Data Analysis with SciPy

Last Updated : 19 Feb, 2024

Scipy is a Python library useful for solving many mathematical equations and algorithms. It is designed on the top of Numpy library that gives more extension of finding scientific mathematical formulae like Matrix Rank, Inverse, polynomial equations, LU Decomposition, etc. Using its high-level functions will significantly reduce the complexity of the code and helps better in analyzing the data.

In this article, we will explore What is SciPy, the Installation of SciPy, and How Data Analysis with SciPy works and Compute pivoted LU decomposition.

Table of Content

What is SciPy?

SciPy is an interactive Python session used as a data-processing library that is made to compete with its rivalries such as MATLAB, Octave, R-Lab, etc. It has many user-friendly, efficient, and easy-to-use functions that help to solve problems like numerical integration, interpolation, optimization, linear algebra, and statistics. The benefit of using the SciPy library in Python while making ML models is that it makes a strong programming language available for developing fewer complex programs and applications.

Installation of SciPy

To install SciPy in your system, you can use Python package manager pip. Before proceeding, make sure that you have Python already installed in your system. Here's the step to install Python in your system.

Step 1: Firstly, Open terminal and Command Prompt in your system.

Step 2: Run the installation Command to install SciPy in your system.

```
pip install scipy
```

Step 3: Pip will be download and install SciPy along with dependencies. This process will may take some time depends on internet connection.

Step 4: To verify installation, you need to import SciPy in a Python script or interactive shell.

```
import scipy

print(scipy.__version__)
```

Now the installation of SciPy is successfully completed.

How does Data Analysis work with SciPy?

Data Preparation

- Import the necessary libraries: import numpy as np and import scipy as sp.
- Load or generate your dataset using NumPy or pandas.

Exploratory Data Analysis (EDA)

- Use descriptive statistics from SciPy's stats module to gain insights into the dataset.

- Calculate measures such as mean, median, standard deviation, skewness, kurtosis, etc.

Python3

```
from scipy import stats
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
# Calculate mean and standard deviation
mean_val = np.mean(data)
std_dev = np.std(data)
# Perform basic statistical tests
t_stat, p_value = stats.ttest_1samp(data, popmean=5)
print("t_stat:" , t_stat)
print("p_value:", p_value)
```

Output:

```
t_stat: 0.0
p_value: 1.0
```

Statistical Hypothesis Testing

Use SciPy's stats module for various hypothesis tests such as t-tests, chi-square tests, ANOVA, etc.

> *# Example of a t-test*
>
> *t_stat, p_value = stats.ttest_ind(group1, group2)*

Regression Analysis

Utilize the linregress function for linear regression analysis.

> *# Linear regression*

slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)

Signal and Image Processing

- Use the scipy.signal module for signal processing operations.
- Explore scipy.ndimage for image processing.

from scipy import signal, ndimage

 # Example of convolution using signal processing

result = signal.convolve2d(image, kernel, mode='same', boundary='wrap')

Optimization

Employ the optimization functions in SciPy to find optimal parameter values.

from scipy.optimize import minimize

 # Define an objective function

def objective_function(x):

*return x[0]**2 + x[1]**2*

result = minimize(objective_function, [0, 0])

Import SciPy

Once SciPy is installed , you need to import the SciPy module(s)

Python3

```
# import numpy library
```

```
import numpy as np
A= [[1, 2,],
    [5, 6,]]
```

Linear Algebra

Determinant of a Matrix

Python3

```
# importing linalg function from scipy
from scipy import linalg
A= [[1, 2,],
    [5, 6,]]

# Compute the determinant of a matrix
linalg.det(A)
```

Output:

```
-4.0
```

Compute pivoted LU decomposition of a Matrix

LU decomposition is a method that reduce matrix into constituent parts that helps in easier calculation of complex matrix operations. The decomposition methods are also called matrix factorization methods, are base of linear algebra in computers, even for basic operations such as solving systems of linear equations, calculating the inverse, and calculating the determinant of a matrix. The decomposition is: A = P L U where P is a permutation matrix, L lower triangular with unit diagonal elements, and U upper triangular.

Python3

```
P, L, U = linalg.lu(A)
print(P)
print(L)
print(U)
```

```
# print LU decomposition
print(np.dot(L,U))
```

Output:

```
[[0. 1. 0.]
[0. 0. 1.]
[1. 0. 0.]]
[[1.         0.         0.        ]
 [0.14285714 1.         0.        ]
 [0.57142857 0.5        1.        ]]
[[7.         8.         8.        ]
 [0.         0.85714286 1.85714286]
 [0.         0.         0.5       ]]
[[0.14285714 1.         0.        ]
 [0.57142857 0.5        1.        ]
  [1.         0.         0.        ]]
```

Eigen values and Eigen vectors of this matrix

Python3

```
eigen_values, eigen_vectors = linalg.eig(A)
print(eigen_values)
print(eigen_vectors)
```

Output:

```
array([ 15.55528261+0.j,  -1.41940876+0.j,  -0.13587385+0.j])
array([[-0.24043423, -0.67468642,  0.51853459],
       [-0.54694322, -0.23391616, -0.78895962],
       [-0.80190056,  0.70005819,  0.32964312]])
```

Solving systems of linear equations can also be done

Python3

```
v = np.array([[2, 4],[3,8]])
```

```
print(v)
s = linalg.solve(A,v)
print(s)
```

Output:

```
[[2 4]
 [3 8]]
[[-1.5  -2.  ]
 [ 1.75  3.  ]]
```

Sparse Linear Algebra

SciPy has some routines for computing with sparse and potentially very large matrices. The necessary tools are in the submodule scipy.sparse.

Let’s look on how to construct a large sparse matrix

Python3

```
# import necessary modules
from scipy import sparse
# Row-based linked list sparse matrix
A = sparse.lil_matrix((1000, 1000))
print(A)
A[0,:100] = np.random.rand(100)
A[1,100:200] = A[0,:100]
A.setdiag(np.random.rand(1000))
print(A)
```

Output:

```
  (0, 0)    0.7948113035416484
  (0, 1)    0.22210781047073025
  (0, 2)    0.1198653673336828
  (0, 3)    0.33761517140362796
  (0, 4)    0.9429097039125192
  (0, 5)    0.32320293202075523
  (0, 6)    0.5187906217433661
  (0, 7)    0.7030189588951778
```

```
  (0, 8)    0.363629602379294
  (0, 9)    0.9717820827209607
  (0, 10)    0.9624472949421112
  (0, 11)    0.25178229582536416
  (0, 12)    0.49724850589238545
  (0, 13)    0.30087830981676966
  (0, 14)    0.2848404943774676
  (0, 15)    0.036886947354532795
```

Linear Algebra for Sparse Matrices

Python3

```
from scipy.sparse import linalg
# Convert this matrix to Compressed Sparse Row format.
A.tocsr()
A = A.tocsr()
b = np.random.rand(1000)
ans = linalg.spsolve(A, b)
# it will print ans array of 1000 size
print(ans)
```

Output:

```
[-4.67207136e+01 -3.69332972e+02  3.69393775e-01  6.32141409e-02
  3.33772205e+00  5.10104872e-01  3.07850190e+00  1.94608719e+01
  1.49997674e+00  1.04751174e+00  9.23616608e-01  8.14103772e-01
  8.42662424e-01  2.28221903e+00  4.92361307e+01  6.74574814e-01
  3.06515031e-01  3.36481843e-02  9.55613073e-01  7.22911464e-01
  2.70518013e+00  1.25039001e+00  1.37825326e-01  3.95005049e-01
  4.04480605e+00  7.72817743e-01  2.14200400e-01  7.06283767e-01
  1.12635170e-01  5.98880840e+00  4.37382510e-01  8.05571435e-01
 ......................................................................
 ......................................................]
```

Python3

```
from scipy import integrate
f = lambda y, x: x*y**2
i = integrate.dblquad(f, 0, 2, lambda x: 0, lambda x: 1)
```

```
# print the results
print(i)
```

Output:

```
(0.6666666666666667, 7.401486830834377e-15)
```

There is a lot more that SciPy is capable of, such as Fourier Transforms, Bessel Functions, etc.

Previous Article
NumPy Tutorial - Python Library

Next Article
Introduction to TensorFlow

Similar Reads

Factor Analysis | Data Analysis

Factor analysis is a statistical method used to analyze the relationships among a set of observed variables by explaining the correlations or covariances between...

13 min read

RFM Analysis Analysis Using Python

In this article, we are going to see Recency, Frequency, Monetary value analysis using Python. But first, let us understand the RFM analysis briefly. What is RFM...

3 min read

Difference Between Factor Analysis and Principal Component Analysis

Factor Analysis (FA) and Principal Component Analysis (PCA) are two pivotal techniques used for data reduction and structure detection. Despite their...

4 min read

Different Sources of Data for Data Analysis

Data collection is the process of acquiring, collecting, extracting, and storing the voluminous amount of data which may be in the structured or unstructured form...

5 min read

SciPy - Integration of a Differential Equation for Curve Fit

In Machine Learning, often what we do is gather data, visualize it, then fit a curve in the graph and then predict certain parameters based on the curve fit. If we...

2 min read

SciPy | Curve Fitting

Given a Dataset comprising of a group of points, find the best fit representing the Data.We often have a dataset comprising of data following a general path, but...

4 min read

SciPy - Cluster

Clustering is nothing but it is the procedure of dividing the datasets into groups consisting of similar data points. In this procedure, the data points in the same...

4 min read

Multidimensional image processing using Scipy in Python

SciPy is a Python library used for scientific and technical computing. It is built on top of NumPy, a library for efficient numerical computing, and provides many...

14 min read

How to Compute Entropy using SciPy?

Entropy is a fundamental concept in measuring the uncertainty or randomness in a dataset. Entropy plays a very significant role in machine learning models such...

6 min read

SciPy - Agglomerative Clustering

Agglomerative clustering, also known as hierarchical clustering, is one of the most popular clustering techniques in data analysis and machine learning. It...

4 min read

Article Tags : Machine Learning data-science python

Practice Tags : Machine Learning python