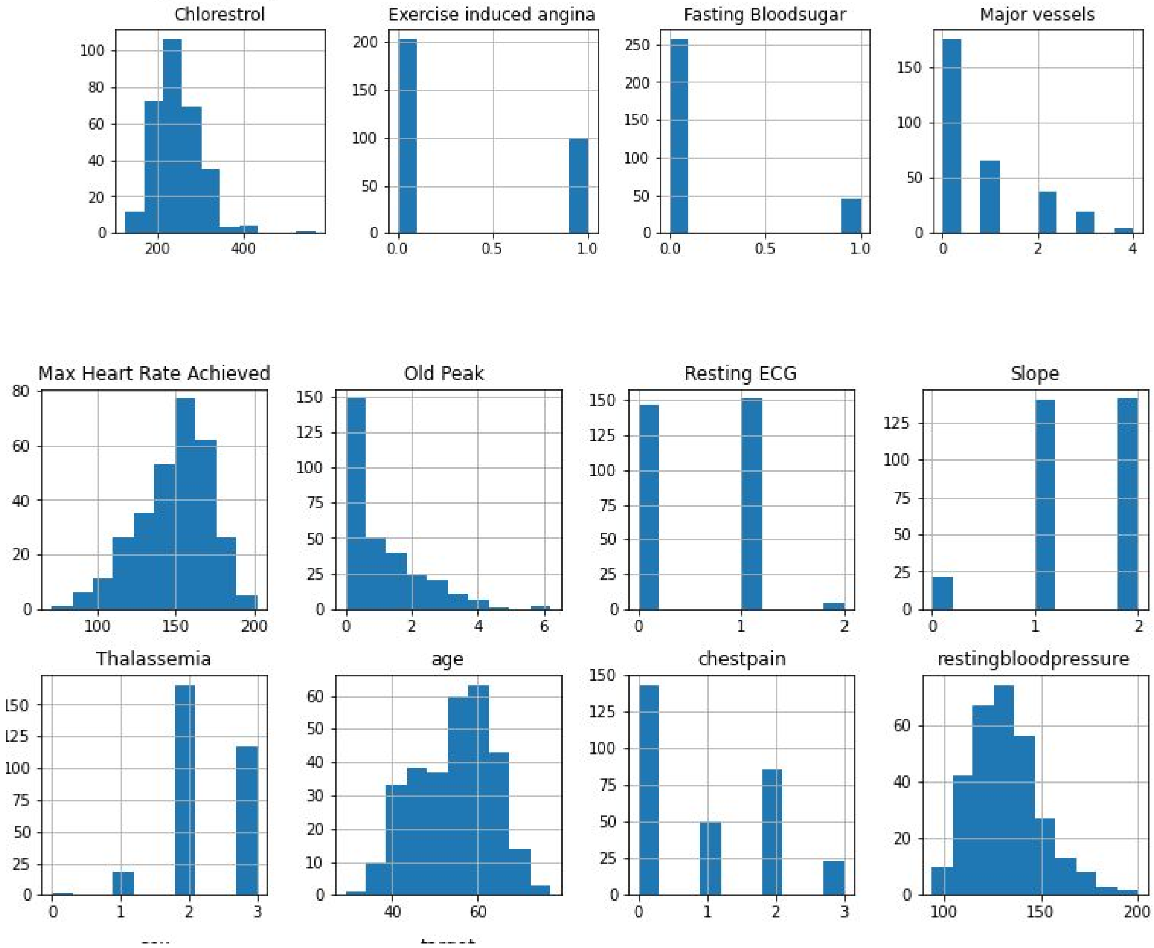


Fig 5.5.4 Here 1 means male and 0 denotes female. We observe females having heart disease are comparatively less when compared to males. Males have low heart diseases as compared to females in the given dataset.

5.6 HISTOGRAM

```
heart.hist(figsize=(13,13))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8d909379b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d907bc978>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d907eccc0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d907a1f98>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8d9075c2b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d90711630>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d906c19b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d906f3cf8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8d906f3d68>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d90666470>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d906177f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d93a0d198>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8d910c4cf8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d9083a518>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d908d64a8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8d9098b048>]],
      dtype=object)
```



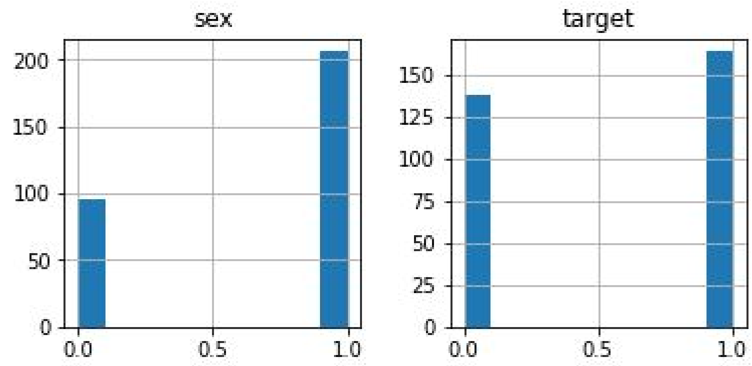


Fig 5.6.1 : histogram constructed between all the columns present

The above figure conveys the histogram plotted between the data which is present.

CHAPTER 6

FEATURE SELECTION

6.1 Select relevant features for the analysis

```
[ ] #checked whether there are balanced number of o's and 1's
    heart['target'].value_counts()

    1    526
    0    499
    Name: target, dtype: int64
```

Fig 6.1.1 : imbalanced data

```
[ ] from sklearn.utils import resample

    # Separate majority and minority classes
    heart_majority = heart[heart.target==0]
    heart_minority = heart[heart.target==1]

    # Downsample majority class
    heart_majority_downsampled = resample(heart_majority,
                                      replace=True,    # sample without replacement
                                      n_samples=526,     # to match minority class
                                      random_state=123) # reproducible results

    # Combine minority class with downsampled majority class
    heart_downsampled = pd.concat([heart_majority_downsampled, heart_minority])

    # Display new class counts
    heart_downsampled.target.value_counts()

    1    526
    0    526
    Name: target, dtype: int64
```

Fig: 6.1.2 balancing the dataset

6.2 TRAIN-TEST-SPLIT

Splitting the data : after the preprocessing is done then the data is split into train and test sets.

● In Machine Learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'test set'. An important point to note is that during training the classifier only uses the training set . The test set must not be used during training the classifier. The test set will only be available during testing the classifier.

● training set - a subset to train a model.(Model learns patterns between Input and Output)

● test set - a subset to test the trained model.(To test whether the model has correctly learnt)

● The amount or percentage of Splitting can be taken as specified (i.e. train data = 75% , test data =25% or train data = 80% , test data= 20%)

● First we need to identify the input and output variables and we need to separate the input set and output set.

● In scikit learn library we have a package called model selection in which train_test_split method is available .we need to import this method.

● This method splits the input and output data to train and test based on the percentage specified by the user and assigns them to four different variables(we need to mention the variables) .

```
[ ]  #Splitting the dataset into training and test data.
     # 80% of the data will be in training data and 20% of the data will be in testing
     X = heart.drop(['target'],axis=1)
     y = heart.target
     from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.2,random_state=0)
```

Fig 6.2.1 importing train_test_split and splitting the data

6.3 FEATURE SCALING

Feature Scaling--> when applied, this units and scaling will be removed To make the data unitless and scale less, we have to apply Feature Scaling

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
StandardScaler = StandardScaler()
columns_to_scale = ['age','restingbloodpressure','Chlorestrol','Max Heart Rate Achieved','Old Peak ']
heart[columns_to_scale] = StandardScaler.fit_transform(heart[columns_to_scale])
heart.head()
```

	age	sex	chestpain	restingbloodpressure	Chlorestrol	Fasting Bloodsugar	Resting ECG	Max Heart Rate Achieved	Exercise induced angina	Old Peak	Slope	Major vessels	Thalassemia	target
0	-0.267966	1	0	-0.376556	-0.667728	0	1	0.806035	0	-0.037124	2	2	3	0
1	-0.157260	1	0	0.478910	-0.841918	1	0	0.237495	1	1.773958	0	0	3	0
2	1.724733	1	0	0.764066	-1.403197	0	1	-1.074521	1	1.342748	0	0	3	0
3	0.728383	1	0	0.935159	-0.841918	0	1	0.499898	0	-0.899544	2	1	3	0
4	0.839089	0	0	0.364848	0.919336	1	1	-1.905464	0	0.739054	1	3	2	0

Fig 6.3.1 : feature scaling data

I have used Standard Scalar from sklearn to scale my dataset

CHAPTER 7

MODEL BUILDING AND EVALUATION

Approach I : Naive Bayes Classifier

7.1 Brief about the algorithms used

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.

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.

$$P(h\backslash D)=P(D\backslash h)P(h)/P(D)$$

- 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.

7.1.1 Train the model

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)

    from sklearn.naive_bayes import GaussianNB
    classifier = GaussianNB()
    classifier.fit(X_train,y_train)

    GaussianNB(priors=None, var_smoothing=1e-09)
```

Fig 7.1.1 : Applying Naive Bayes algorithm

7.1.2 Predicting on Train Data

We can predict the training data using predict function on X_train

```
[ ]  y_train_pred = classifier.predict(X_train)
     y_train==y_train_pred
```

```
210     True
124     True
145     True
380     True
143     True
        ...
292    False
431     True
78      True
366     True
39     False
Name: target, Length: 241, dtype: bool
```

Fig 7.1.2. predicting on train data

7.1.2.1 Visualizing the confusion matrix using heat map

```
from sklearn.metrics import confusion_matrix, accuracy_score
conf = confusion_matrix(y_train, y_train_pred)
conf
sns.heatmap(confusion_matrix(y_train, y_train_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d8d52f208>
```

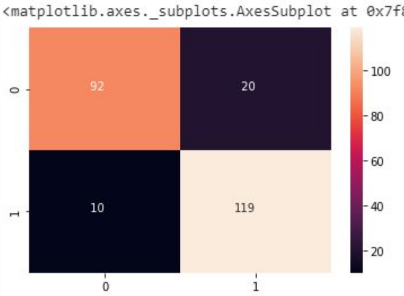


Fig 7.1.2.1 showing confusion matrix

7.1.2.2 Print the classification report and check the accuracy of the training data

```
print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.82      0.86       112
           1       0.86      0.92      0.89       129

    accuracy                           0.88       241
   macro avg       0.88      0.87      0.87       241
weighted avg       0.88      0.88      0.87       241
```

Fig 7.1.2.2 classification report for training data

7.1.2.3 Finding the accuracy score:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_train,y_train_pred)
```

```
0.8755186721991701
```

Fig 7.1.2.3 accuracy score

We got an accuracy score of about 86% to the training data which is considered to be a good score.

Now let's check the same for test data

7.1.3 Predicting on Test Data

We can predict the testing data using predict function on X_test

```
y_test_pred = classifier.predict(X_test)
y_test==y_test_pred
```

```
233     True
101     True
215     True
377    False
131     True
       ...
274    False
89      True
391    False
331     True
363    False
Name: target, Length: 61, dtype: bool
```

Fig 7.1.3. predicting on test data

7.1.3.1 Visualizing the confusion matrix using heat map

```
from sklearn.metrics import confusion_matrix, accuracy_score
conf = confusion_matrix(y_test, y_test_pred)
conf
sns.heatmap(confusion_matrix(y_test, y_test_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d8d47cc88>
```

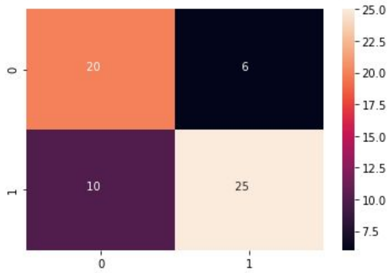


Fig 7.1.3.1 showing confusion matrix

7.1.3.2 Print the classification report to know the accuracy score of testing data

```
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.67      0.77      0.71        26
           1       0.81      0.71      0.76        35

    accuracy                           0.74        61
   macro avg       0.74      0.74      0.74        61
weighted avg       0.75      0.74      0.74        61
```

Fig 7.1.3.2 classification report for testing data

7.1.3.3 Finding the accuracy score:

```
[52] from sklearn.metrics import accuracy_score
     accuracy_score(y_test,y_test_pred)
```

```
0.7377049180327869
```

Fig 7.1.3.3 accuracy score

We got an accuracy score of around 74%, therefore we can say that it is a best fit

Approach II : Logistic Regression

7.2 Brief about the algorithms used

In statistics, the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

7.2.1 Train the model

Import Logistic Regression from sklearn.linear_model and create an object. Fit the model on training data

```
[ ]  from sklearn.linear_model import LogisticRegression
     log_reg = LogisticRegression() # creating an object for Logistic Regression
     ## We have to apply this object(log_reg) to the training data
     log_reg.fit(X_train, y_train) # with help of fit method we are fitting the
                                   ##Logistic Regression on training data
     ## objectName.fit(InputData, OutputData)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Fig 7.2.1. importing logistic regression

7.2.2 Predicting on Train data

We can predict the training data using predict function on X_train

```
[ ]  y_train_pred = log_reg.predict(X_train)
```

```
[ ]  y_train ==y_train_pred
```

```
81      True
193     True
70      True
719    False
628     True
       ...
425     True
271     True
143     True
50      True
232     True
Name: target, Length: 241, dtype: bool
```

Fig 7.2.2. predicting on train data

7.2.2.1 Visualizing the confusion matrix using heat map

```
[ ]  from sklearn.metrics import confusion_matrix, accuracy_score
     conf = confusion_matrix(y_train, y_train_pred)
     conf
     sns.heatmap(confusion_matrix(y_train, y_train_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2e210b6f28>
```

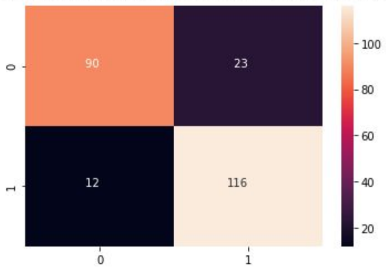


Fig 7.2.2.1 visualizing confusion matrix of train data

7.2.2.2 Find the accuracy score

```
from sklearn.metrics import accuracy_score
accuracy_score(y_train, y_train_pred)
```

```
0.8547717842323651
```

Fig 7.2.2.2 .accuracy score of train data

7.2.2.3 Print the classification report and check the accuracy of the training data

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.80      0.84       113
           1       0.83      0.91      0.87       128

    accuracy                           0.85       241
   macro avg       0.86      0.85      0.85       241
weighted avg       0.86      0.85      0.85       241
```

Fig 7.2.2.3 classification report of train data

We got an accuracy score of around 86% for the training data which is considered to be a good score.

Now let's check the same for test data.

7.2.3 Predicting on test data

We can predict the testing data using predict function on X_test

```
[ ] y_test_pred = log_reg.predict(X_test)
```

```
y_test==y_test_pred
```

```
342     True
191     True
349    False
288     True
56      True
        ...
182     True
878     True
27     False
128     True
102     True
Name: target, Length: 61, dtype: bool
```

Fig 7.2.3 predicting on test data

7.2.3.1 Visualizing the confusion matrix using heat map

```
[ ] from sklearn.metrics import confusion_matrix, accuracy_score
    conf = confusion_matrix(y_test, y_test_pred)
    conf
    sns.heatmap(confusion_matrix(y_test, y_test_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2e210b6748>
```

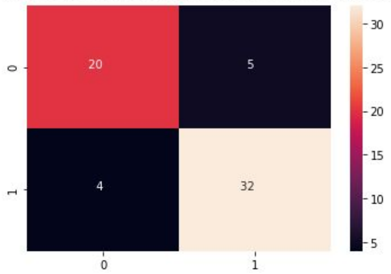


Fig 7.2.3.1 visualizing confusion matrix of test data

7.2.3.2 Find the accuracy score

```
[ ] accuracy_score(y_test, y_test_pred)
```

```
0.8524590163934426
```

Fig 7.2.3.2 accuracy score of test data

7.2.3.3 Print the classification report and check the accuracy of the testing data

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.80      0.82        25
           1       0.86      0.89      0.88        36

    accuracy                           0.85        61
   macro avg       0.85      0.84      0.85        61
weighted avg       0.85      0.85      0.85        61
```

Fig 7.2.3.3 classification report of testing data

We got an accuracy score of around 85%, therefore we can say that it is a best fit and the model predicted very well.

Approach 3:

7.3 KNN CLASSIFIER:

A supervised machine learning algorithm is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.
The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. However, it is more widely used in classification problems in the industry. It is used for classification and regression of known data where usually the target variable is known beforehand.
K nearest neighbors is an algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). It should also be noted that all three distance measures are only valid for continuous variables.
The 'k' stands for the number of nearest neighbors for the newly entered value.
The kNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. kNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph
This works based on minimum distance from the query instance to the training samples to determine the k-nearest neighbors. After we gather these k-nearest neighbors, we take the simple majority of these k nearest neighbors to be the prediction of the query instance. Methods of calculating distance between points:
The first step is to c
calculate the distance between the new point and each training point. There are various methods for calculating this distance, of which the most commonly known methods are – Euclidean, Manhattan and Hamming distance.

1. Euclidean Distance: Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (y).
2. Manhattan Distance/City Block Distance: This is the distance between real vectors using the sum of their absolute difference.

7.3.1 Train the model

Import KNN and create an object for that class

```
[53] from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.2, random_state = 2)
     from sklearn.neighbors import KNeighborsClassifier
     Knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')

     # Apply the knn object on the dataset(Training Phase)
     # Syntax: objectName.fit(Input, Output)
     Knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

Fig 7.3.1. importing KNN

7.3.2 Predicting on Train data

```
# Predictions on the data
#predict function--> gives the predicted values
# Syntax:objectname.predict(Input)
y_train_pred = Knn.predict(X_train)
y_train==y_train_pred
```

```
367    True
125    True
10     True
193    True
88     True
       ...
85     True
23     True
78     True
16     True
225    True
Name: target, Length: 241, dtype: bool
```

Fig 7.3.2 predicting on training data

7.3.2.1 Visualizing the confusion matrix using heat map

```
from sklearn.metrics import confusion_matrix, accuracy_score
conf = confusion_matrix(y_train, y_train_pred)
conf
sns.heatmap(confusion_matrix(y_train, y_train_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2e204c7240>
```

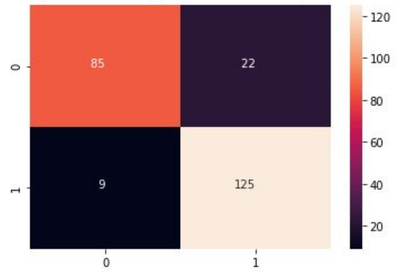


Fig 7.3.2.1 visualizing confusion matrix of train data

7.3.2.2 Print the classification report and check the accuracy of the data

```
[56] # Check the accuracy, classification report
     from sklearn.metrics import classification_report
     print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.79      0.85       107
           1       0.85      0.93      0.89       134

    accuracy                           0.87       241
   macro avg       0.88      0.86      0.87       241
weighted avg       0.87      0.87      0.87       241
```

Fig 7.3.2.2 classification report of data

7.3.2.3 Find the accuracy score

```
from sklearn.metrics import accuracy_score
accuracy_score(y_train,y_train_pred)
```

```
0.8713692946058091
```

Fig 7.3.2.3 accuracy score of train data

We got an accuracy score of around 88% for the training data which is considered to be a good score.

Now let's check the same for test data.

7.3.3 Predicting on test data

We can predict the testing data using predict function on X_test

```
# Prediction on  training data
y_test_pred = Knn.predict(X_test)
y_test==y_test_pred
```

```
114    False
719    False
102     True
32      True
720    False
       ...
233     True
108    False
210     True
323     True
105     True
Name: target, Length: 61, dtype: bool
```

Fig 7.3.3 predicting on test data

7.3.3.1 Visualizing the confusion matrix using heat map

```
from sklearn.metrics import confusion_matrix, accuracy_score
conf = confusion_matrix(y_test, y_test_pred)
conf
sns.heatmap(confusion_matrix(y_test, y_test_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseline'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2e1ff9e470>
```

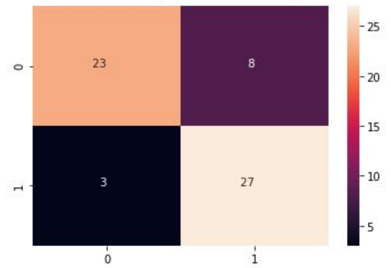


Fig 7.3.3.1 visualizing confusion matrix of test data

7.3.3.2 Print the classification report and check the accuracy of the testing data

```
[61] # Classification Report for Test Data
     print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.74      0.81        31
           1       0.77      0.90      0.83        30

    accuracy                           0.82        61
   macro avg       0.83      0.82      0.82        61
weighted avg       0.83      0.82      0.82        61
```

Fig 7.3.3.2 classification report of testing data

7.3.3.3 Find the accuracy score

```
[60] from sklearn.metrics import accuracy_score
     accuracy_score(y_test,y_test_pred)
```

```
0.819672131147541
```

Fig 7.3.3.3 accuracy score of test data

We got an accuracy score of around 83%

CHAPTER 8

Tuning the hyper-parameters

8.1 GridSearchCV

Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. Typical examples include C, kernel and gamma for Support Vector Classifier, alpha for Lasso, etc.

```
from sklearn import svm
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto'),
        'params' : {
            'C': [1,10,20],
            'kernel': ['rbf','linear']
        }
    },
    'knn': {
        'model': KNeighborsClassifier(),
        'params' : {}

    },
    'naive_bayes_gaussian': {
        'model': GaussianNB(),
        'params': {}
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear',multi_class='auto'),
        'params': {
            'C': [1,5,10]
```

Fig 8.1.1 : importing data

8.2 Checking best_estimator_.score

```
scores = []

for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=5, return_train_score=False)
    clf.fit(X,y)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
```

```
heart = pd.DataFrame(scores,columns=['model','best_score','best_params'])
heart
```

	model	best_score	best_params
0	svm	0.834536	{'C': 1, 'kernel': 'rbf'}
1	knn	0.804645	{}
2	naive_bayes_gaussian	0.807760	{}
3	logistic_regression	0.841038	{'C': 1}

Fig 8.2.1 Best accuracy score

CHAPTER 9

CONCLUSION

In this project, I used Machine Learning to predict whether a person is suffering from a heart disease. After importing the data, I analysed it using plots. I then applied 4 Machine Learning algorithms, K Neighbors Classifier, logistic regression ,naive bayes classifier,svm. I varied parameters across each model to improve their scores. In the end, Logistic Regression achieved the highest score of accuracy 85% as compared to other algorithms

```
models=['Logistic Regression','NaiveBayes','kNeighboursClassifier','svm']
accuracy_scores=[0.85,0.80,0.80,0.83]
plt.bar(models,accuracy_scores,color=['orange','grey','pink','black'])
plt.ylabel("accuracy scores")
plt.title("which model has high accuracy")
plt.show()
```

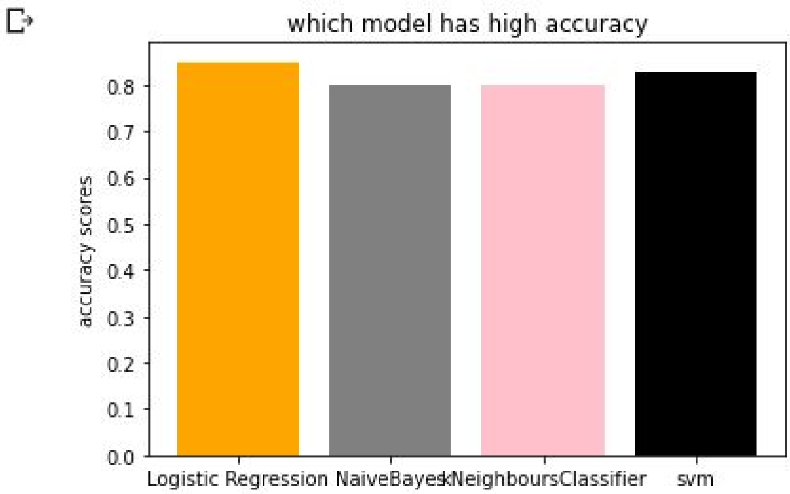


Fig 9.1 : showing high accuracy model

the accuracy for the test set achieved by logistic regression is: 0.85

the accuracy for the test set achieved by naive bayes:0.80

the accuracy for the test set achieved by knn is:0.80

the accuracy for the test set achieved by svm is:0.83

CHAPTER 10

REFERENCES

1.https://medium.com/code-heroku/introduction-to-exploratory-data-analysis-eda-c0257f888676

2.https://en.wikipedia.org/wiki/Machine_learning

3.https://www.edureka.co/blog/what-is-data-science/

4.https://www.edureka.co/blog/data-science-applications/

5.https://en.wikipedia.org/wiki/Naive_Bayes_classifier

6.https://en.wikipedia.org/wiki/Logistic_regression

7.https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm