Data Analysis with Python

1.Concept Overview -Python

1.1.variable

Container to store values

Code

```
A=5
```

1.2.Print:

It is a function which used to display

Code:

```
a=5
print("I am",a,"years old")
```

Output:

I am 5 years old

1.3.Operators

1.3.1 Arithmetic operators: These operators are used to perform mathmetical operations on variables.

```
a=5
b=25
print(a+b)
```

Output:

30

```
#power
print(2**5)
```

Output:

32

```
#floor div
print(25//2)
```

Output:

12

1.3.2 Relational operators: These are used to find the realtion between the operands.

```
a=6
b=6
print(a==b)
print(a>b)
print(a<b)
print(a!=b)
print(a>=b)
print(a<=b)
```

```
Output:
True
False
False
False
True
True
```

1.3.3 logical operators: These operators are used to check whether the expression is true or false.

```
a=9
b=25
print((a>b) and (a<b)) #false and true
Output:
False
print((a>b) or (a<b)) #false and true
Output:
True
```

1.3.4 Membership operators: These operators are used to check whether the value or variable exists in a sequence(list,string,tuple,sets,dictionary)

```
a="private"
print("a" in a)
print("a" not in a)
Output:
True
False
```

1.4 Control flow-Conditional statements: These statements are used to check the conditions.

```
#condition if
#write a program to get a number from the user and check whether the number is +ve or -ve
a=int(input("enter a number:"))
print(type(a))
if a>0:
  print(a,"is positive")
else:
  print(a,"is -ve")
```

Output:

```
enter a number:5
<class 'int'>
5 is positive
```

1.4.1 #zero is +ve or -ve

```
a=int(input("enter a value"))
if a>0:
   print(a,"is +ve")
elif a==0:
   print(a,"is neutral")
else:
   print(a,"is -ve")
```

Output:

```
enter a value0
0 is neutral
```

1.5.Control flow-Looping statements

For loop: It is used to repeatedly execute a group of statements as long As the condition is true.

```
n=int(input("enter a value"))
for i in range(1,11,1):
a=n*i
print(a)
```

Output:

```
enter a value6
6
12
18
24
30
36
42
48
54
60
```

While loop: It is used to repeatedly execute a block of code until the Condition is true.

```
a=int(input("enter a value"))
```

```
i=1
while(i<=10):
  n=a*i
  i=i+1
  print(n)
```

Output:

```
enter a value5
5
10
15
20
25
30
35
40
45
50
```

1.6 Data slicing:

It is used for the extraction of a part of a string,list or tuple.

```
a="python is easy"
print(a[::-1])
print(a[0:6])
```

Output:

```
ysae si nohtyp
Python
```

1.6.1

```
a="python is easy"
print(a[::2])
```

Output:

```
pto ses
```

1.7. Type casting:

1.7.1 Implict type casting: When the data type conversion takes place during compilation or during the runtime.

```
a=9.8
b=7
print(a*b) #interpreter automatically changes data into float
```

Output:

68.60000000000001

1.7.2.Explict type casting: Data type conversion performed by the user.

```
a=9.8
b=7
print(int(a*b))
```

Output:

68

1.7.3

```
a="45"
 a=int(a)
 print(type(a))
```

Output:

<class 'int'>

Collections

List:

1. List is a collection of elements
2. Heterogeneous
3. mutable(modifiable)

```
#creating a list:
l1=[3,"valli",90,6.5,7]
print(l1)
```

Output:

[3, 'valli', 90, 6.5, 7]

```
#Display the elements in line by line
 for i in l1:
 print(i)
```

Output:

3

valli

90

6.5

7

```
#append function
l1.append("sailu")
print(l1)
```

Output:

[3, 'valli', 90, 6.5, 7, 'sailu']

#insert function

```
l1.insert(3,"hello")
print(l1)
```

Output:

[3, 'valli', 90, 'hello', 2, 6.5, 7, 'sailu']

#extending the list

```
l2=[5,"cse"]
l1.extend(l2)
print(l1)
```

Output:

[3, 'valli', 90, 'hello', 2, 6.5, 7, 'vyshu','sailu', 5, 'cse']

#pop function

```
l1.pop(2)
print(l1)
```

Output:

[3, 'valli', 2, 6.5, 7, 'vyshu', 'sailu', 5, 'cse']

#remove function

```
l1.remove(2)
print(l1)
```

Output:

[3, 'valli', 6.5, 7, 'vyshu', 'sailu', 5, 'cse']

#max function

```
l3=[12,7,4,13.7]
max(l3)
```

Output:

13.7

List comprehension

1. iterations
2. applies some functions on every element
3. conditions
4. output:list

#squaring the elements in list

```
l4=[12,7,4,13.7]
l5=[i**2 for i in l4]
```

```
print(l5)
```
Output:

[144, 49, 16, 187.68999999999997]

#squaring the elements in the list which is greater than 50

```
l4=[54,7,4,13.7]
l5=[i**2 for i in l4 if i>50]
print(l5)
```
Output:

[2916]

#type of list

```
type(l2)
```
Output:

List

#problem

#the salaries of 5 employees in a company is taken as a list.the tax is 10% if the salary is less than or equal to 50000

#or it is 15%

#create a new list with tax amount

#[67000,45000,89000,34000,50000]

#list_name=[(body of if) if (condition) else (body of else) iterator]

```
sal=[67000,45000,89000,34000,50000]
tax=[]
for i in sal:
  if i<=50000:
    t=i*0.1
    tax.append(t)
  else:
    t=i*0.15
    tax.append(t)
print(tax)
```
Output:

[10050.0, 4500.0, 13350.0, 3400.0, 5000.0]

#list_name=[(body of if) if (condition) else (body of else) iterator]

```
sal=[67000,45000,89000,34000,50000]
tax=[i*0.1 if i<=50000 else i*0.15 for i in sal]
```

```
print(tax)
```

Output:[10050.0, 4500.0, 13350.0, 3400.0, 5000.0]

Numpy:

Numpy is a python library used for working with numerical data in python.

```
#importing
import numpy as np
#creating 1D array
A=np.array([1,2,3,4,5])
print(type(A))
```

Output:

```
<class 'numpy.ndarray'>
#creating 2D array
B=np.array([[2,3,4],[7,8,9]])
print(B)
```

Output:

```
[[2 3 4]
 [7 8 9]]
#creating a 3D array- rows,columns,groups
C=np.array([[[2,3,4],[5,6,7]],[[1,8,9],[0,4,5]]])
print(C)
```

Output:

```
[[[2 3 4]
  [5 6 7]]
 [[1 8 9]
  [0 4 5]]]
#checking dimensions
print(A.ndim)
print(B.ndim)
print(C.ndim)
```

Output:

1

2

3

```
#ones
```

```
D=np.ones((2,3,2)) #groups,rows,columns,3D
print(D)
```

Output:

```
[[[1. 1.]
  [1. 1.]
  [1. 1.]]

 [[1. 1.]
  [1. 1.]
  [1. 1.]]]
```

```
#ones
E=np.ones((3,2))
print(E)
```

Output:

```
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

```
#zeros
F=np.zeros((3,2))
print(F)
```

Output:

```
[[0. 0.]
 [0. 0.]
 [0. 0.]]
J=np.eye(4)
print(J)
```

Output:

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

Arange:The arange function in python is used to create a sequence of Numbers with a specified start,stop and step value.

```
H=np.arange(3,31,3) #start,stop,step
print(H)
```

Output:

```
[ 3  6  9 12 15 18 21 24 27 30]
#arange with reshape
H=np.arange(3,31,3).reshape(5,2)
print(H)
Output:
[[ 3  6]
 [ 9 12]
 [15 18]
 [21 24]
 [27 30]]
```

Linspace:It is a function that generates a sequence of evenly spaced numbers over specified range.

```
V=np.linspace(12,24,10)
print(V)
```

Output:

```
[12.         13.33333333 14.66666667 16.         17.33333333 18.66666667
 20.         21.33333333 22.66666667 24.        ]
```

Reshape:This function is used to reshape an array into a given shape without changing data.

```
L=np.arange(1,7).reshape(2,3)
print(L)
R=np.arange(9,15).reshape(2,3)
print(R)
Output:
[[1 2 3]
 [4 5 6]]
[[ 9 10 11]
 [12 13 14]]
#sum of arrays
print(L+R)
```

Output:

```
[[10 12 14]
 [16 18 20]]
```

#sum function

```
G=np.sum((L,R))
print(G)
```
Output:
```
90
```
#sum function using axis=0
```
G=np.sum((L,R),axis=0)
print(G)
```
Output:
```
[[10 12 14]
 [16 18 20]]
```
#sum function using axis=1
```
G=np.sum((L,R),axis=1)
print(G)
```
Output:
```
[[ 5  7  9]
 [21 23 25]]
```
#linspace : It is used to create an array with equally spaced values between two numbers
```
M=np.linspace(1,2,6)
print(M)
```
Output:
```
[1.  1.2 1.4 1.6 1.8 2. ]
#
h=np.ones((4,2))
g=np.ones((4,2))
print(np.sum((h,g),axis=0))
```
Output:
```
[[2. 2.]
 [2. 2.]
 [2. 2.]
 [2. 2.]]
#product
A=np.array([[1,1],[0,1]])
B=np.array([[2,0],[3,4]])
print(A*B)
```
Output:

```
[[2 0]
 [0 4]]
#product
print(A@B)
```

Output:

```
[[5 4]
 [3 4]]
```

#problem:

```
#b=np.array[25,289,361,81] find square root and iterate through result values output 5 square is 25 by using np.sqrt
b=np.array( [25,289,361,81])
for i in b:
  print(int(np.sqrt(i)),"square is",i)
```

Output:

```
5 square is 25
17 square is 289
19 square is 361
9 square is 81
```

Array joining

```
a=np.array([34,35,36,37,38,39])
a.resize(2,3)
b=np.array([4,5,6,7,8,9])
b.resize(2,3)
print(np.vstack((a,b)))
print("\n")
print(np.hstack((a,b)))
```

Output:

```
[[34 35 36]
 [37 38 39]
 [ 4  5  6]
 [ 7  8  9]]


[[34 35 36  4  5  6]
 [37 38 39  7  8  9]]
```

Dstack:it is used to stack arrays in sequence depth wise.

```
a=np.arange(30).reshape(2,3,5)
print(a)
print("output of dstack")
print(np.dstack(a))
#no.of rows becomes no.of groups
#columns becomes rows
#group becomes columns
```

Output:

```
[[[ 0  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]]]
output of dstack
[[[ 0 15]
  [ 1 16]
  [ 2 17]
  [ 3 18]
  [ 4 19]]

 [[ 5 20]
  [ 6 21]
  [ 7 22]
  [ 8 23]
  [ 9 24]]

 [[10 25]
  [11 26]
  [12 27]
  [13 28]
  [14 29]]]
```

#Random module is subpackage of numpy

Example:

```
a=np.random.rand(1)
print(a)
```

Output:

```
[0.66463289]
```

Example:

```
a=np.random.rand(8,4) #rand-range is between 0 and 1
print(a)
```

Output:

```
[[0.86793957 0.20550871 0.56757905 0.85394866]
 [0.63239592 0.14391372 0.389903   0.34102883]
 [0.25517135 0.36435822 0.07377094 0.02046665]
 [0.24820571 0.67029435 0.5019085  0.31999846]
 [0.45923602 0.92441907 0.1682318  0.37349023]
 [0.77982353 0.11648227 0.9405154  0.18739525]
 [0.82948274 0.7246973  0.55465982 0.3635287 ]
 [0.98206701 0.74445092 0.43170092 0.70714976]]
```

Example:

```
a=10*np.random.rand(8,4)
print(a)
```

Output:

```
[[5.44601317 5.98726441 7.70631605 7.7068417 ]
 [1.67336756 6.97480975 1.03651707 3.82562451]
 [9.89885977 1.23724356 5.43990632 0.21200397]
 [5.88096494 4.2766098  2.78442444 8.19555105]
 [9.99290092 9.3625434  2.55097112 8.71467748]
 [9.88690419 9.827445   1.21206775 6.63063275]
 [7.60806905 5.18914903 3.45710376 6.21997742]
 [9.74972197 4.32029902 2.53673393 8.41650447]]
```

#floor funtion

```
a=np.floor(10*np.random.rand(8,4))
print(a)
```

Output:

```
[[0. 6. 4. 2.]
 [5. 2. 7. 2.]
 [5. 5. 6. 7.]
 [8. 3. 8. 6.]
 [9. 0. 0. 4.]
 [1. 0. 8. 0.]
 [7. 9. 5. 2.]
 [8. 5. 1. 5.]]
```

Example:

```
a=np.arange(1,33).reshape(8,4)
print(a)
```

Output:

```
[[ 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]]
```

Vsplit:

Vsplit function is used to split the array into two subarrays along the vertical axis(rows).

Example-1:

```
np.vsplit(a,4) #here 4 is the no of splits we require
```

Output:

```
[array([[1, 2, 3, 4],
        [5, 6, 7, 8]]),
 array([[ 9, 10, 11, 12],
        [13, 14, 15, 16]]),
 array([[17, 18, 19, 20],
        [21, 22, 23, 24]]),
 array([[25, 26, 27, 28],
        [29, 30, 31, 32]])]
```

Example-2:

```
np.vsplit(a,3)
```

Output:

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-6a62d06336cc> in <cell line: 1>()
----> 1 np.vsplit(a,3)
```

---

3 frames

```
/usr/local/lib/python3.10/dist-packages/numpy/lib/shape_base.py in split(ary,
indices_or_sections, axis)
    870         N = ary.shape[axis]
    871         if N % sections:
--> 872             raise ValueError(
    873                 'array split does not result in an equal division') from None
    874     return array_split(ary, indices_or_sections, axis)

ValueError: array split does not result in an equal division
```

Example-3:

```
np.vsplit(a,(3,5)) #split it after 3rd row and 5th row
```

Output:

```
[array([[ 1,  2,  3,  4],
        [ 5,  6,  7,  8],
        [ 9, 10, 11, 12]]),
 array([[13, 14, 15, 16],
        [17, 18, 19, 20]]),
 array([[21, 22, 23, 24],
        [25, 26, 27, 28],
        [29, 30, 31, 32]])]
```

hsplit:

hsplit function is used to split the array into two subarrays along the horizontal axis(columns).

Example-1:

```
a=np.arange(1,33).reshape(4,8)
print(a)
```

Output:

```
[[ 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]]
```

Example-2:

```
np.hsplit(a,4) #spliting takes place from left to right
#split divide the array into 4 parts
```

Output:

```
[array([[ 1,  2],
        [ 9, 10],
        [17, 18],
        [25, 26]]),
 array([[ 3,  4],
        [11, 12],
        [19, 20],
        [27, 28]]),
 array([[ 5,  6],
        [13, 14],
        [21, 22],
        [29, 30]]),
 array([[ 7,  8],
        [15, 16],
        [23, 24],
        [31, 32]])]
```

Example-3:

```
np.hsplit(a,(3,7)) #spliting takes place after 3rd column and 7th column
```

Output:

```
[array([[ 1,  2,  3],
        [ 9, 10, 11],
        [17, 18, 19],
        [25, 26, 27]]),
```

```
 array([[ 4,  5,  6,  7],
        [12, 13, 14, 15],
        [20, 21, 22, 23],
        [28, 29, 30, 31]]),
 array([[ 8],
        [16],
        [24],
        [32]])]
```

Example-4:

```
np.hsplit(a,(2,5)) #split takes place after 2nd column and 5th column
```

Output:

```
[array([[ 1,  2],
        [ 9, 10],
        [17, 18],
        [25, 26]]),
 array([[ 3,  4,  5],
        [11, 12, 13],
        [19, 20, 21],
        [27, 28, 29]]),
 array([[ 6,  7,  8],
        [14, 15, 16],
        [22, 23, 24],
        [30, 31, 32]])]
```

Trigonometry

Example-1:

```
np.pi
```

Output:

```
3.141592653589793
```

Example-2:

```
A=[np.pi/4,np.pi/3,np.pi/2,np.pi]
print(A)
```

Output:

```
[0.7853981633974483, 1.0471975511965976, 1.5707963267948966, 3.141592653589793]
```

Example-3:

```
B = np.rad2deg(A)
print(B)  #convert radians to degrees
```

Output:

[45. 60. 90. 180.]

Example-4:

```
np.deg2rad(B) #convert degree to radians
```

Output:

array([0.78539816, 1.04719755, 1.57079633, 3.14159265])

Example-5:

```
np.sin(1) #radians
```

Output:

0.8414709848078965

Example-6:

```
np.cos(1)
```

Output:

0.5403023058681398

Example-7:

```
np.tan(1)
```

Output:

1.5574077246549023

Statistics

Example-1:

```
ar=np.array([23,45,67,89,21,34])
np.mean(ar)
```

Output:

46.5

Example-2:

```
ar=np.array([23,45,67,89,21,34])
np.median(ar)
```

Output:

39.5

Example-3:

```
ar=np.array([23,45,67,89,21,34])
np.std(ar)
```

Output:

24.452334585202017

Example-4:

```
ar=np.array([23,45,67,89,21,34])
np.var(ar)
```

Output:

597.9166666666666

Inverse matrix:

```
C=np.arange(1,5).reshape(2,2)
print(C)
```

Output:

```
[[1 2]
 [3 4]]
```

#logic for inverse matrix

```
np.linalg.inv(C)
```

Output:

```
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
```

Max function():It is used to find the maximum element in the matrix.

Example-1:

```
C=np.floor(10*np.random.rand(24)).reshape(6,4)
print(C)
```

Output:

```
[[9. 5. 4. 6.]
 [2. 4. 1. 5.]
 [1. 4. 5. 5.]
 [3. 1. 9. 7.]
 [7. 3. 6. 9.]
 [0. 1. 5. 9.]]
```

Example for find max element

```
print(np.argmax(C))
```

Output:

0

Example-2:

```
c=10*np.random.rand(24).reshape(6,4)
print(c)
```

Output:

```
[[9.15519817 8.39266107 0.76805167 8.43663521]
```

```
[9.28316067 5.63640124 5.04580318 9.51054732]
 [0.71929046 4.36124117 7.86106573 6.51879388]
 [1.16087787 9.53980816 8.89155739 0.36573456]
 [4.67095163 8.77191641 6.75421363 9.81088343]
 [4.69871256 1.8691479  0.13198657 0.63774885]]
```

Example for find max element

```
print(np.argmax(c))
```

Output:

19

Example for finding max element in rows:

```
print(np.argmax(c,axis=1))
```

Output:

[0 3 2 1 3 0]

Example for finding max element in columns:

```
print(np.argmax(c,axis=0))
```

Output:

[1 3 3 4]

Example for find the minimum element:

```
print(np.argmin(c))
```

Output:

22

Example for finding min element in columns:

```
print(np.argmin(c,axis=0))
```

Output:

[2 5 5 3]

Example for finding min element in rows:

```
print(np.argmin(c,axis=1))
```

Output:

[2 2 0 3 0 2]

Search

Example-1:

```
a=np.array([34,56,7,17,88,91])
print(np.where(a%2==0))
print(np.where(a%2==1))
```

Output:

(array([0, 1, 4]),)

(array([2, 3, 5]),)

Example-2:

```
a=np.array([24,16,7,17,54,60])
print(np.where(a%6==0))
```

Output:

(array([0, 4, 5]),)

Example-3:

```
a=np.array([2,4,7,3,6]) #searchsort
x=np.searchsorted(a,3)
print(x)
```

Output:

1

Example-4:

```
a=np.array([2,4,6,7,8])
x=np.searchsorted(a,7)
print(x)
```

Output:

3

Example-5:

```
a=np.array(['banana','apple','cherry']) #sorting
print(np.sort(a))
```

Output:

['apple' 'banana' 'cherry']

Example-6:

```
a=np.array(['true','false','true'])
print(np.sort(a))
```

Output:

['false' 'true' 'true']

Example-7:

```
b=np.array([[3,2,4],[5,0,1]])
print(np.sort(b))
```

Output:

[[2 3 4]
 [0 1 5]]

Filtering:

filter() function is to process an iterable and extract those items that satisfy a given condition.

Example-1:

```
b=np.array([40,43,50,44,67,78])
flit=np.where(b%2==0)          #flit may be list or array
print(flit)
```

Output:

```
(array([0, 2, 3, 5]),)
b[flit]
```

Output:

```
array([40, 50, 44, 78])
```

Iterating through two arrays:

Example-1:

```
names=np.array(["valli","sudha","susmitha","priya"])
initials=np.array(["p","m","u","s"])
for i,j in zip(initials,names):
  print(i,".",j)
```

Output:

```
p . valli
m . sudha
u . susmitha
s . priya
```

Example-2:

```
a=np.array([10,20,30,40,50,60])
b=np.array([20,21,22,23,24,25])
n=np.multiply(a,b)     #multiplication
print(n)
```

Output:

```
[ 200  420  660  920 1200 1500]
```

Example for divide two arrays:

```
a=np.array([10,20,30,40,50,60])
b=np.array([20,21,22,23,24,25])
print(np.divide(a,b))
```

Output:

```
[0.5        0.95238095 1.36363636 1.73913043 2.08333333 2.4        ]
```

Example for modulo division on two arrays:

```
a=np.array([10,20,30,40,50,60])
b=np.array([20,21,22,23,24,25])
print(np.mod(a,b))
```

Output:

```
[10 20  8 17  2 10]
```

Example:

```
a=np.array([10,20,30,40,50,60])
b=np.array([20,21,22,23,24,25])
print(np.divmod(a,b))
```

Output:

```
(array([0, 0, 1, 1, 2, 2]), array([10, 20,  8, 17,  2, 10]))
```

Logarithms

Example-1:

```
a1=np.arange(1,10)
print(a1)
print(np.log2(a1))
```

Output:

```
[1 2 3 4 5 6 7 8 9]
[0.         1.         1.5849625  2.         2.32192809 2.5849625
 2.80735492 3.         3.169925  ]
```

Example-2:

```
a=1.2
print(np.log(a)) #natural log e
```

Output:

```
0.1823215567939546
```

Example-3:

```
a=1.2
print(np.log2(a)) #log base 2
```

Output:

```
0.2630344058337938
```

Example-4:

```
a=1.2
print(np.log10(a)) #log base 10
```

Output:

```
0.07918124604762482
```

Example-5:

```
a=np.array([1,1.2,3,4])
print(np.log(a)) #natural log e
```

Output:

```
[0.         0.18232156 1.09861229 1.38629436]
```

Other mathmetical functions

Example-1:

```
c=np.array([5,6,3,8,])
print(np.cumprod(c))
```

Output:

```
[  5  30  90 720]
```

Example-2:

```
c=np.array([5,6,3,8])
print(np.sum(c))
```

Output:

22

Example-3:

```
a=np.array([10,15,25,15])
n=np.diff(a)
print(n)
```

Output:

```
[  5  10 -10]
```

Example-4:

```
a=455
b=665
print(np.lcm(a,b))
```

Output:

8645

Example-5:

```
a=455
b=665
print(np.gcd(a,b))
```

Output:

35

Example-6:

```
a=np.array([12,15,60])
gc=np.gcd(a)
print(gc)
```

Output:

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-99-9784b52f7527> in <cell line: 2>()
      1 a=np.array([12,15,60])
----> 2 gc=np.gcd(a)
      3 print(gc)
TypeError: gcd() takes from 2 to 3 positional arguments but 1 were given
```

Example-7:

```
a=np.array([12,15,60])
gc=np.gcd.reduce(a) #takes multiple inputs and gives single  output
print(gc)
```

Output:

3

Example-8:

```
a=np.array([12,15,60])
lc=np.lcm.reduce(a)
print(lc)
```

Output:

60

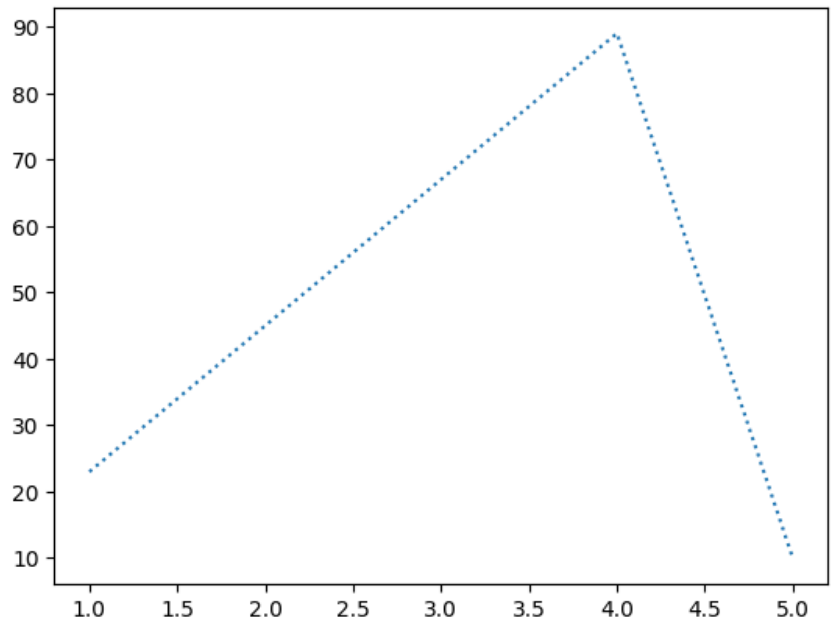
Metaplotlib:

Matplotlib is a cross_platform,data visualization and graphical ploting library for python and its numerical extension numpy.

Example-1:

```
a=[23,45,67,89,10]  #corona cases in first five days
b=[1,2,3,4,5]
plt.plot(b,a,linestyle=":") #plot (x,y) #
plt.show()
```

Output:

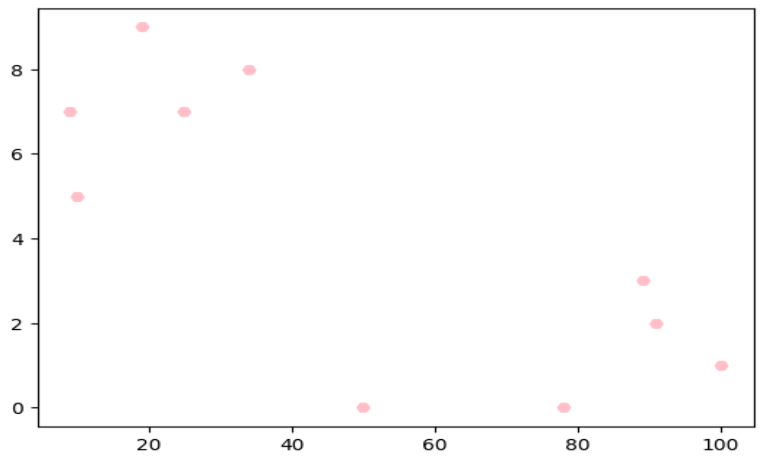


Example-2:

Runs scored by 10 players [100,50,91,78,89,25,34,19,9,10] wickets taken by same 10 new players [1,0,2,0,3,7,8,9,7,5] from the clusters.

```
a=[100,50,91,78,89,25,34,19,9,10]
b=[1,0,2,0,3,7,8,9,7,5]
plt.scatter(a,b,color="pink",marker="H")
plt.show()
```

Output:

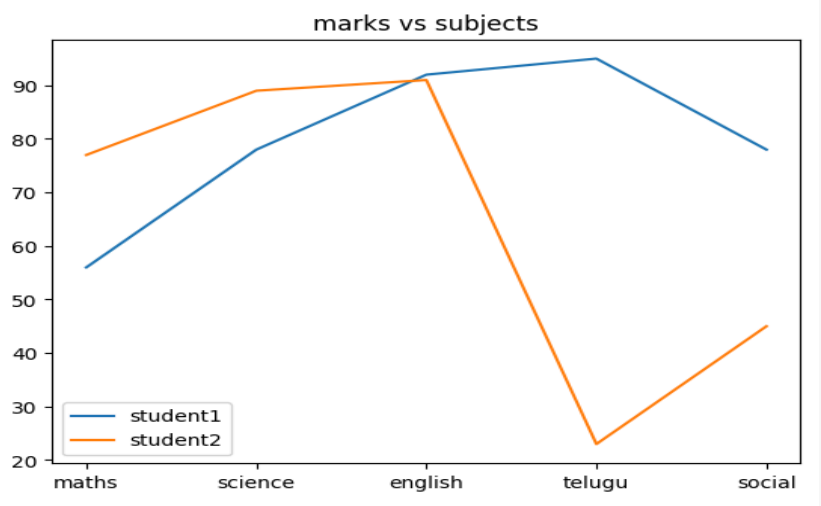


Example-3:

plot the scores of 2 students in 5 different subjects. subjects as x axis and marks as y axis.

```
stu1=[56,78,92,95,78]
stu2=[77,89,91,23,45]
sub=["maths","science","english","telugu","social"]
plt.plot(sub,stu1,label="student1")
plt.plot(sub,stu2,label="student2")
plt.title("marks vs subjects")
plt.legend()
```

Output:



subplot (rows,columns,position)

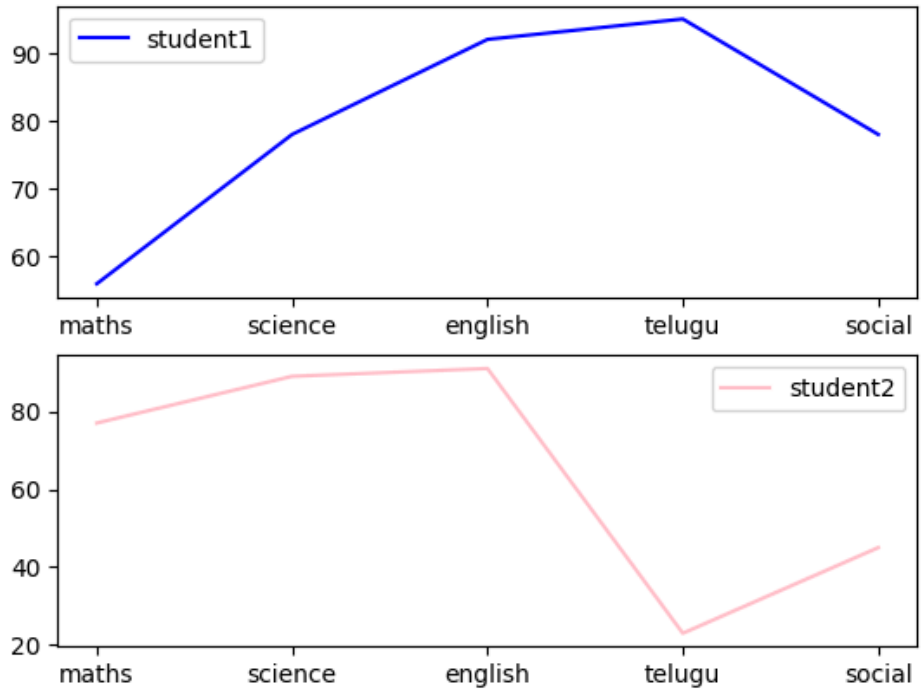
Example:

```
stu1=[56,78,92,95,78]
stu2=[77,89,91,23,45]
sub=["maths","science","english","telugu","social"]
plt.subplot(2,1,1)
plt.plot(sub,stu1,label="student1",color="b")
plt.legend()
plt.subplot(2,1,2)
```

plt.plot(sub,stu2,label="student2",color="pink")

plt.legend()

Output:



Example-2:

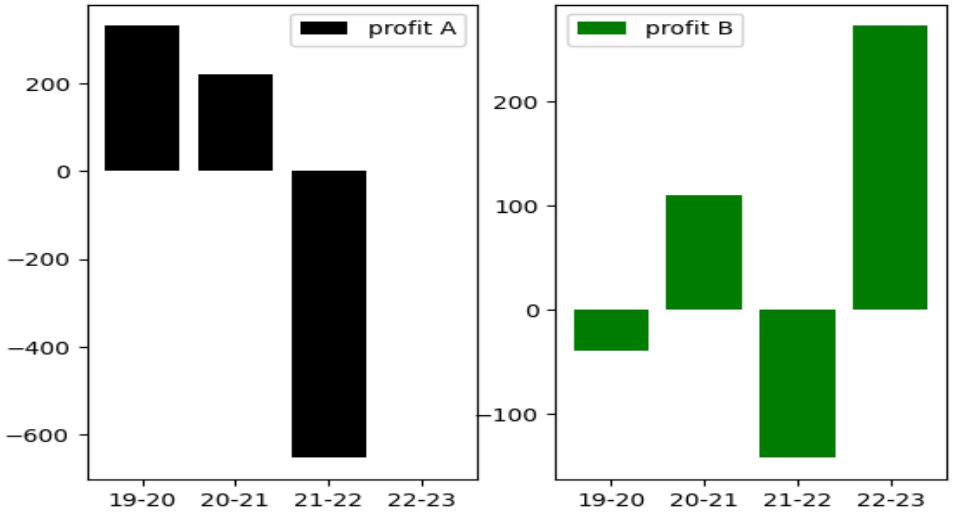
```
A=[230,560,780,127,128]
B=[200,160,270,127,400]
print("revenue of A",A)
print("revenue of B",B)
profA=np.diff(A)
print("profits of A",profA)
profB=np.diff(B)
print("profits of B",profB)
years=["19-20","20-21","21-22","22-23"]
plt.subplot(1,2,1)
plt.bar(years,profA,color="k",label="profit A")
```

```
plt.legend(loc = "best")
plt.subplot(1,2,2)
plt.bar(years,profB,color="g",label="profit B")
plt.legend(loc = "best")
```

Output:

```
revenue of A [230, 560, 780, 127, 128]
revenue of B [200, 160, 270, 127, 400]
profits of A [ 330  220 -653    1]
profits of B [ -40  110 -143  273]
```

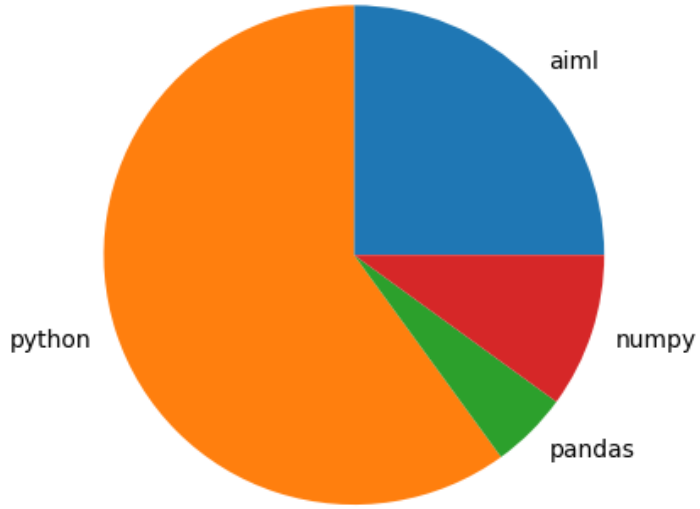
<matplotlib.legend.Legend at 0x78b0fc5d6cb0>



Piechart:

```
a=np.array([25,60,5,10])
v=["aiml","python","pandas","numpy"]
plt.pie(a,labels=v)
plt.show()
```

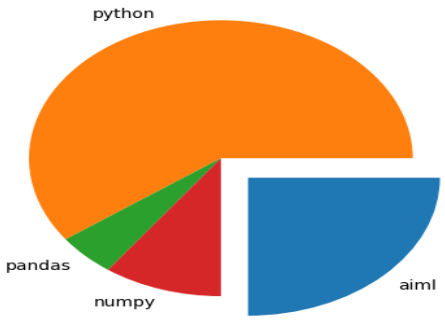
Output:



Explode:

```
a=np.array([25,60,5,10])
v=["aiml","python","pandas","numpy"]
explo=[0.2,0,0,0]
plt.pie(a,labels=v,explode=explo,startangle=270)
plt.show()
```

Output:



Example:

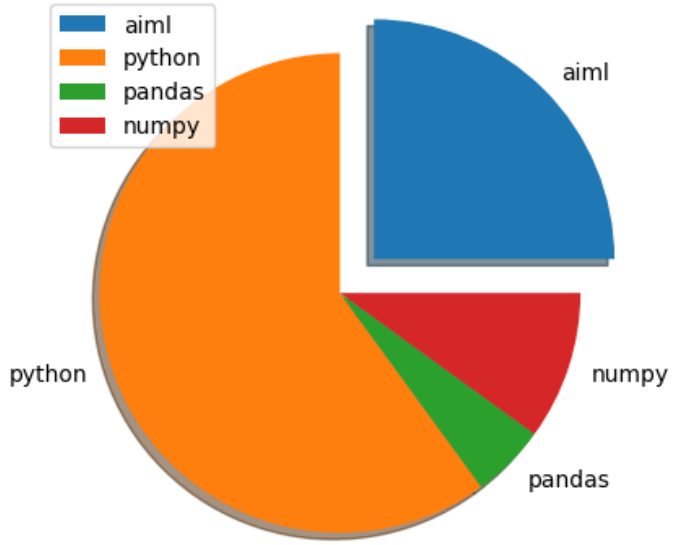
```
a=np.array([25,60,5,10])
v=["aiml","python","pandas","numpy"]
explo=[0.2,0,0,0]
```

```
plt.pie(a,labels=v,explode=explo,shadow=True)

plt.legend()

plt.show()
```

Output:



Pandas:

1. used for data manipulation (data cleaning, organizing data)
2. creates data frames from excel,csv,txt,DBs
3. Dataframe (rows and columns readable by python)
4. data cleaning by dropping or replacing with mean
5. Visualize the data

Importing the pandas

```
import pandas as pd
```

Example:

```
names=['valli','priya','sasi','lavanya','fathima']

index=[40,42,42,43,45]

seri=pd.Series(names,index)
```

```
print(seri)
```

Output:

```
40      valli
42      priya
42       sasi
43    lavanya
45    fathima
dtype: object
```

Importing files

- for csv and txt: read_csv
- for excel: read_excel

Example for reading the csv file

```
df=pd.read_csv("/content/census2011.csv")
df.head(10)
```

Output:

	Ranking	District	State	Population	Growth	Sex-Ratio	Literacy
0	1	Thane	Maharashtra	11,060,148	36.01%	886	84.53
1	2	North Twenty Four Parganas	West Bengal	10,009,781	12.04%	955	84.06
2	3	Bangalore	Karnataka	9,621,551	47.18%	916	87.67

3	4	Pune	Maharashtra	9,429,408	30.37 %	915	86.15
4	5	Mumbai Suburban	Maharashtra	9,356,962	8.29 %	860	89.91
5	6	South Twenty Four Parganas	West Bengal	8,161,961	18.17 %	956	77.51
6	7	Barddhaman	West Bengal	7,717,563	11.92 %	945	76.21
7	8	Ahmadabad	Gujarat	7,214,225	24.03 %	904	85.31
8	9	Murshidabad	West Bengal	7,103,807	21.09 %	958	66.59
9	10	Jaipur	Rajasthan	6,626,178	26.19 %	910	75.51

```
df.tail(10)
```

Output:

```
df.tail(10)
```

	Ranking	District	State	Population	Growth	Sex-Ratio	Literacy
600	631	Diu	Daman and Diu	52,074	17.77 %	1031	83.46
601	632	Longleng	Nagaland	50,484	-21.57 %	905	72.17
602	633	Tawang	Arunachal Pradesh	49,977	28.40 %	714	59.00
603	634	North Sikkim	Sikkim	43,709	6.53 %	767	78.01
604	635	Mahe	Puducherry	41,816	13.54 %	1184	97.87
605	636	Nicobars	Andaman and Nicobar Islands	36,842	-12.42 %	777	78.06
606	637	Upper Siang	Arunachal Pradesh	35,320	5.87 %	889	59.99
607	638	Lahul and Spiti	Himachal Pradesh	31,564	-5.00 %	903	76.81
608	639	Anjaw	Arunachal Pradesh	21,167	14.19 %	839	56.46
609	640	Dibang Valley	Arunachal Pradesh	8,004	10.07 %	813	64.10

Example for reading for text file

```
dft=pd.read_csv("/content/grades.txt",sep=" ")

dft.head()
```

Output:

	Names	Initials	SEM 1	SEM 2	SEM 3	Grade
0	Joe	K	9.8	10.0	9.9	A+
1	Rajesh	M	8.9	9.1	9.3	A
2	Kissan	V	9.9	9.3	9.2	A
3	Mary	N	7.7	8.0	7.1	B
4	Jeen	K	9.8	9.1	9.9	A+

Example for reading excel file

```
dfe=pd.read_excel("/content/census2011.xlsx")

dfe.head()
```

Output:

	Ranking	District	State	Population	Growth	Sex-Ratio	Literacy
0	1	Thane	Maharashtra	11,060,148	36.01 %	886	84.53
1	2	North Twenty Four Parganas	West Bengal	10,009,781	12.04 %	955	84.06
2	3	Bangalore	Karnataka	9,621,551	47.18 %	916	87.67
3	4	Pune	Maharashtra	9,429,408	30.37 %	915	86.15
4	5	Mumbai Suburban	Maharashtra	9,356,962	8.29 %	860	89.91

Example for describing the file

```
dft=pd.read_csv("/content/grades.txt")

print(df.describe)
```

Output:

```
<bound method NDFrame.describe of      Ranking                  District          State  \
0          1                    Thane    Maharashtra
1          2  North Twenty Four Parganas    West Bengal
```

2	3	Bangalore	Karnataka
3	4	Pune	Maharashtra
4	5	Mumbai Suburban	Maharashtra
..	...	...	...
605	636	Nicobars	Andaman and Nicobar Islands
606	637	Upper Siang	Arunachal Pradesh
607	638	Lahul and Spiti	Himachal Pradesh
608	639	Anjaw	Arunachal Pradesh
609	640	Dibang Valley	Arunachal Pradesh

	Population	Growth	Sex-Ratio	Literacy
0	11,060,148	36.01 %	886	84.53
1	10,009,781	12.04 %	955	84.06
2	9,621,551	47.18 %	916	87.67
3	9,429,408	30.37 %	915	86.15
4	9,356,962	8.29 %	860	89.91
..	...	...	...	...
605	36,842	-12.42 %	777	78.06
606	35,320	5.87 %	889	59.99
607	31,564	-5.00 %	903	76.81
608	21,167	14.19 %	839	56.46
609	8,004	10.07 %	813	64.10

[610 rows x 7 columns]>

Example for shape of file

```
df=pd.read_csv("/content/census2011.csv")
print(df.shape) #shape of rows and columns
print(df.shape[0]) #display shape of rows
print(df.shape[1]) #diplay shape of columns
```

Output:

```
(610, 7)
610
7
```

Example:

```
print(df.columns)
```

Output:

```
Index(['Ranking', 'District', 'State', 'Population', 'Growth', 'Sex-Ratio',
       'Literacy'],
```

Accessing Data:

- loc-accepts column names and index
- iloc-accepts only index

Example to access rows:

```
print(df[2:5])
```

Output:

	Ranking	District	State	Population	Growth	Sex-Ratio	\
2	3	Bangalore	Karnataka	9,621,551	47.18 %	916	
3	4	Pune	Maharashtra	9,429,408	30.37 %	915	
4	5	Mumbai Suburban	Maharashtra	9,356,962	8.29 %	860	

Literacy

2 87.67

3 86.15

4 89.91

Example for rows of specified columns

```
print(dft.loc[2:5,"Names"])
```

Output:

2 Kissan

3 Mary

4 Jeen

5 Raj

Name: Names, dtype: object

Example for iloc:

```
print(dft.iloc[2:5,:3])
```

Output:

Names Initials SEM1

2 Kissan V 9.9

3 Mary N 7.7

4 Jeen K 9.8

Example:

```
dfn=pd.read_csv("/content/grades_withnulls.csv")
dfn.head()
```

Output:

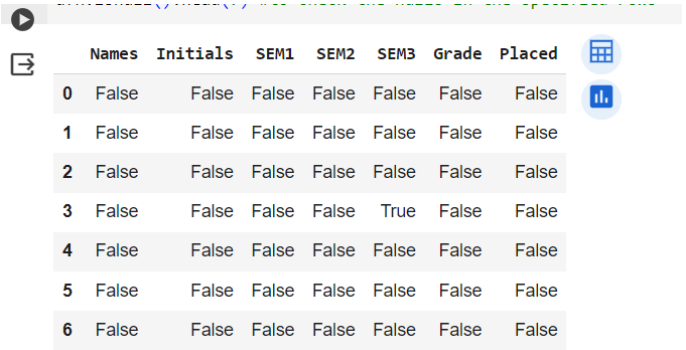
	Names	Initials	SEM1	SEM2	SEM3	Grade	Placed
0	Joe	K	9.8	10.0	9.9	A+	1
1	Rajesh	M	8.9	9.1	9.3	A	1

2	Kissan	V	9.9	9.8	10.0	A	0
3	Mary	N	7.7	8.0	NaN	B	0
4	Jeen	K	9.8	9.1	9.9	A+	1

Example for null function:

```
dfn.isnull().head(7)  #to check the nulls in the specified rows
```

Output:



	Names	Initials	SEM1	SEM2	SEM3	Grade	Placed
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	True	False	False
4	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False

Example for sum of nulls:

```
dfn.isnull().sum()
```

Output:

```
Names       0
Initials    0
SEM1        3
SEM2        0
SEM3        1
Grade       0
Placed      0
dtype: int64
```

Example to view total nulls

```
dfn.isnull().sum().sum()
```

Output:

4

Example for dropping the nulls

```
dfc=dfn.dropna()
```

Dfc

Output:

```
dfc
```

	Names	Initials	SEM1	SEM2	SEM3	Grade	Placed
0	Joe	K	9.8	10.0	9.9	A+	1
1	Rajesh	M	8.9	9.1	9.3	A	1
2	Kissan	V	9.9	9.8	10.0	A	0
4	Jeen	K	9.8	9.1	9.9	A+	1
5	Raj	M	8.9	9.1	9.3	A	1
6	Hassan	V	9.9	9.0	9.2	A	1
7	Mari	N	7.7	8.0	7.1	B	1
11	Maya	N	7.7	8.0	7.1	B	0
12	Jolin	K	9.8	9.1	9.9	A+	1
13	Rajesh	M	8.9	9.1	9.3	A	1
14	Riya	M	9.3	9.9	10.0	A	1
15	Sana	V	9.9	9.3	9.2	A	0
16	Mark	N	7.7	8.0	7.0	B	0

Example for saving the changes:

```
dfc1=dfn.fillna(5)
```

dfc1

Output:

	Names	Initials	SEM1	SEM2	SEM3	Grade	Placed
0	Joe	K	9.8	10.0	9.9	A+	1
1	Rajesh	M	8.9	9.1	9.3	A	1
2	Kissan	V	9.9	9.8	10.0	A	0
3	Mary	N	7.7	8.0	5.0	B	0
4	Jeen	K	9.8	9.1	9.9	A+	1
5	Raj	M	8.9	9.1	9.3	A	1
6	Hassan	V	9.9	9.0	9.2	A	1
7	Mari	N	7.7	8.0	7.1	B	1
8	Jess	K	5.0	9.1	9.9	A+	1
9	Rajini	M	5.0	9.1	9.3	A	0
10	Kiran	V	5.0	9.3	9.2	A	0
11	Maya	N	7.7	8.0	7.1	B	0
12	Jolin	K	9.8	9.1	9.9	A+	1
13	Rajesh	M	8.9	9.1	9.3	A	1
14	Riya	M	9.3	9.9	10.0	A	1
15	Sana	V	9.9	9.3	9.2	A	0
16	Mark	N	7.7	8.0	7.0	B	0

Cleaning with mean

Example of mean:

```
m=dfn['SEM3'].mean()

print(m)
```

Output:

```
9.100000000000001
```

Example:

```
dfc2=dfn.fillna(m)

dfc2
```

Output:

	Names	Initials	SEM1	SEM2	SEM3	Grade	Placed
0	Joe	K	9.8	10.0	9.9	A+	1
1	Rajesh	M	8.9	9.1	9.3	A	1
2	Kissan	V	9.9	9.8	10.0	A	0
3	Mary	N	7.7	8.0	9.1	B	0
4	Jeen	K	9.8	9.1	9.9	A+	1
5	Raj	M	8.9	9.1	9.3	A	1
6	Hassan	V	9.9	9.0	9.2	A	1
7	Mari	N	7.7	8.0	7.1	B	1
8	Jess	K	9.1	9.1	9.9	A+	1
9	Rajini	M	9.1	9.1	9.3	A	0
10	Kiran	V	9.1	9.3	9.2	A	0
11	Maya	N	7.7	8.0	7.1	B	0
12	Jolin	K	9.8	9.1	9.9	A+	1
13	Rajesh	M	8.9	9.1	9.3	A	1
14	Riya	M	9.3	9.9	10.0	A	1
15	Sana	V	9.9	9.3	9.2	A	0
16	Mark	N	7.7	8.0	7.0	B	0

Example for removing the duplicates:

```
dropped=dfc2.drop_duplicates()

print(dropped)
```

Output:

```
     Names Initials  SEM1  SEM2  SEM3 Grade  Placed
0      Joe        K   9.8  10.0   9.9    A+       1
1   Rajesh        M   8.9   9.1   9.3     A       1
2   Kissan        V   9.9   9.8  10.0     A       0
3     Mary        N   7.7   8.0   9.1     B       0
4     Jeen        K   9.8   9.1   9.9    A+       1
5      Raj        M   8.9   9.1   9.3     A       1
6   Hassan        V   9.9   9.0   9.2     A       1
7     Mari        N   7.7   8.0   7.1     B       1
8     Jess        K   9.1   9.1   9.9    A+       1
9   Rajini        M   9.1   9.1   9.3     A       0
10   Kiran        V   9.1   9.3   9.2     A       0
11    Maya        N   7.7   8.0   7.1     B       0
12   Jolin        K   9.8   9.1   9.9    A+       1
14    Riya        M   9.3   9.9  10.0     A       1
15    Sana        V   9.9   9.3   9.2     A       0
16    Mark        N   7.7   8.0   7.0     B       0
```

Columns

Example for rename the column:

```
dfc2.rename(columns={"Grade":"GPA"}) #df.rename(columns={old:new})
```

Output:

	Names	Initials	SEM1	SEM2	SEM3	GPA	Placed
0	Joe	K	9.8	10.0	9.9	A+	1
1	Rajesh	M	8.9	9.1	9.3	A	1
2	Kissan	V	9.9	9.8	10.0	A	0
3	Mary	N	7.7	8.0	9.1	B	0
4	Jeen	K	9.8	9.1	9.9	A+	1
5	Raj	M	8.9	9.1	9.3	A	1
6	Hassan	V	9.9	9.0	9.2	A	1
7	Mari	N	7.7	8.0	7.1	B	1
8	Jess	K	9.1	9.1	9.9	A+	1
9	Rajini	M	9.1	9.1	9.3	A	0
10	Kiran	V	9.1	9.3	9.2	A	0
11	Maya	N	7.7	8.0	7.1	B	0
12	Jolin	K	9.8	9.1	9.9	A+	1
13	Rajesh	M	8.9	9.1	9.3	A	1
14	Riya	M	9.3	9.9	10.0	A	1
15	Sana	V	9.9	9.3	9.2	A	0
16	Mark	N	7.7	8.0	7.0	B	0

Example for rename the column permentently:

```
dfc2.rename(columns={"Grade":"GPA"},inplace=True)

dfc2.head()
```

Output:

	Names	Initials	SEM1	SEM2	SEM3	GPA	Placed
0	Joe	K	9.8	10.0	9.9	A+	1
1	Rajesh	M	8.9	9.1	9.3	A	1
2	Kissan	V	9.9	9.8	10.0	A	0
3	Mary	N	7.7	8.0	9.1	B	0
4	Jeen	K	9.8	9.1	9.9	A+	1

Example for average of columns:

```
dfc2['Avg_score']=(dfc2['SEM1']+dfc2['SEM2']+dfc2['SEM3'])/3 #df['newcol']=values

dfc2.head()
```

Output:

	Names	Initials	SEM1	SEM2	SEM3	GPA	Placed	Avg_score
0	Joe	K	9.8	10.0	9.9	A+	1	9.900000
1	Rajesh	M	8.9	9.1	9.3	A	1	9.100000
2	Kissan	V	9.9	9.8	10.0	A	0	9.900000
3	Mary	N	7.7	8.0	9.1	B	0	8.266667
4	Jeen	K	9.8	9.1	9.9	A+	1	9.600000

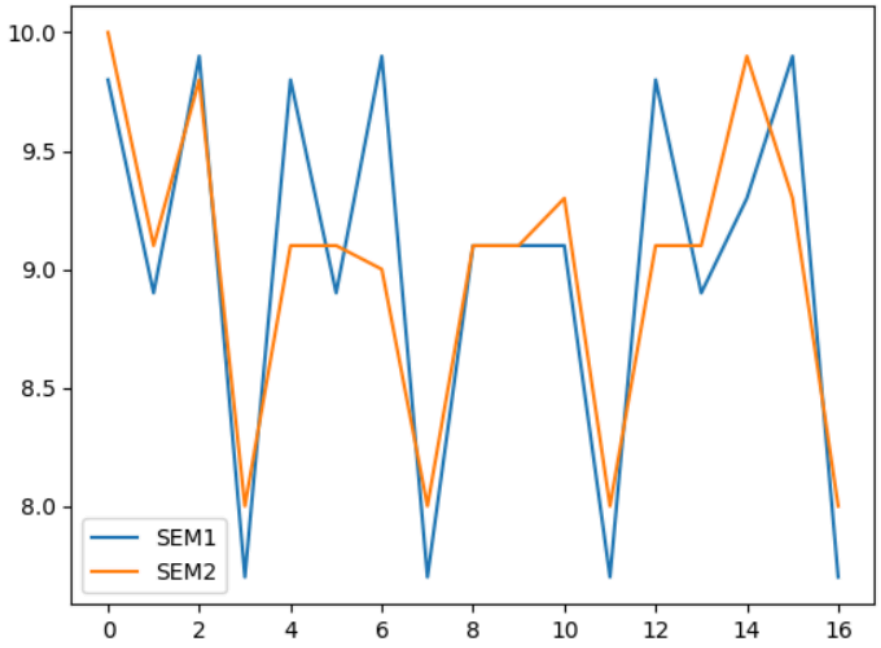
Plotting:

Example:

```
dfc2[['SEM1','SEM2']].plot.line()
```

Output:

<Axes: >

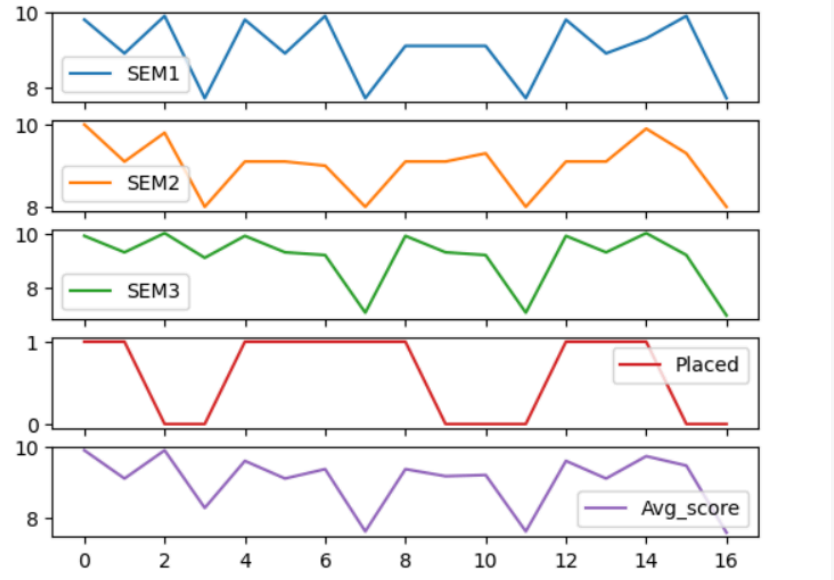


Example for subplots:

```
dfc2.plot.line(subplots=True)
```

Output:

```
array([<Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >], dtype=object)
```



Seaborn:

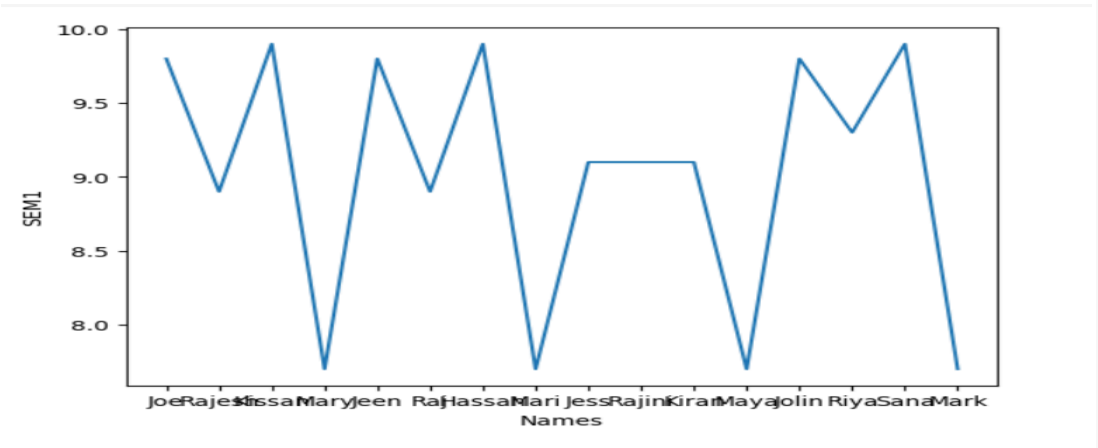
Seaborn is a library mostly used for statistical plotting in Python.

```
import seaborn as sns
```

Example for lineplot:

```
pl=sns.lineplot(x="Names",y="SEM1",data=dfc2)
```

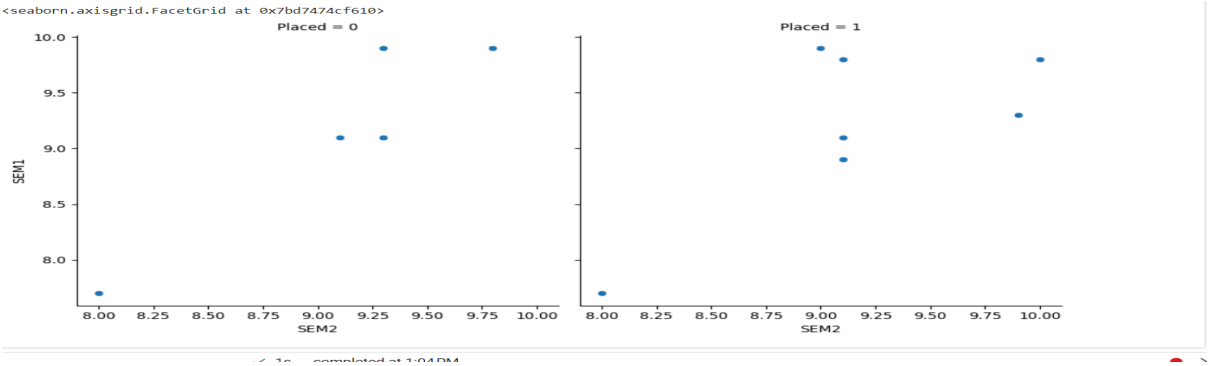
Output:



Example for relplot:

```
sns.relplot(
    data=dfc2,
    x="SEM2",y="SEM1",col="Placed"
)
```

Output:



Problem:

Load diabets.csv create a relplot with age in the x axis and class as columns.

```
dfd=pd.read_csv("/content/diabetcsv.csv")

dfd.head()
```

Output:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive

Dfd

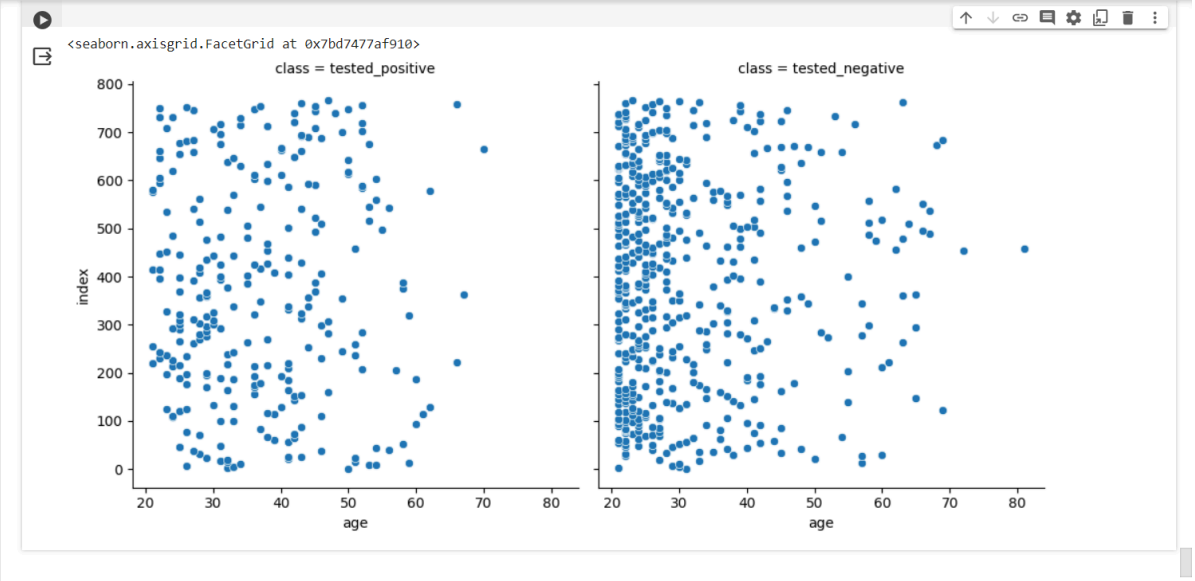
Output:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	tested_negative
764	2	122	70	27	0	36.8	0.340	27	tested_negative
765	5	121	72	23	112	26.2	0.245	30	tested_negative
766	1	126	60	0	0	30.1	0.349	47	tested_positive
767	1	93	70	31	0	30.4	0.315	23	tested_negative

768 rows × 9 columns

```
dfd['index']=(range(0,768))
sns.relplot(
   data=dfd,
   x="age",y="index",col="class"
)
```

Output:



Seaborn inbuilt datasets

1. tips
2. dowjones
3. fmri
4. dots
5. healthexp

To load dataset uses

=>load_dataset("dataset_name")

Example-1:

tips=sns.load_dataset("tips")

Output:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Example-2:

```
fmri=sns.load_dataset("fmri")
fmri.head()
```

Output:

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

Example-3:

```
dowjones=sns.load_dataset("dowjones")
dowjones.head()
```

Output:

	Date	Price
0	1914-12-01	55.00
1	1915-01-01	56.55
2	1915-02-01	56.00
3	1915-03-01	58.30
4	1915-04-01	66.45

Example-4:

```
dots=sns.load_dataset("dots")
dots.head()
```

Output:

	align	choice	time	coherence	firing_rate	
0	dots	T1	-80	0.0	33.189967	
1	dots	T1	-80	3.2	31.691726	
2	dots	T1	-80	6.4	34.279840	
3	dots	T1	-80	12.8	32.631874	
4	dots	T1	-80	25.6	35.060487	

Example-5:

```
healthexp=sns.load_dataset("healthexp")
healthexp.head()
```

Output:

	Year	Country	Spending_USD	Life_Expectancy
0	1970	Germany	252.311	70.6
1	1970	France	192.143	72.2
2	1970	Great Britain	123.993	71.9
3	1970	Japan	150.437	72.0
4	1970	USA	326.961	70.9

```
tips.head()
```

Output:

	Year	Country	Spending_USD	Life_Expectancy
0	1970	Germany	252.311	70.6
1	1970	France	192.143	72.2
2	1970	Great Britain	123.993	71.9
3	1970	Japan	150.437	72.0
4	1970	USA	326.961	70.9

Example-6:

```
tips=sns.load_dataset("tips")
tips.head()
```

Output:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Catplot:

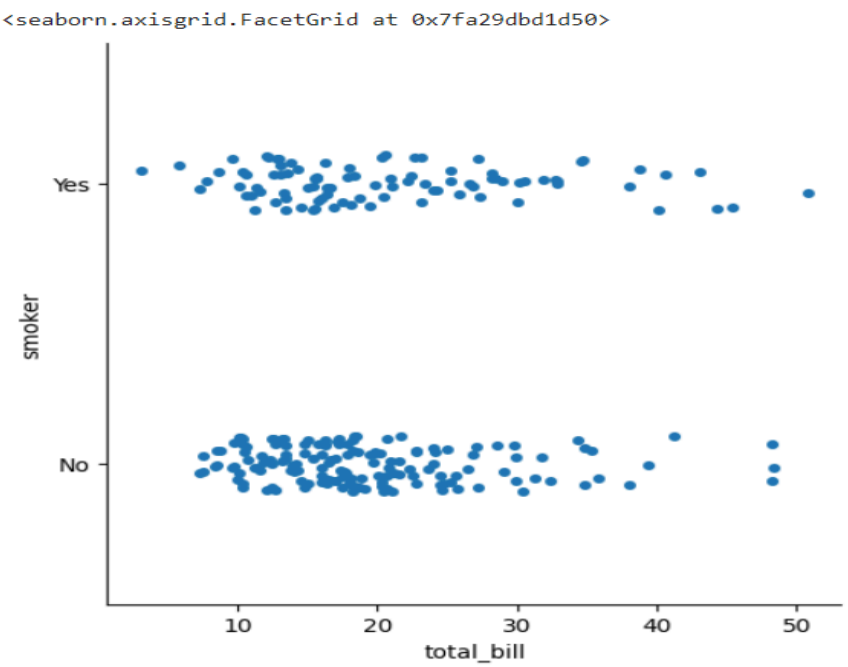
The catplot function in the python is used to create categorical Plots.

Example:

```
sns.catplot(data=tips,x="total_bill",y="smoker")
```

Output:

```
<seaborn.axisgrid.FacetGrid at 0x7fa29dbd1d50>
```



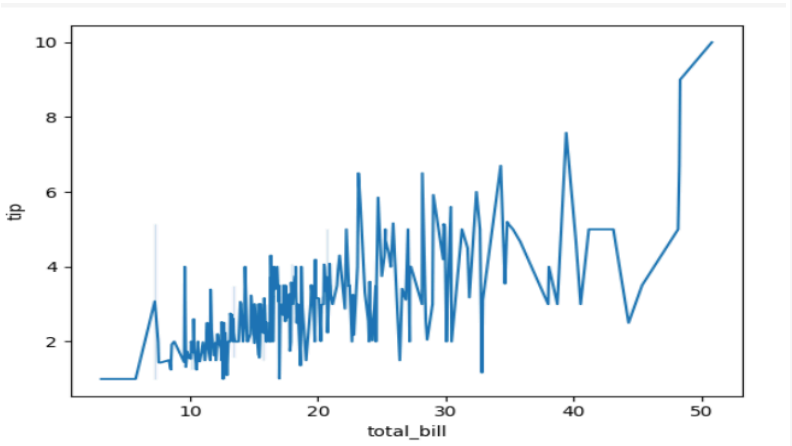
Lineplot:

A line plot in seaborn can be created using the lineplot() or relplot().Lineolot()function is used when you want to create a single line plot,while the replot() functions used when you want to create a line plot with multiple lines on different facets.

Example-1:

```
pl=sns.lineplot(x='total_bill',y='tip',data=tips)
```

Output:

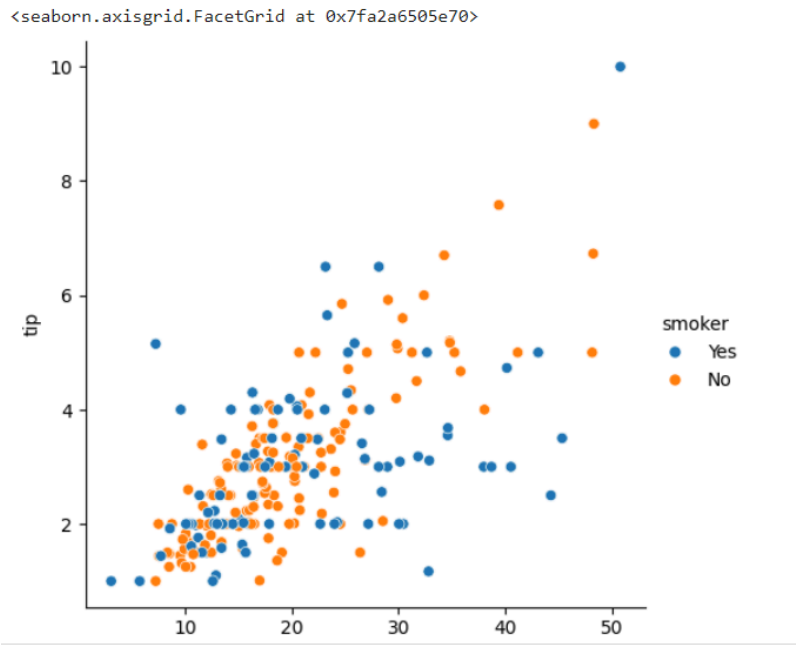


Example-2:

```
sns.relplot(data=tips,x='total_bill',y='tip',hue='smoker')
```

Output:

```
<seaborn.axisgrid.FacetGrid at 0x7fa2a6505e70>
```



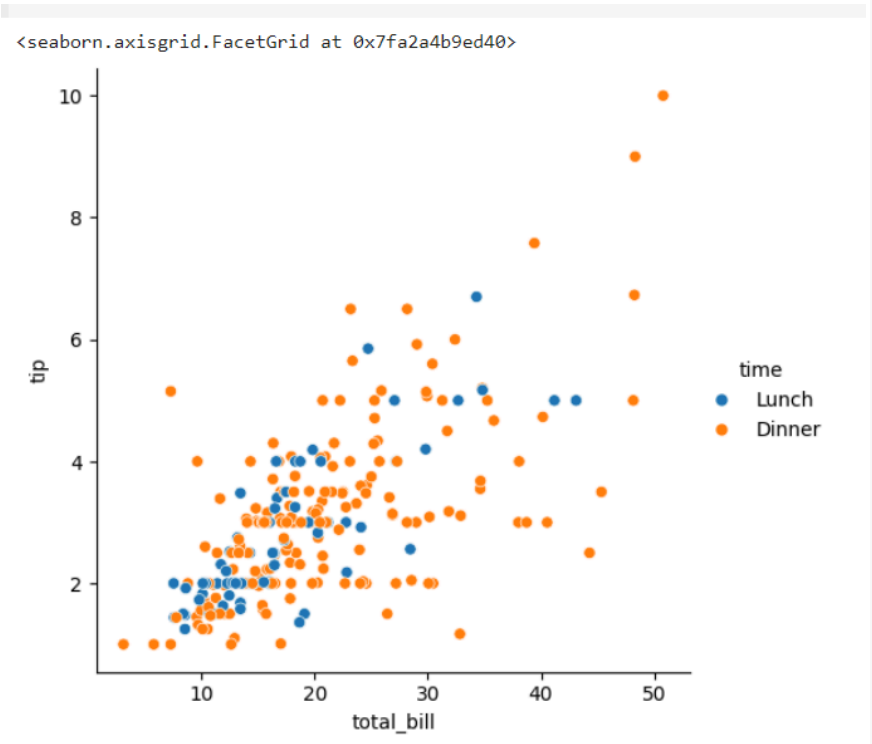
creating difference based on a column via colours

Example:

```
sns.relplot(data=tips,x='total_bill',y='tip',hue='time')
```

Output:

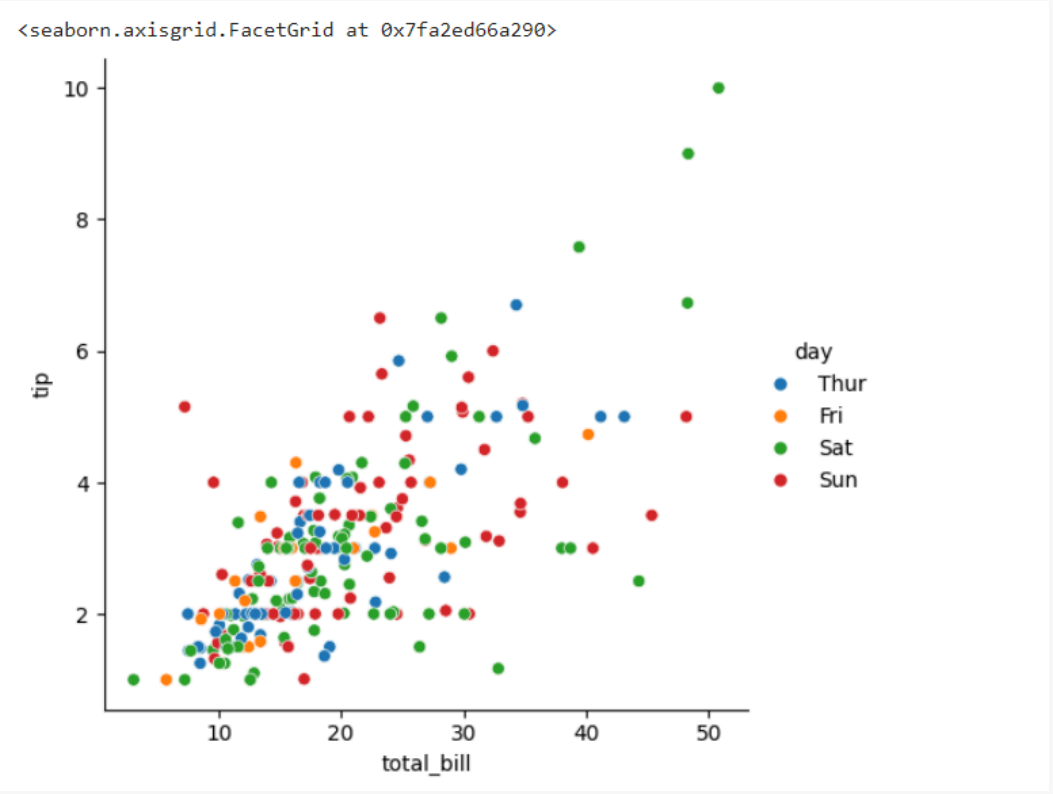
```
<seaborn.axisgrid.FacetGrid at 0x7fa2a4b9ed40>
```



Example-2:

```
sns.relplot(data=tips,x='total_bill',y='tip',hue='day')
```

Output:



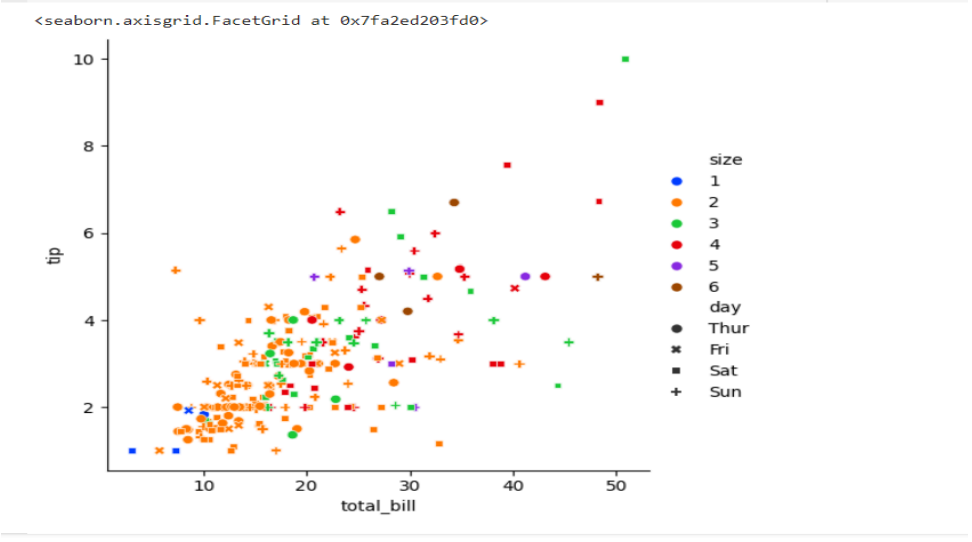
Note:style- uses for different markers for different categories

hue- different colour for diff category**.**

Example-1:

```
sns.relplot(data=tips,x='total_bill',y='tip',hue='size',style='day',palette='bright')
```

Output:



Colour Palette

->Pastel

->Bright

->dark

->muted

->colour blind

->deep

Example:

```
tips=sns.load_dataset("dowjones")
tips.head()
```

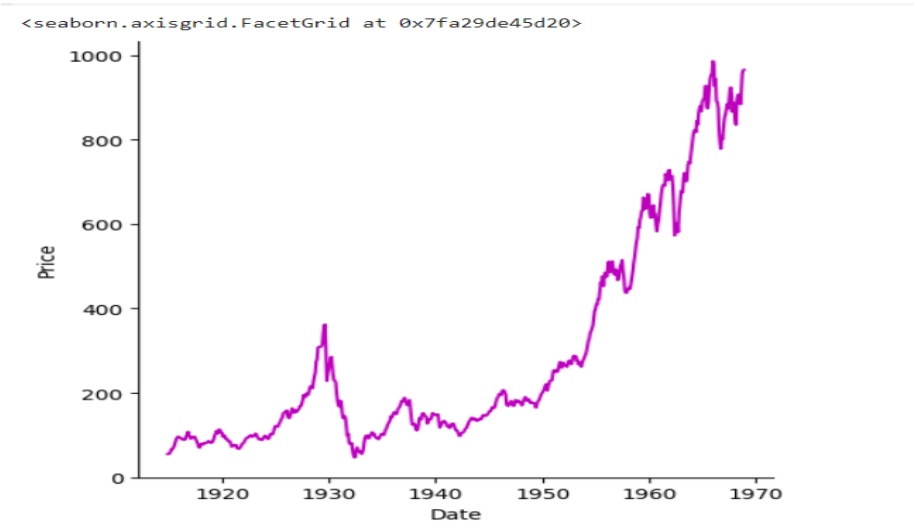
Output:

	Date	Price
0	1914-12-01	55.00
1	1915-01-01	56.55
2	1915-02-01	56.00
3	1915-03-01	58.30
4	1915-04-01	66.45

Example-2:

```
sns.relplot(data=tips,x="Date",y="Price",kind="line",color='m')
```

Output:



Example:

```
tips=sns.load_dataset("fmri")
tips.head()
```

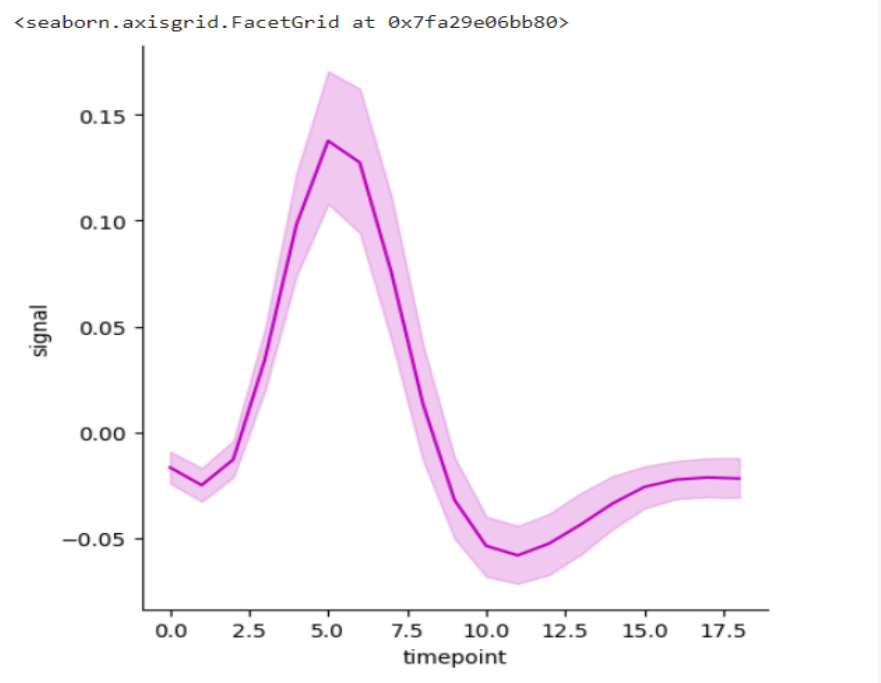
Output:

	subject	timepoint	event	region	signal	
0	s13	18	stim	parietal	-0.017552	
1	s5	14	stim	parietal	-0.080883	
2	s12	18	stim	parietal	-0.081033	
3	s11	18	stim	parietal	-0.046134	
4	s10	18	stim	parietal	-0.037970	

Example-2:

```
sns.relplot(data=tips,x="timepoint",y="signal",kind="line",color='m')
```

Output:

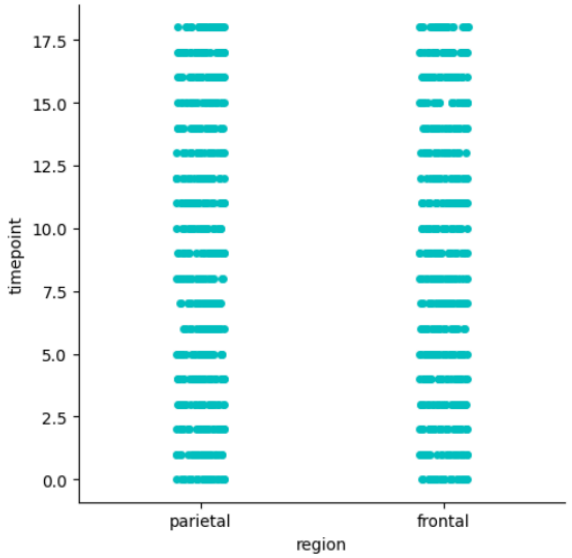


Example-3:

```
sns.catplot(data=fmri,x="region",y="timepoint",color="c")
```

Output:

```
<seaborn.axisgrid.FacetGrid at 0x7fa29ffeb160>
```

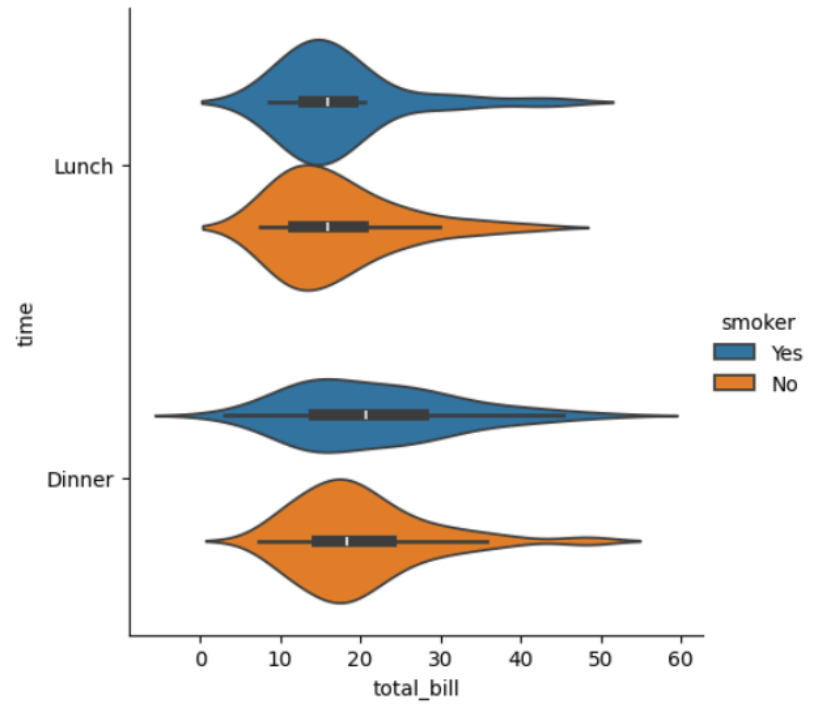


Example-4:

```
sns.catplot(data=tips,x="total_bill",y="time",kind="violin",hue="smoker")
```

Output:

```
<seaborn.axisgrid.FacetGrid at 0x7fa29dafdc60>
```



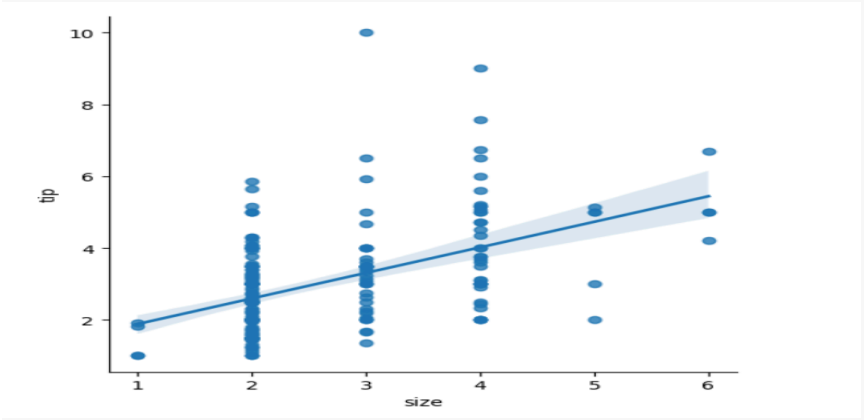
- plot the points

- fit the line
- select the line with least difference(linear fit)
- implot to create a function

Example:

```
sns.lmplot(x='size',y='tip',data=tips);
```

Output:



4.Web scrapping:

4.1. Introduction

* Loading data from websites
* unstrucutred in HTML
* convertible into spresadsheets/DB
* Major websites have their APIs for web scrapping

Purpose of scrabber:

* Extract all the data on particular sites
* Specific data that a user wants

Process:

* URL
* HTML codes
* Scrapes the required data
* saves it in required format(csv,xlsx,Json)

Applications:

* Email marketing
* Sentiment Analysis
* New monitoring
* Market Research
* Price Monitoring

Libraries:

->Beautifulsoup :

=>It is a python library. |

=>It takes input-**URL** and output-**HTML.|->static website**

->Requests #to access the webpage. |

->Selenium

->Pandas #to create,save,manipulate the data from the webpage.

->Web driver

->Webdriver_manager

To install any library:

!pip install library

!pip install–upgrade library #to upgrade to recent vision

Importing subpackage:

From parent import child

Eg:from bs4 import beautifulsoup.

Beautifulsoup:

*used to scrape data from static website.

*package bs4:subpackage beautifulsoup.

Function	**Purpose**	**Attributes**
Beautifulsoup()	To extract html code from a web page	.text .html
find()	To find first element of a kind	(‘element_name)
find_all()	To find all elements of a kind	(‘element_name)

Requests:

*used to send request to a web page.

*get(‘url’)

Web scrapping:

Problem-1:

program flow

1. import libraries
2. save the url
3. using requests.get() #access the web page
4. using beautifulsoup()#access the HTML code

```python
#importing libraries
from bs4 import BeautifulSoup
import requests
#save the url
url="https://monkeytype.com/"
#using requests.get()
req=requests.get(url)
#using BeautifulSoup()
bs=BeautifulSoup(req.text,"html")
print(bs)
```

Output:

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>Canva</title>
<meta content="maximum-scale=1.0, width=500" name="viewport"/>
<meta content="yes" name="apple-mobile-web-app-capable"/>
<style>@font-face{font-family:Open Sans;font-stretch:normal;font-style:normal;font-weight:300;src:url(data:font/woff2;base64,d09GMgABAAAAADIgAA8AAAAAYfAAADHEAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGk4bixAcPAZgP1NUQ
<style>a:hover{text-decoration:none}#error h1{display:inline-block;font-size:220%;font-weight:700;margin:-.2em 0 .4em}[dir=ltr] #errorMessage p.message-align,body{text-align:left}[dir=rtl] #errorMessage p.m
<meta content="375" http-equiv="refresh"/>
</head>
<body>
<div id="errorContainer"><img id="logo" src="
<div class="errorView" id="error">
<div id="errorMessage">
<div><form class="challenge-form" id="challenge-form"><div id="cf-please-wait"><div id="spinner"><div id="cf-bubbles"><div class="bubbles"></div><div class="bubbles"></div><div class="bubbles"></div></div><
<ul>
<li><span class="i18n" id="ray-id">Ray ID</span>: 855b0585fabf0b5e</li>
<li><span class="i18n" id="your-ip-address">Your IP address</span>: 34.90.67.65</li>
<li><span class="i18n" id="detected-location">Detected location</span>: NL</li>
</ul>
<p class="message i18n" id="checking-connection">
We’re currently checking your connection. This shouldn’t take long. <br/>
If you keep getting this page, check if there’s a <a href="https://www.canvastatus.com/" target="_blank">problem with Canva</a>.
</p>
</div>
</div>
</div>
<script>const bundles={en:{"access-denied":"Access denied","can-not-use-canva":"Due to your country, region, or IP address, you can’t use Canva.","try-refreshing-no-access":"Try refreshing the page.<br/>If
<script crossorigin="anonymous" data-cf-beacon='{"rayId":"830a53f3164f3078","b":1,"version":"2023.10.0","token":"9e2e0ea430cb46cd8c857cc5c188a9e6"}' defer="" integrity="sha512-euoFGowhlaLqXsPWQ48qSkBSCFs3DF
<script crossorigin="anonymous" data-cf-beacon='{"rayId":"855b0585fabf0b5e","b":1,"version":"2024.2.0","token":"ef88e2b2e2894e859fa7553dc8d0441c"}' defer="" integrity="sha512-euoFGowhlaLqXsPWQ48qSkBSCFs3DPF
</body>
</html>
```

Problem-2:

Exacting a table from HTML code of any website and saving it as .csv file.

Program flow

1.Import libraries.

```python
from bs4 import BeautifulSoup
```

```
import requests

import pandas as pd
```

2.save the url.

```
url="https://www.forbesindia.com/article/explainers/top-10-richest-people-india/85909/1"
```

3.using requests.gets(),access the web page.

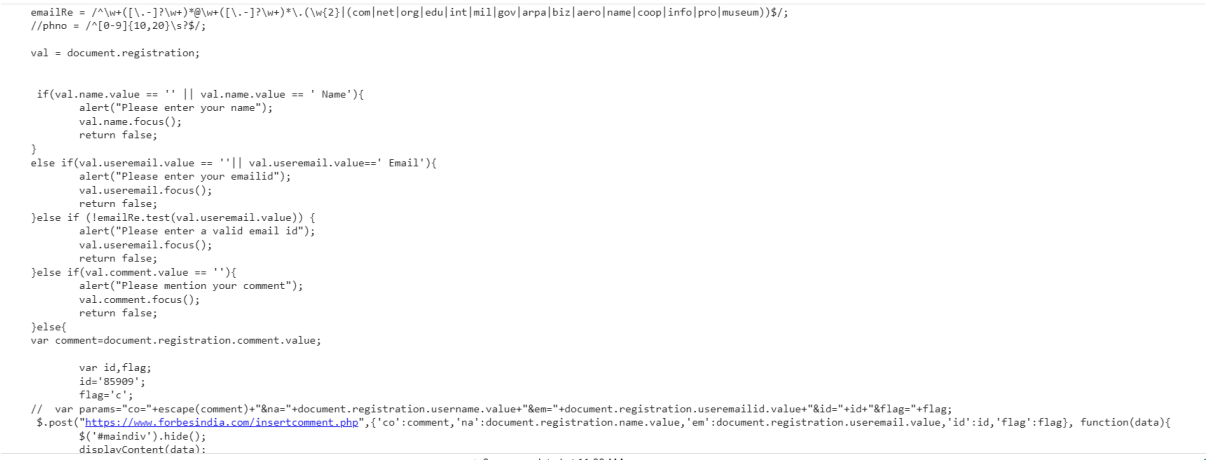
```
req=requests.get(url)
```

4.using BeautifulSoup()

```
bs=BeautifulSoup(req.text,"html")

print(bs)
```

Output:



```
emailRe = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*\.(\w{2}|(com|net|org|edu|int|mil|gov|arpa|biz|aero|name|coop|info|pro|museum))$/;
//phno = /^[0-9]{10,20}\s?$/;

val = document.registration;


 if(val.name.value == '' || val.name.value == ' Name'){
        alert("Please enter your name");
        val.name.focus();
        return false;
}
else if(val.useremail.value == ''|| val.useremail.value==' Email'){
        alert("Please enter your emailid");
        val.useremail.focus();
        return false;
}else if (!emailRe.test(val.useremail.value)) {
        alert("Please enter a valid email id");
        val.useremail.focus();
        return false;
}else if(val.comment.value == ''){
        alert("Please mention your comment");
        val.comment.focus();
        return false;
}else{
var comment=document.registration.comment.value;

        var id,flag;
        id='85909';
        flag='c';
//  var params="co="+escape(comment)+"&na="+document.registration.username.value+"&em="+document.registration.useremailid.value+"&id="+id+"&flag="+flag;
 $.post("https://www.forbesindia.com/insertcomment.php",{'co':comment,'na':document.registration.name.value,'em':document.registration.useremail.value,'id':id,'flag':flag}, function(data){
        $('#maindiv').hide();
        displayContent(data);
```

5.using find,access the table.

```
t=bs.find('table')

print(t)
```

Output:

```
print(t)
```

```
<td style="border: 1px solid black; padding: 8px;">$113.0 B</td>
<td style="border: 1px solid black; padding: 8px;">Reliance Industries</td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#2 Gautam Adani <br/></td>
<td style="border: 1px solid black; padding: 8px;">16<br/></td>
<td style="border: 1px solid black; padding: 8px;">$81.2 B</td>
<td style="border: 1px solid black; padding: 8px;">Adani Group</td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#3 Shiv Nadar <br/></td>
<td style="border: 1px solid black; padding: 8px;">37<br/></td>
<td style="border: 1px solid black; padding: 8px;">$37.1 B</td>
<td style="border: 1px solid black; padding: 8px;">HCL Technologies   <br/></td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#4 Savitri Jindal &amp;amp; family <br/></td>
<td style="border: 1px solid black; padding: 8px;">58<br/></td>
<td style="border: 1px solid black; padding: 8px;">$28.9 B</td>
<td style="border: 1px solid black; padding: 8px;">JSW Group<br/></td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#5 Cyrus Poonawalla <br/></td>
<td style="border: 1px solid black; padding: 8px;">68<br/></td>
<td style="border: 1px solid black; padding: 8px;">$25.6 B</td>
<td style="border: 1px solid black; padding: 8px;">Serum Institute of India<br/></td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#6 Dilip Shanghvi <br/></td>
<td style="border: 1px solid black; padding: 8px;">69<br/></td>
<td style="border: 1px solid black; padding: 8px;">$25.5 B</td>
```

6.using find_all,access the rows of the table.

```
headings=table.find_all('th')

print(headings)
```

Output:

```
[<th style="border: 1px solid black; padding: 8px;"><strong>Name &amp;amp; India Rank</strong></th>, <th style="border: 1px solid black; padding: 8px;"><strong>Global Rank</strong></th>, <th style="border: 1px
```

7.using text,extract only the text(removing html tags)

```
head=[i.text for i in headings ]

print(head)
```

Output:

['Name & India Rank', 'Global Rank', 'Net worth (US$)', 'Company']

8.Create a dataframe using pandas.

```
df=pd.DataFrame(columns=headings)

print(df)
```

Output:

```
Empty DataFrame
Columns: [([Name & India Rank], [Global Rank], [Net worth (US$)], [Company])]
Index: []
```

9.push all the extracted data into dataframes.

```
a=0

for i in rows[1:9]:

 data=i.find_all('td')

 er=[i.text for i in data]

 df.loc[a]=er

 a=a+1

Df
```

Output:

	Name & India Rank	Global Rank	Net worth (US$)	Company
0	#1 Mukesh Ambani	11	$113.0 B	Reliance Industries
1	#2 Gautam Adani	16	$81.2 B	Adani Group
2	#3 Shiv Nadar	37	$37.1 B	HCL Technologies
3	#4 Savitri Jindal & family	58	$28.9 B	JSW Group
4	#5 Cyrus Poonawalla	68	$25.6 B	Serum Institute of India
5	#6 Dilip Shanghvi	69	$25.5 B	Sun Pharmaceutical Industries Ltd
6	#7 Kumar Birla	97	$18.9 B	Aditya Birla Group
7	#8 Kushal Pal Singh	98	$18.9 B	DLF Limited

10.using to_csv,save the dataframe in .csv format

```
df.to_csv("richmen.csv")
```

Output:

	Name & India Rank	Global Rank	Net worth (US$)	Company
0	#1 Mukesh Ambani	11	$113.0 B	Reliance Industries
1	#2 Gautam Adani	16	$81.2 B	Adani Group
2	#3 Shiv Nadar	37	$37.1 B	HCL TechnologiesÂ Â
3	#4 Savitri Jindal & family	58	$28.9 B	JSW Group
4	#5 Cyrus Poonawalla	68	$25.6 B	Serum Institute of India
5	#6 Dilip Shanghvi	69	$25.5 B	Sun Pharmaceutical Industries Ltd
6	#7 Kumar Birla	97	$18.9 B	Aditya Birla Group
7	#8 Kushal Pal Singh	98	$18.9 B	DLF Limited

Accessing elements:

*Go to webpage

*Right click->Inspect

*Click on

*Now hover on the element you want to access

*The html tags/code for that specific element is highlighted

*Extract the element name and import in your program using XPATH or By ID.

Selenium:

*used to scrape data from static website.

*subpackage:webdriver.

Function	**Purpose**	**Attributes**
ChromeOptions()	Creates an instance of chrome	
.get	Access a webpage	‘url’
find_element	To find first element of a kind	By.ID By.XPATH
.click()	To click a button in webpage	

Problem-3:(Selenium)

Extracting dell laptops data from amazon.in website and saving it as.csv Name of the laptop,price,No of reviews.

installing libraries:

!pip install selenium

Output:

```
Requirement already satisfied: selenium in c:\users\valli\anaconda3\lib\site-packages (4.17.2)
Requirement already satisfied: urllib3[socks]<3,>=1.26 in c:\users\valli\anaconda3\lib\site-packages (from selenium) (1.26.16)
Requirement already satisfied: trio~=0.17 in c:\users\valli\anaconda3\lib\site-packages (from selenium) (0.24.0)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\valli\anaconda3\lib\site-packages (from selenium) (0.11.1)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\valli\anaconda3\lib\site-packages (from selenium) (2023.7.22)
Requirement already satisfied: typing_extensions>=4.9.0 in c:\users\valli\anaconda3\lib\site-packages (from selenium) (4.9.0)
Requirement already satisfied: attrs>=20.1.0 in c:\users\valli\anaconda3\lib\site-packages (from trio~=0.17->selenium) (22.1.0)
Requirement already satisfied: sortedcontainers in c:\users\valli\anaconda3\lib\site-packages (from trio~=0.17->selenium) (2.4.
0)
Requirement already satisfied: idna in c:\users\valli\anaconda3\lib\site-packages (from trio~=0.17->selenium) (3.4)
Requirement already satisfied: outcome in c:\users\valli\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\valli\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.3.0)
Requirement already satisfied: cffi>=1.14 in c:\users\valli\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.15.1)
Requirement already satisfied: wsproto>=0.14 in c:\users\valli\anaconda3\lib\site-packages (from trio-websocket~=0.9->selenium)
(1.2.0)
Requirement already satisfied: PySocks!=1.5.7,<2.0,>=1.5.6 in c:\users\valli\anaconda3\lib\site-packages (from urllib3[socks]<
3,>=1.26->selenium) (1.7.1)
Requirement already satisfied: pycparser in c:\users\valli\anaconda3\lib\site-packages (from cffi>=1.14->trio~=0.17->selenium)
(2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\valli\anaconda3\lib\site-packages (from wsproto>=0.14->trio-websocket~
=0.9->selenium) (0.14.0)
```

!pip install webdriver_manager

Output:

```
Collecting webdriver_manager
  Obtaining dependency information for webdriver_manager from https://files.pythonhosted.org/packages/b1/51/b5c11cf739ac4eecde6
11794a0ec9df420d0239d51e73bc19eb44f02b48b/webdriver_manager-4.0.1-py2.py3-none-any.whl.metadata
  Downloading webdriver_manager-4.0.1-py2.py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: requests in c:\users\valli\anaconda3\lib\site-packages (from webdriver_manager) (2.31.0)
Requirement already satisfied: python-dotenv in c:\users\valli\anaconda3\lib\site-packages (from webdriver_manager) (0.21.0)
Requirement already satisfied: packaging in c:\users\valli\anaconda3\lib\site-packages (from webdriver_manager) (23.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\valli\anaconda3\lib\site-packages (from requests->webdriver
_manager) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\valli\anaconda3\lib\site-packages (from requests->webdriver_manager)
(3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\valli\anaconda3\lib\site-packages (from requests->webdriver_manag
er) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\valli\anaconda3\lib\site-packages (from requests->webdriver_manag
er) (2023.7.22)
Downloading webdriver_manager-4.0.1-py2.py3-none-any.whl (27 kB)
Installing collected packages: webdriver_manager
Successfully installed webdriver_manager-4.0.1
```

Importing package:

```
from webdriver_manager.chrome import ChromeDriverManager
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
```

Launching chrome:

```
#define options and  set browser capabilities
options=webdriver.ChromeOptions()
options.add_argument('--some-option')
#create webdriver instance with options
driver=webdriver.Chrome(options=options)
```

#access browser capabilities

browser_name=options.to_capabilities()['browserName']

print(browser_name)

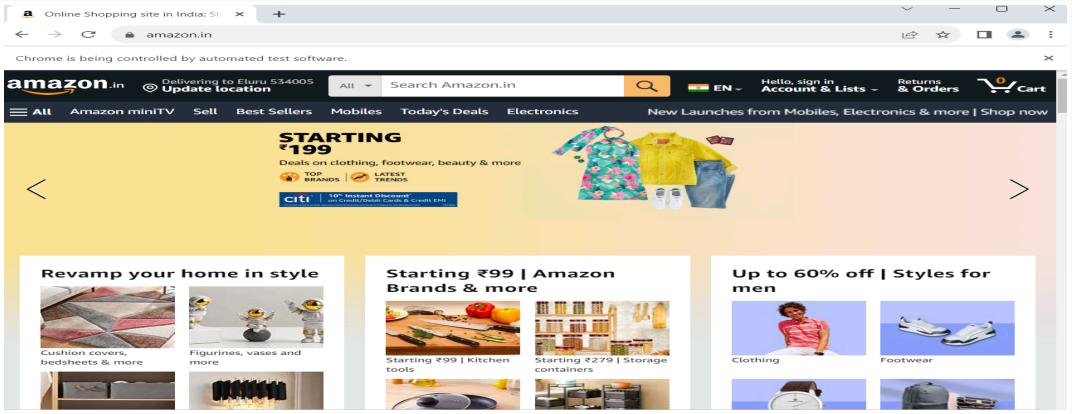
Output:

Chrome

Negative to amazon:

driver.get('https://www.amazon.in/')

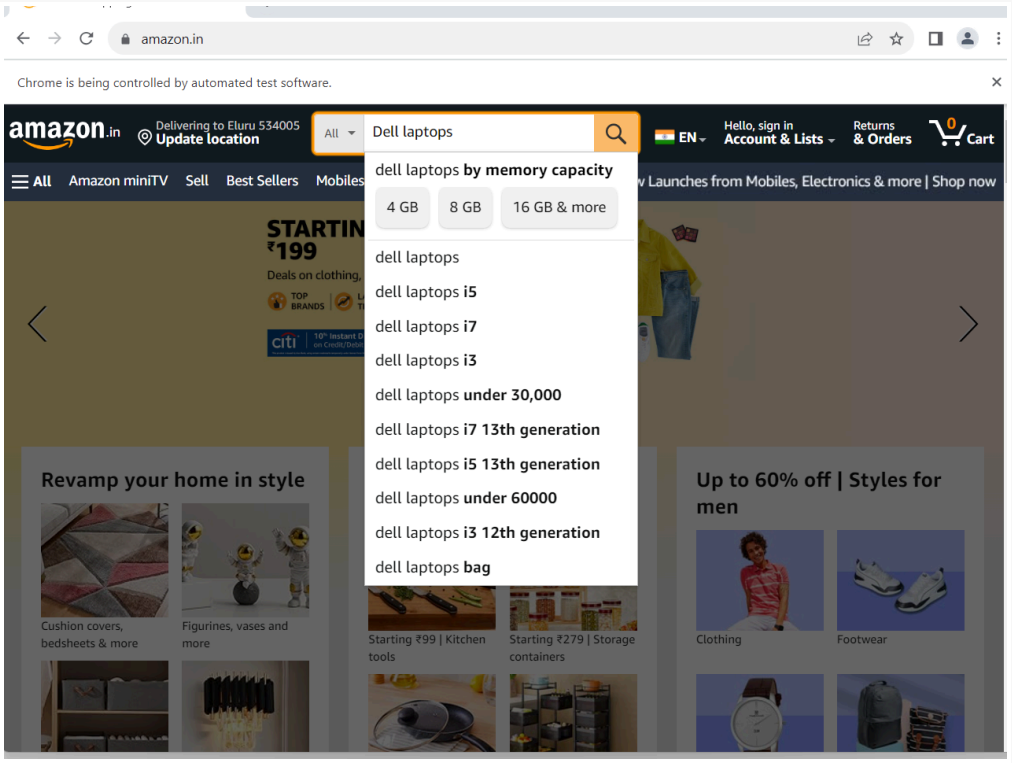
Output:



search=driver.find_element(By.ID,"twotabsearchtextbox")

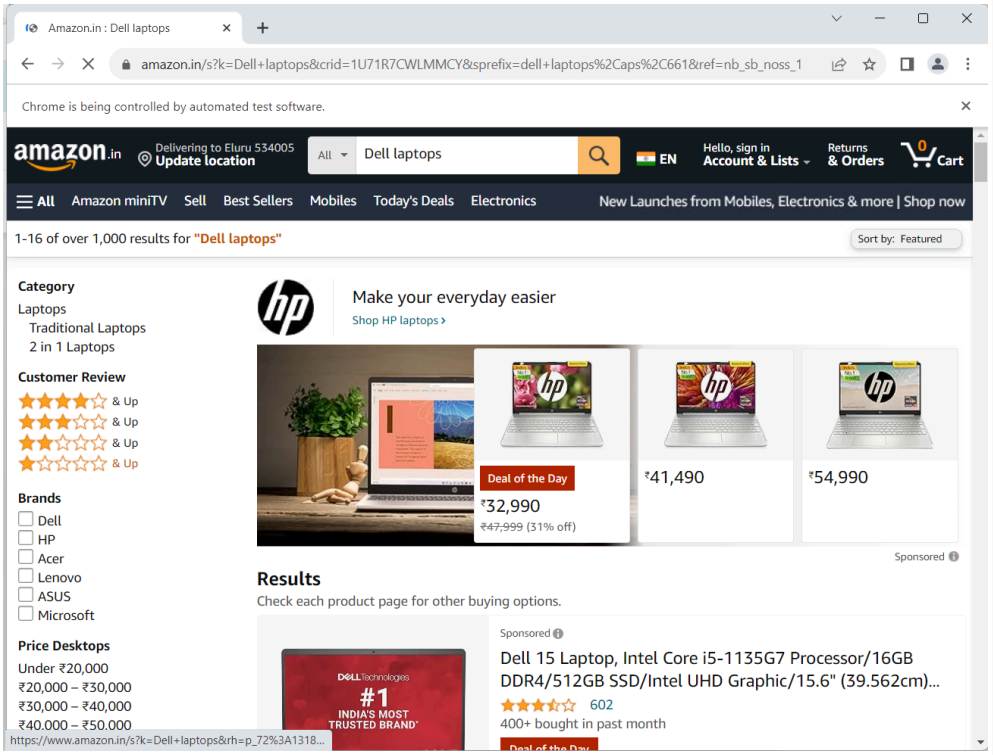
search.send_keys("Dell laptops")

Output:



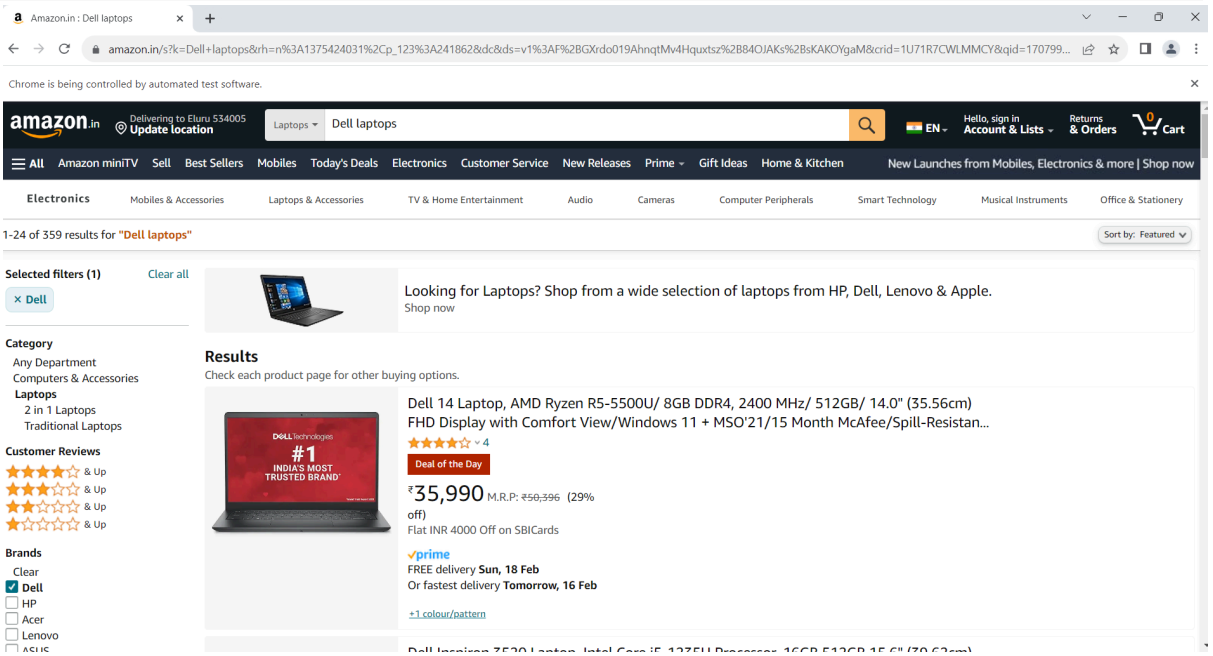
driver.find_element(By.ID,"nav-search-submit-button").click()

Output:



driver.find_element(By.XPATH,"//span[text()='Dell']").click()

Output:



#list of laptops

names=driver.find_elements(By.XPATH,"//span[@class='a-size-medium a-color-base a-text-normal']")

l_names=[i.text for i in names]

print(l_names)

Output:

```
print(l_names)
```

```
['Dell 14 Laptop, AMD Ryzen R5-5500U/ 8GB DDR4, 2400 MHz/ 512GB/ 14.0" (35.56cm) FHD Display with Comfort View/Windows 11 + MSO
\'21/15 Month McAfee/Spill-Resistant Keyboard/Carbon Black/ 1.48kg', 'Dell Inspiron 3520 Laptop, Intel Core i5-1235U Processor,
16GB,512GB,15.6" (39.62cm) FHD Display, Backlit Keyboard, Win 11 + MSO\'21,15 Month McAfee, Silver, Thin & Light-1.65kg', 'Dell
Inspiron 3535 Laptop, AMD 7 Series Ryzen 5-7520U Processor/8GB/512GB/15.6" (39.62cm) FHD WVA AG 120Hz 250 nits Display/Win 11 +
MSO\'21/15 Month McAfee/Platinum Silver/1.63kg', 'Dell 14 Laptop, 12th Gen Intel Core i5-1235U Processor, 16GB, 512GB, 14.0" (3
5.56cm) FHD Display, Windows 11 + MSO\'21, Spill-Resistant Keyboard, 15 Month Mcafee, Black, Thin & Light- 1.48kg', 'Dell 14 La
ptop, 12th Gen Intel Core i3-1215U Processor/8GB/512GB SSD/Intel UHD Graphics/14.0"(35.56cm) FHD/Windows 11 + MSO\'21/15 Month
McAfee/Spill-Resistant Keyboard/Grey/Thin & Light 1.48kg', 'Dell Inspiron 7430 2in1 Touch Laptop, 13th Gen Intel Core i3-1315U/
8GB/1TB SSD/14.0" (35.56cm) FHD+,16:10 Aspect Ratio/Backlit KB+FPR/Win 11+MSO\'21/15 Months McAfee/Platinum Silver/Thin & Light
-1.58kg', 'Dell 15 Laptop, Intel Core i5-1135G7 Processor/ 8GB/ 1TB+256GB SSD/15.6"(39.62cm) FHD Display/Mobile Connect/Windows
11 + MSO\'21/15 Month McAfee/Spill-Resistant Keyboard/Black/Thin & Light 1.69kg', 'Dell Inspiron 3530 Laptop, 13th Gen Intel Co
re i5-1335U Processor/16GB/1TB SSD/15.6" (39.62cm) FHD Display/Backlit Keyboard/Platinum Silver/Win 11 + MSO\'21/15 Month McAfe
e/Thin & Light- 1.66kg', 'Dell G15 5520 Gaming Laptop, Intel i5-12500H/16GB DDR5/1TB SSD/15.6" (39.62cm) FHD WVA AG 120Hz 250 n
its/NVIDIA RTX 3050, 4 GB GDDR6/Win 11 + MSO\'21/15 Months McAfee/Backlit KB/Dark Shadow Grey/2.81kg', 'Dell 15 Laptop, Intel C
ore i3-1115G4 Processor/8GB DDR4/512GB SSD/Intel UHD Graphics/15.6" (39.6cm) FHD 120Hz Refresh, 250 nits/Mobile Connect/Win 11+
MSO\'21/15 Month McAfee/Black/Thin & Light-1.66kg', 'Dell 14 Laptop, Intel Core i5-1135G7 Processor/8GB/512GB/Intel UHD Graphi
c/14.0" (35.6cm) FHD with Comfort View/Spill-Resistant Keyboard/Thin & Light 1.48kg/ Win 11+ MSO\'21/15 Month McAfee/Titan Gre
y', '(Renewed) DELL Latitude 5490 Core i5 8th Gen Laptop, 16 GB RAM, 512GB SSD, Intel HD Graphics, 14 inch (36.83 cms) HD Scree
n, Windows 11 (Upgraded), MS Office, Black, Slim', 'Dell 14, Intel 12th Gen i5-1235U Laptop/8GB/512GB SSD/14.0" (35.56cm) FHD T
Ã¢V Rheinland Certified Comfortview to Reduce Harmful Blue Light Emission/Windows 11 + MSO\'21/Black/15 Month McAfee,1.48kg',
'Dell New 14" Latitude 3420- i3 11th Gen || 8 GB || 512 GB SSD || 14" Full HD 1920 x 1080 Pixels || Ubuntu-Dos || 1 Year Onsite
with ADP Warranty', 'Dell Inspiron 5430 Laptop, 13th Gen Intel Core i7-1360P Processor/16GB/1TB SSD/14.0" (35.56cm) FHD+ WVA 25
0 nits/Backlit KB + FPR/Win 11 + MSO\'21/15 Month McAfee/Platinum Silver/Thin & Light- 1.59kg', 'Dell Inspiron 3530 Laptop, Int
el Core i7-1355U Processor, 16GB, 512GB, 15.6" (39.62cm) FHD WVA AG 120Hz, Backlit KB + FPR, Win 11 + MSO\'21, 15 Month McAfee,
Silver, 1.62kg', 'Dell New G15-5515 Gaming Laptop, AMD Ryzen5-5600H, Win 11 + MSO\'21, 16Gb Gddr4, 512Gb SSD, Nvidia RTX 3050
(4Gb Gddr6), 15.6" (39.62Cms) FHD AG 250 Nits 120Hz, Backlit KB Orange, 2.57Kgs', 'Dell 14 Premium Metal Laptop, AMD Ryzen 5-56
25U, 16GB, 512GB, 14.0" (35.56CM) FHD Display, Backlit KB + FPR, Win 11 + MSO\'21, 15 Month McAfee, Dark Silver, Thin & Light-
```

#length of laptop names

print(len(l_names))

Output:

24

#printing the price of laptops

prices=driver.find_elements(By.XPATH,"//span[@class='a-price-whole']")

l_price=[i.text for i in prices]

print(l_price)

Output:

['34,990', '33,990', '35,990', '35,990', '55,280', '38,990', '49,990', '35,990', '57,990', '44,990', '67,490', '75,990', '33,990', '44,990', '23,649', '46,990', '30,630', '83,490', '71,490', '78,490', '50,490', '37,990', '37,817', '34,380', '81,490', '82,990', '87,490']

#popping the duplicate prices

```
l_price.pop(0) #3 times repeated
print(l_price)
```

Output:

['35,990', '55,280', '38,990', '49,990', '35,990', '57,990', '44,990', '67,490', '75,990', '33,990', '44,990', '23,649', '46,990', '30,630', '83,490', '71,490', '78,490', '50,490', '37,990', '37,817', '34,380', '81,490', '82,990', '87,490']

#printing the length of prices:

```
print(len(l_price))
```

Output:

24

#printing the reviews

```
reviews=driver.find_elements(By.XPATH,"//span[@class='a-size-base s-underline-text']")
l_reviews=[i.text for i in reviews]
print(l_reviews)
```

Output:

['4', '2', '4', '72', '239', '179', '607', '13', '517', '631', '2', '506', '138', '1', '82', '1', '71', '1', '186', '151', '195', '11', '27', '7']

#printing the length of review

```
print(len(l_reviews))
```

Output:

24

#creating the dataframes:

```
import pandas as pd
headings=['Laptop','Price','Review']
df=pd.DataFrame(columns=headings)
print(df)
```

Output:

```
Empty DataFrame
Columns: [Laptop, Price, Review]
Index: []
df['Laptop']=l_names
df['Price']=l_price
df['Review']=l_reviews
print(df)
```

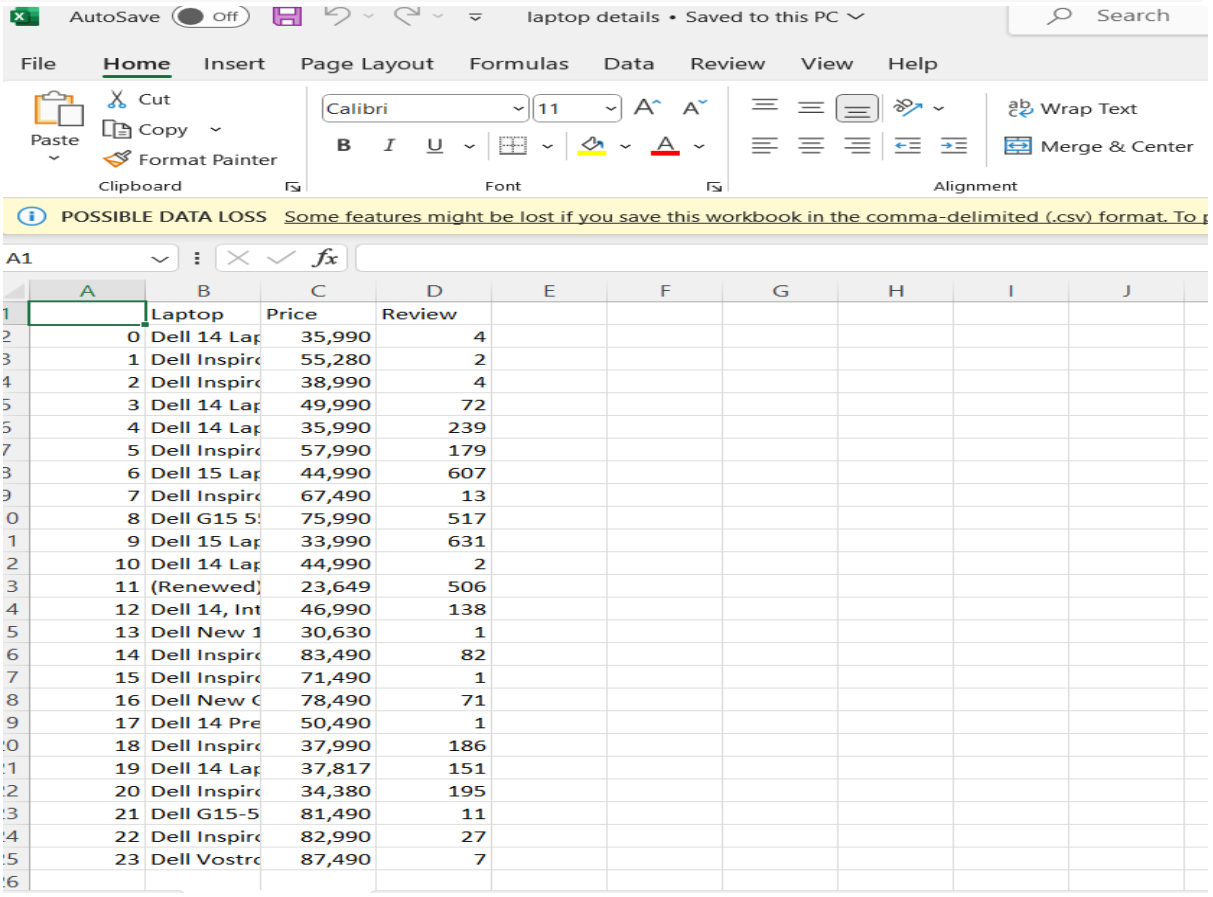
Output:

	Laptop	Price	Review
0	Dell 14 Laptop, AMD Ryzen R5-5500U/ 8GB DDR4, ...	35,990	4
1	Dell Inspiron 3520 Laptop, Intel Core i5-1235U...	55,280	2
2	Dell Inspiron 3535 Laptop, AMD 7 Series Ryzen ...	38,990	4
3	Dell 14 Laptop, 12th Gen Intel Core i5-1235U P...	49,990	72
4	Dell 14 Laptop, 12th Gen Intel Core i3-1215U P...	35,990	239
5	Dell Inspiron 7430 2in1 Touch Laptop, 13th Gen...	57,990	179
6	Dell 15 Laptop, Intel Core i5-1135G7 Processor...	44,990	607
7	Dell Inspiron 3530 Laptop, 13th Gen Intel Core...	67,490	13
8	Dell G15 5520 Gaming Laptop, Intel i5-12500H/1...	75,990	517
9	Dell 15 Laptop, Intel Core i3-1115G4 Processor...	33,990	631
10	Dell 14 Laptop, Intel Core i5-1135G7 Processor...	44,990	2
11	(Renewed) DELL Latitude 5490 Core i5 8th Gen L...	23,649	506
12	Dell 14, Intel 12th Gen i5-1235U Laptop/8GB/51...	46,990	138
13	Dell New 14" Latitude 3420- i3 11th Gen || 8 G...	30,630	1
14	Dell Inspiron 5430 Laptop, 13th Gen Intel Core...	83,490	82
15	Dell Inspiron 3530 Laptop, Intel Core i7-1355U...	71,490	1
16	Dell New G15-5515 Gaming Laptop, AMD Ryzen5-56...	78,490	71
17	Dell 14 Premium Metal Laptop, AMD Ryzen 5-5625...	50,490	1
18	Dell Inspiron 3520 Laptop,12th Gen Intel Core ...	37,990	186
19	Dell 14 Laptop, AMD Ryzen 5-5500U/ 8GB- 2 DIMM...	37,817	151
20	Dell Inspiron 3520 Laptop, Intel Core i3-1215U...	34,380	195
21	Dell G15-5530 Gaming Laptop, Intel Core i5-134...	81,490	11
22	Dell Inspiron 7430 2in1 Touch Laptop, Intel Co...	82,990	27
23	Dell Vostro 5620 Laptop, Intel Core i7-1260P P...	87,490	7

#saving the csv file

df.to_csv('laptop details.csv')

Output:



API(Application Programming Interface)

API:

*It is a set of protocols.

*It is a server used to retrieve and save the data.

Purpose:

->Collection of small apps together forms a big app.

->Every app has a API.

Eg:In order to build the OLA app,it uses different API.

It takes the login page from some developer and uses API.

Similarly it takes the API of google maps,paytm for making payments and so on.Likes it uses different API's and forms a big app.

Usually if APIs were free then many people would access it.If more than 1000 apps access then it will crash.In order to avoid that,API keys are introduced.

The API keys will not only avoid crashing

PUBLIC API:

Accessible to every user.

PRIVATE API:

Accessible only to the subscribes.

->One website accessing data from database,the connection is straightforward.

->One website accessing data from different belonging to different companies needs an API.

Eg:

OLA

->App1:login

->App2:Location

->App3:payment

And so on..

Step-1:

```
import requests
```

Step-2:

```
page=requests.get('https://randomfox.ca/floof')
```

Step-3:

```
print(page.status_code)
```

Output:

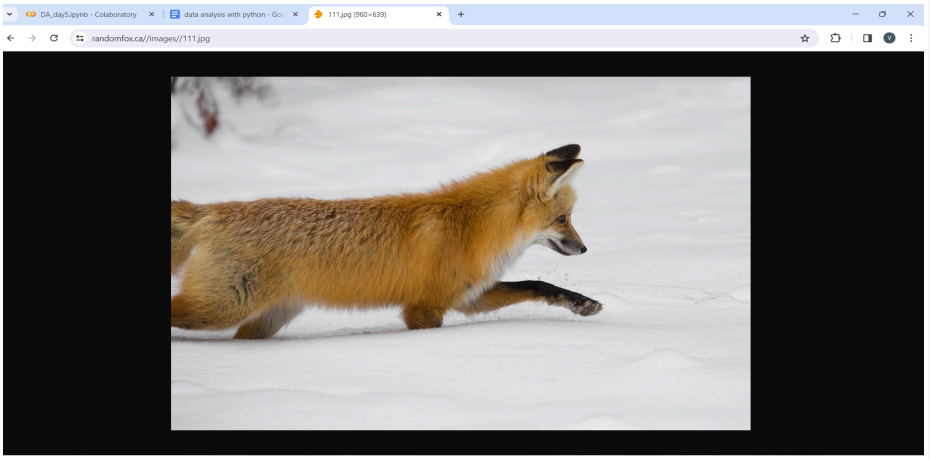
200

Step-4:

```
print(page.text)
```

Output:

{"image":"https:\/\/randomfox.ca\/images\/111.jpg","link":"https:\/\/randomfox.ca\/?i=111"}



Extracting data from Coinmarketcap API:

Program flow:

Step-1:Go to Coinmarketcap Website

Step-2:Go to Products->API and obtain API key

Step-3:Go to documentation to obtain API code!

Step-4:Know your code!

Step-5:Run it in any python environment

Step-6:Obtain data in the form of json

Step-7:Normalize the data into a dataframe

Step-8:Save it if you needed.

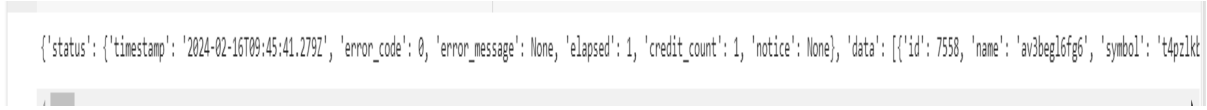
```
from requests import Request, Session
from requests.exceptions import ConnectionError, Timeout, TooManyRedirects
import json

url = 'https://sandbox-api.coinmarketcap.com/v1/cryptocurrency/listings/latest'
parameters = {
  'start':'1',
  'limit':'5000',
  'convert':'USD'
}
headers = {
  'Accepts': 'application/json',
  'X-CMC_PRO_API_KEY': '10308d0a-e505-499a-aba9-57ed557a8b9d',
}
```

```
session = Session()
session.headers.update(headers)


try:
  response = session.get(url, params=parameters)
  data = json.loads(response.text)
  print(data)
except (ConnectionError, Timeout, TooManyRedirects) as e:
  print(e)
```

Output:



Step-2:

```
import pandas as pd
norm=pd.json_normalize(data['data'])
norm
```

Output:

	id	name	symbol	slug	cmc_rank	num_market_pairs	circulating_supply	total_supply	max_supply	infinite_supply	...	quote.USD.price	quote.USD.volume_24h	quote.USD.volume_chang
0	7558	av3begl6fg6	t4pzlkbsyo	x15rb6g226	1627	9987	6966	3137	2836	None	...	0.981952	258	0.8
1	4227	gjm9568m2a	11qu74eb3gne	38mxkka8iu1	395	5878	7702	6747	7090	None	...	0.920645	13	0.1
2	4143	c604f0pp0ve	n97ohyef7mk	cysa91mbpcm	4859	9806	4728	9773	6771	None	...	0.668168	2792	0.7
3	8753	62dz5hm84v4	qmtr8igng7s	02n1rz5ngxdy	9751	1691	1109	6668	5523	None	...	0.762547	2197	0.7
4	1469	ikun0vxyelm	dnbrjbmx6hm	9a4w59udvlo	888	9315	6000	6348	1316	None	...	0.392649	9377	0.6
5	9789	7owwkl0wr7r	vwyhatw5tup	ubb071f6u5	3422	1511	7708	3997	4647	None	...	0.955100	2911	0.7
6	7067	y96flekzv5c	fzimw5s8lxn	nakyernfp5b	8243	7324	1658	254	6607	None	...	0.067676	3514	0.2
7	9085	caflm1803a5	qhwbxxrxzvk	ehcfry93cyk	9199	9991	4295	3731	4843	None	...	0.183867	1916	0.3
8	5116	u95mhd9fti	uw5wlckrzv	68a3v2nmb78	2787	9921	189	2460	4046	None	...	0.398316	7935	0.2
9	1418	ifnv5gl4wke	zpoksfibt4p	v2x0oip2y1q	4190	2244	5322	1781	759	None	...	0.601261	5317	0.6

10 rows × 26 columns

Machine Learning:

->Like humans learn from their past experiences,machines learn from Past data.

->Training a machine on various things is called Machine Learning.

Case study:

A person listen to 100 songs.He liked some songs and some are disliked.

->Now based on the likes and dislikes the machine is going to plot a Graph between temp and intensity.

->Now the machine will suppose to suggest a song based on the temp and intensity of the song.

Types of machine learning:

Supervised.

Steps:

*It will train the machine with algorithms to predict the outcomes using the labelled data.

*It takes a feature for every label.

*Predict the output based on the feature.

*No.of inputs=any one from the no.of outputs.

Eg:

Assume and training a machine with a rupees,euro,ditham currency coins.

Training machine with a feature called “weight”

Label the data as currency(input) and predict the output based on feature.

If we give the grams weight of any of three coins it will output as any one of these label.

->when we provide a new coin ,it will look for a feature value.If the feature value is matched it will give that label as output.

Supervised learning algorithm:

->Linear algorithm

->Logistic regression

->Decision tree

->Random forest etc.

Linear Regression:

->Linear regression is also type of machine-learning algorithm .

->Supervised machine-learning algorithm.

->Learns from the labelled datasets and maps the data points to the most optimized linear functions.

->These points can be used fo prediction on new datasets.

Dependent and Independent variable:

->**Independent:**

The independent variable is the cause.Its value is independent of other variables in your study.

->**Dependent:**

The dependent variable is the effect.Its value depends on changes in the independent variable.

Case study:

Consider measurements of a chemical reaction:

The mass of the product increases with time.

The observation are:

Time(m)	5	7	12	16	20
Mass(gms)	40	120	180	210	240

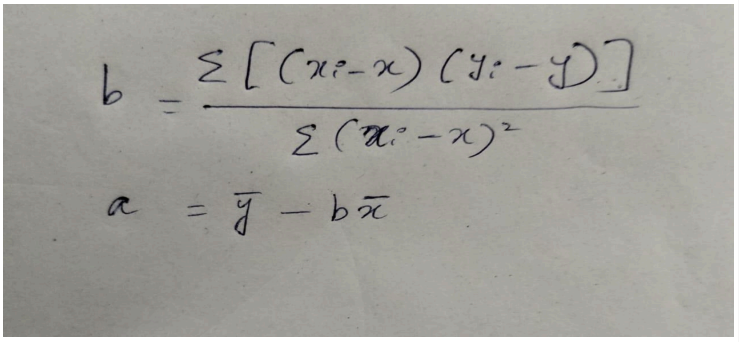
Time-Independent

Mass-Dependent

According to this case study,we have 5 values for features 5 labels(for each feature).

1.Find the mean of dependent and independent variable.

2.Linear equation or regression is y=a+bx.



fir-Process of training your model(algorithm).

step-1:

```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
```

Step-2:

```
 t=[[5],[7],[12],[16],[20]] #2-D array
#t=time
m=[40,120,180,210,240] #1-D array
#m=mass
LR.fit(t,m)
```

Output:

```
▾ LinearRegression
LinearRegression()
```

Step-3:

LR.predict([[5.5]])

Output:

array([78.64935065])

Logistic regression:

*Used for binary classification.

*Supervised learning algorithm.

Binary classification:

Eg:

Segregating Food:

Y=0,Class A-Healthy food.

Y=1,Class B-Unhealthy food.

Case study:

->Let us take data from football matches.

->Based on the distance between the player and goal post,we are going to predict whether it is goal are not!

->Let us plot some trails now

*when distance =2m,goal is scored,Y=1

*when distance =4m,goal is scored,Y=1

*for 5,7,10,20,22m ,it always a gaol,Y=1

*when distance =23m,for 15 trails

Few are goals Y=1,few are failures Y=0

Sigmoid Function:

->Sigmoid function is a S-shaped function with peak at 1 and 0 at valley.

->When model is very confident,Narrow decision boundary.

->When model is not confident ,wide decision boundary.

After plotting the distance against the goal between 20m and 30m the probability reduces from 1 to 0.

But logistic regression only accepts 2 classes.

So a threshold variable (0.5) is set.

The model prediction:

*P>0.5,Its a Goal!-Class A(Y=1)

*p<0.5,Its a Miss!-Class B(Y=0)

Step-1:

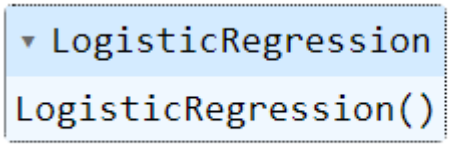
```
import numpy as np
from sklearn.linear_model import LogisticRegression
```

Step-2;

```
# Distance and corresponding probability data
```

```
distances = np.array([1,2,5,10,15,20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30,35,40,41,47,50]).reshape(-1, 1)
probabilities = np.array([1,1,1,1,1,1,0.9, 0.85, 0.73, 0.67, 0.5, 0.47, 0.39, 0.31, 0.25,
0.15,0,0,0,0,0])
#convert probabilities to binary lables
threshold=0.5
binary_lables=(probabilities>threshold).astype(int)
#create and fit logistic regression model
logr=LogisticRegression()
logr.fit(distances,binary_lables)
```

Output:



Step-3:

```
logr.predict([[10]])
```

Output:

```
array([1])
```

Logistic Statistic:

Step-1:

```
#predict 100 distances between 1 and 50
#Generate distances for prediction
dist=np.linspace(1,50,100).reshape(-1,1)
print(dist)
#make predictions using the model
prob=logr.predict_proba(dist)[:,1] #predictions
print(prob)
```

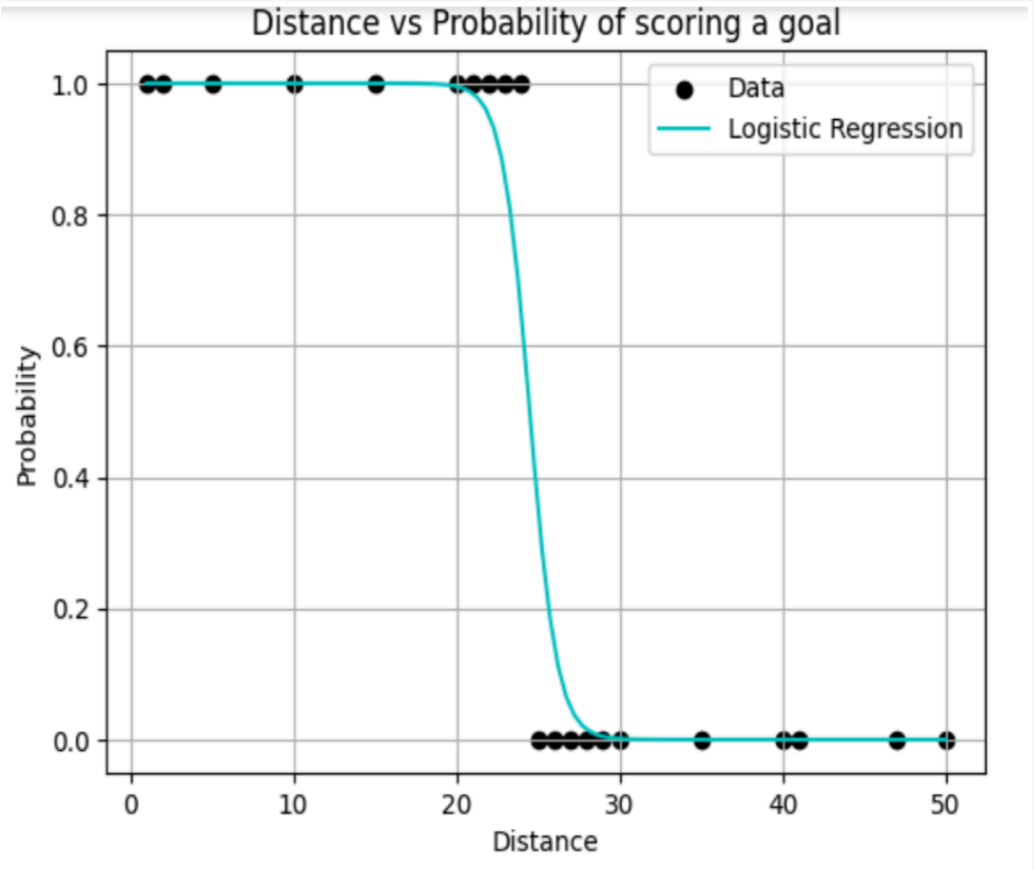
Output:

```
[[ 1.        ]
 [ 1.49494949]
 [ 1.98989899]
 [ 2.48484848]
 [ 2.97979798]
 [ 3.47474747]
 [ 3.96969697]
 [ 4.46464646]
 [ 4.95959596]
 [ 5.45454545]
 [ 5.94949495]
 [ 6.44444444]
 [ 6.93939394]
 [ 7.43434343]
 [ 7.92929293]
 [ 8.42424242]
 [ 8.91919192]
 [ 9.41414141]
```

Step-2:

```
import matplotlib.pyplot as plt
#Plotting actual data-train
plt.scatter(distances,binary_lables,color='black',label='Data')
#Plotting test data with predictions-valid/test
plt.plot(dist,prob,color='c',label='Logistic Regression')
plt.title('Distance vs Probability of scoring a goal')
plt.xlabel('Distance')
plt.ylabel('Probability')
plt.legend()
plt.grid(True)
plt.show()
```

Output:



Decision trees:

->Decision trees in machine learning provide an effective method for making decisions because they lay out the problem and all the possible outcomes.

->Have Nodes and Leaves.

->Node:Condition having True and False Branches.

->Leaf:Result-showing the dataset that is true and false to the condition.

Case study:

*Taking s dataset of 26 states with features like literacy,Cleanlines,Crime Rate and targeting (prediction) Good or Bad state!

*Good is called Target variable here,it has values of 0s and 1s.

->Now let us create a Decision tree:

*Decision tree recurrently (continuously) splits the data until it gets pure leaves.

*Let us view a DT based on Crime Rate.

Building decision tree:

Node-1:CR>60

True=[C,Q,X=0] (pure leave).

False=A,E,F,G,I,K,L,N,P,R,U,V=0 [B,D,H,I,M,O,S,T,W,Y,Z=1] (Mixed leaf).

Reason: The mixed leaf has target variable both 0s and 1s.Hence this data is splited once again.

Node-2:CR>50

True=[A,E,F,G,I,K,N,P,R,U,V=O](pure leaf)

False=[B,D,H,J,M,O,S,T,W,Y,Z=1] (pure leaf)

Results:

Condition	Good
CR>60	0
CR<60	Can't be determined
CR>50	1

CR=63,Good=0

CR=45,Good=1

Step-1:

```
import pandas as pd
```

Step-2:reading the text file

```
df=pd.read_csv('/content/demodt.txt')
```

Step-3:

```
df.head()
```

Output:

	State	Literacy	Cleanliness	Crime_Rate	Good
0	A	92	90	54	0
1	B	56	67	50	1
2	C	78	85	62	0
3	D	63	72	48	1
4	E	85	79	55	0

Step-4:

```
from sklearn.tree import DecisionTreeClassifier
```

Step-5:

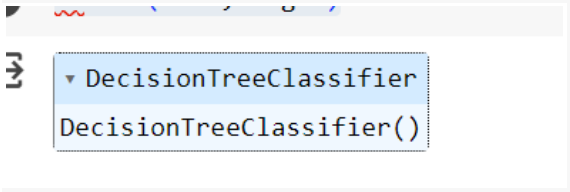
```
dt=DecisionTreeClassifier()
```

Step-6:

```
target=df.Good
 feat_list=['Literacy','Cleanliness','Crime_Rate']
 feat=df[feat_list]
```

```
dt.fit(feat,target)
```

Output:



Step-7:

```
pred=dt.predict([[90,90,63]])
print(pred)
```

Output:

0

```
#Predict:
if pred==[0]:
  print('Bad')
else:
  print('Good')
```

Output:

Bad

1.3Random_forest:

->Collection of marry decision trees!

->Keywords:Bootstrapping,Aggression.

Case study:

Y-target

X0,X1,X2,X3,X4-features

Bootstrapping:

Splitting the parent dataset into child datasets(having same no of rows and should have different row combination).

Convolutional Neural Networking:

● A convolutional NeuralNetwork (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision

● It consists of Three layers

1. **Input Layers:**

It's the layer in which we give input to our model.In CNN generally the input will be a image this layer holds the raw data

2. **Hidden layers:**

→Convolutional Layer

: ● This is the layer ,which is used to extract features from the input dataset .It applies a set of learnable filters known as the kernel to input images.The filters/kernel are smaller matrices usually 2x2, 3x3and 5x5 shape, it slides over the input image data and compute the dot product between kernel weight and the corresponding input image patch.

● The output of this layer is referred to as featured maps.Suppose we use a total of 12 filters for this layer we will get an output volume of dimensions 32x32x12.

→Activation Layer :

● By adding an activation function to the output of the preceding layer,activation layers add non linearly to the network.It i will apply an element-wise activation function to the output to the convolutional

● Thevolumeremains unchanged hence output volume will have dimensions 32*332*12.

→Pooling Layer:

● This layer is periodically inserted in the converts and its main function is to reduce the size of volume which makes the computation fast reduces memory and also

● If we use a max pool with 2x2 filters and stride 2,the resultant volume will be of dimension 16x16x12.

3. **Output layer:** The output from the fully connected layers is then fed into a logistic . classification tasks like sigmoid or softmax which converts the output

Activation Function:

❖ The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it.The purpose of the activation function is to introduce non linearity into the output of a neuron.

❖ We know the neural network has neurons that work in correspondence with weight bias and their respective activation function. In a neural network we would update the weights and biases of the neurons on the bias of the error at the output. This process is known as back propagation. Activation functions make the back propagation possible since the gradients are supplied along with the error to update the weights and biases.

● Tanh:

=>Tanh Function

Valin Range-130

future new

hes Usually ined in hulen layers of a neural network act) values les between 1th 1 hence the mean the the hidden layer comes out be Our very chine ic it, henting the data by bringing

● Sigmoid :

=>Sigmoid Function

It's a function which is plotted as a ‘s’ shaped graph.
Equation A=1/(1+ex)
Nature: Non-linear that X values lie between-2 to 2, Y values are very steep. This means small changes in x would also bring about large changes in the value Y
Value Range: 0 to 1.
Usually use tly used in cutput liver either or 1 of a binary as value for sigmoid classification, where lies between 0 and function lies be it can be predicted easily to be I if val and otherwise vallen

● Relu:

1. It stands for Rectified linear unit.It is the most widely used activation function.Chiefly implemented in hidden layers of Neural network.
2. Equation:-A(x)=max(0,x).It gives an output x if x is positive and 0 otherwise.
3. Value range:-[0,inf)
4. Nature:-Nonlinear which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the Relu function.
5. Uses:-Relu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations.At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.
6. In other words,RELU learns much faster than sigmoid and Tanh functions.

● Softmax:

1. The softmax function is also a type of sigmoid function but id handy where we are trying to handle multi class classification problems.
2. Nature: non-linear
3. Uses:- The softmax function was commonly found in the output layer of the image classification problems.the softmax function would squeeze the outputs for each class between 0 and 1 and would also dividenby the sum of the outputs. 4. Output: The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

If your output is for binary classification then,sigmoid function is a very natural choice for the output layer.

STATISTICAL ANALYSIS:

Statistical analysis using SciPy and statsmodels for hypothesis testing,regression analysis and ANOVA.

SciPy is an open-source scientific computing library for python that builds on Numpy.It provides many additional functionalities compared to Numpy,including optimization,integration,interpolation,eigenvalue problems,signal and Image processing,statistical distributions,and much more.

HYPOTHESIS TESTING(T-test):

1. T-test to compare the mean value of two groups :group A and group B
2.Performing the t-test stats.ttest_ind()
performs on independent samples t-test(2 sample t-test)
3.Tstat variable stores the tstatistical value which measures the differences between mean of the 2 groups relative to their variation within the groups
4.P-value variable stores the probability of observing the given result(less or more extreme under the null hypothesis(no difference between the group)).
5.If the p-value is predefined significance level example 0.05,we reject the null hypothesis,there is a statistically significant difference between the means of the 2 groups.

ANOVA(Analysis of variances)

->it is a statistical test used to compare the means of two or more groups to determine if there are any statistical difference between them it is an extension of the t-test
->it is allowing for the comparison of mean value across multiple groups simultaneously.

Null Hypothesis:

->The mean of all groups are equal

Alternative hypothesis:

->Atleast one group mean is different some others.

Assumptions:

Independent observation-Data point within each group are independent of each other.
Normality-Each group data follows a normal distribution.
Homogeneity of variances-The variances of the data in each group are equal.

ANOVA test statistic(f_statistic):

F-statistic is calculated based on the ratio of the between group variability to the within group variability
->If the f_statistic is large (greater than expected under Null hypothesis) it suggests that the group means are not equal.

P_value:

->P-value variable stores the probability of observing the given result(less or more extreme under the null hypothesis(no difference between the group)).
->If the p-value is predefined significance level example 0.05,we reject the null hypothesis,there is a statistically significant difference between the means of the 2 groups.

Interpolation:

->if the p_value is less than the significance level reject the Null hypothesis and conclude that there is significant difference between at least 1 pair of group means.
->If the p_value is greater than the significance level ,there is no difference.

Reinforcement.

Unsupervised.