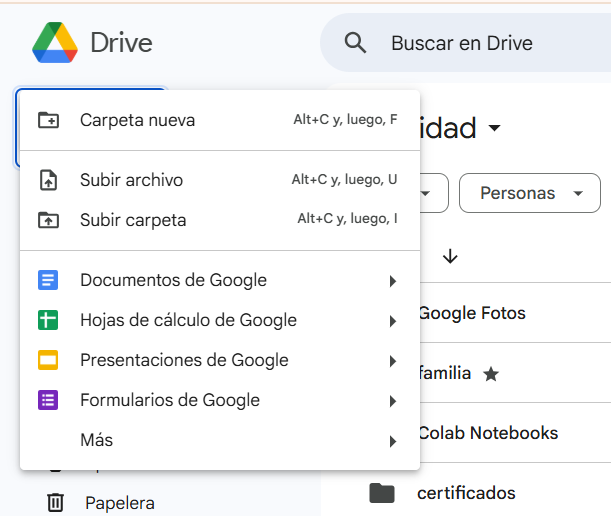
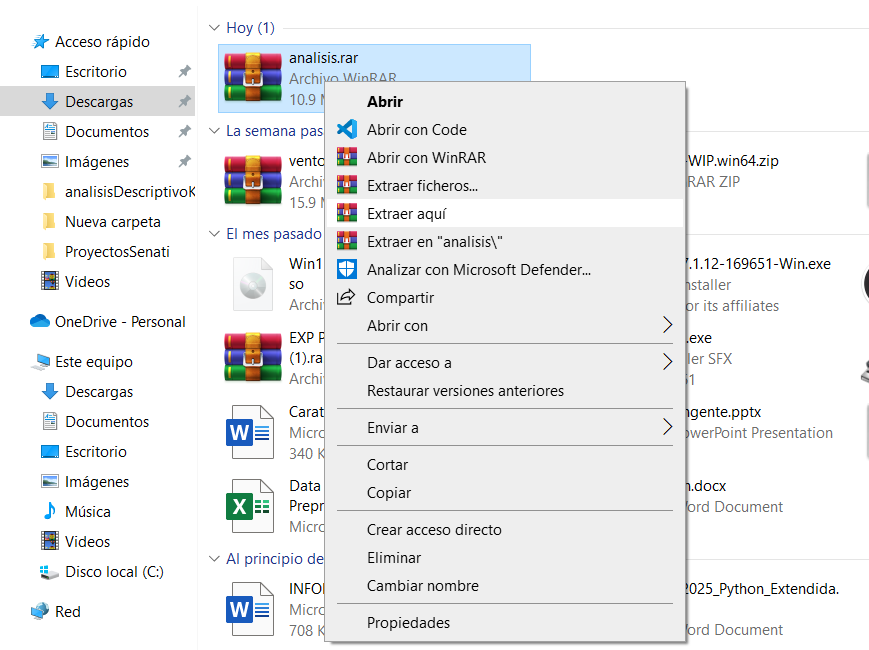
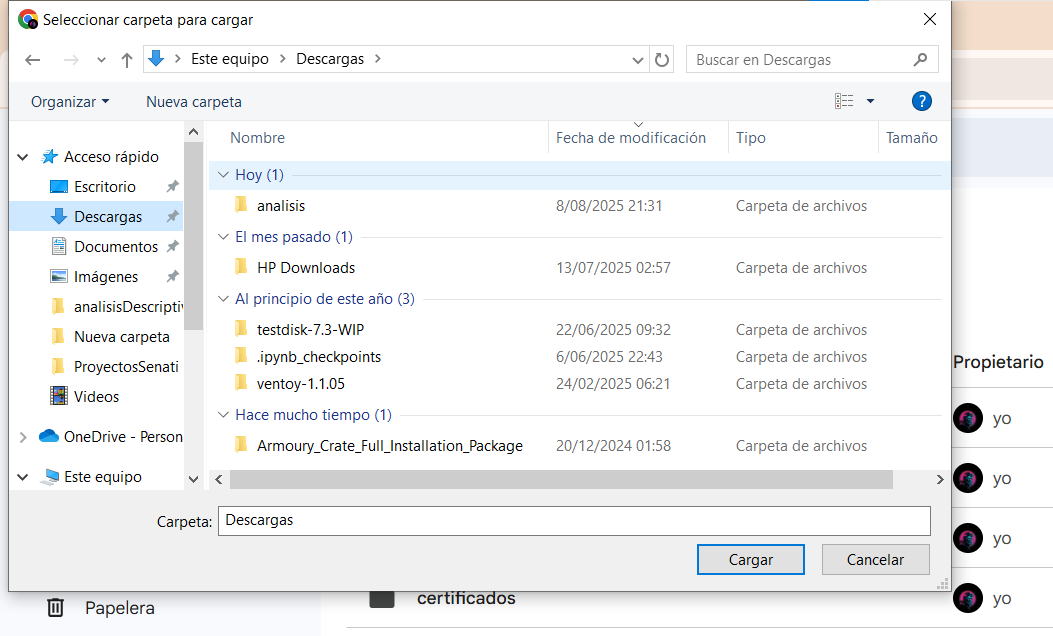
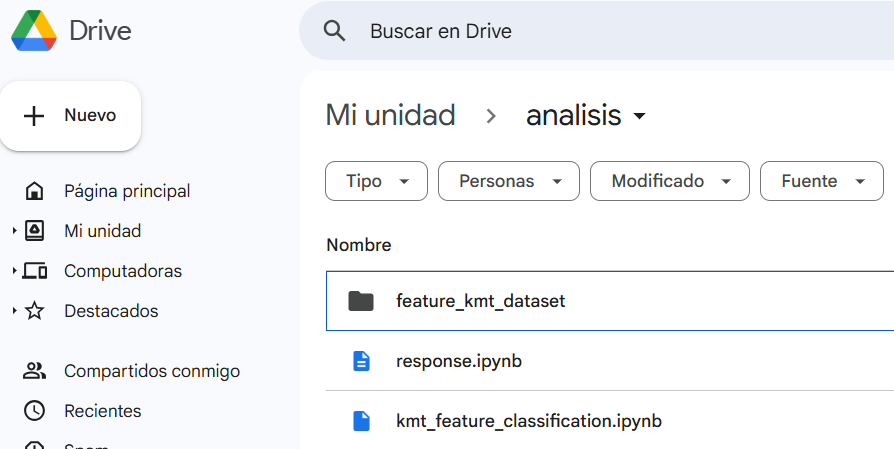
# Obtención de gráficos con Python, paso a paso.

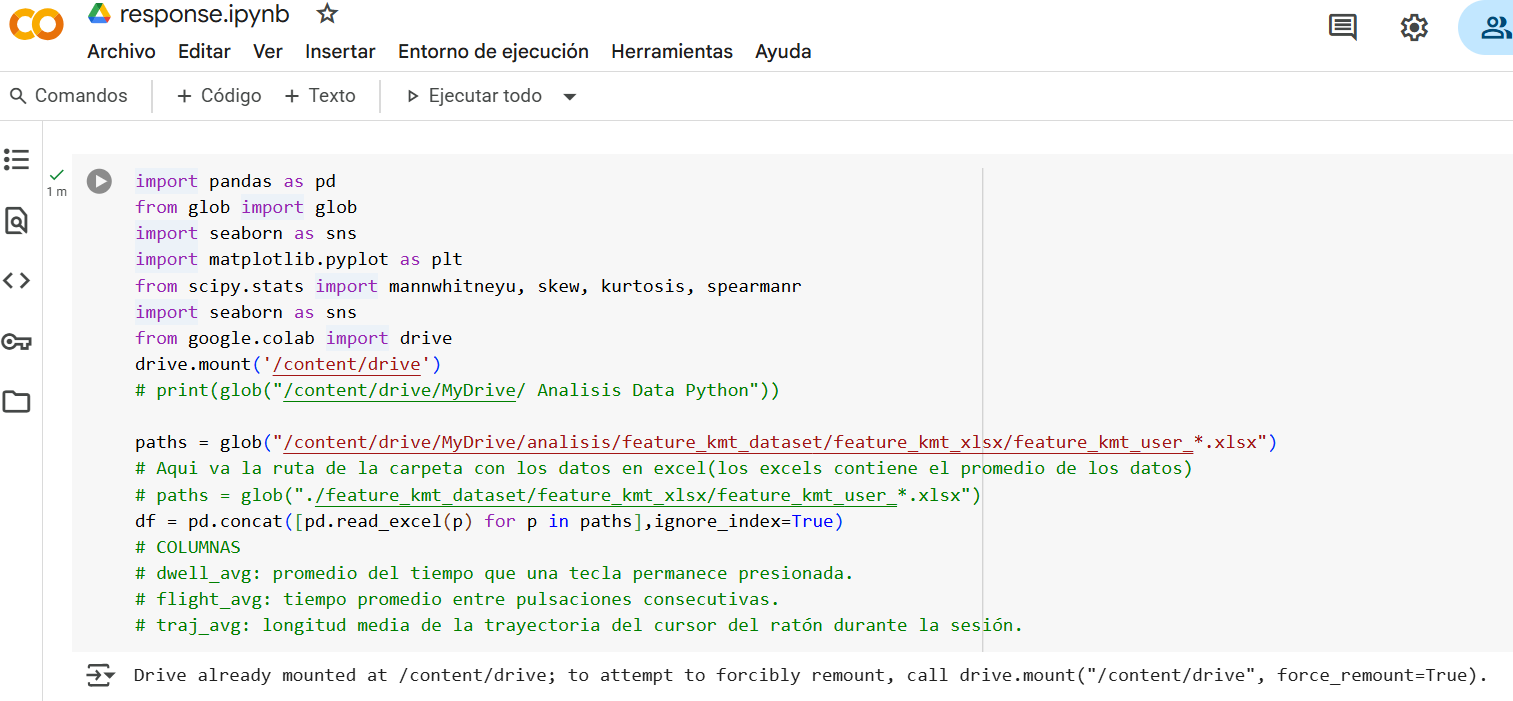
1. Descomprimir el archivo proporcionado, ingrese a su drive, seleccione y cargue la carpeta analisis



Ingresar a ese archivo

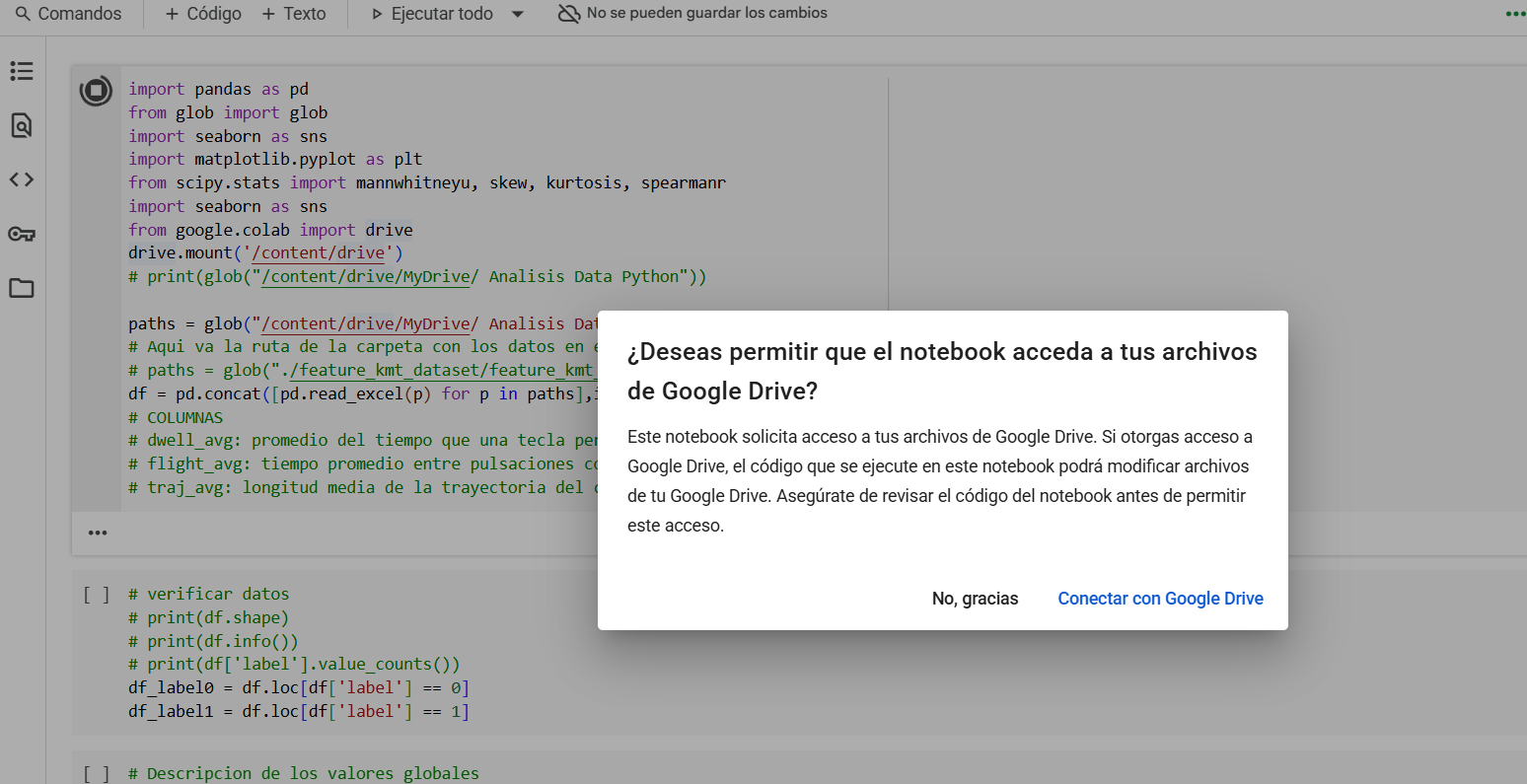


1. **Primera celda:** carga las librerías necesarias para hacer el análisis, las librerías usadas son:
   1. Pandas, Biblioteca principal para manipulación y análisis de datos en Python.
   2. Seaborn: Biblioteca para visualización estadística de alto nivel; facilita histogramas, boxplots, pairplots, etc. Se integra con pandas y matplotlib.
   3. Matplotlib: Biblioteca base para gráficos en Python.
   4. scipy.stats: Funciones estadísticas:
      1. mannwhitneyu: prueba no paramétrica para comparar dos distribuciones (como alternativa a t-test si no hay normalidad).
      2. skew: asimetría de la distribución (si cola está a la derecha/izquierda).
      3. kurtosis: curtosis (concentración/colas; ojo con la definición — ver nota más abajo).
      4. spearmanr: correlación de rango de Spearman (no paramétrica, mide relación monotónica).

****

Presione play para cargar los archivos y ejecutar el primer script

1. Si Google le pide permisos del drive, concédalos. Presione Conectar con Google Drive

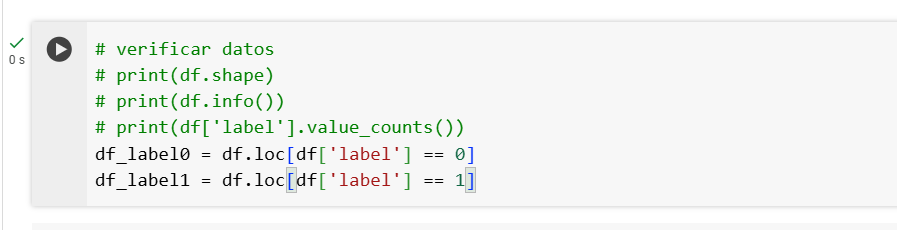


1. En la primera celda, esto crea un DataFrame para manejar toda la data y hacer mas sencillo el análisis, todo se almacena en la variable ‘df ’

df = pd.concat([pd.read\_excel(p) for p in paths],ignore\_index=True)

1. Aqui se verifico los datos, si desea probarlo, borre ‘#’ y ejecute la celda presionando play.

Tambien se separo en 2 variables los datos de usuarios legitimos e ilegitimos para posteriores análisis

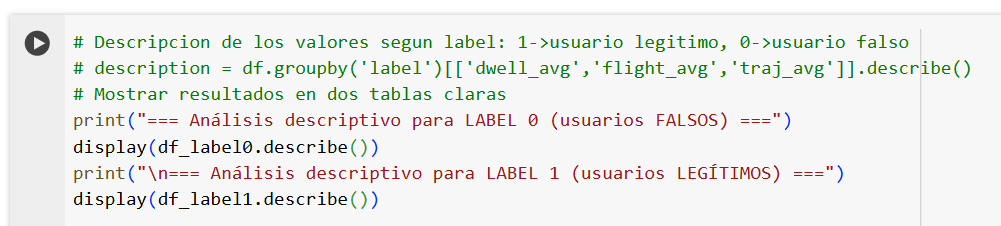


1. En esta celda se hace una descripción de los datos(ignorar unnamed, count y label). Se obtiene la media, desviación estándar, mínimo, máximos y los cuartiles. Al ejecutar la celda, debería salir una tabla igual a esta, recuerde que las celdas se ejecutando dando botón de **play**:



El método **.describe()** genera la descripción de la data

1. Aqui se hace una descripción de cada data separado por usuario legítimos e ilegítimos, se obtiene 2 tablas



1. Aqui se obtiene 3 histogramas(histplot) que están separados por clase para comparaciones entre usuarios legitimos e ilegitimos.

En los bucles **for**:

* Este bucle recorre una lista de nombres de columnas del DataFrame df.
* En cada vuelta, la variable col toma uno de esos nombres.
* Esto permite repetir el mismo gráfico para cada variable sin tener que copiar el código tres veces.

En plt.figure(): Genera la figura pero aun vacia

En sns.histplot(): dibuja el histograma con ciertos parámetros;

**data=df**: Indica que los datos provienen del DataFrame df.

**x=col:** El nombre de la columna actual del bucle ('dwell\_avg', 'flight\_avg' o 'traj\_avg'). Esa será la variable que se graficará en el eje horizontal.

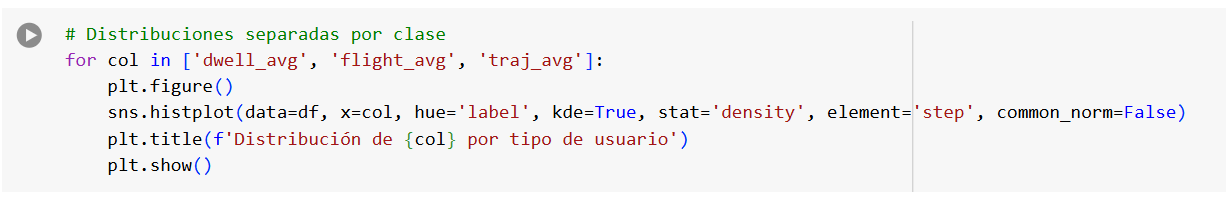
**hue='label':** Seaborn usará la columna 'label' para separar y colorear los datos según su categoría (por ejemplo: “legítimo” vs “impostor”).

**kde=True:** Además del histograma, dibuja la curva de densidad suavizada (Kernel Density Estimate).

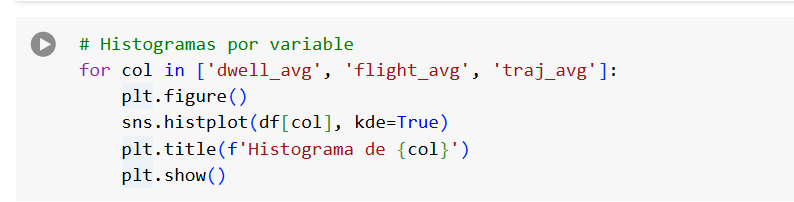
**stat='density':** En lugar de mostrar el conteo de datos por barra, muestra densidad de probabilidad (área total = 1). Esto facilita la comparación entre grupos con diferente cantidad de datos.

**element='step':** El histograma se dibuja con **contornos** tipo escalón, no con barras rellenas.  
Esto hace más fácil ver varias distribuciones sobrepuestas.

**common\_norm=False**  
Por defecto, Seaborn normaliza todas las series juntas para que la suma total sea 1.  
Con False, normaliza cada clase (hue) de forma independiente, de modo que puedas comparar la forma de las distribuciones sin que una clase con más datos “aplane” la otra.



1. Se obtiene otros 3 histogramas pero esta vez por variable(no es de mayor relevancia)



 **for col in [...]**: recorre las tres columnas y repite el gráfico para cada una.

 **plt.figure()**: crea una nueva figura vacía para no sobreescribir gráficos anteriores.

 **sns.histplot(df[col], kde=True)**:

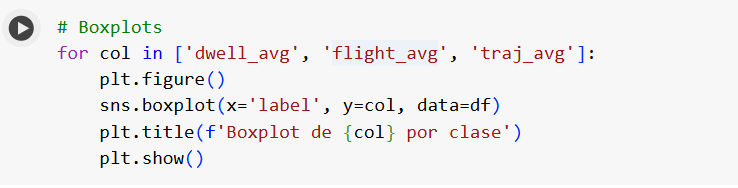
Grafica el histograma de la columna.

kde=True añade una curva de densidad suavizada sobre el histograma.

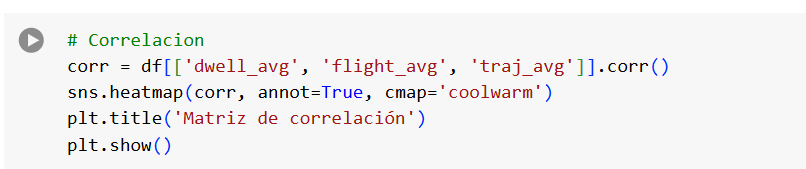
 **plt.title(...)**: pone un título con el nombre de la variable.

 **plt.show()**: muestra el gráfico en pantalla.

1. Boxplot, Diagrama de cajas utilizado para visualizar la distribución de los datos, identificar su tendencia central y detectar posibles valores atípicos (outliers). Permite observar la concentración de los datos en torno a la mediana y la dispersión de los mismos.



1. Matriz de correlación, muestra como las variables están correlacionadas. Este grafico nos sugiere que no hay una relación directa completamente entre las variables ya que miden comportamientos distintos, nos da a entender que esto es perfecto para un modelo de machine learning como el SVM; porque significa que cada variable podría aportar **información única** al modelo.



1. Test de U de Mann-Whiteney : Prueba estadística no paramétrica, esto sirve para comparar las 2 datas de usuarios legitimos e ilegitimos, para ver si hay una diferencia significativa entre sus distribuciones. Se analiza cada variable.



* **Recorre cada variable** (dwell\_avg, flight\_avg, traj\_avg).
* **Separa los datos en dos grupos** según la columna label (0 y 1, por ejemplo, usuarios legítimos vs impostores).
* **mannwhitneyu(...)**:

Aplica la **prueba U de Mann–Whitney**, que compara si dos grupos provienen de distribuciones diferentes (prueba no paramétrica).

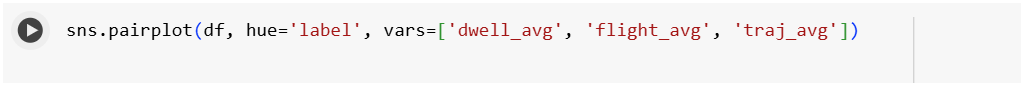
Devuelve:

U o *U-statistic*: medida de la diferencia en rangos entre grupos.

p: valor p para evaluar significancia estadística.

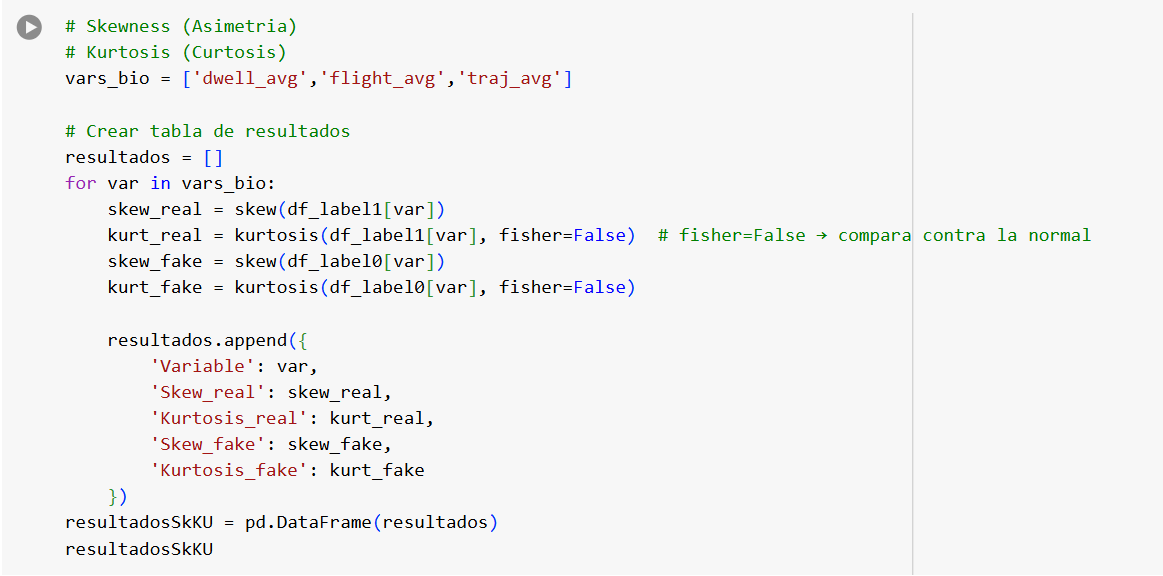
* **alternative='two-sided'**: prueba bilateral (busca diferencias en cualquier dirección).
* **print(...)**: muestra el nombre de la variable, el estadístico U y el valor p con formato reducido de decimales.

1. Pair plot: Ayuda a entender las relaciones entre las variables de los datos mostrando sus interacciones.



**sns.pairplot(...)**: Genera una matriz de gráficos de dispersión (*scatter plots*) para explorar las relaciones bivariadas entre las variables seleccionadas (dwell\_avg, flight\_avg, traj\_avg). El parámetro hue='label' colorea los puntos según la clase (label), facilitando la identificación de patrones o separaciones entre grupos. También incluye histogramas o KDE en la diagonal para mostrar la distribución univariada de cada variable.

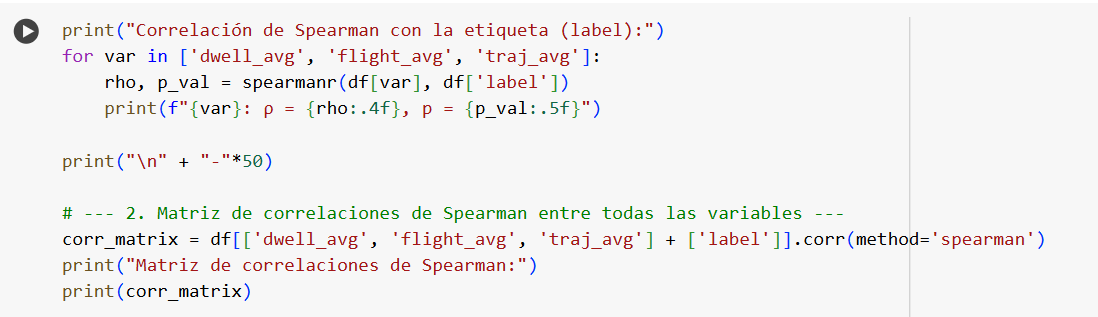
1. Asimetria y curtosis: genera una tabla con los datos(explicado mas detalladamente en el otro documento)



Este código evalúa la forma de la distribución de cada variable biométrica (dwell\_avg, flight\_avg, traj\_avg) de manera separada para usuarios legítimos (df\_label1) y no legítimos (df\_label0).

* **skew()**: mide la asimetría de la distribución (cola hacia la derecha o izquierda).
* **kurtosis(fisher=False)**: mide la curtosis en su definición de Pearson (normal = 3), indicando el grado de concentración y el peso de las colas.  
  Los resultados se almacenan en una lista de diccionarios y luego se convierten en un DataFrame (resultadosSkKU) para facilitar su visualización y análisis comparativo entre ambos grupos.

1. Correlacion de spearman:



Toda la data se analizó con Python en Google colab por la ventaja que no es necesario instalar nada en el computador para que los scripts funcionen.