Statistics using Numpy and Scipy

February 20, 2025

1 Basic Statistical Functions using Numpy and Scipy

Using NumPy Module

[ ]:
```
import numpy as np
```

Statistical Operation

1. mean()- This function basically calculates the average from an array of data.

[ ]:
```
x=np.array([1,2,3,4,5])
print("Output of list items in x\n")
print(x.mean())
```

```
Output of list items in x

3.0
```

Here below, first we have created the 2D array named array1. We then calculated the mean using np.mean().

np.mean(array1) - calculates the mean over the entire array

np.mean(array1, axis=0) - calculates the mean along vertical axis

np.mean(array1, axis=1) calculates the mean along horizontal axis

[ ]:
```
# create a 2D array
array1 = np.array([[1, 3],
                   [5, 7]])

# calculate the mean of the entire array
result1 = np.mean(array1)
print("Entire Array:",result1)  # 4.0

# calculate the mean along vertical axis (axis=0)
result2 = np.mean(array1, axis=0)
print("Along Vertical Axis:",result2)  # [3. 5.]

# calculate the mean along  (axis=1)
result3 = np.mean(array1, axis=1)
```

```
print("Along Horizontal Axis :",result3)  # [2. 6.]
```

```
Entire Array: 4.0
Along Vertical Axis: [3. 5.]
Along Horizontal Axis : [2. 6.]
```

2. median()- This function calculates the statistical median of the elements of the array given.

[ ]:
```
x=np.array([1,2,3,4,5])
print("Output of odd no. of list items in x\n")
print(np.median(x))
```

```
Output of odd no. of list items in x

3.0
```

[ ]:
```
x=np.array([1,2,3,4,5,6])
print("Output of even no. of list items in x\n")
print(np.median(x))
```

```
Output of even no. of list items in x

3.5
```

Calculation of the median is not just limited to 1D array. We can also calculate the median of the 2D array.

In a 2D array, median can be calculated either along the horizontal or the vertical axis individually, or across the entire array.

When computing the median of a 2D array, we use the axis parameter inside np.median() to specify the axis along which to compute the median.

If we specify,

axis = 0, median is calculated along vertical axis axis = 1, median is calculated along horizontal axis If we don't use the axis parameter, the median is computed over the entire array.

[ ]:
```
# create a 2D array
array1 = np.array([[2, 4, 6],
                   [8, 10, 12],
                   [14, 16, 18]])

# compute median along horizontal axis
result1 = np.median(array1, axis=1)

print("Median along horizontal axis :", result1)

# compute median along vertical axis
result2 = np.median(array1, axis=0)
```

```
print("Median along vertical axis:", result2)

# compute median of entire array
result3 = np.median(array1)

print("Median of entire array:", result3)
```

```
Median along horizontal axis : [ 4. 10. 16.]
Median along vertical axis: [ 8. 10. 12.]
Median of entire array: 10.0
```

3.std()- This function calculates the statistical standard deviation of the elements of the array given

The standard deviation is a measure of the spread of the data in the array. It gives us the degree to which the data points in an array deviate from the mean.

Smaller standard deviation indicates that the data points are closer to the mean

Larger standard deviation indicates that the data points are more spread out.

[ ]:
```
# create a numpy array
marks = np.array([76, 78, 81, 66, 85])

# compute the standard deviation of marks
std_marks = np.std(marks)
print("Standard Deviation of the above marks list")
print(std_marks)
```

```
6.368673331236263
```

In a 2D array, standard deviation can be calculated either along the horizontal or the vertical axis individually, or across the entire array.

Similar to mean and median, when computing the standard deviation of a 2D array, we use the axis parameter inside np.std() to specify the axis along which to compute the standard deviation.

[ ]:
```
# create a 2D array
array1 = np.array([[2, 5, 9],
                   [3, 8, 11],
                   [4, 6, 7]])

# compute standard deviation along horizontal axis
result1 = np.std(array1, axis=1)
print("Standard deviation along horizontal axis:", result1)

# compute standard deviation along vertical axis
result2 = np.std(array1, axis=0)
print("Standard deviation  along vertical axis:", result2)

# compute standard deviation of entire array
```

```
result3 = np.std(array1)
print("Standard deviation of entire array:", result3)
```

```
Standard deviation along horizontal axis: [2.86744176 3.29983165 1.24721913]
Standard deviation  along vertical axis: [0.81649658 1.24721913 1.63299316]
Standard deviation of entire array: 2.7666443551086073
```

4. percentile()- In NumPy, we use the percentile() function to compute the nth percentile of a given array.

[ ]:
```
# create an array
array1 = np.array([1, 3, 5, 7, 9, 11, 13, 15, 17, 19])

# compute the 25th percentile of the array
result1 = np.percentile(array1, 25)
print("25th percentile:",result1)

# compute the 75th percentile of the array
result2 = np.percentile(array1, 75)
print("75th percentile:",result2)
```

```
25th percentile: 5.5
75th percentile: 14.5
```

5. min()- Finds the minimum value of the array.
6. max()- Finds the maximum value of the array.

[ ]:
```
# create an array
array1 = np.array([2,6,9,15,17,22,65,1,62])

# find the minimum value of the array
min_val = np.min(array1)

# find the maximum value of the array
max_val = np.max(array1)

# print the results
print("Minimum value:", min_val)
print("Maximum value:", max_val)
```

```
Minimum value: 1
Maximum value: 65
```

7. cov()- This function calculates the statistical covarience between 2 matrices or arrays of same dimension

[ ]:
```
x = np.array([[0, 3, 4], [1, 2, 4], [3, 4, 5]])

print("Shape of array:\n", np.shape(x))
```

```
print("Covariance matrix of x:\n", np.cov(x))
```

```
Shape of array:
 (3, 3)
Covariance matrix of x:
 [[4.33333333 2.83333333 2.        ]
 [2.83333333 2.33333333 1.5       ]
 [2.         1.5        1.        ]]
```

Using Scipy Module

[ ]:
```
from scipy import stats
```

1. mode()- It takes an array as input and returns the mode along with their corresponding counts.

Mode is the data item in the dataset having largest occurance.:

[ ]:
```
data = [1, 2, 2, 3, 1, 1, 5]

print("Mode:", stats.mode(data))
```

```
Mode: ModeResult(mode=1, count=3)
```