Domando Datos Reales con Pandas

Limpieza, Análisis y Series Temporales

Curso de IA Aplicada al Agro

11 de enero de 2026

Índice general

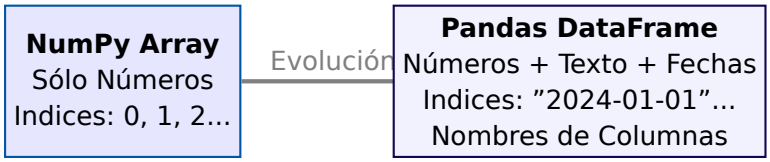
Introducción: ¿Por qué NumPy no es suficiente?

En la Semana 02 aprendimos que NumPy es rapidísimo para matrices numéricas. Pero el mundo real no son solo números; son etiquetas, fechas y datos sucios.

Imagina una tabla de Excel de una estación meteorológica. Tiene:

- *Fechas* (Time Series).
- *Texto* (Nombres de sensores, ubicación).
- *Huecos* (Celdas vacías porque se fue la luz).

Para esto existe **Pandas**. Es una librería construida sobre NumPy que añade etiquetas y manejo de errores.



1 Capítulo I: El DataFrame

El objeto principal de Pandas es el ‘DataFrame‘. Piensa en él como una hoja de cálculo programable.

1.1 Creación y Carga de Datos

Rara vez creamos DataFrames a mano; usualmente los cargamos desde archivos.

Listing 1: Cargar un CSV de Cosecha

```python
import pandas as pd

# Leer archivo separado por comas
# index_col=0 usa la primera columna como indice (ID)
df = pd.read_csv('cosecha_2024.csv', index_col=0)

# Ver las primeras 5 filas
print(df.head())

# Ver informacion tecnica (tipos de datos, memoria)
print(df.info())
```

1.2 Selección de Datos: loc vs iloc

Esta es la fuente de confusión #1.

- `.loc[]`: Busca por Etiqueta (Nombre).
- `.iloc[]`: Busca por Índice Numérico (Posición).

```python
# Seleccionar columna 'Humedad' de la fila 'Sensor_A'
dato = df.loc['Sensor_A', 'Humedad']

# Seleccionar fila 0, columna 2
dato_raw = df.iloc[0, 2]
```

2 Capítulo II: Limpieza de Datos (Data Cleaning)

En el agro, los datos son ruidosos. Un sensor de humedad puede desconectarse y enviar valores vacíos o basura.

El Peligro del NaN

En computación científica, **NaN** significa *Not a Number*. Si intentas sumar `5 + NaN`, el resultado es `NaN`. Un solo dato faltante puede arruinar todo el cálculo de promedio anual.

2.1 Detectar Datos Faltantes

Pandas nos permite ver dónde están los huecos.

```
# Crear un reporte de datos faltantes por columna
print(df.isna().sum())

# Salida ejemplo:
# Temperatura      0
# Humedad         15  <-- 15 lecturas perdidas!
# pH               2
```

2.2 Estrategias de Corrección

Como ingenieros, debemos decidir qué hacer con esos huecos.
1. Eliminar (Drop): Si hay pocos datos malos, bórralos. 2. Imputar (Fill): Rellenar con un promedio o el valor anterior.

Listing 2: Sanitización de Sensores

```
# ESTRATEGIA 1: Borrar filas con huecos
df_limpio = df.dropna()

# ESTRATEGIA 2: Rellenar (Ideal para series de tiempo)
# 'ffill' (Forward Fill) toma el ultimo valor valido y lo repite.
# Si a las 10:00 hizo 25C y a las 11:00 es NaN, asume 25C.
df_relleno = df.fillna(method='ffill')

# ESTRATEGIA 3: Rellenar con promedio
promedio = df['Humedad'].mean()
df['Humedad'] = df['Humedad'].fillna(promedio)
```

3 Capítulo III: Filtrado y Estadística

Podemos hacer preguntas complejas a los datos sin usar bucles.

Caso de Uso: Alerta de Hongos

Los hongos proliferan con humedad alta ($> 80\%$) y temperaturas moderadas ($> 20°C$). Filtremos esos días.

Listing 3: Filtrado Multicriterio

```python
# Filtro booleano (Igual que NumPy, pero con nombres)
riesgo_hongos = df[ (df['Humedad'] > 80) & (df['Temperatura'] > 20) ]

print("Dias con riesgo de hongos:")
print(riesgo_hongos)
```

3.1 Agrupación (GroupBy)

El poder de las tablas dinámicas de Excel, pero en código.

```python
# Supongamos que tenemos columna 'Zona' (Norte, Sur, Este)
# Queremos el promedio de rendimiento por zona.

resumen = df.groupby('Zona')['Rendimiento'].mean()
print(resumen)
```

Reto Semanal: El Dataset Sucio

Se te entregará un archivo `clima_corrupto.csv`.

1. Cárgalo en un DataFrame.
2. Encuentra cuántas filas tienen `NaN`.
3. Rellena las temperaturas faltantes usando interpolación lineal (‘.interpolate()‘).
4. Calcula la temperatura promedio mensual.
5. Exporta el resultado a `clima_limpio.csv`.

```python
# Pista para exportar
df.to_csv('clima_limpio.csv')
```