Numpy

Ferramentas para Data Science

Carolina Ribeiro Xavier
CCOMP - UFSJ

Numpy

NumPy é o pacote fundamental para computação científica em Python. É uma biblioteca que fornece um objeto de array multidimensional, vários objetos derivados e uma variedade de rotinas para operações rápidas em arrays, incluindo matemática, lógica, manipulação de forma, classificação, seleção, entrada/saída, transformadas discretas de Fourier, álgebra linear básica, operações estatísticas básicas, simulação aleatória e muito mais. (link)

Criação de Arrays com NumPy

- Criando arrays simples:

```
import numpy as np
arr = np.array([1, 2, 3, 4])
```

- Arrays multidimensionais:

```
arr_2d = np.array([[1, 2], [3, 4]])
```

- Funções úteis: ‘np.zeros()‘, ‘np.ones()‘, ‘np.arange()‘.

Indexação e Slicing

- Indexação básica de arrays:

```
arr[0]  # Acessa o primeiro elemento
```

- Slicing:

```
arr[1:3]  # Acessa elementos do índice 1 até 2
```

Operações Básicas em Arrays

- Soma, subtração, multiplicação e divisão diretamente nos arrays:

```
arr + 2  # Soma de cada elemento com 2
arr * 3  # Multiplicação de cada elemento por 3
```

- Operações entre arrays:

```
arr1 + arr2  # Soma elemento a elemento
```

Vetorização em NumPy

- A vetorização permite operações rápidas, sem loops explícitos.
- Exemplo:

```
import numpy as np
arr = np.arange(1000000)
resultado = arr * 2  # Mais rápido que usar loops
```

Broadcasting em NumPy

- O broadcasting permite que operações sejam realizadas em arrays de formas diferentes.
- Exemplo de broadcasting:

```
matriz = np.array([[1, 2, 3], [4, 5, 6]])
vetor = np.array([1, 0, 1])
resultado = matriz + vetor
```

Álgebra Linear com NumPy

- Operações como multiplicação de matrizes e decomposições podem ser feitas com NumPy.
- Exemplo de multiplicação de matrizes:

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
resultado = np.dot(A, B)
```

Decomposição de Matrizes com NumPy

- Decomposição em valores singulares (SVD):

```
U, S, V = np.linalg.svd(A)
```

- Autovalores e autovetores:

```
eigenvalues, eigenvectors = np.linalg.eig(A)
```

Manipulação Avançada de Arrays

- Reshaping de arrays:

```
arr = np.arange(12).reshape(3, 4)
```

- Transposição de arrays:

```
arr_T = arr.T
```

Operações Estatísticas

- Cálculos de média, mediana e desvio padrão:

  `np.mean(arr), np.median(arr), np.std(arr)`

- Percentis e quantis:

  `np.percentile(arr, 25), np.quantile(arr, 0.5)`

Manipulação de Dados Faltantes

- Detectando valores ausentes:

```
np.isnan(arr)  # Verifica valores NaN
```

- Substituindo valores faltantes:

```
arr[np.isnan(arr)] = 0  # Substitui NaN por 0
```

Funções Universais (ufuncs)

- Funções universais para operações eficientes:

```
np.sqrt(arr)  # Raiz quadrada de cada elemento
np.log(arr)   # Logaritmo natural de cada elemento
```

Distribuições no NumPy

O NumPy oferece uma série de distribuições para simulação de dados aleatórios. Alguns exemplos incluem:
Distribuição Normal (Gaussiana):

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

- `np.random.normal(loc, scale, size)`: Gera dados com média μ e desvio padrão σ.

Distribuições no NumPy

Distribuição Exponencial:

$$X \sim \text{Exp}(\lambda)$$

- `np.random.exponential(scale, size)`: Gera dados com média $1/\lambda$.

Distribuição Uniforme:

$$X \sim \text{Uniform}(a, b)$$

- `np.random.uniform(low, high, size)`: Gera dados entre a e b.

Distribuições no NumPy

Distribuição Binomial:

$$X \sim \text{Binomial}(n, p)$$

- `np.random.binomial(n, p, size)`: Gera dados com n tentativas e probabilidade de sucesso p.

Exemplo de uso:

- `np.random.normal(0, 1, 1000)`: Gera 1000 amostras de uma distribuição normal padrão.

Correlação em NumPy

A correlação é uma medida estatística que indica a relação entre duas variáveis. No NumPy, você pode calcular a correlação de várias maneiras.

Correlação de Pearson:

Mede a relação linear entre duas variáveis. Ou seja, ela quantifica o grau de associação linear entre as variáveis. Se os pontos de dados seguem uma linha reta (ou seja, uma relação linear), a correlação de Pearson será alta, positiva ou negativa.

Requisitos: Assume que as variáveis seguem uma distribuição normal e que a relação entre elas é linear.

Variação: Valores próximos a $+1$ indicam uma correlação positiva forte, -1 indicam uma correlação negativa forte e 0 indica nenhuma correlação linear.

- `np.corrcoef(X, Y)`: Calcula a matriz de correlação de Pearson entre X e Y.

Correlação em NumPy

Correlação de Spearman:
Mede a relação monotônica entre duas variáveis, o que significa que ela avalia se as variáveis tendem a aumentar ou diminuir juntas, mas não precisa ser uma relação linear. Ou seja, ela mede a associação entre as classificações (ranks) das variáveis, não os valores exatos.
Requisitos: Não faz suposições sobre a distribuição dos dados, sendo uma técnica não paramétrica. Isso significa que a correlação de Spearman pode ser usada mesmo quando os dados não seguem uma distribuição normal ou quando a relação não é linear, mas apenas monotônica (sempre crescente ou decrescente).
Variação: Assim como Pearson, valores próximos de $+1$ indicam uma forte correlação positiva, -1 uma forte correlação negativa, e 0 indica nenhuma relação monotônica.

- `scipy.stats.spearmanr(X, Y)`: Calcula a correlação de Spearman, que é uma versão não paramétrica da correlação de Pearson.

Correlação em NumPy

Correlação de Kendall:

- `scipy.stats.kendalltau(X, Y)`: Calcula a correlação de Kendall, outra medida não paramétrica.

Exemplo de uso em NumPy:

- `np.corrcoef(X, Y)`: Retorna uma matriz com a correlação entre os arrays X e Y.

Correlação em NumPy

Exemplo de código:

```
X = np.random.normal(0, 1, 1000)
Y = 2 * X + np.random.normal(0, 1, 1000)
corr_matrix = np.corrcoef(X, Y)
```

A saída da correlação de Pearson será uma matriz com valores entre -1 e 1.

Exercício Prático - Análise e Simulação de Dados Científicos com NumPy

Descrição do Desafio:
Você foi designado para criar uma análise e simulação numérica de dados em um estudo de variabilidade climática. A partir de dados aleatórios simulando temperaturas ao longo de anos, você deverá calcular estatísticas, detectar anomalias e realizar uma simulação simples de "cenário futuro" com mudanças climáticas.

Exercício Prático - Análise e Simulação de Dados Científicos com NumPy

Objetivos da Tarefa Gerar e Preparar Dados:
Gere um array que simule as temperaturas médias mensais (em °C) de uma cidade ao longo de 10 anos (120 valores). A distribuição deve ser centrada em 15°C, com variação sazonal e ruído aleatório.
Estatísticas e Análise Exploratória:
Calcule estatísticas básicas: média anual, desvio padrão e percentis (25%, 50%, 75%) das temperaturas.
Detecte meses "anômalos" onde a temperatura esteja 2 desvios padrão acima ou abaixo da média anual. Simulação de Cenário Futuro:

Exercício Prático - Análise e Simulação de Dados Científicos com NumPy

Simule um aumento gradual de temperatura de 0.1°C por ano para os próximos 50 anos. Gere um gráfico comparando a série histórica original com a projeção futura. Análise Comparativa e Visualização:
Crie visualizações (histogramas e box plots) para comparar a distribuição de temperaturas dos primeiros 10 anos com a projeção dos 50 anos seguintes.
Escreva um relatório, pode ser intercalado com o código no python notebook com a descrição dos dados e a explicação dos achados.