guayerd

Fundamentos IA

Análisis con Python

Clase 6

En colaboración con

IBM SkillsBuild





- ¿Qué recuerdan de la clase anterior?
- ¿Qué esperan aprender?
- ¿Tienen alguna pregunta?

En colaboración con

IBM SkillsBuild

Contenidos Por temas

Copilot Chat y promptsDemo asincrónica

06

• Limpieza y transformación

• Estadística aplicada

08

Visualización



En colaboración con IBM **SkillsBuild**

Objetivos de la clase



- Pandas
- Lectura de archivos
- Estructuras principales
- Inspección y limpieza

En colaboración con

IBM SkillsBuild

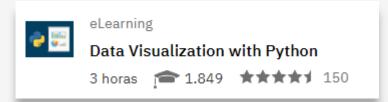
Análisis con Python

Limpieza y transformación

guayerd

En colaboración con IBM **SkillsBuild**

Plataforma Skill Build: Python





eLearning

Utilizar la IA generativa para el desarrollo de software



En colaboración con IBM **SkillsBuild**

Limpieza y transformación de datos

Etapa 3 del ciclo de vida del dato

Proceso técnico para preparar datos antes del análisis.

- Elimina errores, inconsistencias y valores irrelevantes
- Estandariza formatos y estructuras
- Mejora la calidad y utilidad de los datos

Estandarizar = criterios



Formatos comunes

- CSV: datos tabulares separados por comas
- **JSON**: objetos anidados y flexibles
- Excel: hojas de cálculo
- Bases de datos: estructuras relacionales



Pandas integra el trabajo con todos estos formatos



¿Qué es?

Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos.



Principales características

- → Estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
- → Permite leer y escribir fácilmente archivos 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 muy eficiente.



Comandos

- Instalación: pip install pandas
- Uso: import pandas as pd



Estructuras principales

Series (.s)

- Arreglo unidimensional
- Compuesta por índices y valores
- Similar a una columna

DataFrame (.df)

- Tabla bidimensional
- Colección de Series alineadas por índice
- Base para análisis y transformación



Lectura de archivos

CSV

- df_csv = pd.read_csv("nombre_archivo.csv")
- Útil: encoding='utf-8', sep=';'

JSON

- df_json = pd.read_json("nombre_archivo.json")
- Para listas: orient='records'

Excel

- df_xls = pd.read_excel("nombre_archivo.xlsx")
- Especificar hoja: sheet_name="nombre_hoja"

Siempre devuelve un DataFrame listo para trabajar

Inspección inicial

Aspecto	Comando	Descripción	
Estructura	df.shape	Dimensiones del conjunto	
Tipos	df.dtypes	Variables numéricas y categóricas	
Completitud	df.isnull().sum()	Valores faltantes	
Muestra	df.head()	Primeros registros	
Resumen	df.info()	Información general	

Desafíos comunes en datos

- Valores faltantes: celdas vacías o NaN
- **Duplicados:** registros repetidos exactos
- Inconsistencias: formatos diferentes para mismo dato
- Valores atípicos: datos extremos que sesgan análisis
- **Tipos incorrectos:** números como texto, fechas mal formateadas



Qué son las Series en Pandas ?

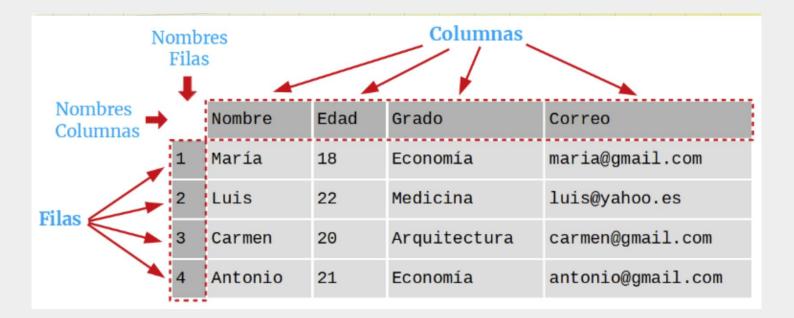
- → Estructuras similares a los arrays de una dimensión.
- → Sus elementos tienen que ser del mismo tipo, y su tamaño es inmutable, es decir, no se puede cambiar, aunque sí su contenido.
- → Dispone de un índice que asocia un nombre a cada elemento de la serie, a través de la cuál se accede al elemento.

```
Crear series
>>> import pandas as pd
>>> s = pd.Series(['Matemáticas', 'Historia', 'Economía', 'Programación', 'Inglés'], dtype='string')
>>> print(s)
     Matemáticas
       Historia
      Economía
                                                                                      A partir de
    Programación
                                                                                        una lista
         Inglés
dtype: string
                              >>> import pandas as pd
                              >>> s = pd.Series({'Matemáticas': 6.0, 'Economía': 4.5, 'Programación': 8.5})
                              >>> print(s)
                              Matemáticas
                                            6.0
    A partir de un
                                            4.5
                              Economía
                              Programación 8.5
      diccionario
                              dtype: float64
```

DataFrame en Pandas

- → Define un conjunto de datos estructurado en forma de tabla donde cada columna es un objeto de tipo Series, y las filas son registros que pueden contener datos de distintos tipos.
- → Un DataFrame contiene dos índices, uno para las filas y otro para las columnas.
- → Se puede acceder a sus elementos mediante los nombres de las filas y las columnas.

Ejemplo:



Crear DataFrames

```
>>> import pandas as pd
>>> df = pd.DataFrame([['María', 18], ['Luis', 22], ['Carmen', 20]], columns=['Nombre', 'Edad'])
>>> print(df)
   Nombre Edad
0 María 18
1 Luis 22
2 Carmen 20
```



A partir de una lista

A partir de una lista de diccionarios



```
>>> import pandas as pd
>>> df = pd.DataFrame([{'Nombre':'María', 'Edad':18}, {'Nombre':'Luis', 'Edad':22}, {'Nombre':'Carmen'}])
>>> print(df)
0 María 18.0
```

Luis 22.0

Crear DataFrames a partir de un array:

Crear DataFrames a partir de un archivo .csv o Excel

```
>>> df = pd.read_csv('colesterol.csv', sep=';', decimal=',')
>>> print(df.head())
                           nombre edad sexo
                                                      altura colesterol
                                               peso
       José Luis Martínez Izquierdo
                                              85.0
                                                    1.79
                                                                  182.0
0
                    Rosa Díaz Díaz
                                    32
                                              65.0
                                                     1.73
                                                                  232.0
             Javier García Sánchez
                                    24
                                               NaN
                                                     1.81
                                                                  191.0
               Carmen López Pinzón
                                    35
                                              65.0
                                                    1.70
                                                                  200.0
              Marisa López Collado
                                    46
                                               51.0
                                                      1.58
                                                                  148.0
```

- → read_csv(fichero.csv, sep=separador, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal)
- read_excel(fichero.xlsx, sheet_name=hoja, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal)

¿Qué harías en cada situación?



- Dataset con 10% de valores faltantes distribuidos aleatoriamente
- Registros de clientes con emails duplicados pero datos diferentes
- Precios con valores negativos en sistema de inventario
- Fechas en formatos: "2024-01-15", "15/01/2024", "Jan 15, 2024"

guayerd

En colaboración con

Tratamiento de valores faltantes

Estrategia	Comando(s)	Descripción	
Detección	df.isnull(), df.isna()	Identificar valores faltantes	
Eliminación	df.dropna()	Eliminar registros con pocos casos	
Valor constante	df.fillna(0)	Rellenar con cero o texto	
Valor estadístico	df.fillna(df['columna'].median())	Rellenar con promedio o mediana	

Eliminación de duplicados

Estrategia	Comando(s)	Descripción	
Detección	df.duplicated()	Marca filas duplicadas	
Eliminación completa	inación completa df.drop_duplicates() Remueve todos los duplic		
Por columnas específicas df.drop_duplicates(subset=['col1', 'col2'])		Detecta duplicados según columnas elegidas	
Conservar primera/última Parámetro keep Ma		Mantiene el primer o último registro	

Inconsistencias de formato

Tipo de dato	Comando(s) / Acción	Descripción
Texto	.str.lower(), .str.strip()	Normalizar mayúsculas/minúsculas y espacios
Fechas	pd.to_datetime()	Convertir a formato fecha uniforme
Números	pd.to_numeric()	Ajustar separadores decimales
Categorías	_	Estandarizar variaciones del mismo valor

Manejo de valores atípicos

Valores extremos que se alejan significativamente del resto de los datos.

- **Detección visual:** boxplots muestran valores extremos
- Filtrado por rango: remover valores fuera de límites lógicos
- Criterio de dominio: usar conocimiento del área
- Verificación manual: confirmar si son errores o valores reales

Tipos de datos incorrectos

Estrategia	Comando(s)	Descripción	
Verificar tipos	df.dtypes Identificar tipos actuale		
Conversión manual df.astype()		Cambiar tipo de columna	
Fechas pd.to_datetime()		Convertir a formato fecha	
Numéricos	pd.to_numeric()	Forzar conversión numérica	

Transformaciones básicas

Operación	Comando(s)	Descripción
Filtrado	df[condición]	Seleccionar subconjuntos específicos
Agrupación	df.groupby()	Calcular estadísticas por categorías
Ordenamiento	df.sort_values()	Organizar registros por columnas
Selección	df[['col1', 'col2']]	Elegir columnas específicas

Café de barrio



- 1. Calcular correlación entre temperatura y ventas
- 2. Identificar el mes con mejor retorno publicitario
- 3. Analizar relación personal vs satisfacción cliente
- 4. Proponer estrategia basada en datos

Mes	Ventas (\$)	Temp (°C)	Publicidad (\$)	Personal	Satisfacción
Ene	15,000	18	800	4	4.2
Feb	22,000	25	1,200	5	4.5
Mar	18,000	22	900	4	4.1
Abr	28,000	28	1,500	6	4.8
May	25,000	30	1,300	5	4.6

Proyecto Tienda Aurelion

- **Documentación:** notebook Markdown
- Desarrollo técnico: programa Python
- Visualización de datos: dashboard en Power BI
- Presentación oral: problema, solución y hallazgos





En colaboración con IBM **SkillsBuild**

Limpieza de datos

Trabajo en equipo



- 1. Usar **Copilot para analizar problemas** con dataset
- 2. Limpiar la base de datos
- 3. Documentar con Copilot cada paso aplicado



Retro ¿Cómo nos vamos?

- ¿Qué fue lo más útil de la clase?
- ¿Qué parte te costó más?
- ¿Qué te gustaría repasar o reforzar?

En colaboración con

IBM SkillsBuild