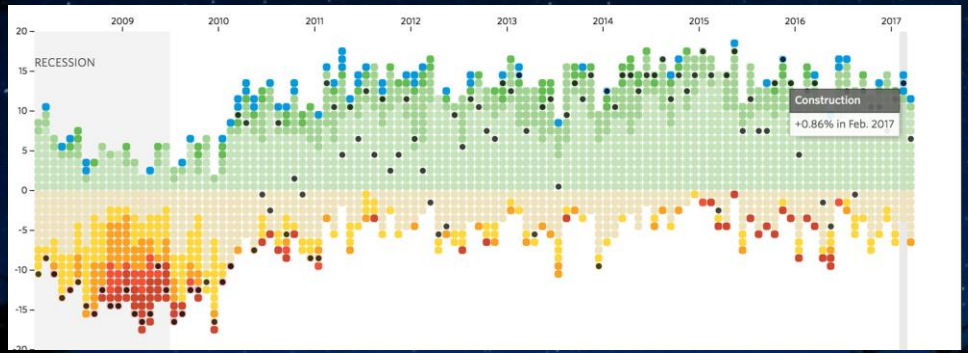
Visualización y Preprocesamiento

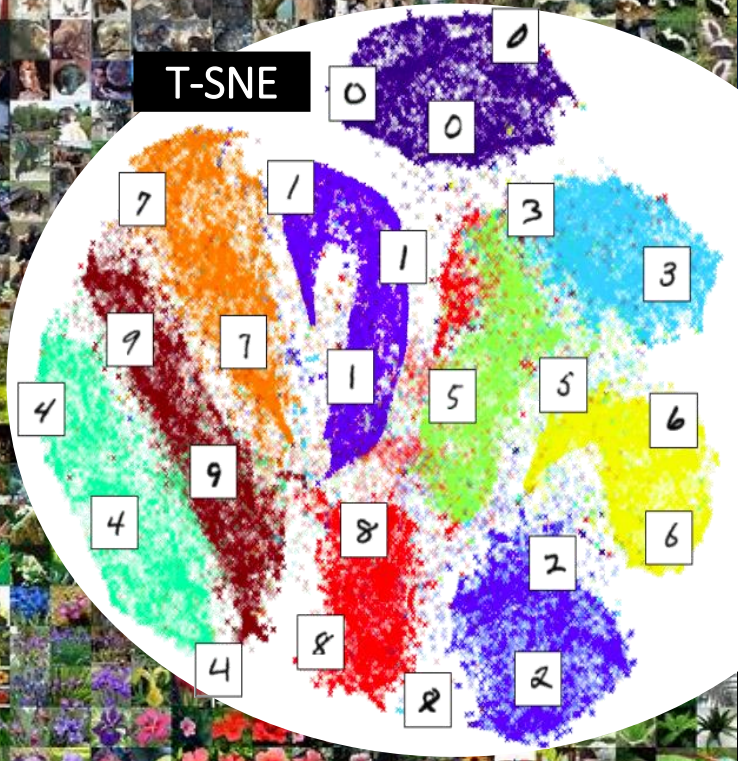
Visualización

Visualizar los datos puede ser una tarea esencial para entender el dominio en el que estamos trabajando.
Al tener más de 2 variables, visualizar puede no ser trivial.

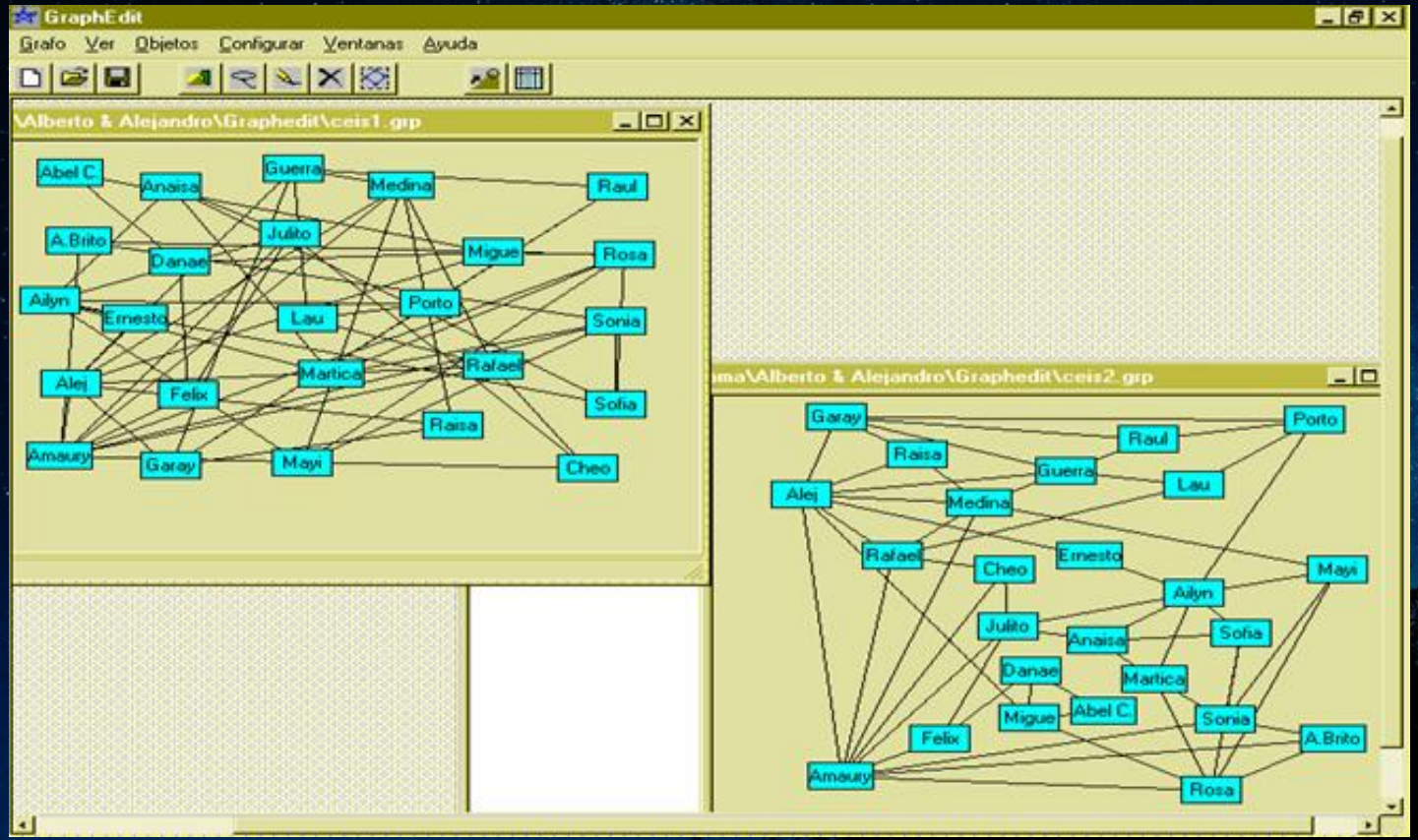


Visualización

Visualizar datos N-D en un espacio 2-D.



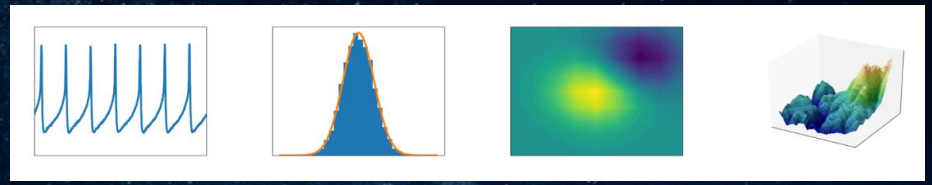
Visualización



Visualización

Matplotlib

matplotlib Version 2.2.2



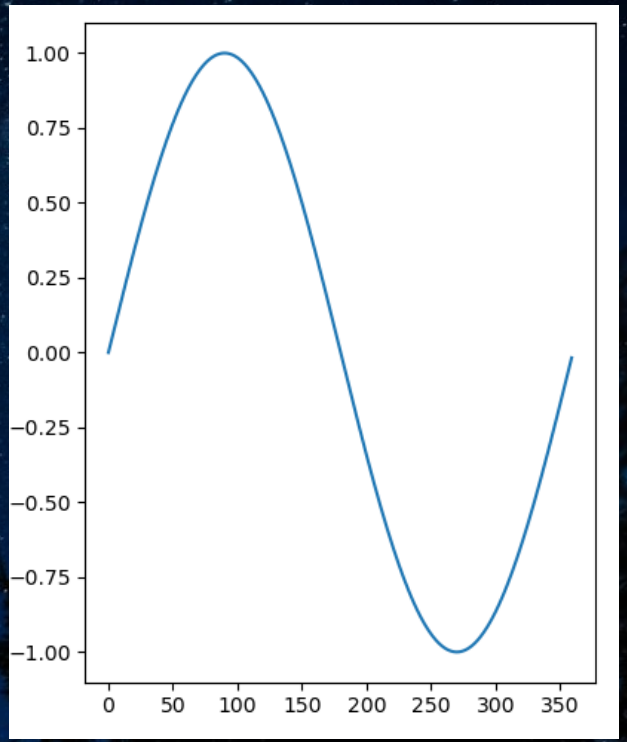
Es la librería gráfica más utilizada para realizar gráficos en Python.
https://matplotlib.org/tutorials

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

Matplotlib

La función plot(x,y) permite realizar un gráfico 2D de la variable *y* en función de *x*.
Copie y pegue el siguiente código en un nuevo documento de Python.

```
import numpy as np
import matplotlib.pyplot as plt
x= np.arange(0, 360)
y= np.sin(x * np.pi / 180.)
plt.plot(x,y)
```



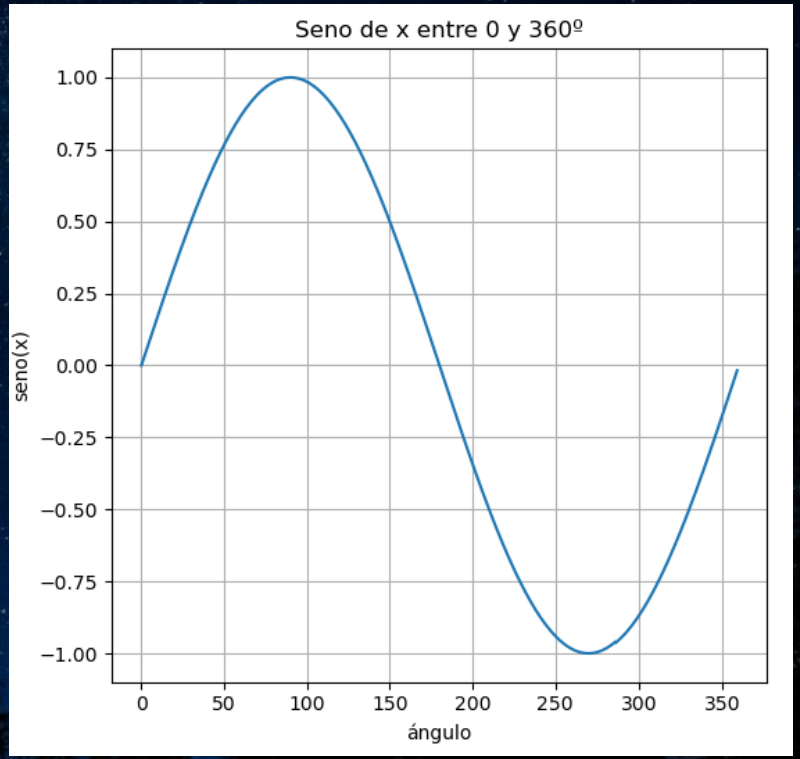
Matplotlib

```
plt.figure()
```

Nueva figura. Sino, todo lo ejecutado por matplotlib será impreso en la figura activa.

Matplotlib permite adornar las figuras con mucha información adicional.

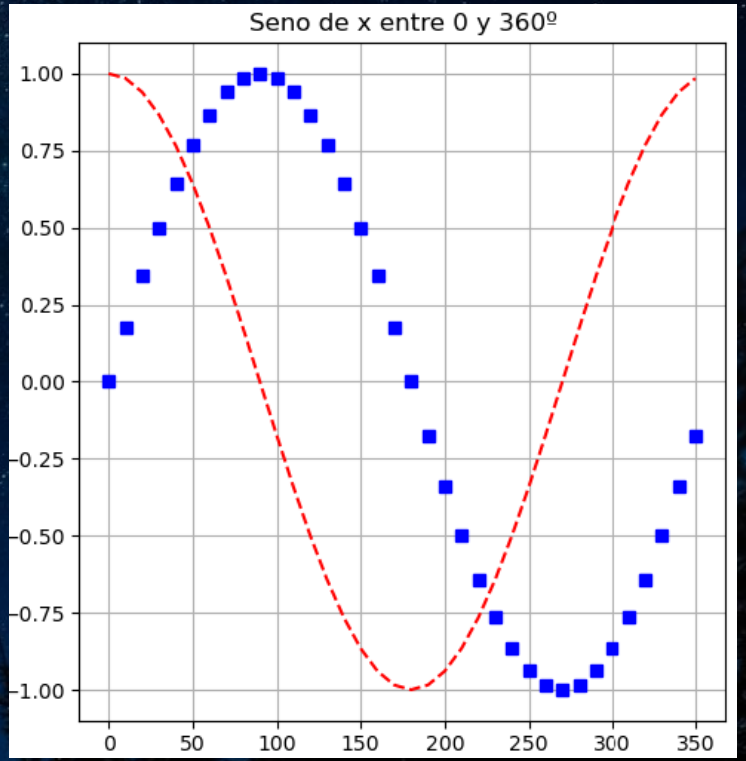
```
plt.xlabel('ángulo')
plt.ylabel('seno(x) ')
plt.title('Seno de x entre 0 y 360º')
plt.grid()
plt.savefig('figura.png')
```



Matplotlib

Graficando con diferentes marcadores y líneas.

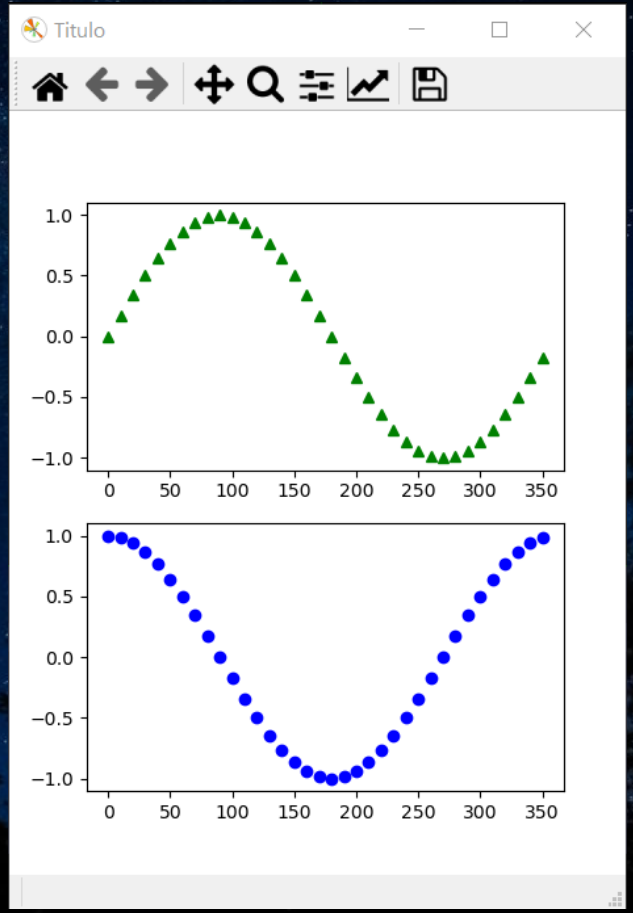
```
y= np.sin(x * np.pi / 180.)
plt.plot(x,y, 'bs')
y= np.cos(x * np.pi / 180.)
plt.plot(x,y, '--r')
```



Matplotlib

Graficando en varias subfiguras.

```
plt.figure('Titulo')

plt.subplot(211) # 2 x 1, indice=1
y= np.sin(x * np.pi / 180.)
plt.plot(x, y, 'g^')

plt.subplot(212) # 2 x 1, indice=2
y= np.cos(x * np.pi / 180.)
plt.plot(x, y, 'bo')
```

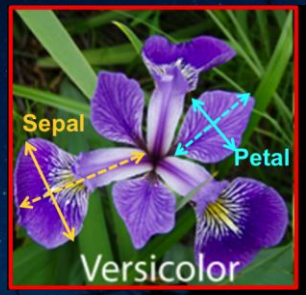


Visualización Iris Dataset

El dataset “Iris” está almacenado en un archivo “cvs” (Comma Separated Values). Estos son archivos de texto plano, donde cada registro está delimitado por un Enter (retorno de carro), y cada columna por una coma (,).

El dataset contiene información sobre 3 especies distintas de flores. Contiene 4 atributos: tamaño y largo del pétalo, tamaño y largo del sépalo.

	A	B	C	D	E
1	sepal_length	sepal_width	petal_length	petal_width	name
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3.0	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5.0	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5.0	3.4	1.5	0.2	setosa



Pandas

https://pandas.pydata.org/

- Pandas es una librería de código abierto, fácil de usar, que provee manejo para carga y análisis de datos, entre otras funcionalidad.
- Es una librería muy utilizada para cargar archivos CSV o XLS.
- Cada columna tiene nombres, y pueden contener información de diferente tipo.
- Muchas funciones de visualización y preprocesamiento.

Pandas

Abriendo Iris Dataset con Pandas

```
import pandas as pd
iris = pd.read_csv("iris.csv")
```

Crea un DataFrame

Pandas - Iris

Desde una consola, con este comando podemos ver los primeros datos en el dataframe.

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	name
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Pandas - Iris

Desde una consola, con este comando podemos ver un análisis estadístico rápido de las variables.

```
iris.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Pandas - Iris

Cuenta la cantidad de ítems en la columna seleccionada.

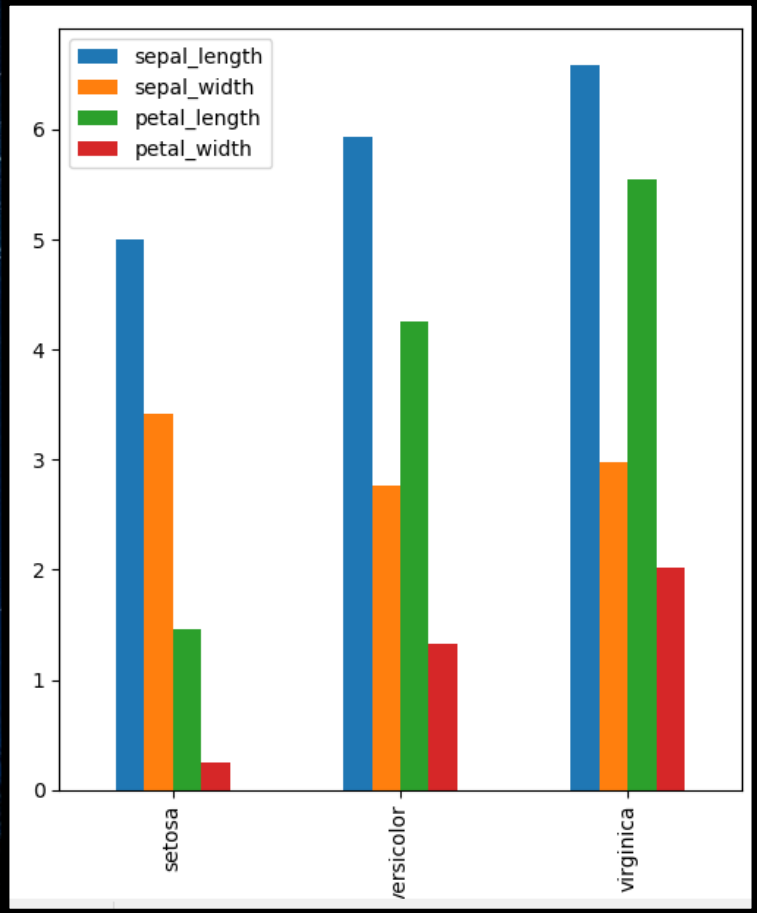
```
iris['name'].value_counts()
```

```
versicolor    50
setosa        50
virginica     50
```

Pandas - Iris

Gráfico de barras de la media de cada atributo, diferenciado por clase.

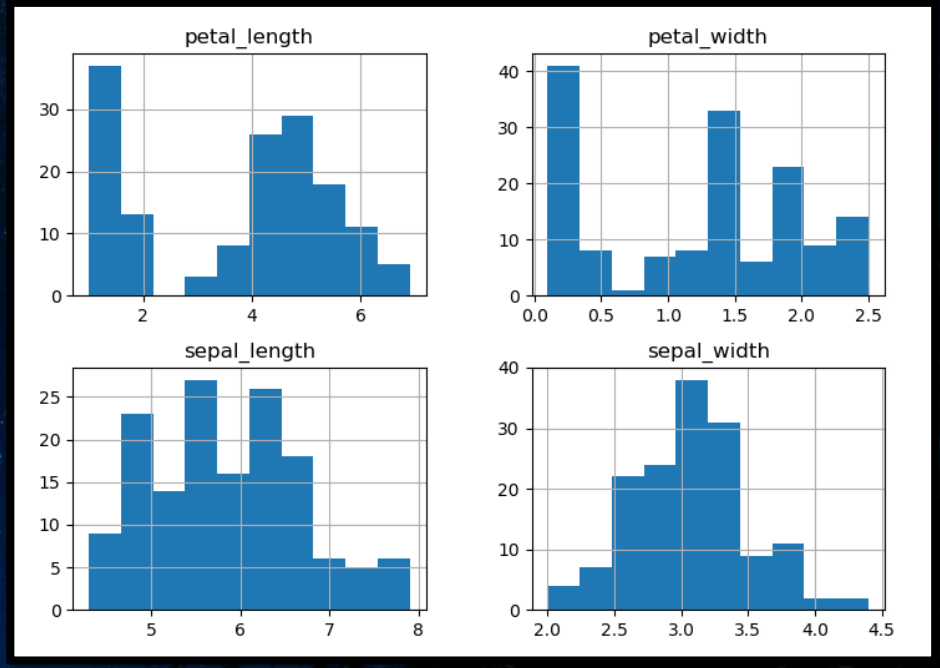
```
plot_data= iris.groupby('name').mean()

plot_data.plot(kind='bar')
```



Pandas - Iris

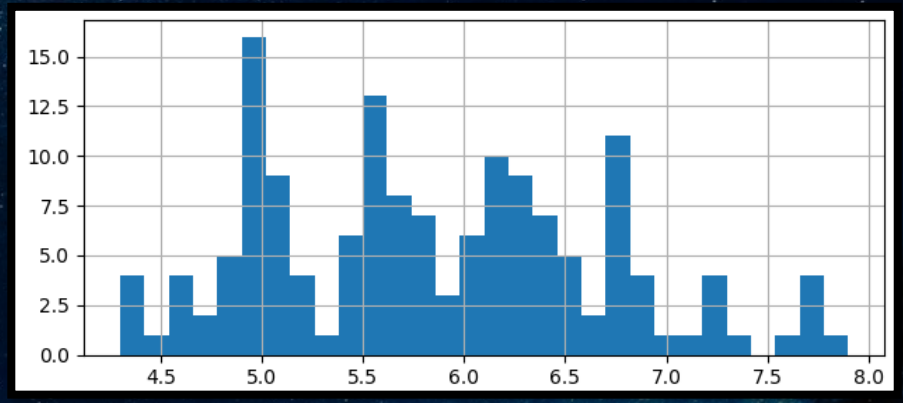
Crea un histograma para cada columna del dataframe

```
iris.hist()
```



Pandas - Iris

Crea un histograma para la columna “sepal_length”




```
iris.sepal_length.hist(bins=30)
```

Pandas - Iris

Matriz de correlación entre features

```
iris.corr()
```

Index	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1	0.962757
petal_width	0.817954	-0.356544	0.962757	1

Pandas - Iris

Nombre	Tipo	Tamaño	Valor
ax	axes._subplots.Axes…	1	AxesSubplot object of matplotlib.axes._subplots module
blue	tuple	3	(0.16696655132641292, 0.48069204152249134, 0.7291503267973857)
class_number	int64	1	2
data	Array of float64	(150, 5)	[[-0.90068117 1.03205722 -1.3412724 -1.312 [-1.14 ...
f	figure.Figure	1	Figure object of matplotlib.figure module
fig	figure.Figure	1	Figure object of matplotlib.figure module
iris	DataFrame	(150, 5)	Column names: sepal_length, sepal_width, petal_length, petal_width, na

Explorador de variables | Gráfico

Visualización del dataframe con Spyder

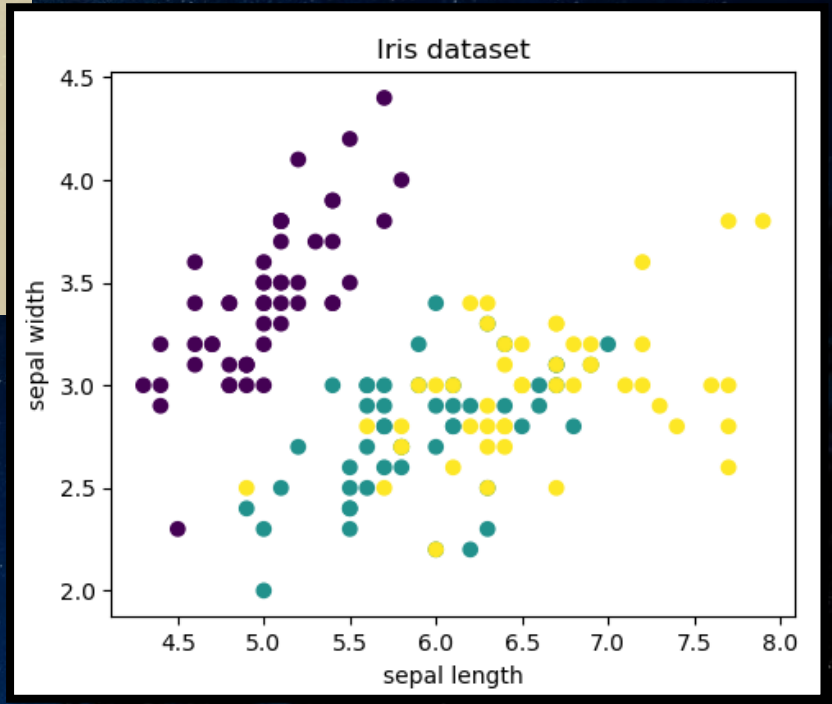
Index	sepal_length	sepal_width	petal_length	petal_width	name
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa

Scatter

Scatter permite visualizar una dispersión de puntos en un espacio 2D.

Al tener 4 variables (iris), no podemos visualizar todas al mismo tiempo, pero podemos hacer un corte 2D para dos de ellas. Equivalente a mirar las estrellas en la noche, sin saber a qué distancia están.

```
plt.scatter(iris.sepal_length,
            iris.sepal_width,
            c=iris.name)
plt.title('Iris dataset')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
```

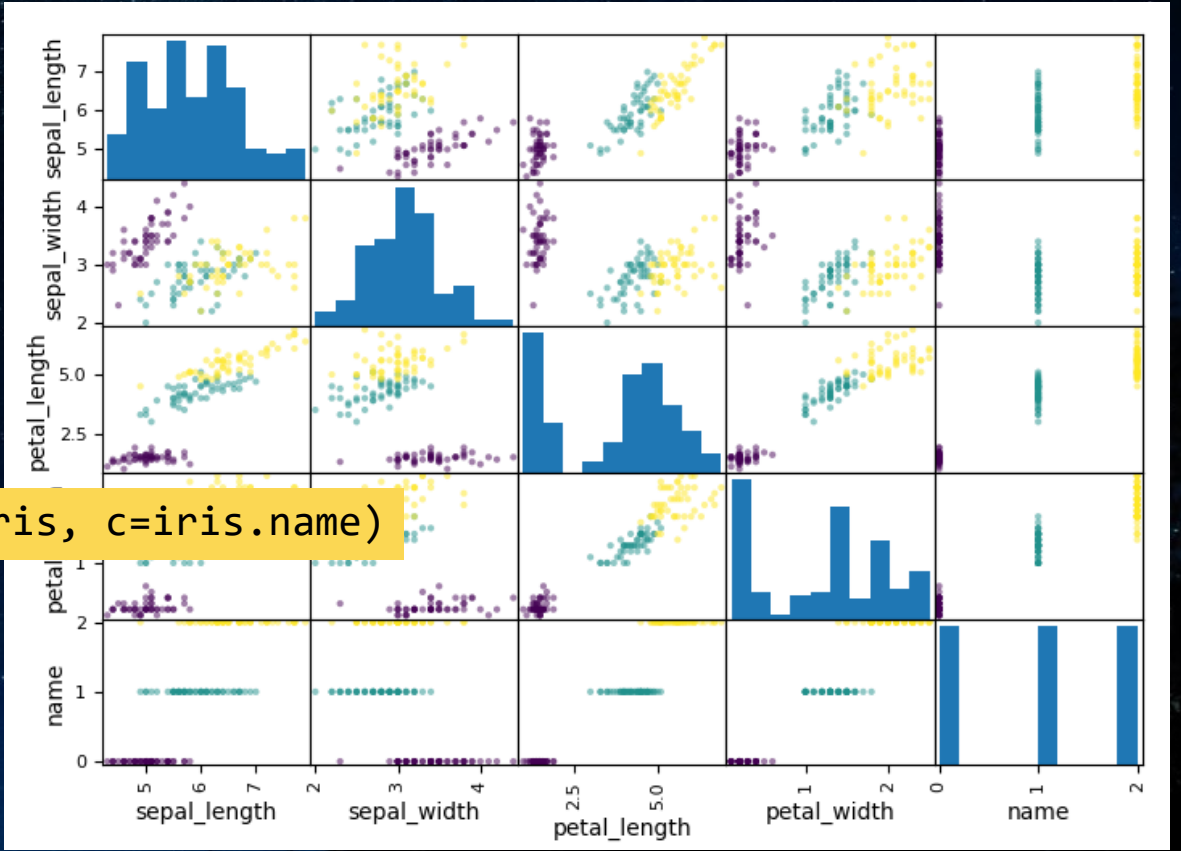


Pandas - Iris

Matriz con scatter 2D para cada par de features

(Cuidado si son muchos features !)

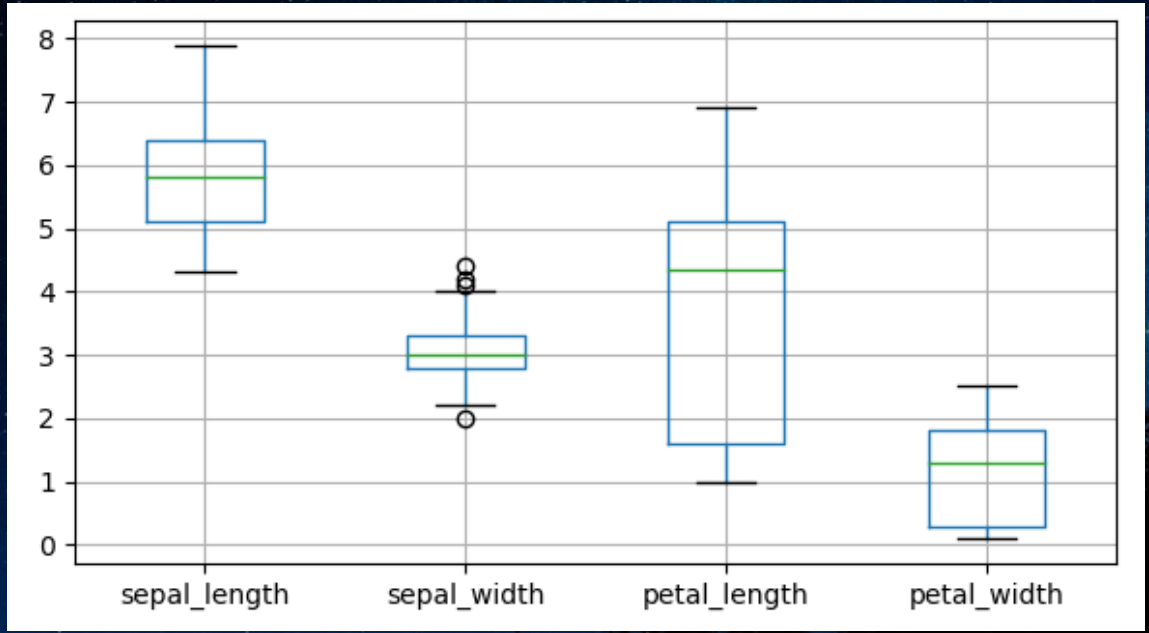
```
pd.plotting.scatter_matrix(iris, c=iris.name)
```



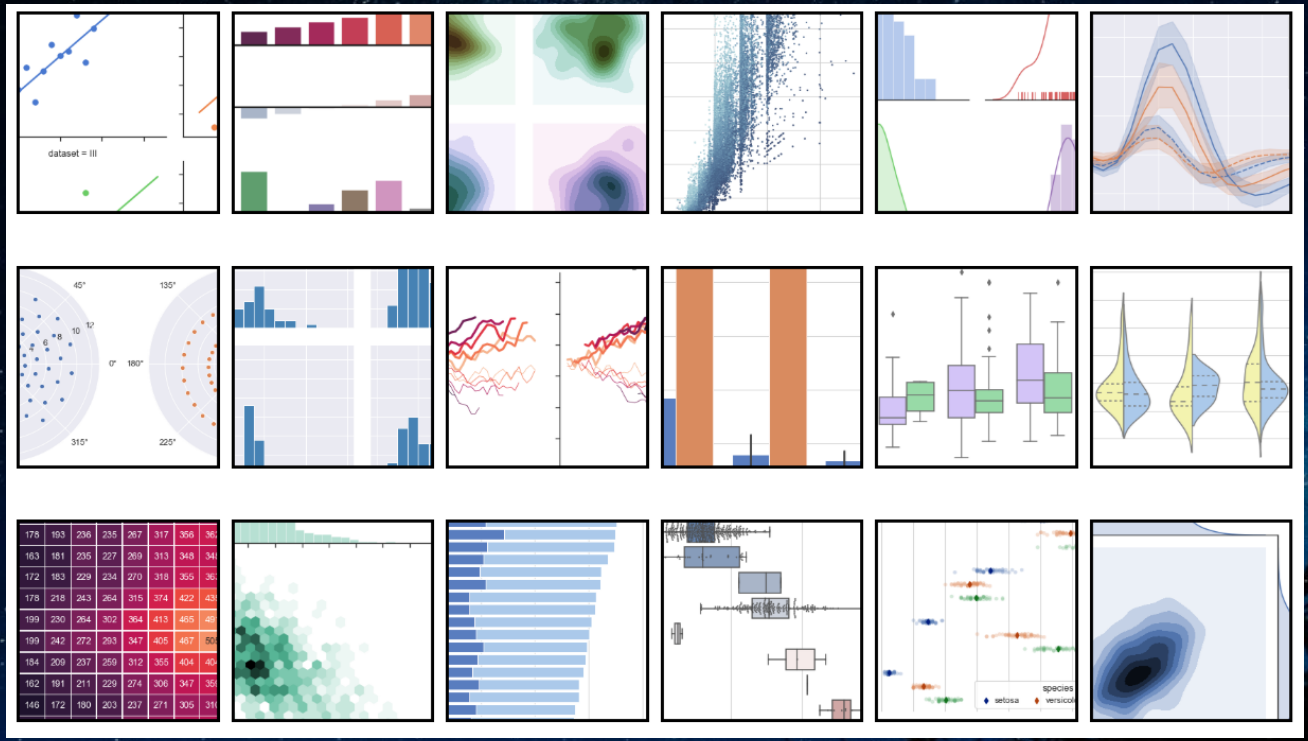
Pandas - Iris

Diagramas de Caja

```
iris.boxplot()
```



Seaborn visualization

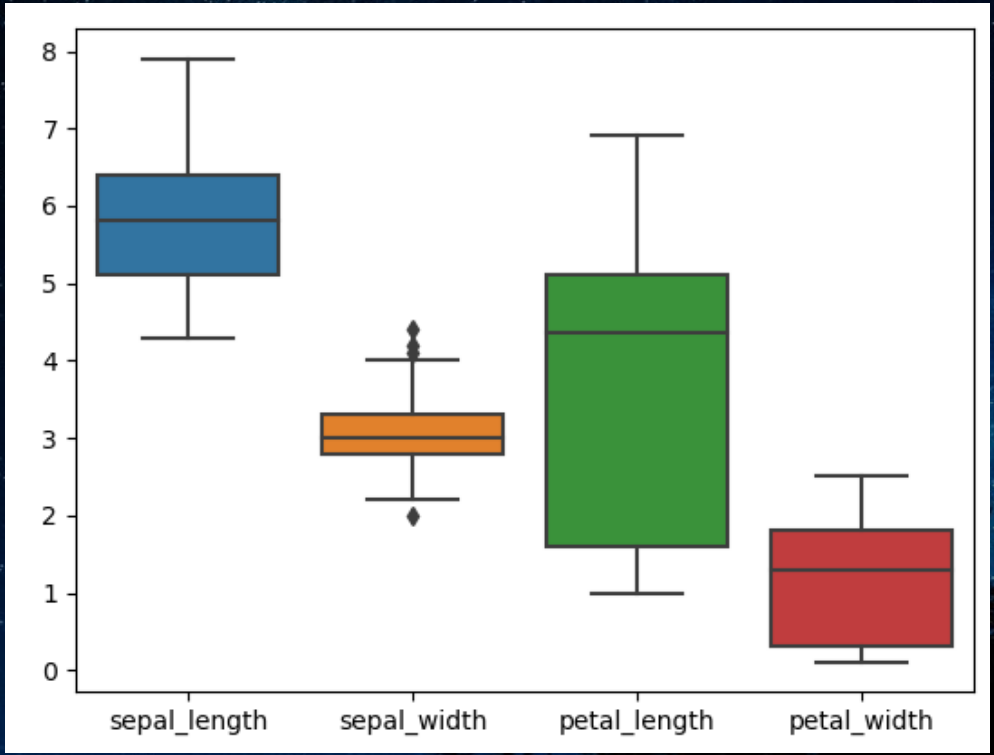


https://seaborn.pydata.org/

https://www.kaggle.com/noelano/seaborn-visualization-on-iris-data-set

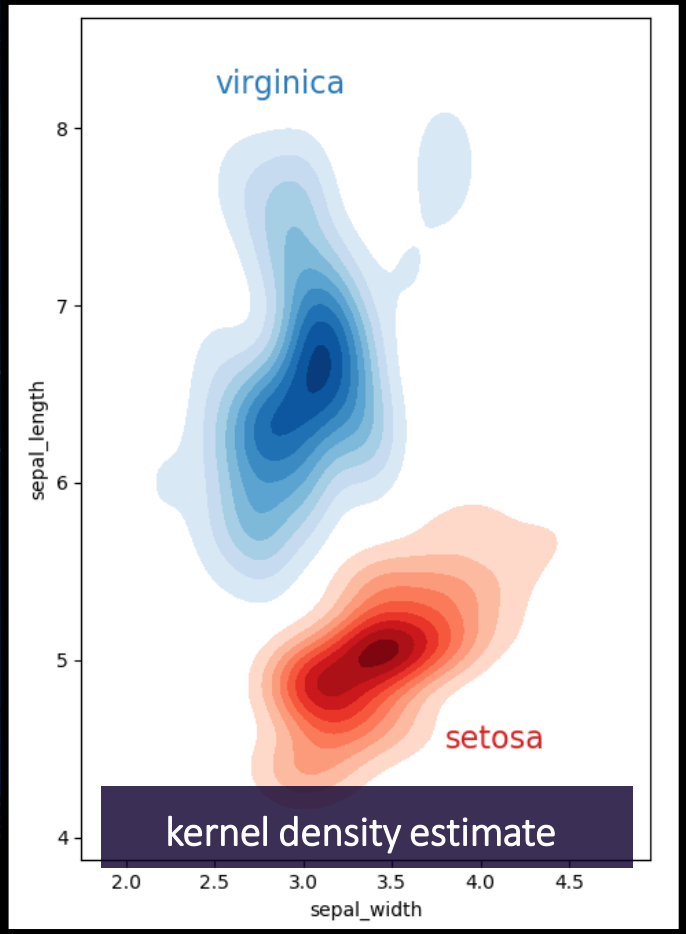
Seaborn visualization

```
sns.boxplot(data=iris)
```



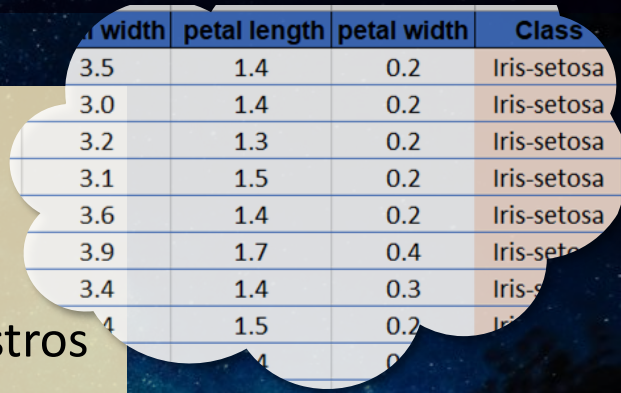
Seaborn visualization

```
setosa = iris.query("name == 'setosa'")
virginica = iris.query("name == 'virginica'")

# Set up the figure
f, ax = plt.subplots(figsize=(8, 8))
ax.set_aspect("equal")

# Draw the two density plots
ax = sns.kdeplot(setosa.sepal_width,
setosa.sepal_length,
                 cmap="Reds", shade=True,
shade_lowest=False)
ax = sns.kdeplot(virginica.sepal_width,
virginica.sepal_length,
                 cmap="Blues", shade=True,
shade_lowest=False)

# Add labels to the plot
red = sns.color_palette("Reds")[-2]
blue = sns.color_palette("Blues")[-2]
ax.text(2.5, 8.2, "virginica", size=16, color=blue)
ax.text(3.8, 4.5, "setosa", size=16, color=red)
```



Preprocesamiento

Preprocesamiento de datos



width	petal length	petal width	Class
3.5	1.4	0.2	Iris-setosa
3.0	1.4	0.2	Iris-setosa
3.2	1.3	0.2	Iris-setosa
3.1	1.5	0.2	Iris-setosa
3.6	1.4	0.2	Iris-setosa
3.9	1.7	0.4	Iris-set
3.4	1.4	0.3	Iris-s
4	1.5	0.2	Iri
	4	0	

3 pasos importantes:

- Revisar valores nulos/faltantes

  Un dataset real puede tener muchos registros corruptos

- Manejar datos nominales

  Pasar de texto a números (con imágenes este paso no existe)

- Normalizar datos

  Los modelos de ML que veremos necesitan datos normalizados.

Valores nulos

¿Qué hacer con los valores nulos?

No siempre es simple la decisión.

- **Eliminar filas**: lo más simple es siempre que haya un valor nulo eliminar todo el registro.
- **Eliminar Columnas:** otra opción puede ser eliminar una variable particular que contenga una proporción muy grande de valores nulos.
- **Reemplazar:** Se debe tener mucho cuidado en este caso, ya que estamos inventando los datos. Existe diferentes aproximaciones:
  - **Reemplazar por la media:** Muy peligroso en Machine Learning! Podrían quedar registros incoherentes.
  - **Reemplazar por un valor** calculado con una regresión u otro algoritmo de ML.
  - **Análisis experto**. En ocasiones la respuesta puede ser más simple de lo imaginado si conocemos bien el dominio.

Nominal a categórico

Convertir atributo nominal a categóricos (setosa=0, vers= 1, virg= 2).

Es necesario para que los algoritmos interpreten el atributo “etiqueta”.

```
iris.name= pd.Categorical(iris.name)
iris.name= iris.name.cat.codes
```

Índice	sepal_length	sepal_width	petal_length	petal_width	name
0	5	2	3.5	1	1
1	6	2.2	5	1.5	2
2	6.2	2.2	4.5	1.5	1
3	6	2.2	4	1	1
4	5	2.3	3.3	1	1
5	6.3	2.3	4.4	1.3	1
6	5.5	2.3	4	1.3	1
7	4.5	2.3	1.3	0.3	0

Guardar previamente los nombres de las etiquetas

```
Labels= np.unique(iris.name)
```

Nominal a One-hot

Cambiar un atributo nominal (texto) a codificación one-hot.

La codificación one-hot tiene tantas columnas como etiquetas existan, pero solo una está activa a la vez.

```
iris= pd.get_dummies(data=iris, columns=['name'])
```

sepal_length	sepal_width	petal_length	petal_width	name_setosa	name_versicolor	name_virginica
4.5	2.3	1.3	0.3	1	0	0
4.4	3.2	1.3	0.2	1	0	0
5	3.5	1.6	0.6	1	0	0
5.1	3.8	1.9	0.4	1	0	0
4.8	3	1.4	0.3	1	0	0
			.2	1	0	0
			.2	1	0	0
5.3	3.7	1.5	0.2	1	0	0
5	3.3	1.4	0.2	1	0	0
7	3.2	4.7	1.4	0	1	0
6.4	3.2	4.5	1.5	0	1	0

Detectar valores nulos

Reporte de valores nulos por columna

```
iris.isnull().sum()
```

Detectar cantidad total de valores nulos

```
iris.isnull().sum().sum()
```

Eliminar filas o columnas

Eliminar cualquier fila con valores nulos

```
iris= iris.dropna()
```

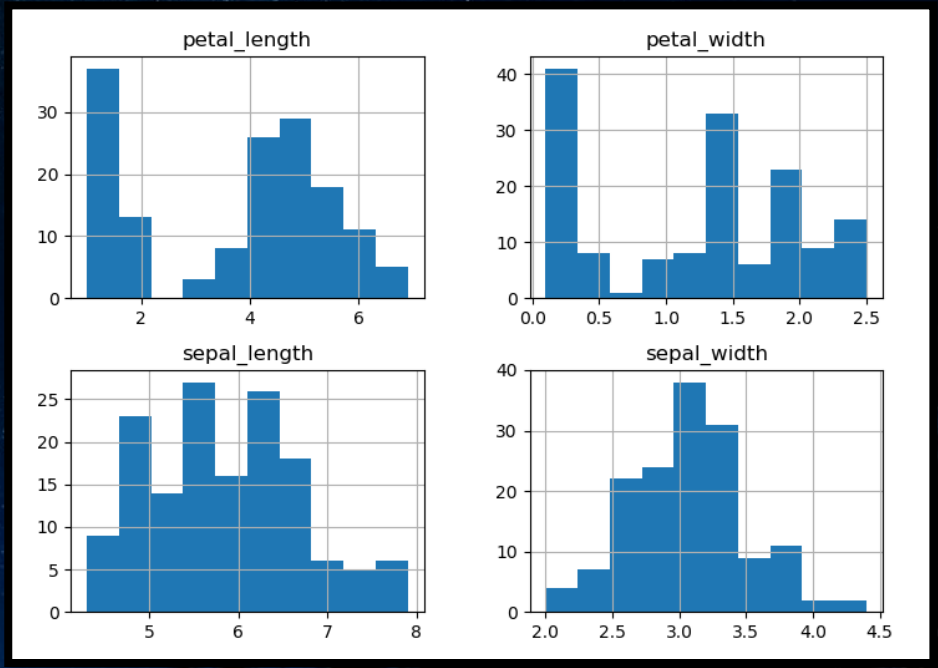
Eliminar la columna “name”.

```
X = iris.drop('name',axis=1)
```

Normalización de datos

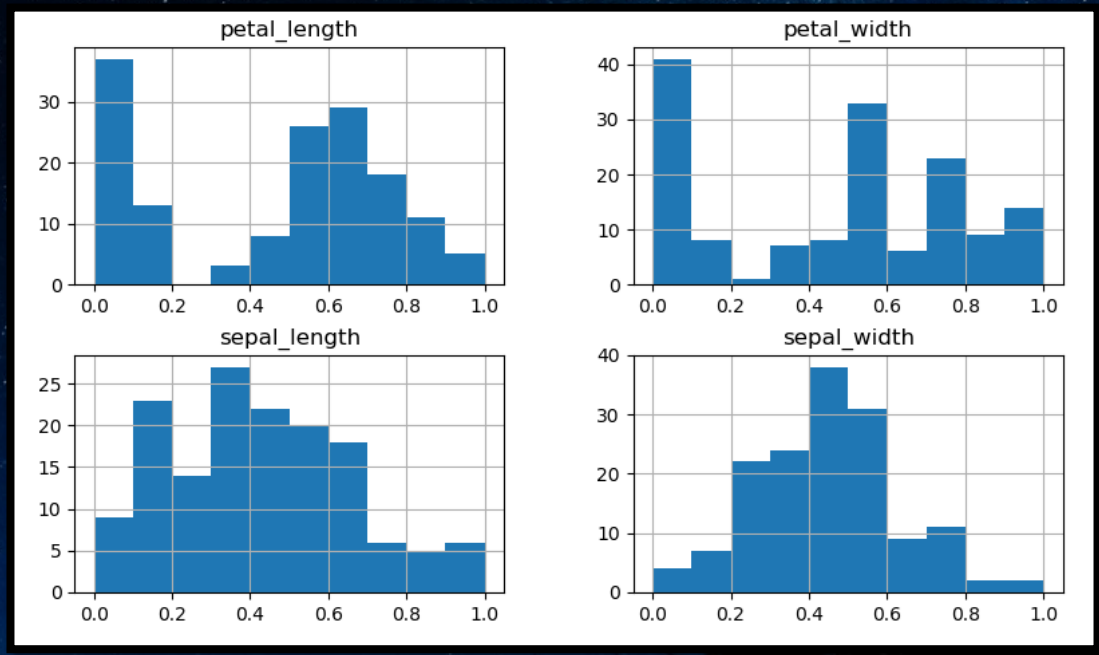
Para varios modelos que veremos, es un problema que cada feature posea su propio rango de datos. Necesitamos normalizarlos de algún modo

Ej. Iris Dataset



Normalización de datos

Min-max (0-1): x= (x-min(x)) / (max(x)-min(x))



Normalización de datos

Z-score (μ y σ): x= (x-mean(x)) / std(x)