

**DAY 1 : 23-02-2024**

**Review of core python concepts:datatypes,operators**

**,control flow,functions,modules,packages.**

**IDE-Integrated Development Environment(Google Colab)**

**Git hub repository creation(folder name-DATAANALYSIS)**

**Numpy:**

**Numpy is used for numerical or scientific computation in arrays,vectors,matrices.**

**NumPy is used for numerical or scientific computation in arrays,vectors and matrices.**

* **NumPy has many built-in mathematical and logical functions that operate on entire arrays such as sin, cos, exp, sqrt, log, abs, sum, min, max etc.**
* **It lets you create multidimensional arrays and perform operations on entire arrays rather than looping through each element one by one. This makes code faster and cleaner.**
* **NumPy has functions for linear algebra, Fourier transforms, random number generation, and more. These are useful for data science, engineering, and scientific applications.**

**As we are using google colab it is a default package available while we are using python in it. To reassure having NumPy package use the import statement at the beginning itself. It is as follows:**

**“Import numpy as np”**

**Now lets look into some functions that are included in the numpy package and are significantly used to perform operations on arrays and matrices.**

**1)np.array():It is a numpy function to create a new NumPy array from the elements passed to it.**

**eg:arr = np.array([1, 2, 3, 4, 5], dtype=float)**

**output:[1 2 3 4 5]**

**The dtype=float argument specifies the data type of the array. Here we explicitly defined it to**

**be float, otherwise integers would have been used by default.**

**2)np.arange(): It returns evenly spaced values within a specified interval as a numpy array.**

**eg:arrange\_a=np.arrange(10)**

**output:[0 1 2 3 4 5 6 7 8 9]**

**3)np.zeros(): It is useful Numpy functions to quickly initialize arrays with 0s.**

**eg:zeros\_arr=np.zeros((3,3),dtype=float)**

**output:[[0. 0. 0.]**

**[0. 0. 0.]**

**[0. 0. 0.]]**

**4)np.ones(): It is useful Numpy functions to quickly initialize arrays with 1s.**

**eg:ones\_arr=np.ones((3,3),dtype=float)**

**Output:{[1. 1.]**

**[1. 1.]]**

**5).reshape():Reshaping allows us to change the structure of the array without changing the data it**

**Contains.**

**eg:arr=arr.reshape(5,1)--->5-rows,1-column**

**Output:[[1.]**

**[2.]**

**[3.]**

**[4.]**

**[5.]]**

**6).slice():The slice function in NumPy allows you to slice or subset NumPy arrays by specifying start,**

**stop, and step values.**

**eg:sliced\_arr=arr[2:4]--->2-start,4-stop**

**output:[3. 4.]**

**7)np.split(): It provides is an easy way to chop up arrays into smaller array chunks for further analysis and processing.**

**eg:b=np.split(a,4)--->divides the array elements into 4 even subarrays.**

**output:[array([1, 2]), array([3, 4]), array([5, 6]), array([7, 8])]**

**8)ni.linalg.eig():This function calculates the eigenvalues and eigenvectors of the input matrix.**

**Eigenvalues and vectors are related to matrix diagonalization and have many scientific and**

**engineering applications.**

**eg:a=np.linalg.eig(c)----->eig is a method present in linalg sub package.**

**output:EigResult(eigenvalues=array([5.80198014e-02, 6.89419802e+01]), eigenvectors=array([[-0.75781077, -0.40313049],**

**[ 0.65247439, -0.91514251]]))**

**9)np.dot():This computes the matrix multiplication between the two arrays. The arrays must have**

**matching dimensions for the multiplication to be possible.**

**eg:np.dot(a,b)---->a,b are two vectors on which dot operation should be performed.**

**output:[[19 22]**

**[43 50]]**

**10)np.loadtxt & np.savetxt:np.loadtxt and np.savetxt allow loading and saving of numpy array data from/to files - extremely useful for data input/output.**

**eg:datadata=np.loadtxt("/content/drive/MyDrive/dataset/dap.txt.txt",dtype=int)**

**d = np.savetxt("/content/date.txt",data)**

**print(d)**

**print(data)**

**Output:[1 2 3 4 5 6]**

**11)np.random.random\_integers(): generates random integers between a range**

**The first two arguments specify the minimum and maximum integer (inclusive)**

**Size specifies number of random numbers to generate**

**eg:a=np.random.random\_integers(0,2)**

**output:0**

**12)(.shape & .ndim):These attributes on any NumPy array returns its dimensionality and size along each dimension.**

**eg:a=np.array([[1,2],[3,4]])**

**print(a)**

**print(a.ndim)**

**print(a.shape)**

**output:[[1 2]**

**[3 4]]**

**2**

**(2, 2)**

**13)np.linespace():returns evenly spaced numbers over a specified interval.The first two arguments are the start and end of the sequence.The third argument specifies the number of samples to generate.**

**eg:b=np.linespace(0.8,2,5)**

**output:print(b)**

**[0.8 1.1 1.4 1.7 2. ]**

**14).sum(),sum(axis=0),sum(axis=1):lets you easily sum either the entire array, or along a particular dimension by specifying the axis parameter.axis=0 iterates through columns,axis=1goes**

**through rows for summation.**

**Pandas-loading the dataset,viewing dataset,performing pre processing operations**

**eg:print(a.sum())**

**Output:6**

**eg:print(a.sum(axis=0))**

**Output:[2,2,2]**

**eg:print(a.sum(axis=1))**

**Output:[3,3]**

**15)max(),max(axis=0),max(axis=1):lets you easily get maximum values either of the entire array, or along a dimension by specifying axis.**

**eg:=a=np.array([[12,4],[7,8],[99,45]])**

**print(a)**

**print(a.max())**

**print(a.max(axis=1))**

**print(a.max(axis=0))**

**output:[[12 4]**

**[ 7 8]**

**[99 45]]**

**99**

**[12 8 99]**

**[99 45]**

**16).cumsum(),cumsum(axis=0),cumsum(axis=1): calculates cumulative sum along an axis.**

**cumsum(axis=0), calculates cumulative sum along each row,cumsum(axis=1).**

**eg:a=np.array([[2,3],[5,6]])**

**print(a)**

**print(a.cumsum(axis=1))**

**print(a.cumsum(axis=0))**

**output:[[2 3]**

**[5 6]]**

**[[ 2 5]**

**[ 5 11]]**

**[[2 3]**

**[7 9]]**

**17)vstack(),hstack(),dstack(): vstack stacks arrays vertically,hstack stacks arrays horizontally.**

**Dstack-number of rows become number of groups,columns becomes rows,groups becomes columns**

**eg:a=np.array([2,5,3,4,7,8])**

**a.resize(2,3)**

**b=np.array([2,7,9,5,6,7])**

**b.resize(2,3)**

**print(np.vstack((a,b)))**

**print(np.hstack((a,b)))**

**output:[[2 5 3]**

**[4 7 8]**

**[2 7 9]**

**[5 6 7]]**

**[[2 5 3 2 7 9]**

**[4 7 8 5 6 7]]**

**eg:a=np.arange(30).reshape(2,3,5)**

**print(a)**

**print(np.dstack(a))**

**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]]]**

**[[[ 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]]]**

**18)eye() function:- It is used to return a two-dimensional array with ones (1) on the diagonal and zeros (0) elsewhere.**

**19)full() function :- It is used to return a new array of a given shape and data type filled with fill\_value .**

**.itemsize function:-itemsize returns the size (in bytes) of each element of a NumPy array.**

**Eg:- import numpy as np**

**a1=np.eye(4)**

**print(a1)**

**a1 = np.full((3,3),4)**

**print(a1)**

**print(a1.itemsize)**

**Output:-[[1. 0. 0. 0.]**

**[0. 1. 0. 0.]**

**[0. 0. 1. 0.]**

**[0. 0. 0. 1.]]**

**[[4 4 4]**

**[4 4 4]**

**[4 4 4]]**

**8**

**20)asarray()function :- It is used when we want to convert the input to an array.**

**21)type():- It is a built-in function that returns the type of the objects/data elements stored in any data type or returns a new type object depending on the arguments passed to the function.**

**Eg:- x=[1 ,2,3,4,5]**

**#symmetric matrix**

**a=np.asarray(x)**

**print(a)**

**print(type(a))**

**Output:- [1 2 3 4 5]**

**<class 'numpy.ndarray'>**

**22) Inner function:- Inner functions, also known as nested functions, are functions that you define inside other functions.**

**Outer Function (Enclosing Function): -This is the function that contains another function, often referred to as the inner function.**

**23) Cross function:- The cross product of a and b in is a vector perpendicular to both a and b. If a and b are arrays of vectors, the vectors are defined by the last axis of a and b by default, and these axes can have dimensions 2 or 3.**

**Eg:- x=np.array([1,4,0],float)**

**y=np.array([2,2,1],float)**

**print(np.inner(x,y))**

**print(np.outer(x,y))**

**print(np.cross(x,y))**

**Output:- 10.0**

**[[2. 2. 1.]**

**[8. 8. 4.]**

**[0. 0. 0.]]**

**[ 4. -1. -6.]**

**24) Rint function:-The rint function is used to round numerical values to the nearest integer. It comes as part of NumPy, which is a library of the high-level coding language, Python.**

**Eg:- a =np.array([1,2,3,4,5,6,7] ,dtype=float)**

**np.rint(a)**

**Output:- array([1., 2., 3., 4., 5., 6., 7.])**

**25)True\_divide:- Returns a true division of the inputs, element-wise. Instead of the Python traditional 'floor division', this returns a true division. True division adjusts the output type to present the best answer, regardless of input types.**

**26)unique() function:- It finds the unique elements of an array and returns these unique elements as a sorted array.**

**27)union1d() function:- It finds the union of two arrays and return the unique, sorted array of values that are in either of the two input arrays.**

**intersection() function:- It is a built-in set method used to find the common elements between two or more sets. It returns a new set containing elements present in all input sets.**

**28)Set difference() method:- It returns a set that contains the difference between two sets. Meaning: The returned set contains items that exist only in the first set, and not in both sets.**

**Eg :- #true\_divide function**

**a =np.array([1,2,3,4,5] ,dtype=float)**

**b=np.array([6,7,8,9,2] ,dtype=float)**

**np.true\_divide(a,b)**

**#unique function**

**c =np.array([1,1,2,2,3,3] ,dtype=int)**

**np.unique(c)**

**#union1d function**

**a=np.array([[2,2],[3,3]])**

**b=np.array([[5,5],[6,6]])**

**np.union1d(a,b)**

**#intersect function**

**np.intersect1d(a,b)**

**#set difference function**

**np.setdiff1d(a,b)**

**Output:- array([0.16666667, 0.28571429, 0.375 , 0.44444444, 2.5 ])**

**array([1, 2, 3])**

**array([2, 3, 5, 6])**

**29)Hypothesis function:- It helps in making an observation and experiments possible. It becomes the start point for the investigation. Hypothesis helps in verifying the observations. It helps in directing the inquiries in the right direction.**

**Eg:- a=8**

**b=6**

**c=np.hypot(a,b)**

**print(c)**

**Output:- 10.0**

**array([], dtype=int64)**

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

**Divmod function:- The divmod function in Python returns the quotient and remainder as output as a tuple.**

**Mod function():-The % symbol in Python is called the Modulo Operator. It returns the remainder of dividing the left hand operand by right hand operand.**

**Div function():- the method in Python is used to perform division operations on the DataFrame.**

**Eg:- #divmod() Function quoitent and remainder will consider**

**a = np.array([10,21,30,40,50,60])**

**b = np.array([20,21,2,20,25,25])**

**c=np.divmod(a,b)**

**print(c)**

**#only remainder will consider**

**c=np.mod(a,b)**

**print(c)**

**#quoitents in float**

**c=np.divide(a,b)**

**print(c)**

**#multiply the array**

**c=np.multiply(a,b)**

**print(c)**

**Output:- (array([ 0, 1, 15, 2, 2, 2]), array([10, 0, 0, 0, 0, 10]))**

**[10 0 0 0 0 10]**

**[ 0.5 1. 15. 2. 2. 2.4]**

**[ 200 441 60 800 1250 1500]**

**random .normal:-creates an array of specified shape and fills it with random values which is actually a part of Normal(Gaussian)Distribution.**

**Eg:- from numpy import random**

**x=random.normal(size=(2,3))**

**print(x)**

**x=random.normal(loc=1,scale=2,size=(2,3))**

**print(x)**

**Output:- [[ 0.299838 0.01567101 -0.25837536]**

**[ 1.76202508 0.81030512 -0.72923644]]**

**[[1.65479787 0.63090917 1.32747607]**

**[0.14878016 1.33332735 0.38140278]]**

**Random .binomial():- random. binomial(n, p, size=None) Draw samples from a binomial distribution. Samples are drawn from a binomial distribution with specified parameters, n trials and p probability of success where n an integer >= 0 and p is in the interval [0,1]. ( n may be input as a float, but it is truncated to an integer in use). Random.poisson():- The random. poisson method returns an array of length n . This is the bunch of samples drawn from the Poisson distribution modeled with Lambda equal to l .**

**Random .choice():-The choices() method returns a list with the randomly selected element from the specified sequence.**

**Eg:- from numpy import random**

**x=random.binomial(n=12,p=0.5,size=10)**

**print(x)**

**x=np.random.poisson(lam=2,size=10)**

**print(x)**

**x =np.random.choice([2, 4])**

**print(x)**

**y=np.random.choice([3, 5], p=[0.5, 0.5], size=(3, 5))**

**print(y)**

**Output:- [4 5 2 6 6 5 6 4 8 7]**

**[7 2 1 3 2 3 1 3 5 1]**

**4**

**[[3 3 3 5 5]**

**[3 5 3 5 3]**

**PANDAS:**

**1) Pandas are used by using import the pandas modules.**

**Ex:- import pandas as pd**

**2) Series() Method:- This method is used to print the data in the form of a series.**

**Ex:**

**import pandas as pd**

**a=[“pandas”,”modules”,”method”]**

**r=pd.Series(a,index=[1,2,3])**

**print(r)**

**3)To read the data form by using two ways**

**1)csv**

**2)excel**

**Ex:**

**t=pd.read\_cvs(“file path”)**

**t=pd.read\_excel(“file path”)**

**4) head() Method :- This Method is used to print the first 5 lines in the file by default.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**print(df.head())**

**// first 5 lines**

**print(df.head(10))**

**// first 10 lines**

**5) tail() Method :- This Method is used to print the last 5 lines in the file by default.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**print(df.tail()) // last 5 lines**

**print(df.tail(10))**

**// last 10 lines**

**6) describe :- This method is used to calculate the mean,std,count ect in the column wise.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**print(df.describe)**

**7) describe().T :- It is used to transpose of the describe.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**print(df.describe().T)**

**8) shape Method:- This method is used to print the number of rows,columns.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**print(df.shape)**

**//prints no.of rows,columns.**

**print(df.shape[0])**

**print(df.shape[1])**

**//prints the rows.**

**//prints the columns.**

**9) columns:- prints the column names in list form.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**print(df.columns)**

**10) copy() Method:- Copy the one dataframe to another dataframe.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**df1=df.copy()**

**print(df1)**

**11) isnull() Method:- This method is used if any column has none value in that place it**

**returns the True otherwise False.**

**Ex:**

**12) Slicing :**

**import pandas as pd**

**df1 = pd.read\_cvs(“file path”)**

**t=df1.isnull()**

**p=df1.isnull().head()**

**q=df1.isnull().tail()**

**r=df1.isnull().sum()**

**import pandas as pd**

**df1 = pd.read\_cvs(“file path”)**

**t=df1[‘column name’]**

**//prints the particular column.**

**p=df1[[‘column1 name’,’column2 name’......]] //prints the particular n column.**

**q=df1[df1.index==1]**

**//prints the particular row.**

**print(df1[df1.index.isin(range(2,5)]) //prints the 2 to 4 rows with columns**

**13) loc:**

**1) loc[row number] //returns the particular row number.**

**2) loc[start:end,”column\_name”]**

**// prints the start index to end index of a**

**particular column.**

**3) loc[[row1 number,row2 number….]]**

**//returns the n row number.**

**4) loc[ start:end,[”column1\_name”,”column2\_name”,.......]]**

**// prints the start index to end index of a particular column.**

**14) dropna() Method:- It is used to delete the missing data in the data frame.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**df1=df.copy()**

**t=df1.dropna()**

**//delete temporary**

**print(t)**

**t=df1.dropna(inplace=True)**

**//delete permanently**

**t=df1.dropna(inplace=True,axis=1) //delete permanently the column**

**t=df1.dropna(inplace=Trueaxis=0) ////delete permanently the row**

**15) fillna() Method:- It is used to fill the data with the default value of missing data in the**

**data frame.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**df1=df.copy()**

**print(df1.fillna(10))**

**16) drop\_duplicates() Method:- This method deletes the duplicate values in a given**

**table.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**df1=df.copy()**

**df1.drop\_duplicates(inplace=True)**

**print(df1)**

**17) rename() Method:- rename the given column.**

**Ex:**

**import pandas as pd**

**df = pd.read\_cvs(“file path”)**

**df1=df.copy()**

**df1.rename(columns = {‘Grade’:’GPA’},inplace=True)**

**print(df1)**

**DAY:- 3 (26-02-2024)**

**MATPLOTLIB :**

**Most of the Matplotlib utilities lies under the pyplot submodule, and are usually**

**imported under the plt**

**Import matplotlib.pyplot as plt**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**plt.plot(x,y)**

**plt.show()**

**1) The scatter() function plots one dot for each observation. It needs two arrays of the same**

**length, one for the values of the x-axis, and one for values on the y-axis.**

**Ex:- Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**plt.scatter(x,y)**

**plt.show()**

**2) color or the c argument**

**Ex:- Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**plt.scatter(x,y,color=”yellow”)**

**plt.show()**

**3) size argument :**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**s=np.array[1,2,3,4,5]**

**plt.scatter(x,y,size=s)**

**plt.show()**

**4) marker to emphasise each point with a specified marker**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**plt.scatter(x,y,marker=’o’)**

**plt.show()**

**5) shortcut string notation parameter to specify the marker.This parameter is also called**

**fmt, and is written with this syntax:**

**Marker|line|color**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**plt.scatter(x,y,o:r) // marker=o,line=dotted,color=red**

**plt.scatter(x,y,o-r) // line=solid line**

**plt.scatter(x,y,o--r) // line=dashed line**

**plt.scatter(x,y,o-.r) // line=dashed,dotted**

**plt.show()**

**6) Marker size(ms):**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**plt.scatter(x,y,marker=’o’,ms=5)**

**plt.show()**

**7)markeredgecolor (mec):**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**plt.scatter(x,y,marker=’o’,ms=5,mec=’r’)**

**plt.show()**

**8)markerfacecolor (mfc):**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array[1,2,3,4,5]**

**y=np.array[1,2,3,4,5]**

**plt.scatter(x,y,marker=’o’,ms=5,mfc=’r’)**

**plt.show()**

**9) line arguments:**

**1) linestyle:- plt.scatter(x,y,linestyle=’dotted’)**

**2) linecolor:- plt.scatter(x,y,color=”yellow”)**

**3) linewidth:- plt.scatter(x,y,linewidth=5)**

**10)xlabel() :- give name to x axis Ex:- plt.xlabel(“x label name”)**

**11)ylabel() :- give name to y axis Ex:- plt.ylabel(“y label name”)**

**12)title() :- give the title for the graph Ex:- plt.title(“title name”)**

**13)pie():-function to draw pie charts**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array([1,2,3,4,5])**

**plt.pie(x)**

**plt.show()**

**14)label parameter:-must be an array with one label for each wedge**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array([1,2,3,4,5])**

**y=[“1”,”2”,3”,”4”,”5”]**

**plt.pie(x,labels=y)**

**plt.show()**

**15)The explode parameter, if specified, and not none, must be an array with one value for**

**each wedge**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array([1,2,3,4,5])**

**y=[“1”,”2”,3”,”4”,”5”]**

**z=[0.2,0,0,0,0]**

**plt.pie(x,labels=y,explode=z)**

**plt.show()**

**16) shadow parameter:**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array([1,2,3,4,5])**

**y=[“1”,”2”,3”,”4”,”5”]**

**z=[0.2,0,0,0,0]**

**plt.pie(x,labels=y,explode=z,shadow=True)**

**plt.show()**

**17) color parameter:**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array([1,2,3,4,5])**

**y=[“1”,”2”,3”,”4”,”5”]**

**p=[“black”,”yellow”,”red”,”blue”,”green”]**

**z=[0.2,0,0,0,0]**

**plt.pie(x,labels=y,colors=p)**

**plt.show()**

**14)bar():-function to draw bar graphs**

**Import matplotlib.pyplot as plt**

**Import numpy as np**

**x=np.array([1,2,3,4,5])**

**plt.bar(x)**

**plt.show()**

**SEABORN:This library based on matplotlib**

**1)by importing seaborn as sns**

**Ex:**

**Import seaborn as sns**

**2) we load the data set by using**

**Import seaborn as sns**

**sns.load\_dataset(“dataset name”)**

**3) different methods:**

**1)barplot():- bar graph**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**sns.load\_dataset(“tips”)**

**sns.barplot(x=”days”,y=”total\_bill”,data=”iris”)**

**plt.show()**

**2)boxplot():**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**sns.load\_dataset(“tips”)**

**sns.boxplot(x=”days”,y=”total\_bill”,data=”iris”)**

**plt.show()**

**3)violin plot()**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**sns.load\_dataset(“tips”)**

**sns.violinplot(x=”days”,y=”total\_bill”,data=”iris”)**

**plt.show()**

**4)lineplot()**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**sns.load\_dataset(“tips”)**

**sns.lineplot(x=”days”,y=”total\_bill”,data=”iris”)**

**plt.show()**

**5)heatmap()**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**sns.load\_dataset(“tips”)**

**sns.heatmap(x=”days”,y=”total\_bill”,data=”iris”)**

**plt.show()**

**6)jointplot()**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**sns.load\_dataset(“tips”)**

**sns.jointplot(x=”days”,y=”total\_bill”,data=”iris”)**

**plt.show()**

**6)countplot()**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**sns.load\_dataset(“tips”)**

**sns.countplot(x=”days”,y=”total\_bill”,data=”iris”)**

**plt.show()**

**7)lm plot()**

**Ex:**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**sns.load\_dataset(“tips”)**

**sns.lmplot(x=”days”,y=”total\_bill”,data=”iris”**

**plt.show()**

**DAY 4 : 27-02-2024**

**Neural networks serves as neurons**

**Neural networks contains three layers**

**->input layer**

**->hidden layer**

**->output layer**

**hidden layer-convolutional,max pooling,dense layers**

**(max) pooling-max pooling,avg pooling layer**

**image with width 32,height 32,depth 3**

**\*\*convolution layer- used to extract the features from input dataset**

**appilies set of learnable filters called kernels**

**kernels are smaller matrices usually 2x2 3x3 5x5 shape**

**the output of this layer is reffered asd feature maps**

**the size will be 32x32x12**

**\*\*Activation layer**

**By adding an activation function to the output of the preceeding layer,activation layer add non linearity to the network**

**It will apply an element wise activation function to the output of the convolutional layer**

**The volume remains unchanged hence the output vlume will have dimensions 32x32x12**

**\*\*Pooling layer**

**This layer is periodically inserted in the covnets and its main function is to reduce the size of the volume which makes the computation fast**

**reduces memory and also prevents overfitting**

**Two common types of pooling layer are max pooling and avg pooling**

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

**\*\*Dense layer**

**last layer of hidden layer**

**used to process the output**

**\*\*Output layer**

**The output from the fully connected layer is then fed in to a logistic function for classification tasks**

**like sigmoid or softmax which converts the ouput of each class into the probability score of the each class**

**\*\*Activation Function**

**The activation function decides whether a neuron should be activated or not by calculate the weighted sum and further adding bias to it**

**The purpose of the activation function is to function is to intoduce 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 nurons on the biases of the eeror at the ouput**

**This process is known as back propogation**

**//activation function make the back-propagation possible since the gradients are suppiled along with the error to update the weights and biases**

**\*Tanh**

**\*Sigmoid**

**\*Relu**

**\*Softmax**

**\*\*Sigmoid Function**

**It is a function which is plotted as "S" shaped graph**

**eq:A=1/(1+e^-x)**

**Nature:Non-linear**

**notice X value btw -2 to +2**

**value range:0 to 1**

**Uses:Usaually used in output layer of a binary classification,where result is either 0 or 1,as value for sigmoid function lies between 0 and 1 only**

**so,results can be predicted easily to be 1 if value is greater than 0.5 and 0 otherwise**

**Mostly used in output layer but we can use in hidden layer**

**\*\*Tanh Function**

**Value range:-1 to +1**

**nature:Non-linear**

**Uses:Usually used in hidden layers of a netral networks hence its values ranges between -1 to +1**

**\*\*Relu function**

**It stands for rectified linear unit.mostly used in activation func .chiefly implemented in hidden layer of Neural Network**

**Eq:A(x)=max(0,x).it gives op x if x is+ve and 0 otherwise**

**value range:[0,inf)**

**Non-Linear(Rectified linear)we can easily back propagate the function and have multiple layers of neurons being activated by the relu function**

**Uses:Computationally expenses is more than Tanh function and sigmoid because it invloves simpler mathematcal operations**

**at a time oly a few neurons are activated making the network spare making it efficient and easy for computation**

**In simple words,Relu learns much faster than sigmoid and Tanh function**

**\*\*Softmax**

**The softmax function is also a type of sigmoid function but is handy when we trying to handle multi-class classification problems**

**Nature:Non-linear**

**Uses:The softmax func commonly is found in o/p layer of image classification problem.The softmax func would squeeze the o/p f each class**

**between 0 and 1 would also divide by the sum of the o/p**

**Output:The softmax func is ideally used in the o/p layer of the classifier where we are actuallt trying to attain the probailities to define the class of each input**

**The basic rule of thumb if you don't know which function is to use just simply use Relu function**

**\*\*RNN**

**simple rnn layer**

**activation layer**

**dense layer**

**.fitten() is used for reshape and to resize all the images to same size**

**Statistical analysis using scipy and ststsmodels for hypothesis testing,regression analysis and ANOVA**

**Scipy ia an open source scientific computing library for python that builds on Numpy.it provides many additional functionalities compared to Numpy,including optimization,integration,eigen value problems,signal and image processing,statistical distributions and much more**

**Statsmodels is a python library that provides classes and functions for estimating and testing statistical models.it is built on top of Numpy,scipy,and matplotlib,and it integrates with pandas for data handling.statsmodels includes a wide range of statistical models and tests,making it a powerful tool for statistical analysis and hypothesis testing**

**HYPOTHESIS TESTING:**

**If the pvalue is less than an pre determined significant level(such as 0.05) it is typically interpreted as sufficient evidence to reject the null hypothesis**

**If the p value is greater then the significant level there is not enough evidence to reject the null hypothesis**

**Together t\_static and p\_value it is determining whether the observed sample data provides enough evidence to support a claim on hypothesis**

**About the population.**

HADOOP LIBRARY:

it is an open source framewotrk designed to process and store big data in a distributed environment .

programming model is a simple one called as map reduce and distributed file sysytem it is called HDFS[HADOOP DISTRIBUTED FILE SYSTEM].

system that stores data across multilple machines it provides highend accesss to application data

MAP REDUCE:

A programming model for processing and generating large data set it i9nvolves two phases ma

MAP PHASE-where data is divided into smaller chunks and proceesed in parallel

REDUCE PHASE

where result from the map phase are aggregated to reduce the final output

YARN:[yET ANOTHER RESOURCE NEGOTIATOR]

It is a resource management layer that manages resources in hadoop clusters and scheduling user application

EX1:

word count using HADOOP:

from the larger collection of document we are going to count number of words by using hadoop.

step1:INPUT DATA

we have large collection of data stored in HTFS

STEP2:MAP PHASE

In this phase each document is divided into words,each word is emitted as a key-value pair[word,one]word is the key and one is the value.

STEP3: SHUFFLE AND SORT PHASE:

The output of the map phase is sorted based on the keys all values corresponding to the same key are grouped togeather.

STEP4: REDUCEPHASE

IN this phase each unique word is passed to a reducer . the reducer sums up the value corresponding to each word giving the count of occurence of that word.

the output of the redce phase is the final word count .

STEP5:OUTPUT DATA

the final output containing is stored in HDFS or in any other safe location

CONCLUSION:

Hadoop provides scalable and cost effective solution for processing and storing big data by distrbuting data and computation across multiple rows

it enables the processing of large data set

EX:

word count

PINCIPLE: hadoop map reduce framework.