DATA SCIENCE

(Prediction Of Heart Disease Occurance)

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

BACHELOROFTECHNOLOGY

IN

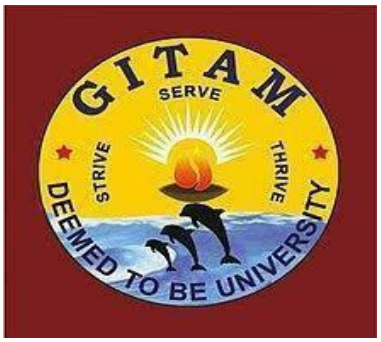
COMPUTER SCIENCE ENGINEERING

submitted by

PERALA LIKHITA

(221710312041)

under the guidance of



DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT(GITAM)

(Declared as Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS MARCH-2020

DECLARATION

I submit this industrial training work entitled "**Prediction of heart disease occurance**" to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "Bachelor of Technology" in "Computer Science Engineering". I declare that it was carried out independently by me under the guidance of _____, _____, 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

Place: HYDERABAD PERALA LIKHITA

Date: 221710312041

INTERNSHIP CERTIFICATE

ACKNOWLEDGEMENT

Our project would not have been successful without the help of several people. we would like to thank the personalities who were part of our project in numerous ways, those who gave us outstanding support from the birth of the project.

We are extremely thankful to our honorable Pro-Vice Chancellor, Prof. N. Siva Prasad for providing necessary infrastructure and resources for the accomplishment of our project.

We are highly indebted to Prof. N. Seetha Ramaiah, Principal, School of Technology, for his support during the tenure of the project.

We are very much obliged to our beloved Prof. S. Phani Kumar, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this project and encouragement in completion of this project.

We hereby wish to express our deep sense of gratitude to _____, _____, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by him for the success of the Internship Project.

We are also thankful to all the staff members of the Computer Science and Engineering department who have cooperated in making our Internship Project a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our internship
Project.

Sincerely

PERALA LIKHITA

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on car Evaluation Dataset. My perception of understanding the given data set has been in the view of undertaking a client's requirement of overcoming the stagnant point of sales of the products being manufactured by client.

To get a better understanding and work on a strategical approach for solution of the client, I have adapted the view point of looking at the dataset features like buying and maintenance cost because these features are the most important features for any kind of customer.

1. INFORMATION ABOUT DATA SCIENCE:

1.1 What is Data Science ……. 10

1.2 Need of Data Science ……. 10

1.3 1.3 Uses of Data Science …….. 10

2. INFORMATION ABOUT MACHINE LEARNING :

2.1 Introduction …….. 11

2.2 Importance of Machine Learning ……. 11

2.3 Uses of Machine Learning …… 12

2.4 Types of Machine Learning …… 12

2.4.1 Supervised Learning

2.4.2 Unsupervised Learning

2.4.3 Sem Supervised Learning

2.5 Relation between Data Mining, Machine Learning and Deep Learning ……..14

3. INFORMATION ABOUT PYTHON :

3.1 Introduction…… 15

3.2 History of Python …….. 15

3.3 Features …… 15

3.4 Setup of Python…….. 15

3.4.1 Installation(Python)

3.4.2 Installation(Anaconda)

3.5 Variable Types ……..16

3.5.1 Numbers

3.5.2 Strings

3.5.3 Lists

3.5.4 Tuples

3.5.5 Dictionaries

3.6 Functions …… 19

3.6.1 Defining a Function

3.6.2 Calling a Function

3.7 OOPs Concepts …… 20

3.7.1 Class

3.7.2 __init__ method in class

4. Prediction Of Heart Disease Occurance

4.1 Project Requirements ……21

4.1.1 Packages used

4.1.2 Versions of the packages

4.1.3 Algorithms used

4.2 Problem Statement ……..21

4.3 Data set Description ……21

4.4 Objective of the Case Study ……21

5. DATA PREPROCESSING/FEATURE ENGINEERING AND EDA

5.1 Prepossessing of Data……..22

5.1.1 Getting the Data set

5.1.2 Importing Libraries

5.1.3 Importing the Data set

5.1.4 Handing Missing values

5.2 Generating Plots…… 23

5.2.1 Visualize the data between all the Features

5.2.2 Visualize the data between Target and the Features

6. FEATURE SELECTION:

6.1 Select relevant features for the analysis ……. 25

6.2 Train-Test-Split ……..25

7. MODEL BUILDING AND EVALUATION:

7.1 Logistic Regression……..28

7.2 K Nearest Neighbor Classifier……..30

7.3 Naïve Bayes……..34

8. Conclusion ……. 37

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 and 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

Apart from the applications mentioned above, data science is also used in Marketing,Finance, Human Resources, Health Care, Government,Augmented Reality,Robots, Self Driving Cars

2. Information About Machine Learning 2.1 Introduction:

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

2.2 Importance of Machine Learning:

Consider some of the instances where machine learning is applied: the self-driving Google car, 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 techniques.

The process flow depicted here represents how machine learning works.

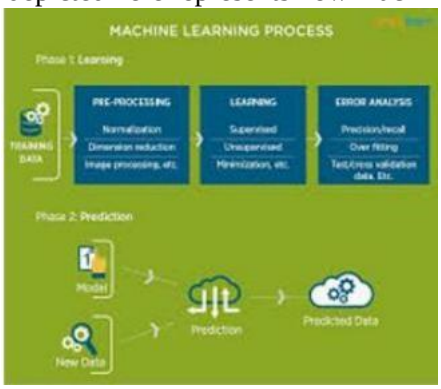


Fig 2.2.1 : The Process Flow

2.3 Uses of Machine Learning

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always 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 Algorithms

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

2.4.1 Supervised Learning :

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

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labeled training data set – that is, a data set 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 multi class classification.

2.4.2 Unsupervised Learning :

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

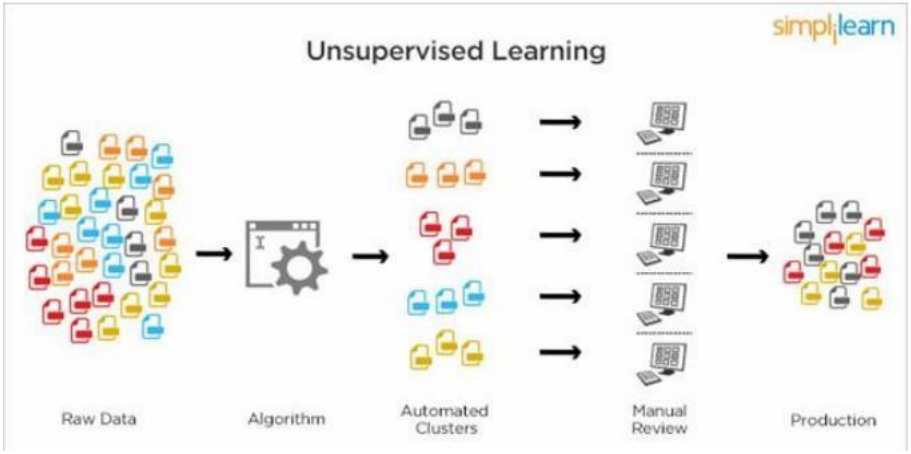


Fig 2.4.2.1: Unsupervised Learning

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.

2.4.3 Semi Supervised Learning:

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

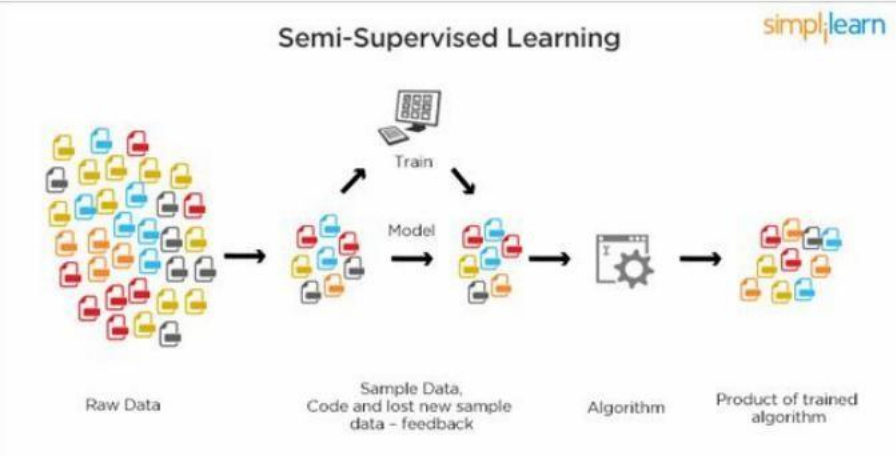


Fig 2.4.3.1: Semi Supervised Learning

2.5 Relation between Data Mining, Machine Learning and Deep Learning

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining 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.

3. INFORMATION ABOUT PYTHON

Basic programming language used for machine learning is : PYTHON

3.1 Introduction

- 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.

3.2 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

3.3 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.

3.4 Python Setup

- 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.

3.4.1 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.

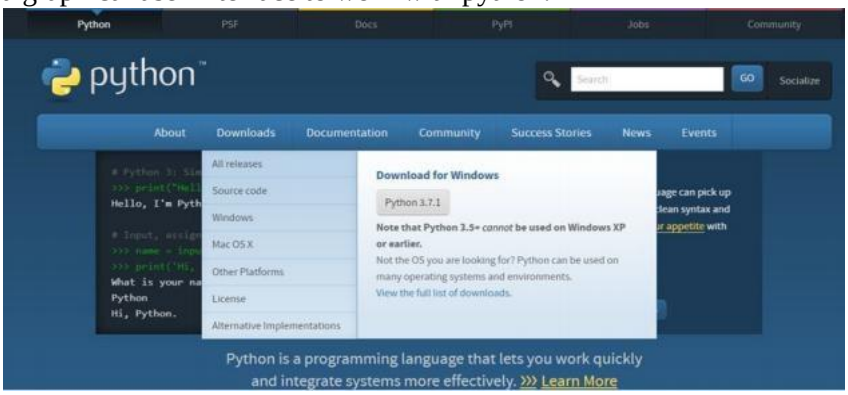


Fig 3.4.1.1: Python download

3.4.2 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)

Fig 3.4.2.1: Anaconda download

3.5 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 five standard data types –
- Numbers
- Strings
- Lists
- Tuples
- Dictionary

3.5.1 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).

3.5.2 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.

3.5.3 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.

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 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 Python Dictionary:

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

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

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

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

3.6 Python Functions

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.

3.6.2 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 Oops 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:
  - 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
```

Fig 3.6.2.1: Defining a Class

3.7.2 __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__().

4. Project Name : PREDICTION OF HEART DISEASES OCCURANCE

4.1 Project Requirements

4.1.1 Packages used.

- ✓ Pandas
- ✓ Matplotlib
- ✓ Numpy
- ✓ Sea born
- ✓ Scikit-learn

4.1.2 Versions of the packages.

- ✓ Pandas : 0.25.1
- ✓ Matplotlib : 3.1.1
- ✓ Numpy : 1.16.5
- ✓ Seaborn:0.10.1 ✓ Scikit-learn:0.23.1

4.1.3 Algorithms used.

- ✓ Logistic Regression
- ✓ K Neighbours classifier
- ✓ Naïve Bayes

4.2 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.

4.3 Data set Description

1.Age (In years)

2.Sex 1 - Male 0 - Female

3.CP (Chest Pain Type) 0 - Typical Angina (Heart related) 1 - Atypical Angina (Non-heart related) 2 - Non-Anginal pain (Non-heart related) 3 - Asymptomatic (No disease)

4.TRESTBPS (Resting Blood Pressure (in mm Hg on admission to the hospital))

5.CHOL (Serum Cholestoral in mg/dl) Healthy serum cholesterol is less than 200 mg/dL

6.FPS (Fasting blood sugar > 120 mg/dl) 1 - True 0 - False

7.RESTECH (Resting Electro Cardio Graphic results)

8.THALACH (Maximum heart rate achieved)

9.EXANG (Exercise induced Angina) 1 - Yes 0 - No

10.OLDPEAK (ST depression induced by exercise relative to rest)

11.SLOPE (Slope of the peak exercise ST segment) 12.CA (Number of major vessels (0-3) colored by Flouroscopy)

13.THAL 0 - Normal 1 - Fixed defect 2 - Reversible defect

14.TARGET 1 - Heart Problem 0 - No Heart Problem
Note: Source (Kaggle)

4.4 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.

5. Data Prepossessing/Feature Engineering and EDA

5.1 Preprocessing of The Data

Prepossessing 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.

```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plts
```

Fig 5.1.2.1: 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 data set from a comma separated values file and we can assign it to a Data Frame 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 data frame.

```
data = pd.read_csv("HeartDisease.csv")
data.head()
```

	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

Fig 5.1.3.1: Reading the data set

5.1.4 Handling Missing Values:

In my Data set there are no missing values.

Missing values can be handled in many ways using some inbuilt methods.

5.2 Generating Plots

5.2.1 Visualize the data between all the Features

Using inline matplotlib I generated the graphs (Histograms) to get count of each label inside categorical feature:

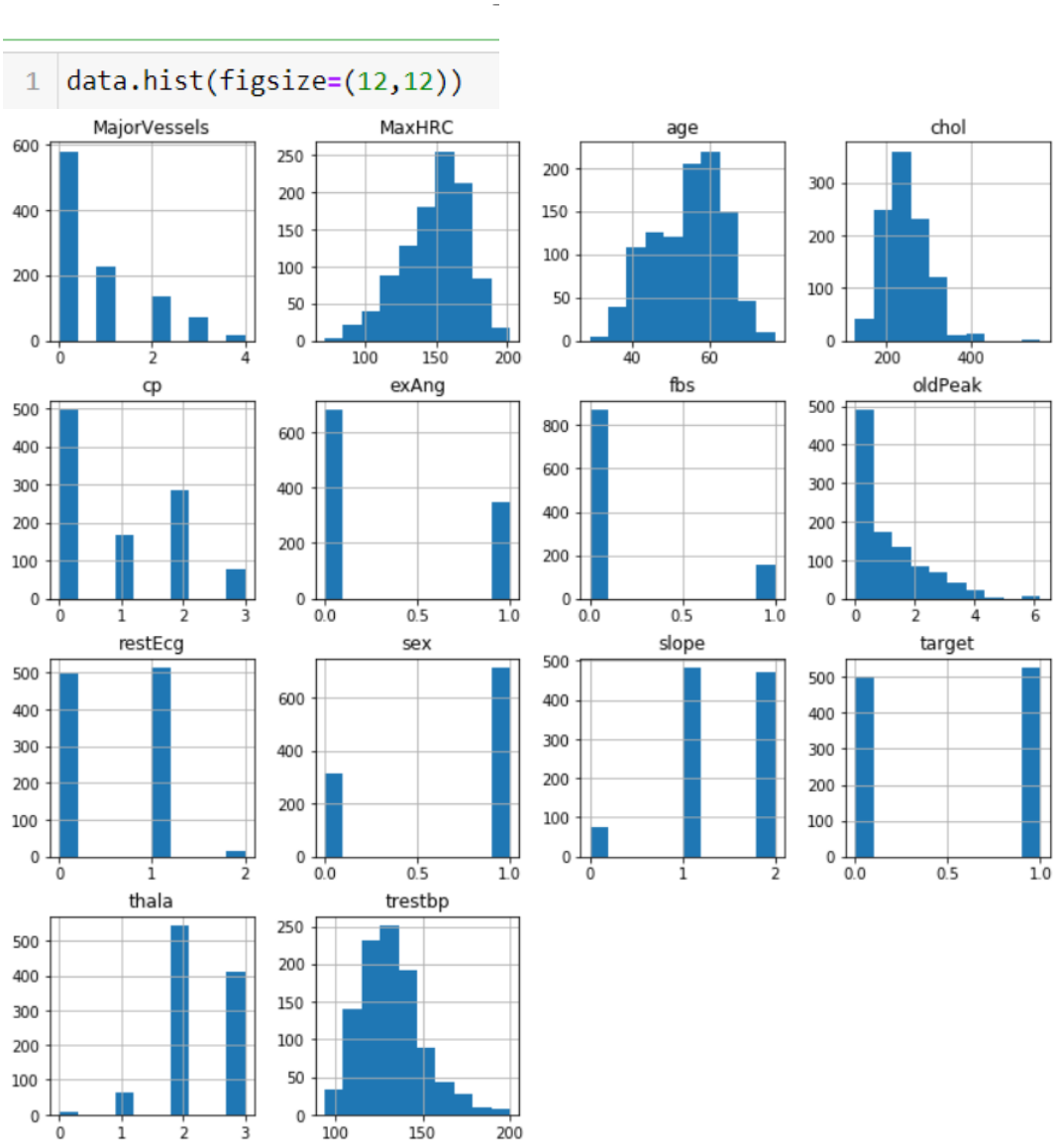


Fig: 5.2.1.1: visualization of data in between all the features

5.2.2 Visualize the data between Target and the Features

Now for visualization of data against the target value I used seaborn

- The below figure is about heart disease frequency for age

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

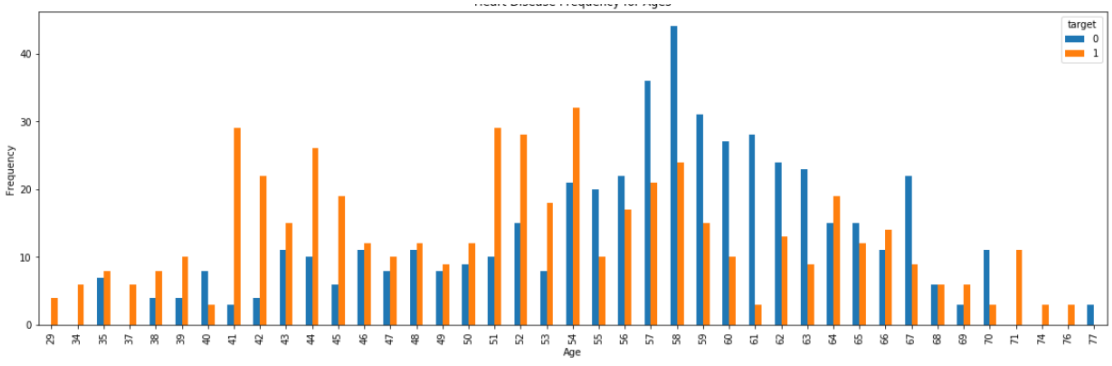


Fig 5.2.2.1

- The below figure is about heart disease frequency for male and female

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

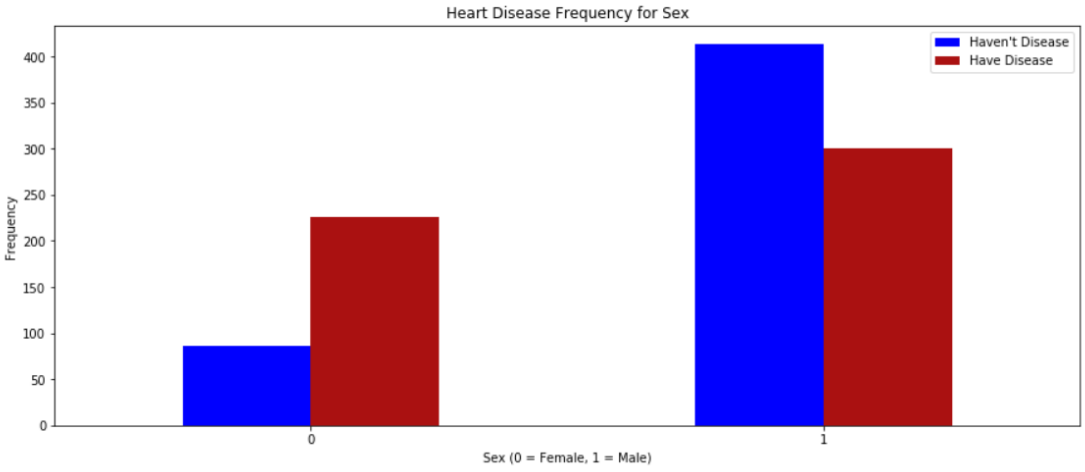


Fig 5.2.2.2

- The below figure is about Thalassemia vs Cholesterol

```
plt.figure(figsize=(8,6))
sns.scatterplot(x='chol',y='thala',data=data,hue='target')
plt.show()
```

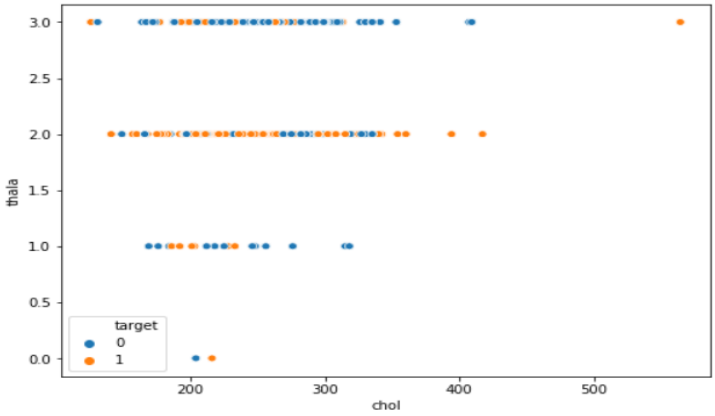


Fig 5.2.2.3

- The below figure is about Thalassemia vs Resting blood Pressure

```
plt.figure(figsize=(8,6))
sns.scatterplot(x='thala',y='trestbp',data=data,hue='target')
plt.show()
```

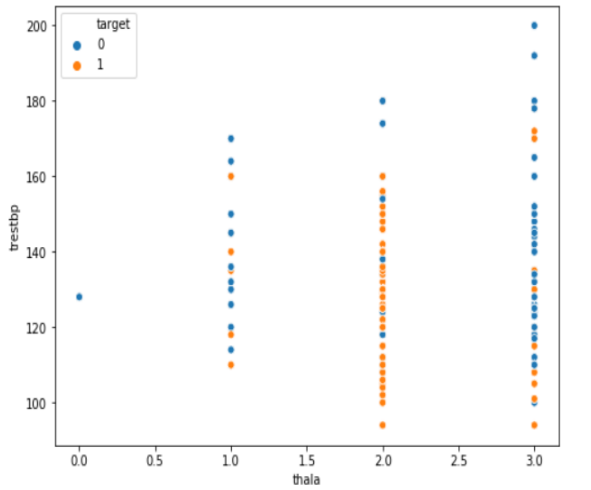


Fig 5.2.2.4

- The below figure is about age vs maximum heart disease rate

```
plt.scatter(x=data.age[data.target==1], y=data.thala[(data.target==1)], c="green")
plt.scatter(x=data.age[data.target==0], y=data.thala[(data.target==0)])
plt.legend(["Disease", "Not Disease"])
plt.xlabel("Age")
plt.ylabel("Maximum Heart Rate")
plt.show()
```

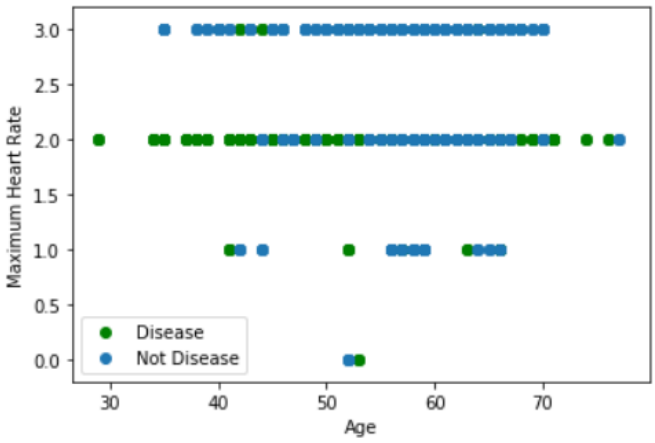


Fig 5.2.2.5

- The below figure is about Fasting blood sugar data

```
pd.crosstab(data.fbs,data.target).plot(kind="bar",figsize=(20,10),color=['#4286f4','#f49242'])
plt.title("Heart disease according to FBS")
plt.xlabel('FBS- (Fasting Blood Sugar > 120 mg/dl) (1 = false; 0 = true)')
plt.xticks(rotation=90)
plt.legend(["Haven't Disease", "Have Disease"])
plt.ylabel('Disease or not')
plt.show()
```

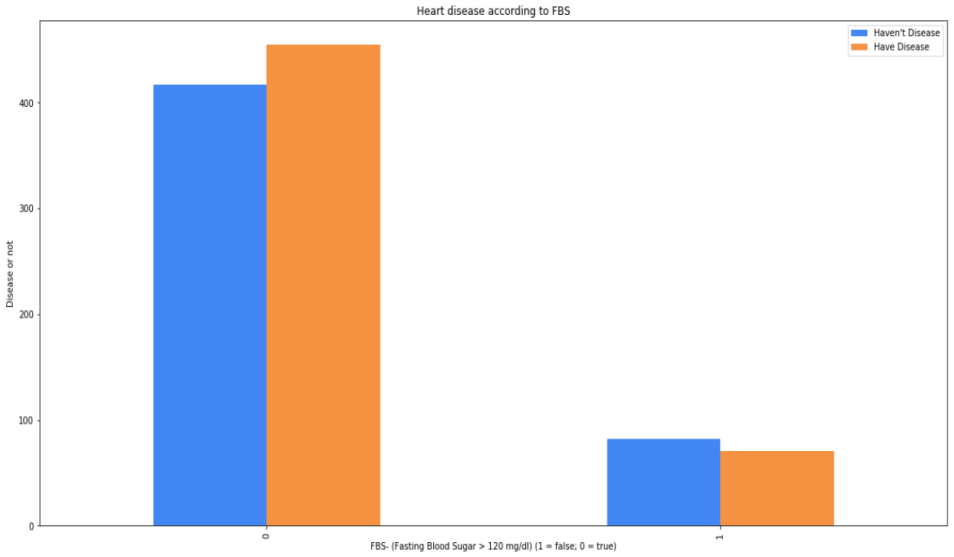


Fig 5.2.2.6

6. Feature Selection

6.1 Select relevant features for the analysis

Splitting the data into independent and Dependent features in my scenario independent values are all features other than the Target value

```
target = ['value']
reject = target
features = [x for x in data_os.columns if x not in reject]
x = data_os[features]
y = data_os[target]
```

Fig 6.1.1: splitting data

6.2 Train and Test Split:

Splitting data into Train and Test

```
# Preparing Input and Output
# Drop the id and diagnosis columns
X = data.drop(['target'], axis=1)
X.head()
```

	age	sex	cp	trestbp	chol	fbs	restEcg	MaxHRC	exAng	oldPeak	slope	MajorVessels	thala
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2

Fig 6.2.1: data

```
# Preparing Training and Testing Data
# Storing 70% of the data(569 rows) 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=10)
```

```
print(X_train.shape)
```

```
(717, 13)
```

```
print(X_test.shape)
```

```
(308, 13)
```

```
print(y_train.shape)
```

```
(717,)
```

```
print(y_test.shape)
```

```
(308,)
```

```
X_train
```

	age	sex	cp	trestbp	chol	fbs	restEcg	MaxHRC	exAng	oldPeak	slope	MajorVessels	thala
750	55	1	1	130	262	0	1	155	0	0.0	2	0	2
807	44	1	2	130	233	0	1	179	1	0.4	2	0	2
687	58	1	0	125	300	0	0	171	0	0.0	2	2	3
651	41	1	1	120	157	0	1	182	0	0.0	2	0	2
826	42	1	2	130	180	0	1	150	0	0.0	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
156	40	1	3	140	199	0	1	178	1	1.4	2	0	3
123	65	0	2	140	417	1	0	157	0	0.8	2	1	2
369	51	1	2	110	175	0	1	123	0	0.6	2	0	2
320	53	0	0	130	264	0	0	143	0	0.4	1	0	2
527	62	0	0	124	209	0	1	163	0	0.0	2	0	2

717 rows × 13 columns

Fig.6.2.3 Train and test split data

7.MODEL BUILDING AND EVALUATION

I used three Algorithm.

7.1 Logistic Regression:

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().

```
# 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
```

```
C:\Users\Likhita\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:940: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
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)
```

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

Fig.7.1.1 : 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.

```
# Confusion matrix for training data
# Confusion matrix(Actual Values, Predicted values)
from sklearn.metrics import confusion_matrix, accuracy_score
conf = confusion_matrix(y_train, y_train_pred)
conf
```

```
array([[279,  67],
       [ 30, 341]], dtype=int64)
```

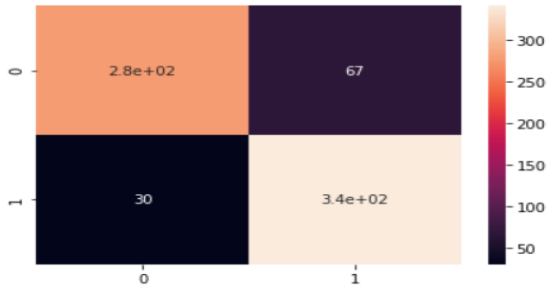
Fig 7.1.2: Logistic Regression algorithm Evaluation using Confusion matrix

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.

```
sns.heatmap(confusion_matrix(y_train, y_train_pred), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b4b7042188>
```




```
# supported values for ha are : 'center', 'right', 'left'
# supported values for va are : 'top', 'bottom', 'center', 'baseline', 'center_baseline
# fmt can also be given as d(integer formatting)
sns.heatmap(confusion_matrix(y_train, y_train_pred), annot=True, fmt='3.0f', annot_kws={'size':'10', "ha": 'right',"va": 'baseli
ne'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b4b70dde08>
```

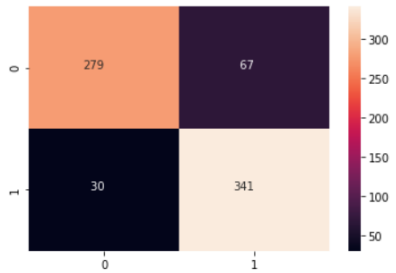


Fig 7.1.3 Logistic Regression algorithm visualizing Confusion matrix using Heatmap

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.

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

```
0.8668831168831169
```

Fig7.1.4: Accuracy using Logistic Regression algorithm

7.2 K NEAREST NEIGHBORS CLASSIFIER:

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.

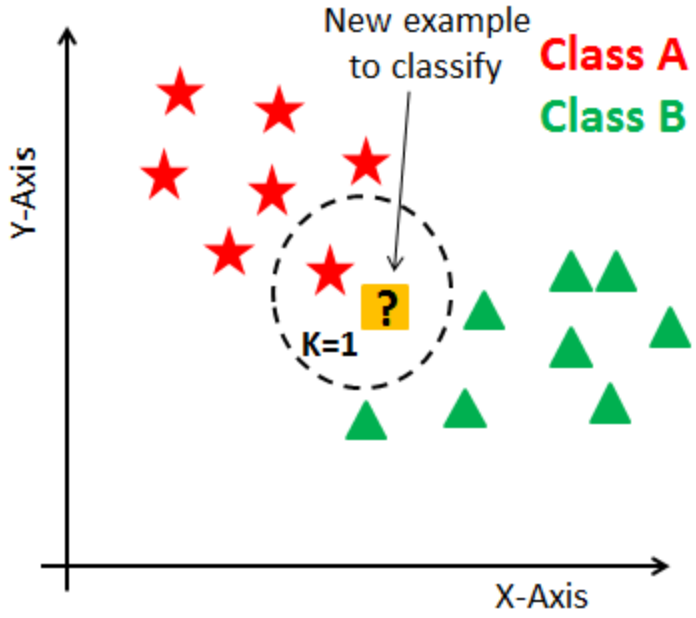


Fig 7.2.1

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

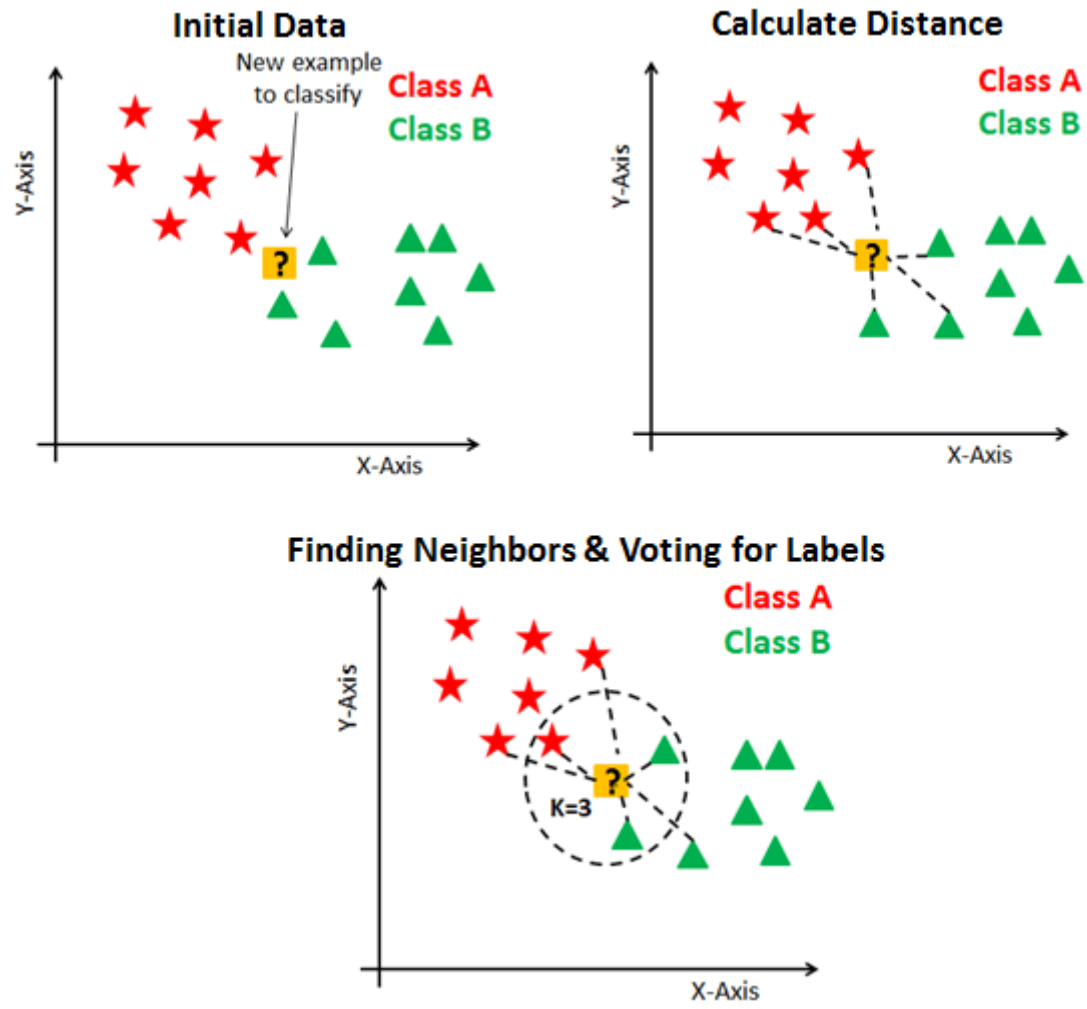


Fig 7.2.2

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
```

	age	sex	cp	trestbp	chol	fbs	restEcg	MaxHRC	exAng	oldPeak	slope	MajorVessels	thala
0	0.274315	0.676252	-0.922581	1.149447	0.531841	-0.441588	0.879214	-2.642109	1.387995	0.120332	-0.610776	0.243072	1.106221
1	1.045625	0.676252	1.982940	-1.257175	-0.680257	-0.441588	-1.002566	-0.209837	1.387995	0.627351	-0.610776	-0.725166	-0.509177
2	1.706747	0.676252	-0.922581	-0.111165	1.455343	-0.441588	-1.002566	-1.730007	-0.720464	1.134370	-0.610776	2.179549	-0.509177
3	-0.607181	0.676252	0.045926	-0.111165	0.377923	-0.441588	0.879214	0.962865	-0.720464	-0.386687	1.005188	-0.725166	-0.509177
4	0.604876	0.676252	-0.922581	-0.111165	0.127808	-0.441588	0.879214	-0.209837	1.387995	0.289338	1.005188	0.243072	1.106221
...	...	...	...	...	...	...	...	...	...	...	...	...	...
303	-1.158117	-1.478738	1.014433	-1.371776	-2.027032	-0.441588	0.879214	1.136599	-0.720464	-0.386687	-0.610776	-0.725166	-0.509177
304	-1.819239	0.676252	1.014433	0.347240	-1.372884	-0.441588	0.879214	1.049732	-0.720464	-0.893707	1.005188	3.147788	-0.509177
305	1.265999	0.676252	-0.922581	1.607851	-0.353183	-0.441588	-1.002566	-0.470438	-0.720464	1.049866	1.005188	-0.725166	-2.124575
306	-0.056246	0.676252	1.014433	-0.684170	0.224006	-0.441588	-1.002566	-0.079537	-0.720464	-0.555694	-0.610776	-0.725166	1.106221
307	1.376186	0.676252	-0.922581	-0.397667	0.147048	2.264554	0.879214	0.615398	-0.720464	-0.724700	-0.610776	1.211311	1.106221

308 rows × 13 columns

Fig 7.2.4 : 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.

```
final_test_pred=model1.predict(scaled_x_test)
#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))
```

Fig 7.2.5 : 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>
```

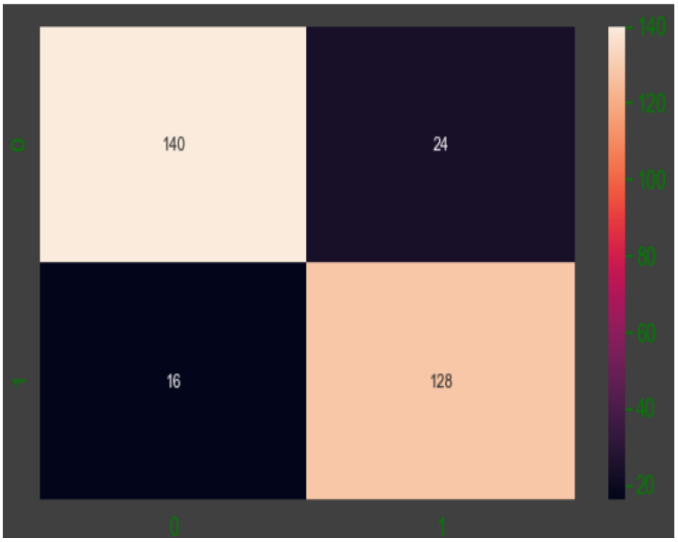


Fig 7.2.6: K-Nearest Neighbor algorithm visualizing Confusion matrix using Heatmap

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.

```
predicted_acc_test=accuracy_score(y_train, final_train_pred)
predicted_acc_test
```

```
0.8814504881450488
```

Fig7.2: Accuracy using K-earest Neighbor algorithm

7.3 NAÏVE BAYES:

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.

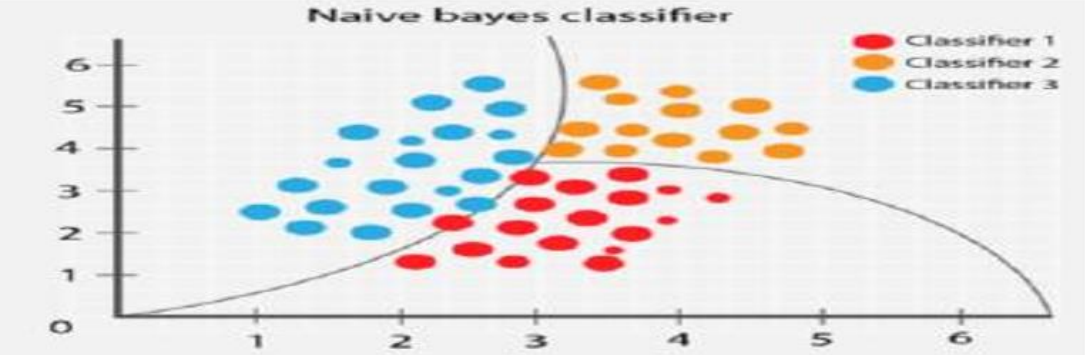


Fig7.3.1: Naive bayes classifier

Model Development and Prediction:

```
#Apply the Naive Bayes Algorithm
#Import BernNB
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)
```

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

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

Fig 7.3.2: 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)
```

Fig 7.3.3: 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)
```

```
0.827922077922078
```

Fig 7.3.4: Accuarcy using Naive bayes classifier

8. 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.