PIPELINE EVERPEAK

January 17, 2026

1 Construcción de pipeline completo para la limpieza de datos de EverPeak

1.0.1 Creación de función básica para reemplazar valores centinela

En la base de datos, la columna costumer_age presenta valores erroneos, el código los reemplazará por NaN´s para no tener sesgos estadísticos.

[3]:
```
# Función para reemplazar Sentinels
def reemplazar_sentinels(df, sentinels):
    df["customer_age"] = df["customer_age"].replace(sentinels, pd.NA)
    return df
# ----------------------------------------------------------------------
# Importar librería y leer datos
import pandas as pd
df = pd.read_csv("/datasets/everpeak_retail.csv")

# observar valores ausentes iniciales
print("Valores ausentes", df["customer_age"].isna().sum())

# Fijar valores a corregir
valores_erroneos = [-999, 999, 0, -1]

# Aplicar función, guardar resultados y observar cambios
df = reemplazar_sentinels(df,valores_erroneos)
print("Valores ausentes", df["customer_age"].isna().sum())
```

```
Valores ausentes 150
Valores ausentes 175
```

1.0.2 Mejora de la función para limpieza de sentinels en múltiples columnas

Ahora vamos a mejorar esta función para que pueda limpiar valores erroneos en múltiples columnas númericas añadiendo un bucle que recorra las diferenes columnas númericas, reemplace los valores centinelas por NaNs y regrese el DF corregido

[4]:
```
 # Crear función mejorada
def reemplazar_sentinels(df, sentinels, numeric_cols):
    for col in numeric_cols:
```

```
        df[col] = df[col].replace(sentinels, pd.NA)
    return df

# ------------------------------------------------------------------
# Importar librería y leer datos
import pandas as pd
df = pd.read_csv("/datasets/everpeak_retail.csv")

# observar valores ausentes iniciales
print("Valores ausentes iniciales:")
print(df[["customer_age", "price"]].isna().sum())

# Fijar valores a corregir y columnas
valores_erroneos = [-999, 999, 0, -1]
columnas_numericas = ["customer_age", "price"]

# Aplicar función y observar cambios
df = reemplazar_sentinels(df, valores_erroneos, columnas_numericas)
print("\nValores ausentes después:")
print(df[["customer_age", "price"]].isna().sum())
```

```
Valores ausentes iniciales:
customer_age    150
price             0
dtype: int64

Valores ausentes después:
customer_age    175
price             2
dtype: int64
```

1.0.3 Ahora crearemos una función para rellenar valores ausentes

Esta función convertirá columnas a tipo númerico y rellenará los valores ausentes con el promedio

[5]:
```
# Crear función para rellenar valores ausentes
def rellenar_ausentes(df, cols_fill):
    for col in cols_fill:
        df[col]= pd.to_numeric(df[col], errors= 'coerce') # convertir a numérico
        df[col].fillna(df[col].mean(), inplace= True) #rellenar con promedio␣
 ↪usando inplace=True
    return df

# ------------------------------------------------------------------
# Importar librería y leer datos
import pandas as pd
df = pd.read_csv("/datasets/everpeak_retail.csv")
```

```
# Observar valores ausentes iniciales
print("Valores ausentes iniciales:")
print(df[["customer_age", "price"]].isna().sum())

# Definir columnas a rellenar
columnas_rellenar = ["customer_age", "price"]

# Aplicar función y observar cambios
df = rellenar_ausentes(df, columnas_rellenar)
print("\nValores ausentes después:")
print(df[["customer_age", "price"]].isna().sum())
```

```
Valores ausentes iniciales:
customer_age    150
price             0
dtype: int64

Valores ausentes después:
customer_age    0
price           0
dtype: int64
```

Ahora que ya tenemos nuestras dos funciones principales para la limpieza de datos, se procede a crear el pipeline de limpieza para automatizar la limpieza de datos ante posibles cambios futuros.

1.0.4 Creación de pipeline para limpieza automática de datos

[6]:
```
# Funciones auxiliares (ya creadas en chunks anteriores)
def reemplazar_sentinels(df, sentinels, numeric_cols):
    for col in numeric_cols:
        df[col] = df[col].replace(sentinels, pd.NA)
    return df

def rellenar_ausentes(df, cols_fill):
    for col in cols_fill:
        df[col] = pd.to_numeric(df[col], errors='coerce')
        df[col].fillna(df[col].mean(), inplace=True)
    return df

# Crear función pipeline
def limpiar_df(df):
    valores_erroneos = [-999, 999, 0, -1] #valores erroneos definidos
    columnas_numericas = ["customer_age", "price"] #columnas numericas definidas

    df = reemplazar_sentinels(df, valores_erroneos, columnas_numericas)
    df = rellenar_ausentes(df, columnas_numericas)

    return df
```

```python
# ------------------------------------------------------------------
# Importar librería y leer datos
import pandas as pd
df = pd.read_csv("/datasets/everpeak_retail.csv")

# observar valores ausentes iniciales
print("Valores ausentes iniciales:")
print(df[["customer_age", "price"]].isna().sum())

# Aplicar pipeline completo
df = limpiar_df(df)
print("\nValores ausentes después del pipeline:")
print(df[["customer_age", "price"]].isna().sum())
```

```
Valores ausentes iniciales:
customer_age    150
price             0
dtype: int64

Valores ausentes después del pipeline:
customer_age    0
price           0
dtype: int64
```

1.0.5 Conclusiones

Se creó un pipeline completo de limpieza de datos que incluye:

1.- Función limpiar_df que integra dos procesos esenciales

2.- Reemplazo de valores centinela (sentinel values) como -999, 999, 0, -1

3.- Relleno de valores ausentes usando la mediana

4.- Aplicación correcta del orden: primero sentinels, luego ausentes

Conceptos clave utilizados:

1.- Pipeline de limpieza: Secuencia ordenada de transformaciones

2.- Valores centinela: Códigos que representan datos faltantes o erróneos

3.- Imputación con mediana: Método robusto para manejar valores ausentes

4.- Funciones modulares: Código reutilizable y bien estructurado

Habilidades técnicas aplicadas:

1.- Definición de listas de columnas específicas

2.-Uso de pandas para manipulación de datos

3.-Creación de funciones con parámetros claros

4.-Aplicación secuencial de transformaciones

Objetivos logrados

En el mundo real, la limpieza de datos consume el 80% del tiempo de un analista. Este proceso implicó:

1.-Identificar y corregir problemas comunes en datasets

2.-Automatizar procesos repetitivos

3.-Mantener la integridad de los datos

4.-Crear código reutilizable para futuros proyectos