DATA SCIENCE

(PREDICTION OF HEART DISEASES)

SUMMER INTERNSHIP SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENT FOR UNDERGRADUATE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

BY

BORKUT SREEJA

221710304012

UNDER THE GUIDANCE OF

Mrs.Manisha Patil

Assistant Professor

GITAM

(DEEMED TO BE UNIVERSITY)

(Estd. u/s 3 of the UGC Act, 1956)

VISAKHAPATNAM • HYDERABAD • BENGALURU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM School of Technology

GITAM (Deemed to be University),Hyderabad-502329

DECLARATION

I Submit this industrial training work entitled **“PREDICTION OF HEART DISEASE”** 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 patil, 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 **Borkut Sreeja**

Date : 14-07-2020 **221710304012**

CERTIFICATE

This is to certify that the Industrial Training Report entitled PREDICTION OF HEART DISEASE" is being submitted by Borkut Sreeja(221710304012) 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 2020-21

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.Manisha Patil	**Prof.S.Phani Kumar**
Assistant Professor	Professor and HOD
Department of CSE	Department of CSE

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 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 Patil** 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.

BORKUT SREEJA

221710304012

ABSTRACT

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 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 Artificial Neural Network (ANN),Decision tree, Fuzzy Logic, K-Nearest Neighbour(KNN), Naïve Bayes This paper provides an insight of the existing algorithm and it gives an overall summary of the existing work.

Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of clinical data analysis. The amount of data in the healthcare industry is huge. Data mining turns the large collection of raw healthcare data into information that can help to make informed decisions and predictions.

Borkut Sreeja

221710304012

TABLE OF CONTENTS

1.INFORMATION ABOUT DATA SCIENCE

1.1 What is Data Science

Data science is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. Data science is related to data mining, deep learning and big data

1.2 Need of Data Science

Companies need to use data to run and grow their everyday business. The fundamental goal of data science is to help companies make quicker and better decisions, which can take them to the top of their market, or at least – especially in the toughest red oceans be a matter of long-term survival Industries require data scientists to assist them in making a smarter decision. To predict the information everyone requires data scientists. Big Data andd Data Science hold the key to the future. Data Science is important for better marketing.

1.3 Uses of Data Science

1. Internet Search

2. Targeted Advertising and re-targeting

3. Website Recommendation Systems

4. Advanced Image Recognition

5. Speech Recognition

6. Price Comparison Websites

7. Airline Route Planning
8. Fraud and Risk Detection

9. Delivery Logistics

2. Information About Machine Learning

2.1 Introduction:

Machine Learning(ML) is the scientific study of algorithms and statistical models thatcomputer 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, cybersex 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 technique

The process flow depicted here represents how machine learning works.

Figure1: 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 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.

2.4 TYPES OF MACHINE LEARNING:

2.4.1 SUPERVISED LEARNING:

Supervised learning is the most popular paradigm for machine learning. It is the easiest to understand and the simplest to implement. It is very similar to teaching a child with the use of flash cards.
Supervised learning is often described as task-oriented because of this. It is highly focused on a singular task, feeding more and more examples to the algorithm until it can accurately perform on that task.
Over time, the algorithm will learn to approximate the exact nature of the relationship between examples and their labels. When fully-trained, the supervised learning algorithm will be able to observe a new, never-before-seen example and predict a good label for it.

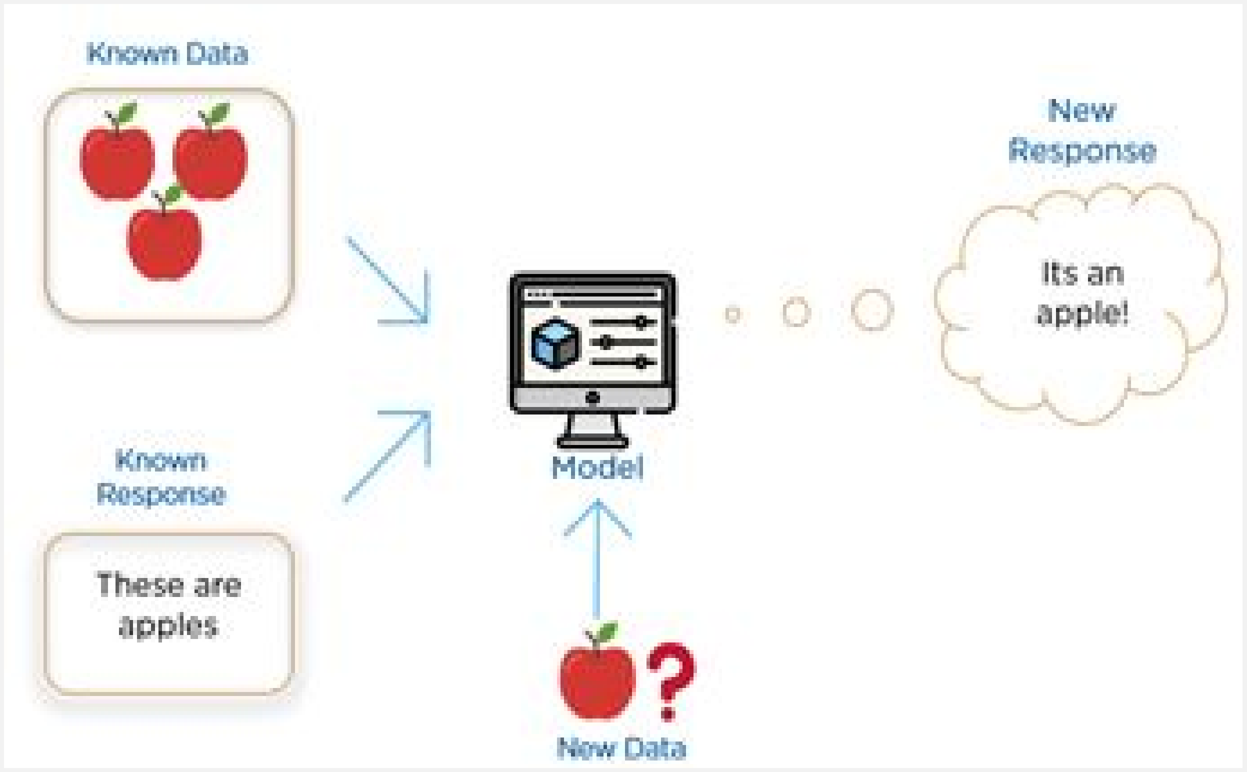


Figure2: supervised learning

Supervised learning is often described as task-oriented because of this. It is highly focused on a singular task, feeding more and more examples to the algorithm until it can accurately perform on that task. Some algorithms may be specifically designed for classification (such as logistic regression) or regression (such as linear regression) and some may be used for both types of problems with minor modifications (such as artificial neural networks).

2.4.2 UnsupervisedLearning:

Unsupervised learning is very much the opposite of supervised learning. It features no labels. Instead, our algorithm would be fed a lot of data and given the tools to understand the properties of the data. From there, it can learn to group, cluster, and/or organize the data in a way such that a human (or other intelligent algorithm) can come in and make sense of the newly organized data.

What makes a data insupervised learning such an interesting area is that an overwhelming majority on this world is unlabeled. Having intelligent algorithms that can take our terabytes and terabytes of unlabeled data and make sense of it is a huge source of potential profit for many industries. That alone could help boost productivity in a number of fields.

For example, what if we had a large database of every research paper ever published and we had an unsupervised learning algorithms that knew how to group these in such a way so that you were always aware of the current progression within a particular domain of research. Now, you begin to start a research project yourself, hooking your work into this network that the algorithm can see. As you write your work up and take notes, the algorithm makes suggestions to you about related works, works you may wish to cite, and works that may even help you push that domain of research forward. With such a tool, your productivity can be extremely boosted.

Because unsupervised learning is based upon the data and its properties, we can say that unsupervised learning is data-driven.

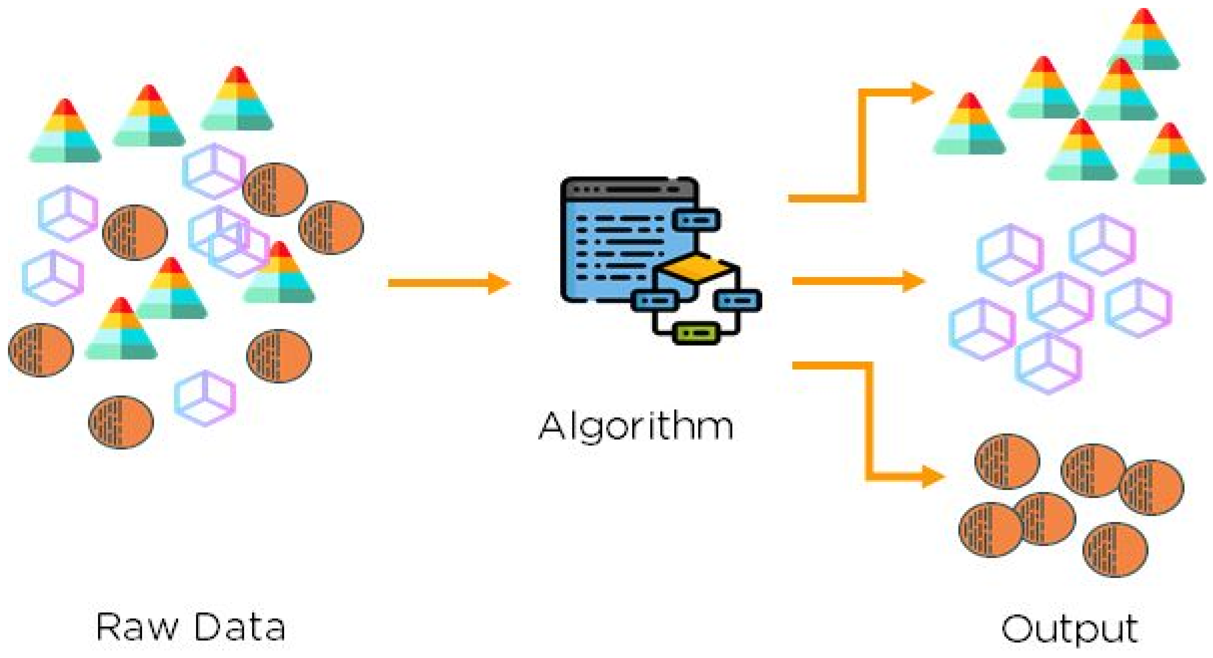


Figure 3: unsupervised learning

2.4.3 SemiSupervisedLearning:

In this type of learning, the algorithm is trained upon a combination of labeled and unlabeled data. Typically, this combination will contain a very small amount of labeled data and a very large amount of unlabeled data. The basic procedure involved is that first, the programmer will cluster similar data using an unsupervised learning algorithm and then use the existing labeled data to label the rest of the unlabeled data. The typical use cases of such type of algorithm have a common property among them – The acquisition of unlabeled data is relatively cheap while labeling the said data is very expensive.

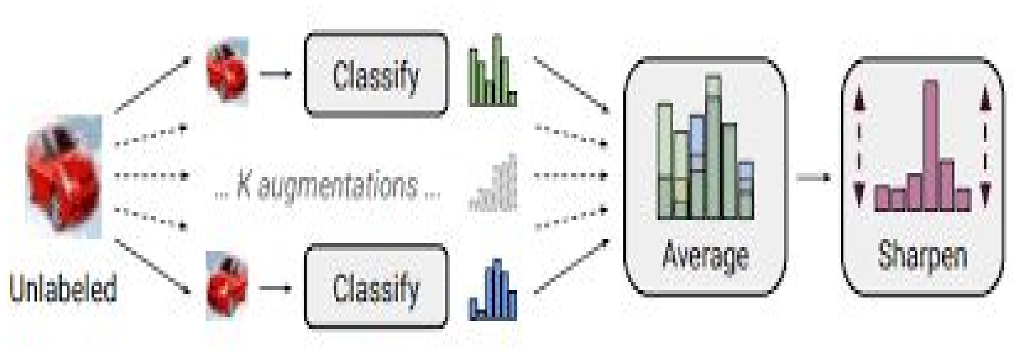


Figure 4:semi supervised learning

Intuitively, one may imagine the three types of learning algorithms as Supervised learning where astudent is under the supervision of a teacher at both home and school, Unsupervised learning where a student has to figure out a concept himself and Semi-Supervised learning where a teacher teaches a few concepts in class and gives questions as homework which are based on similar concepts.

2.5 Relation between Data Mining, Machine Learning and Deep Learning:

Both data mining and machine learning fall under the aegis of Data Science, which makes sense since they both use data. Both processes are used for solving complex problems, so consequently, many people (erroneously) use the two terms interchangeably. This isn't so surprising, considering that machine learning is sometimes used as a means of conducting useful data mining. While data gathered from data mining can be used to teach machines, so the lines between the two concepts become a bit blurred.

a branch of machine learning called deep learning (DL) appeared. The popularity of machine learning and the development of the computing capacity of computers enabled this new technology. Deep learning as a concept is very similar to machine learning but uses different algorithms. While machine learning works with regression algorithms or decision trees, deep learning uses neural networks that function very similarly to the biological neural connections of our brain.

3. INFORMATION ABOUT PYTHON

3.1 Introduction:

Python is an open-source (free) programming language that is used in web programming, data science, artificial intelligence, and many scientific applications. Learning Python allows the programmer to focus on solving problems, rather than focusing on syntax.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.
- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.

Python can be used for rapid prototyping, or for production-ready software development.

3.2 History of Python:

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in December 1989 by Guido Van Rossum at CWI in Netherland
- In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce.
- Python 2.0 added new features such as list comprehensions, garbage collection systems.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released.

3.3 Features of Python:

- Easy to Learn and Use
- Expressive Language
- Interpreted Language

- Cross-platform Language
- Free and Open Source
- Object-Oriented Language
- Extensible
- GUI Programming Support
- Dynamic Memory Allocation

3.4 PythonSetup :

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

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

3.4.1 Installation(using python IDLE):

● Installing python is generally easy ,and nowadays many Linux and MacOS

Distributions include are centpython.

● Download python from www.python.org

● When the download is completed, double click the file and follow the instructions to installit.

● When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Figure 5:python download

3.4.2 Installation(usingAnaconda):

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)

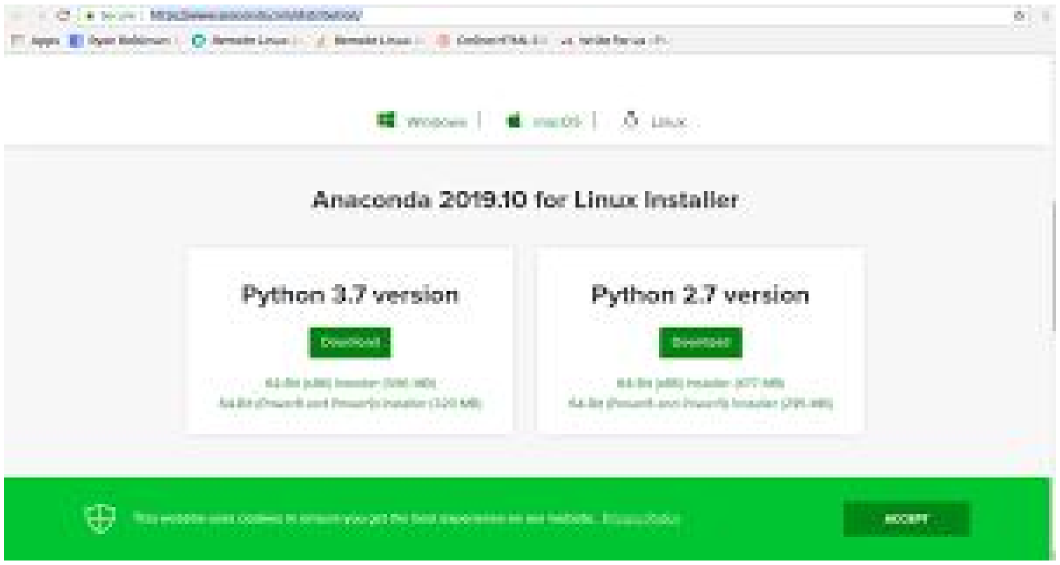


Figure6: installation of anaconda for linux

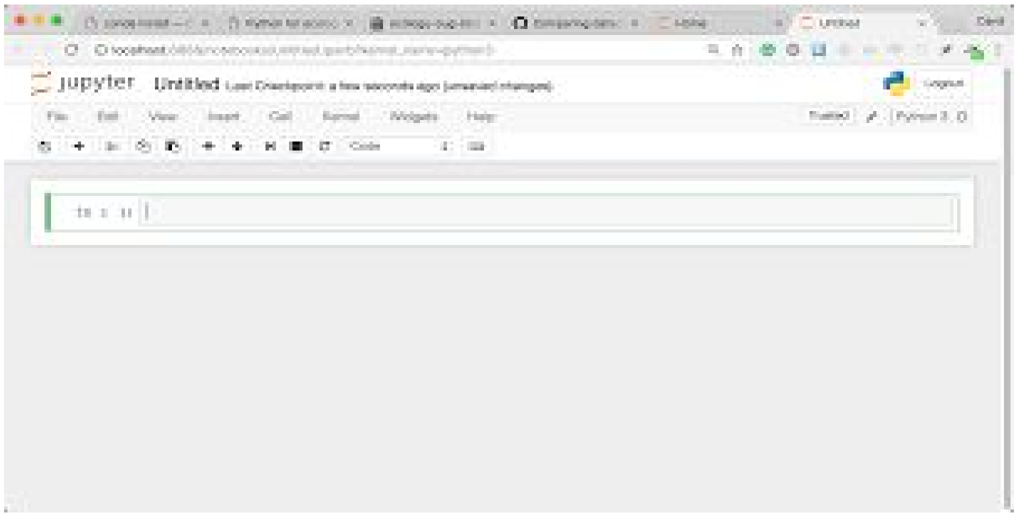


Figure 7: jupyter notebook

3.5 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 –

- Numbers
- Strings
- Lists
- Tuples
- Dictionary

3.5.1 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).

3.5.2 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.

3.5.3 Lists:

● Lists are the most versatile of Python's compound data types.
● A list contains items separated by commas and enclosed within square brackets 11 ([]).
● 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.

3.5.4 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: 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.

3.5.5 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.

3.6 FUNCTION:

3.6.1 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.

3.7 PYTHON USING OOP's CONCEPTS:

3.7.1 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.

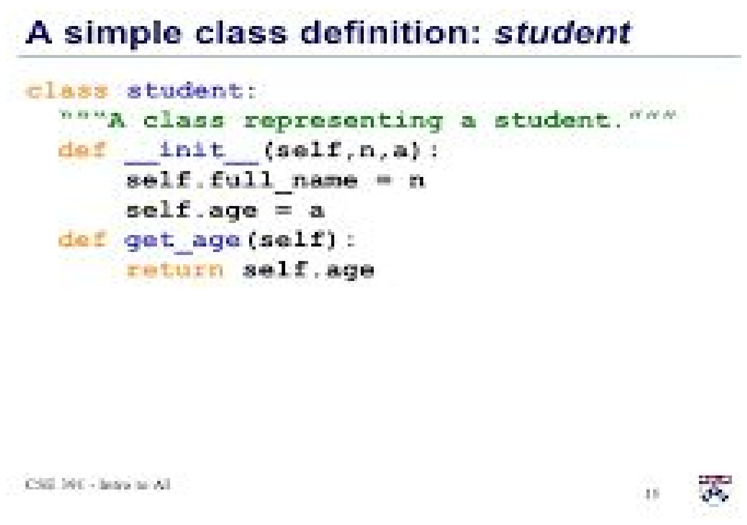
● Data member: A class variable or instance variable that holds data associated with a class and its objects.

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

● Defining a Class:

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(i.e. ():) when we define a class. Similarly, the body of our class is indented like a indented like a functions body



A simple class definition: *student*

```
class student:
  """A class representing a student."""
  def __init__(self,n,a):
      self.full_name = n
      self.age = a
  def get_age(self):
      return self.age
```

CSE 391 - Intro to AI

15

Figure 8: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 4 CASE STUDY

4.1 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.

4.2 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

5. MODEL BUILDING

5.1 Preprocessing of The Data:

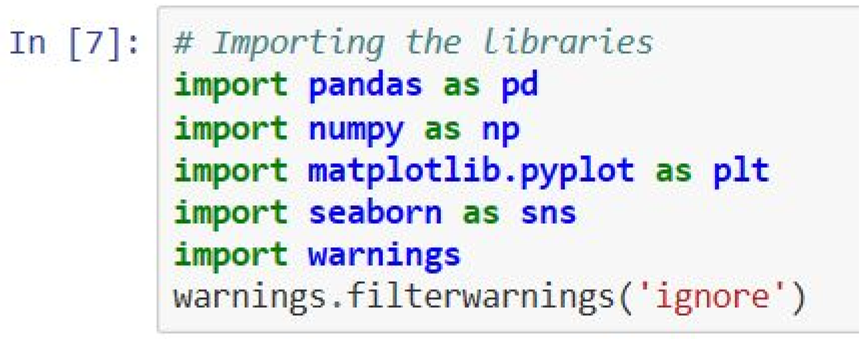
Preprocessing of the data actually involves the following steps:

5.1.1 Getting the Data Set:

We can get the data set from the database or we can get the data from client.

5.1.2 IMPORTING THE LIBRARIES:

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



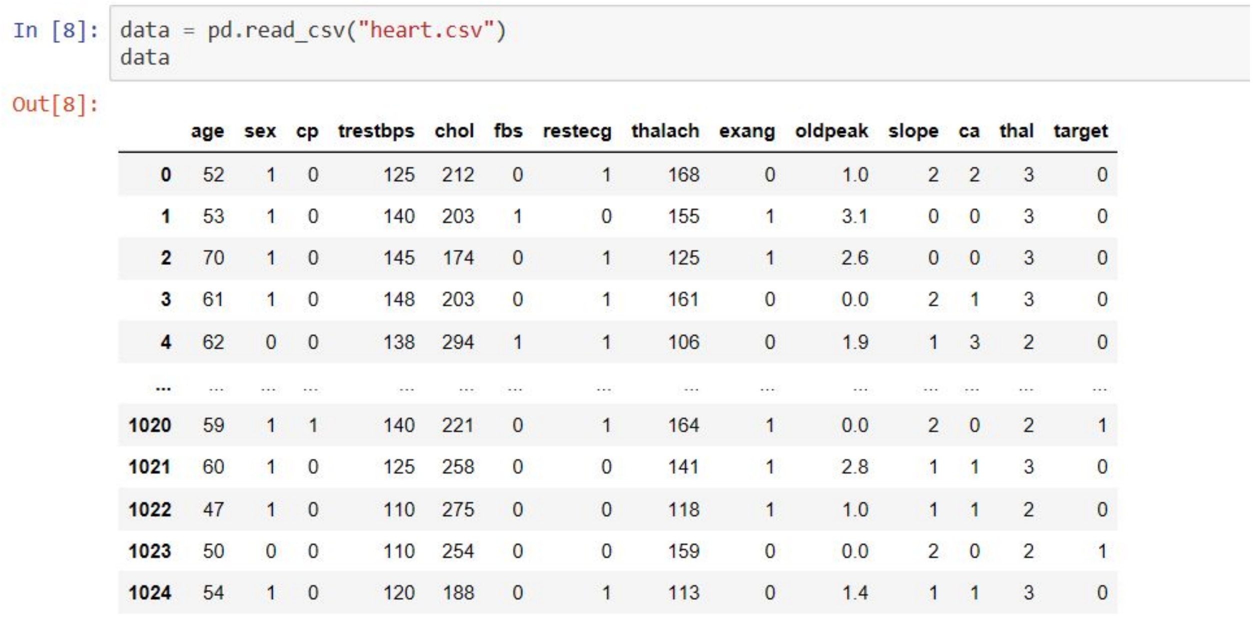
```
In [7]: # Importing the libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')
```

Figure 9:Importing Libraries

5.1.3. 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.

READING THE DATASET



```
In [8]: data = pd.read_csv("heart.csv")
        data
Out[8]:
```

	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

Figure 10:reading the dataset

5.2 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)

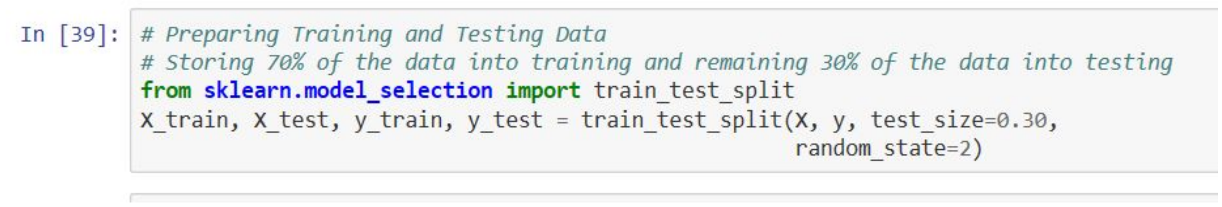
- test set - a subset to test the trained model.(To test whether the model has correctly learnt)

- 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)



```
In [39]: # 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 11:IMPORTING TRAIN _TEST_SPLIT

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

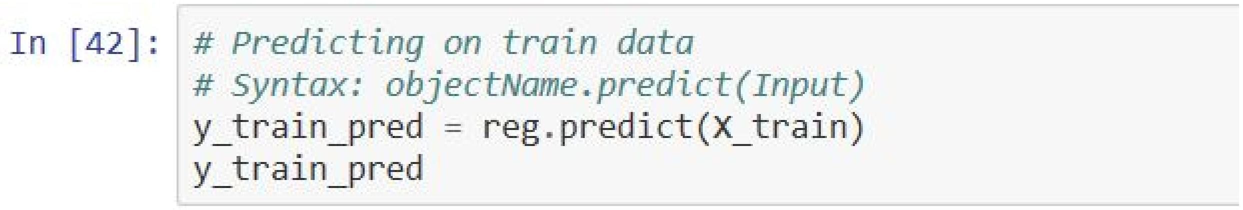
- We need to train the model based on our train set (that we have obtained from splitting)

- 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.

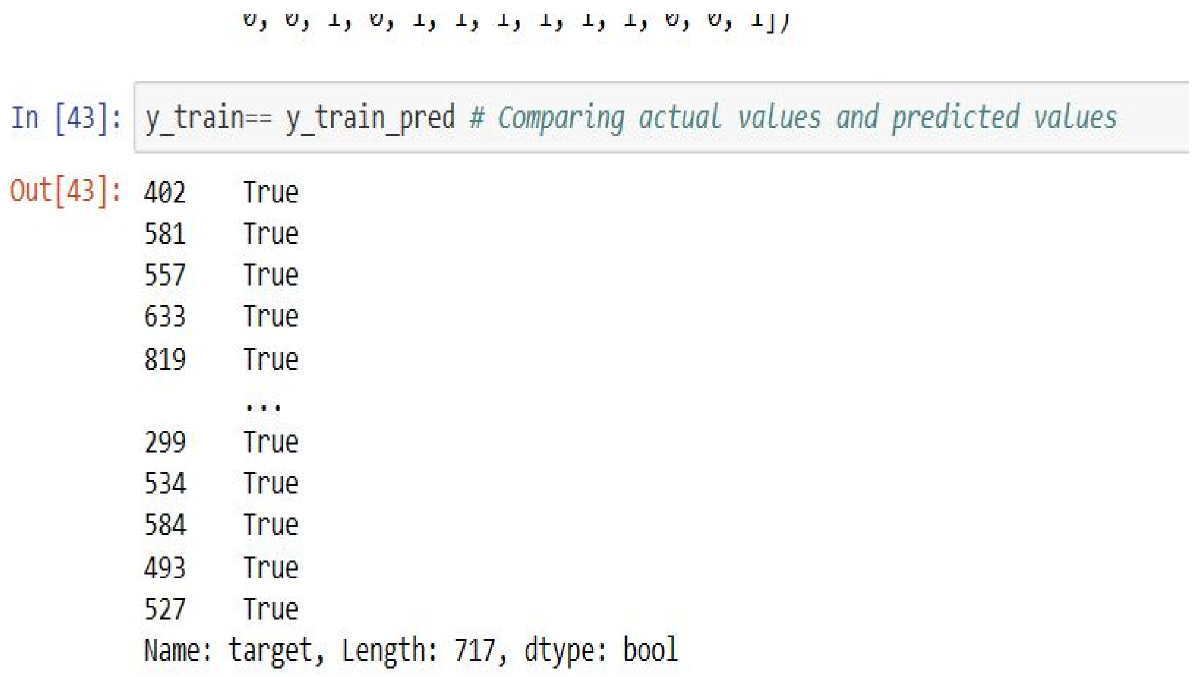
  - If the predicted values and the original values are close then we can say

that model is trained with good accuracy



```
In [42]: # Predicting on train data
         # Syntax: objectName.predict(Input)
         y_train_pred = reg.predict(X_train)
         y_train_pred
```

Figure 12: predicting



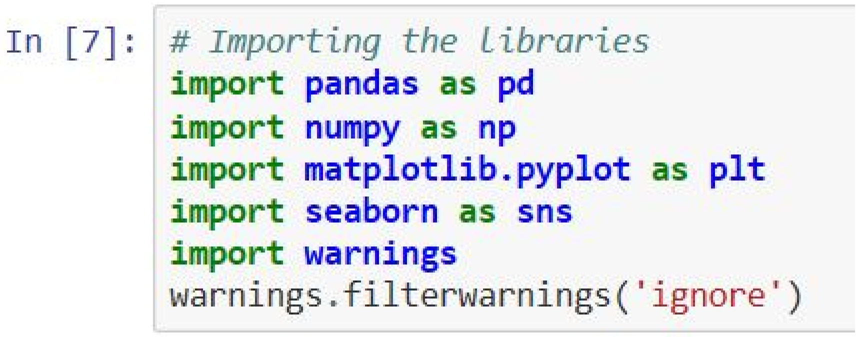
```
       0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1])

In [43]: y_train== y_train_pred # Comparing actual values and predicted values

Out[43]: 402    True
         581    True
         557    True
         633    True
         819    True
                ...
         299    True
         534    True
         584    True
         493    True
         527    True
         Name: target, Length: 717, dtype: bool
```

Figure 13: comparing the predicted value with the original one.

5.3 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.

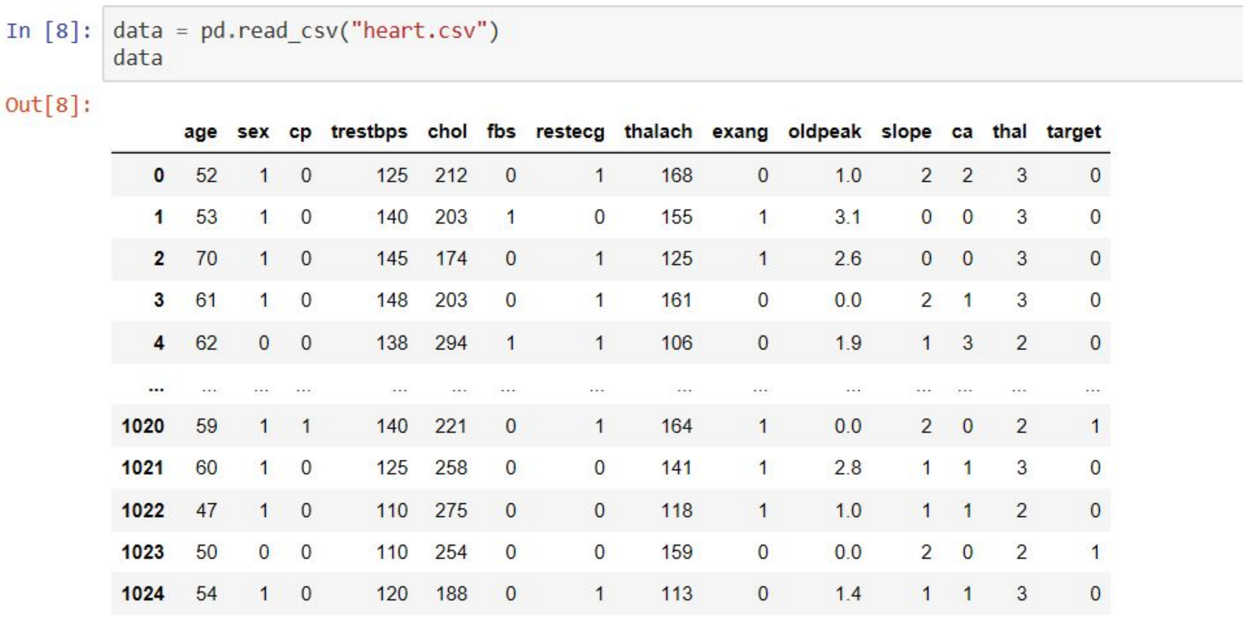
- Importing the required libraries



```
In [7]: # 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 14: Importing the libraries

- Reading the data set



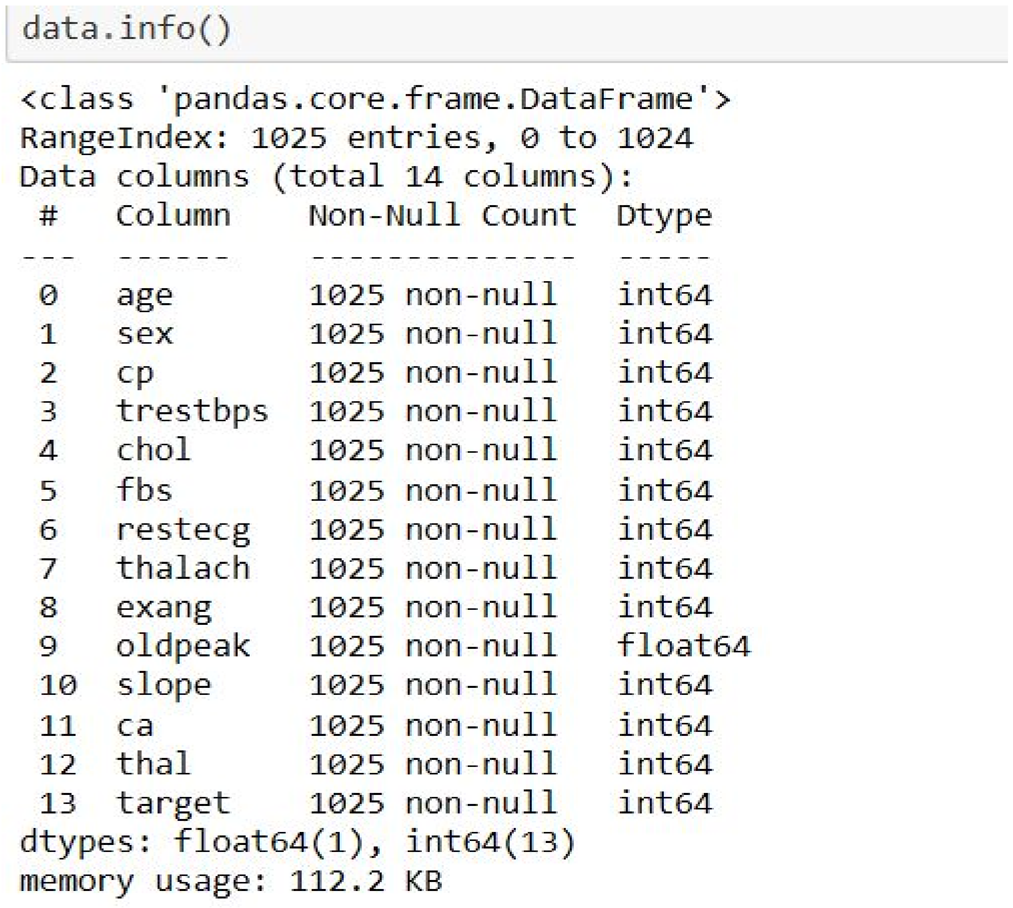
```
In [8]: data = pd.read_csv("heart.csv")
        data

Out[8]:
```

	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

Figure15: 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

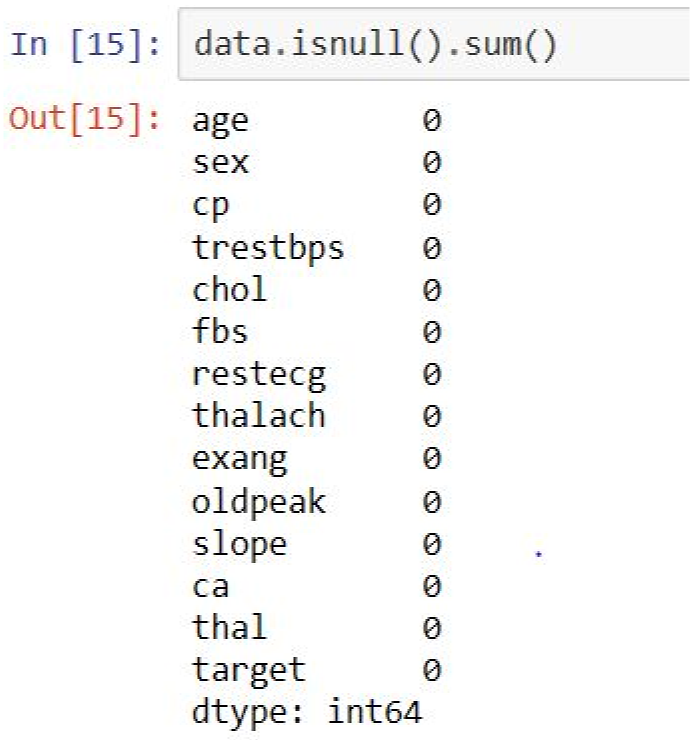
- Using fillna() method each and every missing value is replaced by 0.



```
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 16:Before handling missing values



```
In [15]: data.isnull().sum()

Out[15]: 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
```

Figure 17:After handling missing values

5.3 Caterorical Data:

Machine Learning models are based on equations, we need to replace the text numbers. So that we can include the numbers in the equations.

● Categorical Variables are of two types: Nominal and Ordinal

● Nominal: The categories do not have any numeric ordering in between them. They don't have any ordered relationship between each of them. Examples: Male or Female, any colour

● Ordinal: The categories have a numerical ordering in between them. Example: Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer satisfaction survey, high low medium

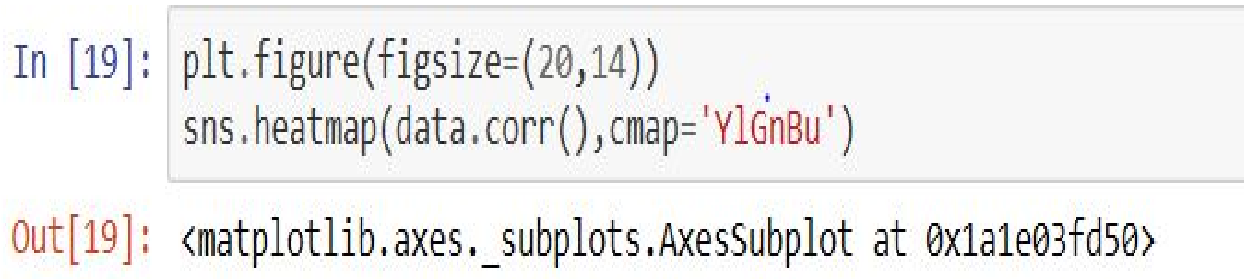
● Categorical data can be handled by using dummy variables, which are also called as indicator variables.

● **Handling categorical data using dummies:**

In pandas library we have a method called get_dummies() which creates dummy variables for those categorical data in the form of 0's and 1's.
Once these dummies got created we have to concat this dummy set to our dataframe or we can add that dummy set to the dataframe**.**

5.2 Generating Plots :

5.2.1 Visualize the data between all the Features:



```
In [19]: plt.figure(figsize=(20,14))
         sns.heatmap(data.corr(),cmap='YlGnBu')

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1e03fd50>
```

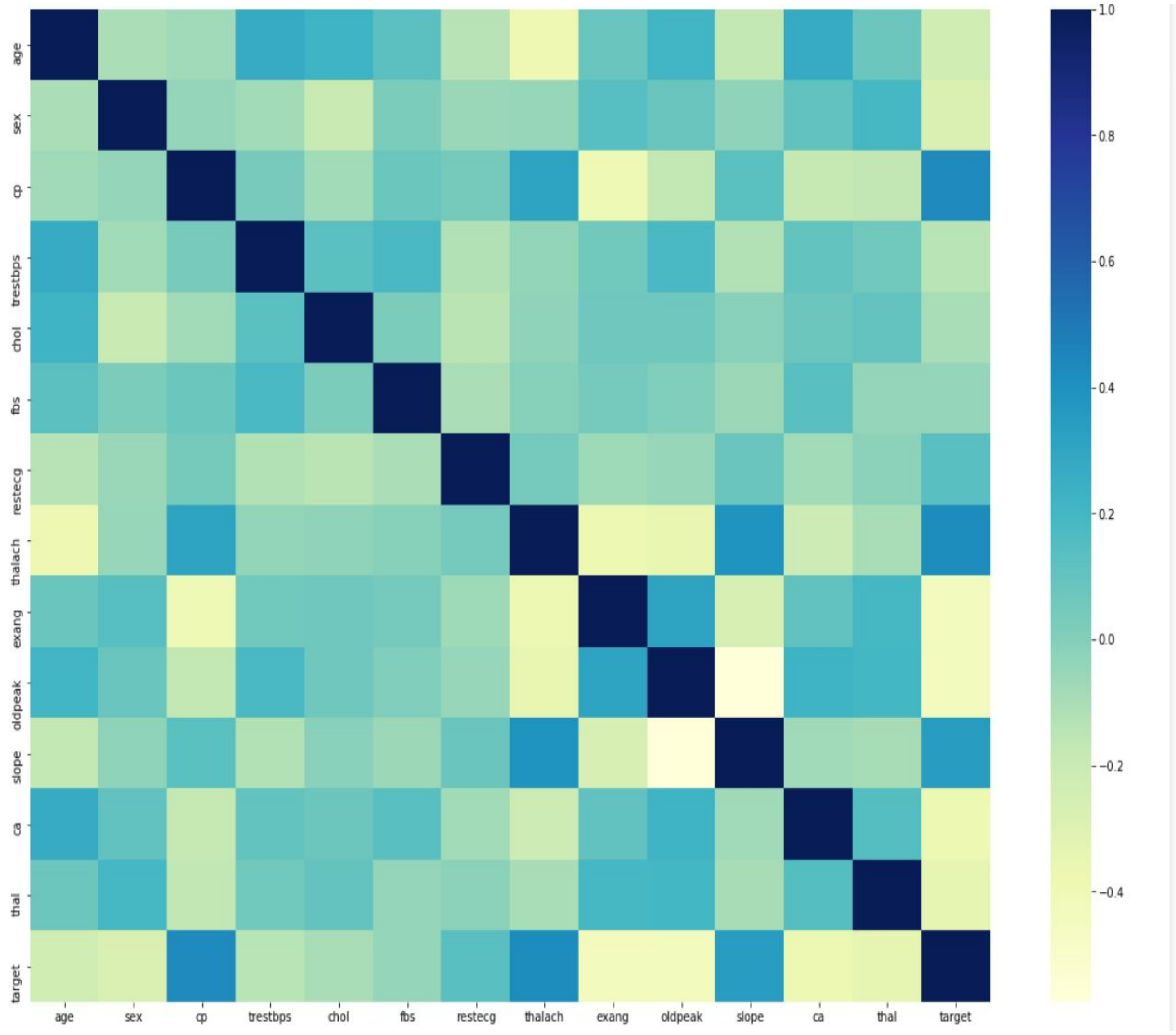
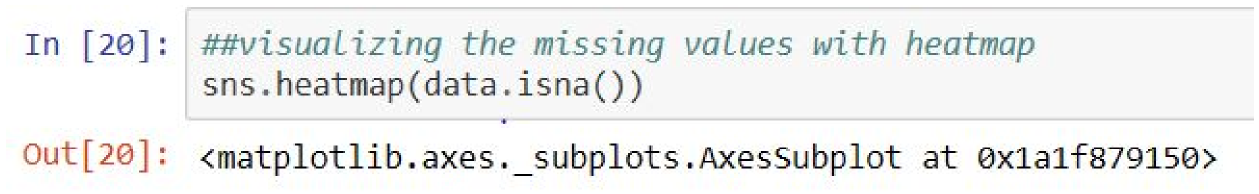


Figure 18:Visualizing the data between all the Features

5.2.2 Visualize the data betweenTarget and the Features:



```
In [20]: ##visualizing the missing values with heatmap
         sns.heatmap(data.isna())

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1f879150>
```

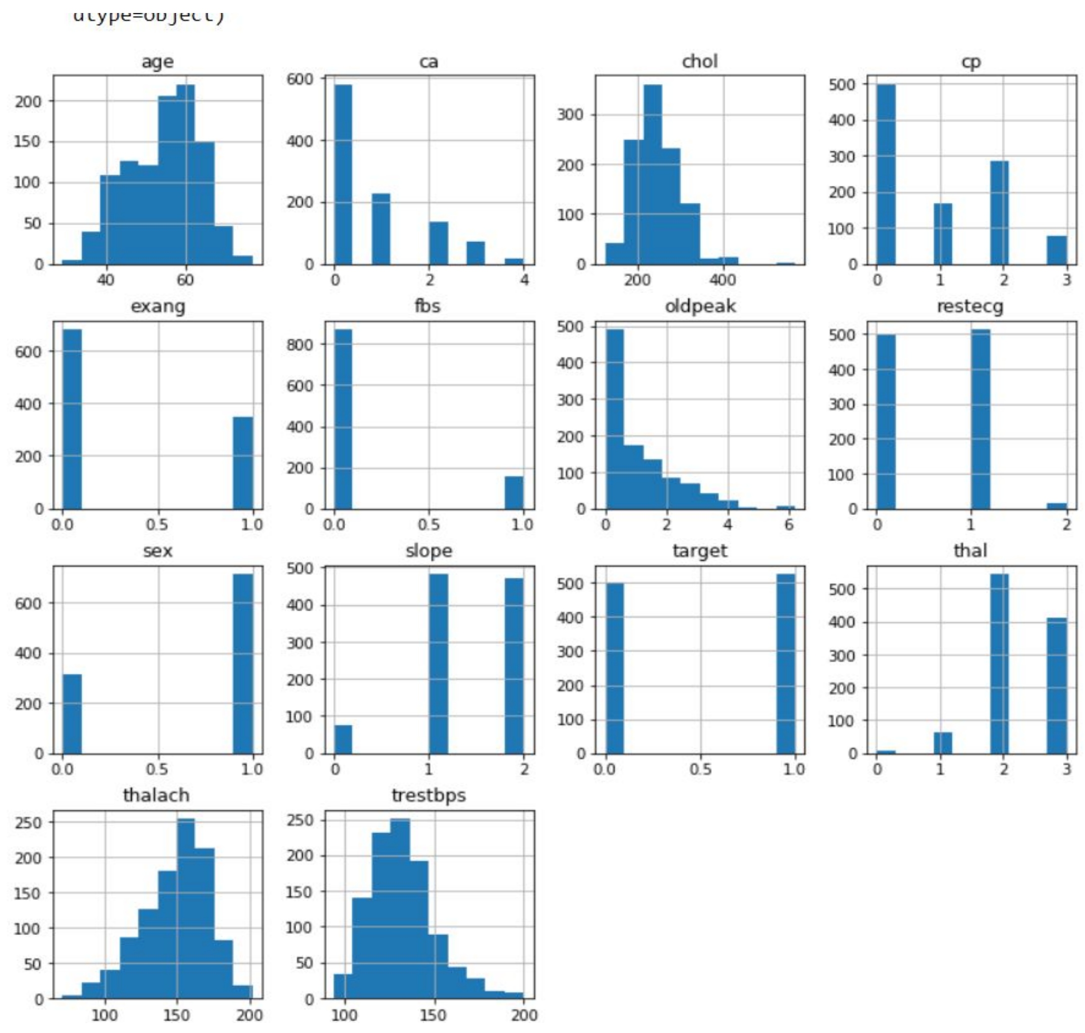


Figure 19:Visualize the data betweenTarget and the Features

6.ALGORITHMS USED

6.1 Naive 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.

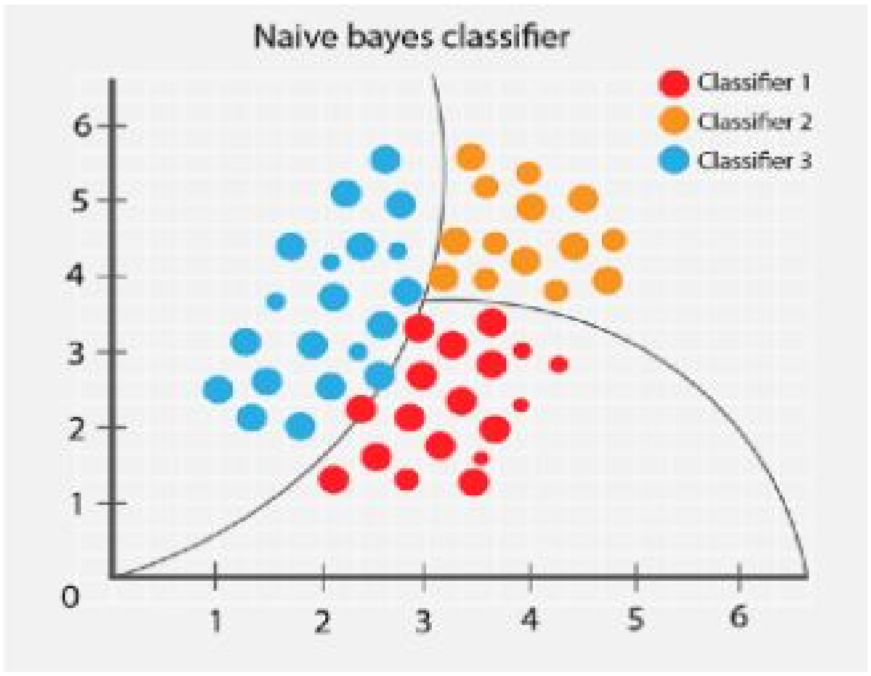
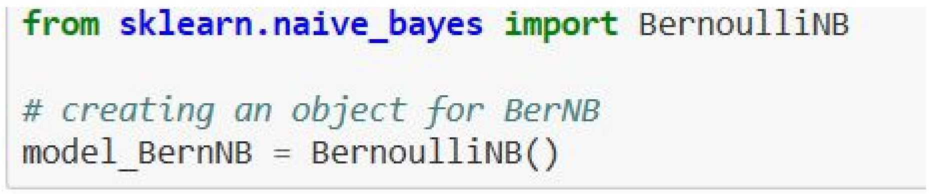
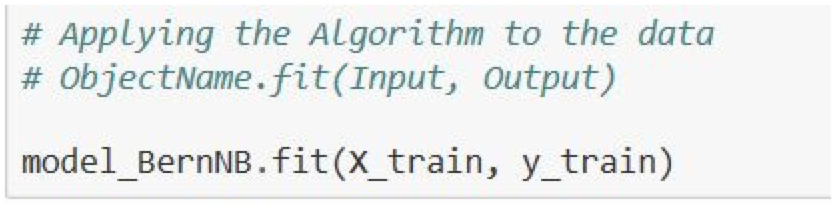


Figure 20:Naive Bayes classifier

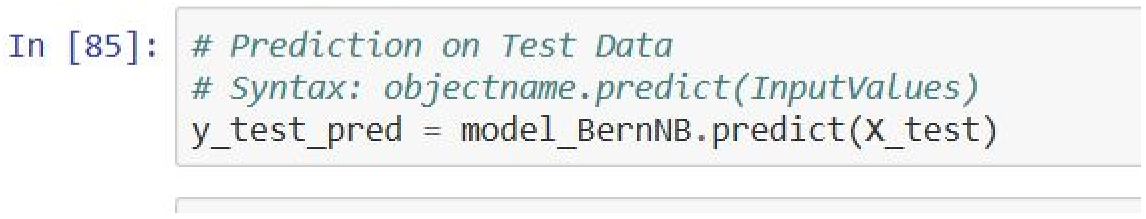
Model Development and Prediction:



```
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)
```

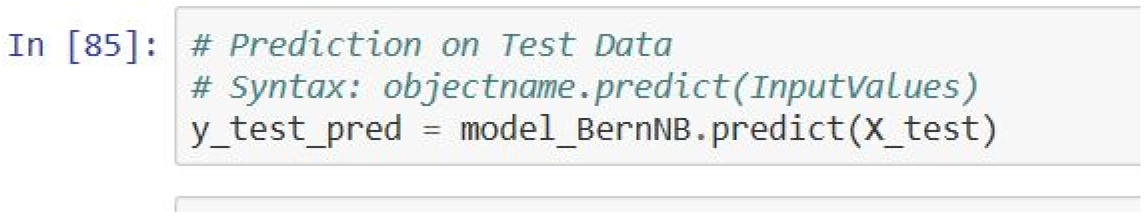


```
In [85]: # Prediction on Test Data
         # Syntax: objectname.predict(InputValues)
         y_test_pred = model_BernNB.predict(X_test)
```

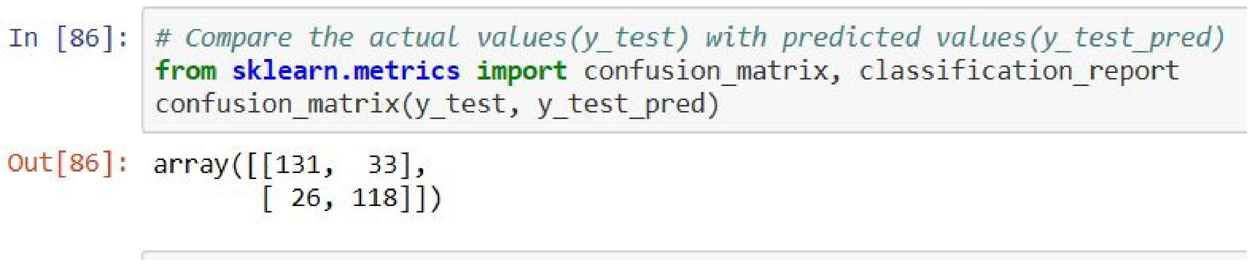
Figure 21: Naive bayes classifier Development and prediction

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



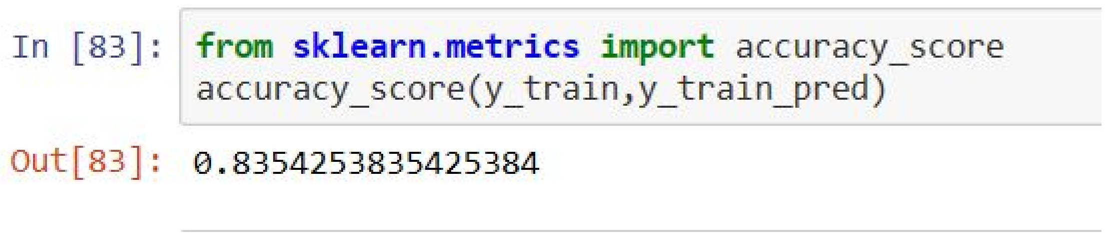
```
In [86]: # Compare the actual values(y_test) with predicted values(y_test_pred)
         from sklearn.metrics import confusion_matrix, classification_report
         confusion_matrix(y_test, y_test_pred)

Out[86]: array([[131,  33],
                [ 26, 118]])
```

Figure 22: Naive bayes 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.



```
In [83]: from sklearn.metrics import accuracy_score
         accuracy_score(y_train,y_train_pred)

Out[83]: 0.8354253835425384
```

Figure 23: Accuracy using Naive bayes classifier

6.2LOGISTIC 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.

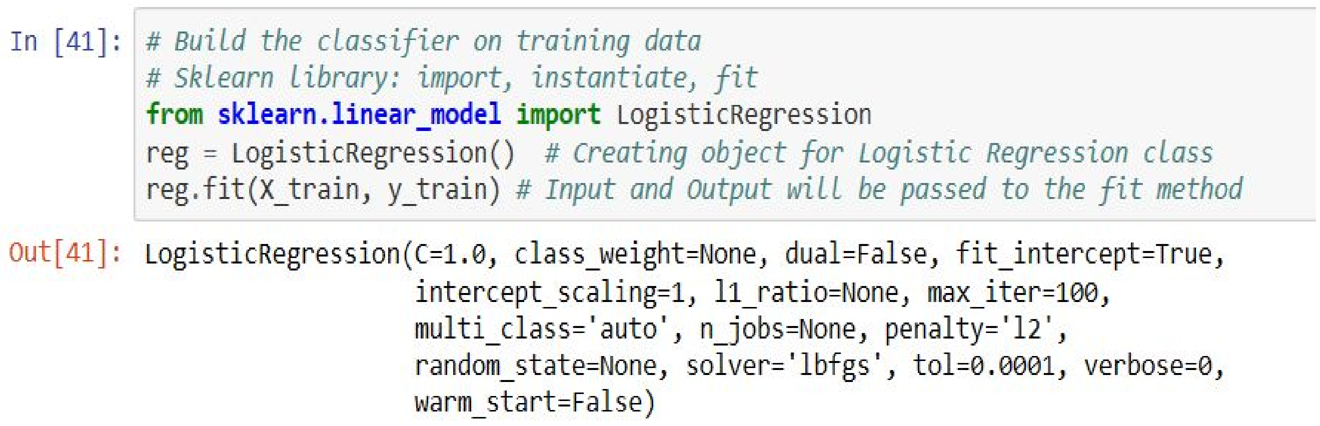
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.

- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

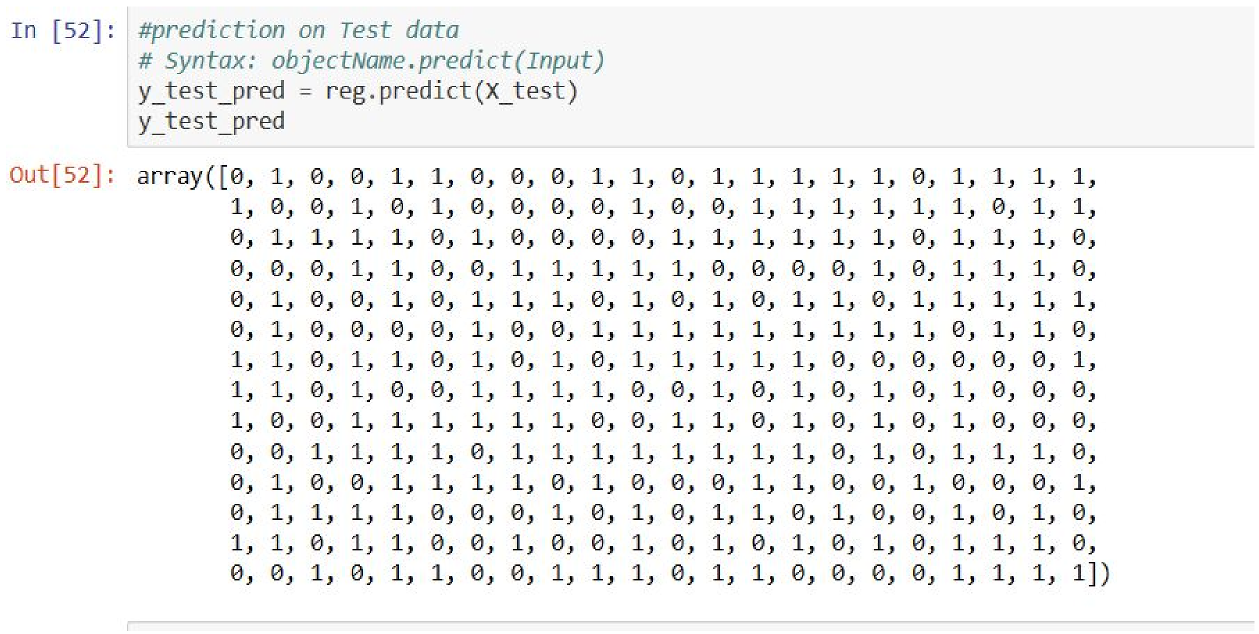
Model Development and Prediction:

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 [41]: # 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

Out[41]: 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)
```

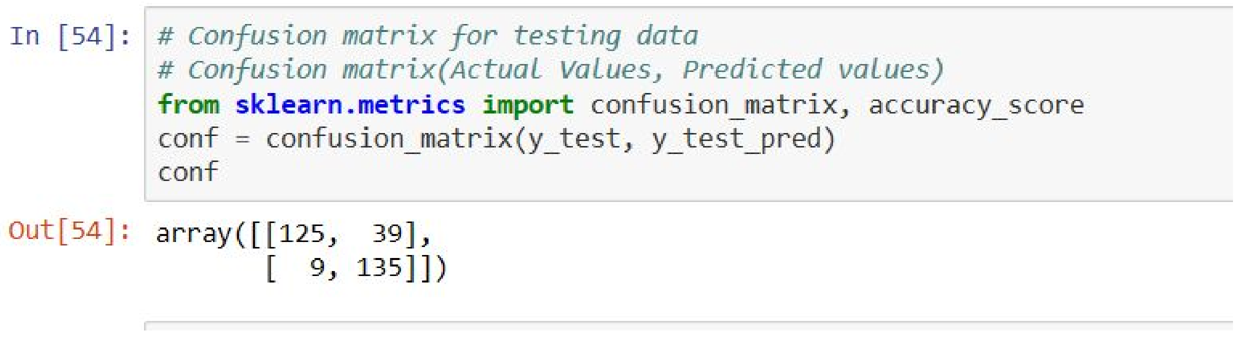


```
In [52]: #prediction on Test data
         # Syntax: objectName.predict(Input)
         y_test_pred = reg.predict(X_test)
         y_test_pred

Out[52]: array([0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
                1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1,
                0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
                0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
                0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
                0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
                1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
                1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
                0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
                0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
                0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0,
                1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0,
                0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1])
```

Figure 24: 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 [54]: # 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[54]: array([[125,  39],
                [  9, 135]])
```

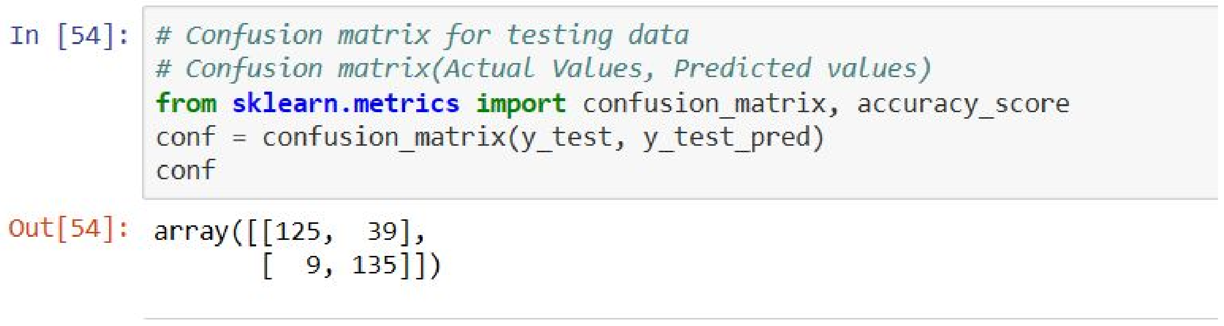
Figure: Logistic Regression algorithm Evaluation using Confusion matrix

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.

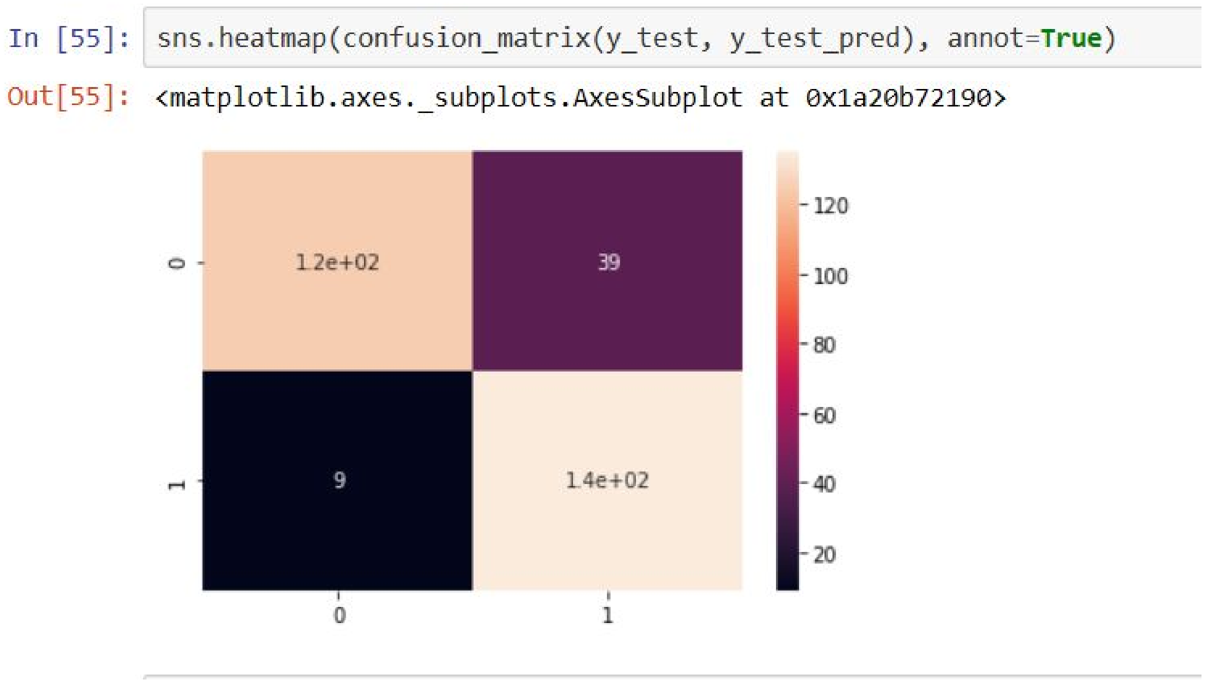
Visualizing Confusion Matrix using Heatmap:

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 [54]: # 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[54]: array([[125,  39],
                [  9, 135]])
```



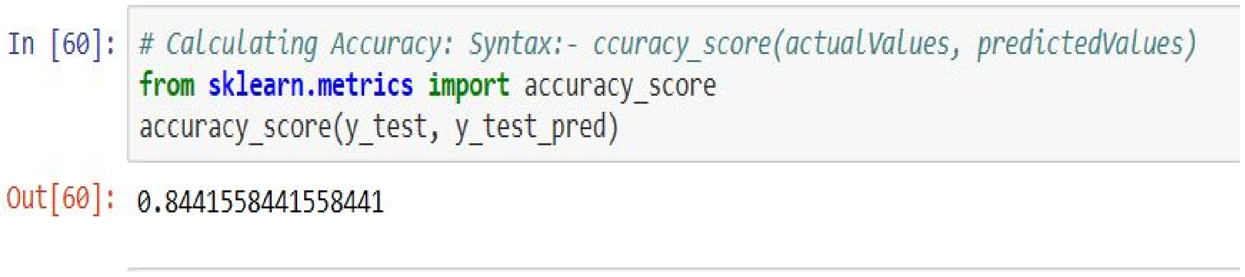
```
In [55]: sns.heatmap(confusion_matrix(y_test, y_test_pred), annot=True)

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x1a20b72190>
```

Figure 25: Logistic Regression algorithm visualising 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 [60]: # Calculating Accuracy: Syntax:- ccuracy_score(actualValues, predictedValues)
         from sklearn.metrics import accuracy_score
         accuracy_score(y_test, y_test_pred)

Out[60]: 0.8441558441558441
```

Figure 26: Accuracy using Logistic Regression algorithm.

6.3 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.

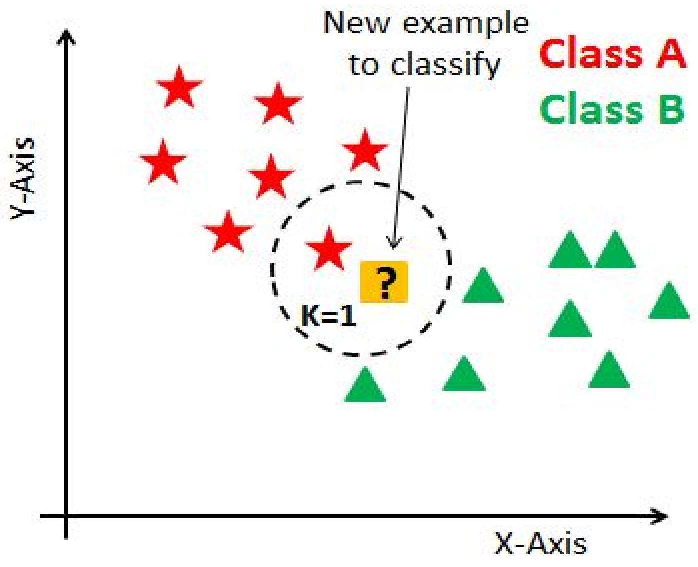


Figure 27:example to classify

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

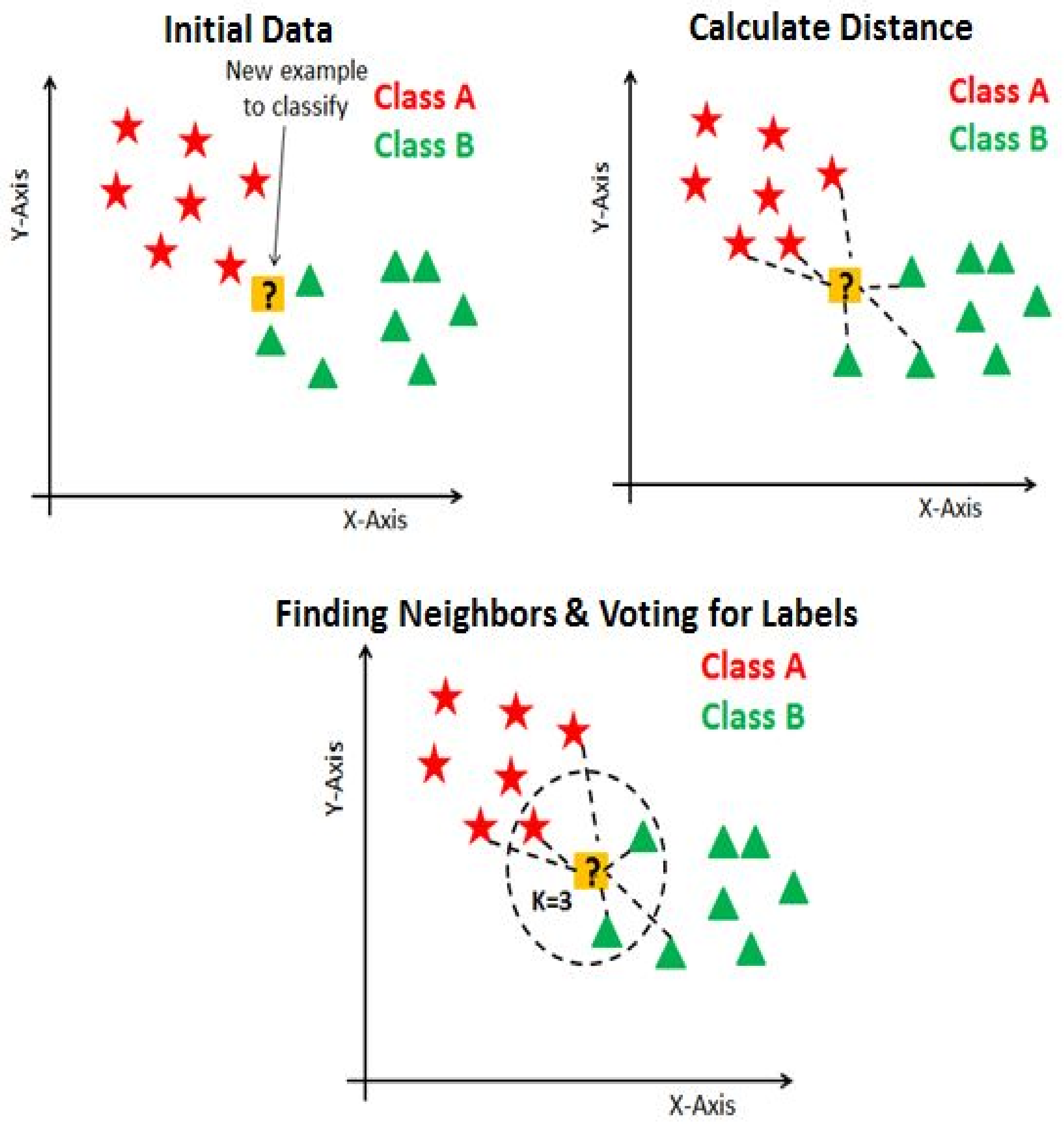
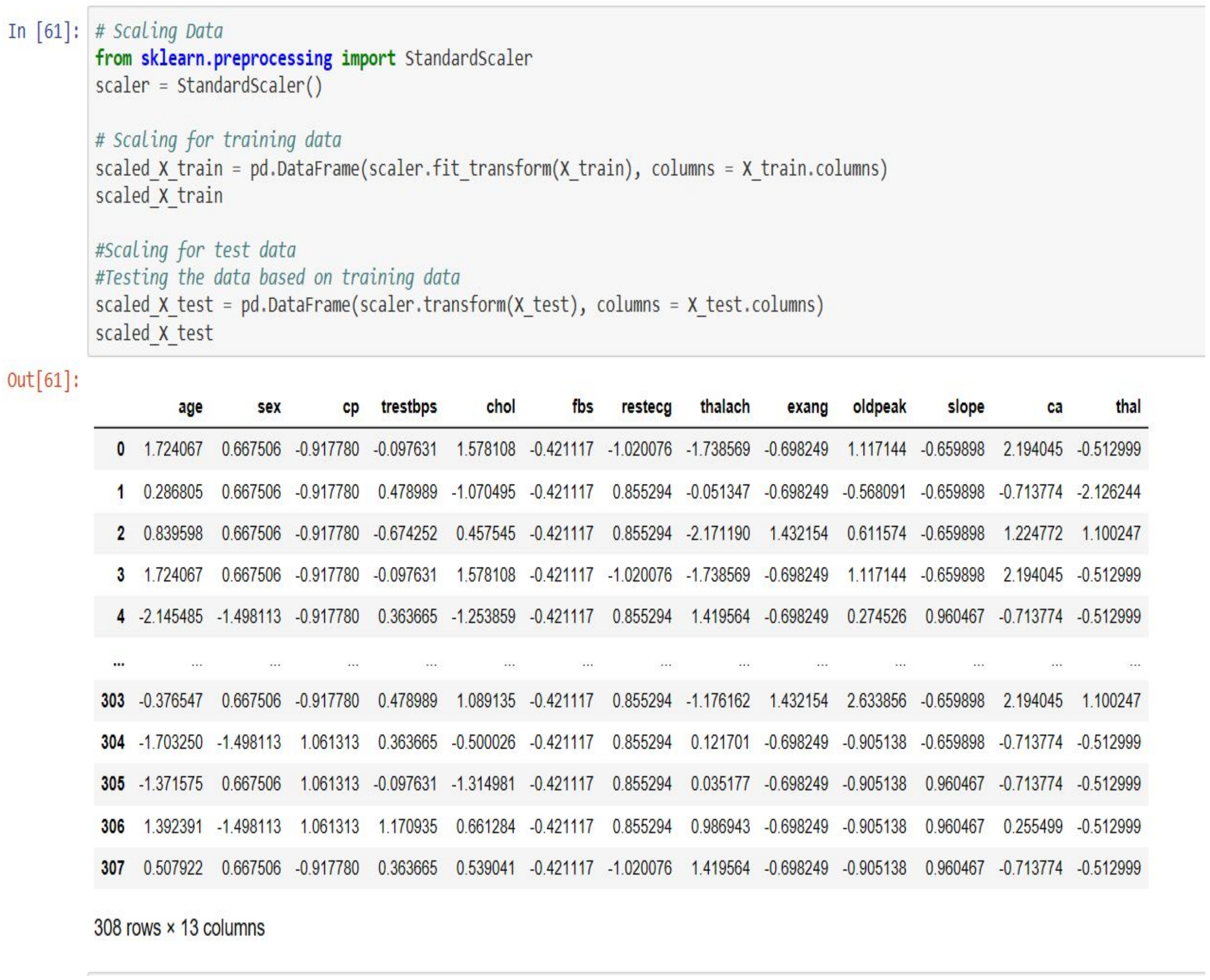


Figure 28:Types of KNN

Model Development and Prediction:

Before splitting if you apply standardization -- you are considering whole you are including test data also into training data

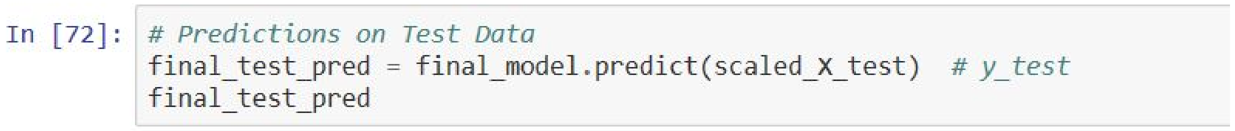


In [61]:
```
# Scaling Data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# 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
```

Out[61]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	1.724067	0.667506	-0.917780	-0.097631	1.578108	-0.421117	-1.020076	-1.738569	-0.698249	1.117144	-0.659898	2.194045	-0.512999
1	0.286805	0.667506	-0.917780	0.478989	-1.070495	-0.421117	0.855294	-0.051347	-0.698249	-0.568091	-0.659898	-0.713774	-2.126244
2	0.839598	0.667506	-0.917780	-0.674252	0.457545	-0.421117	0.855294	-2.171190	1.432154	0.611574	-0.659898	1.224772	1.100247
3	1.724067	0.667506	-0.917780	-0.097631	1.578108	-0.421117	-1.020076	-1.738569	-0.698249	1.117144	-0.659898	2.194045	-0.512999
4	-2.145485	-1.498113	-0.917780	0.363665	-1.253859	-0.421117	0.855294	1.419564	-0.698249	0.274526	0.960467	-0.713774	-0.512999
...	...	...	...	...	...	...	...	...	...	...	...	...	...
303	-0.376547	0.667506	-0.917780	0.478989	1.089135	-0.421117	0.855294	-1.176162	1.432154	2.633856	-0.659898	2.194045	1.100247
304	-1.703250	-1.498113	1.061313	0.363665	-0.500026	-0.421117	0.855294	0.121701	-0.698249	-0.905138	-0.659898	-0.713774	-0.512999
305	-1.371575	0.667506	1.061313	-0.097631	-1.314981	-0.421117	0.855294	0.035177	-0.698249	-0.905138	0.960467	-0.713774	-0.512999
306	1.392391	-1.498113	1.061313	1.170935	0.661284	-0.421117	0.855294	0.986943	-0.698249	-0.905138	0.960467	0.255499	-0.512999
307	0.507922	0.667506	-0.917780	0.363665	0.539041	-0.421117	-1.020076	1.419564	-0.698249	-0.905138	0.960467	-0.713774	-0.512999

308 rows × 13 columns

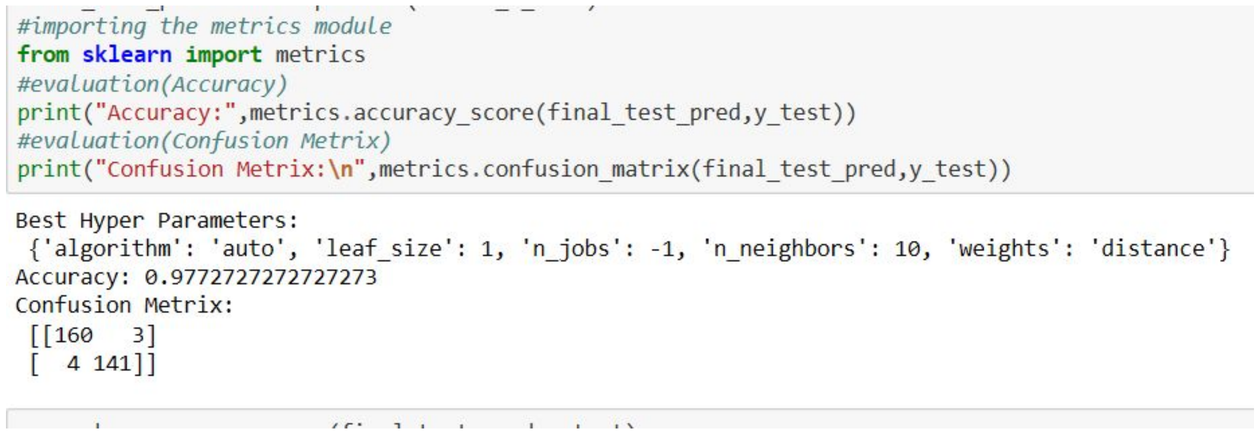


In [72]:
```
# Predictions on Test Data
final_test_pred = final_model.predict(scaled_X_test)  # y_test
final_test_pred
```

Figure 29: 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))
```

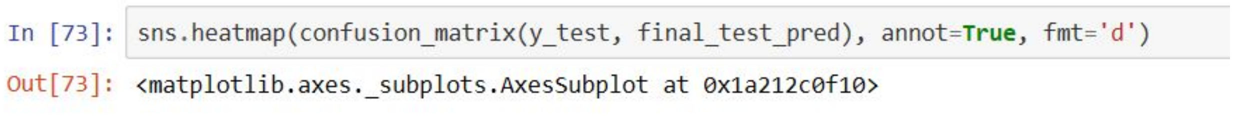
```
Best Hyper Parameters:
 {'algorithm': 'auto', 'leaf_size': 1, 'n_jobs': -1, 'n_neighbors': 10, 'weights': 'distance'}
Accuracy: 0.9772727272727273
Confusion Metrix:
 [[160   3]
 [  4 141]]
```

Figure 30: 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 [73]: sns.heatmap(confusion_matrix(y_test, final_test_pred), annot=True, fmt='d')
Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x1a212c0f10>
```

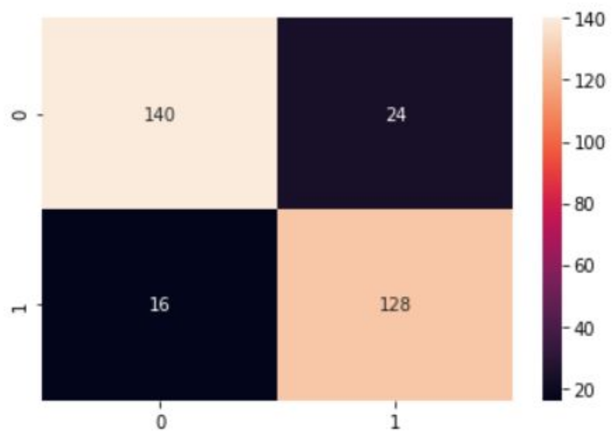
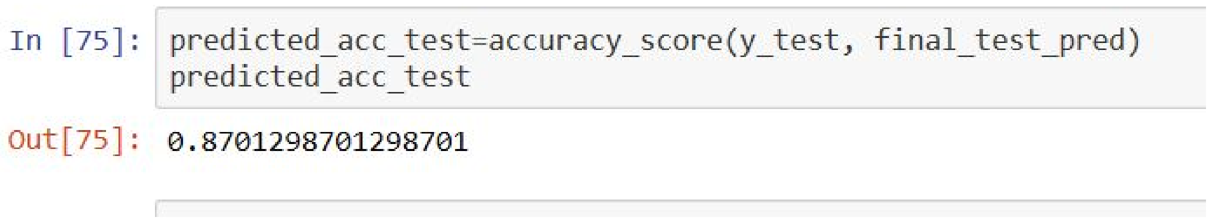


Figure 31: K-Nearest Neighbor algorithm visualizing Confusion matrix using Heatmap.

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 [75]: predicted_acc_test=accuracy_score(y_test, final_test_pred)
         predicted_acc_test
Out[75]: 0.8701298701298701
```

Figure 32: Accuracy using K-Nearest Neighbor algorithm.

7.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

8.REFERENCES:

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

2. https://medium.com/code-heroku/introduction-to-exploratory-data-analysis-eda-c 0257f888676

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