TAREA: Limpieza de Datos

Nombre: Jaime Alexis Hernandez Franco

Materia: Introducción a la Ciencia de Datos

Profesor: Jaime Alejandro Romero Sierra

Fecha: 20/10/2025

Repositorio GitHub: Alexiiiiss0122/CIENCIA-DE-DATOS-2

Descripción inicial de la base de datos

La base contiene información sobre clientes bancarios, sus características personales, financieras y de contacto, así como si se suscribieron a una campaña de marketing.

Columna	Descripción
age	Edad del cliente
job	Ocupación
marital	Estado civil
education	Nivel educativo
balance	Saldo promedio en la cuenta
housing	Tiene crédito hipotecario (yes/no)
loan	Tiene otro préstamo (yes/no)
contact	Tipo de contacto (celular o teléfono)
campaign	Número de contactos realizados durante la campaña
previous	Número de contactos previos
poutcome	Resultado de la campaña anterior
subscribed	Si el cliente se suscribió (yes/no)

Proceso de limpieza de datos

Se realizó un análisis completo para detectar valores nulos, duplicados y errores de texto. Los valores faltantes se rellenaron con el promedio (para columnas numéricas) o con la moda (para columnas categóricas). Los valores anómalos como 'Auto%#' y 'unknown' fueron eliminados y no remplazados.Se eliminaron duplicados también.

Ejemplos de técnicas aplicadas:

- Relleno de nulos con promedio o moda usando fillna().

- Conversión de columnas numéricas con pd.to_numeric(no se pudo con balance).

- Eliminacon de palabras inválidas ('Auto%#', 'unknown').

- Eliminación de duplicados con drop_duplicates().

```
#Importamos la libreria que vamos a ocupar
import pandas as pd
```

✓ 1.2s Python

```
# Cargar la base sucia
data= pd.read_csv("Base_Sucia.csv")
data
```

✓ 0.0s Python

	age	job	marital	education	balance	housing	loan	contact	campaign	previous	poutcome	subscribed
0	56.0	management	married	tertiary	90489	no	yes	telephone	36.0	2.0	nonexistent	yes
1	69.0	management	married	tertiary	40823	no	yes	cellular	33.0	4.0	failure	yes
2	46.0	technician	divorced	unknown	67555	yes	no	cellular	6.0	1.0	success	yes
3	32.0	admin	single	tertiary	84190	no	yes	telephone	3.0	2.0	success	yes
4	60.0	self-employed	divorced	primary	45418	no	no	cellular	35.0	0.0	failure	no
...	...	...	...	...	...	...	...	...	...	...	...	...
11671	44.0	self-employed	divorced	primary	17282	yes	NaN	cellular	24.0	2.0	success	no
11672	NaN	self-employed	single	secondary	NaN	yes	no	telephone	13.0	4.0	success	no
11673	44.0	retired	married	NaN	13685	yes	no	telephone	46.0	3.0	nonexistent	yes
11674	66.0	blue-collar	married	secondary	62446	no	yes	telephone	25.0	3.0	nonexistent	no
11675	64.0	blue-collar	divorced	secondary	-889	no	no	telephone	22.0	2.0	nonexistent	yes

11676 rows × 12 columns

```
# Información general
data.info()
```

[4] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11676 entries, 0 to 11675
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         11326 non-null  float64
 1   job         11326 non-null  object
 2   marital     11326 non-null  object
 3   education   11326 non-null  object
 4   balance     11326 non-null  object
 5   housing     11326 non-null  object
 6   loan        11326 non-null  object
 7   contact     11326 non-null  object
 8   campaign    11326 non-null  float64
 9   previous    11101 non-null  float64
 10  poutcome    11326 non-null  object
 11  subscribed  11326 non-null  object
dtypes: float64(3), object(9)
memory usage: 1.1+ MB
```

```
data.head()
```

[5] ✓ 0.0s Python

	age	job	marital	education	balance	housing	loan	contact	campaign	previous	poutcome	subscribed
0	56.0	management	married	tertiary	90489	no	yes	telephone	36.0	2.0	nonexistent	yes
1	69.0	management	married	tertiary	40823	no	yes	cellular	33.0	4.0	failure	yes
2	46.0	technician	divorced	unknown	67555	yes	no	cellular	6.0	1.0	success	yes
3	32.0	admin	single	tertiary	84190	no	yes	telephone	3.0	2.0	success	yes
4	60.0	self-employed	divorced	primary	45418	no	no	cellular	35.0	0.0	failure	no

```
#Generamos un df con True donde hay NaN y False donde hay datos
data.isnull()
```

[6] ✓ 0.0s Python

	age	job	marital	education	balance	housing	loan	contact	campaign	previous	poutcome	subscribed
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
11671	False	False	False	False	False	False	True	False	False	False	False	False
11672	True	False	False	False	True	False	False	False	False	False	False	False
11673	False	False	False	True	False	False	False	False	False	False	False	False
11674	False	False	False	False	False	False	False	False	False	False	False	False
11675	False	False	False	False	False	False	False	False	False	False	False	False

11676 rows × 12 columns

```
# Contar los valores nulos
data.isnull().sum()
```

[7] ✓ 0.0s Python

```
age           350
job           350
marital       350
education     350
balance       350
housing       350
loan          350
contact       350
campaign      350
previous      575
poutcome      350
subscribed    350
dtype: int64
```

```
#Rellenamos las columnas numericas que tienen los valores nulos por el promedio
columnas_numericas = ["age", "campaign", "previous"]

for col in columnas_numericas:
    data[col] = data[col].fillna(data[col].mean())
```

[8] ✓ 0.0s Python

```
#Rellenar los valores nulos en las columnas categoricas por la moda
columnas_categoricas = ["job", "marital", "education", "housing", "loan", "contact", "poutcome", "subscribed"]

for col in columnas_categoricas:
    moda = data[col].mode()[0]
    data[col] = data[col].fillna(moda)
    print(f"Columna '{col}'los valores nulos han sido reemplazados por la moda ('{moda}')")
```

[9] ✓ 0.0s Python

```
Columna 'job'los valores nulos han sido reemplazados por la moda ('unemployed')
Columna 'marital'los valores nulos han sido reemplazados por la moda ('divorced')
Columna 'education'los valores nulos han sido reemplazados por la moda ('primary')
Columna 'housing'los valores nulos han sido reemplazados por la moda ('no')
Columna 'loan'los valores nulos han sido reemplazados por la moda ('yes')
Columna 'contact'los valores nulos han sido reemplazados por la moda ('telephone')
Columna 'poutcome'los valores nulos han sido reemplazados por la moda ('success')
Columna 'subscribed'los valores nulos han sido reemplazados por la moda ('yes')
```

```
#Convierto los valores nulo de la columna balance a ceros
data["balance"] = data["balance"].fillna(0)
```

[10] ✓ 0.0s Python

```
[10]  #Convierto los valores nulo de la columna balance a ceros
      data["balance"] = data["balance"].fillna(0)
      ✓ 0.0s  Python

[11]  data2=data.isnull().sum()
      ✓ 0.0s  Python

[12]  #Checamos otra vez
      data2
      ✓ 0.0s  Python

...   age          0
      job          0
      marital      0
      education    0
      balance      0
      housing      0
      loan         0
      contact      0
      campaign     0
      previous     0
      poutcome     0
      subscribed   0
      dtype: int64
```

```
[13]  #Vemos las palabras de cada columna
      for col in ["job", "marital", "education", "housing", "loan", "contact", "poutcome", "subscribed"]:
          print(f"\nColumna: {col}")
          print(data[col].unique())
      ✓ 0.0s  Python

...
      Columna: job
      ['management' 'technician' 'admin' 'self-employed' 'services' 'unemployed'
       'entrepreneur' 'retired' 'student' 'blue-collar' 'Auto%#']

      Columna: marital
      ['married' 'divorced' 'single']

      Columna: education
      ['tertiary' 'unknown' 'primary' 'secondary']

      Columna: housing
      ['no' 'yes' 'Auto%#']

      Columna: loan
      ['yes' 'no']

      Columna: contact
      ['telephone' 'cellular']

      Columna: poutcome
      ['nonexistent' 'failure' 'success' 'Auto%#']

      Columna: subscribed
      ['yes' 'no' 'Auto%#']
```

```python
#Eliminar palabras que son invalidas en las columnas
valores_invalidos = ["Auto%#", "unknown"]

columnas_a_revisar = ["job", "education", "housing", "poutcome", "subscribed"]

for col in columnas_a_revisar:
    data = data[~data[col].isin(valores_invalidos)]
```

[14] ✓ 0.0s Python

```python
#Verificar que ya no esten
for col in columnas_a_revisar:
    print(f"\nColumna: {col}")
    print(data[col].unique())
```

[15] ✓ 0.0s Python

```
Columna: job
['management' 'admin' 'self-employed' 'services' 'retired' 'technician'
 'unemployed' 'entrepreneur' 'blue-collar' 'student']

Columna: education
['tertiary' 'primary' 'secondary']

Columna: housing
['no' 'yes']

Columna: poutcome
['nonexistent' 'failure' 'success']

Columna: subscribed
['yes' 'no']
```

```python
#Checamos los duplicados
data.duplicated().sum()
```

[17] ✓ 0.0s Python

```
np.int64(596)
```

```python
#Eliminamos duplicados
data = data.drop_duplicates()
```

[18] ✓ 0.0s Python

```python
data.duplicated().sum()
```

[23] ✓ 0.0s Python

```
np.int64(0)
```

```
#Checamos nombre de las columnas
data.columns
```

✓ 0.0s Python

```
Index(['age', 'job', 'marital', 'education', 'balance', 'housing', 'loan',
       'contact', 'campaign', 'previous', 'poutcome', 'subscribed'],
      dtype='object')
```

```
#Renombramos las columnas a español
data=data.rename(columns={'age': 'Edad', 'job':'trabajo', 'education':'educacion', 'balance':'saldo', 'marital':'estado_civil', 'l
  'contact':'contacto', 'campaing':'campañas', 'previous':'contactos_previos', 'subscribed':'suscrito'})
data
```

✓ 0.0s Python

	Edad	trabajo	estado_civil	educacion	saldo	credito_vivienda	prestamo	contacto	campaign	contactos_previos	poutcom
0	56.000000	management	married	tertiary	90489	no	yes	telephone	36.0	2.0	nonexiste
1	69.000000	management	married	tertiary	40823	no	yes	cellular	33.0	4.0	failu
3	32.000000	admin	single	tertiary	84190	no	yes	telephone	3.0	2.0	succe
4	60.000000	self-employed	divorced	primary	45418	no	no	cellular	35.0	0.0	failu
6	38.000000	self-employed	divorced	primary	46695	no	no	telephone	7.0	0.0	succe
...	...	...	...	...	...	...	...	...	...	...	
11668	36.000000	entrepreneur	divorced	tertiary	52	yes	yes	cellular	21.0	0.0	nonexiste
11670	64.000000	services	divorced	tertiary	90206	yes	yes	cellular	14.0	1.0	succe
11671	44.000000	self-employed	divorced	primary	17282	yes	yes	cellular	24.0	2.0	succe
11672	46.181529	self-employed	single	secondary	0	yes	no	telephone	13.0	4.0	succe

Al final checamos el dataset y guaramos la base de datos limpia.

```
data
```

[22] ✓ 0.0s Python

	Edad	trabajo	estado_civil	educacion	saldo	credito_vivienda	prestamo	contacto	campaign	contactos_previos	poutcom
0	56.000000	management	married	tertiary	90489	no	yes	telephone	36.0	2.0	nonexiste
1	69.000000	management	married	tertiary	40823	no	yes	cellular	33.0	4.0	failu
3	32.000000	admin	single	tertiary	84190	no	yes	telephone	3.0	2.0	succe
4	60.000000	self-employed	divorced	primary	45418	no	no	cellular	35.0	0.0	failu
6	38.000000	self-employed	divorced	primary	46695	no	no	telephone	7.0	0.0	succe
...	...	...	...	...	...	...	...	...	...	...	
11668	36.000000	entrepreneur	divorced	tertiary	52	yes	yes	cellular	21.0	0.0	nonexiste
11670	64.000000	services	divorced	tertiary	90206	yes	yes	cellular	14.0	1.0	succe
11671	44.000000	self-employed	divorced	primary	17282	yes	yes	cellular	24.0	2.0	succe
11672	46.181529	self-employed	single	secondary	0	yes	no	telephone	13.0	4.0	succe
11673	44.000000	retired	married	primary	13685	yes	no	telephone	46.0	3.0	nonexiste

7567 rows × 12 columns

```
data.to_csv("Base_limpia_Bank.csv", index=False)
```

[25] ✓ 0.0s Python

Conclusiones

Durante la limpieza se identificaron valores nulos, duplicados y errores introducidos intencionalmente para practicar el proceso. Se aplicaron técnicas de imputación y corrección de registros, manteniendo la integridad de la base y traducción de las columnas para facilitar su entendimiento. El resultado final es un conjunto de datos limpio, y listo para su análisis.