**DATA SCEINCE & MACHINE LEARNING**

**LAB RECORD**

**LAB CYCLE 1**

1. Program to Print all non-Prime Numbers in an Interval

**Program**

lower = int(input("Enter the lower"))  
upper = int(input("Enter the upper"))  
for num in range(lower,upper+1):  
 if num>1:  
 for m in range(2,num):  
 if(num % m)==0:  
 print(num)

break

**Output**

Enter the lower1

Enter the upper10

4

6

8

9

10

1. Program to print the first N Fibonacci numbers.

**Program**

n = int(input("Enter the value of n: "))  
a = 0  
b = 1  
sum = 0  
count = 1  
print("Fibonacci Series: ", end = " ")  
while(count <= n):  
 print(sum, end = " ")  
 count += 1  
 a = b  
 b = sum  
 sum = a + b

**Output**

Enter the value of n: 6

Fibonacci Series: 0 1 1 2 3 5

1. Given sides of a triangle, write a program to check whether given triangle is an isosceles, equilateral or scalene.

**Program**

print("lengths of the triangle sides: ")  
x = int(input("x: "))  
y = int(input("y: "))  
z = int(input("z: "))  
  
if x == y == z:  
 print("Equilateral triangle")  
elif x==y or y==z or z==x:  
 print("isosceles triangle")  
else:  
 print("Scalene triangle")

**Output**

lengths of the triangle sides:

x: 20

y: 35

z: 20

isosceles triangle

1. Program to check whether given pair of number is coprime

**Program**

num1 = int(input("enter the first number: "))  
num2 = int(input("enter the second number: "))  
  
mn = min(num1, num2)  
  
for i in range(1, mn + 1):  
  
 if num1 % i == 0 and num2 % i == 0:  
 hcf = i  
  
if hcf == 1:  
 print("Co-Prime.")  
  
else:  
 print(" not Co-Prime.")

**Output**

enter the first number: 3

enter the second number: 5

Co-Prime.

1. Program to find the roots of a quadratic equation(rounded to 2 decimal places)

**Program**

import cmath  
a = int(input("Enter a:"))  
b = int(input("Enter b:"))  
c = int(input("Enter c:"))  
  
  
d = (b\*\*2) - (4\*a\*c)  
  
  
sol1 = (-b-cmath.sqrt(d))/(2\*a)  
sol2 = (-b+cmath.sqrt(d))/(2\*a)  
  
print('The solution are {0} and {1}'.format(sol1,sol2))

**Output**

Enter a:3

Enter b:5

Enter c:2

The solution are (-1+0j) and (-0.6666666666666666+0j)

1. Program to check whether a given number is perfect number or not(sum of factors =number)

**Program**

n = int(input("Enter any number:"))  
sum1 = 0  
for i in range(1, n):  
 if(n % i == 0):  
 sum1 = sum1 + i  
if (sum1 == n):  
 print("The number is a Perfect number")  
else:  
 print("The number is not a Perfect number")

**Output**

Enter any number:23

The number is not a Perfect number

1. Program to display amstrong numbers upto 1000

**Program**

lower = int(input("enter lower"))  
upper = int(input("enter upper"))  
  
for num in range(lower, upper + 1):  
  
 order = len(str(num))  
  
 sum = 0  
  
 temp = num  
 while temp > 0:  
 digit = temp % 10  
 sum += digit \*\* order  
 temp //= 10  
  
 if num == sum:  
 print(num)

**Output**

enter lower1

enter upper1000

1

2

3

4

5

6

7

8

9

153

370

371

407

1. Store and display the days of a week as a **List, Tuple, Dictionary, Set.** Also demonstrate different ways to store values in each of them. Display its type also.

**Program**

list = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"]  
print(type(list))  
print(list)  
tuple = ("Sun","Mon","Tue","Wed","Thu","Fri","Sat")  
print(type(tuple))  
print(tuple)  
set = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"}  
print(type(set))  
print(set)  
dict = {  
 "d1" : "Sun",  
 "d2" : "Mon",  
 "d3" : "Tue",  
 "d4" : "Wed",  
 "d5" : "Thu",  
 "d6" : "Fri",  
 "d7" : "Sat"  
}  
print(type(dict))  
print(dict)

**Output**

<class 'list'>

['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']

<class 'tuple'>

('Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat')

<class 'set'>

{'Thu', 'Fri', 'Sun', 'Mon', 'Sat', 'Tue', 'Wed'}

<class 'dict'>

{'d1': 'Sun', 'd2': 'Mon', 'd3': 'Tue', 'd4': 'Wed', 'd5': 'Thu', 'd6': 'Fri', 'd7': 'Sat'}

1. Write a program to add elements of given 2 lists

**Program**

l1 = [5, 10, 15, 20, 25, 30]  
l2 = [2, 4, 6, 8, 10, 12]  
  
print("list 1: " + str(l1))  
print("list 2: " + str(l2))  
  
res = []  
for x in range(0, len(l1)):  
 res.append(l1[x] + l2[x])  
  
print("Addition of the list l1 and l2 is:" + str(res))

**Output**

list 1: [5, 10, 15, 20, 25, 30]

list 2: [2, 4, 6, 8, 10, 12]

Addition of the list l1 and l2 is:[7, 14, 21, 28, 35, 42]

1. Write a program to find the sum of 2 matrices using nested List.

**Program**

rows = int(input("Enter the Number of rows : "))  
column = int(input("Enter the Number of Columns: "))  
  
print("Enter the elements of First Matrix:")  
matrix\_a = [[int(input()) for i in range(column)] for i in range(rows)]  
print("First Matrix is: ")  
for n in matrix\_a:  
 print(n)  
  
print("Enter the elements of Second Matrix:")  
matrix\_b = [[int(input()) for i in range(column)] for i in range(rows)]  
for n in matrix\_b:  
 print(n)  
  
result = [[0 for i in range(column)] for i in range(rows)]  
  
for i in range(rows):  
 for j in range(column):  
 result[i][j] = matrix\_a[i][j] + matrix\_b[i][j]  
  
print("The Sum of Above two Matrices is : ")  
for r in result:  
 print(r)

**Output**

Enter the Number of rows : 2

Enter the Number of Columns: 2

Enter the elements of First Matrix:

1

2

3

4

First Matrix is:

[1, 2]

[3, 4]

Enter the elements of Second Matrix:

1

2

3

4

[1, 2]

[3, 4]

The Sum of Above two Matrices is :

[2, 4]

[6, 8]

1. Write a program to perform bubble sort on a given set of elements.

**Program**

def bubble\_sort(list1):  
 for i in range(0, len(list1) - 1):  
 for j in range(len(list1) - 1):  
 if (list1[j] > list1[j + 1]):  
 temp = list1[j]  
 list1[j] = list1[j + 1]  
 list1[j + 1] = temp  
 return list1  
list1 = [4, 9, 8, 6, 1, 2]  
print("The unsorted list is: ", list1)  
print("The sorted list is: ", bubble\_sort(list1))

**Output**

The unsorted list is: [4, 9, 8, 6, 1, 2]

The sorted list is: [1, 2, 4, 6, 8, 9]

1. Program to find the count of each vowel in a string(use dictionary)

**Program**

string=input("Enter string:")  
vowels=0  
for i in string:  
 if(i=='a' or i=='e' or i=='i' or i=='o' or i=='u' or i=='A' or i=='E' or i=='I' or i=='O' or i=='U'):  
 vowels=vowels+1  
print("Number of vowels:",vowels)

**Output**

Enter string:hello

Number of vowels: 2

1. Write a Python program that accept a positive number and subtract from this number the sum of its digits and so on. Continues this operation until the number is positive

**Program**

def repeat\_times(n):  
 s = 0  
 n\_str = str(n)  
 while (n > 0):  
 n -= sum([int(i) for i in list(n\_str)])  
 n\_str = list(str(n))  
 s += 1  
 return s  
print(repeat\_times(9))  
print(repeat\_times(21))

**Output**

1

3

1. Write a Python program that accepts a 10 digit mobile number, and find the digits which are absent in a given mobile number

**Program**

def absent\_digits(n):  
 all\_nums = set([0,1,2,3,4,5,6,7,8,9])  
 n = set([int(i) for i in n])  
 n = n.symmetric\_difference(all\_nums)  
 n = sorted(n)  
 return n  
print(absent\_digits([7,3,5,6,1,6,0,1,0,0]))

**Output**

[2, 4, 8, 9]

**LAB CYCLE 2**

**1. Create a three dimensional array specifying float data type and print it.**

**Program**

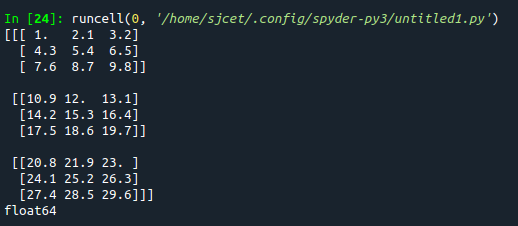
import numpy as np

array = np.arange(1,30,1.1).reshape(3,3,3)

print(array)

print(array.dtype)

**Output**



**2. Create a 2 dimensional array (2X3) with elements belonging to complex data type and print it. Also display**

**a. the no: of rows and columns**

**b. dimension of an array**

**c. reshape the same array to 3X2**

**Program**

import numpy as np

array = np.arange(6).astype(complex).reshape(2,3)

print(array)

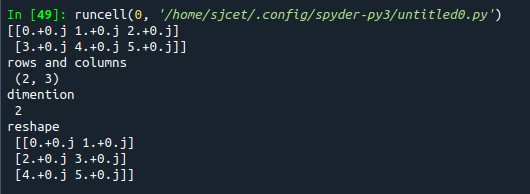
print("rows and columns\n", array.shape)

print("dimention\n", array.ndim)

arr2 = array.reshape(3,2)

print("reshape\n", arr2)

**Output**



**3. Familiarize with the functions to create**

**a) an uninitialized array**

**b) array with all elements as 1,**

**c) all elements as 0**

**Program**

import numpy as np

a = []

print(a)

#1

print(np.empty(2))

#2

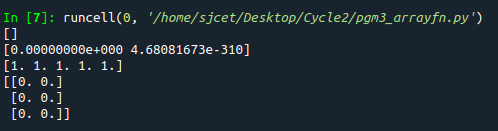
print(np.ones(5))

#3

a=(3,2)

print(np.zeros(a))

**Output**



**4. Create an one dimensional array using arange function containing 10 elements.**

**Display**

**a. First 4 elements**

**b. Last 6 elements**

**c. Elements from index 2 to 7**

**Program**

import numpy as np

print(np.arange(1, 11, 1))

# 1

print(np.arange(1, 5, 1))

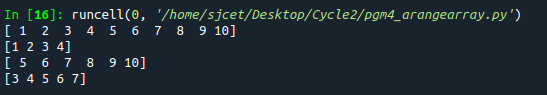
# 2

print(np.arange(5, 11, 1))

# 3

print(np.arange(3, 8, 1))

**Output**



**5. Create an 1D array with arange containing first 15 even numbers as elements**

**a. Elements from index 2 to 8 with step 2(also demonstrate the same**

**using slice function)**

**b. Last 3 elements of the array using negative index**

**c. Alternate elements of the array**

**d. Display the last 3 alternate elements**

**Program**

import numpy as np

array=np.arange(0,15,2)

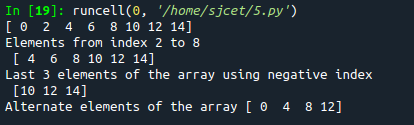
print(array)

print("Elements from index 2 to 8\n",array[2:8])

print("Last 3 elements of the array using negative index\n",array[-3:])

print("Alternate elements of the array",array[::2])

**Output**



**6. Create a 2 Dimensional array with 4 rows and 4 columns.**

**a. Display all elements excluding the first row**

**b. Display all elements excluding the last column**

**c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row**

**d. Display the elements of 2 nd and 3 rd column**

**e. Display 2 nd and 3 rd element of 1 st row**

**f. Display the elements from indices 4 to 10 in descending order(use**

**–values)**

**Program**

import numpy as np

x = np.array([[2, 4, 6,1], [6, 8, 10,1],[1, 2, 1,1], [1, 1, 1,1]])

print(x)

#1

print("excluding first row\n",x[1:])

#2

print("excluding last column\n",x[:,:3])

#3

print("Display the elements of 1st and 2nd column in 2nd and 3rd row")

print(x[1:3,0:2])

#4

print("dispaly 2 and 3 element",x[:1,1:3])

#5

print("display 2nd and 3rd element of 1st row")

print(x[0:1,1:3])

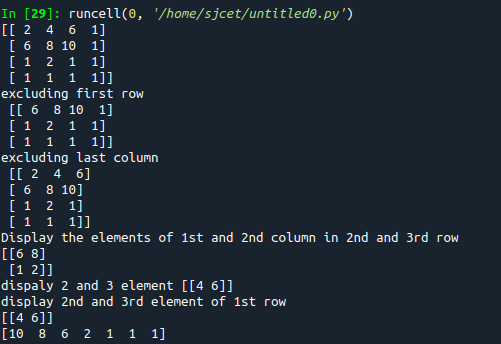
#6

arr=np.array([1,6,8,10,1,1,2])

sarr=np.sort(arr)[::-1]

print(sarr)

**Output**



**7. Create two 2D arrays using array object and**

**a. Add the 2 matrices and print it**

**b. Subtract 2 matrices**

**c. Multiply the individual elements of matrix**

**d. Divide the elements of the matrices**

**e. Perform matrix multiplication**

**f. Display transpose of the matrix**

**g. Sum of diagonal elements of a matrix**

**Program**

import numpy as np

array = np.arange(1,5,1).reshape(2,2)

array2 = np.arange(6,10,1).reshape(2,2)

print ("\n", array)

print ("\n", array2)

print ("\nsum of two 2darrays is: \n", array + array2)

print ("\n2darrays subtracted: \n", array - array2)

print ("\nproduct of individual elements: \n", array \* array2)

print ("\n2darrays divided: \n", array / array2)

matrixprod = np.matmul(array, array2)

print("\nproduct of two matrices\n", matrixprod)

transpose = np.transpose(array)

print("\ntranspose of 1st array\n", transpose)

transpose2 = np.transpose(array2)

print("\ntranspose of 2nd array\n", transpose2)

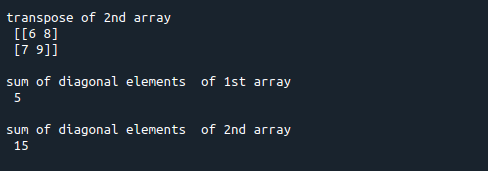
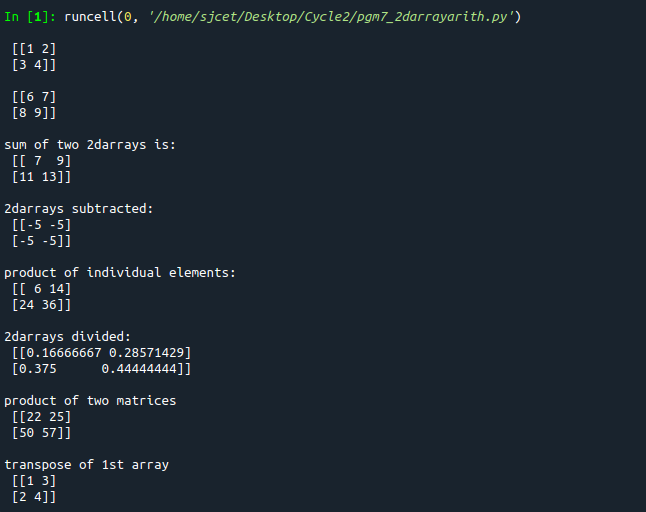
sumofd = np.trace(array)

print("\nsum of diagonal elements of 1st array\n", sumofd)

sumofd2 = np.trace(array2)

print("\nsum of diagonal elements of 2nd array\n", sumofd2)

**Output**



**8. Demonstrate the use of insert() function in 1D and 2D array**

**Program**

import numpy as np

arr1 = np.arange(10, 16)

print("1D ARRAY ")

print("\nThe array is: ", arr1)

obj = 2

value = 40

arr = np.insert(arr1, obj, value, axis=None)

print("\nAfter inserting the new array is: ")

print(arr)

print("\nShape of the new array is : ", np.shape(arr))

print("\n2D ARRAY ")

arr1 = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9), (50, 51, 52)])

print("\nThe array is: ")

print(arr1)

print("\nThe shape of the array is: ", np.shape(arr1))

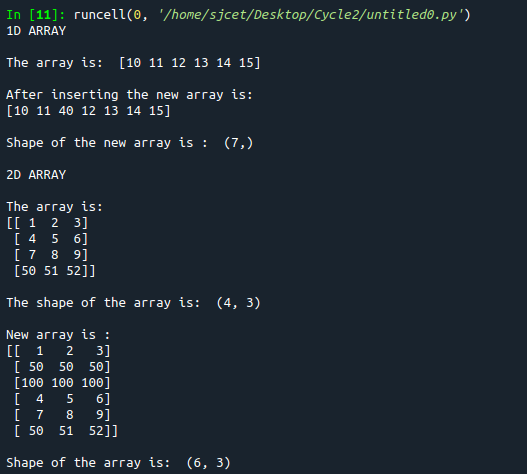
a = np.insert(arr1, 1, [[50], [100]], axis=0)

print("\nNew array is : ")

print(a)

print("\nShape of the array is: ", np.shape(a))

**Output**



**9. Demonstrate the use of diag() function in 1D and 2D array.**

**Program**

**Program**

import numpy as np

a= np.array([[3, 6,7,8]])

b=np.array([[3, 6,8,7], [4, 2,1,0],[3,1,3,3],[1,1,2,2]])

print("\n",a)

print("\n",b)

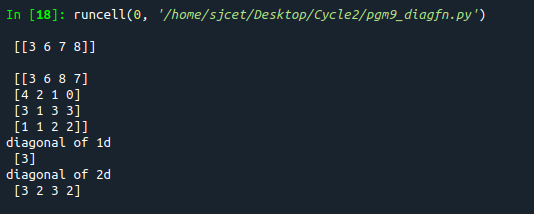
x=np.diag(a)

y=np.diag(b)

print("diagonal of 1d\n",x)

print("diagonal of 2d\n",y)

**Output**



**10. Demonstarte the use of append() function in 1D and 2D**

**array.**

**Program**

import numpy as np

a = np.array([(1, 2, 3), (4, 5, 6)])

b = np.array([1, 2, 3])

print("First array:")

print(a)

print("Second array")

print(b)

print("\n")

print("Append elements to array:")

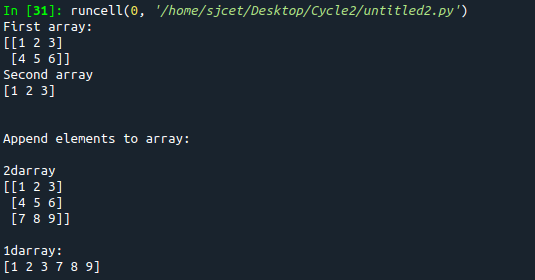
print("\n2darray")

print(np.append(a, [7, 8, 9]).reshape(3, 3))

print("\n1darray:")

print(np.append(b, [7, 8, 9]))

**Output**



**11. Demonstarte the use of sum() function in 1D and 2D array.**

**Program**

import numpy as np

a=np.array([0.4,0.5])

b=np.sum(a)

print ("\nsum:", b)

**Output**



**12.Create a 1 Dimensional array .Display the elements from indices 4 to 10 in descending order(use–values)**

**Program**

import numpy as np

a = np.array([1,2,8,9,3,4,5,6,7,10])

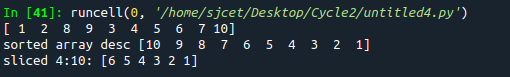
print(a)

array\_copy = np.sort(a)[::-1]

print("sorted array desc",array\_copy)

print("sliced 4:10:",array\_copy[4:10])

**Output**



**Part-2**

**2. Create a square matrix with random integer values(use randint()) and use**

**appropriate functions to find:**

**i) inverse**

**ii) rank of matrix**

**iii) Determinant**

**iv) transform matrix into 1D array**

**v) eigen values and vectors**

**Program**

import numpy as np

arr1 = np.arange(10, 16)

print("1D ARRAY ")

print("\nThe array is: ", arr1)

obj = 2

value = 40

arr = np.insert(arr1, obj, value, axis=None)

print("\nAfter inserting the new array is: ")

print(arr)

print("\nShape of the new array is : ", np.shape(arr))

print("\n2D ARRAY ")

arr1 = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9), (50, 51, 52)])

print("\nThe array is: ")

print(arr1)

print("\nThe shape of the array is: ", np.shape(arr1))

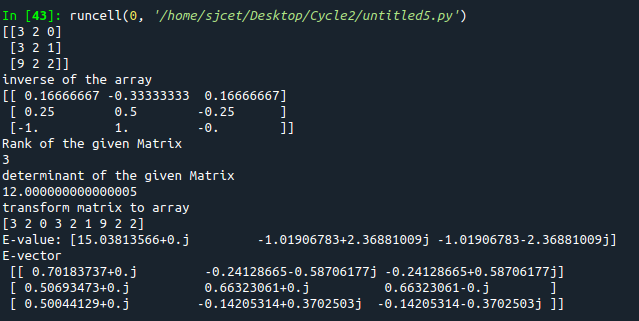
a = np.insert(arr1, 1, [[50], [100]], axis=0)

print("\nNew array is : ")

print(a)

print("\nShape of the array is: ", np.shape(a))

**Output**



**2. Create a matrix X with suitable rows and columns**

**i) Display the cube of each element of the matrix using different methods**

**(use multiply(), \*, power(),\*\*)**

**ii) Display identity matrix of the given square matrix.**

**iii) Display each element of the matrix to different powers.**

**iv) Create a matrix Y with same dimension as X and perform the operation X 2 +2Y**

**Program**

import numpy as np

arr1=np.arange(4,8).reshape(2,2)

print("\narray\n",arr1)

#1

print("\ncube using power\n")

b=np.power(arr1,3)

print(b)

f=arr1\*\*3

print("\n",f)

print("\ncube using multiply\n")

c=np.multiply(arr1,arr1)

d=np.multiply(c,arr1)

print("\n",d)

e=arr1\*arr1\*arr1

print("\n",e)

#2

print("\nidentity matrix\n")

h=np.identity(2)

print("\n",h)

#3

print("\nDisplay each element of the matrix to different powers.")

i=np.power(arr1[0][0],2)

j=np.power(arr1[0][1],3)

k=np.power(arr1[1][0],4)

m=np.power(arr1[1][1],5)

print("\n",i)

print("\n",j)

print("\n",k)

print("\n",m)

#4

print("\nCreate a matrix Y with same dimension as X and perform the operation X 2 +2Y")

arr2=np.arange(0,4).reshape((2,2))

print("\n",arr2)

n=np.power(arr1,2)

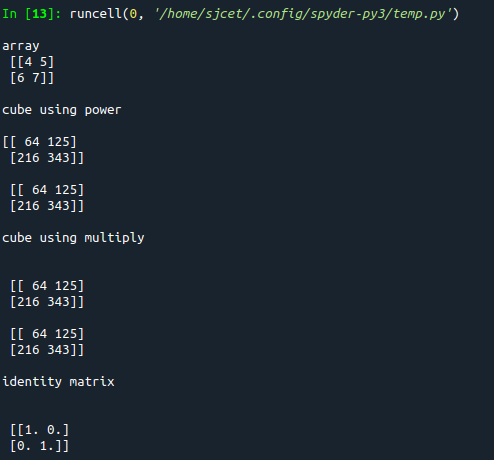
o=np.multiply(2,arr2)

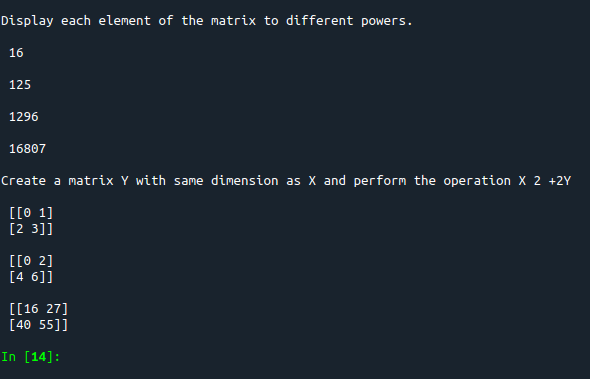
print("\n",o)

p=np.add(n,o)

print("\n",p)

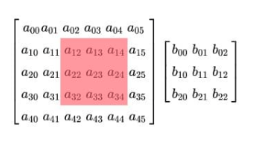
**Output**





**3. Multiply a matrix with a submatrix of another matrix and replace the same in larger**

**matrix.**

****

**Program**

import numpy as np

print("\nLarge matrix")

arr1=np.arange(1,37).reshape((6,6))

print(arr1)

print("\nsmall matrix")

arr2=np.arange(1,10).reshape((3,3))

print(arr2)

print("\ncutout portion")

a=arr1[0:3,1:4]

print(a)

print("\nmultiplying with smaller matrix")

c=np.multiply(a,arr2)

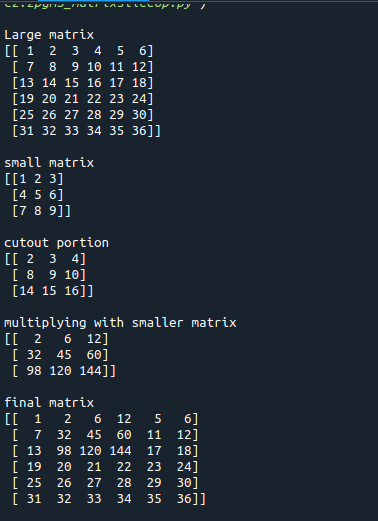
print(c)

print("\nfinal matrix")

arr1[0:3,1:4]=c

print(arr1)

**Output**



**4. Given 3 Matrices A, B and C. Write a program to perform matrix multiplication of**

**the 3 matrices.**

**Program**

import numpy as np

#matrix multiplication of 3 matrices.

arr1=np.arange(1,10).reshape((3,3))

arr2=np.arange(11,20).reshape((3,3))

arr3=np.arange(21,30).reshape((3,3))

print("\n1stmatrix")

print(arr1)

print("\n2ndmatrix")

print(arr2)

print("\n3rdmatrix")

print(arr3)

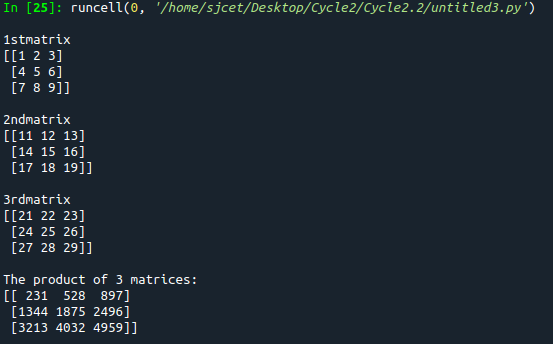
a=np.multiply(arr1,arr2)

b=np.multiply(a,arr3)

print("\nThe product of 3 matrices:")

print(b)

**Output**



**5. Write a program to check whether given matrix is symmetric or Skew Symmetric.**

**Program**

import numpy as np

a= np.array([[1, 2, 3],

[2, 1, 2],

[3, 2, 1]])

x=np.transpose(a)

print(a)

c=np.array\_equal(a, x)

print("symmetric or not:" ,c)

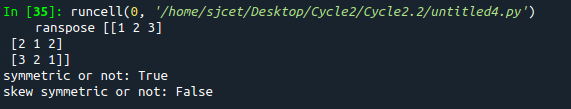
b=np.negative(a)

print(b)

d=np.array\_equal(x,b)

print("skew symmetric or not:",d)

**Output**



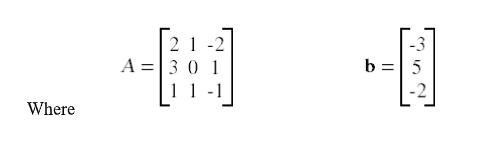
**6. Write a program to find out the value of X using solve(), given A and b as above**

**Solving systems of equations with numpy**

**One of the more common problems in linear algebra is solving a matrix-vector equation.**

**Here is an example. We seek the vector x that solves the equation**

**A X = b**

****

**And X=A -1 b.**

**Numpy provides a function called solve for solving such eauations.**

**Program**

import numpy as np

a=np.array([[2,1,-2],[3,0,1],[1,1,-1]])

b=np.array([[-3],[5],[-2]])

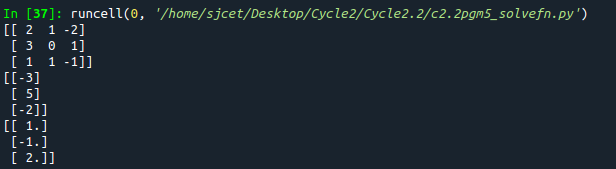
print(a)

print(b)

x = np.linalg.solve(a, b)

print(x)

**Output**



**7. Write a program to perform the SVD of a given matrix. Also reconstruct the given matrix**

**from the 3 matrices obtained after performing SVD.**

**Singular value Decomposition**

**Matrix decomposition, also known as matrix factorization, involves describing a given**

**matrix using its constituent elements.**

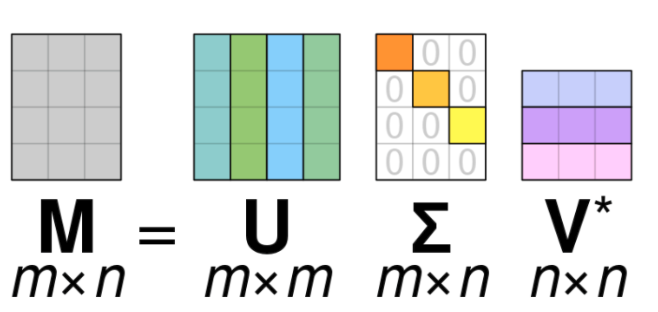
**The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for**

**reducing a matrix to its constituent parts in order to make certain subsequent matrix**

**calculations simpler. This approach is commonly used in reducing the no: of attributes in**

**the given data set.**

**M= U ∑V^T**

****

** M-is original matrix we want to decompose**

** U-is left singular matrix (columns are left singular vectors). U columns contain**

**eigenvectors of matrix MMᵗ**

** Σ-is a diagonal matrix containing singular (eigen) values.**

** V-is right singular matrix (columns are right singular vectors). V columns contain**

**eigenvectors of matrix MᵗM**

**Numpy provides a function for performing svd, which decomposes the given matrix into 3**

**matrices.**

**Program**

from numpy import array

from scipy.linalg import svd

from numpy import dot

from numpy import diag

A = array([[1, 2,1], [3, 4,2], [5, 6,4]])

print(A)

U, s, VT = svd(A)

print(U)

print(s)

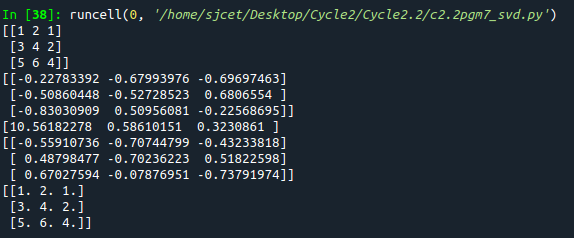
print(VT)

Sigma=diag(s)

B=U.dot(Sigma.dot(VT))

print(B)

**Output**

****

**Lab Cycle 3**

**Demonstrate creating various types of charts and plots using functions in mathplotlib library**

**1.Sarah bought a new car in 2001 for $24,000. The dollar value of her car changed each year as shown in the table below.**

**Value of Sarah's Car**

**Year Value**

**2001 $24,000**

**2002 $22,500**

**2003 $19,700**

**2004 $17,500**

**2005 $14,500**

**2006 $10,000**

**2007 $ 5,800**

**Represent the following information using a line graph with following style properties**

1. **X- axis - Year**

**Y –axis - Car Value**

1. **title –Value Depreciation (left Aligned)**
2. **Line Style dashdot and Line-color should be red**
3. **point using \* symbol with green color and size 20**

**Subplot() provides multiple plots in one figure.**

import matplotlib.pyplot as plt

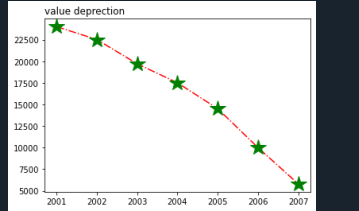
import numpy as np

xpoints = np.array([2001,2002,2003,2004,2005,2006,2007])

ypoints = np.array([24000,22500,19700,17500,14500,10000,5800])

plt.plot(xpoints,ypoints,'-.r',marker ='\*' ,ms=20, mec="g", mfc ='g')

plt.title("valuedeprection",loc="left")



**2.Following table gives the daily sales of the following items in a shop**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Day** | **Mon** | **Tues** | **Wed** | **Thurs** | **Fri** |
| **Drinks** | **300** | **450** | **150** | **400** | **650** |
| **Food** | **400** | **500** | **350** | **300** | **500** |

**Use subplot function to draw the line graphs with grids(color as blue  and line style dotted) for the above information as 2 separate graphs in two rows**

1. **Properties for the Graph 1:**
2. **X label- Days of week**
3. **Y label-Sale of Drinks**
4. **Title-Sales Data1 (right aligned)**
5. **Line –dotted with cyan color**
6. **Points- hexagon shape with color magenta and outline black**
7. **Properties for the Graph 2:**
8. **X label- Days of Week**
9. **Y label-Sale of Food**
10. **Title-Sales Data2 ( center aligned)**
11. **Line –dashed with yellow color**
12. **Points- diamond shape with color green and outline red**

import numpy as np

import matplotlib.pyplot as plt

#plot1

x= np.array(['mon','tue','wed','thur','fri'])

y = np.array([300,450,150,400,600])

plt.subplot(2,1,1)

plt.plot(x,y,color ='cyan', linestyle='dotted',marker ='h' ,ms=20, mec="black", mfc ='m')

plt.grid(color ='green', linestyle='dotted')

plt.title("sales data 1",loc="right")

plt.xlabel("days of week")

plt.ylabel("drinks")

#plot2

x= np.array(['mon','tue','wed','thur','fri'])

y = np.array([400,500,350,300,500])

plt.subplot(2,1,2)

plt.plot(x,y,'.-y',marker = 'd',ms=20,mec='r',mfc='g')

plt.grid(color ='green', linestyle='dotted')

plt.title("sales data2",loc="right")

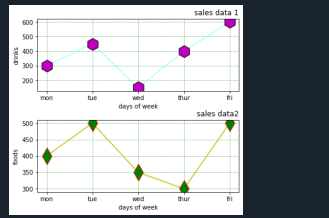
plt.xlabel("days of week")

plt.ylabel("foods")plt.subplots\_adjust(top=2.5,

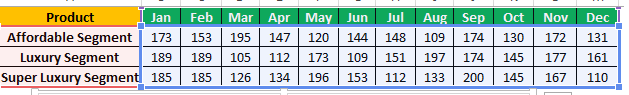
bottom=1.5,

wspace=0.4,

hspace=0.4 )



**3.Create scatter plot for the below data:(use Scatter function)**



**Create scatter plot for each Segment with following properties within one graph**

* **X Label- Months of Year with font size 18**
* **Y-Label- Sales of Segments**
* **Title –Sales Data**
* **Color for Affordable segment- pink**
* **Color for Luxury Segment- Yellow**
* **Color for Super luxury segment-blue**

import matplotlib.pyplot as plt

import numpy as np

z=np.array(['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'])

x = np.array([173,153,195,147,120,144,148,109,174,130,172,131])

plt.scatter(z,x,color="pink")

plt.xlabel("month of year",fontsize="18")

plt.ylabel("sales of segment")

plt.title(("sales data"))

z=np.array(['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'])

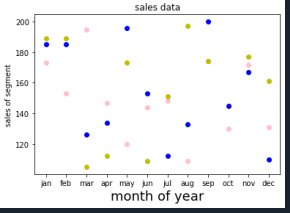
y = np.array([189,189,105,112,173,109,151,197,174,145,177,161])

plt.scatter(z, y,color="y")

q=np.array(['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'])

z = np.array([185,185,126,134,196,153,112,133,200,145,167,110])

plt.scatter(q,z,color="b")

****

**4.Display the above data using multiline plot( 3 different lines in same graph)**

* **Display the description of the graph in upper right corner(use legend())**
* **Use different colors and line styles for 3 different lines**

import matplotlib.pyplot as plt

x1 = ['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']

x2 = [173, 153, 195, 147, 120, 144, 148, 109, 174, 130, 172, 131]

plt.plot(x1, x2, color = 'hotpink', label = 'line 1', ls = '-.')

y1 = ['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']

y2 = [189, 189, 105, 112, 173, 109, 151, 197, 174, 145, 177, 161]

plt.plot(y1, y2, color = 'y', label = 'line 2', linestyle = 'dashed')

z1 = ['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']

z2 = [185, 185, 126, 134, 196, 153, 112, 133, 200, 145, 167, 110]

plt.plot(z1, z2, color = 'blue', label = 'line 3',linestyle = 'dotted')

plt.title("sales data")

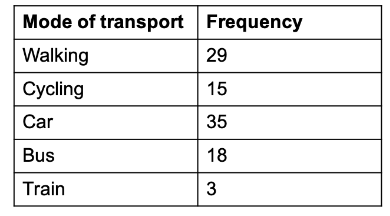
plt.xlabel("Months")

plt.ylabel("sales of segments")

plt.legend()

plt.show()

1. **100 students were asked what their primary mode of transport for getting to school was. The results of this survey are recorded in the table below. Construct a bar graph representing this information.**



**Create a bar graph with**

* **X axis -mode of Transport and Y axis ‘frequency’**
* **Provide appropriate labels and title**
* **Width .1, color  green**

import matplotlib.pyplot as plt

import numpy as np

x = np.array(["Walking", "Cycling", "Car", "Bus", "Train"])

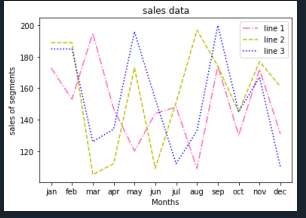
y = np.array([29, 15, 35, 18, 3])

plt.xlabel("mode of transport")

plt.ylabel("frequency")

plt.bar(x,y, color ="g", width = 0.1)

plt.show()

****

1. **We are provided with the height of 30 cherry trees.**

**The height of the trees (in inches): 61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87.Create a histogram with a bin size of 5**

import matplotlib.pyplot as plt

height = [61, 63, 64, 66, 68, 69, 71, 71.5,

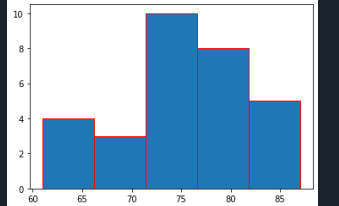
72, 72.5, 73, 73.5, 74, 74.5, 76,

76.2,76.5, 77, 77.5, 78, 78.5, 79,

79.2, 80, 81, 82, 83, 84, 85, 87]

plt.hist(height, edgecolor="red",bins=5)

plt.show()



**DATA HANDLING USING ‘Pandas’ and DATA VISUALIZATION USING ‘Seaborn’**

Using the pandas function read\_csv(), read  the given ‘iris’ data set.

1.Use appropriate functions in pandas to display

1. Shape of the data set
2. First 5 and last five rows of data set(head and tail)
3. Size of dataset
4. No:of samples available for each variety
5. Description of the data set( use describe

**code**

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

col=['sepal\_length','sepal\_width','petal\_length','petal\_width','variety']

iris=pd.read\_csv("iris.csv",names=col)

print(col)

#shape 1

print(iris.shape)

#2

print(iris.head())

print(iris.tail())

#3

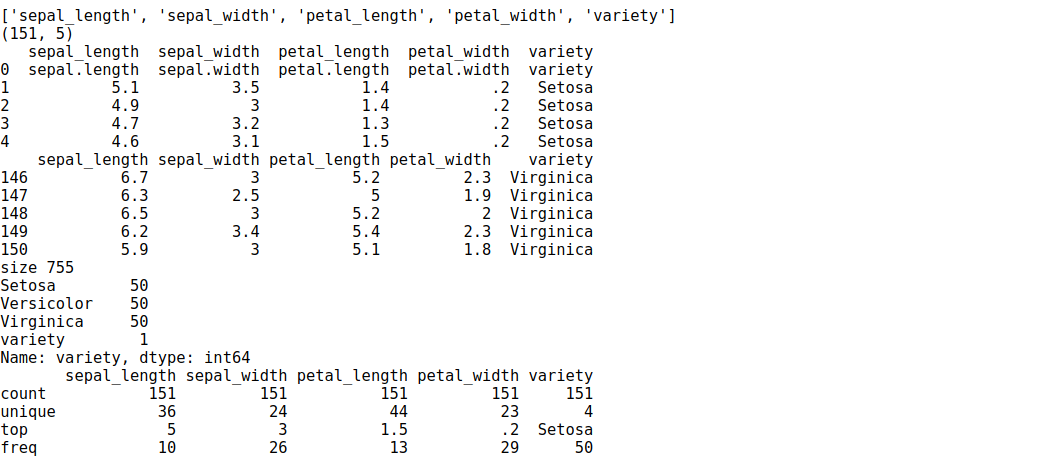
print("size",iris.size)

#4

print(iris["variety"].value\_counts())

#5

print(iris.describe())

****

**2. Use  pairplot() function to display pairwise relationships between attributes. Try different kind of plots  {‘*scatter’, ‘kde’, ‘hist’, ‘reg’}* and different kind of markers**

**code**

ir = sns.load\_dataset('iris')

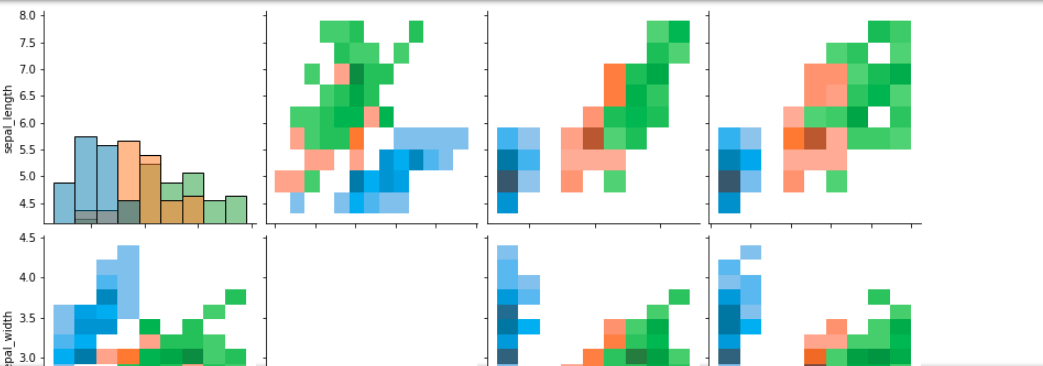
g = pd.DataFrame(ir)

g1=sns.pairplot(g,kind="hist", hue="species",height=3)

g1=sns.pairplot(g,kind="kde" , hue="species", vars=["sepal\_length","petal\_length"])

g1=sns.pairplot(g,kind="scatter" ,hue="species",markers=["o","s","D"])

g1=sns.pairplot(g,kind="reg" ,hue="species")



#### **3.using the iris data set,get familiarize with functions:**

**1)displot()**

**2) histplot()**

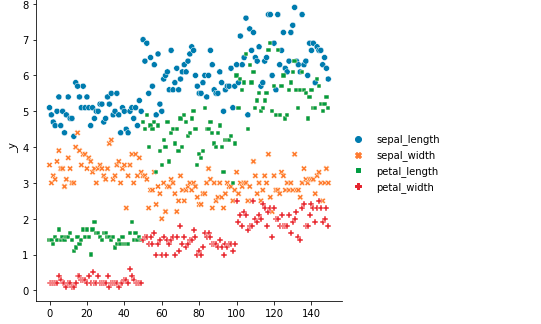
**3) relplot()**

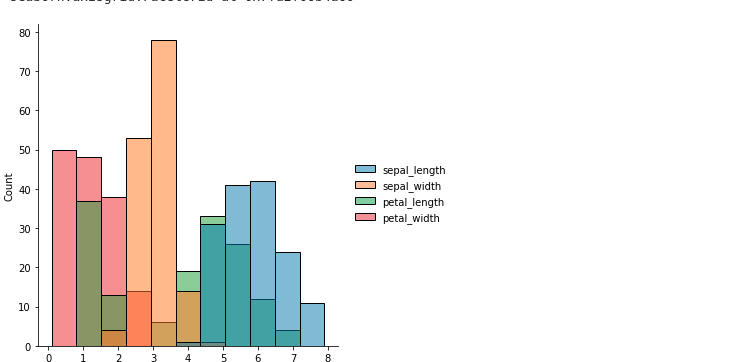
**code**

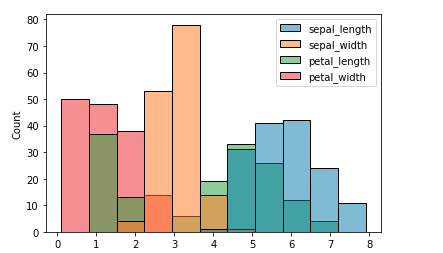
sns**.**relplot(data**=**ir)

sns**.**displot(data**=**ir)

sns**.**histplot(data**=**ir,)

****

****

****

**LAB CYCLE 4**

**1.Using the iris data set implement the KNN algorithm. Take different values for Test and training data set. Also use different values for k. Also find the accuracy level.**

**CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv("iris.csv")

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=5)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix

print(classification\_report(y\_test, y\_pred))

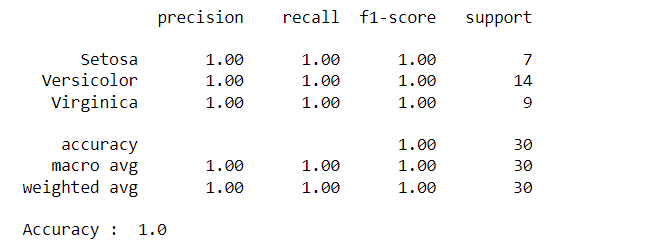
from sklearn.metrics import accuracy\_score

print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

df

**OUTPUT**



Reference: <https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/>

**2.Download another data set suitable for the KNN and implement the KNN algorithm. Take different values for Test and training data set. Also use different values for k.**

**CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv("cancer.csv")

dataset.head()

dataset.info()

X = dataset.iloc[:, 2:35].values

print(X)

y = dataset.iloc[:, 1].values

print(y)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=5)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix

print(classification\_report(y\_test, y\_pred))

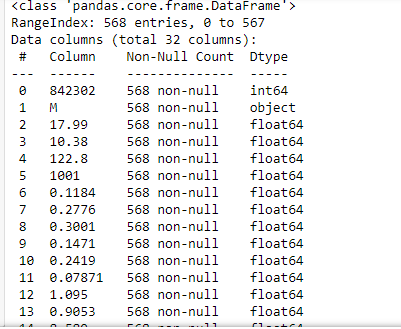
from sklearn.metrics import accuracy\_score

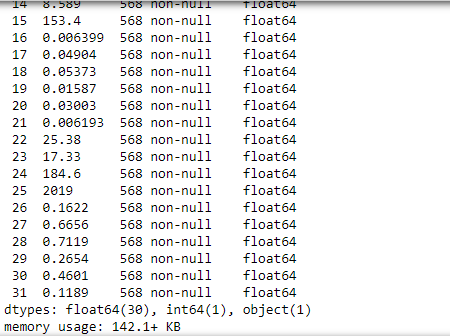
print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

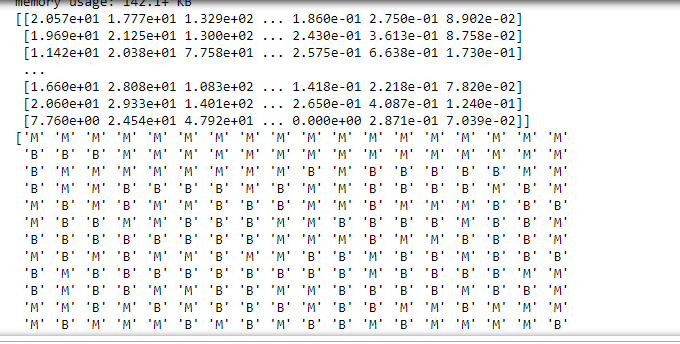
df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

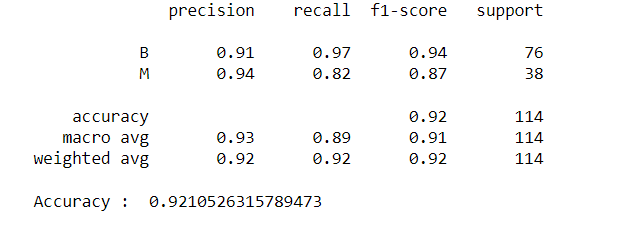
df

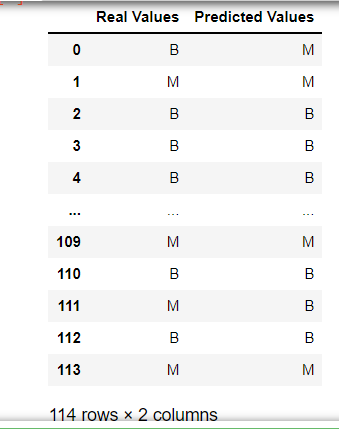
**OUTPUT**











**3.Using iris data set, implement naive bayes classification for different naive Bayes classification algorithms. ((i) gaussian (ii) bernoulli etc)**

1. **Find out the accuracy level w.r.t to each algorithm**
2. **Display the no:of mislabeled classification from test data set**
3. **List out the class labels of the mismatching records**

**i)CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('iris.csv')

X = dataset.iloc[:,:4].values

y = dataset['variety'].values

dataset.head(5)

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)

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

y\_pred

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

from sklearn.metrics import accuracy\_score

print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

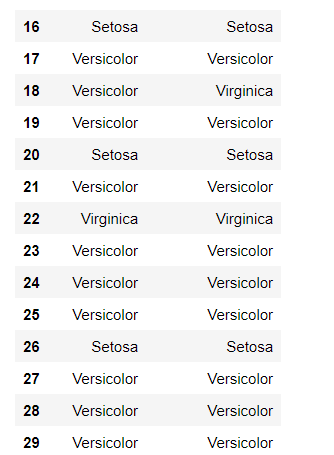
cm

df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

df

**OUTPUT**





**CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('iris.csv')

X = dataset.iloc[:,:4].values

y = dataset['variety'].values

dataset.head(5)

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)

from sklearn.naive\_bayes import BernoulliNB

classifier = BernoulliNB()

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

y\_pred

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

from sklearn.metrics import accuracy\_score

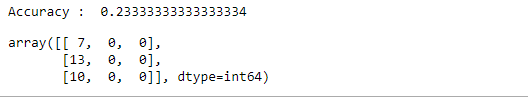
print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

cm

# df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

# df

**OUTPUT**



**ii)CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('iris.csv')

X = dataset.iloc[:,:4].values

y = dataset['variety'].values

dataset.head(5)

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)

from sklearn.naive\_bayes import BernoulliNB

classifier = BernoulliNB()

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

y\_pred

#from sklearn.metrics import confusion\_matrix

#cm = confusion\_matrix(y\_test, y\_pred)

from sklearn.metrics import accuracy\_score

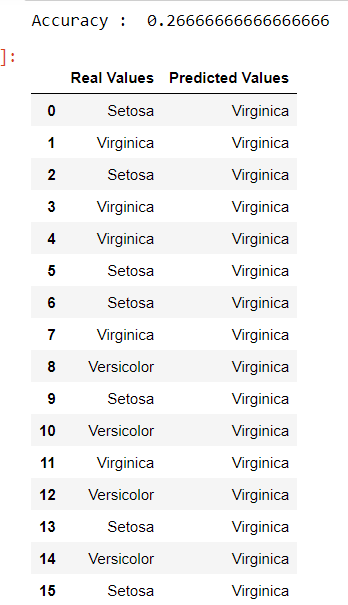
print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

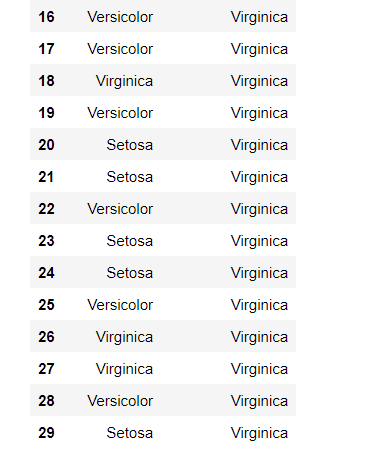
#cm

df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

df

**OUTPUT**





References:

<https://towardsdatascience.com/machine-learning-basics-naive-bayes-classification-964af6f2a965>

[https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive\_bayes](https://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.naive_bayes)

**4.Use car details CSV file and implement decision tree algorithm**

1. **Find out the accuracy level.**
2. **Display the no: of mislabelled classification from test data set**
3. **List out the class labels of the mismatching records**

References:

<https://www.24tutorials.com/machine-learning/case-study-decision-tree-model-for-car-quality/>

<https://notebook.community/bMzi/ML_in_Finance/0210_DecisionTrees>

<https://stackabuse.com/decision-trees-in-python-with-scikit-learn/>

For Data Sets Refer:

[https://www.kaggle.com](https://www.kaggle.com/) ( for data set)

<http://archive.ics.uci.edu/ml/datasets.php>

**REGRESSION**

Implement Simple and multiple linear regression for the data sets ‘student\_score.csv’ and ‘company\_data .csv’ respectively

Ref:  <https://stackabuse.com/linear-regression-in-python-with-scikit-learn/>

**Simple Linear Regression**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

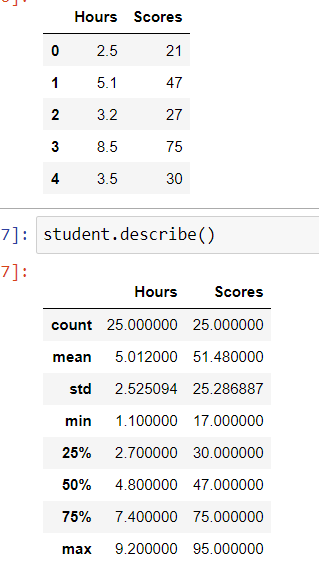
#data set contains details of no.of hours spend by students for studt and their marks

student = pd.read\_csv('student\_scores.csv')

student.head()

student.describe()

**OUTPUT:**



student.info()

**OUTPUT:**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 25 entries, 0 to 24

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Hours 25 non-null float64

1 Scores 25 non-null int64

dtypes: float64(1), int64(1)

memory usage: 528.0 bytes

import matplotlib.pyplot as plt

Xax=student.iloc[:,0]

Yax=student.iloc[:,1]

plt.scatter(Xax,Yax)

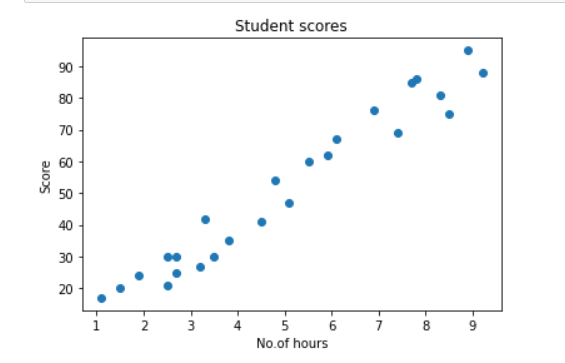
plt.xlabel("No.of hours")

plt.ylabel("Score")

plt.title("Student scores")

plt.show()

**OUTPUT:**



#Perform the simple linear regression model

#Equation: Y=w0+w1.x

#Here Y(marks)=w0+w1.x

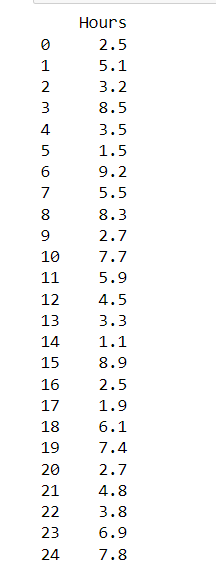
#Create x as hours and Y as marks

X = student.iloc[:, :-1]

y = student.iloc[:, 1]

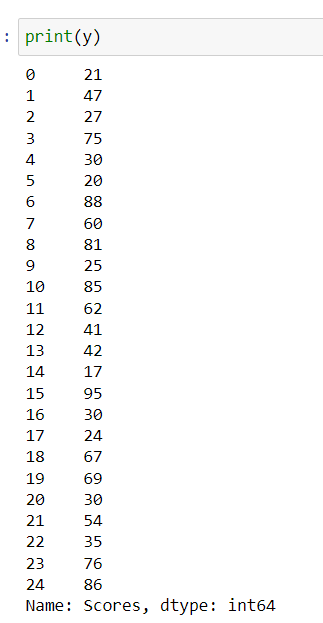
print(X)

**OUTPUT:**



print(y)

**OUTPUT:**

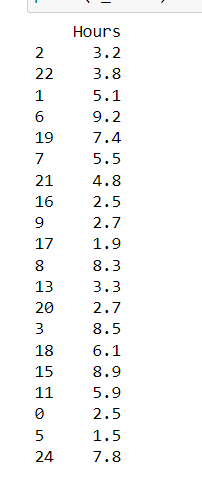


from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

print(X\_train)

**OUTPUT:**



from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

**OUTPUT:**

LinearRegression()

print(regressor.intercept\_)

**OUTPUT:**

3.4157789720979608

print(regressor.coef\_)

**OUTPUT:**

[9.544138]

y\_pred = regressor.predict(X\_test)

for(i,j) in zip(y\_test,y\_pred):

if i!=j:

print("Actual value :",i,"Predicted value :",j)

print("Number of mislabeled points from test data set :", (y\_test != y\_pred).sum())

**OUTPUT:**

Actual value : 76 Predicted value : 69.2703311556657

Actual value : 41 Predicted value : 46.36439996138126

Actual value : 85 Predicted value : 76.90564155376049

Actual value : 17 Predicted value : 13.914330769478324

Actual value : 30 Predicted value : 36.82026196376275

Number of mislabeled points from test data set : 5

from sklearn import metrics

print("Mean Absolute error :", metrics.mean\_absolute\_error(y\_test,y\_pred))

print("Mean Squared error :", metrics.mean\_squared\_error(y\_test,y\_pred))

print("Root Mean Squared error :", np.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

**OUTPUT:**

Mean Absolute error : 6.0188716892478995

Mean Squared error : 39.124239242163426

Root Mean Squared error : 6.254937189306014

import matplotlib.pyplot as plt

c=X\_test['Hours'].count()

xax=np.arange(c)

print(xax)

X\_axis = np.arange(len(xax))

plt.bar(X\_axis-0.2, y\_test, 0.6, label='Actual')

plt.bar(X\_axis+0.2, y\_pred, 0.6, label='Predicted')

plt.xlabel("Test Records")

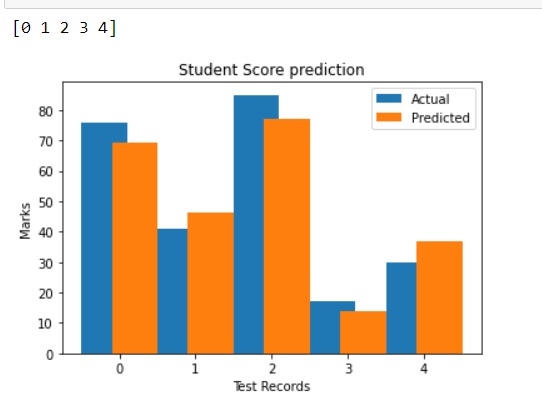
plt.ylabel("Marks")

plt.title("Student Score prediction")

plt.legend()

plt.show()

**OUTPUT:**



**Multiple Linear Regression**