PART1:NUMPY

Set of methods and functions for efficient array creation, manipulation, and computation provided by Numpy.

Common Array Creation Functions:

- *np.array()*: Creates an array from a Python list or tuple.
- *np.zeros()*: Creates an array filled with zeros.
- *np.ones()*: Creates an array filled with ones.
- *np.eye()*: Creates an array filled with ones on a specified diagonal and zeros elsewhere.It is used for identity array.
- *np.empty()*: Creates an array with uninitialized (random) content.
- *np.arange()*: Creates an array with evenly spaced values within a given interval.
- *np.linspace()*: Creates an array with a specified number of evenly spaced values over a given interval.
- *np.random.rand()*: Creates an array of random floats between 0 and 1.
- *np.random.randint()*: Creates an array of random integers within a specified range.

Array Attributes:

- *arr.shape*: Returns a tuple representing the dimensions of the array.
- *arr.ndim*: Returns the number of dimensions of the array.
- *arr.size*: Returns the total number of elements in the array.
- *arr.dtype*: Returns the data type of the elements in the array

Iterating Numpy Array:

- *nditer()*
- *Ndenumerate()*

Common Array Manipulation Methods and Functions:

- *arr.reshape()*: Gives a new shape to an array without changing its data.
- *arr.flatten()*: Returns a copy of the array flattened to one dimension.
- *arr.ravel()*: Returns a view of the array flattened to one dimension (modifying the view affects the original array).
- *np.concatenate()*: Joins a sequence of arrays along an existing axis.
- *np.split()*: Splits an array into multiple sub-arrays.
- *arr.transpose()* or *arr.T*: Transposes the array.
- *np.sort()*: Returns a sorted copy of an array.
- *np.where()*: Returns elements chosen from *x* or *y* depending on *condition*.

Common Universal Functions (ufuncs) for Element-wise Operations:

- **Mathematical:** *np.add()*, *np.subtract()*, *np.multiply()*, *np.divide()*, *np.power()*, *np.sqrt()*, *np.exp()*, *np.log()*, *np.sin()*, *np.cos()*, *np.abs()*.
- **Statistical:** *np.mean()*, *np.median()*, *np.std()*, *np.var()*, *np.min()*, *np.max()*, *np.sum()*.

Input/Output Functions:

- *np.save()*: Saves an array to a binary file in NumPy *.npy* format.
- *np.load()*: Loads an array from a *.npy* file.
- *np.loadtxt()*: Loads data from a text file.
- *np.savetxt()*: Saves an array to a text file.

PART2:PANDAS LIBRARY:

Pandas (stands for Python Data Analysis) is an open-source software library designed for **data manipulation** and **analysis**.

Why we use Pandas:

- Reading and writing data from various file formats like CSV, Excel and SQL databases.
- Cleaning and preparing data (handling missing values, filtering, removing duplicates).
- Merging, joining, and reshaping datasets.
- Performing statistical analysis and descriptive statistics.
- Visualizing data quickly.

Pandas DataFrame:

A DataFrame is a two-dimensional, size-mutable and potentially heterogeneous tabular data structure with labeled axes (rows and columns).

A DataFrame is similar to a table with rows and columns. It helps in handling large amounts of data, performing calculations, filtering information with ease.

SERIES:

Series is like a column of table(It is 1D Array)

- Series will be generated by Numpy Array,List,Tuple and Dictionary
- Attributes:(Size,dtype,name,is_unique,index,value)

- Methods:(head(),tail(),sample(),value_counts(),sort_values(),sort_index(),Concept of method chaining.
- Math Function:Count,Product,Sum,mean,mode,std,var,min,max,describe
- Indexing:Integer Indexing i.e. positive & fancy indexing but negative indexing not work for integer only for string ,Slicing
- Editing Series:If indexing is not available and assign the values it will create new entry.We can edit through slicing as well as fancy indexing or label
- Null variation:isnull(),hasnan(),notnull() all work in series as well as dataframe
- Unique and nunique.
- Series with python functionality:

✓ (len/type/dir/sorted/max/min)
✓ Type conversion
✓ Membership operator
✓ Looping
✓ Arithmetic operator
✓ Relational operator
✓ Plotting graph:

DATA FRAME:

✓ **Creating DataFrame**
Using List,Dict and read_csv
✓ **Attributes and Methods**
Shape,dtype,columns,index,values,head,tail,sample,info,describe,isnull,isnull.sum(),duplicated,dupilicated().sum(),rename,inplace
✓ **Maths Functions**
Sum,min,max,mean,mode,var
✓ **Selecting cols from dataframe**
✓ **Selecting rows from dataframe**
✓
iloc:search using index position,set index(colname)
loc:search using index labels

✓ **Some Impotant Functions**

1. **astype**
2. **between**
3. **drop_duplicates**
4. **dropna**
5. **fillna**
6. **Drop(columns=['name'])**

7. **Apply**
8. **Isin**
9. **Duplicated**
10. **Set_index**
11. **Reset_inex**
12. **iloc() & loc()**
13. **Group_by():use in loop,get_group(),apply() by using use define function,min(),max(),sum(),describe(),value_counts(),len(),size(), agg():passing value in dict**
14. **Concat():concat(axis=1),(ignore_index=True),(join="inner"),multi indexing:(keys=(['g1','g2']))**
15. **Merge():how=left,outer,inner,right,cross,on=columnname**
16. **Join():it works on uneven columns dataframe but ignore rest values how=left,outer,inner,right index="lsuffix/rsuffix",index=["a","b"]**

PART3:MATPLOTLIB

2D Line Plot:

It is Bivariant Analysis,Apply on only 2 Columns.
Use case:Time Series Analysis
Categorical->Numerical
Numerical->Numerical
It is faster tnan scatter

1.using different attributes:color,linestyle,linewidth,marker,markersize,label
2. using title
3. using xlabel & ylabel
4. using legend() attribute='best' default
5. using grid()
6. using plt.show()#to plot graph in website not imp for console

Scatter Plot:

It is Bivariant Analysis,Apply on only 2 Columns.
Use case:Find Correlation
Numerical->Numerical
It is slower than 2d plot

Bar Chart:

It is Bivariant Analysis,Apply on only 2 Columns.
Use case:Aggregate analysis of group
Numerical->Categorical

Stacked bar:Stacked bar plots represent different groups on the highest of 1 another. The peak of the bar depends on the resulting height of the mixture of the results of the groups. It goes from rock bottom to the worth rather than going from zero to value.

Histogram:

It is Univariant analysis.
Only for Numerical column:It divides numerical into categorical range(bins)
Use case:Frequency count

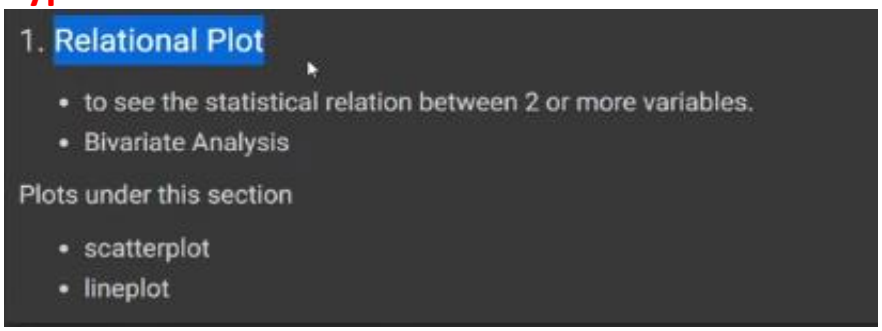
Pie Chart:

It is Univariant analysis.
Categorical-Numerical
Use case:To find contribution on standard scale

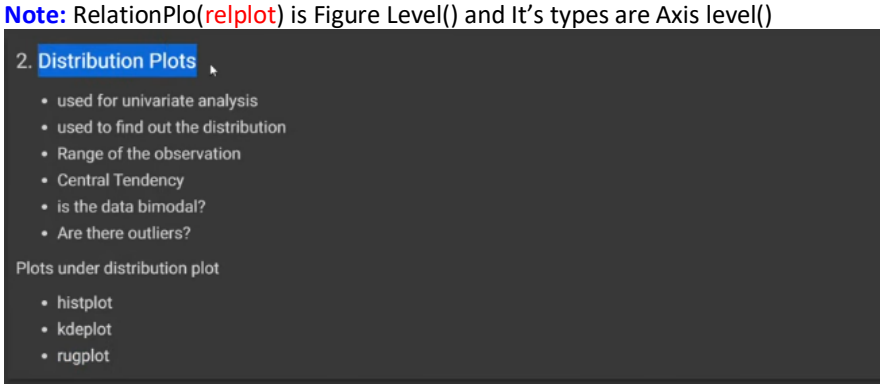
PART4:SEABORN

Seaborn is a Python library built on top of Matplotlib that focuses on statistical data visualization. It provides high-level functions, built-in themes, and automatic handling of datasets, allowing users to create informative and visually appealing plots with minimal code. Seaborn is widely used for exploring trends, relationships and distributions in data.

Types of Seaborn Plot:



Note: RelationPlo(relplot) is Figure Level() and It's types are Axis level()



Remark:bimodal :How many multiple peaks.

Note: Distribution Plot(displot) is Figure Level() and It's types are Axis level()

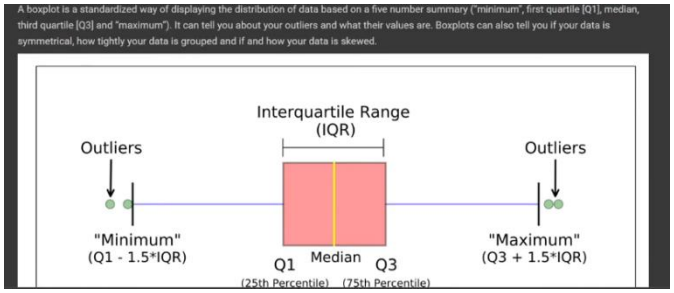
3. Matrix Plot:

- Heat Map
- Cluster Map

4. Categorical Plot: Only Numerical->Categorical

- **Categorical Scatter Plot:(Bivariant)**
  - ✓ Strip Plot
  - ✓ Swarm Plot

- **Categorical Distribution plot:(Univariant)**
  - ✓ Box Plot



Note:Seaborn gives u comparision between numerical and categorical data

- ✓ Violin Plot(Boxplot+KDEplot)

- **Categorical Estimated Plot:(For Central Tendency)**
- ✓ Bar Plot
- ✓ Count Plot
