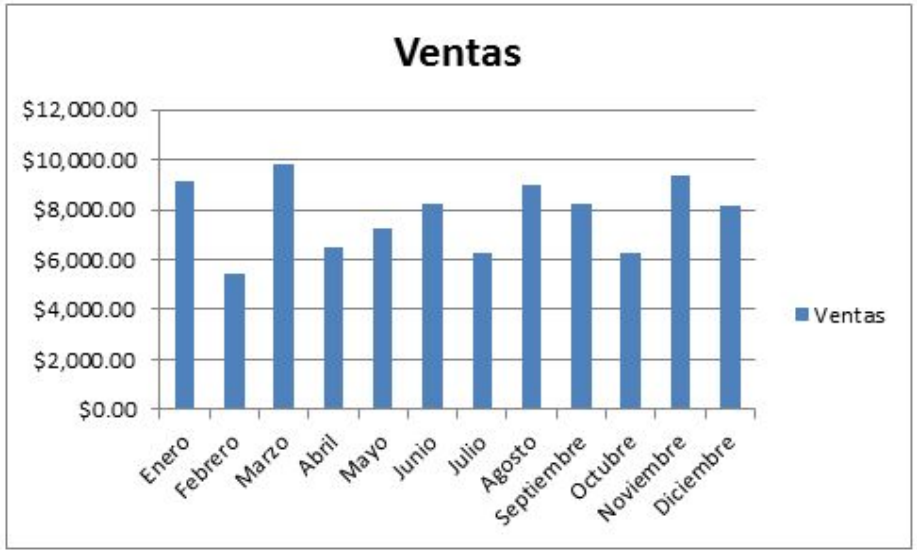
Visualización de datos con Python

Juan Carlos Pérez González

- Una vez que se obtienen los datos, se cargan y se procesan, pasamos a la **representación gráfica de los mismos** ya que, a nivel de registro, su interpretación es más difícil ( )
- Así, por ejemplo, el estudio de variables como ventas por sexo o por código postal, nos puede indicar hacia donde podemos y/o debemos realizar un mayor esfuerzo en nuestras campañas de marketing.
- Lo anterior, obtenido a partir de gráficos resulta más eficiente y rápido ya que nuestra interpretación de la realidad que nos rodea es eminentemente visual: colores, direcciones, volúmenes, superficies….

¿Que se ve mejor?

Mes	Ventas
Enero	$9,132.00
Febrero	$5,414.00
Marzo	$9,852.00
Abril	$6,524.00
Mayo	$7,218.00
Junio	$8,214.00
Julio	$6,249.00
Agosto	$8,957.00
Septiembre	$8,218.00
Octubre	$6,241.00
Noviembre	$9,354.00
Diciembre	$8,173.00



Técnicas de representación

A continuación repasamos otra vez las representaciones gráficas más habituales.

En el anterior caso, tomamos como base las opciones gráficas que nos daba Excel.

Trabajaremos las obtenidas a partir de las librerías Python para su construcción, especialmente ***matplotlib***

Tampoco nos olvidaremos los llamados **cuadros de mando y BI (Business Intelligence)** que estudiaremos en la unidad 5 y 6.

Finalmente indicar que estas herramientas permiten la construcción de gráficos de forma visual ***(Drag&Drop),*** es decir, sin recurrir a lenguajes de programación

En resumen, todo ello con el objetivo de entender mejor los datos obtenidos y tratados en **apoyo a la toma de decisiones** en relación con ellos.

Gráfico de puntos

Muy útiles para estudiar **tendencias y estudios de correlación**. Su forma incluso da idea de la fórmula matemática asociada.

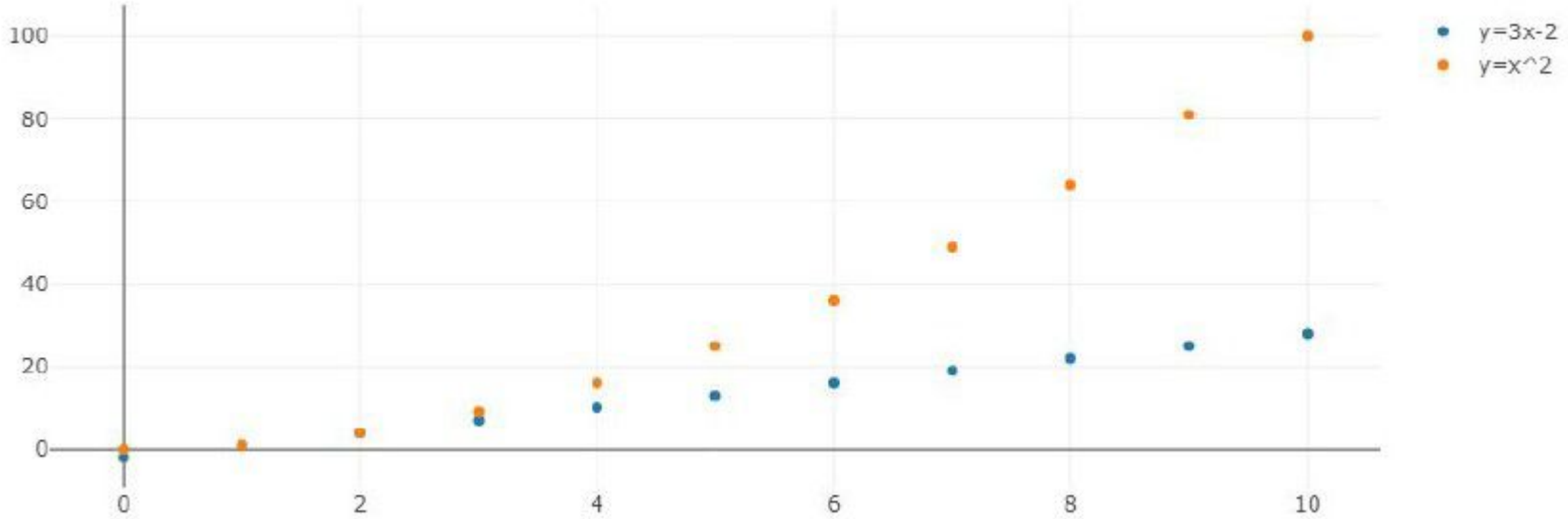


Gráfico de líneas

Deriva de la anterior con el trazado de la línea que une los puntos para dar **sensación de continuidad** y obtención de cualquier valor de datos **más allá de los empleados** en su construcción.

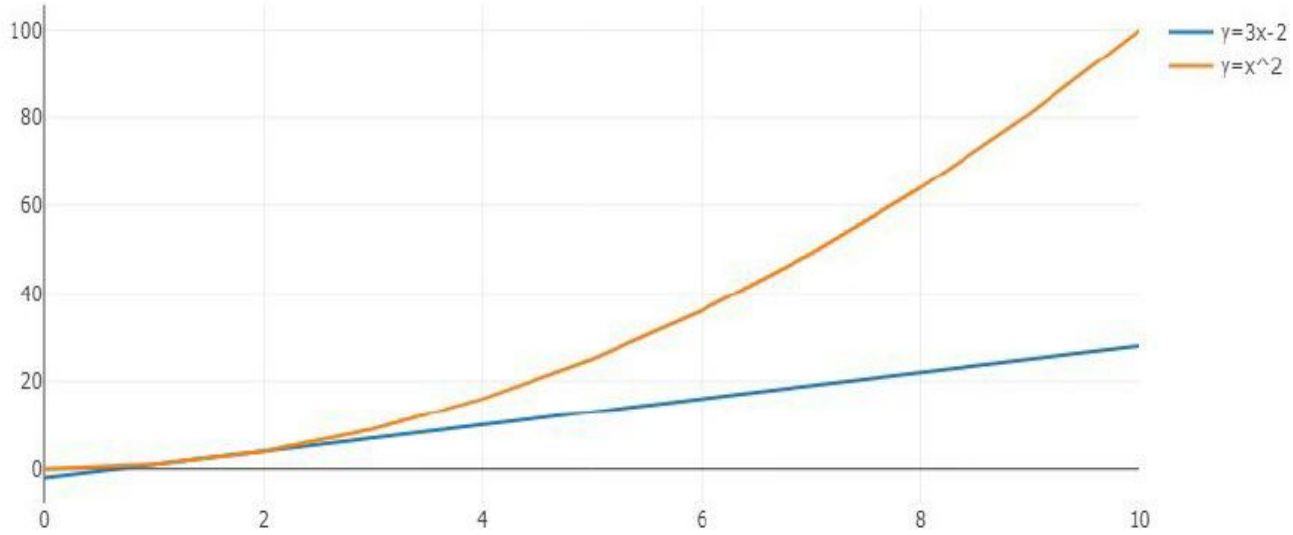


Gráfico de líneas y puntos

A veces interesa combinar ambos.

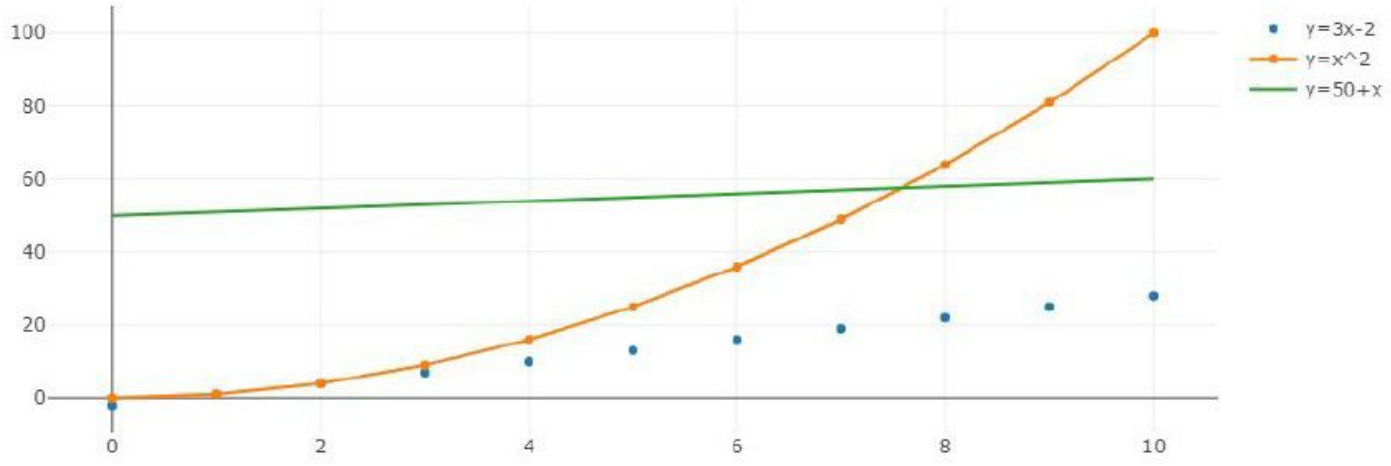
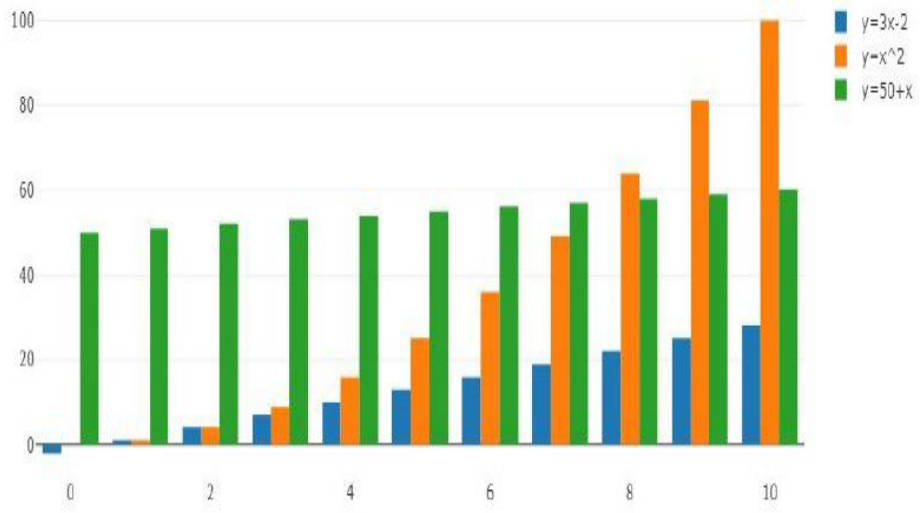
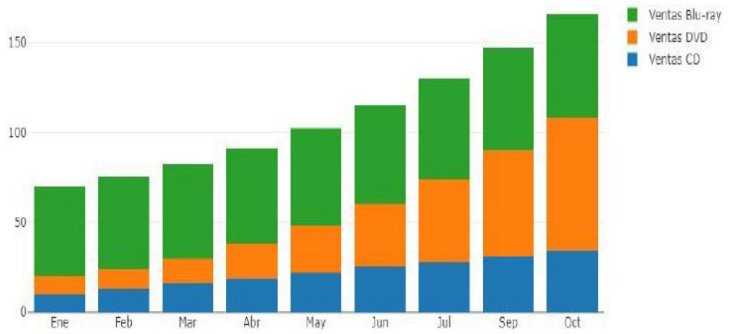
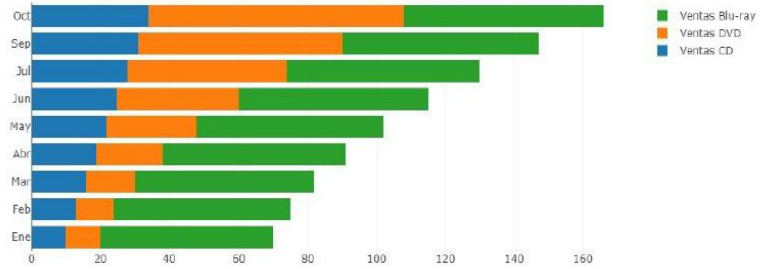


Gráfico de barras y barras apiladas

Válidos para muestras de distintos tipos y **comparativas** rápidas. El gráfico de barras apiladas también puede ser horizontal.

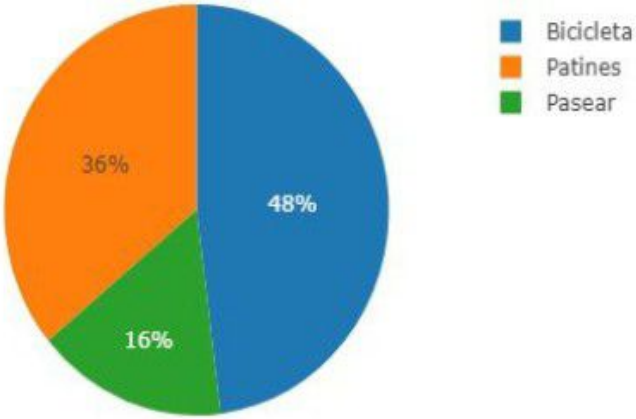


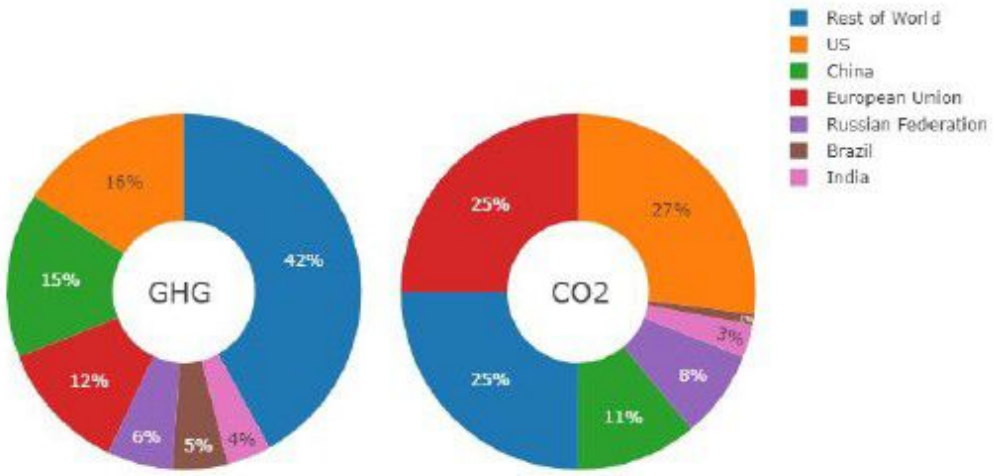




Gráficos circulares

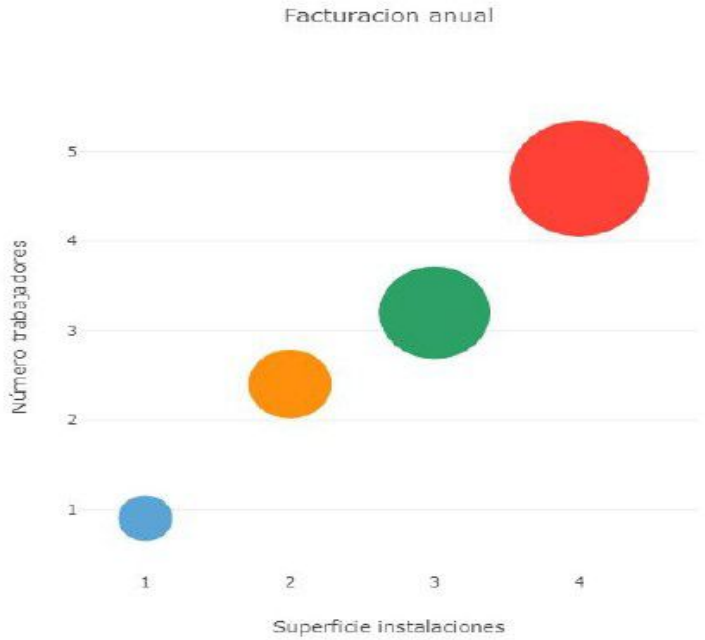
Permite ver la distribución de valores

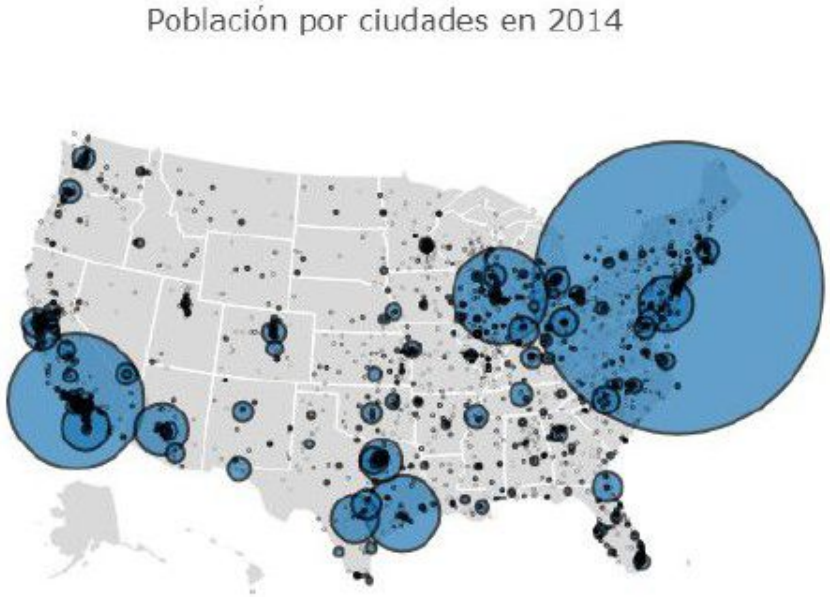




Gráficos de burbujas

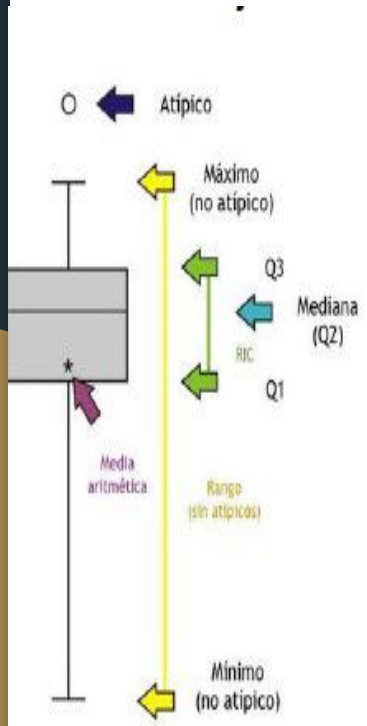
Tiene la ventaja de que permite **unir 3D en un gráfico 2D.** Aquí se observa la relación entre nº de trabajadores, m2 de instalaciones y ventas. Otra opción es sobre mapas.





Gráficos de plotbox

Permiten representar **numerosos valores estadísticos** de una serie de forma unificada.

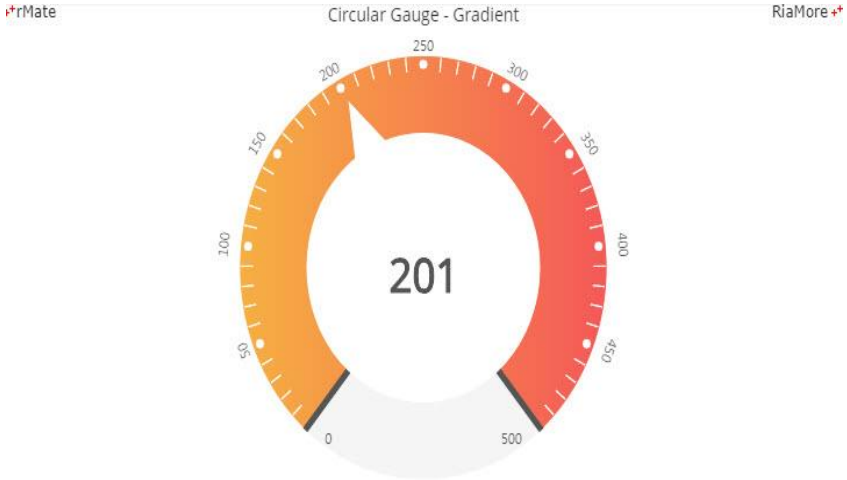


Mapas de colores

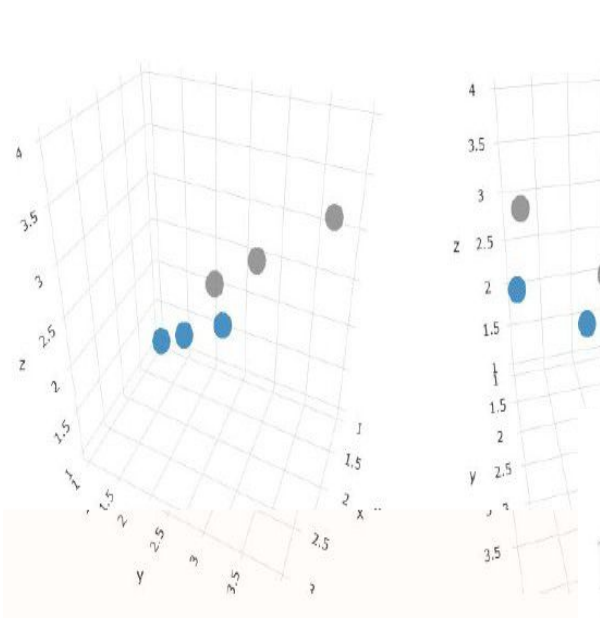
También permite **opciones 3D sobre un gráfico 2D,** en este caso la 3D es la intensidad de color.

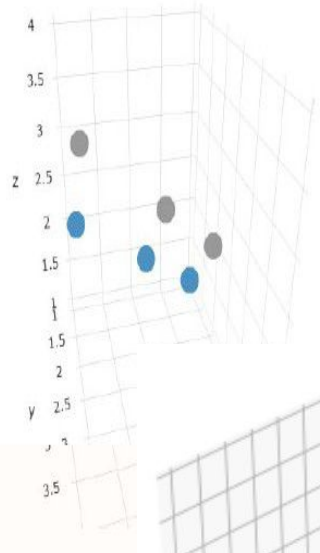
Indicadores

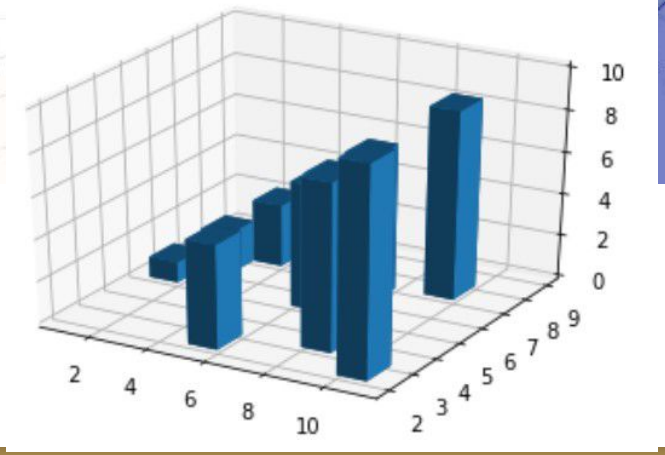
Hay diferentes formatos. Por ejemplo, el tipo “Velocímetro”. Son muy **dinámicos y permiten comparativas entre rangos e incluso colores** para indicarnos si es una zona correcta y/o peligrosa o no (p.e. temperaturas elevadas)

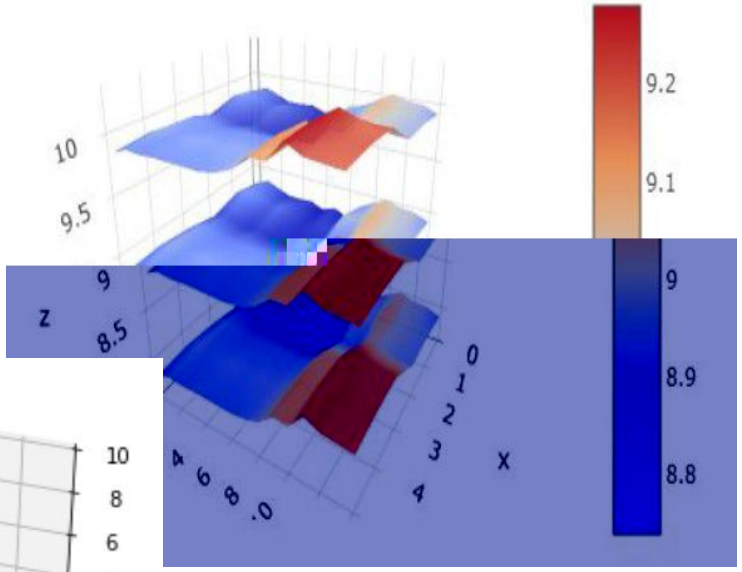


Gráficos de puntos 3D, barras 3D y de superficie 3D









Gráficos con Python (I)

Realizar gráficos con los datos en Python es muy útil como parte de un **análisis exploratorio de datos** (EDA), ya que posteriormente veremos herramientas más potentes.

Así, el científico de datos debe **analizar la información presente** para determinar **si puede servir para resolver el problema.**

Por ejemplo, si van a alimentar a un algoritmo de ML utilizado para predecir una variable binaria, es importante disponer de suficientes muestras de entrenamiento de ambos tipos o si hay muestras atípicas, con gran desviación.

Python dispone del paquete ***matplotlib*** para la realización de gráficos. Existen paquetes adicionales que, basándose en matplotlib, **extienden su funcionalidad**. Uno de los más usados es ***seaborn***.

El ***módulo principal*** de esta librería es ***pyplot***, así lo primero que debemos hacer es importarlo:

```
import matplotlib.pyplot as plt
```

Para estos primeros ejemplos lo haremos con un dataframe sencillo.

Matplotlib y Seaborn son paquetes muy extensos, imposibles de cubrir en su totalidad. Veremos algunas de sus opciones.

Pyplot (I)

El elemento que contiene el gráfico es la ***figura,*** la cual puede tener 1 o N gráficos o ***plots***

fig = plt.figure() La figura creada toma por defecto la escala en pulgadas

Podemos crear la figura con otro tamaño de dos formas:

fig = plt.figure(figsize = (10,6)) → *Solo afecta a esta figura*

plt.rc('figure', figsize = (10,6)) → *Afecta a cualquier figura futura*

Matplotlib almacena la configuración en la propiedad rcParams. Podemos modificar el comportamiento por defecto alterando los valores de esa propiedad.

Con esto ya podríamos generar un gráfico.

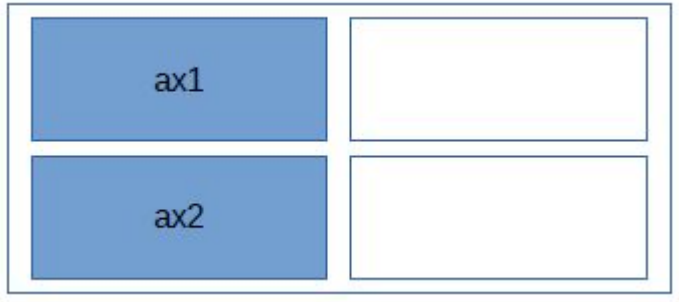
Pyplot (II)

Sin embargo, puede que queramos mostrar más gráficos, para ello se crean ***subplots***

fig = plt.figure()

ax1 = fig.add_subplot(2, 2, 1)

ax2 = fig.add_subplot(2, 2, 3)



*.**add_subplot** devuelve los ejes de un gráfico (objeto Axes). Posteriormente, con este objeto podemos dibujar la gráfica.*

En este caso estamos creando:

- *un gráfico en una figura de 2x2 gráficos (primeros 2 parámetros) y es el de la esquina superior izquierda*
- *un gráfico en una figura de 2x2 gráficos y es el de la esquina inferior izquierda*

Pyplot (III)

Hay otro modo:

fig, axes = plt.subplots(2,2,)

.subplots evuelve directamente dos objetos:

- la figura
- una matriz de objetos de tipo Axes, que definen cada gráfico

Recibe el número de gráficos a incluir en la figura, en horizontal y vertical.

Cada Axes dónde se pintará un gráfico

Pyplot (IV)

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt
```

Gráficos de Líneas ⇒ `df.plot(x='columna_x', y='columna_y', kind='line')`

Gráficos de Barras: ⇒ `df.plot(x='columna_x', y='columna_y', kind='bar')`

Gráficos de Barras Apiladas: ⇒ `df.plot(x='columna_x', y='columna_y', kind='bar', stacked=True)`

Gráficos de Histogramas: ⇒ `df['columna'].plot(kind='hist', bins=30)`

Gráficos de dispersión: ⇒ `df.plot(x='columna_x', y='columna_y', kind='scatter')`

Gráficos de Caja (Box Plots): ⇒ `df.plot(kind='box')`

Gráficos de Área: ⇒ `df.plot(x='columna_x', y='columna_y', kind='area')`

Mapas de Calor (Heatmaps): ⇒

```
matriz_correlacion = df.corr()

sns.heatmap(matriz_correlacion, annot=True, cmap='coolwarm')
```

Finalmente, visualizarla ⇒

Pyplot (V)

Aunque ***matplotlib*** no requiere trabajar con dataframes. Puedes **crear gráficos a partir de datos** organizados en listas, trabajaremos con ellos ya que resulta más fácil.

- Mediante las distintas opciones **podemos darle nombre a los ejes, añadir leyendas**, etc.
- Pandas tiene integración con matplotlib, de modo que dispone de varios métodos para la creación de los distintos gráficos.
- Agrega de forma automática leyendas, nombre de ejes, etc.

Documentación de referencia:

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html

A partir de aquí trabajaremos con el fichero

VUESTRO TURNO