Documento 3: Arrays y Operaciones en NumPy
Curso Intermedio de Inteligencia Artificial
Contenido de Clase: Sesión 3

Continuación de NumPy

Funciones avanzadas de NumPy
- Funciones Universales (ufuncs):
  - Las funciones universales son funciones que operan elemento por elemento en arrays de NumPy, lo que permite realizar operaciones rápidas y eficientes.
  - Ejemplo de uso de funciones universales:

```
python
import numpy as np

arr = np.array([1, 2, 3, 4])
print(np.sqrt(arr))  # [1.  1.41421356  1.73205081  2. ]
print(np.exp(arr))   # [  2.71828183   7.3890561   20.08553692  54.59815003]
```

- Manipulación avanzada de arrays:
  - Reshape: Cambiar la forma de un array sin cambiar sus datos.

```
python
arr = np.array([1, 2, 3, 4, 5, 6])
arr2 = arr.reshape((2, 3))
print(arr2)
# Output:
# [[1 2 3]
#  [4 5 6]]
```

  - Ravel: Aplanar un array multidimensional.

```
python
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr.ravel())
# Output: [1 2 3 4 5 6]
```

Operaciones Matemáticas y Estadísticas con NumPy
- Operaciones algebraicas:
  - Producto punto (dot product) y producto cruzado (cross product):

```
python
vec1 = np.array([1, 2, 3])
vec2 = np.array([4, 5, 6])
print(np.dot(vec1, vec2))  # Producto punto: 32
print(np.cross(vec1, vec2))  # Producto cruzado: [-3  6 -3]
```

- Operaciones estadísticas:
  - Cálculo de la media, varianza y correlación:

```
python
arr = np.array([1, 2, 3, 4, 5])
print(np.mean(arr))  # Media: 3.0
print(np.var(arr))   # Varianza: 2.0
print(np.corrcoef(arr))  # Correlación: matriz de correlación
```

Referencias:
- Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*.
- Oliphant, T. E. (2006). *Guide to NumPy*.

Ejemplos de Industria

1. Finanzas: Modelado de Portafolios
- Descripción:
  - En el sector financiero, NumPy se utiliza para modelar y simular portafolios de inversiones, permitiendo cálculos rápidos y precisos de métricas de riesgo y retorno.
  - Ejemplo práctico: Calcular el rendimiento y el riesgo de un portafolio utilizando la varianza y la covarianza.
  - Código de ejemplo:

```
python
import numpy as np

# Retornos históricos de dos activos
retornos = np.array([[0.1, 0.2, 0.15], [0.05, 0.1, 0.2]])
cov_matrix = np.cov(retornos)
print("Matriz de covarianza:", cov_matrix)

# Calcular el rendimiento esperado y la varianza del portafolio
pesos = np.array([0.5, 0.5])
rendimiento_esperado = np.dot(pesos, np.mean(retornos, axis=1))
varianza_portafolio = np.dot(pesos.T, np.dot(cov_matrix, pesos))
print("Rendimiento esperado:", rendimiento_esperado)
print("Varianza del portafolio:", varianza_portafolio)
```

2. Salud: Procesamiento de Señales Médicas
- Descripción:
  - NumPy se utiliza en el procesamiento de señales médicas, como el análisis de señales ECG (electrocardiograma) para detectar anomalías en el ritmo cardíaco.
  - Ejemplo práctico: Filtrar una señal de ECG para eliminar el ruido.

- Código de ejemplo:

```
python
import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt

# Señal de ejemplo de ECG con ruido
t = np.linspace(0, 1, 500)
ecg_signal = np.sin(2 * np.pi * 5 * t) + 0.5 * np.random.randn(500)

# Filtrar la señal
b, a = signal.butter(4, 0.1)
ecg_filtered = signal.filtfilt(b, a, ecg_signal)

# Graficar la señal original y la señal filtrada
plt.plot(t, ecg_signal, label='Señal con ruido')
plt.plot(t, ecg_filtered, label='Señal filtrada', linewidth=2)
plt.legend()
plt.show()
```

3. Marketing: Análisis de Datos de Clientes

- Descripción:
  - Las empresas de marketing utilizan NumPy para analizar grandes volúmenes de datos de clientes, identificando patrones y segmentando audiencias para campañas personalizadas.
  - Ejemplo práctico: Realizar clustering de clientes basado en su comportamiento de compra.
  - Código de ejemplo:

```
python
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Datos de comportamiento de compra (ejemplo)
datos_clientes = np.array([[100, 10], [200, 20], [300, 30], [400, 40], [500, 50]])

# Aplicar K-means clustering
kmeans = KMeans(n_clusters=2)
kmeans.fit(datos_clientes)
etiquetas = kmeans.labels_

# Graficar los clusters
plt.scatter(datos_clientes[:, 0], datos_clientes[:, 1], c=etiquetas)
plt.xlabel('Gasto')
plt.ylabel('Compras')
plt.title('Clustering de Clientes')
```

```
plt.show()
```

Referencias adicionales:
- Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*.
- Oliphant, T. E. (2006). *Guide to NumPy*.
- McKinney, W. (2012). *Python for Data Analysis*.

Ejercicios prácticos

Manipulación avanzada de arrays con NumPy
- Ejercicio 1: Crear un array de 10x10 con valores aleatorios y encontrar los valores máximo y mínimo.

```python
arr = np.random.rand(10, 10)
max_val = np.max(arr)
min_val = np.min(arr)
print(f"Valor máximo: {max_val}, Valor mínimo: {min_val}")
```

- Ejercicio 2: Realizar operaciones de suma y multiplicación en arrays bidimensionales.

```python
arr1 = np.array([[1, 2, 3], [4, 5, 6]])
arr2 = np.array([[7, 8, 9], [10, 11, 12]])
suma = np.add(arr1, arr2)
multiplicacion = np.multiply(arr1, arr2)
print("Suma:\n", suma)
print("Multiplicación:\n", multiplicacion)
```

- Ejercicio 3: Calcular la media y la desviación estándar de un array bidimensional.

```python
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
media = np.mean(arr)
desviacion_estandar = np.std(arr)
print(f"Media: {media}, Desviación estándar: {desviacion_estandar}")
```

Referencias
- Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*.

- Oliphant, T. E. (2006). *Guide to NumPy*.
- McKinney, W. (2012). *Python for Data Analysis*.