**NumPy** es una biblioteca fundamental para la computación científica en Python. Proporciona un objeto de matriz multidimensional (ndarray), varias funciones derivadas, herramientas para integrar código C/C++ y Fortran, capacidades de álgebra lineal, transformada de Fourier y generación de números aleatorios. A continuación se presenta un resumen de las funciones y métodos más importantes de NumPy en el contexto de la ciencia de datos.

**1. Creación de Arrays**

**- np.array():** Crea un array desde listas de Python.

**- np.zeros():** Crea un array lleno de ceros.

**- np.ones():** Crea un array lleno de unos.

**- np.full():** Crea un array lleno de un valor específico.

**- np.empty():** Crea un array sin inicializar.

**- np.arange():** Crea un array con valores dentro de un intervalo dado.

**- np.linspace():** Crea un array con valores espaciados uniformemente dentro de un intervalo.

**- np.eye():** Crea una matriz identidad.

**2. Propiedades y métodos del array**

**- ndarray.shape:** Devuelve una tupla con las dimensiones del array.

**- ndarray.size:** Devuelve el número total de elementos del array.

**- ndarray.ndim:** Devuelve el número de dimensiones del array.

**- ndarray.dtype:** Devuelve el tipo de datos de los elementos del array.

**- ndarray.astype():** Convierte el array a un tipo de datos diferente.

**3. Operaciones aritméticas y estadísticas**

**- np.add(), np.subtract(), np.multiply(), np.divide():** Operaciones aritméticas elementales.

**- np.power():** Eleva los elementos de un array a una potencia.

**- np.sqrt():** Calcula la raíz cuadrada de cada elemento del array.

**- np.exp():** Calcula la exponencial de todos los elementos del array.

**- np.log():** Calcula el logaritmo natural de cada elemento del array.

**- np.mean():** Calcula la media aritmética.

**- np.median():** Calcula la mediana.

**- np.std():** Calcula la desviación estándar.

**- np.var():** Calcula la varianza.

**- np.sum():** Calcula la suma de los elementos.

**- np.min(), np.max():** Encuentra el valor mínimo y máximo, respectivamente.

**4. Manipulación de arrays**

**- np.reshape():** Cambia la forma de un array sin cambiar sus datos.

**- np.ravel():** Aplana un array.

**- np.transpose():** Transpone un array.

**- np.concatenate():** Une una secuencia de arrays a lo largo de un eje existente.

**- np.split():** Divide un array en múltiples sub-arrays.

**- np.stack():** Une una secuencia de arrays a lo largo de un nuevo eje.

**- np.hstack(), np.vstack():** Apila arrays horizontal y verticalmente, respectivamente.

**5. Indexación y Slicing**

**- Indexación básica:** array[index]

**- Slicing:** array[start:stop:step]

**- Indexación booleana:** array[condition]

**- Indexación avanzada:** array[[row\_indices], [col\_indices]]

**6. Álgebra lineal**

**- np.dot():** Producto escalar de dos arrays.

**- np.matmul():** Producto matricial.

**- np.linalg.inv():** Calcula la inversa de una matriz.

**- np.linalg.det():** Calcula el determinante de una matriz.

**- np.linalg.eig():** Calcula los valores y vectores propios.

**- np.linalg.svd():** Descomposición en valores singulares.

**7. Generación de números aleatorios**

**- np.random.rand():** Genera un array de dimensiones dadas con valores aleatorios en el intervalo [0, 1).

**- np.random.randn():** Genera un array de dimensiones dadas con valores aleatorios de una distribución normal estándar.

**- np.random.randint():** Genera un array de enteros aleatorios.

**- np.random.choice():** Genera una muestra aleatoria de una array dada.

**- np.random.seed():** Establece la semilla para el generador de números aleatorios.

import numpy as np

# Creación de arrays

a = np.array([1, 2, 3])

b = np.zeros((2, 3))

c = np.ones((3, 3))

d = np.arange(10)

e = np.linspace(0, 1, 5)

# Operaciones aritméticas

f = a + 2

g = a \* b

# Estadísticas

mean\_a = np.mean(a)

std\_a = np.std(a)

# Álgebra lineal

h = np.dot(a, a)

i = np.linalg.inv(np.array([[1, 2], [3, 4]]))

# Números aleatorios

j = np.random.rand(2, 3)

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

# Manipulación de arrays

l = np.concatenate((a, d))

m = b.reshape(3, 2)

# Indexación y slicing

n = a[1]

o = d[2:5]

p = d[d > 5]

Este conjunto de funciones y métodos de *NumPy* proporciona una base sólida para la manipulación de datos, cálculos aritméticos y operaciones avanzadas en la ciencia de datos.