Day 3 : Data Visualization & Outlier detection

★ np.random.random()

↓

random no. bet^n 0 to 1

★ np.random.randint(0,10,20)

↗

20 vals bet^n 0 & 10

To change its dim^n :

add → .reshape(4,5)

OR

np.random.randint (0,10, size=(4,5))

To make 3-d array

np.random.randint(0,10,size=(2,3,5))

data = np.random.randn(1000)

★ data.mean() ,data.std()

★ a.sum (axis= 0, keepdims = True)

↗

can be 0,1,2...

→ keepdims → Keep dimensions

If it is false, array will be 1-dim.

★ np.linspace (0, 10, 11)

start ↗ end ↑ ↖ no of points

★ sin wave

```
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x,y)
```

★
```
r = 10
theta = np.linspace(0, 2* np.pi, 200)
x = r * np.cos(theta)
y = r * np.sin(theta)
plt.plot(x,y)
```

To fix up scale:

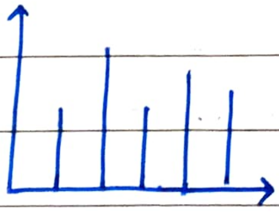
```
plt.figure(figsize=(4,4))
```

★ If we write
plt. scatter( ) → marks it as points & not line.

★ noise = np. random .randn (1000)
~~y=~~ y-mod = y + noise
It'll look like a line, but they are all points.

★ x = np.random.randn (10000) ← creates a normal distrubution
o = plt.hist (x , bins = 120)

With randint → you get



★ plt.pie (10,20,30)

* SeaBorn

umport seaborn as sns

x = np.random.randn (1000)

sns.distplot (x, bins=30 , kde = False, rug=True)

eg:

data = sns.load_dataset("iris") # iris dataset

sns ~~data~~.displot.(x, bins=30, kde = False, rug=

kde → approximation

sns.scallerplot (data ["total_bill"], data ["tip"],
hue=~~data~~=data ["smokers"])

sns.violinplot(x = data.total_bill)

Loading datasets

from sklearn.datasets import load_boston

x, y = load_boston (return X_y = True)

sns.violin plot (~~x~~ = y)

Outliers are used in fraud detection