Obtención y Preparación

de Datos

Sesión 2

Pandas: La Herramienta Esencial para Análisis de Datos en Python

- Pandas es una de las librerías más populares y poderosas de Python para la manipulación y análisis de datos. Desarrollada por Wes McKinney en 2008, se ha convertido en una herramienta esencial para científicos de datos, analistas e ingenieros debido a su facilidad de uso y capacidad para manejar grandes volúmenes de datos de manera eficiente.

- Esta librería de código abierto está construida sobre NumPy y proporciona estructuras de datos flexibles que permiten trabajar con datos tabulares, estructurados y series temporales. Su nombre proviene de "Panel Data", un término econométrico que hace referencia a conjuntos de datos multidimensionales.

Características Principales

✅ Estructuras Optimizadas

Manejo eficiente de datos tabulares en estructuras como DataFrame y Series, diseñadas específicamente para el análisis de datos.

✅ Compatibilidad

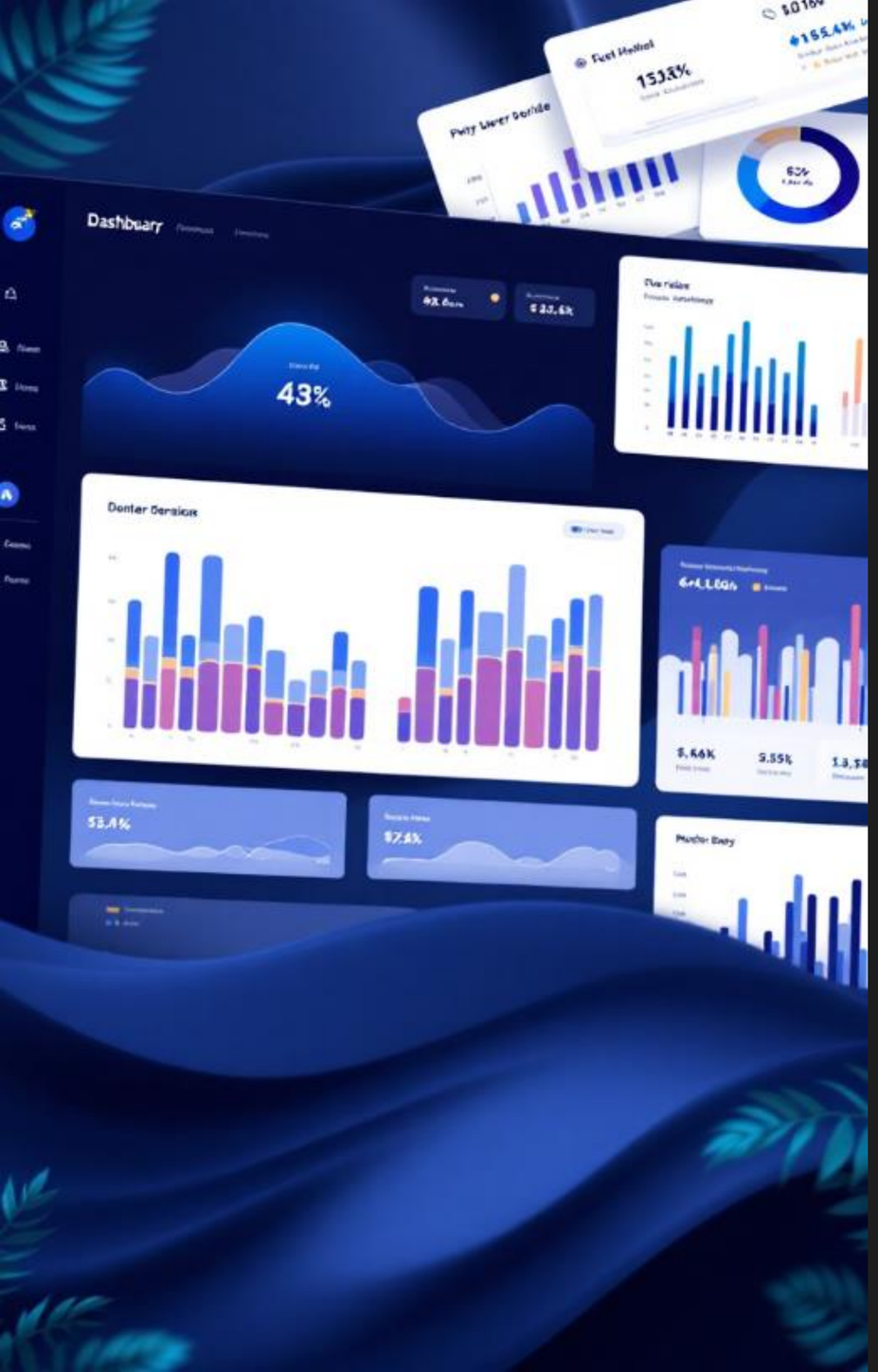
Alta integración con librerías como NumPy, Matplotlib y Scikit-learn, formando un ecosistema completo para ciencia de datos.

✅ Versatilidad

Facilidad de lectura y escritura de múltiples formatos como CSV, JSON, Excel y SQL, simplificando el flujo de trabajo con datos.

✅ Rendimiento

Optimización mediante operaciones vectorizadas basadas en NumPy y uso eficiente de memoria para procesar grandes conjuntos de datos.



Aplicaciones Principales de Pandas

Manipulación y Limpieza

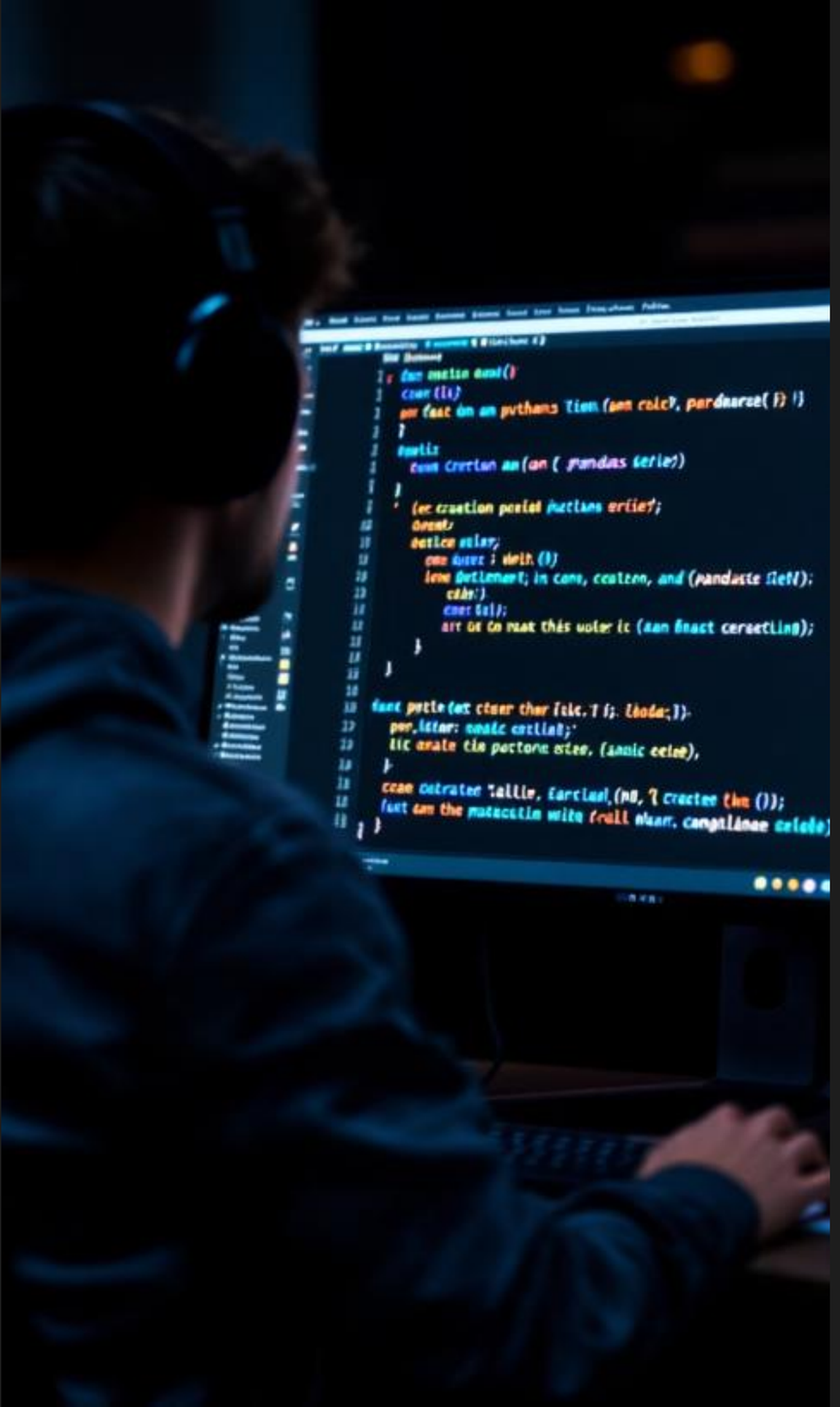
Permite manejar datos faltantes, eliminar duplicados, transformar columnas y normalizar datos de manera eficiente, facilitando la preparación de datos para análisis posteriores.

Análisis Exploratorio

Facilita la inspección y descripción de los datos a través de estadísticas básicas, visualizaciones y resúmenes rápidos, permitiendo entender la estructura y patrones en los datos.

Integración

Se integra perfectamente con otras herramientas del ecosistema de ciencia de datos como NumPy, Matplotlib, Seaborn y Scikit-learn, creando un flujo de trabajo completo.



El Tipo de Dato Serie en Pandas

Definición

Una Serie en Pandas es una estructura de datos unidimensional, similar a un vector en NumPy o una lista en Python, pero con índices personalizables. Cada elemento está asociado con una etiqueta (índice).

Características

Es unidimensional, contiene una sola columna de valores. Cada elemento tiene un índice asociado que puede ser numérico o de tipo string. Soporta operaciones matemáticas y funciones estadísticas aplicadas eficientemente.

Flexibilidad

Se basa en NumPy, lo que le otorga alto rendimiento en cálculos numéricos. Permite la conversión entre listas, diccionarios y arreglos de NumPy de manera sencilla.

Creación de Series en Pandas

Desde una Lista

La forma más básica de crear una Serie es a partir de una lista de Python. Pandas asignará automáticamente índices numéricos comenzando desde 0.

```
import pandas as pd
datos = [10, 20, 30, 40]
serie = pd.Series(datos, index=['a', 'b', 'c', 'd'])
print(serie)
```

Con Índices Personalizados

Podemos especificar nuestros propios índices al crear una Serie, lo que facilita el acceso a los datos mediante etiquetas significativas.

```
serie = pd.Series([100, 200, 300], index=["a", "b", "c"])
print(serie)
```

Desde un Diccionario

Al crear una Serie desde un diccionario, las claves se convierten en los índices y los valores en los datos de la Serie.

```
datos = {"Manzanas": 5, "Peras": 3, "Naranjas": 8}
serie = pd.Series(datos)
print(serie)
```

Acceso y Operaciones con Series

1

Acceso por Índice

Podemos acceder a los valores de una Serie de manera similar a los diccionarios en Python, utilizando el índice como clave.

```
print(serie["Peras"])  # Devuelve 3
print(serie[0])        # Devuelve 5 (accediendo por índice numérico)
```

2

Acceso a Múltiples Elementos

El slicing permite seleccionar subconjuntos de una Serie.

```
print(serie[["Manzanas", "Naranjas"]])  # Devuelve los valores de las claves seleccionadas
```

3

Filtrado de Datos

Podemos aplicar condiciones para seleccionar elementos específicos.

```
print(serie[serie > 4])  # Devuelve solo los valores mayores a 4
```

4

Operaciones Matemáticas

Pandas permite realizar operaciones matemáticas sobre los elementos de una Serie de forma rápida y eficiente, como suma, resta, multiplicación y división.

```
numeros = pd.Series([1, 2, 3, 4, 5])

print(numeros * 2)  # Multiplicación por escalar
print(numeros + 10) # Suma de un valor a cada elemento
print(numeros.mean())  # Media de los valores
print(numeros.max())   # Valor máximo
print(numeros.min())   # Valor mínimo
```

El Tipo de Dato DataFrame en Pandas

1

Estructura Bidimensional

Un DataFrame es una estructura de datos bidimensional, similar a una tabla de base de datos o una hoja de cálculo. Se compone de filas y columnas, donde cada columna puede contener diferentes tipos de datos.

2

Colección de Series

Cada columna de un DataFrame representa una Serie de Pandas, lo que permite tener diferentes tipos de datos en cada columna mientras se mantiene la estructura tabular.

3

Indexación Avanzada

Soporta indexación avanzada y selección de datos mediante etiquetas o índices numéricos, facilitando el acceso a subconjuntos específicos de datos.

Selección de Datos en DataFrames

Seleccionar varias columnas

```
print(df[["Nombre", "Edad"]])  # Devuelve solo las columnas seleccionadas
```

Selección de filas

```
# Seleccionar una fila por su índice con loc
print(df.loc[1])  # Devuelve la fila con índice 1

# Seleccionar una fila por su posición con iloc
print(df.iloc[2])  # Devuelve la tercera fila
```

Selección múltiples filas

```
print(df.loc[0:2])  # Devuelve las filas con índice de 0 a 2
```

Métodos de Exploración de DataFrames

Pandas proporciona métodos rápidos para explorar la estructura y el contenido de un DataFrame.

head() y tail()

Estos métodos permiten visualizar las primeras o últimas filas del DataFrame.

```
print(df.head(2))  # Primeras 2 filas
print(df.tail(1))  # Última fila
```

Métodos de Exploración de DataFrames

Info()

Muestra un resumen del DataFrame, incluyendo tipos de datos y valores nulos.

```
print(df.info())
```

Describe()

Genera estadísticas descriptivas para las columnas numéricas.

```
print(df.describe())
```

Métodos de Sumarización en Pandas

min()

Valor Mínimo

Encuentra el valor mínimo en cada columna numérica

max()

Valor Máximo

Encuentra el valor máximo en cada columna numérica

mean()

Promedio

Calcula el promedio de los valores en cada columna

sum()

Suma Total

Suma todos los valores en cada columna numérica

count()

Contar

Cuenta los valores no nulos en cada columna

median()

Mediana

Calcula la mediana de cada columna

Métodos de Sumarización en Pandas

Pandas permite obtener información estadística de manera rápida y eficiente a través de estos métodos de sumarización. Estos métodos pueden aplicarse tanto a Series como a DataFrames completos, facilitando el análisis exploratorio de datos y la obtención de insights preliminares.

Ejemplo

```
print(df["Edad"].min())     # Valor mínimo
print(df["Edad"].max())     # Valor máximo
print(df["Edad"].count())   # Número de elementos
print(df["Edad"].mean())    # Media (promedio)
print(df["Edad"].median())  # Mediana
print(df["Edad"].sum())     # Suma total de los valores
```

Métodos para Datos Categóricos

unique()

Devuelve los valores únicos en una columna.

```
print(df['Ciudad'].unique())  # Salida: ['Madrid' 'Barcelona' 'Valencia']
```

nunique()

Cuenta el número de valores únicos en una columna.

```
print(df['Ciudad'].nunique())  # Salida: 3
```

value_counts()

Cuenta la frecuencia de cada valor único en una columna.

```
print(df['Ciudad'].value_counts())  # Salida: Madrid       1
                                    #         Barcelona    1
                                    #         Valencia     1
```

Preguntas

Sección de preguntas

Obtención y Preparación

de Datos

Continúe con las actividades