



Analítica de Datos y  
Herramientas de Inteligencia Artificial

**Reporte de actividad 1.2 avance**

Profesor: Alfredo García Suárez

Camila Trujillo Beristain | A01737170  
Bernardo Quintana López | A01658064  
Fernando Guadarrama González | A01379340  
Mauricio Goris García | A01736428

**Campus Puebla**

30 de marzo de 2025

## Reporte Explicativo del Análisis de Datos

Este análisis es del código con proceso completo de análisis de datos, desde la carga y limpieza hasta la visualización, detección de valores atípicos, etc.

El código comienza con la instalación de paquetes necesarios: Estos paquetes proporcionan herramientas para el análisis exploratorio de datos (funpymodeling) y para trabajar con archivos Excel (openpyxl).

```
%pip install funpymodeling
%pip install openpyxl
✓ 2.9s Python
```

Después se importan las bibliotecas esenciales para el análisis y la carga de los datos por el archivo csv, se realiza la exploración de datos, y la búsqueda de valores nulos por columna.

```
#Buscamos valores nulos y calculamos porcentajes por columna
Execute Cell (^Enter) los1 = data.isnull().sum()
print(valores_nulos1)

[ ]

...  Administrador      0
      Usuario         0
      botón correcto   762
      tiempo de interacción 762
      mini juego      156
      número de interacción 762
      color presionado 762
      dificultad      0
      fecha           0
      Juego           0
      auto push       762
      tiempo de lección 177
      tiempo de sesión 606
      dtype: int64
```

Esto indica que varias columnas tienen valores nulos, especialmente "botón correcto", "tiempo de interacción", "número de interacción", "color presionado" y "auto push" con 762 valores nulos cada una.

Se verifican los tipos de datos de cada columna y separamos las variables por numéricas y categóricas: para las variables numéricas generales (excluyendo "tiempo de sesión" y "tiempo de lección"), se reemplazan los valores nulos con la media y para "tiempo de sesión" y "tiempo de lección", se calcula la media excluyendo ceros y luego se reemplazan los nulos:

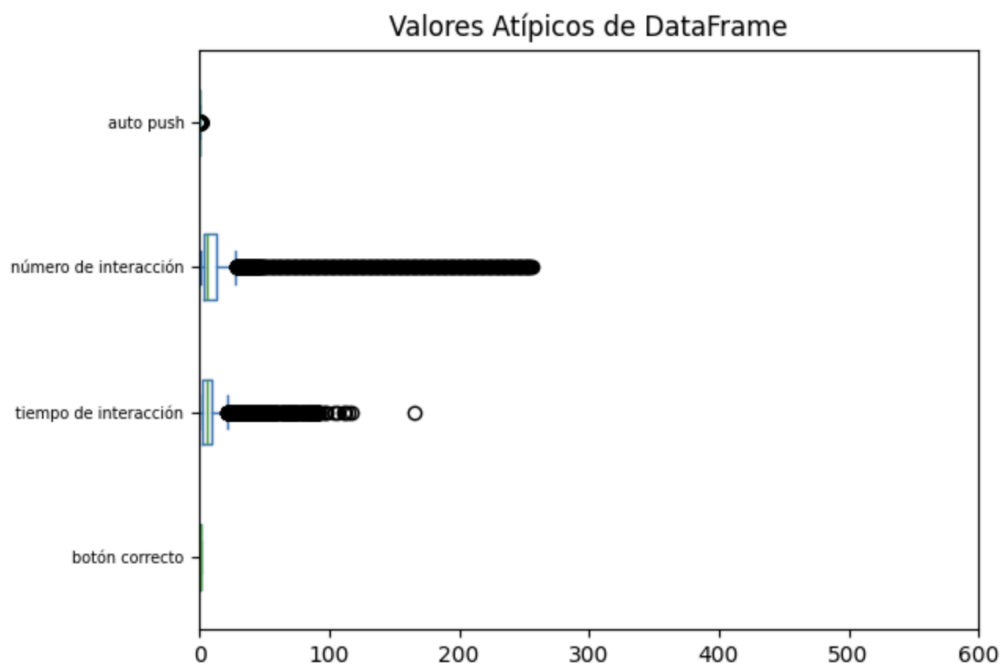
```
media_sesion_sin_ceros = tiempo_sesion.mean()
media_leccion_sin_ceros = tiempo_leccion.mean()
```

```
tiempo_sesion_sin_nulos = tiempo_sesion.fillna(media_sesion_sin_ceros)
tiempo_leccion_sin_nulos = tiempo_leccion.fillna(media_leccion_sin_ceros)
```

Posterior a eso, para la visualización de los datos se crea un diagrama de caja para identificar valores atípicos:

```
fig = plt.figure(figsize = (15,10))
numericas_generales_sin_nulos.plot(kind='box', vert=False)
plt.xlim([0, 600])
plt.title('Valores Atípicos de DataFrame')
plt.yticks(fontsize=7, rotation=0)
plt.show()
```

<Figure size 1500x1000 with 0 Axes>



Después de que el diagrama de caja horizontal mostrara valores atípicos, se hace una detección y manejo de ellos, donde se calculan los límites para valores atípicos usando el rango intercuartílico (IQR):

```

y=numericas_generales_sin_nulos

percentile25=y.quantile(0.25) #Q1
percentile75=y.quantile(0.75) #Q3
iqr= percentile75 - percentile25

Limite_Superior_iqr= percentile75 + 1.5*iqr
Limite_Inferior_iqr= percentile25 - 1.5*iqr
print("Limite superior permitido", Limite_Superior_iqr)
print("Limite inferior permitido", Limite_Inferior_iqr)

```

```

Limite superior permitido botón correcto      2.50000
tiempo de interacción      21.81771
número de interacción      28.84852
auto push      0.00000
dtype: float64
Limite inferior permitido botón correcto      -1.500000
tiempo de interacción      -9.731186
número de interacción      -12.509112
auto push      0.000000
dtype: float64

```

Después se realizó un proceso de limpieza de datos en Python donde primero se filtraron outliers, luego se detectaron valores nulos (669 en "tiempo de interacción", 371 en "número de interacción" y 816 en "auto push") usando `isnull().sum()`, y finalmente se creó una copia del dataframe rellenando los nulos con la media redondeada de cada columna, evidenciando errores tipográficos como "isnu1" y "filing" en lugar de las funciones correctas, pero demostrando claramente el flujo típico de limpieza: filtrar, analizar nulos e imputar valores.

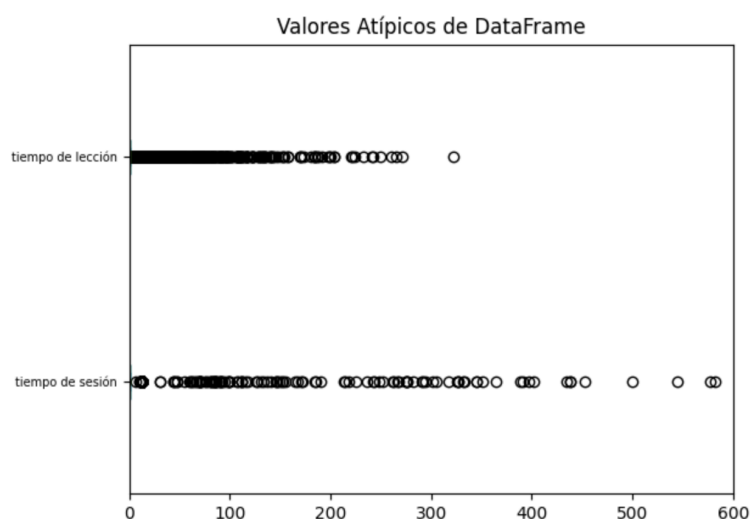
Posterior motramos los atípicos de dataframe en las columnas numéricas, graficando con el diagrama de caja, y mostrando la distribución de los datos y sus valores extremos.

```

fig = plt.figure(figsize = (15,10))
numericas_con0.plot(kind='box', vert=False)
plt.xlim([0, 600])
plt.title('Valores Atípicos de DataFrame')
plt.yticks(fontsize=7, rotation=0)
plt.show()

```

<Figure size 1500x1000 with 0 Axes>



De ahí se realizó un análisis de datos para las variables "tiempo de sesión" y "tiempo de lección", calculando los percentiles 1% y 99% para identificar valores atípicos. Se intentó filtrar los datos conservando solo los valores dentro de estos límites (0.0 como mínimo y 296.97/118.95 como máximo). Para terminar detectando 59 valores nulos en cada columna del filtrado.

```
valores_nulos3=data4_igr.isnull().sum()
valores_nulos3

[ ]

... tiempo de sesión      59
    tiempo de lección     59
    dtype: int64
```

Posterior se hacen operaciones de manipulación de datos con pandas, incluyendo la copia de un DataFrame combinado (data3\_igredata4\_igr), el relleno de valores nulos con la media redondeada de otro conjunto de datos (data4\_igr), y la visualización de las primeras filas de data3\_igr, donde todas las columnas ("tiempo de sesión" y "tiempo de lección") aparecen con valores 0.0. Luego, verificamos la ausencia de valores nulos y finalmente se concatenan tres DataFrames: uno cualitativo con valores nulos reemplazados por "Sin Dato" y dos copias de data3\_igr.

```
valores_nulos4=data5_igr.isnull().sum()
valores_nulos4

[ ]

... tiempo de sesión      0
    tiempo de lección     0
    dtype: int64
```

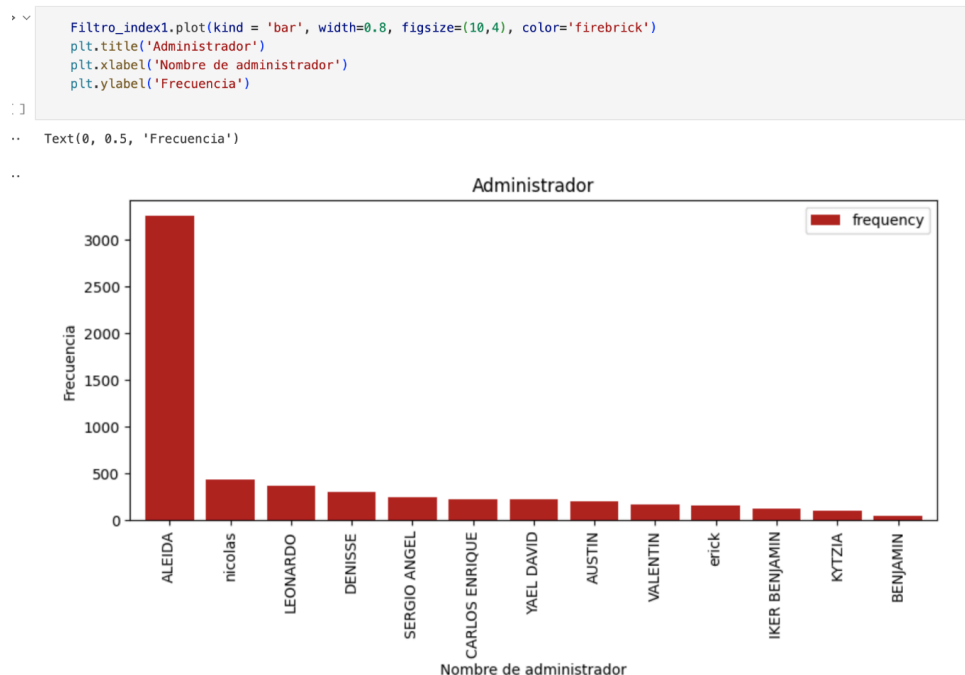
Posterior confirmamos si los nulos se han ido correctamente, y efectivamente ya no había ni uno:

```
valores_nulos5=data_final.isnull().sum()
valores_nulos5
```

```
Administrador      0
Usuario            0
mini juego         0
color presionado   0
dificultad         0
fecha              0
Juego              0
botón correcto     0
tiempo de interacción 0
número de interacción 0
auto push          0
tiempo de sesión   0
tiempo de lección  0
dtype: int64
```

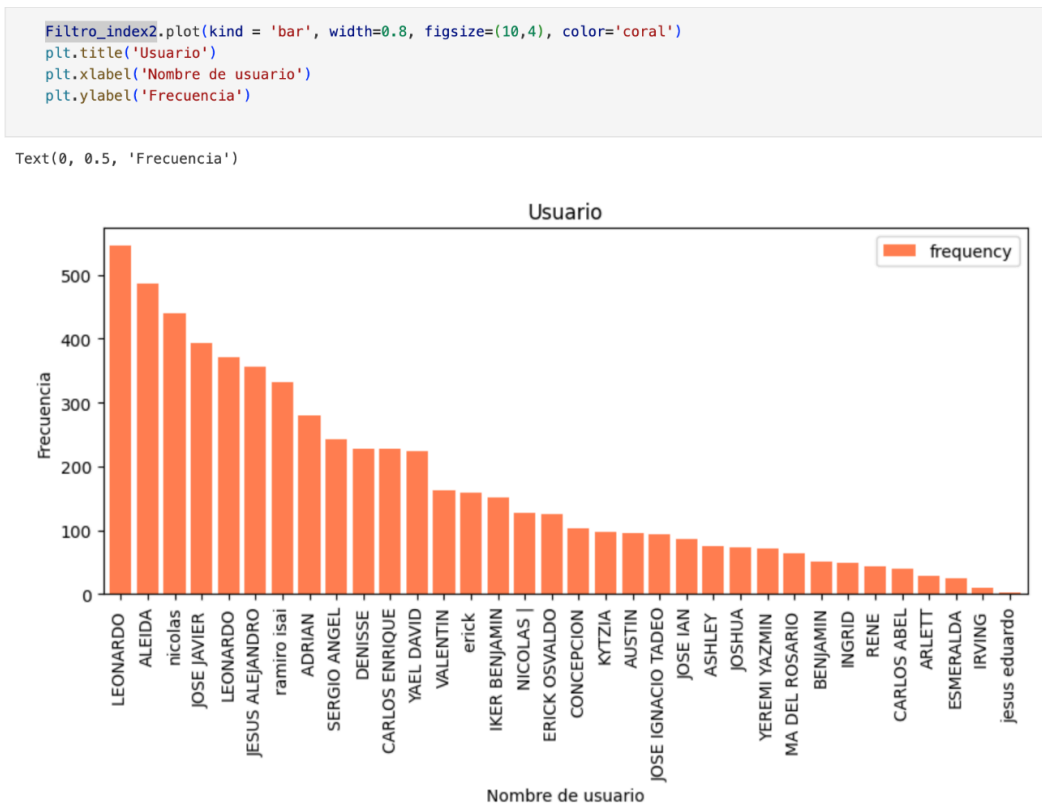
Después generamos una tabla de frecuencias a partir de datos agrupados por la columna 'Administrador'. Primero, creamos la tabla (table1), luego eliminamos las columnas de porcentajes (table2), y finalmente establecimos 'Administrador' como índice (Filtration). El resultado es una lista de administradores con sus respectivas frecuencias (conteos de apariciones), donde "ALEDA" tiene la mayor frecuencia (3260) y "BENJAMIN" la menor (51), mostrando los datos ordenados de mayor a menor frecuencia.

De ahí, se generó un gráfico de barras a partir de un DataFrame o Serie de pandas llamado Filtro\_index1, mostrando la frecuencia (cantidad de ocurrencias) de distintos administradores. Aquí está el desglose:



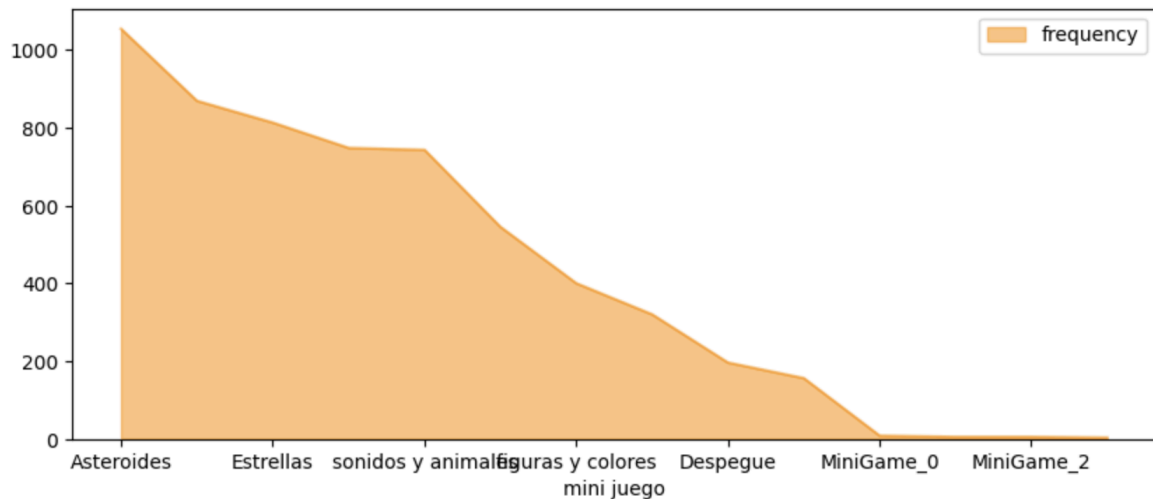
El gráfico muestra la distribución de administradores, donde cada barra corresponde a un nombre y su altura refleja su frecuencia.

Se creó un gráfico de barras a partir del DataFrame `Filtro_index2` usando `matplotlib`, configurando barras de color coral con un ancho de 0.8 y un tamaño de figura de 10x4 pulgadas.

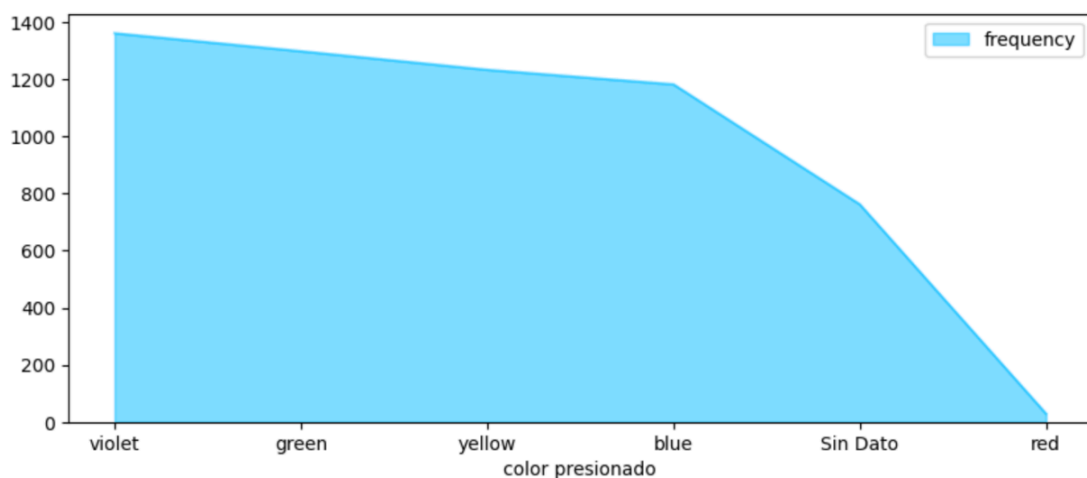


El gráfico muestra la frecuencia de cada usuario, con el título "Usuario", el eje X etiquetado como "Nombre de usuario" y el eje Y como "Frecuencia", proporcionando una visualización clara de la distribución de datos.

Posterior se creó un gráfico que muestra un diagrama de áreas apiladas en tono naranja oscuro semitransparente, representando la frecuencia de uso de distintos mini-juegos (como "Asteroides" y "MiniGame\_0") en un rango de 0 a 1000. El eje X etiqueta los nombres de los juegos.



Se generó después un gráfico donde se representa la frecuencia con que se presionaron diferentes colores, donde el violeta (1391) y verde (1298) son los más frecuentes, seguidos por amarillo (1233), azul (1162), "Sin Dato" (762) y rojo (29). El gráfico, usa un formato de área apilada para comparar visualmente las frecuencias, con los colores en el eje X y los valores numéricos (0-1400) en el eje Y.

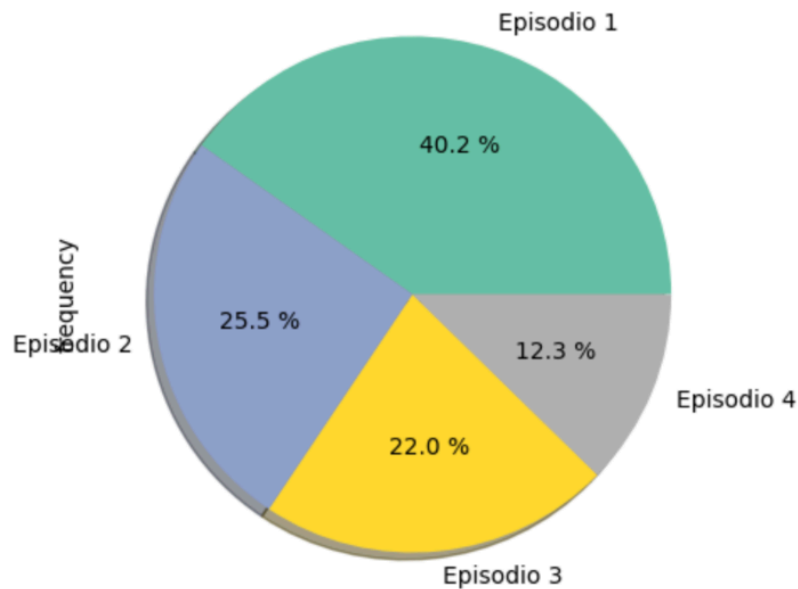


Después, se creó otro gráfico de pastel que muestra la distribución porcentual de cuatro niveles de dificultad (Episodic 1 al 4) en un conjunto de datos.

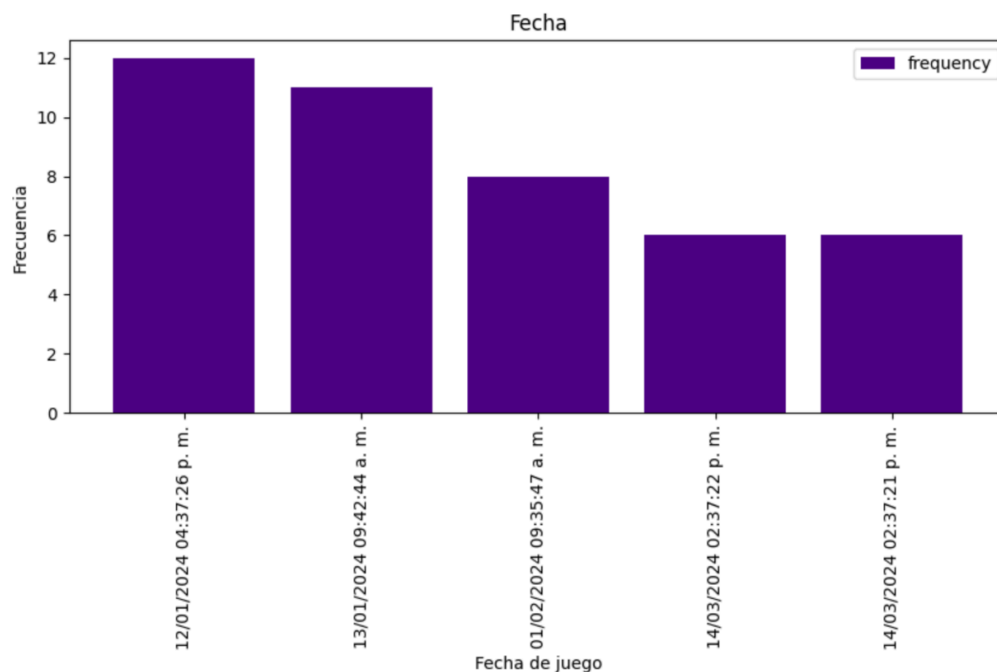
El nivel Episodic 1 es el más frecuente con 2357 casos (40.2%), seguido por Episodic 2 (1493, 25.5%), Episodic 3 (1282, 22%) y Episodic 4 (723, 12.3%).

El gráfico destaca claramente las proporciones relativas de cada categoría, mostrando los valores porcentuales.

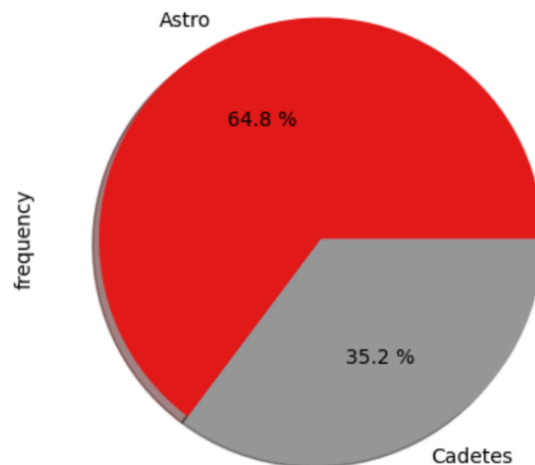




Después mostramos los registros filtrados de frecuencias por fecha (solo valores >5), donde se observan fechas como "12/01/2024" con 12 repeticiones, y posterior a ello, se generó el gráfico de barras moradas a partir de esos datos, con ejes etiquetados ("Fecha de juego" vs "Frecuencia"), que visualiza cómo la fecha "12/01/2024 04:37:26 p.m." tuvo el pico máximo (12 registros), destacando los días con mayor actividad.



Y después a ello, se realizó un gráfico de pastel que compara la participación de dos juegos, Astro (64.8%, 3,799 registros) y Cadetes (35.2%, 2,086 registros), generado a partir de una tabla de frecuencias. Este gráfico se visualiza con porcentajes precisos y destaca claramente la predominancia de Astro, que representa casi dos tercios de los registros, logrando una comparación de forma visual inmediata entre ambos juegos.



### Categorización de la Variable "Tiempo de Interacción"

Para facilitar el análisis e interpretación de los datos, realizamos una categorización de la variable cuantitativa "tiempo de interacción". Este procedimiento nos permite agrupar los datos numéricos en intervalos definidos, transformándolos en una variable categórica más fácil de estudiar mediante frecuencias o gráficos.

#### 1. Cálculo de valores mínimos y máximos

Primero, identificamos los valores extremos de la variable:

```
Max=data_final['tiempo de interacción'].max()
Min=data_final['tiempo de interacción'].min()
Límites= [Min, Max]
Límites
```

[1642]

... [0.0, 21.78444]

Python

Esta etapa es fundamental para asegurarnos de que todos los valores estén contemplados dentro de los intervalos que se construirán más adelante.

## 2. Definición de intervalos y categorías

Una vez conocidos los límites, establecimos los intervalos de tiempo que permitirían clasificar los datos en distintos niveles de interacción. Estos intervalos no fueron generados automáticamente, sino definidos manualmente con base en criterios prácticos, lo que permitió asignar etiquetas significativas.

```
intervalos=np.linspace(0.0000, 21.78444, 4)
intervalos

array([ 0.        ,  7.26148, 14.52296, 21.78444])

#Creamos las categorías
categorias= ["Bajo 0-7", "Medio 7-14", "Alto 14-21"]
```

## 3. Transformación de la variable numérica en categórica

Con los intervalos y etiquetas definidos, procedimos a aplicar la función `pd.cut()` de la librería `pandas`, que permite convertir una columna numérica en una categórica con base en los rangos establecidos:

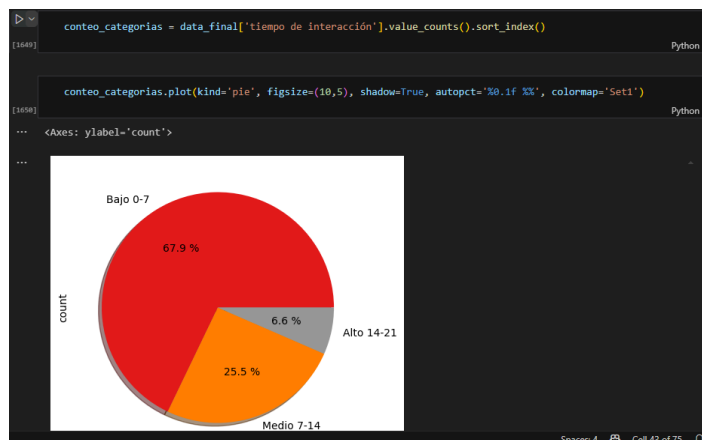
```
#Finalmente creamos las categorías en la columna numérica
data_final['tiempo de interacción']=pd.cut(x= data_final['tiempo de interacción'], bins=intervalos, labels=categorias)
data_final['tiempo de interacción']
```

5723	Medio 7-14
5724	Medio 7-14
5725	Medio 7-14
5726	Bajo 0-7
5727	Bajo 0-7
5728	Bajo 0-7
5729	Bajo 0-7
5730	Medio 7-14
5731	Medio 7-14
5732	Bajo 0-7
5733	Bajo 0-7
5734	Bajo 0-7
5735	Bajo 0-7
5736	Medio 7-14
5737	Medio 7-14

Este paso realiza automáticamente la asignación de etiquetas a cada fila del dataset, dependiendo del valor que tenga en la variable "tiempo de interacción". De este modo, cada usuario quedó clasificado en uno de los cinco niveles definidos.

## 4. Conteo de frecuencias por categoría

Finalmente, para analizar la distribución de los datos resultantes, calculamos la cantidad de observaciones que pertenecen a cada categoría:



Este conteo nos permite visualizar cuántos usuarios se ubican en cada nivel de interacción, lo que a su vez ofrece una visión más clara del comportamiento general de los usuarios dentro de la plataforma educativa.

Este proceso de categorización no sólo mejora la comprensión del fenómeno estudiado, sino que también nos facilita la creación de visualizaciones más claras, como gráficos de barras o pastel, que permiten comunicar los resultados a audiencias no técnicas.

## Categorización de la Variable "Tiempo de Lección"

Al igual que con el tiempo de interacción, la variable tiempo de lección representa una medición continua que indica cuánto tiempo dedicaron los usuarios a estudiar o completar lecciones dentro de la plataforma. Para facilitar su análisis y comparación entre diferentes grupos de usuarios, esta variable fue transformada en una variable categórica mediante un proceso de clasificación por intervalos.

### 1. Identificación del rango de valores

Primero determinamos los valores extremos (mínimo y máximo) de la variable, con el fin de comprender la dispersión de los datos y definir límites adecuados para los intervalos:

```
Max1=data_final['tiempo de lección'].max()
Min1=data_final['tiempo de lección'].min()
Limites1= [Min1, Max1]
Limites1
```

[1652] ... [0.0, 117.8544] Python

Esto nos permitió obtener una visión clara de los valores más bajos y más altos registrados por los usuarios en esta dimensión.

## 2. Establecimiento de intervalos y categorías

Después, definimos manualmente los intervalos que agrupan el tiempo de lección en diferentes niveles. A cada intervalo se le asignó una etiqueta representativa que facilita la interpretación:

```
intervalos1 = np.linspace(-0.1, 117.8544, 4)
np.set_printoptions(suppress=True)
intervalos1
```

[1656] ... array([ -0.1 , 39.21813333, 78.53626667, 117.8544 ])

```
#Creamos las categorías
categorias1 = ["Bajo 0-39", "Medio 39-78", "Alto 78-118"]
```

[ ] Python

Estas categorías nos permiten resumir el comportamiento de los usuarios respecto a su tiempo dedicado al estudio de lecciones.

## 3. Transformación de datos numéricos a categorías

Utilizando `pd.cut()`, se asignaron las etiquetas correspondientes a cada observación del conjunto de datos:

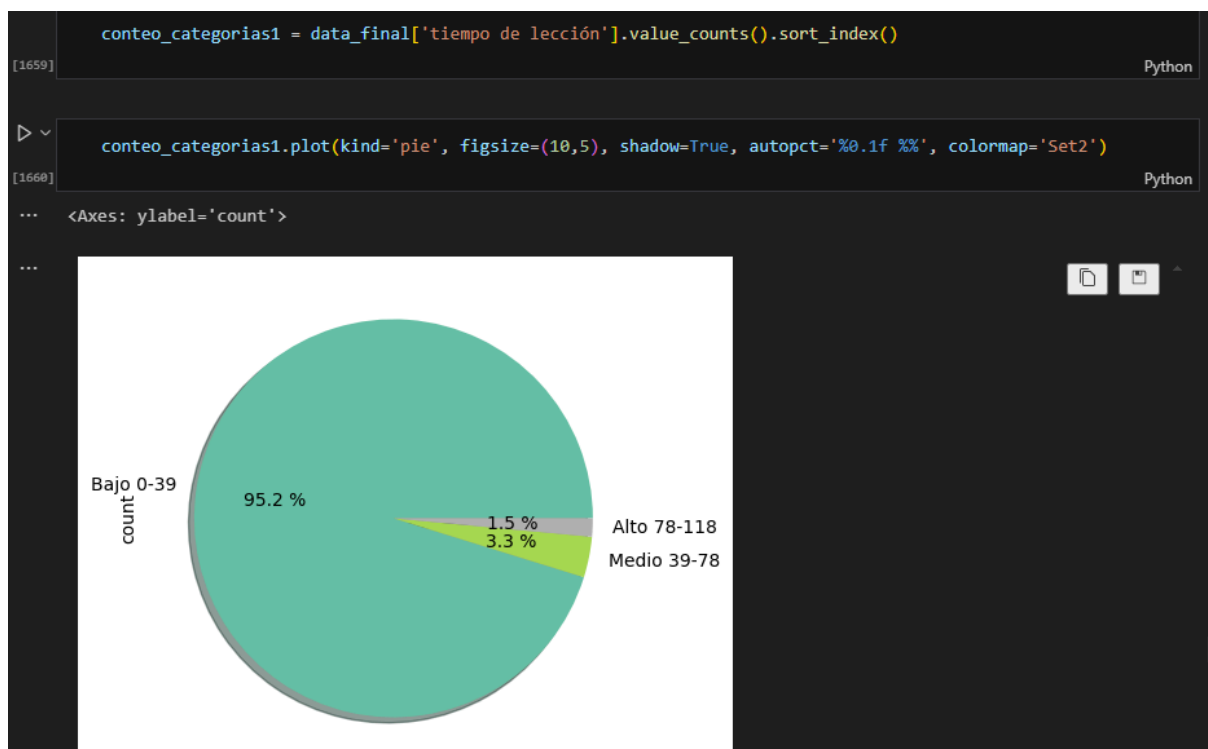
```
#Finalmente creamos las categorías en la columna numérica
data_final['tiempo de lección'] = pd.cut(x=data_final['tiempo de lección'], bins=intervalos1, labels=categorias)
data_final['tiempo de lección']
```

5671	Bajo 0-39
5672	Bajo 0-39
5673	Bajo 0-39
5674	Alto 78-118
5675	Bajo 0-39
5676	Bajo 0-39
5677	Bajo 0-39
5678	Bajo 0-39
5679	Bajo 0-39
5680	Bajo 0-39
5681	Bajo 0-39
5682	Bajo 0-39
5683	Alto 78-118
5684	Bajo 0-39
5685	Bajo 0-39
5686	Bajo 0-39
5687	Bajo 0-39
5688	Bajo 0-39
5689	Bajo 0-39
5690	Alto 78-118
5691	Bajo 0-39
5692	Bajo 0-39

Esta transformación convierte los valores numéricos continuos en una variable categórica, facilitando su análisis a través de frecuencias o gráficos.

#### 4. Análisis de la distribución por categoría

Finalmente, se contó cuántos usuarios quedaron en cada categoría de tiempo de lección:



Esto nos permitió identificar qué tan común era cada nivel de dedicación y hacer comparaciones entre los grupos.

## Categorización de la Variable "Tiempo de Sesión"

La variable tiempo de sesión refleja el tiempo total que un usuario permanece conectado en la plataforma. Esta métrica también fue categorizada para facilitar su interpretación, siguiendo el mismo procedimiento que en los casos anteriores.

### 1. Obtención de valores extremos

Calculamos los valores mínimo y máximo de esta variable para determinar el rango total de tiempos registrados:

```
Max2=data_final['tiempo de sesión'].max()
Min2=data_final['tiempo de sesión'].min()
Límites2= [Min2, Max2]
Límites2
```

[1662] Python

... [0.0, 294.3935]

Esto nos permitió tener una base sólida para definir los intervalos de categorización.

### 2. Definición de intervalos y etiquetas

Al igual que en los casos anteriores, agrupamos los datos en cinco categorías, definidas de forma manual para reflejar distintos niveles de uso de la plataforma:

```
intervalos3 = np.linspace(-0.1, 294.3935, 4)
np.set_printoptions(suppress=True)
intervalos3
```

[1666] Python

... array([-0.1, 98.0645, 196.229, 294.3935])

```
#Creamos las categorías
categorías3= ["Bajo 0-98", "Medio 98-196", "Alto 196-294"]
```

[1667] Python

+ Code + Markdown

### 3. Conversión de variable continua a categórica

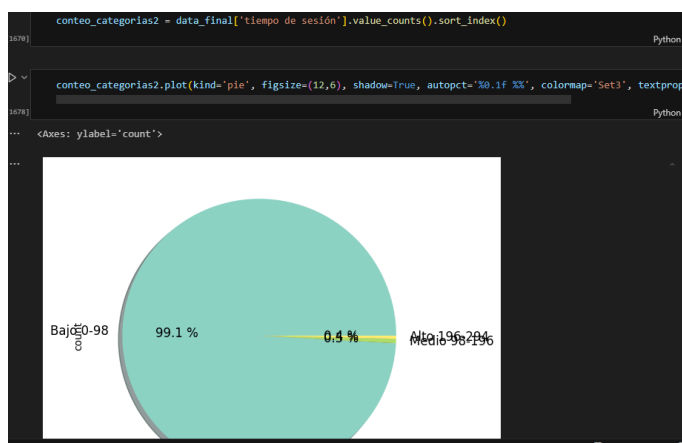
Utilizamos la función `pd.cut()` para asignar a cada observación una categoría específica según el intervalo en el que se encontraba:

```
#Finalmente creamos las categorías en la columna numérica
data_final['tiempo de sesión']=pd.cut(x= data_final['tiempo de sesión'], bins=intervalos3, labels=categorias3,
data_final['tiempo de sesión']

Python
5779      Bajo 0-98
...
5780      Bajo 0-98
5781      Bajo 0-98
5782      Bajo 0-98
5783      Bajo 0-98
5784      Bajo 0-98
5785      Bajo 0-98
5786      Bajo 0-98
5787      Bajo 0-98
5788      Bajo 0-98
5789      Bajo 0-98
5790      Bajo 0-98
5791      Bajo 0-98
5792      Bajo 0-98
5793      Bajo 0-98
5794      Bajo 0-98
5795      Bajo 0-98
5796      Bajo 0-98
5797      Bajo 0-98
5798      Bajo 0-98
5799      Bajo 0-98
5800      Bajo 0-98
```

#### 4. Conteo de frecuencias de las categorías

Finalmente, analizamos la distribución de los datos por categoría para comprender mejor el comportamiento general de los usuarios:



Este análisis nos resultó útil para identificar los patrones de permanencia dentro de la plataforma, revelando qué tan comprometidos están los usuarios en términos de tiempo de conexión.



## **Conclusión**

Este análisis de datos permitió comprender de manera más clara el comportamiento de los usuarios, identificando patrones en variables como el tiempo de sesión, tiempo de lección y tiempo de interacción. A través de un proceso de limpieza de datos, detección y tratamiento de valores atípicos, así como la imputación de valores nulos, se garantizó una base de datos más estructurada y confiable. La categorización de las variables facilitó su interpretación y permitió segmentar a los usuarios en distintos grupos según su nivel de actividad. Además, las visualizaciones generadas ayudaron a identificar tendencias clave, proporcionando información valiosa para la toma de decisiones y posibles mejoras.