UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
Centro de e-Learning

UNIDAD DIDÁCTICA III

MACHINE LEARNING

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo I – Nivel Inicial I

Unidad III – Inspección y visualización de datos I.

Presentación:

Dado que ya hemos aprendido a adquirir los datos de diferentes formatos mediante el uso de **pandas**, nos dispondremos en el transcurso de esta unidad a utilizar la librería **Matplotlib** de forma de visualizar los datos adquiridos.

Matplotlib es una herramienta imprescindible en el análisis ya que nos permite presentar la información en una variedad enorme de formatos tanto de 2D como 3D.

Objetivos:

Que los participantes:

Aprendan a visualizar los datos adquiridos con pandas.

Exploren diferentes formatos de representación de datos.

Bloques temáticos:

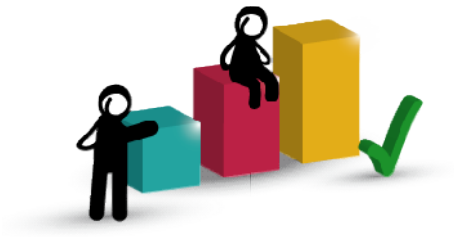
1.- Matplotlib – Descripción.

2.- Gráficos de línea.

3.- Gráficos de dispersión.

4.- Polinomios.

5.- Mapa de colores.



Consignas para el aprendizaje colaborativo

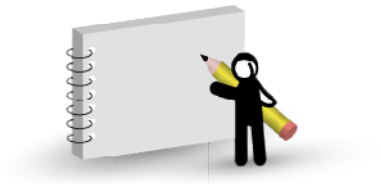
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Matplotlib - Descripciones

Vamos a iniciar a trabajar con matplotlib aprendiendo a agregar comentarios y descripciones a los gráficos, para ello crearemos un gráfico con tres curvas en el dominio x = [0, 2] y adicionamos 100 puntos equiespaciados dentro del dominio mediante:

```
x = np.linspace(0, 2, 100)
```

Luego podemos crear las curvas utilizando el método plot() en donde pasamos como parámetros:

- Primero los puntos correspondientes a la coordenada en x.
- Segundo los puntos de la coordenada en y.
- Tercero el color de línea o el tipo de representación para los puntos de la línea, si en lugar de poner 'pink' ponemos '+' cada punto es representado por un signo de (+).
- Cuarto el atributo **label** con la descripción de la curva. Para que la descripción salga en pantalla debemos agregamos el método **legend()**.

```
plt.plot(x, x, 'pink', label='lineal')
plt.plot(x, x**2, label='cuadratica')
plt.plot(x, x**3, label='cubica')

plt.legend()
```

Colores

Para los colores también tenemos otras opciones como:

- Dar el valor hexadecimal de forma análoga a como se hace en una página web dentro de código css. Los colores representan tres canales de colores:
  - Canal rojo #ff0000
  - Canal verde #00ff00

- Canal azul #0000ff

- Se puede dar una abreviatura para el color, por ejemplo 'm' representa el color magenta.
- Se puede especificar el parámetro c con un valor de C0 al C9, lo cual agrega colores con buen contraste preestablecidos.
- Los colores pueden ser dados en canales de colores rgb en donde se dan tres parámetros entre 0 y 1 representando los canales rojo, verde y azul de la siguiente manera: c=(1,0.1,1)
- Los colores pueden ser dados en canales de colores rgba en donde se dan tres parámetros entre 0 y 1 representando los canales rojo, verde y azul y un cuarto parámetro que determina el nivel de opacidad, de la siguiente manera: c=(1,0.1,1,0.5)

Descripción de curva.

La descripción de la curva puede posicionarse en la pantalla si tomamos la longitud de cada eje como si estuviera en el intervalo **(0,1),** por lo que podemos en base a esto agregar dentro de **legend()** la ubicación según cada eje con el uso del parámetro **loc**. De esta forma podemos darle una posición específica dentro del gráfico de la siguiente forma: **plt.legend(loc=(0.5,0.7))**

Descripciones de título y ejes

Para agregar descripciones a los ejes y título utilizamos los métodos:

- xlabel() para agregar una descripción en el eje x.
- ylabel() para agregar una descripción en el eje y.
- title() para adicionar un título al gráfico.

```
plt.xlabel('Eje X')
plt.ylabel('Eje Y')
plt.title("Estructura de gráfico")
```

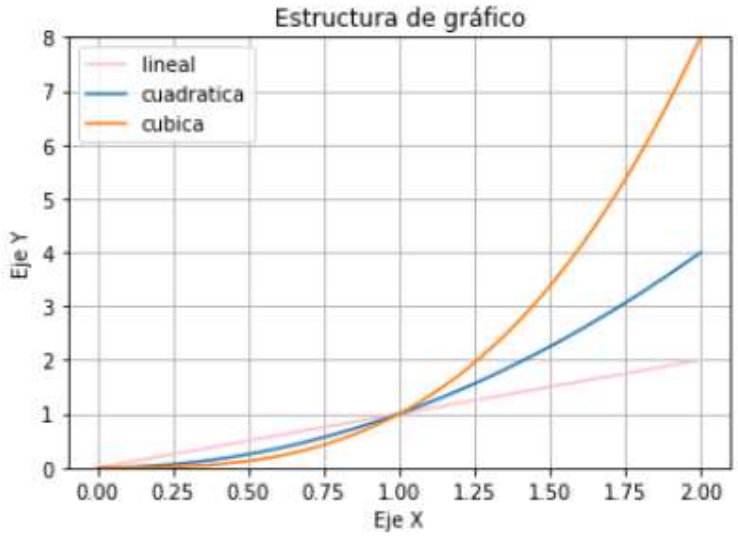
Límites en ejes

También podemos especificar los límites de la representación de datos en los ejes mediante:

- plt.ylim(valor mínimo, valor máximo) para el eje y.
- plt.xlim(valor mínimo, valor máximo) para el eje x.

```
plt.ylim(0,2)
plt.ylim(0,8)
```

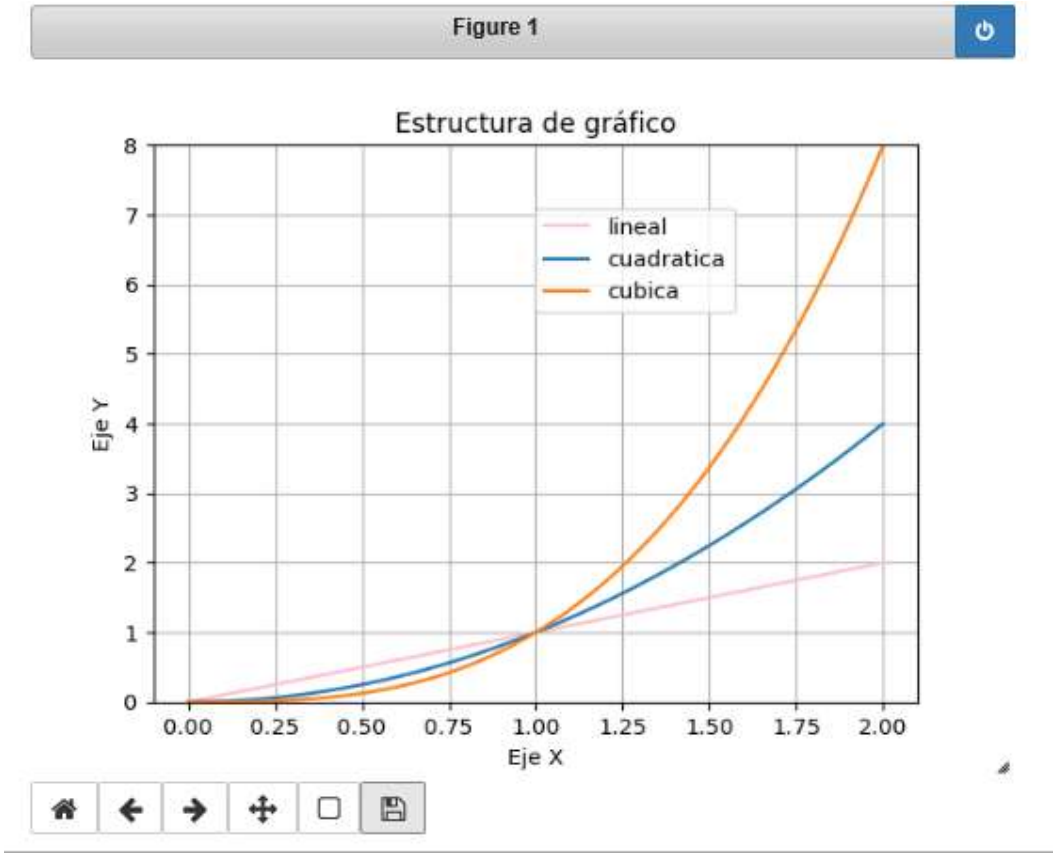
Si queremos adicionar una grilla lo podemos realizar mediante el método **grid()**. La representación visual nos queda:



Si al inicio de nuestro código agregamos la línea:

```
%matplotlib notebook
```

Nuestro gráfico se vuelve interactivo y aparecen herramientas de navegación de forma de poder hacer zoom en un área o desplazar el gráfico para posicionarnos en un punto específico, también aparece un tirador en el margen inferior derecho desde el cual podemos aumentar o disminuir el tamaño del gráfico:

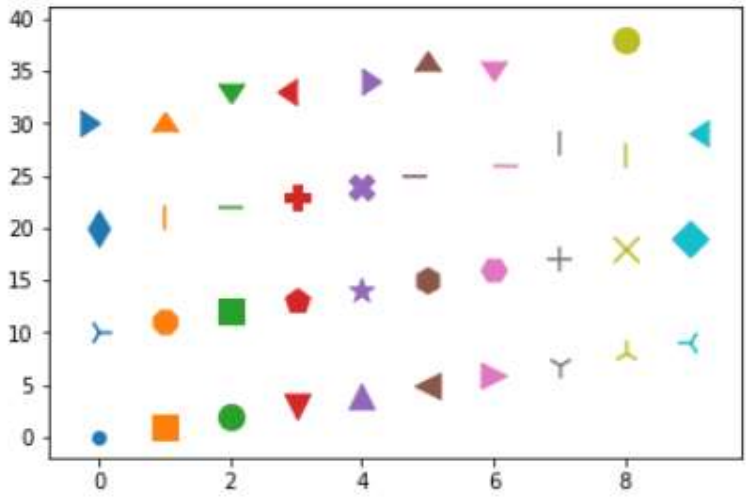


Icono	Descripción
Casa	Restablecer vista original
Flecha izquierda	Volver a la vista anterior
Flecha derecha	Ir a la siguiente vista
Flecha en cuatro direcciones	Paneo manteniendo presionada la tecla izquierda del mouse y Zoom con la tecla de flecha derecha en la pantalla.
Rectángulo	Zoom de enmarcado.
Disquete	Guardar

Tipos de marcas

En lugar de obtener puntos en las dispersiones, podemos agregar distintos tipo de indicadores, podemos verlos todos al ejecutar el siguiente código:

```
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D

for i,marker in enumerate(Line2D.markers):
    plt.scatter(i%10,i,marker=marker,s=150)
plt.show()
```



El tamaño del indicador lo puedo modificar con el parámetro **s**.

Estilos de línea

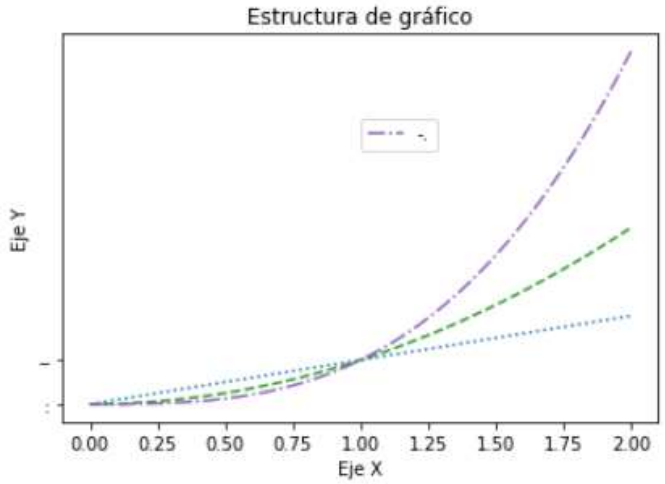
Podemos agregar diferentes tipos de trazado de línea utilizando los siguientes parámetros:

Parámetro	Tipo de línea
'-'	Línea llena
'--'	Línea discontinua

'-.'	Línea de raya punto
'.'	Línea de puntos grandes
':'	Línea de puntos

Veamos un ejemplo:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 2, 100)
plt.plot(x, x, '.','.')
plt.plot(x, x**2, '--','--')
plt.plot(x, x**3,'-.', label='-.')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')
plt.title("Estructura de gráfico")
plt.show()
```



2.- Gráficos de línea

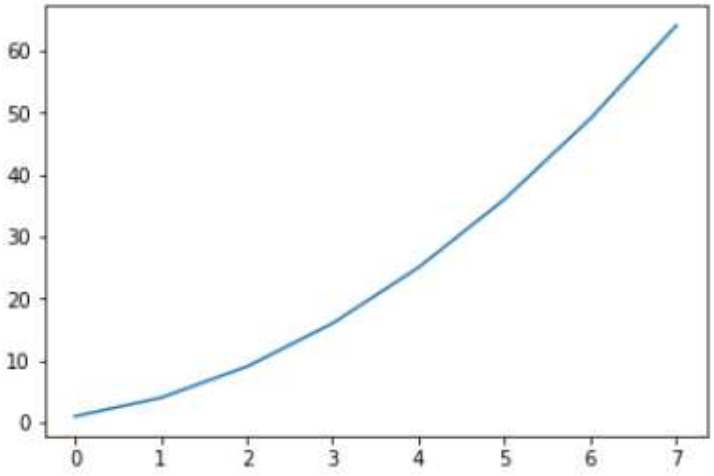
Los gráficos de línea podemos realizarlos a partir de:

- Dos grupos de datos, uno para el eje x y otro para el eje y como vimos en el punto anterior.
- Un grupo de datos para el eje y.

Ejemplo de un solo grupo de datos

Si en lugar de especificar las los datos pertenecientes a las dos coordenadas, presentamos solo un grupo de datos, matplotlib presenta los mismos asignándoles un valor x entero creciente como se puede ver a continuación.

```
y2 = [1,4,9,16,25,36,49,64]
plt.plot(y2)
plt.show()
```



3.- Gráficos de dispersión

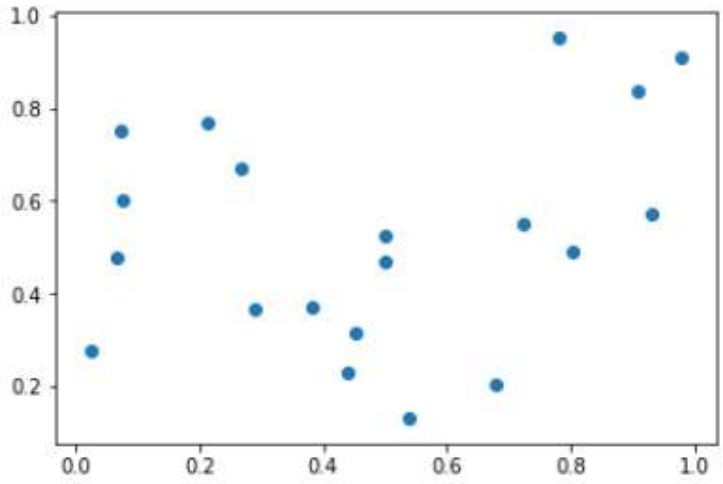
Para generar datos aleatorios se suele partir de un valor llamado "**semilla**", al igual valor de semilla se obtiene un mismo grupo de datos de dispersión. Para especificar una semilla utilizamos el método de numpy: **numpy.random.seed(valor).**

A partir de esta semilla podemos crear valores de dispersión mediante el métodos de numpy:

numpy.random.rand(número_de_arrays_generados, elementos_de_la_dispersión)

Un ejemplo podría quedar así:

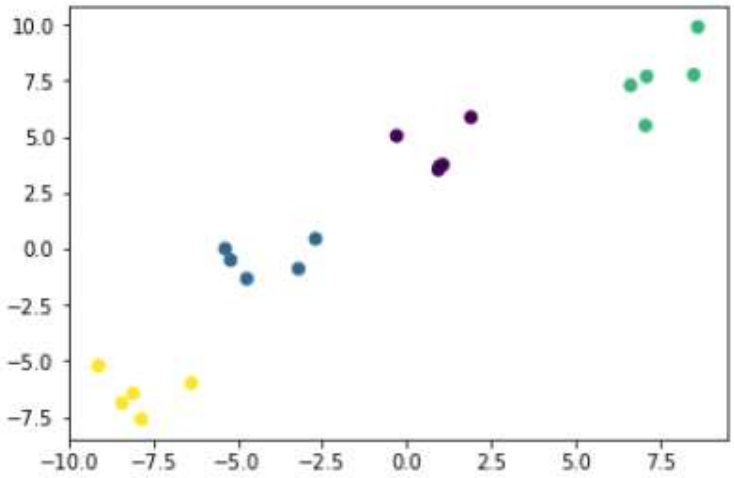
```
import numpy as np
np.random.seed(7)
datos = np.random.rand(2,20)
plt.scatter(datos[0],datos[1])
plt.show()
```



Grupos de dispersión

En el caso de que tuviéramos que representar en un mismo gráfico varios grupos de dispersiones, podemos realizarlo con la ayuda del método **make_blobs**, el cual genera puntos basados en la distribución Gaussiana isotrópica según se muestra a continuación:

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
datos_mb,features = make_blobs(n_samples=20, centers=4, random_state=3)
plt.scatter(datos_mb[:, 0], datos_mb[:, 1], marker='o', c=features)
plt.show()
```



Aquí los parámetros son:

Parámetro	Descripción
seed	Semilla
n_samples (int)	El número total de puntos divididos en partes iguales entre los grupos. El valor por defecto es 100
centros (int o matriz de forma [n_centers,n_features]):	El número de centros a generar, o las ubicaciones de los centros fijos. El valor por defecto es 3
cluster_std (flotante)	La desviación estándar de los grupos. El valor por defecxto es 1.0
shyffle (boolean)	Mezclar las muestras.El valor por defecto es = Verdadero
random_state (int or None)	Semilla utilizada para el generador de números aleatorios, si el valor es None, utiliza el valor de np.random.

4.- Polinomios

En ocasiones podemos intentar ajustar una curva por un polinomio, como puede ser en una representación lineal, en este caso podemos utilizar el método polyfit() el cual tiene tres parámetros:

- Puntos en el eje x
- Puntos en el eje y
- Grado del polinomio.

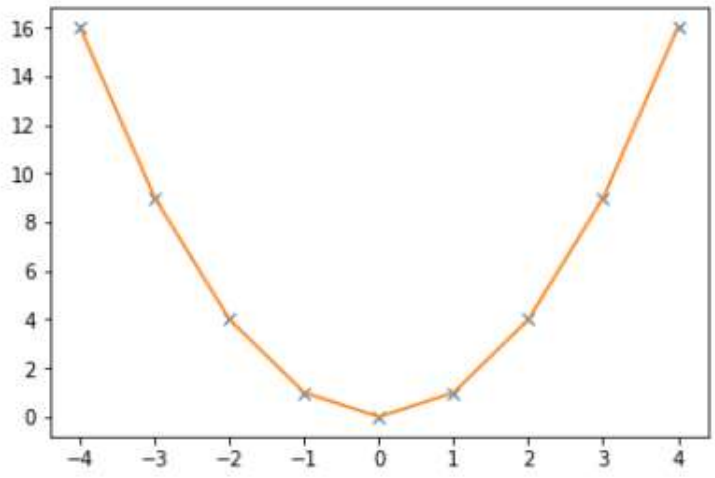
El método nos retorna los coeficientes del polinomio, recordemos que la ecuación de un polinomio de grado 'n' es:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots\ldots + a_0 \, x^0$$

Los datos retornados son guardados en un array.

Para graficar el polinomio nos valemos del método poly1d() como se muestra a continuación en donde se ajustan los puntos por un polinomio de grado 2:

```
x = list(range(-4 , 5))
y = [i**2 for i in x]
z = np.polyfit(x, y, 2)

p = np.poly1d(z)
plt.plot(x,y,'x',x,p(x))
```



5.- Mapa de colores

Cuando vamos a utilizar mapas de colores para representar nuestros datos, tenemos que tener en cuenta elegir aquel que mejor represente el conjunto de datos.

Los investigadores han encontrado que el cerebro humano percibe cambios en el parámetro de luminosidad como cambios en los datos mucho mejor que, por ejemplo, cambios en el tono. Por lo tanto, el espectador interpretará mejor los mapas de colores que tienen una luminosidad que aumenta monótonamente a través del mapa de colores.

Los mapas de colores a menudo se dividen en varias categorías según su función, por ejemplo:

- **Secuencial:** Es un cambio en la luminosidad y, a menudo, saturación del color de forma incremental, a menudo con un solo tono; Se debe utilizar para representar información que tiene ordenamiento.
- **Divergente:** Es un cambio en la luminosidad y posiblemente saturación de dos colores diferentes que se encuentran en el medio en un color insaturado; debe usarse cuando la información que se está trazando tiene un valor medio crítico, como la topografía o cuando los datos se desvían alrededor de cero.
- **Cualitativo:** A menudo son colores variados; debe utilizarse para representar información que no tiene orden o relaciones.

Secuencial

Veamos cómo trabajar con mapas del tipo secuencial, se puede encontrar una referencia completa de todos los tipos de mapas en el siguiente link:

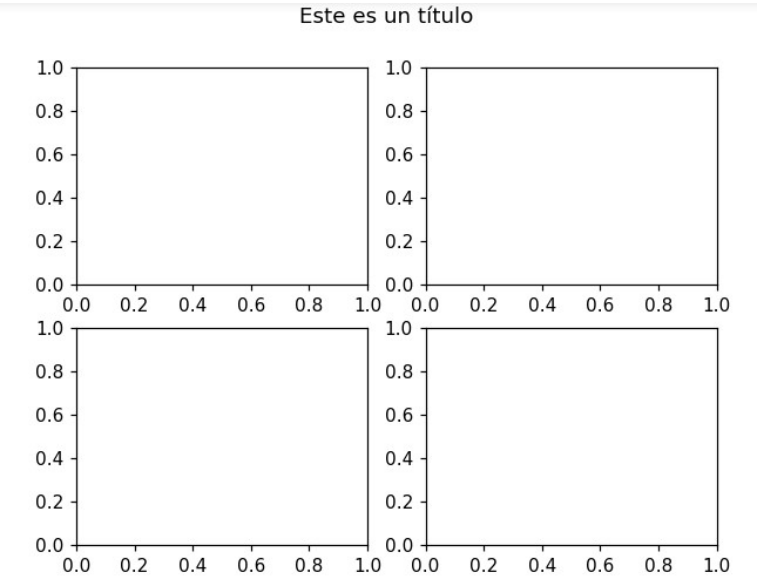
https://matplotlib.org/2.1.0/tutorials/colors/colormaps.html#mycarta-banding

Para comprender el siguiente script debemos tener en mente que en matplotlib se denomina:

- figure: Al área que contiene la figura completa a representar.
- axes: a cada figura representada dentro de figure.
- axis: a los ejes de referencia de cada axes.
- figsize: es el tamaño de cada axes.

Asi por ejemplo si queremos crear cuadro figuras (axes) dentro de un área (gigu

```
import matplotlib.pyplot as plt

import numpy as np

fig, axes = plt.subplots(2, 2)  # Figura con grillas de ejes 2x2

fig.suptitle('Este es un título')
```

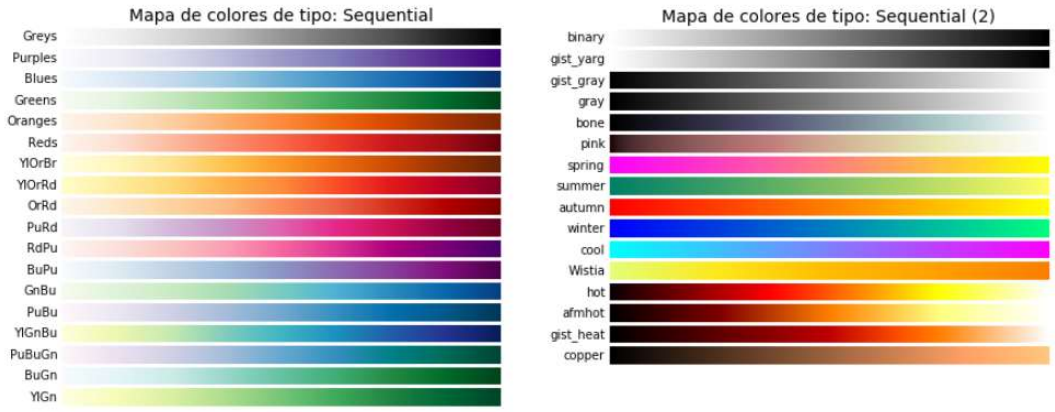


En el siguiente script vamos a utilizar la función mat_color_gratients() para generar los listados de gradientes de colores del tipo secuencial que podemos utilizar en matplotlib.

```
import numpy as np

import matplotlib.pyplot as plt


cmaps = [

        ('Sequential', [
```

```
            'Greys', 'Purples', 'Blues', 'Greens', 'Oranges', 'Reds',
            'YlOrBr', 'YlOrRd', 'OrRd', 'PuRd', 'RdPu', 'BuPu',
            'GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn', 'YlGn']),
        ('Sequential (2)', [
            'binary', 'gist_yarg', 'gist_gray', 'gray', 'bone', 'pink',
            'spring', 'summer', 'autumn', 'winter', 'cool', 'Wistia',
            'hot', 'afmhot', 'gist_heat', 'copper']),
        ]
gradient = np.linspace(0, 1, 256)
gradient = np.vstack((gradient, gradient))
def plot_color_gradients(clase_de_mapa, lista_de_datos):
    numero_de_filas = len(lista_de_datos)
    altura = 0.35 + 0.15 + (numero_de_filas + (numero_de_filas-1)*0.1)*0.22
    fig, axes = plt.subplots(nrows=numero_de_filas, figsize=(6.4, altura))
    fig.subplots_adjust(top=1-.35/altura, bottom=.15/altura, left=0.2, right=0.99)
    axes[0].set_title('Mapa de colores de tipo: ' + clase_de_mapa , fontsize=14)
    for ax, nombre in zip(axes, lista_de_datos):
        ax.imshow(gradient, aspect='auto', cmap=plt.get_cmap(nombre))
        ax.text(-.01, .5, nombre, va='center', ha='right', fontsize=10,transform=ax.transAxes)
    for ax in axes:
        ax.set_axis_off()
for clase_de_mapa, lista_de_datos in cmaps:
    plot_color_gradients(clase_de_mapa, lista_de_datos)
```

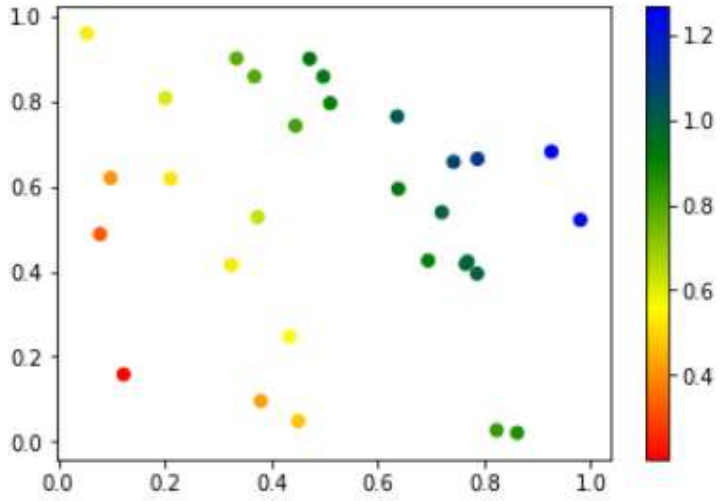
```
plt.show()
```



Un ejemplo muy útil de mapa de color, sería asignar colores en una escala según algún criterio específico, por ejemplo en el siguiente script se crea un mapa de valores aleatorios en donde se utiliza el valor en **x** más el valor en **y** sobre dos para representar el color de la dispersión.

```
import numpy as np

import matplotlib.pyplot as plt

import matplotlib.colors

np.random.seed(7)

x,y = zip(*np.random.rand(50,2))

color=[]

print(len(x))

for i in range(0,50):

    color.append(x[i]+y[i]/2)

cmap = matplotlib.colors.LinearSegmentedColormap.from_list("", ["red","yellow","green", "blue"])
```

```
plt.scatter(x,y,c=color, cmap=cmap)

plt.colorbar()

plt.show()
```



Bibliografía utilizada y sugerida

Libros

Programming Python 5th Edition – Mark Lutz – O'Reilly 2013

Mastering Exploratory Analysis with pandas - By Harish Garg - September 2018

Manual online

(2019, 01). Obtenido 01, 2019, de https://pandas.pydata.org/

(2019, 01). Obtenido 01, 2019, de https://matplotlib.org/

Lo que vimos

En esta unidad hemos trabajado con la librería Matplotlib para realizar la representación visual de datos.

Lo que viene:

En la siguiente unidad comenzaremos a ver cómo tratar los datos adquiridos y visualizados.