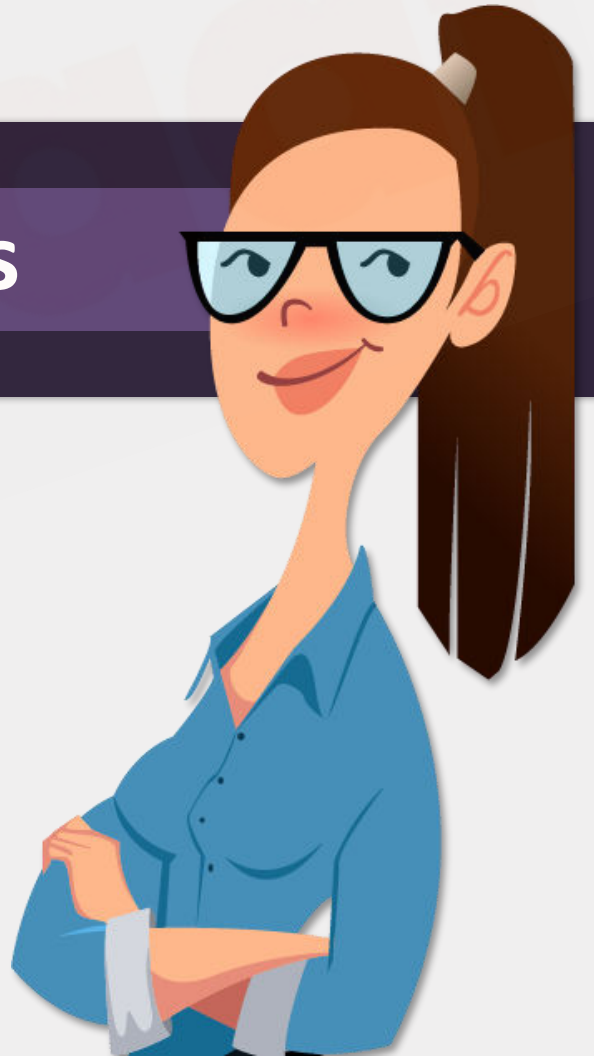
IntelliPaat

SciPy Training for Beginners



IntelliPaat

What is SciPy?

- Free and open-source Python library
- Used for scientific computing and technical computing
- Contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing

SciPy

What are the characteristics of SciPy?

IntelliPaat

SciPy

Characteristics of SciPy

01 Contains toolboxes dedicated to common issues in Scientific Computing

02 Different submodules corresponds to different applications

03 Operates on NumPy array, NumPy and SciPy work hand in hand

04 Adds significant power to the interactive Python session

05 Routines are tested and optimized

SciPy Sub-packages

IntelliPaat

SciPy

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

IntelliPaat

Working with SciPy Sub-packages

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.cluster()?

- Provides kmeans() function
- Useful while dividing data into clusters of related attributes
- Divide n observations into k clusters

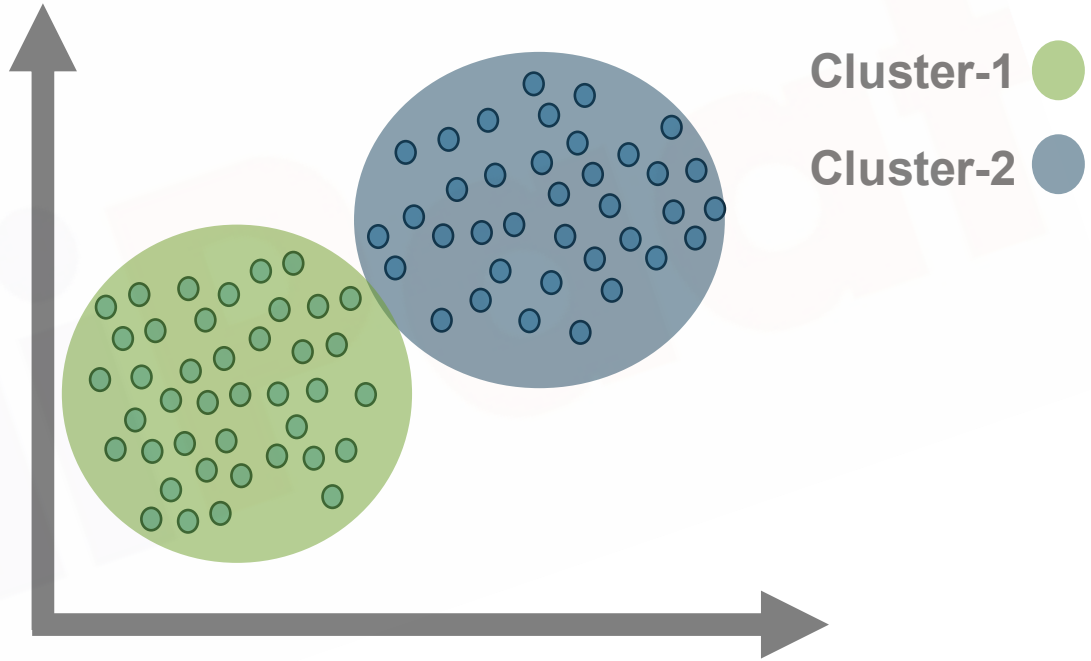
IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

What is clustering?



Dividing data into groups

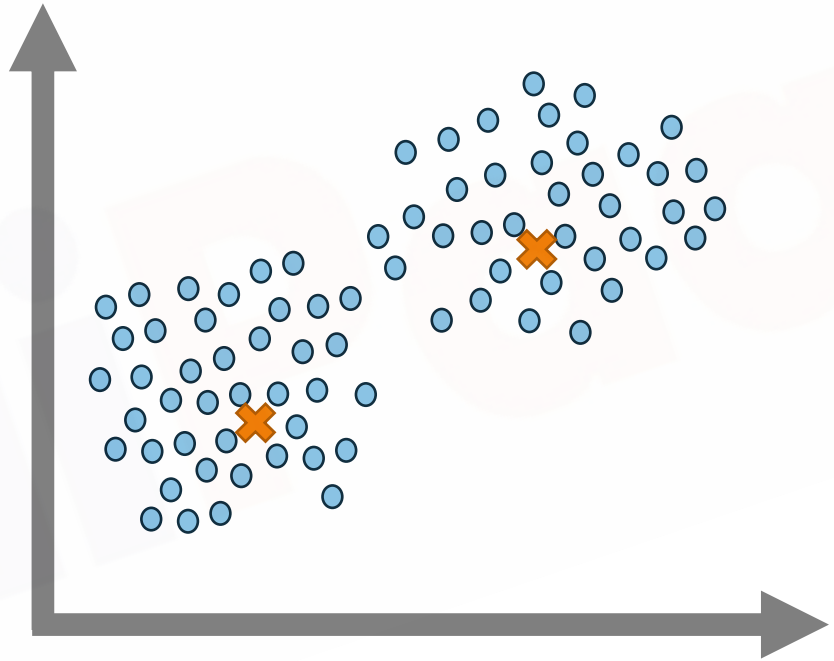
IntelliPaat

SciPy

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

How do we find clusters?



Step:1 Randomly select two cluster centers

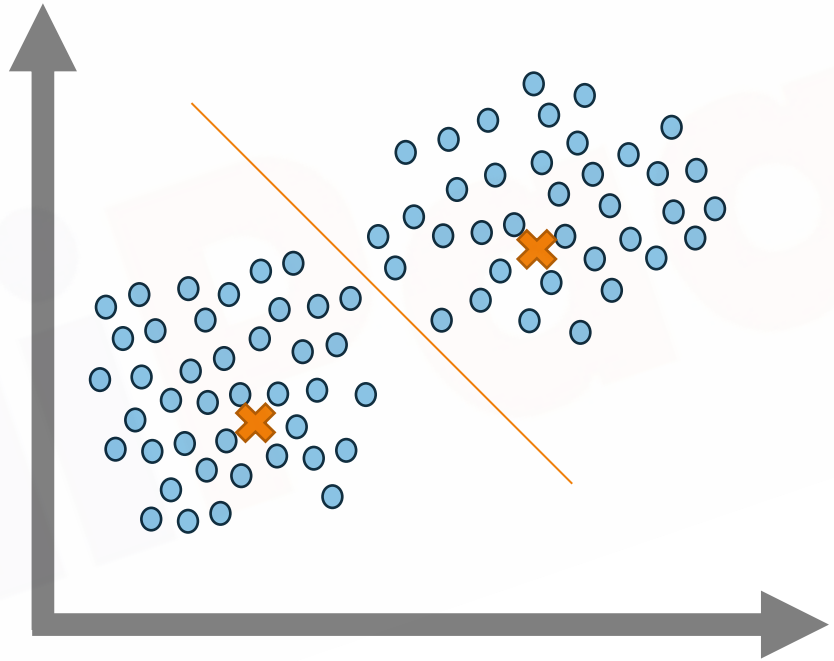
IntelliPaat

SciPy

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

How do we find clusters?



Step:2 Assign members to the cluster

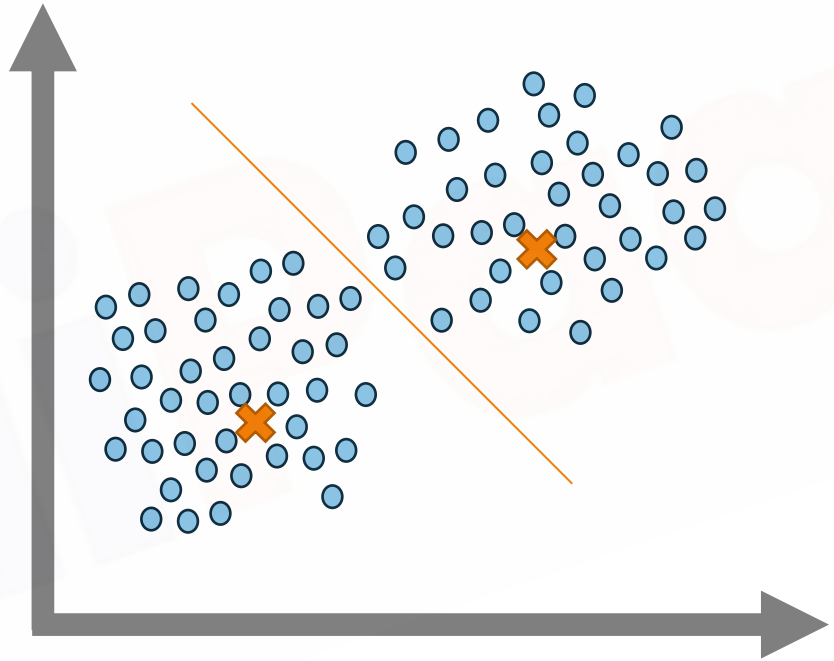
IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we find clusters?



Step:3 Calculate the mean distance of each member

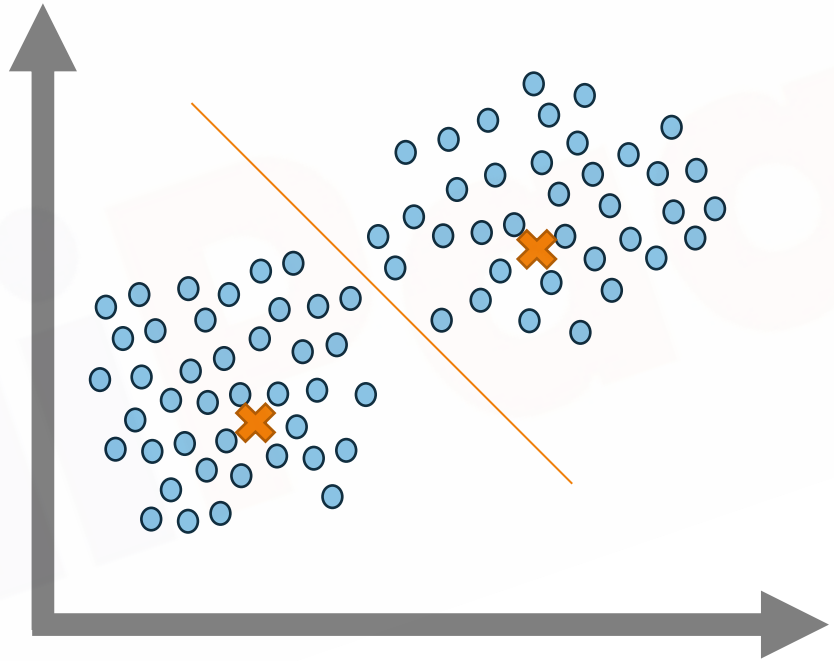
IntelliPaat

SciPy

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

How do we find clusters?



Step:4 Shift the cluster center towards mean

Repeat **Step:2** and **Step:3** until no member changes the group

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we find cluster?

Example:

	X	Y	Dist-k1	Dist-k2	Min	Cluster
	1	1	8	33.62	8	1
	2	2	2	19.22	2	1
	4	4	2	2.42	2	1
	5	5	8	0.02	0.02	2
	6	6	18	1.62	1.62	2
K1(Centroid1)	3	3				
K2(Centroid2)	5.1	5.1				

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we find cluster?

Example:

X	Y	Cluster-1		Cluster-2	
1	1	1	1		
2	2	2	2		
4	4	4	4		
5	5			5	5
6	6			6	6
	Mean	2.333	2.333	5.5	5.5

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we find cluster?

Example:

X	Y	Cluster-1		Cluster-2	
1	1	1	1		
2	2	2	2		
4	4	4	4		
5	5			5	5
6	6			6	6
	Mean	2.333	2.333	5.5	5.5

- Now, consider (2.33, 2.33) and (5.5, 5.5) as K1 and K2
- Again, find distance of each point from K1 and K2
- Based on minimum distance put them in clusters
- Repeat until no member changes the cluster

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we find cluster using SciPy?

```
In [24]: import pandas as pd
         #importing data
         data=pd.read_excel('somecars1.xlsx')
         data
```

Out[24]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
5	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
6	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
7	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
8	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
9	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
10	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
11	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
12	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
13	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we find cluster using SciPy?

```
In [25]: import pandas as pd
         from scipy.cluster.vq import kmeans,vq
         #data importing
         data=pd.read_excel('somecars1.xlsx')
         #Find out centroid with the help of kmeans function, here k=3
         centroid,_ = kmeans(data,3)
         #Find out cluster index for each record with vector  quintization function(i.e. vq(data,centroid))
         idx, _ = vq(data,centroid)
         #print clusetr index
         idx
```

```
Out[25]: array([0, 0, 0, 2, 1, 2, 1, 0, 0, 0, 0, 2, 2, 2, 1, 1, 1, 0, 0, 0, 0, 2,
                2, 1, 1, 0, 0, 0, 1, 0, 1, 0])
```

k-means without data whitening

```
In [32]: print(centroid)
```

```
[[2.45000000e+01 4.62500000e+00 1.22293750e+02 9.68750000e+01
  4.00250000e+00 2.51800000e+00 1.85431250e+01 7.50000000e-01
  6.87500000e-01 4.12500000e+00 2.43750000e+00]
 [1.70142857e+01 7.42857143e+00 2.76057143e+02 1.50714286e+02
  2.99428571e+00 3.60142857e+00 1.81185714e+01 2.85714286e-01
  0.00000000e+00 3.00000000e+00 2.14285714e+00]
 [1.46444444e+01 8.00000000e+00 3.88222222e+02 2.32111111e+02
  3.34333333e+00 4.16155556e+00 1.64044444e+01 0.00000000e+00
  2.22222222e-01 3.44444444e+00 4.00000000e+00]]
```

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we find cluster using SciPy?

```
In [27]: import pandas as pd
         from scipy.cluster.vq import kmeans,vq,whiten
         #data importing
         data=pd.read_excel('somecars1.xlsx')
         #whitening data
         data = whiten(data)
         #Find out centroid with the help of kmeans function, here k=3
         centroid,_ = kmeans(data,3)
         #Find out cluster index for each record with vector  quintization function(i.e. vq(data,centroid),
         idx, _ = vq(data,centroid)
         #print clusetr index
         idx
```

```
Out[27]: array([0, 0, 1, 2, 2, 2, 2, 1, 1, 0, 0, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2,
                2, 2, 2, 1, 1, 1, 0, 0, 0, 1])
```

k-means with data whitening

```
In [32]: print(centroid)
```

```
[[ 2.65026134  4.38861477  2.79561906  2.69379731  5.87571578  4.14784379
   9.96498874  0.28797293  0.          4.13118224  1.75228367]
 [ 4.49485899  2.27557803  0.86186686  1.22454955  7.73557558  2.373431
  10.88086523  1.83255502  1.48080818  5.63343032  0.9721294 ]
 [ 3.11866282  3.73844962  1.70065265  2.62499698  7.33480435  3.24639015
   9.36352368  0.57594586  1.45436517  6.09841187  3.05526383]]
```

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

Why do we use scipy.stats()?

- All of the statistical functions are located in the sub-package scipy.stats.
- Contains a large number of probability distributions
- Use info(stats) to see.all the functions

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

Example:

Nutritionists measured the sugar content (in grams) for 13 drinks at Jake's Java coffee shop. The collected data looks like this

Sugar
15
18
20
26
32
38
32
24
21
16
12
11
14

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

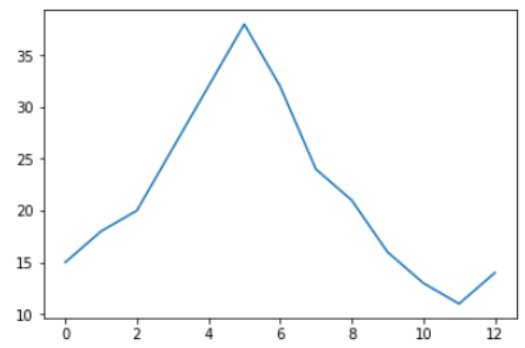
Example:

The drinks had a mean of 21.5g and a standard deviation of 8g, and the distribution was roughly symmetric.

Sugar
15
18
20
26
32
38
32
24
21
16
12
11
14

```
In [52]: #import numpy
         import numpy as np
         #create the marks array
         coffee = np.array([15,18,20,26,32,38,32,24,21,16,13,11,14])
         print(coffee.mean(), coffee.std())
         #let us see the data distribution by plotting it
         import matplotlib.pyplot as pyplot
         plt.plot(range(13),coffee)
```

```
21.53846153846154 8.063358573156098
Out[52]: [<matplotlib.lines.Line2D at 0x295efe4a668>]
```



IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

Example:

Now, a Grande Mocha Cappuccino at Jake's Java contains 14g of sugar. Calculate the standardized score (z-score) for the Grande Mocha Cappuccino.

Sugar
15
18
20
26
32
38
32
24
21
16
12
11
14

```
In [47]: #import numpy
         import numpy as np
         #create the marks array
         coffee = np.array([15,18,20,26,32,38,32,24,21,16,13,11,14])
         from scipy import stats
         #find the zscore
         print(stats.zscore(coffee))
```

```
[-0.81088562 -0.43883222 -0.19079662  0.55331019  1.297417    2.0415238
  1.297417    0.30527459 -0.06677882 -0.68686782 -1.05892123 -1.30695683
 -0.93490342]
```

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

Example:

Now, a Grande Mocha Cappuccino at Jake's Java contains 14g of sugar. Calculate the standardized score (z-score) for the Grande Mocha Cappuccino.

Sugar
15
18
20
26
32
38
32
24
21
16
12
11
14

```
In [47]: #import numpy
         import numpy as np
         #create the marks array
         coffee = np.array([15,18,20,26,32,38,32,24,21,16,13,11,14])
         from scipy import stats
         #find the zscore
         print(stats.zscore(coffee))
```

```
[-0.81088562 -0.43883222 -0.19079662  0.55331019  1.297417    2.0415238
  1.297417    0.30527459 -0.06677882 -0.68686782 -1.05892123 -1.30695683
 -0.93490342]
```

z-score of the Mocha Cappuccino is **-0.9**

What does this mean?

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

Example:

Now, a Grande Mocha Cappuccino at Jake's Java contains 14g of sugar. Calculate the standardized score (z-score) for the Grande Mocha Cappuccino.

Sugar
15
18
20
26
32
38
32
24
21
16
12
11
14

What does this mean?

A. This drink has 0.9g of sugar less than the mean of the 13 drinks.

B. This drink is 0.9 standard deviations below the mean of the 13 drinks.

C. About 90% of drinks have a lower sugar content than this drink.

D. About 90% of drinks have a higher sugar content than this drink.

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

Example:

Now, a Grande Mocha Cappuccino at Jake's Java contains 14g of sugar. Calculate the standardized score (z-score) for the Grande Mocha Cappuccino.

Sugar
15
18
20
26
32
38
32
24
21
16
12
11
14

What does this mean?

A. This drink has 0.9g of sugar less than the mean of the 13 drinks.

B. This drink is 0.9 standard deviations below the mean of the 13 drinks.

C. About 90% of drinks have a lower sugar content than this drink.

D. About 90% of drinks have a higher sugar content than this drink.

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

Example:

Dataset consists of frequency of people going to gym and frequency of smoking

Gym going / Smoking	Always	Never	Sometimes
Heavy	7	1	3
Never	87	18	84
Occasional	12	3	4
Regular	9	1	7

Test if the relation is significant or not..

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

Example:

Dataset consists of frequency of people going to gym and frequency of smoking

Gym going / Smoking	Always	Never	Sometimes
Heavy	7	1	3
Never	87	18	84
Occasional	12	3	4
Regular	9	1	7

Test if the relation is significant or not..

Find the **p value.**.

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

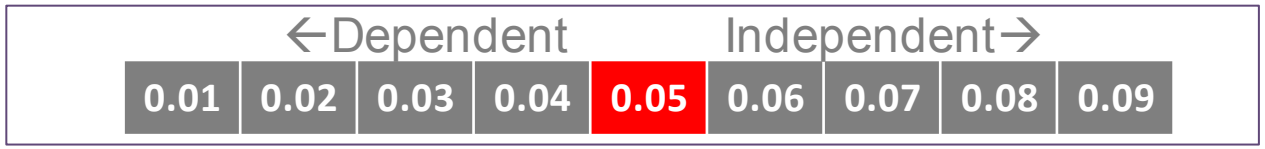
Example:

Dataset consists of frequency of people going to gym and frequency of smoking

Gym going / Smoking	Always	Never	Sometimes
Heavy	7	1	3
Never	87	18	84
Occasional	12	3	4
Regular	9	1	7

Test if the relation is significant or not..

If **p value** is less than **0.05** than there is a dependency otherwise not



IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

```
In [46]: import numpy as np
         from scipy import stats
         obs= np.array([[7,1,3],[87,18,84],[12,3,4],[9,1,7]])
         chi2,p,dof,expected=stats.chi2_contingency(obs)
         p
```

```
Out[46]: 0.48284216946545633
```

P value is greater than 0.05

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.stats()?

```
In [46]: import numpy as np
         from scipy import stats
         obs= np.array([[7,1,3],[87,18,84],[12,3,4],[9,1,7]])
         chi2,p,dof,expected=stats.chi2_contingency(obs)
         p
```

```
Out[46]: 0.48284216946545633
```

P value is greater than 0.05

No dependency between frequency of smoking and frequency of going to gym for our observed data.

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.optimize()?

- Provides algorithms for function minimization (scalar or multi-dimensional)
- Can also be used for curve fitting and root finding

IntelliPaat

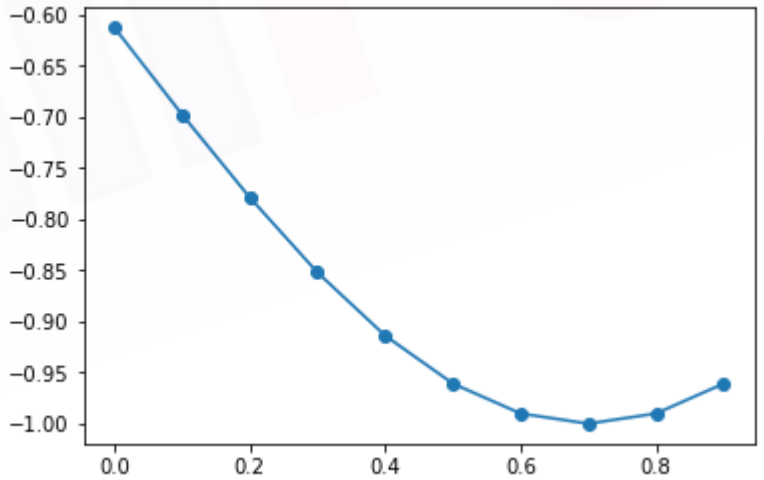
SciPy

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

Why do we use scipy.optimize()?

```
In [80]: import matplotlib.pyplot as plt
         import numpy as np
         x= np.arange(0.0, 1.0, 0.1)
         def f(x):
             return -np.exp(-(x-0.7)**2)
         plt.plot(x,f(x),'o-')
```



IntelliPaat

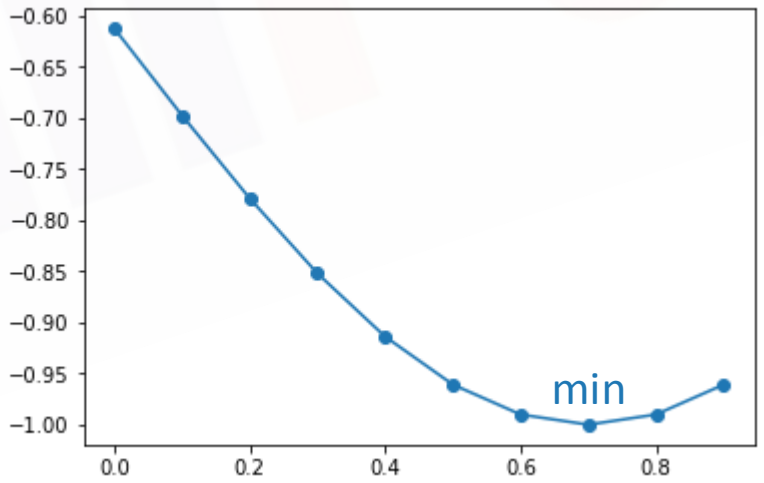
SciPy

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

Why do we use scipy.optimize()?

```
In [80]: import matplotlib.pyplot as plt
         import numpy as np
         x= np.arange(0.0, 1.0, 0.1)
         def f(x):
             return -np.exp(-(x-0.7)**2)
         plt.plot(x,f(x),'o-')
```



IntelliPaat

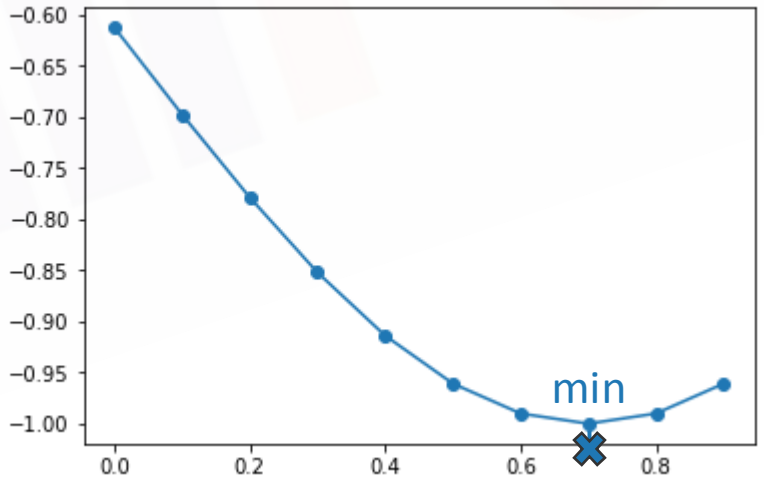
Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.optimize()?

```
In [80]: import matplotlib.pyplot as plt
         import numpy as np
         x= np.arange(0.0, 1.0, 0.1)
         def f(x):
             return -np.exp(-(x-0.7)**2)
         plt.plot(x,f(x),'o-')
```



Manually: f(0.1), f(0.2), ..., f(1). But when data is more??

IntelliPaat

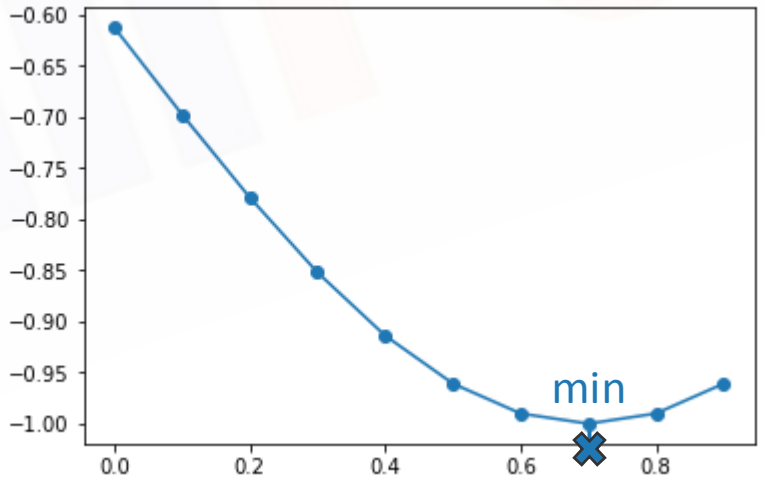
Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.optimize()?

```
In [80]: import matplotlib.pyplot as plt
         import numpy as np
         x= np.arange(0.0, 1.0, 0.1)
         def f(x):
             return -np.exp(-(x-0.7)**2)
         plt.plot(x,f(x),'o-')
```



That is when scipy.optimize() comes in handy.

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.optimize()?

```
In [55]: from scipy import optimize
         import numpy as np
         def f(x):
             return -np.exp(-(x-0.7)**2)
         result = optimize.minimize_scalar(f)
         x_min = result.x
         x_min
```

```
Out[55]: 0.6999999997839409
```

This means that at x=0.699 we get a minimum out of our function.

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.integrate()?

- When analytical integration becomes difficult, numerical integration comes handy.
- Offers number of routines for performing numerical integration.
- Offers **quad** for single integration, **dblquad** for double integration, **tplquad** for triple integration, **nquad** for n-fold multiple integration.

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.integrate()?

Example:

If f(x) = x^2, y = f(x) then find the value of $\int_a^b y\ dx$

$$where\ a = 0\ and\ b = 1$$

Mathematically,

y = $x^n = x^2$

$$\int_a^b x^n dx = \frac{a^{(n+1)}}{(n+1)} - \frac{b^{(n+1)}}{(n+1)}$$

Here, n=2,

$$\int_0^1 x^2 dx = \frac{1^{(2+1)}}{(2+1)} - 0 = 0.333$$

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.integrate()?

```
In [82]: import scipy.integrate as intg

         def integrad(x):
             return x**2
         ans, _ = intg.quad(integrad,0,1)
         ans
```

```
Out[82]: 0.33333333333333337
```

Similarly,

For double integration use dblquad

For triple integration use tplquad

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.linalg()?

- Provides standard linear algebra operation
- Compute determinant of a square matrix
- Compute inverse of a square matrix
- Compute eigen values of the matrix
- Can find out singular-value decomposition, commonly used in statistics and signal processing

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.linalg()?

Determinant of a square matrix

```
In [84]: from scipy import linalg
         import numpy as np
         data = np.array([[1,2,3],[3,4,5],[5,6,7]])
         linalg.det(data)
```

```
Out[84]: -1.1842378929335004e-15
```

Inverse of a square matrix

```
In [85]: from scipy import linalg
         import numpy as np
         data = np.array([[1,2,3],[3,4,5],[5,6,7]])
         linalg.inv(data)
```

```
Out[85]: array([[-1.18515780e+15,  2.37031559e+15, -1.18515780e+15],
                [ 2.37031559e+15, -4.74063119e+15,  2.37031559e+15],
                [-1.18515780e+15,  2.37031559e+15, -1.18515780e+15]])
```

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.linalg()?

Eigen values of a square matrix

```
In [86]: from scipy import linalg
         import numpy as np
         data = np.array([[1,2,3],[3,4,5],[5,6,7]])
         linalg.eigvals(data)
```

```
Out[86]: array([ 1.29282032e+01+0.j, -9.28203230e-01+0.j,  6.16237757e-16+0.j])
```

IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.fftpack()?

- Computes fast Fourier transforms (FFTs)
- An efficient implementation of the discrete Fourier Transform (DFT)

It provides fucntions such as:

- scipy.fftpack.fft() to compute the FFT
- scipy.fftpack.fftfreq() to generate the sampling frequencies
- scipy.fftpack.ifft() to compute the inverse FFT, from frequency space to signal space

IntelliPaat

Sub-packages in SciPy

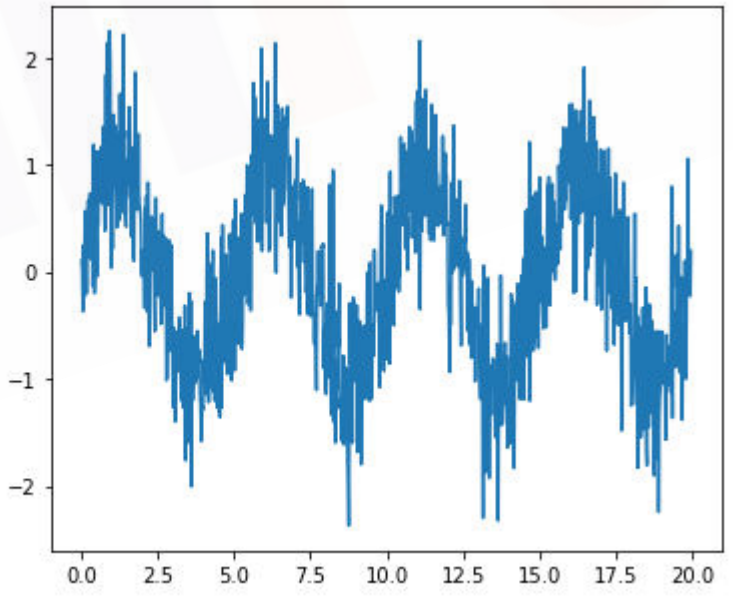
- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.fftpack()?

Generate a noisy signal:

```
In [90]: import numpy as np
         time_step = 0.02
         period = 5.
         time_vec = np.arange(0, 20, time_step)
         sig = np.sin(2 * np.pi / period * time_vec) + 0.5 *np.random.randn(time_vec.size)
         plt.figure(figsize=(6, 5))
         plt.plot(time_vec, sig)
         plt.show()
```



IntelliPaat

SciPy

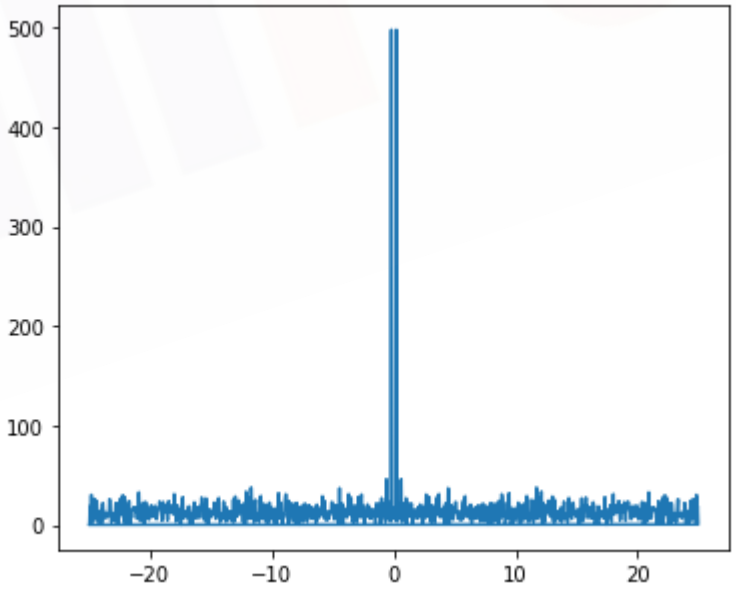
Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

How do we use scipy.fftpack()?

Applying fft:

```
In [92]: from scipy import fftpack
         sample_freq = fftpack.fftfreq(sig.size, d = time_step)
         sig_fft = fftpack.fft(sig)
         power = np.abs(sig_fft)
         sample_freq = fftpack.fftfreq(sig.size, d=time_step)
         plt.figure(figsize=(6, 5))
         plt.plot(sample_freq, power)
         plt.show()
```



IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

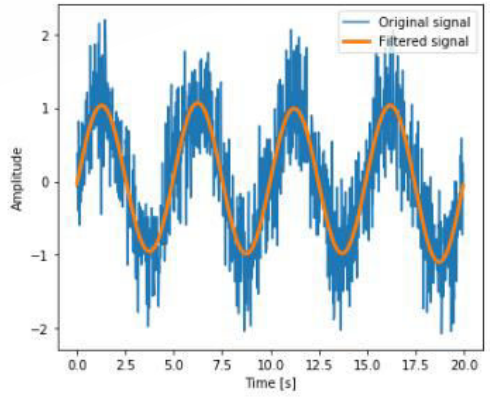
How do we use scipy.fftpack()?

Applying ifft to get the noiseless signal:

```
In [10]: #Filter out the sample frequencies that are greater than 0 with numpy.where(condition)
pos_mask = np.where(sample_freq > 0)
#Apply the fiter on smaple_freq and store the +ve sample_freq on freqs
freqs = sample_freq[pos_mask]

#print(power[pos_mask].argmax())
#Find the peak frequency, here we focus on only the positive frequencies
peak_freq = freqs[power[pos_mask].argmax()]
#now get an array copy of the signal where we already applied fft.
high_freq_fft = sig_fft.copy()
#assign the ones greater than peak freq as 0 in order to remove the noise
high_freq_fft[np.abs(sample_freq) > peak_freq] = 0

#print(high_freq_fft)
#Now apply inverese fft on the new high_freq_fft this will be the filtered signal
filtered_sig = fftpack.ifft(high_freq_fft)
#plot
plt.figure(figsize=(6, 5))
#now plot the original signal for reference
plt.plot(time_vec, sig, label='Original signal')
#now plot the filtered signal
plt.plot(time_vec, filtered_sig, linewidth=3, label='Filtered signal')
#add label, legend
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
plt.legend(loc='best')
#show
plt.show()
```



IntelliPaat

Sub-packages in SciPy

- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

Why do we use scipy.signal()?

- Used for typical signal processing(1D Dimensional and periodic signals)
- Say, you need to resample a signal with n data points to x data points, scipy.signal is used. Scipy resamples data points by using FFT.

IntelliPaat

Sub-packages in SciPy

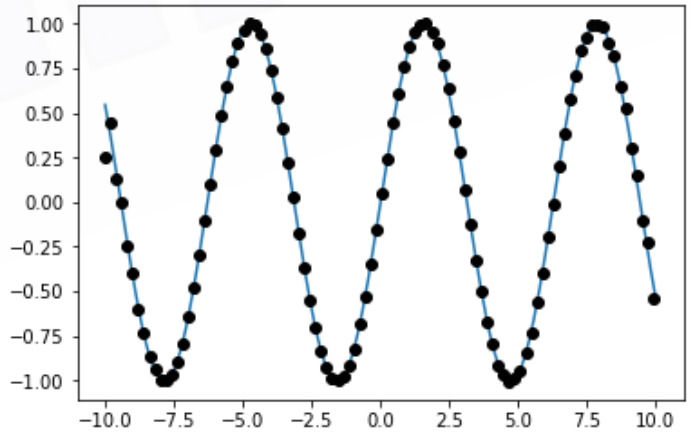
- scipy.cluster
- scipy.stats
- scipy.optimize
- scipy.integrate
- scipy.linalg
- scipy.fftpack
- scipy.signal

SciPy

How do we use scipy.signal()?

Resampling:

```
In [33]: #scipy.signal uses FFT to resample a 1D signal.
         import matplotlib.pyplot as plt
         import numpy as np
         from scipy import signal
         #Now let us create a signal with 200 data point
         t = np.linspace(-10, 10, 200) #Defining Time Interval
         y = np.sin(t)
         x_resampled=signal.resample(y, 100) #Number of required samples is 100
         plt.plot(t, y)
         #for x axis slice t into 2 step size
         plt.plot(t[::2], x_resampled, 'o')
         plt.show()
```



QUIZ

Quiz 1

Which of the following will be used to find out the zscore of the given numpy array?

```
from scipy import stats
import numpy as np
numarr=np.array([9,8,7,6,5,4,3,2,1,0])
```

A	np.stats.zscore(numarr)
B	stats.zscore(numarr)
C	stats.zcores(numarr)
D	np.stats.zscores(numarr)

Answer 1

Which of the following will be used to find out the zscore of the given numpy array?

A	np.stats.zscore(numarr)
B	stats.zscore(numarr)
C	stats.zcores(numarr)
D	np.stats.zscores(numarr)

```
from scipy import stats
import numpy as np
numarr=np.array([9,8,7,6,5,4,3,2,1,
0])
```

Quiz 2

Find out the correct output:

```
import numpy as np
from scipy import stats
new=np.array([[0.1,1,2],[3,0.2,1],[
1,0.5,4]])
_,p,_,_
=stats.chi2_contingency(new)
```

A	0.2464
B	1.0929
C	2.3111
D	None of the above

Answer 2

Find out the correct output:	
A	0.2464
B	1.0929
C	2.3111
D	None of the above

```
import numpy as np
from scipy import stats
new=np.array([[0.1,1,2],[3,0.2,1],[
1,0.5,4]])
_,p,_,_
=stats.chi2_contingency(new)
```

Quiz 3

Find out the correct output:	
A	8.37
B	-8.37
C	1
D	None of the above

```
from scipy import linalg
import numpy as np
new=np.array([[0.1,1,2],[3,0.2,1],[
1,0.5,4]])
linalg.det(new)
```

Answer 3

Find out the correct output:	
A	8.37
B	-8.37
C	1
D	None of the above

```
from scipy import linalg
import numpy as np
new=np.array([[0.1,1,2],[3,0.2,1],[
1,0.5,4]])
linalg.det(new)
```

Quiz 4

Find out the correct cluster index array for the following:

```
from scipy.cluster.vq import
kmeans,vq,whiten
import matplotlib.pyplot as plt
new=[0.1,1,3,0.2,1,0.5,12,0.2,10,
0.5,10,2,5,0.8]
plt.scatter(range(len(new)),new)
centroid,_ = kmeans(new,2)
idx, _ = vq(new,centroid)
idx
```

A array([1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1])

B array([1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1])

C array([1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1])

D array([1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1])

Answer 4

Find out the correct cluster index array for the following:

A	array([1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1])
B	array([1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1])
C	array([1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1])
D	array([1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1])

```
from scipy.cluster.vq import
kmeans,vq,whiten
import matplotlib.pyplot as plt
new=[0.1,1,3,0.2,1,0.5,12,0.2,10,
0.5,10,2,5,0.8]
plt.scatter(range(len(new)),new)
centroid,_ = kmeans(new,2)
idx, _ = vq(new,centroid)
idx
```

Quiz 5

Which of the following is used to find double integration?

A	doublequad
B	dblquad
C	quadquad
D	None

Answer 5

Which of the following is used to find double integration?

A	doublequad
B	dblquad
C	quadquad
D	None

Quiz 6

What is the output of the following?

```
import numpy
array_1 =
numpy.array([[1,2,3],[0,0,0]])
array_2 =
numpy.array([[0,0,0],[7,8,9]])
print(numpy.concatenate((array_1
, array_2), axis = 1))
```

A	[[1 2 3 0 0 0 0 0 0 7 8 9]]
B	[[1 2 3 0 0 0]
C	[[1 2 3] [7 8 9]]
D	None

Answer 6

What is the output of the following?

```
import numpy
array_1 =
numpy.array([[1,2,3],[0,0,0]])
array_2 =
numpy.array([[0,0,0],[7,8,9]])
print(numpy.concatenate((array_1
, array_2), axis = 1))
```

A	[[1 2 3 0 0 0 0 0 0 7 8 9]]
B	[[1 2 3 0 0 0]
C	[[1 2 3] [7 8 9]]
D	None

Quiz 7

What is the output?

```
import numpy
my_array = numpy.array([1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9])
print(numpy.ceil(my_array))
```

A. [1. 2. 3. 4. 5. 6. 7. 8. 9.]

B. [2. 3. 4. 5. 6. 7. 8. 9. 10.]

C. [.1 .2 .3 .4 .5 .6 .7 .8 .9]

D. [.2 .3 .4 .5 .6 .7 .8 .9 .10]

Answer 7

What is the output?

```
import numpy
my_array = numpy.array([1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9])
print(numpy.ceil(my_array))
```

A. [1. 2. 3. 4. 5. 6. 7. 8. 9.]

B. [2. 3. 4. 5. 6. 7. 8. 9. 10.]

C. [.1 .2 .3 .4 .5 .6 .7 .8 .9]

D. [.2 .3 .4 .5 .6 .7 .8 .9 .10]

IntelliPaat

www.intellipaat.com

CALL US NOW

India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com