Obtención y Preparación

de Datos

Sesión 4

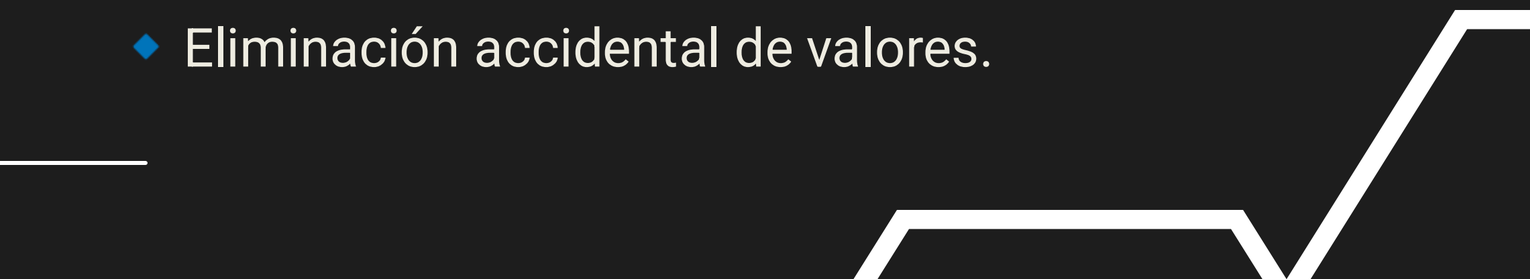
¿Qué es un Valor Perdido?

Datos ausentes en un conjunto de datos.

En Pandas, se representan con NaN.

Causas comunes:

- Errores en la recolección de datos (por ejemplo, sensores defectuosos).
- Registros incompletos (campos no ingresados en encuestas).
- Conversión de formatos (al importar datos de distintas fuentes).
- Eliminación accidental de valores.



Identificación de Valores Perdidos

✅ isnull(): Identifica valores nulos.

✅ notnull(): Identifica valores no nulos.

✅ sum(): Cuenta la cantidad de valores nulos por columna.

Ejemplo isnull()

```
import pandas as pd

df = pd.DataFrame({"Nombre": ["Ana", "Carlos", "Luis", None],
                   "Edad": [25, 30, None, 40]})

print(df.isnull())  # Muestra un DataFrame con True donde hay valores nulos
print(df.isnull().sum())  # Cuenta los valores nulos por columna
```

Tratamiento de Valores Perdidos

Si los valores perdidos son pocos y no afectan significativamente el análisis, pueden eliminarse con dropna():

```
df_limpio = df.dropna()  # Elimina todas las filas con al menos un NaN
df_sin_columnas_nulas = df.dropna(axis=1)  # Elimina columnas con NaN
```

Consideraciones:

- axis=0: Elimina filas (por defecto).
- axis=1: Elimina columnas.
- how="any": Elimina si hay al menos un NaN en la fila/columna.
- how="all": Solo elimina si todas las celdas son NaN.

Imputación de Datos

En lugar de eliminar datos, podemos reemplazar los valores perdidos con fillna()

Ejemplo:

```
df["Edad"].fillna(df["Edad"].mean(), inplace=True)  # Rellena con la media
df["Nombre"].fillna("Desconocido", inplace=True)  # Rellena valores de texto
```

Imputación de Valores Cualitativos

Para datos categóricos, se puede usar la moda (valor más frecuente):

```
df["Categoría"].fillna(df["Categoría"].mode()[0], inplace=True)
```

Detección y Filtrado de Outliers

Calculo a través de los cuartiles. Un valor se considera atípico si se encuentra fuera del rango esperado, determinado por el rango intercuartílico (IQR).

El IQR se calcula como la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1):

$$IQR = Q3 - Q1$$

Un dato se considera un outlier si está fuera del siguiente rango:

$$Q1 - 1.5 \times IQR \quad \text{o} \quad Q3 + 1.5 \times IQR$$

¿Qué son los cuartiles?

Los cuartiles son valores que dividen un conjunto de datos ordenados en cuatro partes iguales, cada una con aproximadamente el 25% de los datos. Se utilizan en estadística para analizar la distribución de los datos y detectar valores atípicos.

Principales cuartiles:

- Primer cuartil (Q1): Es el valor por debajo del cual se encuentra el 25% de los datos. También se conoce como percentil 25.

- Segundo cuartil (Q2): Es la mediana, que divide el conjunto en dos mitades, dejando el 50% de los datos por debajo.

- Tercer cuartil (Q3): Es el valor por debajo del cual está el 75% de los datos. También se conoce como percentil 75.

¿Qué son los cuartiles?

Ejemplo en Python

```python
import numpy as np

Q1 = df["Edad"].quantile(0.25)  # Primer cuartil
Q3 = df["Edad"].quantile(0.75)  # Tercer cuartil
IQR = Q3 - Q1

# Filtramos los datos dentro del rango permitido
df_filtrado = df[(df["Edad"] >= Q1 - 1.5*IQR) & (df["Edad"] <= Q3 + 1.5*IQR)]
```

¿Qué es el Data Wrangling?

El data wrangling es el proceso de transformar, limpiar y estructurar datos en un formato adecuado para el análisis.

Principales Tareas de Data Wrangling

- Limpieza de datos: Eliminación de valores faltantes, duplicados y errores.
- Transformación de datos: Cambio de formatos, creación de nuevas variables.
- Integración de datos: Combinación de diferentes fuentes de datos.

Técnicas de Manipulación de Datos

Muestreo aleatorio (sample())

```
df_muestra = df.sample(n=5)  # Selecciona 5 registros aleatorios
```

Permutación de la Data (np.random.permutation())

```
df = df.iloc[np.random.permutation(len(df))]
```

Ordenamiento (sort_values())

```
df_ordenado = df.sort_values(by="Edad", ascending=False)
```

Detección y Eliminación de Duplicados

Identificación de duplicados duplicated()

```
df = pd.DataFrame({
    'A': [1, 2, 2, 3],
    'B': [4, 5, 5, 6]
})
print(df.duplicated())
```

Eliminación de duplicados drop_duplicates():

```
df_sin_duplicados = df.drop_duplicates()
print(df_sin_duplicados)
```

Reemplazo de Valores

Podemos reemplazar valores específicos con replace()

```
df["Categoría"] = df["Categoría"].replace({"M": "Masculino", "F": "Femenino"})
```

También podemos reemplazar múltiples valores:

```
df.replace(["Desconocido", "N/A"], np.nan, inplace=True)
```

Discretización y Binning

Discretización y binning son técnicas utilizadas para transformar variables continuas en variables discretas.

Sintaxis básica:

```
pd.cut(x, bins, labels=False, include_lowest=False)
```

Ejemplo en Python

Para agrupar valores numéricos en intervalos, usamos pd.cut():

```
df["Rango_Edad"] = pd.cut(df["Edad"], bins=[0, 18, 35, 60, 100], labels=["Joven", "Adulto Joven", "Adulto", "Mayor"])
```

Aplicación de Funciones y Expresiones Lambda

- En Pandas, podemos aplicar funciones a los datos de forma eficiente mediante el método apply()

```
import pandas as pd

df = pd.DataFrame({"Nombre": ["Ana", "Carlos", "Luis"],
                   "Edad": [25, 30, 40]})

# Aplicar una función para calcular la edad en 10 años
df["Edad_Futura"] = df["Edad"].apply(lambda x: x + 10)
print(df)
```

- Las expresiones lambda son funciones anónimas que permiten realizar operaciones rápidas sin necesidad de definir una función tradicional

```
df["Nombre_Mayus"] = df["Nombre"].apply(lambda x: x.upper())
print(df)
```

Manipulación de DataFrames

- Agregar Columnas: df['Nueva'] = valores

```
df["Salario"] = [5000, 6000, 7000]  # Agregar columna
```

- Eliminar Columnas: df.drop(columns=['Columna'])

```
df = df.drop(columns=["Salario"])  # Eliminar columna
```

- Redimensión: melt()

```
df_melt = df.melt(id_vars=["Nombre"], value_vars=["Edad", "Edad_Futura"], var_name="Variable", value_name="Valor")
print(df_melt)
```

Renombrar y Redefinir Índices

- set_index(): Define índice.

```
df = df.set_index("Nombre")  # Usar "Nombre" como índice
```

- reset_index(): Restablece índice original.

```
df = df.reset_index()  # Volver al índice original
```

- rename(): Cambia nombres de columnas.

```
df = df.rename(columns={"Edad": "Años", "Categoria_Edad": "Grupo_Etario"})
print(df)
```

Preguntas

Sección de preguntas

Obtención y Preparación

de Datos

Continúe con las actividades