OBTENCIÓN DE DATOS DESDE ARCHIVOS

M3: OBTENCIÓN Y PREPARACIÓN DE DATOS

|AE3: APLICAR TÉCNICAS DE EXTRACCIÓN DE DATOS DESDE DISTINTAS FUENTES UTILIZANDO LIBRERÍAS UTILITARIAS DE PYTHON PARA SU POSTERIOR USO.

alkemy

Introducción👩‍💻

En el análisis de datos, la obtención y manipulación eficiente de la información es un paso fundamental en cualquier flujo de trabajo de ciencia de datos, ingeniería de datos y análisis empresarial. La calidad de los datos impacta directamente en la precisión de los resultados obtenidos en modelos estadísticos, visualización de datos y aprendizaje automático. Por ello, contar con herramientas que permitan extraer, limpiar y estructurar datos de diversas fuentes de manera eficiente es crucial para cualquier profesional del área.

Python se ha convertido en un lenguaje estándar para la manipulación de datos, y una de sus bibliotecas más potentes es **Pandas**, la cual ofrece métodos sencillos y eficientes para la lectura y escritura de archivos en múltiples formatos. En este manual, exploraremos cómo utilizar Pandas para obtener datos desde archivos **CSV** y **Excel**, así como extraer tablas directamente desde páginas web.

El propósito de este manual es proporcionar una comprensión clara sobre cómo trabajar con datos almacenados en archivos externos, cómo estructurarlos adecuadamente en un **DataFrame**, y cómo exportarlos para su posterior procesamiento y análisis. Se incluirán ejemplos prácticos y buenas prácticas para maximizar la eficiencia en la manipulación de datos en Pandas.

Aprendizaje esperado

Cuando finalices la lección serás capaz de:

1. Comprender la importancia de la obtención de datos estructurados en procesos de análisis y modelado.
2. Leer y escribir archivos CSV de manera eficiente utilizando Pandas, entendiendo sus diversas opciones y configuraciones.
3. Leer y escribir archivos Excel mediante la integración de librerías como xlrd y openpyxl.
4. Extraer tablas de páginas web y transformarlas en estructuras de datos utilizables en Python.
5. Aplicar buenas prácticas en la carga y exportación de datos, minimizando errores de formato y optimizando el rendimiento en la manipulación de archivos grandes.
6. Automatizar la obtención y procesamiento de datos, reduciendo la dependencia de procesos manuales en la estructuración de información.

OBTENCIÓN DE DATOS DESDE ARCHIVOS CSV

1. ARCHIVOS CSV

Los archivos CSV (Comma-Separated Values) son ampliamente utilizados en la ciencia de datos y en el almacenamiento de datos estructurados debido a su simplicidad y compatibilidad con diversas plataformas. Estos archivos contienen información organizada en filas y columnas, separadas por comas, punto y coma u otros delimitadores.

Pandas proporciona una serie de herramientas que facilitan la lectura y escritura de archivos CSV, permitiendo a los usuarios cargar, analizar y modificar grandes volúmenes de datos con facilidad. Además, Pandas permite manejar diferentes codificaciones, delimitadores y formatos de archivo sin comprometer la integridad de los datos.

2. LEYENDO UN ARCHIVO CSV

El método read_csv() de Pandas es la forma más eficiente de leer archivos CSV y convertirlos en un DataFrame, la estructura de datos tabular de Pandas.

Link al archivo: datos.csv

```
import pandas as pd

# Cargar un archivo CSV
df = pd.read_csv('datos.csv')
print(df.head())  # Muestra las primeras filas
```

En ocasiones, los archivos CSV utilizan delimitadores distintos a la coma. Para estos casos, se puede especificar el delimitador correcto mediante el parámetro sep:

```
# Leer un CSV con punto y coma como separador
df = pd.read_csv('datos.csv', sep=';')
```

Otras opciones útiles incluyen:

- header=None: Indica que el archivo no tiene encabezados, lo que permite asignar nombres de columnas manualmente.
- names=['Col1', 'Col2']: Permite asignar nombres personalizados a las columnas.
- na_values=['?', 'N/A']: Especifica valores que deben considerarse como nulos.

```
# Leer un archivo CSV y asignar nombres de columnas
df = pd.read_csv('datos.csv', names=['Nombre', 'Edad', 'Ciudad'])
```

3. ESCRIBIENDO UN ARCHIVO CSV

Para guardar un DataFrame en un archivo CSV, se utiliza el método to_csv().

```
# Guardar un DataFrame en un archivo CSV
df.to_csv('nuevo_archivo.csv', index=False)
```

Es posible personalizar la exportación con opciones como:

- sep=';': Define un delimitador diferente.
- encoding='utf-8': Especifica la codificación del archivo para evitar problemas con caracteres especiales.
- header=False: Omite los nombres de las columnas.

```
# Guardar un archivo CSV con separador personalizado
df.to_csv('nuevo_archivo.csv', sep=';', encoding='utf-8', index=False)
```

ARCHIVOS EXCEL

Los archivos Excel son ampliamente utilizados en análisis de datos, finanzas y reportes empresariales debido a su capacidad de organizar información en múltiples hojas de cálculo. Pandas permite leer y escribir archivos Excel con gran facilidad, integrando librerías externas para manejar distintos formatos.

4. LIBRERÍA XLRD

xlrd fue tradicionalmente utilizada para leer archivos Excel en Pandas. Sin embargo, debido a cambios recientes en Pandas, ahora se recomienda usar openpyxl para manejar archivos .xlsx modernos.

Para garantizar compatibilidad con archivos Excel, se recomienda instalar openpyxl si aún no está disponible:

```
pip install openpyxl
```

5. LEYENDO UN ARCHIVO EXCEL

Para cargar datos desde un archivo Excel en un DataFrame, se usa el método read_excel().

Link al archivo: datos.xlsx

```
# Leer un archivo Excel
df = pd.read_excel('datos.xlsx', engine='openpyxl')
```

Si el archivo contiene varias hojas, se puede especificar la hoja deseada:

```
# Leer una hoja específica
df = pd.read_excel('datos.xlsx', sheet_name='Ventas', engine='openpyxl')
```

6. ESCRIBIENDO UN ARCHIVO EXCEL

Para exportar datos en formato Excel, se usa to_excel().

```
# Guardar un DataFrame en un archivo Excel
df.to_excel('nuevo_archivo.xlsx', index=False, engine='openpyxl')
```

Para escribir en múltiples hojas:

```
with pd.ExcelWriter('multiples_hojas.xlsx', engine='openpyxl') as writer:
    df.to_excel(writer, sheet_name='Hoja1')
    df.to_excel(writer, sheet_name='Hoja2')
```

7. LEER TABLAS WEB CON PANDAS

El método read_html() de Pandas permite extraer datos estructurados de sitios web que contienen tablas en formato HTML.

```
# Leer una tabla desde Wikipedia
url = 'https://es.wikipedia.org/wiki/Lista_de_pa%C3%ADses_por_PIB_(nominal)'
df_list = pd.read_html(url)

# Mostrar la primera tabla encontrada
df = df_list[0]
print(df.head())
```

Cierre ✍️

La correcta obtención y manipulación de datos es una habilidad fundamental en el análisis de datos, la ciencia de datos y la inteligencia de negocios. A lo largo de este manual, se han explorado diversas técnicas para leer y escribir archivos en formatos CSV y Excel, además de la extracción de información desde páginas web. Estas herramientas permiten a los profesionales estructurar, limpiar y transformar datos de diversas fuentes, facilitando su análisis y optimización en entornos de programación.

El dominio de la carga y exportación de datos con Pandas proporciona una base sólida para desarrollar flujos de trabajo eficientes, minimizar errores en el procesamiento de información y automatizar tareas repetitivas. Además, la integración de Pandas con otras bibliotecas de Python permite ampliar su funcionalidad, aplicándola a contextos más avanzados como bases de datos, procesamiento en la nube y aprendizaje automático.

Referencias📚

1. The pandas development team. (n.d.). Pandas documentation. https://pandas.pydata.org/docs/
2. McKinney, W. (2017). Python for data analysis: Data wrangling with pandas, NumPy, and IPython (2nd ed.). O'Reilly Media.
3. Müller, A. C., & Guido, S. (2017). Introduction to machine learning with Python: A guide for data scientists. O'Reilly Media.
4. Molin, S. (2019). Hands-on data analysis with pandas: Efficiently perform data collection, wrangling, analysis, and visualization using Python. Packt Publishing.

¡Muchas gracias!

Nos vemos en la próxima lección