Actividad R.1: Exploratory Data Analysis

Inteligencia artificial avanzada para la ciencia de datos I (Gpo 102)
Campus Guadalajara

Héctor Manuel Cárdenas Yáñez | A01634615
Siddhartha López Valenzuela | A00227694
Álvaro Morán Errejón | A01638034
Isaí Ambrocio | A01625101

26/08/2023

Actividad R.1: Exploratory Data Analysis

Las librerías que fueron utilizadas para los fines de investigación y análisis de los datos fueron numpy, pandas, matplotlib.pyplot y seaborn.

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

sns.set_theme()
```

Comenzamos analizando los datos de un solo DataFrame debido a la gran cantidad de datos en los 15 csv. Previamente importamos los datos de todos los archivos solo en distintos DataFrames, pero, para fines de esta actividad se documentó el análisis del "df501". Por medio de la descripción de los datos podemos observar la media, mínimo, desviación estándar, máximo y sus respectivos cuartiles (25%, 50% y 75%).

```
df = df501
df.describe()
```

	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	label
count	103860.000000	103860.000000	103860.000000	103860.000000	103860.000000	103860.000000	103860.000000
mean	-0.880883	-0.069506	-0.003201	-0.794307	-0.095594	-0.318354	3.237069
std	0.340989	0.128599	0.417656	0.599406	0.210591	0.547369	2.860101
min	-2.895264	-1.157471	-2.204834	-7.942139	-3.600830	-4.193604	1.000000
25%	-1.015625	-0.137939	-0.268066	-1.003174	-0.187256	-1.046875	1.000000
50%	-0.920898	-0.062500	-0.136963	-0.937500	-0.080322	-0.131348	1.000000
75%	-0.767517	-0.006836	0.234375	-0.203125	-0.015625	0.080078	6.000000
max	0.239746	0.893555	0.979004	1.159424	3.234863	1.861572	8.000000

Al inspeccionar los datos implementando la función dtypes pudimos observar que la mayoria de los datos son de tipo flotante a excepción de label, siendo de tipo integer. Las etiquetas de las variables que mencionamos son: walking, shuffling, stairs (ascending), stairs (descending), standing, sitting, lying. Es importante mencionar que no existe la etiqueta número dos en la base de datos que obtuvimos de Kaggle.

```
df.dtypes
df['label'].unique()
```

```
timestamp     object
back_x       float64
back_y       float64
back_z       float64
thigh_x      float64
thigh_y      float64
thigh_z      float64
label          int64
dtype: object
```

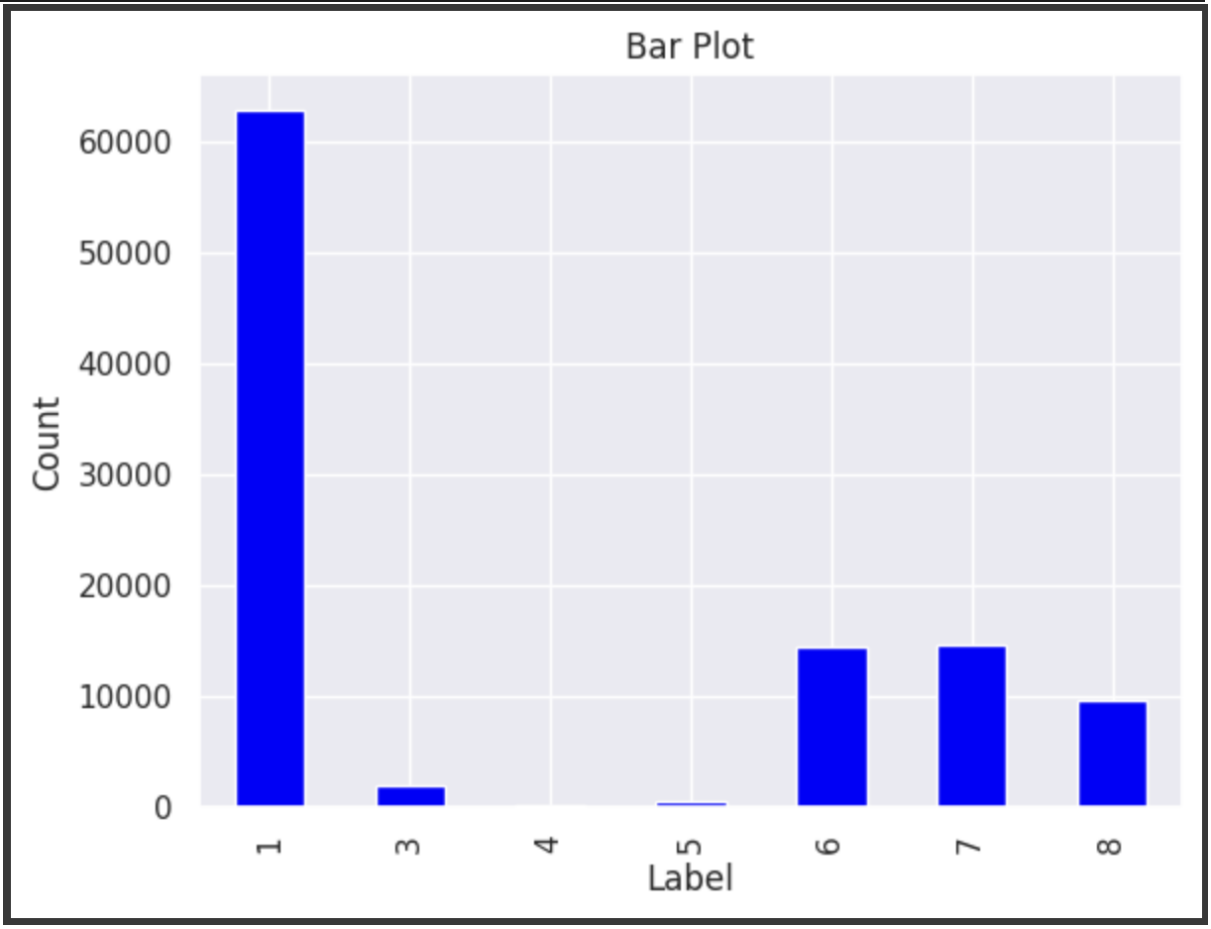
```
array([6, 3, 1, 7, 8, 5, 4])
```

Las etiquetas son las siguientes(La etiqueta 2 no aparece en Kaggle):

- 1: walking
- 3: shuffling
- 4: stairs (ascending)
- 5: stairs (descending)
- 6: standing
- 7: sitting
- 8: lying

En la siguiente gráfica se muestra un time series para cada variable. Es preciso resaltar que esto es solo una pequeña fracción del dataset. Ya que si utilizamos el dataset completo la gráfica se vuelve muy grande y el tiempo de carga aumenta. Este gráfico nos ayuda a analizar cuáles etiquetas son las que más se repiten. También de primera mano podemos observar que la mayoría de los datos pertenecen a la primera etiqueta “walking”.

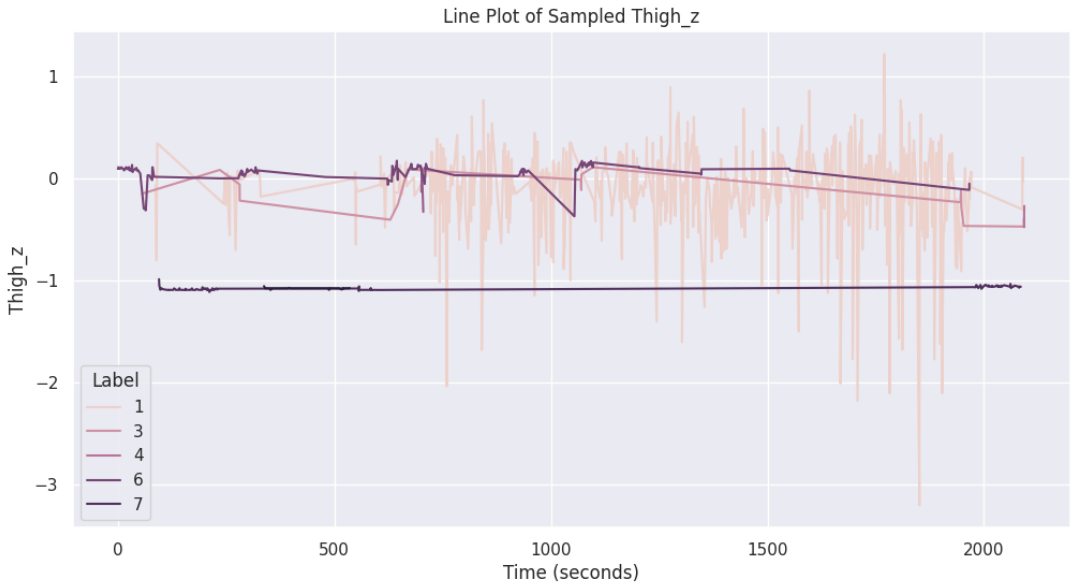
```
df['label'].value_counts().sort_index().plot(kind='bar', color='blue')
plt.xlabel('Label')
plt.ylabel('Count')
plt.title('Bar Plot')
plt.show()
```

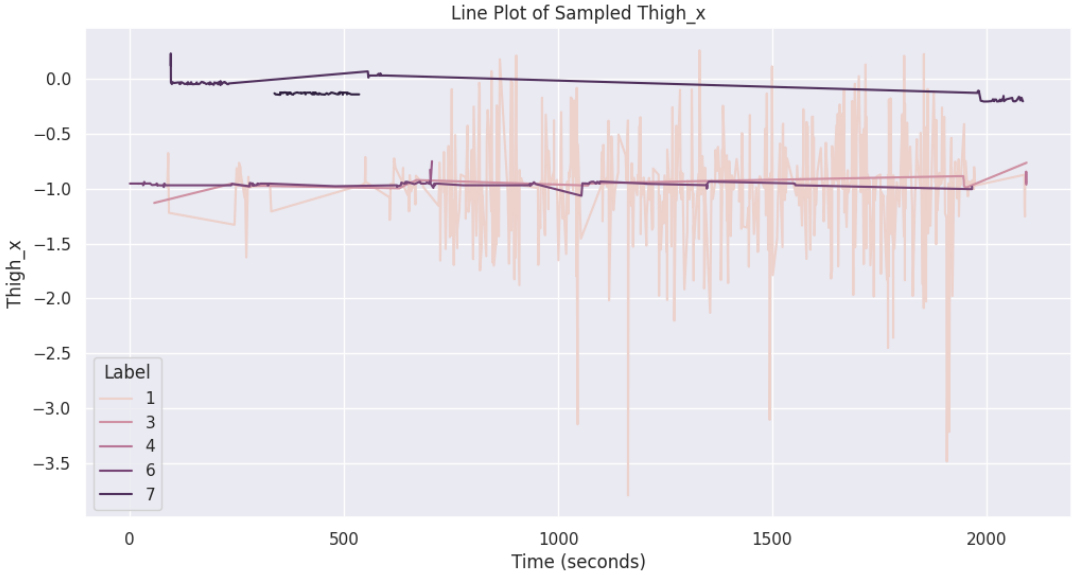


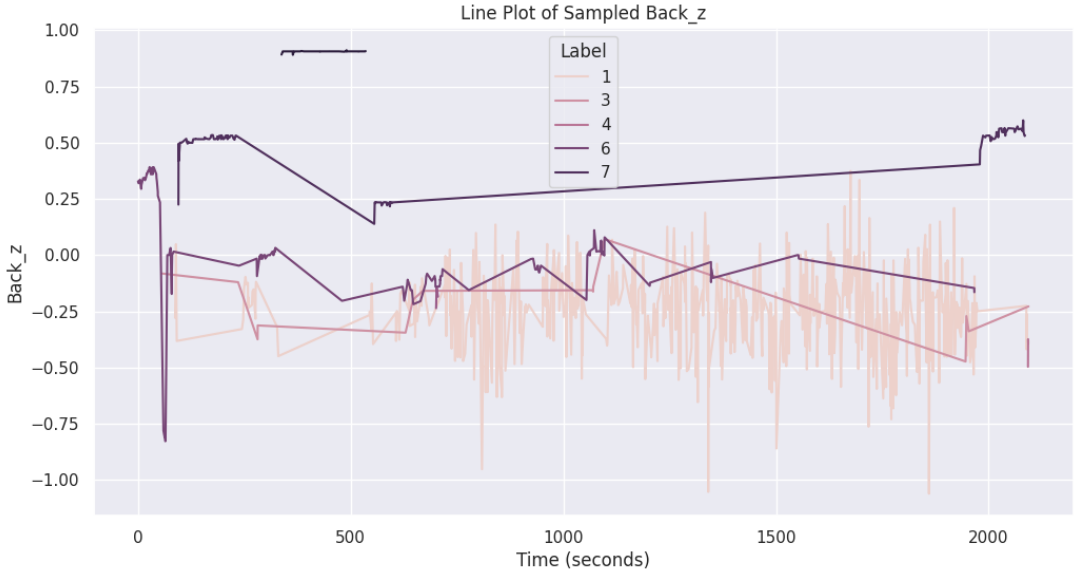
El siguiente código y sus gráficas nos proporcionan un analisis más detallado del comportamiento de todas la columnas del DataFrame (back_x, back_y, back_z, thigh_z, thigh_y, thigh_z) para cada una de las etiquetas con respecto del tiempo. La gráficas dibujan una distinción clara entre las etiquetas, tanto con respecto de la duración cómo del impacto o esfuerzo. Analizando cómo están distribuidos los datos y la cantidad de los mismos. Con ello podemos crear una hipótesis inicial de cuáles datos son más relevantes o cuales deberían de tener mayor impacto para poder crear un modelo que proporcione un análisis concreto.

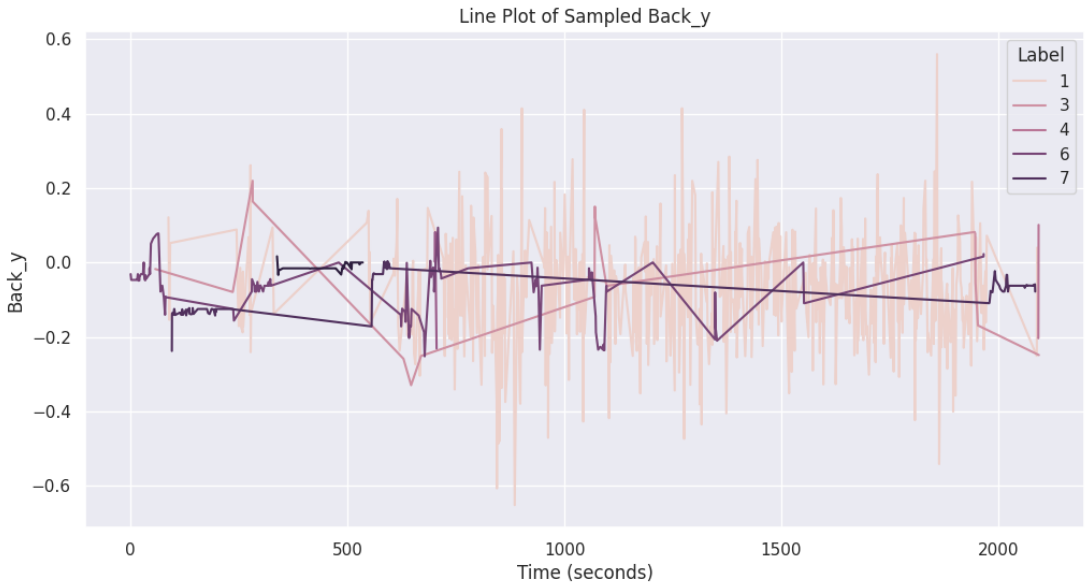
Es importante mencionar que también es una fracción de la información. De analizar más datos y concatenar los 15 archivos csv se debería cambiar la variable de frac para ajustar el modelo.

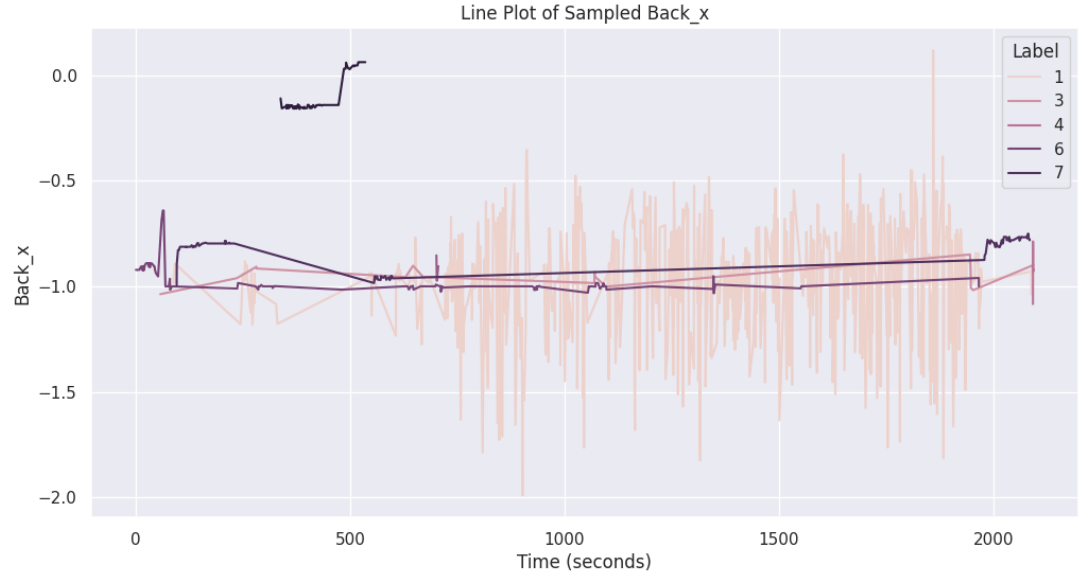
```
###SOLO CORRER ESTE CODIGO SI EL DATASET ESTA PEQUEÑO (QUE NO ESTEN LOS
15 CSV CONCATENADOS). DE LO CONTRARIO CAMBIAR LA VARIABLE "FRAC"
df['timestamp'] = pd.to_datetime(df['timestamp'])

# muestra de datos usada
sampled_df = df.sample(frac=0.01) # Adjust the fraction as needed

# cambiar a segundos el timestamp de inicio a fin
sampled_df['timestamp_seconds'] = (sampled_df['timestamp'] -
sampled_df['timestamp'].min()).dt.total_seconds()

# line plots para cada columna
variables_to_plot = ['back_x', 'back_y', 'back_z', 'thigh_x',
'thigh_y', 'thigh_z']

plt.figure(figsize=(12, 6))

for variable in variables_to_plot:
plt.figure(figsize=(12, 6))
sns.lineplot(data=sampled_df, x='timestamp_seconds', y=variable,
hue='label')
plt.xlabel('Time (seconds)')
plt.ylabel(f'{variable.capitalize()}')
plt.title(f'Line Plot of Sampled {variable.capitalize()}')
plt.legend(title='Label')
plt.show()
```

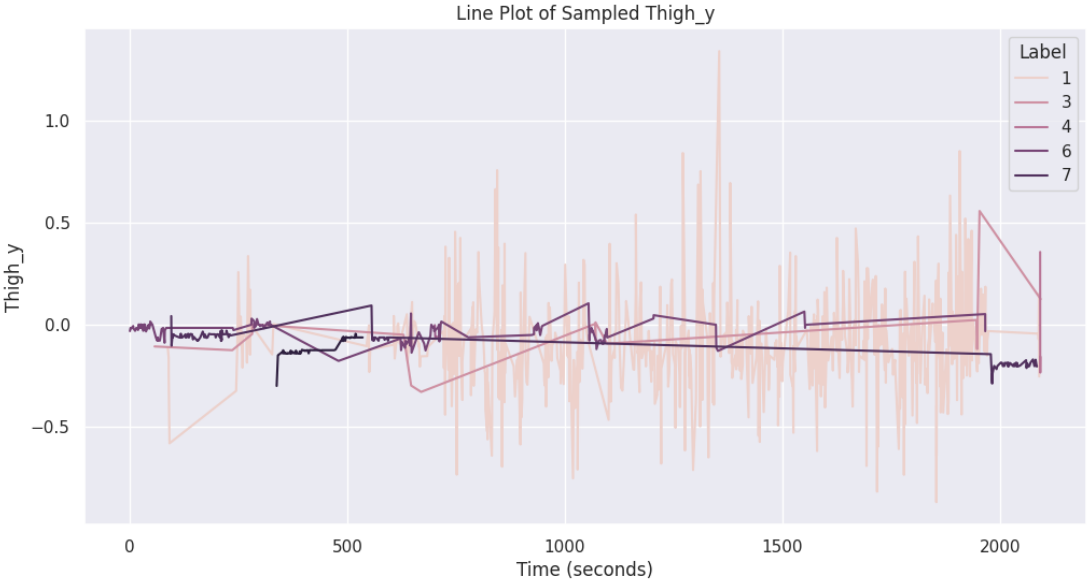

Line Plot of Sampled Thigh_z
Thigh_z
Time (seconds)
Label
1
3
4
6
7


Line Plot of Sampled Thigh_x
Thigh_x
Time (seconds)
Label
1
3
4
6
7


Line Plot of Sampled Back_z
Back_z
Time (seconds)
Label
1
3
4
6
7

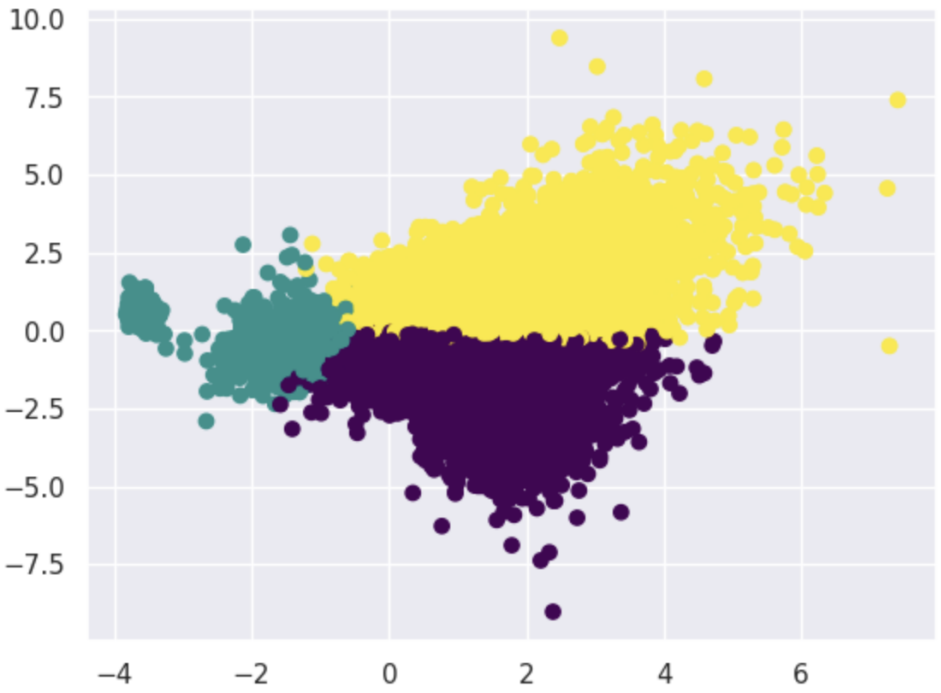

Line Plot of Sampled Back_y
Back_y
Time (seconds)
Label
1
3
4
6
7


Line Plot of Sampled Back_x
Back_x
Time (seconds)
Label
1
3
4
6
7


Line Plot of Sampled Thigh_y
Thigh_y
Time (seconds)
Label
1
3
4
6
7

Para finalizar, implementamos un data clustering para poder analizar cómo están organizados los datos, que tendencias tienen. Podemos observar claramente que se dividen en tres mayores distinciones. También podemos asumir que existe una mayor cantidad o mayor dispersión entre ciertas variables. Lo cual debemos tomar en cuenta al evaluar los diferentes modelos para poder elegir el mejor para este caso.

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

clusters = ['back_x', 'back_y', 'back_z', 'thigh_x', 'thigh_y',
'thigh_z']
df_clusters = df[clusters]
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_clusters)

# hacer el elbow method o cualquier otro metodo para determinar
num_clusters, falta!!!
num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
cluster_labels = kmeans.fit_predict(scaled_data)

pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

plt.scatter(pca_data[:, 0], pca_data[:, 1], c=cluster_labels,
cmap='viridis')
plt.show()
```



Conclusión Individual

Gracias al análisis exploratorio de los datos pudimos tener un mayor acercamiento a la información contenida en el data set y así poder comprender de mejor manera los mismos. Algo que pudimos observar es que en el primer DataFrame no hay datos faltantes, esto nos facilitará más el proceso, ya que no tendremos que hacer imputaciones; sin embargo, tenemos que hacer lo correspondiente con los catorce DataFrames restantes. Del mismo modo, al tener un pequeño panorama de la cantidad de etiquetas que más se repiten, podemos observar y darnos una idea de por qué hay una gran cantidad de etiquetas inclinada a cuatro en específico. Esto nos permite abordar de mejor manera el cómo hacer los análisis futuros y también, hacer las modelaciones pertinentes.