TALENTO TECH

TIC

Universidad Tecnológica de Bolívar

Pandas 2

Ejecutor técnico: Jorge Luis Villalba Acevedo

www.utb.edu.co/talento-tech

Handling Missing Data

Resumen General

El manejo de datos faltantes es una parte crucial en el análisis de datos. En pandas, los valores faltantes están representados generalmente por `NaN` (Not a Number) para valores numéricos y `None` para valores de tipo objeto. Pandas proporciona varias funciones para *identificar*, *eliminar* o *reemplazar* datos faltantes.

1. Identificación de Datos Faltantes

Funciones clave:

Función	Descripción	Ejemplo de uso
`isna`	Devuelve una estructura de datos booleana indicando si los valores son nulos (NaN o None)	`df.isna()`
`notna`	Devuelve una estructura de datos booleana indicando los valores no nulos	`df.notna()`

Ejemplo:

```
import pandas as pd
import numpy as np

data = pd.Series([1, np.nan, 3, None, 5])
```

2. Eliminación de Datos Faltantes

Funciones clave:

Función	Descripción	Ejemplo de uso
`dropna()`	Elimina las filas o columnas con datos faltantes	`df.dropna(axis=0)` (elimina filas) o `df.dropna(axis=1)` (elimina columnas)

Ejemplo:

```
df = pd.DataFrame({
    'A': [1, 2, np.nan],
    'B': [5, np.nan, np.nan],
    'C': [7, 8, 9]
})
```

Ejemplo:

```
# Elimina filas con al menos un NaN
df_cleaned = df.dropna()
```

3. Rellenar Datos Faltantes

Funciones clave:

Función	Descripción	Ejemplo de uso
`fillna()`	Rellena valores nulos con un valor específico o utilizando métodos como interpolación	`df.fillna(0)` o `df.fillna(method='ffill')`

Ejemplo:

```
df_filled = df.fillna(0)
```

Otro ejemplo con propagación hacia adelante (`ffill`):

```
df_filled_ffill = df.fillna(method='ffill')
```

4. Reemplazo de Datos Faltantes

Funciones clave:

Función	Descripción	Ejemplo de uso
`replace()`	Reemplaza valores específicos en un DataFrame o Serie	`df.replace(np.nan, 0)`

Ejemplo:

```
df_replaced = df.replace(np.nan, 99)   # Reemplaza todos los NaN con 99
```

Data Transformation

Resumen General

La transformación de datos es una de las tareas más comunes en el análisis de datos. pandas proporciona una amplia variedad de herramientas para reordenar, filtrar, modificar y agrupar datos de manera eficiente.

1. Eliminar Duplicados

Funciones clave:

Función	Descripción	Ejemplo de uso
`duplicated()`	Devuelve una Serie booleana indicando si una fila es duplicada	`df.duplicated()`
`drop_duplicates()`	Elimina filas duplicadas	`df.drop_duplicates()`

Ejemplo:

```
import pandas as pd

df = pd.DataFrame({
    'A': ['foo', 'bar', 'foo', 'bar', 'foo'],
    'B': [1, 2, 1, 2, 3]
})

df_unique = df.drop_duplicates()
```

2. Reemplazar Valores

Funciones clave:

Función	Descripción	Ejemplo de uso
`replace()`	Reemplaza valores específicos	`df.replace('foo', 'baz')`

Ejemplo:

```
df_replaced = df.replace('foo', 'baz')
```

3. Renombrar Índices o Columnas

Funciones clave:

Función	Descripción	Ejemplo de uso
`rename()`	Cambia los nombres de columnas o índices	`df.rename(columns={'A': 'Category'})`

Ejemplo:

```
df_renamed = df.rename(columns={'A': 'Category'})
```

4. Mapear o Aplicar Funciones a Columnas

Funciones clave:

Función	Descripción	Ejemplo de uso
`map()`	Aplica una función o diccionario de reemplazo a una columna Serie	`df['A'].map({'foo': 'FOO', 'bar': 'BAR'})`
`apply()`	Aplica una función a cada columna o fila del DataFrame	`df.apply(np.sum)` (aplica una suma por columnas)
`applymap()`	Aplica una función elemento por elemento en un DataFrame	`df.applymap(lambda x: x * 2)`

Ejemplo:

```
# Usando map para reemplazar valores en una columna
df_mapped = df['A'].map({'foo': 'FOO', 'bar':
'BAR'})

# Usando apply para sumar por columnas
df_sum = df.apply(lambda x: x.sum())

# Usando applymap para multiplicar cada valor por 2
df_multiplied = df.applymap(lambda x: x * 2)
```

5. Discretización y Binning

Funciones clave:

Función	Descripción	Ejemplo de uso
`cut()`	Divide los datos en intervalos o bins	`pd.cut(df['B'], bins=[0, 1, 2, 3])`
`qcut()`	Divide los datos en quantiles	`pd.qcut(df['B'], q=4)`

Ejemplo:

```
# Crear intervalos (bins)
# Crear quantiles
quantiles = pd.qcut(df['B'], q=2)
print(quantiles)
```

```
0    (0.999, 2.0]
1    (0.999, 2.0]
2    (0.999, 2.0]
3    (0.999, 2.0]
4      (2.0, 3.0]
Name: B, dtype: category
Categories (2, interval[float64, right]): [(0.999, 2.0] <
(2.0, 3.0]]
```

Ejemplo:

```
# Crear intervalos (bins)
bins = pd.cut(df['B'], bins=[0, 1, 2, 3])
print(bins)
```

```
0    (0, 1]
1    (1, 2]
2    (0, 1]
3    (1, 2]
4    (2, 3]
Name: B, dtype: category
Categories (3, interval[int64, right]): [(0, 1] < (1, 2] < (2, 3]]
```

6. Detección y Filtrado con Expresiones Condicionales

Funciones clave:

Función	Descripción	Ejemplo de uso
Filtrado con condicionales	Filtra filas basadas en condiciones lógicas	`df[df['A'] == 'foo']`

Ejemplo:

```
# Filtrar filas donde 'A' es 'foo'
df_filtered = df[df['A'] == 'foo']
```

Referencias

1. **McKinney, W. (2017).** *Python for data analysis: Data wrangling with pandas, numpy, and ipython* (2nd ed.). O'Reilly Media.
2. **Sweigart, A. (2019).** *Automate the boring stuff with Python: Practical programming for total beginners* (2nd ed.).
3. **González Duque, R. (2015).** *Python para todos*. Independently published.
4. **Bosch, J. (2020).** *Introducción a la programación en Python: Algoritmos y lógica para principiantes*. Independently published.