**Fundamentals of Data Science**

**Syntax**

**Basics: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| Install | ! pip install library\_name |
| Import | Import library\_name as short\_name |
| Import from | from library\_name import module\_name |

**Numpy as np**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| Arange (range in Numpy) | np.arange(start , end + 1, step) |
| Array | np.array(data) |
| Zero’s value array | np.zeros((row,column,No.of arrays)) |
| One’s value array | np.ones(row,column,dtype=datatype)) |
| Required no. value | np.full([row.column],required\_no) |
| Reshape | Data.reshape(row,column) |
| Dimension check | Data.ndim |
| Random float values | np.random.rand(row,column) |
| Random int values | np.random.randint(end,size=(row,column)) |
| Random values with limit | np.linspace(2,6,num=10) 🡪 10 values between 2-6 |

**Operations: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| Add | np.add(a,b) or a+b |
| Subtract | np.substract(a,b) or a-b |
| Multiply | np.multiply(a,b) or a\*b |
| Divide | np.divide(a,b) or a/b |

**Statistical: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| Minimum | np.min(a) |
| Maximum | np.max(a) |
| Mean | np.mean(a) |
| Median | np.median(a) |
| Mode | np.mode(a) 🡪 mode is not available in numpy directly  import from scipy |

**Array Comparison: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| Equal | np.equal(a,b) |
| Array Equal | np.array\_equal(a,b) |

**Indexing: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| Row and Column Indexing | a[start\_row : end\_row + 1 , start\_col : end\_col +1] |
| Value Indexing | a[0] a[7] |

**Trigonometric: - scipy as sc**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| degrees (sin,cos,tan) | special.sindg(degree\_value)  special.cosdg(90)  special.tandg(45) |
| exponential 10\*\*5 | special.exp10(5) |

**Pandas as pd**

**1.Series: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| Series | pd.Series(data) |
| User defined Indexing | pd.Series(data,index= [‘column\_name1’ ,  ‘column\_name2’],dtype=int) |

**2.Data Frame: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| create DataFrame | pd.DataFrame(data) |
| add column | data[‘col\_name’]=pd.DataFrame(l) |
| drop column | data.drop([‘col\_name’),axis=1) |
| Rename column | data.rename(columns={‘old\_col\_name’ : ‘new\_col\_name’}) |
| Dictionary keys | data.keys() |
| Dictionary values | data.values() |
| Import Dataset (read) | pd.read\_csv(‘dataSet\_name.csv’) |
| delete column | del data[‘column\_name’) |
| pop | data.pop(‘col\_name’) |
| drop | data.drop([‘col\_name’],axis=1) |
| indexing call only one column | data[‘column\_name’] |
| indexing | pd.DataFrame(data,index=[0,1,4,7]) |
| call multiple column | data[‘col\_name’],data[‘cal\_name’] |
| loc | data.loc[start\_row : end\_row] |
| iloc | data.iloc[start\_row : end\_row + 1] |

**EDA: - Exploratory Data Analysis**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| top 5 | data.head() or data.head(end\_index\_value) |
| bottom 5 | data.tail() |
| shape (no. of rows,col) | data.shape |
| Name of columns | data.columns |
| duplicate value count | data[‘col\_name’].value\_counts() |
| describe (quantile) | data.describe() |
| columns data type | data.dtypes |
| range index | data.index |
| Information of data | data.info() |
| Display full dataset; Full Table | pd.set\_option('display.max\_columns',None)  pd.set\_option('display.max\_rows',None) |

**Null values: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| check null values in total table | data.isnull 🡪true or false |
| check null values in total table | data.isnull.sum() 🡪count null values of each column  drop.isna().sum().sum() 🡪 Total |
| drop null values row wise | data.dropna(axis=0) |
| drop null values column wise | data.drop(axis=1) |
| check null values in one column | data.column\_name.isna().sum() |
| replace null values | data[‘col\_name’].fillna(‘value\_name’,  inplace=True) |
| Mean (numerical values) | data[‘col\_name’].mean() |
| Median(numerical values) | data[‘col\_name’].median() |
| mode(string/object values) | data[‘col\_name’].mode()[0] |
| Replace | data[‘col\_name’].replace(to\_replace=np.nan  ,values=’New\_value’,inplace=True) |
| sorting column | data.sort\_values(‘col\_name’,axis=0,  ascending=True)🡪 True=asc | False=desc |
| sort multiple columns | data.sort\_values(‘col\_name1’,col\_name2’,  axis=0,ascending=True,False)🡪c1=true c2=false |
| percentage | data.isna().sum()\*100/len(data) or  data.isnull().sum()/data.shape[0]\*100 |

**Grouping: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| group by column | data.groupby(‘col\_name’).sum() 🡪shows count of duplicate values |
| group by 2 calumns | data.groupby([‘col\_name1’,’col\_name2’])  .sum() |
| get first row | data.first() |
| groupby and sort | data.groupby9[‘col\_name’],sort=False.sum() |
| get values from group | data.get\_group(‘required\_value\_from\_col1’,  value\_2\_from\_col2) |

**Concatenation: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| join two data sets | pd.concat([data1,data2])🡪 same dataset or diff but should contains one common column |
| ignore Index | pd.concat([data1,data2],ignore\_index=True) |
| for Dictionary | pd.concat([data1,data2],keys=[‘key\_name1’,  ‘key\_name2’]) |

**Merge: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| merge two datasets | pd.merge(data1,data2,on=’commom\_column’) |
| merge with two columns | pd.merge(data1,data2,on=’commom\_column1’,  ‘common\_column2) |
| merge type Inner | pd.merge(data1,data2,on=’commom\_column’  , how=’inner’) |
| merger type outer | pd.merge(data1,data2,on=’commom\_column’  , how=’outer’) |

**Join: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| join two datasets | data.join(data1,lsuffix=’\_a’,rsuffix=’\_b’,  on=’common\_column\_name’) 🡪lsuf=data1 … |
|  |  |

**Data Visualization**

**Matplotlib and seaborn**

**data=a**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| matplotlib | import matplotlib.pyplot as plt |
| seaborn | import seaburn as sns |
| syntax of matplotlib | plt.plot(data,x=’col’,y=’col2’) |
| syntax of seaborn | sns.barplot(data=a,x=’col1’,y=’col2’) or  sns.barplot(x=a[‘col1’],y=[‘col2’]) |
| Heatmap | plt.figure(figsize=((5,5))  sns.heatmap(data) |

**Line plot: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| line plot in matplotlib | plt.plot(data,x,y) 🡪assign values in x & y variable  plt.show() |
| title to plot | plt.title(‘title\_name’) |
| x,y lables | plt.xlabel(‘x-lable\_name’)  plt.ylabel(‘y-lable\_name’) |
| line plot in seaburn | sns.lineplot(data=a,x=’col1’,y=’col2’) |
| matplotlib in detail | plt.plot(x,y,color=’crimson’,linestyle=’:’  ,marker=’o’,linewidth=3,alpha=1) |
| Two-line plots in one plot | just write 2 plot with details one after  one like In detail then plt.show() |

**Subplot: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| plot with details | plt.plot(x,y,color=’r’,linestyle=’:’,  marker=’o’,linewidth=5,alpha=1) |
| 2 different plots  (subplots=fig size)  (subplot=diff plot) | fig=plt.subplots(rows,columns,  figsize=(weidth,height\_of\_total\_figure\_in Inches))  1.🡪plt.subplot(weidth,height,location)  2.🡪plt.subplot(weidth,height,location) |

**Bar plot: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| matplotlib | plt.bar(x,y) |
| seaborn | sns.barplot(x,y) |
| Ignore Warnings | import warnigns  warnings.filterwarnings(‘ignore’) |
| barplot in matplotlib detail | plt.bar(x,y,width=0.7,hatch=’\_’,ec=’r’,  fill=False) |
| seaborn | sns.barplot(x,y,palette=’col\_name1’,  hue=’col\_name2’) |

**Scatter plot: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| matplotlib | plt.scatter(x,y) |
| change background color | sns.set\_style(‘darkgrid’) |
| in detail | plt.scatter(x,y,color=’col1’,marker=’^’,  edgecolor=’olive’,linewidth=0.5) |
| seaborn | sns.scatterplot(data=df,x=’col1’,y=’col2’  ,hue=’col3’,style=’col4’) |
| figure size | fig=plt.subplots(figsize=(10,5)) |
| Heat map sns | sns.heatmap(data) |
| Heat map matplotlib | fig,ax=plt.subplots() here data or col\_nm  img=ax.imshow(data) |
| show values on | sns.heatmap(data,annot=True) |
| correlation | data.corr() |

**Box plot: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| matplotlib | plt.boxplot(data[‘column\_name’]) |
| details | plt.boxplot(col1,showmeans=True)  plt.xlable(‘x-axis’)  plt.ylable(‘y-axis’) |
| seaborn | sns.boxplot(data[‘column\_name’,palette=  ‘Reds\_r’) or  sns.boxplot(data=a,y=’column\_name’ |

**Outliers: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| quantile 1 | q1=data[‘column\_name’].quantile(0.25) |
| quantile 3 | q3=data[‘column\_name’].quantile(0.75) |
| iqr | iqr=q3-q1 |
| Upper limit | upper=q3+1.5\*iqr |
| Lover limit | lower=q1-1.5\*iqr |
| Ignore outlier (data=a) | a[(~((a[‘col1’]<lower) | (a[‘col1’]>upper)))] 🡪 ~ used to reverse the code |

**Histogram: -**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| matplotlib | plt.hist(a[‘column\_name’]) |
| seaborn | sns.histplot(data=a,x=values) |
| distribution line | sns.histplot(x,kde=True) |

**Seaborn: - sns**

|  |  |
| --- | --- |
| **Function** | **Syntax** |
| seaborn full detailed with all mentioned | seaborn.lineplot(data=None, \*, x=None, y=None, hue=None, size=None, style=None, units=None, palette=None, hue\_order=None, hue\_norm=None, sizes=None, size\_order=None, size\_norm=None, dashes=True, markers=None, style\_order=None, estimator='mean', errorbar=('ci', 95), n\_boot=1000, seed=None, orient='x', sort=True, err\_style='band', err\_kws=None, legend='auto', ci='deprecated', ax=None, \*\*kwargs) |