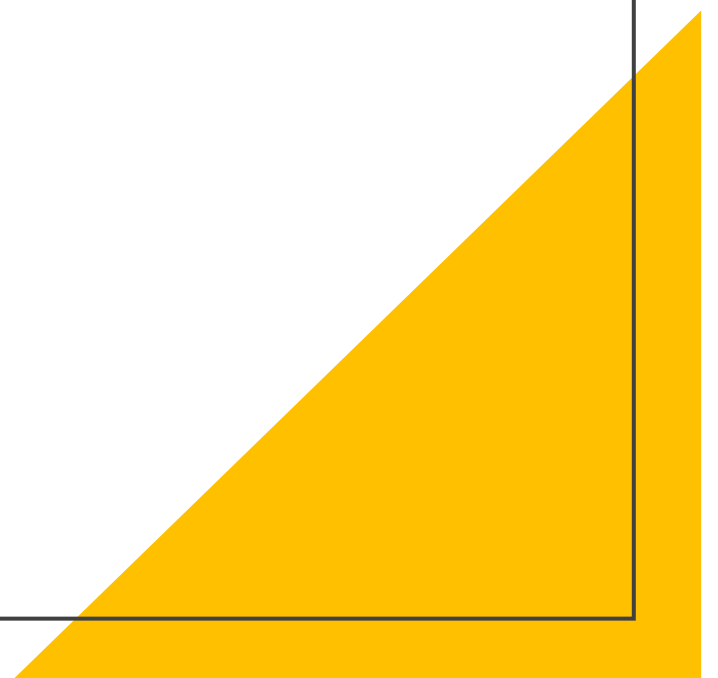


MODULO II - **PREPROCESAMIENTO DE** **DATOS**

DATA SCIENCE CON PYTHON



Introducción

- Los datos faltantes (también conocidos como "valores perdidos" o "missing values") son un aspecto crítico en el análisis de datos por varias razones:
 1. **Impacto en la calidad de los resultados:** pueden llevar a conclusiones erróneas o sesgadas.
 2. **Perdida de información:** Cada valor ausente representa información que no se aprovecha.
 3. **Sesgo en los resultados:** Si los datos faltantes no se manejan de manera apropiada, pueden introducir sesgos en los análisis.
 4. **Problemas de validez y generalización:** Los modelos entrenados en datos con valores faltantes pueden tener dificultades para generalizarse a nuevos datos.
 5. **Problemas éticos y de privacidad:** La omisión de datos puede estar relacionada con problemas de privacidad o con la falta de disponibilidad de ciertos tipos de datos.
 6. **Requisito de completitud:** Algunos algoritmos y técnicas de análisis de datos requieren que los datos estén completos para funcionar correctamente. Si se dejan datos faltantes, se pueden limitar las opciones de análisis.

¿Por qué es

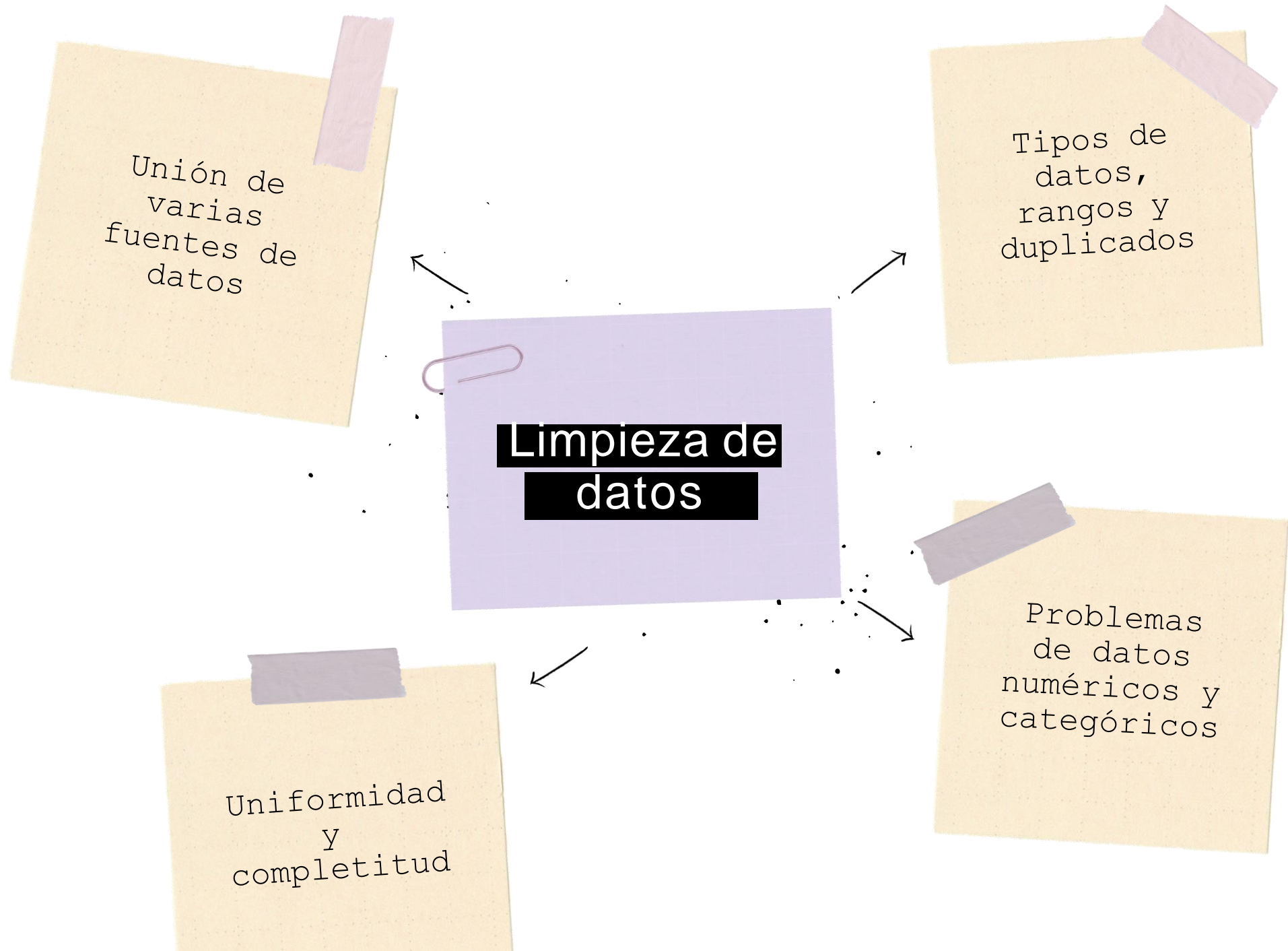
Importante?

Entra basura



Sale basura





Limpieza de datos

Tipos de datos:



```
#Validar tipo de dato
```

```
df.dtypes
```

```
#Cambiar tipo de dato
```

```
df['<nombre_columna>'].astype('<tipo_dato>')
```

object

float

bool

category

datetime

int

Limpieza de datos

Fechas:



```
#Convertir a Datetime
```

```
datetime.strptime(df['<columna_fechas>'], <formato>)
```

```
#Obtener fecha actual
```

```
fecha_actual = dt.date.today( )
```

```
#Si la columna es tipo dato datetime
```

```
df['año'] = df['fecha'].dt.year #Obtener año
```

```
df['mes'] = df['fecha'].dt.month #Obtener mes
```

Limpieza de datos

Duplicados:



```
#Visualizar valores duplicados  
df[df.duplicated()] #Opción 1  
df[df.duplicated(subset = [<nombrs_columnas>], keep = False)] #Opción 2  
  
#Eliminar duplicados  
df.drop_duplicates(inplace = True)
```

Limpieza de datos

Variables categóricas:



#Visualizar valores no definidos

```
df['<col_categorica>'] = pd.cut(df['<col_num>'], bins= <rango_div>, labels = <nombres_div>)
```

#Cambiar valores

```
df['<columna>'] = df['<columna>'].replace({<valor_anterior>:<valor_nuevo>})
```

#Eliminar espacios

```
df['<columna>'].str.strip()
```

#Pasar a minúsculas

```
df['<columna>'] = df['<columna>'].str.lower()
```

#Pasar a mayúsculas

```
df['<columna>'] = df['<columna>'].str.upper()
```


Limpieza de datos

Variables no definidas:



```
#Visualizar valores no definidos  
inconsistentes = set(df1['<nombre_columna>']).difference(df2['<nombre_columna>'])  
  
#Eliminar valores no definidos  
df = df[~df['<columna>'].isin(inconsistentes)]
```

Datos nulos

Estos valores pueden afectar el rendimiento y la capacidad de predicción de nuestros modelos. ¿Cómo podemos identificarlos?

- NaN
- None
- 0
- Null
- Na
- Not Available

Media,
moda,
mediana

Imputar
valores
con ML

valor
anterior
o valor
siguiente

```
#Visualizar cantidad de valores nulos
df.isna().sum()

#Graficar valores nulos
import missingno as msno
msno.bar(df) #Opción 1
msno.matrix(android_games) #Opción 2

#Eliminar valores nulos
df.dropna()


#Reemplazar valores nulos
df.fillna(value=values, inplace=True)
```

Cómo identificar y contabilizar datos faltantes en un conjunto de datos utilizando Pandas y NumPy.

Para identificar y contabilizar datos faltantes en un conjunto de datos utilizando Pandas y NumPy, puedes seguir estos pasos:

- **Paso 1: Importar las bibliotecas**

python

 Copy code

```
import pandas as pd
import numpy as np
```

- **Paso 2: Cargar los datos**

Carga tu conjunto de datos en un DataFrame de Pandas. Utilizando la función `pd.read_csv()` o `pd.read_excel()`

python

 Copy code

```
# Ejemplo de carga de datos desde un archivo CSV
df = pd.read_csv('tu_archivo.csv')
```

Cómo identificar y contabilizar datos faltantes en un conjunto de datos utilizando Pandas y NumPy.

- **Paso 3: Identificar datos faltantes**

- `isna()` o `isnull()`: Estas funciones devuelven un DataFrame booleano del mismo tamaño que el DataFrame original, donde `True` indica la presencia de un valor faltante y `False` indica un valor presente.

```
python Copy code  
  
# Identificar valores faltantes en todo el DataFrame  
missing_data = df.isna() # También puedes usar df.isnull()  
  
# Identificar valores faltantes en una columna específica  
missing_data_column = df['nombre_de_columna'].isna()
```

Cómo identificar y contabilizar datos faltantes en un conjunto de datos utilizando Pandas y NumPy.

- **Paso 4: Contabilizar datos faltantes**

- Una vez identificado los datos faltantes, puedes contarlos para obtener una idea de cuántos valores faltantes hay en tu conjunto de datos. Puedes usar la función `sum()` después de aplicar los métodos `isna()` o `notna()`.

```
python Copy code  
  
# Contar valores faltantes en todo el DataFrame  
missing_count = df.isna().sum()  
  
# Contar valores faltantes en una columna específica  
missing_count_column = df['nombre_de_columna'].isna().sum()
```

Cómo identificar y contabilizar datos faltantes en un conjunto de datos utilizando Pandas y NumPy.

- **Paso 4: Contabilizar datos faltantes**

- Una vez identificado los datos faltantes, puedes contarlos para obtener una idea de cuántos valores faltantes hay en tu conjunto de datos. Puedes usar la función `sum()` después de aplicar los métodos `isna()` o `notna()`.

```
python Copy code  
  
# Contar valores faltantes en todo el DataFrame  
missing_count = df.isna().sum()  
  
# Contar valores faltantes en una columna específica  
missing_count_column = df['nombre_de_columna'].isna().sum()
```