DAY 1: 23-02-2024

Review of core Python concepts: data types, operators, control flow, functions, modules, packages

1. Data Types:

Python supports various data types including:

Numeric types: Integers (int), Floats (float), Complex numbers (complex)

Sequence types: Lists (list), Tuples (tuple), Strings (str)

Mapping type: Dictionary (dict)

Set types: Set (set), Frozen set (frozenset)

Boolean type: Boolean (bool)

None type: None (NoneType)

2. Operators:

Python supports different types of operators:

Arithmetic operators: +, -, , /, %, //, *

Comparison operators: ==, !=, <, >, <=, >=

Assignment operators: =, +=, -=, =, /=, %=, //=, *=

Logical operators: and, or, not

Bitwise operators: &, |, ^, ~, <<, >>

Membership operators: in, not in

Identity operators: is, is not

3. Control Flow:

Python provides several control flow statements:

Conditional statements: if, elif, else

Looping statements: for, while

Control flow statements: break, continue, pass

Exception handling: try, except, finally, raise

4. Functions:

Functions in Python are defined using the def keyword and can take parameters and return values. Key aspects include:

Parameter passing: Parameters can be positional or keyword.

Variable number of arguments: *args and **kwargs.

Anonymous functions: lambda functions.

Scope: Variables defined inside a function have local scope by default.

5. Modules:

Modules in Python are simply Python files with a .py extension. They can contain functions, classes, and variables. You can import modules using the import statement.

6. Packages:

Packages are a way of organizing modules into a hierarchical structure. A package is essentially a directory containing Python modules and a special _init_.py file. You can import packages using dot notation.

- **IDE:-** integrated Development Environment (Google CoLab)
- GIT Hub repository creation(Folder Name-Data Analysis)
- **Numpy**: Numpy is for numerical or scientific computations in arrays,vectors and Matrics
- To import a numpy by import numpy as np
- To create an array by using numpy like

  Ex:- import numpy as np
  a=np.array([1,2,3,4])   //np is short form of numpy

1) **ones() Method :-** Print/Display the array with Ones(1's) by using ones() method

Ex:- a = np.ones((r,c),dtype= data type)
-> r = number of rows
-> c = number of columns
->dtype means data type (int,float etc….)

Ex:- import numpy as np
a=np.ones((3,3),dtype=int)

2) **zeros() Method :-** Same like we can print the array with Zeros(0's) by using zeros() method

Ex:- a = np.zeros((r,c),dtype= data type)
-> r = number of rows
-> c = number of columns
-> dtype means data type (int,float etc….)

Ex:- import numpy as np
a=np.zeros((3,3),dtype=int)

NOTE:- If doesn't consider the dtype(data type) by default it can prints in float

3) **arange() Method:-** It is used to to arrange the number from specific number to specific number

Ex:- a=np.arange(x,y,z,dtype= data type)
-> x = Starting number
-> y = Ending number
-> z = Step number
-> dtype means data type (int,float etc….)

Ex:- import numpy as np
a=np.arange(10)
a=np.arange(1,20)
a=np.arange(1,20,5)
a=np.arange(1,20,5,dtype=int)

4) **reshape() Method:-** This method is used to change the shape of the array.

Ex:- b=a.reshape(r,c)
-> a = created array
-> r = number of rows
-> c = number of columns

Ex:- import numpy as np

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

5) **split() Method:-** Split method is used split the array in number of times

Ex:- `b=np.split(a,n)`
-> a = created array
-> n = number of divisions/split

6)**T (Transpose):-** Returns Transpose of the matrix

Ex:- `b=a.T`
-> a = created array

7)**.dot() Method:-** This method returns the matrix multiplication of given matrices

Ex:- `c=np.dot(a,b)`
-> a=first matrix stored variable
-> b=second matrix stored variable

8)**linalg.eig() Method (Linear algebra eigenvalues):-** Compute the eigenvalues and right eigenvectors of a square array.

Ex:- `b=np.linalg.eig(a)`
-> a = created array

9) **random.rand():-**This method is used to print random number

Ex:-
```
import numpy as np
a=np.random.rand()
```

10) **random.randint() Method:-** It Returns the random value between the values

Ex:-
```
import numpy as np
a=np.random.randint(2,10)
```

11) **type() Method:-** Returns the which type of the variable

Ex:-
```
import numpy as np
a=np.array([1,2,3,4])
print(type(a))
```

12) **ndim**(Number of dimensions):- To give the dimension of array.

Ex:-
```
import numpy as np
a=np.array([1,2,3,4])
print(a.ndim)
```

13)**shape** :- Returns the shape of the array.

```
import numpy as np
a=np.array([1,2,3,4])
```

```
print(a.shape)
```

14) **linspace()**(Line space) **Method:-** Create array filled evenly spaced values.

Ex:-
```
import numpy as np
a=np.linspace(0.8,2,5)
print(a)
```

15) If we can use **'*'** to multiply the elements in matrices.

Ex:-
```
import numpy as np
a=np.array([1,2,3,4])
b=np.array([5,6,7,8])
c=a*b
```

16) If we can use **'@ , .dot()'** to perform the matrix multiplication.

Ex:-
```
import numpy as np
a=np.array([1,2,3,4])
b=np.array([5,6,7,8])
c=a@b
```

17) **sum():-** To perform the addition between the 2 arrays.

Ex:-
```
import numpy as np
a=np.ones((3,3),dtype=int)
print(a.sum())
```

18) **axis:-** 1) If axis=1 refers to the row
2) If axis=0 refers to the column

Ex:-
```
import numpy as np
a=np.ones((3,3),dtype=int)
print(a.sum(axis=1))  //row
print(a.sum(axis=0))  //column
```

19) **floor() method :-** Returns the value of the ignores the decimal values
20) **ceil() method :-** Returns the value of the round the decimal value and returns
21) **max() Method:-** Returns the maximum value of the array
22) **min() Method:-** Returns the minimum value of the array
23) **cumsum() Method:-** Returns the cumulative sum of array/matrix
24) **resize() Method:-** we can modify the matrix /array size.

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5,6])
p=a.resize(2,3)
```

25) **vstack()**:-This method is used to join the two maurices in vertical order

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5,6])
```

```
b=np.array([7,8,9,10,11,12])
a.resize(2,3)
a.resize(2,3)
p=np.vstack(a,b)
```

26) **hstack()**:-This method is used to join the two maurices in horizontal order

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5,6])
b=np.array([7,8,9,10,11,12])
a.resize(2,3)
a.resize(2,3)
p=np.hstack(a,b)
```

27) **dstack() Method**:- number of rows become number of groups number of columns become rows group become column

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5,6])
b=np.array([7,8,9,10,11,12])
a.resize(2,3)
a.resize(2,3)
p=np.dstack(a,b)
```

DAY 2:-(24-02-2024)

1) **eye() Method:-** This method is used to print the diagonal values with **"1"**

Ex:-
```
import numpy as np
a=np.eye(4)
print(a)
```

2) **full() Method:-** This method is used to fill the given matrix with given number

Ex:-
```
import numpy as np
a=np.full([2,2],2)
print(a)
```

3) **asarray() Method:-** This is a symmetric array of the array. It is one of the method to create the array

Ex:-
```
import numpy as np
x=[1,2,3,4,5]
a=np.asarray(x)
print(a)
```

4) **inner() method :-** It is the sum of the element wise multiplication in an array.
5) **outer() method :-** The 1st array element is multiplied with the 2nd array of all elements.
6) **cross() Method :-** cross multiplication of 1st array and 2nd array

7) **rint() Method:-** This method is returns the round the elements in array to nearest integer
8)**true_divide() method** :- This method is used for the 1st element array with 2nd array elements wise.

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5])
b=np.array([2,3,4,5,6])
c=np.true_divide(a,b)
print(c)
```

9) **unique()** :- this method removes all duplicate values and prints.

Ex:-
```
import numpy as np
a=np.array([1,1,2,2,3,3,4,4,5,5])
b=np.unique(a)
print(b)
```

10) **union1d() Method** :- It returns the to join the all array without the duplicate values and prints them in one array.

Ex:-
```
import numpy as np
a=np.array([[1,2,3,4,5],[8,4,9,8,1]])
b=np.array([[2,3,4,5,6],[4,6,8,9,3]])
c=np.union1d(a,b)
print(c)
```

11) **intersect1d() Method** :- It returns the to common elements in two arrays

Ex:-
```
import numpy as np
a=np.array([[1,2,3,4,5],[8,4,9,8,1]])
b=np.array([[2,3,4,5,6],[4,6,8,9,3]])
c=np.intersect1d(a,b)
print(c)
```

12) **setdiff1d() Method** :- It returns the first array elements without which element is common in both arrays.

Ex:-
```
import numpy as np
a=np.array([[1,2,3,4,5],[8,4,9,8,1]])
b=np.array([[2,3,4,5,6],[4,6,8,9,3]])
c=np.setdiff1d(a,b)
print(c)
```

13)**hypot() Method**:- This method is works on hypotenuse formula

Ex:-
```
import numpy as np
a=8
b=6
c=np.hypot(a,b)
print(c)
```

14) **sin() Method** :- It can return the sin of values.

```
import numpy as np
x=90
c=np.sin(x)
print(c)
```

15) **divmod() Method** :- It can print the two arrays. The 1st array is division of both arrays in element wise, 2nd array is modular division of both arrays by element wise.

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5])
b=np.array([2,3,4,5,6])
c=np.divmod(a,b)
print(c)
```

16) **mod() Method** :- It can print the modular division (remainder) of both arrays in element wise.

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5])
b=np.array([2,3,4,5,6])
c=np.mod(a,b)
print(c)
```

17) **div() Method** :- It can print the exact division of both arrays in element wise

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5])
b=np.array([2,3,4,5,6])
c=np.div(a,b)
print(c)
```

18) **multiply() Method**:- This method is used to multiply the elements in arrays in element wise.

Ex:-
```
import numpy as np
a=np.array([1,2,3,4,5])
b=np.array([2,3,4,5,6])
c=np.mul(a,b)
print(c)
```

19) **random.normal() Method**:- This method returns random values in the given size of the matrix.

Ex:-
```
from numpy import random
a=random.normal(size(2,3))
print(a)
a=random.normal(loc=1,scale=2,size(2,3))
->loc=mean
```

->scale=standard deviation

20) **random.binomial() Method**:-

Ex:-
```
from numpy import random
a=random.binomial(n=12,p=0.5,size=10)
print(a)
```

21) **random.poisson() Method**:-

Ex:-
```
from numpy import random
a=random.binomial(lam=2,size=10)
print(a)
```
-> lam=lamida

22)**random.choice() Method**:- This method is used to print the random value in a given range.

Ex:-
```
from numpy import random
a=random.choice([1,2,3,4,5,6,7,8,9,10])
print(a)
```

DAY 2:-(24-02-2024)

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”)
Ex:-     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])       //prints the rows.
print(df.shape[1])       //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:-
```
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()
```

12) **Slicing :-**
```
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:

- ➢ Matplotlib is a versatile Python library used for data visualisation in fields like data analysis, scientific computing, and machine learning.
- ➢ It offers various plot types, including line plots, scatter plots, and histograms, making it suitable for exploratory data analysis.
- ➢ Matplotlib is also employed in machine learning projects for visualising model performance and feature distributions.
- ➢ Additionally, it supports geospatial data visualisation and interactive plotting capabilities, enabling users to create maps and dynamic visualisations.
- ➢ Overall, its flexibility, extensive functionality, and ease of use make it a popular choice.
- ➢ To import matplotlib library the syntax is:
  - **"import matplotlib.pyplot as plt" .**

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:

- ➢ Seaborn, a Python data visualisation library, simplifies the creation of statistical graphics for data exploration and analysis.
- ➢ It offers high-level interfaces for visualising relationships between variables, handling categorical data, and creating heatmaps and matrices.
- ➢ It provides tools for visualising time series data and customising plot aesthetics.
- ➢ Seamlessly integrating with Pandas, Seaborn enables users to leverage its visualisation capabilities directly on DataFrame objects.
- ➢ To import seaborn the syntax is:
  - Import seaborn as sns.

- ➢ Seaborn has 14 -15 datasets in it as default.They are:

1. Anscombe: Anscombe's quartet dataset.
2. attention: Response times in a psychological experiment.
3. brain_networks: Coordinates of networks in the human brain.
4. car_crashes: Insurance data about car crashes.
5. diamonds: Characteristics of diamonds.
6. dots: Lateralized response times in a psychological experiment.
7. exercise: Measurements of exercise patterns.
8. flights: Data about flights.
9. fmri: Functional magnetic resonance imaging (fMRI) data.
10. gammas: Brain activity during exposure to gamma rays.
11. iris: Iris flower data.
12. mpg: Miles per gallon (MPG) and various car attributes.
13. planets: Exoplanets data.
14. tips: Restaurant tips data.

- ➢ To load these default datasets the command is as follows:
  - **var_name=sns.load_dataset("dataset name")**

➢FUNCTIONS IN SEABORN:

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()
```

7)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()
```

8)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)

Machine learning can be done in 3 steps
1)training
2)testing
3)processing
Types of machine learning
1)supervised machine learning : labelled
2)unsupervised machine learning : unlabelled
3)semi-supervised machine learning : both labelled and unlabelled

NEURAL NETWORK: inter-connection of the neurons

CNN:
A convolutional neural network(CNN) is a type of deep learning.neural network architecture commonly used in computer vision.computer vision is a field of artificial intelligence that enables a computer to understand and interpret the image or visual data .

Types of layers:
1.input layer
2.hidden layer
convolutional layer
Activation layer
max pooling layer, average layer
dense layer
3.output layer

1.INPUT LAYER:
Its the layer in which we give input to our model.In CNN,generally,the input will be an image or a sequence of images.

2.CONVOLUTIONAL LAYER:
This is the layer,which is used to extract the feature from input dataset.it applies a set of learnable filters known as the kernels/filters to the input images.
the output of this layer is referred ad feature maps.suppose we use a total of 12 filters for this layer we'll get an output volume of dimension 32x32x12

3.ACTIVATION LAYER

By adding an activation function to the output of the preceding layer,activation layers add nonlinearity to the network.it will apply an element wise activation function to the output of the convolution.

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.

Activation function make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

TYPES OF ACTIVATION FUNCTION:

1)tanh - range: -1 to +1,nature-nonlinear,hidden layer,back-propagation.

2)sigmoid - A=1/(1+e-x) formula,range=0 to 1,it is used in output layer of a binary classification.If it is less than 0.5 then it is considered as 0 else 1.

3)relu - formula A(x)=max(0,x),range : [0,infinity],nature:- non-linear, multiple layers of neuron being activated by the Relu function.it gives fast response and calculations(computation).it is the best for error corrected fastly. It is a rectified linear,hidden layer

4)softmax - nature : non-linear,output layer,it can handle multi-class classification problems,range: 0 to 1.it is very useful to predict the probability.

4.POOLING LAYER

This layer is periodically inserted in the convnets and its main function is to reduce the size of volume which makes the computation fast, reduce memory and also prevents overfitting.

POOLING LAYERS are two types:

1.max pooling layer-16x16x12

2.average pooling layer

5.OUTPUT LAYER:

The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

DAY-5(28/02/2024)

LINEAR REGRESSION:

- It learns from labelled datasets and maps the data points to the most optimised linear functions.
- These points can be used for prediction on new datasets.
- We have two variables,they are dependent and independent.

The formulae is y=a+bx.To find

1. We have to calculate the mean for both the dependent and independent.
2. Find the differences between x point and x'(mean x).
3. Find the differences between y point and y'(mean y).
4. Find the total sum of squares of (x-x').
5. Find the total sum of products of (x-x') and (y-y').

- b=$\dfrac{\text{sum of product of (x-x') and (y-y').}}{\text{total sum of squares of (x-x')}}$
- Now with the help of **b** we can find **a** by replacing **x** with **x'** and **y** with **y'** in **y=ax+b.**
- The main two functions in this linear regression are fit() and predict().

LOGISTIC REGRESSION:

- It is used for binary classification and is learned from supervised algorithms.
- But the only difference is this will actually tells us which class does our prediction site in.

SIGMOID FUNCTION

- When the model is confident then it shows a narrow decision boundary
- When the model is not confident the it shows a wide decision boundary.

DAY 6:29-3-2024

<u>**DECISION TREE:**</u>

- Decision trees in machine learning provide an effective method for making decisions.
- This is because they lay out the problem and all the possible outcomes.
- They have **nodes** and **leafs.**
- **Nodes:** They have both the true and false branches.
- **leaf:** It has either true or false in other words it shows us the result and where our outcome sits.
- Decision trees recurrently(continuously) split the data until it gets pure nodes(leaf).
-

DISADVANTAGES:

- It has no accuracy.
- It won't get to a conclusion until it recurrently splits the data which wastes time.

<u>**RANDOM FOREST:**</u>

- Collection of many decision trees is called Random Forest.
- We can frequently find **bootstrapping** and **aggregation.**
- Random forest creates a new dataset from the existing or the original dataset.
- This process is called **"bootstrapping".**
- This gives the more accurate answers when compared to the decision tree.

<u>CONDITIONS:</u>

- Select rows that are different and cover all the rows.
- Select features that are different and cover all features.
- Selecting the same data(rows) and same features have no use.

<u>**DAY 7:1-3-2024**</u>

BIG DATA CONCEPTS:

- ➔ Big data refers to vast and complex datasets that cannot be effectively managed or processed using traditional data processing tools

- ➔ The concept of big data is often characterised by four key dimensions known as 4 V's:**volume,velocity,variety,veracity**.

❖ VOLUME:

- ➢ It is the immense amount of data that is generated by sources like social media,online transactions.
- ➢ sd

❖ VELOCITY:

- ➢ Velocity denotes the speed at which data is generated,collected and processed.
- ➢ In today's world, data is produced rapidly in real time from sources like social media updates,sensor data from iot devices,online transactions and more.
- ➢ Efficient processing of this data is crucial for real time analytics and decision making.

❖ VARIETY:

- ➢ Variety refers to the diversity of data types and sources.
- ➢ Data comes in various forms including structured data,semi structured data,and unstructured data.
- ➢ Big data platforms must be able to handle this diverse range of data types effectively.

❖ VERACITY:

- ➢ It emphasises quality and reliability of the data.
- ➢ Due to the sheer(bulk) volume and variety of data sources,ensuring data quality and reliability can be challenging.
- ➢ Big data analysis often involves dealing with incomplete,inconsistent,or inaccurate data requiring robust mechanisms for data cleaning,validation,and quality assurance.

SCIPY(STATISTICAL ANALYSIS AND STAT'S MODEL):

- 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.

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

HYPOTHESIS TESTING:

t_statistic :

- ➔ The t_static is a measure of how many standard deviations the sample mean is away from the hypothesised population mean relative to that variability in the sample.
- ➔ It is calculated as the difference between the sample mean and the population mean divided by the standard error of the sample mean.
- ➔ The larger the absolute value of the t_static the stronger the evidence against the null hypothesis.

P_value:

- ➔ The value is the probability of observing a text statistic or one or more (extreme) if the null hypothesis is true.
- ➔ It quantifies the strength of the evidence against the null hypothesis.
- ➔ A small p value (typically less than 0.05) integrates that the observed data is unlikely under the assumption that the null hypothesis is true leading to the rejection of the null hypothesis in favour of the alternative hypothesis.
- ➔ Conversely a large p_value suggests that the observed data is under the null hypothesis leading to a failure to reject the null hypothesis.

HYPOTHESIS TESTING:

- ➔ If the p_value isb less than the predetermined significance value(suggest 0.05) it is typically interpreted as sufficient evidence to reject the null hypothesis.
- ➔ If the p_value is greater than the significance 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 of hypothesis about the population

<u>DAY 9-04/03/2024</u>

<u>HADOOP:</u>

- ➔ <u>It is an open source framework designed to process and store big data</u> in a distributed environment
- ➔ Programming model is a simple one called as map reduce and distributed file system.it is called HDFS(hadoop distributed file system)

HDFS:

- System that stores data across multiple machines.it provides high end access to application data

MAP REDUCE:

- ➔ A programming model for processing and generating large datasets.
- ➔ It involves two phases
  - ◆ **Map phase**-where data is divided into smaller chunks and processed in parallel
  - ◆ **reduce phase**-where the result from the map phase are aggregated to produce the final output

YARN(yet another resource negotiator)

- ➔ It is a resource management layer that manages resources in hadoop cluster 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.there are 5 steps

Step1-input data:

- we have a large collection of documents stored in htms

Step2-map phase:

- In this phase each document is divided into words each word is emitted as key value pair (word,1)word is the key 1 is the value

Step3-shuffle and sort phase:

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

Step4-reduce phase

- 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 occurrence of that word.
- The output of the reduce phase is the final word count

` **Step5-output data**

- The final output containing the count of each word is stored in hdfs or any other desired location

Conclusion:

Hadoop provides a scalable and cost effective solution for processing and storing big data by distributing data and computation across multiple nodes it enables the processing of large dataset

Ex:word count principle hadoop map reduce framework.