

Aprendizaje Automático Profundo

Clase 1 - 2019

Profs: Franco Ronchetti - Facundo Quiroga



Horarios

LUNES – 15 -17:30hs – Sala de PC 1º piso

Primera mitad de consulta práctica.

A las 16hs comienza la teoría.

VIERNES – 9 - 11hs – Aula 10-A

Consulta práctica

Régimen de aprobación

Régimen de promoción

- Asistir al 70% de las clases teóricas y prácticas.
- Aprobar el examen que se tomará al finalizar el curso. Este examen cuenta con dos recuperatorios.
- Para promocionar la materia deberá obtener una calificación mayor o igual a 6 (seis) puntos.

Régimen convencional

- Examen escrito con nota mayor o igual a 4 (cuatro) puntos para obtener la cursada. Final escrito convencional.

Comunicación



Cartelera virtual



Material: <https://tinyurl.com/AAP-UNLP-2019>



Aprendizaje.Automatico.UNLP@gmail.com



gitter.im/aa2019unlp/community

Cronograma

Semana	Teoría	Práctica	Fecha
1	Introducción al aprendizaje automático y aprendizaje profundo. Python. Visualización y preprocesamiento.	1 - Visualización	26/8
2	Regresión Lineal		2/9
3	Regresión Logística. Introducción a Keras.	2 – Modelos básicos con Keras	9/9
4	Redes Neuronales 2. Capas. Funciones de activación. Tensor Flow/Keras. CPU vs. GPU.		16/9
5	Evaluación de Modelos 1. Etapas en un proyecto de aprendizaje automático. Métricas para evaluar modelos. Generalización y Sobreajuste.	3- Validación de modelos	23/9
6	Redes Convolucionales 1. Clasificación de Imágenes.		30/9
7	Redes Convolucionales 2. Filtros convolucionales. Pooling.	4 – Imágenes y CNN	7/10
-	Feriado. No hay clases toda la semana.		14/10
8	Tópicos avanzados.	5- Tópicos avanzados	21/10
9	Tópicos avanzados.		28/10
10	Repaso general.		4/11
11	1ra. Fecha de parcial. Muestra el viernes 15/11		11/11
12	Lunes Feriado. Consulta el viernes 22/11		18/11
13	2da. Fecha de parcial.		25/11
14	Muestra de exámenes de la 2da. Fecha. Consultas referidas a la 3era. Fecha de parcial		2/12
15	3ra. Fecha de parcial		9/12
16	Muestra de exámenes de la 3da. fecha		16/12

Objetivos

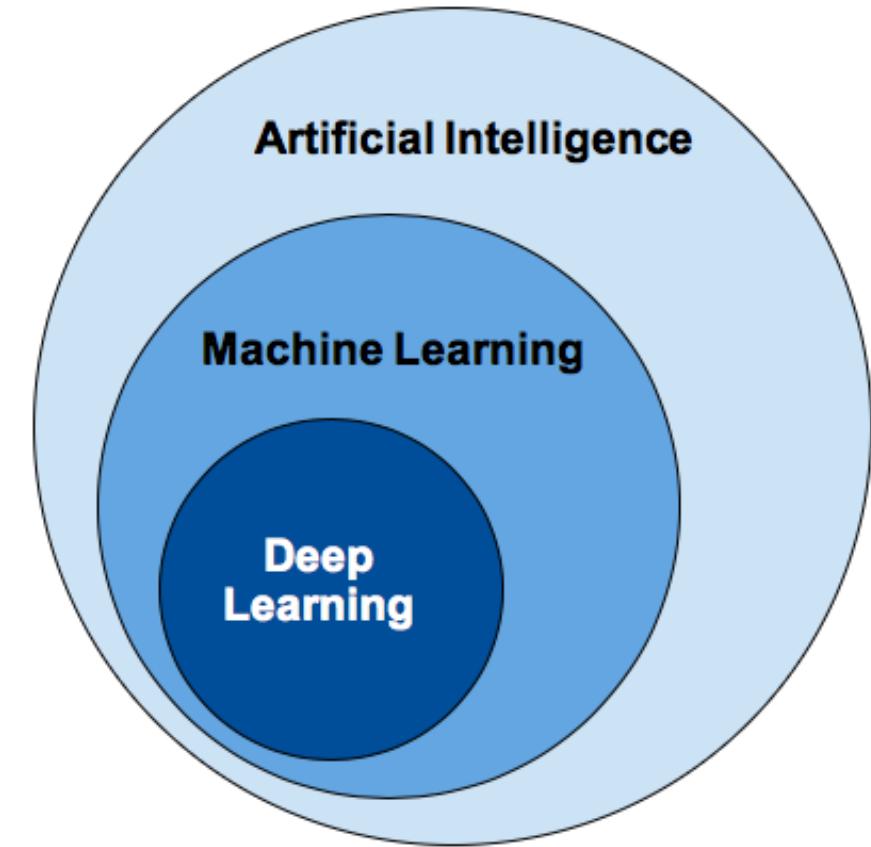
- Conocer los fundamentos básicos del aprendizaje automático.
- Conocer los principales modelos y algoritmos del aprendizaje automático profundo.
- Aprender lo que es una imagen digital y cuáles son sus propiedades.
- Entender los métodos de validación que se utilizan para los algoritmos planteados.
- Conocer herramientas y frameworks actuales en el área para poder llevar a cabo un desarrollo específico.

INTRODUCCIÓN

Aprendizaje Automático

El **Aprendizaje Automático (Machine Learning)** es una rama de la Inteligencia Artificial que estudia mecanismos que permite a las computadoras aprender a clasificar o predecir en base a experiencias.

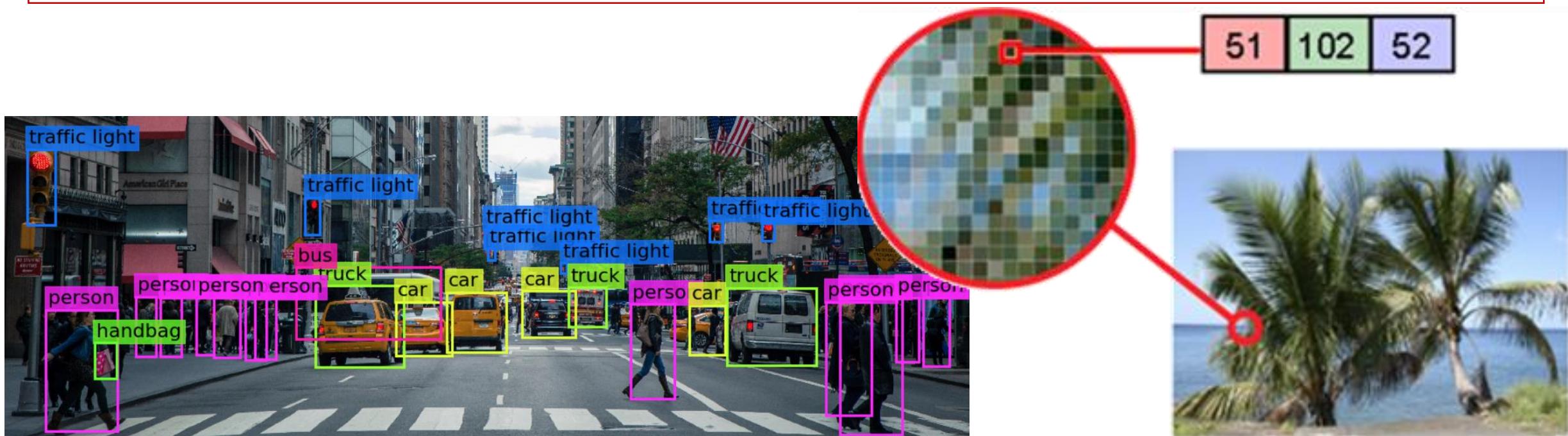
Definición de Arthur Samuel (1959): “área de estudio que da a las computadoras la habilidad de aprender sin ser explícitamente programadas”.



Visión por Computadora. Historia

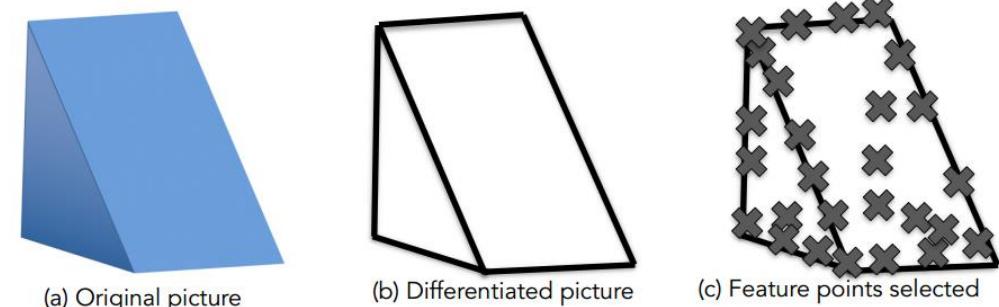
Conjunto de técnicas informáticas desarrolladas con el fin de interpretar y procesar imágenes digitales. Se intenta imitar (o mejorar) el sistema de visión humano.

Entre otras áreas, la Visión utiliza al Machine Learning para generar modelos capaces de reconocer objetos en imágenes/videos.

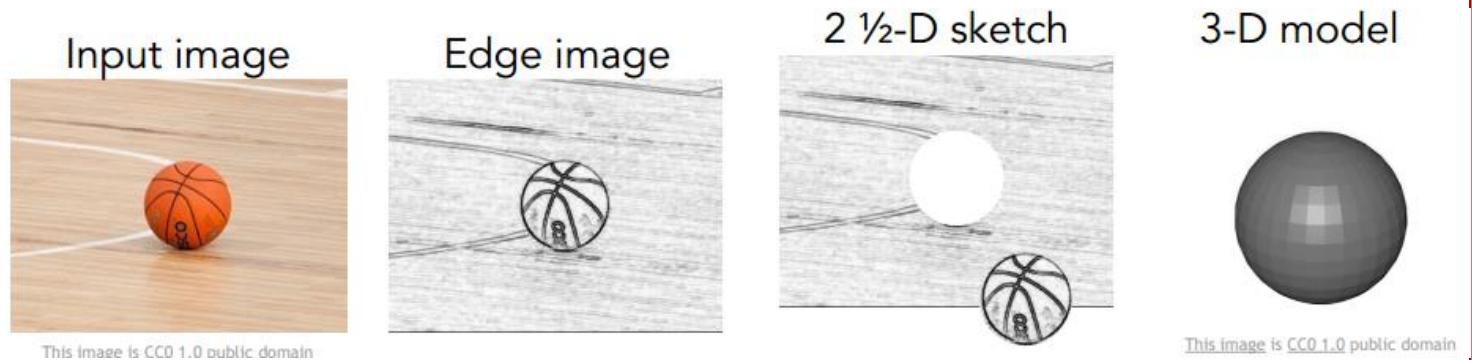


Visión por Computadora. Historia

- Años 60.
Reconstrucción de formas geométricas.
The MIT summer vision Project

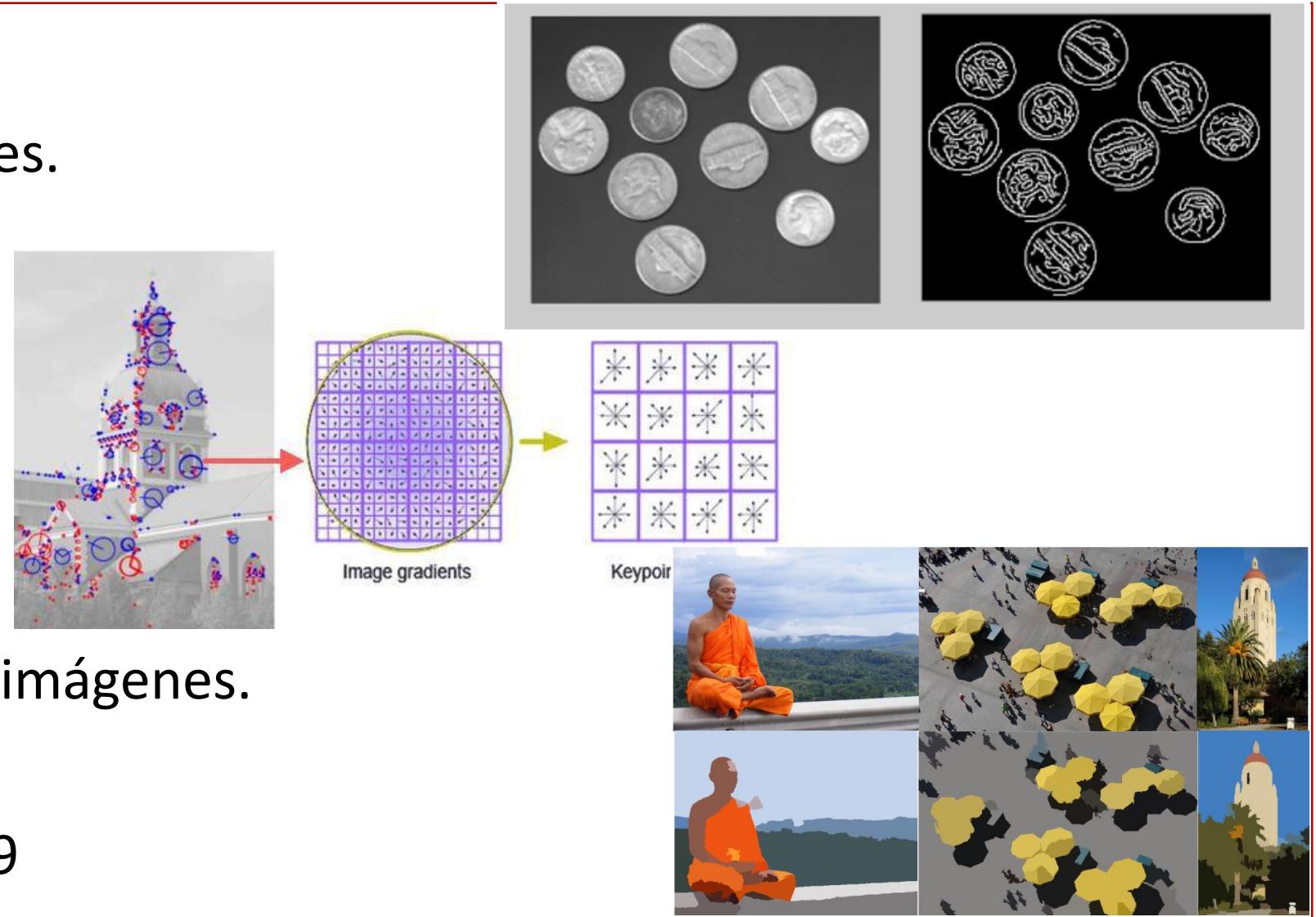


- Años 70.
Detección de bordes y
reconstrucción 3D.



Visión por Computadora. Historia

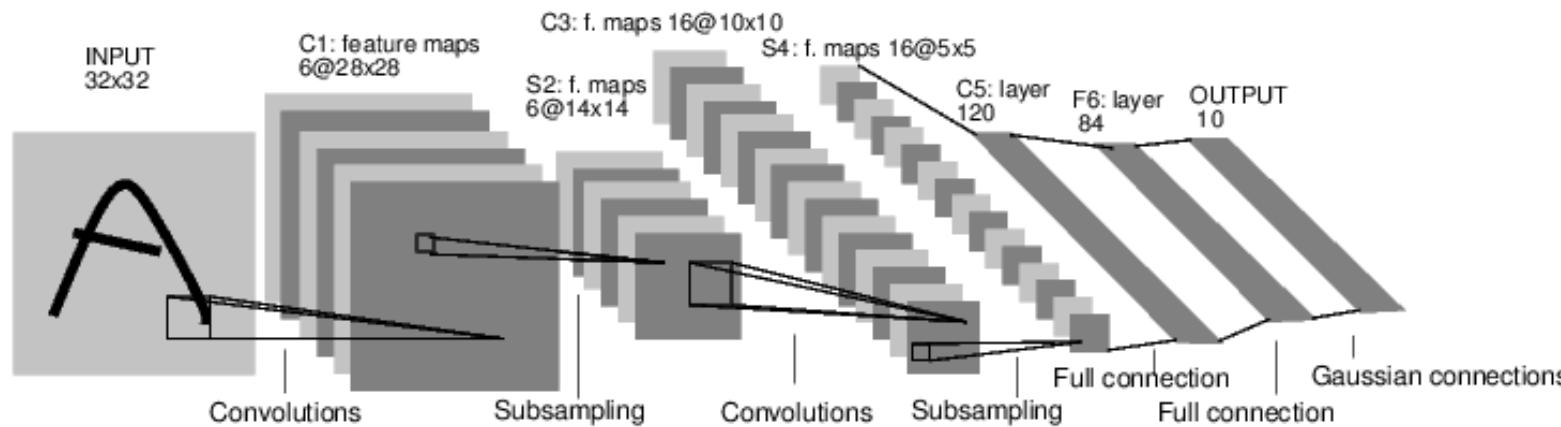
- Años 80.
David Lowe. Bordes.



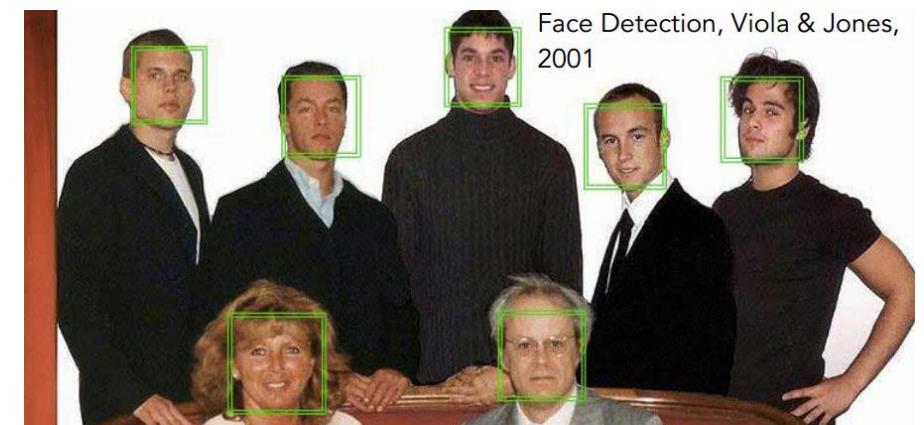
- Años 90.
Segmentación de imágenes.
Shi & Malik. 1997
D Lowe. SIFT. 1999

Visión por Computadora. Historia

- 1998. LeCun. Primera CNN. Reconocimiento de dígitos. No fue muy popular. Tampoco había GPU ni grandes bases de datos en imágenes.

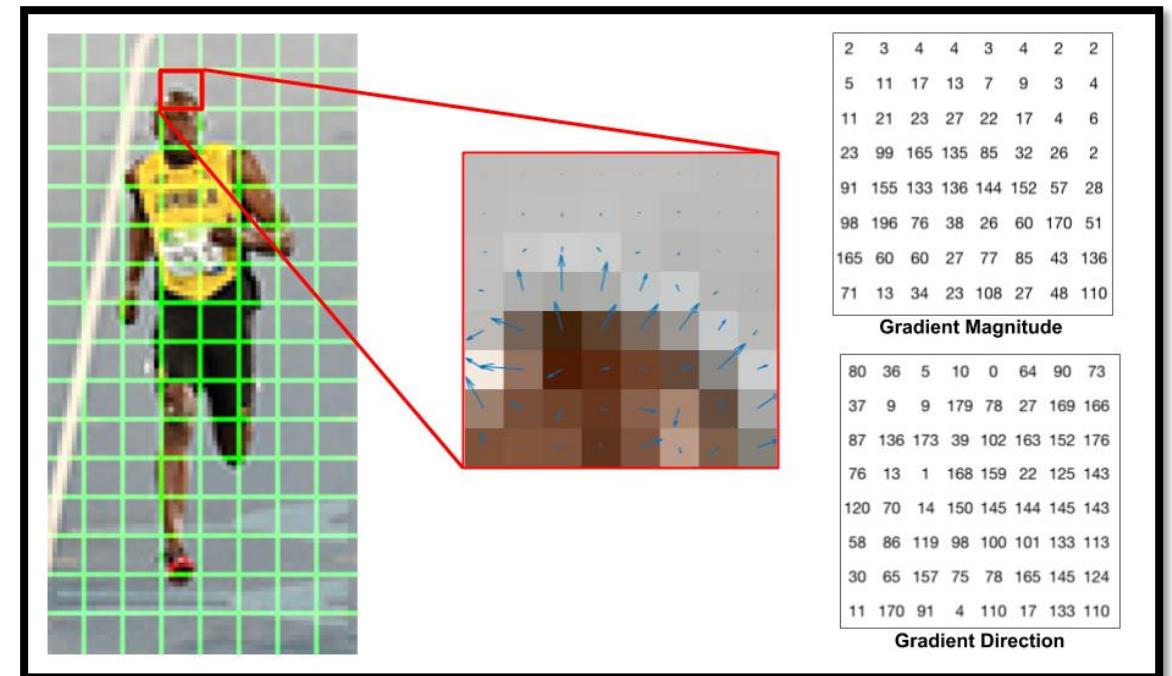
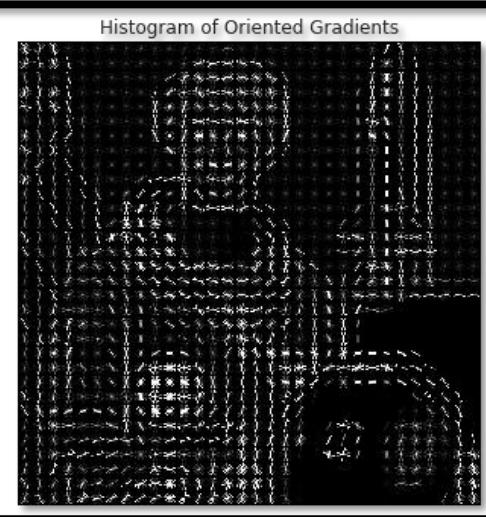


- 2001. Face detection. Viola & Jones.
Fuji lanzó la primera cámara con *face detection* en 2006.



Visión por Computadora. Historia

- 2005. Histogram of Gradients (HoG) Dalal & Triggs.



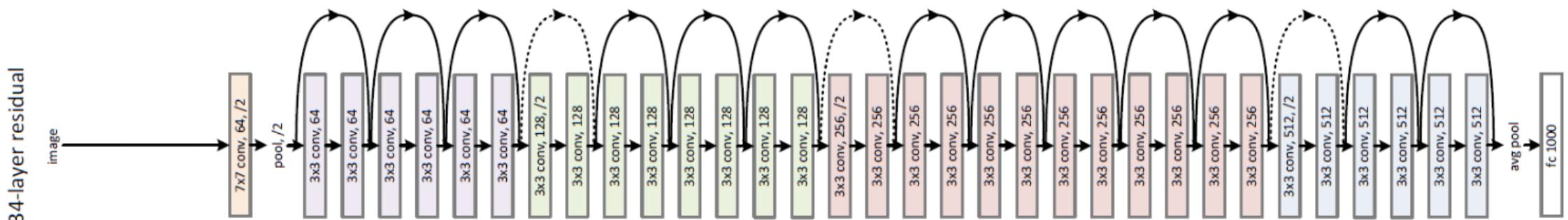
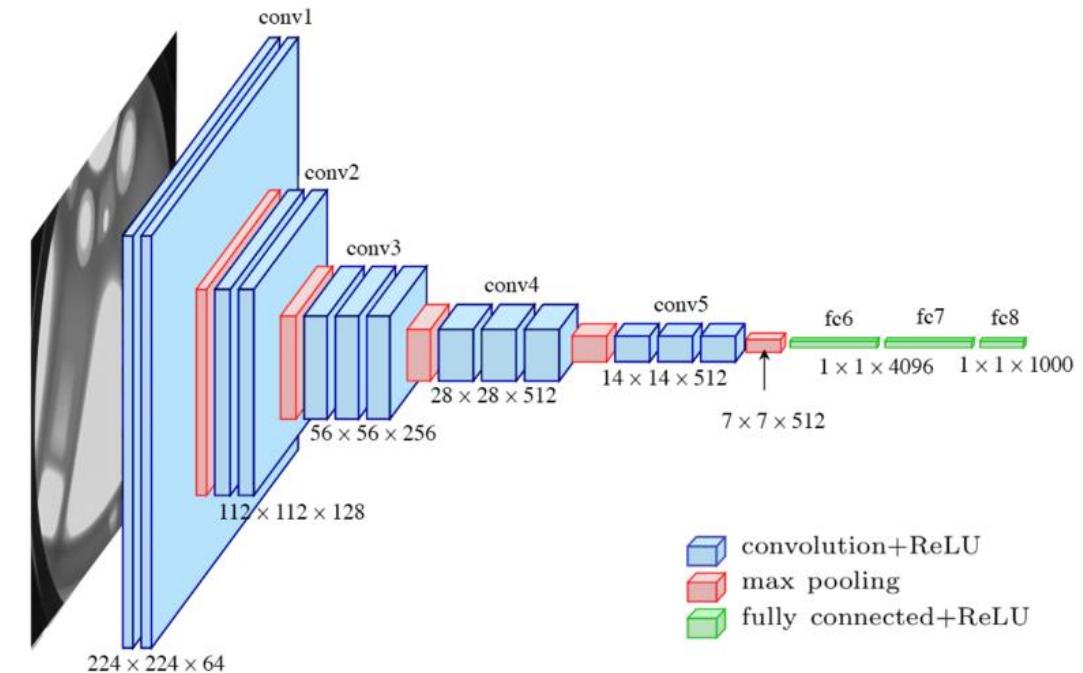
Visión por Computadora. Historia

- Actualidad

2014 VGG. 19 capas.

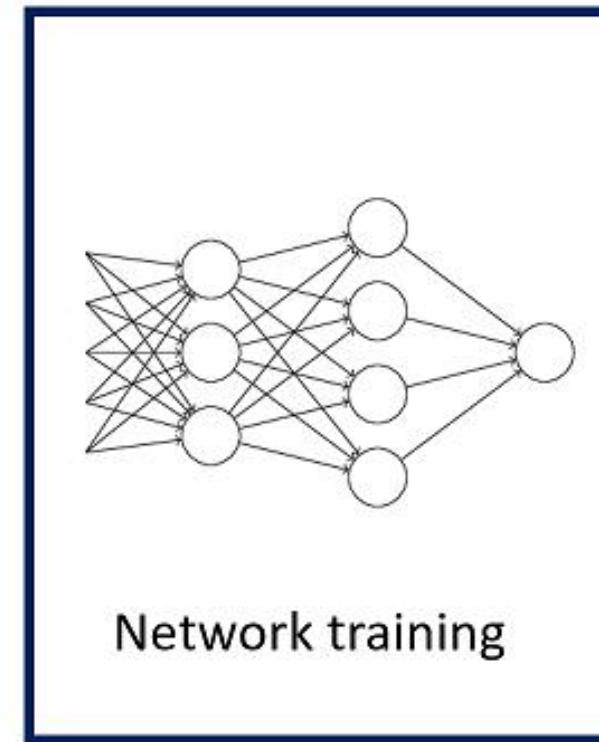
2014. GoogLeNet. 22 capas.

2015. Resnet (Microsoft). 152 capas.



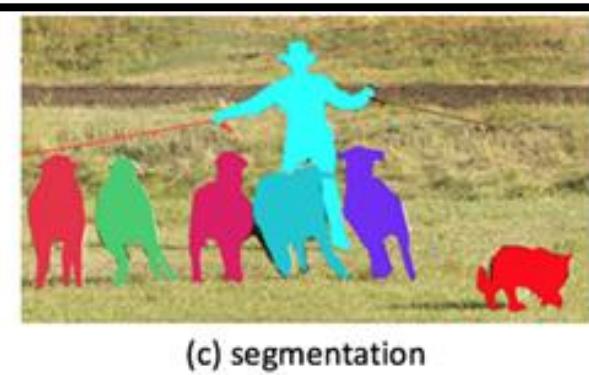
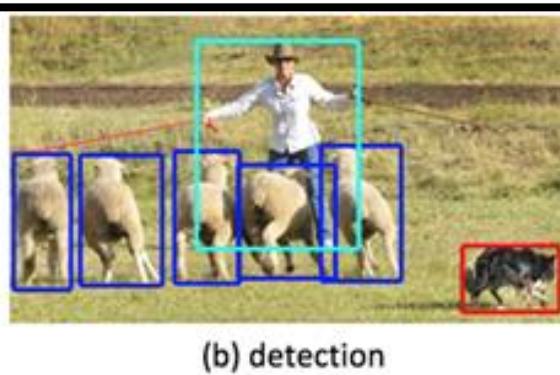
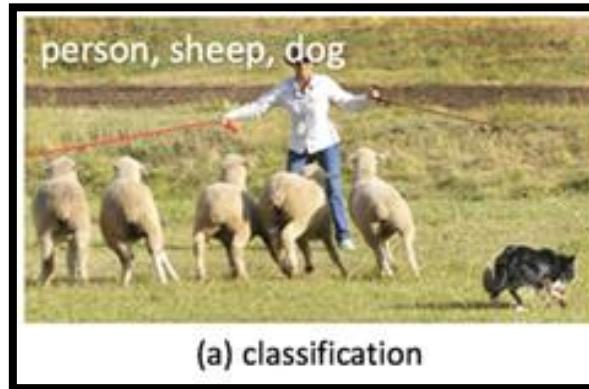
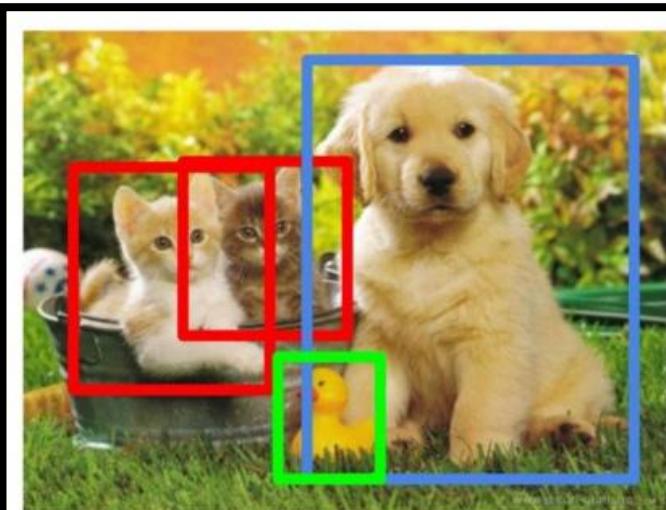
Aprendizaje Automático. Ejemplos. OCR

0000000000000000
1111111111111111
2222222222222222
3333333333333333
4444444444444444
5555555555555555
6666666666666666
7777777777777777
8888888888888888
9999999999999999

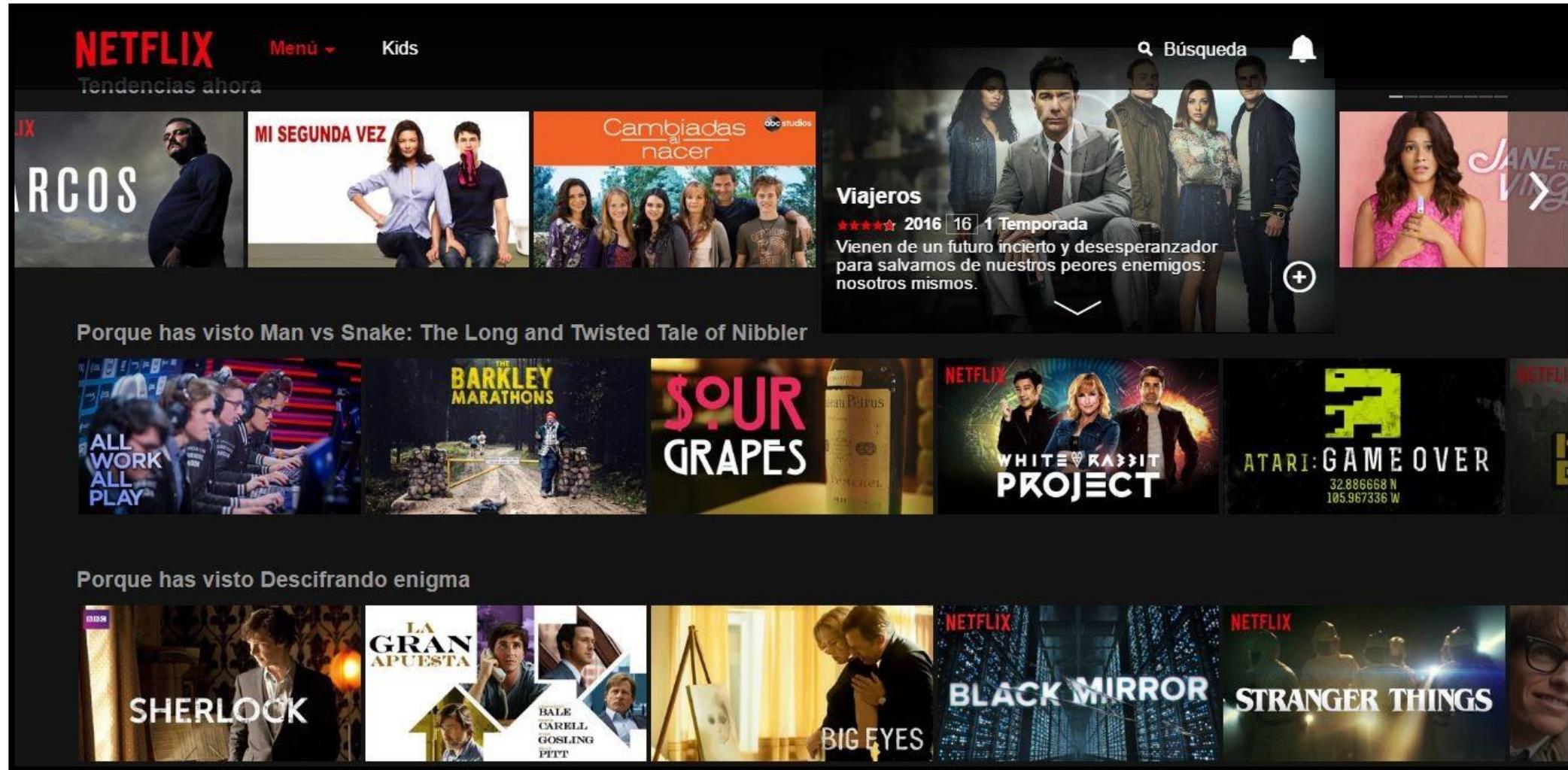


0
1
2
3
4
5
6
7
8
9

Ejemplos. Segmentación de objetos/ímagenes



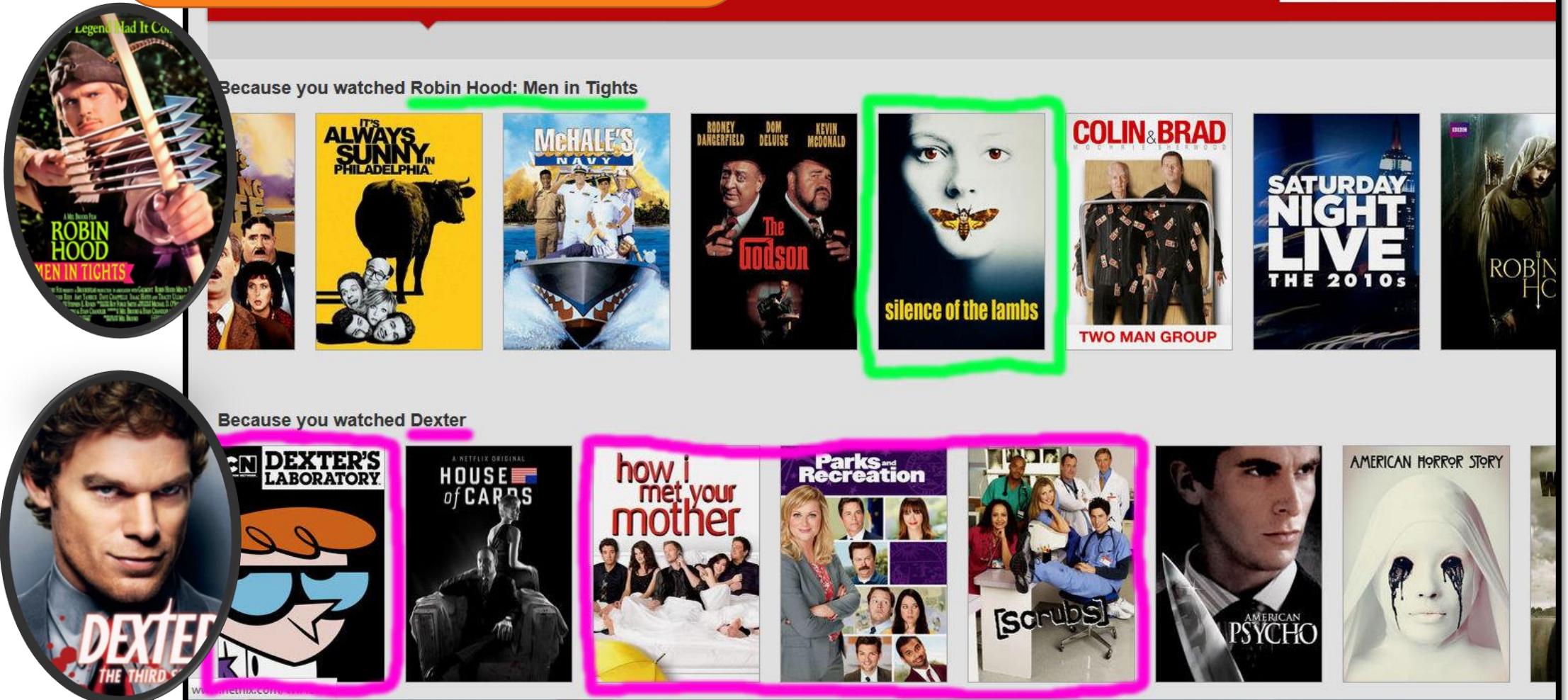
Ejemplos. Sistemas recomendadores.



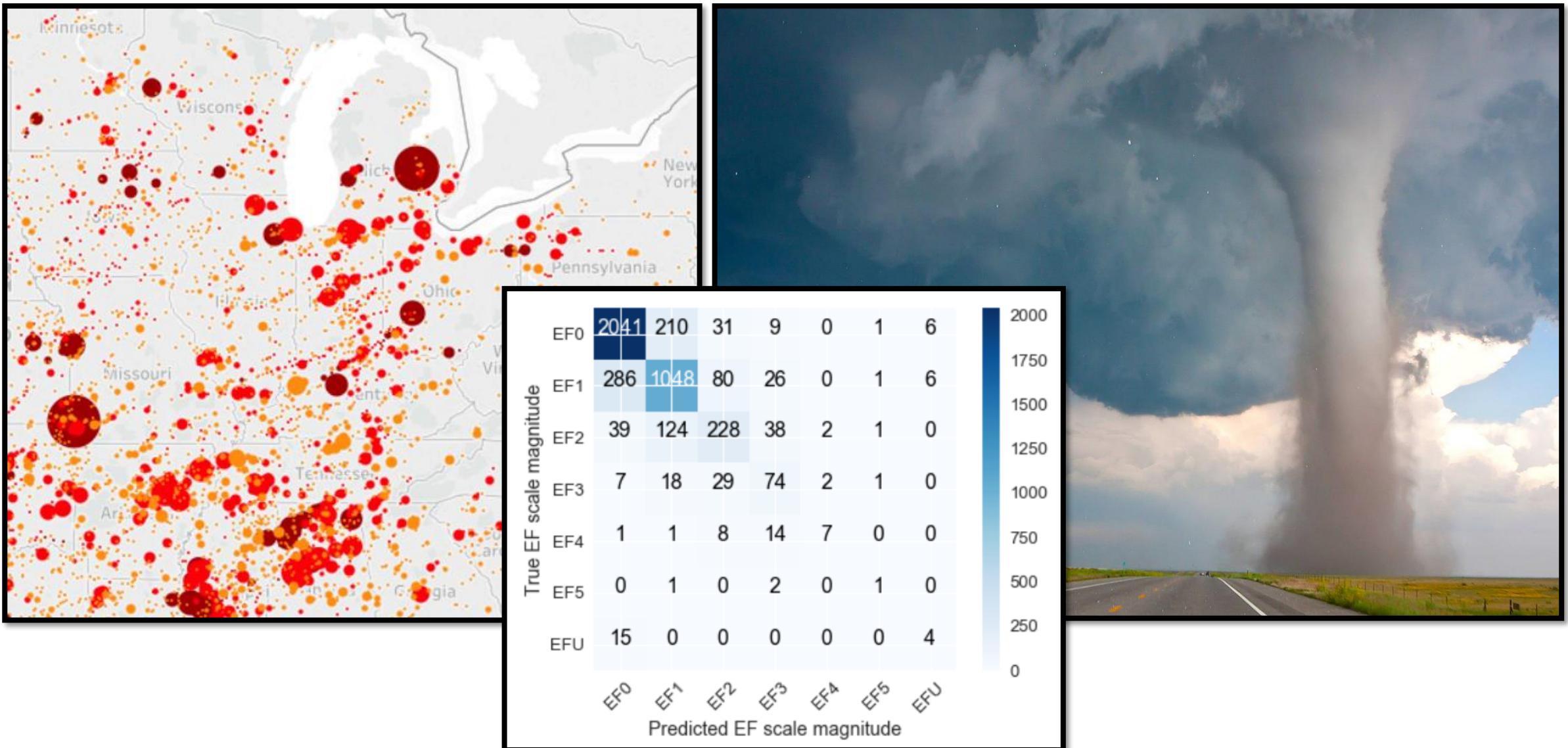
Ejemplos. Sistemas recomendadores.

¡¡Todavía mucho a resolver!!

Movies, TV shows, actors, directors, genres

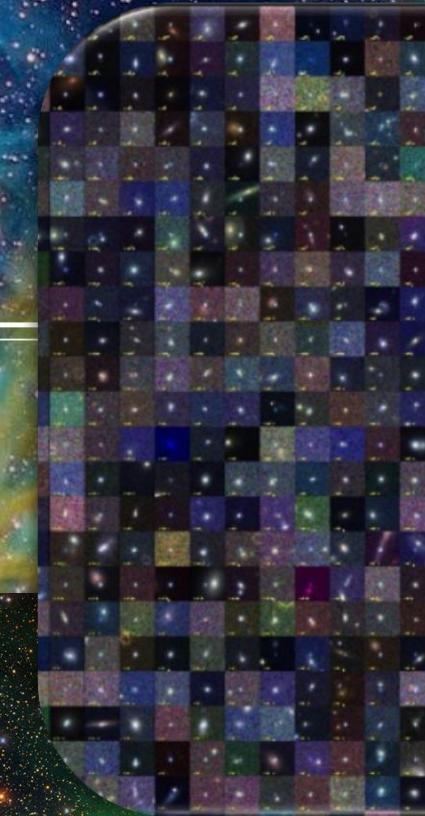
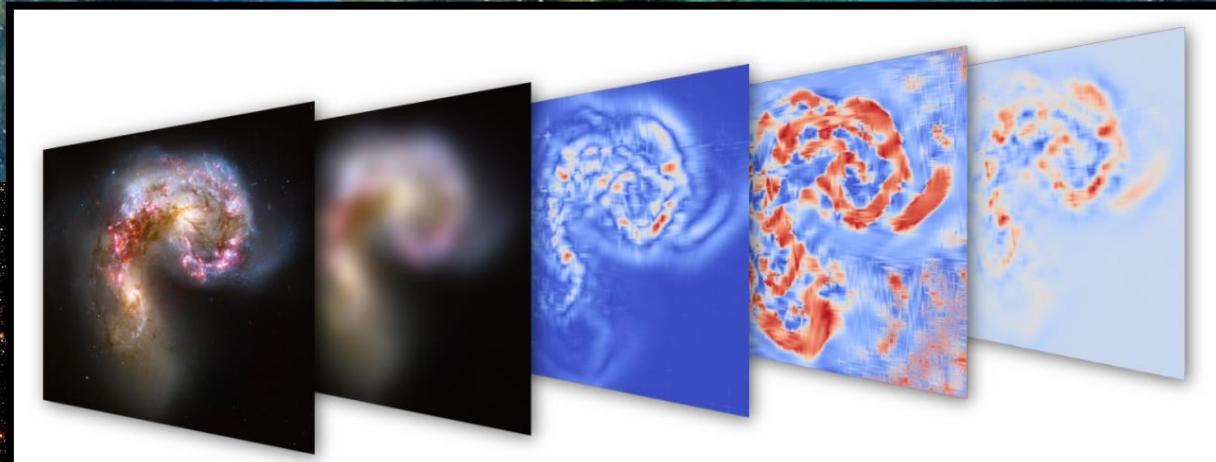


Ejemplos. Meteorología.



Ejemplos. Astronomía.

NUEVO OBSERVATORIO VIRTUAL ARGENTINO
NOVA
ARGENTINE VIRTUAL OBSERVATORY



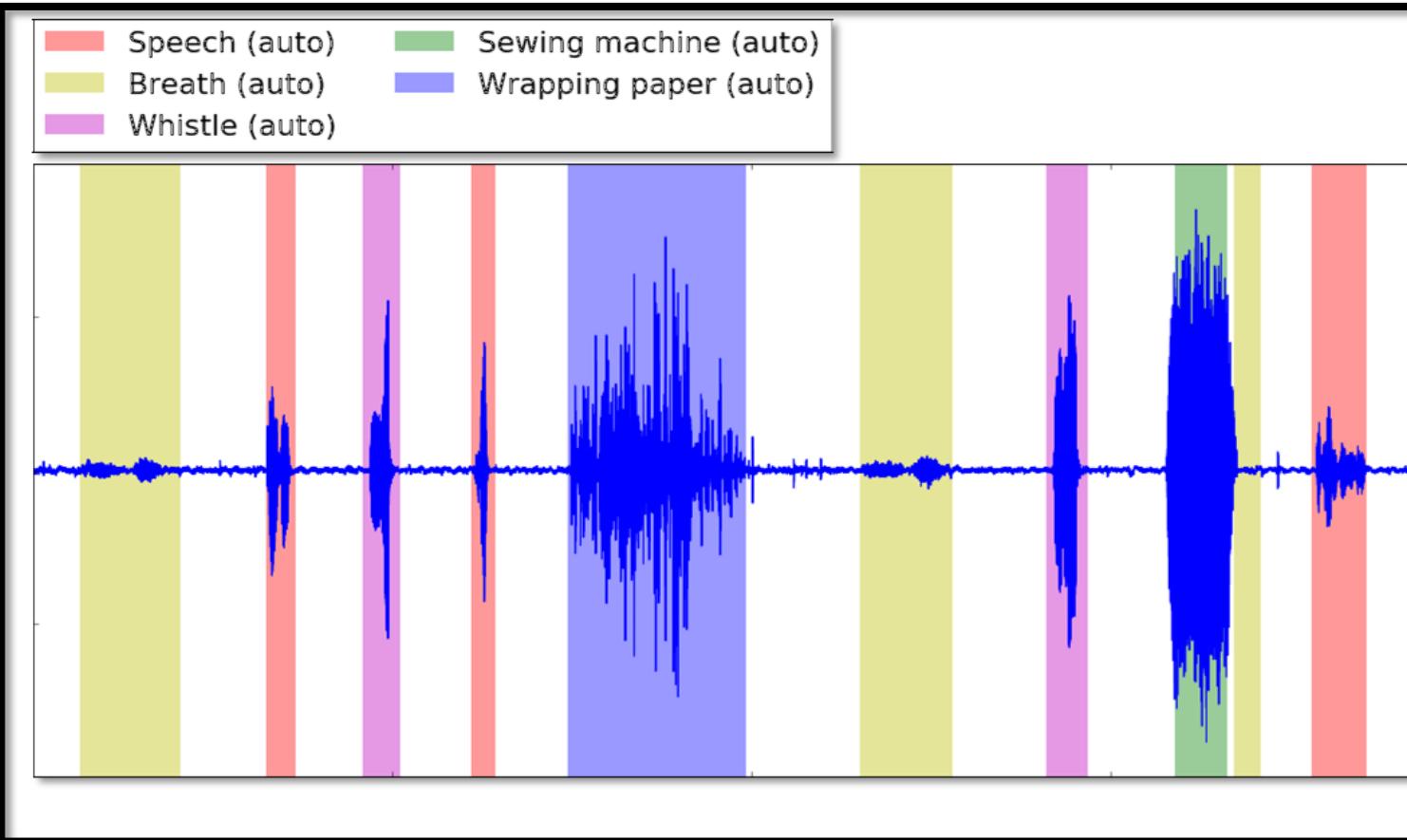
The SDSS telescopes collect over 200GB of data every night

Over 200 million galaxies have been photographed

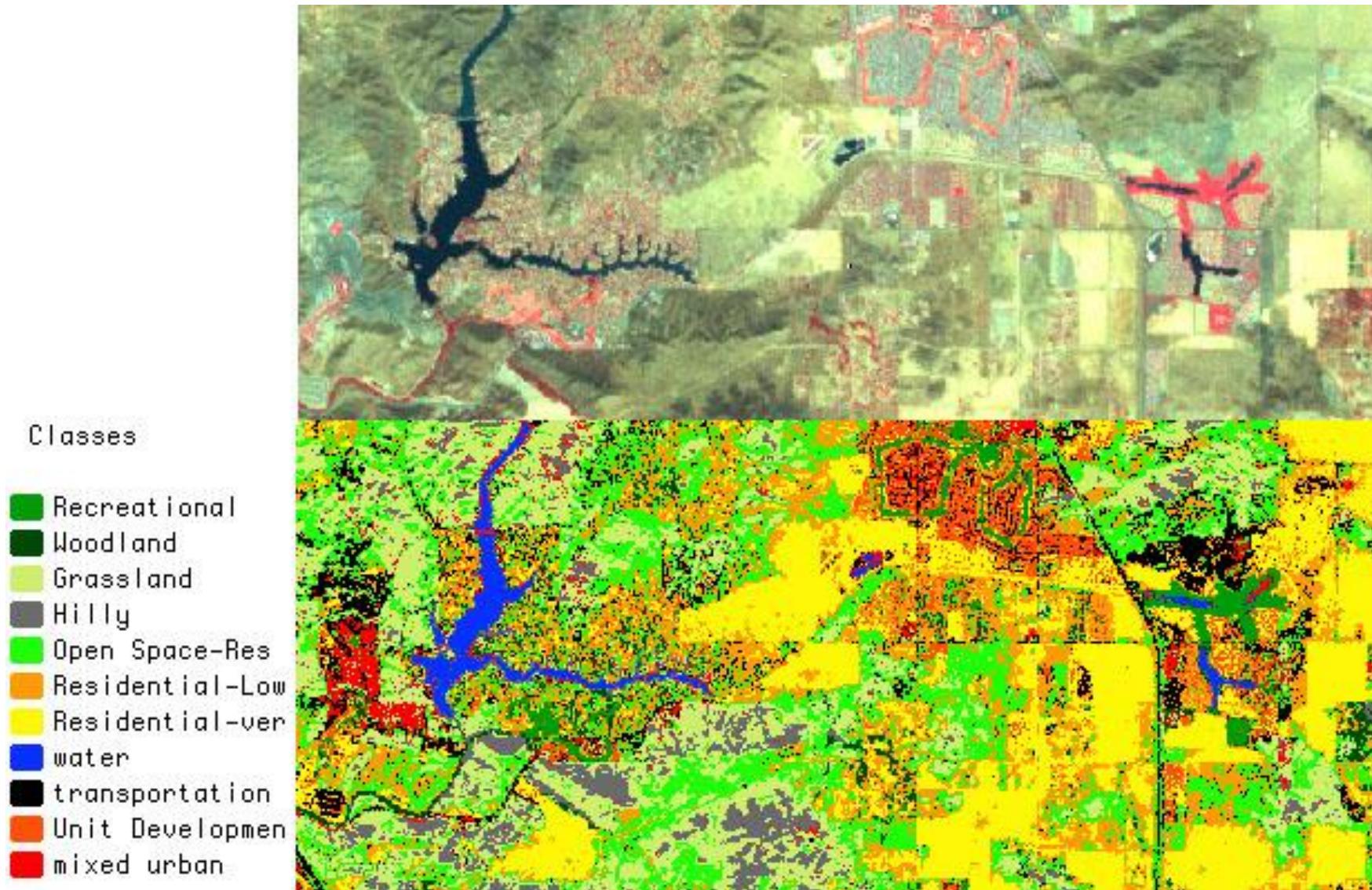
Data has become a huge bottle neck in modern astronomy

Future telescopes will collect even more data

Ejemplos. Reconocimiento de Voz.



Ejemplos. Segmentación de imágenes satelitales.

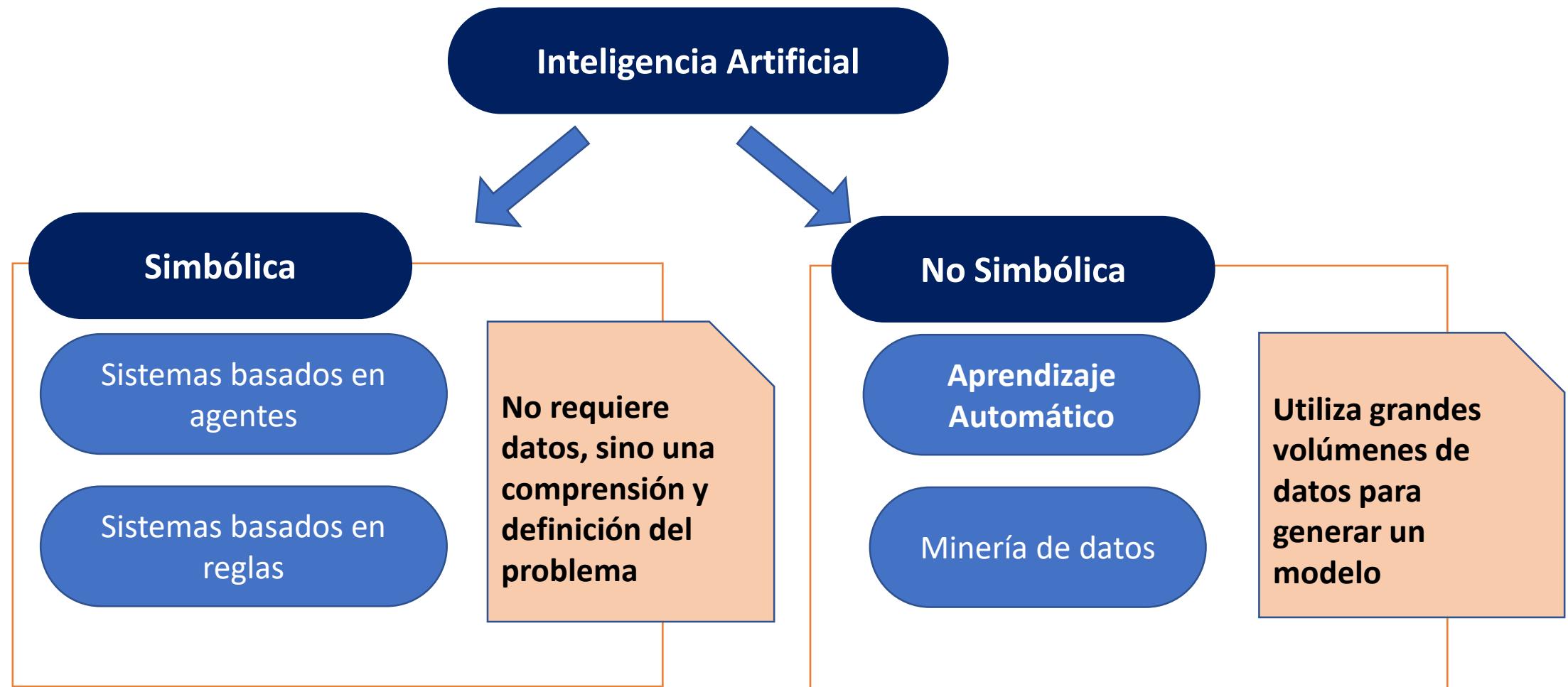


Ejemplos. Generación de arte digital.



[https://deeplearning4j.org/deeplearning4j-deeppaint](https://deeplearning4j.org/deeplearning4j-deepdream)

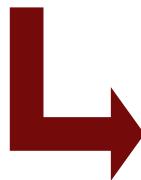
Aprendizaje Automático. Taxonomía.



Aprendizaje Automático – Tipos de algoritmos

Aprendizaje Supervisado

Se utilizan cuando existe un conjunto de datos de los cuales se conoce la clase a la que pertenecen. Generalmente se intenta minimizar cierto error de clasificación o regresión.



- Redes Neuronales
- Regresión lineal
- Máquinas de Vectores Soporte

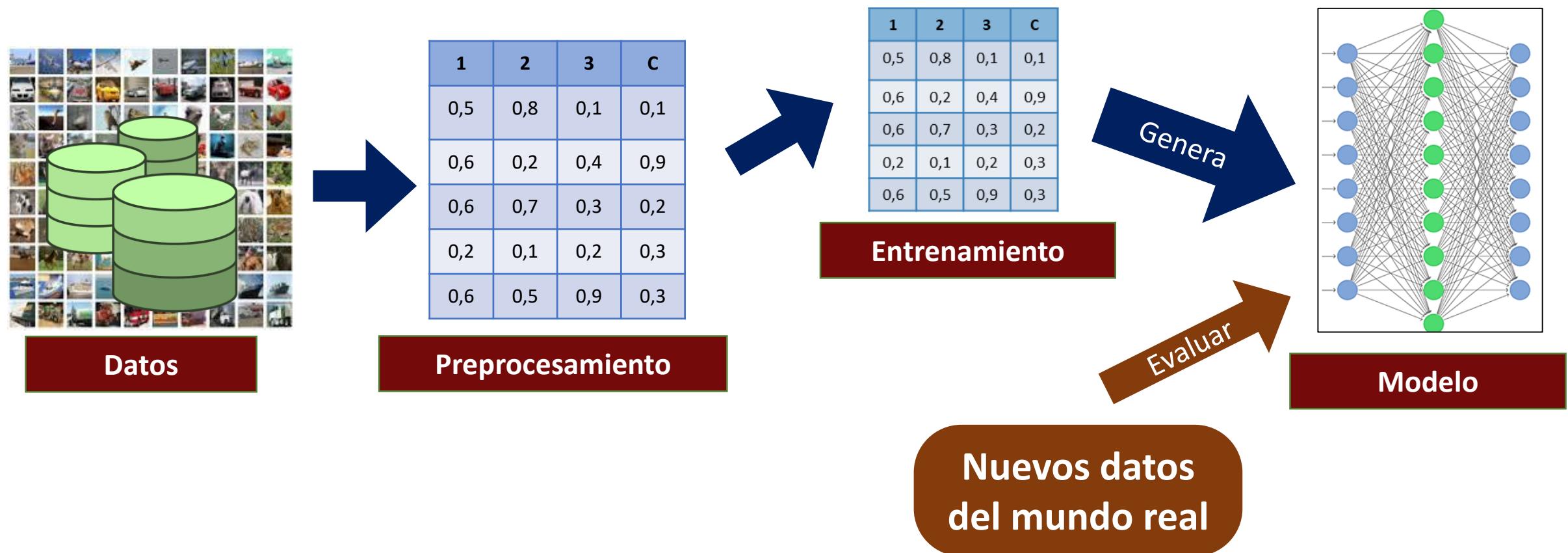
Aprendizaje No Supervisado

Se utilizan cuando existe un conjunto de datos de los cuales no se conoce su clase. El objetivo aquí es agrupar los datos que posean características similares. Descubrir nuevos patrones



- Clustering

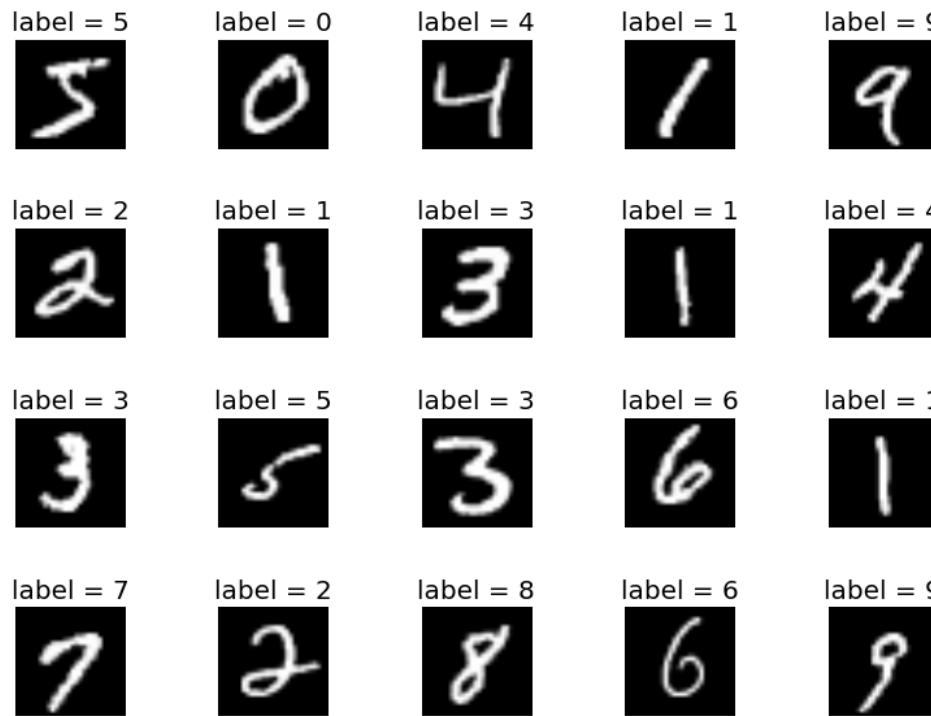
Aprendizaje Automático. Pipeline general.



Bases de datos actuales

MNIST

70 mil patrones de números escritos a mano por 250 personas

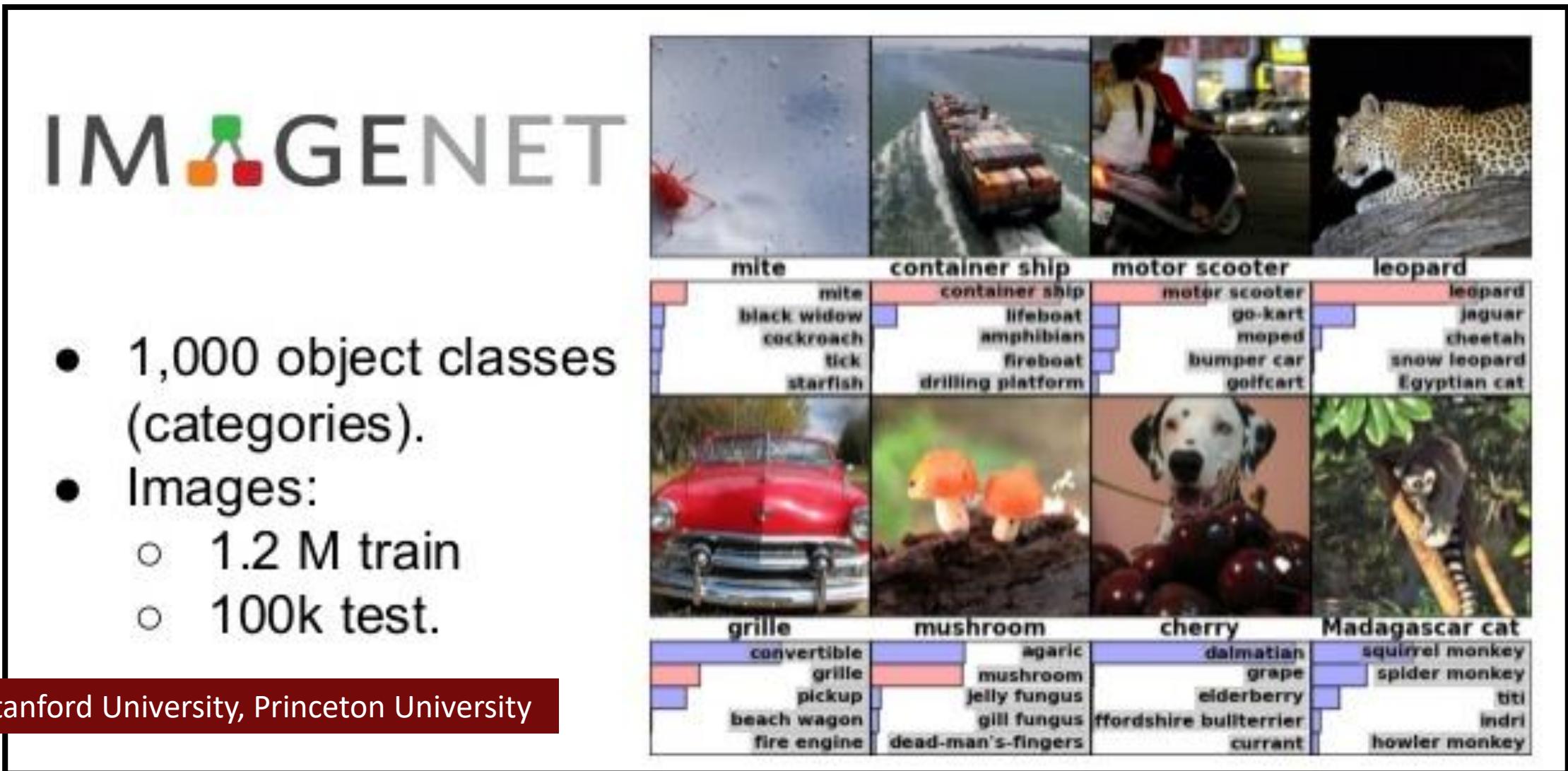


RWTH-PHOENIX-Weather

53Gb de videos con más de 300 sentencias de la lengua de señas alemana.



Bases de datos actuales



- 1,000 object classes (categories).
 - Images:
 - 1.2 M train
 - 100k test.

Stanford University, Princeton University

Bases de datos actuales

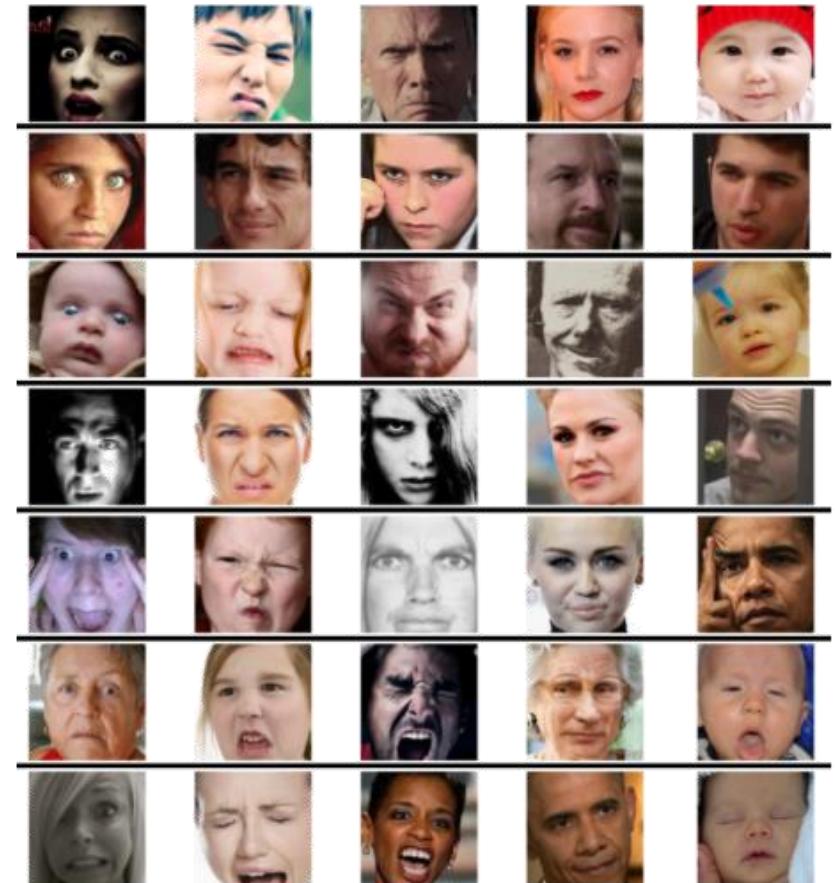
Celeba

200K imágenes de celebridades con anotaciones de diferentes atributos



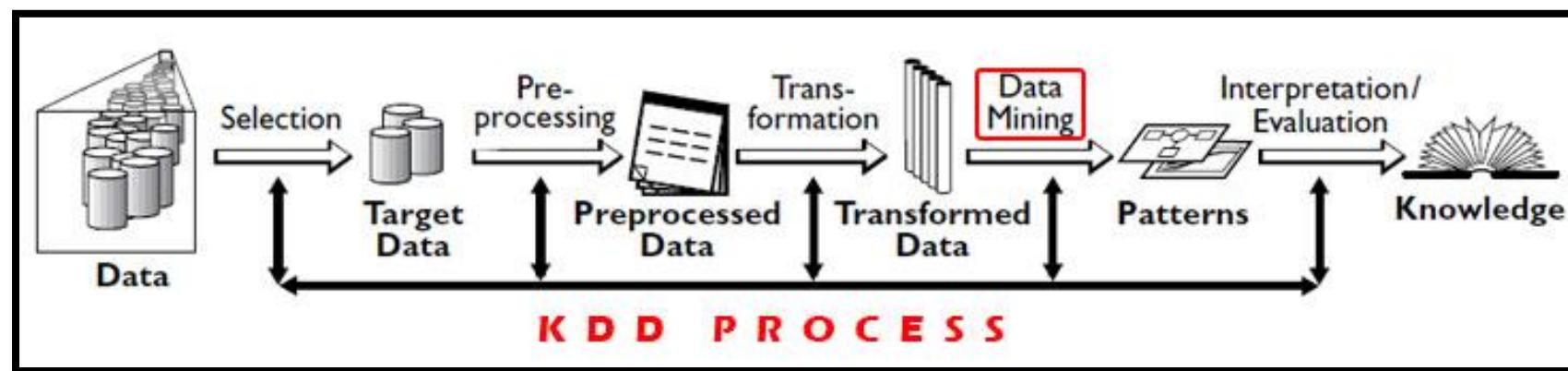
Affect Net

1 millón de imágenes de emociones humanas obtenidas desde internet



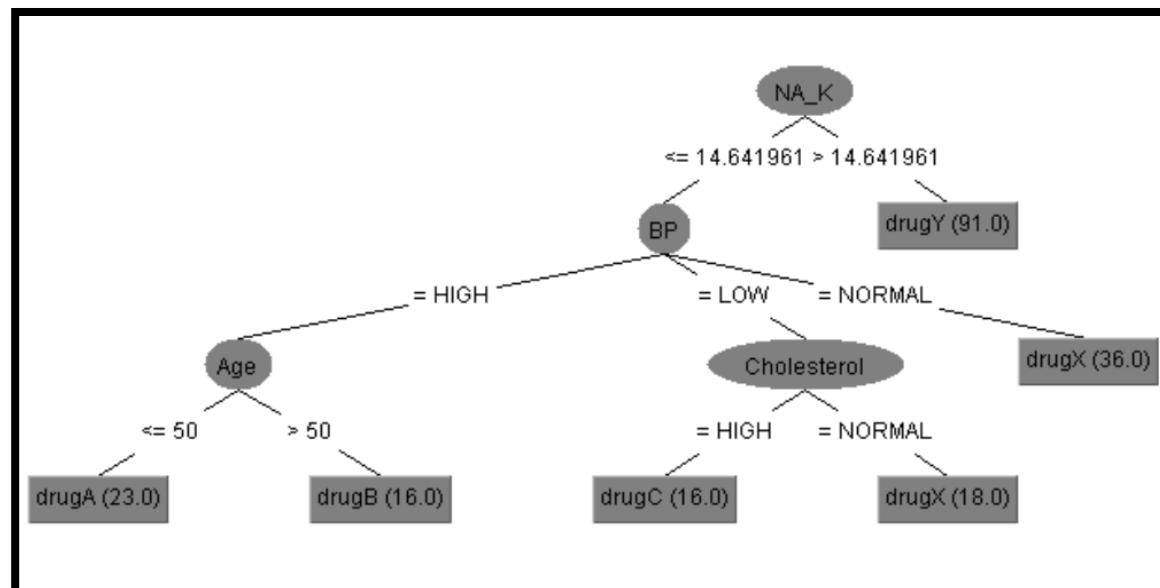
Minería de datos vs Aprendizaje Automático

- La Minería de Datos busca el descubrimiento del conocimiento sin una hipótesis preconcebida.
- “KDD” (Knowledge Discovery in Databases) es el proceso no trivial de identificar patrones a partir de los datos, que sean novedosos y potencialmente útiles.
- La minería de Datos va más allá de clasificar patrones como “bien o “mal”, que es algo que intentamos hacer con Aprendizaje Automático.
- No obstante las técnicas utilizadas son muy similares.



Minería de datos vs Aprendizaje Automático

- Modelos Descriptivos: Muestran nuevas relaciones entre las variables.
- Modelos Predictivos: En base al modelo que gobierna el sistema es posible predecir hechos futuros.



Algunas técnicas que no usaremos en este curso.

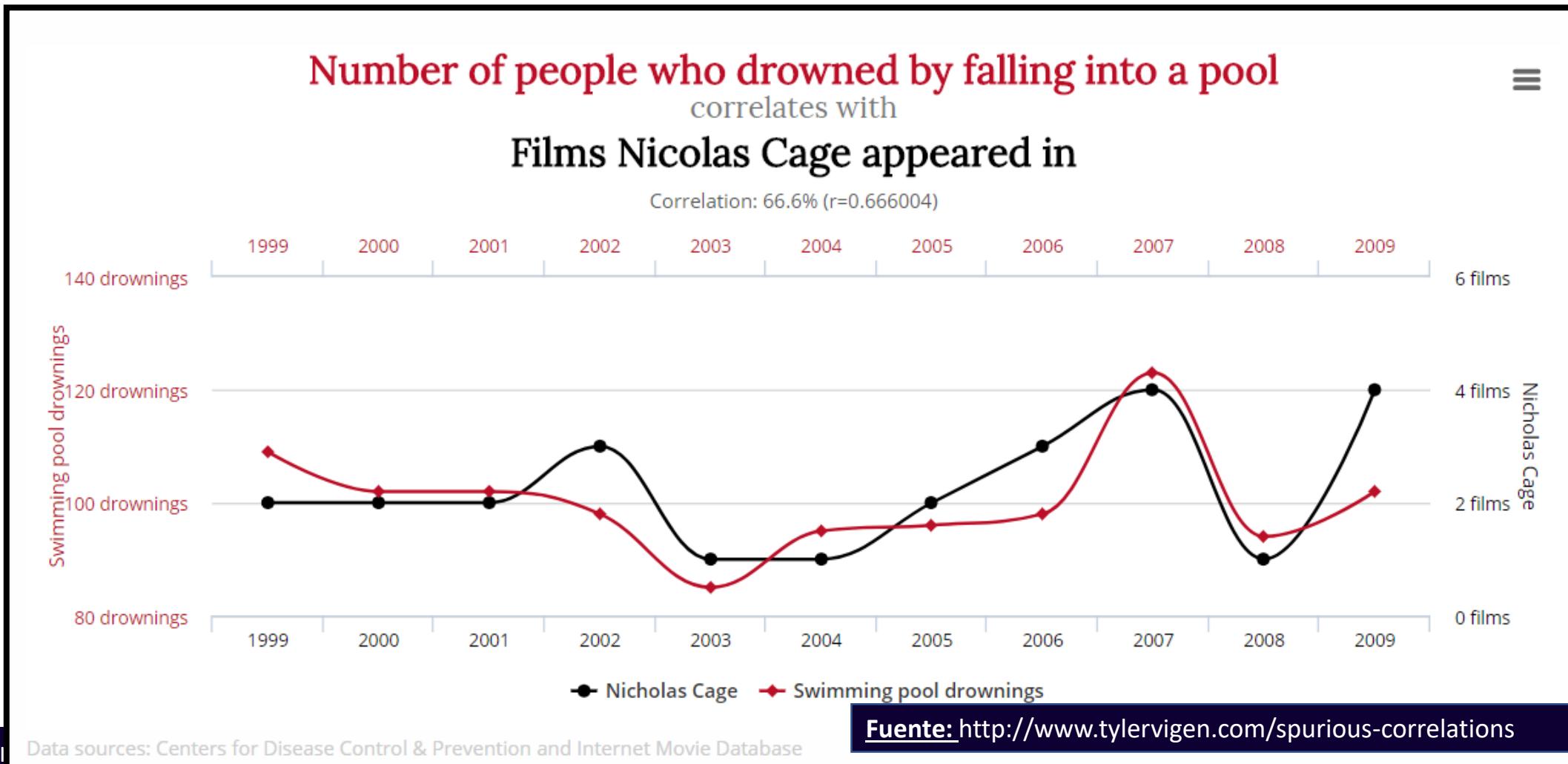
IDC	D-créditos (años)	C-créditos (pesos)	Salario (pesos)	Casa propia	Cuentas Morosas	...	Devuelve crédito
101	15	60000	2200	Si	2	...	no
102	2	30000	3500	Si	0	...	Si
103	9	9000	1700	Si	1	...	No
104	15	18000	1900	No	0	...	Si
105	10	24000	2100	no	0	...	No
...

Reglas obtenidas

- Si cuentas-Morosas > 0 entonces Devuelve-credito = no
- Si Cuentas-Morosas=0 Y $[(\text{Salario}>2500) \text{ O } (\text{D-credito}>10)]$ entonces Devuelve-credito= si

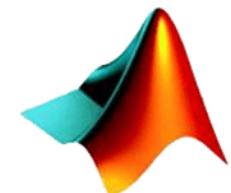
Aprendizaje Automático y Big Data

El aumento masivo de los datos en algunos casos puede ayudar, pero también puede traer correlaciones aleatorias. Siempre es importante utilizar el conocimiento de dominio.



Aprendizaje Automático. Lenguajes/librerías

Hace algunos años...



MATLAB®

Orientado a Minería de Datos



Hoy



python™



julia



Deep Learning

K Keras PYTORCH

theano



TensorFlow

ETC...

Machine Learning - Modelos

Modelos de caja blanca



Modelos de caja negra



VS.

Es posible interpretar el modelo:

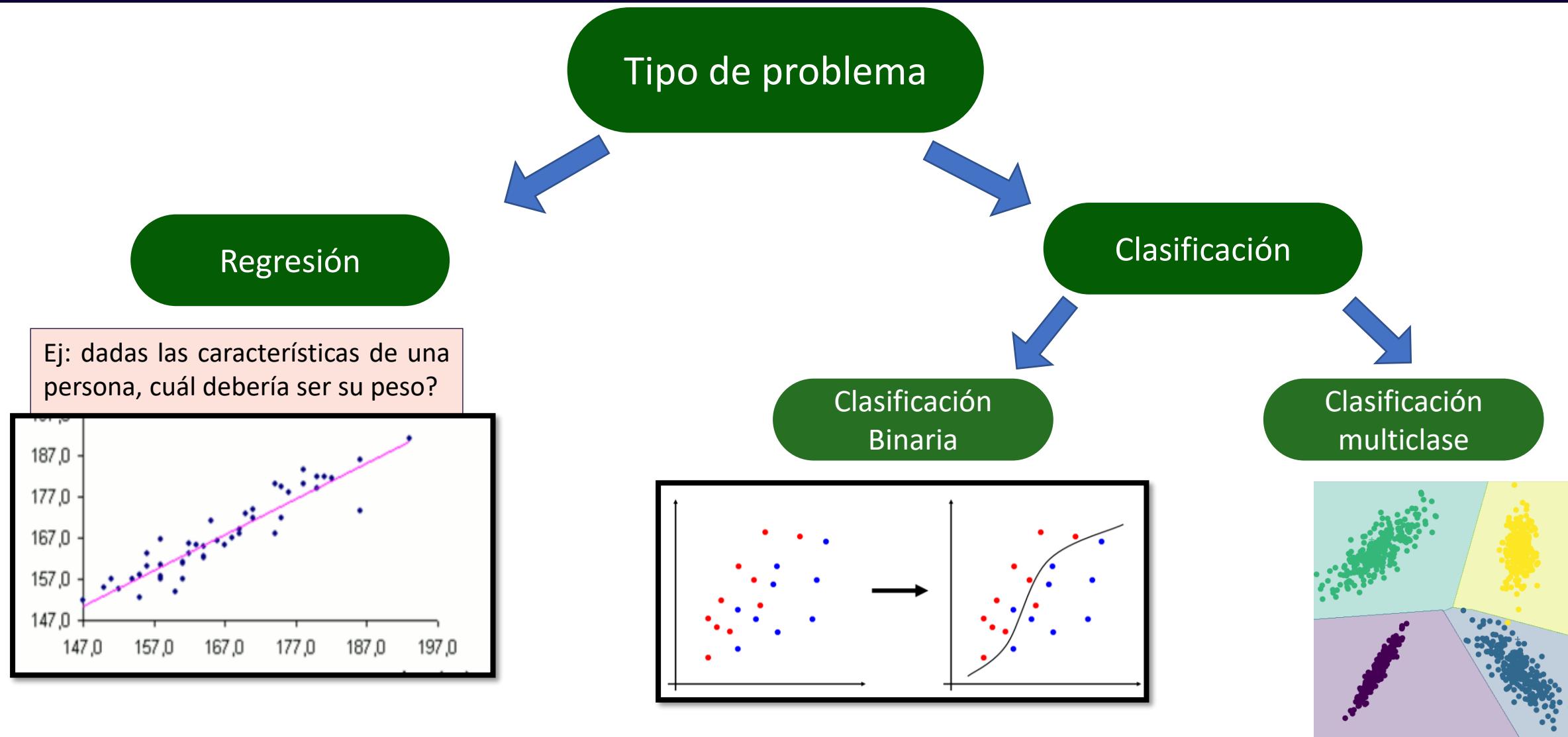
- Regresión lineal
- Modelos probabilísticos

NO es posible interpretar el modelo:

- Redes Neuronales
- En cierta medida: Deep Learning

TIPOS DE PROBLEMAS

Aprendizaje Automático. Tipos de problemas.



Problemas de Regresión

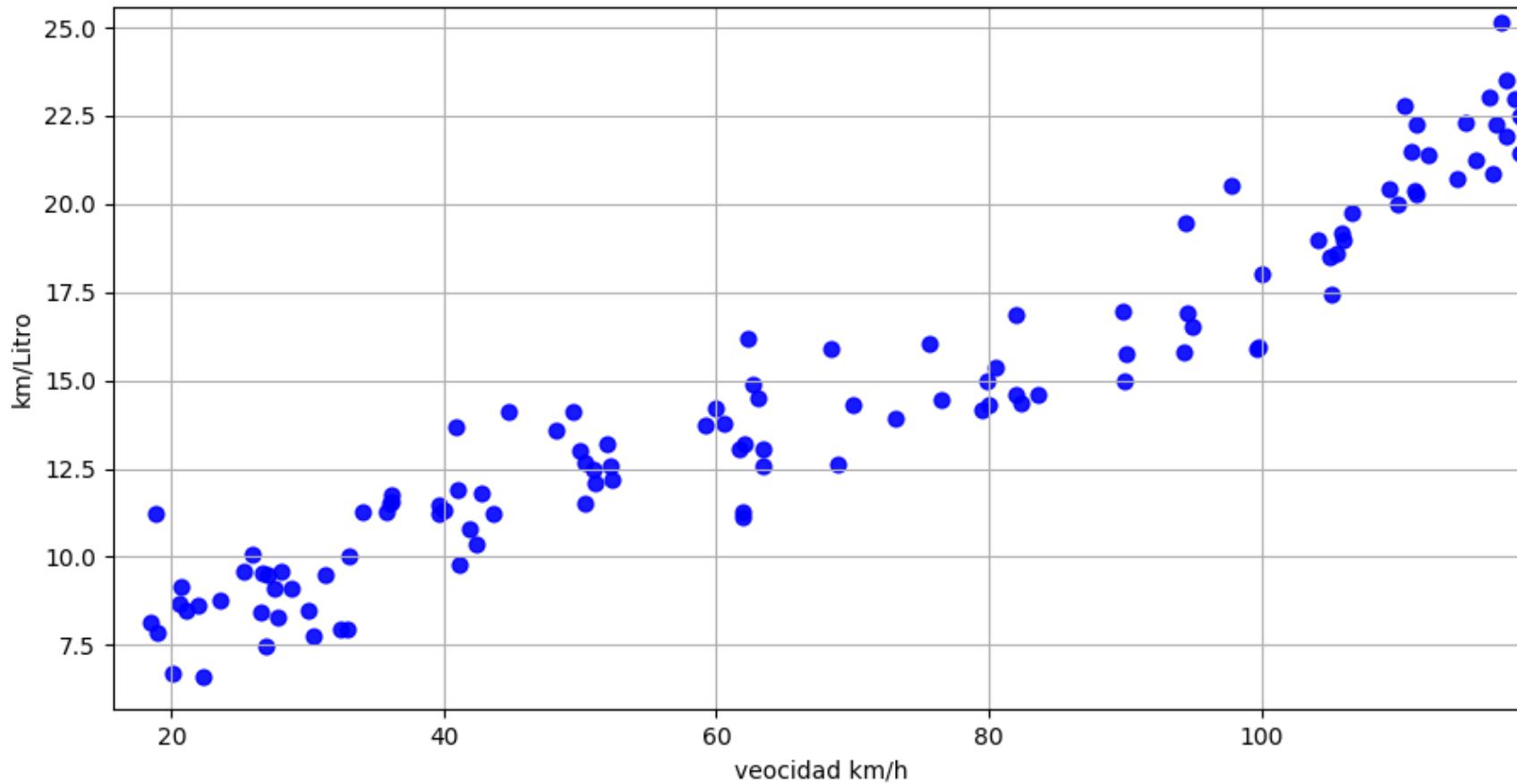
Supongamos el siguiente problema:

Queremos predecir cuál es el rendimiento (km/L) que tiene un auto en base a la velocidad a la que circula

velocidad km/h	km/Litro
20	8.00
75	15.20
22	8.60
10	18.00
51	12.50
52	13.20
60	14.20

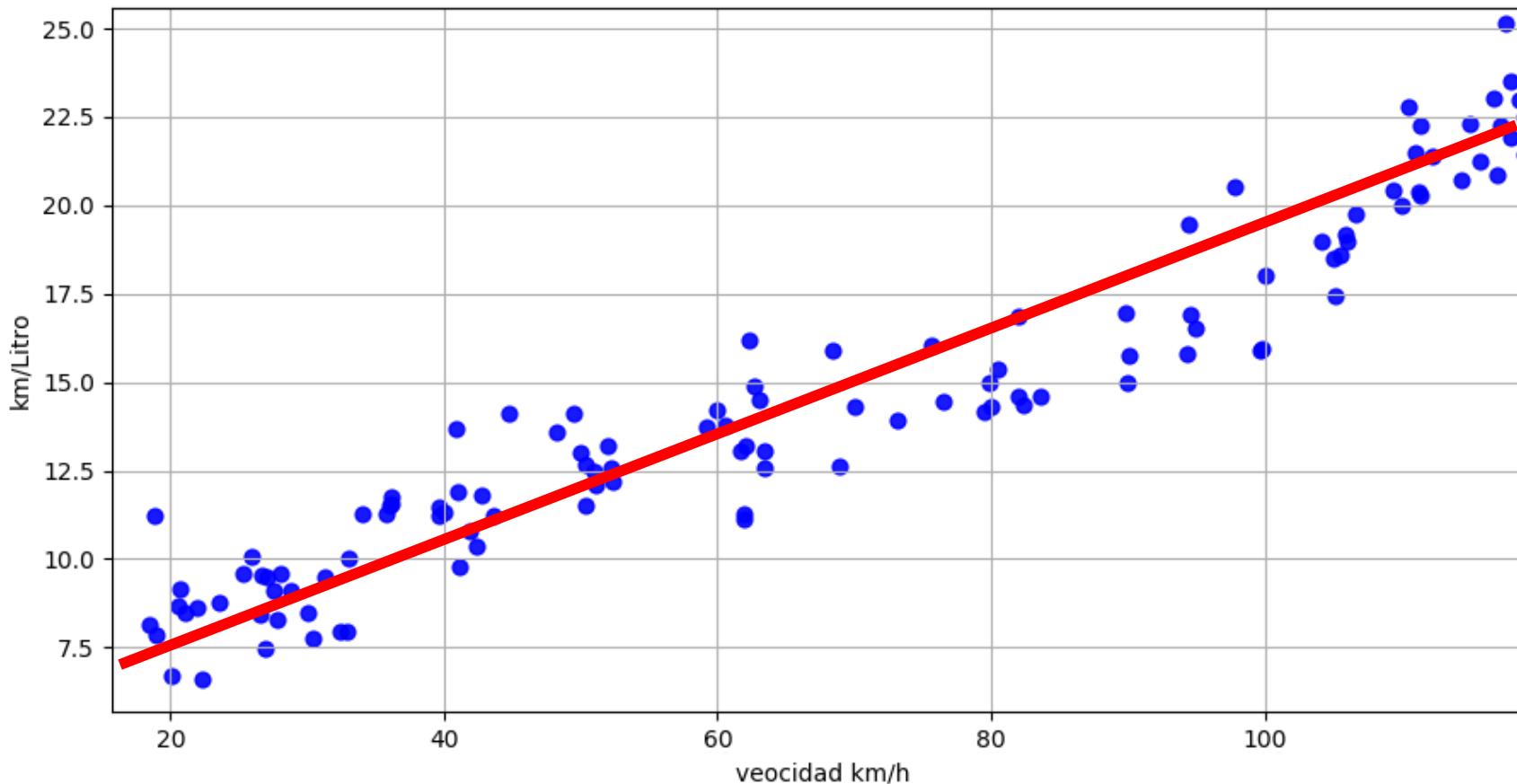
Problemas de Regresión

¿Cómo podemos generar un modelo que resuelva este problema?



Problemas de Regresión

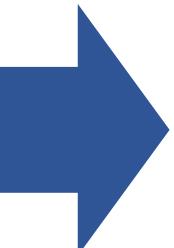
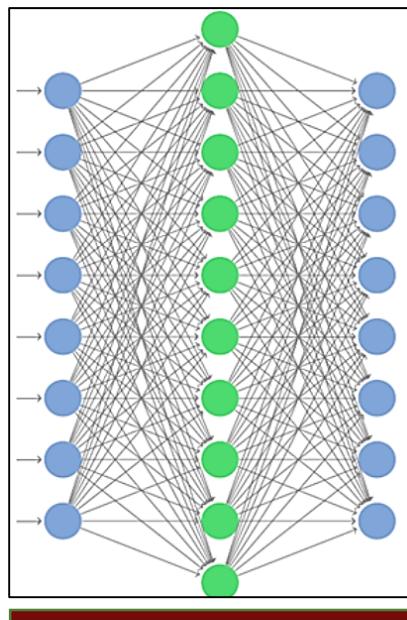
¿Cómo podemos generar un modelo que resuelva este problema?



Podríamos generar una ecuación lineal.
El modelo va a tener un error (ya que muy pocos elementos pasan por la recta), pero es simple y funciona “bien”.

Problemas de Clasificación

Nuestro Objeto



Gato

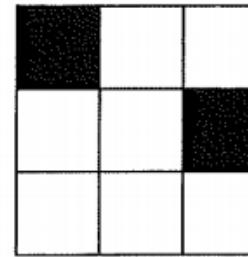
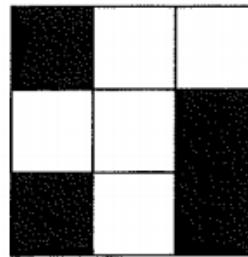
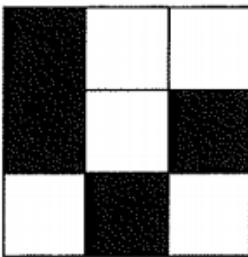
Perro

Ave

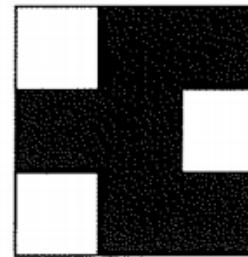
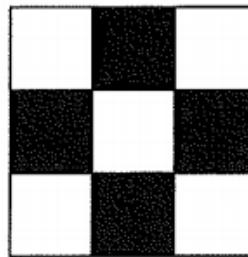
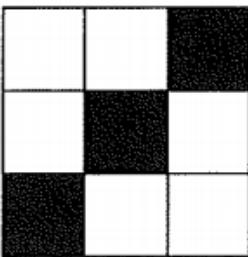
Rana

Ciervo

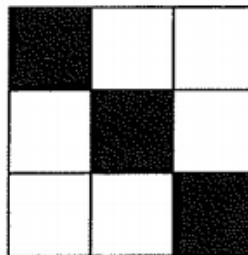
Problemas de Clasificación



$$f = -1$$



$$f = +1$$



$$f = ?$$

En el Aprendizaje Automático, el modelo aprenderá en base a los datos. Si los datos no son representativos, el modelo no generalizará bien a nueva información.

Problemas de Clasificación

Por ejemplo, ¿cómo agruparía las bolas del juego de Pool en diferentes conjuntos?

¿Qué característica necesito?

¿Cuántos conjuntos necesito?

Podrían agruparse por color (rojas, amarillentas, azuladas).

En orden por sus valores.

Por su función en el juego (lisas, rayadas).



"you need to know the question you are trying to answer"

- Jason Bell 2015

Problemas de Clasificación

Supongamos el siguiente problema:

Queremos clasificar dos especies de flores según el largo y ancho de sus pétalos.
Tenemos la siguiente información. X1= Largo, X2= Ancho.

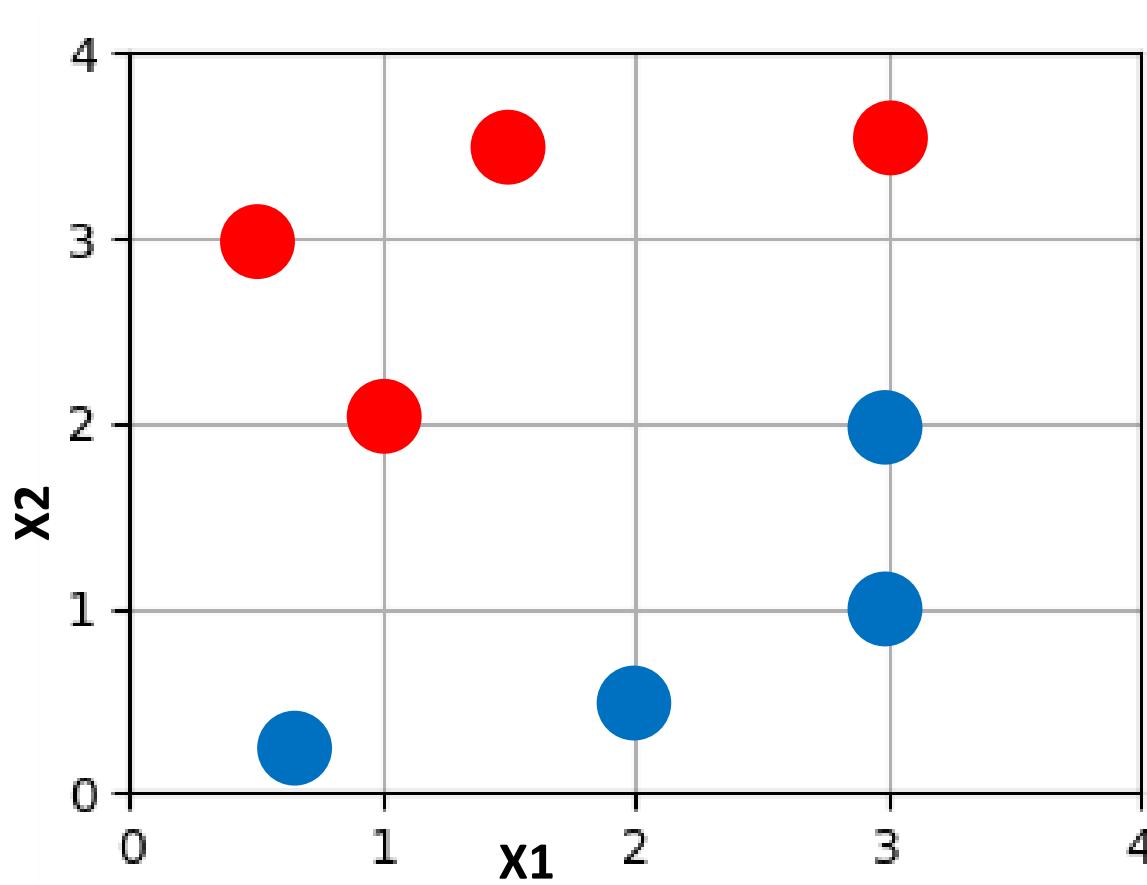
x1	x2	Clase
0,7	0,2	0
2	0,5	0
3	1	0
3	2	0
0,5	3	1
1	2	1
1,5	3,5	1
3	3,5	1



A veces, la “clase” estará en formato nominal (texto).
Ej. “flor_tipo_1” y “flor_tipo_2”

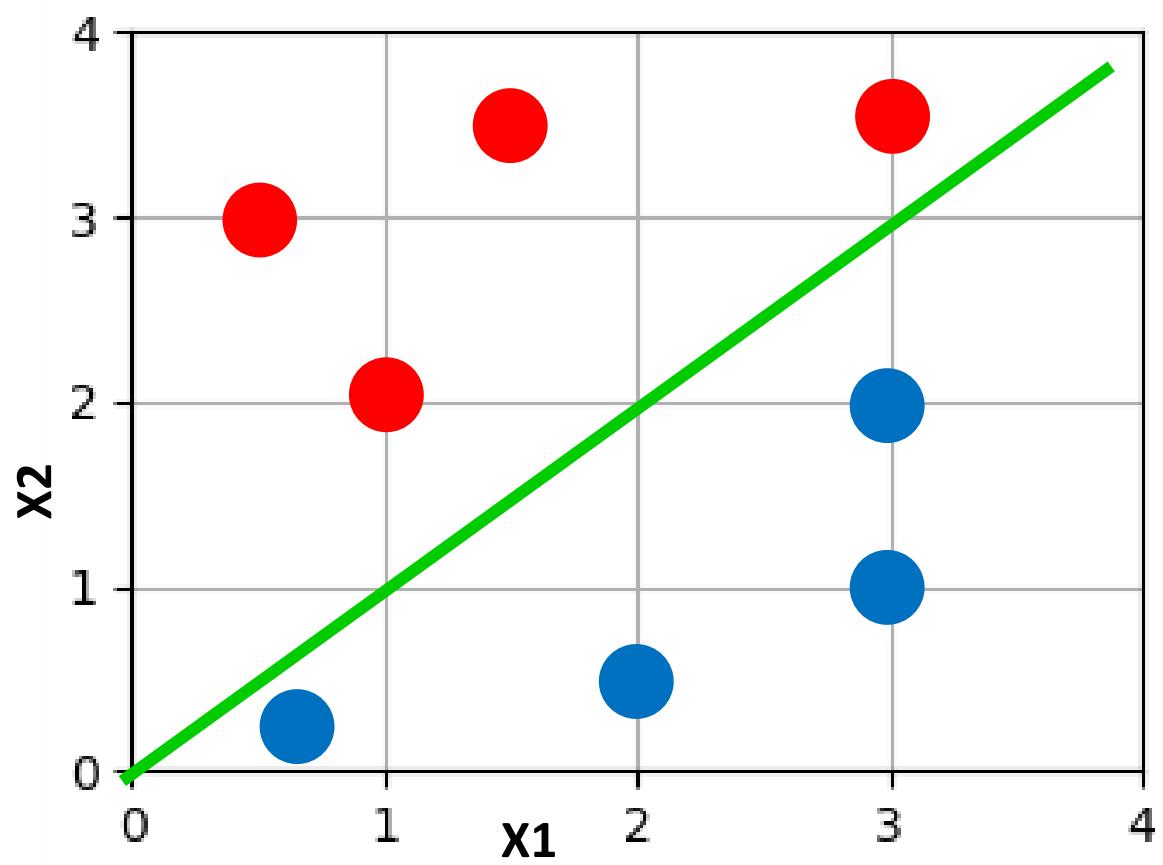
Problemas de Clasificación

Visualización en 2D



x_1	x_2	Clase
0,7	0,2	0
2	0,5	0
3	1	0
3	2	0
0,5	3	1
1	2	1
1,5	3,5	1
3	3,5	1

Problemas de Clasificación

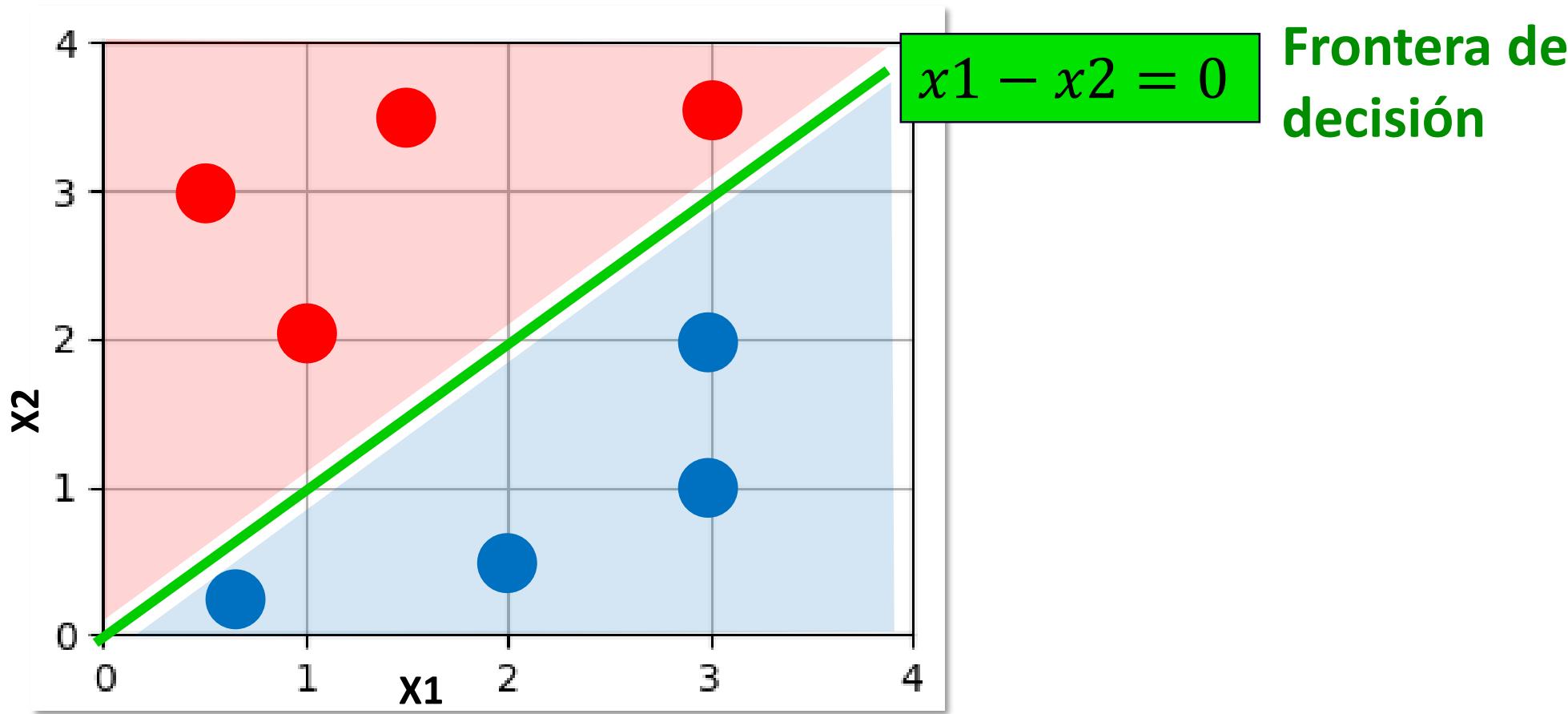


El modelo más simple que podemos pensar: **Lineal**

Ecuación de decisión:
 $x_1 = x_2$

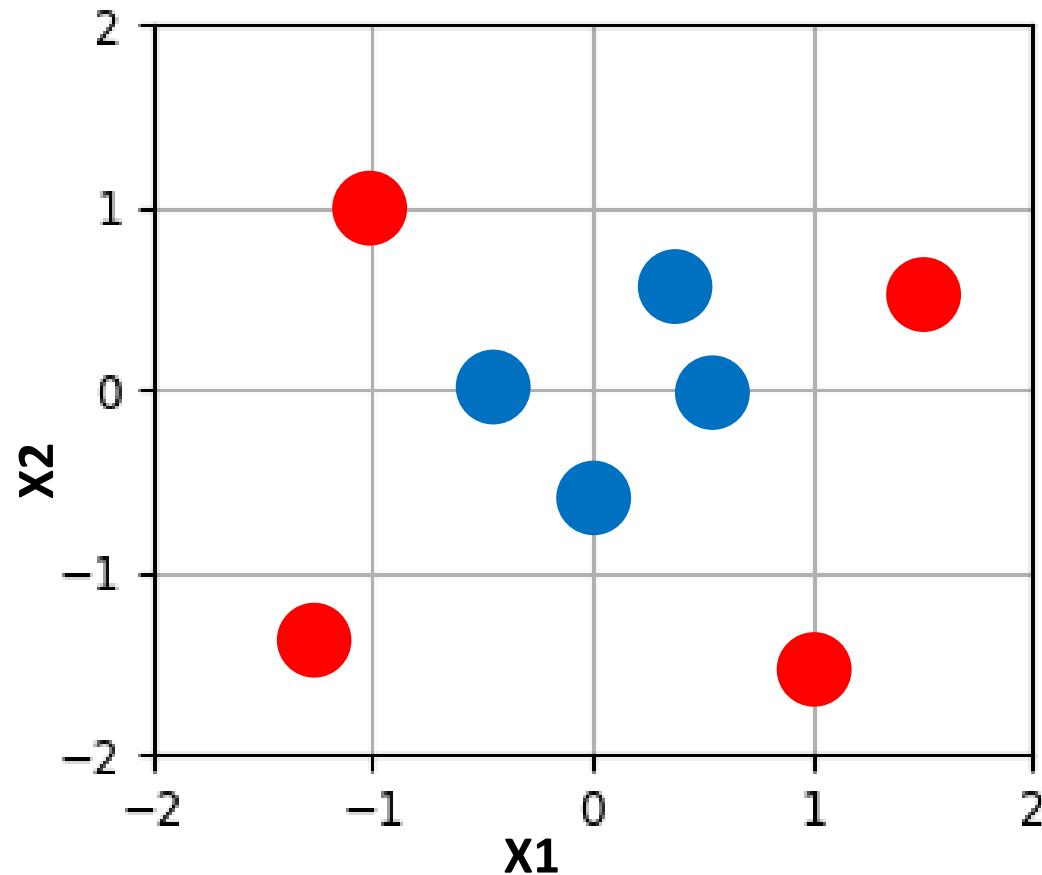
$$g(x) = \begin{cases} 0 & \text{si } x_1 - x_2 > 0 \\ 1 & \text{si } x_1 - x_2 \leq 0 \end{cases}$$

Problemas de Clasificación



Problemas de Clasificación

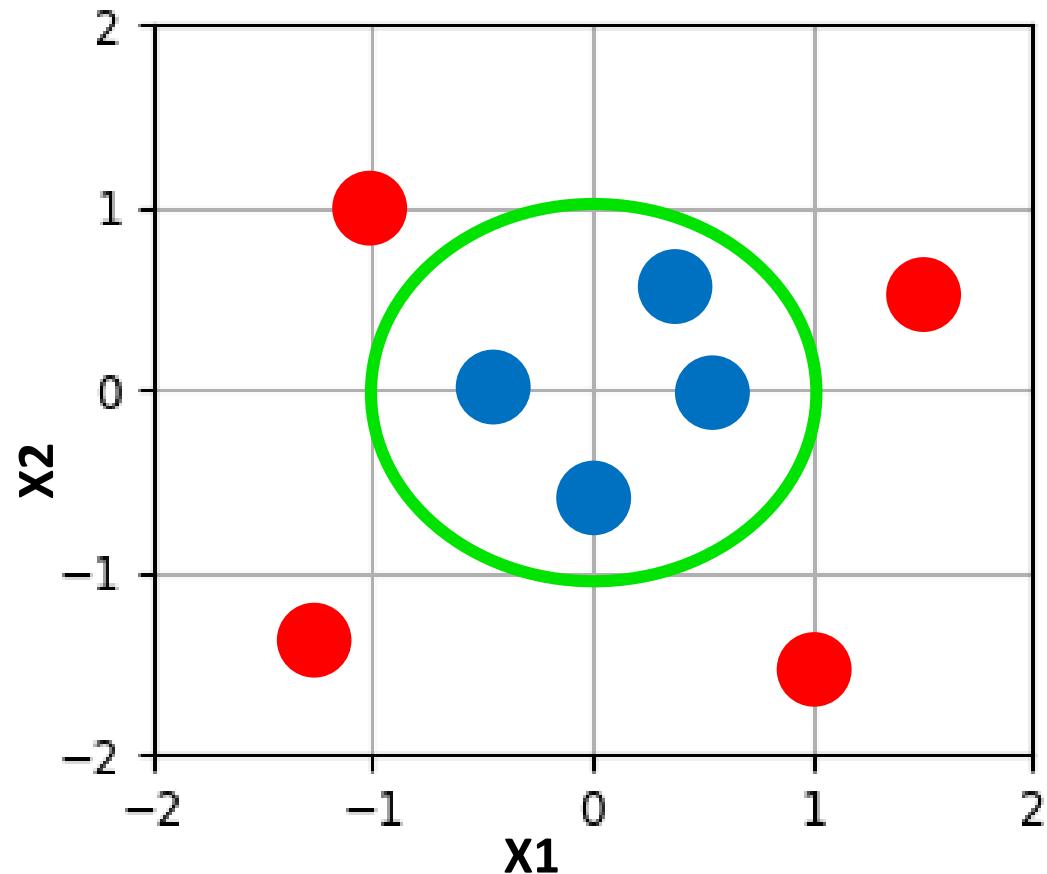
¿Y ahora?



x_1	x_2	Clase
0	-0,5	0
0,5	0	0
-0,5	0	0
0,3	0,5	0
-1,3	-1,3	1
1	-1,5	1
-1	1	1
1,5	0,5	1

Problemas de Clasificación

Un modelo lineal ya no nos sirve.

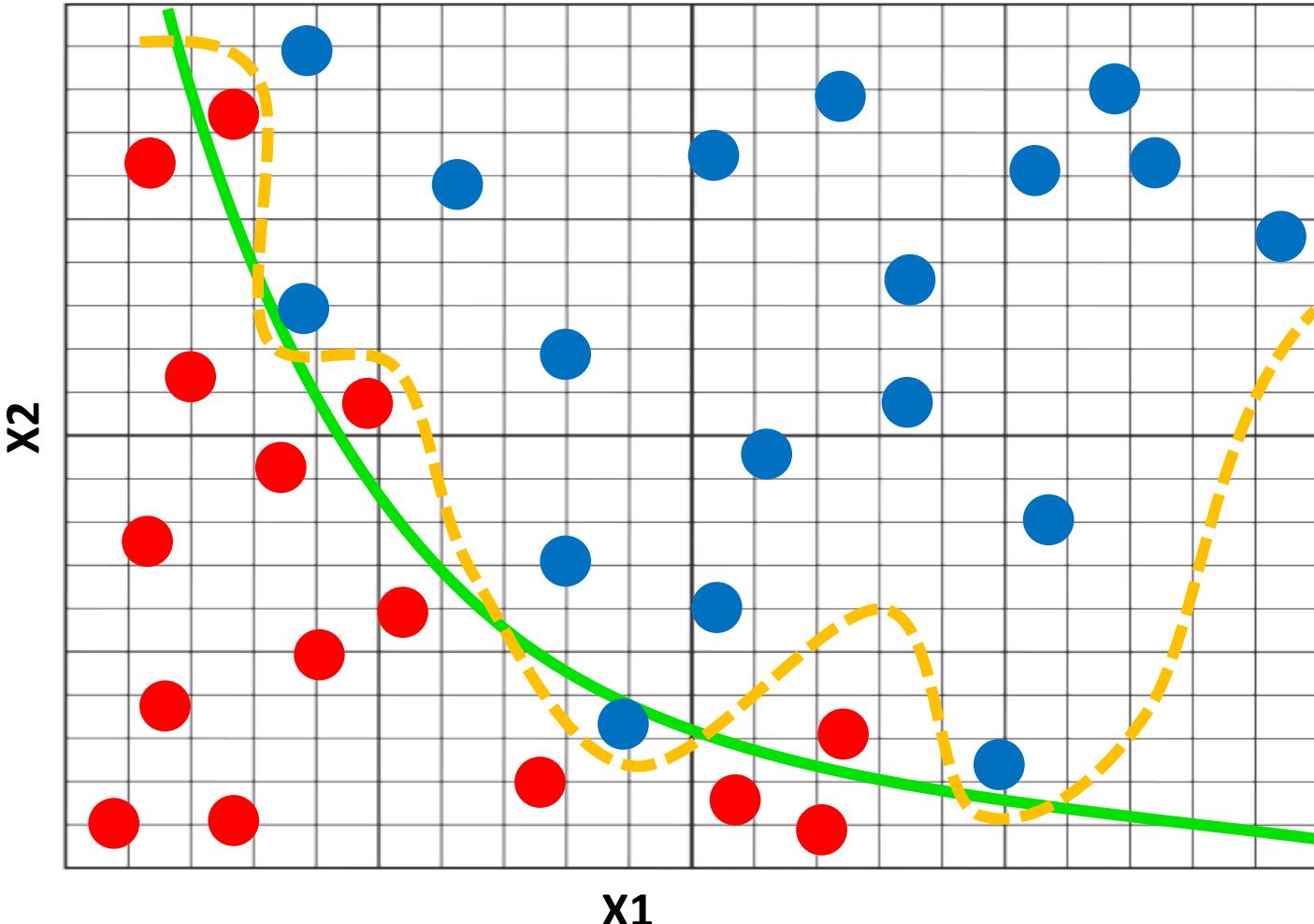


$$f = x_1^2 = x_2^2$$

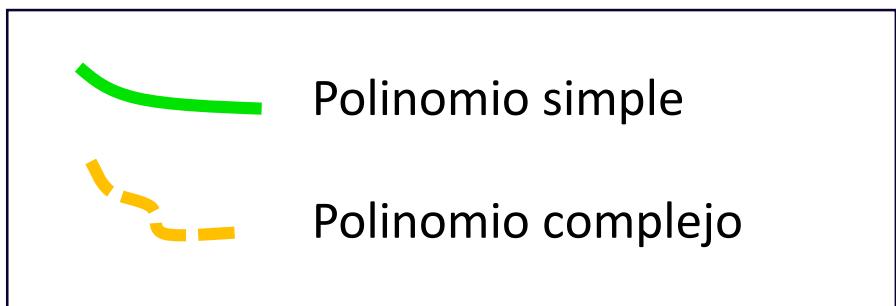
$$g(x) = \begin{cases} 1 & \text{si } x_1^2 - x_2^2 > 0 \\ 0 & \text{si } x_1^2 - x_2^2 \leq 0 \end{cases}$$

Generalización de un modelo

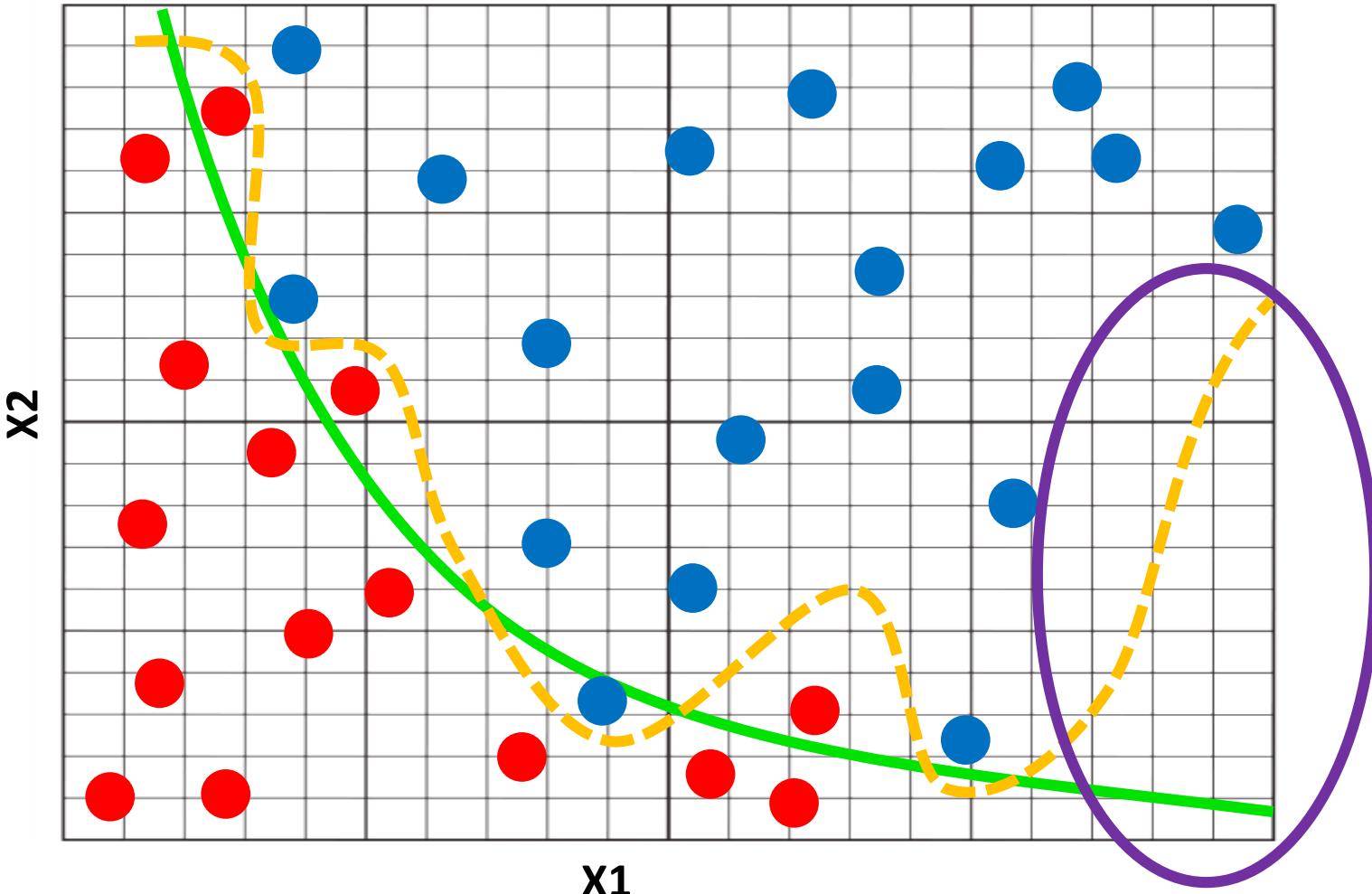
¿Qué curva es mejor?



Generalmente, un modelo más simple lo vamos a considerar mejor, ya que generaliza de forma más efectiva, aunque no sea perfecto.

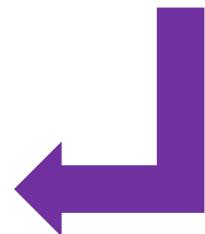


Generalización de un modelo

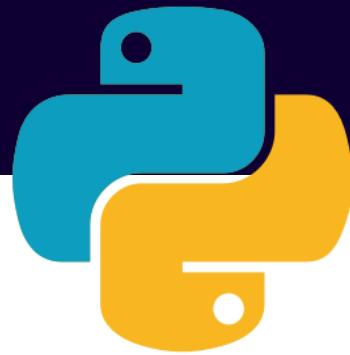


¿Qué ocurre en esta área?

Si no hay datos, esa zona no importa para un modelo de Aprendizaje Automático. Pero podría ser una zona del espacio a considerar en futuros datos (desconocidos por el modelo).



PYTHON



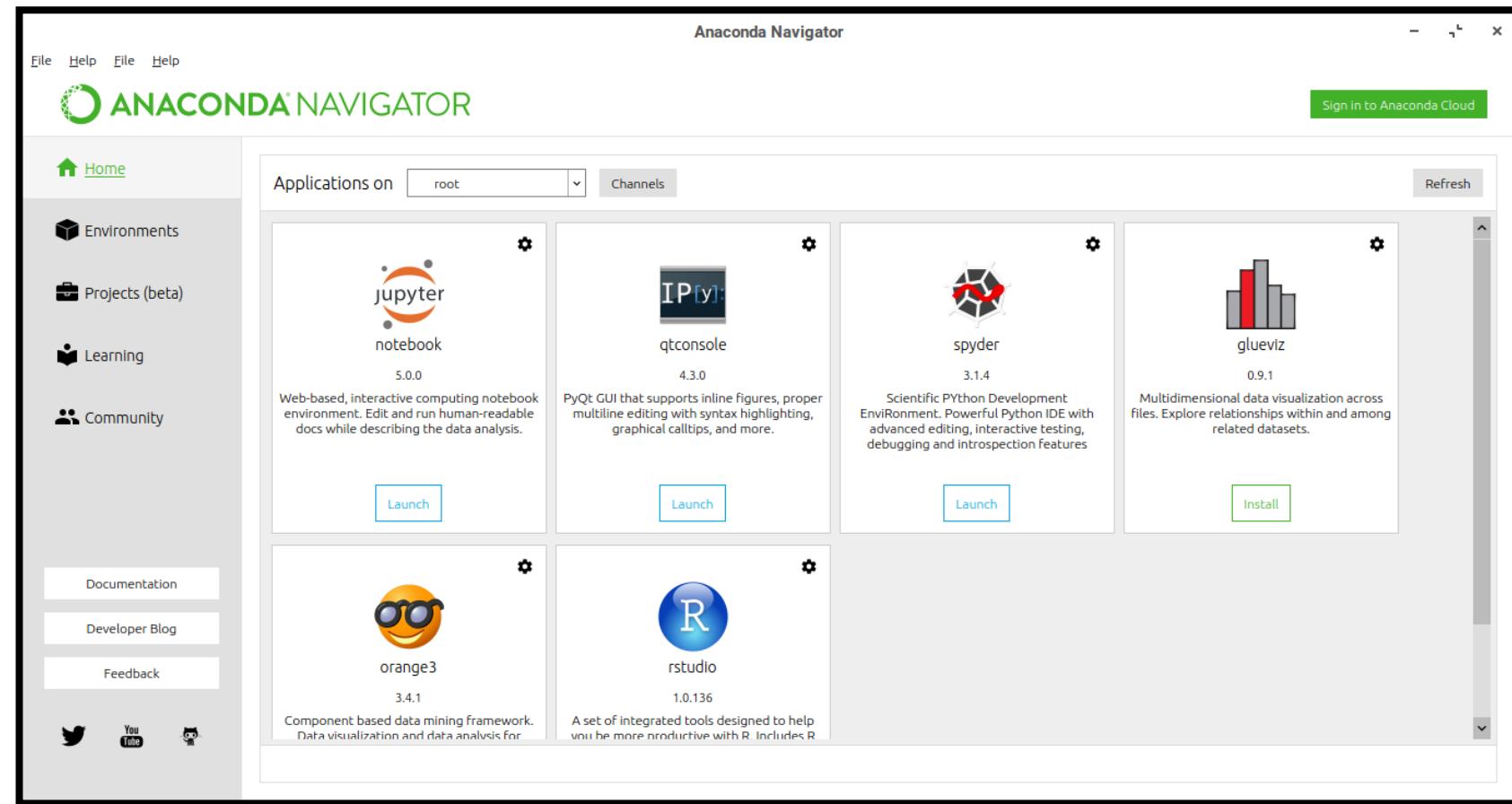
Python

<http://www.python.org.ar/>
<https://www.python.org/>

- Python es un lenguaje de programación de alto nivel.
- Es interpretado y tiene una filosofía que hace hincapié en una sintaxis que favorezca un código legible.
- Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

```
(base) C:\Users\Franco>python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Anaconda es una distribución de Python libre y abierta.
- Orientado a la ciencia de datos.
- Tiene más de 250 paquetes (librerías) multiplataforma (Windows, Linux, Mac).
- Posee aplicaciones de alto nivel tanto para desarrollo como para gestión de paquetes, visualización, etc.



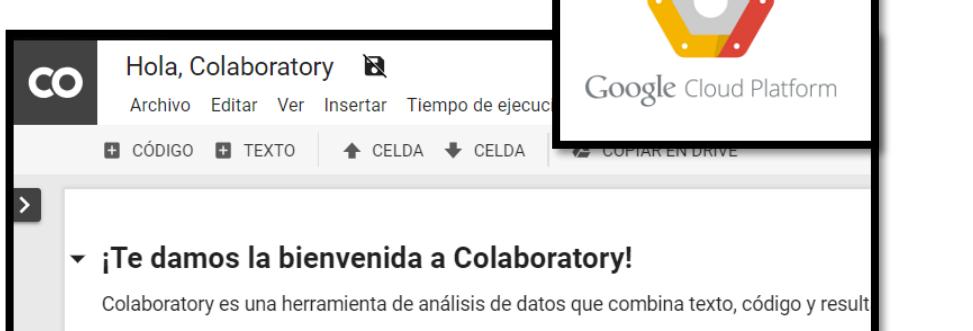
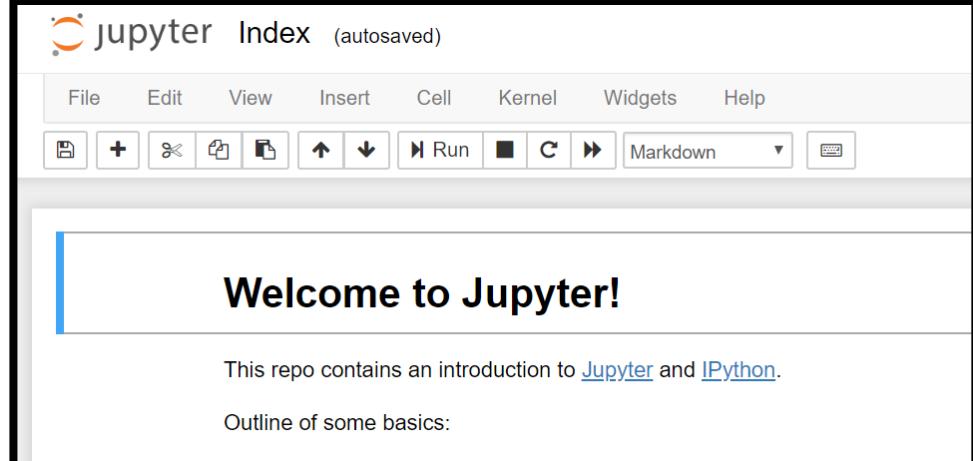


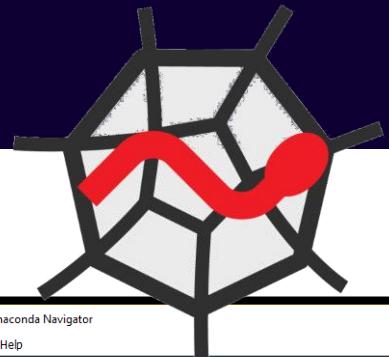
Jupyter

```
Anaconda Prompt  
C:\Users\Franco>set "KERAS_BACKEND=theano"  
(base) C:\Users\Franco>jupyter notebook
```

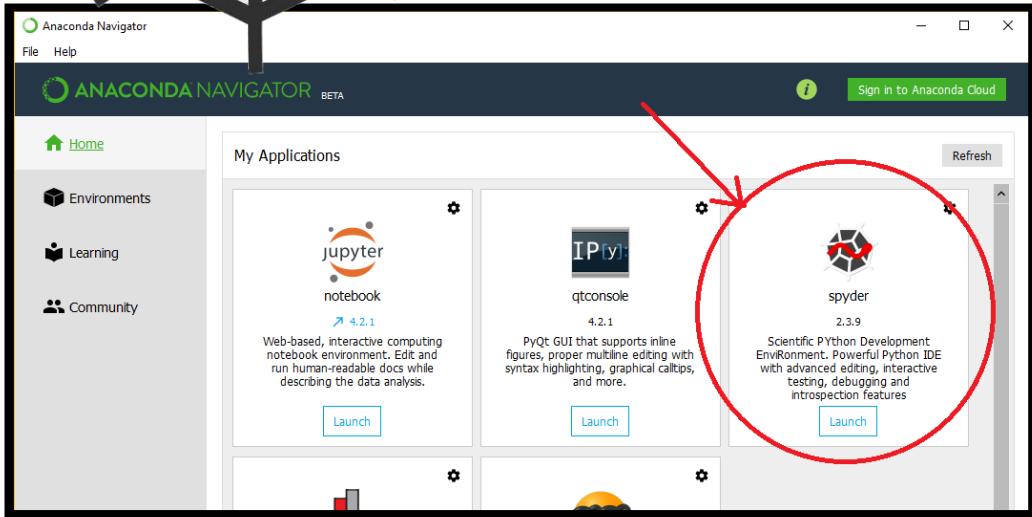


- Es un entorno web, de código abierto, que permite mezclar código con texto libre. Creado en 2014.
- Los archivos están organizados como “notebooks” (archivos .ipynb) que corren en un servidor (local o remoto).
- Ejecuta entornos iPython, pero tiene soporte para lenguajes como Julia, R, Haskell y Ruby.





Spyder



- Spyder es un entorno de desarrollo para el lenguaje Python.
- Permite codificar, así como visualizar el contenido de las variables, ejecutar código, depurar, etc.

Spyder (Python 3.6)

Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Proyectos Herramientas Ver Ayuda

Editor - D:\GDrive\LDI\Investigación\faces\faces_classifier.py Explorador de variables

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Jun 4 12:37:24 2018
4
5 @author: Franco
6 """
7 #%%
8 import numpy as np
9 from skimage.feature import hog
10 from skimage.feature import local_binary_patt
11 from skimage import exposure
12 import matplotlib.pyplot as plt
13 from sklearn.model_selection import train_test_split
14 from sklearn.neural_network import MLPClassifier
15 from sklearn.svm import SVC
16 from sklearn.metrics import confusion_matrix,
17 import itertools
18 import pickle
19
20 # Constantes =====
21 img_size= 64
22 # Parámetros para features
23 # HOG
24 features= 'hog' # original, gray, hog
25 hog_pixels_per_cell=(8,8)
26 hog_bins= 9
```

Explorador d... Explorador ... Ay... Perfilad... Análisis estático...

Nombre	Tipo	Tamaño	Valor
X	uint8	(11780, 64, 64)	ndarray obje...
X_features	float64	(11780, 608)	array([[0.43...
X_test	float64	(118, 608)	array([[0.18...
X_train	float64	(11662, 608)	array([[0.17...
Y	uint8	(11780,)	ndarray obje...

Terminal de IPython

Precision: 1.00
Accuracy: 1.00

In [6]: 1+1
Out[6]: 2

In [7]: x = 5

In [8]: x**2
Out[8]: 25

Terminal de IPython Historial de comandos

Permisos: RW Fin de línea: CRLF Codificación: UTF-8 Línea: 162 Columna: 48 Memoria: 63 %

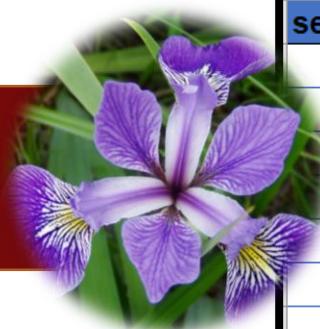
VISUALIZACIÓN Y PREPROCESAMIENTO DE DATOS

Machine Learning - Datos

Datos de entrada

- Pueden existir dos tipos principales de datos: numéricos y nominales (cuantitativos y cualitativos). Para utilizar algoritmos de Machine Learning siempre necesitamos datos numéricos.
- En ocasiones es necesario pre-procesar los datos antes de poder utilizarlos.
- Y en la mayoría de los casos, los datos no están completos, presentan valores nulos, o no tienen un formato amigable!

Un Dataset etiquetado

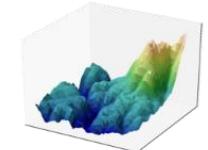
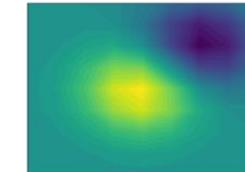
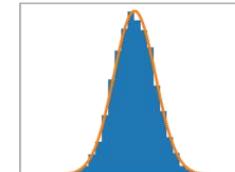
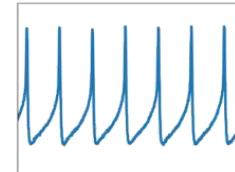


sepal length	sepal width	petal length	petal width	Class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa

Visualización



Version 2.2.2



Es la librería gráfica más utilizada para realizar gráficos en Python.

<https://matplotlib.org/tutorials>

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

Visualización

matplotlib.pyplot.plot

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)
```

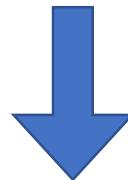
Visualización

Matplotlib permite adornar las figuras con mucha información adicional.

Por ejemplo:

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

`plt.figure()`

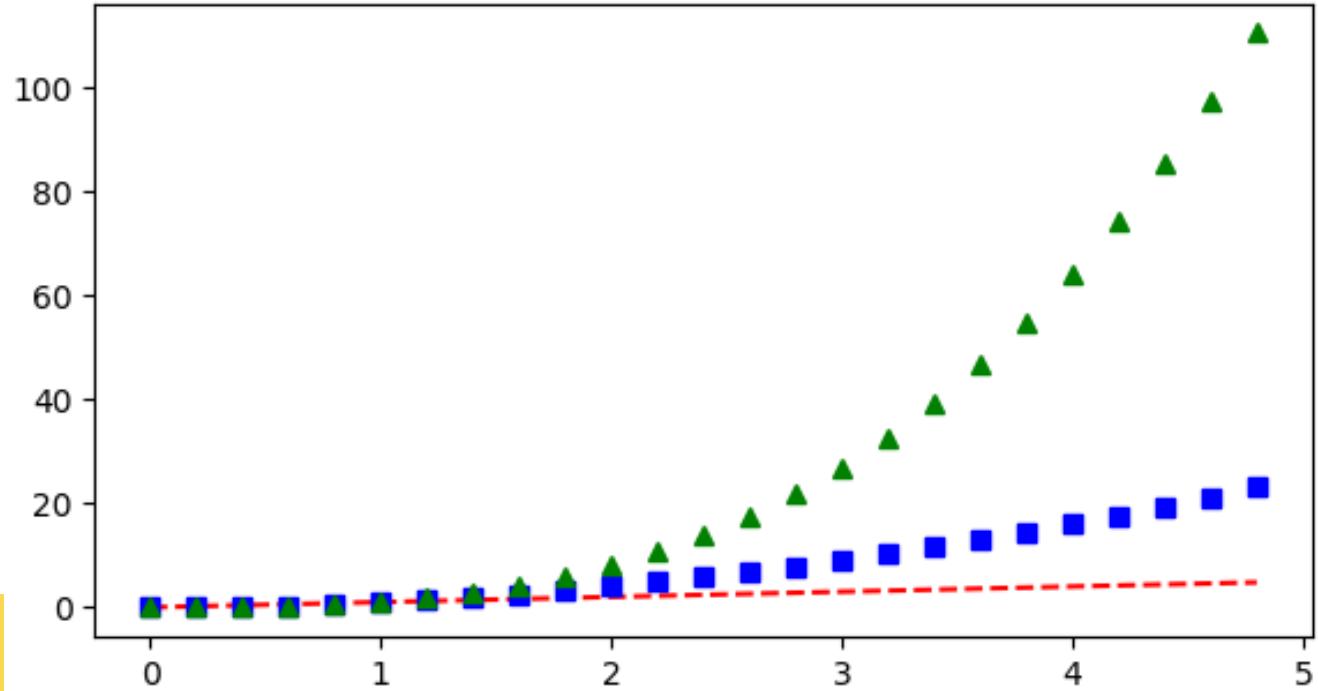


Este comando permite crear una figura nueva (nueva ventana). De lo contrario, todo lo ejecutado por *matplotlib* será en la figura activa.

Visualización

También podemos omitir graficar las líneas, y visualizar sólo los puntos que pasamos como argumento.

```
plt.figure()
# rango de datos
t = np.arange(0., 5., 0.2)
# lineas rojas, cuadrados azules y triángulos verdes
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



Visualización

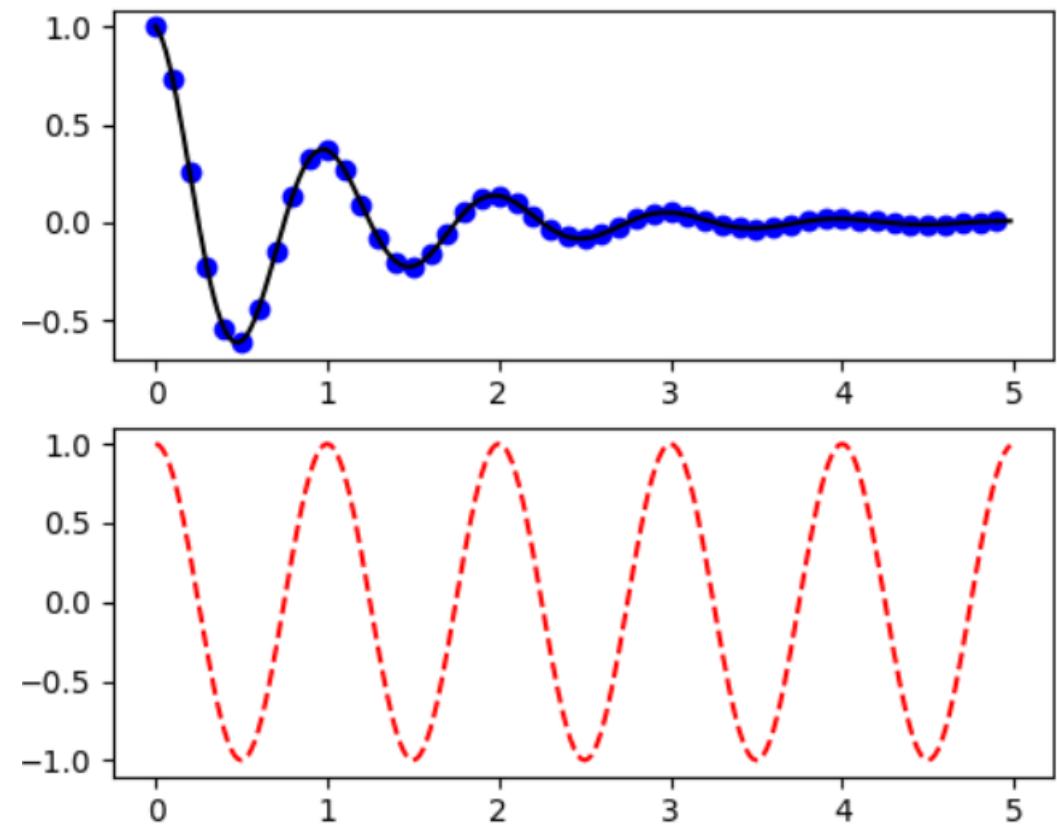
Graficando en varias subfiguras.

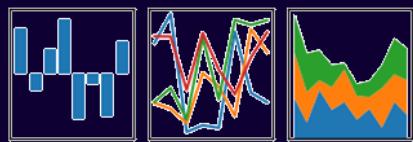
```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211) # 2 x 1, indice=1
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212) # 2 x 1, indice=2
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```





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, ya que es una versión de una hoja de cálculo (Excel) en Python. Cada columna tiene nombres, y pueden contener información de diferente tipo.

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

Crea un DataFrame

Retorna los valores
como un Array de
NumPy

Scatter plot

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

```
import pandas as pd
import matplotlib.pyplot as plt

raw_data = {'nombre': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
            'apellido': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze'],
            'femenino': [0, 1, 1, 0, 1],
            'edad': [42, 52, 36, 24, 73],
            'altura': [180, 170, 165, 175, 158],
            'peso': [77, 80, 61, 90, 65]} # esto es un diccionario

df = pd.DataFrame(raw_data, columns = ['nombre', 'apellido', 'edad', 'femenino', 'altura', 'peso'])
```

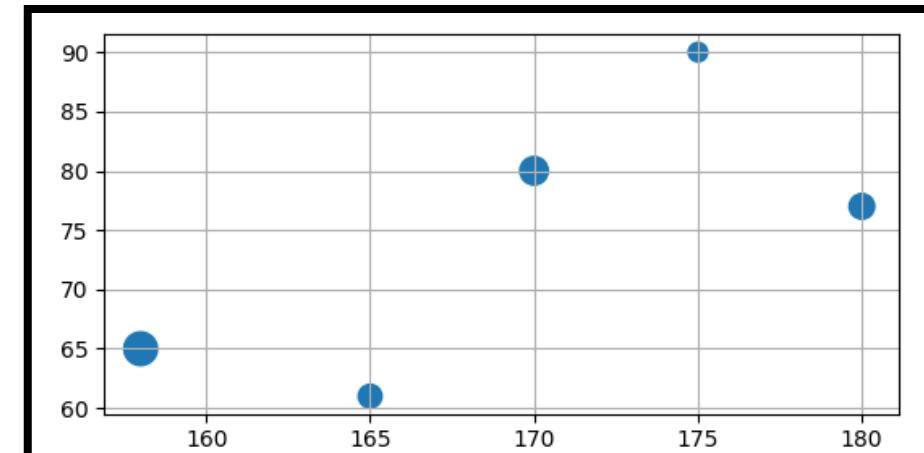
Diccionario de python

Crea un DataFrame con la información.

```
plt.figure()
plt.scatter(df.altura, df.peso, s=df.edad*3), plt.grid()
```

Los parámetros “x” e “y” son arreglos.

Grafica la altura en función del peso. El tamaño del punto es la edad



Ejemplo: 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 (,). Algunos archivos guardados en formato español podrían estar separados por “punto y coma” (;).

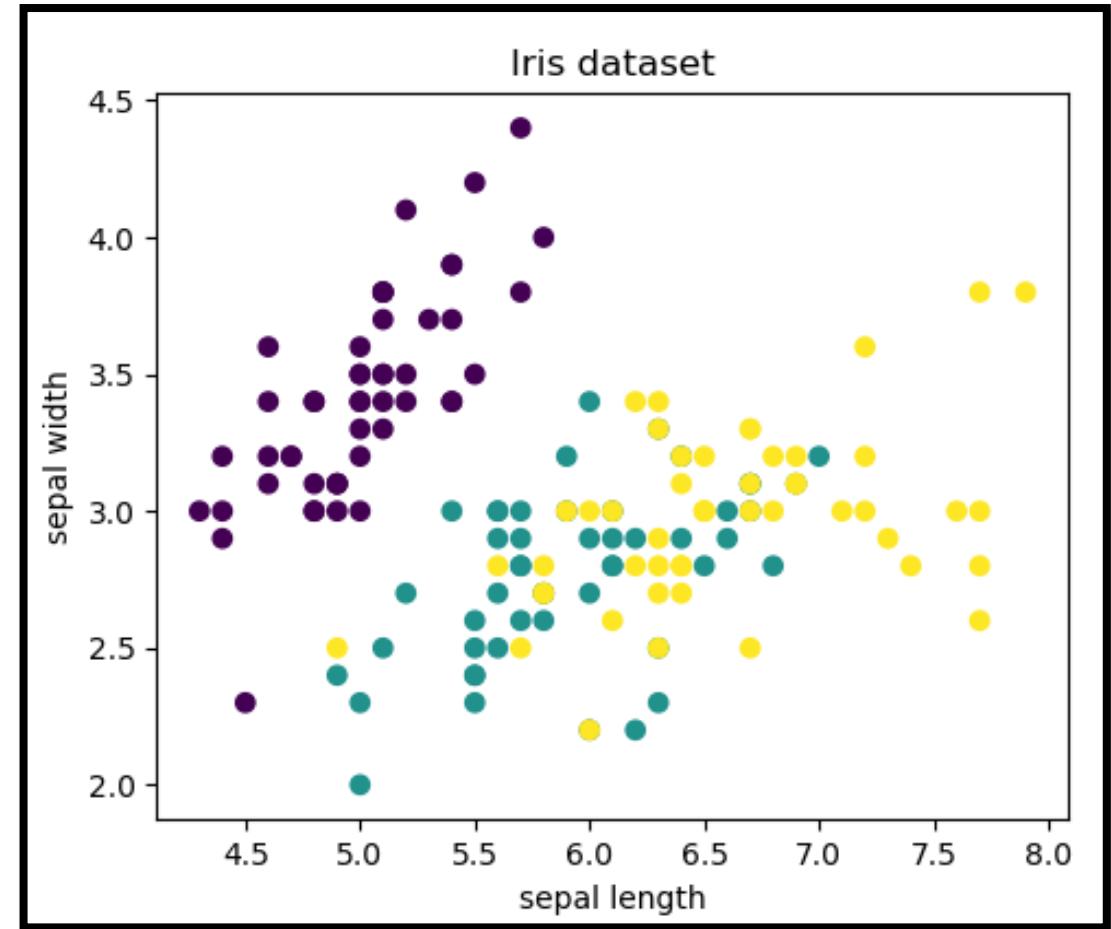
También es posible visualizar los datos en un entorno conocido, como podría ser MS Excel. El dataset contiene información sobre 3 especies distintas de flores. Contiene 4 atributos: tamaño y largo del pétalo, tamaño y largo del sépalo.



Visualización Iris

Al tener 4 variables, no podemos visualizar todas al mismo tiempo, pero podemos hacer un corte 2D para dos de ellas.

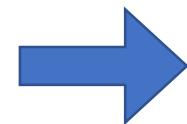
```
iris = pd.read_csv("iris.csv")
plt.figure()
plt.scatter( iris.sepal_length,
             iris.sepal_width,
             c=iris.name)
plt.title('Iris dataset')
plt.xlabel('sepal length')
plt.ylabel('sepal width')
```



Visualización con Pandas

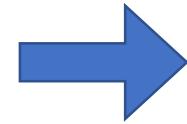
Pandas nos permite visualizar de forma rápida los datos con algunos comandos

```
iris.columns
```



Visualizar los nombres de los *features*:
*'sepal_length', 'sepal_width',
'petal_length', 'petal_width', 'name'*

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

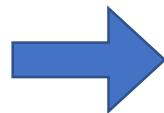


Visualizar los nombres de las clases:
'setosa', 'versicolor', 'virginica'

Visualización con Pandas

Pandas nos permite visualizar de forma rápida los datos con algunos comandos

`iris.head()`



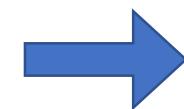
Si estamos en una consola, con este comando podemos ver los primeros datos en el dataframe

	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

Visualización con Pandas

Pandas nos permite visualizar de forma rápida los datos con algunos comandos

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
7	5	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

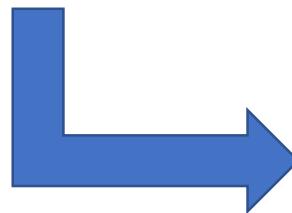


Visualización del
dataframe con Spyder

Visualización con Pandas

Pandas nos permite visualizar de forma rápida los datos con algunos comandos

`iris.corr()`



Matriz de correlación entre features

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

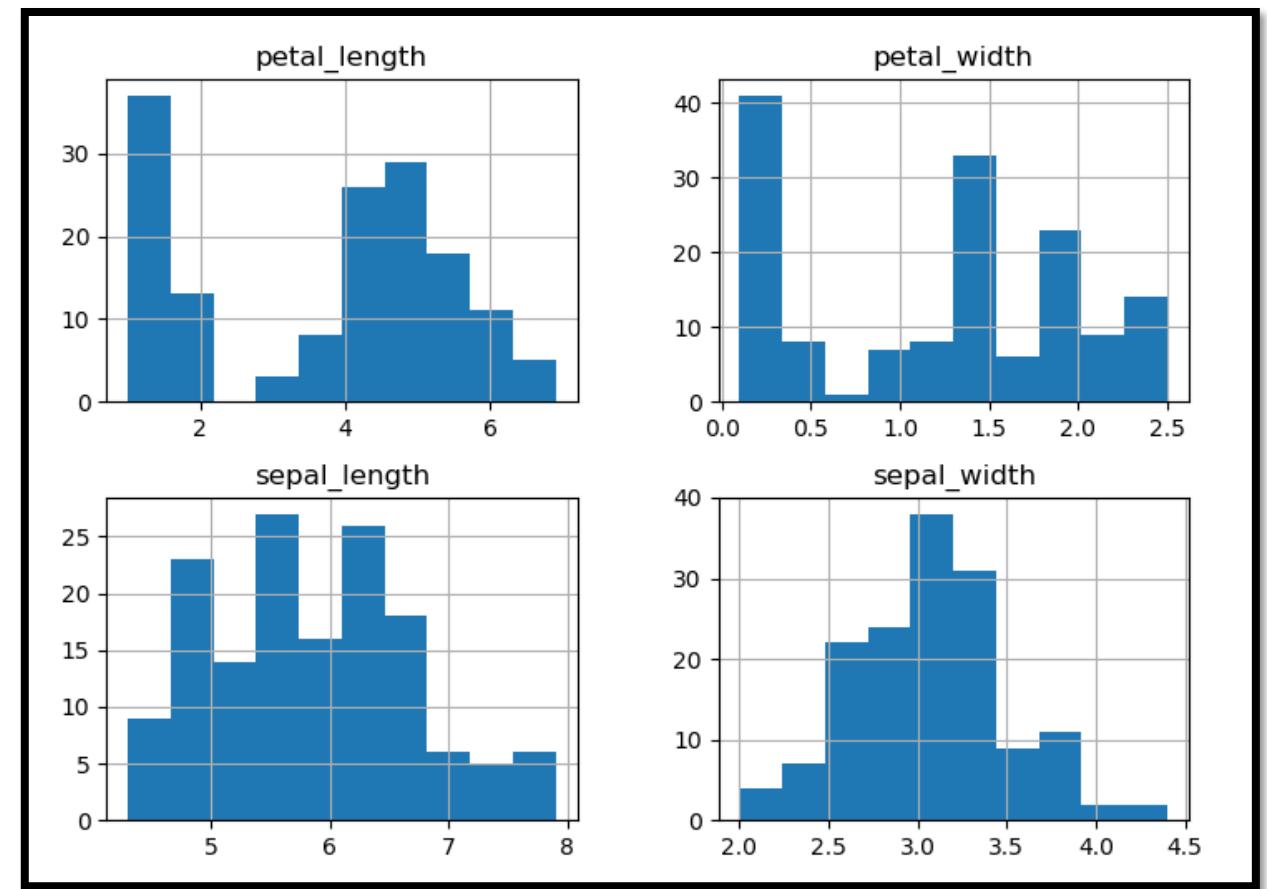
Visualización con Pandas

Pandas nos permite visualizar de forma rápida los datos con algunos comandos

`iris.hist()`



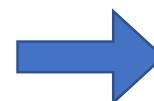
Crea un histograma para cada columna del dataframe



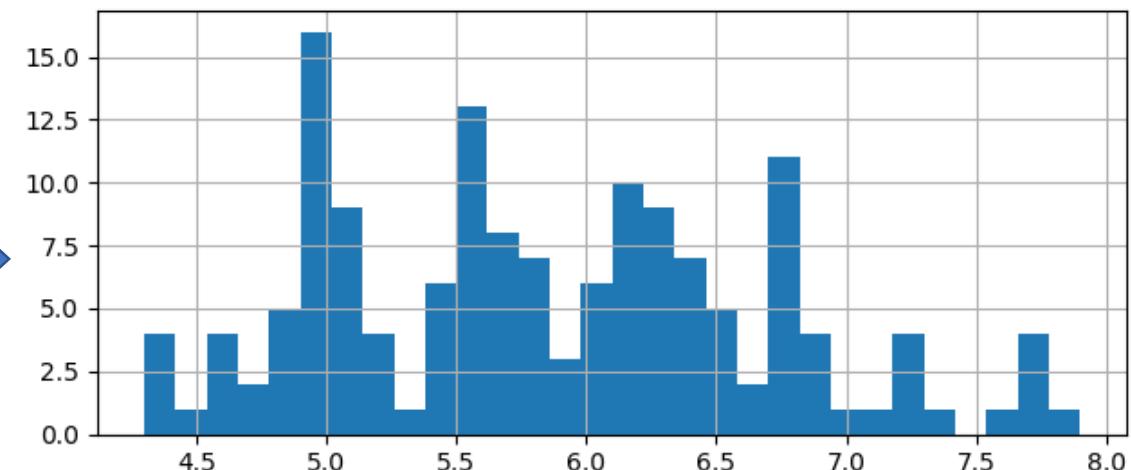
Visualización con Pandas

Pandas nos permite visualizar de forma rápida los datos con algunos comandos

```
iris['sepal_length'].hist(bins=30)
```



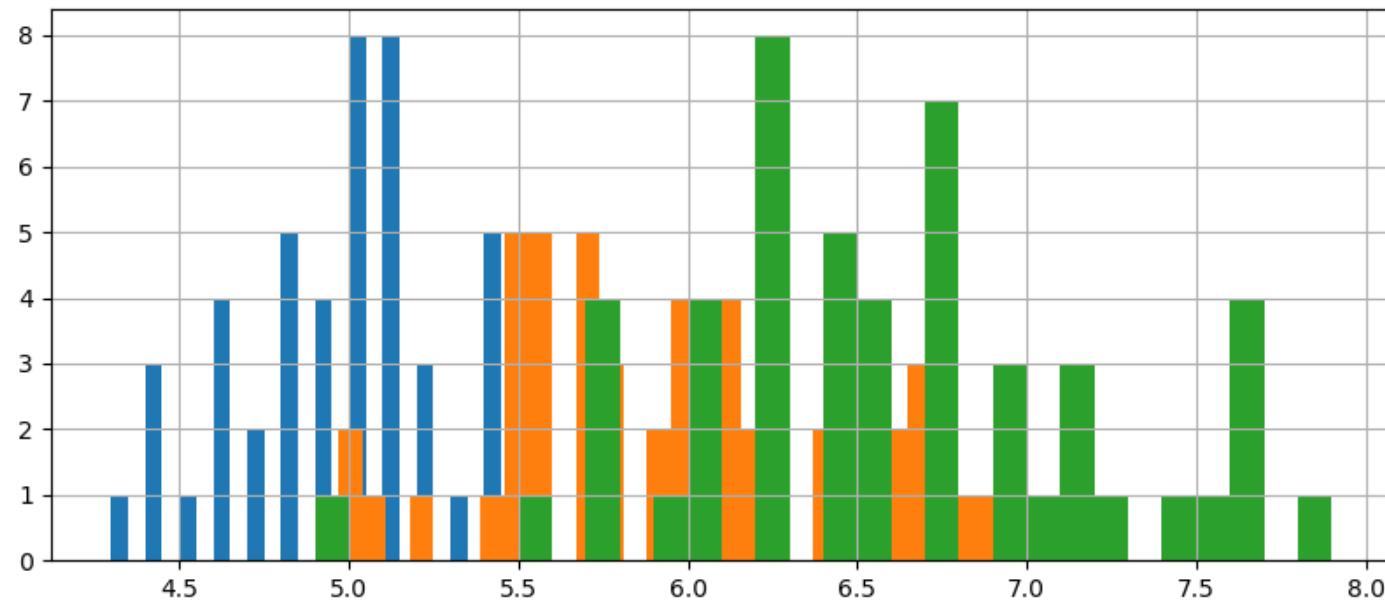
Otro modo de indexar el dataframe:
iris.sepal_length.hist(bins=30)



Crea un histograma para la columna
“sepal_length”

Visualización con Pandas

```
for class_number in np.unique(iris.name):
    iris['sepal_length'].iloc[np.where(iris.name == class_number)[0]].hist(bins=30)
```

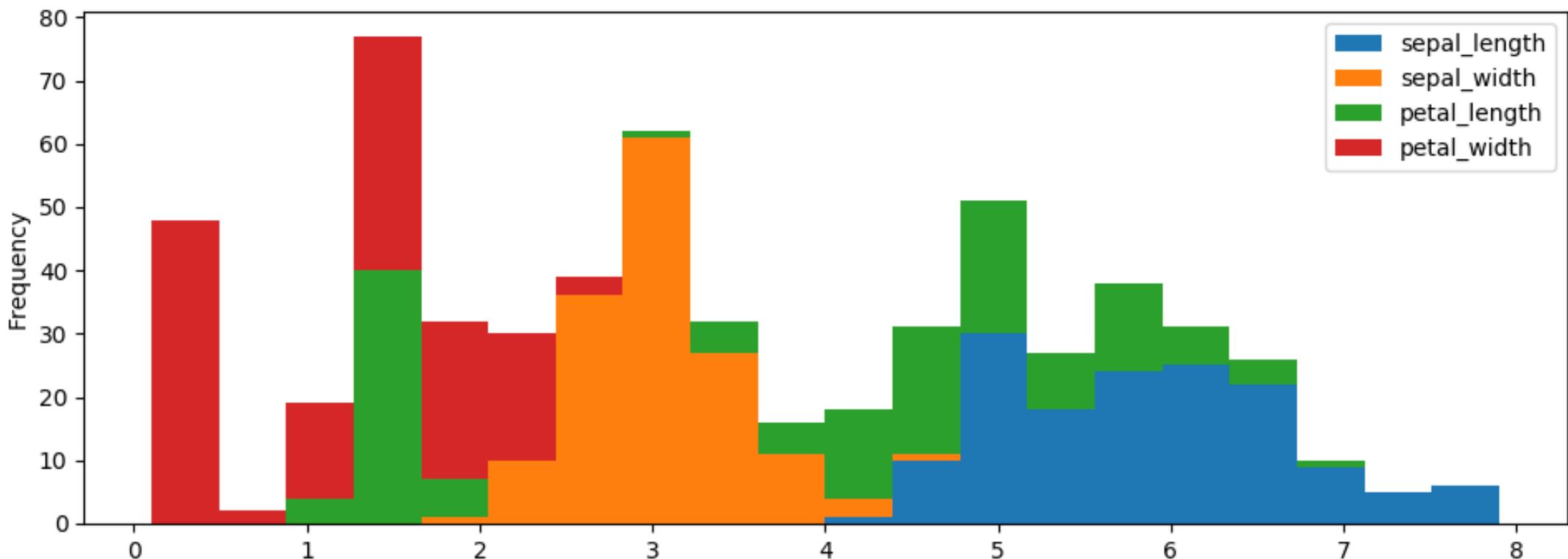


Crea un histograma de
“sepal_length” para cada
clase

Visualización con Pandas

```
iris.plot.hist(stacked=True, bins=20)
```

Crea un histograma para cada feature

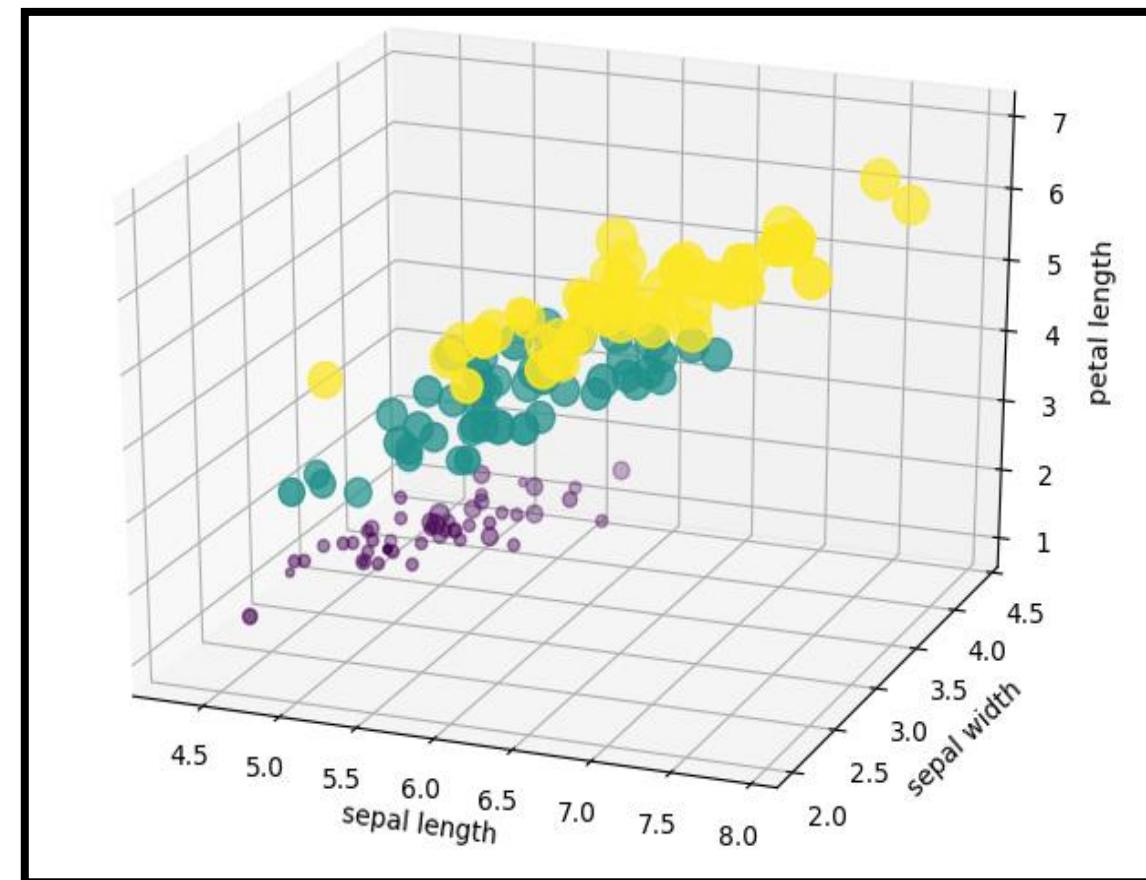


Visualización con Pandas

Una opción (no muy buena) para visualización de 4 variables podría ser graficar en 3D y utilizar la 4ta variable como tamaño del marcador.

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(    iris.sepal_length,
             iris.sepal_width,
             iris.petal_length,
             c=iris.name,
             s=iris.petal_width*100  )

plt.title('Iris dataset')
ax.set_xlabel('sepal length')
ax.set_ylabel('sepal width')
ax.set_zlabel('petal length')
```



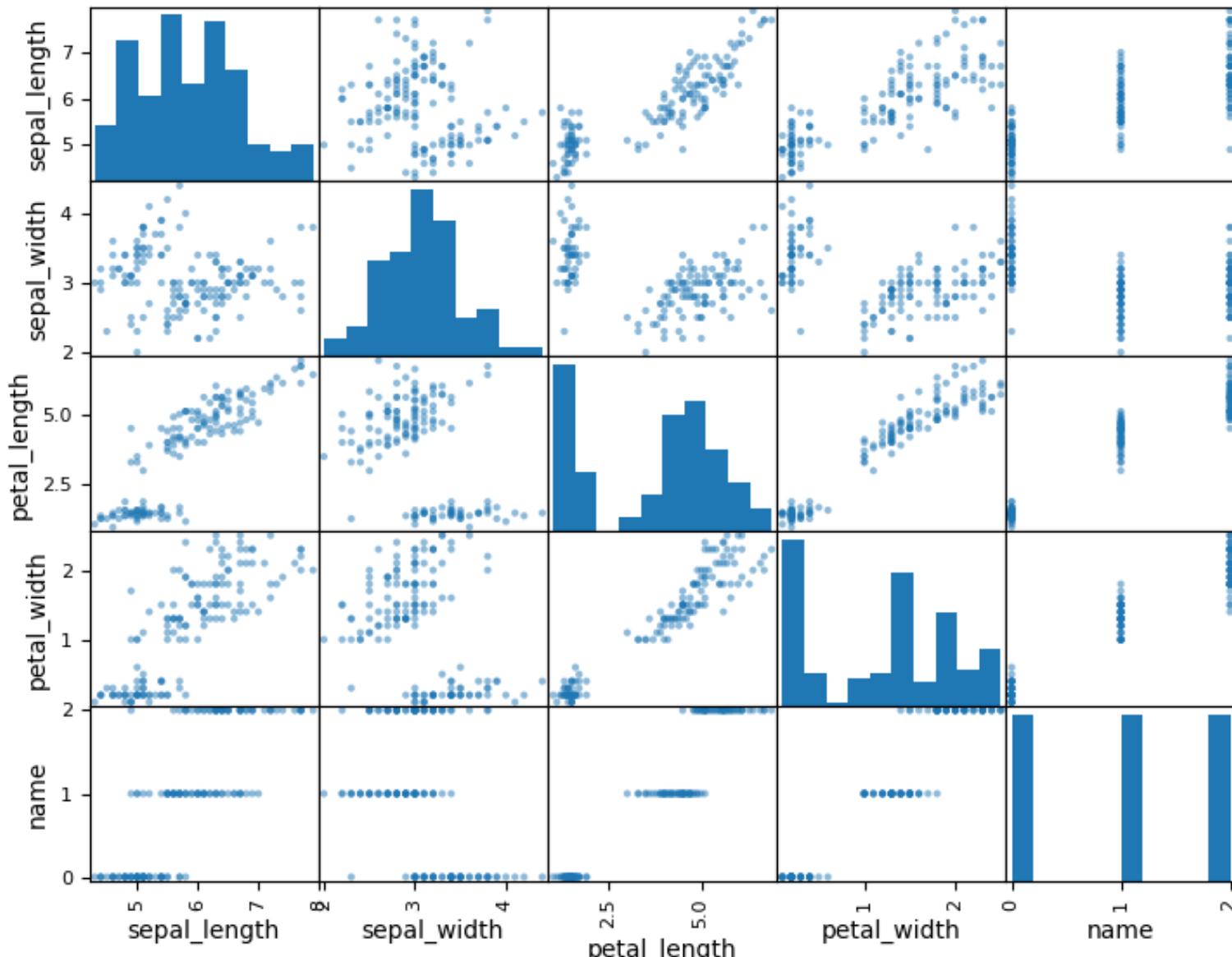
Visualización con Pandas

Matriz con scatter 2D para cada par de features

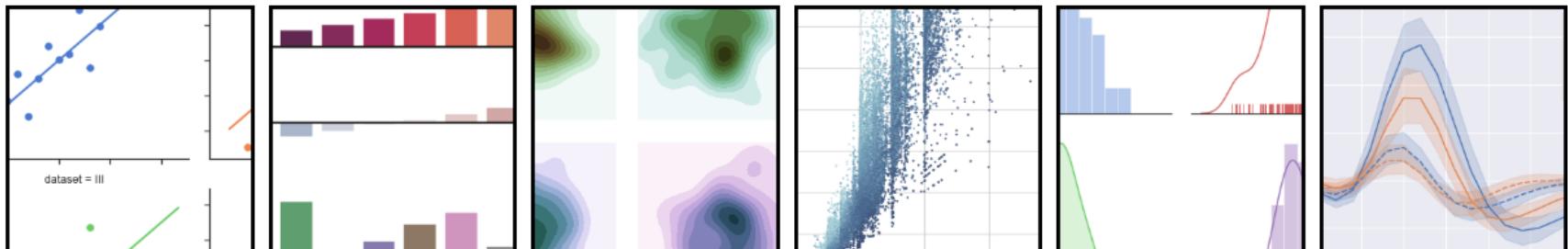
```
pd.scatter_matrix(iris)
```

Para meditar: son cortes en 2D de un espacio \mathbb{R}^4 .

No estamos viendo toda la información.

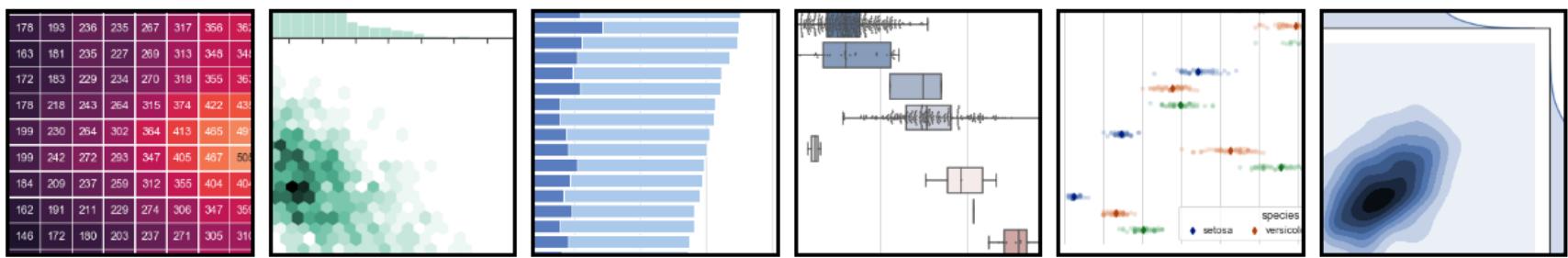
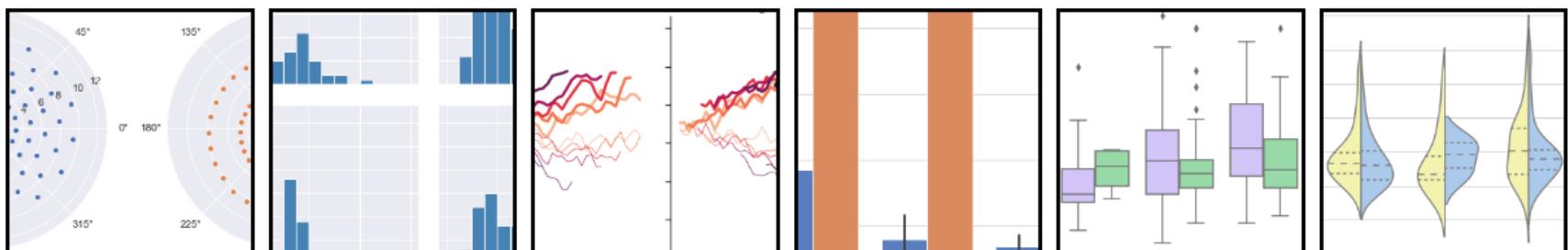


Visualización con Seaborn



<https://seaborn.pydata.org/>

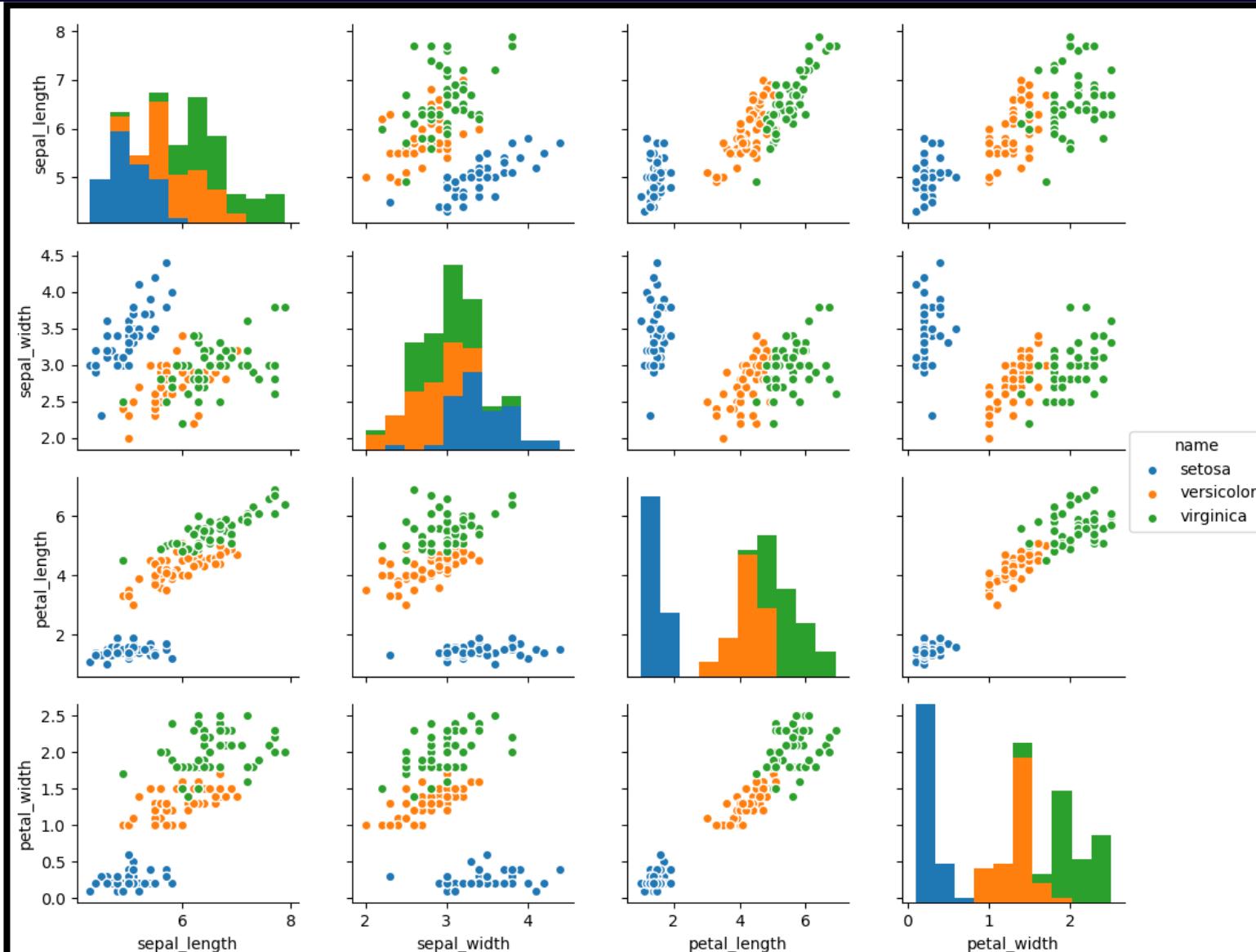
<https://www.kaggle.com/noelano/seaborn-visualization-on-iris-data-set>



Visualización con Seaborn

```
sns.pairplot(iris, hue='name')
```

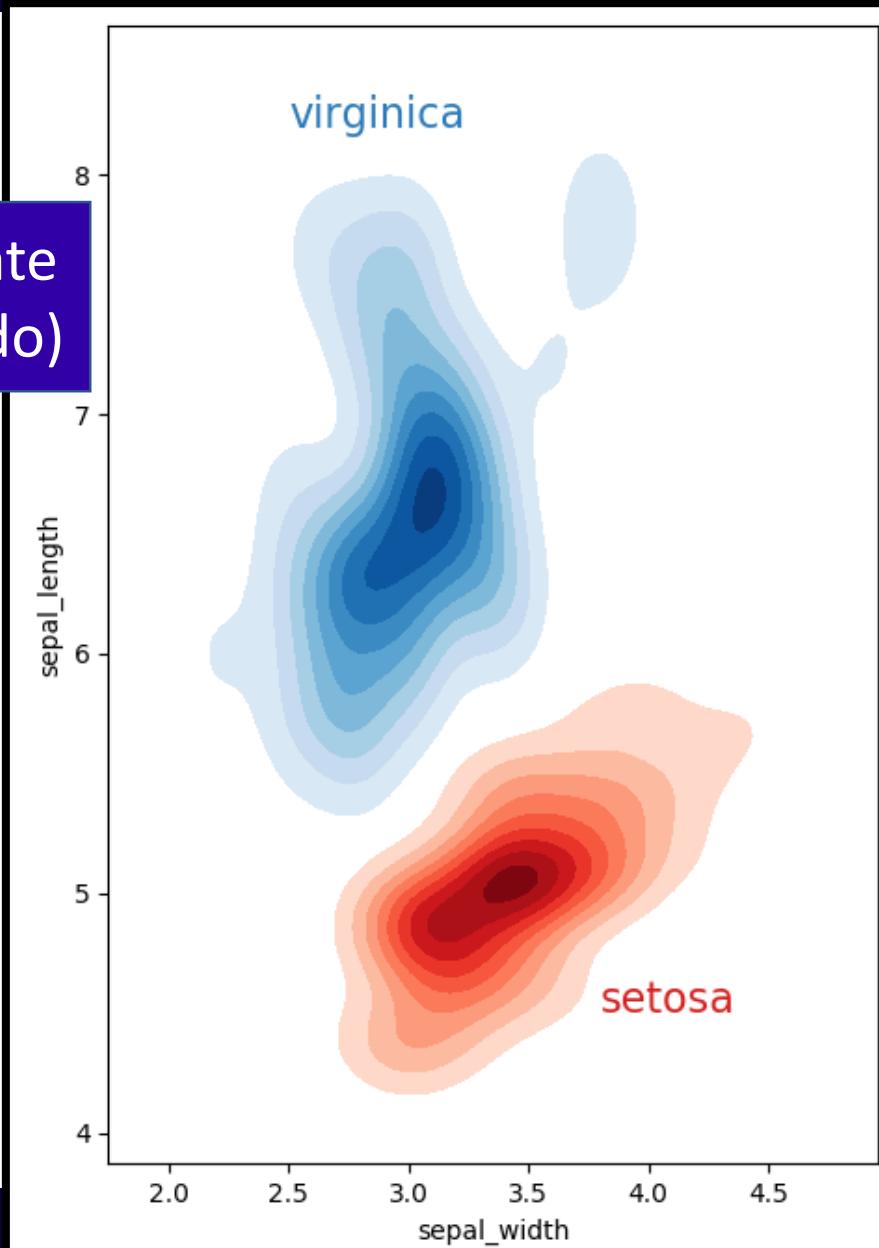
Iris Dataset
pairplot con SeaBorn
(igual a scatter_matrix)



Visualización con Seaborn

```
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)
```

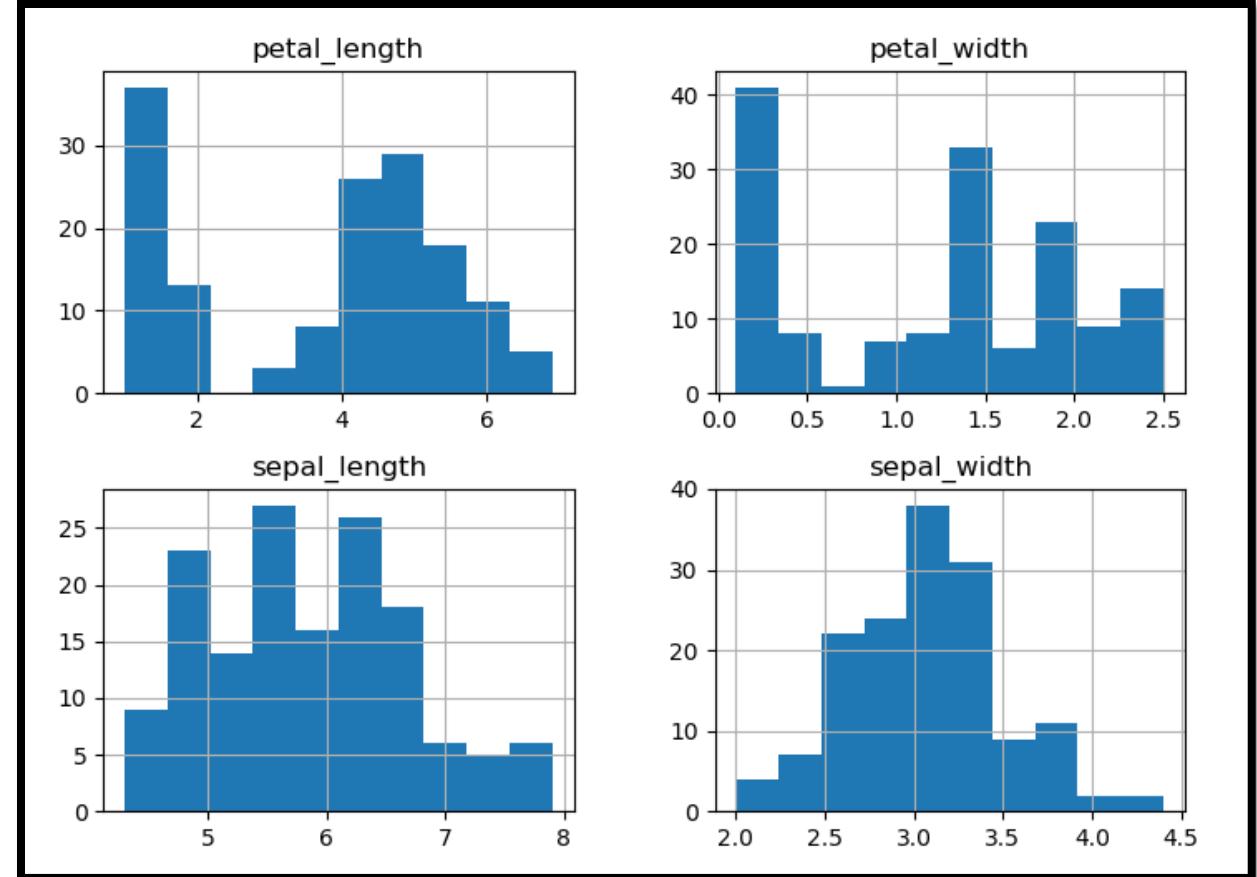
kernel density estimate
(univariado o bivariado)



Normalización de datos

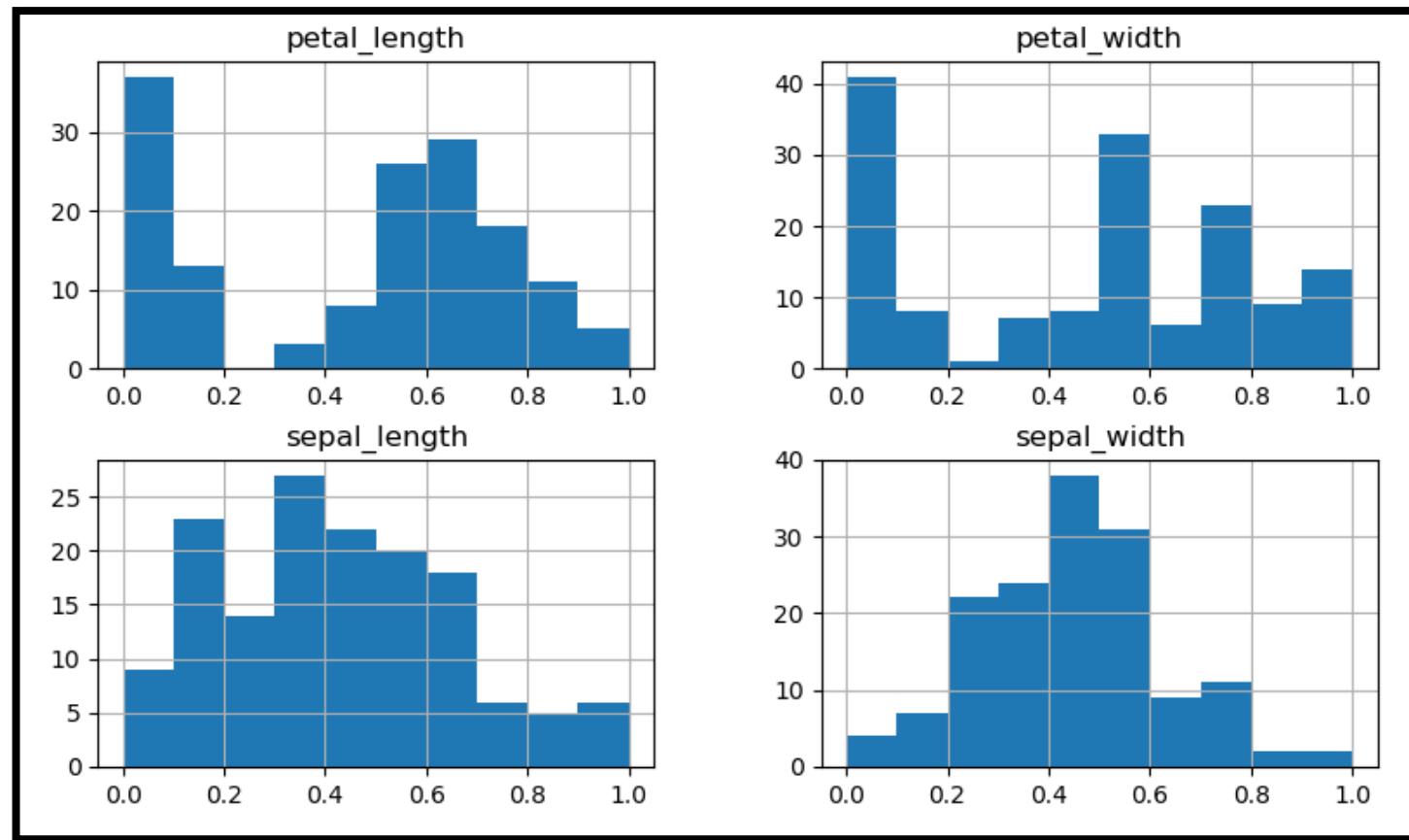
Para los modelos que veremos es un problema que cada *feature* posea su propio rango de datos.

Necesitamos normalizarlos de algún modo



Normalización de datos

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



Normalización de datos

Z-score (μ y σ): $x = (x - \text{mean}(x)) / \text{std}(x)$

