

**DR. A. Q. KHAN INSTITUTE OF COMPUTER
SCIENCES
AND INFORMATION TECHNOLOGY
KAHUTA**



BACHELOR OF SCIENCE IN COMPUTER ENGINEERING

LAB MANUAL

Artificial Intelligence

Lab Instructor: Engr.Tajummal Hussain

Version 1.0

CLO & PLO Mapping

COURSE LEARNING OUTCOMES:

Upon successful completion of the course, the student will be able to:

Sr. #	Artificial Intelligence (Lab)	Bloom's Taxonomy Level	PLO
1	<i>Practice Basic concept of python programming language to analyze and solve complex problems</i>	P3	2
2	<i>Make Artificial Intelligence based models on different datasets to produce efficient results on real world problems</i>	P4	3
3	<i>Practice different Machine learning models and data preprocessing technique on (Anaconda Distribution) using advance data processing and data learning libraries</i>	P3	5
4	<i>Classify objects on the basis of object classifying and recognition techniques with human interaction</i>	A4	10

Mapping between PLO VS CLO:

PLO No. CLO No.	2	3	5	10
1	✓			
2		✓		
3			✓	
4				✓

LIST OF EXPERIMENTS

Sr. #	LAB EXPERIMENTS	CLO
Lab 1	<i>Introduction about Artificial Intelligence & Application Environment Setup:</i> Try to Install Anaconda software on system	1
Lab 2	<i>Python Programming-I:</i> Practice Basics of python string, input statements, decision making statement, list, dictionary and tuples	1
Lab 3	<i>Python Programming-II:</i> Practice: Basics of python loops functions and lambda functions.	1
Lab 4	<i>Introduction about Numpy:</i> Practice Numpy function	1
Lab 5	<i>Introduction to Pandas:</i> Practice Basic Pandas Function	1
Lab 6	<i>Introduction to Matplotlib and Seaborn:</i> Practice Basics of Matplotlib and Seaborn	1
Lab 7	<i>Linear Regression:</i> Practice and make linear regression Algorithm	2,3
Lab 8	<i>Logistic Regression:</i> Practice and make Logistic Regression Algorithm	2,3
Lab 9	<i>Support Vector Machine:</i> Practice and make support vector machines on the given dataset	2,3
Lab 10	<i>Random Forest and Decision Trees:</i> Classify a dataset by implementing random forest and decision trees	4
Lab 11	<i>K-Means and DBSCAN:</i> Classify a dataset by implementing K-Means and dbscan	4
Lab 12	<i>Open Cv:</i> Practice basic opencv functions	1
Lab 13	<i>Naive Baye's Algorithm:</i>	2

	Make and implement Naïve Baye's algorithm	
Lab 14	Implementing Machine learning Algorithm on Diabetes set: Make and implement 6 to 8 Algorithms on diabetes dataset	2

LAB 01

(CLO 1-3, PLO 1-5,P3)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	<i>Introduction about Artificial Intelligence & Application</i>
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	<i>Environment Setup:</i> Try to Install Anaconda software on system

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Installing Anaconda & Applications of AI

Task no. 1:

Download & install Anaconda on windows operating system and explain installation process with screenshots.

Solution:**Introduction:****What is Anaconda Distributions?**

Anaconda is a freemium open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system conda. With over 4.5 million users, the open source Anaconda Distribution is the easiest way to do Python data science and machine learning. It includes hundreds of popular data science packages and the conda package and virtual environment manager for Windows, Linux, and MacOS. Conda makes it quick and easy to install, run, and upgrade complex data science and machine learning environments like scikit-learn, TensorFlow, and SciPy. What is Anaconda Distribution

How it is different from Python Distribution?

Anaconda Data Science Libraries

Over 1,000 Anaconda-curated and community data science packages

Develop data science projects using your favorite IDEs, including Jupyter, JupyterLab, Spyder and RStudio

Analyze data with scalability and performance with Dask, numpy, pandas and Numba
Visualize your data with Bokeh, Datashader, Holoviews or Matplotlib

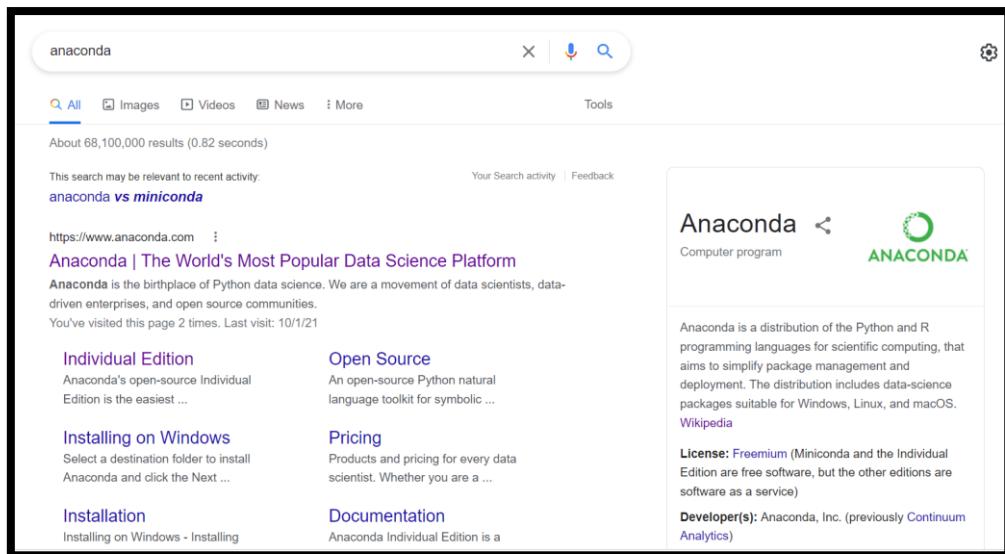
Create machine learning and deep learning models with Scikit-learn, Tensorflow, h2o and theano

Download & install Anaconda on windows operating system:

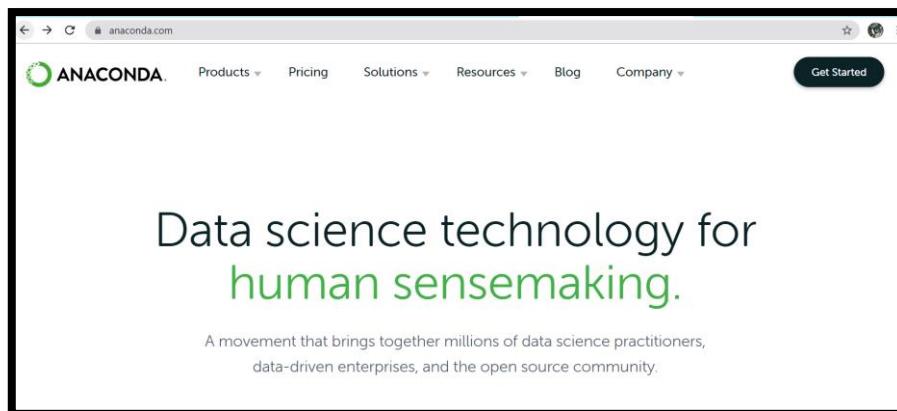
Open your browser

Search for Anaconda

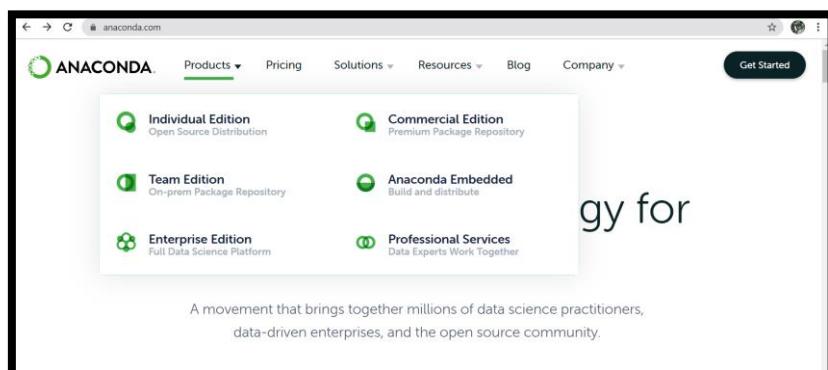
There you will see the offical website <https://www.anaconda.com/>



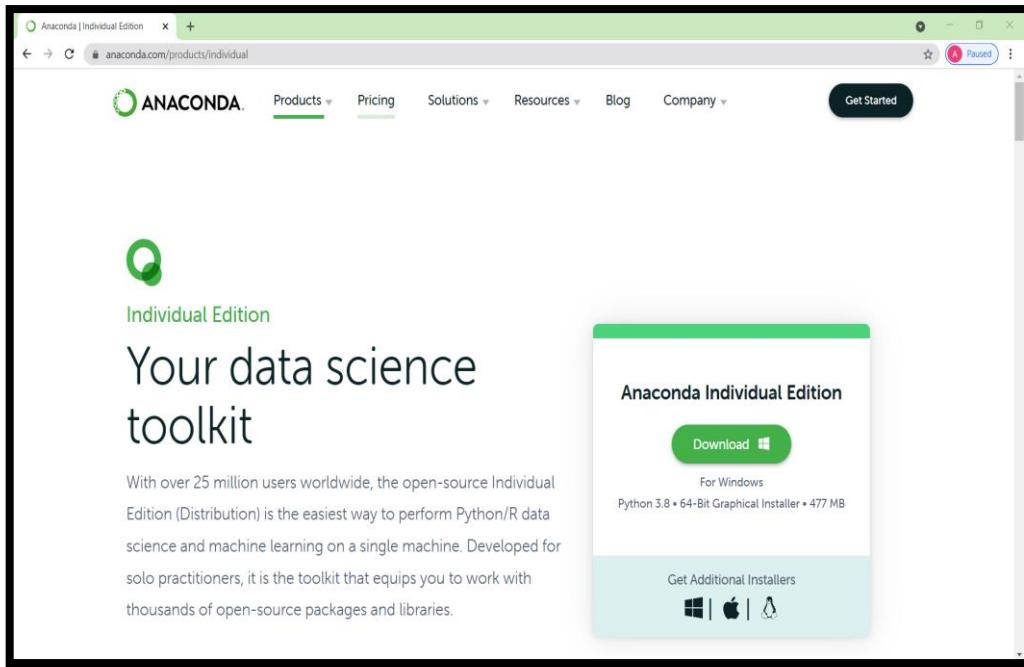
Open the official link



Go in the product section and select the individual edition as we want to use it only for an individual



Afterwards you'll see this window in front of you



It has various options for different operating systems with different 3 and 6 bit options

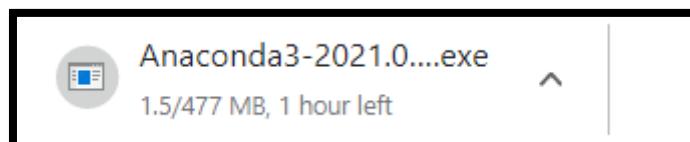
To install it on Windows choose the desired option

My system is 64 bit and it will be compatible with the 64-bit one

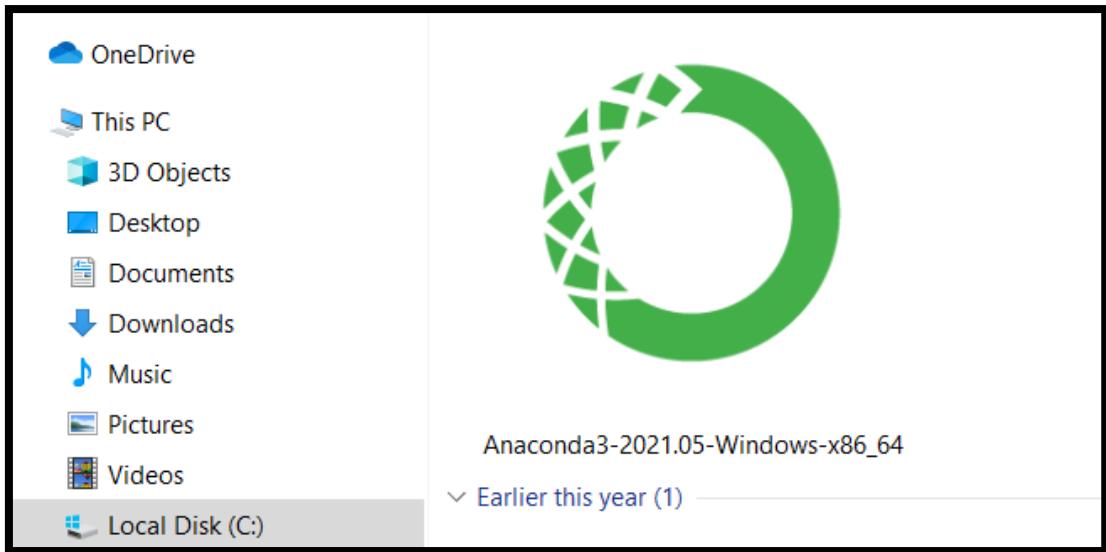
Click on the desired option



After clicking downloading will began i.e. Exe file



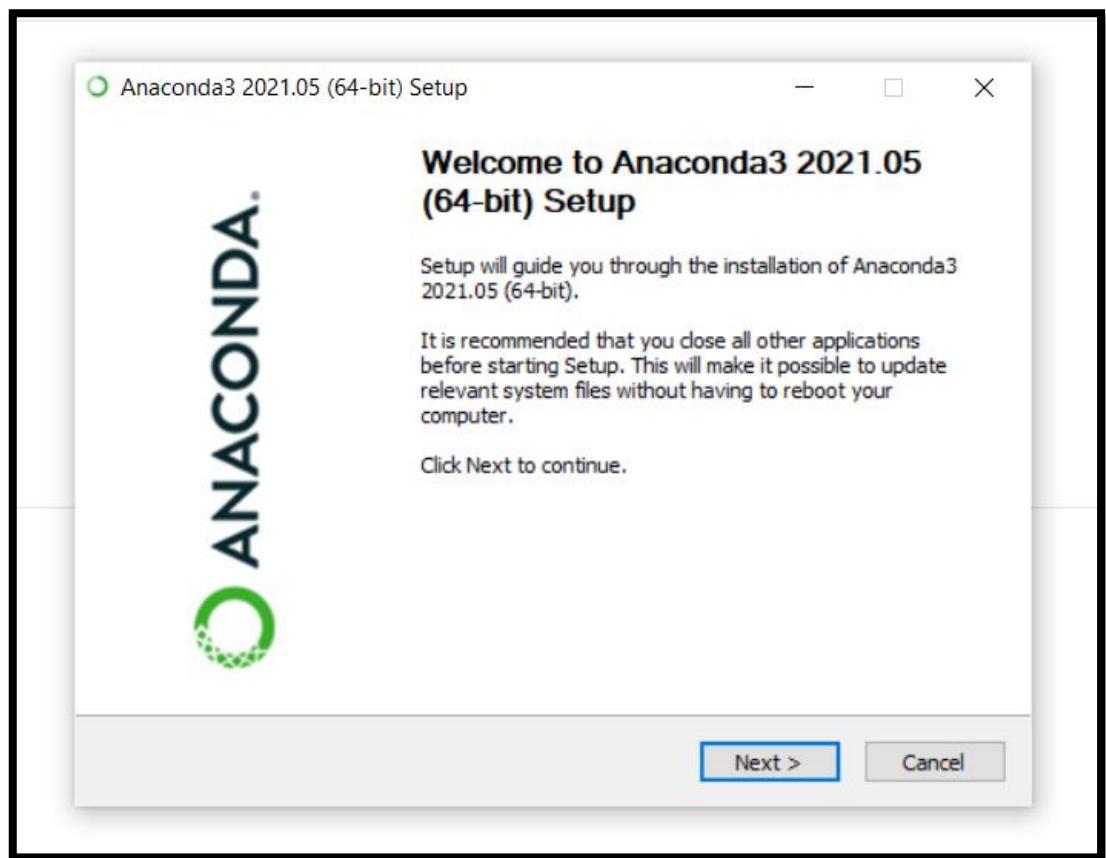
Here we can see the setup has been downloaded successfully on the PC



Open the downloaded file

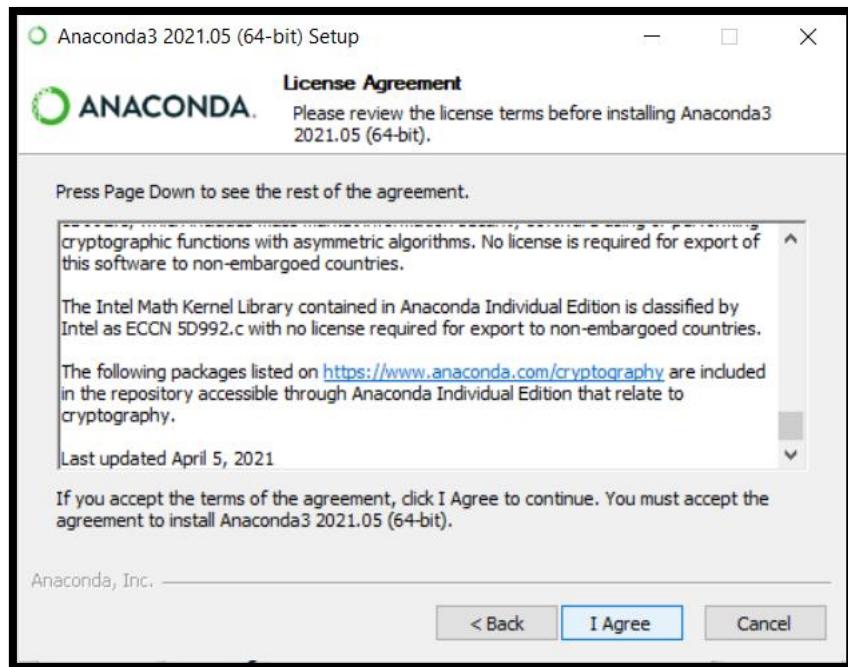
Next what you'll see will be this window

Click next from here to continue



Next the license agreement will appear in front of you

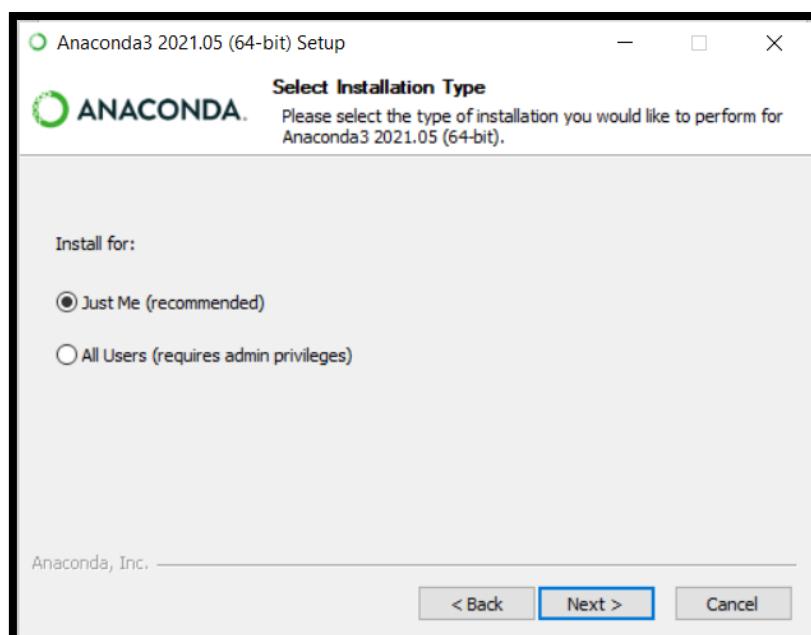
Read the agreement and that click I agree if you have no issues from the information provided above



After agreeing with the license agreement you have to select installation type

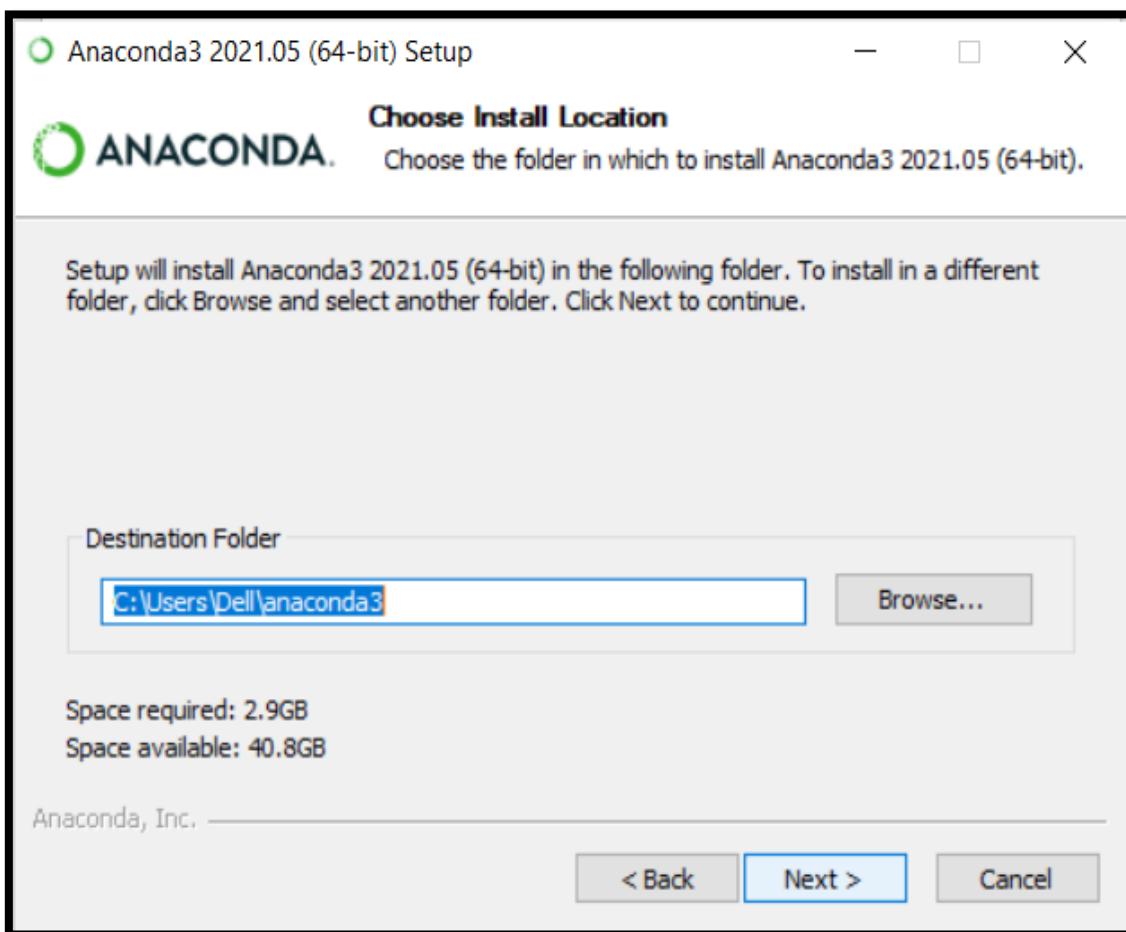
If you want to use it only for yourself as I want to select just me and vice versa

Then click Next



Choose the location for the installation of the setup on your PC

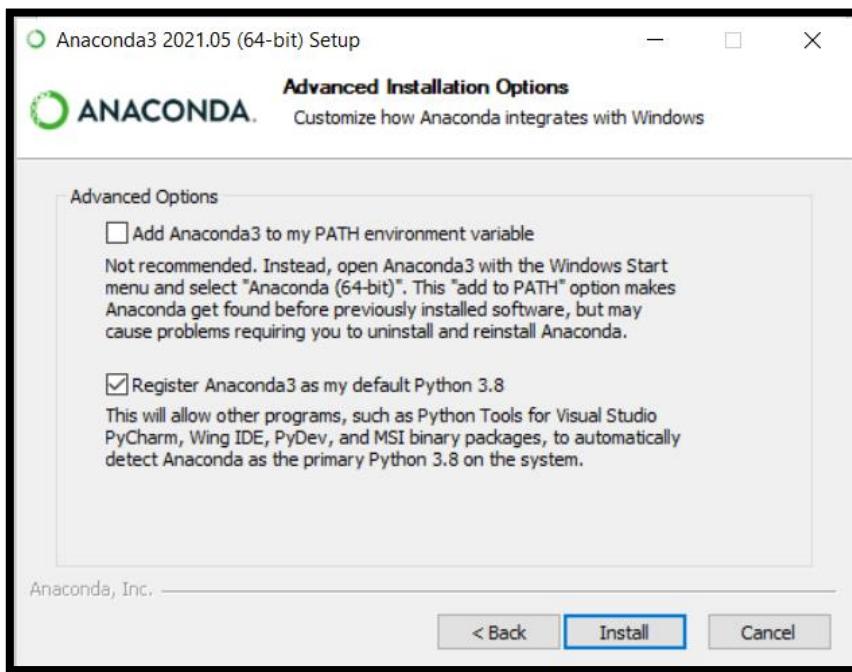
After choosing the location click next



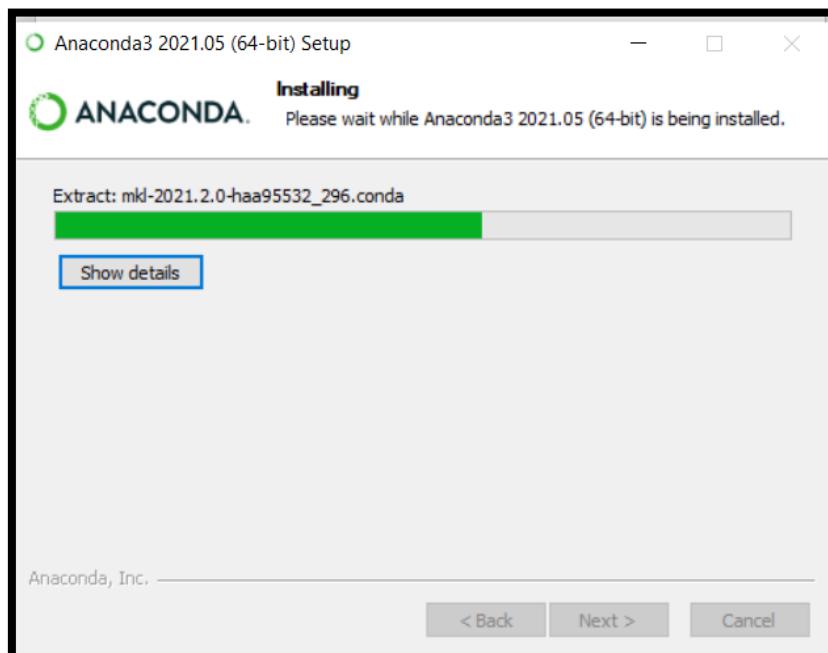
After selecting the location you'll see the advanced installation options

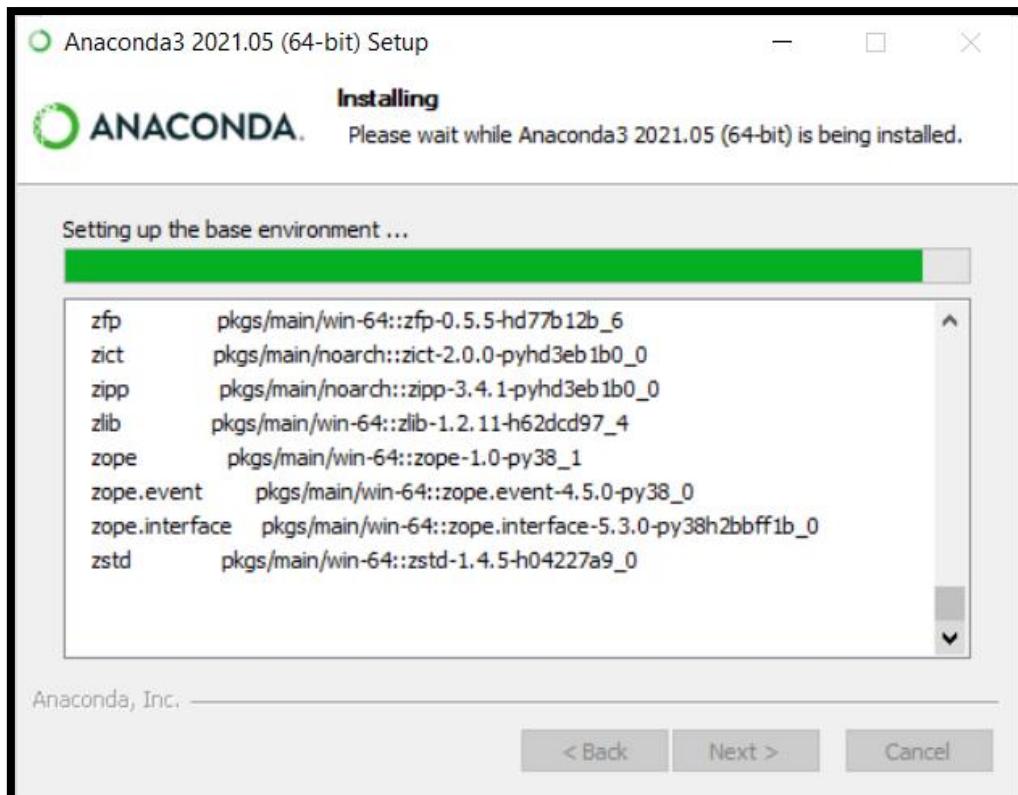
Make sure that the desired option has been selected . Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, accept the default and leave this box checked.

Than click install

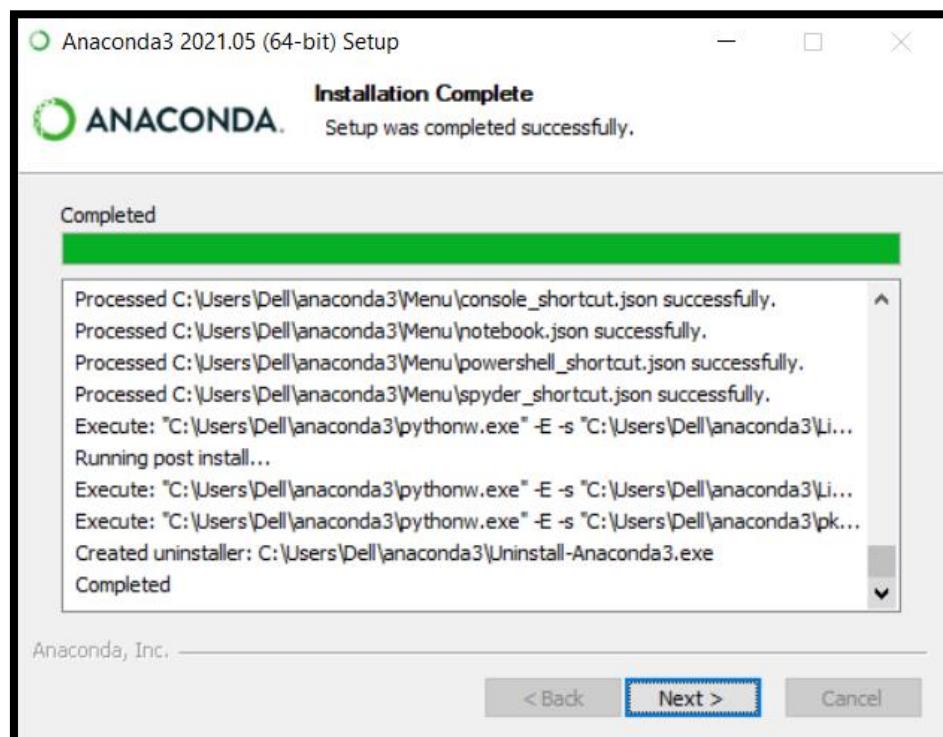


Installation will began

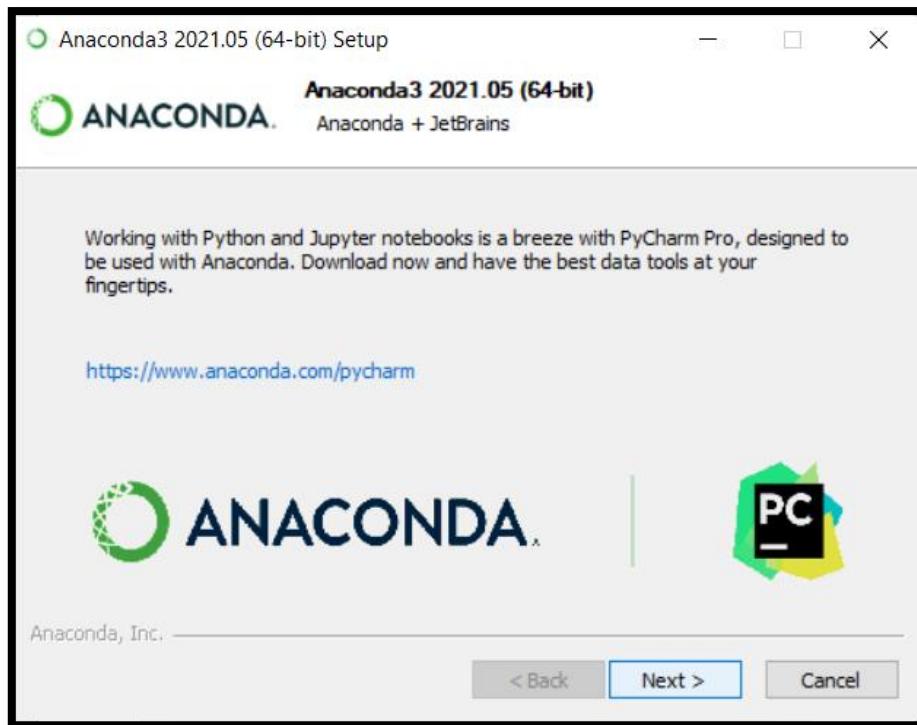




Once the installation has been completed Click next

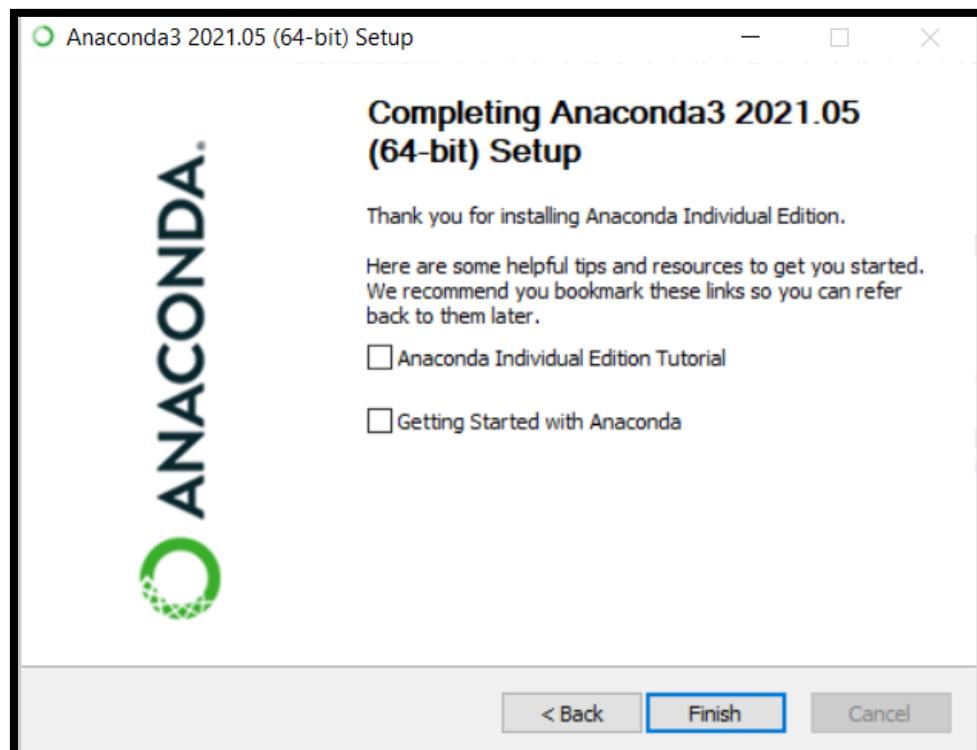


Click Next after reading the instructions in order to continue



I have un checked the boxes but if you want to watch the tutorial or start anaconda you can leave these boxes checked

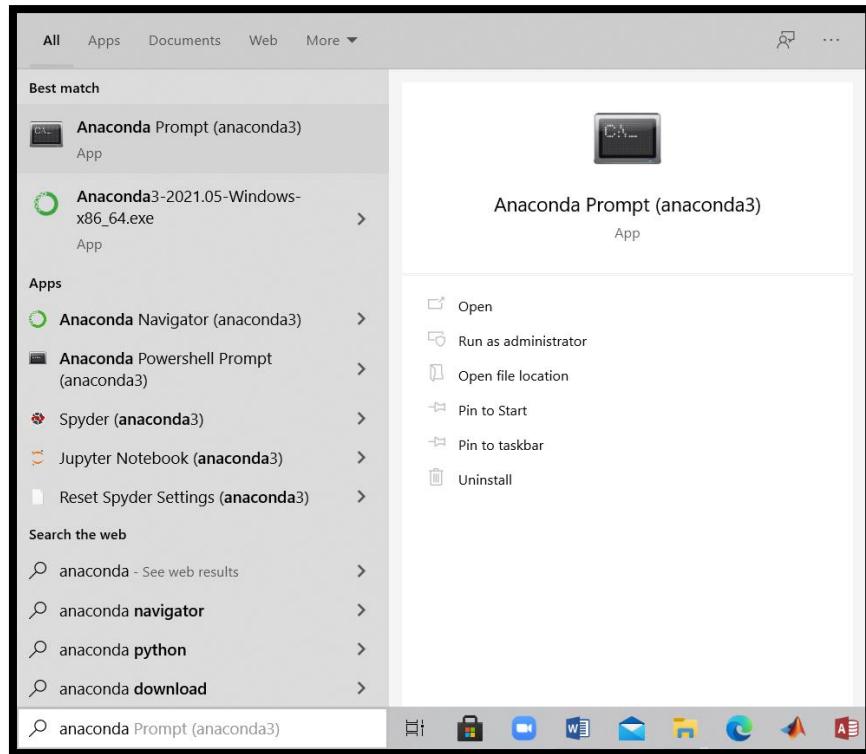
After wards click finish



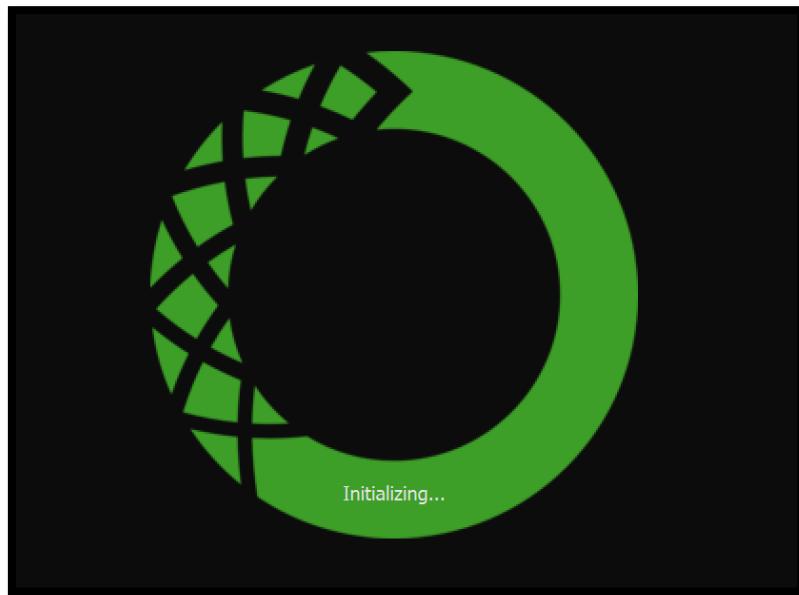
Go on the search bar of the PC and search for Anaconda

There you will see various anaconda option as shown below

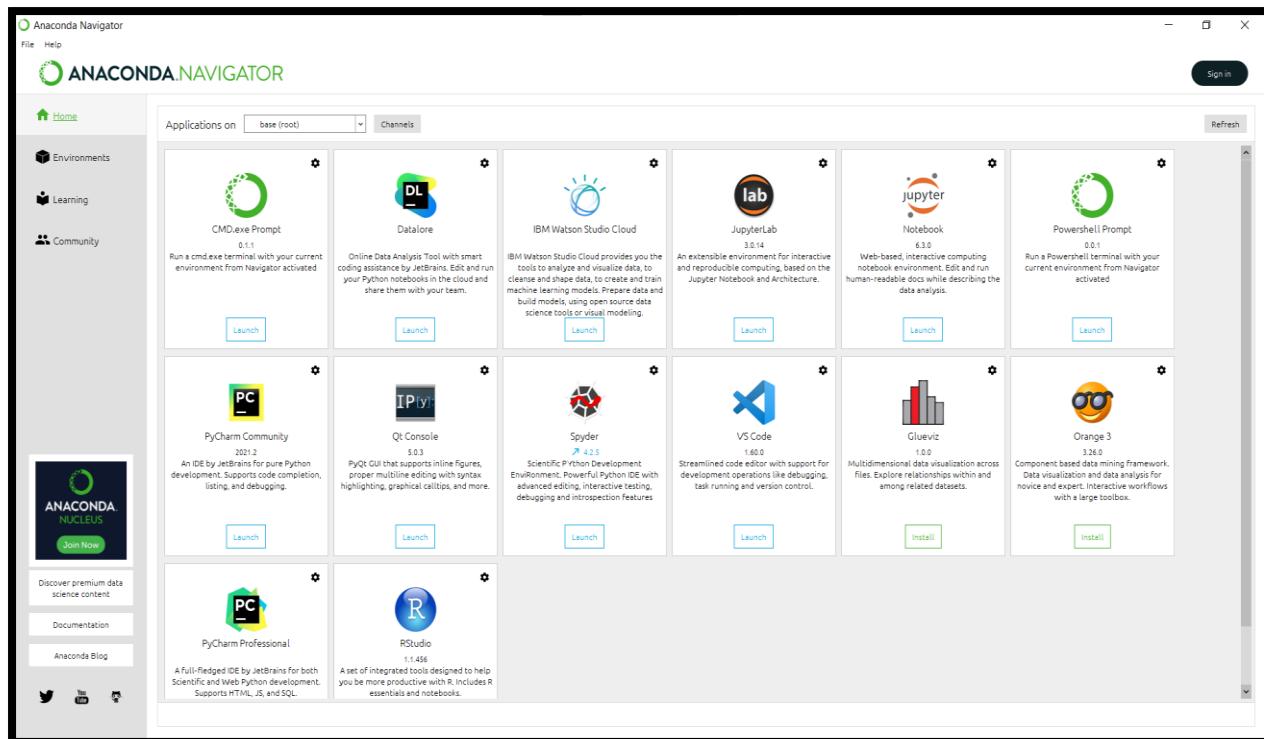
Select Anaconda Navigator



Anaconda will start initializing

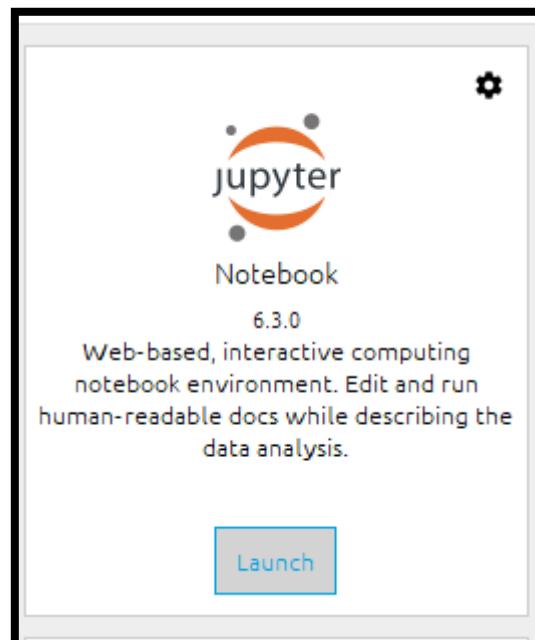


Then you'll see the anaconda environment with warious IDEs such as jupyter notebook,pycharm,spyder etc



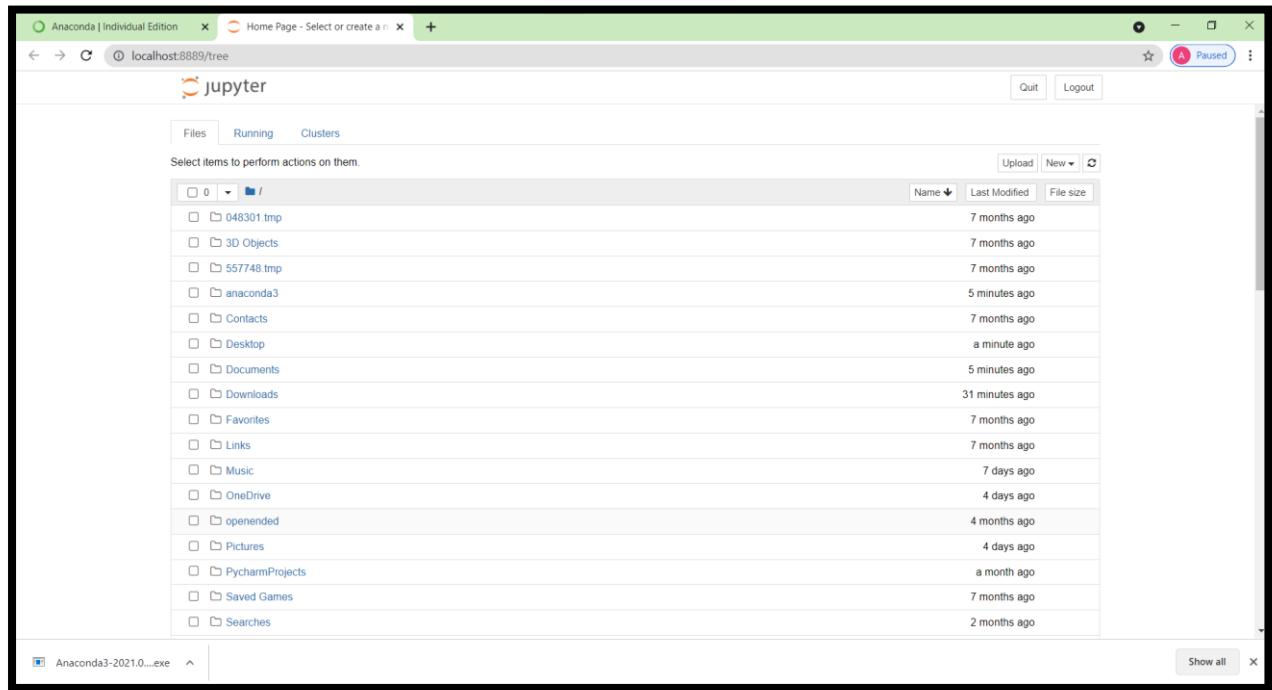
For printing Your Name:

Launch the Jupyter notebook

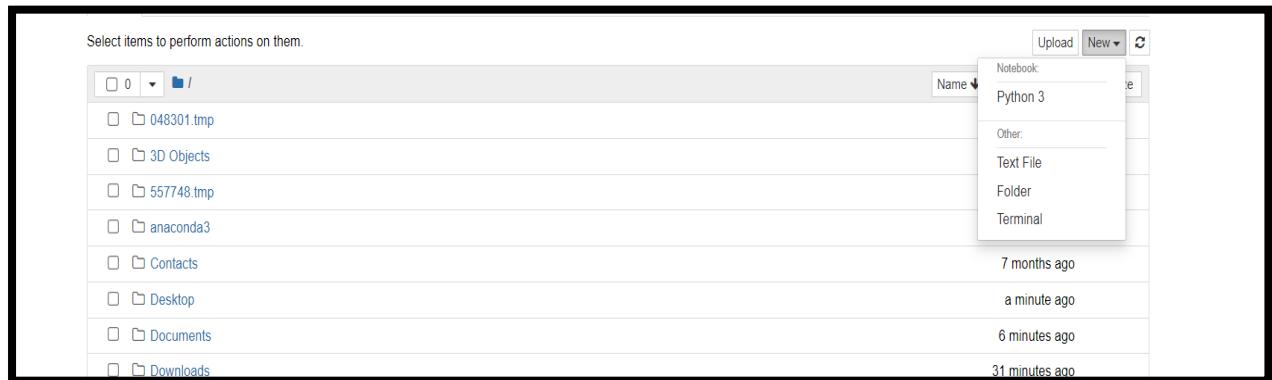


Launching notebook

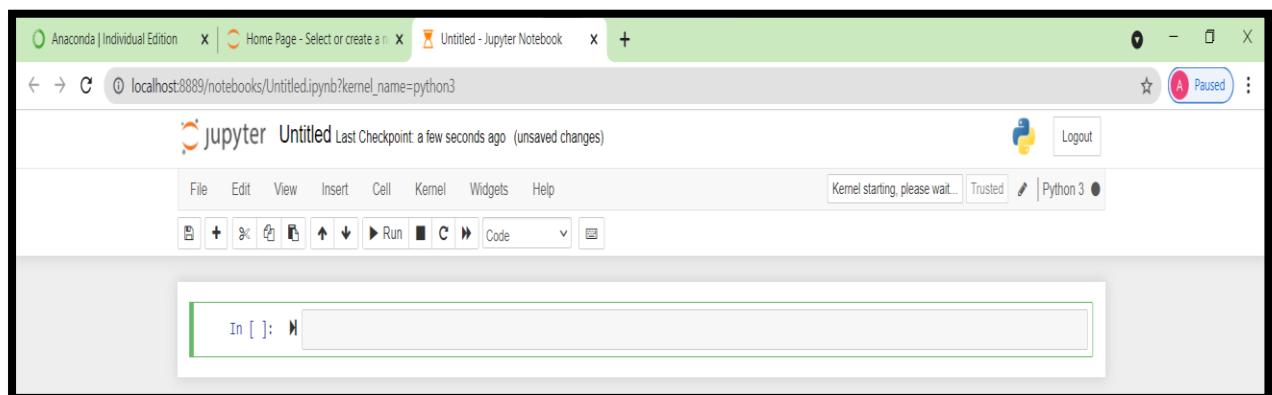
After launching you'll see this window



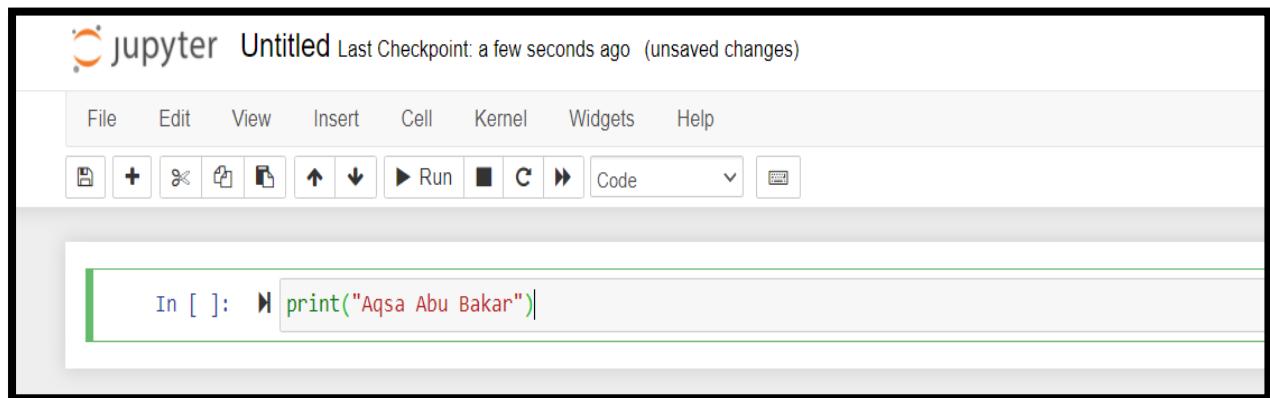
Click new and select python 3 to open jupyter notebook's file



You'll see this type of environment

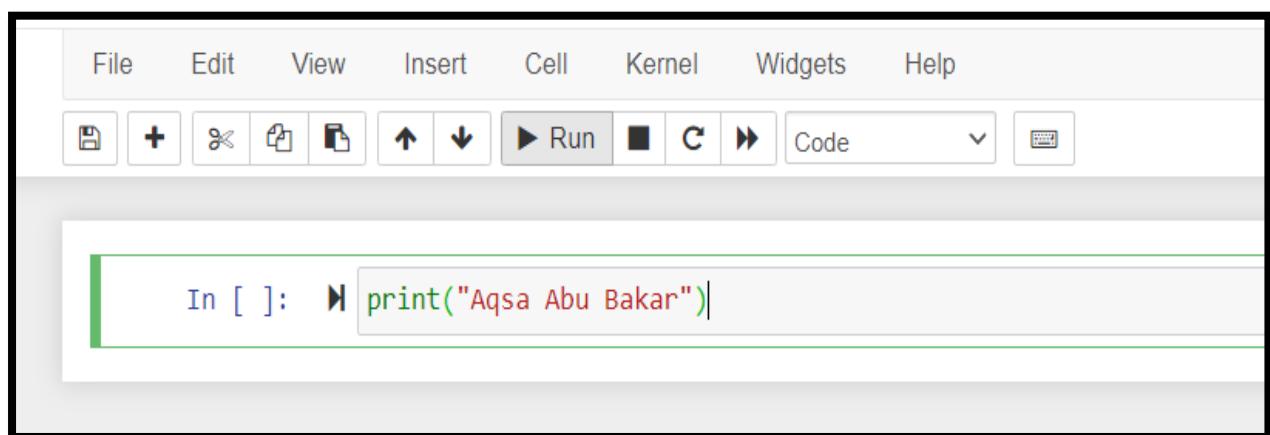


Write the command to be printed as print (“Your name”)

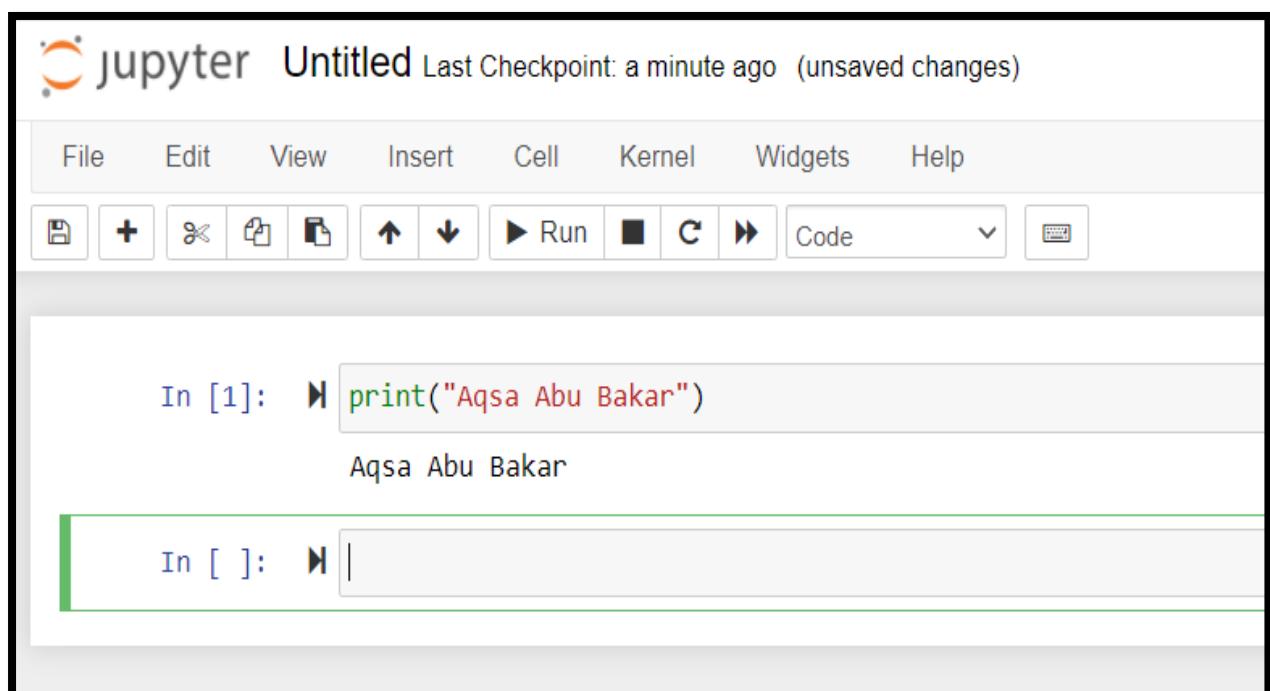


A screenshot of a Jupyter Notebook interface. The title bar says "jupyter Untitled Last Checkpoint: a few seconds ago (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations like save, new, delete, and run, along with a "Code" dropdown. A code cell is active, indicated by a green border. The cell contains the Python code: `In []: print("Aqsa Abu Bakar")`.

Click Run to execute your instruction

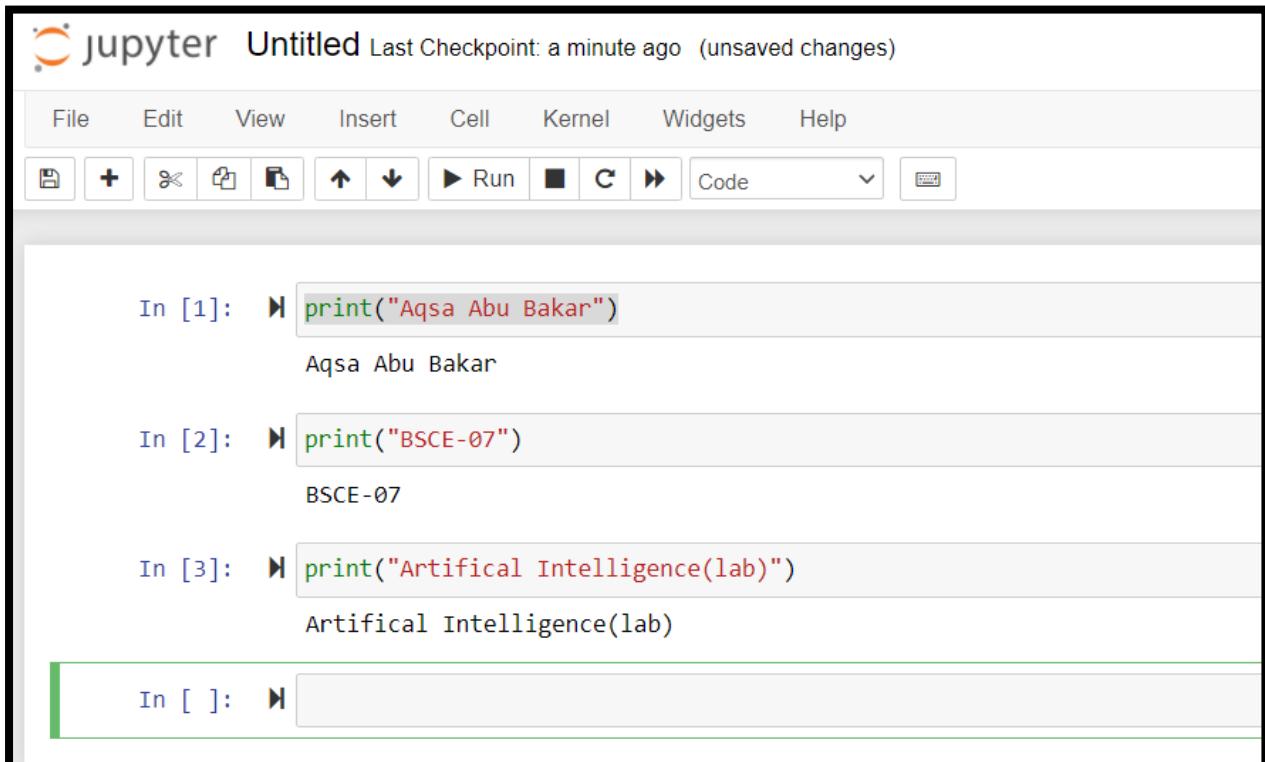


A screenshot of a Jupyter Notebook interface, similar to the one above. It shows a code cell with the same Python code: `In []: print("Aqsa Abu Bakar")`. The "Run" button in the toolbar is highlighted.



A screenshot of a Jupyter Notebook interface showing the execution results. The code cell from the previous screenshots has been run. The output shows the text "Aqsa Abu Bakar" printed below the code. A new code cell is visible at the bottom, indicated by a green border and the text "In []:".

Some more print statements



The screenshot shows a Jupyter Notebook interface with the title "jupyter Untitled Last Checkpoint: a minute ago (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with various icons for file operations like save, new, and run. The notebook area contains three code cells:

- In [1]: `print("Aqsa Abu Bakar")`
Aqsa Abu Bakar
- In [2]: `print("BSCE-07")`
BSCE-07
- In [3]: `print("Artifical Intelligence(lab)")`
Artifical Intelligence(lab)

A green box highlights the input field for In []:.

Task no. 2:

Explain 5 real world examples of Artificial intelligence

Answer:

Artificial intelligence:

The Oxford Dictionary defines artificial intelligence as:

“The theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.”

The Oxford Dictionary of Phrase and Fable (2 ed.)

Essentially, artificial intelligence is the method by which a computer is able to act on data through statistical analysis, enabling it to understand, analyze, and learn from data through specifically designed algorithms. This is an automated process. Artificially intelligent machines can remember behavior patterns and adapt their responses to conform to those behaviors or encourage changes to them.

“From facial recognition to autocorrect, AI can be found all around us. Not only does it serve as a form of entertainment, but also provides countless utilities that mankind has become dependent on.”

The most important technologies that make up AI are machine learning (ML), deep learning, and natural language processing (NLP).

Artificial intelligence is a vast field and has now become an essential part of our everyday life some of its examples from real world are as follows

Recommendation systems:

With every day developing online businesses and stores, the usage of recommendation engines has become a significant part of the e-commerce industry. All of these Netflix “Other Movies You May Enjoy” and “Customer who bought this item also bought...” on Amazon, Facebook “People you may know” are the best practices of recommendation system usage.

A recommendation engine or a recommender system is a tool used by developers to foresee the users' choices in a huge list of suggested items.

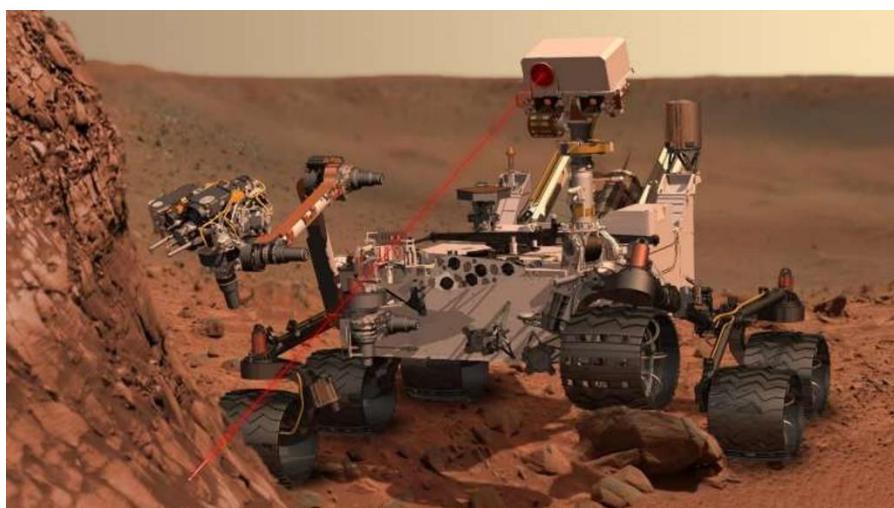
Generally, algorithms developed for recommendation systems rely on purchases and page views done before. What is more, today there are many services suggesting in-the-moment recommendations, as they use artificial intelligence for analyzing interactions of the users and find visually proper products that will interest any individual customer. Due to AI, recommendation engines make quick and to-the-point recommendations tailored to each customer's needs and preferences.



With the usage of artificial intelligence, online searching is improving as well, since it makes recommendations related to the user's visual preferences rather than product descriptions. Seemingly, **artificial intelligence consulting engines** may become the alternatives of search fields since they help users find items or content that they may not find in another way. That's why today recommendation engines play an essential role for sites like Amazon, Facebook, YouTube and so on.

Space Exploration

Space expeditions and discoveries always require analyzing vast amounts of data. Artificial Intelligence and Machine learning is the best way to handle and process data on this scale. After rigorous research, astronomers used Artificial Intelligence to sift through years of data obtained by the Kepler telescope in order to identify a distant eight-planet solar system.



Artificial Intelligence is also being used for NASA's next rover mission to Mars, the Mars 2020 Rover. The AEGIS, which is an AI-based Mars rover is already on the red planet. The rover is responsible for autonomous targeting of cameras in order to perform investigations on Mars.

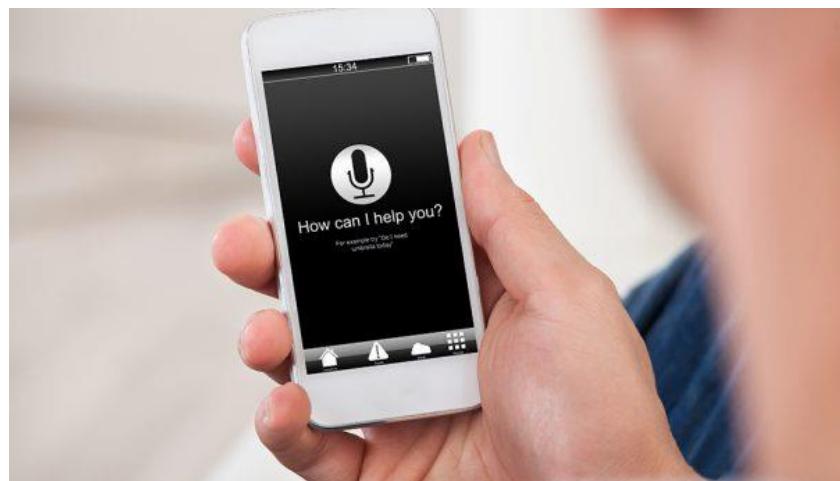
Digital Assistants:

When we have our hands full, we often resort to ordering digital assistants to perform tasks on our behalf. When you are driving, you might ask the assistant to call your mom (Don't text and drive, kids). A virtual assistant like Siri is an example of an AI that will access your contacts, identify the word "Mom", and call the number. These assistants use NLP, ML, statistical

analysis, and algorithmic execution to decide what you are asking for and try to get it for you. Voice and image search work in much the same way.



We also refer to them as *virtual assistants*, *predictive chatbots*, *mobile assistants*, *voice assistants*, or *virtual digital assistants*. If your smartphone, for example, has a digital assistant, you can tell it to do things for you, i.e., speak to it rather than type in instructions.



Here are some well-known examples:

Google Assistant.

Siri on iPads and iPhones.

Mycroft on Android phones.

Bixby on Samsung phones.

Amazon Echo on smart speakers.

Google Home on smart speakers.

Cortana on computers that use Microsoft systems.

E-Payments:

Having to run to the bank for every transaction is an enormous waste of time and AI is playing a part in why you haven't been to a bank branch in 5 years. Banks are now leveraging artificial intelligence to facilitate customers by simplifying payment processes.

Intelligent algorithms have made it possible to make deposits, transfer money, and even open accounts from anywhere, leveraging AI for security, identity management, and privacy controls.



Even potential fraud can be detected by observing users' credit card spending patterns. This is also an example of artificial intelligence. The algorithms know what kind of products User X buys, when and from where they are typically bought, and in what price bracket they fall.



When there is an unusual activity that does not fit in with the user profile, the system can generate an alert or a prompt to verify transactions.

Chatbots:

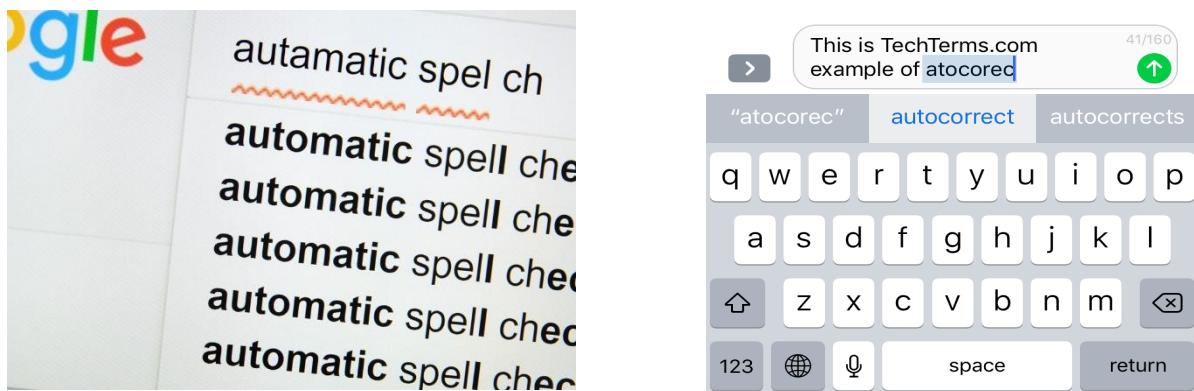
As a customer, interacting with customer service can be time-consuming and stressful. For companies, it's an inefficient department that is typically expensive and hard to manage. One increasingly popular artificially intelligent solution to this is the use of AI chatbots. The programmed algorithms enable machines to answer frequently asked questions, take and track orders, and direct calls.



Chatbots are taught to impersonate the conversational styles of customer representatives through natural language processing (NLP). Advanced chatbots no longer require specific formats of inputs (e.g. yes/no questions). They can answer complex questions requiring detailed responses. In fact, if you give a bad rating for the response you get, the bot will identify the mistake it made and correct it for next time, ensuring maximum customer satisfaction.

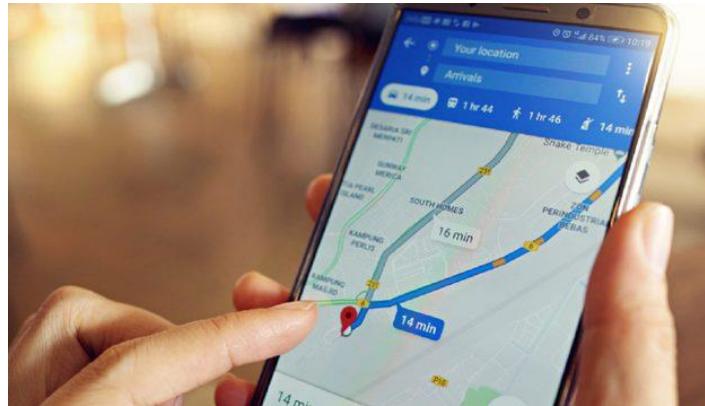
Text Editors or Autocorrect:

AI algorithms use machine learning, deep learning, and natural language processing to identify incorrect usage of language and suggest corrections in word processors, texting apps, and every other written medium, it seems. Linguists and computer scientists work together to teach machines grammar, just like you were taught at school. The algorithms are taught through high-quality language data so when you use a comma incorrectly, the editor will catch it.



Maps and Navigation:

AI has drastically improved traveling. Instead of having to rely on printed maps or directions, you can now use Waze or Google, or Apple Maps on your phone and type in your destination.



So how does the application know where to go? What's more, the optimal route, road barriers, and traffic congestions. Not too long ago, only satellite-based GPS was available, but now, artificial intelligence is being incorporated to give users a much more enhanced experience. Using machine learning, the algorithms remember the edges of the buildings that it has learned which allows for better visuals on the map, and recognition and understanding of house and building numbers. The application has also been taught to understand and identify changes in traffic flow so that it can recommend the route that avoids roadblocks and congestion.

LAB 02

(CLO 1-3, PLO 1-5,P3)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	<i>Python Programming-I:</i>
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice Basics of python string, input statements, decision making statement, list, dictionary and tuples

Basics of Python

(String, Input statement, decision-making statements, list, dictionaries etc.)

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Task no. 1:

Create variable of all types, print values and the types of all variables.

Solution:

Task 1

```
In [18]: ┆ i=10  
         print("The data type of i is",type(i))
```

The data type of i is <class 'int'>

```
In [19]: ┆ x='abc'  
         print("The data type of x is",type(x))
```

The data type of x is <class 'str'>

```
In [20]: ┆ z=2+3j  
         print("The data type of z is",type(z))
```

The data type of z is <class 'complex'>

```
In [21]: ┆ j=0.9  
         print("The data type of j is",type(j))
```

The data type of j is <class 'float'>

```
In [22]: ┆ check=True  
         print("The data type of check is",type(check))
```

The data type of check is <class 'bool'>

Task no. 2:

Take input from user (two numbers) and perform all arithmetic and logical operation on it and print results on display.

Solution:

With numeric values in logical operations:

Task 2

```
In [17]: x1=input('Enter 1st number = ')
y1=input('Enter 2nd number = ')
x=int(x1)
y=int(y1)
add=x+y
sub=x-y
mul=x*y
div=x/y
mod=x%y
exp=x**y
a=x and y
b=x or y
c=not x
#print
print("The addition of numbers is ",add)
print("The subtraction of numbers is ",sub)
print("The multiplication of numbers is ",mul)
print("The division of numbers is ",div)
print("The modulus of numbers is ",mod)
print("The exponent of numbers is ",exp)
print("The AND of numbers is ",a)
print("The OR of numbers is ",b)
print("The NOT of numbers is ",c)
```

```
Enter 1st number = 3
Enter 2nd number = 2
The addition of numbers is 5
The subtraction of numbers is 1
The multiplication of numbers is 6
The division of numbers is 1.5
The modulus of numbers is 1
The exponent of numbers is 9
The AND of numbers is 2
The OR of numbers is 3
The NOT of numbers is False
```

With bool values in logical operations:

Task 2

```
In [4]: x1=input('Enter 1st number = ')
y1=input('Enter 2nd number = ')
x=int(x1)
y=int(y1)
x2=True
y2=False
add=x+y
sub=x-y
mul=x*y
div=x/y
mod=x%y
exp=x**y
a=x2 and y2
b=x2 or y2
c=not x2
#print
print("\n      Arithematic Operations i.e.+,-,*,/,%,**  \n")
print("The addition of numbers is ",add)
print("The subtraction of numbers is ",sub)
print("The multiplication of numbers is ",mul)
print("The division of numbers is ",div)
print("The modulus of numbers is ",mod)
print("The exponent of numbers is ",exp)
print("\n      Logical Operations i.e.AND,OR,NOT  \n")
print("The AND of numbers is ",a)
print("The OR of numbers is ",b)
print("The NOT of numbers is ",c)
```

```
Enter 1st number = 2
Enter 2nd number = 4
```

Arithematic Operations i.e.+,-,*,/,%,**

The addition of numbers is 6
The subtraction of numbers is -2
The multiplication of numbers is 8
The division of numbers is 0.5
The modulus of numbers is 2
The exponent of numbers is 16

Logical Operations i.e.AND,OR,NOT

The AND of numbers is False
The OR of numbers is True
The NOT of numbers is False

Task no. 3:

Take a string (a long Statement) from user

Find the length of the string

Display the first and last character of the string

Capitalize all alphabets

Split string in to words (split using ‘ ’)

count the occurrence of an Alphabet (i.e : a)

replace the second word of a string with ‘Artificial’

reverse the complete string

Solution:

Task 3

```
In [13]: str=input("Enter a long string : ")
```

Enter a long string : Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision

```
In [14]: print("The length of given string is =",len(str))
```

The length of given string is = 237

```
In [15]: print("The first character of the given string is ",str[0])
```

The first character of the given string is A

```
In [16]: print("The last character of the given string is ",str[-1])
```

The last character of the given string is n

```
In [17]: print("The given string in upper case : ")
print(str.upper())
```

The given string in upper case :

ARTIFICIAL INTELLIGENCE IS THE SIMULATION OF HUMAN INTELLIGENCE PROCESSES BY MACHINES, ESPECIALLY COMPUTER SYSTEMS. SPECIFIC APPLICATIONS OF AI INCLUDE EXPERT SYSTEMS, NATURAL LANGUAGE PROCESSING, SPEECH RECOGNITION AND MACHINE VISION

```
In [18]: print("The splited string : ")
print(str.split(' '))
```

The splited string :
['Artificial', 'intelligence', 'is', 'the', 'simulation', 'of', 'human', 'intelligence', '', 'processes', 'by', 'machines', 'especially', 'computer', 'systems.', 'Specific', 'applications', '', 'of', 'AI', 'include', 'expert', 'systems', 'natural', 'language', 'processing', 'speech', 'recognition', '', 'and', 'machine', 'vision']

```
In [19]: print("In the given string a has occurred : ",str.count('a'))
```

In the given string a has occurred : 13

```
In [20]: print("The replaced string is up: ",str.replace('!!!','Artifical'))
```

The replaced string is up: Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision

```
In [21]: print("The reversed string is :",str[::-1])
```

The reversed string is : noisiv enihcam dna noitingocer hceeps ,gnisssecorp egaugnal larutan ,smetsys trepxe edulcni IA fo snoitacilppa cificepS .smetsys retupmoc yllaicepse ,senihcam yb sessecorp ecnegilletni namuh fo noitalumis eht si ecnegille tni laicifitrA

Task no. 4:

Take three integer input from user in a single line and find the average

Solution:

Task 4

```
In [22]: x,y,z=input('Enter three numbers with spaces to distinguish them = ').split(' ')
x1=int(x)
x2=int(y)
x3=int(z)
avg=(x1+x2+x3)/3
print("The average of the numbers entered is = ",avg)
```

Enter three numbers with spaces to distinguish them = 5 17 9
The average of the numbers entered is = 10.333333333333334

Or

Task 4

```
In [10]: x,y,z=input('Enter three numbers = ').split(' ')
x1=int(x)
x2=int(y)
x3=int(z)
avg=(x1+x2+x3)/3
print("The average of the numbers entered is =",avg)
```

```
Enter three numbers = 1 2 3
The average of the numbers entered is = 2.0
```

Task 4

```
In [11]: x,y,z=input('Enter three numbers = ').split(' ')
x1=int(x)
x2=int(y)
x3=int(z)
avg=(x1+x2+x3)/3
print("The average of the numbers entered is =",avg)
```

```
Enter three numbers = 7 15 67
The average of the numbers entered is = 29.666666666666668
```

Task no. 5:

Create a list of 10 values

Display the elements of a list

Update the value of 3rd index

Reverse the list

Add 2 more elements in 2nd index and print list

Pop element of 1 index and display list

Solution:

Task 5

```
In [30]: └─ list=[1,2,'A','Aqsa',3,4,'E','Esha','Ayesha','Ayra']
      print("The elements of list are = ",list)

      The elements of list are = [1, 2, 'A', 'Aqsa', 3, 4, 'E', 'Esha', 'Ayesha', 'Ayra']

In [31]: └─ list[2]='B'
      print("The updated list with element of 3rd index = ",list)

      The updated list with element of 3rd index = [1, 2, 'B', 'Aqsa', 3, 4, 'E', 'Esha', 'Ayesha', 'Ayra']

In [32]: └─ print("The reversed list is :",list[::-1])

      The reversed list is : ['Ayra', 'Ayesha', 'Esha', 'E', 4, 3, 'Aqsa', 'B', 2, 1]

In [33]: └─ list.insert(1,45)
      list.insert(1,66)
      print("The list after inserting elements of 2nd index is :",list)

      The list after inserting elements of 2nd index is : [1, 66, 45, 2, 'B', 'Aqsa', 3, 4, 'E', 'Esha', 'Ayesha', 'Ayra']

In [34]: └─ list.pop(0)
      print("The list after popping element of 1st index is :",list)

      The list after popping element of 1st index is : [66, 45, 2, 'B', 'Aqsa', 3, 4, 'E', 'Esha', 'Ayesha', 'Ayra']
```

Or

Task 5

```
In [35]: └─ list=[1,2,'A','Aqsa',3,4,'E','Esha','Ayesha','Ayra']
      print("The elements of list are = ",list)

      The elements of list are = [1, 2, 'A', 'Aqsa', 3, 4, 'E', 'Esha', 'Ayesha', 'Ayra']

In [36]: └─ list[3]='B'
      print("The updated list with element on 3rd index i.e. 4th element is = ",list)

      The updated list with element on 3rd index i.e. 4th element is = [1, 2, 'A', 'B', 3, 4, 'E', 'Esha', 'Ayesha', 'Ayra']

In [37]: └─ print("The reversed list is :",list[::-1])

      The reversed list is : ['Ayra', 'Ayesha', 'Esha', 'E', 4, 3, 'B', 'A', 2, 1]

In [38]: └─ list.insert(2,45)
      list.insert(2,66)
      print("The list after inserting elements on 2nd index i.e. 3rd element is :",list)

      The list after inserting elements on 2nd index i.e. 3rd element is : [1, 2, 66, 45, 'A', 'B', 3, 4, 'E', 'Esha', 'Ayesha', 'Ayra']

In [39]: └─ list.pop(1)
      print("The list after popping element of 1st index i.e. 2nd element is :",list)

      The list after popping element of 1st index i.e. 2nd element is : [1, 66, 45, 'A', 'B', 3, 4, 'E', 'Esha', 'Ayesha', 'Ayra']
```

Task no. 6:

Create dictionary

Display dictionary

Print keys and value separately

Insert new key-value pair

Copy dictionary into new dictionary

Solution:

Task 6

```
In [39]: mydict={ "Artifical" : "made or produced by human beings rather than occurring naturally, especially as a copy of something natural",  
           "Intelligence" : "the ability to acquire and apply knowledge and skills.",  
           2021:"The current year"  
         }  
  
In [40]: print("The Dictionary is \n")  
print(mydict)  
  
The Dictionary is  
  
{'Artifical': 'made or produced by human beings rather than occurring naturally, especially as a copy of something natural',  
 'Intelligence': 'the ability to acquire and apply knowledge and skills.', 2021: 'The current year'}  
  
In [41]: print("The keys in the given Dictionary are \n")  
print(mydict.keys())  
  
The keys in the given Dictionary are  
  
dict_keys(['Artifical', 'Intelligence', 2021])  
  
In [42]: print("The values in the given Dictionary are \n")  
print(mydict.values())  
  
The values in the given Dictionary are  
  
dict_values(['made or produced by human beings rather than occurring naturally, especially as a copy of something natural',  
 'the ability to acquire and apply knowledge and skills.', 'The current year'])
```

```
In [43]: print("The original dictionary is\n\n",mydict)
updatedict={"Aqsa":"Height/limit/shore/Remote/Farthest",
"laboratory":'''a room or building equipped for scientific experiments,
research, or teaching, or for the manufacture of drugs or chemicals'''}
print("\n      Inserting new key pair in dictionary \n\nUpdated Dictionary is \n")
mydict.update(updatedict)
print(mydict)
```

The original dictionary is

```
{'Artifical': 'made or produced by human beings rather than occurring naturally, especially as a copy of something natural',
 'Intelligence': 'the ability to acquire and apply knowledge and skills.', 2021: 'The current year'}
```

Inserting new key pair in dictionary

Updated Dictionary is

```
{'Artifical': 'made or produced by human beings rather than occurring naturally, especially as a copy of something natural',
 'Intelligence': 'the ability to acquire and apply knowledge and skills.', 2021: 'The current year', 'Aqsa': 'Height/limit/shore/Remote/Farthest',
 'laboratory': 'a room or building equipped for scientific experiments, \nresearch, or teaching, or for the manufacture of drugs or chemicals'}
```

```
In [46]: print("The original dictionary is\n\n",mydict)
dict_copy=mydict.copy()
print("\nThe copied dictionary is\n\n",dict_copy)
```

The original dictionary is

```
{'Artifical': 'made or produced by human beings rather than occurring naturally, especially as a copy of something natural',
 'Intelligence': 'the ability to acquire and apply knowledge and skills.', 2021: 'The current year', 'Aqsa': 'Height/limit/shore/Remote/Farthest',
 'laboratory': 'a room or building equipped for scientific experiments, \nresearch, or teaching, or for the manufacture of drugs or chemicals'}
```

The copied dictionary is

```
{'Artifical': 'made or produced by human beings rather than occurring naturally, especially as a copy of something natural',
 'Intelligence': 'the ability to acquire and apply knowledge and skills.', 2021: 'The current year', 'Aqsa': 'Height/limit/shore/Remote/Farthest',
 'laboratory': 'a room or building equipped for scientific experiments, \nresearch, or teaching, or for the manufacture of drugs or chemicals'}
```

LAB 03

(CLO 1-3, PLO 1-5,P3)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	<i>Python Programming-II</i>
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice: Basics of python loops functions and lambda functions.

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Basics of Python-II

(Loops, functions, lambda function)

Lab Tasks:

Task no. 1:

Compare a and b and display whether a is greater than, less than or equal to b where a=5 and b=2

Task 1

```
# a=5
# b=2
if(a>b):
    print('a is greater than b')
elif(a<b):
    print('a is less than b ')
else:
    print('a and b are equal')

a is greater than b
```

Task no. 2:

Compare a and b and display whether a is greater than, less than or equal to, take values of a and b from the user

Task 2

```
a=int(input('Enter value of a :'))
b=int(input('Enter value of b :'))
if(a>b):
    print('a is greater than b')
elif(a<b):
    print('a is less than b ')
else:
    print('a and b are equal')
```

```
Enter value of a :8
Enter value of b :9
a is less than b
```

Task 2

```
a=int(input('Enter value of a :'))
b=int(input('Enter value of b :'))
if(a>b):
    print('a is greater than b')
elif(a<b):
    print('a is less than b ')
else:
    print('a and b are equal')
```

```
Enter value of a :5
Enter value of b :2
a is greater than b
```

Task 2

```
In [4]: # a=int(input('Enter value of a :'))
# b=int(input('Enter value of b :'))
if(a>b):
    print('a is greater than b')
elif(a<b):
    print('a is less than b ')
else:
    print('a and b are equal')
```

```
Enter value of a :5
Enter value of b :5
a and b are equal
```

Task no. 3:

Implement for loop, remove newline, inserting character/symbol in between characters using print statement.

Task 3

```
In [11]: # print("Implementing for loop :")
# for i in "PYTHON":
#     print(i)
print("\n\nRemoving Newline from print statement : ")
for i in "PYTHON":
    print(i,end='')
print("\n\nAdding different characters/symbol in print statement : ")
for i in "PYTHON":
    print(i,end='*')

Implementing for loop :
P
Y
T
H
O
N

Removing Newline from print statement :
PYTHON

Adding different characters/symbol in print statement :
P*Y*T*H*O*N*
```

Task no. 4:

Perform the tasks in task 3 with list

Task 4

```
: # list=[1,4,5,6,8]
# print("Implementing for loop with list :")
# for i in list:
#     print(i)
print("\n\nRemoving Newline from print statement : ")
for i in list:
    print(i,end='')
print("\n\nAdding different characters/symbol in print statement : ")
for i in list:
    print(i,end=',',)

Implementing for loop with list :
1
4
5
6
8

Removing Newline from print statement :
14568

Adding different characters/symbol in print statement :
1,4,5,6,8,
```

Task no. 5:

By using for loop print counting from 1 to 10 and later change the step size.

Task 5

```
In [14]: ┌─▶ print("Print counting 1-10 by using for loop")
      ┌─▶   for i in range(1,11):
      ┌─▶     print(i)
      ┌─▶ #steps size =2
      ┌─▶ print("\nChanging stepsize to 3")
      ┌─▶ for i in range(1,11,3):
      ┌─▶   print(i)

Print counting 1-10 by using for loop
1
2
3
4
5
6
7
8
9
10

Changing stepsize to 3
1
4
7
10
```

Task no. 6:

Print table of any number entered by the user by using for loop

Task 6

```
In [25]: ┌─▶ a=int(input('Enter the number for which you want to print table : '))
      ┌─▶ for i in range(1,11):
      ┌─▶   print(a, " x ",i," = ",a*i)

Enter the number for which you want to print table : 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Task no. 7:

Print the keys of a dictionary using for loop

Task 7

```
#print keys
mydict={ "Artifical" : "made or produced by human beings rather than occurring naturally, especially as a copy of something r
    "Intelligence" : "the ability to acquire and apply knowledge and skills.",
    2021:"The current year"
}
for i in mydict:
    print(i)
```

Artifical
Intelligence
2021

Task no. 8:

Print the values of a dictionary

Task 8

```
#print values
mydict={ "Artifical" : "made or produced by human beings rather than occurring naturally, especially as a copy of something r
    "Intelligence" : "the ability to acquire and apply knowledge and skills.",
    2021:"The current year"
}
for i in mydict:
    print(mydict[i])
```

made or produced by human beings rather than occurring naturally, especially as a copy of something natural
the ability to acquire and apply knowledge and skills.
The current year

Task no. 9:

Print complete dictionary

Task 9

```
In [30]: #print complete dictionary
mydict={ "Artifical" : "made or produced by human beings rather than occurring naturally, especially as a copy of something r
    "Intelligence" : "the ability to acquire and apply knowledge and skills.",
    2021:"The current year"
}
for i in mydict:
    print(i, " : ",mydict[i])
```

Artifical : made or produced by human beings rather than occurring naturally, especially as a copy of something natural
Intelligence : the ability to acquire and apply knowledge and skills.
2021 : The current year

Task no. 10:

Print even numbers in the range of 1 to 100

Task 10

```
▶ for i in range(1,101):
    if(i%2==0):
        print(i,end=",")
```

2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,94,96,98,100,

Task no. 11:

Print even numbers amongst the range of 1 to 101 with a step size of 5 and count them

Task 11

```
▶ count =0
for i in range(1,101,5):
    if(i%2==0):
        print(i,end=",")
    count=count+1
print("\nNo. of even numbers in the given range are : ",count)
```

6,16,26,36,46,56,66,76,86,96,
No. of even numbers in the given range are : 10

Task no. 12:

Print a list using for loop by taking input from the user

Task 12

```
▶ lis=[]
for i in range(1,11):
    a=int(input("Enter the element : "))
    lis.append(a)
print(lis)
```

Enter the element : 1
Enter the element : 2
Enter the element : 3
Enter the element : 4
Enter the element : 5
Enter the element : 6
Enter the element : 7
Enter the element : 8
Enter the element : 9
Enter the element : 10
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Task no. 13:

Print counting from 1 to 10 using while loop

Task 13

```
▶ i=0
  while(i<10):
    print(i)
    i=i+1
```

```
0
1
2
3
4
5
6
7
8
9
```

Task no. 14:

Take input from user and find the greatest number using while loop

Task 14

```
▶ lis=[]
  i=0
  j=1
  while (i<10):
    a=int(input("Enter the element : "))
    lis.append(a)
    i=i+1
  maxi=lis[0]
  while (j<10):
    if(maxi<lis[j]):
      maxi=lis[j]
    j=j+1
  print("The greatest number amongst the list is : ",maxi)
```

```
Enter the element : 1
Enter the element : 2
Enter the element : 3
Enter the element : 4
Enter the element : 9
Enter the element : 5
Enter the element : 6
Enter the element : 7
Enter the element : 8
Enter the element : 1
The greatest number amongst the list is : 9
```

Task no. 15:

Implement functions

Task 15

```
► #def func_name (parameters):
    #return
def function_name():
    print("function of printing")
    return

► function_name()
    function of printing
```

Task no. 16:

Find the sum of a and b by function

Task 16

```
► def function_name1(a,b):
    print("Sum",a+b)
    return

► function_name1(3,2)
    Sum 5
```

Task no. 17:

Find the sum of a and b by return statement of function

Task 17

```
In [27]: ► def function_name1(a,b):
    return a+b

In [28]: ► function_name1(3,2)

Out[28]: 5
```

Task no. 18:

Pass the values as labeled arguments in task 17

Task 18

```
In [10]: ► function_name1(a=3,b=2)  
Out[10]: 5
```

Task no. 19:

Demonstrate how function acts if only one argument is passed out of 2

Task 19

```
► def function_name1(a=1,b=1):  
    print("Sum",a+b)  
    return  
  
► function_name1(a=3)  
Sum 4  
  
► function_name1(b=4)  
Sum 5
```

Task no. 20:

Demonstrate the use of lambda function

Task 20

```
► #Lambda function mean anonymous functions  
a=lambda x:x**2  
print(a(2))
```

4

Task no. 21:

Demonstrate the use of lambda function

Task 21

```
▶ a=lambda x,y:x**y  
    print(a(2,3))
```

8

Task no. 22:

Demonstrate what happens when an argument is passed in the variable with which lambda function is stored

Task 22

```
▶ def func(n):  
    return lambda n:n*2
```

```
▶ a=func(2)  
    print(a(10))
```

20

Home Tasks:

Task no. 1:

Write a program that ask user to enter integer number and print the Table of that number

Home tasks

Task 1

```
► a=int(input('Enter the integer for which you want to print table : '))
  for i in range(1,11):
    print(a," x ",i," = ",a*i)

Enter the integer for which you want to print table : 87
87 x 1 = 87
87 x 2 = 174
87 x 3 = 261
87 x 4 = 348
87 x 5 = 435
87 x 6 = 522
87 x 7 = 609
87 x 8 = 696
87 x 9 = 783
87 x 10 = 870
```

Task no. 2:

Write a program to enter integer numbers (10) store in a list and print the sum of all elements

Task 2

```
► sum1=0
  lis=[]
  for i in range(1,11):
    k=int(input("Enter number "))
    lis.append(k)
    sum1=sum1+k
  print(lis)
  print("The sum of numbers entered is :",sum1)

Enter number 1
Enter number 2
Enter number 3
Enter number 4
Enter number 5
Enter number 6
Enter number 7
Enter number 8
Enter number 9
Enter number 10
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The sum of numbers entered is : 55
```

OR

```
► sum1=0
lis=[]
j=0
i=0
while i<10:
    k=int(input("Enter number "))
    lis.append(k)
    i=i+1
while j<10:
    sum1=sum1+lis[j]
    j=j+1
print(lis)
print("The sum of numbers entered is :",sum1)
```

```
Enter number 1
Enter number 2
Enter number 3
Enter number 4
Enter number 5
Enter number 6
Enter number 7
Enter number 8
Enter number 9
Enter number 10
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The sum of numbers entered is : 55
```

Task no. 3:

Write a program that ask user to enter integer number and print the given number is it even or odd

Task 3

```
► a=int(input("Enter a number :"))
if(a%2==0):
    print("The number is even")
else:
    print("The number is odd")
```

```
Enter a number :88
The number is even
```

Task 3

```
► a=int(input("Enter a number :"))
  if(a%2==0):
    print("The number is even")
  else:
    print("The number is odd")
```

Enter a number :77
The number is odd

Task no. 4:

Write a program that use any mathematical Expression using Lambda function

Task 4

```
: ► a=lambda x,y,z:5*x+3*y+9*z
  print(a(2,3,1))
```

28

Task no. 5:

Write a program that generate 100 random numbers 1-100 and print on screen

Task 5

```
► import random
  i=0
  while(i<100):
    print(random.randint(1,100))
    i=i+1
```

```
39  
74  
36  
21  
26  
64  
46  
17  
96  
73  
45  
87  
62  
30  
41  
71  
76  
37  
52  
88  
68
```

```
89  
32  
41  
90  
7  
81  
75  
75  
18  
73  
46  
7  
6  
44  
10  
93  
67  
76  
95  
64  
45  
18  
100  
71  
38  
49  
25  
44  
36  
32
```

```
32  
4  
16  
5  
69  
41  
9  
19  
4  
92  
63  
92  
22  
45  
62  
22  
63  
86  
4  
91  
15  
28  
69  
18  
78  
70  
9  
66  
30  
56  
49  
69  
55  
72  
61  
93  
45  
36  
89  
91  
11  
44  
92
```

```
12  
74  
90  
59  
68  
100  
78
```

Task no. 6:

Write a program that take 3 integer numbers from user and find the maximum number using Decision Making statements

Task 6

```
► a=int(input("Enter 1st number : "))  
b=int(input("Enter 2nd number : "))  
c=int(input("Enter 3rd number : "))  
if(a>b and a>c):  
    print("a is greatest")  
elif( b>a and b>c):  
    print("b is greatest")  
else:  
    print("c is greatest")
```

```
Enter 1st number : 9  
Enter 2nd number : 5  
Enter 3rd number : 1  
a is greatest
```

Task 6

```
► a=int(input("Enter 1st number : "))
b=int(input("Enter 2nd number : "))
c=int(input("Enter 3rd number : "))
if(a>b and a>c):
    print("a is greatest")
elif( b>a and b>c):
    print("b is greatest")
else:
    print("c is greatest")
```

```
Enter 1st number : 6
Enter 2nd number : 9
Enter 3rd number : 5
b is greatest
```

Task 6

```
► a=int(input("Enter 1st number : "))
b=int(input("Enter 2nd number : "))
c=int(input("Enter 3rd number : "))
if(a>b and a>c):
    print("a is greatest")
elif( b>a and b>c):
    print("b is greatest")
else:
    print("c is greatest")
```

```
Enter 1st number : 1
Enter 2nd number : 5
Enter 3rd number : 9
c is greatest
```

Task no. 7:

Write a program that print even numbers between the range of 1-200

Task 7

```
► for i in range(1,200):
    if(i%2==0):
        print(i)
```

2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84

84
86
88
90
92
94
96
98
100
102
104
106
108
110
112
114
116
118
120
122
124
126
128
130
132
134
136
138
140
142
144
146
148
150
152
154
156
158
160
162
164
166

168
170
172
174
176
178
180
182
184
186
188
190
192
194
196
198

Task no. 8:

Write a program that asks the user for a weight in kilograms and converts it to pounds.

Task 8

```
a=int(input("Enter the mass in kilograms(kg) :"))
b=lambda a:a*2.20462
print(a,"kg mass is equal to ",b(a)," pound")
```

```
Enter the mass in kilograms(kg) :35
35 kg mass is equal to  77.1617  pound
```

Task no. 9:

Write a program that generates a list of 20 random numbers between 1 and 100.

Print the list.

Print the average of the elements in the list.

Print the largest and smallest values in the list.

Print the second largest and second smallest entries in the list

Print how many even numbers are in the list.

Task 9

```
► import random
lis=[]
great=0;
great2=0;
mini=2;
mini=0;
summ=0;
i=0;
j=1;
count=0;
while(i<20):
    a=random.randint(1,100)
    summ=summ+a
    lis.append(a)
    if(a%2==0):
        count=count+1
    i=i+1

print(lis,end=",")
lis.sort()
x=len(lis)
print("\nThe smallest number in the list is : ",lis[0])
print("The second smallest number in the list is : ",lis[1])
print("The largest number in the list is : ",lis[x-1])
print("The second largest number in the list is : ",lis[x-2])
print("The average of the list is : ",summ/20)
print("The number of even numbers in the list is : ",count)
```

```
[93, 99, 87, 52, 34, 10, 100, 70, 36, 26, 62, 22, 42, 61, 8, 94, 4, 33, 39, 38],
The smallest number in the list is : 4
The second smallest number in the list is : 8
The largest number in the list is : 100
The second largest number in the list is : 99
The average of the list is : 50.5
The number of even numbers in the list is : 14
```

Or

Task 9

```
]▶ import random
lis=[]
great=0;
great2=0;
mini=2;
mini=0;
summ=0;
i=0;
j=1;
count=0;
while(i<20):
    a=random.randint(1,100)
    summ=summ+a
    lis.append(a)
    if(a%2==0):
        count=count+1
    i=i+1

print(lis,end="")
lis.sort()
print("\nThe smallest number in the list is : ",lis[0])
print("The second smallest number in the list is : ",lis[1])
print("The largest number in the list is : ",lis[-1])
print("The second largest number in the list is : ",lis[-2])
print("The average of the list is : ",summ/20)
print("The number of even numbers in the list is : ",count)

[58, 5, 45, 87, 85, 57, 46, 20, 52, 38, 7, 13, 16, 36, 51, 44, 69, 34, 13, 84],
The smallest number in the list is : 5
The second smallest number in the list is : 7
The largest number in the list is : 87
The second largest number in the list is : 85
The average of the list is : 43.0
The number of even numbers in the list is : 10
```

Task no. 10:

Write a program that asks the user for an integer and creates a list that consists of the factors of that integer.

Task 10

```
▶ lis=[]
a=int(input("Enter an integer to create a list of it's factors: "))
for i in range (1,a+1):
    if(a%i==0):
        lis.append(i)
print("The list of Factors is : ",lis)

Enter an integer to create a list of it's factors: 99
The list of Factors is : [1, 3, 9, 11, 33, 99]
```

Task 10

```
► lis=[]
a=int(input("Enter an integer to create a list of it's factors: "))
for i in range (1,a+1):
    if(a%i==0):
        lis.append(i)
print("The list of Factors is : ",lis)
```

```
Enter an integer to create a list of it's factors: 54
The list of Factors is : [1, 2, 3, 6, 9, 18, 27, 54]
```

Task no. 11:

Write a function called sum_digits that is given an integer num and returns the sum of the digits of num

Task 11

```
► def sum_digits(num):
    sum_di=0
    for i in str(num):
        sum_di+=int(i)
    print("The sum of the digits of the given num ",num," is ",sum_di)

► sum_digits(123456789)
The sum of the digits of the given num 123456789  is  45
```

Task no. 12:

Write a function called number_of_factors that takes an integer and returns how many factors the number has.

Task 12

```
: ► def number_of_factors(num):
    count=0
    for i in range (1,num+1):
        if(num%i==0):
            count=count+1
    print("The given number has ",count," factors")

: ► number_of_factors(69)
The given number has 4 factors
```

Task 12

```
► def number_of_factors(num):
    lis=[]
    count=0
    for i in range (1,num+1):
        if(num%i==0):
            lis.append(i)
            count=count+1
    print("The given number has ",count," factors i.e. ",lis)

► number_of_factors(88)
The given number has 8 factors i.e. [1, 2, 4, 8, 11, 22, 44, 88]
```

Task no. 13:

Write a function called is_sorted that is given a list and returns True if the list is sorted and False otherwise.

Method 1:

Method 1

```
► def is_sorted(lis):
    flag=0
    for i in range(0,len(lis)-1):
        if(lis[i]<=lis[i+1]):
            flag=1
        else:
            flag=0
            break
    if(flag==0):
        print("No, the list is not sorted")
    else:
        print("Yes, the given list is sorted")

► a=[1,2,3,4,5,6,7,8,10]
is_sorted(a)

Yes, the given list is sorted
```

Method 1

```
► def is_sorted(lis):
    flag=0
    for i in range(0,len(lis)-1):
        if(lis[i]<=lis[i+1]):
            flag=1
        else:
            flag=0
            break
    if(flag==0):
        print("No, the list is not sorted")
    else:
        print("Yes, the given list is sorted")
```

```
► a=[1,2,3,4,10,6,7,8,1]
    is_sorted(a)
```

No, the list is not sorted

Method 2:

Task 13

```
► def is_sorted(lis):
    flag=0
    if(lis==sorted(lis)):
        print("Yes, the given list is sorted")
    else:
        print("No, the list is not sorted")
```

```
► a=[1,2,3,4,5,6,7,8,9]
    is_sorted(a)
```

Yes, the given list is sorted

Task 13

```
► def is_sorted(lis):
    flag=0
    if(lis==sorted(lis)):
        print("Yes, the given list is sorted")
    else:
        print("No, the list is not sorted")
```

```
► a=[1,2,3,4,9,6,7,8,7]
    is_sorted(a)
```

No, the list is not sorted

Task no. 14:

Write a function called root that is given a number x and an integer n and returns $x^{(1/n)}$. In the function definition, set the default value of n to 2.

Task 14

```
In [36]: ┏ def root(x,n=2):  
      return(x**(1/n))
```

```
In [38]: ┏ root(12)
```

```
Out[38]: 3.4641016151377544
```

Task 14

```
In [36]: ┏ def root(x,n=2):  
      return(x**(1/n))
```

```
In [39]: ┏ root(x=4)
```

```
Out[39]: 2.0
```

Task no. 15:

Write a program that asks the user for their name and how many times to print it. The program should print out the user's name the specified number of times.

Task 15

```
┏ a=input("Enter your name : ")  
b=int(input("How many times you want to print your name ? "))  
count=1;  
for i in range (0,b):  
    print(count,".",a)  
    count=count+1
```

```
Enter your name : Aqsa  
How many times you want to print your name ? 6  
1 . Aqsa  
2 . Aqsa  
3 . Aqsa  
4 . Aqsa  
5 . Aqsa  
6 . Aqsa
```

Task no. 16:

Use a for loop to print a triangle like the one below. Allow the user to specify how high the triangle should be.

```
*  
**  
***  
****
```

Task 16

```
▶ a=int(input("Enter the height of the triangle : "))  
    for i in range (0,a):  
        for j in range (0,i+1):  
            print("*",end='')  
        print("\n")
```

```
Enter the height of the triangle : 3  
*  
  
**  
  
***
```

Task 16

```
▶ a=int(input("Enter the height of the triangle : "))  
    for i in range (0,a):  
        for j in range (0,i+1):  
            print("*",end='')  
        print("\n")
```

```
Enter the height of the triangle : 6  
*  
  
**  
  
***  
  
****  
  
*****
```

Task no. 17:

Generate a random number between 1 and 10. Ask the user to guess the number and print a message based on whether they get it right or not.

Task 17

```
▶ import random
a=random.randint(1,10)
b=int(input("Guess the number from 1 to 10 : "))
if(a==b):
    print("Welldone!!! You got it Right")
else:
    print("Oops!!! you got the wrong\nThe correct answer was ",a)
```

```
Guess the number from 1 to 10 : 2
Welldone!!! You got it Right
```

Task 17

```
▶ import random
a=random.randint(1,10)
b=int(input("Guess the number from 1 to 10 : "))
if(a==b):
    print("Welldone!!! You got it Right")
else:
    print("Oops!!! you got the wrong\nThe correct answer was ",a)
```

```
Guess the number from 1 to 10 : 4
Oops!!! you got the wrong
The correct answer was 1
```

Task no. 18:

Write a program that generates 100 random integers that are either 0 or 1. Then find the longest run of zeros, the largest number of zeros in a row. For instance, the longest run of zeros in [1,0,1,1,0,0,0,0,1,0,0] is 4.

Task no. 19:

Write a program that asks the user to enter a length in feet. The program should then give the user the option to convert from feet into inches, yards, miles, millimeters, centimeters, meters, or kilometers. Say if the user enters a one, then the program converts to inches, if they enter a 2, then the program converts to yards, etc. While this can be done with if statements, it is much shorter with lists and it is also easier to add new conversions if you use lists.

Task 19

```
► print("-----")
feet = int(input("Enter a length in feet: "))
print(''-----
Press 1 to convert in inches(in)
Press 2 to convert in yards(yd)
Press 3 to convert in miles(mi)
Press 4 to convert in millimeter(mm)
Press 5 to convert in centimeter(cm)
Press 6 to convert in meter(m)
Press 7 to convert in kilometer(km)
-----')
a=int(input("Enter your choice : "))
print("-----")
inch = feet * 12
yd = feet * 0.33333
mi = feet * 0.000189393939
mm = feet * 304.8
cm = feet * 30.48
m = feet * 0.3048
km = feet * 0.0003048
convt=[inch,yd,mi,mm,cm,m,km]
print("> ",convt[a-1])
```

```
-----
Enter a length in feet: 4
-----
Press 1 to convert in inches(in)
Press 2 to convert in yards(yd)
Press 3 to convert in miles(mi)
Press 4 to convert in millimeter(mm)
Press 5 to convert in centimeter(cm)
Press 6 to convert in meter(m)
Press 7 to convert in kilometer(km)
-----
Enter your choice : 1
-----
> 48
```

```

-----
Enter a length in feet: 34
-----
Press 1 to convert in inches(in)
Press 2 to convert in yards(yd)
Press 3 to convert in miles(mi)
Press 4 to convert in millimeter(mm)
Press 5 to convert in centimeter(cm)
Press 6 to convert in meter(m)
Press 7 to convert in kilometer(km)
-----
Enter your choice : 5
-----
> 1036.32

```

```

-----
Enter a length in feet: 87
-----
Press 1 to convert in inches(in)
Press 2 to convert in yards(yd)
Press 3 to convert in miles(mi)
Press 4 to convert in millimeter(mm)
Press 5 to convert in centimeter(cm)
Press 6 to convert in meter(m)
Press 7 to convert in kilometer(km)
-----
Enter your choice : 7
-----
> 0.0265176

```

Task no. 20:

Implement the functionality of the ATM with initial balance Rs:50,000.

Task 20

```

► import getpass
from datetime import datetime
import random
import getpass
print("\t\tATM")
Amount=50000
#print("Please insert your card ") not applicable here but physically
card_num=int(input("Enter your card number : "))
p=getpass.getpass(prompt="Insert your secret pin : ")
if(card_num==1234567890123456 and p=='1234'):
    print('')
    1. 1,000 PKR
    2. 2000 PKR
    3. 5000 PKR
    4. 10,000 PKR
    5. 20,000 PKR
    6. Other Amount
    7. Check Account Balance''')
acc=int(input("Enter your Choice : "))
if(acc==1):
    slip=input("do you want balance slip ? y/n : ")
    a=1000
    Amount=Amount-a

```

```

elif(acc==2):
    slip=input("do you want balance slip ? y/n : ")
    q=2000
    Amount=Amount-a
elif(acc==3):
    slip=input("do you want balance slip ? y/n : ")
    a=5000
    Amount=Amount-q
elif(acc==4):
    slip=input("do you want balance slip ? y/n : ")
    a=10000
    Amount=Amount-a
elif(acc==5):
    slip=input("do you want balance slip ? y/n : ")
    a=25000
    Amount=Amount-a
elif(acc==6):
    a=int(input("Enter your desired amount : "))
    slip=input("do you want balance slip ? y/n : ")
    Amount=Amount-a
elif(acc==7):
    print("Your current account balance is ",Amount)
if(slip=='y'):
    print("Transaction Sucessfull !!!")
    print("\n\n-----")
    print("\t",datetime.now())

```

```

print("\t\tWITHDRAWAL")
print("-----")
print("\tCard No : 123*****456")
print("\tAccount :",random.randint(1000,9999),"*****",random.randint(1000,9999))
print("\tSTAN   :",random.randint(1000000,9999999))
print("\tTransaction Amount : PKR",a)
print("\tAvailable Balance : PKR",Amount)
print("\n\t\tTHANK YOU\u0001f600")
print("-----")
else:
    print("Transaction Sucessfull !!!")
else:
    print("The information you entered is wrong !!!")

```

ATM

Enter your card number : 1234567890123456

Insert your secret pin :

1. 1,000 PKR
2. 2000 PKR
3. 5000 PKR
4. 10,000 PKR
5. 20,000 PKR
6. Other Amount
7. Check Account Balance

Enter your Choice : 3

do you want balance slip ? y/n : y

Transaction Sucessfull !!!

2021-10-18 22:26:16.394128

WITHDRAWAL

Card No : 123*****456

Account : 4135 ***** 4832

STAN : 2915967

Transaction Amount : PKR 5000

Available Balance : PKR 48000

THANK YOU 😊

LAB 04

(CLO 1-2, PLO 1-5,P3)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	<i>Introduction about Numpy</i>
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice Basic Data Manipulation operations using Pandas Library and Function

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

NUMPY

Lab Tasks:

Introduction to NUMPY:

What is NumPy?

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
- NumPy stands for Numerical Python.

Why Use NumPy?

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
- Arrays are very frequently used in data science, where speed and resources are very important.

Why is NumPy Faster Than Lists?

- NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.
- This behavior is called locality of reference in computer science.
- This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

NumPy as np:

- NumPy is usually imported under the np alias. (*In Python alias are an alternate name for referring to the same thing.*)
- Create an alias with the as keyword while importing

Import numpy as np

- Now the NumPy package can be referred to as np instead of numpy.

Create a NumPy ndarray Object:

- NumPy is used to work with arrays. The array object in NumPy is called ndarray.

- We can create a NumPy ndarray object by using the array() function.
- To create an ndarray, we can pass a list, tuple or any array-like object into the array() method, and it will be converted into an ndarray

Task no. 1:

Take 2 lists and multiply both you'll see that error occurs repeat the process but by converting them toarray by numpy.array()

Solution:

Lab Task

```

: ┌─[1,2,3]
  12=[4,5,6]
  print(l1*l2)

-----
TypeError                                     Traceback (most recent call last)
<ipython-input-1-c12a32d3adc3> in <module>
      1 l1=[1,2,3]
      2 l2=[4,5,6]
----> 3 print(l1*l2)

TypeError: can't multiply sequence by non-int of type 'list'

```

Task 1

```

import numpy as np
l1=[1,2,3]
l2=[4,5,6]
A1=np.array([l1])
A2=np.array([l2])
print("1st array (A1) = ",A1)
print("2nd array (A2) = ",A2)
print("Multiplying both (A1 X A2) = ",A1*A2)
#[[]]multi idimensional because list was passed within a list

1st array (A1) =  [[1 2 3]]
2nd array (A2) =  [[4 5 6]]
Multiplying both (A1 X A2) =  [[ 4 10 18]]

```

Task no. 2:

Demonstrate the use of numpy.dtype and numpy.shape() functions

Solution:

Checking the Data Type of an Array:

The NumPy array object has a property called dtype that returns the data type of the array

Get the Shape of an Array:

NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

Task 2

```
❷ import numpy as np
l1=[1,2,3]
l2=[4,5,6]
A1=np.array(l1)
A2=np.array(l2)
print("A1 = ",A1)
print("A2 = ",A2)
A=A1*A2
print("A1 x A2 = ",A)
print("> The type of array using type : ",type(A))
print("> The type of array using dtype : ",A.dtype)#no() with dtype bcz it is attribute not a function
print("> The dimension of array : ",A.shape)

A1 = [1 2 3]
A2 = [4 5 6]
A1 x A2 = [ 4 10 18]
> The type of array using type : <class 'numpy.ndarray'>
> The type of array using dtype : int32
> The dimension of array : (3,)
```

Task no. 3:

The size of an array created with numpy.array() is int32 convert it to int 8

Solution:

When declaring the array we can reduce it's size as demonstrated in the task below

Task 3

```
▶ import numpy as np
11=[1,2,3]
12=[4,5,6]
A1=np.array(11,np.int8)
A2=np.array(12,np.int8)      #reducing size from 32 to 8
print("A1 = ",A1)
print("A2 = ",A2)
A=A1*A2
print("A1 x A2 = ",A)
print("> The type of array using type : ",type(A))
print("> The type of array using dtype : ",A.dtype)
print("> The dimension of array : ",A.shape)

A1 = [1 2 3]
A2 = [4 5 6]
A1 x A2 = [ 4 10 18]
> The type of array using type : <class 'numpy.ndarray'>
> The type of array using dtype : int8
> The dimension of array : (3,)
```

Task no. 4:

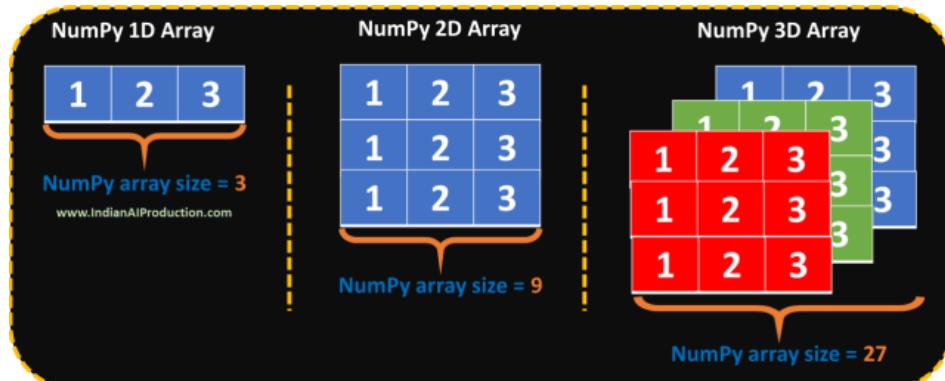
Demonstrate the use of numpy.size() functions

Solution:

Get the Size of an Array:

To find python **NumPy array size** use size() function. The NumPy size() function has two arguments. First is an array, required an argument need to give array or array name. Second is an axis, default an argument. The axis contains none value, according to the requirement you can change it. The **np.size()** function count items from a given array and give output in the form of a number as size.

np.size(array, axis=None)



Task 4

```
▶ import numpy as np
t1=(1,2,3)
t2=(4,5,6)
A1=np.array(t1,np.int8)
A2=np.array(t2,np.int8)      #reducing size from 32 to 8
print("A1 = ",A1)
print("A2 = ",A2)
A=A1*A2
print("A1 x A2 = ",A)
print("> The type of array using type : ",type(A))
print("> The type of array using dtype : ",A.dtype)
print("> The size of array using dtype : ",A.size)
print("> The dimension of array : ",A.shape) #dimension of arrays

A1 = [1 2 3]
A2 = [4 5 6]
A1 x A2 = [ 4 10 18]
> The type of array using type : <class 'numpy.ndarray'>
> The type of array using dtype : int8
> The type of array using dtype : 3
> The dimension of array : (3,)
```

Task no. 5:

Create a 2D array using numpy.array()

Solution:

Task 5

```
▶ import numpy as np
l1=[1,2,3]
l2=[4,5,6]
A=np.array((l1,l2))
print("> The type of array using dtype : ",A.dtype)
print("> The dimension of array : ",A.shape)

> The type of array using dtype : int32
> The dimension of array : (2, 3)
```

Task no. 6:

Create a 1 D array by passing a list

Solution:

Task 6

```
: ┌ import numpy as np
A=np.array([4,5,6])
print("The type of array using dtype : ",A.dtype)
print("The dimension of array : ",A.shape)
print("The size of array using dtype : ",A.size)

> The type of array using dtype : int32
> The dimension of array : (3,)
> The size of array using dtype : 3
```

Task no. 7:

Create a 2 D array by passing lists

Solution:

Task 7

```
┐ import numpy as np
A=np.array(([1,2,3] ,[4,5,6]))
print("The type of array using dtype : ",A.dtype)
print("The dimension of array : ",A.shape)
print("The size of array using dtype : ",A.size)

> The type of array using dtype : int32
> The dimension of array : (2, 3)
> The size of array using dtype : 6
```

Task no. 8:

Create a 4 x 4 matrix

Solution:

Task 8

```
▶ import numpy as np
r1=[1,2,3,4]
r2=[4,5,6,7]
r3=[1,3,5,7]
r4=[2,4,6,8]
A=np.array((r1,r2,r3,r4))
print("\t4 x 4 Matrix \n",A)
print("> The type of array using dtype : ",A.dtype)
print("> The dimension of array : ",A.shape)
print("> The size of array using dtype : ",A.size)
```

```
        4 x 4 Matrix
[[1 2 3 4]
 [4 5 6 7]
 [1 3 5 7]
 [2 4 6 8]]
> The type of array using dtype :  int32
> The dimension of array :  (4, 4)
> The size of array using dtype :  16
```

Task no. 9:

Replace 10 on the 2nd index of a 2D array

Solution:

Task 9

```
▶ arr=np.array([[1,4,5,6]])
print("Original array = ",arr)
arr[0,1]=10
print("After Replacing",arr)
#rows and columns because its 2d
```

```
Original array =  [[1 4 5 6]]
After Replacing [[ 1 10  5  6]]
```

Task no. 10:

Replace 10 on the 2nd index of a 1D array

Solution:

Task 10

```
► arr=np.array([1,4,5,6])
print("Original array = ",arr)
arr[1]=10
print("After Replacing",arr)
```

```
Original array = [1 4 5 6]
After Replacing [ 1 10  5  6]
```

Task no. 11:

Create a 5 x 5 matrix of all zeros by setting values of both rows and column

Solution:

The **numpy.zeros()** function returns a new array of given shape and type, with zeros.

Task 11

```
► z=np.zeros([5,5])
print("5 x 5 Matrix of zeros \n",z)
print("> The type of array using dtype : ",z.dtype)
print("> The dimension of array : ",z.shape)
print("> The size of array using dtype : ",z.size)
```

```
5 x 5 Matrix of zeros
[0. 0. 0. 0. 0.]
> The type of array using dtype : float64
> The dimension of array : (5,)
> The size of array using dtype : 5
```

Task no. 12:

Create a 5×5 matrix of all zeros by only passing 1 argument.

Solution:

Task 12

```
: ┏━ z=np.zeros([5])
  └━ print("5 x 5 Matrix of zeros \n",z)
      print("> The type of array using dtype : ",z.dtype)
      print("> The dimension of array : ",z.shape)
      print("> The size of array using dtype : ",z.size)

  5 x 5 Matrix of zeros
  [0. 0. 0. 0. 0.]
  > The type of array using dtype :  float64
  > The dimension of array :  (5,)
  > The size of array using dtype :  5
```

Task no. 13:

Create an array from 1 to 100 by `numpy.arange()`

Solution:

Numpy.arange():

Values are generated within the half-open interval $[start, stop)$ (in other words, the interval including *start* but excluding *stop*). For integer arguments the function is equivalent to the Python built-in *range* function, but returns an ndarray rather than a list.

When using a non-integer step, such as 0.1, the results will often not be consistent. It is better to use **numpy.linspace** for these cases.

Task 13

```
┏━ arr=np.arange(1,100)
  └━ print("Printing range = \n",arr)
      print("> The type of array using dtype : ",arr.dtype)
      print("> The dimension of array : ",arr.shape)
      print("> The size of array using dtype : ",arr.size)

  Printing range =
  [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
  25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
  49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
  73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96
  97 98 99]
  > The type of array using dtype :  int32
  > The dimension of array :  (99,)
  > The size of array using dtype :  99
```

Task no. 14:

Create an array from 1 to 100 by numpy.arange() with a stepsize of 10

Solution:

Task 14

```
# arr=np.arange(1,100,10)
print("Printing range = \n",arr)
print("> The type of array using dtype : ",arr.dtype)
print("> The dimension of array : ",arr.shape)
print("> The size of array using dtype : ",arr.size)

Printing range =
[ 1 11 21 31 41 51 61 71 81 91]
> The type of array using dtype :  int32
> The dimension of array :  (10,)
> The size of array using dtype :  10
```

Task no. 15:

Create an array of 100 elements ranging from 2 to 3

Solution:

numpy.linspace():

- *Return evenly spaced numbers over a specified interval.*
- Returns *num* evenly spaced samples, calculated over the interval [*start*, *stop*].
- The endpoint of the interval can optionally be excluded.

Task 15

```
# arr=np.linspace(2,3,100)
print("Printing 100 values between 2 and 3 = \n",arr)
print("> The type of array using dtype : ",arr.dtype)
print("> The dimension of array : ",arr.shape)
print("> The size of array using dtype : ",arr.size)

Printing 100 values between 2 and 3 =
[2.          2.01010101 2.02020202 2.03030303 2.04040404 2.05050505
 2.06060606 2.07070707 2.08080808 2.09090909 2.1010101 2.11111111
 2.12121212 2.13131313 2.14141414 2.15151515 2.16161616 2.17171717
 2.18181818 2.19191919 2.2020202 2.21212121 2.22222222 2.23232323
 2.24242424 2.25252525 2.26262626 2.27272727 2.28282828 2.29292929
 2.3030303 2.31313131 2.32323232 2.33333333 2.34343434 2.35353535
 2.36363636 2.37373737 2.38383838 2.39393939 2.4040404 2.41414141
 2.42424242 2.43434343 2.44444444 2.45454545 2.46464646 2.47474747
 2.48484848 2.49494949 2.50505051 2.51515152 2.52525253 2.53535354
 2.54545455 2.55555556 2.56565657 2.57575758 2.58585859 2.59595956
 2.60606061 2.61616162 2.62626263 2.63636364 2.64646465 2.65656566
 2.66666667 2.67676768 2.68686869 2.6969697 2.70707071 2.71717172
 2.72727273 2.73737374 2.74747475 2.75757576 2.76767677 2.77777778
 2.78787879 2.7979798 2.80808081 2.81818182 2.82828283 2.83838384
 2.84848485 2.85858586 2.86868687 2.87878788 2.88888889 2.8989899
 2.90909091 2.91919192 2.92929293 2.93939394 2.94949495 2.95959596
 2.96969697 2.97979798 2.98989899 3.        ]
> The type of array using dtype :  float64
> The dimension of array :  (100,)
> The size of array using dtype :  100
```

Task no. 16:

Create an identity matrix

Solution:

Np.identity():

- return the identity array.
- The identity array is a square array with ones on the main diagonal.

Task 16

```
► i=np.identity(5)
print("Identity Matrix =\n",i)
print("> The type of array using dtype : ",i.dtype)
print("> The dimension of array : ",i.shape)
print("> The size of array using dtype : ",i.size)

Identity Matrix =
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
> The type of array using dtype :  float64
> The dimension of array :  (5, 5)
> The size of array using dtype :  25
```

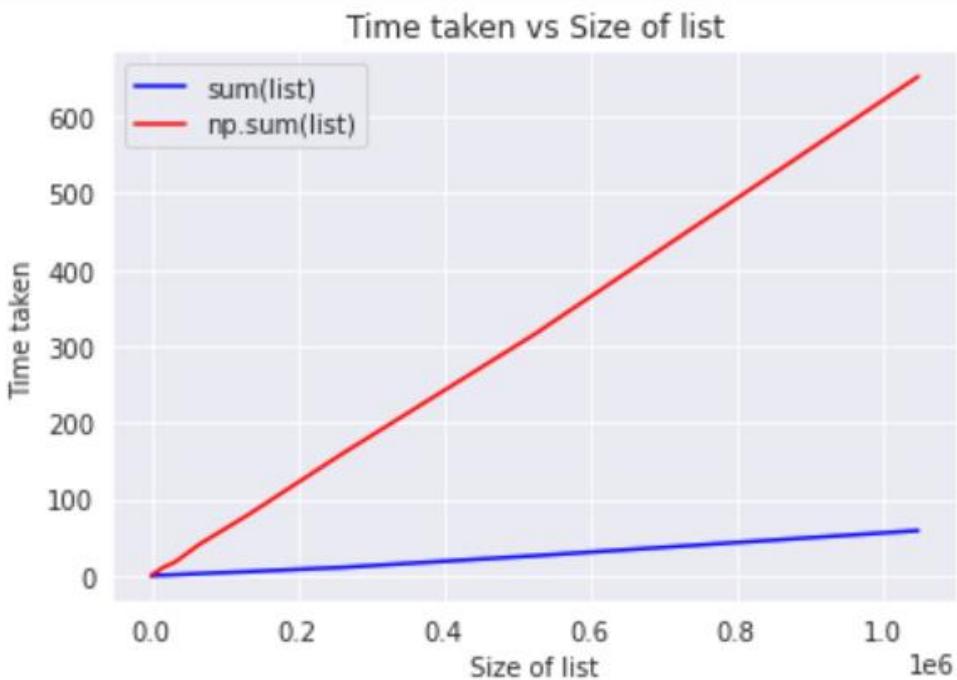
Task no. 17:

Find the sum of identity matrix

Solution:

Np.Sum(i) VS i.sum():

- Python's sum iterates over the iterable (in this case the list or array) and adds all elements.
- NumPy's sum **method** iterates over the stored C array and adds these C values and finally wraps that value in a Python type (in this case numpy.int32 (or numpy.int64)) and returns it.
- NumPy's sum **function** converts the input to an array (at least if it isn't an array already) and then uses the NumPy sum **method**.



It is to be noted that adding C values from a C array is much faster than adding Python objects, which is why the NumPy functions can be much faster. But converting a Python list to a NumPy array is relatively slow and then you still have to add the C values which is why for **lists** the Python's sum will be faster. Also, why is Python's sum on an array so slow. This has to do with the fact that Python's sum simply iterates over whatever you pass in. In case of a list it gets the stored Python object but in case of a 1D NumPy array there are no stored Python objects, just C values, so Python & NumPy have to create a Python object (an numpy.int32 or numpy.int64) for each element and then these Python objects have to be added. The creating the wrapper for the C value is what makes it really slow. This posts differentiates between the two in terms of their working and their performance when it comes to lists and arrays. Do try the same out with different objects such as tuples and ranges.

Task 17

```
|: ┌ print("> The sum of identity matrix is : ",np.sum(i))
|: ┌ print("> The sum of identity matrix is : ",i.sum())
> The sum of identity matrix is : 5.0
> The sum of identity matrix is : 5.0
```

Task no. 18:

Create a 4 x 4 matrix and find the sum of all columns

Solution:

numpy.sum(arr, axis) : This function returns the sum of array elements over the specified axis.

Parameters:

arr: input array.

axis : axis along which we want to calculate the sum value. Otherwise, it will consider arr to be flattened(works on all the axis). axis = 0 means along the column and axis = 1 means working along the row.

Return : Sum of the array elements (a scalar value if axis is none) or array with sum values along the specified axis.

Task 18

```
: ┏━ r1=[4,1,3,2]
  r2=[1,0,0,1]
  r3=[2,4,3,0]
  r4=[4,1,1,1]
  arr=np.array([r1,r2,r3,r4])
  print(arr)
  #print("sum of row ",np.sum(arr[1,:]))
  print("sum of row ",arr.sum(axis=1)) #if axis=0 column wise sum
  print("> The type of array using dtype : ",arr.dtype)
  print("> The dimension of array : ",arr.shape)
  print("> The size of array using dtype : ",arr.size)

  [[4 1 3 2]
   [1 0 0 1]
   [2 4 3 0]
   [4 1 1 1]]
  sum of row  [10  2  9  7]
  > The type of array using dtype :  int32
  > The dimension of array :  (4, 4)
  > The size of array using dtype :  16
```

Task no. 19:

Find the sum of rows and columns of an array

Solution:

Numpy.max():

Numpy max() function is used to get a maximum value along a specified axis.

Axis=0 for columns

Axis=1 for rows

Task 19

```
▶ print("Maximum value in row = ",arr.max(axis=1))
print("Maximum value in column = ",arr.max(axis=0))

Maximum value in row = [4 1 4 4]
Maximum value in column = [4 4 3 2]
```

Task no. 20:

Find the transpose of the matrix

Solution:

Numpy.T():

Permute the dimensions of an array except that self is returned if self.ndim < 2.

Task 20

```
▶ Atrans=arr.T
print("Transpose of the given matrix is = \n",Atrans)

Transpose of the given matrix is =
[[4 1 2 4]
 [1 0 4 1]
 [3 0 3 1]
 [2 1 0 1]]
```

Task no. 21:

Demonstrate the use of numpy.flat function

Solution:

Numpy.flat():

A 1-D iterator over the array.

This is a **numpy.flatiter** instance, which acts similarly to, but is not a subclass of, Python's built-in iterator object.

Task 21

```
▶ Aflat=Atrans.flat  
print("Flat matrix = ",Aflat)  
Flat matrix = <numpy.flatiter object at 0x000002B2B82F8B10>
```

Task no. 22:

Use reshape command to convert 4 x 4 matrix to 8 x 2

Solution:

Reshaping arrays:

- Reshaping means changing the shape of an array.
- The shape of an array is the number of elements in each dimension.
- By reshaping we can add or remove dimensions or change number of elements in each dimension.
- The numpy.reshape() function shapes an array without changing the data of the array.

Can We Reshape Into any Shape?

Yes, as long as the elements required for reshaping are equal in both shapes. We can reshape an 8 elements 1D array into 4 elements in 2 rows 2D array but we cannot reshape it into a 3 elements 3 rows 2D array as that would require $3 \times 3 = 9$ elements.

Task 22

```
▶ print("Reshaping to 8 x 2 = \n",arr.reshape(8,2)) #when reshaping number of elements should be equal  
Reshaping to 8 x 2 =  
[[4 1]  
[3 2]  
[1 0]  
[0 1]  
[2 4]  
[3 0]  
[4 1]  
[1 1]]
```

Task no. 23:

Demonstrate the use of numpy.ravel()

Solution:

numpy.ravel()

- Return a contiguous flattened array.
- A 1-D array, containing the elements of the input, is returned. A copy is made only if needed.
- As of NumPy 1.10, the returned array will have the same type as the input array. (for example, a masked array will be returned for a masked array input)

Task 23

```
▶ a=arr.reshape(8,2)
b=a.ravel()      #convert in single vector
print("Ravel = ",b)
```

```
Ravel = [4 1 3 2 1 0 0 1 2 4 3 0 4 1 1 1]
```

Task no. 24:

Demonstrate the use of argmax, argmin, argsort

Solution:

- The **numpy.argmax()** function returns indices of the max element of the array in a particular axis.
- The **numpy.argmin()** method returns indices of the min element of the array in a particular axis.
- **numpy.argsort()** function is used to perform an indirect sort along the given axis using the algorithm specified by the kind keyword. It returns an array of indices of the same shape as arr that would sort the array.

Task 24

```
: ▶ a=[1,16,31,4]
arr=np.array(a)
print(arr)
print("Index of maximum value = ",arr.argmax()) #arg provide indexes
print("Index of minimum value = ",arr.argmin())
print("Sorted indexes = ",arr.argsort())

[ 1 16 31  4]
Index of maximum value =  2
Index of minimum value =  0
Sorted indexes =  [0 3 1 2]
```

Task no. 25:

Demostrate the use of numpy.full(),vstack(),hstack(),column_stack()

Solution:

Numpy.full():

numpy.full(shape, fill_value, dtype = None, order = ‘C’) : Return a new array with the same shape and type as a given array filled with a fill_value.

Numpy.vstack():

Stack arrays in sequence vertically (row wise).This is equivalent to concatenation along the first axis after 1-D arrays of shape (N,) have been reshaped to (1,N). Rebuilds arrays divided by **vsplit**.

Numpy.hstack():

Stack arrays in sequence horizontally (column wise).This is equivalent to concatenation along the second axis, except for 1-D arrays where it concatenates along the first axis. Rebuilds arrays divided by **hsplit**.

Numpy.column_stack():

Stack 1-D arrays as columns into a 2-D array.Take a sequence of 1-D arrays and stack them as columns to make a single 2-D array. 2-D arrays are stacked as-is, just like with **hstack**. 1-D arrays are turned into 2-D columns first.

Task 25

```
► f1=np.full((2,2),5)
   print("\nf1 = \n",f1)
   f2=np.full((2,2),[[2,91],[4,86]])
   print("\nf2 = \n",f2)
   a=np.vstack([f1,f2])
   print("\nvstack = \n",a)
   b=np.hstack([f1,f2])
   print("\nhstack = \n",b)
   c=np.column_stack([f1,f2])
   print("\ncolumnstack = \n",c)
```

```
f1 =  
[[5 5]  
[5 5]]  
  
f2 =  
[[ 2 91]  
[ 4 86]]  
  
vstack =  
[[ 5 5]  
[ 5 5]  
[ 2 91]  
[ 4 86]]  
  
hstack =  
[[ 5 5 2 91]  
[ 5 5 4 86]]  
  
columnstack =  
[[ 5 5 2 91]  
[ 5 5 4 86]]
```

Task no. 26:

Save and load a matrix in the memory

Solution:

Saving and loading in numpy():

Save an array to a binary file in NumPy .npy format.

numpy.load() function return the input array from a disk file with npy extension(.npy).

Task 26

```
In [27]: np.save('untitled.npy',a)  
  
In [28]: np.load('untitled.npy')  
  
Out[27]: array([[ 5,  5],  
                 [ 5,  5],  
                 [ 2, 91],  
                 [ 4, 86]])
```

Task no. 27:

Demonstrate the use of numoy.dot() and compare it with simple multiplication

Solution:

Simple multiplication vs numpy.dot():

In Python if we have two numpy arrays which are often referred as a vector. The “*” operator and numpy.dot() work differently on them. It's important to know especially when you are dealing with data science or competitive programming problem.

Working of ‘*’ operator:

“*” operation carries out element-wise multiplication on array elements. The element at $a[i][j]$ is multiplied with $b[i][j]$. This happens for all elements of array.

Example:

Let the two 2D array are v1 and v2:-

v1 = [[1, 2], [3, 4]]

v2 = [[1, 2], [3, 4]]

Output:

[[1, 4]

[9, 16]]

The diagram shows the element-wise multiplication of two 2x2 matrices. On the left, matrix A is shown with elements 1, 2 in the top row and 3, 4 in the bottom row. On the right, matrix B is shown with elements 1, 2 in the top row and 3, 4 in the bottom row. An asterisk (*) indicates the multiplication operation between the two matrices. The resulting matrix on the right has elements 1, 4 in the top row and 9, 16 in the bottom row, representing the element-wise product of A and B.

Working of numpy.dot():

It carries of normal matrix multiplication . Where the condition of number of columns of first array should be equal to number of rows of second array is checked than only

numpy.dot() function take place else it shows an error .

Example:

Let the two 2D array are v1 and v2:-

v1=[[1, 2], [3, 4]]

v2=[[1, 2], [3, 4]]

Than numpy.dot(v1, v2) gives **output** of :-

[[7 10]

[15 22]]

Task 27

```
:  ► import numpy as np
f1=np.full((2,2),5)
print("\nf1 = \n",f1)
f2=np.full((2,2),[[2,91],[4,86]])
print("\nf2 = \n",f2)
print("point to point multiplication = ",f1*f2)
print("point to point multiplication = ",np.dot(f1,f2))

f1 =
[[5 5]
 [5 5]]

f2 =
[[ 2 91]
 [ 4 86]]
point to point multiplication =  [[ 10 455]
 [ 20 430]]
point to point multiplication =  [[ 30 885]
 [ 30 885]]
```

LAB 05

(CLO 1-3, PLO 1-5,P3)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	Introduction to Pandas
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice Basic Data Manipulation operations using Pandas Library and Function

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Pandas

Task no. 1:

Create a Dictionary and convert them into data frames also check its datatype

Data Frames:

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure.

Lab Tasks(Pandas,Matplot lib,Seaborn)

Task 1

```
► #pandas are used in machine Learning for data manipulation
#data frame(multidimensional i.e. columns & rows) and series(single column only)

► #create a dictionary
Stu_Dict=[{"Name":["Aqsa","Esha","Ayesha","Ayra","Arfa","Afsa","Abdul","Saadia","Abu Bakar","Atif"],
           "ID":["SID-1","SID-2","SID-3","SID-4","SID-5","SID-6","SID-7","SID-8","SID-9","SID-10"],
           "Roll_no":[1,2,3,4,5,6,7,8,9,10],
           "Semester": [7,7,7,6,6,6,5,8,8]}

► #convert into data frames
import pandas as pd
data=pd.DataFrame(Stu_Dict)
print(data)
#to check data type
print("\n\nThe data type of above given syntax is :",type(data))

      Name      ID  Roll_no  Semester
0    Aqsa  SID-1       1        7
1     Esha  SID-2       2        7
2   Ayesha  SID-3       3        7
3     Ayra  SID-4       4        7
4     Arfa  SID-5       5        6
5     Afsa  SID-6       6        6
6     Abdul  SID-7       7        6
7    Saadia  SID-8       8        5
8  Abu Bakar  SID-9       9        8
9      Atif  SID-10      10       8

The data type of above given syntax is : <class 'pandas.core.frame.DataFrame'>
```

Task no. 2:

Demonstrate the use of describe function for a data frame

Describe():The describe() method is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

Task 2

```
print("-----")
print("  Using Describe function")
print("-----\n")
print(data.describe())#describe the characteristics of the numeric value in the data set i.e. describes data
```

Using Describe function

	Roll_no	Semester
count	10.00000	10.00000
mean	5.50000	6.70000
std	3.02765	0.948683
min	1.00000	5.00000
25%	3.25000	6.00000
50%	5.50000	7.00000
75%	7.75000	7.00000
max	10.00000	8.00000

Task no. 3:

Demonstrate the use of head function

Head():

head() returns the first n rows(obsserve the index values). The default number of elements to display is five, but you may pass a custom number.

Task 3

```
:  print("-----")
:  print("\tUsing Head function")
:  print("-----\n")
:  print(data.head()) # top/upper most rows are displayed
```

Using Head function

	Name	ID	Roll_no	Semester
0	Aqsa	SID-1	1	7
1	Esha	SID-2	2	7
2	Ayesha	SID-3	3	7
3	Ayra	SID-4	4	7
4	Arfa	SID-5	5	6

Task no. 4:

Demonstrate the use of tail function

tail() returns the last n rows(obsserve the index values). The default number of elements to display is five, but you may pass a custom number.

Task 4

```
▶ print("-----")
print("\tUsing Tail function")
print("-----\n")
print(data.tail()) # Last/lower rows are displayed
```

Using Tail function

	Name	ID	Roll_no	Semester
5	Afsa	SID-6	6	6
6	Abdul	SID-7	7	6
7	Saadia	SID-8	8	5
8	Abu Bakar	SID-9	9	8
9	Atif	SID-10	10	8

Task no. 5:

Demonstrate the use of info function

Info():

The info() function is used to print a concise summary of a DataFrame. This method prints information about a DataFrame including the index dtype and column dtypes, non-null values and memory usage. Whether to print the full summary

Task 5

```
▶ print("-----")
print("\tUsing info function")
print("-----\n")
print(data.info()) # columns
```

Using info function

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Name        10 non-null    object  
 1   ID          10 non-null    object  
 2   Roll_no     10 non-null    int64  
 3   Semester    10 non-null    int64  
dtypes: int64(2), object(2)
memory usage: 448.0+ bytes
None
```

Task no. 6:

Convert the data frame in a variable to CSV file

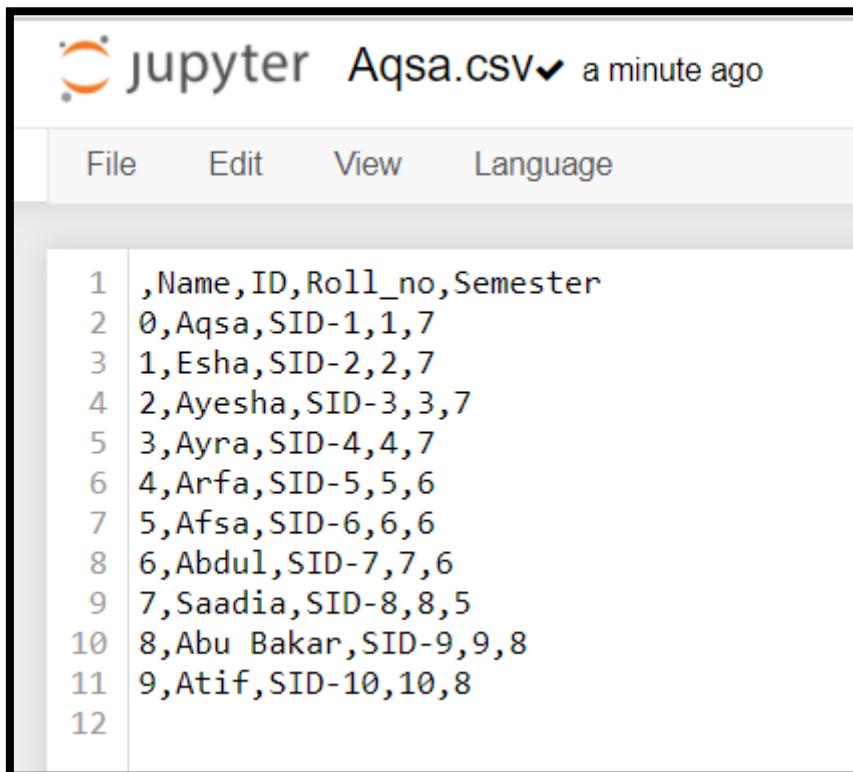
Data.to_csv:

Pandas DataFrame to_csv() function converts DataFrame into CSV data. We can pass a file object to write the CSV data into a file. Otherwise, the CSV data is returned in the string format.

Task 6

```
► #converting to csv file having indexes  
data.to_csv('Aqsa.csv')
```

CSV File:



The screenshot shows a Jupyter Notebook interface with a single cell containing the output of a CSV file. The cell title is "Aqsa.csv" and it was run "a minute ago". The output displays 12 rows of data, each consisting of a row number and four columns: Name, ID, Roll_no, and Semester. The data is as follows:

	Name	ID	Roll_no	Semester
1	Aqsa	SID-1	1	7
2	Esha	SID-2	2	7
3	Ayesha	SID-3	3	7
4	Ayra	SID-4	4	7
5	Arfa	SID-5	5	6
6	Afsa	SID-6	6	6
7	Abdul	SID-7	7	6
8	Saadia	SID-8	8	5
9	Abu Bakar	SID-9	9	8
10	Atif	SID-10	10	8
11				
12				

Task no. 7:

Remove the indexes from the csv file

If we want to convert this DataFrame to a CSV file without the index column, we can do it by setting the index to be False in the to_csv() function. As seen in the output, the DataFrame does have an index, but since we set the index parameter to False , the exported CSV file won't have that extra column.

Task 7

```
#removing indexes from csv  
data.to_csv('Without_index',index=False)
```

CSV File:

	Name	ID	Roll_no	Semester
1	Aqsa	SID-1	1,7	
2	Esha	SID-2	2,7	
3	Ayesha	SID-3	3,7	
4	Ayra	SID-4	4,7	
5	Arfa	SID-5	5,6	
6	Afsa	SID-6	6,6	
7	Abdul	SID-7	7,6	
8	Saadia	SID-8	8,5	
9	Abu Bakar	SID-9	9,8	
10	Atif	SID-10	10,8	
11				
12				

Task no. 8:

Read from CSV File

read_csv is an important pandas function to read csv files and do operations on it.

Task 8

```
► df=pd.read_csv('Aqsa.csv')  
print(df)
```

	Unnamed: 0	Name	ID	Roll_no	Semester
0	0	Aqsa	SID-1	1	7
1	1	Esha	SID-2	2	7
2	2	Ayesha	SID-3	3	7
3	3	Ayra	SID-4	4	7
4	4	Arfa	SID-5	5	6
5	5	Afsa	SID-6	6	6
6	6	Abdul	SID-7	7	6
7	7	Saadia	SID-8	8	5
8	8	Abu Bakar	SID-9	9	8
9	9	Atif	SID-10	10	8

Task no. 9:

Use *describe*,*head*,*tail* and *info* function for CSV file

Task 9

```
► df=pd.read_csv('Aqsa.csv')  
print("\nDescribe Function :\n\n",df.describe())  
print("\nHead Function :\n\n",df.head())  
print("\nTail Function :\n\n",df.tail())  
print("\nInfo Function :\n\n")  
print(df.info())
```

Describe Function :

```
      Unnamed: 0    Roll_no    Semester
count      10.00000   10.00000   10.000000
mean       4.50000   5.50000   6.700000
std        3.02765   3.02765   0.948683
min        0.00000   1.00000   5.000000
25%       2.25000   3.25000   6.000000
50%       4.50000   5.50000   7.000000
75%       6.75000   7.75000   7.000000
max       9.00000   10.00000  8.000000
```

Head Function :

```
      Unnamed: 0      Name     ID    Roll_no    Semester
0            0      Aqsa  SID-1        1        7
1            1      Esha  SID-2        2        7
2            2    Ayesha  SID-3        3        7
3            3      Ayra  SID-4        4        7
4            4      Arfa  SID-5        5        6
```

Tail Function :

```
      Unnamed: 0      Name     ID    Roll_no    Semester
5            5      Afsa  SID-6        6        6
6            6     Abdul  SID-7        7        6
7            7    Saadia  SID-8        8        5
8            8  Abu Bakar  SID-9        9        8
9            9      Atif  SID-10       10        8
```

Info Function :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #  Column      Non-Null Count  Dtype  
--- 
 0  Unnamed: 0    10 non-null    int64  
 1  Name         10 non-null    object  
 2  ID           10 non-null    object  
 3  Roll_no      10 non-null    int64  
 4  Semester     10 non-null    int64  
dtypes: int64(3), object(2)
memory usage: 528.0+ bytes
None
```

Task no. 10:

Access a column by its name

Task 10

```
: ┆ #excessing single column  
df['Name']
```

```
[21]: 0          Aqsa  
       1          Esha  
       2         Ayesha  
       3          Ayra  
       4          Arfa  
       5          Afsa  
       6         Abdul  
       7        Saadia  
       8      Abu Bakar  
       9          Atif  
Name: Name, dtype: object
```

Task no. 11:

Access the 1st element of a column

Task 11

```
: ┆ #excessing 1st element of name column  
df['Name'][0]
```

```
[23]: 'Aqsa'
```

Task no. 12:

Update a value in the column

Task 12

```
▶ #updating value in column  
df['Name'][0] = "Sierra"  
print("Updated frame = \n", df)
```

```
Updated frame =  
      Unnamed: 0      Name       ID  Roll_no  Semester  
0            0    Sierra  SID-1       1         7  
1            1      Esha  SID-2       2         7  
2            2   Ayesha  SID-3       3         7  
3            3     Ayra  SID-4       4         7  
4            4     Arfa  SID-5       5         6  
5            5     Afsa  SID-6       6         6  
6            6    Abdul  SID-7       7         6  
7            7   Saadia  SID-8       8         5  
8            8  Abu Bakar  SID-9       9         8  
9            9     Atif  SID-10      10         8
```

Task no. 13:

Find the columns and indexes in a data frame

DataFrame.columns

The column labels of the DataFrame.

DataFrame.index

The index (row labels) of the DataFrame.

Task 13

```
▶ print("Columns in data Frame = ", df.columns)  
print("Indexes in data Frame = ", df.index)
```

```
Columns in data Frame = Index(['Unnamed: 0', 'Name', 'ID', 'Roll_no', 'Semester'], dtype='object')  
Indexes in data Frame = RangeIndex(start=0, stop=10, step=1)
```

Task no. 14:

Create a series of 50 random numbers and check their data type and shape

Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.

Task 14

```
#printing series
import numpy as np
s=pd.Series(np.random.rand(50))
print(s)
print("Using dtype : ",np.dtype(s))
print("Using type : ",type(s))
print("Using shape : ",np.shape(s))
```

```
0    0.113932
1    0.298590
2    0.650400
3    0.781452
4    0.485316
5    0.100169
6    0.596690
7    0.382132
8    0.428653
9    0.002994
10   0.325282
11   0.798802
12   0.065001
13   0.646920
14   0.187512
15   0.706715
16   0.452282
17   0.022998
18   0.812367
19   0.102254
20   0.833611
21   0.467484
22   0.766674
23   0.270217
24   0.230594
25   0.558150
26   0.354865
27   0.971753
28   0.467195
```

```
29   0.388652
30   0.017492
31   0.503365
32   0.611916
33   0.774081
34   0.616089
35   0.139845
36   0.890991
37   0.766778
38   0.107543
39   0.385073
40   0.270829
41   0.133072
42   0.923962
43   0.824831
44   0.418110
45   0.359430
46   0.717067
47   0.360459
48   0.819464
49   0.502900
dtype: float64
Using dtype :  float64
Using type : <class 'pandas.core.series.Series'>
Using shape :  (50,)
```

Task no. 15:

Create a 50 x 5 data set from random values

Task 15

```
▶ #creating 50 x 5 data set from random values  
dataf=pd.DataFrame(np.random.rand(50,5))  
print(dataf)
```

0	0.094962	0.967399	0.870565	0.152312	0.788467
1	0.410245	0.430224	0.741084	0.686065	0.882180
2	0.831359	0.813040	0.651404	0.411039	0.311182
3	0.583211	0.989624	0.432424	0.822741	0.578906
4	0.672311	0.962876	0.347203	0.499454	0.272077
5	0.282468	0.692010	0.085945	0.236833	0.620393
6	0.825945	0.845408	0.345600	0.477379	0.576652
7	0.570140	0.509748	0.302237	0.590486	0.906585
8	0.838134	0.199787	0.116606	0.676304	0.869001
9	0.770665	0.464550	0.203465	0.339543	0.592015
10	0.035244	0.937535	0.496007	0.510602	0.639925
11	0.217091	0.872672	0.716851	0.023274	0.742585
12	0.310766	0.141865	0.062944	0.347324	0.748089
13	0.921795	0.519870	0.977449	0.447276	0.755288
14	0.646517	0.585067	0.304640	0.670232	0.691378
15	0.313117	0.263691	0.482159	0.425714	0.479258
16	0.358009	0.913182	0.081566	0.089380	0.347058
17	0.252173	0.020024	0.211393	0.565776	0.737216
18	0.740405	0.661884	0.740350	0.847878	0.180923
19	0.430050	0.644161	0.273812	0.662908	0.008411
20	0.297693	0.275966	0.660264	0.752230	0.008089
21	0.993513	0.284765	0.387372	0.245187	0.907365
22	0.462112	0.666517	0.794615	0.379071	0.858499
23	0.830467	0.922895	0.362753	0.442325	0.066728
24	0.887667	0.574220	0.558578	0.776452	0.222403
25	0.087204	0.578986	0.618343	0.296245	0.010016
26	0.881723	0.642017	0.148139	0.393283	0.561768
27	0.603282	0.328548	0.555388	0.453224	0.075012
28	0.048082	0.374357	0.799392	0.178253	0.783095
29	0.968448	0.684144	0.453310	0.377193	0.109069
30	0.194752	0.887519	0.647682	0.477079	0.817529
31	0.902753	0.769571	0.606887	0.350952	0.709144
32	0.926063	0.347599	0.596461	0.923828	0.736493
33	0.842098	0.640310	0.856703	0.283169	0.639744
34	0.777959	0.703186	0.673254	0.135506	0.202832
35	0.446190	0.919686	0.494257	0.613030	0.761471
36	0.310280	0.805924	0.943170	0.854213	0.704417
37	0.159416	0.259688	0.863710	0.343536	0.087491
38	0.880394	0.431513	0.274577	0.046114	0.221450
39	0.984284	0.572065	0.433954	0.649175	0.158406
40	0.476638	0.923671	0.624877	0.020481	0.293166
41	0.169351	0.043293	0.419437	0.972303	0.254046
42	0.804143	0.772792	0.234535	0.912694	0.994494
43	0.898257	0.935198	0.684354	0.051571	0.410761
44	0.407155	0.074626	0.582713	0.314586	0.716908
45	0.837013	0.508406	0.793916	0.045091	0.259168
46	0.294357	0.009460	0.744084	0.623614	0.844292
47	0.060526	0.446291	0.593897	0.506072	0.099769
48	0.460307	0.184787	0.432950	0.648736	0.420524
49	0.495737	0.918560	0.116149	0.839676	0.065992

Task no. 16:

Find the minimum maximum and mean values column wise in a dataset

Task 16

```
► #min,max,mean column wise
print("-----\n")
print("The minimum value in the dataset \n\n",dataf.min())
print("-----\n")
print("The maximum value in the dataset \n\n",dataf.max())
print("-----\n")
print("The mean of the dataset \n\n",dataf.mean())
-----
```

The minimum value in the dataset

```
0    0.035244
1    0.009460
2    0.062944
3    0.020481
4    0.008089
dtype: float64
-----
```

The maximum value in the dataset

```
0    0.993513
1    0.989624
2    0.977449
3    0.972303
4    0.994494
dtype: float64
-----
```

The mean of the dataset

```
0    0.549849
1    0.578424
2    0.507989
3    0.467748
4    0.494555
dtype: float64
```

Task no. 17:

Find the maximum value in 1st column

Task 17

```
► #for finding max in 1 column  
print("Maximum value is 1st column is ",dataf[1].max)
```

```
Maximum value is 1st column is <bound method NDFrame._add_numeric_operations.<locals>.max of 0      0.967399  
1    0.430224  
2    0.813040  
3    0.989624  
4    0.962876  
5    0.692010  
6    0.845408  
7    0.509748  
8    0.199787  
9    0.464550  
10   0.937535  
11   0.872672  
12   0.141865  
13   0.519870  
14   0.585067  
15   0.263691  
16   0.913182  
17   0.020024  
18   0.661884  
19   0.644161  
20   0.275966  
21   0.284765  
22   0.666517  
23   0.922895  
24   0.574220  
25   0.578986  
26   0.642017  
27   0.328548  
28   0.374357  
29   0.684144  
30   0.887519  
31   0.769571  
32   0.347599  
33   0.640310  
34   0.703186  
35   0.919686  
36   0.805924  
37   0.259688  
38   0.431513  
39   0.572065  
40   0.923671  
41   0.043293  
42   0.772792  
43   0.935198  
44   0.074626  
45   0.508406  
46   0.009460  
47   0.446291  
48   0.184787  
49   0.918560  
Name: 1, dtype: float64>
```

Task no. 18:

Convert the dataset into numpy array and also take transpose of it

Task 18

```
► d1=dataf.to_numpy()
print("-----")
print("\t\tConverting to numpy array \n\n",d1)
print("\n-----")
print("\t\tTaking Transpose \n\n",d1.T)
```

Converting to numpy array

```
[[0.09496224 0.96739889 0.87056457 0.15231237 0.78846697]
[0.41024459 0.43022436 0.74108434 0.6860652 0.88218016]
[0.83135923 0.8130404 0.65140396 0.41103938 0.31118153]
[0.58321149 0.98962406 0.43242438 0.82274073 0.57890574]
[0.67231134 0.96287554 0.3472932 0.49945402 0.2720768]
[0.28246783 0.69201017 0.08594535 0.23683349 0.62039289]
[0.82594469 0.84540792 0.34559998 0.47737903 0.5766515]
[0.5701397 0.50974771 0.30223669 0.59048574 0.90658528]
[0.8381343 0.19978683 0.11660604 0.67630383 0.86900145]
[0.77066515 0.46455027 0.2034653 0.33954297 0.59201452]
[0.03524388 0.93753501 0.49600665 0.51060246 0.63992535]
[0.21709121 0.87267215 0.7168508 0.02327441 0.74258531]
[0.31076565 0.14186465 0.06294447 0.34732414 0.74808945]
[0.92179504 0.51986998 0.97744945 0.44727618 0.7552876]
[0.6465167 0.58506677 0.30464031 0.67023152 0.69137835]
[0.31311718 0.26369111 0.48215915 0.42571432 0.47925752]
[0.35800938 0.91318212 0.08156639 0.08937985 0.34705844]
[0.25217341 0.0200244 0.21139318 0.56577601 0.73721602]
[0.74040469 0.66188429 0.74034999 0.84787762 0.18092331]
[0.4300497 0.64416091 0.27381183 0.66290824 0.00841065]
[0.29769328 0.27596616 0.66026412 0.75222983 0.00808919]
[0.99351306 0.28476486 0.38737162 0.2451871 0.90736508]
[0.46211246 0.66651683 0.79461462 0.37907137 0.85849947]
[0.83046676 0.92289529 0.36275287 0.44232542 0.06672824]
[0.88766734 0.57422013 0.5585783 0.77645232 0.22240326]
[0.08720351 0.57898581 0.6183432 0.29624487 0.01001607]
[0.88172301 0.64201675 0.14813874 0.39328251 0.56176787]
[0.60328222 0.32854837 0.55538755 0.45322375 0.07501245]
[0.04808203 0.3743566 0.79939198 0.17825271 0.78309538]
[0.96844763 0.68414372 0.45331019 0.37719319 0.10906852]
[0.19475195 0.88751857 0.6476824 0.47707893 0.8175287]
[0.90275286 0.76957111 0.60688723 0.35095242 0.70914356]
[0.92066292 0.34759895 0.59646138 0.92382782 0.73649342]
[0.84209806 0.64031026 0.85670297 0.28316937 0.63974355]
[0.77795894 0.70318622 0.67325439 0.13550629 0.20283215]
[0.44618954 0.91968553 0.49425728 0.61302968 0.76147076]
[0.31027992 0.80592376 0.94316983 0.85421259 0.70441692]
[0.15941623 0.25968786 0.86371041 0.34353635 0.08749113]
[0.8803942 0.43151266 0.27457681 0.04611443 0.22145019]
[0.98428366 0.57206545 0.43395425 0.64917548 0.15840648]
[0.47663808 0.92367105 0.62487715 0.02048144 0.29316619]
[0.16935096 0.04329326 0.41943695 0.97230346 0.25404552]
[0.88414328 0.77279214 0.23453457 0.9126944 0.99449371]
[0.89825708 0.93519816 0.68435428 0.05157074 0.41076144]
[0.40715546 0.07462614 0.58271275 0.31458574 0.7169084]
[0.837013 0.50840572 0.79391627 0.04509143 0.25916818]
[0.29435712 0.00946035 0.74408407 0.62361411 0.84429175]
[0.06052577 0.44629128 0.59389692 0.50607193 0.09976903]
[0.46030698 0.18478665 0.43295016 0.64873587 0.42052419]
[0.49573711 0.91855996 0.11614903 0.83967602 0.06599247]]
```

Taking Transpose

```
[[0.09496224 0.41024459 0.83135923 0.58321149 0.67231134 0.28246783  
0.82594469 0.5701397 0.8381343 0.77066515 0.03524388 0.21709121  
0.31076565 0.92179504 0.6465167 0.31311718 0.35800938 0.25217341  
0.74040469 0.4300497 0.29769328 0.99351306 0.46211246 0.83046676  
0.88766734 0.08720351 0.88172301 0.60328222 0.04808203 0.96844763  
0.19475195 0.90275286 0.92606292 0.84209806 0.77795894 0.44618954  
0.31027992 0.15941623 0.8803942 0.98428366 0.47663808 0.16935096  
0.88414328 0.89825708 0.40715546 0.837013 0.29435712 0.06052577  
0.46030698 0.49573711]  
[0.96739889 0.43022436 0.8130404 0.98962406 0.96287554 0.69201017  
0.84540792 0.50974771 0.19978683 0.46455027 0.93753501 0.87267215  
0.14186465 0.51986998 0.58506677 0.26369111 0.91318212 0.0200244  
0.66188429 0.64416091 0.27596616 0.28476486 0.66651683 0.92289529  
0.57422013 0.57898581 0.64201675 0.32854837 0.3743566 0.68414372  
0.88751857 0.76957111 0.34759895 0.64031026 0.70318622 0.91968553  
0.88592376 0.25968786 0.43151266 0.57206545 0.92367105 0.04329326  
0.77279214 0.93519816 0.07462614 0.50840572 0.00946035 0.44629128  
0.18478665 0.91855996]  
[0.87056457 0.74108434 0.65140396 0.43242438 0.3472032 0.08594535  
0.34559998 0.30223669 0.11660604 0.2034653 0.49600665 0.7168508  
0.06294447 0.97744945 0.30464031 0.48215915 0.08156639 0.21139318  
0.74034999 0.27381183 0.66026412 0.38737162 0.79461462 0.36275287  
0.5585783 0.6183432 0.14813874 0.55538755 0.79939198 0.45331019  
0.6476824 0.60688723 0.59646138 0.85670297 0.67325439 0.49425728  
0.94316983 0.86371041 0.27457681 0.43395425 0.62487715 0.41943695  
0.23453457 0.68435428 0.58271275 0.79391627 0.74408407 0.59389692  
0.43295016 0.11614903]  
[0.15231237 0.6860652 0.41103938 0.82274073 0.49945402 0.23683349  
0.47737903 0.59048574 0.67630383 0.33954297 0.51060246 0.02327441  
0.34732414 0.44727618 0.67023152 0.42571432 0.08937985 0.56577601  
0.84787762 0.66290824 0.75222983 0.2451871 0.37907137 0.44232542  
0.77645232 0.29624487 0.39328251 0.45322375 0.17825271 0.37719319  
0.47707893 0.35095242 0.92382782 0.28316937 0.13550629 0.61302968  
0.85421259 0.34353635 0.04611443 0.64917548 0.02048144 0.97230346  
0.9126944 0.05157074 0.31458574 0.04509143 0.62361411 0.50607193  
0.64873587 0.83967602]  
[0.78846697 0.88218016 0.31118153 0.57890574 0.2720768 0.62039289  
0.5766515 0.90658528 0.86900145 0.59201452 0.63992535 0.74258531  
0.74808945 0.7552876 0.69137835 0.47925752 0.34705844 0.73721602  
0.18092331 0.00841065 0.00808919 0.90736508 0.85849947 0.06672824  
0.22240326 0.01001607 0.56176787 0.07501245 0.78309538 0.10906852  
0.8175287 0.70914356 0.73649342 0.63974355 0.20283215 0.76147076  
0.70441692 0.08749113 0.22145019 0.15840648 0.29316619 0.25404552  
0.99449371 0.41076144 0.7169084 0.25916818 0.84429175 0.09976903  
0.42052419 0.06599247]]
```

Task no. 19:

Change names of the columns.

Task 19

```
► #changing name of columns
dataf.columns=['A','B','C','D','E']
print(dataf)

      A        B        C        D        E
0  0.094962  0.967399  0.870565  0.152312  0.788467
1  0.410245  0.430224  0.741084  0.686065  0.882180
2  0.831359  0.813040  0.651404  0.411039  0.311182
3  0.583211  0.989624  0.432424  0.822741  0.578906
4  0.672311  0.962876  0.347203  0.499454  0.272077
5  0.282468  0.692010  0.085945  0.236833  0.620393
6  0.825945  0.845408  0.345600  0.477379  0.576652
7  0.570140  0.509748  0.302237  0.590486  0.906585
8  0.838134  0.199787  0.116606  0.676304  0.869001
9  0.770665  0.464550  0.203465  0.339543  0.592015
10 0.035244  0.937535  0.496007  0.510602  0.639925
11 0.217091  0.872672  0.716851  0.023274  0.742585
12 0.310766  0.141865  0.062944  0.347324  0.748089
13 0.921795  0.519870  0.977449  0.447276  0.755288
14 0.646517  0.585067  0.304640  0.670232  0.691378
15 0.313117  0.263691  0.482159  0.425714  0.479258
16 0.358009  0.913182  0.081566  0.089380  0.347058
17 0.252173  0.020024  0.211393  0.565776  0.737216
18 0.740405  0.661884  0.740350  0.847878  0.180923
19 0.430050  0.644161  0.273812  0.662908  0.008411
20 0.297693  0.275966  0.660264  0.752230  0.008089
21 0.993513  0.284765  0.387372  0.245187  0.907365
22 0.462112  0.666517  0.794615  0.379071  0.858499
23 0.830467  0.922895  0.362753  0.442325  0.066728
24 0.887667  0.574220  0.558578  0.776452  0.222403
25 0.087204  0.578986  0.618343  0.296245  0.010016
26 0.881723  0.642017  0.148139  0.393283  0.561768
27 0.603282  0.328548  0.555388  0.453224  0.075012
28 0.048082  0.374357  0.799392  0.178253  0.783095
29 0.968448  0.684144  0.453310  0.377193  0.109069
30 0.194752  0.887519  0.647682  0.477079  0.817529
31 0.902753  0.769571  0.606887  0.350952  0.709144
32 0.926063  0.347599  0.596461  0.923828  0.736493
33 0.842098  0.640310  0.856703  0.283169  0.639744
34 0.777959  0.703186  0.673254  0.135506  0.202832
35 0.446190  0.919686  0.494257  0.613030  0.761471
36 0.310280  0.805924  0.943170  0.854213  0.704417
37 0.159416  0.259688  0.863710  0.343536  0.087491
38 0.880394  0.431513  0.274577  0.046114  0.221450
39 0.984284  0.572065  0.433954  0.649175  0.158406
40 0.476638  0.923671  0.624877  0.020481  0.293166
41 0.169351  0.043293  0.419437  0.972303  0.254046
42 0.804143  0.772792  0.234535  0.912694  0.994494
43 0.898257  0.935198  0.684354  0.051571  0.410761
44 0.407155  0.074626  0.582713  0.314586  0.716908
45 0.837013  0.508406  0.793916  0.045091  0.259168
46 0.294357  0.009460  0.744084  0.623614  0.844292
47 0.060526  0.446291  0.593897  0.506072  0.099769
48 0.460307  0.184787  0.432950  0.648736  0.420524
49 0.495737  0.918560  0.116149  0.839676  0.065992
```

Task no. 20:

Display column B and C from the dataset

Task 20

```
► #gives error
# print(dataaf['B', 'C'])
# pass as a list to display columns
print(dataaf[['B', 'C']])
```

	B	C
0	0.967399	0.870565
1	0.430224	0.741084
2	0.813040	0.651404
3	0.989624	0.432424
4	0.962876	0.347203
5	0.692010	0.085945
6	0.845408	0.345600
7	0.509748	0.302237
8	0.199787	0.116606
9	0.464550	0.203465
10	0.937535	0.496007
11	0.872672	0.716851
12	0.141865	0.062944
13	0.519870	0.977449
14	0.585067	0.304640
15	0.263691	0.482159
16	0.913182	0.081566
17	0.020024	0.211393
18	0.661884	0.740350
19	0.644161	0.273812
20	0.275966	0.660264
21	0.284765	0.387372
22	0.666517	0.794615
23	0.922895	0.362753
24	0.574220	0.558578
25	0.578986	0.618343
26	0.642017	0.148139
27	0.328548	0.555388
28	0.374357	0.799392
29	0.684144	0.453310
30	0.887519	0.647682
31	0.769571	0.606887
32	0.347599	0.596461
33	0.640310	0.856703
34	0.703186	0.673254
35	0.919686	0.494257
36	0.805924	0.943170
37	0.259688	0.863710
38	0.431513	0.274577
39	0.572065	0.433954
40	0.923671	0.624877
41	0.043293	0.419437
42	0.772792	0.234535
43	0.935198	0.684354
44	0.074626	0.582713
45	0.508406	0.793916
46	0.009460	0.744084
47	0.446291	0.593897
48	0.184787	0.432950
49	0.918560	0.116149

Task no. 21:

Use head function

Task 21

```
▶ print(dataf.head())
      A        B        C        D        E
0  0.094962  0.967399  0.870565  0.152312  0.788467
1  0.410245  0.430224  0.741084  0.686065  0.882180
2  0.831359  0.813040  0.651404  0.411039  0.311182
3  0.583211  0.989624  0.432424  0.822741  0.578906
4  0.672311  0.962876  0.347203  0.499454  0.272077
```

Task no. 22:

Demonstrate the use of iloc function

Iloc: iloc is integer position-based, so you have to specify rows and columns by their integer position values (0-based integer position).

Task 22

```
▶ print(dataf.iloc[:,0:2]) #access using indexes so well use iloc if we have to access column we have to give proper name of col
      A        B
0  0.094962  0.967399
1  0.410245  0.430224
2  0.831359  0.813040
3  0.583211  0.989624
4  0.672311  0.962876
5  0.282468  0.692010
6  0.825945  0.845408
7  0.570140  0.509748
8  0.838134  0.199787
9  0.770665  0.464550
10 0.035244  0.937535
11 0.217091  0.872672
12 0.310766  0.141865
13 0.921795  0.519870
14 0.646517  0.585067
15 0.313117  0.263691
16 0.358009  0.913182
17 0.252173  0.020024
18 0.740405  0.661884
19 0.430050  0.644161
20 0.297693  0.275966
21 0.993513  0.284765
22 0.462112  0.666517
23 0.830467  0.922895
24 0.887667  0.574220
25 0.087204  0.578986
26 0.881723  0.642017
27 0.603282  0.328548
28 0.048082  0.374357
29 0.968448  0.684144
30 0.194752  0.887519
31 0.902753  0.769571
32 0.926063  0.347599
33 0.842098  0.640310
34 0.777959  0.703186
35 0.446190  0.919686
36 0.310280  0.805924
37 0.159416  0.259688
38 0.880394  0.431513
39 0.984284  0.572065
40 0.476638  0.923671
41 0.169351  0.043293
42 0.804143  0.772792
43 0.898257  0.935198
44 0.407155  0.074626
45 0.837013  0.508406
46 0.294357  0.009460
47 0.060526  0.446291
48 0.460307  0.184787
49 0.495737  0.918560
```

Task no. 23:

Print column A to C and find the value on 0,0

Loc:

loc is label-based, which means that you have to specify rows and columns based on their row and column labels.

Task 23

```
: ➜ print(dataaf.loc[:, "A":"C"])
print("\nvalue on 0,0 using iloc :", dataaf.iloc[0,0])
print("\nvalue on 0,0 using loc : ", dataaf.loc[0, "A"])

      A        B        C
0  0.057626  0.762036  0.780251
1  0.065522  0.555099  0.711657
2  0.654892  0.376889  0.398426
3  0.753755  0.499594  0.588186
4  0.000876  0.604123  0.154795
5  0.488721  0.204174  0.180558
6  0.121139  0.503304  0.422514
7  0.078020  0.204490  0.649066
8  0.236422  0.361264  0.418481
9  0.950761  0.423689  0.874363
10 0.767785  0.478894  0.309020
11 0.272708  0.444775  0.038812
12 0.944971  0.868106  0.206321
13 0.208303  0.918467  0.167093
14 0.844017  0.992720  0.669934
15 0.800014  0.496467  0.891681
16 0.169285  0.814668  0.838501
17 0.772124  0.516441  0.021734
18 0.994474  0.946151  0.564170
19 0.782237  0.793920  0.717260
20 0.567038  0.428268  0.364362
21 0.909236  0.672828  0.425320
22 0.270920  0.526777  0.845231
23 0.876884  0.870506  0.004045
24 0.618712  0.975558  0.667233
25 0.424044  0.659217  0.910856
26 0.017806  0.162268  0.839673
27 0.382670  0.183211  0.225167
28 0.286211  0.529198  0.154539
29 0.091717  0.705808  0.805876
30 0.323401  0.676608  0.218677
31 0.056152  0.919117  0.013344
32 0.215945  0.122286  0.149577
33 0.156992  0.556132  0.257969
34 0.593733  0.642551  0.319709
35 0.544364  0.627443  0.819122
36 0.549163  0.832855  0.935878
37 0.984776  0.428544  0.206843
38 0.660637  0.741615  0.275145
39 0.620504  0.992189  0.785994
40 0.673893  0.887586  0.369984
41 0.117486  0.171085  0.287998
42 0.162369  0.721662  0.991039
43 0.546014  0.281963  0.021852
44 0.389452  0.671938  0.027151
45 0.745482  0.455814  0.893112
46 0.082729  0.457692  0.320132
47 0.734280  0.103402  0.023114
48 0.379591  0.954774  0.606877
49 0.156945  0.913780  0.002900

value on 0,0 using iloc : 0.05762556309821454
value on 0,0 using loc : 0.05762556309821454
```

Task no. 24:

Print 1st 12 elements of column 2 and 4

Task 24

```
▶ print(dataf.iloc[0:12,2:4]) #in iloc 2nd value after column is discarded but not in loc
```

	C	D
0	0.870565	0.152312
1	0.741084	0.686065
2	0.651404	0.411039
3	0.432424	0.822741
4	0.347203	0.499454
5	0.085945	0.236833
6	0.345600	0.477379
7	0.302237	0.590486
8	0.116606	0.676304
9	0.203465	0.339543
10	0.496007	0.510602
11	0.716851	0.023274

LAB 06

(CLO 1-3, PLO 1-5,P3)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	Introduction to Matplotlib and Seaborn
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice Basics Data visualization techniques with the help of graphes using Matplotlib and Seaborn Libraries

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Matplotlib, seaborn

Task no. 1:

Import matplotlib

Task 25 (matplotlib)

```
► import numpy as np  
import matplotlib.pyplot as plt #from matplotlib import pyplot as plt
```

Task no. 2:

Use plot and show functions to create a graph

Plot():

The plot() function is used to draw points (markers) in a diagram.

By default, the plot() function draws a line from point to point.

The function takes parameters for specifying points in the diagram.

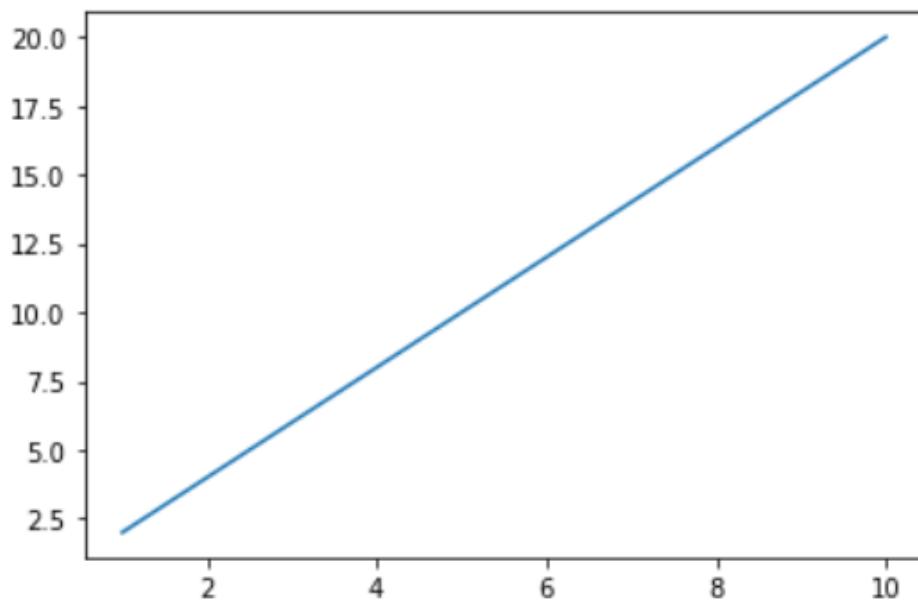
Parameter 1 is an array containing the points on the **x-axis**.

Parameter 2 is an array containing the points on the **y-axis**.

Task 26

```
▶ x=np.array((1,2,3,4,5,6,7,8,9,10))  
y=x*2  
print(x)  
  
plt.plot(x,y)  
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```



Task no. 3:

Add labels and title to the graph

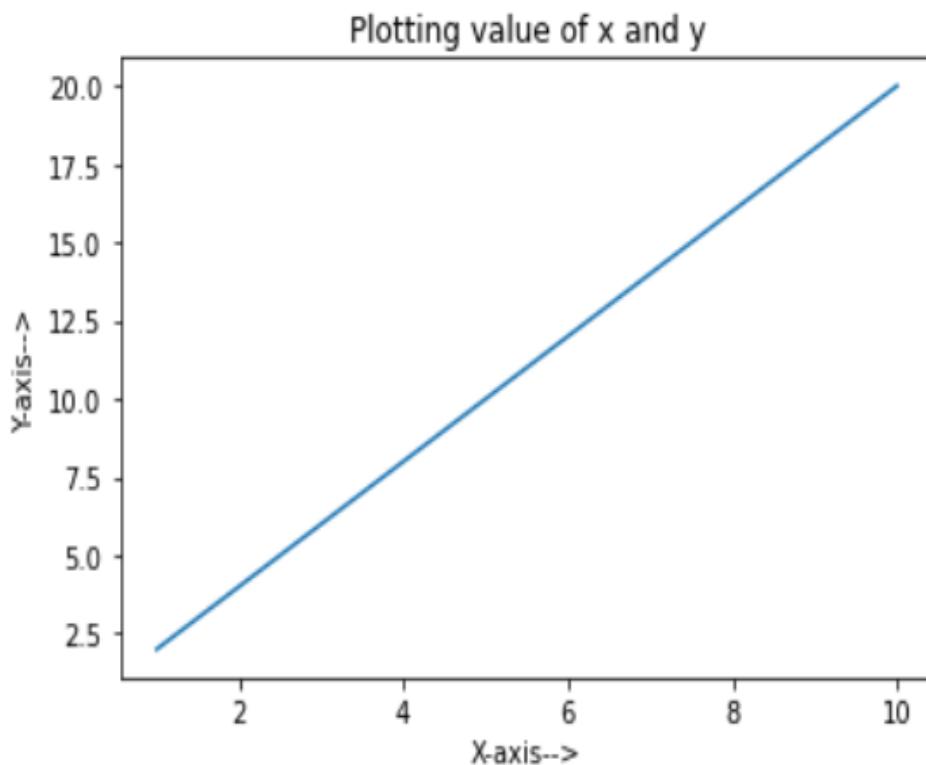
With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.

With Pyplot, you can use the title() function to set a title for the plot.

Task 27

```
▶ x=np.array((1,2,3,4,5,6,7,8,9,10))
y=x*2
print(x)
print(y)
plt.xlabel('X-axis-->')
plt.ylabel('Y-axis-->')
plt.title('Plotting value of x and y')
plt.plot(x,y)
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[ 2  4  6  8 10 12 14 16 18 20]
```



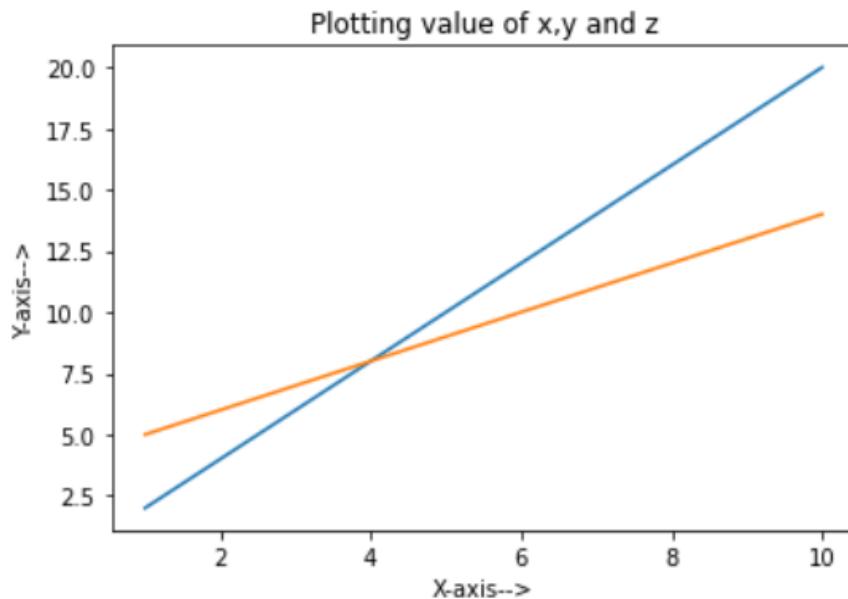
Task no. 4:

Plot using 3 variables on a single graph

Task 28

```
#applying Labels
x=np.array((1,2,3,4,5,6,7,8,9,10))
y=x*2
z=x+4
print(x)
print(y)
print(z)
plt.xlabel('X-axis-->')
plt.ylabel('Y-axis-->')
plt.title('Plotting value of x,y and z')
plt.plot(x,y)
plt.plot(x,z)
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[ 2  4  6  8 10 12 14 16 18 20]
[ 5  6  7  8  9 10 11 12 13 14]
```



Task no. 5:

Change the color linestyle and linewidth of the graph

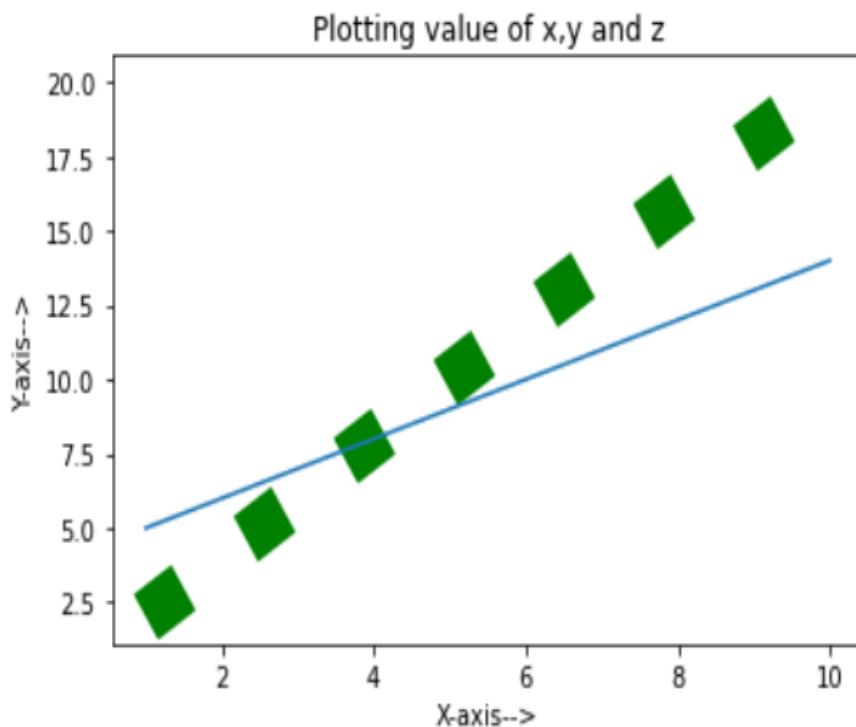
You can use the keyword argument linestyle, or shorter ls, to change the style of the plotted line

You can use the keyword argument linewidth or the shorter lw to change the width of the line

Task 29

```
▶ #applying styles to plot
x=np.array((1,2,3,4,5,6,7,8,9,10))
y=x*2
z=x+4
print(x)
print(y)
print(z)
plt.xlabel('X-axis-->')
plt.ylabel('Y-axis-->')
plt.title('Plotting value of x,y and z')
plt.plot(x,y,color='g',linestyle=":",linewidth=20)
plt.plot(x,z)
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[ 2  4  6  8 10 12 14 16 18 20]
[ 5  6  7  8  9 10 11 12 13 14]
```



Task no. 6:

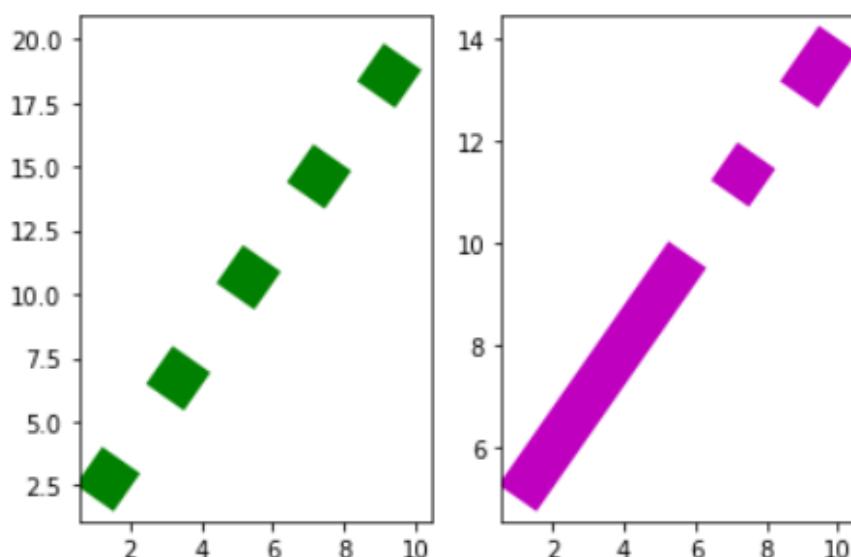
Plot using subplot

With the subplots() function you can draw multiple plots in one figure

Task 30

```
▶ #appying styles to plot
x=np.array((1,2,3,4,5,6,7,8,9,10))
y=x*2
z=x+4
print(x)
print(y)
print(z)
plt.xlabel('X-axis-->')
plt.ylabel('Y-axis-->')
plt.title('Plotting value of x,y and z')
plt.subplot(1,2,1)
plt.plot(x,y,color='g',linestyle=":",linewidth=20)
plt.subplot(1,2,2)
plt.plot(x,z,color='m',linestyle="-.",linewidth=20)
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[ 2  4  6  8 10 12 14 16 18 20]
[ 5  6  7  8  9 10 11 12 13 14]
```



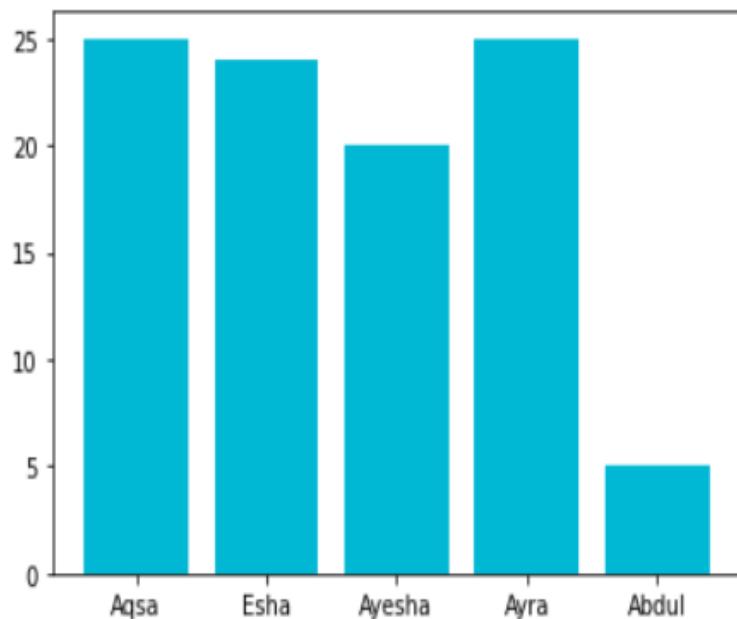
Task no. 7:

Print the marks of students w.r.t their names using Dictionary

Task 31

```
stu_mark={"Aqsa":25,"Esha":24,"Ayesha":20,"Ayra":25,"Abdul":5}
print(stu_mark)
k=stu_mark.keys()
v=stu_mark.values()
print(k)
print(v)
plt.bar(k,v,color="#00b8d4")
plt.show()
```

```
{'Aqsa': 25, 'Esha': 24, 'Ayesha': 20, 'Ayra': 25, 'Abdul': 5}
dict_keys(['Aqsa', 'Esha', 'Ayesha', 'Ayra', 'Abdul'])
dict_values([25, 24, 20, 25, 5])
```



Task no. 8:

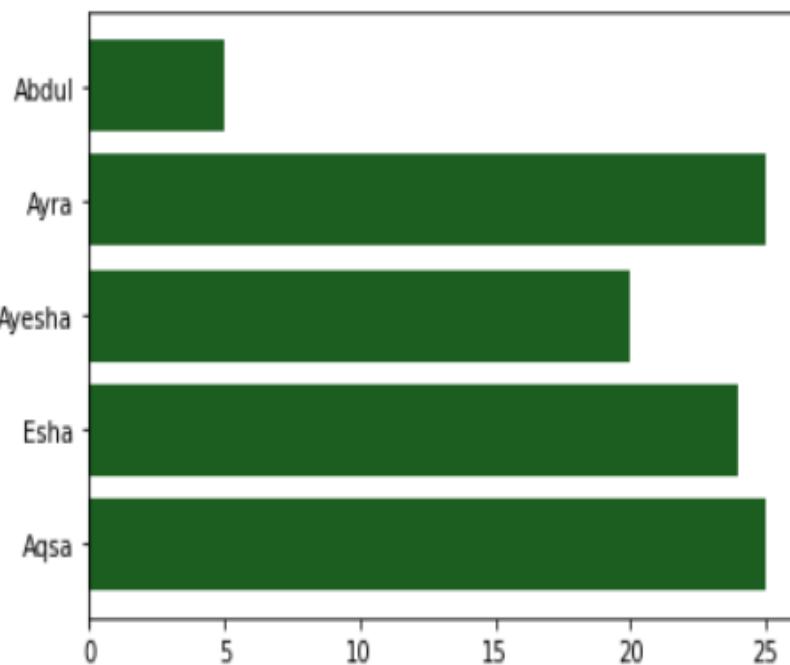
Plot a horizontal bar graph

If you want the bars to be displayed horizontally instead of vertically, use the barh() function

Task 32

```
#for horizontal plotting
stu_mark={"Aqsa":25,"Esha":24,"Ayesha":20,"Ayra":25,"Abdul":5}
print(stu_mark)
k=list(stu_mark.keys())
v=list(stu_mark.values())
print(k)
print(v)
plt.barh(k,v,color="#1b5e20")
plt.show()
```

```
{'Aqsa': 25, 'Esha': 24, 'Ayesha': 20, 'Ayra': 25, 'Abdul': 5}
['Aqsa', 'Esha', 'Ayesha', 'Ayra', 'Abdul']
[25, 24, 20, 25, 5]
```



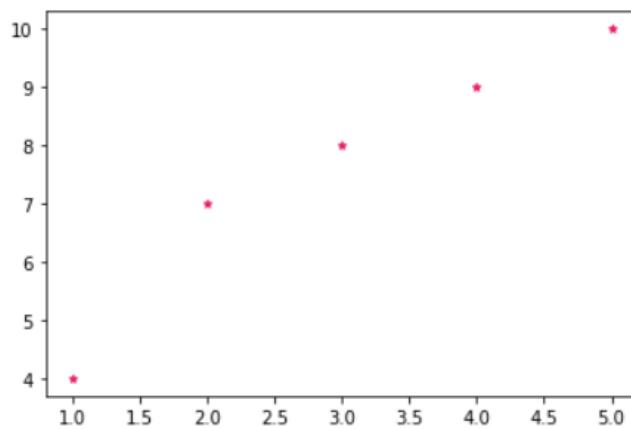
Task no. 9:

Plot using scatter function

The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis

Task 33

```
▶ a=[1,2,3,4,5]
  b=[4,7,8,9,10]
  plt.scatter(a,b,color="#e91e63",marker='*',s=20) #s=size of marker
  plt.show()
```



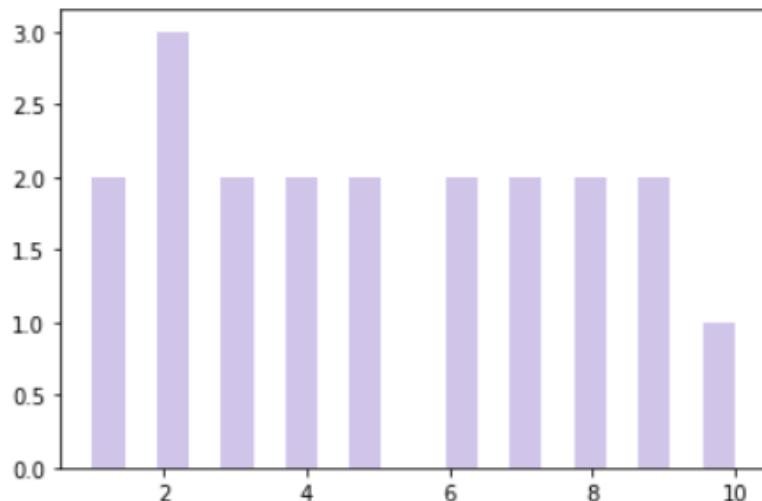
Task no. 10:

Plot a histogram

A histogram is a graph showing *frequency distributions*.

Task 34

```
▶ a=[1,2,3,4,5,6,7,8,9,10,9,8,7,6,5,4,3,2,1,2]
  plt.hist(a,bins=20,color='#d1c4e9')
  plt.show()
```



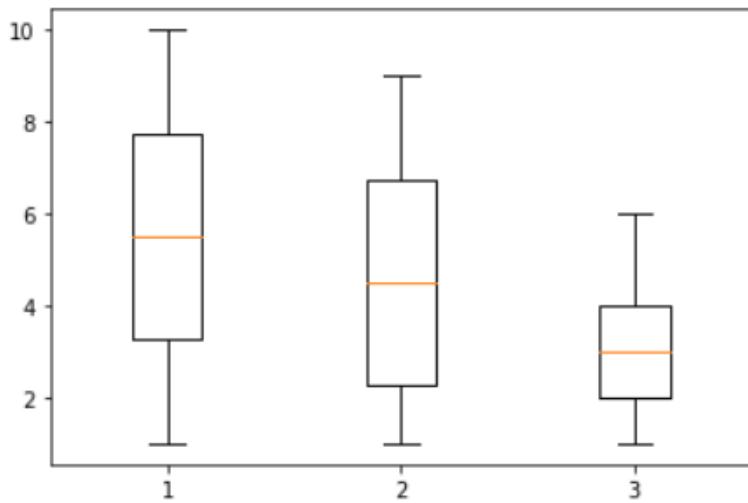
Task no. 11:

Demonstrate the use boxplot

A Box Plot is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum. In the box plot, a box is created from the first quartile to the third quartile, a vertical line is also there which goes through the box at the median. Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution.

Task 35

```
► l1=[1,2,3,4,5,6,7,8,9,10]
  l2=[3,4,5,6,7,1,2,8,9,1]
  l3=[1,2,3,4,1,2,3,4,5,6]
  data=list([l1,l2,l3])
  plt.boxplot(data)
  plt.show()
```



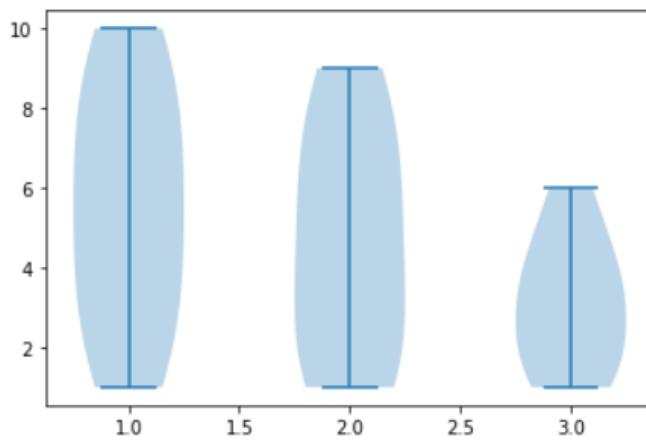
Task no. 12:

Demonstrate the use of violin plot

A violin plot allow to visualize the distribution of a numeric variable for one or several groups. Seaborn is particularly adapted to build it thanks to its violin() function. Violinplots deserve more attention compared to boxplots that can sometimes hide features of the data.

Task 36

```
► l1=[1,2,3,4,5,6,7,8,9,10]
  l2=[3,4,5,6,7,1,2,8,9,1]
  l3=[1,2,3,4,1,2,3,4,5,6]
  data=list([l1,l2,l3])
  plt.violinplot(data)
  plt.show()
```

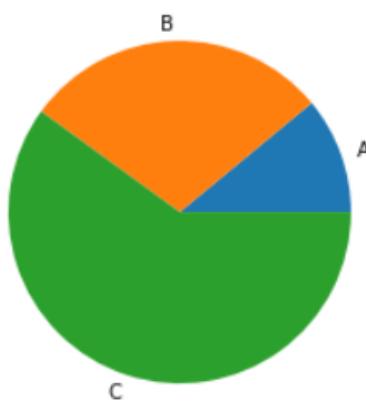


Task no. 13:

Demonstrate the use of pie function

Task 37

```
► label=["A","B","C"]
  values=[11,29,60]
  plt.pie(values,labels=label)
  plt.show()
```



LAB 07

(CLO 2-3, PLO 3-5,P3,P4)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	Linear Regression Model
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice and make linear regression Algorithm

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

*Linear
Regression
and logistic
regression*

Lab Tasks (Algorithms) Linear Regression

- :

```
#ML is the set of algorithms to train the model for specific objective
#classification
#binary:True and False only 2 classes
#Multiple:Grades
#ML Design process--> 7steps
#kaggle.com multiple datasets for ML
#iris dataset of flower
#amanest dataset hand wrritten digits
#caras, tensor flow ,open cv for image based
#kmeans ki short coming db scan na cover thi clustering algo
```
- :

```
#linear regression is a staistical model that can be used to find the relationship between predictors,featuresm i/p,o/p
#dependent and independent variable
#y=mx+c
#y=output/dependent variable
#x=input/independent variable
#m=slope
#c= constant /y intercept where line touches y axis
#slope + or -
#y=mx+c for +
#y=-mx+c for -
#positive slope
#x      on      y
#marketing depends on sales
#salary on experience
#distance on speed
#negative slope
#time on speed
#y=b0+b1x1+b2x2.....
#y=output
#b0=constant
#b1b2=slope
#x1x2=input
```

Task no . 1:

Explain Linear Regression with example (Lab work)

Answer:

➤ **Regression:**

Regression analysis is a form of predictive modelling technique, which investigates the relationship between dependent and independent variables.

Uses of regression:

Three major uses for regression analysis are

- Determining the strength of predictors.
- Forecasting an effect.
- Trend forecasting.

Linear Regression:

Linear Regression is a statistical method that can be used to find the relationship between predictor features and independent or dependent variables. It solves regression problem. The graph shows straight line.

Use of linear regression:

Linear regression is used for

- Evaluating Trends and Sales Estimates.
- Analyzing the impact of Price Changes.
- Assessment of risk in financial services and insurance domain.

Formula:

$$y = mx + c$$

Where as

y = Output / Dependent Variable.

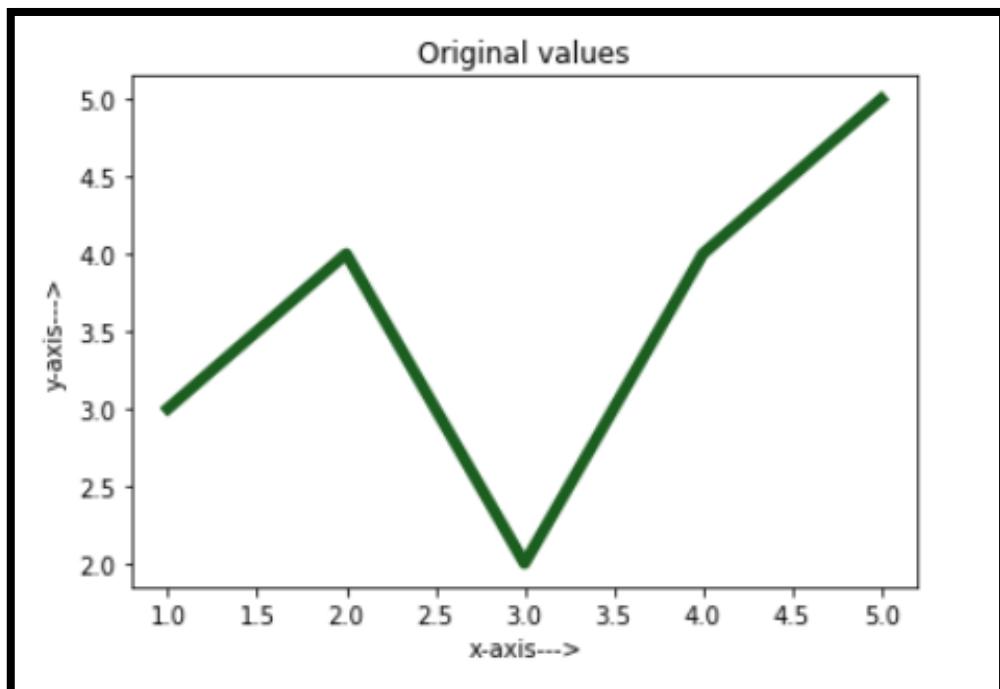
m = Slope.

X = Input / Independent Variable.

C = Constant / y-intercept.

For example:

<i>x</i>	<i>y (Actual Values)</i>
1	3
2	4
3	2
4	4
5	5



First we find mean of x and y values by using formula.

$$\text{Mean} = \frac{\text{Sum of all values}}{\text{Total Values.}}$$

Mean of x:

$$\sum \bar{x} = \frac{1 + 2 + 3 + 4 + 5}{5}$$

$$\sum \bar{x} = 3$$

Mean of y:

$$\sum \bar{y} = \frac{3 + 4 + 2 + 4 + 5}{5}$$

$$\sum \bar{y} = 3.6$$

Now, we use linear equation.

$$y = mx + c \quad \text{equation (A)}$$

$$\bar{y} = m\bar{x} + c \quad \text{equation (B)}$$

Now we find slope by using formula.

$$m = \frac{\sum(x - \bar{x})(y - \bar{y})}{(x - \bar{x})^2}$$

Put the values in formula.

$$m = \frac{(1-3)(3-3.6)+(2-3)(4-3.6)+(3-3)(2-3.6)+(4-3)(4-3.6)+(5-3)(5-3.6)}{(1-3)^2+(2-3)^2+(3-3)^2+(4-3)^2+(5-3)^2}$$

$$m = \frac{4}{10} = 0.4$$

Now, we find c. Put the values in equation (B)

$$3.6 = 0.4(3) + c$$

$$c = 3.6 - 1.2$$

$$c = 2.4$$

Prediction values of y:

We find prediction values of y by using values of m ,x, and c in equation (A).

y₁:

$$y_1 = 1(0.4) + 2.4 = 2.8$$

y₂:

$$y_2 = 2(0.4) + 2.4 = 0.8 + 2.4 = 3.2$$

y₃:

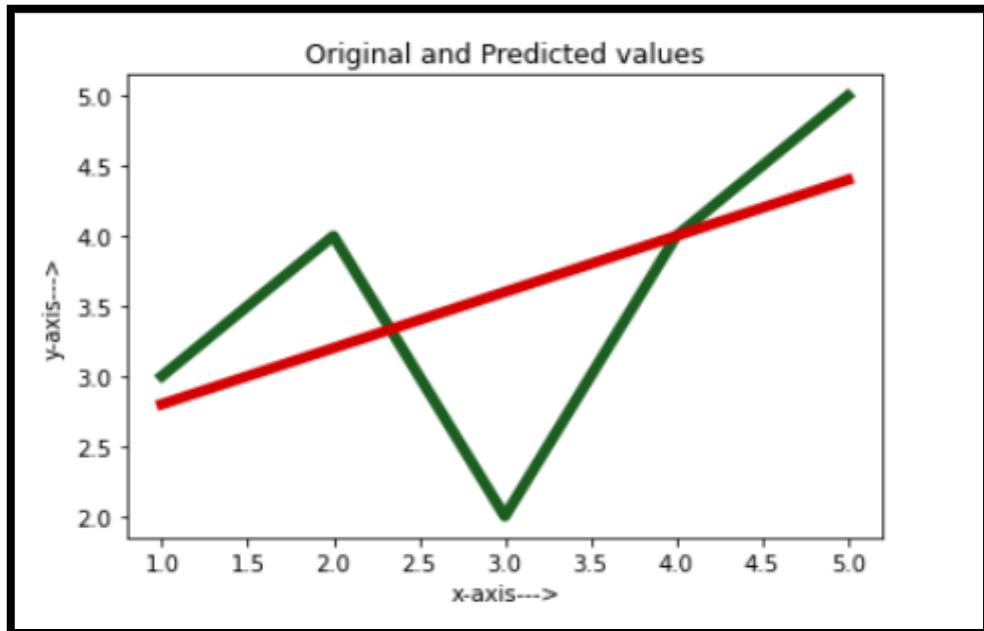
$$y_3 = 3(0.4) + 2.4 = 1.2 + 2.4 = 3.6$$

y₄:

$$y_4 = 4(0.4) + 2.4 = 1.6 + 2.4 = 4.0$$

y_5 :

$$y_5 = 5(0.4) + 2.4 = 2.0 + 2.4 = 4.4$$



R^2 :

$$R^2 = \frac{\sum(y_p - \bar{y})}{(\bar{y} - \bar{y})^2}$$

$$R^2 = \frac{(2.8 - 3.6)^2 + (3.2 - 3.6)^2 + 3.6 - 3.6)^2 + (4 - 3.6)^2 + (4.4 - 3.6)^2}{(3 - 3.6)^2 + (4 - 3.6)^2 + (2 - 3.6)^2 + (4 - 3.6)^2 + (5 - 3.6)^2}$$

$$R^2 = 0.307$$

$$R^2 = 0.307 \times 100\% = 30.7\%$$

R-squared : R-squared value is a statistical measure of how close the data are to the fitted the regression line. It is also known as coefficient of determination, or the coefficient of multiple determination.

If value of $R^2 = 1$ its mean no error occur 100 % line is fitted.

If value of $R^2 = 0$ its mean error occur 100 % line is not fitted.

Task no . 2:

Implement Linear Regression from scratch (Lab work)

Implementing Linear Regression from scratch

Creating 2 arrays using numpy

```
▶ #example solved on register
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,3,4,5]) #actual value of x
y=np.array([3,4,2,4,5]) # actual value of y
print(x,"\\nshape of x = ",x.shape)
print(y,"\\nshape of y = ",y.shape)
```

```
[1 2 3 4 5]
shape of x =  (5,)
[3 4 2 4 5]
shape of y =  (5,)
```

Finding mean

```
▶ #finding means
x_mean=np.mean(x)
y_mean=np.mean(y)
print("mean of x = ",x_mean)
print("mean of y = ",y_mean)
```

```
mean of x =  3.0
mean of y =  3.6
```

Finding slope

```
▶ #finding slope
m_num=0
m_denom=0
n=len(x)
print("length of loop = ",n)
for i in range(n):
    m_num+=(x[i]-x_mean)*(y[i]-y_mean)
    m_denom+=(x[i]-x_mean)**2
m=m_num/m_denom
print("slope = ",m)

length of loop =  5
slope =  0.4
```

Finding c i.e. y intercept

```
: ▶ c=y_mean-(m*x_mean)
    print("constant = ",c)

constant =  2.4
```

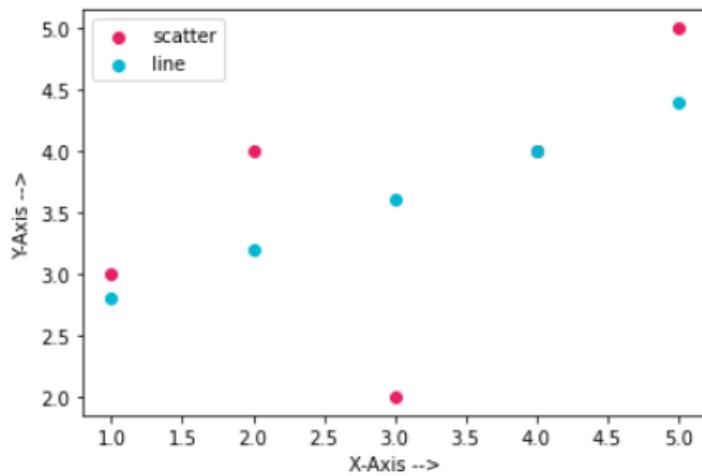
Predicting value of y

```
▶ #predicting value of y
y_predicted=m*x+c #passing complete list as x
print(y_predicted)

[2.8 3.2 3.6 4.  4.4]
```

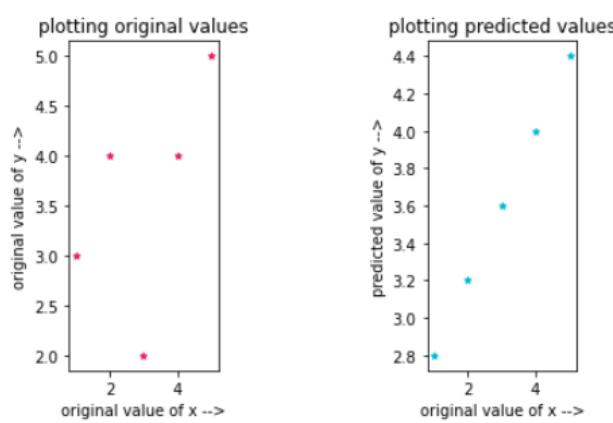
Plotting the original and predicted values

```
#Method done in class
plt.scatter(x,y,color="#e91e63",label='scatter')
plt.xlabel("X-Axis -->")
plt.ylabel("Y-Axis -->")
plt.scatter(x,y_predicted,color="#00b8d4",label='line')
plt.legend()
plt.show()
```



#By subploting

```
plt.subplot(1,3,1)
plt.scatter(x,y,color="#e91e63",label='scatter',marker='*',s=20) #s=size of marker
plt.xlabel("original value of x -->")
plt.ylabel("original value of y -->")
plt.title("plotting original values")
plt.subplot(1,3,3)
plt.scatter(x,y_predicted,color="#00b8d4",label='line',marker='*',s=20) #s=size of marker
plt.xlabel("original value of x -->")
plt.ylabel("predicted value of y -->")
plt.title("plotting predicted values")
plt.show()
```



Finding Rsquare

```
: ┌─▶ r_num=0
  r_denum=0
  for i in range(n):
    r_num+=(y_predicted[i]-y_mean)**2
    r_denum+=(y[i]-y_mean)**2
  r=r_num/r_denum
  print("R square = ",r)
```

R square = 0.3076923076923078

Calculating best fit and error percentage

```
└─▶ b=r*100
  e=100-b
  print("Predicted values are best fit upto ",b,"% which is approx = ",round(b),"%" )
  print("Predicted values have Error upto ",e,"% which is approx = ",round(e),"%" )
```

Predicted values are best fit upto 30.76923076923078 % which is approx = 31 %
Predicted values have Error upto 69.23076923076923 % which is approx = 69 %

Task no . 3:

Implement Linear Regression on **HeadBrain** dataset from scratch (Lab work)

Implement Linear Regression on HeadBrain dataset from scratch

```
: ┌─▶ import pandas as pd
  dt=pd.read_csv('headbrain.csv')
  print(dt)
```

	Gender	Age	Range	Head Size(cm^3)	Brain Weight(grams)
0	1	1	1	4512	1530
1	1	1	1	3738	1297
2	1	1	1	4261	1335
3	1	1	1	3777	1282
4	1	1	1	4177	1590
..
232	2	2	2	3214	1110
233	2	2	2	3394	1215
234	2	2	2	3233	1104
235	2	2	2	3352	1170
236	2	2	2	3391	1120

[237 rows x 4 columns]

Using Describe,shape and type function

```
▶ print("Describe = \n",dt.describe())
print("Describe = \n",dt.head())
print("Shape = ",dt.shape)

Describe =
   Gender  Age Range  Head Size(cm^3)  Brain Weight(grams)
count  237.000000  237.000000      237.000000      237.000000
mean    1.434599    1.535865    3633.991561    1282.873418
std     0.496753    0.499768    365.261422     120.340446
min     1.000000    1.000000    2720.000000     955.000000
25%    1.000000    1.000000    3389.000000    1207.000000
50%    1.000000    2.000000    3614.000000    1280.000000
75%    2.000000    2.000000    3876.000000    1350.000000
max     2.000000    2.000000    4747.000000    1635.000000

Describe =
   Gender  Age Range  Head Size(cm^3)  Brain Weight(grams)
0        1         1          4512           1530
1        1         1          3738           1297
2        1         1          4261           1335
3        1         1          3777           1282
4        1         1          4177           1590

Shape =  (237, 4)
```

Finding means

```
▶ x_mean=np.mean(x)
y_mean=np.mean(y)
print("mean of x = ",x_mean)
print("mean of y = ",y_mean)

mean of x =  3.0
mean of y =  3.6
```

Finding slope

```
▶
m_num=0
m_denum=0
n=len(x)
print("length of loop = ",n)
for i in range(n):
    m_num+=(x[i]-x_mean)*(y[i]-y_mean)
    m_denum+=(x[i]-x_mean)**2
m=m_num/m_denum
print("slope = ",m)

length of loop =  5
slope =  0.4
```

Finding Y intercept

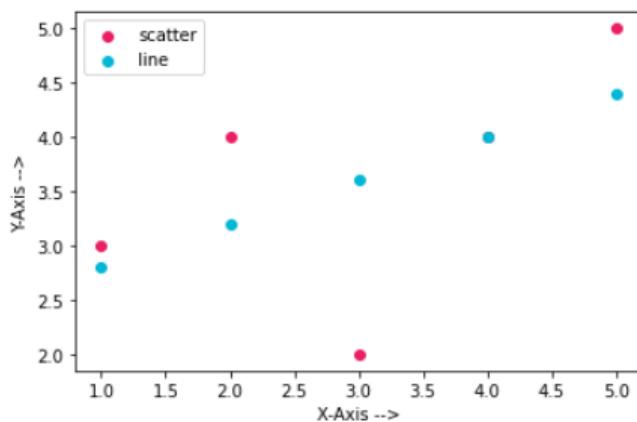
```
: ┌ #finding constant  
c=y_mean-(m*x_mean)  
print("constant = ",c)  
  
constant = 2.4
```

Predicting value of y

```
: ┌ #predicting value of y  
y_predicted=m*x+c #passing complete list as x  
print(y_predicted)  
  
[2.8 3.2 3.6 4. 4.4]
```

Plotting original and predicted values

```
: ┌ #Method done in class  
plt.scatter(x,y,color="#e91e63",label='scatter')  
plt.xlabel("X-Axis -->")  
plt.ylabel("Y-Axis -->")  
plt.scatter(x,y_predicted,color="#00b8d4",label='line')  
plt.legend()  
plt.show()
```



Finding rsquare

```
▶ r_num=0  
r_denum=0  
for i in range(n):  
    r_num+=(y_predicted[i]-y_mean)**2  
    r_denum+=(y[i]-y_mean)**2  
r=r_num/r_denum  
print("R square = ",r)
```

R square = 0.3076923076923078

Calculating best fit and error percentage

```
▶ b=n*100  
e=100-b  
print("Predicted values are best fit upto ",b,"% which is approx = ",round(b),"%")  
print("Predicted values have Error upto ",e,"% which is approx = ",round(e),"%")
```

Predicted values are best fit upto 30.76923076923078 % which is approx = 31 %
Predicted values have Error upto 69.23076923076923 % which is approx = 69 %

Task no . 4:

Implement Linear Regression using built-in Model (Lab work)

Implement Linear Regression using built-in Model

```
▶ #import numpy as np  
#from sklearn.linear_model import LinearRegression #library for machine learning algo  
#from sklearn.model_selection import train_test_split ,split data test and train  
#from sklearn.metrics import r2_score,mean_squared_error,accuracy_score,to find rsquare accuracy etc  
#from matplotlib import pyplot as plt
```

Importing libraries

```
▶ import numpy as np  
from sklearn.linear_model import LinearRegression  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import r2_score,mean_squared_error,accuracy_score  
from matplotlib import pyplot as plt
```

Creating 2 arrays using numpy

```
: ┌─▶ x=np.array([1,2,3,4,5]).reshape(-1,1) #actual value of x
  y=np.array([3,4,2,4,5]).reshape(-1,1) # actual value of y
  print(x,"\\nshape of x = ",x.shape)
  print(y,"\\nshape of y = ",y.shape)

[[1]
 [2]
 [3]
 [4]
 [5]]
shape of x =  (5, 1)
[[3]
 [4]
 [2]
 [4]
 [5]]
shape of y =  (5, 1)
```

Training

```
: ┌─▶ linear=LinearRegression()
  linear.fit(x,y)

[59]: LinearRegression()
```

Testing

```
: ┌─▶ test=np.array([2,28,54,36,8]).reshape(-1,1)

: ┌─▶ y_pred=linear.predict(test)
  print(y_pred)

[[ 3.2]
 [13.6]
 [24. ]
 [16.8]
 [ 5.6]]
```

Rsquare

```
[]: # print("score",linear.score(x,y))
      score 0.3076923076923078

[]: # print("R2 = ",r2_score(y,y_pred))
#gives error if no. of y is not equal to no. of y pred
      R2 = -141.38461538461544
```

LAB 08

(CLO 2-3, PLO 3-5,P3,P4)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	Logistic Regression
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice and make Logistic Regression Algorithm

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Explain and implement Logistic Regression on any dataset

Answer:

Logistic Regression:

Logistic Regression produces result in a binary format which is used to predict the outcome of a categorical dependent variable. It solves classification problems. The graph represent S-Curve.

So the outcome should be discrete / categorical such as:

- 0 or 1.
- Yes or No.
- True or False.
- High and Low.

Logistic regression will allow you to analyze the set of variables and predict a categorical outcome. Since here we need to predict whether she will get into school or not, which is a classification problem, logistic regression will be used.

Why are we not using Linear regression in this case?

The reason is that linear regression is used to predict a continuous quantity, rather than categorical one. Here we are going to predict whether or not your sister is going to get into grad school. So that is clearly a categorical outcome. So when the result in outcome can take only classes of values, like two classes of values, it is sensible to have a model that predicts the value as either 0 or 1, or in a probability form ha ranges between zero and one. So linear regression does not have this ability. If we use linear regression to model a binary outcome, the resulting

model will not predict y values in the range of 0 and 1, because linear regression works on continuous dependent variables, and not on categorical variables. That's why we make use of logistic regression. So we understand that linear regression is used to predict continuous quantities, and logistic regression is used to predict categorical quantities .

Logistic regression is primary technique. It is similar to linear regression because it belongs to generate linear models. It belongs to same class as linear regression , but there is no reason behind the name logistic regression. Logistic regression is mainly used for classification purpose because here we will have to predict a dependent variable which is categorical in nature .

Logistic regression equation is derived from same equation, except we need to make a few alterations, because the output is only categorical. So logistic regression does not calculate necessary the outcome as zero or one.

Logistic Regression Equation:

The logistic regression equation is derived from the Straight line Equation.

Equation of **straight line :**

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \varepsilon$. Range is from $-\infty$ to ∞ .

whereas

$\beta_0 = c$ = Constant or $y_{\text{intercept}}$.

$\beta_1 = m$ = Slope.

x = Input / Independent Variable.

y = Output / Dependent Variable.

Let's try to reduce the Logistic Regression Equation from Straight Line Equation.

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots$$

In logistic regression y can be only from 0 to 1.

Now , to get the range of y between 0 and infinity , let's transform y .

$$\frac{y}{y-1}$$

where as $y = 0$ then and $y = 1$ then infinity.

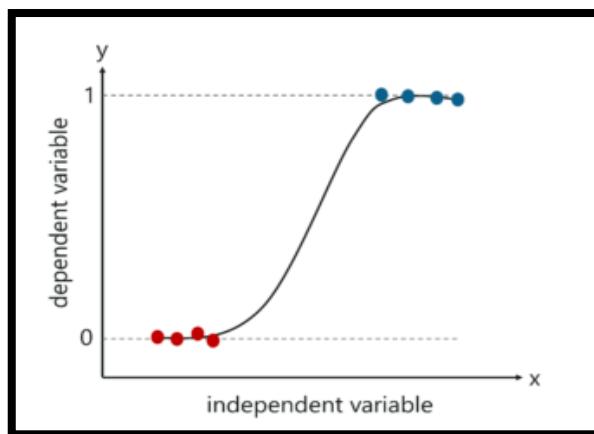
Now, the range is between 0 to infinity.

Let us transform it further to get range between $-\infty$ and ∞ .

$$\log\left[\frac{y}{1-y}\right] \Rightarrow y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots \text{ (Final logistic regression).}$$

How logistic regression work:

Take a look at this graph.



Now I told you that the outcome in a logistic regression is categorical. Your outcome will either be 0 or 1, or it will be a probability from 0 to 1. Now, some of you might think that why do we have an S curve. We can obviously have a straight line. We have something known as a Sigmoid curve, because we have values ranging between zero and one, which will basically show the probability. So maybe our output will be 0.7, which is a probability value. if it is 0.7, it means that your outcome is basically one. So that's why we have this sigmoid curve like this.

First we take a linear regression equation.

$$y = \beta_0 + \beta_1 x + \epsilon$$

Represent a relationship between $p(X) = \Pr(Y=1|X)$ and X . here because this \Pr denotes Probability and this value basically denotes that the probability of $y = 1$, given some value of x , If you wanted to calculate probability using the linear regression model, then the probability will look something like

$$P(X) = \frac{e^{(\beta_0 + \beta_1 x)}}{e^{(\beta_0 + \beta_1 x)} + 1}$$

Now the next step is to calculate something known as a logic function. Now, the logic function is nothing, but it is a link function that is represented as an S curve or as a sigmoid curve that ranges between the value 0 and 1 it basically calculates the probability of the output variable.

$$P(X) = \frac{e^{(\beta_0 + \beta_1 x)}}{e^{(\beta_0 + \beta_1 x)} + 1}$$

$$p(e^{(\beta_0 + \beta_1 x)} + 1) = e^{(\beta_0 + \beta_1 x)}$$

$$p \cdot e^{(\beta_0 + \beta_1 x)} + p = e^{(\beta_0 + \beta_1 x)}$$

$$p = e^{(\beta_0 + \beta_1 x)} - p \cdot e^{(\beta_0 + \beta_1 x)}$$

$$p = e^{(\beta_0 + \beta_1 x)}(1 - p)$$

$$\frac{p}{(1 - p)} = e^{(\beta_0 + \beta_1 x)}$$

$$\ln \left[\frac{p}{(1 - p)} \right] = \beta_0 + \beta_1 x$$

Uses of logistic regression:

Logistic regression is used across many scientific fields. In Natural Language Processing (NLP), it's used to determine the sentiment of movie reviews, while in Medicine it can be used to determine the probability of a patient developing a particular disease.

Implement Logistic Regression:

Following are these.

- **Collect Data:** Importing Libraries.
- **Analyzing Data:** Creating different plots to check relationships between them.
- **Data Wrangling:** Clean the data by removing Nan values and unnecessary columns in data set.
- **Train and Test Data:** Build the model on train data and predict the output on test data.
- **Accuracy Check:** Calculate accuracy to check how accurate your values are.

From Scratch:

Implementing Logistic Regression from Scratch

Collect Data

```
# import pandas as pd
# import numpy as np
# import seaborn as sns
# import matplotlib.pyplot as plt
# import math
titanic_data=pd.read_csv("titanic.csv")
titanic_data.head(10)
```

04]:

	Passengerid	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0.0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0.0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0.0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1.0	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1.0	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

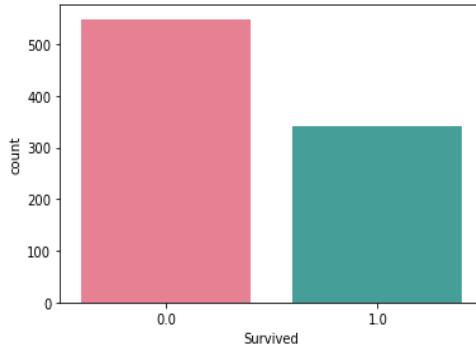
```
# print("Number of passengers in original data:" +str(len(titanic_data.index)))
```

Number of passengers in original data:1309

Analyzing Data

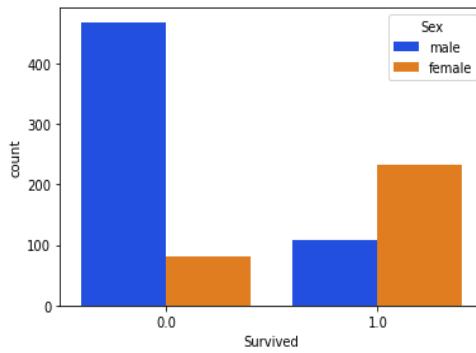
```
: sns.countplot(x="Survived",data=titanic_data,palette="husl")
```

```
[06]: <AxesSubplot:xlabel='Survived', ylabel='count'>
```



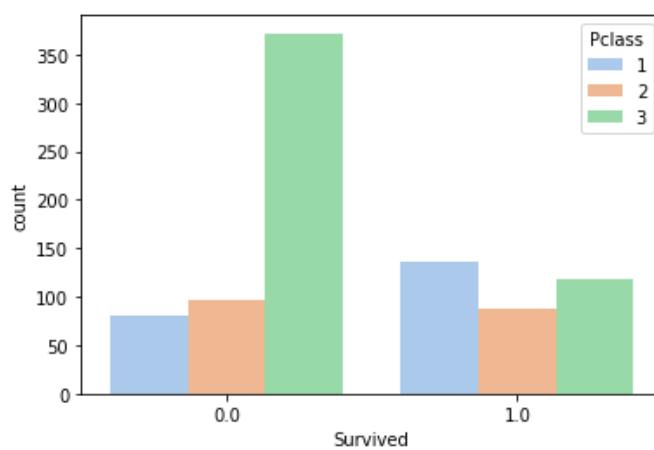
```
: sns.countplot(x="Survived",hue="Sex",data=titanic_data,palette="bright")
```

```
[07]: <AxesSubplot:xlabel='Survived', ylabel='count'>
```



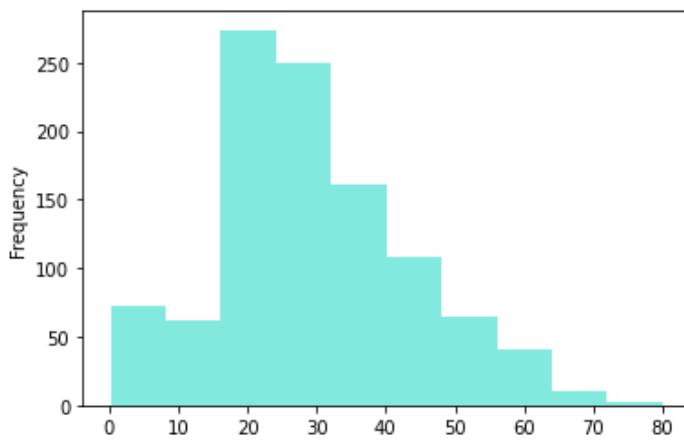
```
▶ sns.countplot(x="Survived",hue="Pclass",data=titanic_data,palette="pastel")
```

```
[08]: <AxesSubplot:xlabel='Survived', ylabel='count'>
```



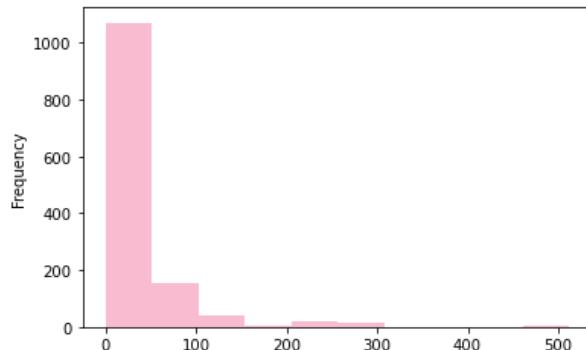
```
titanic_data["Age"].plot.hist(color="#82e9de")
```

```
[9]: <AxesSubplot:ylabel='Frequency'>
```



```
titanic_data["Fare"].plot.hist(color="#f8bbd0")
```

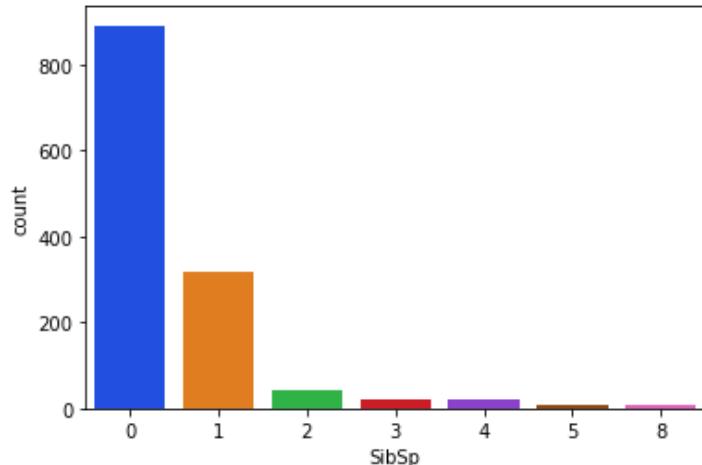
```
[0]: <AxesSubplot:ylabel='Frequency'>
```



```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
  0   PassengerId 1309 non-null   int64  
  1   Survived     891 non-null   float64 
  2   Pclass       1309 non-null   int64  
  3   Name         1309 non-null   object  
  4   Sex          1309 non-null   object  
  5   Age          1046 non-null   float64 
  6   SibSp       1309 non-null   int64  
  7   Parch       1309 non-null   int64  
  8   Ticket      1309 non-null   object  
  9   Fare         1308 non-null   float64 
  10  Cabin        295 non-null   object  
  11  Embarked    1307 non-null   object  
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

```
▶ sns.countplot(x="SibSp",data=titanic_data,palette="bright")  
2]: <AxesSubplot:xlabel='SibSp', ylabel='count'>
```



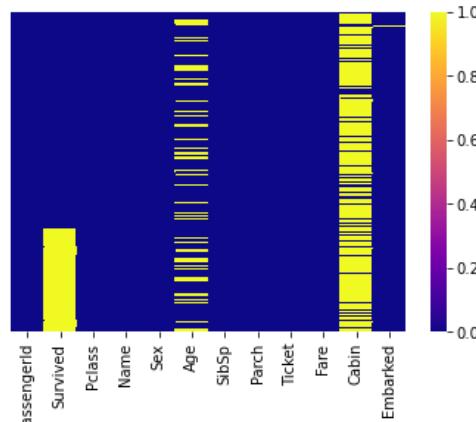
Data Wrangling

```
▶ titanic_data.isnull()  
3]:  
   PassengerId  Survived  Pclass  Name  Sex  Age  SibSp  Parch  Ticket  Fare  Cabin  Embarked  
0        False    False  False  False  False  False  False  False  False  False  False  True  False  
1        False    False  False  False  False  False  False  False  False  False  False  False  False  
2        False    False  False  False  False  False  False  False  False  False  False  False  True  False  
3        False    False  False  False  False  False  False  False  False  False  False  False  False  False  
4        False    False  False  False  False  False  False  False  False  False  False  False  True  False  
...      ...    ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  
1304     False     True  False  False  False  True  False  False  False  False  False  True  False  
1305     False     True  False  
1306     False     True  False  True  False  
1307     False     True  False  False  False  True  False  False  False  False  False  True  False  
1308     False     True  False  False  False  True  False  False  False  False  False  True  False  
1309 rows × 12 columns
```

```
▶ titanic_data.isnull().sum()  
4]:  
PassengerId      0  
Survived       418  
Pclass          0  
Name            0  
Sex             0  
Age          263  
SibSp          0  
Parch          0  
Ticket          0  
Fare           1  
Cabin       1014  
Embarked        2  
dtype: int64
```

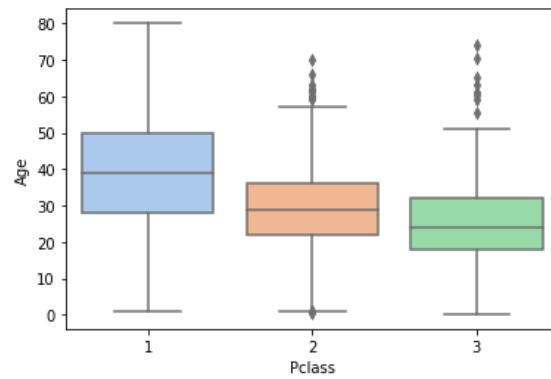
```
| sns.heatmap(titanic_data.isnull(), yticklabels=False, cmap = "plasma")
```

```
15]: <AxesSubplot:>
```



```
| sns.boxplot(x="Pclass",y="Age",data=titanic_data,palette="pastel")
```

```
16]: <AxesSubplot:xlabel='Pclass', ylabel='Age'>
```



```
| titanic_data.head(5)
```

```
17]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2 3101282	7.9250	NaN	S
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
| titanic_data.drop("Cabin",axis =1,inplace = True)
```

```
| titanic_data.head(5)
```

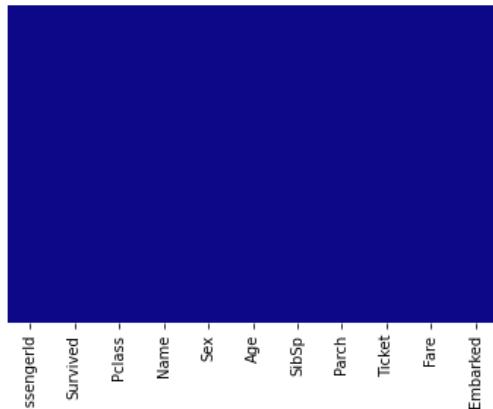
```
19]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2 3101282	7.9250	S
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S

```
| titanic_data.dropna(inplace=True)
```

```
▶ sns.heatmap(titanic_data.isnull(),yticklabels=False,cbar=False,cmap="plasma")
```

```
[1]: <AxesSubplot:>
```



```
▶ titanic_data.isnull().sum()
```

```
[6]: PassengerId      0
Survived        0
Pclass          0
Name            0
Sex             0
Age             0
SibSp          0
Parch          0
Ticket         0
Fare            0
Embarked       0
dtype: int64
```

```
▶ titanic_data.head(2)
```

```
[7]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C

```
▶ sex=pd.get_dummies(titanic_data["Sex"],drop_first=True)
```

```
▶ sex.head(5)
```

```
[69]:
```

```
male
```

0	1
1	0
2	0
3	0
4	1

```
▶ Pcl=pd.get_dummies(titanic_data["Pclass"],drop_first=True)
Pcl.head(5)
```

```
Pcl=pd.get_dummies(titanic_data["Pclass"],drop_first=True)
Pcl.head(5)
```

70]:

	2	3
0	0	1
1	0	0
2	0	1
3	0	0
4	0	1

```
embark=pd.get_dummies(titanic_data["Embarked"],drop_first=True)
embark.head(5)
```

71]:

	Q	S
0	0	1
1	0	0
2	0	1
3	0	1
4	0	1

```
titanic_data=pd.concat([titanic_data,sex,embark,Pcl],axis =1)
titanic_data.head(5)
```

2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	male	Q	S	2	3
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S	1	0	1	0	1
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599	71.2833	C	0	0	0	0	0
2	3	1.0	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	STON/O2. 3101282	7.9250	S	0	0	1	0	1
3	4	1.0	1	Allen, Mr. William Henry	male	35.0	1	0	113803	53.1000	S	0	0	1	0	0
4	5	0.0	3				0	0	373450	8.0500	S	1	0	1	0	1

```
titanic_data.drop(['Sex','Embarked','PassengerId','Name','Ticket'],axis = 1, inplace=True)
```

```
titanic_data.head()
```

4]:

	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S	2	3
0	0.0	3	22.0	1	0	7.2500	1	0	1	0	1
1	1.0	1	38.0	1	0	71.2833	0	0	0	0	0
2	1.0	3	26.0	0	0	7.9250	0	0	1	0	1
3	1.0	1	35.0	1	0	53.1000	0	0	1	0	0
4	0.0	3	35.0	0	0	8.0500	1	0	1	0	1

```
titanic_data.drop(["Pclass"],axis =1,inplace=True)
```

Train and Test Data

```
| X= titanic_data.drop( "Survived", axis=1)
| y= titanic_data["Survived"]
|
| from sklearn.model_selection import train_test_split
|
| X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
|
| from sklearn.linear_model import LogisticRegression
|
| logmodel=LogisticRegression()
|
| logmodel.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
predictions = logmodel.predict(X_test)
```

```
| from sklearn.metrics import classification_report
|
| classification_report(y_test,predictions)
|:
|      precision    recall   f1-score   support
| 0       0.72      0.72      0.72      88
| 1       0.76      0.76      0.76     214
|   weighted avg       0.77      0.77      0.77     214
|   macro avg       0.76      0.76      0.76     126
|   accuracy         0.76
|
| X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
|
| from sklearn.metrics import confusion_matrix
|
| confusion_matrix(y_test,predictions)
|:
| array([[102,  24],
|        [ 25,  63]], dtype=int64)
```

Accuray Check

```
] : | from sklearn.metrics import accuracy_score
] : | accuracy_score(y_test,predictions)
[189]: 0.7710280373831776
]
] : | ac = accuracy_score(y_test,predictions)
] : | ok = ac*100
] : | ok
[190]: 77.10280373831776
```

Using Built in Commands:

Implement Logistic Regression on any dataset

```
► pip install pydataset
```

```
Requirement already satisfied: pydataset in c:\users\dell\anaconda3\lib\site-packages (0.2.0)
Requirement already satisfied: pandas in c:\users\dell\anaconda3\lib\site-packages (from pydataset) (1.2.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\dell\anaconda3\lib\site-packages (from pandas->pydataset) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\dell\anaconda3\lib\site-packages (from pandas->pydataset) (2021.1)
Requirement already satisfied: numpy>=1.16.5 in c:\users\dell\anaconda3\lib\site-packages (from pandas->pydataset) (1.20.1)
Requirement already satisfied: six>=1.5 in c:\users\dell\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas->pydataset) (1.15.0)
```

Import Libraries

```
► #import libraries
  from pydataset import data
  import pandas as pd
  import numpy as np
  from sklearn.linear_model import LogisticRegression
  from sklearn.model_selection import train_test_split
```

Fetching Data

```
: ► #get data
  titanic=data('titanic')
  titanic.sample(5)
```

```
[3]:
```

	class	age	sex	survived
1055	3rd class	adults	man	no
1184	3rd class	adults	women	no
42	1st class	adults	man	yes
1217	3rd class	adults	women	no
1211	3rd class	adults	women	no

Feature engineering(oneHot)

```
▶ #feature engineering(oneHot)
titanic=pd.get_dummies(titanic,drop_first=True)
titanic.head()
```

[4]:

	class_2nd class	class_3rd class	age_child	sex_women	survived_yes
1	0	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	1
5	0	0	0	0	1

Test Train Split

```
▶ #test train split
X_train,X_test,y_train,y_test=train_test_split(titanic.drop('survived_yes',axis=1),titanic['survived_yes'])
```

Train the Model

```
▶ #train the model
LogReg=LogisticRegression(solver='lbfgs')
LogReg.fit(X_train,y_train)
```

[6]: LogisticRegression()

Predictions

```
: └─ #predicting if a class 1 child-age girl survived  
LogReg.predict(np.array([[0,0,1,1]]))[0]
```

```
t[9]: 1
```

```
: └─ #predicting if a class 3 adultage male survived  
LogReg.predict(np.array([[0,1,0,0]]))[0]
```

```
[10]: 0
```

Scoring the Model

```
└─ #scoring the model  
LogReg.score(X_test,y_test)
```

```
11]: 0.7659574468085106
```

Understanding the score

```
└─ #understanding the score  
prediction=(LogReg.predict(X_test)>.5).astype(int)  
np.sum(prediction == y_test)/len(y_test)
```

```
12]: 0.7659574468085106
```

LAB 09

(CLO 2-3, PLO 3-5,P3-P4)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	Support Vector Machine
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice and make support vector machines and kernel functions on the different dataset

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Support vector machine

Support Vector Machines (SVM) Algorithm

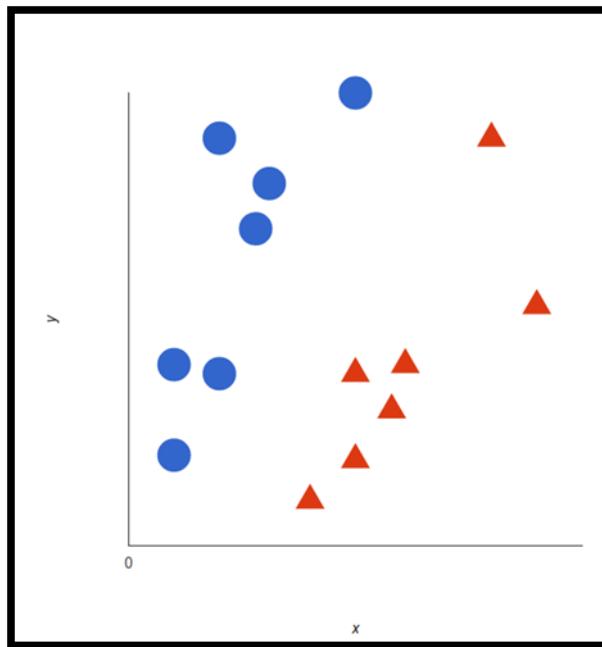
What is Support Vector Machines?

A support vector machine (SVM) is a supervised [machine learning](#) model that uses [classification algorithms](#) for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

Compared to newer algorithms like neural networks, they have two main advantages: higher speed and better performance with a limited number of samples (in the thousands). This makes the algorithm very suitable for text classification problems, where it's common to have access to a dataset of at most a couple of thousands of tagged samples.

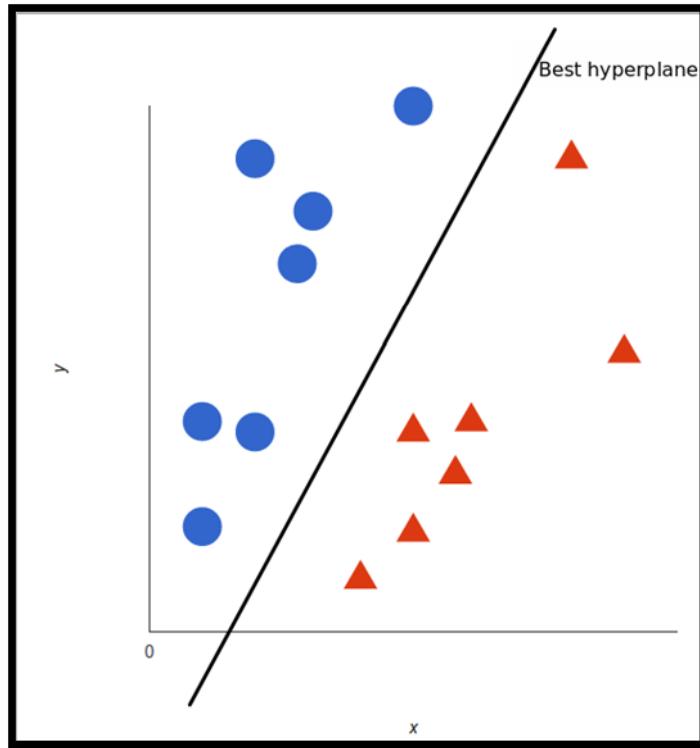
How Does SVM Work?

The basics of Support Vector Machines and how it works are best understood with a simple example. Let's imagine we have two tags: *red* and *blue*, and our data has two [features](#): x and y . We want a classifier that, given a pair of (x,y) coordinates, outputs if it's either *red* or *blue*. We plot our already labeled training data on a plane:



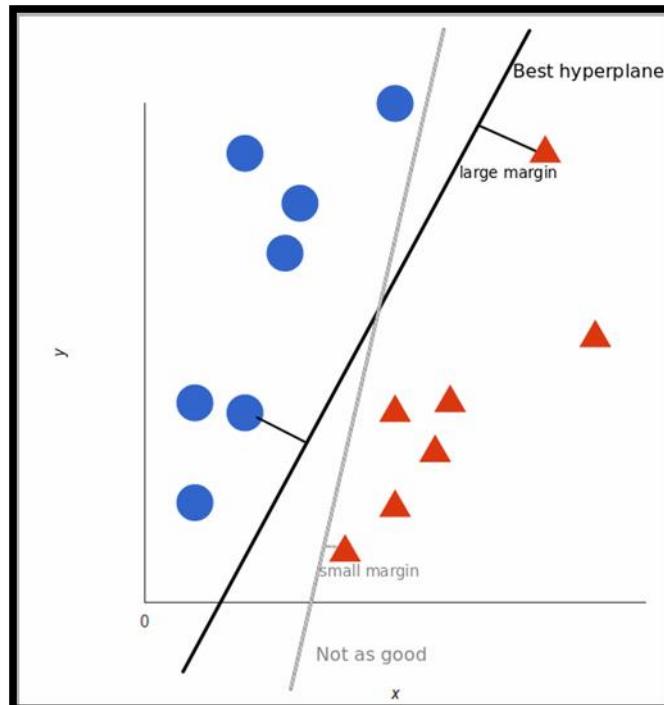
Our labeled data

A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the **decision boundary**: anything that falls to one side of it we will classify as *blue*, and anything that falls to the other as *red*.



In 2D, the best hyperplane is simply a line

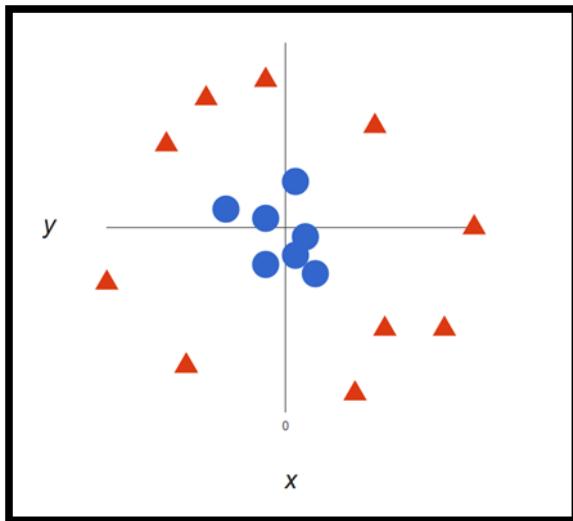
But, what exactly is *the best* hyperplane? For SVM, it's the one that maximizes the margins from both tags. In other words: the hyperplane (remember it's a line in this case) whose distance to the nearest element of each tag is the largest.



Not all hyperplanes are created equal

Nonlinear data:

Now this example was easy, since clearly the data was linearly separable — we could draw a straight line to separate *red* and *blue*. Sadly, usually things aren't that simple. Take a look at this case:

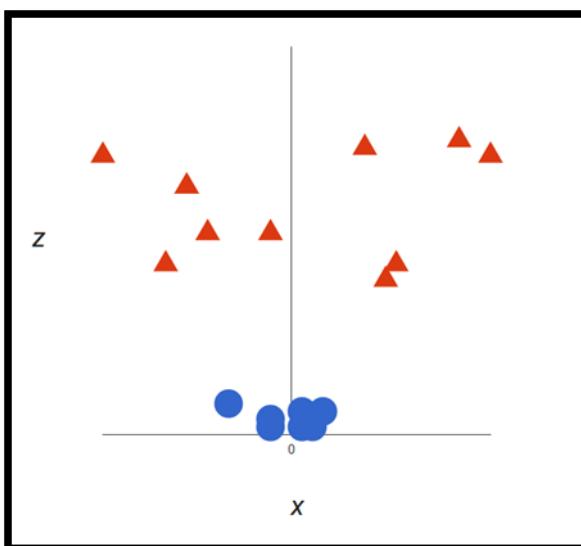


A more complex dataset

It's pretty clear that there's not a linear decision boundary (a single straight line that separates both tags). However, the vectors are very clearly segregated and it looks as though it should be easy to separate them.

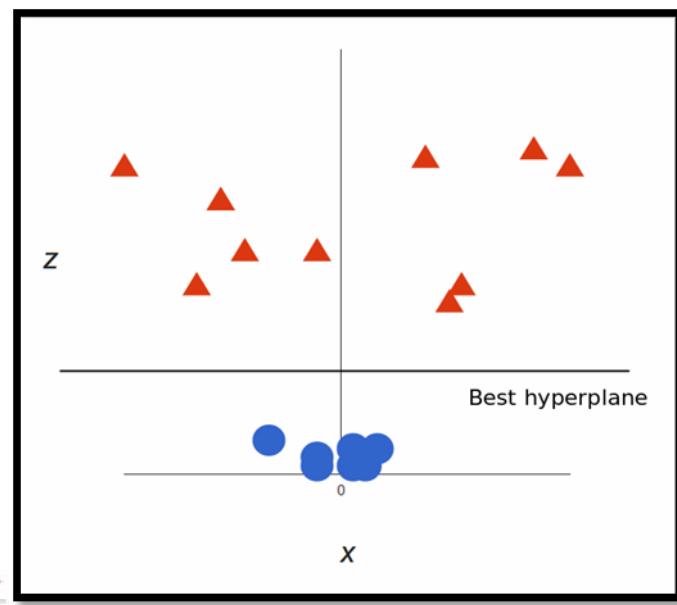
So here's what we'll do: we will add a third dimension. Up until now we had two dimensions: x and y . We create a new z dimension, and we rule that it be calculated a certain way that is convenient for us: $z = x^2 + y^2$ (you'll notice that's the equation for a circle).

This will give us a three-dimensional space. Taking a slice of that space, it looks like this:



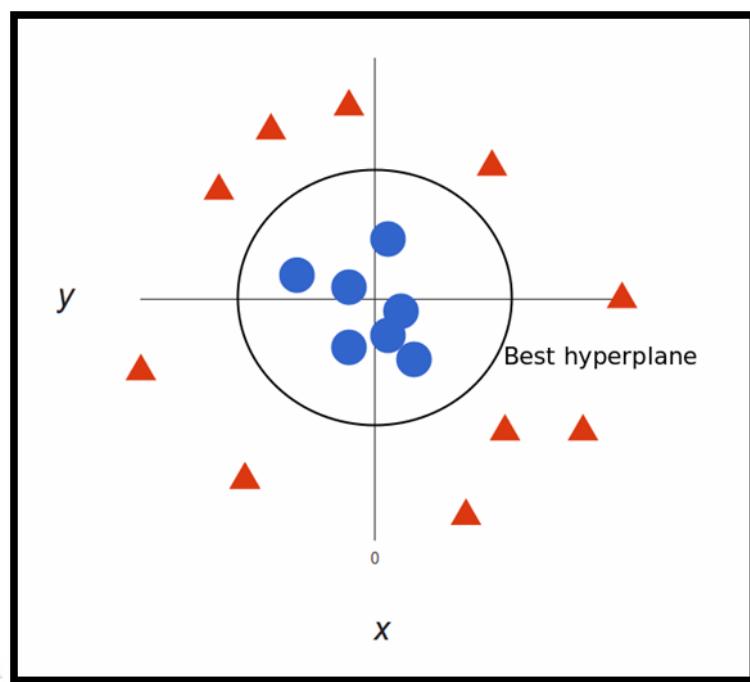
From a different perspective, the data is now in two linearly separated groups

What can SVM do with this? Let's see:



That's great! Note that since we are in three dimensions now, the hyperplane is a plane parallel to the x axis at a certain z (let's say $z = 1$).

What's left is mapping it back to two dimensions:



Back to our original view, everything is now neatly separated

And there we go! Our decision boundary is a circumference of radius 1, which separates both tags using SVM. Check out this 3d visualization to see another example of the same effect:

The kernel trick:

In our example we found a way to classify nonlinear data by cleverly mapping our space to a higher dimension. However, it turns out that calculating this transformation can get pretty computationally expensive: there can be a lot of new dimensions, each one of them possibly involving a complicated calculation. Doing this for every vector in the dataset can be a lot of work, so it'd be great if we could find a cheaper solution.

And we're in luck! Here's a trick: SVM doesn't need the actual vectors to work its magic, it actually can get by only with the [dot products](#) between them. This means that we can sidestep the expensive calculations of the new dimensions.

This is what we do instead:

- Imagine the new space we want:
$$z = x^2 + y^2$$
- Figure out what the dot product in that space looks like:
$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= xa \cdot xb + ya \cdot yb + za \cdot zb \\ \mathbf{a} \cdot \mathbf{b} &= xa \cdot xb + ya \cdot yb + (xa^2 + ya^2) \cdot (xb^2 + yb^2) \end{aligned}$$
- Tell SVM to do its thing, but using the new dot product — we call this a [kernel function](#).

That's it! That's the **kernel trick**, which allows us to sidestep a lot of expensive calculations. Normally, the kernel is linear, and we get a linear classifier. However, by using a nonlinear kernel (like above) we can get a nonlinear classifier without transforming the data at all: we only change the dot product to that of the space that we want and SVM will happily chug along.

Note that the kernel trick isn't actually part of SVM. It can be used with other linear classifiers such as logistic regression. A support vector machine only takes care of finding the decision boundary.

Using SVM with Natural Language Classification:

So, we can classify vectors in multidimensional space. Great! Now, we want to apply this algorithm for [text classification](#), and the first thing we need is a way to transform a piece of text into a vector of numbers so we can run SVM with them. In other words, which **features** do we have to use in order to classify texts using SVM?

The most common answer is word frequencies, [just like we did in Naive Bayes](#). This means that we treat a text as a bag of words, and for every word that appears in that bag we have a feature. The value of that feature will be how frequent that word is in the text.

This method boils down to just counting how many times every word appears in a text and dividing it by the total number of words. So in the sentence "*All monkeys are primates but not all primates are monkeys*" the word *monkeys* has a frequency of $2/10 = 0.2$, and the word *but* has a frequency of $1/10 = 0.1$.

For a more advanced alternative for calculating frequencies, we can also use [TF-IDF](#).

Now that we've done that, every text in our dataset is represented as a vector with thousands (or tens of thousands) of dimensions, every one representing the frequency of one of the words of the text. Perfect! This is what we feed to SVM for training. We can improve this by using [preprocessing techniques](#), like stemming, removing stopwords, and using n-grams.

Choosing a kernel function:

Now that we have the feature vectors, the only thing left to do is choosing a kernel function for our model. Every problem is different, and the kernel function depends on what the data looks like. In our example, our data was arranged in concentric circles, so we chose a kernel that matched those data points.

Considering those, what is best for [natural language processing](#)? Do we need a nonlinear classifier? Alternatively, is the data linearly separable? It turns out that it is best to stick to a linear kernel. Why?

Back in our example, we had two features. Some real uses of SVM in other fields may use tens or even hundreds of features. Meanwhile, NLP classifiers use *thousands* of features, since they can have up to one for every word that appears in the training data. This changes the problem a little bit: while using nonlinear kernels may be a good idea in other cases, having this many features will end up making nonlinear kernels over fit the data. Therefore, it is best to just stick to a good old linear kernel, which actually results in the best performance in these cases.

SVM Implementation in Python:

Support Vector Machines

```
► #supervised Mode  
#support both classification and regression mostly for classification  
#work with N-D data  
#multiclasses  
#for linear  
#line drawn called hyper plan  
#2 objects which ever is nearest in each class line is drawn from there edges  
#line drawn called support vectors  
#diff b/w both support vectors is called margin  
#line to be choosen shall have greatest margin it is best fit line ,if less margin underfitting  
#for non linear  
#low dimension to high dimension chnage position  
# kernal function is an equation by default it is linear other include sigmoid,RBF,ploynomial  
# when multiclassess build one to many relation
```

Import Libraries

```
► import pandas as pd  
  
► import matplotlib.pyplot as plt  
from sklearn.svm import SVC #support vector classifier  
from sklearn.model_selection import train_test_split
```

Read Dataset

```
► df=pd.read_csv('Iris.csv')  
df.head()
```

5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Seperating Features and Labels

```
: ➜ x=df.iloc[:,0:4]  
x.head()
```

t[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4

```
: ➜ y=df.iloc[:, -1]  
y.head()
```

t[7]: 0 Iris-setosa
1 Iris-setosa
2 Iris-setosa
3 Iris-setosa
4 Iris-setosa
Name: Species, dtype: object

Splitting the data

```
▶ xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

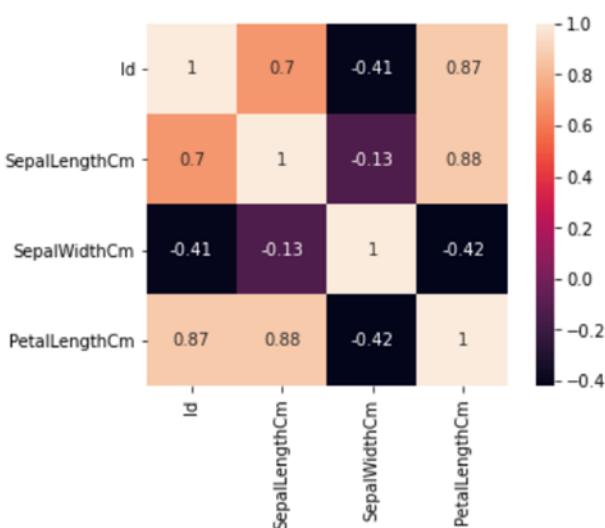
▶ print("Xtrain = ",xtrain.shape)
print("Xtest = " ,xtest.shape)
print("Ytrain = " ,ytrain.shape)
print("Ytest = " ,ytest.shape)

Xtrain =  (120, 4)
Xtest =  (30, 4)
Ytrain =  (120,)
Ytest =  (30,)
```

finding correlations between xtrain variables i.e. columns

```
▶ #finding correlations b/w xtrain variables i.e. columns
import seaborn as sns
corrmat=xtrain.corr()
sns.heatmap(corrmat,annot=True,square=True)

0]: <AxesSubplot:
```



Support Vector Classification

```
▶ model=SVC()  
#mode=SVC(kernel='linear')  
model.fit(xtrain,ytrain)
```

1]: SVC()

Predictions

```
▶ ypred=model.predict(xtest)  
print("Predictions = \n",ypred)  
  
Predictions =  
['Iris-versicolor' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'  
'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'  
'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'  
'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'  
'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'  
'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'  
'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'  
'Iris-versicolor' 'Iris-setosa']
```

Finding Accuracy Score

```
▶ print('Accuracy score = ',model.score(xtrain,ytrain))
```

Accuracy score = 0.9833333333333333

```
▶ from sklearn.metrics import accuracy_score  
print("Accuracy = \n",accuracy_score(ytest,ypred))
```

Accuracy =
0.9666666666666667

Kernel=linear

Support Vector Machines

```
▶ #supervised Mode  
#support both classification and regression mostly for classification  
#work with N-D data  
#multiclasses  
#for linear  
#line drawn called hyper plan  
#2 objects which ever is nearest in each class line is drawn from there edges  
#line drawn called support vectors  
#diff b/w both support vectors is called margin  
#line to be chosen shall have greatest margin it is best fit line ,if less margin underfitting  
#for non linear  
#low dimension to high dimension change position  
# kernel function is an equation by default it is linear other include sigmoid,RBF,polynomial  
# when multiclass build one to many relation
```

Import Libraries

```
▶ import pandas as pd  
  
▶ import matplotlib.pyplot as plt  
from sklearn.svm import SVC #support vector classifier  
from sklearn.model_selection import train_test_split
```

Read Dataset

```
▶ df=pd.read_csv('Iris.csv')  
df.head()
```

5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Separating Features and Labels

```
: ➜ x=df.iloc[:,0:4]  
x.head()
```

t[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4

```
: ➜ y=df.iloc[:, -1]  
y.head()
```

t[7]: 0 Iris-setosa

1 Iris-setosa

2 Iris-setosa

3 Iris-setosa

4 Iris-setosa

Name: Species, dtype: object

Splitting the data

```
➜ xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
```

```
➜ print("Xtrain = ",xtrain.shape)  
print("Xtest = " ,xtest.shape)  
print("Ytrain = " ,ytrain.shape)  
print("Ytest = " ,ytest.shape)
```

Xtrain = (120, 4)

Xtest = (30, 4)

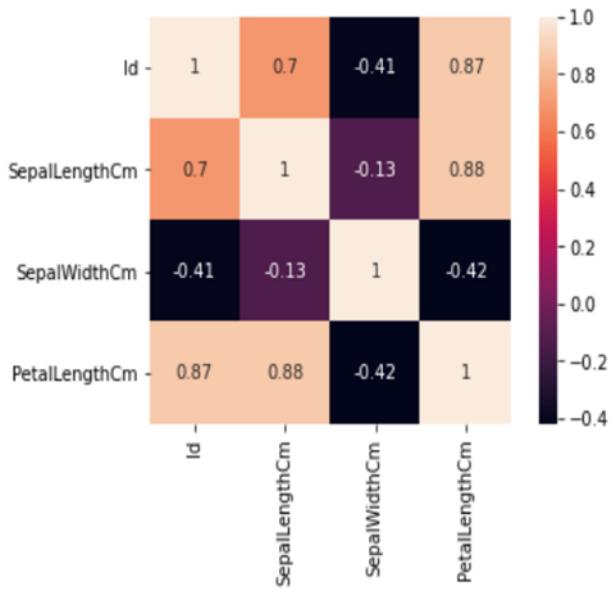
Ytrain = (120,)

Ytest = (30,)

finding correlations between xtrain variables i.e. columns

```
▶ #finding correlations b/w xtrain variables i.e. columns
import seaborn as sns
corrmat=xtrain.corr()
sns.heatmap(corrmat,annot=True,square=True)
```

0]: <AxesSubplot:>



Support Vector Classification

```
▶ model=SVC(kernel='linear')
model.fit(xtrain,ytrain)
```

43]: SVC(kernel='linear')

Predictions

```
▶ ypred=model.predict(xtest)
print("Predictions = \n",ypred)

Predictions =
['Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
'Iris-versicolor' 'Iris-virginica']
```

Finding Accuracy Score

```
▶ print('Accuracy score = ',model.score(xtrain,ytrain))

Accuracy score =  1.0

▶ from sklearn.metrics import accuracy_score
print("Accuracy = \n",accuracy_score(ytest,ypred))

Accuracy =
1.0
```

Kernel=sigmoid

Import Libraries

```
▶ import pandas as pd

▶ import matplotlib.pyplot as plt
from sklearn.svm import SVC #support vector classifier
from sklearn.model_selection import train_test_split
```

Read Dataset

```
▶ df=pd.read_csv('Iris.csv')  
df.head()
```

5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Separating Features and Labels

```
: ▶ x=df.iloc[:,0:4]  
x.head()
```

t[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4

```
: ▶ y=df.iloc[:, -1]  
y.head()
```

t[7]: 0 Iris-setosa
1 Iris-setosa
2 Iris-setosa
3 Iris-setosa
4 Iris-setosa
Name: Species, dtype: object

Splitting the data

```
▶ xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

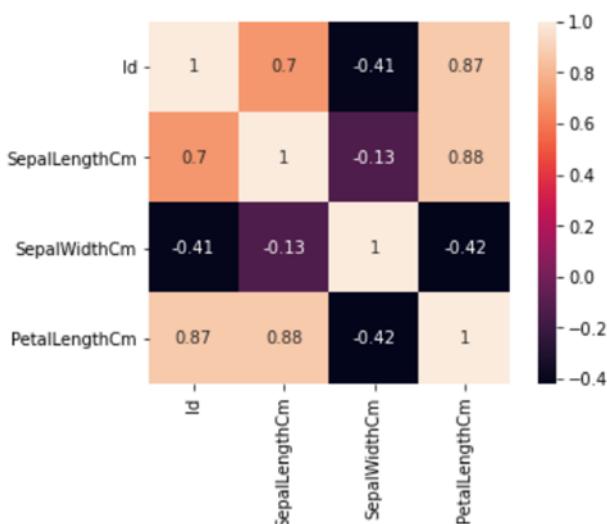
▶ print("Xtrain = ",xtrain.shape)
print("Xtest = " ,xtest.shape)
print("Ytrain = " ,ytrain.shape)
print("Ytest = " ,ytest.shape)

Xtrain =  (120, 4)
Xtest =  (30, 4)
Ytrain =  (120,)
Ytest =  (30,)
```

finding correlations between xtrain variables i.e. columns

```
▶ #finding correlations b/w xtrain variables i.e. columns
import seaborn as sns
corrrmat=xtrain.corr()
sns.heatmap(corrrmat,annot=True,square=True)
```

0]: <AxesSubplot:>



Support Vector Classification

```
▶ model=SVC(kernel='sigmoid')  
model.fit(xtrain,ytrain)
```

```
[7]: SVC(kernel='sigmoid')
```

Kernel=RBf

Import Libraries

```
▶ import pandas as pd  
  
▶ import matplotlib.pyplot as plt  
from sklearn.svm import SVC #support vector classifier  
from sklearn.model_selection import train_test_split
```

Read Dataset

```
▶ df=pd.read_csv('Iris.csv')  
df.head()
```

```
5]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Seperating Features and Labels

```
: ┌─▶ x=df.iloc[:,0:4]  
x.head()
```

t[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4

```
: ┌─▶ y=df.iloc[:, -1]  
y.head()
```

t[7]: 0 Iris-setosa
1 Iris-setosa
2 Iris-setosa
3 Iris-setosa
4 Iris-setosa
Name: Species, dtype: object

Splitting the data

```
▶ xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

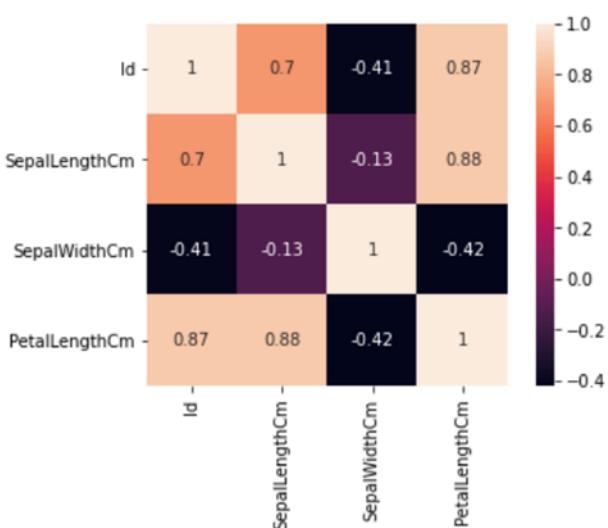
▶ print("Xtrain = ",xtrain.shape)
print("Xtest = " ,xtest.shape)
print("Ytrain = " ,ytrain.shape)
print("Ytest = " ,ytest.shape)

Xtrain =  (120, 4)
Xtest =  (30, 4)
Ytrain =  (120,)
Ytest =  (30,)
```

finding correlations between xtrain variables i.e. columns

```
▶ #finding correlations b/w xtrain variables i.e. columns
import seaborn as sns
corrmat=xtrain.corr()
sns.heatmap(corrmat,annot=True,square=True)

0]: <AxesSubplot:
```



Support Vector Classification

```
▶ model=SVC(kernel='rbf')
model.fit(xtrain,ytrain)

3]: SVC()
```

Predictions

```
▶ ypred=model.predict(xtest)
print("Predictions = \n",ypred)

Predictions =
['Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
'Iris-versicolor' 'Iris-virginica']
```

Finding Accuracy Score

```
: ▶ print('Accuracy score = ',model.score(xtrain,ytrain))

Accuracy score =  1.0

: ▶ from sklearn.metrics import accuracy_score
print("Accuracy = \n",accuracy_score(ytest,ypred))

Accuracy =
1.0
```

Kernel = Polynomial

Import Libraries

```
▶ import pandas as pd  
  
▶ import matplotlib.pyplot as plt  
from sklearn.svm import SVC #support vector classifier  
from sklearn.model_selection import train_test_split
```

Read Dataset

```
▶ df=pd.read_csv('Iris.csv')  
df.head()
```

5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Seperating Features and Labels

```
: ➜ x=df.iloc[:,0:4]  
x.head()
```

t[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4

```
: ➜ y=df.iloc[:, -1]  
y.head()
```

t[7]: 0 Iris-setosa
1 Iris-setosa
2 Iris-setosa
3 Iris-setosa
4 Iris-setosa
Name: Species, dtype: object

Splitting the data

```
▶ xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

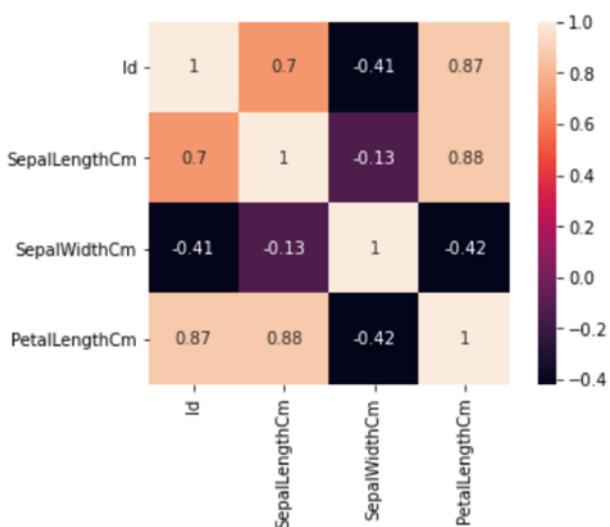
▶ print("Xtrain = ",xtrain.shape)
print("Xtest = " ,xtest.shape)
print("Ytrain = ",ytrain.shape)
print("Ytest = " ,ytest.shape)

Xtrain =  (120, 4)
Xtest =  (30, 4)
Ytrain =  (120,)
Ytest =  (30,)
```

finding correlations between xtrain variables i.e. columns

```
▶ #finding correlations b/w xtrain variables i.e. columns
import seaborn as sns
corrmat=xtrain.corr()
sns.heatmap(corrmat,annot=True,square=True)
```

0]: <AxesSubplot:>



Support Vector Classification

```
▶ model=SVC(kernel='poly')
model.fit(xtrain,ytrain)

33]: SVC(kernel='poly')
```

Predictions

```
▶ ypred=model.predict(xtest)
print("Predictions = \n",ypred)

Predictions =
['Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-versicolor' 'Iris-setosa']
```

Finding Accuracy Score

```
▶ print('Accuracy score = ',model.score(xtrain,ytrain))

Accuracy score =  0.975

▶ from sklearn.metrics import accuracy_score
print("Accuracy = \n",accuracy_score(ytest,ypred))

Accuracy =
0.9333333333333333
```

Finding Accuracy Score of all 4 kernel types

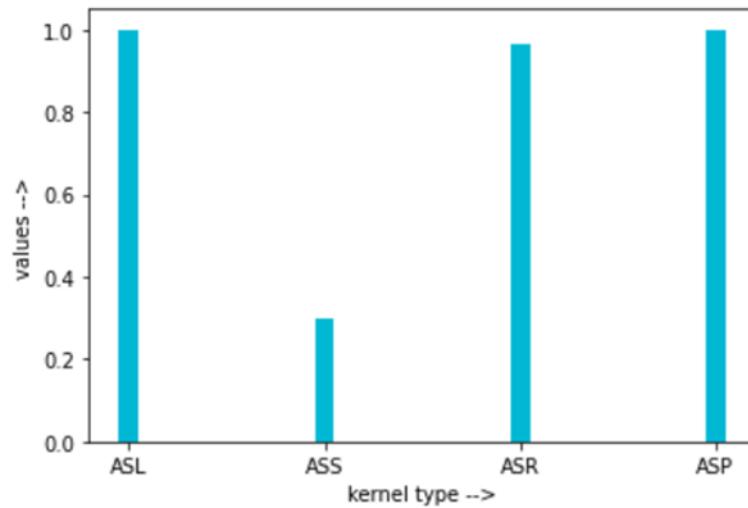
```
▶ s1=model_linear.score(xtrain,ytrain)
  s2=model_sigmoid.score(xtrain,ytrain)
  s3=model_rbf.score(xtrain,ytrain)
  s4=model_poly.score(xtrain,ytrain)
  print('Accuracy score (linear)= ',s1)
  print('Accuracy score (sigmoid)= ',s2)
  print('Accuracy score (rbf)= ',s3)
  print('Accuracy score (polynomial)= ',s4)
```

```
Accuracy score (linear)=  1.0
Accuracy score (sigmoid)=  0.21666666666666667
Accuracy score (rbf)=  0.9916666666666667
Accuracy score (polynomial)=  0.975
```

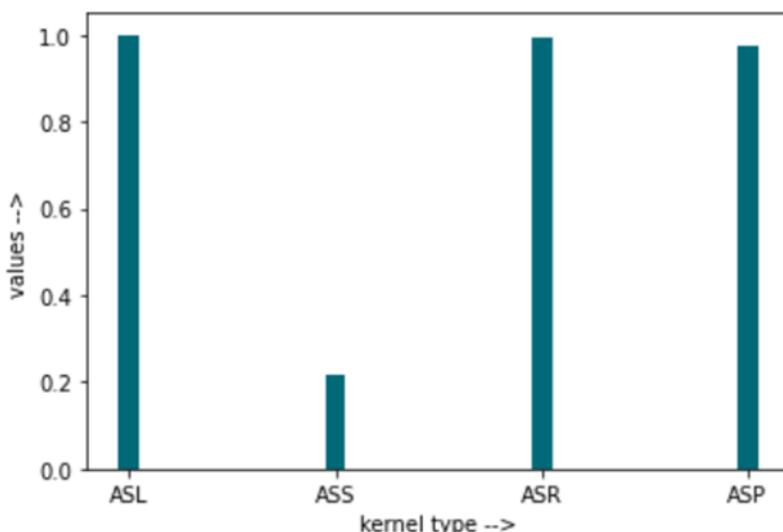
```
▶ from sklearn.metrics import accuracy_score
  as1=accuracy_score(ytest,yp1)
  as2=accuracy_score(ytest,yp2)
  as3=accuracy_score(ytest,yp3)
  as4=accuracy_score(ytest,yp4)
  print("Accuracy(linear)) = \n",as1)
  print("Accuracy(sigmoid)) = \n",as2)
  print("Accuracy(rbf)) = \n",as3)
  print("Accuracy(polynomial)) = \n",as4)
```

```
Accuracy(linear)) =
1.0
Accuracy(sigmoid)) =
0.3
Accuracy(rbf)) =
0.9666666666666667
Accuracy(polynomial)) =
1.0
```

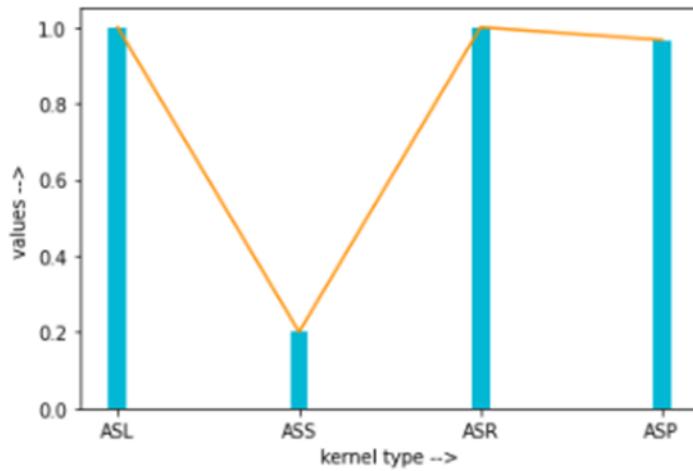
```
► import matplotlib.pyplot as plt
import numpy as np
scores=np.array((as1,as2,as3,as4))
values=np.array(['ASL','ASS','ASR','ASP'])
plt.bar(values,scores,width=0.1,color="#00b8d4")
plt.xlabel('kernel type -->')
plt.ylabel('values -->')
plt.show()
```



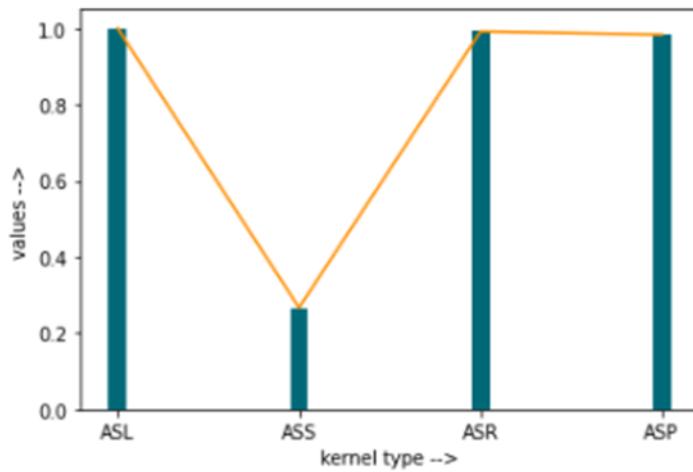
```
► import matplotlib.pyplot as plt
import numpy as np
score=np.array((s1,s2,s3,s4))
value=np.array(['ASL','ASS','ASR','ASP'])
plt.bar(value,score,width=0.1,color="#006978")
plt.xlabel('kernel type -->')
plt.ylabel('values -->')
plt.show()
```



```
▶ import matplotlib.pyplot as plt
import numpy as np
scores=np.array((as1,as2,as3,as4))
values=np.array(['ASL','ASS','ASR','ASP'])
plt.bar(values,scores,width=0.1,color="#00b8d4")
plt.plot(scores,color="#ff8f00")
plt.xlabel('kernel type -->')
plt.ylabel('values -->')
plt.show()
```



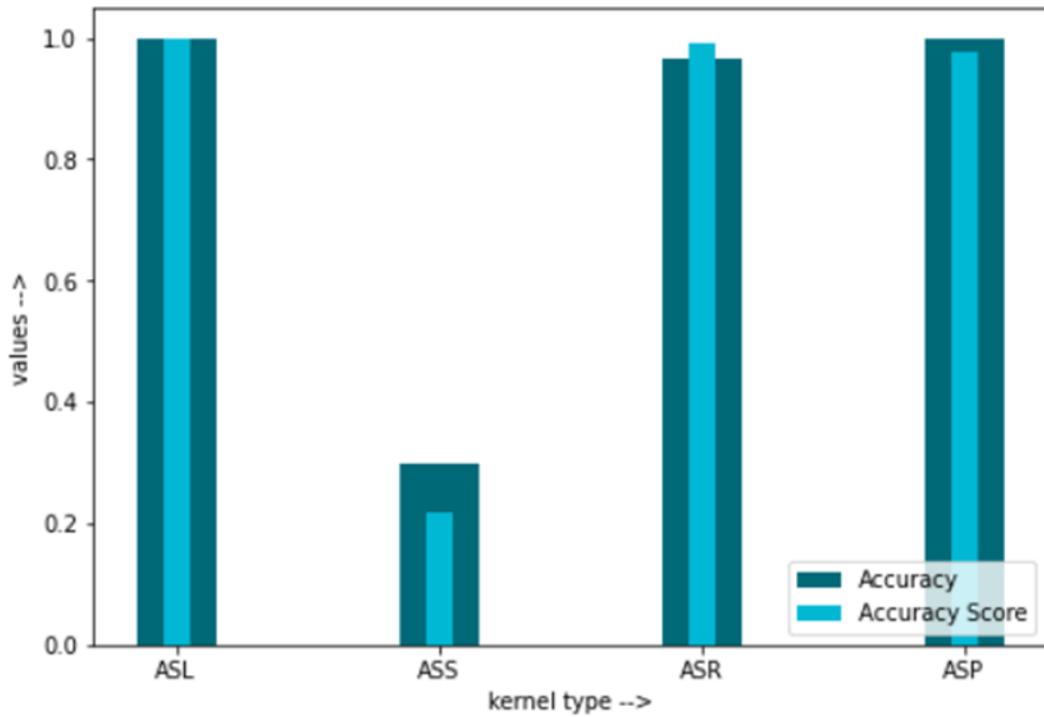
```
▶ import matplotlib.pyplot as plt
import numpy as np
score=np.array((s1,s2,s3,s4))
value=np.array(['ASL','ASS','ASR','ASP'])
plt.plot(score,color="#ff8f00")
plt.bar(value,score,width=0.1,color="#006978")
plt.xlabel('kernel type -->')
plt.ylabel('values -->')
plt.show()
```



```

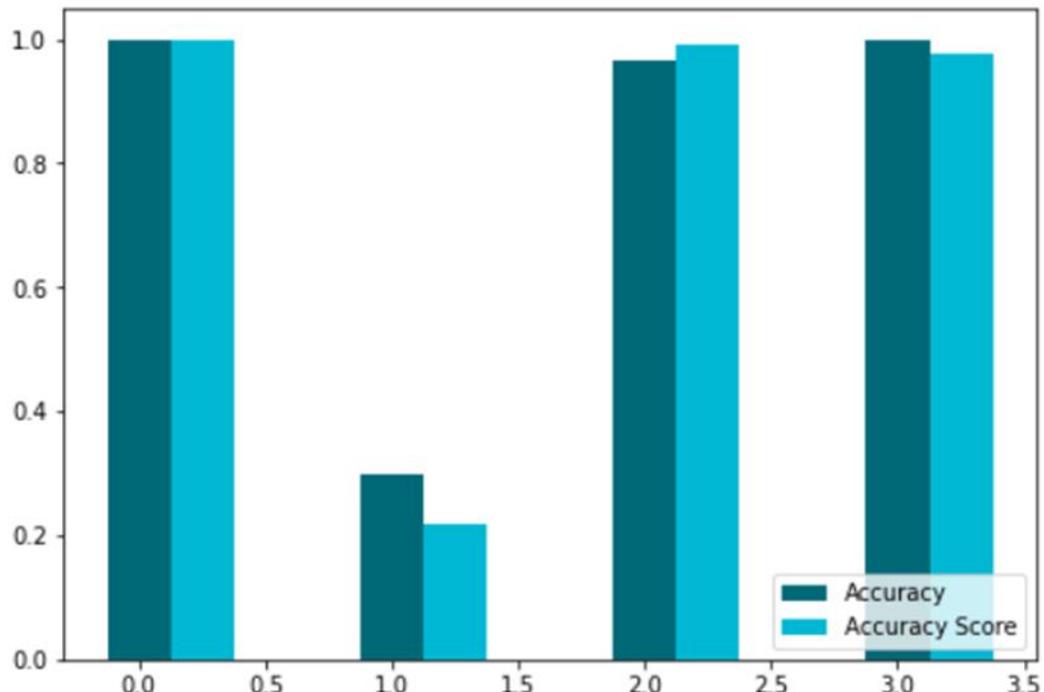
import matplotlib.pyplot as plt
import numpy as np
scores=np.array((as1,as2,as3,as4))
score=np.array((s1,s2,s3,s4))
value=np.array(['ASL','ASS','ASR','ASP'])
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(value,scores,width=0.3,color="#006978")
ax.bar(value,score,width=0.1,color="#00b8d4")
plt.xlabel('kernel type -->')
plt.ylabel('values -->')
plt.legend(["Accuracy", "Accuracy Score"], loc ="lower right")
plt.show()

```



```
▶ import numpy as np
import matplotlib.pyplot as plt
scores=np.array((as1,as2,as3,as4))
score=np.array((s1,s2,s3,s4))
X = np.arange(4)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(X + 0.00, scores, color = '#006978', width = 0.25)
ax.bar(X + 0.25, score, color = '#00b8d4', width = 0.25)
plt.legend(["Accuracy", "Accuracy Score"], loc ="lower right")
```

```
.]: <matplotlib.legend.Legend at 0x259a3e2ce50>
```



Question no. 2:

Implement the given dataset in python

- a) Load the dataset
- b) Remove the Null values from data
- c) Convert categorical values
- d) Separate the features and label from the data
- e) Split the training and test set
- f) Apply any classification model
- g) Test on test data set
- h) Print accuracy and score of the model

Import libraries

```
▶ import pandas as pd
    import matplotlib.pyplot as plt
    from sklearn.preprocessing import LabelEncoder,scale,StandardScaler
    from sklearn.metrics import accuracy_score
    from sklearn.model_selection import train_test_split
    import seaborn as sns
    import numpy as np
    from sklearn.naive_bayes import GaussianNB
```

a) Load the dataset

```
▶ df=pd.read_csv('data_car_csv.csv',encoding='unicode_escape')
    df.describe()
    df.head()
```

[2]:

	Age	Gender	City	Make	Model	Sum Insured	Label
0	45 Years	Male	Rawalpindi	S.Alto	2016	1057515	Accidental Claim
1	38 Years	Male	Rawalpindi	H.City	2016	1225000	No Claim
2	49 Years	Male	Rawalpindi	H.Civic	2013	2000000	Fraud Claim
3	40 Years	Male	Rawalpindi	S.Swift	2017	1225000	Fraud Claim
4	70 Years	Male	Rawalpindi	H.City	2015	1700000	Accidental Claim

b) Remove the Null values from data

```
▶ df.isna().head()  
df.isnull().sum()  
df.fillna('bfill',inplace=True)  
df.isna().sum()
```

```
[3]: Age          0  
Gender        0  
City          0  
Make          0  
Model         0  
Sum Insured   0  
Label         0  
dtype: int64
```

c) Convert categorical values

```
▶ df.head()  
lb=LabelEncoder()  
lb.fit(df['Gender'])  
df['Gender']=lb.transform(df['Gender'])  
lb1=LabelEncoder()  
lb1.fit(df['Age'])  
df['Age']=lb1.transform(df['Age'])  
lb2=LabelEncoder()  
lb2.fit(df['City'])  
df['City']=lb2.transform(df['City'])  
lb3=LabelEncoder()  
lb3.fit(df['Make'])  
df['Make']=lb3.transform(df['Make'])
```

d) Separate the features and label from the data

```
▶ x=df.iloc[:,0:6]
x.head()
```

[5]:

	Age	Gender	City	Make	Model	Sum Insured
0	23	4	9	9	2016	1057515
1	14	4	9	1	2016	1225000
2	28	4	9	2	2013	2000000
3	17	4	9	14	2017	1225000
4	41	4	9	1	2015	1700000

```
▶ y=df.iloc[:, -1]
y.head()
```

[6]:

```
0    Accidental Claim
1        No Claim
2      Fraud Claim
3      Fraud Claim
4    Accidental Claim
Name: Label, dtype: object
```

e) Split the training and test set

```
: ▶ xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
print("Xtrain = ",xtrain.shape)
print("Xtest = ",xtest.shape)
print("Ytrain = ",ytrain.shape)
print("Ytest = ",ytest.shape)

Xtrain = (2027, 6)
Xtest = (507, 6)
Ytrain = (2027,)
Ytest = (507,)
```

f) Apply any classification model

```
Model=GaussianNB()
Model.fit(xtrain,ytrain)
print('Accuracy score = ',Model.score(xtrain,ytrain))
```

Accuracy score = 0.23186975826344353

g) Test on test data set

h) Print accuracy and score of the model

```
▶ print("Accuracy = \n",accuracy_score(ytest,ypred))
```

```
Accuracy =
0.26429980276134124
```

LAB 10

(CLO 2-3-4, PLO 3-5-10,P3-P4-A10)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	Random Forest and Decision Trees:
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Classify a dataset by implementing random forest and decision trees

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Random Forest and decision trees

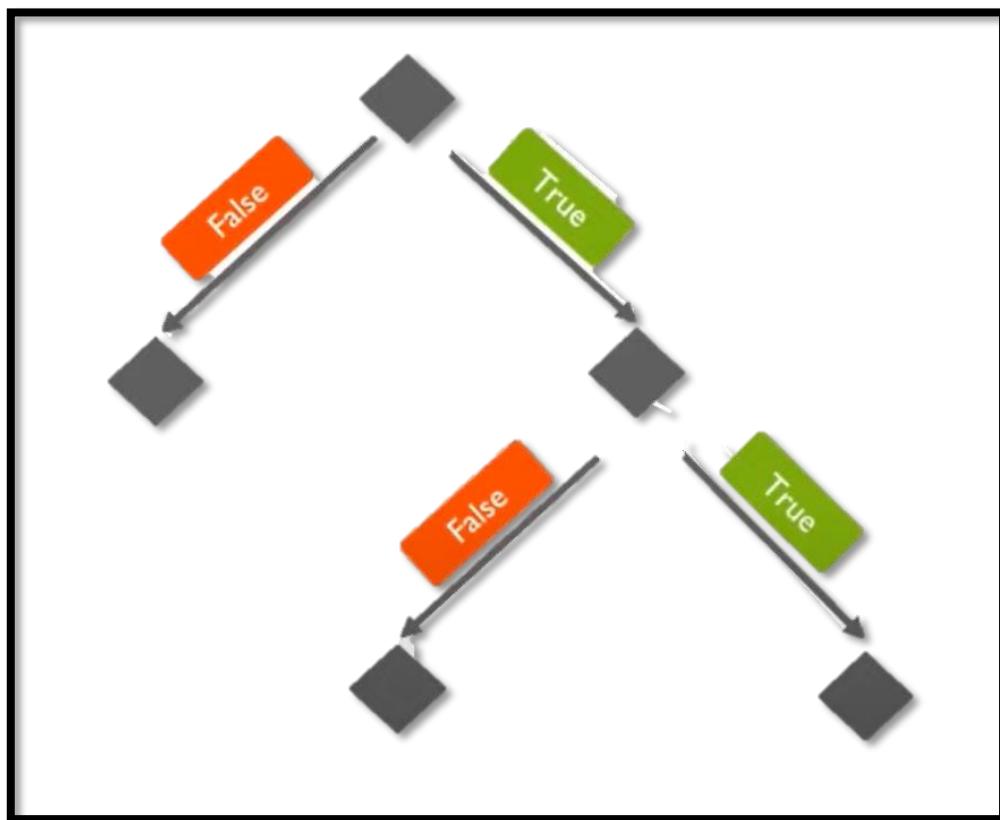
Task no. 1:

Explain and implement Random forest Classification Model

What is Random Forest?

Random Forest is the most used supervised machine learning algorithm for classification and regression

Random Forest are made out of decision trees



Why Random Forest?

Decision Trees are easy to build and interpret.

Then WHY Random Forest?

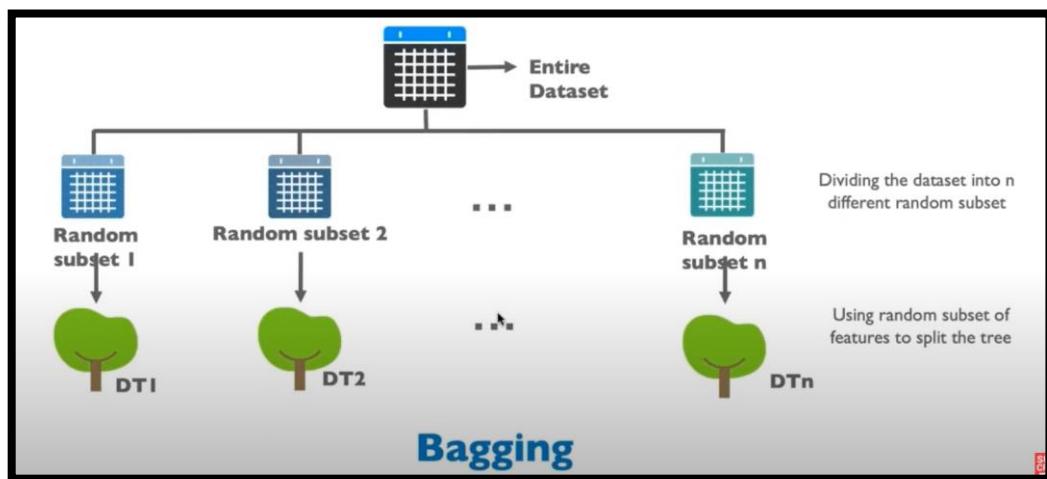
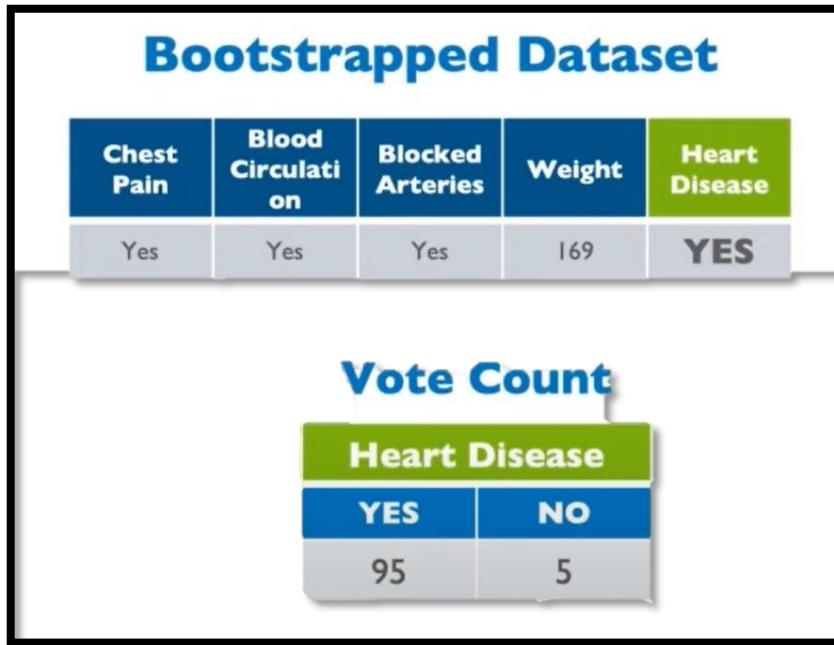
Decision Trees have only one aspect, therefore are less accurate and inflexible when it comes to classifying new samples

Ensemble Learning:

Random forest uses Ensemble learning method in which the predictions are based on the combined results of various individual models

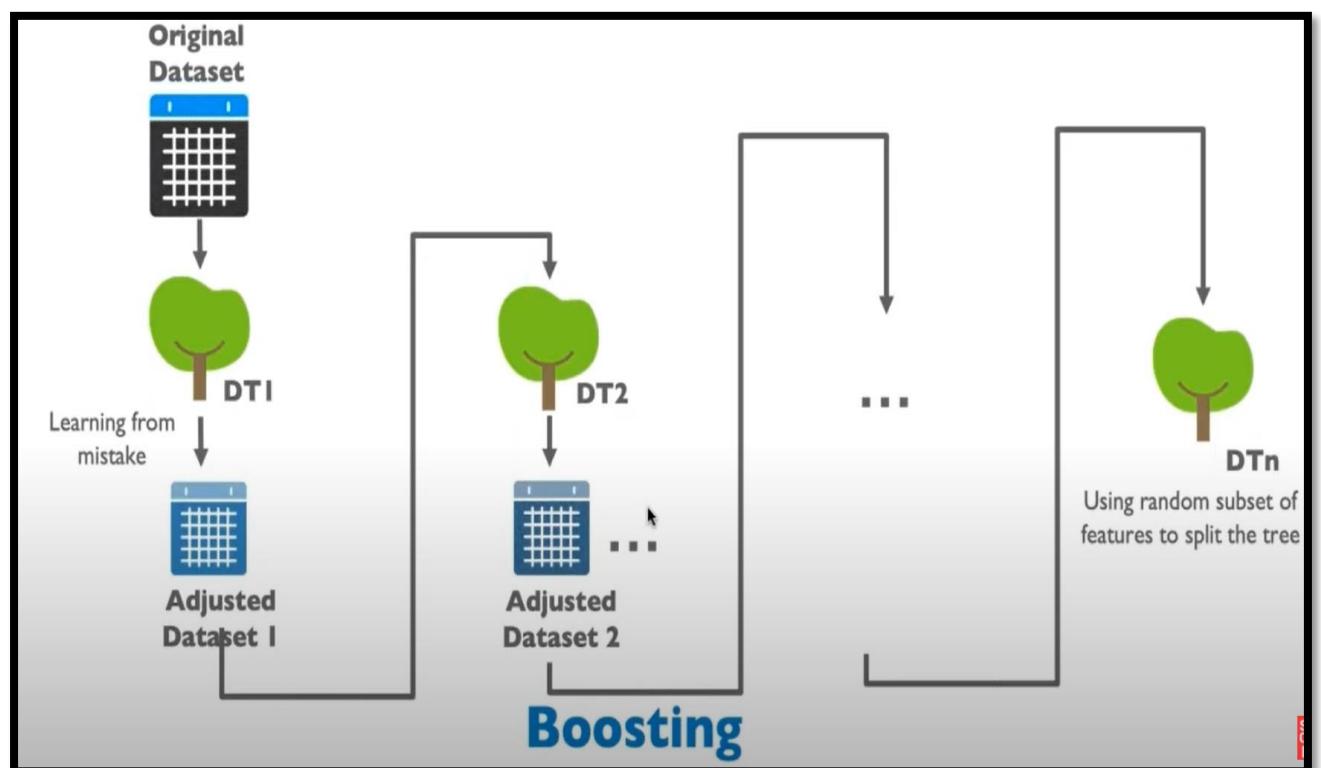
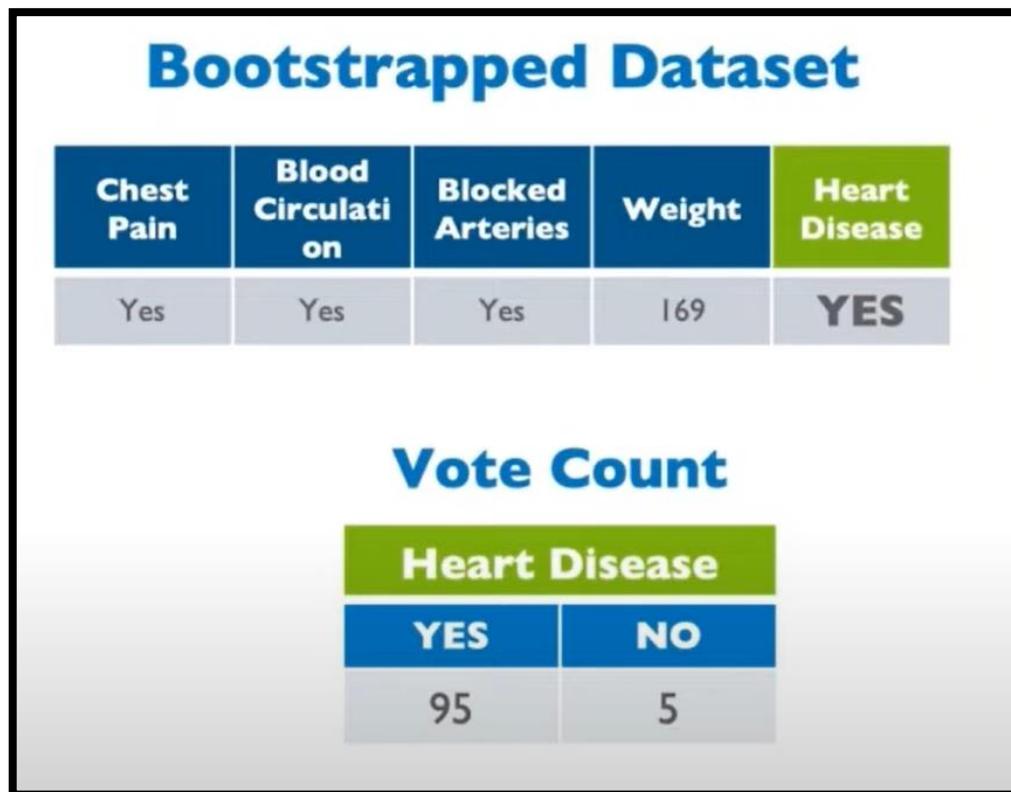
Bagging :

Bootstrapping the data and using its aggregate to make a decision is known as Bagging. In other words, Bagging is training a bunch of individual models parallelly, and each model is trained by a random subset of the data.



Boosting:

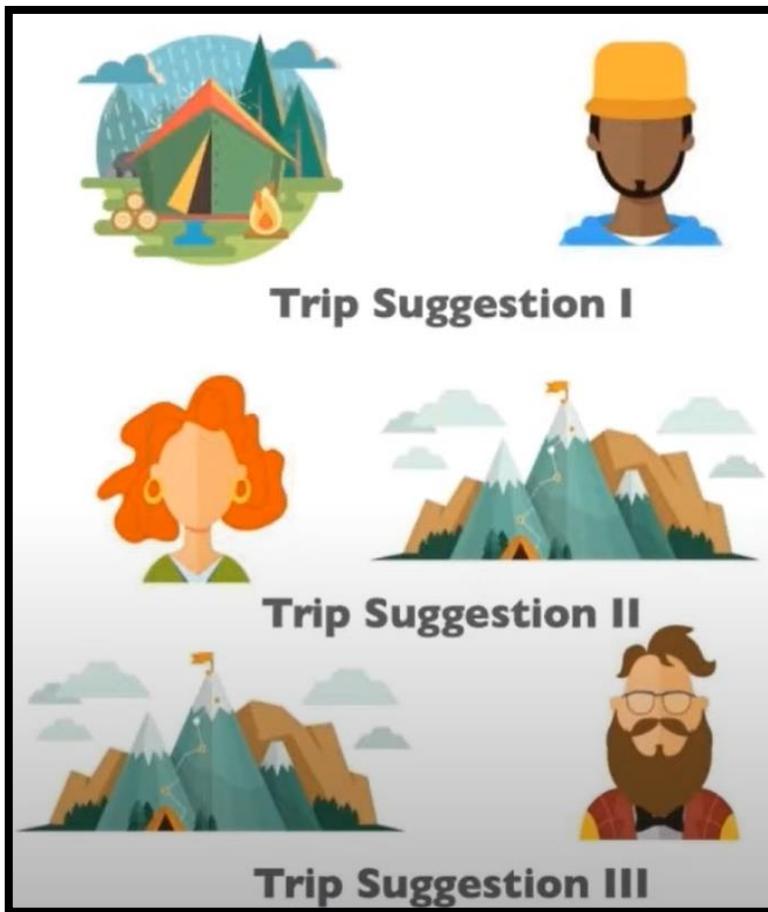
Boosting is training a bunch of individual models in a sequential way. Each individual model learns from mistakes made by the previous model.



Random forest analogy:

Chandler is planning for a one-year vacation trip. So in order to decide which places should he travel to, he asks his friends for their advice.

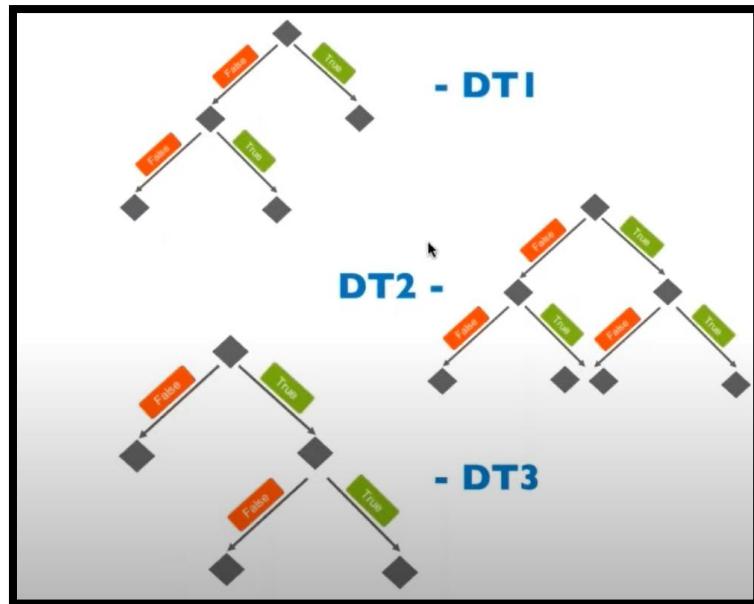
Every friends gave suggestion by asking him few questions



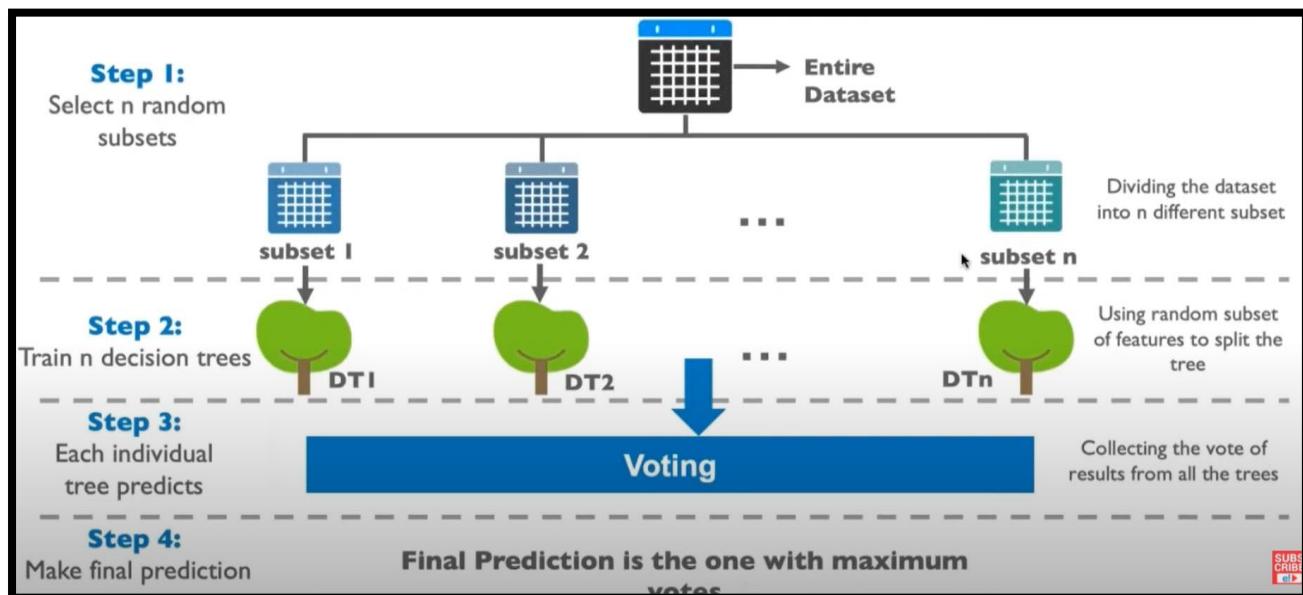
Later on, Chandler asked more of his friends to advise him. Once again, his friends asked him different questions to recommend about the places.

Now after talking to all of his friends he decided to visit the place with most number of votes. the above scenario is a typical example of Random Forest Algorithm.

What is a Random forest:



Creating a random forest steps involved:



Step 1: Creating a Bootstrapped dataset:

The bootstrap method is a technique used to estimate statistics on a population by sampling a dataset with replacement. It can be used to estimate summary statistics such as the mean or standard deviation.

The bootstrap dataset (same size as original) is created by randomly selecting samples from the original dataset.

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

This is our sample dataset...

Note :you can pick same sample more than once

Original Dataset

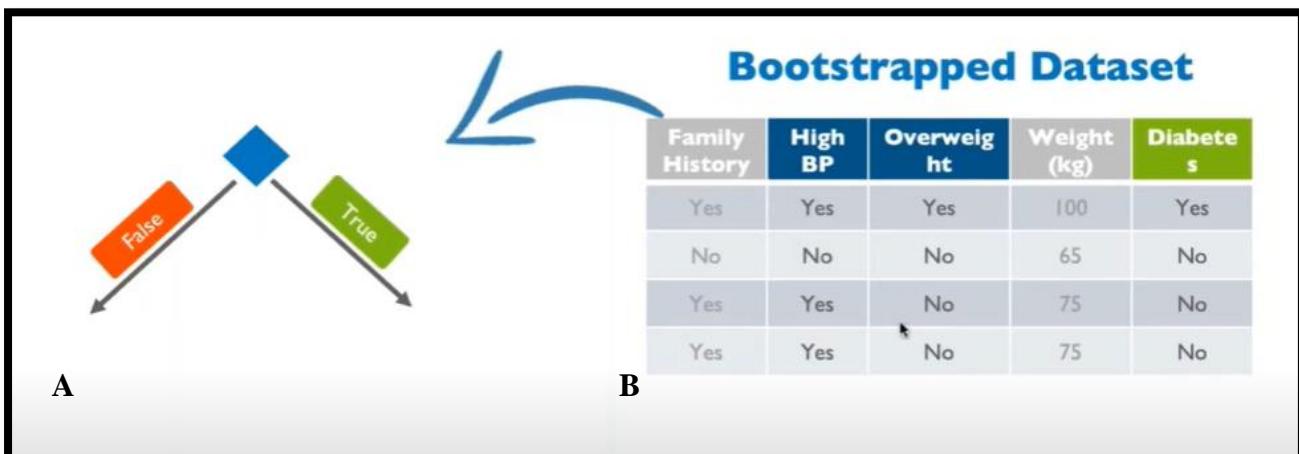
Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No



Step 2:creating decision tree



B.Instead of considering all the 4 variables to figure out how to split the root node. In this example, consider only 2 variables at each Step

A.Create a decision tree (eg: Root node -Overweight) using the bootstrapped dataset. Use only a random subset of variables/column at each step

Bootstrapped Dataset

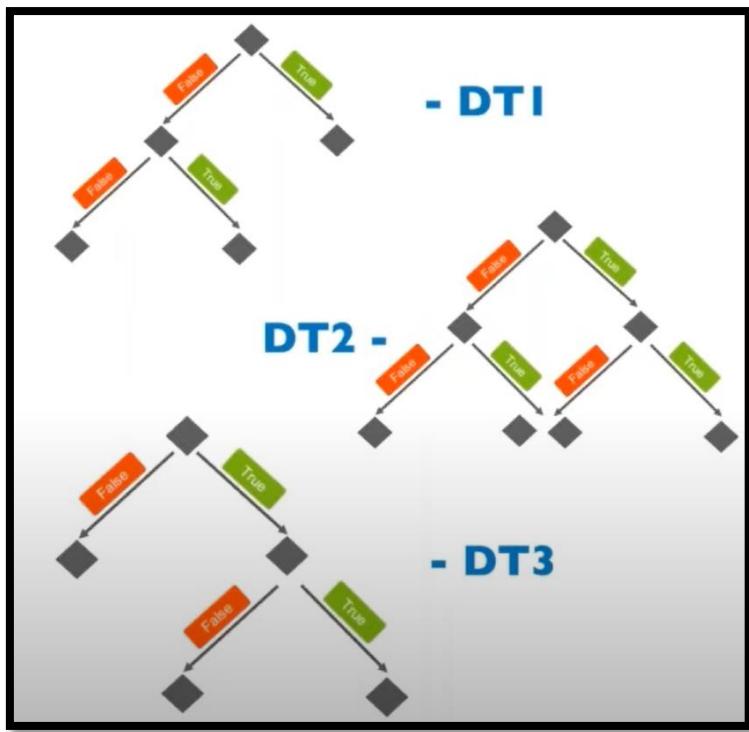
Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

The decision tree is ready using the random subset of variables at each step

Now we need to figure out, how to split samples at this node

Original Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes



Recursively splitting sample at other nodes

Get back to Step I and repeat: Build new bootstrapped dataset and rebuild decision trees considering subset of variables at each step (ideally you have to repeat this step 100's of time)

Step 3 &4: counting the votes for predicting:

Bootstrapped Dataset				
Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	No	75	?

Predict if the patient has diabetes or not

Using a bootstrapped sample and considering only a subset of variables at each step results in a wide variety of trees. This variety is what makes random forest more effective than individual decision trees

Predict if the patient has diabetes or not Let's say we created 100 decision tree and this is the overall vote count of the predictions from all the decision tree. Next thing is to find out the option which received most votes



Bootstrapped Dataset				
Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	No	75	?

Predict if the patient has diabetes or not

Vote Count	
Diabetes	
YES	NO
95	5

In this case, 'YES' received most of the votes, → patient has heart disease

How good is your model:

Original Dataset					Bootstrapped Dataset				
Family History	High BP	Overweight	Weight (kg)	Diabetes	Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No	No	No	No	65	No
Yes	Yes	Yes	100	Yes	Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No	Yes	Yes	No	75	No
Yes	No	Yes	110	Yes	Yes	Yes	No	75	No

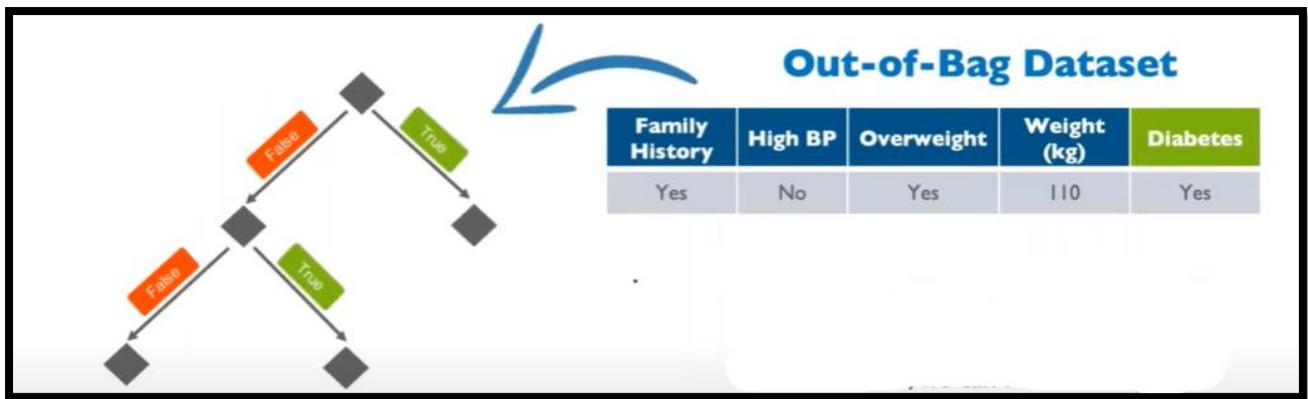
Remember! this entry (almost 1/3rd of the original dataset) was not included in the bootstrapped dataset because of the duplicate entry

Out-of-Bag Dataset

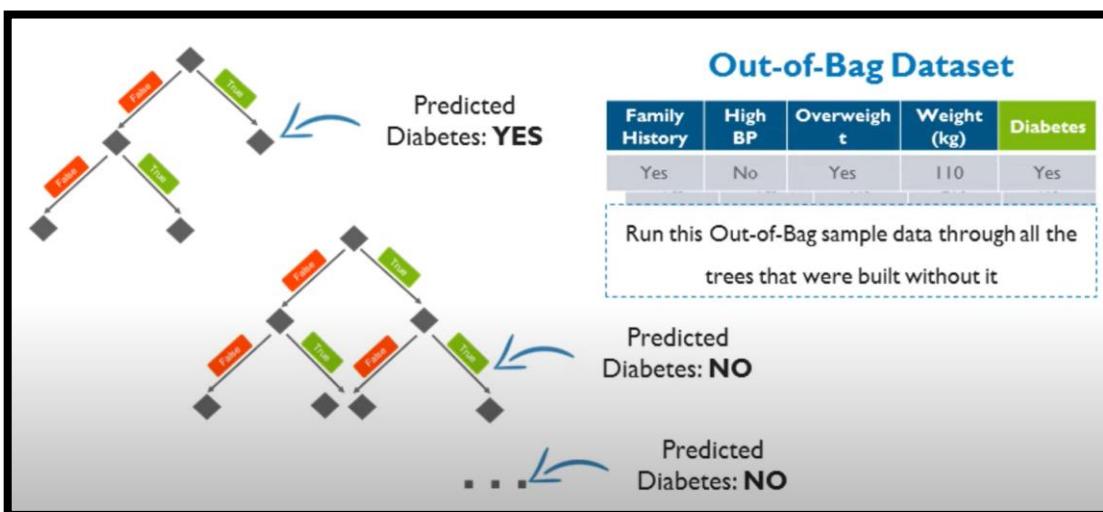
Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	No	Yes	110	Yes

This is the entry that did not end up in the bootstrapped dataset and the collection of such entries as a dataset is known as "Out of Bag Dataset"

NOTE: Just in case, the original dataset were larger, we would have more than 1 entry over here



Since Out-of-Bag Dataset was not used to create this tree, we can run it through and check if it correctly classifies the sample as "Yes" in Diabetes



Next we do the same for all of the other Out of-Bag samples for all the trees, and finally the accuracy of random forest can be measured by the proportion of Out-of-Bag samples that were correctly classified by the algorithm

Classification of Out-of-Bag Dataset	
Diabetes	
YES	NO
1	4
Diabetes	
YES	NO
4	0
Diabetes	
YES	NO
3	1

Out-of-Bag Dataset				
Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No

Next we do the same for all of the other Out-of-Bag samples for all the trees, and finally the accuracy of random forest can be measured by the proportion of Out-of-Bag samples that were correctly classified by the algorithm

Remember when we built our first tree, we only used 2 variables to make decision at each step?

Family History	High BP	Overweight	Weight (kg)	Diabetes	2 Variables
Yes	Yes	Yes	100	Yes	
No	No	No	65	No	
Yes	Yes	No	75	No	
Yes	Yes	No	75	No	

↓

Family History	High BP	Overweight	Weight (kg)	Diabetes	3 Variables
Yes	Yes	Yes	100	Yes	
No	No	No	65	No	
Yes	Yes	No	75	No	
Yes	Yes	No	75	No	

Compare the Out-of-Bag error for a random forest built using 2 variables vs 3 variables and select the most accurate random forest

Import Libraries

```
▶ import pandas as pd  
  
▶ from sklearn.model_selection import train_test_split  
▶ from sklearn.ensemble import RandomForestClassifier  
▶ from sklearn.metrics import accuracy_score  
▶ import matplotlib.pyplot as plt  
▶ import seaborn as sns  
▶ from sklearn import datasets  
  
▶ #df=pd.read_csv('Iris.csv')  
▶ #df.head()
```

Load Dataset

```
▶ data=datasets.load_iris()
print(data.target_names)
print(data.feature_names)

['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

Display first 5 rows

Convert to dataframe

```
: ▶ df=pd.DataFrame({'sp_len':data.data[:,0],  
'sp_wdh':data.data[:,1],  
'ptl_len':data.data[:,2],  
'ptl_wdh':data.data[:,3],  
'ouptut' :data.target}  
)  
df
```

t[7]:

	sp_len	sp_wdh	ptl_len	ptl_wdh	ouptut
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

Checking head,shape,size

```
: ┌ df.head()  
t[8]:  
      sp_len  sp_wdh  ptl_len  ptl_wdh  ouptut  
    0      5.1      3.5      1.4      0.2      0  
    1      4.9      3.0      1.4      0.2      0  
    2      4.7      3.2      1.3      0.2      0  
    3      4.6      3.1      1.5      0.2      0  
    4      5.0      3.6      1.4      0.2      0  
  
: ┌ df.shape  
t[9]: (150, 5)  
  
: ┌ df.size  
t[10]: 750
```

Separating features and labels

```
: ┌ x=df.iloc[:,0:4]  
y=df.iloc[:, -1]  
  : ┌ x  
t[12]:  
      sp_len  sp_wdh  ptl_len  ptl_wdh  
    0      5.1      3.5      1.4      0.2  
    1      4.9      3.0      1.4      0.2  
    2      4.7      3.2      1.3      0.2  
    3      4.6      3.1      1.5      0.2  
    4      5.0      3.6      1.4      0.2  
    ...     ...      ...      ...      ...  
   145     6.7      3.0      5.2      2.3  
   146     6.3      2.5      5.0      1.9  
   147     6.5      3.0      5.2      2.0  
   148     6.2      3.4      5.4      2.3  
   149     5.9      3.0      5.1      1.8  
  
150 rows × 4 columns
```

```
▶ y  
[3]: 0      0  
     1      0  
     2      0  
     3      0  
     4      0  
     ..  
    145     2  
    146     2  
    147     2  
    148     2  
    149     2  
Name: ouput, Length: 150, dtype: int32
```

Splitting data to test and train

```
▶ xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)  
print("Xtrain = ",xtrain.shape)  
print("Xtest = ",xtest.shape)  
print("Ytrain = ",ytrain.shape)  
print("Ytest = ",ytest.shape)  
  
Xtrain = (120, 4)  
Xtest = (30, 4)  
Ytrain = (120,)  
Ytest = (30,)
```

Algorithm selection

```
:   ▶ RFC=RandomForestClassifier()  
:  
:   ▶ RFC.fit(xtrain,ytrain)  
[16]: RandomForestClassifier()  
:  
:   ▶ RFC.score(xtrain,ytrain)  
[17]: 1.0
```

Predictions

```
▶ yp=RFC.predict(xtest)
```

```
▶ yp
```

```
19]: array([0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 2, 0, 1, 2, 2, 2, 2, 1, 1, 0, 0, 2,  
           2, 2, 2, 0, 1, 0, 0, 2])
```

```
▶ ass=accuracy_score(ytest,yp)
```

```
ass
```

```
20]: 0.9666666666666666
```

Hyper Parameter tuning

```
▶ #RFC=RandomForestClassifier(n_estimators=50)  
#No of samples  
#by default 100
```

For Random values predictions

```
▶ R=RFC.predict([[2,4,1,6]])[0]
```

```
▶ print(data.target_names[R])
```

```
setosa
```

N_estimator:1

Algorithm selection

```
: └─ RFC=RandomForestClassifier(n_estimators=1)
: └─ RFC.fit(xtrain,ytrain)
[16]: RandomForestClassifier(n_estimators=1)
: └─ RFC.score(xtrain,ytrain)
[17]: 0.9833333333333333
```

Predictions

```
└─ yp=RFC.predict(xtest)
└─ yp
[9]: array([0, 2, 0, 2, 2, 2, 1, 0, 2, 1, 0, 1, 2, 0, 1, 1, 0, 2, 1, 1, 1, 2,
          0, 1, 2, 1, 2, 0, 2, 1])
└─ ass=accuracy_score(ytest,yp)
└─ ass
[20]: 0.9666666666666667
```

For Random values predictions

```
└─ R=RFC.predict([[2,4,1,6]])[0]
```

```
└─ print(data.target_names[R])
    versicolor
```

N_estimator:50

Algorithm selection

```
▶ RFC=RandomForestClassifier(n_estimators=50)
▶ RFC.fit(xtrain,ytrain)
[48]: RandomForestClassifier(n_estimators=50)
▶ RFC.score(xtrain,ytrain)
[49]: 1.0
```

Predictions

```
: ▶ yp=RFC.predict(xtest)
:
: ▶ yp
[51]: array([0, 1, 0, 0, 0, 2, 1, 2, 2, 2, 0, 2, 1, 1, 0, 2, 2, 0, 2, 1, 2,
         2, 0, 1, 0, 0, 2, 0, 2])
:
: ▶ ass=accuracy_score(ytest,yp)
ass
[52]: 1.0
```

For Random values predictions

```
▶ R=RFC.predict([[2,4,1,6]])[0]
```

```
▶ print(data.target_names[R])
```

setosa

N_estimator:100

Algorithm selection

```
▶ RFC=RandomForestClassifier(n_estimators=100)
▶ RFC.fit(xtrain,ytrain)
81]: RandomForestClassifier()
▶ RFC.score(xtrain,ytrain)
82]: 1.0
```

Predictions

```
: ▶ yp=RFC.predict(xtest)
:
: ▶ yp
[84]: array([0, 2, 1, 2, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 2, 2, 1, 1, 1, 2, 1, 1, 2,
           2, 1, 0, 0, 0, 0, 0, 1])
:
: ▶ ass=accuracy_score(ytest,yp)
ass
[85]: 1.0
```

For Random values predictions

```
▶ R=RFC.predict([[2,4,1,6]])[0]
```

```
▶ print(data.target_names[R])
setosa
```

Task no. 2:

Explain and Decision Tree Classification Model

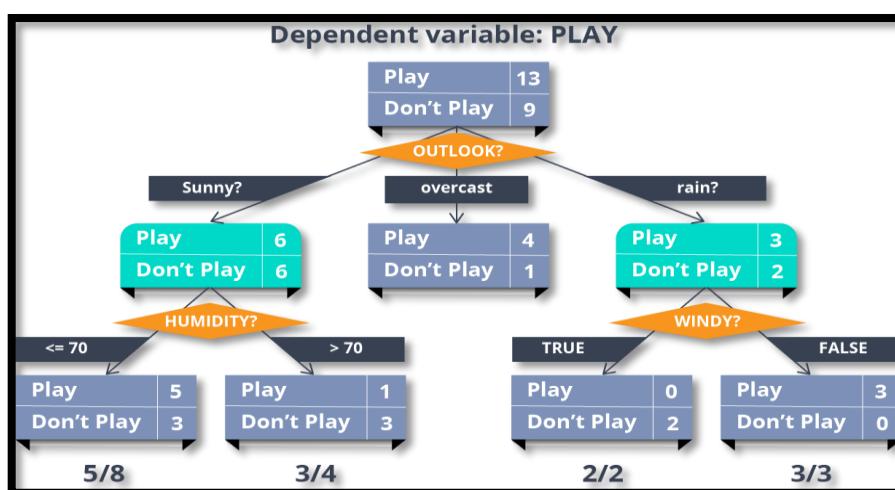
A *Decision Tree* has many analogies in real life and turns out, it has influenced a wide area of Machine Learning, covering both Classification and Regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making.

What is a Decision Tree?

A decision tree is a map of the possible outcomes of a series of related choices. It allows an individual or organization to weigh possible actions against one another based on their costs, probabilities, and benefits.

As the name goes, it uses a tree-like model of decisions. They can be used either to drive informal discussion or to map out an algorithm that predicts the best choice mathematically.

A decision tree typically starts with a single node, which branches into possible outcomes. Each of those outcomes leads to additional nodes, which branch off into other possibilities. This gives it a tree-like shape.



There are three different types of nodes: chance nodes, decision nodes, and end nodes. A chance node, represented by a circle, shows the probabilities of certain results. A decision node, represented by a square, shows a decision to be made, and an end node shows the final outcome of a decision path.

Advantages & Disadvantages of Decision Trees

Advantages

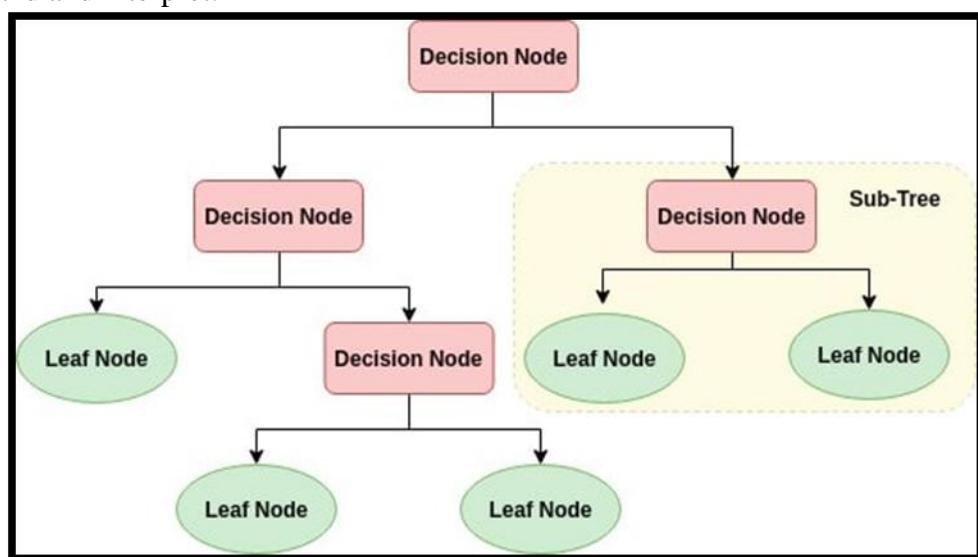
- Decision trees generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are capable of handling both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

Disadvantages

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and a relatively small number of training examples.
- Decision trees can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. [Pruning algorithms](#) can also be expensive since many candidate sub-trees must be formed and compared.

Decision Tree Algorithm

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.



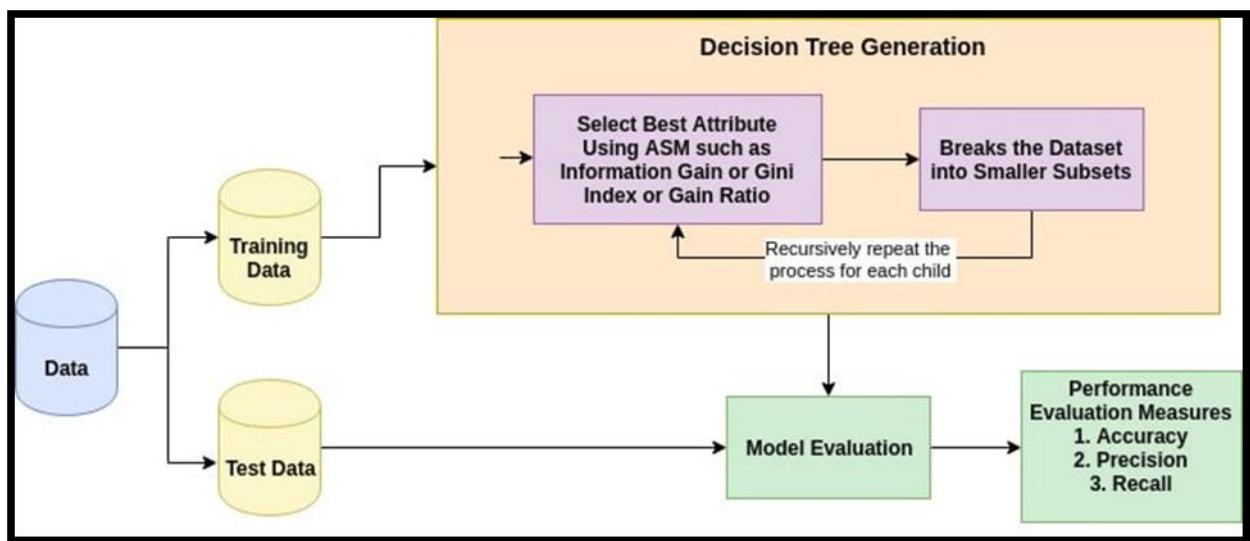
Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision

tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

How does the Decision Tree algorithm work?

The basic idea behind any decision tree algorithm is as follows:

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.
2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
 - o All the tuples belong to the same attribute value.
 - o There are no more remaining attributes.
 - o There are no more instances.



Attribute Selection Measures

Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner. It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node. ASM provides a rank to each feature(or attribute) by explaining the given dataset. Best score attribute will be selected as a splitting attribute

([Source](#)). In the case of a continuous-valued attribute, split points for branches also need to define. Most popular selection measures are Information Gain, Gain Ratio, and Gini Index.

Information Gain

Shannon invented the concept of entropy, which measures the impurity of the input set. In physics and mathematics, entropy referred as the randomness or the impurity in the system. In information theory, it refers to the impurity in a group of examples. Information gain is the decrease in entropy. Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

$$\text{Info}(D) = - \sum_{i=1}^m p_i \log_2 p_i$$

Where, P_i is the probability that an arbitrary tuple in D belongs to class C_i .

$$\text{Info}_A(D) = \sum_{j=1}^V \frac{|D_j|}{|D|} \times \text{Info}(D_j)$$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

Where,

- $\text{Info}(D)$ is the average amount of information needed to identify the class label of a tuple in D .
- $|D_j|/|D|$ acts as the weight of the j th partition.
- $\text{Info}_A(D)$ is the expected information required to classify a tuple from D based on the partitioning by A .

The attribute A with the highest information gain, $\text{Gain}(A)$, is chosen as the splitting attribute at node $N()$.

Gain Ratio

Information gain is biased for the attribute with many outcomes. It means it prefers the attribute with a large number of distinct values. For instance, consider an attribute with a unique

identifier such as customer_ID has zero info(D) because of pure partition. This maximizes the information gain and creates useless partitioning.

C4.5, an improvement of ID3, uses an extension to information gain known as the gain ratio. Gain ratio handles the issue of bias by normalizing the information gain using Split Info. Java implementation of the C4.5 algorithm is known as J48, which is available in WEKA data mining tool.

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

Where,

- $|D_j|/|D|$ acts as the weight of the jth partition.
- v is the number of discrete values in attribute A.

The gain ratio can be defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

The attribute with the highest gain ratio is chosen as the splitting attribute ([Source](#)).

Gini index

Another decision tree algorithm CART (Classification and Regression Tree) uses the Gini method to create split points.

$$Gini(D) = 1 - \sum_{i=1}^m P_i^2$$

Where, pi is the probability that a tuple in D belongs to class Ci.

The Gini Index considers a binary split for each attribute. You can compute a weighted sum of the impurity of each partition. If a binary split on attribute A partitions data D into D1 and D2, the Gini index of D is:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

In case of a discrete-valued attribute, the subset that gives the minimum gini index for that chosen is selected as a splitting attribute. In the case of continuous-valued attributes, the strategy is to select each pair of adjacent values as a possible split-point and point with smaller gini index chosen as the splitting point.

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

The attribute with minimum Gini index is chosen as the splitting attribute.

Decision Tree

Import Datasets

```
▶ import pandas as pd
  from sklearn.model_selection import train_test_split
  from sklearn.tree import DecisionTreeClassifier
  from sklearn.metrics import accuracy_score
  import matplotlib.pyplot as plt
  import seaborn as sns
  from sklearn import datasets
```

Load Dataset

```
▶ data=datasets.load_iris()
  print(data.target_names)
  print(data.feature_names)

['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

Convert to Dataframe

```
[1]: df=pd.DataFrame({'sp_len':data.data[:,0],  
                   'sp_wdh':data.data[:,1],  
                   'ptl_len':data.data[:,2],  
                   'ptl_wdh':data.data[:,3],  
                   'ouputut' :data.target})  
df
```

[3]:

	sp_len	sp_wdh	ptl_len	ptl_wdh	ouputut
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

Checking head,shape,size

```
[4]: df.head()
```

[4]:

	sp_len	sp_wdh	ptl_len	ptl_wdh	ouputut
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
[5]: df.shape
```

[5]: (150, 5)

```
[6]: df.size
```

[6]: 750

Separating features and labels

```
In [8]: x=df.iloc[:,0:4]  
y=df.iloc[:, -1]
```

```
In [8]: x
```

[8]:

	sp_len	sp_wdh	ptl_len	ptl_wdh
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
In [8]: y
```

[8]: 0 0
1 0
2 0
3 0
4 0
..
145 2
146 2
147 2
148 2
149 2

Name: output, Length: 150, dtype: int32

Splitting Data

```
: ┌─▶ xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
  print("Xtrain = ",xtrain.shape)
  print("Xtest = " ,xtest.shape)
  print("Ytrain = " ,ytrain.shape)
  print("Ytest = " ,ytest.shape)

Xtrain =  (120, 4)
Xtest =  (30, 4)
Ytrain =  (120,)
Ytest =  (30,)
```

Algorithm Selection

```
: ┌─▶ clf = DecisionTreeClassifier()
:
: ┌─▶ clf.fit(xtrain,ytrain)
[12]: DecisionTreeClassifier()

: ┌─▶ clf.score(xtrain,ytrain)
[13]: 1.0
```

Predictions

```
┐ yp=clf.predict(xtest)

┐ yp
[5]: array([1, 2, 2, 0, 1, 2, 0, 0, 0, 2, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1,
  2, 1, 1, 1, 0, 2, 2, 0])
```

Accuracy Score

Accuracy Score

```
▶ ass=accuracy_score(ytest,yp)  
ass
```

```
[6]: 0.9333333333333333
```

For Random values predictions

```
▶ C=clf.predict([[2,4,1,6]])[0]
```

```
print(data.target_names[C])
```

```
versicolor
```

LAB 11

(CLO 2-3-4, PLO 3-5-10,P3-P4-A4)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	K-Means and DBSCAN Algorithms
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Classify a data points by implementing K-Means and dbscan

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

K-means and

DBSCAN

Question no. 1:

Solve and implement K- Means clustering algorithm on IRIS dataset

Solution:

What is Clustering?

Clustering is dividing data points into homogeneous classes or clusters:

- Points in the same group are as similar as possible
- Points in different group are as dissimilar as possible

When a collection of objects is given, we put objects into group based on similarity.

Application of Clustering:

Clustering is used in almost all the fields. You can infer some ideas from Example 1 to come up with lot of clustering applications that you would have come across.

Listed here are few more applications, which would add to what you have learnt.

- Clustering helps marketers improve their customer base and work on the target areas. It helps group people (according to different criteria's such as willingness, purchasing power etc.) based on their similarity in many ways related to the product under consideration.
- Clustering helps in identification of groups of houses on the basis of their value, type and geographical locations.
- Clustering is used to study earth-quake. Based on the areas hit by an earthquake in a region, clustering can help analyse the next probable location where earthquake can occur.

Clustering Algorithms:

A Clustering Algorithm tries to analyse natural groups of data on the basis of some similarity. It locates the centroid of the group of data points. To carry out effective clustering, the algorithm evaluates the distance between each point from the centroid of the cluster.

The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data.

What is K-means Clustering?

K-means (Macqueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining.

K-means Clustering – Example 1:

A pizza chain wants to open its delivery centres across a city. What do you think would be the possible challenges?

- They need to analyse the areas from where the pizza is being ordered frequently.
- They need to understand as to how many pizza stores has to be opened to cover delivery in the area.
- They need to figure out the locations for the pizza stores within all these areas in order to keep the distance between the store and delivery points minimum.

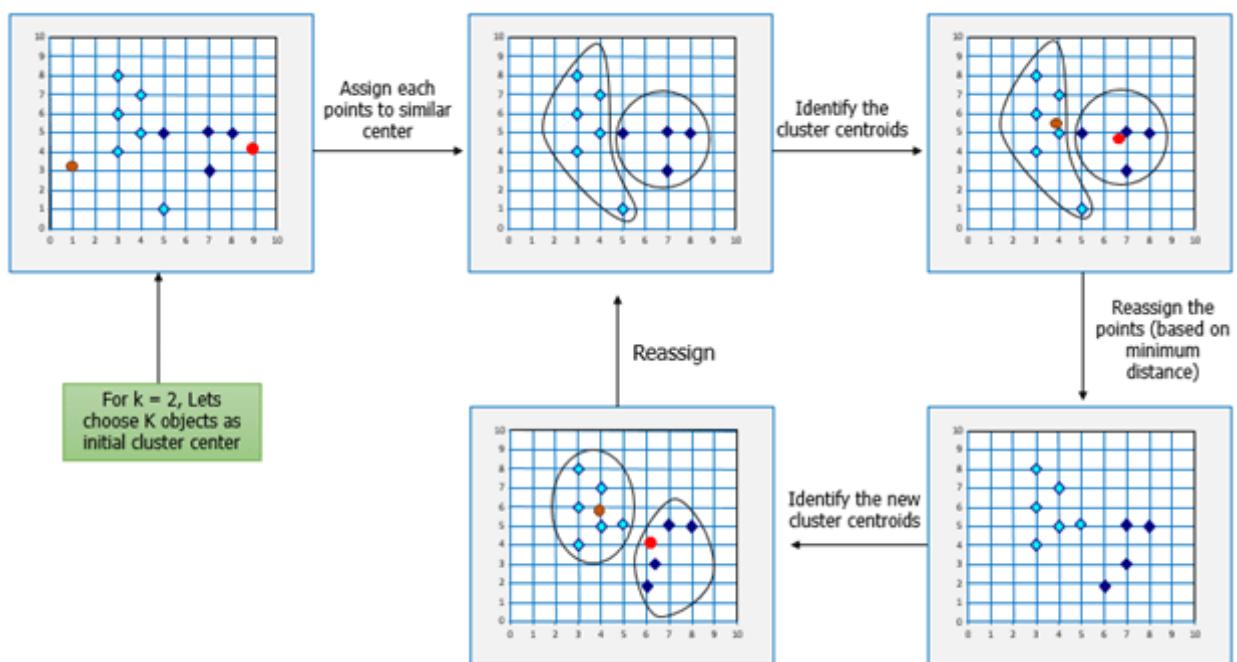
Resolving these challenges includes a lot of analysis and mathematics. We would now learn about how clustering can provide a meaningful and easy method of sorting out such real life challenges. Before that let's see what clustering is.

K-means Clustering Method:

If k is given, the K-means algorithm can be executed in the following steps:

- Partition of objects into k non-empty subsets
- Identifying the cluster centroids (mean point) of the current partition.
- Assigning each point to a specific cluster
- Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum.
- After re-allotting the points, find the centroid of the new cluster formed.

The step by step process:



Solve by KNN using the given Dataset

185	72
170	56
168	60
179	68
182	72
188	77
180	71
180	70
183	84
180	88
180	67
177	76

Solution:

R₁	185	72
R₂	170	56
R₃	168	60
R₄	179	68
R₅	182	72
R₆	188	77
R₇	180	71
R₈	180	70
R₉	183	84
R₁₀	180	88
R₁₁	180	67
R₁₂	177	76

Number of clusters=2

Selecting centroids (randomly)

No. of Centroids=No. of Clusters

R₁ :Centroid 1: 185,72

R₂ :Centroid 2: 170,56

$$\text{Distance formula: } \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\text{Euclidean distance : } \sqrt{(x_0 - x_c)^2 + (y_0 - y_c)^2}$$

For finding in which Cluster thus, R₃ belongs:

$$R_3 = (168, 60)$$

$$C_1 = (185, 72)$$

$$C_2 = (170, 56)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(168 - 185)^2 + (60 - 72)^2}$$

$$C_1 = 20.808$$

$$C_2 = \sqrt{(168 - 170)^2 + (60 - 56)^2}$$

$$C_2 = 4.472$$

As C₁>C₂

R₃ lies in C₂

Changing centroid:

Taking mean of R₂ & C₂

$$= \frac{168+170}{2}, \frac{60+56}{2}$$

$$= 169, 58$$

New C₂=(169,58)

For finding in which Cluster thus, R₄ belongs:

$$R_4 = (179, 68)$$

$$C_1 = (185, 72)$$

$$C_2 = (169, 58)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(179 - 185)^2 + (68 - 72)^2}$$

$$C_1 = 7.211$$

$$C_2 = \sqrt{(179 - 169)^2 + (68 - 58)^2}$$

$$C_2 = 14.14$$

As C₂>C₁

R₄ lies in C₁

Changing centroid:

Taking mean of R₄ & C₁

$$= \frac{185+179}{2}, \frac{72+68}{2}$$

$$= 182, 70$$

$$\text{New } C_1 = (182, 70)$$

For finding in which Cluster thus, R₅ belongs:

$$R_5 = (182, 72)$$

$$C_1 = (182, 70)$$

$$C_2 = (169, 58)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(182 - 182)^2 + (72 - 70)^2}$$

$$C_1 = 2$$

$$C_2 = \sqrt{(182 - 169)^2 + (72 - 58)^2}$$

$$C_2 = 19.10$$

As C₂>C₁

R₅ lies in C₁

Changing centroid:

Taking mean of R₅ & C₁

$$= \frac{182+182}{2}, \frac{72+70}{2}$$

$$= 182, 71$$

$$\text{New } C_1 = (182, 71)$$

For finding in which Cluster thus, R₆ belongs:

$$R_6 = (188, 77)$$

$$C_1 = (182, 71)$$

$$C_2 = (169, 58)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(188 - 182)^2 + (77 - 71)^2}$$

$$C_1 = 8.48$$

$$C_2 = \sqrt{(188 - 169)^2 + (77 - 58)^2}$$

$$C_2 = 26.87$$

As $C_2 > C_1$

R_6 lies in C_1

Changing centroid:

Taking mean of R_6 & C_1

$$= \frac{188+182}{2}, \frac{71+77}{2}$$

$$= 185, 74$$

$$\text{New } C_1 = (185, 74)$$

For finding in which Cluster thus, R_7 belongs:

$$R_7 = (180, 71)$$

$$C_1 = (185, 74)$$

$$C_2 = (169, 58)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(180 - 185)^2 + (71 - 74)^2}$$

$$C_1 = 5.83$$

$$C_2 = \sqrt{(180 - 169)^2 + (71 - 58)^2}$$

$$C_2 = 17.029$$

As $C_2 > C_1$

R_7 lies in C_1

Changing centroid:

Taking mean of R_7 & C_1

$$= \frac{180+185}{2}, \frac{71+74}{2}$$

$$= 182.5, 72.5$$

$$\text{New } C_1 = (182.5, 72.5)$$

For finding in which Cluster thus, R_8 belongs:

$$R_8 = (180, 70)$$

$$C_1 = (182.5, 72.5)$$

$$C_2 = (169, 58)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(180 - 182.5)^2 + (70 - 72.5)^2}$$

$$C_1 = 3.53$$

$$C_2 = \sqrt{(180 - 169)^2 + (70 - 58)^2}$$

$$C_2 = 16.28$$

As $C_2 > C_1$

R_8 lies in C_1

Changing centroid:

Taking mean of R_8 & C_1

$$= \frac{180+182.5}{2}, \frac{70+72.5}{2}$$

$$= 181.25, 71.25$$

$$\text{New } C_1 = (181.25, 71.25)$$

For finding in which Cluster thus, R_9 belongs:

$$R_9 = (183, 84)$$

$$C_1 = (181.25, 71.25)$$

$$C_2 = (169, 58)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(183 - 181.25)^2 + (84 - 71.25)^2}$$

$$C_1 = 12.87$$

$$C_2 = \sqrt{(183 - 169)^2 + (84 - 58)^2}$$

$$C_2 = 29.53$$

As $C_2 > C_1$

R_9 lies in C_1

Changing centroid:

Taking mean of R_9 & C_1

$$= \frac{183+181.25}{2}, \frac{84+71.25}{2}$$

$$= 182.125, 77.62$$

$$\text{New } C_1 = (182.12, 77.62)$$

For finding in which Cluster thus, R₁₀ belongs:

$$R_{10} = (180, 88)$$

$$C_1 = (182.12, 77.62)$$

$$C_2 = (169, 58)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(180 - 182.12)^2 + (88 - 77.62)^2}$$

$$C_1 = 10.59$$

$$C_2 = \sqrt{(180 - 169)^2 + (88 - 58)^2}$$

$$C_2 = 31.95$$

As C₂>C₁

R₁₀ lies in C₁

Changing centroid:

Taking mean of R₁₀ & C₁

$$= \frac{180+182.125}{2}, \frac{88+77.62}{2}$$

$$= 181.0625, 82.81$$

New C₁=(181.06,82.81)

For finding in which Cluster thus, R₁₁ belongs:

$$R_{11} = (180, 67)$$

$$C_1 = (181.06, 82.81)$$

$$C_2 = (169, 58)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(181 - 181.06)^2 + (67 - 82.81)^2}$$

$$C_1 = 15.84$$

$$C_2 = \sqrt{(180 - 169)^2 + (67 - 58)^2}$$

$$C_2 = 14.21$$

As C₁>C₂

R₁₁ lies in C₂

Changing centroid:

Taking mean of R₁₁ & C₂

$$= \frac{180+169}{2}, \frac{67+58}{2}$$

$$= 174.5, 62.5$$

New C₂=(174.5,62.5)

For finding in which Cluster thus, R₁₂ belongs:

$$R_{12}=(177,76)$$

$$C_1=(181.06,82.81)$$

$$C_2=(174.5,62.5)$$

Putting in Euclidean formula:

$$C_1 = \sqrt{(177 - 181.06)^2 + (76 - 82.81)^2}$$

$$C_1 = 7.92$$

$$C_2 = \sqrt{(177 - 174.5)^2 + (76 - 62.5)^2}$$

$$C_2 = 13.72$$

As C₂>C₁

R₁₂ lies in C₁

R₁	185	72	C₂
R₂	170	56	C₁
R₃	168	60	C₁
R₄	179	68	C₁
R₅	182	72	C₁
R₆	188	77	C₁
R₇	180	71	C₁
R₈	180	70	C₁
R₉	183	84	C₁
R₁₀	180	88	C₁
R₁₁	180	67	C₂
R₁₂	177	76	C₁

Implementation on python:

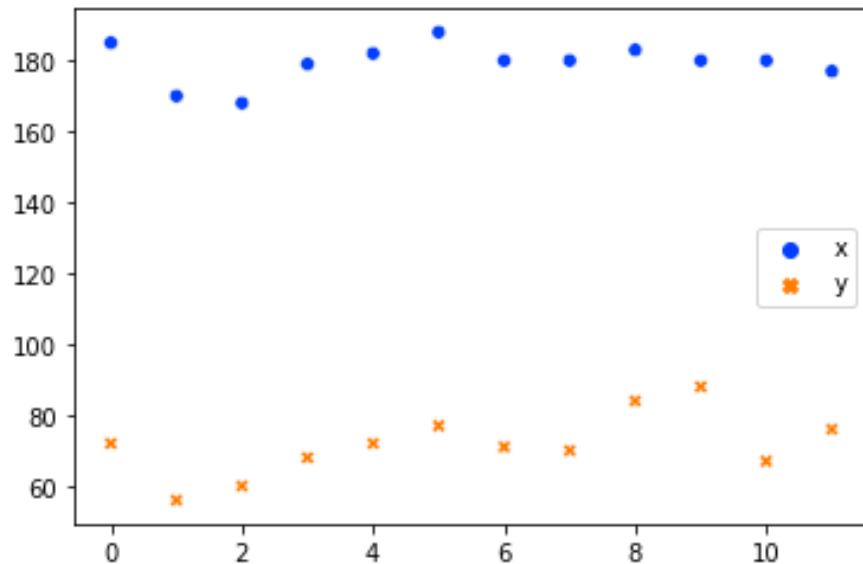
Import Libraries

```
▶ import numpy as np  
    import pandas as pd  
    import seaborn as sns  
    import matplotlib.pyplot as plt
```

Creating the Dataset

```
▶ Dict={ 'x':[185,170,168,179,182,188,180,180,183,180,180,177],  
        'y':[72,56,60,68,72,77,71,70,84,88,67,76]}  
  
▶ Dict  
  
4]: { 'x': [185, 170, 168, 179, 182, 188, 180, 180, 183, 180, 180, 177],  
      'y': [72, 56, 60, 68, 72, 77, 71, 70, 84, 88, 67, 76]}  
  
▶ df=pd.DataFrame(Dict)  
  
▶ df  
  
6]:  
     x   y  
 0  185  72  
 1  170  56  
 2  168  60  
 3  179  68  
 4  182  72  
 5  188  77  
 6  180  71  
 7  180  70  
 8  183  84  
 9  180  88  
10  180  67  
11  177  76
```

```
▶ sns.scatterplot(data=df,palette="bright")
plt.show()
```



For Euclidean Distance

```
▶ n=2 #no.of clusters
xc1=df['x'][0]
xc2=df['x'][1]
yc1=df['y'][0]
yc2=df['y'][1]
```

```
▶ print("Centroid 1 = ",xc1," , ",yc1)
print("Centroid 2 = ",xc2," , ",yc2)
```

Centroid 1 = 185 , 72
Centroid 2 = 170 , 56

```

: ► lis_xcl1=[]
lis_xcl2=[]
lis_ycl1=[]
lis_ycl2=[]
for i in range(2,12):
    x0=df['x'][i]
    y0=df['y'][i]
    euc_dis_c1=((xc1-x0)**2)+((yc1-y0)**2)**(1/2)
    euc_dis_c2=((xc2-x0)**2)+((yc2-y0)**2)**(1/2)
    print("For R",i+1," ( ",x0," , ",y0," ) = \n\n")
    print("Euclidean Distance for C1 = ",round(euc_dis_c1,2))
    print("Euclidean Distance for C2 = ",round(euc_dis_c2,2))
    if(euc_dis_c1>euc_dis_c2):
        print("\n Lies in C2\n")
        lis_xcl1.append(x0)
        lis_ycl1.append(y0)
        xc2=(x0+xc2)/2
        yc2=(y0+yc2)/2
    else:
        print("\n Lies in C1\n")
        lis_xcl2.append(x0)
        lis_ycl2.append(y0)
        xc1=(x0+xc1)/2
        yc1=(y0+yc1)/2
print("C1 = ",round(xc1,2)," , ",round(yc1,2))
print("C2 = ",round(xc2,2)," , ",round(yc2,2))
print("\n\n")

```

```
For R 3 ( 168 , 60 ) =  
  
Euclidean Distance for C1 = 20.81  
Euclidean Distance for C2 = 4.47  
  
Lies in C2  
  
C1 = 185 , 72  
C2 = 169.0 , 58.0  
  
  
For R 4 ( 179 , 68 ) =  
  
Euclidean Distance for C1 = 7.21  
Euclidean Distance for C2 = 14.14  
  
Lies in C1  
  
C1 = 182.0 , 70.0  
C2 = 169.0 , 58.0  
  
  
For R 5 ( 182 , 72 ) =  
  
Euclidean Distance for C1 = 2.0  
Euclidean Distance for C2 = 19.1  
  
Lies in C1  
  
C1 = 182.0 , 71.0  
C2 = 169.0 , 58.0
```

```
For R 6 ( 188 , 77 ) =  
  
Euclidean Distance for C1 = 8.49  
Euclidean Distance for C2 = 26.87  
  
Lies in C1  
  
C1 = 185.0 , 74.0  
C2 = 169.0 , 58.0  
  
  
For R 7 ( 180 , 71 ) =  
  
Euclidean Distance for C1 = 5.83  
Euclidean Distance for C2 = 17.03  
  
Lies in C1  
  
C1 = 182.5 , 72.5  
C2 = 169.0 , 58.0  
  
  
For R 8 ( 180 , 70 ) =  
  
Euclidean Distance for C1 = 3.54  
Euclidean Distance for C2 = 16.28  
  
Lies in C1  
  
C1 = 181.25 , 71.25  
C2 = 169.0 , 58.0
```

For R 9 (183 , 84) =

Euclidean Distance for C1 = 12.87
Euclidean Distance for C2 = 29.53

Lies in C1

C1 = 182.12 , 77.62
C2 = 169.0 , 58.0

For R 10 (180 , 88) =

Euclidean Distance for C1 = 10.59
Euclidean Distance for C2 = 31.95

Lies in C1

C1 = 181.06 , 82.81
C2 = 169.0 , 58.0

For R 11 (180 , 67) =

Euclidean Distance for C1 = 15.85
Euclidean Distance for C2 = 14.21

Lies in C2

C1 = 181.06 , 82.81
C2 = 174.5 , 62.5

For R 12 (177 , 76) =

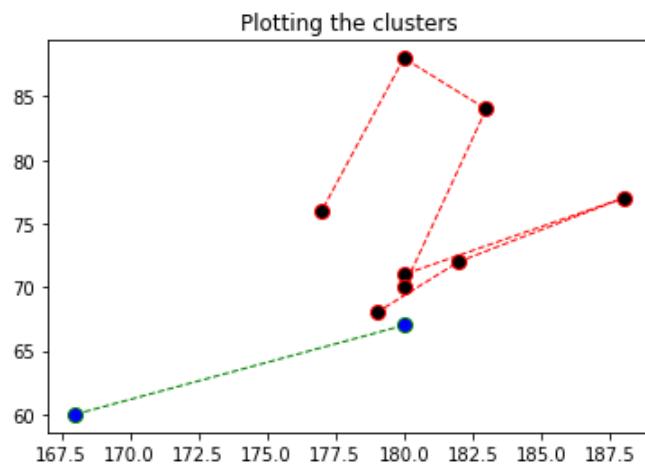
Euclidean Distance for C1 = 7.93
Euclidean Distance for C2 = 13.73

Lies in C1

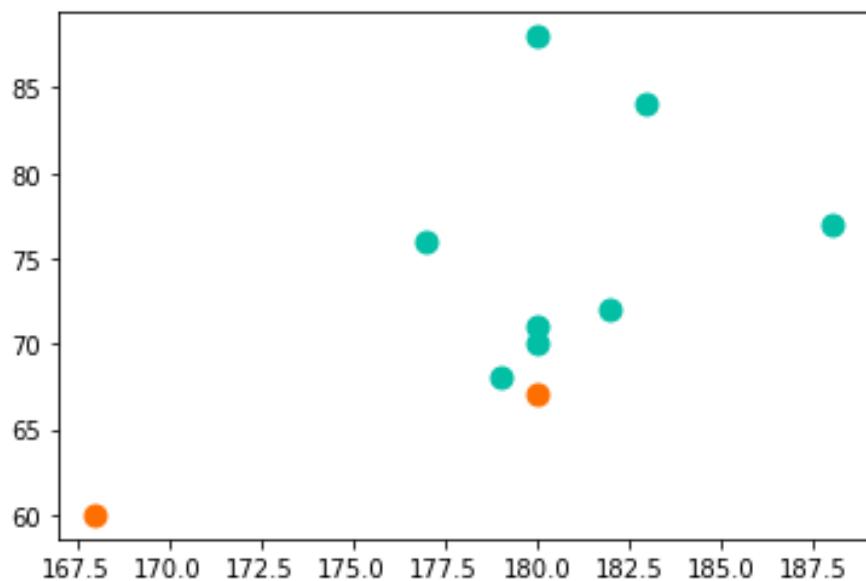
C1 = 179.03 , 79.41
C2 = 174.5 , 62.5

Plotting the results

```
▶ plt.plot(lis_xcl1,lis_ycl1, color='green', linestyle='dashed', linewidth = 1,  
          marker='o', markerfacecolor='blue', markersize=8)  
plt.plot(lis_xcl2,lis_ycl2, color='red', linestyle='dashed', linewidth = 1,  
          marker='o', markerfacecolor='black', markersize=8)  
plt.title('Plotting the clusters')  
plt.show()
```



```
▶ plt.scatter(lis_xcl1,lis_ycl1,color='#ff6f00',s=80)  
plt.scatter(lis_xcl2,lis_ycl2,color='#00bfa5',s=80)  
plt.show()
```



Implementation using Iris dataset:

Import Libraries

```
: └─▶ import numpy as np  
      import matplotlib.pyplot as plt  
      import pandas as pd
```

Create Dataset

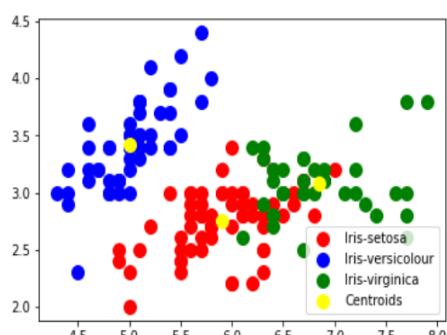
```
└─▶ dataset = pd.read_csv('Iris.csv')  
x = dataset.iloc[:, [1, 2, 3, 4]].values
```

Applying kmeans to the dataset / Creating the kmeans classifier

```
└─▶ kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)  
y_kmeans = kmeans.fit_predict(x)
```

Visualising the clusters

```
: └─▶ plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Iris-setosa')  
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Iris-versicolour')  
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-virginica')  
#Plotting the centroids of the clusters  
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'yellow', label = 'Centroids')  
plt.legend()  
  
t[5]: <matplotlib.legend.Legend at 0x24f9061c730>
```



OR

K Means

Import Libraries

```
[1]: ┆ from sklearn import datasets
      from sklearn.cluster import KMeans
      import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
```

Load Dataset

```
 ┆ iris = datasets.load_iris()
    print (iris.data)
    print (iris.target)

[[5.1 3.5 1.4 0.2]
 [4.9 3. 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5. 3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5. 3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3. 1.4 0.1]
 [4.3 3. 1.1 0.1]
 [5.8 4. 1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1. 0.2]]
```

Separating Features and labels

```
In [2]: x = pd.DataFrame(iris.data, columns=['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width'])  
y = pd.DataFrame(iris.target, columns=['Target'])
```

```
In [3]: x
```

[2]:

	Sepal Length	Sepal Width	Petal Length	Petal Width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
In [3]: y
```

[3]:

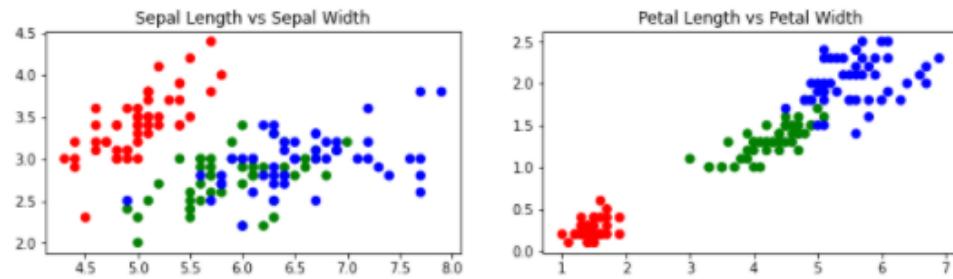
	Target
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	2
147	2
148	2
149	2

150 rows × 1 columns

Plotting

```
plt.figure(figsize=(12,3))
colors = np.array(['red', 'green', 'blue'])
plt.subplot(1, 2, 1)
plt.scatter(x['Sepal Length'], x['Sepal Width'], c=colors[y['Target']], s=40)
plt.title('Sepal Length vs Sepal Width')
plt.subplot(1,2,2)
plt.scatter(x['Petal Length'], x['Petal Width'], c= colors[y.Target], s=40)
plt.title('Petal Length vs Petal Width')

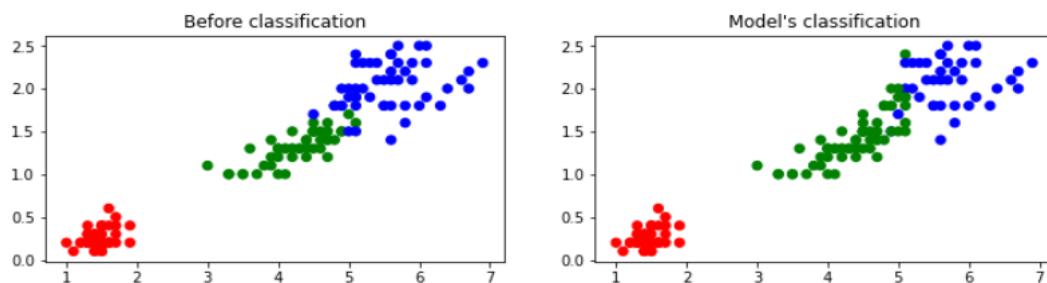
[7]: Text(0.5, 1.0, 'Petal Length vs Petal Width')
```



Algorithm

Plotting Result

```
plt.figure(figsize=(12,3))
colormap = np.array(['red', 'green', 'blue'])
predictedY = np.choose(model.labels_, [1, 0, 2]).astype(np.int64)
plt.subplot(1, 2, 1)
plt.scatter(x['Petal Length'], x['Petal Width'], c=colormap[y['Target']], s=40)
plt.title('Before classification')
plt.subplot(1, 2, 2)
plt.scatter(x['Petal Length'], x['Petal Width'], c=colormap[predictedY], s=40)
plt.title("Model's classification")
```



Question:

DBSCAN

Solution:

Cluster Analysis comprises of many different methods, of which one is the Density-based Clustering Method. DBSCAN stands for **Density-Based Spatial Clustering of Applications with Noise**. For a given set of data points, the DBSCAN algorithm clusters together those points that are close to each other based on any distance metric and a minimum number of points.

DBSCAN works on the idea that clusters are dense groups of points. These dense groups of data points are separated by low-density regions. The real-world data contains outliers and noise. It can also have arbitrary shapes (as shown in figures below), due to which the commonly used clustering algorithms (like k-means) fail to perform properly. For such arbitrary shaped clusters or data containing noise, the density-based algorithms such as DBSCAN are more efficient.



In the above figures, data points are arbitrarily shaped. Density-based clustering algorithms are used to find the high-density regions and group them into clusters

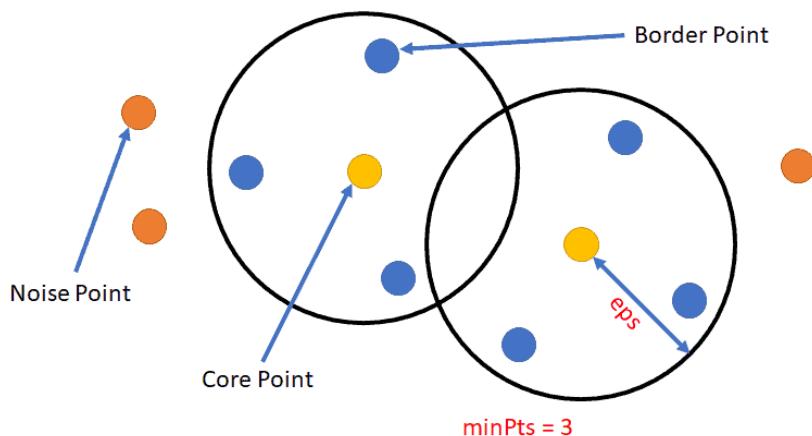
Parameters and Algorithm

DBSCAN Clustering Algorithm requires two parameters:

- **minPts:** The minimum number of data points required for a cluster to form a dense region.
- **eps:** Distance measure that is used to specify the neighborhood of any data point. If the distance between two data points is less than eps, then those two data points are neighbors.

Hence, after the DBSCAN algorithm, based on these two parameters, there are three types of points:

- Core Points:** A data point is said to be a core point if there are at least minPts number of data points within the eps radius.
- Border Points:** A data point is said to be a border point that is in the neighborhood of a core point but has fewer than MinPts within the eps radius.
- Noise/Outlier Points:** A data point is said to be a noise/outlier point if it is neither a core point nor is in the neighborhood of a core point.



Algorithm:

- Firstly, the parameters minPts and eps are defined
- Arbitrarily pick a starting point from the data, its neighborhood is defined within eps distance and if there are at least minPts number of points within this neighborhood, then this data point is classified as a core point.
- The algorithm continues this for all points. It finds the neighboring data points within eps and identifies the core points. For each core point, if that point is not assigned to a cluster, the algorithm creates a new cluster.
- The algorithm then finds all the neighboring points using the distance metrics; and if there are at least minPts points within eps for the considered data point, then all these points are part of the same cluster.
- The algorithm traverses through the unvisited points. If they are not part of any cluster, then marks them as outlier/noise.
- The algorithm terminates when all the data points are visited.
-

$$\epsilon = 1.5$$

$$\text{Minpts}=3$$

x	y
2	10
2	5
8	4
5	8
7	5
6	4
3	2
4	9

Solution:

	x	y
A ₀	2	10
A ₁	2	5
A ₂	8	4
A ₃	5	8
A ₄	7	5
A ₅	6	4
A ₆	3	2
A ₇	4	9

$$\text{Euclidean distance} : \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Calculating Euclidean distance for each

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0	5	8.48528	3.60555	7.07107	7.2111	8.06226	2.23607
A1	5	0	6.08276	4.24264	5	4.12311	3.16228	4.47214
A2	8.48528	6.08276	0	5	1.41421	2	5.38516	6.40312
A3	3.60555	4.24264	5	0	3.60555	4.12311	7.81025	1.41421
A4	7.07107	5	1.41421	3.60555	0	1.41421	5	5
A5	7.2111	4.12311	2	4.12311	1.41421	0	3.60555	5.38516
A6	8.06226	3.16228	5.38516	7.81025	5	3.60555	0	7.07107
A7	2.23607	4.47214	6.40312	1.41421	5	5.38516	7.07107	0

Checking values with following conditions:

$\epsilon = 1.5$

Minpts=3

Firstly points<1.5

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0	5	8.48528	3.60555	7.07107	7.2111	8.06226	2.23607
A1	5	0	6.08276	4.24264	5	4.12311	3.16228	4.47214
A2	8.48528	6.08276	0	5	1.41421	2	5.38516	6.40312
A3	3.60555	4.24264	5	0	3.60555	4.12311	7.81025	1.41421
A4	7.07107	5	1.41421	3.60555	0	1.41421	5	5
A5	7.2111	4.12311	2	4.12311	1.41421	0	3.60555	5.38516
A6	8.06226	3.16228	5.38516	7.81025	5	3.60555	0	7.07107
A7	2.23607	4.47214	6.40312	1.41421	5	5.38516	7.07107	0

Now finding the one where minpts>=3

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0	5	8.48528	3.60555	7.07107	7.2111	8.06226	2.23607
A1	5	0	6.08276	4.24264	5	4.12311	3.16228	4.47214
A2	8.48528	6.08276	0	5	1.41421	2	5.38516	6.40312
A3	3.60555	4.24264	5	0	3.60555	4.12311	7.81025	1.41421
A4	7.07107	5	1.41421	3.60555	0	1.41421	5	5
A5	7.2111	4.12311	2	4.12311	1.41421	0	3.60555	5.38516
A6	8.06226	3.16228	5.38516	7.81025	5	3.60555	0	7.07107
A7	2.23607	4.47214	6.40312	1.41421	5	5.38516	7.07107	0

So the core point in the given dataset is A₄

$$A_4(A_2, A_4, A_5) = A_4(1.41, 0, 1.41)$$

A ₀	Noise
A ₁	Noise
A ₂	Border Point
A ₃	Noise
A ₄	Core point
A ₅	Border Point
A ₆	Noise
A ₇	Noise

Implementation on python:

DBSCAN

Import Libraries

```
▶ from sklearn.datasets import make_blobs  
from sklearn.preprocessing import StandardScaler
```

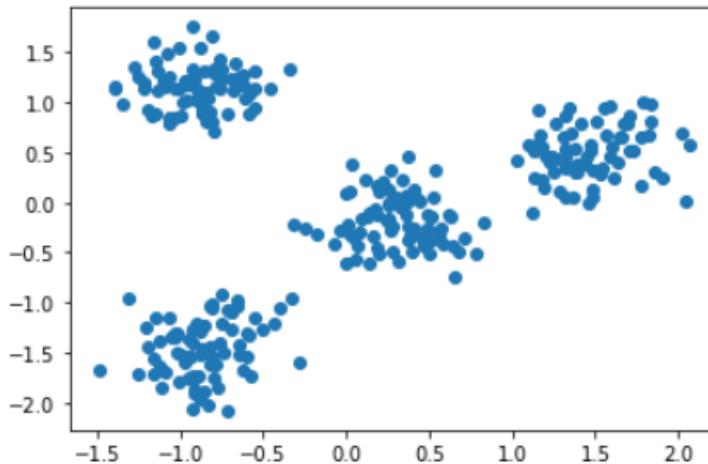
Create the dataset

```
▶ centers = [[2,1], [0,0], [-2,2], [-2,-2]]  
x, y = make_blobs(n_samples=300, centers=centers, cluster_std=0.4)
```

Normalization of the values

```
▶ x = StandardScaler().fit_transform(x)
```

```
▶ import matplotlib.pyplot as plt  
plt.scatter(x[:,0], x[:,1])  
plt.show()
```



Algorithm

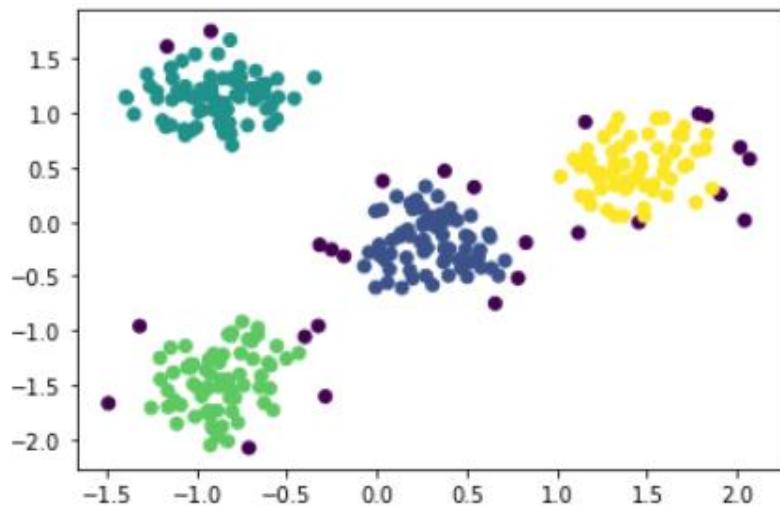
```
▶ from sklearn.cluster import DBSCAN  
db = DBSCAN(eps=0.3, min_samples=25)  
db.fit(x)
```

```
9]: DBSCAN(eps=0.3, min_samples=25)
```

```
▶ y2 = db.fit_predict(x)
```

```
1]: array([ 0,  1,  0,  1,  1,  0,  2,  0,  2,  1,  1,  3,  1,  0,  2,  2, -1,  
         2,  0, -1,  0,  3,  1,  2,  0,  3,  2,  0,  0, -1,  3, -1,  1, -1,  
         1,  3,  2,  0,  2, -1,  1,  0,  0,  3,  0,  3,  2,  0,  3,  1,  0,  
         0, -1,  0,  1,  3,  0,  3,  2,  2,  1,  0,  3,  0,  2,  2,  2,  2,  3,  
         2,  3,  1,  1, -1,  3,  1,  3,  0,  1,  1,  3,  3,  2,  3,  3,  0,  
        -1,  2,  2, -1,  3,  2,  1,  3,  0,  1,  3,  3,  3,  0,  0,  3,  2,  
         2,  0,  0,  1,  3,  1,  1,  0,  2,  2,  3,  3,  1,  2, -1, -1, -1,  
         2,  1,  3,  0,  3,  1,  0,  2,  1,  0,  1,  1,  2,  1,  1,  1,  2,  
         1,  2,  0,  2,  3,  1,  0,  3, -1,  3,  0,  0,  2,  3,  0,  1,  1,  
         3,  0,  0,  0,  1, -1,  0,  3,  3,  1,  1,  1,  3,  3,  0,  2,  0,  
         2,  3,  2, -1,  1,  3,  1,  0,  2, -1,  3,  1,  0,  1,  0,  0,  0,  0,  
         1,  2,  1,  3,  3,  1,  2,  1, -1,  0,  1,  0,  0,  1,  2,  1,  1,  
         0,  3,  0,  2,  0,  3,  3,  1,  1,  1,  3, -1,  2, -1,  2,  2,  1,  
         2,  1,  3,  3,  2,  1,  1,  2,  3,  2,  2, -1, -1,  0,  2,  1,  3,  
         1,  0,  3,  2,  2,  1,  2,  3,  2, -1,  2,  0,  2,  3,  0,  2,  2, -1,  
         0,  1,  0,  1,  3,  2,  2,  2,  0,  2,  0,  0,  2,  2,  2,  2, -1,  3,  
        -1,  3,  3,  3,  3,  1,  1,  1,  3,  3,  2,  1,  2,  1,  0,  2,  3,  
         0,  0,  1,  2,  3,  2,  2,  1,  1,  1,  3], dtype=int64)
```

```
▶ plt.scatter(x[:,0], x[:,1], c=y2)
plt.show()
```



LAB 12

(CLO 3,4, PLO 5-10,P3-A4)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	Computer Vision
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice basic Image Processing Techniques using OPENCV library functions

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Open CV

Question no. 1:

Image Processing-using OpenCV (Fundamentals)

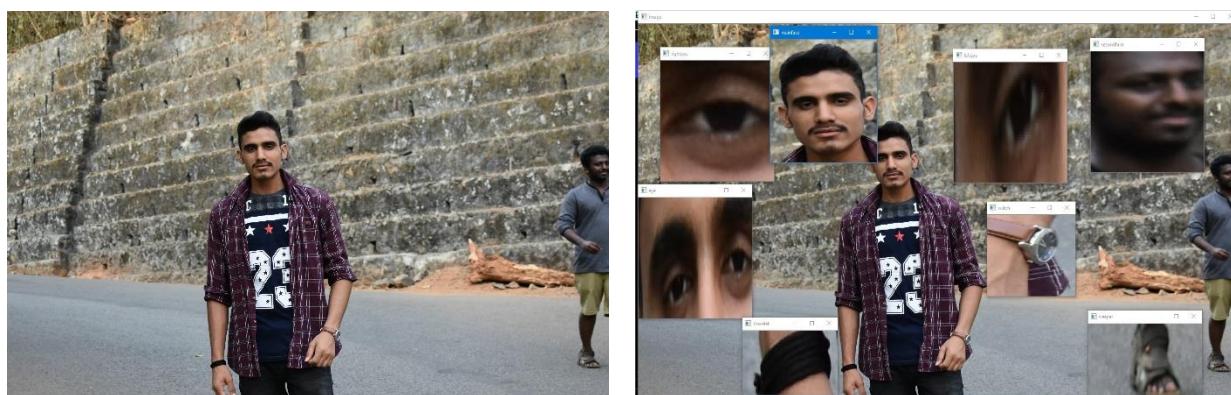
Computer Vision:

Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

OpenCV:

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.



Look at the following images

from the above original image, lots of pieces of information that are present in the original image can be obtained. Like in the above image there are two faces available and the person(I) in the images wearing a bracelet, watch, etc so by the help of OpenCV we can get all these types of information from the original image.

It's the basic introduction to OpenCV we can continue the Applications and all the things in our upcoming articles .

Applications of OpenCV:

There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition
- OpenCV Functionality
- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

Image-Processing:

Image processing is a method to perform some operations on an image, in order to get an enhanced image and/or to extract some useful information from it. If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”.

Digital-Image

An image may be defined as a two-dimensional function $f(x, y)$, where x and y are spatial(plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function $f(x, y)$ at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image. Image processing basically includes the following three steps:

- Importing the image
- Analysing and manipulating the image
- Output in which result can be altered image or report that is based on image analysis

How Does A Computer Read An Image?

We are humans we can easily make it out that is the image of a person who is me. But if we ask computer “is it my photo?”. The computer can’t say anything because the computer is not figuring out it all on its own.

The computer reads any image as a range of values between 0 and 255. For any color image, there are 3 primary channels -red, green and blue.

What is open CV as per it's offical website:

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.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding [18 million](#). The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, [Android](#) and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured [CUDA](#) and [OpenCL](#) interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

Question no. 2:

Lab implementation

Open CV

```
: M import cv2

Reading an Image

: M img=cv2.imread("D:\\7th semester\\AT in CE\\lab\\practice\\cv.png")
: M print(img)

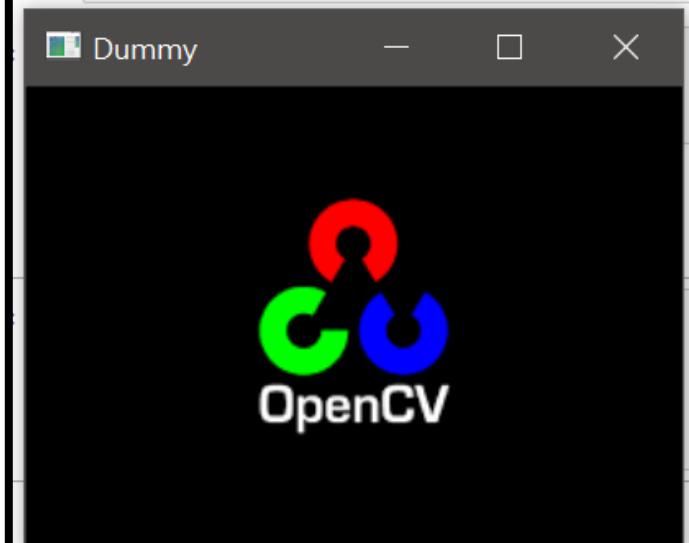
[[[0 0 0]
 [0 0 0]
 [0 0 0]
 ...
 [0 0 0]
 [0 0 0]
 [0 0 0]]

 [[0 0 0]
 [0 0 0]
 [0 0 0]
 ...
 [0 0 0]
 [0 0 0]
 [0 0 0]]

 [[0 0 0]
 [0 0 0]
 [0 0 0]]]
```

Displaying the Image

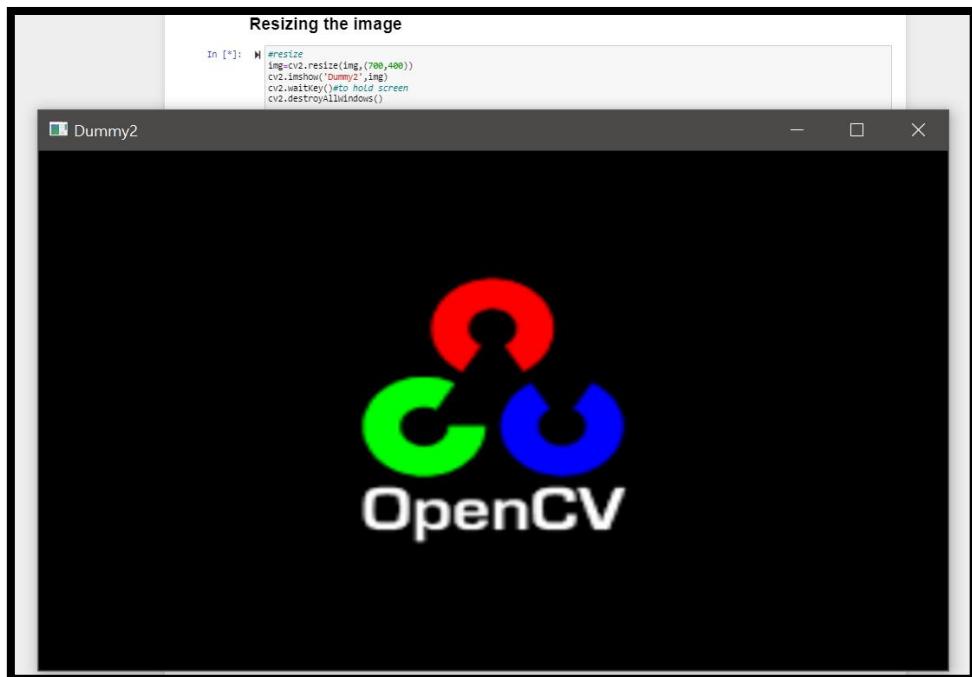
```
M cv2.imshow('Dummy',img)
cv2.waitKey()#to hold screen
cv2.destroyAllWindows()
```



```
▶ # doesn't wait shuts as soon as opens
cv2.imshow('Dummy1',img)
cv2.destroyAllWindows()
```

Resizing the image

```
▶ #resize
img=cv2.resize(img,(700,400))
cv2.imshow('Dummy2',img)
cv2.waitKey()#to hold screen
cv2.destroyAllWindows()
```



Take path as input from user

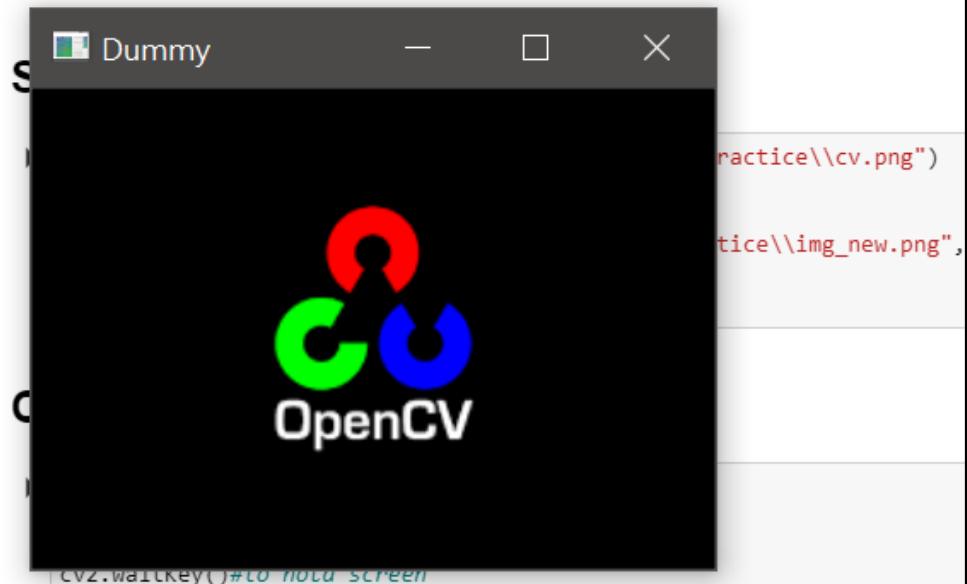
```
#user input
path=input('Enter path = ');
img=cv2.imread(path)
cv2.imshow('Dummy',img)
cv2.waitKey()#to hold screen
cv2.destroyAllWindows()
```

Enter path = D:\\7th semester\\AT in CE\\lab\\practice\\cv.png

Take path as input from user

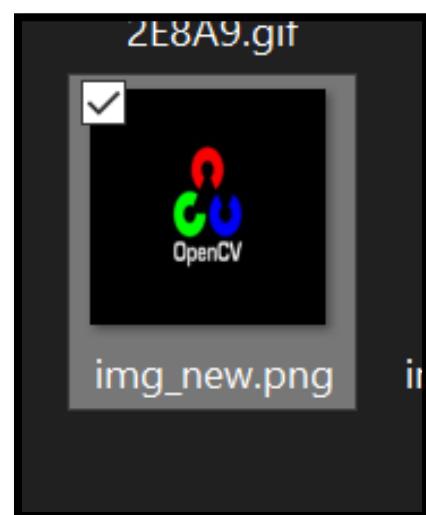
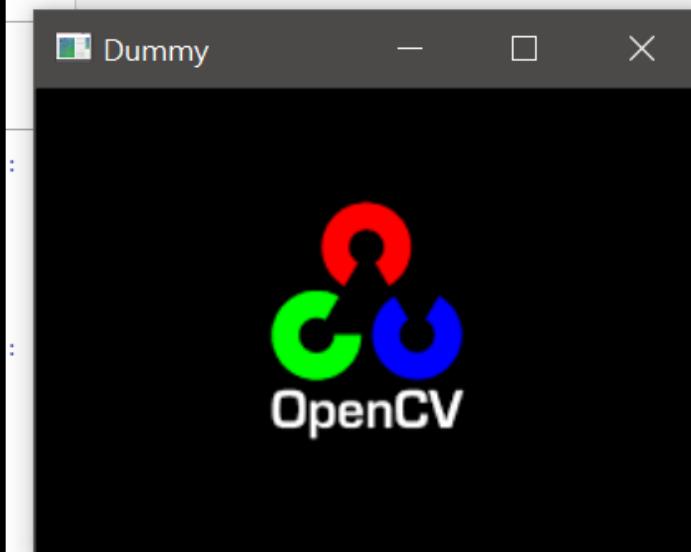
```
#user input
path=input('Enter path = ');
img=cv2.imread(path)
cv2.imshow('Dummy',img)
cv2.waitKey()#to hold screen
cv2.destroyAllWindows()
```

Enter path = D:\\7th semester\\AT in CE\\lab\\practice\\cv.png



Save a Resized image in memory

```
► img=cv2.imread("D:\\7th semester\\AT in CE\\lab\\practice\\cv.png")
  img2=cv2.resize(img,(500,500))
  cv2.imshow('Dummy',img)
  cv2.imwrite("D:\\7th semester\\AT in CE\\lab\\practice\\img_new.png",img2)
  cv2.waitKey()#to hold screen
  cv2.destroyAllWindows()
```



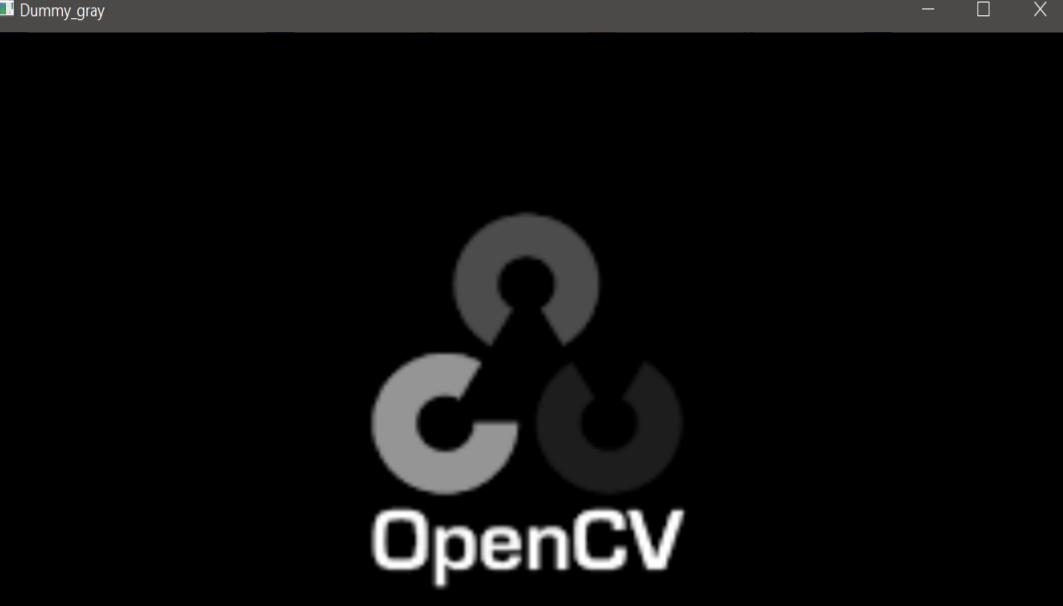
Convert to gray Scale

```
# convert to gray scale  
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
cv2.imshow('Dummy',gray)  
cv2.waitKey()#to hold screen  
cv2.destroyAllWindows()
```



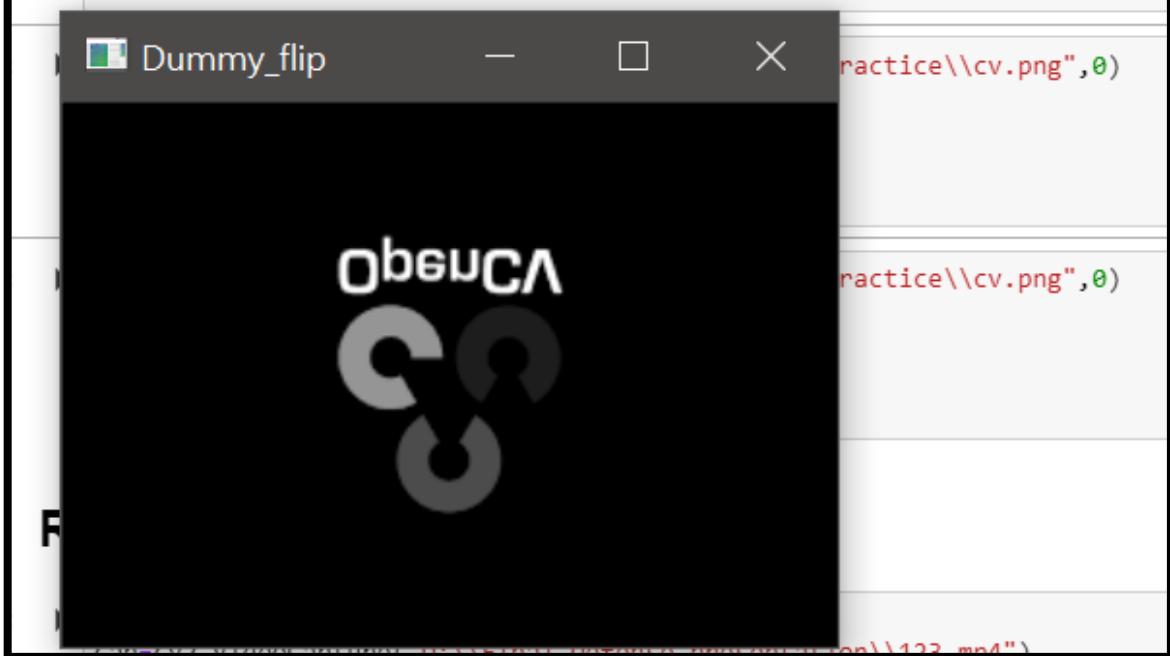
```
img=cv2.imread("D:\\7th semester\\AT in CE\\Lab\\prac
```

```
In [*]: ┆ img=cv2.imread("D:\\7th semester\\AT in CE\\lab\\practice\\cv.png",0)
          img3=cv2.resize(img,(900,500))
          cv2.imshow('Dummy_gray',img3)
          cv2.imwrite("D:\\7th semester\\AT in CE\\lab\\practice\\img_new1.png",img3)
          cv2.waitKey()#to hold screen
          cv2.destroyAllWindows()
```

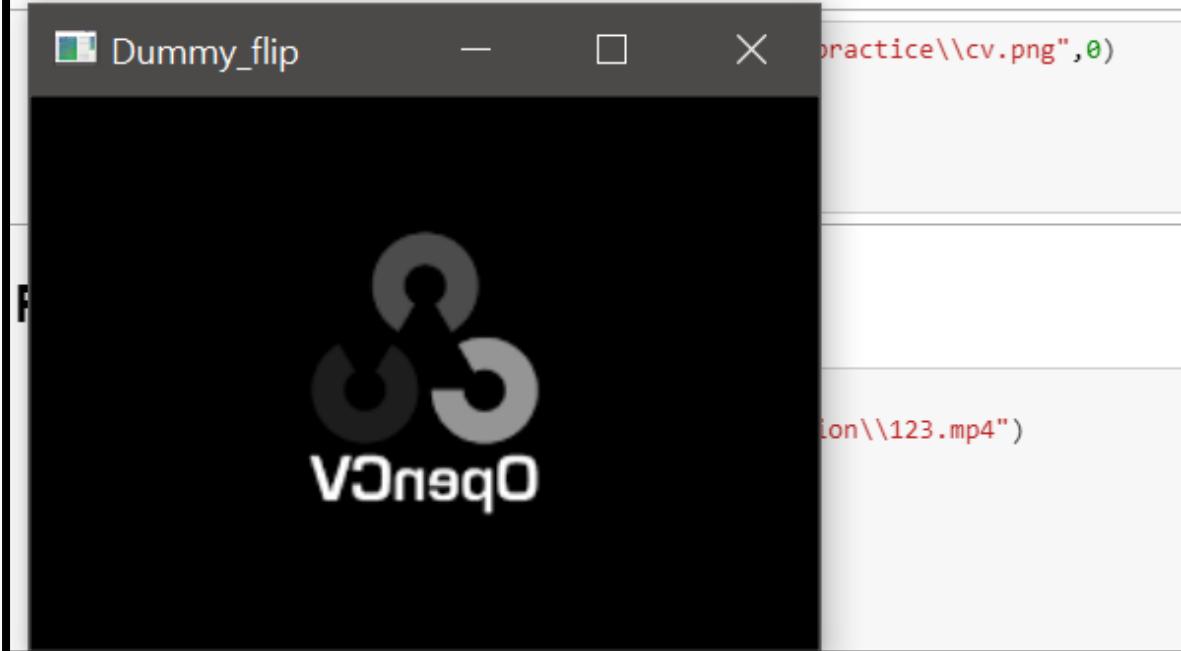


Flip image

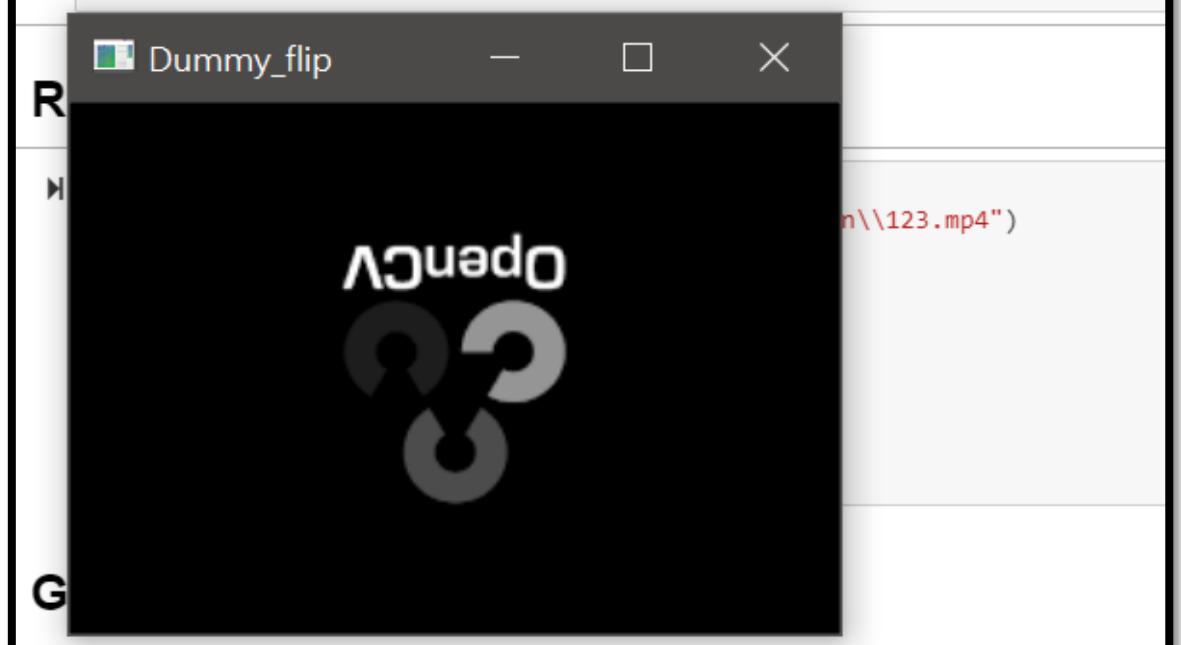
```
 ┆ img=cv2.imread("D:\\7th semester\\AT in CE\\lab\\practice\\cv.png",0)
    img1=cv2.flip(img,0) #upside down
    cv2.imshow('Dummy_flip',img1)
    cv2.waitKey()#to hold screen
    cv2.destroyAllWindows()
```



```
▶ img=cv2.imread("D:\\7th semester\\AT in CE\\lab\\practice\\cv.png",0)
  img1=cv2.flip(img,1) # mirror
  cv2.imshow('Dummy_flip',img1)
  cv2.waitKey()#to hold screen
  cv2.destroyAllWindows()
```



```
▶ img=cv2.imread("D:\\7th semester\\AT in CE\\lab\\practice\\cv.png",0)
  img1=cv2.flip(img,-1) # upside down and inverted
  cv2.imshow('Dummy_flip',img1)
  cv2.waitKey()#to hold screen
  cv2.destroyAllWindows()
```



Read a video

```
► import cv2
cap=cv2.VideoCapture("D:\\Final Defense presentation\\123.mp4")
while True:
    ret,f=cap.read()
    cv2.imshow("Video",f)
    k=cv2.waitKey(25)
    if k==ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```



Gray video

```
▶ import cv2
cap=cv2.VideoCapture("D:\\Final Defense presentation\\123.mp4")
while True:
    ret,f=cap.read()
    img1 = cv2.cvtColor(f, cv2.COLOR_BGR2GRAY)
    cv2.imshow("Video",img1)
    k=cv2.waitKey(25)
    if k==ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```



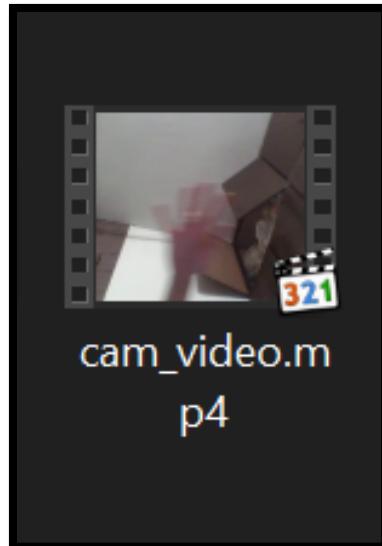
Question no. 3:

Python code to record video using web cam and save in your computer

Record video using web cam and save in your computer

```
▶ import cv2

▶ vid_capture = cv2.VideoCapture(0)
vid_cod = cv2.VideoWriter_fourcc(*'XVID')
#saving in memory
output= cv2.VideoWriter("D:\\7th semester\\AT in CE\\lab\\practice\\cam_video.mp4", vid_cod, 20.0, (640,480))
while(True):
    # Capture each frame of webcam video
    ret,frame = vid_capture.read()
    cv2.imshow("My cam video", frame)
    output.write(frame)
    if cv2.waitKey(1) &0xFF == ord('x'):
        break
vid_capture.release()
output.release()
cv2.destroyAllWindows()
```

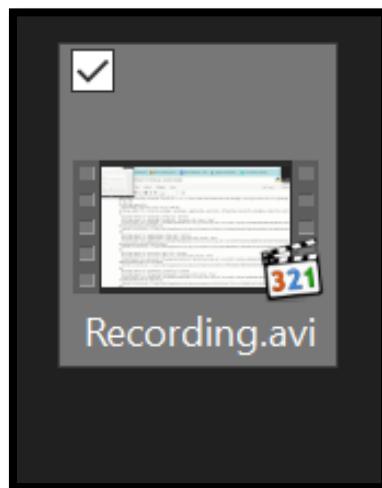


Question no. 4:

Python code for screen recorder save in your computer

Python code for screen recorder save in your computer

```
# import pyautogui
import cv2
import numpy as np
resolution = (1920, 1080)
# Specify video codec
codec = cv2.VideoWriter_fourcc(*"XVID")
# Specify name of Output file path and name
filename = "D:\\7th semester\\AT in CE\\lab\\practice\\Recording.avi"
fps = 60.0
# Creating a VideoWriter object
out = cv2.VideoWriter(filename, codec, fps, resolution)
# Create an Empty window
cv2.namedWindow("Live", cv2.WINDOW_NORMAL)
# Resize
cv2.resizeWindow("Live", 480, 270)
while True:
    # Take screenshot using PyAutoGUI
    img = pyautogui.screenshot()
    # Convert the screenshot to a numpy array
    frame = np.array(img)
    # Convert it from BGR(Blue, Green, Red) to
    # RGB(Red, Green, Blue)
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # Write it to the output file
    out.write(frame)
    # Optional: Display the recording screen
    cv2.imshow('Live', frame)
    # Stop recording when we press 'q'
    if cv2.waitKey(1) == ord('q'):
        break
out.release()
cv2.destroyAllWindows()
```



LAB 13

(CLO 1-3, PLO 3-5,P3-P4)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	Naïve Baye's Algorithm
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Make and implement Naïve Baye's algorithm

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

Naive Bayes

Question no. 3:

Implement Naive Bayes algorithm (Gaussian, Bernoulli etc) on Weather Dataset

Gaussian Naïve Bayes:

Naive Bayes Algorithm(Gaussian)

```
▶ import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,scale,StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import seaborn as sns
import numpy as np
from sklearn.naive_bayes import GaussianNB
```

Load Dataset

```
▶ df=pd.read_csv('testset.csv',encoding='unicode_escape')
df.describe()
```

[2]:

	datetime_utc	_conds	_dewptm	_fog	_hail	_heatindexm	_hum	_precipm	_pressurem	_rain	_snow	_tempm	_thunder	_tornado	_vism	_wdi
0	19961101-11:00	Smoke	9.0	0	0	NaN	27.0	NaN	1010.0	0	0	30.0	0	0	5.0	280
1	19961101-12:00	Smoke	10.0	0	0	NaN	32.0	NaN	-9999.0	0	0	28.0	0	0	NaN	280
2	19961101-13:00	Smoke	11.0	0	0	NaN	44.0	NaN	-9999.0	0	0	24.0	0	0	NaN	280
3	19961101-14:00	Smoke	10.0	0	0	NaN	41.0	NaN	1010.0	0	0	24.0	0	0	2.0	280
4	19961101-16:00	Smoke	11.0	0	0	NaN	47.0	NaN	1011.0	0	0	23.0	0	0	1.2	280
...	
100985	20170424-06:00	Haze	17.0	0	0	NaN	25.0	NaN	1005.0	0	0	34.0	0	0	4.0	320
100986	20170424-09:00	Haze	14.0	0	0	NaN	16.0	NaN	1003.0	0	0	38.0	0	0	4.0	320
100987	20170424-12:00	Haze	12.0	0	0	NaN	14.0	NaN	1002.0	0	0	36.0	0	0	4.0	270
100988	20170424-15:00	Haze	15.0	0	0	NaN	27.0	NaN	1004.0	0	0	32.0	0	0	2.0	320
100989	20170424-18:00	Haze	15.0	0	0	NaN	30.0	NaN	1005.0	0	0	30.0	0	0	2.0	320

100990 rows × 20 columns

Checking for null values

```
3]: └── df.isnull().sum()  
out[3]: datetime_utc      0  
_conds          72  
_dewptm        621  
_fog            0  
_hail            0  
_heatindexm     71835  
_hum            757  
_precipm       100990  
_pressurem      232  
_rain            0  
_snow            0  
_tempm          673  
_thunder          0  
_tornado          0  
_vism           4428  
_wdird          14755  
_wdire          14755  
_wgustm         99918  
_windchillm    100411  
_wspdm          2358  
dtype: int64  
  
4]: └── df.isnull().values.any()  
out[4]: True
```

Filling the null values

```
└── df1=df.fillna(method = 'pad')  
  
└── df1.isnull().sum()  
out[6]: datetime_utc      0  
_conds          0  
_dewptm        0  
_fog            0  
_hail            0  
_heatindexm     47  
_hum            0  
_precipm       100990  
_pressurem      0  
_rain            0  
_snow            0  
_tempm          0  
_thunder          0  
_tornado          0  
_vism            0  
_wdird          0  
_wdire          0  
_wgustm         1665  
_windchillm    755  
_wspdm          0  
dtype: int64
```

Dropping Extra columns

```
: █ df1.drop(['datetime_utc','_precipm','_heatindexm','_snow','_tempm',
    '_thunder','_tornado','_vism','_wdird','_wdire','_wgustm',
    '_windchillm','_wspdm'], axis = 'columns',inplace=True)
```

```
: █ df1.dropna()
```

[8]:

	_conds	_dewptm	_fog	_hail	_hum	_pressurem	_rain
0	Smoke	9.0	0	0	27.0	1010.0	0
1	Smoke	10.0	0	0	32.0	-9999.0	0
2	Smoke	11.0	0	0	44.0	-9999.0	0
3	Smoke	10.0	0	0	41.0	1010.0	0
4	Smoke	11.0	0	0	47.0	1011.0	0
...
100985	Haze	17.0	0	0	25.0	1005.0	0
100986	Haze	14.0	0	0	16.0	1003.0	0
100987	Haze	12.0	0	0	14.0	1002.0	0
100988	Haze	15.0	0	0	27.0	1004.0	0
100989	Haze	15.0	0	0	30.0	1005.0	0

100990 rows × 7 columns

```
█ df1.head()
```

[1]:

	_conds	_dewptm	_fog	_hail	_hum	_pressurem	_rain
0	Smoke	9.0	0	0	27.0	1010.0	0
1	Smoke	10.0	0	0	32.0	-9999.0	0
2	Smoke	11.0	0	0	44.0	-9999.0	0
3	Smoke	10.0	0	0	41.0	1010.0	0
4	Smoke	11.0	0	0	47.0	1011.0	0

Categorical values

```
▶ df1.head()
lb=LabelEncoder()
lb.fit(df1['_conds'])
df1['_conds']=lb.transform(df1['_conds'])

▶ df1.head()
```

[11]:

	_conds	_dewptm	_fog	_hail	_hum	_pressurem	_rain
0	31	9.0	0	0	27.0	1010.0	0
1	31	10.0	0	0	32.0	-9999.0	0
2	31	11.0	0	0	44.0	-9999.0	0
3	31	10.0	0	0	41.0	1010.0	0
4	31	11.0	0	0	47.0	1011.0	0

Separating Features and labels

```
: ▶ x=df1.iloc[:,0:6]
x.head()
```

[12]:

	_conds	_dewptm	_fog	_hail	_hum	_pressurem
0	31	9.0	0	0	27.0	1010.0
1	31	10.0	0	0	32.0	-9999.0
2	31	11.0	0	0	44.0	-9999.0
3	31	10.0	0	0	41.0	1010.0
4	31	11.0	0	0	47.0	1011.0

```
: ▶ y=df1.iloc[:,-1]
y.head()
```

[13]:

```
0    0
1    0
2    0
3    0
4    0
```

Name: _rain, dtype: int64

Splitting test and training data

```
► xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
print("Xtrain = ",xtrain.shape)
print("Xtest = " ,xtest.shape)
print("Ytrain = " ,ytrain.shape)
print("Ytest = " ,ytest.shape)

Xtrain =  (80792, 6)
Xtest =  (20198, 6)
Ytrain =  (80792,)
Ytest =  (20198,)
```

Gaussian Naive Bayes

```
► Model=GaussianNB()
Model.fit(xtrain,ytrain)
print('Accuracy score = ',Model.score(xtrain,ytrain))

Accuracy score =  0.24800722843845927
```

Predictions

```
: ► ypred=Model.predict(xtest)
print("Predictions = \n",ypred)

Predictions =
[1 1 0 ... 1 1 1]
```

Print accuracy and score of the model

```
▶ print("Accuracy = \n",accuracy_score(ytest,ypred))
```

```
Accuracy =  
0.24556886820477275
```

Bernoulli Naïve Bayes:

Naive Bayes Algorithm(Bernoulli)

```
: ▶ import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import LabelEncoder,scale,StandardScaler  
from sklearn.metrics import accuracy_score  
from sklearn.model_selection import train_test_split  
import seaborn as sns  
import numpy as np  
from sklearn.naive_bayes import BernoulliNB
```

Load Dataset

```
▶ df=pd.read_csv('testset.csv',encoding='unicode_escape')  
df.describe()
```

```
[2]:
```

	datetime_utc	_conds	_dewptm	_fog	_hail	_heatindexm	_hum	_precipm	_pressurem	_rain	_snow	_tempm	_thunder	_tornado	_vism	_wdi
0	19961101-11:00	Smoke	9.0	0	0	NaN	27.0	NaN	1010.0	0	0	30.0	0	0	5.0	28C
1	19961101-12:00	Smoke	10.0	0	0	NaN	32.0	NaN	-9999.0	0	0	28.0	0	0	NaN	C
2	19961101-13:00	Smoke	11.0	0	0	NaN	44.0	NaN	-9999.0	0	0	24.0	0	0	NaN	C
3	19961101-14:00	Smoke	10.0	0	0	NaN	41.0	NaN	1010.0	0	0	24.0	0	0	2.0	C
4	19961101-16:00	Smoke	11.0	0	0	NaN	47.0	NaN	1011.0	0	0	23.0	0	0	1.2	C
...	
100985	20170424-06:00	Haze	17.0	0	0	NaN	25.0	NaN	1005.0	0	0	34.0	0	0	4.0	32C
100986	20170424-09:00	Haze	14.0	0	0	NaN	16.0	NaN	1003.0	0	0	38.0	0	0	4.0	32C
100987	20170424-12:00	Haze	12.0	0	0	NaN	14.0	NaN	1002.0	0	0	36.0	0	0	4.0	27C
100988	20170424-15:00	Haze	15.0	0	0	NaN	27.0	NaN	1004.0	0	0	32.0	0	0	2.0	32C
100989	20170424-18:00	Haze	15.0	0	0	NaN	30.0	NaN	1005.0	0	0	30.0	0	0	2.0	32C

100990 rows × 20 columns

Checking for null values

```
In[3]: df.isnull().sum()
```

```
Out[3]: datetime_utc      0
        _conds          72
        _dewptm         621
        _fog            0
        _hail           0
        _heatindexm     71835
        _hum            757
        _precipm        100990
        _pressurem      232
        _rain           0
        _snow            0
        _tempm          673
        _thunder          0
        _tornado          0
        _vism            4428
        _wdird           14755
        _wdire           14755
        _wgustm          99918
        _windchillm     100411
        _wspdm           2358
        dtype: int64
```

```
In[4]: df.isnull().values.any()
```

```
Out[4]: True
```

Filling the null values

```
In[5]: df1=df.fillna(method = 'pad')
```

```
In[6]: df1.isnull().sum()
```

```
Out[6]: datetime_utc      0
        _conds          0
        _dewptm         0
        _fog            0
        _hail           0
        _heatindexm     47
        _hum            0
        _precipm        100990
        _pressurem      0
        _rain           0
        _snow            0
        _tempm          0
        _thunder          0
        _tornado          0
        _vism            0
        _wdird           0
        _wdire           0
        _wgustm          1665
        _windchillm     755
        _wspdm           0
        dtype: int64
```

Dropping Extra columns

```
: df1.drop(['datetime_utc','_precipm','_heatindexm','_snow','_tempm',
           '_thunder','_tornado','_vism','_wdird','_wdire','_wgustm',
           '_windchillm','_wspdm'], axis = 'columns',inplace=True)
```

```
: ► df1.dropna()
```

:[8]:

	<u>conds</u>	<u>dewptm</u>	<u>fog</u>	<u>hail</u>	<u>hum</u>	<u>pressureum</u>	<u>rain</u>
0	Smoke	9.0	0	0	27.0	1010.0	0
1	Smoke	10.0	0	0	32.0	-9999.0	0
2	Smoke	11.0	0	0	44.0	-9999.0	0
3	Smoke	10.0	0	0	41.0	1010.0	0
4	Smoke	11.0	0	0	47.0	1011.0	0
...
100985	Haze	17.0	0	0	25.0	1005.0	0
100986	Haze	14.0	0	0	16.0	1003.0	0
100987	Haze	12.0	0	0	14.0	1002.0	0
100988	Haze	15.0	0	0	27.0	1004.0	0
100989	Haze	15.0	0	0	30.0	1005.0	0

100990 rows × 7 columns

```
▶ df1.head()
```

1:

	<u>conds</u>	<u>dewptm</u>	<u>fog</u>	<u>hail</u>	<u>hum</u>	<u>pressurem</u>	<u>rain</u>
0	Smoke	9.0	0	0	27.0	1010.0	0
1	Smoke	10.0	0	0	32.0	-9999.0	0
2	Smoke	11.0	0	0	44.0	-9999.0	0
3	Smoke	10.0	0	0	41.0	1010.0	0
4	Smoke	11.0	0	0	47.0	1011.0	0

Categorical values

```
df1.head()
lb=LabelEncoder()
lb.fit(df1['_conds'])
df1['_conds']=lb.transform(df1['_conds'])
```

```
df1.head()
```

[11]:

	_conds	_dewptm	_fog	_hail	_hum	_pressurem	_rain
0	31	9.0	0	0	27.0	1010.0	0
1	31	10.0	0	0	32.0	-9999.0	0
2	31	11.0	0	0	44.0	-9999.0	0
3	31	10.0	0	0	41.0	1010.0	0
4	31	11.0	0	0	47.0	1011.0	0

Separating Features and labels

```
: df1.iloc[:,0:6]
x=x.head()
```

[12]:

	_conds	_dewptm	_fog	_hail	_hum	_pressurem
0	31	9.0	0	0	27.0	1010.0
1	31	10.0	0	0	32.0	-9999.0
2	31	11.0	0	0	44.0	-9999.0
3	31	10.0	0	0	41.0	1010.0
4	31	11.0	0	0	47.0	1011.0

```
: df1.iloc[:,-1]
y=y.head()
```

[13]:

```
0    0
1    0
2    0
3    0
4    0
```

```
Name: _rain, dtype: int64
```

Splitting test and training data

```
► xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
  print("Xtrain = ",xtrain.shape)
  print("Xtest = " ,xtest.shape)
  print("Ytrain = " ,ytrain.shape)
  print("Ytest = " ,ytest.shape)

Xtrain =  (80792, 6)
Xtest =  (20198, 6)
Ytrain =  (80792,)
Ytest =  (20198,)
```

Bernoulli Naive Bayes

```
► Model=BernoulliNB()
  Model.fit(xtrain,ytrain)
  print('Accuracy score = ',Model.score(xtrain,ytrain))

Accuracy score =  0.9740320823843945
```

Predictions

```
► ypred=Model.predict(xtest)
  print("Predictions = \n",ypred)

Predictions =
 [0 0 0 ... 0 0 0]
```

Print accuracy and score of the model

```
▶ print("Accuracy = \n",accuracy_score(ytest,ypred))
```

```
Accuracy =
0.9725715417368056
```

LAB 05

(CLO 2-3-4, PLO 3-5-10,P3-P4-A4)

Items	Description
Course Title	Artificial Intelligence Lab
Lab Title	<i>Python Programming-II</i>
Duration	3 Hours
Tool	Anaconda (Jupyter Notebook)
Objective	Practice: Basics of python loops functions and lambda functions.

Rubrics:

Coding/logic	Response	Data processing & visualization	Accuracy	Outcome/Result
04	04	04	04	04

8 ML algorithm on diabetes dataset

Question: Pakistan Diagnosis Center(PDC) needs a Machine Learning based solution in order predict Diabetes of the patient in real time on the bases of different attributes e.g. blood pressure, glucose, BMI, age and so on. PDC assign you to design and develop a solution as a professional ML expert.

- As a professional ML expert you have a test and training dataset.
- You have to apply 6 to 8 ML model to check accuracy in results and you have to select the best model in this aspect.
- You have to design a GUI using flask and any other framework.

First we test and train the data set then find the accuracy of ML models

1) Linear regression

Linear Regression Algorithm

Collect Data

Importing Libraries and dataset

```
> import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
import math  
from sklearn.linear_model import LinearRegression  
  
data=pd.read_csv("diabetes.csv")  
data.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
5	5	116	74	0	0	25.6	0.20
6	3	78	50	32	88	31.0	0.24
7	10	115	0	0	0	35.3	0.13
8	2	197	70	45	543	30.5	0.15
9	8	125	96	0	0	0.0	0.23

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
x=df.drop(['Outcome'], axis=1)
x
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2
...
763	10	101	76	48	180	32.9	0
764	2	122	70	27	0	36.8	0
765	5	121	72	23	112	26.2	0
766	1	126	60	0	0	30.1	0
767	1	93	70	31	0	30.4	0

768 rows × 8 columns

```
y=df['Outcome']
y

0      1
1      0
2      1
3      0
4      1
 ..
763     0
764     0
765     0
766     1
767     0
Name: Outcome, Length: 768, dtype: int64
```

Train and Test Data

Build the model on train data and predict the output on test data

```
# from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_
```

Building Linear Regression Model

Let's create a Linear Regression Model using Scikit-learn.

```
# from sklearn import linear_model
model = linear_model.LinearRegression()
model.fit(x_train, y_train) # Training data is used always
y_pre = model.predict(x_test)
```

Accuracy Check

Calculate accuracy to check how accurate your values are.

```
# from sklearn.metrics import accuracy_score, r2_score
LinearRegression_Score = r2_score(y_test, y_pre)*100
print("accuracy = ", LinearRegression_Score)

accuracy = 31.50886303509085
```

2) Logistic regression

Logistic Regression Algorithm

Collect Data

Importing Libraries and dataset

```
▶ import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math
from sklearn.linear_model import LogisticRegression

data=pd.read_csv("diabetes.csv")
data.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
5	5	116	74	0	0	25.6	0.20
6	3	78	50	32	88	31.0	0.24
7	10	115	0	0	0	35.3	0.13
8	2	197	70	45	543	30.5	0.15
9	8	125	96	0	0	0.0	0.23

```
data.head().transpose()
```

	0	1	2	3	4
Pregnancies	6.000	1.000	8.000	1.000	0.000
Glucose	148.000	85.000	183.000	89.000	137.000
BloodPressure	72.000	66.000	64.000	66.000	40.000
SkinThickness	35.000	29.000	0.000	23.000	35.000
Insulin	0.000	0.000	0.000	94.000	168.000
BMI	33.600	26.600	23.300	28.100	43.100
DiabetesPedigreeFunction	0.627	0.351	0.672	0.167	2.288
Age	50.000	31.000	32.000	21.000	33.000
Outcome	1.000	0.000	1.000	0.000	1.000

```
data.describe ()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabet
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
X= data.drop( "Outcome", axis=1)
y= data["Outcome"]
```

Train and Test Data

Build the model on train data and predict the output on test data

```
# from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X, y,  
                                                test_size=0.3, random_state=1)
```

Building Logistic Regression Model

Let's create a Logistic Regression Model using Scikit-learn.

```
# classification_report(y_test,predictions)  
  
1]:      precision    recall   f1-score   support  
0    0.79      0.90      0.84     146\n      1      0.78      0.58  
0.66      85\n      accuracy           0.78      231\nmacro avg      0.78      0.74      0.75      231\nweighted avg      0.78  
0.78      0.78     231\n  
  
# X_train, X_test, y_train, y_test = train_test_split  
(X, y, test_size=0.3, random_state=1)  
from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test,predictions)  
  
2]: array([[132,  14],  
          [ 36,  49]], dtype=int64)
```

Accuracy Check

Calculate accuracy to check how accurate your values are.

```
# from sklearn.metrics import accuracy_score  
LogisticRegression_Score = accuracy_score(y_test,predictions)*100  
print("accuracy = ", LogisticRegression_Score)  
  
accuracy = 78.35497835497836
```

3) Decision tree

Decision Tree Algorithm

Collect Data

Let's first load the required libraries and load the required Diabetes dataset using pandas read_csv function.

```
❷ import pandas as pd  
df=pd.read_csv("diabetes.csv")  
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables). Our target variable is Outcome and other variables are feature variables.

```
: ❷ x=df.drop(['Outcome'], axis=1)  
x
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2
..
763	10	101	76	48	180	32.9	0
764	2	122	70	27	0	36.8	0
765	5	121	72	23	112	26.2	0
766	1	126	60	0	0	30.1	0
767	1	93	70	31	0	30.4	0

768 rows × 8 columns

```
y=df['Outcome']
y

0    1
1    0
2    1
3    0
4    1
..
763   0
764   0
765   0
766   1
767   0
Name: Outcome, Length: 768, dtype: int64
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. I have divided the train and test size in 1:4 ratio (test_size 0.2).

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split
(x, y, test_size=0.2, random_state=1)
```

Building Decision Tree Model

Let's create a Decision Tree Model using Scikit-learn.

```
# Create Decision Tree classifier object
model = DecisionTreeClassifier()

# Train Decision Tree Classifier
model = model.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = model.predict(x_test)
```

Accuracy Check

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

```
from sklearn import metrics
DTaccuracy = metrics.accuracy_score(y_test, y_pred)*100
print(" Accuracy:",DTaccuracy)
```

Accuracy: 70.12987012987013

4) SVM algorithm

Support Vector Machine (SVM)

Collect Data

Let's first load the required libraries and load the required Diabetes dataset using pandas `read_csv` function.

```
▶ import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import sklearn.preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
from sklearn import svm
```

```
df = pd.read_csv('diabetes.csv')
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	
0	6	148	72	35	0	33.6	0.62	
1	1	85	66	29	0	26.6	0.35	
2	8	183	64	0	0	23.3	0.67	
3	1	89	66	23	94	28.1	0.16	
4	0	137	40	35	168	43.1	2.28	

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabet
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

Data Wrangling

Clean the data by removing Nan values and unnecessary columns in data set.

```
# df.isnull().values.any()
```

```
7]: False
```

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
x = df.iloc[:, :-2]
y = df.iloc[:, -1]
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. I have divided the train and test size in 1:4 ratio (test_size 0.2).

```
# x_train, x_test, y_train, y_test = train_test_split
(x, y, random_state = 0, test_size = 0.2)
```

Building SVM Model

Let's create a SVM Model using Scikit-learn.

```
▶ clf = svm.SVC(kernel='rbf')
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
```

Accuracy Check

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

```
▶ SVM_accuracy= accuracy_score(y_test, y_pred)*100
print("Accuracy:",SVM_accuracy)
```

Accuracy: 79.22077922077922

5) Naive Bayes algorithm

Naive Bayes Algorithm

Collect Data

Let's first load the required libraries and load the required Diabetes dataset using pandas read_csv function.

```
▶ import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

▶ data=pd.read_csv("diabetes.csv")
data.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	F
0	6	148	72	35	0	33.6		0.62
1	1	85	66	29	0	26.6		0.35
2	8	183	64	0	0	23.3		0.67
3	1	89	66	23	94	28.1		0.16
4	0	137	40	35	168	43.1		2.28
5	5	116	74	0	0	25.6		0.20
6	3	78	50	32	88	31.0		0.24
7	10	115	0	0	0	35.3		0.13
8	2	197	70	45	543	30.5		0.15
9	8	125	96	0	0	0.0		0.23

```
data.dtypes
```

Pregnancies	int64
Glucose	int64
BloodPressure	int64
SkinThickness	int64
Insulin	int64
BMI	float64
DiabetesPedigreeFunction	float64
Age	int64
Outcome	int64
dtype: object	

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables). Our target variable is Outcome and other variables are feature variables.

```
from sklearn.model_selection import train_test_split
X= data.drop( "Outcome", axis=1)
y= data[ "Outcome"]
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. I have divided the train and test size in 1:4 ratio (test_size 0.2).

```
▶ X_train, X_test, y_train, y_test = train_test_split  
    (X, y, test_size=0.3, random_state=1)
```

Building Naive Bayes Model

Let's create a Naive Bayes Model using Scikit-learn

```
▶ from sklearn.naive_bayes import GaussianNB  
model=GaussianNB()  
model.fit(X_train, y_train)  
predictions = model.predict(X_test)
```

Accuracy Check

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

```
▶ from sklearn.metrics import accuracy_score  
gnb = accuracy_score(y_test,predictions)*100  
print("accuracy = ", gnb)
```

```
accuracy = 78.35497835497836
```

6) Random forest algorithm

Random Forest Algorithm

Collect Data

Let's first load the required libraries and load the required Diabetes dataset using pandas `read_csv` function.

```
import pandas as pd  
df=pd.read_csv("diabetes.csv")  
df.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	
0	6	148	72	35	0	33.6	0.62	
1	1	85	66	29	0	26.6	0.35	
2	8	183	64	0	0	23.3	0.67	
3	1	89	66	23	94	28.1	0.16	
4	0	137	40	35	168	43.1	2.28	
5	5	116	74	0	0	25.6	0.20	
6	3	78	50	32	88	31.0	0.24	
7	10	115	0	0	0	35.3	0.13	
8	2	197	70	45	543	30.5	0.15	
9	8	125	96	0	0	0.0	0.23	

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables). Our target variable is Outcome and other variables are feature variables.

```
x = df.drop('Outcome', axis=1)  
y = df['Outcome']
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. I have divided the train and test size in 1:4 ratio (test_size 0.33).

```
▶ from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split  
(X, y, test_size=0.33, random_state=66)
```

Building Random Forest Classifier Model

Let's create a Random Forest Classifier Model using Scikit-learn.

```
▶ from sklearn.ensemble import RandomForestClassifier  
rfc = RandomForestClassifier()  
rfc.fit(X_train,y_train)  
rfc_predict = rfc.predict(X_test)
```

Accuracy Check

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

```
▶ from sklearn.metrics import accuracy_score  
RFaccuracy = accuracy_score(y_test,rfc_predict)*100  
print("accuracy = ", RFaccuracy)
```



```
accuracy = 75.98425196850394
```

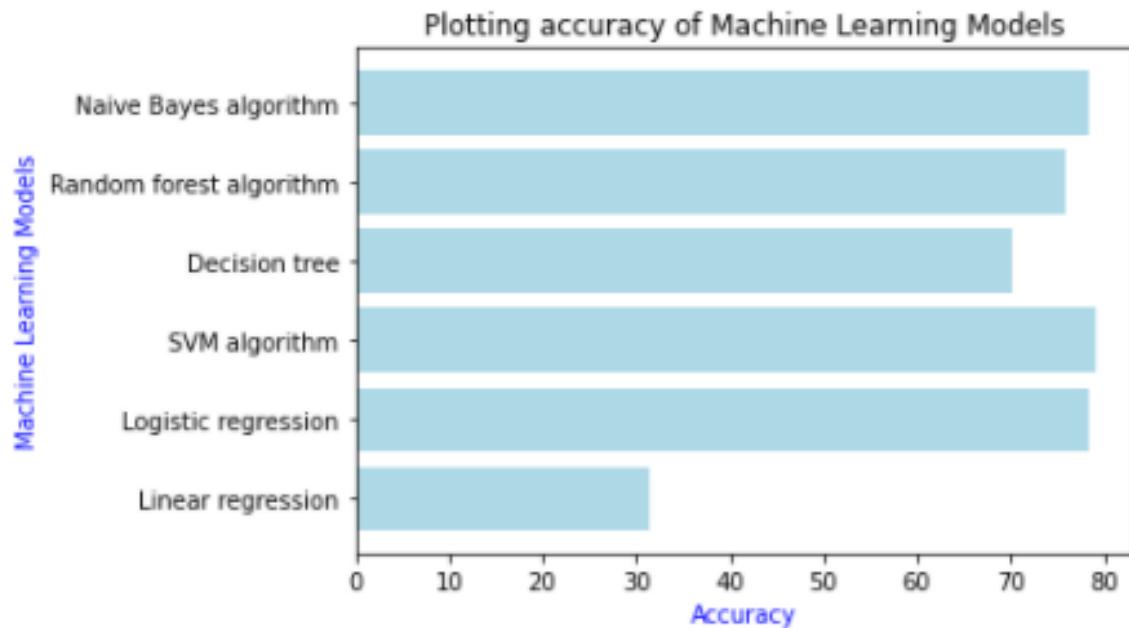
Plotting accuracy of Machine Learning Models

Data declaration

```
a1 = np.array([LinearRegression_Score, LogisticRegression_Score, SVM_accuracy,
               RFaccuracy, gnb])
x = np.array(['Linear regression','Logistic regression','SVM algorithm','Decision tree',
              'Random forest algorithm','Naive Bayes algorithm'])
```

Plotting

```
plt.barh(x,a1,color='lightblue')
plt.title('Plotting accuracy of Machine Learning Models')
plt.ylabel('Machine Learning Models',color='b')
plt.xlabel('Accuracy',color='b')
plt.show()
```



- By analyzing the results of accuracy the best model is **SVM Algorithm**.

SVM Algorithm (Best Accuracy):

Support Vector Machine (SVM)

Collect Data

Let's first load the required libraries and load the required Diabetes dataset using pandas `read_csv` function.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import sklearn.preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
from sklearn import svm
```

```
df = pd.read_csv('diabetes.csv')
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

```
| df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabet
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

Data Wrangling

Clean the data by removing Nan values and unnecessary columns in data set.

```
| df.isnull().values.any()
```

```
?]: False
```

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
| x = df.iloc[:, :-2]  
| y = df.iloc[:, -1]
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. I have divided the train and test size in 1:4 ratio (test_size 0.2).

```
▶ x_train, x_test, y_train, y_test = train_test_split  
    (x, y, random_state = 0, test_size = 0.2)
```

Building SVM Model

Let's create a SVM Model using Scikit-learn.

```
▶ clf = svm.SVC(kernel='rbf')  
clf.fit(x_train,y_train)  
y_pred = clf.predict(x_test)
```

Accuracy Check

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

```
▶ SVM_accuracy= accuracy_score(y_test, y_pred)*100  
print("Accuracy:",SVM_accuracy)  
  
Accuracy: 79.22077922077922
```

Save the model as a pickle in a file

```
▶ pickle.dump(clf,open('dia.pkl','wb'))
```

GUI Code:

Flask

Flask is a web application written in python i.e. it is third party Python library used for developing web application.

Import libraries

```
▶ from flask import Flask,render_template, request  
    import pickle  
    import numpy as np
```

Load the model from the file

```
▶ model = pickle.load(open('dia.pkl','rb'))
```

Run the Webapp

```
▶ app = Flask(__name__)

@app.route('/')
def man():
    return render_template('home.html')

@app.route('/predict',methods=['POST'])
def home():
    data1 = request.form['a']
    data2 = request.form['b']
    data3 = request.form['c']
    data4 = request.form['d']
    data5 = request.form['e']
    data6 = request.form['f']
    data7 = request.form['g']
    arr = np.array([[data1,data2,data3,data4,data5,data6,data7]])
    pred = model.predict(arr)
    return render_template('aftertesting.html',data=pred)

if __name__ == "__main__":
    app.run()

* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production
  deployment.
    Use a production WSGI server instead.
* Debug mode: off

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Home.html:

```
<html>
    <head>          <title>Home</title>
<style>
*{font-family:Times-New-Roman;}
#abc{margin-left:600px;
margin-right:600px;
margin-top:80px;
padding-bottom:80px;
background-color:lightblue; }
</style>
</head>
<body>
    <form id="abc"><br>
<center>
    <img src='static\abc.png' alt="logo" height="200px" width="200px" >
    <h1 style="margin:0px; padding:0px;">Pakistan Diagnosis Center(PDC)</h1>
    <h3>Please enter the required information below</h3><br>
    <form method="POST", action="{{url_for('home')}}">
        <b>
            Pregnancies : <input type="text", name='a', placeholder="Pregnancies" style="margin-left:95px;"> <br><br>
            Glucose : <input type="text", name='b', placeholder="Glucose" style="margin-left:123px;"> <br><br>
            BloodPressure : <input type="text", name='c', placeholder="BloodPressure" style="margin-left:80px;"> <br><br>
            SkinThickness : <input type="text", name='d', placeholder="SkinThickness" style="margin-left:80px;"> <br><br>
            Insulin : <input type="text", name='e', placeholder="Insulin" style="margin-left:132px;"> <br><br>
            BMI : <input type="text", name='f', placeholder="BMI" style="margin-left:147px;"> <br><br>
            DiabetesPedigreeFunction : <input type="text", name='g', placeholder="DiabetesPedigreeFunction"> <br><br><br>    </b>
        </form><input type="submit" , value='predict!' ></center></form>
    </body>
</html>
```

Aftertesting.html:

```
<html>
  <head>
    <title>Home</title>
  </head>
  <body bgcolor=lightblue>
    <center>
      <h1> Pakistan Diagnosis Center(PDC) </h1><br><br>

      <h1> PREDICTION : </h1>

      {@if data == 0%}
      <h1>Patient is not diabetic </h1><br>
      <img src='static\dian.jpg'>

      {@else%}
      <h1>Patient is diabetic</h1><br>
      <img src='static\ok.jpg'>

      {@endif%}

      <br><br>
      <a href='/'>go back to home page</a>

    </center>
  </body>
</html>
```

GUI:



Pakistan Diagnosis Center(PDC)

Please enter the required information below

Pregnancies :

Glucose :

BloodPressure :

SkinThickness :

Insulin :

BMI :

DiabetesPedigreeFunction :



Pakistan Diagnosis Center(PDC)

Please enter the required information below

Pregnancies :

Glucose :

BloodPressure :

SkinThickness :

Insulin :

BMI :

DiabetesPedigreeFunction :

Pakistan Diagnosis Center(PDC)

PREDICTION :

Patient is diabetic



[go back to home page](#)



Pakistan Diagnosis Center(PDC)

Please enter the required information below

Pregnancies :

Glucose :

BloodPressure :

SkinThickness :

Insulin :

BMI :

DiabetesPedigreeFunction :

PREDICTION :

Patient is not diabetic



NON-DIABETIC

[go back to home page](#)



Pakistan Diagnosis Center(PDC)

Please enter the required information below

Pregnancies :

Glucose :

BloodPressure :

SkinThickness :

Insulin :

BMI :

DiabetesPedigreeFunction :