



Karunadu Technologies Private Limited

#17, ATK complex, 4th Floor, Acharya college main road, beside Karur Vysya bank,
Guttebasaveshwaranagar Chikkabanvara, Bengaluru, Karnataka-560090

Internship Report

On

Python With Machine Learning

By

**SHANMUKHA SHIVU AU
(1KI19CS097)**

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE

KALPATARU INSTITUTE OF TECHNOLOGY



List of Contents

Topic	Pg. No
List of Figures	3
Chapter 1: Company Profile	4
1.1: Profile	4
1.1.1: Vision	4
1.1.2: Mission	4
1.1.3: Objectives	5
1.2: Company Products and Services Offered	5
1.2.1: Products	5
1.2.2: Services	6
1.3: Contact Details	7
Chapter 2: Topics Learned During The Course	8
2.1: Python	8
2.1.1 Features of Python	8
2.2: Python libraries for Data Analytics	8
2.2.1: Numpy	9
2.2.2: Pandas	9
2.2.3: Matplotlib	10
2.3: Software Tool Used	11
2.4: Open CV	12
2.4.1: Topics Covered	13
2.5: Machine Learning	13
2.5.1: The Goals of Machine Learning	18
2.5.2: Types of Machine Learning	19
2.5.3: Applications of Python with Machine Learning	19
	22

Chapter 3: Machine Learning Algorithm	23
3.1: Linear Regression	23
3.1.1: Obtaining best fit line (Value of a and b)	24
3.1.2: Advantages and Disadvantages	25
3.2: Multiple Linear Regression	27
3.2.1: Advantages and Disadvantages	27
3.3: Logistic Regression	29
3.3.1: Key Features	29
3.3.2: Advantages and Disadvantages	30
3.4: KNN Regression	32
3.4.1: Features Of KNN Algorithm	32
3.4.2: Working	32
3.4.3: Advantages and Disadvantages	34
3.5: SVM	36
3.5.1: Features Of SVM	36
3.5.2: Working	36
3.5.3: Advantages and Disadvantages	37
3.6: Decision Tree	39
3.6.1: Features Of Decision Tree	39
3.6.2: Working	39
3.6.3: Advantages nd Disadvantages	40
 Chapter 4: Prediction Of Car Price	 43
4.1: Problem Statement	43
4.2: Dataset	43
4.3: Algorithm – Multiple Linear Regression	43
4.4: Programming Steps	44
4.5: Conclusion	48
 Chapter 5: Prediction of Fractured Or Not	 49
5.1: Problem Statement	49
5.2: Dataset	50
5.3: Algorithm – SVM	50
5.4: Programming Steps	50
5.5: Conclusion	54

Chapter 6: Prediction of Loan	
6.1: Problem Statement	55
6.2: Dataset	55
6.3: Algorithm-KNN	55
6.4: Programming Steps	58
6.5: Conclusion	
Chapter 7: Prediction of Hired Or Not	
7.1: Problem Statement	59
7.2: Dataset	59
7.3: Algorithm – Logistic Regression	60
7.4: Programming Steps	64
7.5: Conclusion	
References	65

List Of Figures

Sl. No	Fig. No	Name of the figure	Page No
1.	1.1	Company Logo	4
2.	2.1	Python Logo	8
3.	2.2	Pandas Data Frame components example	11
4.	2.3	Example of Gray and HSV Image	14
5.	2.4	Example of Resized Image	15
6.	2.5	Example of Added and Transparency Image	16
7.	2.6	Example of Rotated Image	17
8.	2.7	Example of Face and Eye Detection	18
9.	2.8	Machine Learning Infrastructure	18
10.	2.9	Flow Model of Algorithms of Supervised Learning	20
11.	2.10	Clustering and Regression	21
12.	2.11	Flow Model of Algorithms of Unsupervised Learning	21
13.	2.12	Flow Model of Algorithms of Reinforcement Learning	22
14.	3.1	Types of Machine Learning Algorithms	23
15.	3.2	Linear Regression	24
16.	3.3	Plot of Input Graph	24
17.	3.4	Example of Linear Regression	26
18.	3.5	Example of Multiple Linear Regression	28
19.	3.6	Linear Regression vs Logistic Regression	30
20.	3.7	Example of Logistic Regression	31
21.	3.8	Plots of ideal KNN Algorithm	33
22.	3.9	Example of KNN Algorithm	35
23.	3.10	Plot of ideal SVM Algorithm	37
24.	3.11	Example of SVM Algorithm	38
25.	3.12	Ideal diagram of a Decision Tree	40
26.	3.13	Example of Decision Tree	41
27.	4.1	Overview of Dataset	43
28.	4.2	Output of Yearly Amount Spent	45
29.	4.3	Implementing Yearly Amount Spent Prediction in PYQT5	46
30.	4.4	PYQT5 Output of Yearly Amount Spent Prediction	47
31.	5.1	Overview of Dataset	49
32.	5.2	Output of Car Name Prediction	51
33.	5.3	Implementing Car Name Prediction in PYQT5	52
34.	5.4	PYQT5 Output of Car Name Prediction	54

CHAPTER 1

Company Profile

It is pleasure in introducing “Karunadu Technologies Private Limited” as a leading IT software solutions and services industry focusing on quality standards and customer values. It is also a leading Skills and Talent Development company that is building a manpower pool for global industry requirements.

1.1 Profile



Fig 1.1 Company Logo

The company offers broad range of customized software applications powered by concrete technology and industry expertise. It also offers end to end embedded solutions and services. They deal with broad range of product development along with customized features ensuring at most customer satisfaction and also empower individual with knowledge, skills and competencies that assist them to escalate as integrated individuals with a sense of commitment and dedication.

1. Vision

To Empower Unskilled Individual with knowledge, skills and technical competencies in the field of Information Technology and Embedded engineering which assist them to escalate as integrated individuals contributing to company's and Nation's growth.

2. Mission

- Provide cost effective and reliable solutions to customers across various latest technologies.
- Offer scalable end-to-end application development and management solutions

- Provide cost effective highly scalable products for varied verticals.
- Focus on creating sustainable value growth through innovative solutions and unique partnerships.
- Create, design and deliver business solutions with high value and innovation by leveraging technology expertise and innovative business models to address long-term business objectives.
- Keep our products and services updated with the latest innovations in the respective requirement and technology.

3. Objectives

- To develop software and Embedded solutions and services focussing on quality standards and customer values.
- Offer end to end embedded solutions which ensure the best customer satisfaction.
- To build Skilled and Talented manpower pool for global industry requirements.
- To develop software and embedded products which are globally recognized.
- To become a global leader in Offering Scalable and cost-effective Software solutions and services across various domains like E-commerce, Banking, Finance, Healthcare and much more.
- To generate employment for skilled and highly talented youth of our Country INDIA.

3. Company Products and Services Offered

1. Products

- **KECMS – Karunadu Enterprise Content Management System**

Karunadu Enterprise Content Management System is a one stop solution for all our enterprise content management System relating to digital asset management, document imaging, workflow systems and records management systems. Increasing digitalization has led to an exponential growth in business content and managing this sea of unstructured data is tedious work. KECMS enables you to create, capture, manage, distribute, archive different forms of content and has much more features.

- **KEMS – Karunadu Education Management System**

Manage diversified data relating to education management on cloud. Educational data including students and staff is gathered over years which contain information from admission/appointment until leaving the Education. Statistical reports for the College/school can be generated along with admission Tracking and result analysis to keep track of progressive improvements of both student and staff.

- **KASS – Karunadu Advanced Security System**

A Complete one stop embedded solution for large apartments. Security system which monitors door breakage, window breakage, gas leakage, motion detection and various other features which can be operated and maintained by centralized monitored system. This Embedded solution enhances the security measures of apartment/building and enhances the security of individuals may be from unintended intervention or from unauthorized access.

2. Services

- **IT Solutions and Services**

Karunadu Technologies is a Bangalore based IT Training and Software Development center with an exclusive expertise in the area of IT Services and Solutions. Karunadu Technologies Pvt. Ltd. is also expertise in Web Designing and Consulting Services.

- **Embedded Design and Development**

Karunadu Technologies Pvt. Ltd. has expertise in Design and development of embedded products and offers solutions and services in field of Electronics.

- **Academic Projects**

Karunadu Technologies Pvt. Ltd. helps students in their academics by imparting industrial experience into projects to strive excellence of students. Karunadu Technologies Pvt. Ltd. encourages students to implement their own ideas to projects keeping in mind "A small seed sown upfront will be nourished to become a large tree one day", thereby focusing the future entrepreneurs. They have a wide range of IEEE projects for B.E, MTech, MCA, BCA, DIPLOMA students for all branches in each and every domain.

- **Implant Training**

Karunadu Technologies Pvt. Ltd. provides Implant training for students according to the interest of students keeping in mind the current technology and academic benefit one obtains after completing the training. Students will be nourished and will be trained throughout with practical experience. Students will be exposed to industrial standards which boost their carrier. Students will become Acquaint to various structural partitions such as labs, workshops, assembly units, stores, and administrative unit and machinery units. They help students to understand their functions, applications and maintenance. Students will be trained from initial stage that is from collection of Project Requirements, Project Planning, Designing, implementation, testing, deployment and maintenance there by helping to understand the business model of the industry. Entire project life cycle will be demonstrated with hands on experience. Students will also be trained about management skills and team building activities. They assure that by end of implant training students will Enhance communication skills and acquire technical skills, employability skills, start-up skills, and will be aware of risks in industry, management skills and many other skills which are helpful to professional engagement.

- **Software Courses**

Karunadu Technologies Pvt. Ltd. provides courses for students according to the interest of students keeping in mind the current technology and assist them for their further Employment. Company provides various courses such as C, C++, VB, DBMS, Dot Net, Core Java and J2EE along with live projects.

1.3 Contact Details



#17, ATK complex, 4th Floor, Acharya College Main Road, Beside KarurVysya Bank, Guttebasaveshwaranagar, Chikkabanvara, Bengaluru, Karnataka- 560090



support@karunadutechnologies.com

CHAPTER 2

Topics Learnt During the Course

The objective of the internship is to apply theoretical knowledge of “Machine Learning using Python” to solve real time complex problems, in order to achieve these following basic concepts were learnt:

- Python
- Machine Learning

2.1 Python

Python is a multiparadigm, general-purpose, interpreted, high-level programming language. Python allows programmers to use different programming styles to create simple or complex programs, get quicker results and write code almost as if speaking in a human language.



Fig 2.1 Python Logo

The topics learnt on Python are as follows:

- Installation of Python.
- Use of variables to store, retrieve and calculate information.
- Utilization of core programming operations such as functions and loops.
- Operation on strings, python supported libraries for Machine Learning.

1. Features of Python

- Extensive support libraries (NumPy for numerical calculations, Pandas for data analytics).
- Open source and community development.

- Dynamically typed language (No need to mention data type based on value assigned, it takes data type).
- Object-oriented language, Portable and Interactive across Operating systems.

2. Python libraries for Data Analytics

Machine Learning, as the name suggests, is the science of programming a computer by which they are able to learn from different kinds of data. A more general definition given by Arthur Samuel is – “Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.” They are typically used to solve various types of life problems.

In the older days, people used to perform Machine Learning tasks by manually coding all the algorithms and mathematical and statistical formula. This made the process time consuming, tedious and inefficient. But in the modern days, it is become very much easy and efficient compared to the olden days by various python libraries, frameworks, and modules. Python libraries that used in Machine Learning are:

- NumPy
- Pandas
- Matplotlib

2.2.1 Numpy

Numpy is basic package for scientific computing. It is the python language implementation which includes powerful N-dimensional array structure, sophisticated functions, Tools that can be integrated into C/C++ and Fortran code, Linear algebra, Fourier transform and Random number features. Besides its obvious scientific uses, numpy can also be used as an efficient multidimensional container of generic data.

The main aspect Numpy is the Numpy array, on which you can do various operations. The key is that a Numpy array isn't just a regular array you'd see in a language like Java or C++, but instead it is like a mathematical object as a vector or a matrix. That means you can do vector and matrix operations like addition, subtraction, and multiplication. The most important aspect of NumPy arrays is that they are optimized for speed.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

- **Basic Array Operations**

NumPy, arrays allow a wide range of operations which can be performed on a particular array or a combination of Arrays. These operations include some basic Mathematical operation as well as Unary and Binary operations.

```
# Python program to demonstrate
# basic operations on single array
import NumPy as np

a = np.array([[1, 2], [3, 4]])# Defining Array 1
b = np.array([[4, 3], [2, 1]]) # Defining Array 2

print ("Adding 1 to every element:", a + 1)# Adding 1 to every element
print ("\n Subtracting 2 from each element:", b - 2)# Subtracting 2 from each element

# sum of array elements

print ("\n Sum of all array "elements: ", a.sum()) # Performing Unary operations

print ("\n Array sum:\n", a + b)# Performing Binary operations
```

2. Pandas

Pandas Data Frame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labelled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas Data Frame consists of three principal components, the data, rows, and columns as shown in the fig 2.2.

The basic operations which can be performed on Pandas Data Frame are:

- Creating a Data Frame
- Dealing with Rows and Columns
- Indexing and Selecting Data
- Working with Missing Data

The diagram shows a Pandas Data Frame with 7 rows and 6 columns. The columns are labeled 'Name', 'Team', 'Number', 'Position', and 'Age'. The rows are indexed from 0 to 6. Annotations include: 'Columns' pointing to the column headers, 'Rows' pointing to the row indices, and 'Data' pointing to the data cells. A pink box highlights the data for the 'Number' column, showing values 0.0, 30.0, 8.0, NaN, 12.0, 7.0, and 11.0. A green box highlights the data for the 'Position' column, showing values PG, SG, PF, PF, PG, C, and SG. A blue box highlights the data for the 'Age' column, showing values 25.0, 27.0, 29.0, 21.0, 22.0, NaN, and 27.0.

	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Fig 2.2 Pandas Data Frame components example

#Python program using Pandas for arranging a given set of data into a table

```
import pandas as pd
```

```
data = {"country": ["Brazil", "Russia", "India", "China", "South Africa"],  
        "capital": ["Brasilia", "Moscow", "New Delhi", "Beijing", "Pretoria"],  
        "area": [8.516, 17.10, 3.286, 9.597, 1.221],  
        "population": [200.4, 143.5, 1252, 1357, 52.98] }  
data table = pd. DataFrame(data)
```

```
print(data table)
```

2.2.3Matplotlib

Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar charts, etc,

#Python program using Matplotlib for forming a linear plot

```
import matplotlib. Pyplot as plt# importing the necessary packages and modules
```

```
import NumPy as np
```

```
x = np. linspace (0, 10, 100)# Prepare the data
```

```
plt. Plot (x, x, label ='linear' )# Plot the data
```

```
plt. Legend() # Add a legend
```

```
plt. Show()# Show the plot
```

2.3 Software tool used

Anaconda is a [free and open-source](#) distribution of the [Python](#) programming language for [scientific computing](#) ([data science](#), [machine learning](#) applications, large-scale data processing, [predictive analytics](#), etc.), that aims to simplify [package management](#) and deployment. Package versions are managed by the [package management system conda](#). Anaconda distribution comes with more than 1,500 packages as well as the [conda](#) package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI). Anaconda Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. There are many applications available by default in navigator; among them is the Spyder.

Spyder is an [open source](#) cross-platform [integrated development environment](#) (IDE) for scientific programming in the [Python language](#). Spyder integrates with a number of prominent packages. Some of the features of Spyder are:

- An editor with [syntax highlighting](#), [introspection](#), [code completion](#).
- Support for multiple [I Python consoles](#).
- The ability to explore and edit [variables](#) from a [GUI](#).
- A Help pane able to retrieve and render rich text [documentation](#) on functions, classes and methods automatically or on-demand.
- A [debugger](#) linked to IP dB, for step-by-step execution.
- A run-time [Profiler](#), to benchmark code.
- Project support, allowing work on multiple development efforts simultaneously.
- A built-in [file explorer](#), for interacting with the filesystem and managing projects.
- A "Find in Files" feature, allowing full [regular expression](#) search over a specified scope.
- An online help browser, allowing users to search and view Python and package documentation inside the IDE.
- A [history log](#), recording every user command entered in each console.
- An internal console, allowing for introspection and control over Spyder's own operation.

4. Open CV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as [Numpy](#) which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e., whatever operations one can do in Numpy can be combined with OpenCV.

Definition

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

1. Topics Covered

- **Converting Image to Grey scale and HSV scale**

Here we convert the given image into a grey scale image i.e., Black and white image. The function used here is `cvtColor` and the code used for grey scale is 'COLOR_BGR2GRAY'. Similarly, we do for HSV scale and the function used is 'COLOR_BGR2HSV'. Following is an example for the above scales.

CODE:

```
import cv2
path=" C:\\Users\\Admin\\Desktop\\ajay\\Images\\cats.jpg"
img = cv2.imread(path)
cv2.imshow("Original Image", img)
img1= cv2.resize(img,(512,512))
cv2.imshow("Resized Image",img1)
grayimg = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
grayimg= cv2.resize(grayimg,(512,512))
```

```
cv2.imshow("Gray Image", grayimg)
having = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
having= cv2.resize(having,(512,512))
cv2.imshow(" HSV Image", having)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

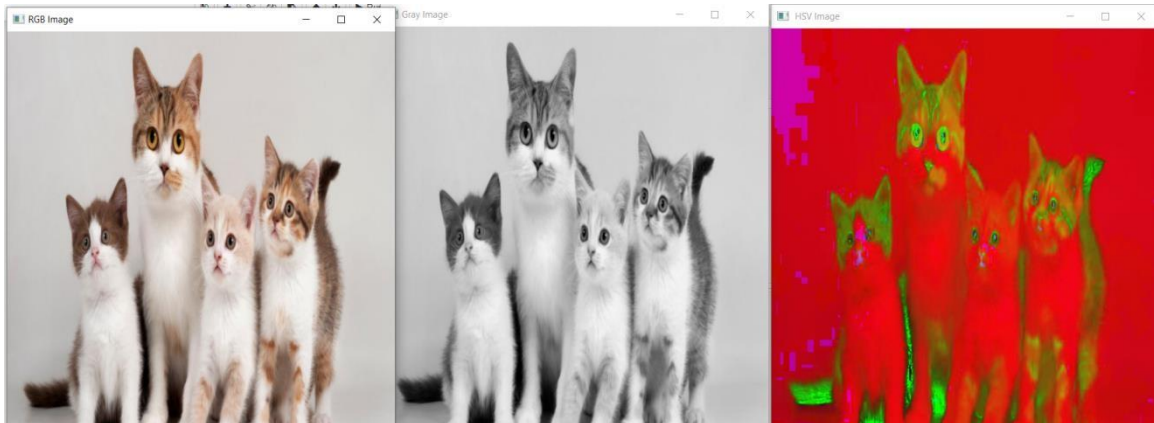


Fig 2.3 Example of Grey and HSV Image

- **Resizing Image**

Here we can resize the original image into whichever size required by us. The function used for resizing is ‘resize’. For further applications we require resizing of images in order to match them for further uses. This is particularly useful for cascading more than two images.

CODE:

```
import cv2
path = "C:\\Users\\Admin\\Desktop\\ajay\\Images\\Image.png"
img = cv2.imread(path)

cv2.imshow("Original Image", img)
img1= cv2.resize(img,(512,512))
cv2.imshow("Resized Image",img1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




Fig 2.4 Example of Resized Image

- **Adding Images and Transparency**

Here in order cascade or adding the images we require the images to be of the same size. We use resize function to resize the image. Here adding the images means in single image we paste them with required transparency. We can adjust the transparency according to our demand.

CODE:

```
import cv2
path1="C:\\Users\\Admin\\Desktop\\ajay\\Images\\Image.png"
img1 = cv2.imread(path1)
path2="C:\\Users\\Admin\\Desktop\\ajay\\Images\\cats.jpg"
img2 = cv2.imread(path2)
img1 = cv2.resize(img1,(512,512))
img2 = cv2.resize(img2,(512,512))
addimages = cv2.add(img1,img2)
addtransperency = cv2.addWeighted(img1,0.8,img2,0.2,0)
cv2.imshow("First Image",img1)
cv2.imshow("Second Image",img2)
cv2.imshow("Added Image", addimages)
cv2.imshow("Added Image with transparency", addtransperency)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

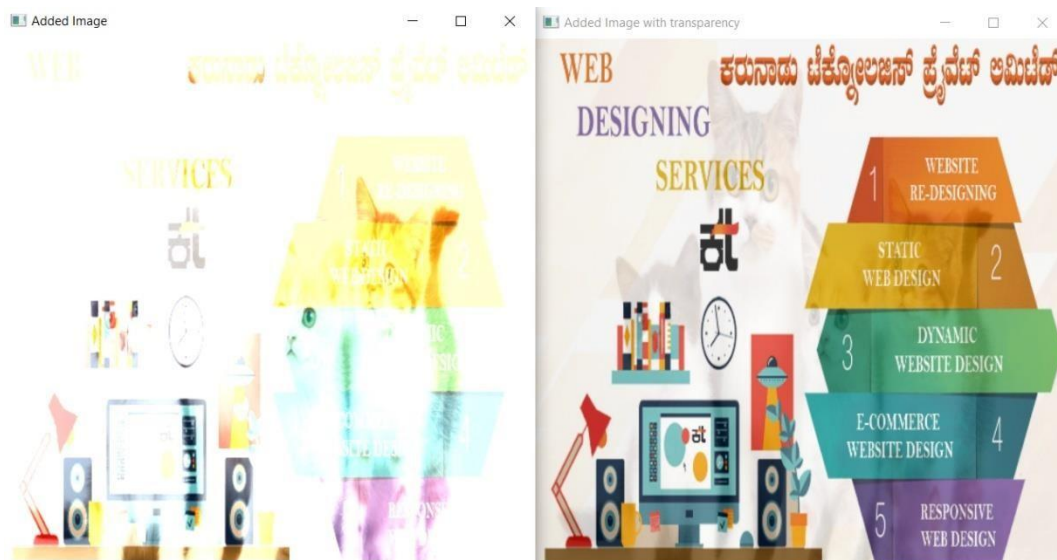


Fig 2.5 Example of Added and Transparency Image

- **Rotating Images**

Here we rotate the images in clockwise or anticlockwise direction in the required degree. The function we use 'rotate' in order to rotate the images.

CODE:

```
import cv2
path="C:\\Users\\Admin\\Desktop\\ajay\\Images\\cats.jpg"
img = cv2.imread(path)

imgresize = cv2.resize(img, (300,300))
img90 = cv2.rotate(imgresize, cv2.ROTATE_90_CLOCKWISE)
img180 = cv2.rotate(imgresize, cv2.ROTATE_180)

img270 = cv2.rotate(imgresize, cv2.ROTATE_90_COUNTERCLOCKWISE)
cv2.imshow("Original Image", img)
cv2.imshow("90degree Image", img90)
cv2.imshow("180degree Image", img180)
cv2.imshow("270degree Image", img270)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

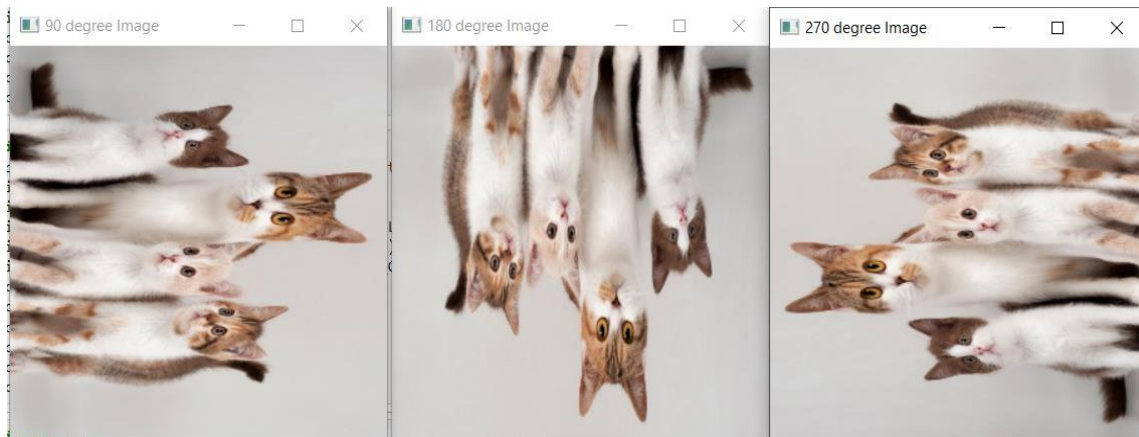


Fig 2.6 Example of Rotated Image

- **Face and Eye Detection**

Here we detect the face and eye of an individual while he is alone or in a group. It is possible to detect more than one face and eye at a time. Here we use haarcascades which is a pre-defined function in order to detect the faces and eyes. It is also possible to detect the faces and eyes in image or video. Similarly, we can do live screening and also, we need to define different function for various tasks.

CODE:

```
import cv2
path="C:\\Users\\Admin\\Desktop\\ajay\\Images\\rotated_face.jpeg"
img = cv2.imread(path)

face_cascade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_
_default.xml')
faces = face_cascade. detectMultiScale(img,1.1,4)
for (x, y, w, h) in faces:
cv2.rectangle(img, (x, y),(x+w,y+h),(255,255,0),8)

cv2.imshow("Face detected", img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

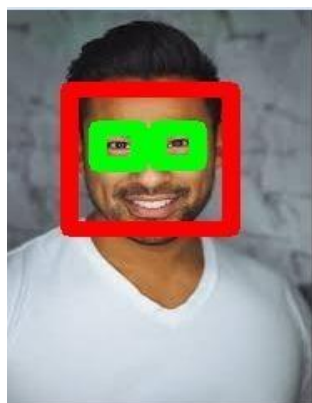


Fig 2.7 Example of Face and Eye Detection.

2.5 Machine Learning

Machine learning (ML) is a type of artificial intelligence ([AI](#)) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning [algorithms](#) use historical data as input to predict new output values.

Definition: Ability of a machine to improve its own performance through the use of software that employs artificial intelligence techniques to mimic the ways by which humans seem to learn, such as repetition and experience.

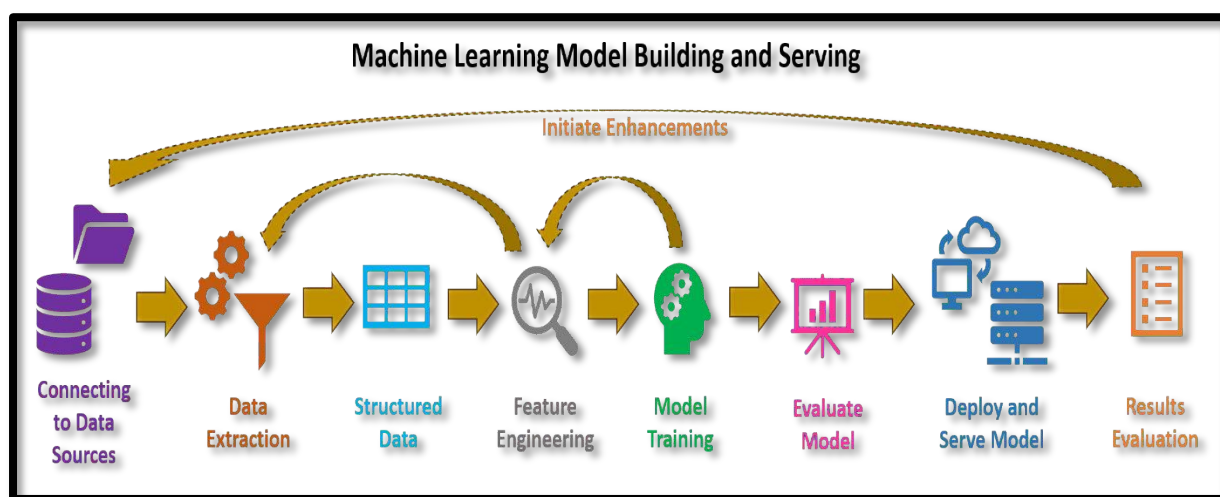


Fig 2.8 Machine Learning Infrastructure

Machine Learning (ML) is a sub-field of Artificial Intelligence (AI) which concerns with developing computational theories of learning and building learning machines. This sequence of learning and building is as shown in fig 2.9. The goal of machine learning, closely coupled with the goal of AI, is to achieve a thorough understanding about the nature

of learning process (both human learning and other forms of learning), about the computational aspects of learning behaviours, and to implant the learning capability in computer systems.

1. The Goals of Machine Learning

The goal of ML, in simple words, is to understand the nature of human and other forms of learning, and to build learning capability in computers. To be more specific, there are three aspects of the goals of ML.

- To make the computers smarter, more intelligent. The more direct objective in this aspect is to develop systems (programs) for specific practical learning tasks in application domains.
- To develop computational models of human learning process and perform computer simulations. The study in this aspect is also called cognitive modelling.
- To explore new learning methods and develop general learning algorithms independent of applications.

2. Types of Machine Learning

□ Supervised learning

Supervised learning is a type of machine learning method in which we provide sample labelled data to the machine learning system in order to train it, and on that basis, it predicts the output. The system creates a model using labelled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is **spam filtering**.

Supervised learning can be grouped further in two categories of algorithms:

- **Classification**: A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.



2.9 Flow Model of Algorithms of Supervised Learning.

□ Unsupervised Learning

Unsupervised learning is the training of machine using information that is neither classified nor labelled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data. Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore, machine is restricted to find the hidden structure in unlabelled data by our-self.

For instance, suppose it is given an image having both dogs and cats which have not seen ever. Thus, machine has no any idea about the features of dogs and cat so we can't categorize it in dogs and cats. But it can categorize them according to their similarities, patterns and differences i.e., one can easily categorize the above picture into two parts. First may contain all pics having dogs in it and second part may contain all pics having cats in it.

Unsupervised learning classified into two categories of algorithms:

- **Clustering**: A clustering problem is where one wants to discover the inherent groupings in the data as shown in fig 2.11, such as grouping customers by purchasing behaviour.
- **Association**: An association rule learning problem is where one wants to discover rules that describe large portions of the data, such as people that buy X also tend to buy Y.

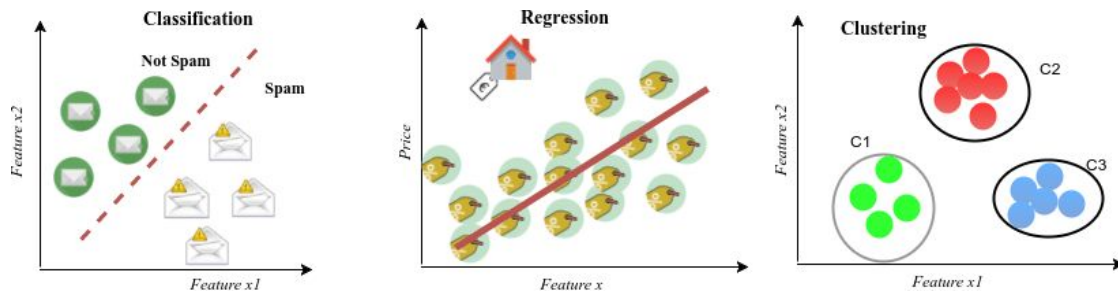
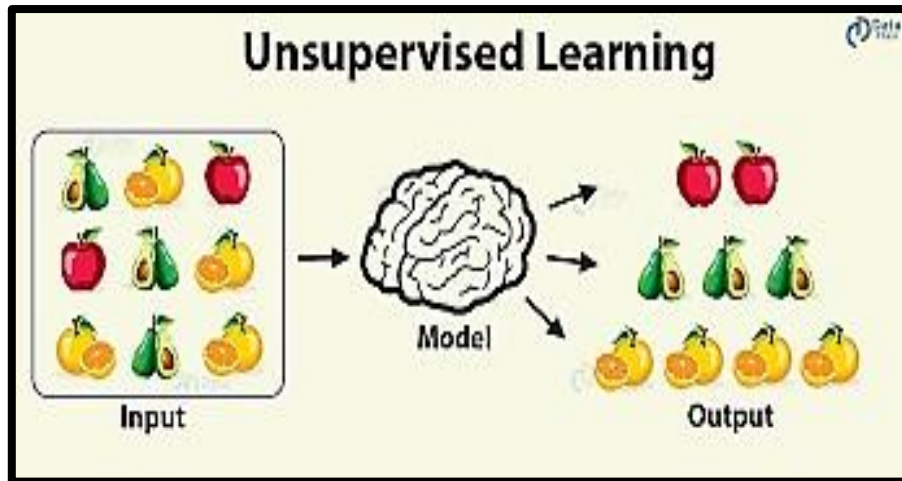


Fig 2.10 Clustering and Regression



2.11 Flow Model of Algorithms of Unsupervised Learning.

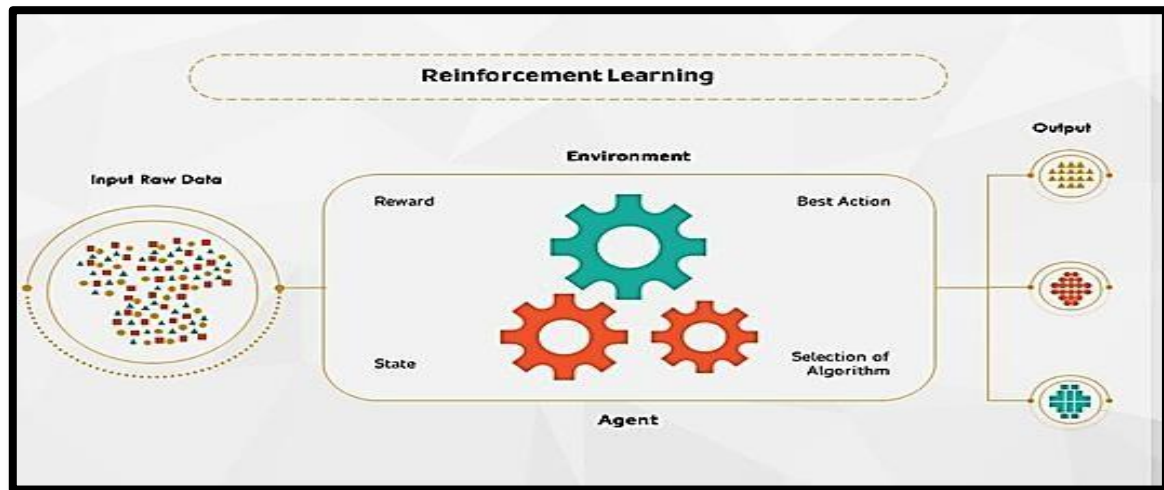
□ Reinforcement learning

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience.

There are two types of Reinforcement learning:

- **Positive:** Positive Reinforcement is defined as when an event, occurs due to a particular behaviour, increases the strength and the frequency of the behaviour. In other words, it has a positive effect on the behaviour. It maximizes performance and sustains change for a long period of time. Too much Reinforcement can lead to overload of states which can diminish the results.

- **Negative:** Negative Reinforcement is defined as strengthening of behaviour because a negative condition is stopped or avoided. It increases behaviour and provide defiance to minimum standard of performance. It only provides enough to meet up the minimum behaviour.



2.13 Flow Model of Algorithms of Reinforcement Learning.

3. Applications of Machine Learning

Machine learning has been recognized as central to the success of Artificial Intelligence, and it has applications in various areas of science, engineering and society. Some of them are:

- Product recommendations (e.g., Amazon etc.)
- Refining the search engine results(e.g., Google)
- Fighting the web spam(e.g., Gmail)
- Video surveillance(e.g., crime alerts)
- Face recognition and many more.

CHAPTER 3

Machine Learning Algorithms

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

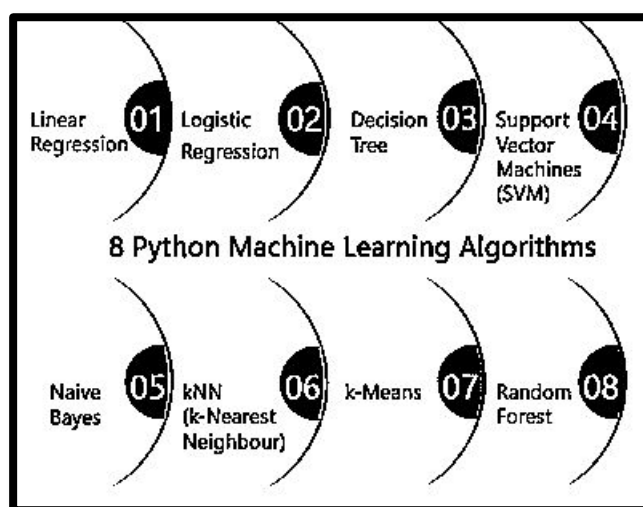


Fig 3.1 Types of Machine Learning Algorithms

3.1 Linear Regression

It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick while learning predictive modelling. In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear. Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables(X) using a best fit straight line (also known as regression line) as shown in fig 3.2. It is represented by an equation $Y = a + b * X + e$, where a is intercept, b is slope of the line and e is error term. This equation can be used to predict the value of target variable based on given predictor variable.

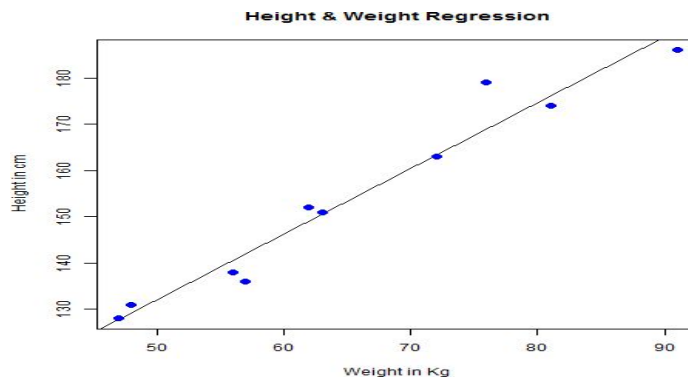
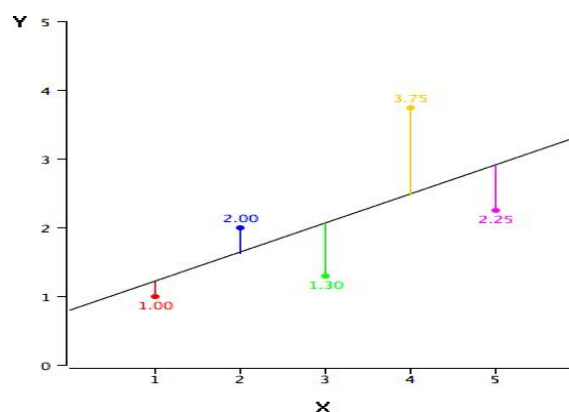


Fig 3.2 Linear Regression

3.1.1 Obtaining best fit line (Value of a and b)

This task can be easily accomplished by Least Square Method. It is the most common method used for fitting a regression line. It calculates the best-fit line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line. Because the deviations are first squared, when added, there is no cancelling out between positive and negative values.



$$\min_w ||Xw - y||_2^2$$

Fig 3.3 Plot of Input Graph

There must be linear relationship between independent and dependent variables. Linear Regression is very sensitive to Outliers. It can terribly affect the regression line and eventually the forecasted values. Simple linear regression is used for finding the relationship between the dependent variable Y and the independent or predictor variable X. Both of these variables are continuous in nature. While performing simple linear regression, we assume that the values of

predictor variable X are controlled. Furthermore, they are not subject to the measurement error from which the corresponding value of Y is observed.

The equation of a simple linear regression model to calculate the value of the dependent variable, Y based on the predictor X is as follows:

$$y_i = b_0 + b_1 x + e_i$$

Where the value of y_i is calculated with the input variable x_i for every i th data point; The coefficients of regressions are denoted by b_0 and b_1 ; the i th value of x has e_i as its error in the measurement.

Regression analysis is implemented to do the following:

- With it, we can establish a linear relationship between the independent and the dependent variables.
- The input variables x_1, x_2, \dots, x_n is responsible for predicting the value of y .
- In order to explain the dependent variable precisely, we need to identify the independent variables carefully. This will allow us to establish a more accurate causal relationship between these two variables.

2. Advantages and Disadvantages

Advantages

Linear regression is an extremely simple method. It is very easy and intuitive to use and understand. A person with only the knowledge of high school mathematics can understand and use it. In addition, it works in most of the cases. Even when it doesn't fit the data exactly, we can use it to find the nature of the relationship between the two variables.

Disadvantages

- By its definition, linear regression only models relationships between dependent and independent variables that are linear. It assumes there is a straight-line relationship between them which is incorrect sometimes. Linear regression is very sensitive to the anomalies in the data (or outliers).
- Take for example most of your data lies in the range 0-10. If due to any reason only one of the data items comes out of the range, say for example 15, this significantly influences the regression coefficients.

- Another disadvantage is that if we have a number of parameters than the number of samples available then the model starts to model the noise rather than the relationship between the variables.

```
def AREA(self):  
    import pandas as pd  
    import matplotlib.pyplot as plt  
    from sklearn import linear_model  
  
    path="C:\\Users\\Admin\\Desktop\\ajay\\Data\\homeprices.csv"  
  
    data=pd.read_csv(path)  
  
    inputdata=data.drop('price','columns')  
    output=data.drop('area','columns')  
  
    model=linear_model.LinearRegression()  
    model.fit(inputdata,output)  
  
    a=float(self.txtarea.text())  
  
    self.lblmessage.setText(str(model.predict([[a]])))
```

HOUSE PRICE PREDICTION

AREA

PREDICT

[[533664.38356164]]

Fig 3.4 Example of Linear Regression

2. Multiple Linear Regression

In many cases, there may be possibilities of dealing with more than one predictor variable for finding out the value of the response variable. Therefore, the simple linear models cannot be utilized as there is a need for undertaking multiple linear regression for analyzing the predictor variables. The difference between simple linear regression and multiple linear regression is that, multiple linear regression has more than 1 independent variables, whereas simple linear regression has only 1 independent variable. Using the two explanatory variables, we can delineate the equation of multiple linear regression as follows:

$$y_i = b_0 + b_1x_1 + b_2x_2 + e_i$$

The two explanatory variables x_1 and x_2 , determine y_i , for the i th data point.

Furthermore, the predictor variables are also determined by the three parameters b_0 , b_1 , and b_2 of the model, and by the residual e_i of the point i from the fitted surface. General Multiple regression models can be represented as:

$$y_i = \sum b_i x_i + e_i$$

Multiple regression suffers from multicollinearity, autocorrelation, heteroskedasticity. Multicollinearity can increase the variance of the coefficient estimates and make the estimates very sensitive to minor changes in the model. The result is that the coefficient estimates are unstable. In case of multiple independent variables, we can go with forward selection, backward elimination and step wise approach for selection of most significant independent variables.

1. Advantages and disadvantages

Advantages

- The ability to determine the relative influence of one or more predictor variables to the criterion value.
- The ability to identify outliers, or anomalies.

Disadvantage

- Any disadvantage of using a multiple regression model usually comes down to the data being used. Two examples of this are using incomplete data and falsely concluding that a correlation is causation.

```
def profit(self):
    import pandas as pd
    import matplotlib.pyplot as plt
    import sklearn
    from sklearn import linear_model
    path = "C:\\Users\\Admin\\Desktop\\ajay\\Data\\50_Startups.csv"
    data = pd.read_csv(path)
    print(data)
    inputs=data.drop('Profit','columns')
    print(inputs)
    outputs=data.drop(['R&D Spend','Administration','Marketing Spend'],'columns')
    print(outputs)
    model = linear_model.LinearRegression()
    model.fit(inputs,outputs)

    RDSpend = self.txtRdsPend.text()
    Administration = self.txtAdministration.text()
    MarketingSpend = self.txtMarketingSpend.text()

    ans = model.predict([[int(RDSpend),int(Administration),int(MarketingSpend)]])

    self.lblMessage.setText(str(ans))
```

predicting the profit using multiple linear regression

R&D Spend	<input type="text" value="2560"/>
Administration	<input type="text" value="25"/>
Marketing Spend	<input type="text" value="2569"/>

[[52254.10201691]]

Fig 3.5 Example of Multiple Linear Regression

3. Logistic Regression

Logistic regression is a [statistical model](#) that in its basic form uses a [logistic function](#) to model a [binary dependent variable](#), although many more complex [extensions](#) exist. In [regression analysis](#), logistic regression is [estimating](#) the parameters of a logistic model (a form of [binary regression](#)). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an [indicator variable](#), where the two values are labelled "0" and "1". In the logistic model, the [log-odds](#) (the [logarithm](#) of the [odds](#)) for the value labelled "1" is a [linear combination](#) of one or more [independent variables](#) ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a [continuous variable](#) (any real value). The corresponding [probability](#) of the value labelled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labelling, the function that converts log-odds to probability is the logistic function, hence the name. The [unit of measurement](#) for the log-odds scale is called a [logit](#), from logistic unit, hence the alternative names. Analogous models with a different [sigmoid function](#) instead of the logistic function can also be used, such as the [probit model](#); the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the [odds ratio](#).

1. Key Features

- Logistic regression predicts whether something is True(1) or False(0) instead, predicting something that is continuous like size.
- It has an S-shaped line.
- We can take our Linear Regression Model and convert it into Logistic Regression model with the help of Sigmoid Function.
- Logistic Regression's ability to provide probabilities and classify new samples using continuous and discrete measurements makes it a popular machine learning method.

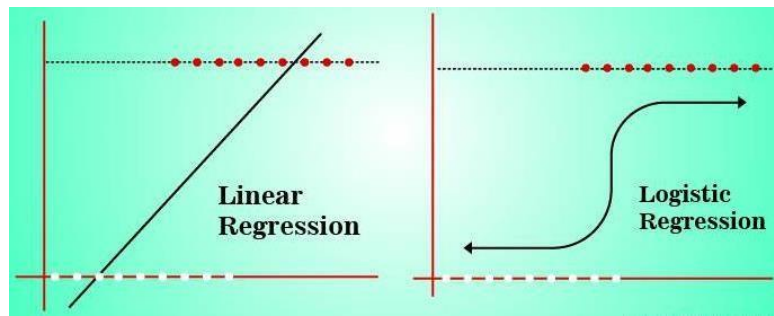


Fig 3.6 Linear Regression v/s Logistic Regression

This is where logistic regression comes into play. In logistic regression, you get a probability score that reflects the probability of the occurrence of the event. An event in this case is each row of the training dataset. It could be something like classifying if a given email is spam, or mass of cell is malignant or a user will buy a product and so on.

2. Advantages and Disadvantages

Advantages

- It doesn't require high computational power.
- Is easily interpretable.
- Is used widely by the data analyst and data scientists.
- Is very easy to implement.
- It doesn't require scaling of features.
- It provides a probability score for observations.

Disadvantages

- While working with Logistic regression you are not able to handle a large number of categorical features/variables.
- It is vulnerable to over fitting.
- It can't solve the non-linear problem with the logistic regression model that is why it requires a transformation of non-linear features.
- Logistic regression will not perform well with independent(X) variables that are not correlated to the target(Y) variable.


```
def buy(self):
    import pandas as pd
    path = "C:\\Users\\Admin\\Desktop\\ajay\\LogisticRegression\\Magzine\\Kid.csv"
    data = pd.read_csv(path)
    print(data)

    inputs = data.drop(['Buy', 'Obs No'], 'columns')
    output = data['Buy']
    print(inputs)
    print(output)

    import sklearn
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(inputs, output, test_size=0.2)
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    print(y_pred)
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    print(cm)
    Income = self.txtincome.text()
    IsFemale = self.txtisfemale.text()
    IsMarried = self.txtismarried.text()
    HasCollege = self.txtthascollege.text()
    IsProfessional = self.txtisprofessional.text()
    IsRetired = self.txtisretired.text()
    Unemployed = self.txtunemployed.text()
    ResidenceLength = self.txtresidenceLength.text()
    DualIncome = self.txtDualIncome.text()
    Minors = self.txtminors.text()
    Own = self.txtown.text()
    House = self.txtHouse.text()
    White = self.txtWhite.text()
    English = self.txtEnglish.text()
    PrevChildMag = self.txtprevchildmag.text()
    PrevParentMag = self.txtprevparentmag.text()
    ans = model.predict([int(Income), int(IsFemale), int(IsMarried), int(HasCollege), int(IsProfessional),
                          int(IsRetired), int(Unemployed), int(ResidenceLength), int(DualIncome),
                          int(Minors), int(Own), int(House), int(White), int(English), int(PrevChildMag), int(PrevParentMag))])

    if ans==0:
        self.lblmessage.setText(str("YES"))
    elif ans == 1:
        self.lblmessage.setText(str("NO"))
```

MAGZINE BUY OR NOT			
INCOME	<input type="text" value="7500"/>	DUAL INCOME	<input type="text" value="1"/>
IS FEMALE	<input type="text" value="1"/>	MINORS	<input type="text" value="0"/>
IS MARRIED	<input type="text" value="1"/>	OWN	<input type="text" value="1"/>
HAS COLLEGE	<input type="text" value="1"/>	HOUSE	<input type="text" value="1"/>
IS PROFESSIONAL	<input type="text" value="1"/>	WHITE	<input type="text" value="1"/>
IS RETIRED	<input type="text" value="0"/>	ENGLISH	<input type="text" value="1"/>
UNEMPLOYED	<input type="text" value="0"/>	PREV CHILD MAG	<input type="text" value="1"/>
RESIDENCE LENGTH	<input type="text" value="15"/>	PREV PARENT MAG	<input type="text" value="0"/>
PREDICT			
NO			

MAGZINE BUY OR NOT			
INCOME	<input type="text" value="2400"/>	DUAL INCOME	<input type="text" value="0"/>
IS FEMALE	<input type="text" value="1"/>	MINORS	<input type="text" value="0"/>
IS MARRIED	<input type="text" value="0"/>	OWN	<input type="text" value="0"/>
HAS COLLEGE	<input type="text" value="1"/>	HOUSE	<input type="text" value="1"/>
IS PROFESSIONAL	<input type="text" value="1"/>	WHITE	<input type="text" value="0"/>
IS RETIRED	<input type="text" value="0"/>	ENGLISH	<input type="text" value="0"/>
UNEMPLOYED	<input type="text" value="0"/>	PREV CHILD MAG	<input type="text" value="0"/>
RESIDENCE LENGTH	<input type="text" value="26"/>	PREV PARENT MAG	<input type="text" value="0"/>
PREDICT			
YES			

Fig 3.7 Example of Logistic Regression

4. KNN

K nearest neighbors or KNN Algorithm is a simple algorithm which uses the entire dataset in its training phase. Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction. KNN is often used in search applications where you are looking for similar items, like find items similar to this one.

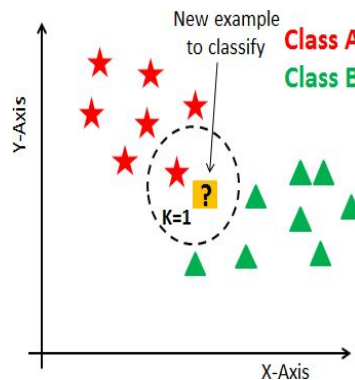
1. Features of KNN Algorithm

- KNN is a Supervised Learning algorithm that uses labelled input data set to predict the output of the data points.
- It is one of the simplest Machine learning algorithms and it can be easily implemented for a varied set of problems.
- It is mainly based on feature similarity. KNN checks how similar a data point is to its neighbor and classifies the data point into the class it is most similar to.
- Unlike most algorithms, KNN is a non-parametric model which means that it does not make any assumptions about the data set. This makes the algorithm more effective since it can handle realistic data.
- KNN is a lazy algorithm; this means that it memorizes the training data set instead of learning a discriminative function from the training data.
- KNN can be used for solving both classification and regression problems.

2. Working

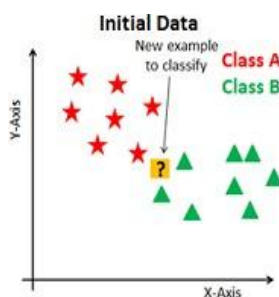
In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm. This is the simplest case. However, the number of neighbours (K) is a hyper parameter that needs to be chosen at the time of model building. Research has shown that no optimal number of neighbors suits all kind of data sets. Each dataset has its own requirements. Generally, Data scientists choose as an odd number if the number of classes is even. We can also check by generating the model on different values of k and check their performance.

Suppose ‘?’ 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 are 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. Then we find the one closest point to ‘?’ and then the label of the nearest point is assigned to ‘?’.

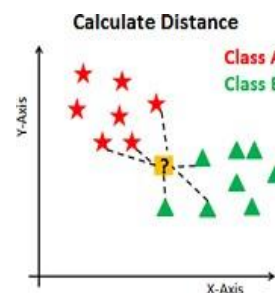


KNN has the following basic steps:

1. Calculate distance



2. Find closest neighbors



3. Vote for labels

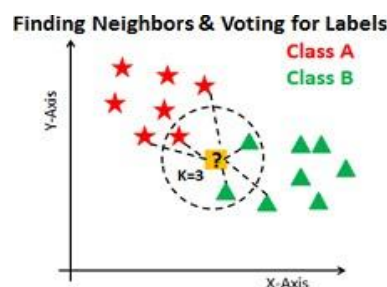


Fig 3.8 Plot of ideal KNN Algorithm

3. Advantages and Disadvantages

Advantages

- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.
- The algorithm is versatile. It can be used for classification, regression, and search
- The training phase of K-nearest neighbor classification is much faster compared to other classification algorithms. There is no need to train a model for generalization that is why KNN is known as the simple and instance-based learning algorithm.
- KNN can be useful in case of nonlinear data. It can be used with the regression problem. Output value for the object is computed by the average of k closest neighbors value.

Disadvantages

- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.
- The testing phase of K-nearest neighbor classification is slower and costlier in terms of time and memory. It requires large memory for storing the entire training dataset for prediction.
- KNN requires scaling of data because KNN uses the Euclidean distance between two data points to find nearest neighbors. Euclidean distance is sensitive to magnitudes. The features with high magnitudes will weigh more than features with low magnitudes.
- KNN also not suitable for large dimensional data.

```
def Diabet(self):
    import pandas as pd
    path="C:\\Users\\Admin\\Desktop\\ajay\\final report\\diabetes2.csv"
    data=pd.read_csv(path)
    print(data)
    import sklearn
    from sklearn.model_selection import train_test_split
    inputs=data.drop('Outcome', 'columns')
    output=data['Outcome']
    x_train,x_test,y_train,y_test=train_test_split(inputs,output,train_size=0.8)
    from sklearn.neighbors import KNeighborsClassifier
    model=KNeighborsClassifier(n_neighbors=27)
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    from sklearn.metrics import confusion_matrix
    cm=confusion_matrix(y_test,y_pred)
    print(cm)
    result = model.predict([[int(self.txtpreg.text()),int(self.txtglu.text()),int(self.txtbp.text()),
                             int(self.txtskinth.text()),int(self.txtinsu.text()),float(self.txtbmi.text()),
                             float(self.txtdpf.text()),int(self.txtage.text())]])
    self.lblmessage.setText(str(result))
```

diabetes

Diabetes

Pregnancies	<input type="text" value="1"/>	Insulin	<input type="text" value="0"/>
Glucose	<input type="text" value="126"/>	BMI	<input type="text" value="30.1"/>
BloodPressure	<input type="text" value="60"/>	Age	<input type="text" value="47"/>
SkinThickness	<input type="text" value="0"/>		
DiabetesPedigreeFunction	<input type="text" value="0.349"/>		

[0]

Fig 3.9 Example of KNN Algorithm

5. SVM

“Support Vector Machine” (SVM) is a supervised [machine learning algorithm](#) that can be used for both classification and regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate.

1. Features of SVM

- SVM is a Supervised Learning algorithm that uses labelled input data set to predict the output of the data points.
- It is one of the simplest Machine learning algorithms and it can be easily implemented for a varied set of problems.
- SVM can be used for solving both classification and regression problems.

2. Working

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x_1 and x_2 . We want a classifier that can classify the pair(x_1 , x_2) of coordinates in either green or blue. So, as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.

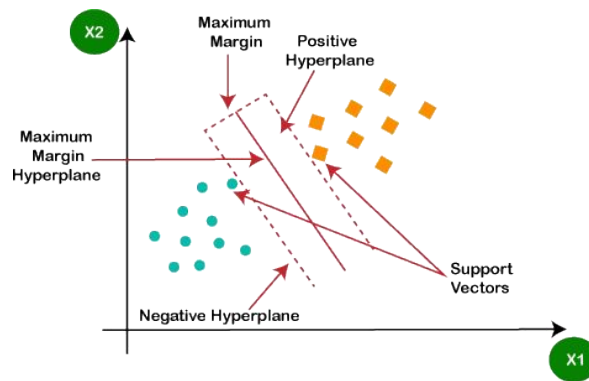


Fig 3.10 Plot of ideal SVM Algorithm

3. Advantages and Disadvantages

Advantages

- SVM works relatively well when there is a clear margin of separation between classes.
- SVM is more effective in high dimensional spaces.
- SVM is effective in cases where the number of dimensions is greater than the number of samples.
- SVM is relatively memory efficient.

Disadvantages

- SVM algorithm is not suitable for large data sets.
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
- In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
- As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.


```
def winequality(self):
    import pandas as pd
    import sklearn
    path = "C:\\Users\\Admin\\Desktop\\ajay\\final report\\winequalitywhite.csv"
    data = pd.read_csv(path)
    print(data)
    data.fixedacidity = data.fixedacidity.fillna(data.fixedacidity.median())
    data.volatileacidity = data.volatileacidity.fillna(data.volatileacidity.median())
    data.citricacid = data.citricacid.fillna(data.citricacid.median())
    data.residualsugar = data.residualsugar.fillna(data.residualsugar.median())
    data.chlorides = data.chlorides.fillna(data.chlorides.median())
    data.freesulfurdioxide = data.freesulfurdioxide.fillna(data.freesulfurdioxide.median())
    data.totalsulfurdioxide = data.totalsulfurdioxide.fillna(data.totalsulfurdioxide.median())
    data.pH = data.pH.fillna(data.pH.median())
    data.density = data.density.fillna(data.density.median())
    data.sulphates = data.sulphates.fillna(data.sulphates.median())
    data.alcohol = data.alcohol.fillna(data.alcohol.median())
    print(data)
    inputs = data.drop(['type', 'quality'], 'columns')
    output = data['quality']
    import sklearn
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(inputs, output, train_size=0.8)
    from sklearn.svm import SVC
    model = SVC()
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    print(cm)
    result = model.predict(sc.fit_transform([[float(self.txtfa.text()), float(self.txtva.text()), float(self.txtca.text()),
                                             float(self.txttrs.text()), float(self.txttc.text()), float(self.txtfs.text()),
                                             float(self.txtts.text()), float(self.txtd.text()), float(self.txtpH.text()),
                                             float(self.txts.text()), float(self.txta.text())])))

    self.lblmsg.setText(str(result))
```

Wine Qualities

Fixed Acidity	7	Total Sulfurdioxide	186
Volatile Acidity	0.23	Density	0.99560
Citric Acid	0.32	pH	3.2
Residual Sugar	8.5	Sulphates	0.4
Chlorides	0.058	Alcohol	10
Free Sulfurdioxide	47		

Predict

[6]

Fig 3.11 Example of SVM Algorithm

6. Decision Tree

Decision Tree is a **supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome**. It is a graphical representation for getting all the possible solutions to a problem/decision on based on given conditions. In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node**. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

1. Features of Decision tree

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.
- It is very easy to understand and implement.

2. Working

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub- nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM)**.

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step3.

Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

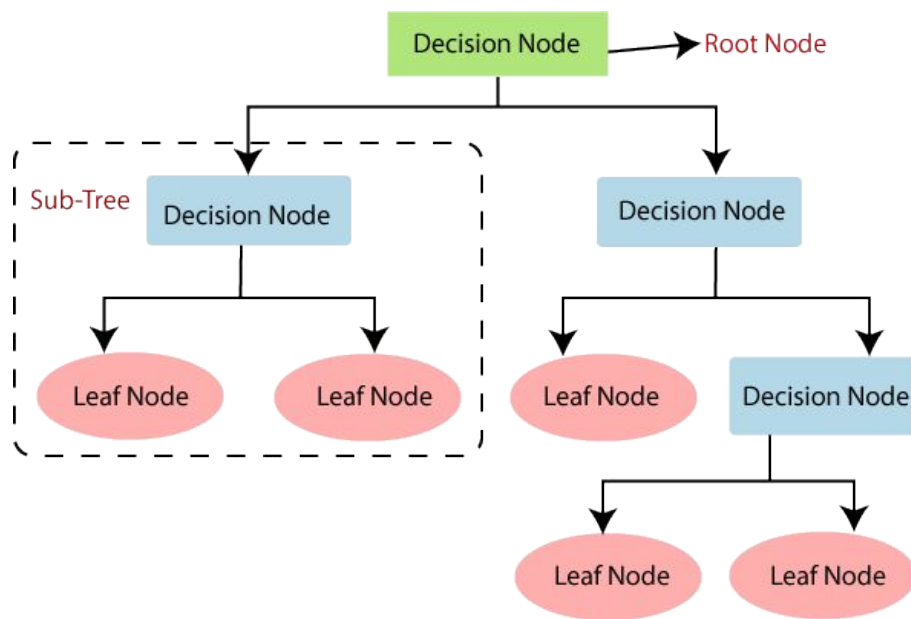


Fig 3.12 Ideal diagram of a Decision Tree

3. Advantages and Disadvantages

Advantages

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages

- The decision tree contains lots of layers, which makes it complex.
- It may have an over fitting issue, which can be resolved using the **Random Forest algorithm**.
- For more class labels, the computational complexity of the decision tree may increase.

```
def loan(self):
    import pandas as pd
    path = "C:\\Users\\Admin\\Desktop\\ajay\\SVM\\loanpredictiondataset\\train.csv"
    data = pd.read_csv(path)
    print(data)
    inputs = data.drop('Loan_Status','columns')
    output = data['Loan_Status']

    import sklearn
    from sklearn.preprocessing import LabelEncoder

    le_Gender = LabelEncoder() # initialize my Label encoder
    inputs['Gender_n'] = le_Gender.fit_transform(inputs['Gender'])
    le_Married = LabelEncoder() # initialize my Label encoder
    inputs['Married_n'] = le_Married.fit_transform(inputs['Married'])
    le_Education = LabelEncoder() # initialize my Label encoder
    inputs['Education_n'] = le_Education.fit_transform(inputs['Education'])
    le_Self_Employed = LabelEncoder() # initialize my Label encoder
    inputs['Self_Employed_n'] = le_Self_Employed.fit_transform(inputs['Self_Employed'])
    le_Property_Area = LabelEncoder() # initialize my Label encoder
    inputs['Property_Area_n'] = le_Property_Area.fit_transform(inputs['Property_Area'])

    inputs = inputs.drop(['Gender','Married','Education','Self_Employed','Property_Area','Loan_ID'],'columns')# removing col
    print(inputs)
    print(output)

    from sklearn import tree
    model = tree.DecisionTreeClassifier()# init model
    model.fit(inputs,output)#traine
    Gender = self.txtgender.text()
    Married = self.txtmarried.text()
    Dependents = self.txtdependents.text()
    Education = self.txteducation.text()
    Self_Employed = self.txtselfemployed.text()
    ApplicantIncome = self.txtapplicantincome.text()
    CoapplicantIncome = self.txtcoapplicantincome.text()
    LoanAmount = self.txtloanamount.text()
    Loan_Amount_Term = self.txtloanamountterm.text()
    Credit_History = self.txtcredithistory.text()
    Property_Area = self.txtpropertyarea.text()

    ans = model.predict([[float(Gender),float(Married),float(Dependents),float(Education),float(Self_Employed)
    ,float(ApplicantIncome),float(CoapplicantIncome),float(LoanAmount),float(Loan_Amount_Term)
    ,float(Credit_History),float(Property_Area)]]])

    if ans == 'Y':
        self.lblmessage.setText(str("YES"))
    elif ans == 'N':
        self.lblmessage.setText(str("NO"))
```

LOAN STATUS

GENDER	<input type="text" value="1"/>	APPLICANT INCOME	<input type="text" value="5849"/>
MARRIED	<input type="text" value="0"/>	COAPPLICANT INCOME	<input type="text" value="0"/>
DEPENDENTS	<input type="text" value="0"/>	LOAN AMOUNT	<input type="text" value="266"/>
EDUCATION	<input type="text" value="0"/>	LOAN_AMOUNT_TERM	<input type="text" value="360"/>
SELF_EMPLOYED	<input type="text" value="0"/>	CREDIT_HISTORY	<input type="text" value="1"/>
PROPERTY_AREA	<input type="text" value="2"/>		

YES

LOAN STATUS

GENDER	<input type="text" value="1"/>	APPLICANT INCOME	<input type="text" value="4583"/>
MARRIED	<input type="text" value="1"/>	COAPPLICANT INCOME	<input type="text" value="1508"/>
DEPENDENTS	<input type="text" value="1"/>	LOAN AMOUNT	<input type="text" value="128"/>
EDUCATION	<input type="text" value="0"/>	LOAN_AMOUNT_TERM	<input type="text" value="360"/>
SELF_EMPLOYED	<input type="text" value="0"/>	CREDIT_HISTORY	<input type="text" value="1"/>
PROPERTY_AREA	<input type="text" value="0"/>		

NO

Fig 3.13 Example of Decision Tree

CHAPTER 4

Project -1 Description

Prediction Of Car Price

In this project, we will work with weight.csv dataset to develop a machine learning algorithm that predicts the weight. A model like this would be very valuable to predict one's weight using height and gender.

1. Problem Statement

Develop a model that has the capacity of predicting price by making use of the information provided in Dataset.

2. Dataset

The dataset used in this project consists of 4 variables: 'Gender', 'Height', 'Weight', 'Index'. The main variable we are interested is 'Weight'. This variable predicts the weight of the person based on the inputs given in dataset.

The dataset consists of below features which can be summarized as follows:

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	Car_Name	301 non-null	object
1	Year	301 non-null	int64
2	Selling_Price	301 non-null	float64
3	Present_Price	301 non-null	float64
4	Kms_Driven	301 non-null	int64
5	Fuel_Type	301 non-null	object
6	Seller_Type	301 non-null	object
7	Transmission	301 non-null	object
8	Owner	301 non-null	int64

	A	B	C	D	E	F	G	H	I
1	Car_Name	Year	Selling_Price	Present_Pri	Kms_Driven	Fuel_Type	Seller_Type	Transmissio	Owner
2	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
3	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
4	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
5	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
6	swift	2014	4.6	6.87	42450	Diesel	Dealer	Manual	0
7	vitara brezza	2018	9.25	9.83	2071	Diesel	Dealer	Manual	0
8	ciaz	2015	6.75	8.12	18796	Petrol	Dealer	Manual	0
9	s cross	2015	6.5	8.61	33429	Diesel	Dealer	Manual	0
10	ciaz	2016	8.75	8.89	20273	Diesel	Dealer	Manual	0
11	ciaz	2015	7.45	8.92	42367	Diesel	Dealer	Manual	0
12	alto 800	2017	2.85	3.6	2135	Petrol	Dealer	Manual	0
13	ciaz	2015	6.85	10.38	51000	Diesel	Dealer	Manual	0
14	ciaz	2015	7.5	9.94	15000	Petrol	Dealer	Automatic	0
15	ertiga	2015	6.1	7.71	26000	Petrol	Dealer	Manual	0
16	dzire	2009	2.25	7.21	77427	Petrol	Dealer	Manual	0
17	ertiga	2016	7.75	10.79	43000	Diesel	Dealer	Manual	0
18	ertiga	2015	7.25	10.79	41678	Diesel	Dealer	Manual	0

Fig 4.1 Overview of Dataset

○ Algorithm –Multiple Linear Regression

It is a very simple python program to implement. Multiple regression is like [linear regression](#), but with more than one independent value, meaning that we try to predict a value based on two or more variables. Multiple Linear Regression is implemented using the LinearRegression class from sklearn.linear_model library.

○ Programming Steps

- This project requires us to predict the weight of a person based on the given input dataset.
- First, we read the given dataset using pandas function.
- Then we print the inputs and output from csv file.
- Label encoding is used for 'Gender' column.
- We initialize the model i.e., **Multiple Linear Regression**.
- We further implement this using Pyqt in order for better representation.

Code:

```
import pandas as ps

path="C:\\Users\\shiva\\OneDrive\\Desktop\\Car Price Prediction\\car.csv"

data = ps.read_csv(path)

print(data)

print(data.info())


import sklearn

from sklearn.preprocessing import LabelEncoder

laFuel_Type=LabelEncoder()

laSeller_Type=LabelEncoder()

laTransmission=LabelEncoder()


data['Fuel_Type_n']=laFuel_Type.fit_transform(data['Fuel_Type'])
data['Seller_Type_n']=laSeller_Type.fit_transform(data['Seller_Type'])
data['Transmission_n']=laTransmission .fit_transform(data['Transmission'])


inputs = data.drop(['Selling_Price','Car_Name','Fuel_Type','Seller_Type','Transmission'],'columns')
output=data.drop(['Car_Name','Year','Present_Price','Kms_Driven','Fuel_Type','Seller_Type','Transmission',
'Owner','Fuel_Type_n','Seller_Type_n','Transmission_n'],'columns')

print("inputs:",inputs)

print("output:",output)


import sklearn          #importing
from sklearn import linear_model

model = linear_model.LinearRegression()  #intilaizing the model
model.fit(inputs,output)
acc=model.score(inputs,output)*100

print("acc :",acc)


res = model.predict([[2014,5.59,27000,0,2,0,1]])  # area,3 column,bedrooms
print("Selling Price:",res)
```

4.2

Output:

output: Selling_Price

0 3.35

1 4.75

2 7.25

3 2.85

4 4.60

.. ...

296 9.50

297 4.00

298 3.35

299 11.50

300 5.30

[301 rows x 1 columns]

acc : 87.85205982910816

Selling Price: [[1564.40341731]]

Implementing Code in PYQT5:

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'carprice.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(979, 835)
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(290, 30, 371, 51))
        self.label.setStyleSheet("color: rgb(255, 255, 255);\n"
"background-color: rgb(0, 0, 0);\n"
"font: 75 14pt \"Palatino Linotype\";")
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(Form)
        self.label_2.setGeometry(QtCore.QRect(50, 120, 111, 61))
        self.label_2.setStyleSheet("color: rgb(0, 0, 0);\n"
"font: 12pt \"MV Boli\";")
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(Form)
        self.label_3.setGeometry(QtCore.QRect(50, 190, 191, 61))
        self.label_3.setStyleSheet("color: rgb(0, 0, 0);\n"
"font: 12pt \"MV Boli\";")
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(Form)
        self.label_4.setGeometry(QtCore.QRect(50, 260, 151, 61))
        self.label_4.setStyleSheet("color: rgb(0, 0, 0);\n"
"font: 12pt \"MV Boli\";")
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(Form)
        self.label_5.setGeometry(QtCore.QRect(50, 330, 111, 61))
        self.label_5.setStyleSheet("color: rgb(0, 0, 0);\n"
"font: 12pt \"MV Boli\";")
        self.label_5.setObjectName("label_5")
        self.label_6 = QtWidgets.QLabel(Form)
        self.label_6.setGeometry(QtCore.QRect(50, 410, 161, 61))
        self.label_6.setStyleSheet("color: rgb(0, 0, 0);\n"
"font: 12pt \"MV Boli\";")
        self.label_6.setObjectName("label_6")
        self.label_7 = QtWidgets.QLabel(Form)
```

Fig 4.3

PYQT5 Output:

The image displays two side-by-side screenshots of a PyQt5 application window titled "Form". The window contains a "CAR PRICE PREDICTION :" header. Below the header, there are seven input fields with labels: "YEAR :", "Present_Price :", "Kms_Driven :", "Owner :", "Fuel_Type_n :", "Seller_Type_n :", and "Transmission_n :". Each field has a text input box. Below these fields is a green button labeled "PRICE". At the bottom, there is a label "Selling_Price :" followed by a text box. In the left screenshot, the text box is empty. In the right screenshot, the text box contains the value "[[1564.40341731]]".

Fig 4.4 PYQT5 Output of Car Price Prediction

4.5 CONCLUSION

The internship aims to use Python programming language for Machine Learning so as to apply the theoretical knowledge to solve real-time and complex problems. The internship helped to find appropriate prediction model to the problems by applying suitable learning algorithm which can be used in future. The internship project assigned by the company helped to improve programming skills and to implement basic knowledge for solving real world problem. In this project **Multiple Linear Regression Algorithm** is used to predict the **Price**.. After having a basic understanding of Supervised learning, we explored the Multiple Linear Regression Algorithm which is used to solve machine learning problem. The Price of a car based on the inputs given in the dataset.

CHAPTER 5

Project -2

Prediction of Fractured Or Not

In this project, we will work quantitative information from digitized images of a diagnostic test for the diagnosis of fractured To develop a machine learning algorithm that predicts if a particular patient has malignant or benign cancer. A model like this would be very valuable in the medical field.

1. Problem Statement

Develop a model that has the capacity of predicting whether the if a particular patient has fractured by making use of the information provided in bdig Dataset.

2. Dataset

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	id	169 non-null	int64
1	age	169 non-null	float64
2	sex	169 non-null	object
3	fracture	169 non-null	object
4	weight_kg	169 non-null	float64
5	height_cm	169 non-null	float64
6	medication	169 non-null	object
7	waiting_time	169 non-null	int64
8	bmd	169 non-null	float64

	A1		fx	id									
	A	B	C	D	E	F	G	H	I	J	K		
1	id	age	sex	fracture	weight_kg	height_cm	medication	waiting_tim	bmd				
2	469	57.052768	F	no fracture	64	155.5	Anticonvuls	18	0.8793				
3	8724	75.741225	F	no fracture	78	162	No medicat	56	0.7946				
4	6736	70.7789	M	no fracture	73	170.5	No medicat	10	0.9067				
5	24180	78.247175	F	no fracture	60	148	No medicat	14	0.7112				
6	17072	54.191877	M	no fracture	55	161	No medicat	20	0.7909				
7	3806	77.177752	M	no fracture	65	168	No medicat	7	0.7301				
8	17106	56.180618	M	no fracture	77	159	No medicat	26	1.0096				
9	23834	49.91614	F	no fracture	59	150	No medicat	9	0.731				
10	2454	68.408403	M	no fracture	64	167	Glucocortic	6	0.6893				
11	2088	66.256648	M	no fracture	72	159.5	No medicat	10	0.9466				
12	5364	45.866582	M	no fracture	62	169	No medicat	12	0.8015				
13	8922	73.970471	F	no fracture	68	164	No medicat	5	0.5793				
14	23890	60.555417	F	no fracture	76	155	No medicat	11	0.976				
15	3047	64.213103	M	no fracture	90	175	Glucocortic	28	0.9184				
16	2179	53.39544	M	no fracture	70	162.5	No medicat	73	0.802				
17	3800	66.825462	M	no fracture	76	171	No medicat	13	0.8033				
18	7528	57.934593	M	no fracture	67	160	Glucocortic	5	0.7978				
19	5288	40.232367	M	no fracture	66	165	No medicat	8	1.039				
20	109	69.048179	F	no fracture	72	154	No medicat	11	0.7861				
21	8622	57.802298	F	no fracture	50	152	No medicat	21	0.8254				
22	23843	58.191329	F	no fracture	59	154	No medicat	10	0.8499				
23	4205	46.08155	M	no fracture	73	171	No medicat	37	0.8982				
24	159	65.795352	F	no fracture	74	145	No medicat	14	0.7738				
25	8960	60.189461	F	no fracture	55	153.5	Glucocortic	22	0.8377				
26	23586	54.992184	M	no fracture	75	172	No medicat	64	0.8126				
27	398	51.430717	F	no fracture	67.5	154	Glucocortic	10	0.9234				
28	6732	70.198573	M	no fracture	68	167	No medicat	5	1.040599				
29	688	77.15502	M	no fracture	69	166	Glucocortic	9	1.002				
30	338	70.421873	F	no fracture	55	162	Anticonvuls	15	0.625				

The overview of the original dataset is shown in fig 4.1., with its original features:

3. Algorithm - SVM

SVM is incredibly easy to implement and very efficient to train. So, it's always better start with a Regression model as a benchmark and try using more complex algorithms from there on. regression is a statistical analysis method used to predict a data value based on prior observations of a data set.

Code:

```
import pandas as ps
path="C:\\Users\\shiva\\OneDrive\\Desktop\\Predcit fracture or not\\bmd.csv"
data = ps.read_csv(path)
print(data)
print(data.info())

import sklearn    #importing
from sklearn.preprocessing import LabelEncoder # which is used to the encode the values
la_medication=LabelEncoder()                  # which used to transfrom the values of specific column of
company
la_sex=LabelEncoder()
la_fracture=LabelEncoder()

data['medication_n']=la_medication.fit_transform(data['medication'])
data['sex_n']=la_sex.fit_transform(data['sex'])
data['fracture_n']=la_fracture.fit_transform(data['fracture'])

inputs = data.drop(['id','fracture','fracture_n','sex','medication'],columns')
output =
data.drop(['id','age','sex_n','medication_n','weight_kg','height_cm','bmd','waiting_time','sex','medication','fracture'],columns')

print(inputs)
print(output)

import sklearn
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(inputs,output,train_size=0.2)
```

```
from sklearn.preprocessing import StandardScaler # increase the accuracy
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)

from sklearn.svm import SVC # importing the model
model=SVC()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print(y_test)
print(y_pred)

acc=model.score(x_train,y_train)*100
print("acc :",acc)

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print("cm=",cm)

import numpy as np
newinputs = np.array([[57.05,64.0,155.5,18,0.8793,0,0]]) #to cal predict ([[age,salary,gender_n]])
newinputs = sc.transform(newinputs)
res = model.predict(newinputs)
print(res)
```

Output:

```
fracture_n
95      1
119     0
3       1
63      1
164     0
..     ...
9       1
130     0
150     0
8       1
18      1

[136 rows x 1 columns]
acc : 96.96969696969697
cm= [[22 17]
     [10 87]]
[1]
```

Implementing Code in PYQT5:

```

QtCore.QMetaObject.connectSlotsByName(Form)
def frac(self):
    import pandas as ps
    path="C:\\Users\\shiva\\OneDrive\\Desktop\\Predcit fracture or not\\bmd.csv"
    data = ps.read_csv(path)
    print(data)
    print(data.info())

    import sklearn          #importing
    from sklearn.preprocessing import LabelEncoder # which is used to the encode the values
    la_medication=LabelEncoder() # which used to transform the values of specific column of compar
    la_sex=LabelEncoder()
    la_fracture=LabelEncoder()

    data['medication_n']=la_medication.fit_transform(data['medication'])
    data['sex_n']=la_sex.fit_transform(data['sex'])
    data['fracture_n']=la_fracture.fit_transform(data['fracture'])

    inputs = data.drop(['id','fracture','fracture_n','sex','medication'], 'columns')
    output = data.drop(['id','age','sex_n','medication_n','weight_kg','height_cm','bmd','waiting_time','sex','medication']

    print(inputs)
    print(output)

    import sklearn
    from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(inputs,output,train_size=0.2)

    from sklearn.preprocessing import StandardScaler # increase the accuracy
    sc=StandardScaler()
    x_train=sc.fit_transform(x_train)
    x_test=sc.transform(x_test)

    inputage = self.ag.text()
    inputweight_kg = self.wkg.text()
    inputheight_cm = self.hcm.text()
    inputwaiting_time = self.wtm.text()
    inputbmd = self.bmd.text()
    inputmedication_n = self.med.text()
    inputsex_n = self.sex_n.text()

    from sklearn.svm import SVC # importing the model
    model=SVC()

    inputwaiting_time = self.wtm.text()
    inputbmd = self.bmd.text()
    inputmedication_n = self.med.text()
    inputsex_n = self.sex_n.text()

    from sklearn.svm import SVC # importing the model
    model=SVC()
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    print(y_test)
    print(y_pred)

    acc=model.score(x_train,y_train)*100
    print("acc :",acc)

    from sklearn.metrics import confusion_matrix
    cm=confusion_matrix(y_test,y_pred)
    print("cm=",cm)

    import numpy as np
    newinputs = np.array([float(inputage),float(inputweight_kg),float(inputheight_cm),int(inputwaiting_time),float(inputbmd),float(inputmedication_n),float(inputsex_n)])
    res = model.predict(newinputs)
    print(res)

    self.fran.setText(str(res))

def retranslateUi(self, Form):
    _translate = QtCore.QCoreApplication.translate
    Form.setWindowTitle(_translate("Form", "Form"))
    self.label.setText(_translate("Form", " FRACTURE PREDICTION"))
    self.label_2.setText(_translate("Form", "age :"))
    self.label_3.setText(_translate("Form", "weight_kg :"))
    self.label_4.setText(_translate("Form", "height_cm :"))
    self.label_5.setText(_translate("Form", "waiting_time :"))
    self.label_6.setText(_translate("Form", "bmd :"))
    self.label_7.setText(_translate("Form", "medication_n :"))
    self.label_8.setText(_translate("Form", "fracture_n :"))
    self.label_9.setText(_translate("Form", "sex_n :"))
    self.button.setText(_translate("Form", "PREDICT"))
    self.textEdit.setText(_translate("Form", "<DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/1
    <html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n
    \"p, li { white-space: pre-wrap; }\n
    \"</style></head><body style=\" font-family:'MV Boli'; font-size:8pt; font-weight:400; font-style:normal;\n
    \">\n
    \"<p align=\"justify\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\n
    \"><p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\n
    \"><p style=\" -qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\n
    \"></p></body></html>

```

Fig 5.3

PYQT5 Output:

The figure displays two screenshots of a PyQt5 application window titled "Form". The window contains a "FRACTURE PREDICTION" form with the following input fields and a "PREDICT" button:

- age :
- weight_kg :
- height_cm :
- waiting_time :
- bmd :
- medication_n :
- sex_n :
- PREDICT**
- fracture_n :
FOR FRACTURED = 1
FOR NOT FRACTURED = 0

The top screenshot shows the form with empty input fields. The bottom screenshot shows the form with the following values entered:

- age : 55
- weight_kg : 64
- height_cm : 132
- waiting_time : 4
- bmd : 0.2
- medication_n : 2
- sex_n : 1
- PREDICT**
- fracture_n : [1]
FOR FRACTURED = 1
FOR NOT FRACTURED = 0

Fig 5.4

5.5 CONCLUSION

The internship aims to use Python programming language for Machine Learning so as to apply the theoretical knowledge to solve real-time and complex problems. The internship helped to find appropriate prediction model to the problems by applying suitable learning algorithm which can be used in future. The internship project assigned by the company helped me to improve my programming skills for solving real world problem. In this project **SVM Algorithm** is used to predict whether the a particular patient has malignant or benign fracture. After having a basic understanding of Supervised learning, we explored the SVM Algorithm which is used to solve machine learning problem.

CHAPTER 6

Loan Prediction

In this project, we will develop and evaluate the performance and the predictive power of a model trained and tested on data collected from magazine dataset. Once we get a good fit, we will use this model to predict whether a person buys a magazine or not. A model like this would be very valuable.

1. Problem Statement

Develop a model that has the capacity of predicting to predict whether a person buys a magazine or not by making use of the information provided in a dataset.

2. Dataset

Data columns (total 13 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	----
0	Loan_ID	614 non-null	object
1	Gender	601 non-null	object
2	Married	611 non-null	object
3	Dependents	599 non-null	object
4	Education	614 non-null	object
5	Self_Employed	582 non-null	object
6	ApplicantIncome	614 non-null	int64
7	CoapplicantIncome	614 non-null	float64
8	LoanAmount	592 non-null	float64
9	Loan_Amount_Term	600 non-null	float64
10	Credit_History	564 non-null	float64
11	Property_Area	614 non-null	object
12	Loan_Status	614 non-null	object

The overview of the original dataset is as shown in fig 5.1., with its original features:

Fig 6.1 Overview of Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360		1 Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360		1 Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360		1 Urban	Y
5	LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360		1 Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360		1 Urban	Y
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360		1 Urban	Y
8	LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360		1 Urban	Y
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360		0 Semiurban	N
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360		1 Urban	Y
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360		1 Semiurban	N
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360		1 Urban	Y
13	LP001027	Male	Yes	2	Graduate		2500	1840	109	360		1 Urban	Y
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360		1 Urban	Y
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360		1 Rural	N
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120		1 Urban	Y
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360		1 Urban	Y
18	LP001034	Male	No	1	Not Graduate	No	3596	0	100	240		Urban	Y
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360		0 Urban	N
20	LP001038	Male	Yes	0	Not Graduate	No	4887	0	133	360		1 Rural	N
21	LP001041	Male	Yes	0	Graduate		2600	3500	115			1 Urban	Y
22	LP001043	Male	Yes	0	Not Graduate	No	7660	0	104	360		0 Urban	N
23	LP001046	Male	Yes	1	Graduate	No	5955	5625	315	360		1 Urban	Y
24	LP001047	Male	Yes	0	Not Graduate	No	2600	1911	116	360		0 Semiurban	N

Code:

```
import pandas as ps
path="C:\\Users\\shiva\\OneDrive\\Desktop\\loanpredictiondataset\\train.csv"
data=ps.read_csv(path)

print(data)
print(data.info())

medianval = data.LoanAmount.median() # to change the value of NaN
print(medianval)
data['LoanAmount']= data.LoanAmount.fillna(data.LoanAmount.median())

medianval = data.Credit_History.median() # to change the value of NaN
print(medianval)
data['Credit_History']= data.Credit_History.fillna(data.Credit_History.median())

medianval = data.Loan_Amount_Term.median() # to change the value of NaN
print(medianval)
data['Loan_Amount_Term']= data.Loan_Amount_Term.fillna(data.Loan_Amount_Term.median())

import sklearn #importing
from sklearn.preprocessing import LabelEncoder # which is used to the encode the values
la_Property_Area=LabelEncoder() # which used to transfrom the values of specific column of company
la_Loan_Status=LabelEncoder()
la_Gender=LabelEncoder()
la_Married=LabelEncoder()
la_Education=LabelEncoder()
la_Self_Employed=LabelEncoder()
```

```
data['Property_Area_n']=la_Property_Area.fit_transform(data['Property_Area'])
data['Loan_Status_n']=la_Property_Area.fit_transform(data['Loan_Status'])
data['Gender_n']=la_Property_Area.fit_transform(data['Gender'])
data['Education_n']=la_Property_Area.fit_transform(data['Education'])
data['Self_Employed_n']=la_Property_Area.fit_transform(data['Self_Employed'])
data['Married_n']=la_Property_Area.fit_transform(data['Married'])

inputs =
data.drop(['Loan_ID','Loan_Status','Loan_Status_n','Property_Area','Gender','Education','Self_Employed','Married','Dependents'],columns')
output =
data.drop(['Married_n','Self_Employed_n','Education_n','Gender_n','Property_Area_n','Loan_Status','Loan_ID','Gender','Married','Dependents','Education','Self_Employed','ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','Property_Area'],columns')
print("inputs:",inputs)
print("output:",output)

import sklearn
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(inputs,output,test_size=0.2)
print("x_train :",x_train)
print("x_test:",x_test)
print("y_train:",y_train)
print("y_test:",y_test)

from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=1)
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
print("y_pred:",y_pred)
print("y_test:",y_test)

from sklearn.metrics import confusion_matrix    #using confusion matrix the acc is calculated
cm=confusion_matrix(y_test,y_pred)
print("cm:",cm)

acc=model.score(inputs,output)*100
print("acc :",acc)
```

Output:

	Loan_Status_n
441	1
11	1
492	1
457	0
306	1

```
[123 rows x 1 columns]
cm: [[14 25]
      [26 58]]
acc : 91.69381107491856
```

```

self.retranslateUi(Form)
QtCore.QMetaObject.connectSlotsByName(Form)
def lon(self):
    import pandas as ps
    path="C:\\Users\\shiva\\OneDrive\\Desktop\\loanpredictiondataset\\train.csv"
    data=ps.read_csv(path)

    print(data)
    print(data.info())

    medianval = data.LoanAmount.median() # to change the value of NaN
    print(medianval)
    data['LoanAmount'] = data.LoanAmount.fillna(data.LoanAmount.median())

    medianval = data.Credit_History.median() # to change the value of NaN
    print(medianval)
    data['Credit_History'] = data.Credit_History.fillna(data.Credit_History.median())

    medianval = data.Loan_Amount_Term.median() # to change the value of NaN
    print(medianval)
    data['Loan_Amount_Term'] = data.Loan_Amount_Term.fillna(data.Loan_Amount_Term.median())

    import sklearn #importing
    from sklearn.preprocessing import LabelEncoder # which is used to the encode the values
    la_Property_Area=LabelEncoder() # which used to transform the values of specific column of com
    la_Loan_Status=LabelEncoder()
    la_Gender=LabelEncoder()
    la_Married=LabelEncoder()
    la_Education=LabelEncoder()
    la_Self_Employed=LabelEncoder()

    data['Self_Employed_n']=la_Property_Area.fit_transform(data['Self_Employed'])
    data['Married_n']=la_Property_Area.fit_transform(data['Married'])

    inputs = data.drop(['Loan_ID', 'Loan_Status', 'Loan_Status_n', 'Property_Area', 'Gender', 'Education', 'Self_Employed', 'Mar
    output = data.drop(['Married_n', 'Self_Employed_n', 'Education_n', 'Gender_n', 'Property_Area_n', 'Loan_Status', 'Loan_ID',
    print("inputs:",inputs)
    print("output:",output)

    inputApplicantIncome = self.ain.text()
    inputCoapplicantIncome = self.cin.text()
    inputLoanAmount = self.lam.text()
    inputLoan_Amount_Term = self.lamt.text()
    inputCredit_History = self.ch.text()
    inputProperty_Area_n = self.pan.text()
    inputGender_n = self.gn.text()
    inputEducation_n = self.edn.text()
    inputSelf_Employed_n = self.sen.text()
    inputMarried_n = self.mn.text()

    import sklearn
    from sklearn.model_selection import train_test_split

    x_train,x_test,y_train,y_test = train_test_split(inputs,output,test_size=0.2)
    print("x_train :",x_train)
    print("x_test:",x_test)
    print("y_train:",y_train)
    print("y_test:",y_test)

    from sklearn.neighbors import KNeighborsClassifier
    model = KNeighborsClassifier(n_neighbors=10)
    model.fit(x_train,y_train)
    y_pred = model.predict(x_test)
    res=model.predict([int(inputApplicantIncome),float(inputCoapplicantIncome),float(inputLoanAmount),float(inputLoan_Am
    print("res :",res)

    self.loss.setText(str(res))

```

Fig 6.2 PYQT Output

6.3 CONCLUSION

The internship aims to use Python programming language for Machine Learning so as to apply the theoretical knowledge to solve real-time and complex problems. The internship helps to integrate corporate experience in college life. The internship helped to find appropriate prediction model to the problems by applying suitable learning algorithm which can be used in future. The internship project assigned by the company helped to improve programming skills and to implement basic knowledge for solving real world problem like White Wine Quality prediction which helped to use appropriate Machine Learning algorithm such as SVM algorithm for building a model that can be used to predict whether a person buys a magazine or not using Python.

CHAPTER 7

Project -4 Description

Prediction of Hire Or Not

In this project, we will develop and evaluate performance and the predictive salary of a model trained and tested on data collected from salary dataset. Once we get a good fit, we will use this model to predict if the salary is greater than or less than and equal to 50K.

1. Problem Statement

Develop a model that has the capacity of predicting the income by making use the information provided in salary dataset.

2. Dataset

Data columns (total 5 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	----
0	Status	30 non-null	int64
1	Interview	30 non-null	int64
2	CGPA	30 non-null	int64
3	Written	30 non-null	int64
4	Aptitude	30 non-null	int64

Fig 7.1 Overview of Dataset

○ **Algorithm - Logistic Regression**

Logistic Regression is such a method and, despite its simplicity, continues to perform fairly well for large training sets. It essentially relies only on the most basic assumption underlying all prediction: that observation with similar characteristics will tend to have similar outcomes.

○ **Programming Steps**

- This project requires us to predict the salary based on the given input dataset.
- First, we read the given dataset using pandas function.
- Then we print the inputs and output from csv file.
- We initialize the model i.e., **Logistic Algorithm**.
- Then we initialize inputs in order to train and test the model for accuracy and make improvements after observing the output.
- Since it is Logistic Algorithm, we can check the accuracy by using the confusion matrix.
- We further implement this using Pyqt in order for better representation.

Code:

```
import pandas as ps
path="C:\\Users\\shiva\\OneDrive\\Desktop\\Hired or not\\hiredornot.csv"
data=ps.read_csv(path)
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

print(data)
print(data.info())

inputs = data.drop('Status','columns')
output = data.drop(['Interview','CGPA','Written','Aptitude'],'columns')
print(inputs)
print(output)

x_train,x_test,y_train,y_test=train_test_split(inputs,output,test_size=0.5)
print("x_train",x_train)
print("x_test",x_test)
print("y_train",y_train)
print("y_test",y_test)

model=LogisticRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print(y_pred)
print(y_test)
res=model.predict([[80,10,79,68]])
print(res)
```

```
from sklearn.metrics import confusion_matrix    #using confusion matrix the acc is calculated
cm=confusion_matrix(y_test,y_pred)
print(cm)

acc=model.score(inputs,output)*100
print("acc :",acc)
```

Output:

```
      Status
3      1
11     0
15     0
5      0
22     0
23     0
18     0
10     0
8      0
0      0
28     1
2      0
24     0
27     1
25     0
[1]
[[11  1]
 [ 3  0]]
acc : 73.33333333333333
```

Implementing Code in PYQT5:

```

QtCore.QMetaObject.connectSlotsByName(Form)
def hr(self):
    import pandas as ps
    path="C:\\Users\\shiva\\OneDrive\\Desktop\\Hired or not\\hiredornot.csv"
    data=ps.read_csv(path)
    import sklearn
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression

    print(data)
    print(data.info())

    inputs = data.drop('Status','columns')
    output = data.drop(['Interview','CGPA','Written','Aptitude'],'columns')
    print(inputs)
    print(output)

    inputInterview = self.int_2.text()
    inputCGPA = self.cgpa.text()
    inputWritten = self.writ.text()
    inputAptitude = self.appt.text()

    x_train,x_test,y_train,y_test=train_test_split(inputs,output,test_size=0.5)
    print("x_train",x_train)
    print("x_test",x_test)
    print("y_train",y_train)
    print("y_test",y_test)
    model=LogisticRegression()
    model.fit(inputs,output)
    y_pred=model.predict(x_test)
    print(y_pred)
    print(y_test)
    res=model.predict([[80,10,79,69]])
    print(res)

    self.hon.setText(str(res))

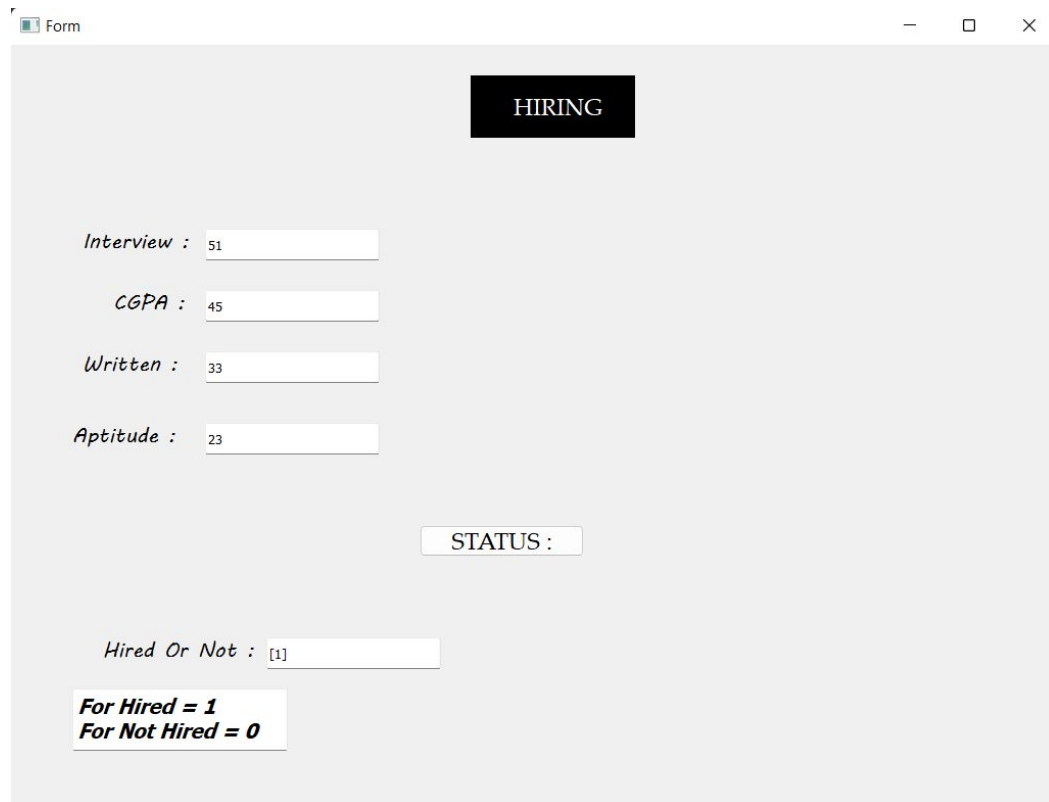
    from sklearn.metrics import confusion_matrix    #using confusion matrix the acc is calculated
    cm=confusion_matrix(y_test,y_pred)
    print(cm)

    acc=model.score(inputs,output)*100
    print("acc :",acc)

```

Fig 7.3

PYQT5 Output:



The screenshot shows a web application window titled "Form". At the top center is a black button with the word "HIRING" in white. Below this, there are four input fields, each preceded by a label: "Interview : 51", "CGPA : 45", "Written : 33", and "Aptitude : 23". The numbers 51, 45, 33, and 23 are already entered in the respective fields. Below these fields is a label "STATUS :". Further down is an output field labeled "Hired Or Not : [1]". At the bottom left, there is a legend box containing the text "For Hired = 1" and "For Not Hired = 0".

Fig 7.4

7.5 CONCLUSION

The internship aims to use Python programming language for Machine Learning so as to apply the theoretical knowledge to solve real-time and complex problems. The internship helped to find appropriate prediction model to the problems by applying suitable learning algorithm which can be used in future. The internship project assigned by the company helped to improve programming skills and to implement basic knowledge for solving real world problem. In this project **Logistic Algorithm** is used to predict the **Hire**. After having a basic understanding of Supervised learning, we explored the KNN Algorithm which is used to solve machine learning problem. The Income varies based on the inputs given in the salary dataset.

REFERENCES

- 1 www.karunadutechnologies.com
- 2 <https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know- article>
- 3 <https://www.edureka.co/blog/machine-learning-algorithms/>
- 4 <https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>