MACHINE LEARNING

(PREDICTION OF HEART OCCURENCE)

Summer Internship Report Submitted in partial fulfillment of the requirement for undergraduate degree of

Bachelor of Technology

In

COMPUTER SCIENCE ENGINEERING

By

B. ALEKYA

221710304011

Under the Guidance of

Mrs..Manisha Patil

Assistance Professor

Department Of ComputerScience and Engineering

GITAM School of Technology

GITAM (Deemed to be University), Hyderabad-502329

DECLARATION

I submit this industrial training work entitled “PREDICTION OF HEART DISESASE OCCURENCE” to GITAM (Deemed To Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of “Bachelor of Technology” in “Computer Science Engineering”. I declare that it was carried out independently by me under the guidance of Mrs.Manisha 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 **B.ALEKYA**

Date:13-07-2020 **221710304011**

CERTIFICATE

This is to certify that the Industrial Training Report entitled "PREDICTION OF HEART DISEASE" is being submitted by B. Alekya(221710304011) in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science Engineering at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-2020

It is faithful record work carried out by her at the Computer Science Engineering Department, GITAM University Hyderabad Campus

under my guidance and supervision.

Mrs.Mansiha Patil	**Dr.K.Manjunathachari**
Assistant Professor	Professor and HOD
Department of CSE	Department of ECE

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 **Prof. N. Siva Prasad**, Pro Vice Chancellor,GITAM Hyderabad and **Prof. N. Seetha Ramaiah,** Principal, GITAM Hyderabad.

I would like to thank respected **Prof. S. Phani Kumari,** 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 a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mrs.Manisha** 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.

B. ALEKYA

221710304011

ABSTRACT

Heart disease is a common problem which can be very severe in old ages and also in people not having a healthy lifestyle. With regular check-up and diagnosis in addition to maintaining a decent eating habit can prevent it to some extent. In this paper we have tried to implement the most sought after and important machine learning algorithm to predict the heart disease in a patient. The decision tree classifier is implemented based on the symptoms which are specifically the attributes required for the purpose of prediction. Using the decision tree algorithm, we will be able to identify those attributes which are the best one that will lead us to a better prediction of the datasets. The decision tree algorithm works in a way where it tries to solve the problem by the help of tree representation. Here each internal node of the tree represents an attribute, and each leaf node corresponds to a class label. The support vector machine algorithm helps us to classify the datasets on the basis of kernel and it also groups the dataset using hyperplane. The main objective of this project is to try and reduce the number of occurrences of the heart diseases in patients.

Heart is the next major organ comparing to brain which has more priority in Human body. It pumps the blood and supplies to all organs of the whole body. Prediction of occurrences of heart diseases in medical field is significant work. Data analytics is useful for prediction from more information and it helps medical centre to predict of various disease. Huge amount of patient related data is maintained on monthly basis. The stored data can be useful for source of predicting the occurrence of future disease. Some of the data mining and machine learning techniques are used to predict the heart disease, such as logistic regreesion , K-Nearest Neighbour(KNN), Naïve Bayes. This paper provides an insight of the existing algorithm and it gives an overall summary of the existing

B.Alekya

221710304011

CHAPTER 1

DATA SCIENCE

1.1 INTRODUCTION:

DATA SCIENCE:

DATA SCIENCE is the area of study which involves extracting insights from vast amounts of data by the use of various scientific methods, algorithms, and processes. It helps you to discover hidden patterns from the raw data. The term Data Science has emerged because of the evolution of mathematical statistics, data analysis, and big data.

Data Science is an interdisciplinary field that allows you to extract knowledge from structured or unstructured data. Data science enables you to translate a business problem into a research project and then translate it back into a practical solution.

MACHINE LEARNING:

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).

1.2 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

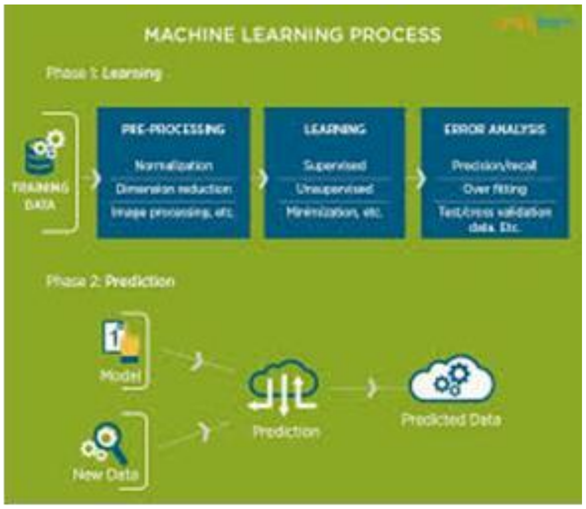


Figure 1 : The Process Flow

1.3 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 being 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.

1.4 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.

1.4.1 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.

1.4.2 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.

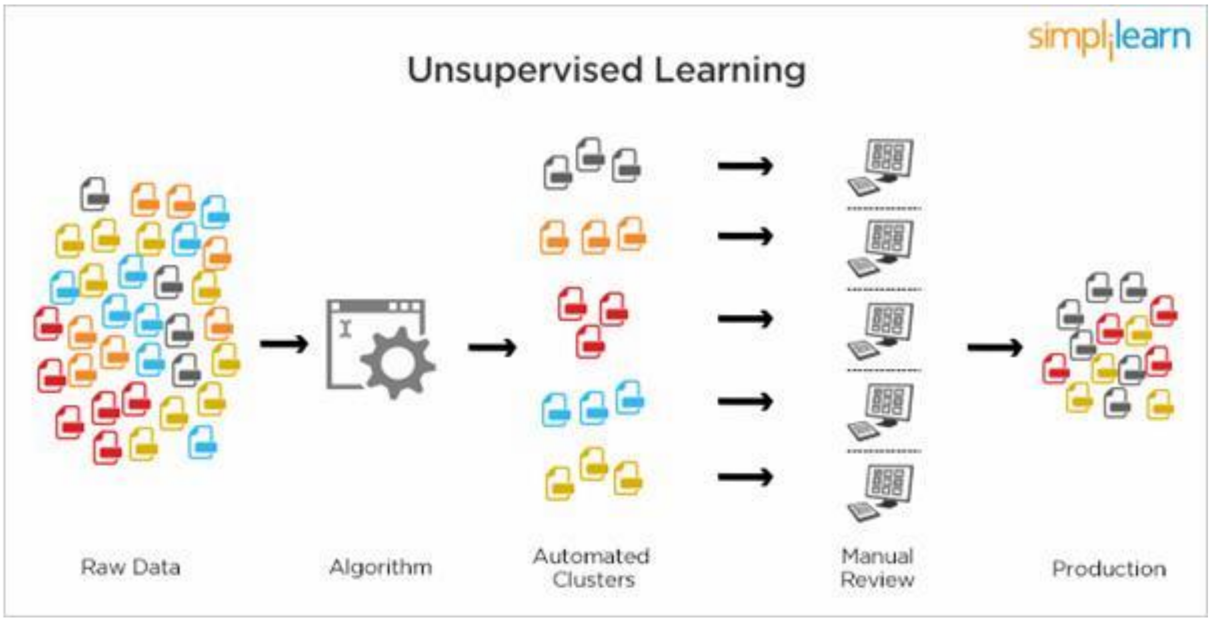


Figure 2 : Unsupervised Learning

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

1.4.3 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.

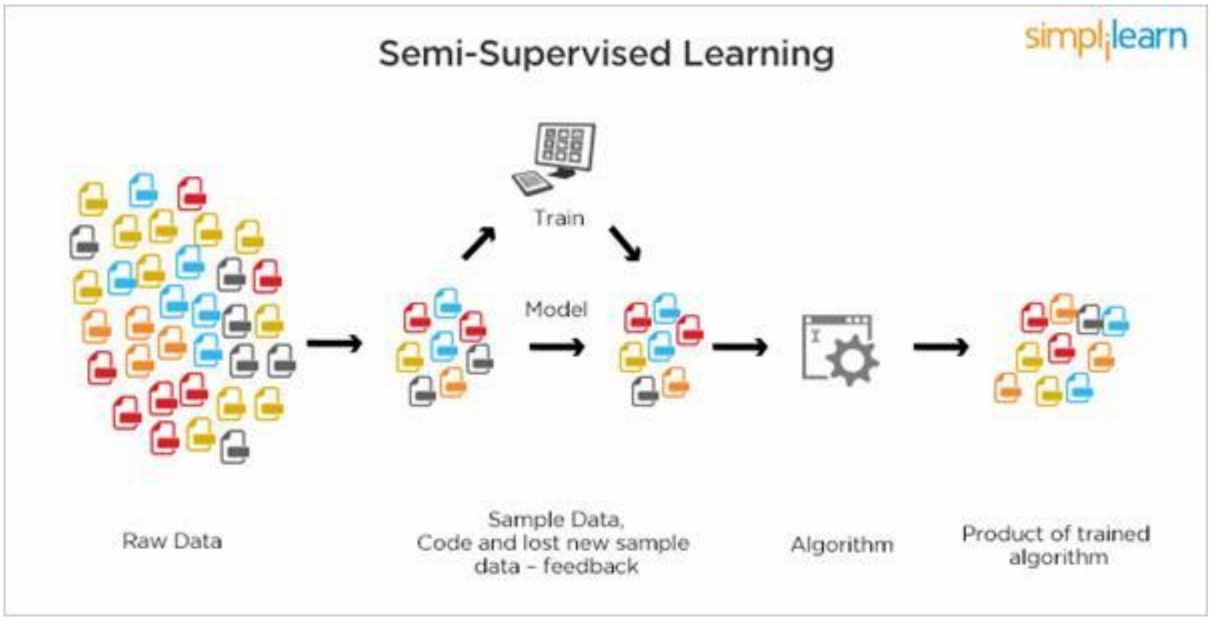

Semi-Supervised Learning
simplilearn
Train
Model
Raw Data
Sample Data, Code and lost new sample data - feedback
Algorithm
Product of trained algorithm

Figure 3 : Semi Supervised Learning

1.5 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 discovers 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 2

PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 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.

2.2 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 python3

2.3 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.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's

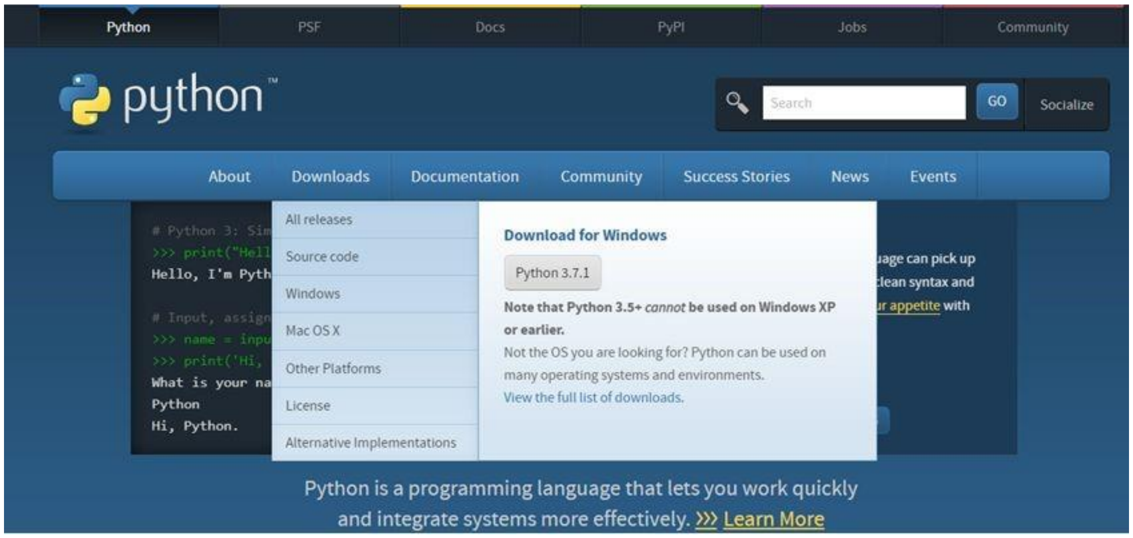
understand how to set up our Python environment.

- The most up-to-date and current source code, binaries, documentation, news, etc., is

available on the official website of Python.

2.4.1 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.



```
# Python 3: Sim
>>> print("Hell
Hello, I'm Pyth

# Input, assign
>>> name = inpu
>>> print('Hi,
What is your na
Python
Hi, Python.
```

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

uage can pick up
clean syntax and
ur appetite with

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Figure 4 : Python download

2.4.2 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 quickly installs and manages packages.
- In WINDOWS:
- In windows
- Step 1: Open Anaconda.com/downloads in web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/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)

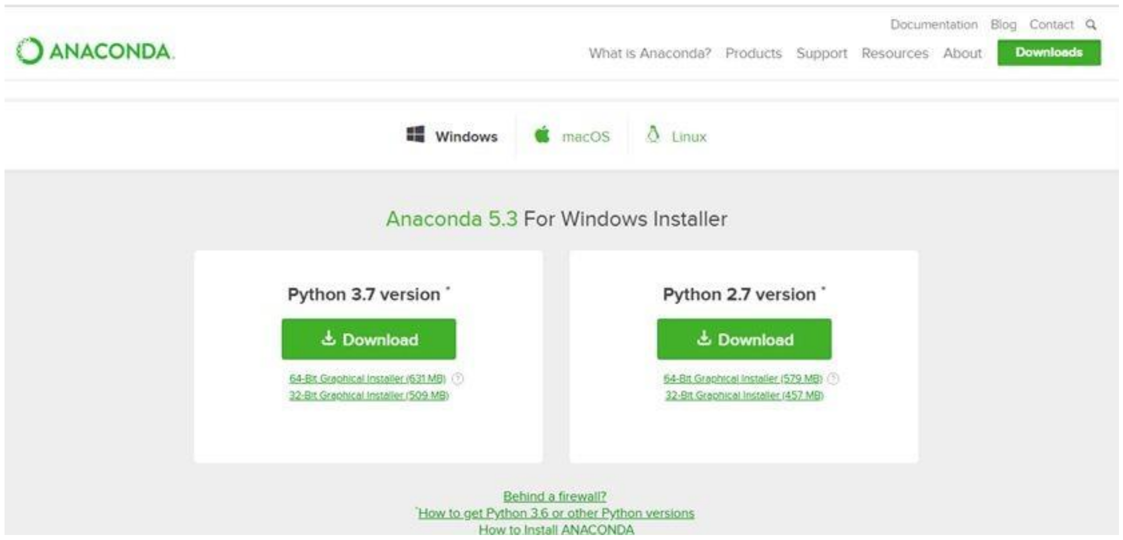

ANACONDA.
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows
macOS
Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
How to get Python 3.6 or other Python versions
How to Install ANACONDA

Figure 5 : Anaconda download

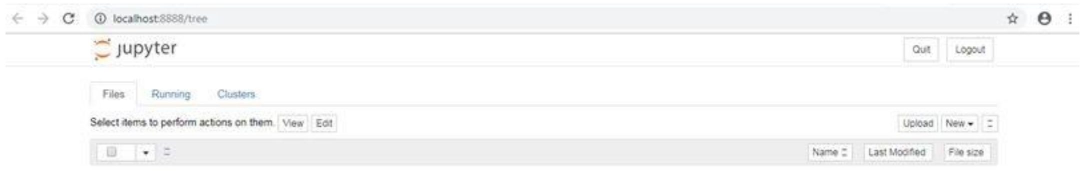


Figure 6 : Jupyter notebook

2.5 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 standard data types –
    - **Strings:**
- **Tuples**
- **Numbers:**
- **Lists:**
    - **Dictionary:**

2.5.1 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).

2.5.2 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.

2.5.3 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 type.
- 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.

2.5.4 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 add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 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.

2.6 PYTHON FUNCTION:

2.6.1 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.

2.6.2 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.

2.7 PYTHON USING OOP's CONCEPTS:

2.7.1 Class:

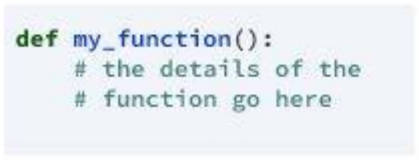
- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

- 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.

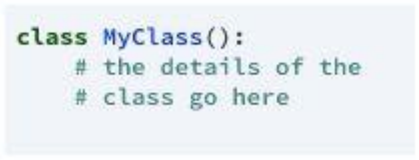
- Data member: A class variable or instance variable that holds data associated with a class and its objects.

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

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



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



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

Figure 7 : Defining a Class

2.7.2 __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 3

CASE STUDY

3.1 PROBLEM STATEMENT:

The problem that we are going to solve here is that given a set of features that describe whether a person is suffering from heart disease or not, our machine learning model must predicts whether the person is suffering from heart disease or not.

3.2 DATA SET:

The given data set consists of the following parameters:

age

sex

chest pain type (4 values)

resting blood pressure

serum cholestoral in mg/dl

fasting blood sugar > 120 mg/dl

resting electrocardiographic results (values 0,1,2)

maximum heart rate achieved

exercise induced angina

oldpeak = ST depression induced by exercise relative to rest the slope of the peak exercise ST segment

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

thal: 0 = normal; 1 = fixed defect; 2 = reversable defect The names and social security numbers of the patients were recently removed from the database, replaced with dummy value

3.3 OBJECTIVE OF THE CASE STUDY:

To get a better understanding and chalking out a plan of action for solution of the client, we have adapted the view point of looking at product categories and for further deep understanding of the problem, we have also considered gender age of the customer and reasoned out the various factors of choice of the products and they purchase , and our primary objective of this case study was to look up the factors which were dampening the sale of products and corelate them to product categories and draft out an outcome report to client regarding the various accepts of a product purchases.

CHAPTER 4

MODEL BUILDING

4.1 PREPROCESSING OF THE DATA:

Preprocessing of the data actually involves the following steps:

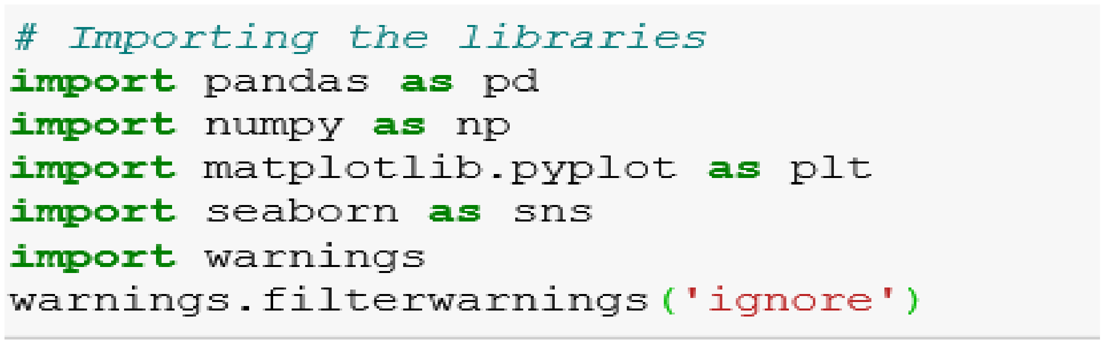
4.1.1 GETTING THE DATASET:

We can get the data set from the database or we

can get the data from client.

4.1.2 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the algorithm.



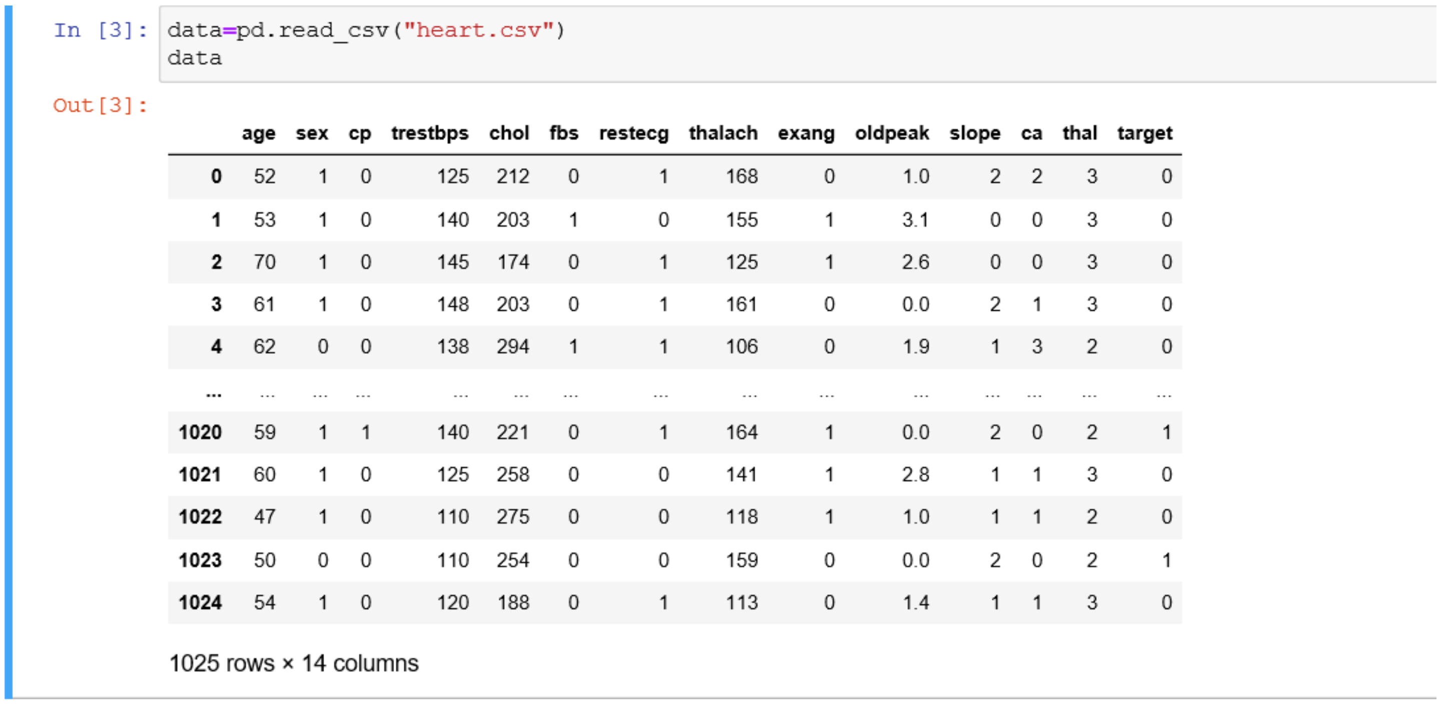
```
# Importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Figure 8 : Importing Libraries

4.1.3 IMPORTING THE DATA-SET:

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 access using the dataframe. Any missing value or NaN value have to be cleaned.



```
In [3]: data=pd.read_csv("heart.csv")
        data
Out[3]:
```

	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

Figure 9 : Reading the dataset

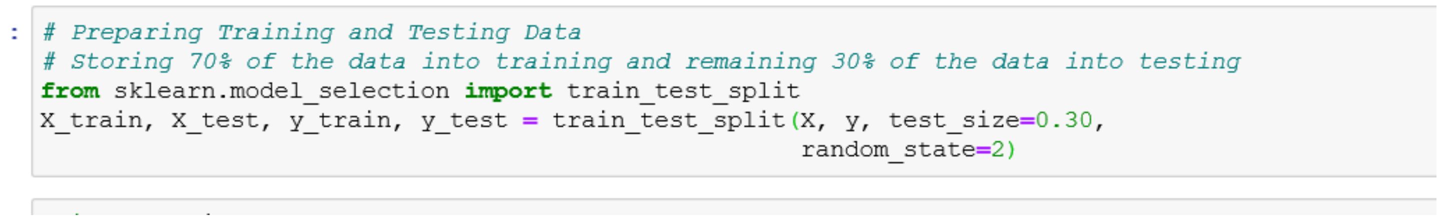
4.2 TRAINING THE MODEL:

Method :

- 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)

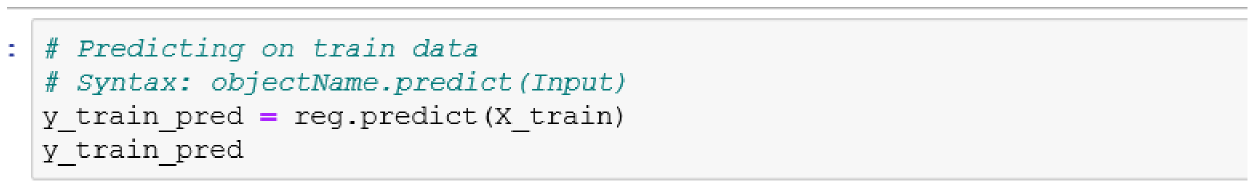


```
# Preparing Training and Testing Data
# Storing 70% of the data into training and remaining 30% of the data into testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
                                                    random_state=2)
```

Figure 10 : importing train_test_split

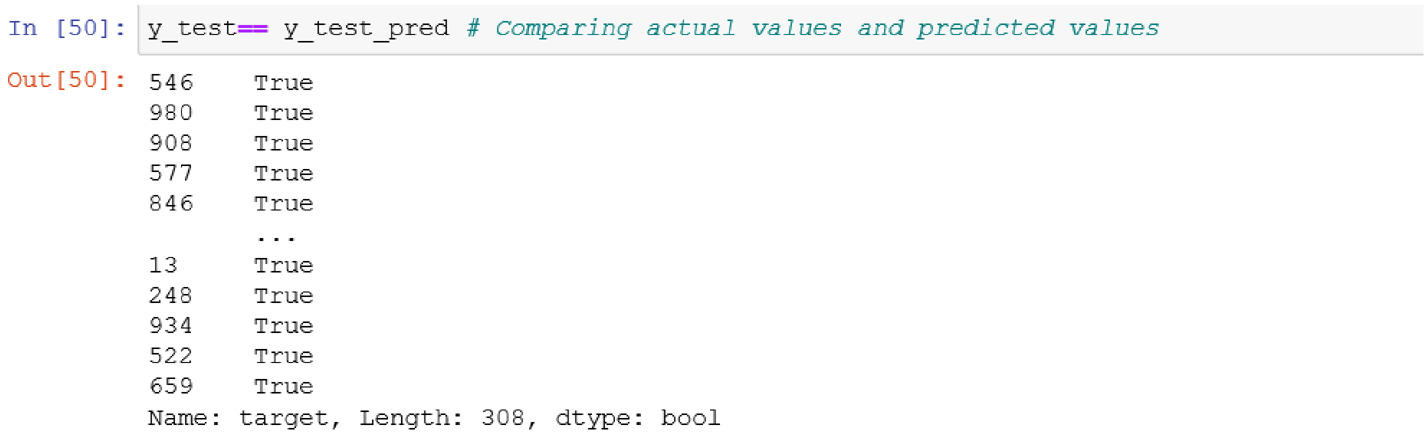
- Then we need to import linear regression method from linear_model package from scikit learn library

- We need to train the model based on our train set (that we have obtained from splitting)
- Then we have to test the model for the test set ,that is done as follows
    - We have a method called predict , using this method we need to predict the output for input test set and we need to compare the out but with the output test data
    - If the predicted values and the original values are close then we can say that model is trained with good accuracy



```
: # Predicting on train data
  # Syntax: objectName.predict(Input)
  y_train_pred = reg.predict(X_train)
  y_train_pred
```

Figure 11 : Predicting



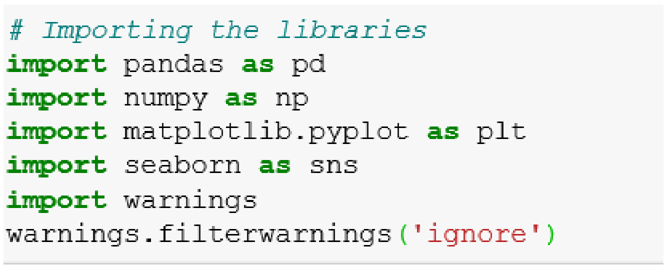
```
In [50]: y_test== y_test_pred # Comparing actual values and predicted values

Out[50]: 546    True
         980    True
         908    True
         577    True
         846    True
                ...
         13     True
         248    True
         934    True
         522    True
         659    True
         Name: target, Length: 308, dtype: bool
```

Figure 12 : comparing the predicted value with the original one

4.3 EVALUATING THE CASE STUDY:

- MULTIPLE LINEAR REGRESSION: A multiple linear regression model allows us to capture the relationship between multiple feature columns and the target column. Here's what the formula looks like: y^=a0+a1x1+a2x2+...+anxn

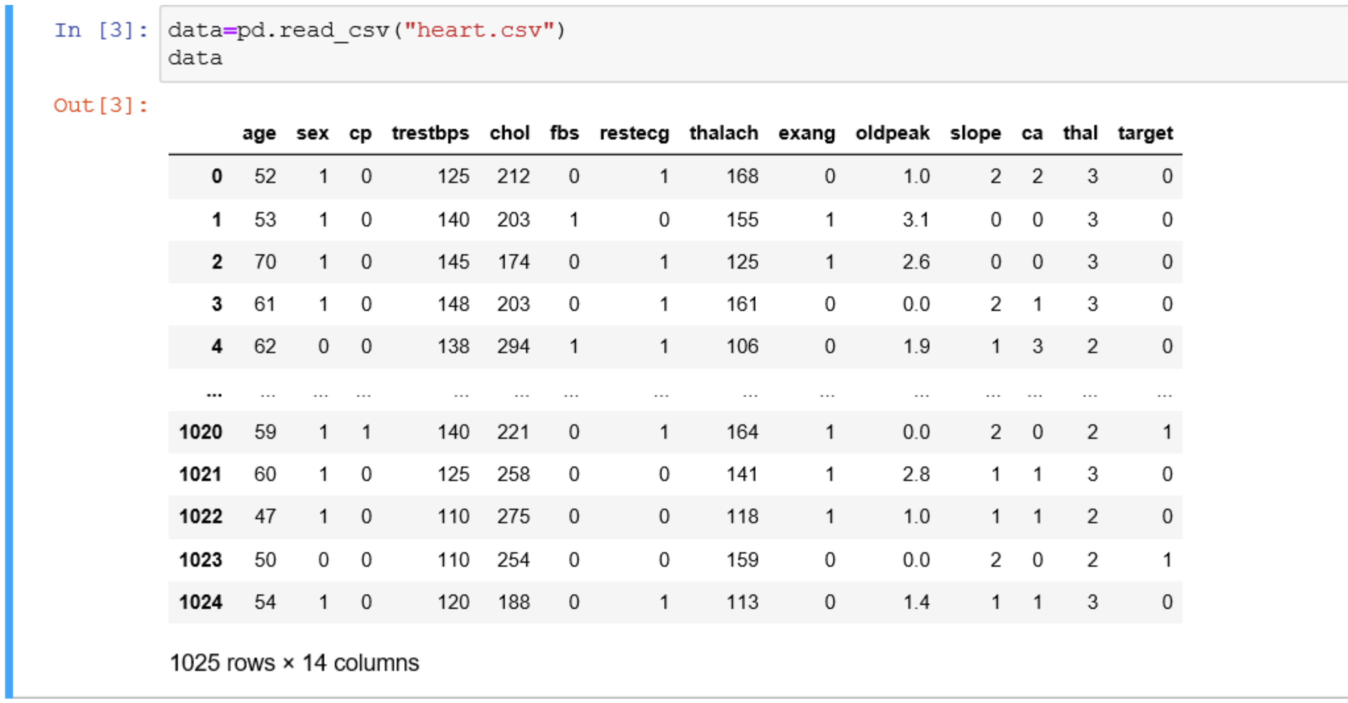
- Importing the required libraries



```
# Importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Figure 13 : Importing the libraries

- Reading the Data-Set



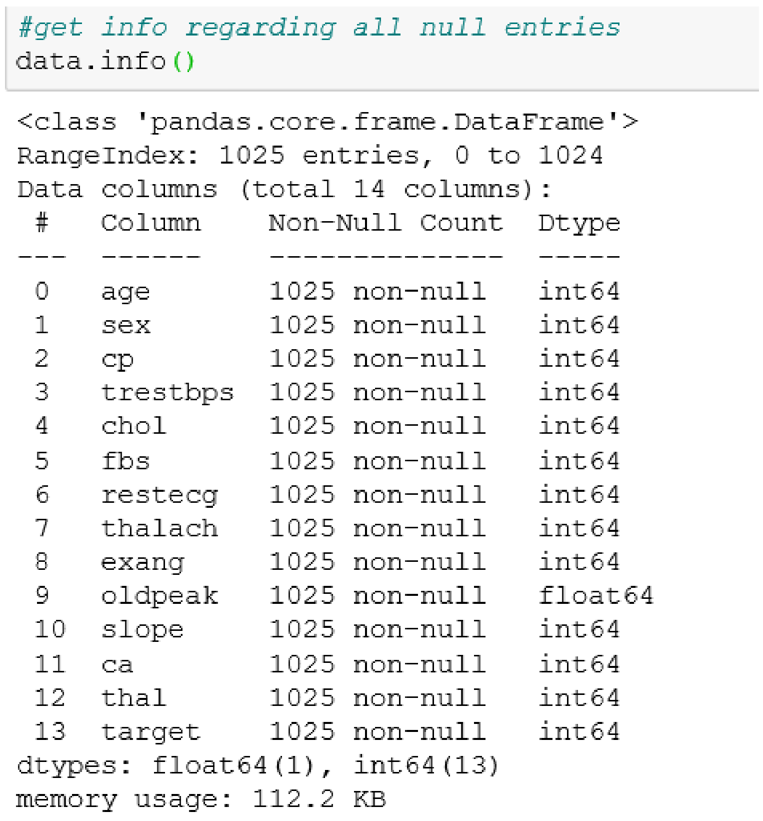
```
In [3]: data=pd.read_csv("heart.csv")
        data
Out[3]:
```

	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

Figure 14 : Reading the Data-Set

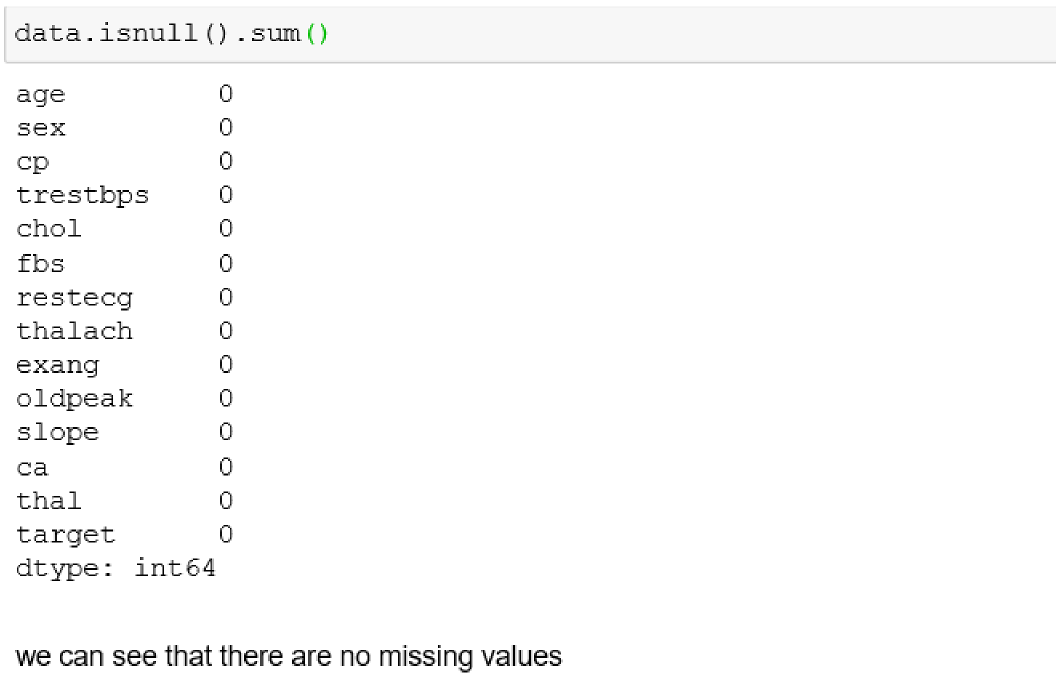
- Handling the missing values
  - There is a method called isnull() which gives the number of missing values in each and every column.
  - Using fillna() method each and every missing value is replaced by 0.



```
#get info regarding all null entries
data.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
```

Figure 15 : Before handling the missing the values



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

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

we can see that there are no missing values

Figure 16 : After handling the missing values

- Dealing with categorical data

- Using labelencoder from preprocessing package which is present in scikit learn, we can get dummies in place of categorical data

- Once we get dummies we need to fit and transform that to our dataframe

4.4 ALGORITHMS USED:

4.4.1 NAÏVE BAYES ALGORITHM:

NB tree is probabilistic model and used for developing real time applications in classification. Ex: Navie Bayes classification is used to filter the spam, predict the heart diseases, classifying documents , sentiment prediction in social media sites, detection of cancer diseases, etc. It make uses of independent feature that's why it is called navie. It depends on the Bayes theorem, even if the value of a feature is modified it doesn't shows any impact on other features. It more applicable for scalable application as it is scalable in nature. It identifies the problems in faster manner as it a probabilistic model. It associate with the conditional probability as well as bayes rule. Naïve Bayes in nature. It is used in many real world applications for classification. For instance, it is used in heart disease prediction, spam filtering, classifying cancer diseases, segmenting documents and predicting sentiments in online reviews. The Naïve Bayes classification technique is based on the Bayes theory. The features it uses are independent and hence the name naïve. It does mean that when a value of a feature is changed, it does not affect other feature directly. This algorithm is found faster as it is probabilistic. It is also scalable in nature and suitable for applications where scalability is in demand. It has its associated concepts like Bayes Rule and conditional probability.

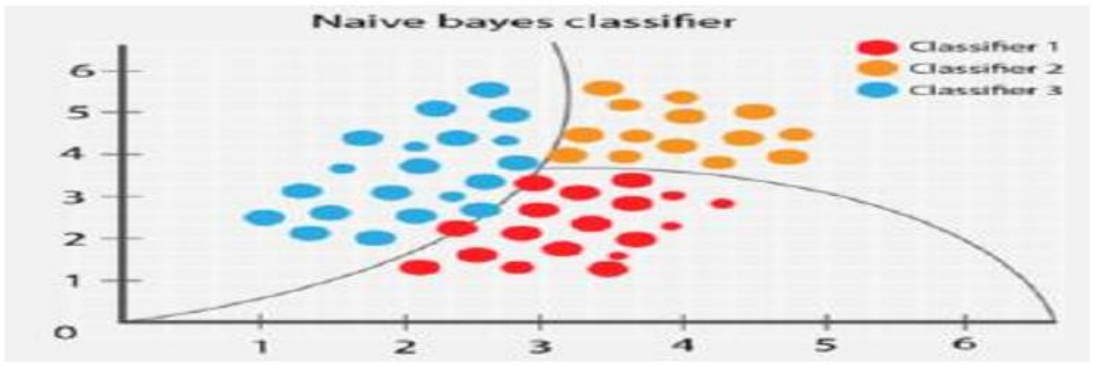
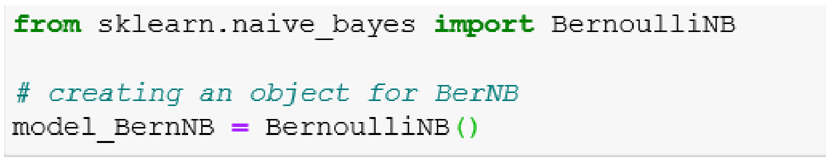
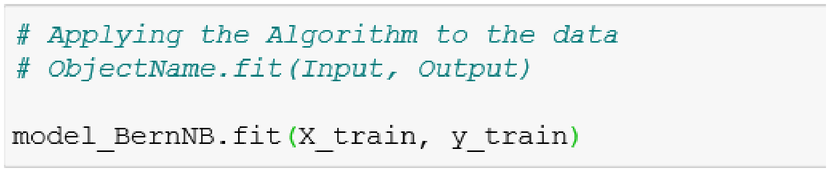


Figure 17: Naive bayes classifier

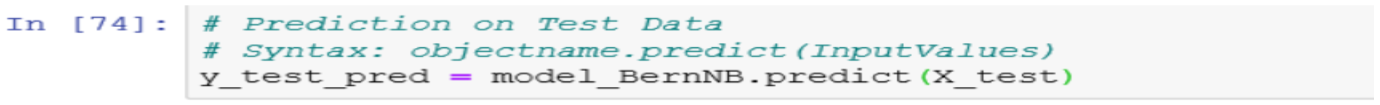
Model Development and Prediction:



```
from sklearn.naive_bayes import BernoulliNB

# creating an object for BerNB
model_BernNB = BernoulliNB()
```



```
# Applying the Algorithm to the data
# ObjectName.fit(Input, Output)

model_BernNB.fit(X_train, y_train)
```

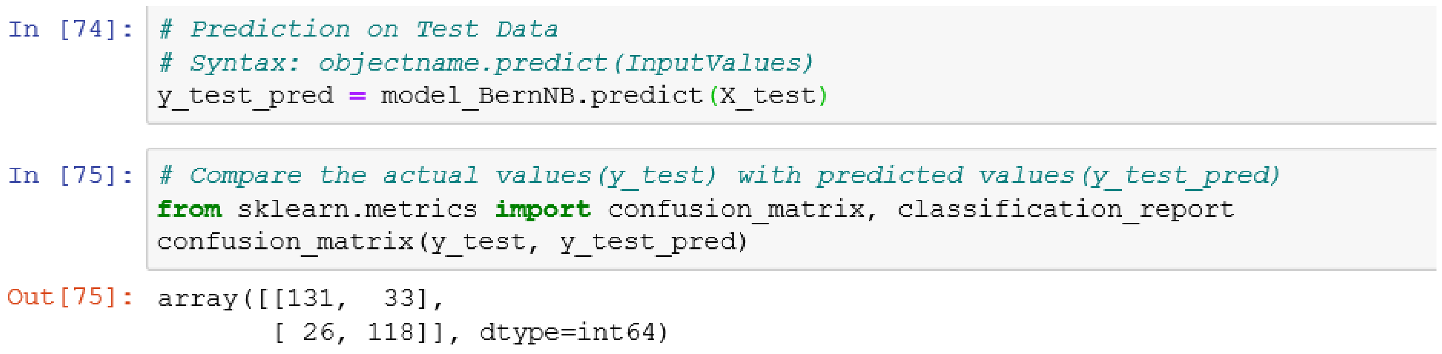


```
In [74]: # Prediction on Test Data
         # Syntax: objectname.predict(InputValues)
         y_test_pred = model_BernNB.predict(X_test)
```

Figure 18: Naive bayes classifier Development and prediction

Model Evaluation using Confusion Matrix:

A confusion matrix is a table that is used to evaluate the performance of a classification model. You can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.



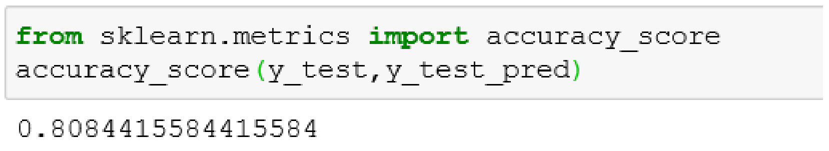
```
In [74]: # Prediction on Test Data
         # Syntax: objectname.predict(InputValues)
         y_test_pred = model_BernNB.predict(X_test)

In [75]: # Compare the actual values(y_test) with predicted values(y_test_pred)
         from sklearn.metrics import confusion_matrix, classification_report
         confusion_matrix(y_test, y_test_pred)

Out[75]: array([[131,  33],
                [ 26, 118]], dtype=int64)
```

Figure 19: Naive bayes classifier Evaluation using Confusion matrix

Confusion Matrix Evaluation Metrics:

Evaluating the model using model evaluation metrics such as accuracy.

Accuracy can be computed by comparing actual test set values and predicted values.



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

```
0.8084415584415584
```

Figure 21: Accuarcy using Naive bayes classifier

4.4.2 LOGISTIC REGRESSION ALGORITHM:

Logistic regression is one of the most popular machine learning algorithms for binary classification. This is because it is a simple algorithm that performs very well on a wide range of problems. Logistic Regression can be used for various classification problems such as spam detection, Diabetes prediction. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

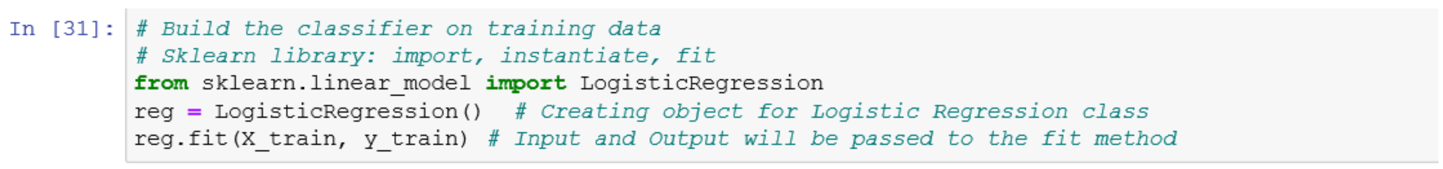
Types of Logistic Regression:

- Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.
- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

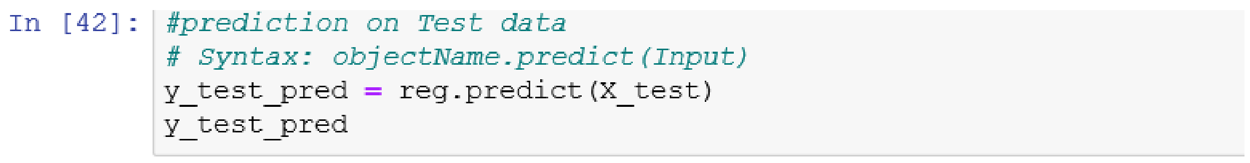
Model Development and Prediction:

First, import the Logistic Regression module and create a Logistic Regression classifier object using LogisticRegression() function.

Then, fit your model on the train set using fit() and perform prediction on the test set using predict().



```
In [31]: # Build the classifier on training data
         # Sklearn library: import, instantiate, fit
         from sklearn.linear_model import LogisticRegression
         reg = LogisticRegression()   # Creating object for Logistic Regression class
         reg.fit(X_train, y_train) # Input and Output will be passed to the fit method
```

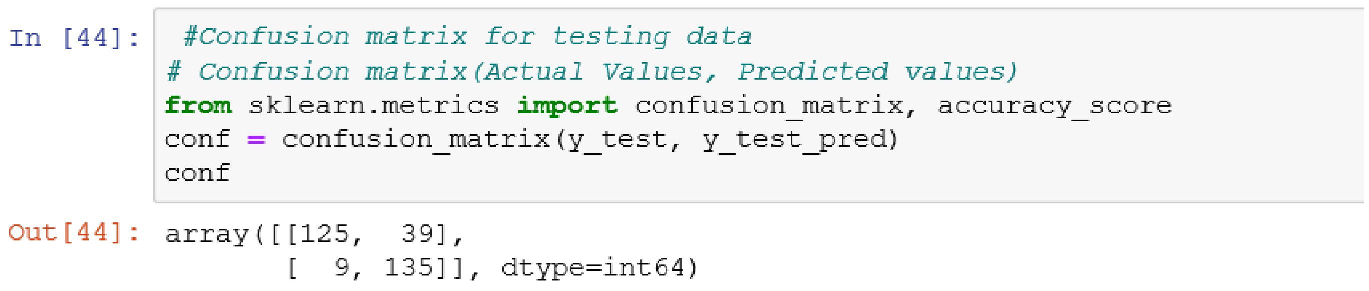


```
In [42]: #prediction on Test data
         # Syntax: objectName.predict(Input)
         y_test_pred = reg.predict(X_test)
         y_test_pred
```

Figure 22: Logistic Regression algorithm Development and prediction

Model Evaluation using Confusion Matrix:

A confusion matrix is a table that is used to evaluate the performance of a classification model. You can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.



```
In [44]:  #Confusion matrix for testing data
         # Confusion matrix(Actual Values, Predicted values)
         from sklearn.metrics import confusion_matrix, accuracy_score
         conf = confusion_matrix(y_test, y_test_pred)
         conf

Out[44]: array([[125,  39],
                [  9, 135]], dtype=int64)
```

Figure 23: Logistic Regression algorithm Evaluation using Confusion matrix

Here, you can see the confusion matrix in the form of the array object. The dimension of this matrix is 2*2 because this model is binary classification. You have two classes 0 and 1. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions.

Visualizing Confusion Matrix using Heatmap:

Visualizing the results of the model in the form of a confusion matrix using matplotlib.

Here, you will visualize the confusion matrix using Heatmap.

```
In [44]:  #Confusion matrix for testing data
         # Confusion matrix(Actual Values, Predicted values)
         from sklearn.metrics import confusion_matrix, accuracy_score
         conf = confusion_matrix(y_test, y_test_pred)
         conf

Out[44]: array([[125,  39],
                [  9, 135]], dtype=int64)

In [45]: sns.heatmap(confusion_matrix(y_test, y_test_pred), annot=True)

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x165c6385388>
```

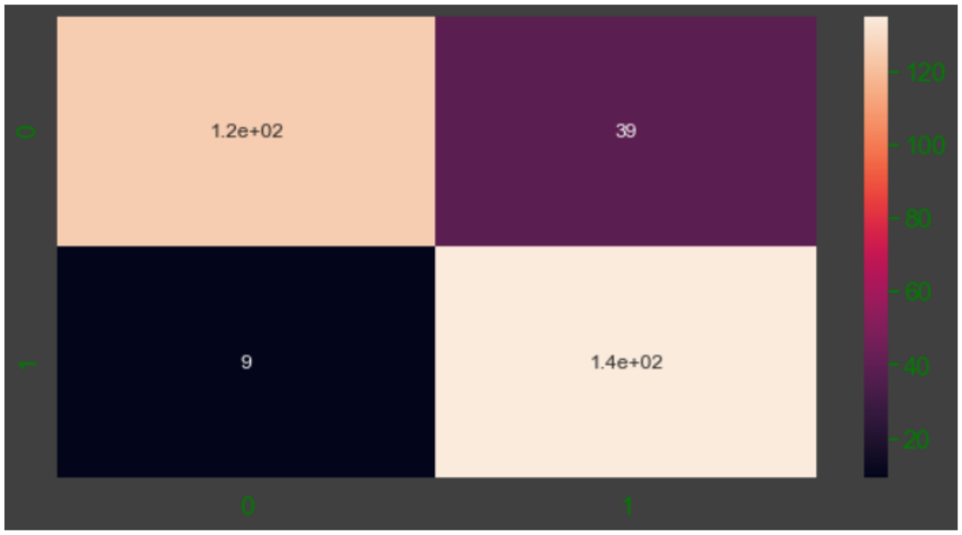
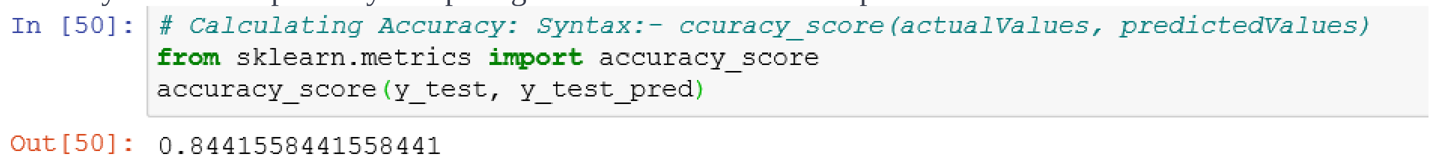


Figure 24: Logistic Regression algorithm visualizing Confusion matrix using Heatmap

Confusion Matrix Evaluation Metrics:

Evaluating the model using model evaluation metrics such as accuracy.

Accuracy can be computed by comparing actual test set values and predicted values.



```
In [50]: # Calculating Accuracy: Syntax:- ccuracy_score(actualValues, predictedValues)
         from sklearn.metrics import accuracy_score
         accuracy_score(y_test, y_test_pred)

Out[50]: 0.8441558441558441
```

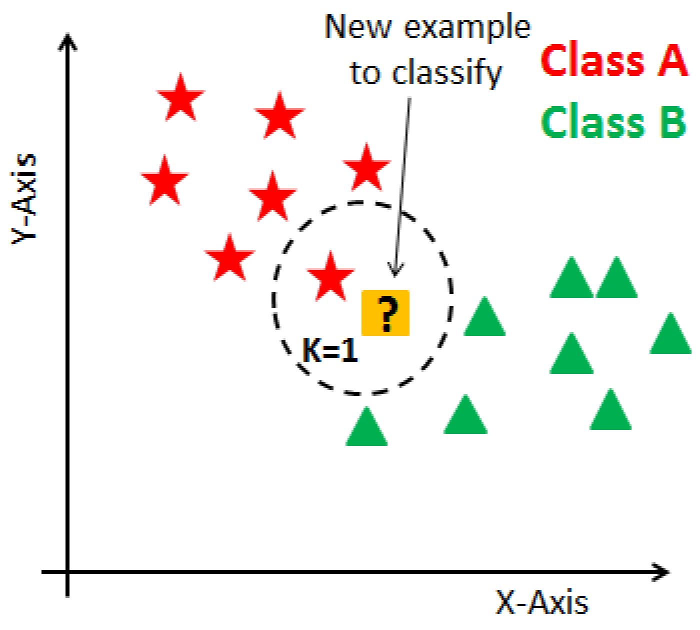
Figure 25: Accuracy using Logistic Regression algorithm

4.4.2 K NEAREST NEIGHBOR(KNN) ALGORITHM:

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition. KNN algorithm used for both classification and regression problems. KNN algorithm based on feature similarity approach. KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

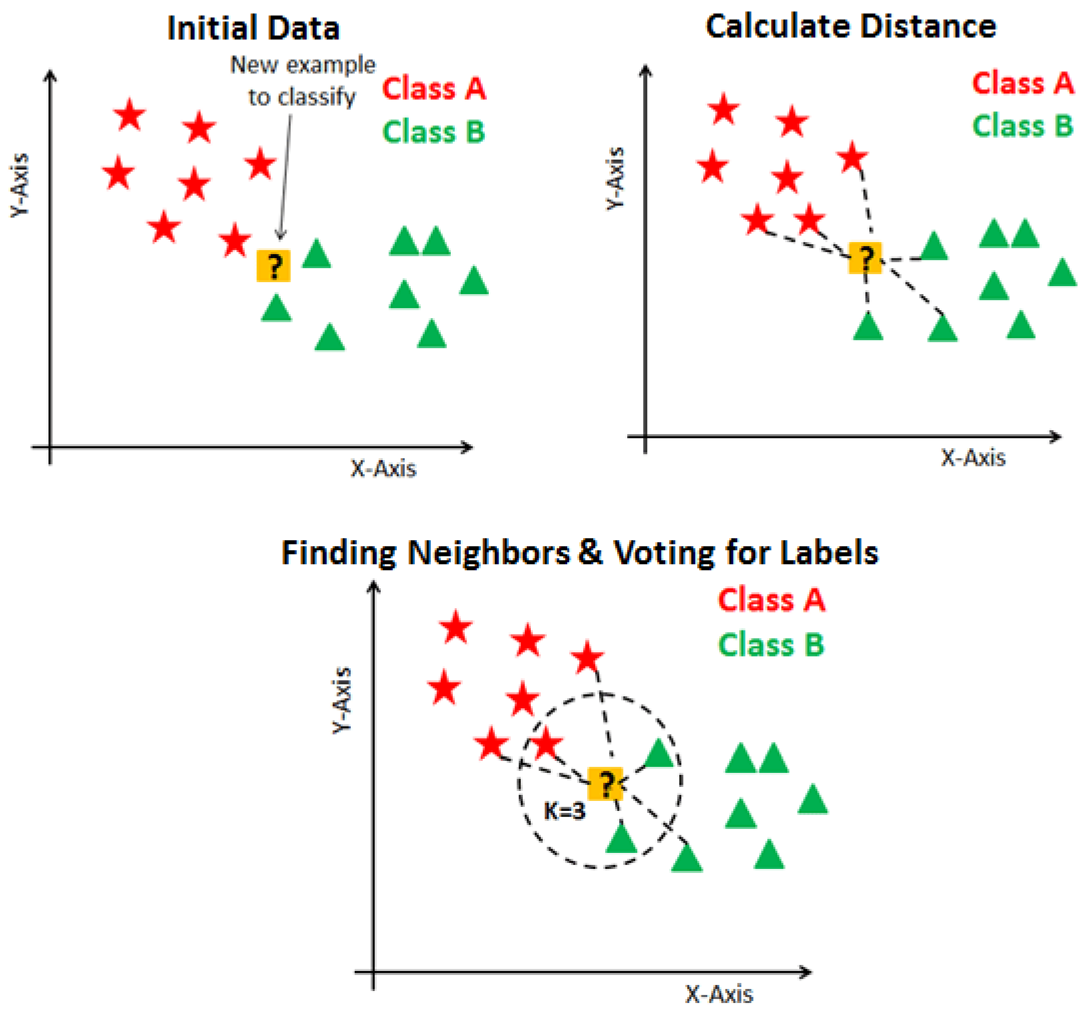
How does the KNN algorithm work?

In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to predict. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.



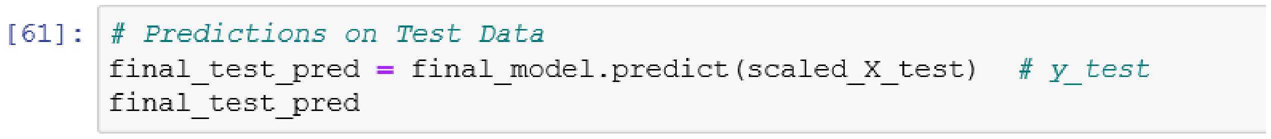
Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNN has the following basic steps:

1. Calculate distance
2. Find closest neighbors
3. Vote for labels



Model Development and Prediction:

```
# Scaling Data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# Scaling for training data
scaled_X_train = pd.DataFrame(scaler.fit_transform(X_train), columns = X_train.columns)
scaled_X_train

#Scaling for test data
#Testing the data based on training data
scaled_X_test = pd.DataFrame(scaler.transform(X_test), columns = X_test.columns)
scaled_X_test
```

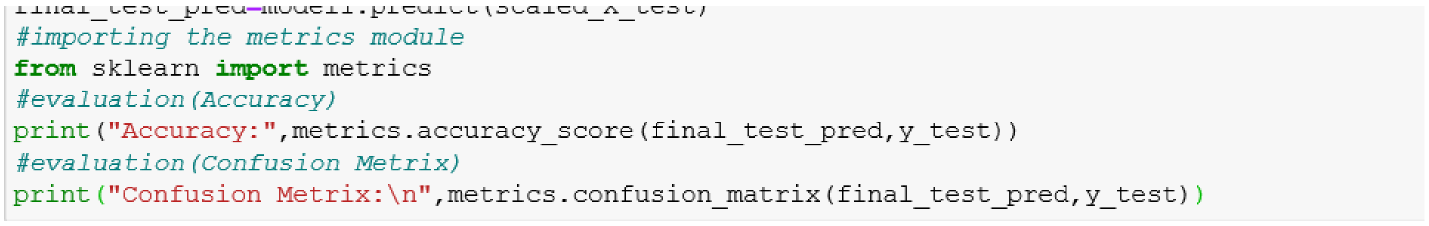


```
[61]: # Predictions on Test Data
      final_test_pred = final_model.predict(scaled_X_test)   # y_test
      final_test_pred
```

Figure 26: K-Nearest Neighbor algorithm Development and prediction

Model Evaluation using Confusion Matrix:

A confusion matrix is a table that is used to evaluate the performance of a classification model. You can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.



```
#importing the metrics module
from sklearn import metrics
#evaluation(Accuracy)
print("Accuracy:",metrics.accuracy_score(final_test_pred,y_test))
#evaluation(Confusion Metrix)
print("Confusion Metrix:\n",metrics.confusion_matrix(final_test_pred,y_test))
```

Figure 27:K-Nearest Neighbor algorithm Evaluation using Confusion matrix

Visualizing Confusion Matrix using Heatmap:

Visualizing the results of the model in the form of a confusion matrix using matplotlib.

Here, you will visualize the confusion matrix using Heatmap.

```
In [62]: sns.heatmap(confusion_matrix(y_test, final_test_pred), annot=True, fmt='d')
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x165c6689788>
```

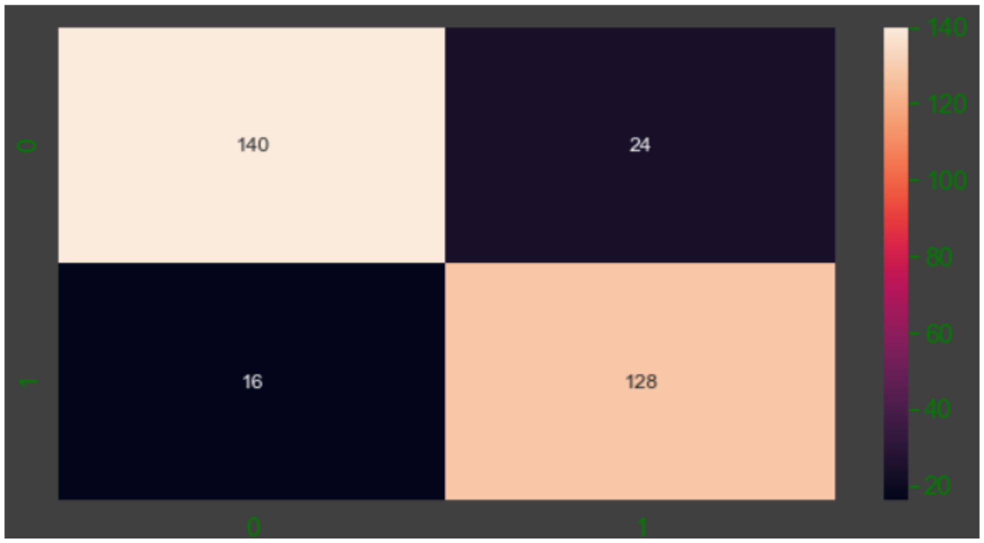


Figure 28: K-Nearest Neighbor algorithm visualizing Confusion matrix using Heatmap

Confusion Matrix Evaluation Metrics:

Evaluating the model using model evaluation metrics such as accuracy.

Accuracy can be computed by comparing actual test set values and predicted values.

```
In [64]: predicted_acc_test=accuracy_score(y_test, final_test_pred)
         predicted_acc_test
Out[64]: 0.8701298701298701
```

Figure 29: Accuracy using K-earest Neighbor algorithm

CONCLUSION:

In this project, the three different machine learning algorithms such as Naïve Bayes, Logistic Regression and k- nearest neighbor are applied to the dataset for the prediction of heart disease occurrence. It utilizes the data such as age, blood pressure, cholesterol, diabetes and then tries to predict the possible heart

disease patient in next 10 years. Family history of heart disease can also be a reason for developing a heart disease. So, this data of the patient can also be included for further increasing the accuracy of the model. This work will be useful in identifying the possible patients who may suffer from heart disease in the next 10 years. This may help in taking preventive measures and hence try to avoid the possibility of
heart disease for the patient. So when a patient is predicted as positive for heart disease, then the
medical data for the patient can be closely analysed by the doctors. An example would be - suppose the
patient has diabetes which may be the cause for heart disease in future and then the patient can be
given treatment to have diabetes in control which in turn may prevent the heart disease. So, this project shows the prediction accuracy of heart disease occurrence for each of the algorithm. Finally we conclude which algorithm is best suitable to solve this problem of heart disease occurrence prediction.

REFERENCES:

[1] https://www.kaggle.com/semakulapaul/cereals-datase t

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

[3] https://en.wikipedia.org/wiki/Machine_learning