step_1_data_cleaning

July 6, 2025

1 Data cleaning - Mobile User Behavior

1.1 librerias y datos

[1]:
```
import pandas as pd
```

[2]:
```
original_dataset = pd.read_csv("user_behavior_dataset_original.csv")
```

1.2 Exploracion inicial

- head() »> para ver los primeros registros
- tail() »> para ver los ultimos registros
- shape »> para ver el numero de filas y columnas. O la forma del dataset
- info() »> para ver informaciones generales del dataset y sus columnas

[4]:
```
original_dataset.head(4)
```

[4]:
```
   User ID    Device Model Operating System  App Usage Time (min/day)  \
0        1  Google Pixel 5          Android                       393
1        2       OnePlus 9          Android                       268
2        3    Xiaomi Mi 11          Android                       154
3        4  Google Pixel 5          Android                       239

   Screen On Time (hours/day)  Battery Drain (mAh/day)  \
0                         6.4                     1872
1                         4.7                     1331
2                         4.0                      761
3                         4.8                     1676

   Number of Apps Installed  Data Usage (MB/day)  Age  Gender  \
0                        67                 1122   40    Male
1                        42                  944   47  Female
2                        32                  322   42    Male
3                        56                  871   20    Male

   User Behavior Class
0                    4
1                    3
```

```
2                    2
3                    3
```

[5]:
```
original_dataset.tail(4)
```

[5]:
```
     User ID        Device Model Operating System  App Usage Time (min/day)  \
696      697        Xiaomi Mi 11          Android                       316
697      698      Google Pixel 5          Android                        99
698      699  Samsung Galaxy S21          Android                        62
699      700           OnePlus 9          Android                       212

     Screen On Time (hours/day)  Battery Drain (mAh/day)  \
696                         6.8                     1965
697                         3.1                      942
698                         1.7                      431
699                         5.4                     1306

     Number of Apps Installed  Data Usage (MB/day)  Age  Gender  \
696                        68                 1201   59    Male
697                        22                  457   50  Female
698                        13                  224   44    Male
699                        49                  828   23  Female

     User Behavior Class
696                    4
697                    2
698                    1
699                    3
```

[6]:
```
original_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   User ID                     700 non-null    int64
 1   Device Model                700 non-null    object
 2   Operating System            700 non-null    object
 3   App Usage Time (min/day)    700 non-null    int64
 4   Screen On Time (hours/day)  700 non-null    float64
 5   Battery Drain (mAh/day)     700 non-null    int64
 6   Number of Apps Installed    700 non-null    int64
 7   Data Usage (MB/day)         700 non-null    int64
 8   Age                         700 non-null    int64
 9   Gender                      700 non-null    object
 10  User Behavior Class         700 non-null    int64
dtypes: float64(1), int64(7), object(3)
```

```
memory usage: 60.3+ KB
```

[7]:
```
original_dataset.shape
```

[7]:
```
(700, 11)
```

1.3 Data cleaning

1.3.1 Valores nulos

Buscar por valores nulos en el dataset

[9]:
```
original_dataset.isnull().sum()
```

[9]:
```
User ID                        0
Device Model                   0
Operating System               0
App Usage Time (min/day)       0
Screen On Time (hours/day)     0
Battery Drain (mAh/day)        0
Number of Apps Installed       0
Data Usage (MB/day)            0
Age                            0
Gender                         0
User Behavior Class            0
dtype: int64
```

1.3.2 Valores duplicados

Buscar por valores duplicados

[11]:
```
original_dataset.duplicated().sum()
```

[11]:
```
np.int64(0)
```

1.3.3 Valores de columnas categoricas

Verificar los valores de cada columna, para buscar por inconsistencias.

[12]:
```
original_dataset["Device Model"].value_counts()
```

[12]:
```
Device Model
Xiaomi Mi 11          146
iPhone 12             146
Google Pixel 5        142
OnePlus 9             133
Samsung Galaxy S21    133
Name: count, dtype: int64
```

[15]:
```
original_dataset["Gender"].value_counts()
```

[15]:
```
Gender
Male      364
Female    336
Name: count, dtype: int64
```

[16]:
```
original_dataset["Operating System"].value_counts()
```

[16]:
```
Operating System
Android    554
iOS        146
Name: count, dtype: int64
```

[17]:
```
original_dataset["User Behavior Class"].value_counts()
```

[17]:
```
User Behavior Class
2    146
3    143
4    139
5    136
1    136
Name: count, dtype: int64
```

Definir la columna “User Behavior Class”, pues el titulo no sugiere de que se trata.

Info del dataset:

- User Behavior Class: Classification of user behavior based on usage patterns (1 to 5).
- Each entry is categorized into one of five user behavior classes, ranging from light to extreme usage.

La columna ‘User Behavior Class’ contiene valores numéricos (del 1 al 5), pero estos números representan categorías de usuarios, no cantidades reales. Por eso, tiene sentido tratar esta columna como una variable categórica ordinal.

Una nueva columna se incluye para convertir los valores numericos a variables categoricas, así:

- 1: ‘Muy bajo’,
- 2: ‘Bajo’,
- 3: ‘Medio’,
- 4: ‘Alto’,
- 5: ‘Muy alto’

[22]:
```
original_dataset["Defined Behavior"] = original_dataset["User Behavior Class"].
 ↪map({
    1: 'Very low',
    2: 'Low',
    3: 'Medium',
    4: 'High',
    5: 'Very high'
}).astype('category')
```

[23]:
```
original_dataset["Defined Behavior"].value_counts()
```

[23]:
```
Defined Behavior
Low          146
Medium       143
High         139
Very high    136
Very low     136
Name: count, dtype: int64
```

1.3.4 Valores de columnas numericas

[24]:
```
original_dataset.describe().T
```

[24]:

	count	mean	std	min	25%	\
User ID	700.0	350.500000	202.216880	1.0	175.75	
App Usage Time (min/day)	700.0	271.128571	177.199484	30.0	113.25	
Screen On Time (hours/day)	700.0	5.272714	3.068584	1.0	2.50	
Battery Drain (mAh/day)	700.0	1525.158571	819.136414	302.0	722.25	
Number of Apps Installed	700.0	50.681429	26.943324	10.0	26.00	
Data Usage (MB/day)	700.0	929.742857	640.451729	102.0	373.00	
Age	700.0	38.482857	12.012916	18.0	28.00	
User Behavior Class	700.0	2.990000	1.401476	1.0	2.00	

	50%	75%	max
User ID	350.5	525.25	700.0
App Usage Time (min/day)	227.5	434.25	598.0
Screen On Time (hours/day)	4.9	7.40	12.0
Battery Drain (mAh/day)	1502.5	2229.50	2993.0
Number of Apps Installed	49.0	74.00	99.0
Data Usage (MB/day)	823.5	1341.00	2497.0
Age	38.0	49.00	59.0
User Behavior Class	3.0	4.00	5.0

1.3.5 Configuracion del nombre de las columnas

Cambiar el nombre de las columnas para evitar posibles conflictos posteriormente.

[25]:
```
# Eliminar espacios al inicio y final de cada nombre de columna

original_dataset.columns = original_dataset.columns.str.strip()
```

[26]:
```
# Convertir todos los nombres de columna a minúsculas

original_dataset.columns = original_dataset.columns.str.lower()
```

[27]:
```
# Reemplazar los espacios internos por guiones bajos

```

```
original_dataset.columns = original_dataset.columns.str.replace(" ", "_")
```

[31]:
```
original_dataset.columns
```

[31]:
```
Index(['user_id', 'device_model', 'operating_system',
       'app_usage_time_(min/day)', 'screen_on_time_(hours/day)',
       'battery_drain_(mah/day)', 'number_of_apps_installed',
       'data_usage_(mb/day)', 'age', 'gender', 'user_behavior_class',
       'defined_behavior'],
      dtype='object')
```

1.4 Exportar el archivo

[32]:
```
original_dataset.to_csv("user_behavior_dataset_clean.csv", index=False)
```

step_2_eda

July 6, 2025

1 EDA - Mobile User Behavior

- Estadística descriptiva

- Análisis univariado

1.1 Importar librerias y cargar los datos

[1]:
```
# Importar librerias

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# Cargar los datos

dataset = pd.read_csv("user_behavior_dataset_clean.csv")
```

1.1.1 Vista general del dataset

Visualizar todas las columnas y algunas metricas básicas

[3]:
```
dataset.describe(include="all").T
```

[3]:

	count	unique	top	freq	mean	\
user_id	700.0	NaN	NaN	NaN	350.5	
device_model	700	5	Xiaomi Mi 11	146	NaN	
operating_system	700	2	Android	554	NaN	
app_usage_time_(min/day)	700.0	NaN	NaN	NaN	271.128571	
screen_on_time_(hours/day)	700.0	NaN	NaN	NaN	5.272714	
battery_drain_(mah/day)	700.0	NaN	NaN	NaN	1525.158571	
number_of_apps_installed	700.0	NaN	NaN	NaN	50.681429	
data_usage_(mb/day)	700.0	NaN	NaN	NaN	929.742857	
age	700.0	NaN	NaN	NaN	38.482857	
gender	700	2	Male	364	NaN	
user_behavior_class	700.0	NaN	NaN	NaN	2.99	
defined_behavior	700	5	Low	146	NaN	

	std	min	25%	50%	75%	max
user_id	202.21688	1.0	175.75	350.5	525.25	700.0
device_model	NaN	NaN	NaN	NaN	NaN	NaN
operating_system	NaN	NaN	NaN	NaN	NaN	NaN
app_usage_time_(min/day)	177.199484	30.0	113.25	227.5	434.25	598.0
screen_on_time_(hours/day)	3.068584	1.0	2.5	4.9	7.4	12.0
battery_drain_(mah/day)	819.136414	302.0	722.25	1502.5	2229.5	2993.0
number_of_apps_installed	26.943324	10.0	26.0	49.0	74.0	99.0
data_usage_(mb/day)	640.451729	102.0	373.0	823.5	1341.0	2497.0
age	12.012916	18.0	28.0	38.0	49.0	59.0
gender	NaN	NaN	NaN	NaN	NaN	NaN
user_behavior_class	1.401476	1.0	2.0	3.0	4.0	5.0
defined_behavior	NaN	NaN	NaN	NaN	NaN	NaN

1.2 Estadística descriptiva (información general del dataset)

Média de las columnas numericas Definición: Es el valor promedio de un conjunto de datos; representa una medida general de tendencia central. Se obtiene al sumar todos los valores y dividirlos entre la cantidad de elementos.

[4]:
```
# Calcular média solamente de las columnas numéricas
dataset.select_dtypes(include='number').mean()
```

[4]:
```
user_id                        350.500000
app_usage_time_(min/day)       271.128571
screen_on_time_(hours/day)       5.272714
battery_drain_(mah/day)       1525.158571
number_of_apps_installed        50.681429
data_usage_(mb/day)            929.742857
age                             38.482857
user_behavior_class              2.990000
dtype: float64
```

Moda de las columnas numéricas Es el valor que más veces se repite en el conjunto de datos. Puede haber más de una moda (bimodal, multimodal) o ninguna. Es útil para identificar los valores más comunes o frecuentes.

[5]:
```
# Calcular la moda de las columnas numéricas, excluyendo la columna 'user_id'
dataset.select_dtypes(include="number").drop(columns='user_id').mode().T
```

[5]:

	0	1
app_usage_time_(min/day)	64.0	NaN
screen_on_time_(hours/day)	1.6	NaN
battery_drain_(mah/day)	490.0	NaN
number_of_apps_installed	10.0	16.0
data_usage_(mb/day)	122.0	284.0
age	34.0	51.0

```
user_behavior_class           2.0     NaN
```

Mediana de las columnas numericas Definición: Es el valor central de los datos cuando se ordenan de menor a mayor. Si hay una cantidad impar de datos, es el del medio; si es par, es el promedio de los dos centrales. Es útil cuando hay valores extremos que pueden distorsionar la media.

[6]:
```
dataset.select_dtypes(include='number').drop(columns='user_id').median()
```

[6]:
```
app_usage_time_(min/day)       227.5
screen_on_time_(hours/day)       4.9
battery_drain_(mah/day)       1502.5
number_of_apps_installed        49.0
data_usage_(mb/day)            823.5
age                             38.0
user_behavior_class              3.0
dtype: float64
```

Rango de las columnas numericas Definición: Es la diferencia entre el valor más alto y el más bajo del conjunto. Da uma idea rápida de la amplitud o dispersión de los datos, aunque no considera cómo están distribuidos internamente.

[7]:
```
col_numericas = dataset.select_dtypes(include='number').drop(columns="user_id")
rango = col_numericas.max() - col_numericas.min()

print(rango)
```

```
app_usage_time_(min/day)       568.0
screen_on_time_(hours/day)      11.0
battery_drain_(mah/day)       2691.0
number_of_apps_installed        89.0
data_usage_(mb/day)           2395.0
age                             41.0
user_behavior_class              4.0
dtype: float64
```

Desviación estandar Definición: Es una medida de dispersión que indica cuánto, en promedio, se alejan los valores respecto a la media. Cuanto mayor es la desviación estándar, más dispersos están los datos; cuanto menor, más agrupados están cerca de la media.

[8]:
```
dataset.select_dtypes(include='number').drop(columns='user_id').std()
```

[8]:
```
app_usage_time_(min/day)      177.199484
screen_on_time_(hours/day)      3.068584
battery_drain_(mah/day)       819.136414
number_of_apps_installed       26.943324
data_usage_(mb/day)           640.451729
age                            12.012916
```

```
user_behavior_class             1.401476
dtype: float64
```

1.3 Análisis univariado

En el análisis univariado, podemos adoptar la siguiente estratégia para analizar cada variable:

- Variable categórica: Desarrollar análisis numérico y análisis gráfico
- Variable numérica: Desarrollar análisis numérico y gráfico.

1.3.1 Análisis de la columna 'device_model' - Columna categórica - Análisis numérico

La primera columna en la que se hará el análisis es la columna "device_model", que informa los modelos que aparecen dentro del dataset

Conteo de frecuencias absolutas Definición: Es la cantidad de veces que aparece cada categoría en la columna. Muestra cuántas observaciones pertenecen a cada grupo.

[9]:
```
dataset['device_model'].value_counts().rename_axis('device_model').
↪reset_index(name='count')
```

[9]:
```
          device_model  count
0        Xiaomi Mi 11    146
1           iPhone 12    146
2      Google Pixel 5    142
3           OnePlus 9    133
4  Samsung Galaxy S21    133
```

Porcentajes (frecuencias relativas) Definición: Es la proporción que representa cada categoría respecto al total, expresada en porcentaje. Ayuda a comparar categorías de diferentes tamaños.

[10]:
```
# Paso 1: Calcular la frecuencia relativa
freq_relativa = dataset["device_model"].value_counts(normalize=True)

# Paso 2: Multiplicar por 100 para obtener porcentajes
porcentajes = freq_relativa * 100

# Paso 3: Convertir a DataFrame y organizar columnas
porcentajes_df = porcentajes.rename_axis('device_model').
↪reset_index(name='percentage')

# Paso 4: Redondear los porcentajes a dos decimales
porcentajes_df['percentage'] = porcentajes_df['percentage'].round(2)

porcentajes_df
```

[10]:
```
          device_model  percentage
0        Xiaomi Mi 11       20.86
```

```
1           iPhone 12       20.86
2      Google Pixel 5       20.29
3           OnePlus 9       19.00
4  Samsung Galaxy S21       19.00
```

Número de categorías únicas Definición: Indica cuántas categorías diferentes existen en la columna. Es útil para entender la diversidad o variedad de valores.

[11]:
```
categorias_unicas = dataset['device_model'].nunique()

print(f"El número de categorias únicas es: {categorias_unicas}")
```

```
El número de categorias únicas es: 5
```

Identificación de la moda Definición: Es la categoría que aparece con mayor frecuencia en la columna. Representa el valor más común o típico de esa variable.

[12]:
```
moda = dataset["device_model"].mode().to_frame(name='mode')
moda
```

[12]:
```
           mode
0  Xiaomi Mi 11
1     iPhone 12
```

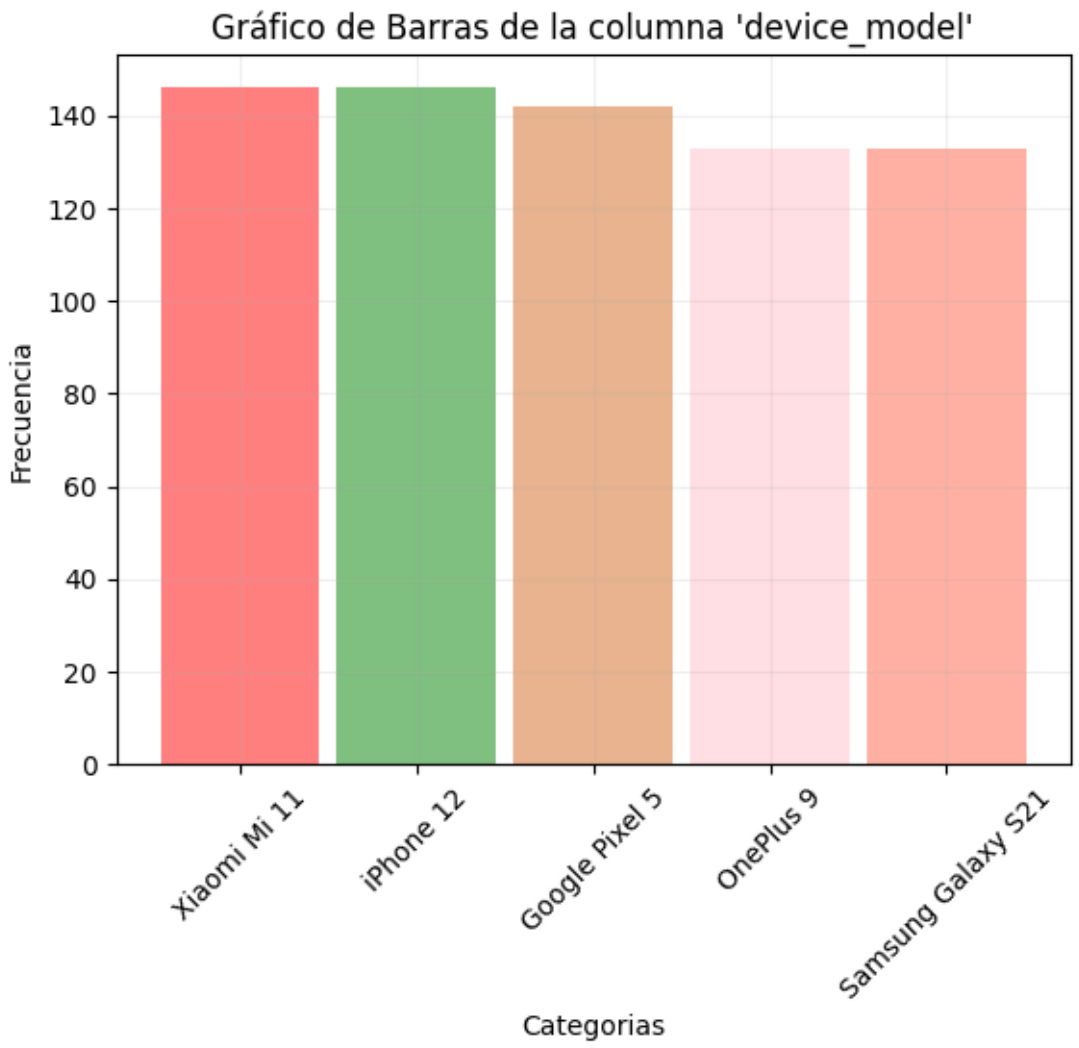
1.3.2 Análisis de la columna 'device_model' - Columna categórica - Análisis gráfico

Creando un gráfico de barras

[31]:
```
conteo = dataset["device_model"].value_counts()
colores = ['red', 'green', 'chocolate', 'pink', 'tomato']

conteo.plot(kind='bar', color=colores, width=0.9, alpha=0.5)

plt.title("Gráfico de Barras de la columna 'device_model'")
plt.xlabel("Categorias")
plt.ylabel("Frecuencia")
plt.grid(alpha=0.2)
plt.xticks(rotation=45)
plt.show()
```



Creando un diagrama de torta (pie chart)

[40]:
```
colores_pie_chart = ['gold', 'lightblue', 'lightcoral', 'lightgreen', 'tomato']

conteo.plot(kind='pie', autopct='%1.f%%', startangle=90,␣
 ↪colors=colores_pie_chart)


plt.title("Gráfico de torta de la columna 'device_model'")
plt.tight_layout()
plt.show()
```

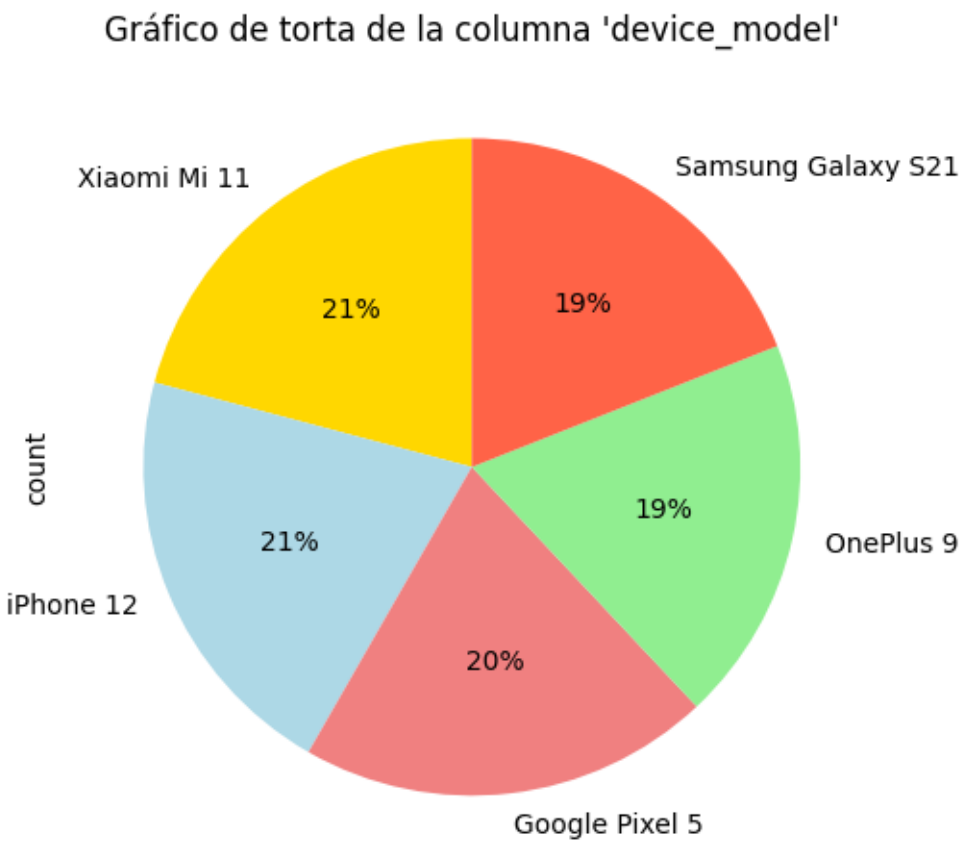


Gráfico de torta de la columna 'device_model'
Xiaomi Mi 11
Samsung Galaxy S21
21%
19%
count
19%
21%
OnePlus 9
iPhone 12
20%
Google Pixel 5