Primer Tarea Explorando Librerías Básicas Python

Diego Alejandro Barragán Vargas
Ingeniero Electrónico y Magister en Ingeniería de la UDFJC

PRIMER ENTREGABLE

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
INGENIERO ELECTRÓNICO Y MAGISTER EN INGENIERÍA
BOGOTÁ D.C, COLOMBIA
FEBRERO 20 DE 2025

Primer Tarea Explorando Librerías Básicas en Python

Presentado por:
Diego Alejandro Barragán Vargas

PRIMER ENTREGABLE

Primer entregable del:
Machine Learning

Profesor encargado de la revisión:
Gerardo Muñoz

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
INGENIERO ELECTRÓNICO Y MAGISTER EN INGENIERÍA
BOGOTÁ D.C, COLOMBIA

FEBRERO 20 DE 2025

Índice general

1 MARCO CONCEPTUAL

En este documento se procederá a mostrar algunas de las librerías clásicas y más básicas de python con el objetivo de aprender y explorar más a fondo el campo del machine learning. Las librerías que se explorarán son:

- **Numpy:**Es una librerías especializada en el cálculo numérico y el análisis de datos, donde se manejan especialmente un gran número de datos. Esta librerías incorpora una clase de objetos denominada arrays que permite la recolección de datos de un mismo tipo en varias dimensiones y también funciones que son eficientes para su manipulación, esta librería permite un procesamiento de vectores y matrices de grandes dimensiones.

  Los arrays son estructuras de datos de un mismo tipo organizado en forma de tabla o cuadrícula con diferentes dimensiones. Estas dimensiones son conocidas como ejes. Para la creación de un array se utiliza la siguiente función de Numpy:

  ```
  np.array(lista): Permite la creación de una tupla o de una lista
  y devuelve la referencia a el. Es importante que los elementos de
  la lista o de la tupla deben ser del mismo tipo. A continuación
  genero un ejemplo de un array:

  #Array de una dimensión
  a1= np.array([1,2,3])
  print(a1)

  [1,2,3]

  #Array de dos dimensiones
  a2 = np.array([[1,2,3],[4,5,6]])
  print(a2)

  [[1,2,3]
   [4,5,6]]
  ```

Otras funciones importantes que existen para esta librería son:

- **np.empty(dimensiones):** rea y devuelve una referencia a un array vacío con las dimensiones especificadas en la tupla dimensiones.
- **np.zeros(dimensiones):**Crea y devuelve una referencia a un array con las dimensiones especificadas en la tupla dimensiones cuyos elementos son todos ceros.
- **np.ones(dimensiones):** Crea y devuelve una referencia a un array con las dimensiones especificadas en la tupla dimensiones cuyos elementos son todos unos.
- **np.full(dimensiones, valor):** Crea y devuelve una referencia a un array con las dimensiones especificadas en la tupla dimensiones cuyos elementos son todos valor.
- **np.identity(n):**Crea y devuelve una referencia a la matriz identidad de dimensión n.
- **np.arange(inicio, fin, salto):** Crea y devuelve una referencia a un array de una dimensión cuyos elementos son la secuencia desde inicio hasta fin tomando valores cada salto.
- **np.linspace(inicio, fin, n):** Crea y devuelve una referencia a un array de una dimensión cuyos elementos son la secuencia de n valores equidistantes desde inicio hasta fin.
- **np.random.random(dimensiones):** Crea y devuelve una referencia a un array con las dimensiones especificadas en la tupla dimensiones cuyos elementos son aleatorios. [1]

- **Matplotlib:** Es una librería open source que permite crear visualizaciones de datos, la desarrolló el neurobiólogo Jhon Hunter en 2002, el objetivo del profesional de la salud era visualizar las señales eléctricas del cerebro de personas epilépticas, con lo que se propuso desarrollar unas funciones similares a las creadas de manera gráficas en Matlab. [2]

- **Pandas:** Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos. Las principales características de pandas son:
  - Define nuevas estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
  - Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL.
  - Permite acceder a los datos mediante índices o nombres para filas y columnas.
  - Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
  - Permite trabajar con series temporales.

- Realiza todas estas operaciones de manera eficiente, aunque se basa en C y C++. [3]

- **Polars:** Es una librería que proporciona bibliotecas de dataframe ultra-rápidos que permite lo siguiente:
  - Utilizar todos los núcleos disponibles en las máquinas
  - optimizar las consultas para reducir las asignaciones de trabajo/memoria innecesarias.
  - Maneja conjuntos de datos muy grandes.
  - Es una API que es robusta y consistente.
  - Se adhiere a un sistema estricto donde los tipos de datos deben conocerse antes de ejecutar la consulta.

- **hvPlot:** Es una librería de alto nivel para la exploración y la generación de tablas y gráficas. Este tiene las siguientes características:
  - Admite una amplia gama de fuente de datos, incluidaa pandas, polars , Xarray, Dask, Streamz, Intake, GeoPandas y NetworkX.
  - Admite los backends gráficos Bokeh, Matplotlib y Plotly.
  - Expone las poderosas herramientas del ecosistema denominado Holo Viz.
  - HvPlot se puede utilizar también para una exploración de datos, desarrollo de informes y aplicaciones de datos.
  - Se comparte una pequeña arquitectura de la conexión de hvplot con otras librerías.

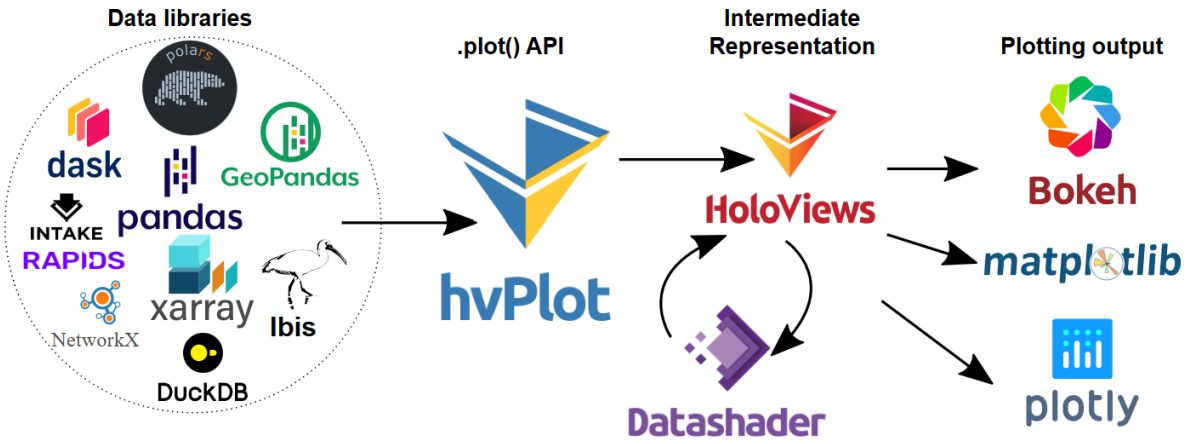

Figura 1.1: Arquitectura de hvplot

1.0.1. Pandas vs Polars

Se tiene lo siguiente:

Características	**Pandas**	**Polars**
Backend	Basado en NumPy (en memoria).	Basado en Apache Arrow (formato columnar eficiente).
Lenguaje	Python (con partes críticas en C/Cython).	Escrito en Rust, con bindings para Python y Node.js.
Ejecución	Eager (ejecución inmediata).	Soporta modo eager y lazy (optimización de consultas).
Manejo de Datos	DataFrames en memoria con índices.	DataFrames sin índices, optimizados para operaciones vectorizadas.
Paralelización	Limitada (depende de NumPy/operaciones por hilo único).	Paralelización nativa (multihilos) y SIMD (optimizaciones a nivel de CPU).
API	Amplia y madura, con sintaxis estilo Python.	Sintaxis funcional y encadenamiento de métodos (pipe).
Integración	Compatible con NumPy, scikit-learn, Matplotlib, etc.	Compatible con Apache Arrow, Pandas, y herramientas modernas (DuckDB, etc.).

Ventajas y Desventajas

- **Ventajas de Pandas**
  - Comunidad enorme y documentación extensa.
  - Madurez (lanzado en 2008).
  - Integración perfecta con el ecosistema Python (ML, visualización).
  - Sintaxis intuitiva para usuarios nuevos.
- **Desventajas de Pandas**
  - Rendimiento limitado en datos grandes (¿1 GB).
  - Consumo alto de memoria.
  - Operaciones complejas requieren trabajo adicional (ej. apply).
- **Ventajas de Polars**
  - Rendimiento 5-10x más rápido que Pandas en operaciones complejas.

- Manejo eficiente de memoria (Apache Arrow).
- Ejecución lazy para optimizar consultas.
- Escalabilidad en datasets grandes (TB-level).

- **Desventajas de Polars**
  - Curva de aprendizaje más pronunciada.
  - Menor cantidad de recursos y ejemplos.
  - Integración limitada con librerías clásicas (ej. scikit-learn).

Comparación de Sintaxis

Ejemplo: Agrupación y Media

```
# Pandas
df.groupby('categoria')['valor'].mean().reset_index()

# Polars
df.group_by('categoria').agg(pl.col('valor').mean())
```

Ejemplo: Lectura y Filtrado

```
# Pandas
df = pd.read_csv('datos.csv')
df_filtrado = df[df['edad'] > 30]

# Polars
df = pl.read_csv('datos.csv')
df_filtrado = df.filter(pl.col('edad') > 30)
```

1.0.2. Matplotlib vs Hvplot

Se tiene lo siguiente:

Características	**Pandas**	**Polars**
Backend	Basado en Python (con partes críticas en C/C++).	Basado en Bokeh (renderizado interactivo).
Tipo de Gráficos	Estáticos (SVG, PNG, PDF).	Interactivos (HTML, notebooks) y estáticos.
Sintaxis	Imperativa (orientada a objetos o estilo MATLAB).	Declarativa (integración con pandas y xarray).
Interactividad	Limitada (requiere widgets o herramientas externas).	Nativa (zoom, pan, tooltips, widgets dinámicos).
Integración	Compatible con NumPy, pandas, Jupyter.	Integra con pandas, xarray, Dask, Panel (para dashboards).
Personalización	Altamente personalizable (ej. ejes, leyendas, estilos).	Personalización más sencilla, enfocada en interactividad.
Rendimiento	Optimizado para datasets pequeños/medianos.	Escalable para datasets grandes (usando Dask o Datashader).
API	Compleja (múltiples capas: pyplot, objetos).	Unificada y concisa (similar a pandas .plot).

1.0.3. Ventajas y Desventajas

- **Ventajas de Matplotlib**
  - Estándar de facto en visualización científica.
  - Control total sobre cada elemento del gráfico (ej. fuentes, colores).
  - Soporte para múltiples formatos de exportación (PDF, SVG, PNG).
  - Comunidad masiva y documentación extensa.
- **Desventajas de Matplotlib**
  - Sintaxis verbosa para gráficos complejos.
  - Interactividad limitada sin librerías adicionales (ej. mpld3).
  - Curva de aprendizaje pronunciada para personalizaciones avanzadas.
- **Ventajas de Hvplot**
  - Interactividad nativa en notebooks y navegadores.

- Sintaxis simplificada (ej. df.hvplot.line(x='col')).
- Integración con el ecosistema HoloViz (Panel, Datashader).
- Ideal para exploración rápida de datos.

- **Desventajas de Hvplot**
  - Menor flexibilidad en estilos y detalles visuales.
  - Dependencia de Bokeh y otras librerías (HoloViews, Param).
  - Menor soporte para gráficos no convencionales (ej. diagramas de radar).

1.0.4. Comparación de Sintaxis

Ejemplo: Gráficos de Líneas

```
# Matplotlib
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.plot(df['x'], df['y'], label='Serie', color='blue', linestyle='--')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')
plt.title('Gráfico de Líneas')
plt.legend()
plt.show()

# hvPlot
import hvplot.pandas
df.hvplot.line(
    x='x', y='y', title='Gráfico de Líneas',
    color='blue', linestyle='dashed', width=500, height=400
)
```

Ejemplo: Histograma

```
# Matplotlib
plt.hist(df['valores'], bins=30, edgecolor='black')
plt.title('Histograma')

# hvPlot
df.hvplot.hist(y='valores', bins=30, title='Histograma')
```

Bibliografía

[1] A. Conalf, "La librerías numpy." [Online]. Available: https://aprendeconalf.es/docencia/python/manual/numpy/

[2] DataScicentest, "Matplotlib: todo lo que tienes que saber sobre la librería python de dataviz." [Online]. Available: https://datascientest.com/es/todo-sobre-matplotlib

[3] A. Conalf, "La librería pandas." [Online]. Available: https://aprendeconalf.es/docencia/python/manual/pandas/