



Cruz Quispe Luis Angel

Escuela de Ingeniería Física  
Universidad Nacional de Ingeniería  
Facultad de Ciencias [20221664C](#)

Rios Cusi Pedro David

Escuela de Ingeniería Física  
Universidad Nacional de Ingeniería  
Facultad de Ciencias [20230216J](#)

Mamani Benites Jimmy

Escuela de Ingeniería Física  
Universidad Nacional de Ingeniería  
Facultad de Ciencias [20211489D](#)

**Resumen**— Este proyecto explora el análisis de datos, machine learning y visualización utilizando las bibliotecas de Python Pandas, Matplotlib y NumPy. Pandas facilita la manipulación de datos tabulares, Matplotlib permite crear diversos gráficos, y NumPy es fundamental para operaciones numéricas eficientes. Se incluyen ejercicios prácticos que demuestran la integración y aplicabilidad de estas herramientas en el análisis y visualización de datos.

## I. INTRODUCCION

Este proyecto explora herramientas y técnicas clave de análisis de datos, machine learning, procesamiento de lenguaje natural, desarrollo web y automatización. Utilizamos las librerías de Python Pandas, Matplotlib y NumPy para la manipulación y visualización de datos.

## II. MARCO TEORICO

### II-A. Librerías

#### **Panda:**

Es una biblioteca de análisis de datos que proporciona estructuras de datos de alto nivel y herramientas para trabajar con datos estructurados o tabulares. Aquí están los puntos clave:

- I. **Dataframe:** Es la estructura central en Pandas, que representa datos en forma de tablas con filas y columnas etiquetadas. Los DataFrames permiten manipular datos de manera similar a una hoja de cálculo.
- II. **Series:** Es otra estructura importante en Pandas, que representa una columna unidimensional en un DataFrame, junto con un índice.
- III. **Operaciones de datos:** Pandas ofrece funcionalidades poderosas para limpiar, transformar, fusionar y analizar datos. Esto incluye manipulaciones de datos como filtros, agregaciones, agrupaciones y pivoteo.

## II-B. Librerías

### Matplotlib:

Es la biblioteca estándar para visualización de datos en Python. Ofrece una variedad de funciones para crear gráficos estáticos, gráficos interactivos, subplots y más. Los aspectos destacados son:

- I. **Estilos de gráficos:** Permite crear una amplia gama de gráficos, incluyendo gráficos de líneas, barras, dispersión, histogramas, diagramas de caja y más.
- II. **{Personalización:** Ofrece control completo sobre la apariencia de los gráficos, incluyendo colores, marcadores, leyendas, etiquetas de ejes y títulos.
- III. **Subplots y figuras multiples:** Permite crear múltiples gráficos dentro de la misma figura y personalizar su disposición.
- IV. **Interactividad:** A través de otras bibliotecas como mpld3 o Plotly, Matplotlib puede ser utilizada para crear gráficos interactivos y widgets en notebooks de Jupyter.

## II-C. Librerías

### Numpy:

Es la biblioteca fundamental para la computación científica en Python. Proporciona soporte para arrays multidimensionales (llamados ndarray), junto con una amplia colección de funciones matemáticas para operar en estos arrays. Algunos conceptos clave incluyen:

- I. **Arrays:** Son estructuras de datos fundamentales en NumPy, que permiten almacenar datos de manera eficiente y realizar operaciones vectorizadas.
- II. **Operaciones vectorizadas:** NumPy permite realizar operaciones rápidas en arrays completos sin necesidad de bucles explícitos, lo cual es crucial para la eficiencia en el procesamiento de datos.
- III. **Funciones matemáticas:** Proporciona funciones para operaciones matemáticas
- IV. básicas y avanzadas, como funciones trigonométricas, álgebra lineal, estadísticas y más.
- V. **Indexación y segmentación avanzadas:** Permite acceder y manipular datos dentro de arrays utilizando técnicas poderosas de indexación y segmentación.

## III. DESARROLLO

Aquí haremos presencia de los códigos de los respectivos de los ejercicios planteados en los cuadernos compartidos en github:

### 1. 1.1 Ejercicios del cuaderno de Pandas:

#### 3.1.1

```
#Ejercicio
...
Agrega las columnas siguientes: edad_mas_5, nombre_completo y genero
edad_mas_5viene de sumar 5 a la etiqueta edad
nombre_completo viene de sumar las etiquetas primer_nombre, _ y ultimo_nombre
genero es una serie de elementos 'F','F','M','M','M','F'.
...
import pandas as pd

data1 = {
    'primer_nombre': ['Amy', 'Amy', 'Jason', 'Nick', 'Stephen', 'Amy'],
    'ultimo_nombre': ['Jackson', 'J', 'Miller', 'Milner', 'L', 'J'],
    'edad': [42, 42, 36, 24, 24, 42]
}

#creo un dataframe de pandas a partir de los datos de data1
df = pd.DataFrame(data1, columns=['primer_nombre', 'ultimo_nombre', 'edad'])

df['edad_mas_5'] = df['edad'] + 5

df['nombre_completo'] = df['primer_nombre'] + ' ' + df['ultimo_nombre'].replace('J', '')

genero = ['F', 'F', 'M', 'M', 'M', 'F']
df['genero'] = genero

# Mostrar el DataFrame resultante
print(df)
```

	primer_nombre	ultimo_nombre	edad	edad_mas_5	nombre_completo	genero
0	Amy	Jackson	42	47	Amy Jackson	F
1	Amy	J	42	47	Amy	F
2	Jason	Miller	36	41	Jason Miller	M
3	Nick	Milner	24	29	Nick Milner	M
4	Stephen	L	24	29	Stephen L	M
5	Amy	J	42	47	Amy	F

3.1.2

```
import pandas as pd
import numpy as np

df = pd.DataFrame({
    'Nombre': ['jack', 'jane', 'jack', 'jane', 'jack', 'jane', 'jack', 'jane'],
    'Estado': ['SFO', 'SFO', 'NYK', 'CA', 'NYK', 'NYK', 'SFO', 'CA'],
    'Edad': np.random.uniform(24, 50, size=8)
})

#df.groupby(['Estado', 'Nombre']) agrupamos datos de variable estado y nombre
tabla_grupo = df.groupby(['Estado', 'Nombre'])['Edad'].mean().reset_index()
#mean() Esto calcula la media de las edades
#reset_index() : Esto convierte los índices de los grupos (Estado y Nombre) en columnas regulares
# y restablece el índice del DataFrame para que sea numérico, comenzando desde 0.

edad_media_estado = df.groupby('Estado')['Edad'].mean()

print("Edad media por Estado:")
print(edad_media_estado)
```

Suma por nombres:

	Estado	Genero	Edad	Salario
Nombre				
jack	SFO	NYK	SFO	ABCC
jane	SFO	CA	NYK	AABA

Edad máxima y salario máximo por nombre/estado:

	Edad	Salario
	max	max
Nombre		
jack	NYK	SFO
jane	CA	NYK

3.1.3

```
1 import pandas as pd
2 import numpy as np
3
4 df = pd.DataFrame({'Nombre': ['jack', 'jane', 'jack', 'jane', 'jack', 'jane', 'jack', 'jane'],
5                       'Estado': ['SFO', 'SFO', 'NYK', 'CA', 'NYK', 'NYK', 'SFO', 'CA'],
6                       'Genero': ['A', 'A', 'B', 'A', 'C', 'B', 'C', 'A'],
7                       'Edad': np.random.uniform(24, 50, size=8),
8                       'Salario': np.random.uniform(3000, 5000, size=8)})
9
10 suma_por_nombres = df.groupby('Nombre').sum()
11 #se está utilizando el método groupby de pandas para agrupar los datos del DataFrame df
12 #por la columna 'Nombre'. Después, se aplica el método sum() sobre los grupos resultantes.
13
14 print("Suma por nombres:")
15 print(suma_por_nombres)
16 print()
17 agregaciones = {
18     'Edad': ['max'],
19     'Salario': ['max']
20 }
21 maximos_por_nombre_estado = df.groupby(['Nombre', 'Estado']).agg(agregaciones)
22
23 print("Edad máxima y salario máximo por nombre/estado:")
24 print(maximos_por_nombre_estado)
```

Edad media por Estado:

Estado	
CA	45.423754
NYK	39.778021
SFO	37.896553

Name: Edad, dtype: float64

### 3.2 Ejercicios del cuaderno de Numpy:

#### 3.2.1

```
import numpy as np
#genero numeros aleatorios
np.random.seed(0)

x1 = np.random.randint(10, size=6)
x2 = np.random.randint(10, size=(5, 4))
x3 = np.random.randint(10, size=(2, 4, 5))
print("Atributos de x1:")
print("Dimensiones (ndim):", x1.ndim)
print("Forma (shape):", x1.shape)
print("Tamaño (size):", x1.size)
print("Tipo de datos (dtype):", x1.dtype)
print("Tamaño de cada elemento (itemsize):", x1.itemsize)
print("Tamaño total en bytes (nbytes):", x1.nbytes)
print()
print("Atributos de x2:")
print("Dimensiones (ndim):", x2.ndim)
print("Forma (shape):", x2.shape)
print("Tamaño (size):", x2.size)
print("Tipo de datos (dtype):", x2.dtype)
print("Tamaño de cada elemento (itemsize):", x2.itemsize)
print("Tamaño total en bytes (nbytes):", x2.nbytes)
print()
print("Atributos de x3:")
print("Dimensiones (ndim):", x3.ndim)
print("Forma (shape):", x3.shape)
print("Tamaño (size):", x3.size)
print("Tipo de datos (dtype):", x3.dtype)
print("Tamaño de cada elemento (itemsize):", x3.itemsize)
print("Tamaño total en bytes (nbytes):", x3.nbytes)
```

```
Atributos de x1:
Dimensiones (ndim): 1
Forma (shape): (6,)
Tamaño (size): 6
Tipo de datos (dtype): int64
Tamaño de cada elemento (itemsize): 8
Tamaño total en bytes (nbytes): 48
```

```
Atributos de x2:
Dimensiones (ndim): 2
Forma (shape): (5, 4)
Tamaño (size): 20
Tipo de datos (dtype): int64
Tamaño de cada elemento (itemsize): 8
Tamaño total en bytes (nbytes): 160
```

```
Atributos de x3:
Dimensiones (ndim): 3
Forma (shape): (2, 4, 5)
Tamaño (size): 40
Tipo de datos (dtype): int64
Tamaño de cada elemento (itemsize): 8
Tamaño total en bytes (nbytes): 320
```

#### 3.2.2

```
import numpy as np
#creo una matriz
matriz = np.array([[ 0,  1,  2,  3],
                   [ 4,  5,  6,  7],
                   [ 8,  9, 10, 11]])

print("Matriz original:")
print(matriz)
print()
submatriz = matriz[:, 1:3]

print("Submatriz extraida (primeras 2 filas y columnas 1 y 2):")
print(submatriz)
print()
submatriz[0, 0] = 99

print("Submatriz después de modificar submatriz[0, 0] a 99:")
print(submatriz)
print()
print("Matriz original después de modificar la submatriz:")
print(matriz)
```

```
Matriz original:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

```
Submatriz extraída (primeras 2 filas y columnas 1 y 2):
[[1 2]
 [5 6]]
```

```
Submatriz después de modificar submatriz[0, 0] a 99:
[[99  2]
 [ 5  6]]
```

```
Matriz original después de modificar la submatriz:
[[ 0 99  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

### 3.3 Ejercicios del cuaderno de matplotlib:

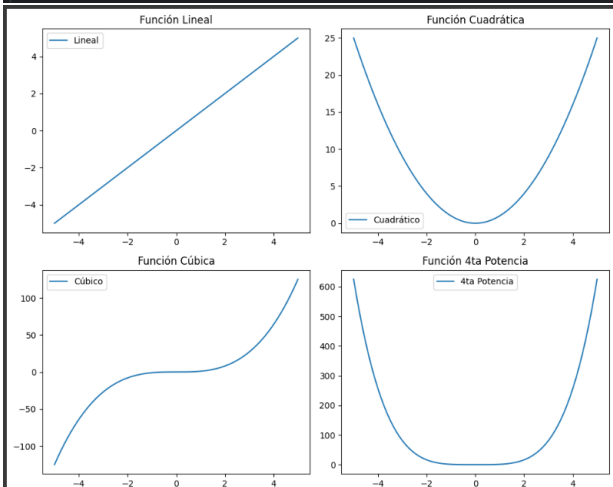
#### 3.3.1

```
import matplotlib.pyplot as plt
import numpy as np
# creo arreglo de valores numericos
x = np.linspace(start=-5, stop=5, num=150)
# creo una figura que contiene un arreglo de subplots
fig, axs = plt.subplots(2, 2, figsize=(10, 8))

axs[0, 0].plot(x, x, label='Lineal')
axs[0, 0].set_title('Función Lineal')
axs[0, 1].plot(x, x**2, label='Cuadrático')
axs[0, 1].set_title('Función Cuadrática')
axs[1, 0].plot(x, x**3, label='Cúbico')
axs[1, 0].set_title('Función Cúbica')
axs[1, 1].plot(x, x**4, label='4ta Potencia')
axs[1, 1].set_title('Función 4ta Potencia')

# Ajustar espacio ente subplots
plt.tight_layout()
for ax in axs.flat:
    ax.legend() # se muestran las leyendas

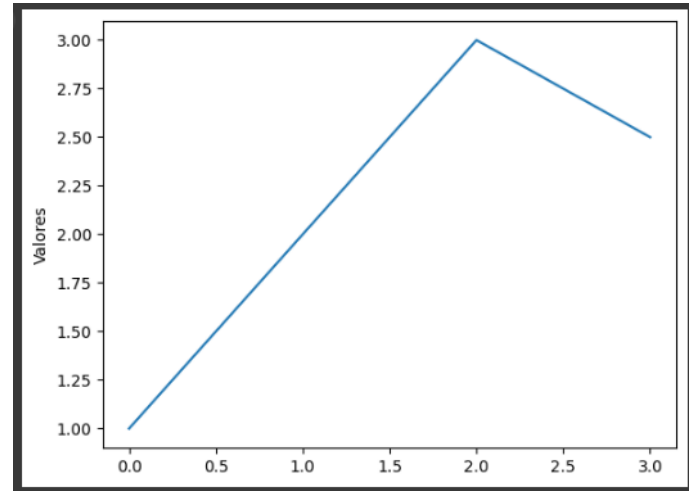
plt.show()
```



#### 3.3.2

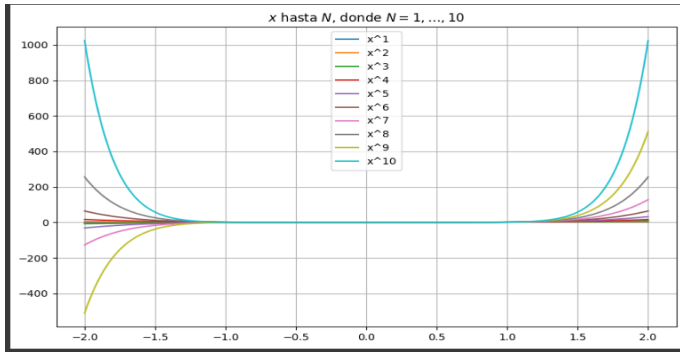
```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 2.5])
plt.ylabel('Valores')
plt.show()
```



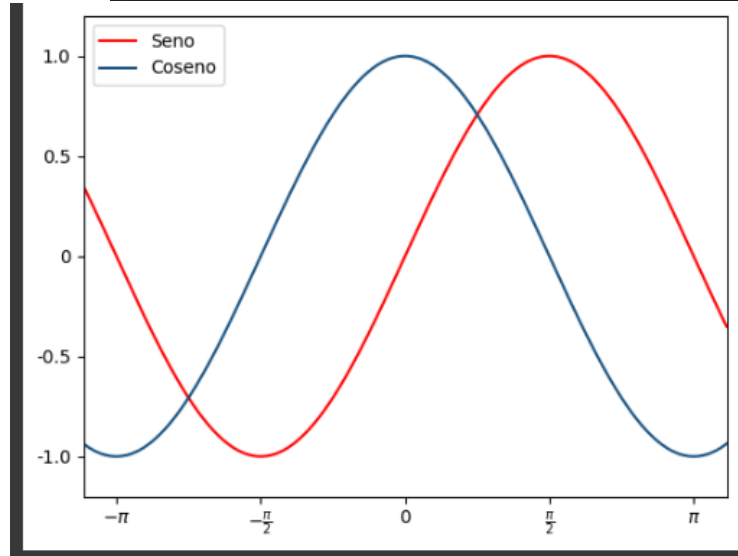
### 3.3.3

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(start=-2, stop=2, num=400)
fig, ax = plt.subplots(figsize=(10, 6))
for N in range(1, 11):
    #establesco "colores"
    ax.plot(x, x**N, label=f'x^{N}')
ax.set_title(r'$x$ hasta $N$, donde $N=1,\ldots,10$')
ax.legend()
plt.grid(True)
plt.show()
```



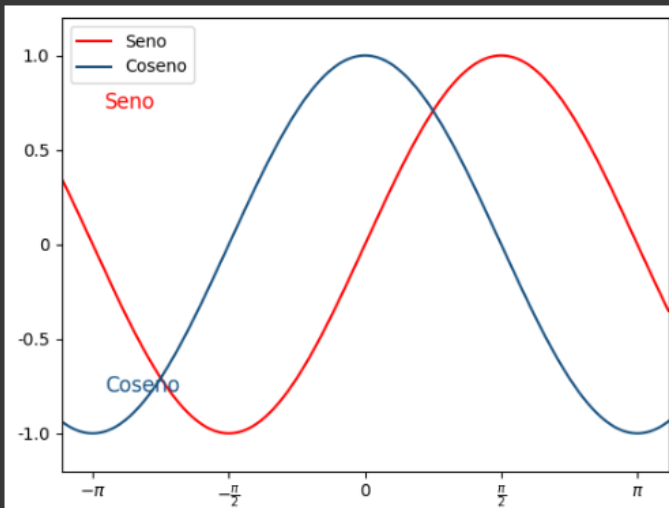
### 3.3.4

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-3.5, 3.5, 400)
seno = np.sin(x)
coseno = np.cos(x)
fig, ax = plt.subplots()
#ponemos "colores"
ax.plot(x, seno, color='red', label='Seno')
ax.plot(x, coseno, color='#165181', label='Coseno')
#limites de eje
ax.set_xlim(-3.5, 3.5)
ax.set_ylim(-1.2, 1.2)
#establesco ticks
ax.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.set_yticks(np.arange(-1, 1.1, 0.5))
ax.set_xticklabels([r'$-\pi$', r'$-\frac{\pi}{2}$', '0', r'$\frac{\pi}{2}$', r'$\pi$'])
ax.set_yticklabels(['-1.0', '-0.5', '0', '0.5', '1.0'])
ax.legend()
plt.show()
```



### 3.3.5

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-3.5, 3.5, 400)
seno = np.sin(x)
coseno = np.cos(x)
fig, ax = plt.subplots()
ax.plot(x, seno, color='red', label='Seno')
ax.plot(x, coseno, color='#165181', label='Coseno')
ax.set_xlim(-3.5, 3.5)
ax.set_ylim(-1.2, 1.2)
ax.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.set_yticks(np.arange(-1, 1.1, 0.5))
ax.set_xticklabels([' $-\pi$ ', ' $-\frac{\pi}{2}$ ', '0', ' $\frac{\pi}{2}$ ', ' $\pi$ '])
ax.set_yticklabels(['-1.0', '-0.5', '0', '0.5', '1.0'])
#agregamos leyenda
ax.legend()
ax.text(-3, 0.75, 'Seno', color='red', fontsize=12, ha='left', va='center')
ax.text(-3, -0.75, 'Coseno', color='#165181', fontsize=12, ha='left', va='center')
plt.show()
```



### 3.3.6

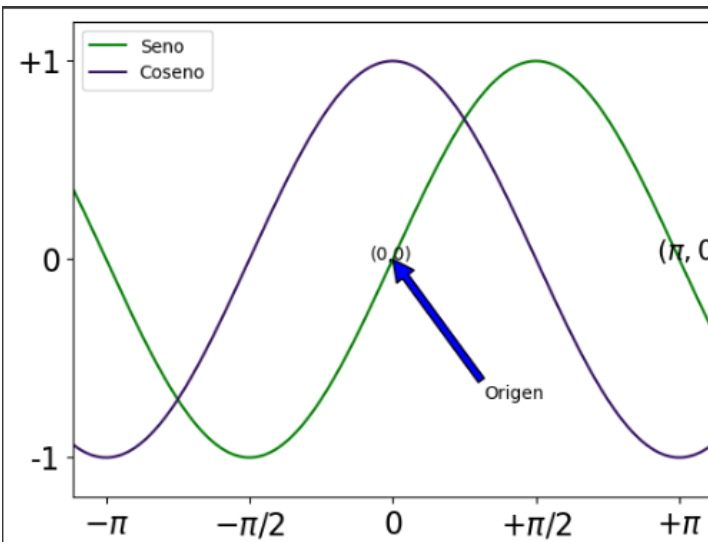
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-3.5, 3.5, 400)
seno = np.sin(x)
coseno = np.cos(x)

fig, ax = plt.subplots()
ax.plot(x, seno, color='green', label='Seno')
ax.plot(x, coseno, color='#341161', label='Coseno')
ax.set_xlim(-3.5, 3.5)
ax.set_ylim(-1.2, 1.2)

ax.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.set_yticks([-1, 0, 1])

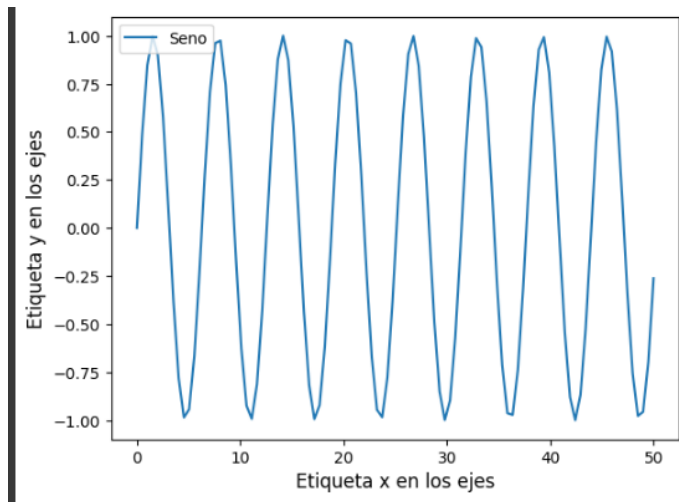
ax.set_xticklabels([' $-\pi$ ', ' $-\pi/2$ ', ' $0$ ', ' $\pi/2$ ', ' $\pi$ '], size=17)
ax.set_yticklabels(['-1', '0', '+1'], size=17)
ax.legend(loc='upper left')
ax.text(-0.25, 0, '(0,0)')
ax.text(np.pi - 0.25, 0, ' $(\pi, 0)$ ', size=15)
ax.annotate('Origen', xy=(0, 0), xytext=(1, -0.7), arrowprops=dict(facecolor='blue'))
plt.show()
```



### 3.3.7

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 50, 100)
y1 = np.sin(x)
y2 = np.cos(x)
fig, ax = plt.subplots()
ax.plot(x, y1, label='Seno')
ax.set_xlabel('Etiqueta x en los ejes', fontsize=12)
ax.set_ylabel('Etiqueta y en los ejes', fontsize=12)
ax.legend()
fig.tight_layout(pad=2.0)

# Mostrar el gráfico
plt.show()
```



## I. CONCLUSIONES

En conclusión, las bibliotecas NumPy, Pandas y Matplotlib se presentan como pilares fundamentales en el ecosistema de Python para el análisis de datos y la visualización.

NumPy proporciona la base eficiente para operaciones numéricas en arrays multidimensionales, esencial para el procesamiento de datos a gran escala. Por otro lado, Pandas simplifica significativamente la manipulación y el análisis de datos tabulares mediante sus estructuras de datos flexibles como DataFrames y Series, permitiendo operaciones complejas con facilidad.

Finalmente, Matplotlib destaca por su capacidad para crear una amplia variedad de gráficos estáticos e interactivos, ofreciendo control detallado sobre la apariencia visual de los datos.

La integración fluida entre estas bibliotecas facilita un flujo de trabajo coherente y eficiente desde la carga de datos hasta su análisis y visualización, convirtiéndolas en herramientas indispensables para cualquier profesional o investigador que trabaje con datos en Python.

## V. BIBLIOGRAFÍA

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.

The pandas development team. (2020). *pandas-dev/pandas: Pandas*. Zenodo. doi:10.5281/zenodo.3509134

The matplotlib development team. (2020). *matplotlib/matplotlib: Matplotlib*. Zenodo. doi:10.5281/zenodo.38932