Comparativa entre librerías de visualización de Python

Juan Luis Serradilla Tormos
Universidad de Murcia (UMU)
Murcia, España
juanluis.serradillat@um.es

***Abstract*—En este trabajo se va a profundizar en el uso de las librerías Matplotlib, Seaborn y Plotly. El documento en cuestión trata de ser una guía a la hora de elegir una de estas librerías, ayudando a ver mejor sus fortalezas, debilidades y especificar sus casos de uso. Primero se explicará el modo en el que usa cada una de ellas, de forma que el lector sepa la estructura básica de un código sencillo. Después, se realizarán tres ejemplos gráficos diferentes con cada una de las librerías (gráficos de violín, gráfico de parejas y gráfico de dispersión 3D); de forma que posteriormente se pueda comparar el desempeño de cada una de las librerías en cada tipo de gráfico. Finalmente, se analizará su desempeño y se obtendrán unas conclusiones en las que, como se ha dicho, se recomendará cada librería en un uso específico, destacando puntos fuertes y débiles.**

***Index terms*—Python, visualización, datos, comparativa, librerías**

I. Introducción

En este trabajo se profundizará en el uso de tres librerías de visualización de datos en Python: Matplotlib, Seaborn y Plotly. Se realizarán ejemplos gráficos con las tres y, posteriormente, se hará una comparativa entre todas, destacando sus puntos débiles y fuertes.

II. Descripción de las librerías

A continuación, se describirán las librerías y el uso que para el que están pensadas.

A. Matplotlib

Matplotlib es una de las librerías más populares para la visualización de datos en Python [1]. Fue diseñada para ser una librería generalista, lo que significa que permite la creación de una amplia variedad de gráficos, desde gráficos de líneas simples hasta visualizaciones complejas como histogramas, diagramas de dispersión o gráficos en 3D.

Debido a su naturaleza generalista, Matplotlib puede ser algo compleja de usar, especialmente para gráficos más avanzados o estéticamente atractivos. Muchas veces, requiere escribir más código para configurar opciones detalladas o personalizaciones. Sin embargo, esta flexibilidad la hace muy poderosa.

La documentación oficial está disponible en https://matplotlib.org, y es muy completa, incluyendo una guía de usuario, tutoriales, ejemplos prácticos y referencias detalladas, lo que facilita el aprendizaje y el uso de la biblioteca.

B. Seaborn

Seaborn es una biblioteca de visualización de datos construida sobre Matplotlib [2]. Su objetivo principal es simplificar la creación de gráficos estadísticos atractivos y fáciles de interpretar. Además, se integra perfectamente con pandas, lo que la hace especialmente útil cuando se trabaja con datos tabulares.

Seaborn se enfoca en gráficos estadísticos como mapas de calor, diagramas de distribución, gráficos de regresión, entre otros. Ofrece estilos predeterminados más estéticos que los de Matplotlib y requiere menos configuración para lograr gráficos visualmente agradables. Sin embargo, tiene menos flexibilidad para gráficos que no sean de tipo estadístico.

La página oficial de Seaborn está disponible en https://seaborn.pydata.org. Su documentación incluye tutoriales detallados y numerosos ejemplos prácticos que ayudan a los usuarios a familiarizarse rápidamente con la biblioteca.

C. Plotly

Plotly es una biblioteca interactiva de visualización de datos que permite crear gráficos dinámicos y altamente personalizables [3]. A diferencia de Matplotlib y Seaborn, Plotly se enfoca en gráficos interactivos que pueden integrarse en aplicaciones web o mostrarse en navegadores, lo que la hace ideal para presentaciones o cuadros de mando.

Ofrece una amplia variedad de gráficos, desde básicos como líneas y barras hasta avanzados como gráficos de mapas, 3D

y gráficos de Sankey. Aunque es muy potente, su enfoque en la interactividad puede requerir un poco más de configuración para los principiantes.

La página oficial de Plotly es https://plotly.com/python, y contiene una extensa colección de tutoriales, ejemplos y documentación para comenzar.

III. Modo de empleo de las librerías

A continuación se profundizará en el uso de cada una de las librerías.

A. *Matplotlib*

El método general para usar Matplotlib es el siguiente:

- Se crea una figura con sus ejes.
- Se añaden gráficas a los ejes que se deseen.
- Se añaden opciones estéticas a estos ejes.
- Se enseña o se muestra la figura.

Un código de ejemplo podría ser el siguiente:

```
import matplotlib.pyplot as plt

# Se crea la figura y los ejes
fig, axes = plt.subplots("Opciones estéticas de ejes
y de la figura")
# Se añaden gráficas a los ejes de la figura
axes[0].plot(...)
axes[1].hist(...)
...
axes[n].bar()
# Se añaden opciones estéticas a los ejes que se
deseen
axes[0].set_xlabel(...)
axes[0].set_title(...)
...
# Se enseña o se muestra la figura
fig.show()      #-> Se enseña
fig.savefig(...) #-> Se guarda
```

B. *Seaborn*

El método general para usar Seaborn es el siguiente:

- Se selecciona el tema del gráfico (opcional).
- Se genera el gráfico.
- Se añaden opciones estéticas con matplotlib.
- Se enseña o se guarda la figura.

Un código de ejemplo podría ser el siguiente:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Configurar el tema estético (opcional)
sns.set_theme(...)
# Se genera el gráfico
ax = sns.barplot(...)
# Se añaden las opciones estéticas deseadas
plt.xlabel(...)
plt.ylabel(...)
...
# Se muestra o se guarda el gráfico
plt.show()      #-> Se muestra
plt.savefig(...) #-> Se enseña
```

C. *Plotly*

El método general para usar Plotly es el siguiente:

- Se crea la figura con el gráfico
- Se añaden opciones estéticas al gráfico
- Se muestra o se guarda el gráfico

```
import plotly.express as px

# Se crea la figura con el gráfico
fig = px.bar(...)
# Se añaden opciones estéticas al gráfico
fig.update_layout(
    xaxis_title=...,
    yaxis_title=...,
    ...
)
# Se muestra o se guarda el gráfico
fig.show()          #-> Se muestra
fig.write_html(...) #-> Se enseña
```

IV. Ejemplos de gráficos

En esta sección se realizarán varios tipos de gráficos con las tres librerías, para así poder comparar el desempeño de cada una de ellas.

Los datos que se utilizarán para realizar las diferentes visualizaciones se han extraído de Kaggle [4]. Los datasets son los siguientes:

- Dataset sobre factor de rendimiento de los estudiantes [5]. La naturaleza de este dataset permite realizar gráficos estadísticos y ver si son útiles para sacar conclusiones de los datos.
- Dataset de datos interesantes para visualizar [6]. Este es un dataset interesante ya que tiene una gran variedad de datos diferentes, lo que permite probar gráficos que el dataset anterior no.

Se van a probar los siguientes gráficos:

- Gráfico de violín
- Gráfico de parejas
- Gráfico 3D

En cada apartado se mostrarán las tres figuras de los gráficos y se referenciarán los códigos utilizados para generarlos, que se encontrarán en los apéndices.

A. *Gráfico de violín*

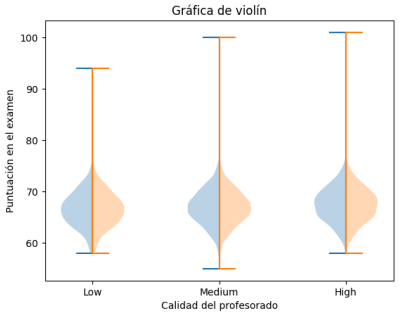

Fig. 1: Gráfico de violín realizado con Matplotlib

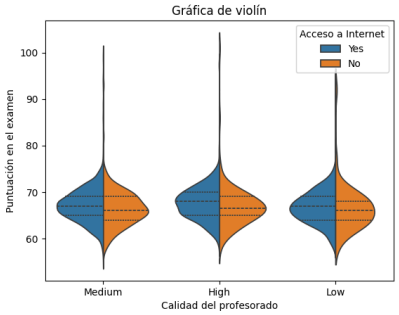

Fig. 2: Gráfico de violín realizado con Seaborn

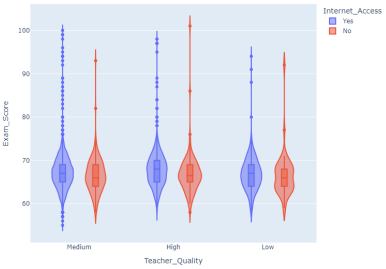

Fig. 3: Gráfico de violín realizado con Plotly

B. *Gráfico de parejas*

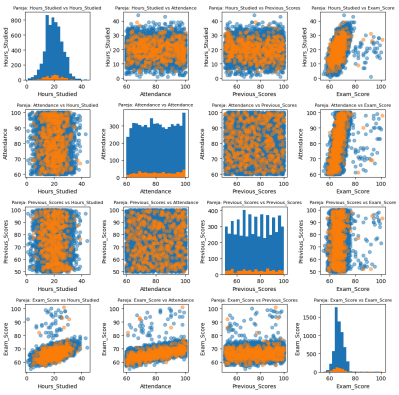

Fig. 4: Gráfico de parejas realizado con Matplotlib

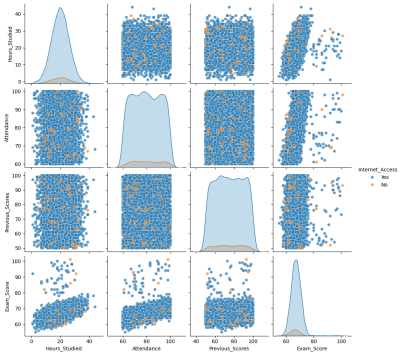

Fig. 5: Gráfico de parejas realizado con Seaborn

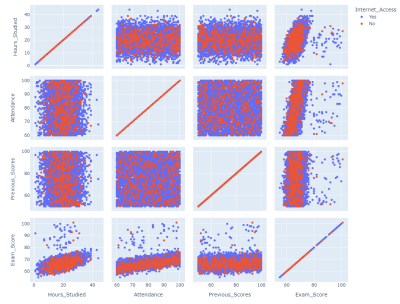

Fig. 6: Gráfico de parejas realizado con Plotly

C. *Gráfico 3D*

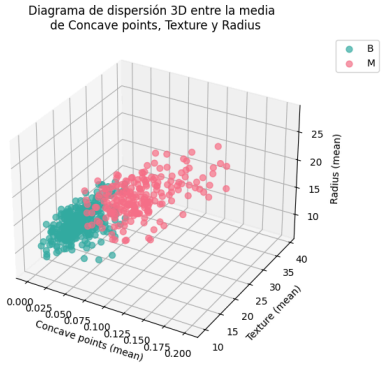

Fig. 7: Gráfico 3D realizado con Matplotlib

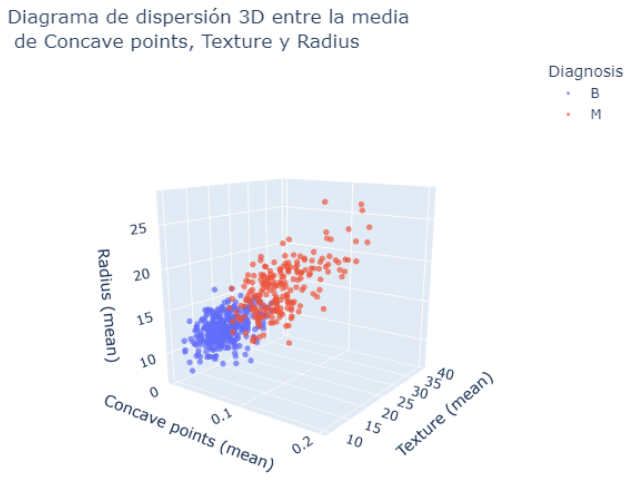

Fig. 8: Gráfico 3D realizado con Plotly

V. Análisis de los gráficos

Ahora se analizarán los gráficos anteriores para comparar los resultados y el código (puede consultarse en la Sección VII).

- **Gráfico de violín:**
  Con este gráfico de violín se quiere ver la distribución de las notas de los alumnos en función de la calidad del profesorado y del acceso a internet.

  Respecto a la calidad de los gráficos, vemos que en el caso de Matplotlib el gráfico no ha conseguido ser muy informativo en comparación al resto. Se puede ver en Fig. 1 como no aparecen reflejados valores estadísticos como la media y los cuartiles, que pueden ser muy útiles para el análisis del gráfico. Además, no se aprecian bien las diferencias entre las distribuciones del lado izquierdo y derecho del gráfico. Estos problemas se solucionan con creces en el gráfico de Seaborn (Fig. 2), donde se aprecian las magnitudes estadísticas necesarias y, además, hay una clara diferencia entre las diferentes distribuciones de notas. Plotly (Fig. 3), por su parte, es bastante similar a Seaborn, con la salvedad de que no se han podido juntar las diferencias de la categoría «acceso a internet» en un mismo «violín». Sin embargo, el gráfico sigue siendo muy descriptivo, al igual que en Seaborn se aprecia como la distribución de las notas de la gente sin acceso a internet es inferior a la gente con acceso. Además, la interactividad de Plotly, aunque no se pueda probar en una imagen estática, es muy útil para poder visualizar valores individuales, se pueden consultar valores como la media, los cuartiles, los valores atípicos, además de poder ampliar el gráfico para un mayor nivel de detalle.

  Respecto al código, se observa en la Sección VII.A como el código de Matplotlib para generar un gráfico de violín es bastante más complicado que el código de Seaborn o el de Plotly. En el caso de Matplotlib se han tenido que preparar los datos previamente para separar por categorías. Después, se han generado dos gráficos de violín diferentes, uno para el lado izquierdo y otro para el lado derecho. Por otro lado, en Seaborn y en Plotly solo se ha tenido que llamar a una función para generar el gráfico y dentro de esta se han especificado los parámetros necesarios.

- **Gráfico de parejas:**
  El gráfico de parejas sirve para comparar las variables numéricas dentro de un dataset, viendo las relaciones de todas las parejas de estas variables. En este ejemplo se querían ver las relaciones entre todas las variables numéricas del dataset de rendimiento de los estudiantes.

  Respecto a la calidad de los gráficos, vemos como tanto Matplotlib (Fig. 4) como Seaborn (Fig. 5) han obtenido resultados muy parecidos. En estos ejemplos se puede analizar tanto la relación entre las parejas con gráficos de dispersión como la distribución de las diferentes clases en una misma variable con histogramas. Se observa como las distribuciones de las parejas con y sin acceso a internet son idénticas, aunque el tamaño de la muestra de alumnos sin acceso a internet es mucho menor. Por otro lado, Plotly solo representa gráficos de dispersión y no los histogramas en el caso de representar una variable consigo misma. Además, la interactividad en este caso no es tan importante y no supone tantas ventajas. Por ello no es la mejor opción para este tipo de gráfico, aunque tampoco tiene un mal desempeño.

  Respecto al código necesario para generar el gráfico (Sección VII.B), Matplotlib vuelve a ser la opción más complicada. Como no tiene una función específica para un gráfico de parejas, es necesario hacer un bucle y para asignar un valor a las diferentes variables y categorías. Por otro lado, Seaborn y Plotly tienen una función integrada que permite generar el gráfico con los parámetros necesarios, siendo así las opciones más sencillas.

- **Gráfico 3D:**
  El gráfico 3D es útil cuando se quieren representar 3 variables en un diagrama de dispersión, por ejemplo. De esta forma puede revelar tendencias entre diferentes

categorías a la hora de dispersarse entre cada una de las tres variables. Para este análisis hemos usado el dataset de visualización de datos, y dentro de este hemos codigo el dataset de muestra de datos de cáncer. Al contener muchas variables numéricas relacionadas entre sí es bastante útil para poder ver el potencial de este tipo de diagramas de dispersión.

Respecto a la calidad de los gráficos, se puede ver como tanto Matplotlib (Fig. 7) como Plotly (Fig. 8) dan resultados estáticos bastante parecidos. En los dos casos se aprecia como los cánceres con etiqueta «B» se agrupan en una zona con valores menores a los de etiqueta «M». Sin embargo, Plotly tiene una gran ventaja en este tipo de gráficos y es su interactividad. En estos gráficos de dispersión 3D hace falta ver la imagen desde diferentes perspectivas para poder ver las tendencias de agrupamiento de los datos. En el caso de Matplotlib es necesario tomar varias instantáneas desde diferentes enfoques, pero gracias a la interactividad de Plotly esto no es necesario y basta con rotar la figura generada, pudiendo ver con mucho detalle desde múltiples enfoques.

Respecto al código necesario (Sección VII.C), lo primero que se puede apreciar es que no se ha podido generar un gráfico de dispersión 3D con Seaborn. Al ser una librería especializada en gráficos estadísticos no dispone de una función para gráficos 3D. Por su parte, Matplotlib sí que permite generar estos gráficos, pero su código es mucho más complejo que el código de Plotly. Para realizar el gráfico hace falta preparar los datos y luego generar un figura especial de ejes 3D para poder crear la imagen. Por su parte, Plotly tiene una función integrada de gráficos 3D, así que lo único que hay que hacer es llamar a la función con los parámetros necesarios para la representación.

VI. Conclusiones

Podemos decir que la librería Matplotlib es la más complicada de utilizar pero, a su vez, la más completa. En el caso de necesitar generar visualizaciones con Python y no saber de que tipo van a ser o si van a abarcar muchas categorías, es recomendable tener siempre esta librería a mano para poder representar cualquier tipo de gráfico. Su aprendizaje y uso es casi indispensable.

Por otro lado, Seaborn es una librería que da resultados mucho más estéticos, visuales con un código más sencillo. Sin embargo, se limita a gráficos estadísticos. En el caso de que solo se quieran mostrar este tipo de figuras es la librería más interesante de aprender, además de la más cómoda de utilizar.

Finalmente, Plotly es una librería con un caso de uso muy concreto, gráficos interactivos. En el caso de mostrar las figuras en documentos estáticos otras librerías como Seaborn hacen la misma función de ser una simplificación de Plotly, pero si se van a mostrar los gráficos en páginas web, dashboards, etc; la interactividad de Plotly destaca enormemente. Es una herramienta muy potente en estos casos y altamente recomendada.

Como conclusión, todas las librerías son bastante completas y tienen un nicho de uso concreto, se pueden utilizar todas perfectamente siempre que se sepa cuál es el objetivo que se quiere conseguir con los gráficos, donde ser van a visualizar y qué se quiere mostrar.

VII. Apéndices

A. *Código de gráficos de violín*

a) *Matplotlib:*

```
plot_data = [
    data_students[data_students["Teacher_Quality"]
== "Low"]["Exam_Score"].values,     # Datos para Low
    data_students[data_students["Teacher_Quality"]
==  "Medium"]["Exam_Score"].values,  #  Datos  para
Medium
    data_students[data_students["Teacher_Quality"]
== "High"]["Exam_Score"].values     # Datos para High
]
plt.violinplot(plot_data, side="low")
plt.violinplot(plot_data, side="high")
plt.xticks(np.arange(1,     4),     labels=["Low",
"Medium", "High"])
plt.title("Gráfica de violín")
plt.xlabel("Calidad del profesorado")
plt.ylabel("Puntuación en el examen")
plt.show()
```

b) *Seaborn:*

```
sns.violinplot(
  data=data_students,
  x="Teacher_Quality",
  y="Exam_Score",
  hue="Internet_Access",
  split=True,
  inner="quart"
)
plt.title("Gráfica de violín")
plt.xlabel("Calidad del profesorado")
plt.ylabel("Puntuación en el examen")
legend = plt.gca().get_legend()
legend.set_title("Acceso a Internet")
plt.show()
```

c) *Plotly:*

```
fig = px.violin(
  data_students,
  x="Teacher_Quality",
  y="Exam_Score",
```

```
  color="Internet_Access", box=True
)
fig.update_layout(height=600, width=775)
fig.show()
```

B. *Código de gráficos de parejas*

a) *Matplotlib:*

```
numerical_columns = [
  "Hours_Studied",
  "Attendance",
  "Previous_Scores",
  "Exam_Score"
]
categorical = "Internet_Access"
plot_dataset = data_students[numerical_columns]
fig, axes = plt.subplots(len(numerical_columns),
len(numerical_columns), figsize=(10, 10))
for cat in ["Yes", "No"]:
                    plot_dataset_filt =
plot_dataset[data_students[categorical] == cat]
  for y,y_label in enumerate(numerical_columns):
    for x,x_label in enumerate(numerical_columns):
      ax = axes[y, x]
      if x == y:
        ax.hist(plot_dataset_filt[x_label], bins=20)
      else:
            ax.scatter(plot_dataset_filt[x_label],
plot_dataset_filt[y_label], alpha=0.5)
      ax.set_xlabel(x_label)
      ax.set_ylabel(y_label)
            ax.set_title(f"Pareja: {y_label} vs
{x_label}", fontsize=8)
plt.tight_layout()
plt.show()
```

b) *Seaborn:*

```
numerical_columns = [
  "Hours_Studied",
  "Attendance",
  "Previous_Scores",
  "Exam_Score"
]
categorical = "Internet_Access"
plot_dataset = data_students[numerical_columns]
numerical_columns.append(categorical)
plot_dataset = data_students[numerical_columns]
sns.pairplot(
  plot_dataset,
  hue=categorical,
  plot_kws={"alpha": 0.7}
)
plt.show()
```

c) *Plotly:*

```
numerical_columns = [
  "Hours_Studied",
  "Attendance",
  "Previous_Scores",
  "Exam_Score"
]
categorical = "Internet_Access"
dataset_columns = numerical_columns.copy()
dataset_columns.append(categorical)
plot_dataset = data_students[dataset_columns]
fig = px.scatter_matrix(
  plot_dataset,
  dimensions=numerical_columns,
  color=categorical
)
fig.update_layout(height=900, width=1100)
fig.show()
```

C. *Código de gráficos 3D*

a) *Matplotlib:*

```
fig = plt.figure(figsize=(6, 6))
ax = fig.add_subplot(projection='3d')
cmap = ListedColormap(sns.color_palette("husl",
2).as_hex())

data_B = data_cancer[data_cancer["Diagnosis"] ==
"B"]
data_M = data_cancer[data_cancer["Diagnosis"] ==
"M"]
sc_B = ax.scatter(
    data_B["Concave points (mean)"],
    data_B["Texture (mean)"],
    data_B["Radius (mean)"],
     s=40, color=cmap(1), label="B", marker='o',
alpha=0.7
)
sc_M = ax.scatter(
    data_M["Concave points (mean)"],
    data_M["Texture (mean)"],
    data_M["Radius (mean)"],
     s=40, color=cmap(0), label="M", marker='o',
alpha=0.7
)

ax.set_xlabel('Concave points (mean)')
ax.set_ylabel('Texture (mean)')
ax.set_zlabel('Radius (mean)')
ax.set_title("Diagrama de dispersión 3D entre la
media \n de Concave points, Texture y Radius")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2)
plt.savefig("scatter_hue", bbox_inches='tight')
plt.show()
```

b) *Plotly:*

```
fig = px.scatter_3d(
  data_cancer,
  x="Concave points (mean)",
  y="Texture (mean)",
  z="Radius (mean)",
  color="Diagnosis",
  opacity=.7
)
fig.update_traces(marker_size=3)
fig.update_layout(
  title_text="Diagrama de dispersión 3D entre la
media <br> de Concave points, Texture y Radius",
  height=600, width=600
)
fig.show()
```

References

[1] J. D. Hunter, «Matplotlib: A 2D graphics environment», *Computing in Science & Engineering*, vol. 9, n.º 3, pp. 90-95, 2007, doi: 10.1109/MCSE.2007.55.

[2] M. L. Waskom, «seaborn: statistical data visualization», *Journal of Open Source Software*, vol. 6, n.º 60, p. 3021, 2021, doi: 10.21105/joss.03021.

[3] N. Kruchten, «An interactive, open-source, and browser-based graphing library for Python». [En línea]. Disponible en: https://github.com/plotly/plotly.py

[4] Kaggle, «Kaggle: Your Machine Learning and Data Science Community». [En línea]. Disponible en: https://www.kaggle.com/

[5] «Student Performance Factors». [En línea]. Disponible en: https://www.kaggle.com/datasets/lainguyn123/student-performance-factors

[6] «Interesting Data to Visualize». [En línea]. Disponible en: https://www.kaggle.com/datasets/alexisbcook/data-for-datavis