**GESTIÓN VALORES NULOS**

**Librerías:**

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.impute import KNNImputer
import seaborn as sns
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)
```

**Tipos nulos y cómo sustituir, teniendo en cuenta el contexto:**

**Columnas categóricas:**

- Si el % de nulos es pequeño y tenemos una **moda** representativa, cambiamos por la moda
- En caso contrario, categorizar como “unknown”

**Columnas numéricas:**

- La **mediana** es más robusta que la media

**Eliminar filas** con valores **nulos en** una **columna:**

```
df.dropna(subset=['columna'], inplace=True)
```

**Subrayar** filas con valores **nulos:**

```
df.style.highlight_null(color='yellow')
```

**PASOS GESTIÓN DE NULOS:**

**1. Conocer el porcentaje de nulos por columna del DF**

```
porc_nulos = (df.isnull().sum() / df.shape[0]) * 100
```

Visualizarlo en un DF solo con las columnas con nulos:

```
df_nulos = pd.DataFrame(porc_nulos, columns = ["%_nulos"])
df_nulos[df_nulos["%_nulos"] > 0]
```

**2. Extraer columnas con valores nulos:**

**Categóricas:**

```
nulos = df[df.columns[df.isnull().any()]].select_dtypes(include = "O").columns
```

**Numéricas:** include = np.number

**3. Conocer la distribución de categorías de las columnas con nulos**

```
for col in nulos:
  display(df[col].value_counts() / df.shape[0] * 100)
```

Saca el porcentaje de cada valor único en las columnas con nulos del paso2

**4. Reemplazar los valores nulos**: **s**i hay un dato dominante, podremos sustituir los nulos por la moda, si no por 'unknown'. **Métodos**:

**.fillna():** rellena los valores nulos con uno **específico**. Sintaxis:
Modificar por moda:

```
moda = df['columna'].mode()[0]
df['columna] = df['columna'].fillna(moda)
```

Modificar por valor predeterminado:

```
df['columna] = df['columna'].fillna('unknown')
```

**SimpleImputer**: igual que fillna. Permite imputar un **mismo valor**, media, mediana o moda o valor constante, a todos los nulos de una columna. Sintaxis:
1. Imputar con la media `imputer = SimpleImputer(strategy='mean')`
   - mean, median, most_frequent, constant
2. Transformar los datos

```
df['columna'] = imputer.fit_transform(df[['columna']])
```

**IterativeImputer**: hace una **predicción** del valor nulo basándose en los datos de toda la tabla. Sintaxis:
1. Crear una instancia del IterativeImputer;

```
imputer = IterativeImputer(max_iter 10, random_state 42)
```

   - max_iter: nº de iteraciones que usará para completar los datos nulos
   - random_state: semilla para el generador de números aleatorios

2. Ajustar y transformar los datos

```
data_imputed = imputer.fit_transform(df[['col1', 'col2', 'col3',]])
```

   - datos: columna(s) que contienen los valores faltantes

3. Sustituir las columnas modificadas si es más de una. Si es una solo hacer el paso2, cambiando data imputed por columna a modificar.

```
df[['col1', 'col2', 'col3',]] = data_imputed
```

**KNNImputer**: hace una **predicción** para los valores nulos **numéricos** basándose en los **valores vecinos**, columnas que le pasamos que podría usar.

1. Crear una instancia del IterativeImputer, recomendable vecinos 3-5

```
imputer = KNNImputer(n_neighbors = 3-5)
```

2. Ajustar y transformar los datos

```
knn_imputed = imputer_knn.fit_transform(df[[‘col1’, ‘col2’]])
```

- col1, col2: columnas con la que completa los nulos

3. Añadir nuevas columnas al DataFrame y comprobar con .describe() que son datos realistas

```
df[[‘col1’, ‘col2’]]= knn_imputed
```

4. Eliminar columnas que están repetidas