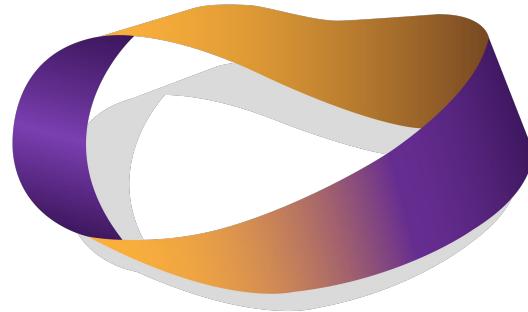


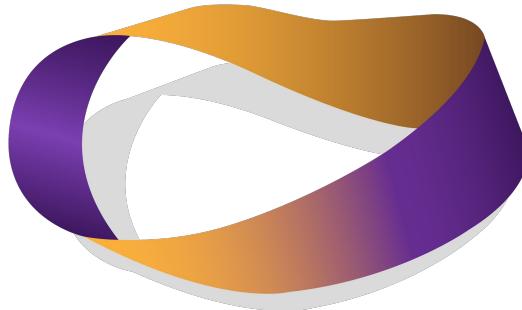
Machine Learning Práctico



ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Panorama general del aprendizaje automático



ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Algoritmos de ML:

Algoritmos de Regresión

- Regresión Lineal (Linear regression)
- Regresión Logística (Logistic Regression)

Algoritmos basados en Instancia

- k vecinos más cercanos (k-Nearest Neighbor kNN)
- Mapa auto organizado (Self-Organizing Map)

Algoritmos de Árbol de Decisión

- Árboles de Clasificación y Regresión (Classification And Regression Tree CART)
- Decisión de Árbol condicional (Decision tree learning)
- Bosques Aleatorios (Random Forest)

Algoritmos Bayesianos

- Clasificador bayesiano ingenuo (Naive Bayes)
- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Red bayesiana (Bayesian Network)

Algoritmos de agrupación (Clustering)

- K-medias (K-Means)
- K-medianas (K-Medians)
- Agrupamiento jerárquico (Hierarchical Clustering)

Máquinas de vectores de soporte (SVM)

Algoritmos de Redes Neuronales

- Compuerta XOR
- Perceptrón
- Retro propagación (Back-Propagation)
- Red de Hopfield (Hopfield Network)
- Red perceptrón multicapa (MLP Multi Layered Perceptron)

Algoritmos de Aprendizaje Profundo

- Redes neuronales convolucionales (Convolutional Neural Networks CNN)
- Redes de gran memoria de corto plazo (Long Short Term Memory Neural Networks LSTM)

Algoritmos de Reducción de Dimensión

- Análisis de componentes principales (Principal Component Analysis PCA)
- t-Incrustación estocástica de vecinos (t-distributed Stochastic Neighbor Embedding t-SNE)

Procesamiento del Lenguaje Natural (NLP)

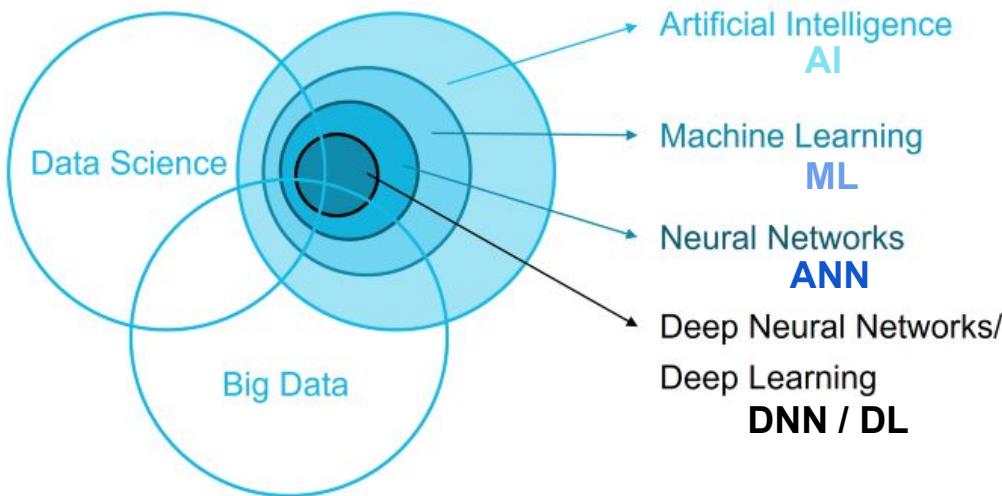
Otros Algoritmos

- algoritmos de Aprendizaje por Reglas de Asociación
- Algoritmos de Conjunto
- Visión por computadora (Computer Vision)
- Sistemas de Recomendación
- Algoritmos genéticos (Genetic Algorithms)

Y MÁS 0_0



¿Qué es Machine Learning?



Inteligencia Artificial (AI): Técnica con la que se pretende que las máquinas imiten el comportamiento humano. (¿Inteligencia?)

Aprendizaje de Máquina o **Aprendizaje automático (ML):** Rama de la Inteligencia Artificial que utiliza métodos estadísticos para que las máquinas aprendan de los datos (experiencia).

Redes Neuronales Artificiales (ANN): Rama del Aprendizaje de Máquina. Es un sistema computacional inspirado en neuronas biológicas.

Redes neuronales profundas o **Aprendizaje profundo (DNN):** Conjunto de técnicas poderosas para el aprendizaje de redes neuronales (Y sí, redes neuronales más profundas)

Machine Learning, Arthur Samuel (1959): Campo de estudio que dota a las computadoras con la habilidad de aprender sin ser explícitamente programadas.

¿Por qué usar aprendizaje automático?

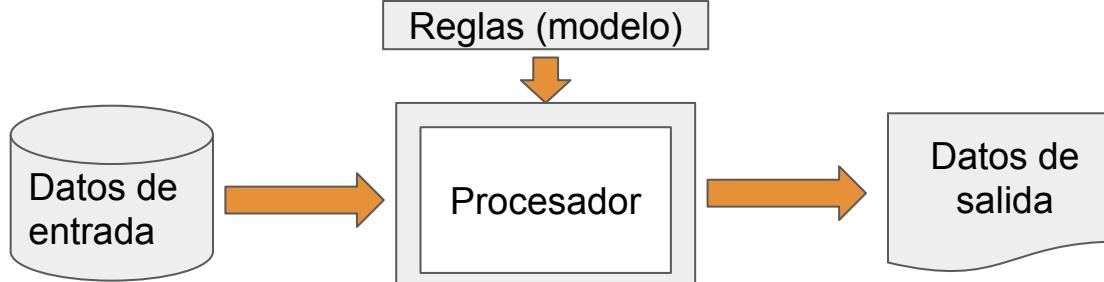
Programación tradicional:

- Existe un algoritmo que resuelve el problema correctamente
- Es computable en un tiempo razonable

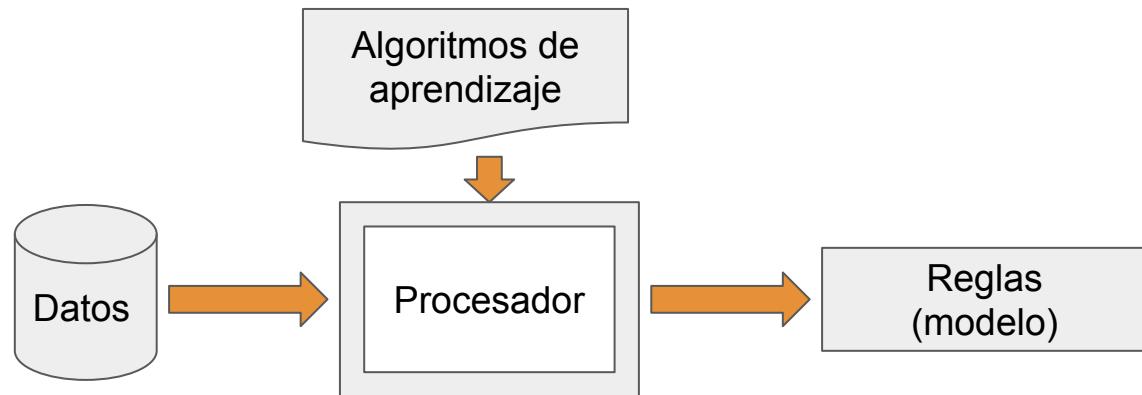
Aprendizaje automático:

- Existen problemas que son demasiado complejos para métodos tradicionales
- No existen algoritmos que resuelvan el problema correctamente
- El sistema es cambiante, y necesita adaptarse frecuentemente
- Ayudarnos a descubrir patrones que no son evidentes

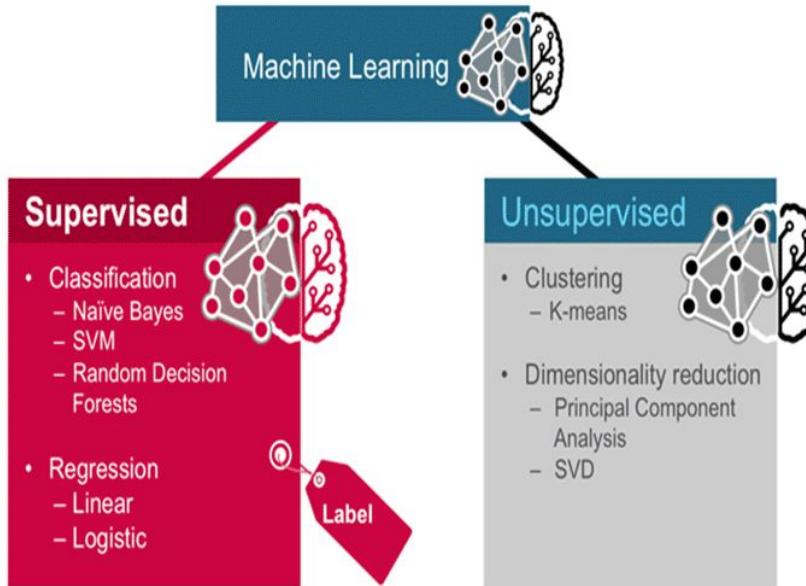
Programación clásica



Aprendizaje automático



¿Cómo puedo saber el alcance de Aprendizaje Automático?



Alcance del curso:

Clasificación, Regresión, Máquinas de soporte vectorial, Árboles de decisión, Regresión logística, Naive Bayes, Vecinos más cercanos, Redes neuronales multicapa, Teorema de “No free Lunch”, modelos ensamblados y bosques aleatorios, reducción de dimensionalidad.

Primer aplicación importante de ML, en los 90's con el filtro de spam

Aprendizaje supervisado y no supervisado

Supervisado

Se cuenta con:

Un conjunto de datos X_m y de etiquetas Y_m
 $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$

Se obtiene:

Un modelo predictivo
Predecir un comportamiento o clase para
datos que no se han visto

$$Y = f(X)$$

Grupos de algoritmos:

- Clasificación
- Regresión

No supervisado

Se cuenta con:

Un conjunto de datos X_m
 $\{(x_1), (x_2), \dots, (x_m)\}$

Se obtiene:

Un modelo descriptivo
Obtener más información sobre los datos:

- Patrones
- Estructuras
- Distribuciones

Grupos de algoritmos:

- Agrupamiento
- Reglas de asociación

Aprendizaje por refuerzo

Se cuenta con:

Agentes, ambientes, estados, acciones y recompensas

Se obtiene:

Un comportamiento del agente ante su ambiente.

Controlado por políticas auto aprendidas que le dicen qué acciones tomar para maximizar recompensas (positivas) y/o minimizar riesgos (negativas)

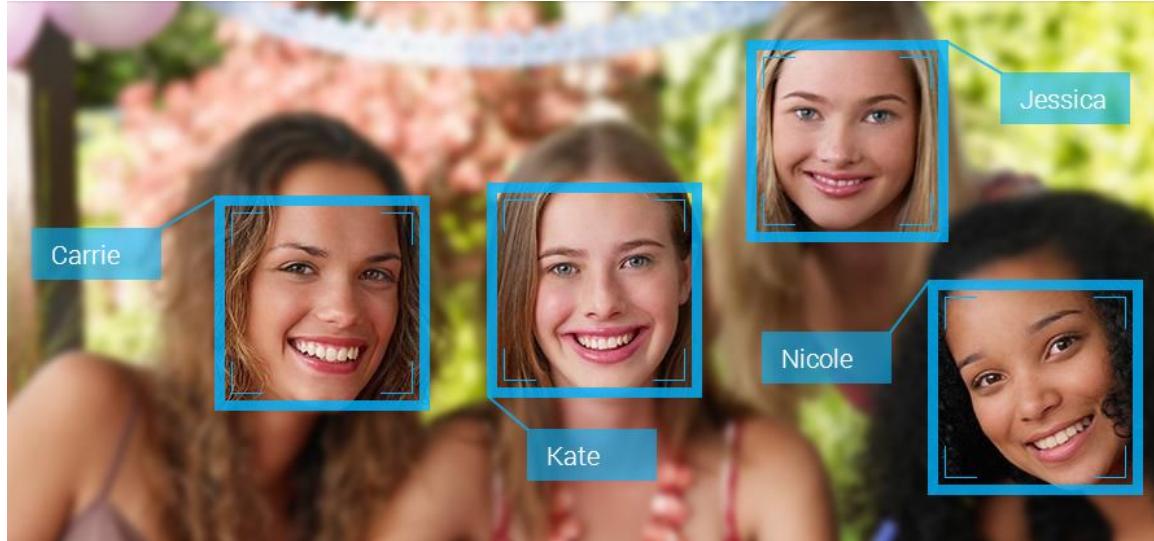
Aplicaciones:

- Juegos (de mesa, videojuegos)
- Brazos robóticos
- Vehículos autónomos

Ejercicio:

Describa un ejemplo de aprendizaje supervisado, no supervisado y por refuerzo

Ejemplo de aprendizaje supervisado



¿Como sabe los nombres?

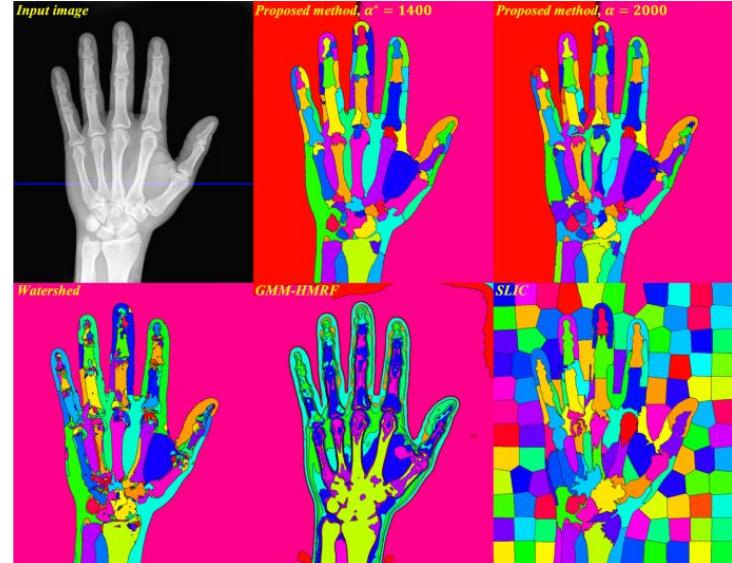
Ejemplo de aprendizaje no supervisado



Análisis de la canasta de mercado:

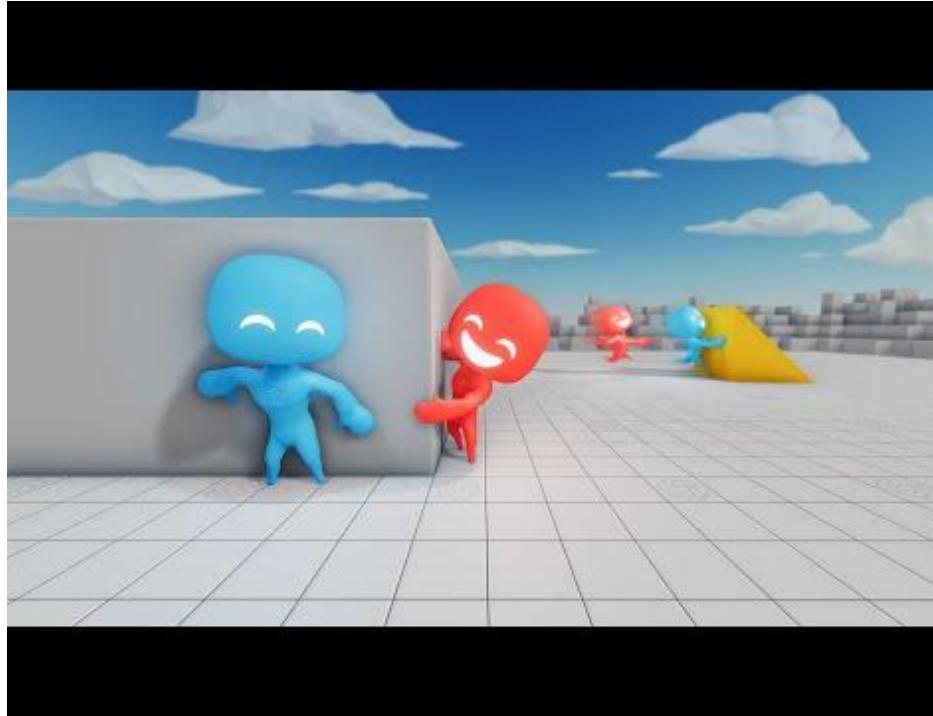
Comprender patrones de compra puede ayudar a aumentar las ventas.

Colocación de artículos A y B, que con frecuencia se compran juntos



Segmentación y búsqueda de patrones

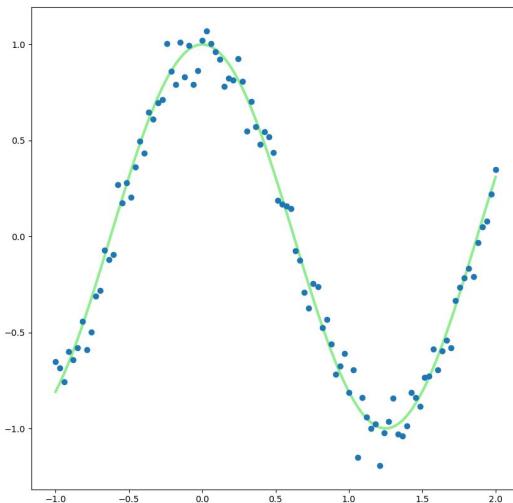
Ejemplo de aprendizaje por reforzamiento



Aprender a esconderse y buscar

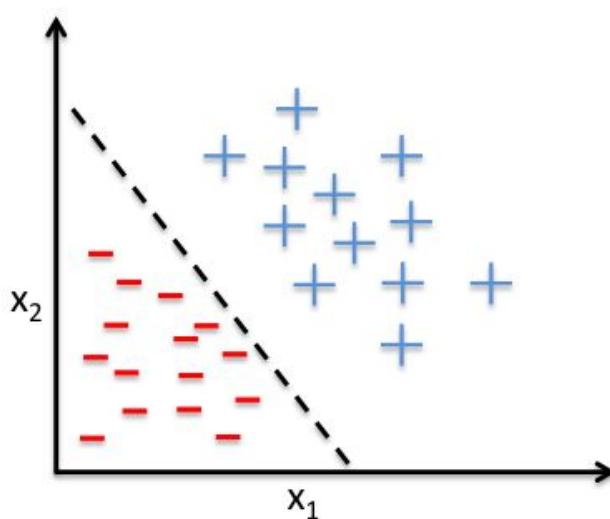
Saber más <https://openai.com/blog/emergent-tool-use/>

Regresión



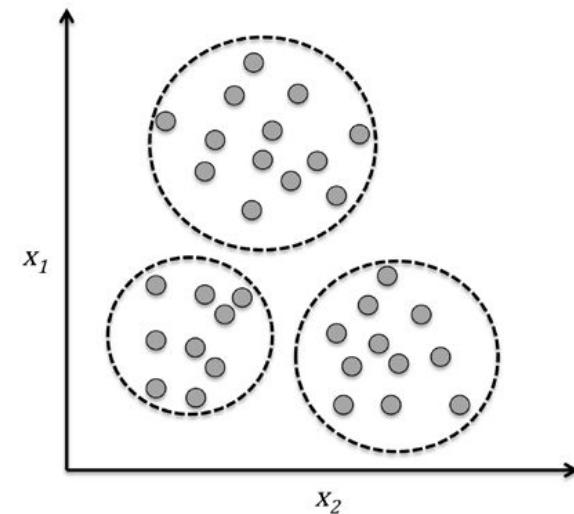
Tomamos el **conjunto de entrenamiento** para tratar de generar una función que se ajuste a los datos (predecir un valor numérico)

Clasificación



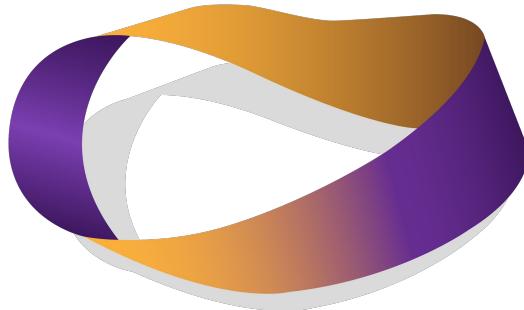
Separa en clases, esto es cuando la variable de salida es una categoría, como como rojo o azul o enfermedad y sin enfermedad.

Agrupamiento



Es la división de los datos en grupos con **rasgos** similares (sin conocer las clases).

Herramientas y ambientes de trabajo



ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

[Python](#): Utilizamos Python 3 porque es un lenguaje fácil de aprender, fácil de entender y de utilizar, sobre todo para prototipado rápido

Colab: [Colaboratory](#) es un entorno de notebook de Jupyter gratuito que no requiere configuración y se ejecuta completamente en la nube.

Ambiente virtual [miniconda](#): Conda es un sistema de gestión de paquetes de código abierto y un sistema de gestión del entorno que se ejecuta en Windows, macOS y Linux

[Scikit-learn](#): Es una biblioteca de aprendizaje automático de software gratuito para el lenguaje de programación Python. implementa muchos algoritmos de Machine Learning eficientemente por lo que es un gran punto de entrada para aprender. [Video](#).

[Tensorflow](#): es una biblioteca más compleja para computación numérica distribuida utilizando grafos de flujo de datos.
[TF - GPU](#)

Deep Learning Framework	Release Year	Written in which language?	CUDA support
TensorFlow	2015	C++, Python	Yes
Keras	2015	Python	Yes
PyTorch	2016	Python, C	Yes
Caffe	2013	C++	Yes
Deeplearning4j	2014	C++, Java	Yes

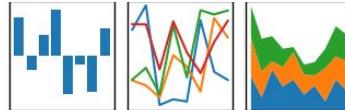
Numpy: Es un paquete para computación científica con Python, ver [tutorial](#). Es de utilidad práctica para manejo de arreglos de objetos N-dimensionales



Pandas: Es una biblioteca de alto desempeño para Python, con manejo de estructuras de datos y herramientas de análisis de datos, ver [documentación](#).



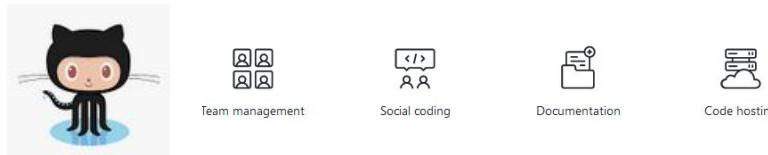
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Matplotlib: Es una biblioteca para generar en pocas líneas: diagramas, histogramas, imágenes, espectros de potencia, diagramas de barras, diagramas de error, diagramas de dispersión y más. Ver [tutorial](#).



GitHub: Es una plataforma para colaboración y control de versiones. Ver [ayuda](#).



Lectura recomendable: [Why You Need Python Environments and How to Manage Them with Conda](#)

Hardware - Software

Requisitos CPU:

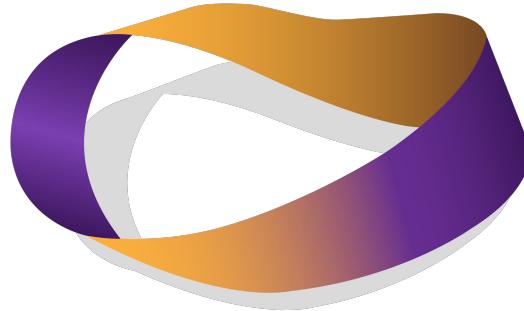
- Laptop
- Instalar [miniconda](#)
- Python 3

Requisitos GPU:

- Nvidia gpu
- Instalar miniconda
- Python 3
- Tensorflow gpu [TF - GPU](#)
- Cuda Toolkit
- CudNN

Seguir las instrucciones del manual de instalación de ambientes para Ubuntu-Mac o Windows, en su material para estudiantes.

¿Existe un modelo perfecto?



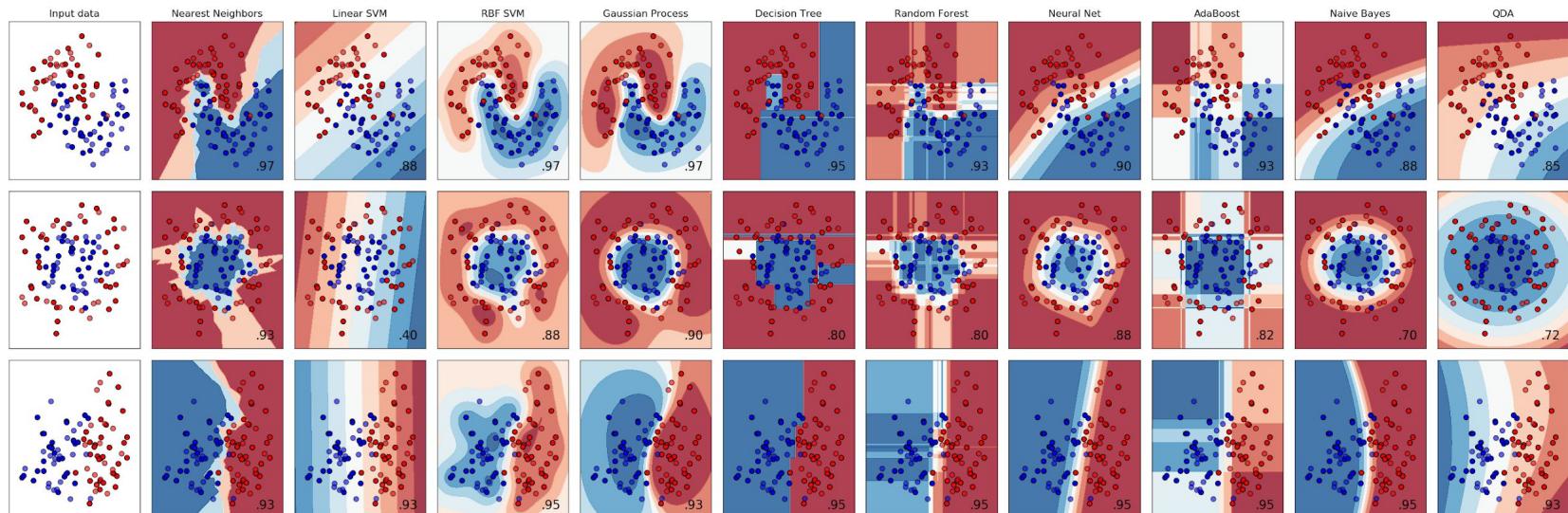
ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Reto: Active su ambiente local creado con miniconda en su equipo, busque “sklearn Classifier comparison” en los documentos y abra la página.

Ejecute el código y analicelo junto al instructor.

Resultado Esperado:



Explorando conjuntos de datos



ACTUMLOGOS

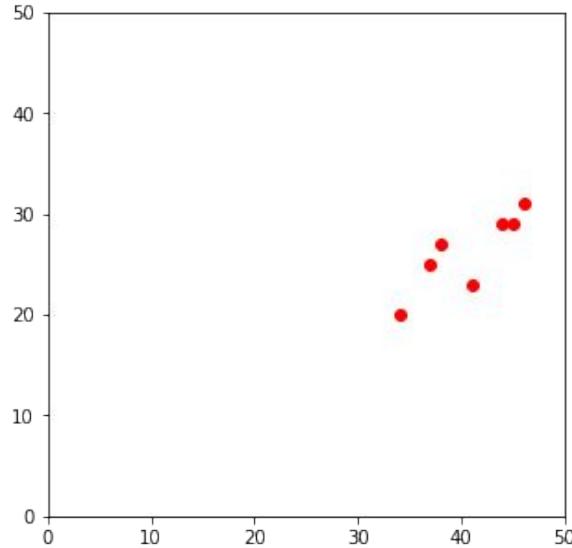
DESARROLLANDO HABILIDADES TECNOLÓGICAS

Reto: En 5 minutos, en la **primer celda** escriba un código que grafique los siguientes puntos: $\{(44, 29), (45, 29), (34, 20), (37, 25), (38, 27), (41, 23), (46, 31)\}$ Utilice listas de numpy, una para los valores x y otra para los valores y ejecute la **segunda celda** para verificar los datos.

Tips:

- Use las bibliotecas matplotlib y numpy
- Use las funciones ->
figure, plot, show y xlim, ylim de la biblioteca pyplot
- Consulte en “San Google”

Resultado Esperado:



```
[44 45 34 37 38 41 46]  
x_list.shape: (7,)  
X_list.shape: (7, 1)  
[[44]  
 [45]  
 [34]  
 [37]  
 [38]  
 [41]  
 [46]]
```

Solución:

```
1 # Importar librerias / packages
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 %matplotlib inline
6 # Definir los datos
7 x_list = np.array([44, 45, 34, 37, 38, 41, 46])
8 y_list = np.array([29, 29, 20, 25, 27, 23, 31])
9
10 # Figura 2D
11 plt.figure(figsize=(5, 5))
12 plt.plot(x_list, y_list, "ro")
13 plt.xlim(0, 50) # Límite del eje x
14 plt.ylim(0,50) # Límite del eje y
15 plt.show() # instrucción para genera el gráfico
```

Regresión lineal

$$\hat{y} = \theta^T x \quad \rightarrow \quad \hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

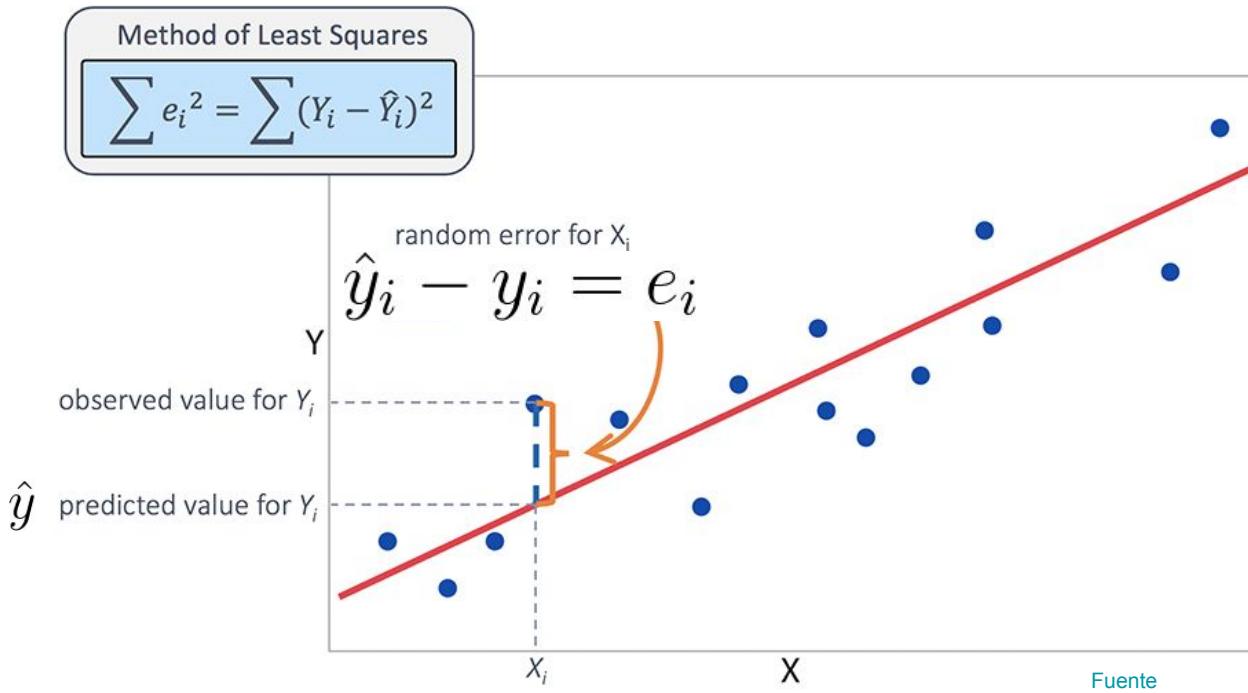
↓ ↓
Valor predicho Sesgo (bias)

x_i rasgos (features)
 θ_i pesos (weights)

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b = w_1 x_1 + w_2 x_2 + \cdots + w_r x_r + b$$

Nomenclatura que usaremos en el curso

Objetivo: Generar la recta que minimice el error, esta será la que mejor se ajuste a los datos



Reto: En 10 minutos complete las **celdas 3 y 4**. Utilice el modelo de regresión lineal implementado con scikit-learn para ajustar los datos generados del código anterior. Guardar la pendiente m , y el cruce con el eje Y b , obtenidos por la regresión para poder graficar el modelo:

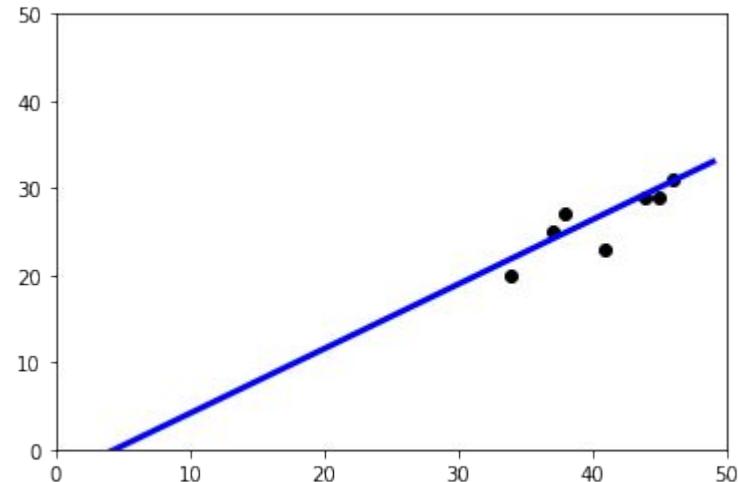
$$f_1(x) = m_1x + b_1$$

Tips:

- Use la biblioteca sklearn
- Use las funciones y métodos->
LinearRegression, fit, predict, scatter
coef_, intercept_
- Consulte en “San Google”

Resultado Esperado:

Modelo 1: $y = 0.726x + -3.260$



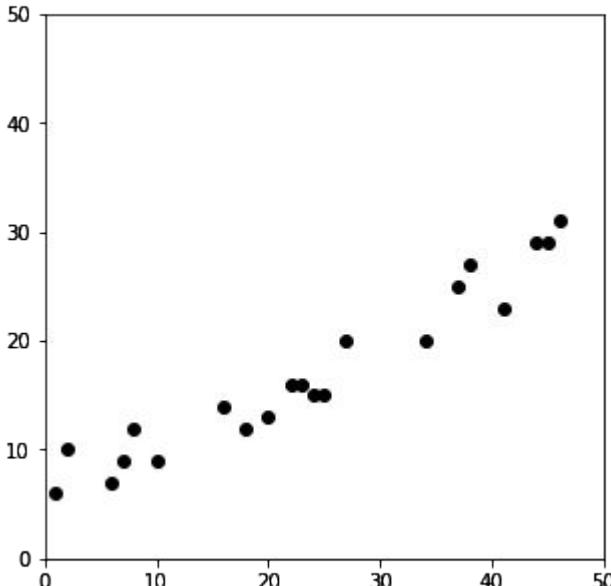
Solución:

```
1 # Importar librerias / packages
2 from sklearn.linear_model import LinearRegression
3
4 def obtener_modelo(X, Y):
5     model = LinearRegression() # Se crea un objeto LinearRegression
6     model.fit(X, Y) # Se ajusta a los datos (regresión)
7     m = model.coef_ # La pendiente del modelo
8     b = model.intercept_ # El cruce con el eje y
9     x_range = np.linspace(0, 50)
10
11     # Obtenemos la recta
12     y_predict = model.predict(x_range.reshape(-1, 1))
13     return m, b, y_predict
```

```
1 ml, bl, fxl = obtener_modelo(X_list, y_list)
2 # Imprimimos el modelo obtenido
3 print("Modelo 1: y = %.3fx + %.3f" %(ml, bl)) # Print con formateo
4
5 # Plot de la regresión
6 plt.scatter(X_list, y_list, color='black') # pone los puntos
7 plt.plot(fxl, color='blue', linewidth=3) # regresión
8 plt.xlim(0, 50) # Límite del eje x
9 plt.ylim(0,50) # Límite del eje y
10 plt.show()
```

¿Y si no tenemos suficientes datos?

Puede que nuestro modelo aprenda una distribución que no es cercana a la realidad y tengamos un sesgo alto:



En 1948, Thomas Dewey se enfrentó al titular Harry Truman para la presidencia de EUA. El Chicago Daily Tribune, un periódico de tendencia republicana realizó una encuesta que pronosticaba una victoria decisiva para Dewey. La noche anterior a las elecciones, el Tribune publica el título ahora infame: DEWEY DERROTA A TRUMAN.

Descubrieron que hubo un sobremuestreo de republicanos en sus datos por una razón bastante simple: **la encuesta se realizó completamente por teléfono**. Dado que **las personas ricas** tenían más probabilidades de tener un teléfono y también **más probabilidades de identificarse como republicanos**, la encuesta se inclinó significativamente hacia Dewey.

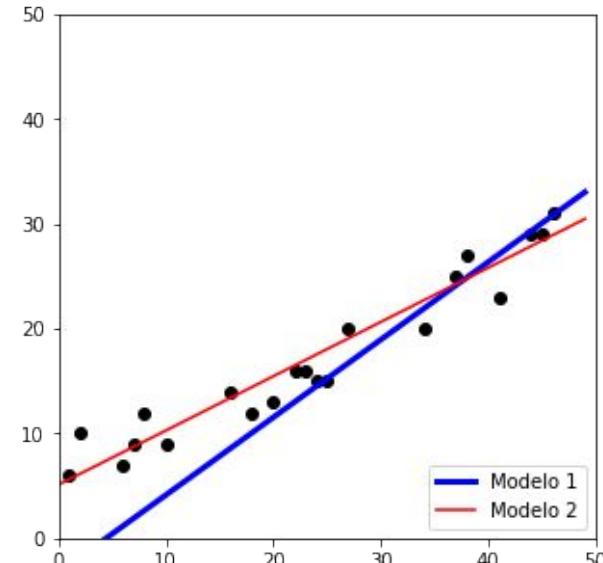
Reto: En 5 min en la **celda 5** obtenga un nuevo modelo que se ajuste a los nuevos datos, aplique las conversiones necesarias a los datos para su utilización. Guardar m_2 y b_2 obtenidos por la regresión para poder graficar el modelo $f_2(x) = m_2x + b_2$

Graifique $f_1(x)$ en el mismo plot para comparar ambos modelos

Tips:

- Reutilice el código ya implementado
- Averigüe cómo poner etiquetas a las funciones del plot
- Consulte en “San Google”

Resultado Esperado:



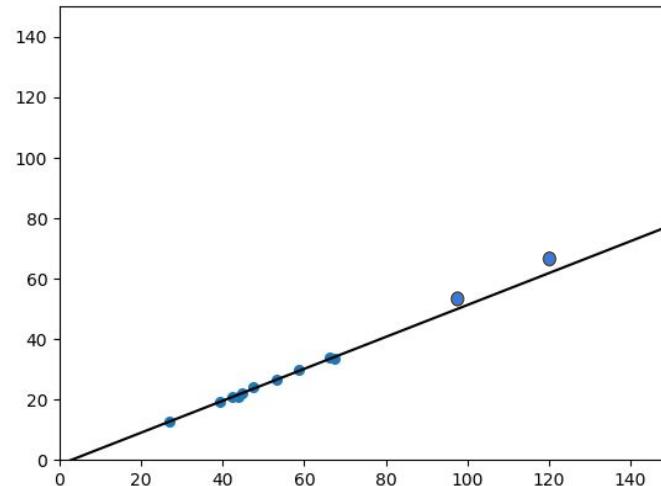
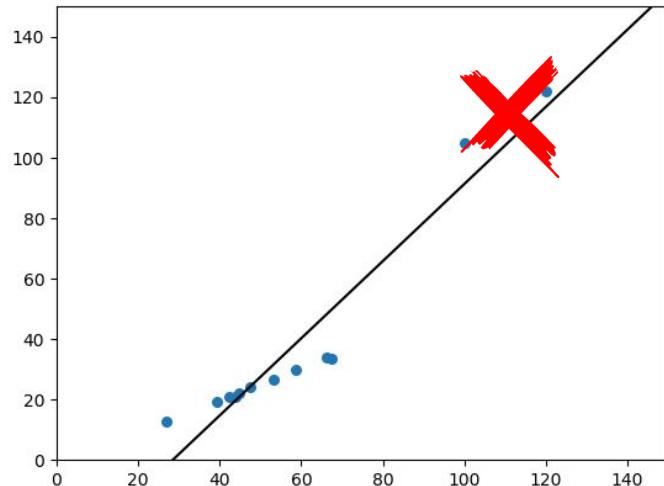
$$\text{Modelo 1: } y = 0.7256x + -3.260$$
$$\text{Modelo 2: } y = 0.507x + 5.122$$

Solución:

```
1 subsample_X = [2, 24, 7, 8, 1, 6, 10, 18, 20, 25, 16, 23, 22, 44, 34, 37, 27, 45, 38, 41, 46]
2 subsample_y = [10, 15, 9, 12, 6, 7, 9, 12, 13, 15, 14, 16, 16, 29, 20, 25, 20, 29, 27, 23, 31]
3 subsample_X = np.array(subsample_X).reshape(-1, 1)
4 subsample_y = np.array(subsample_y)
5
6 m2, b2, fx2 = obtener_modelo(subsample_X, subsample_y)
7
8 plt.figure(figsize=(5, 5))
9 plt.scatter(subsample_X, subsample_y, color='black')
10 plt.plot(fx1, "b", label="Modelo 1", linewidth=3)
11 plt.plot(fx2, "r", label="Modelo 2")
12 plt.legend(loc="lower right")
13 plt.xlim(0, 50) # Límite del eje x
14 plt.ylim(0,50) # Límite del eje y
15 plt.show()
16
17 print("Modelo 1: y = %.3fx + %.3f"%(m1, b1))
18 print("Modelo 2: y = %.3fx + %.3f"%(m2, b2))
```

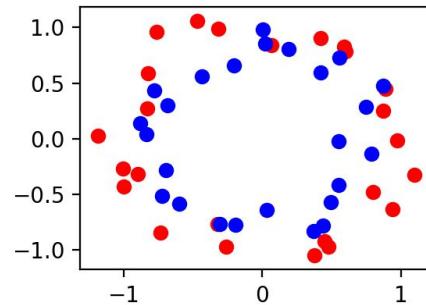
Mala calidad de los datos

¿Y si tenemos datos atípicos?



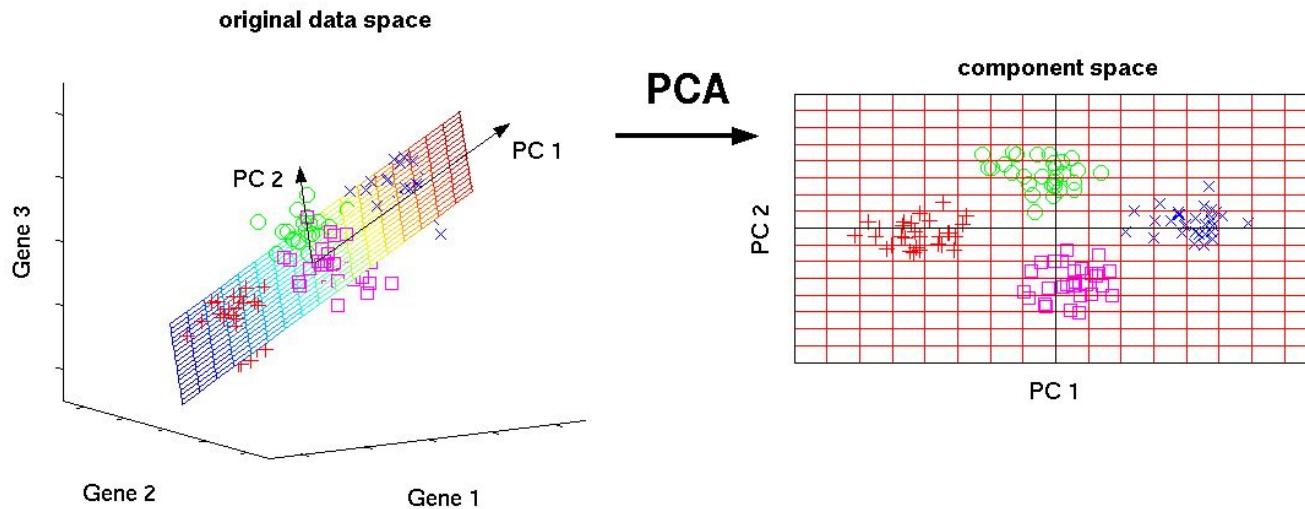
Podría ayudar eliminarlos o corregirlos manualmente

¿Y si tenemos ruido?



- Recolectar más datos puede ayudar
- Aplicar técnicas de preprocessamiento a los datos para eliminación o corrección

¿Y si tenemos rasgos irrelevantes?



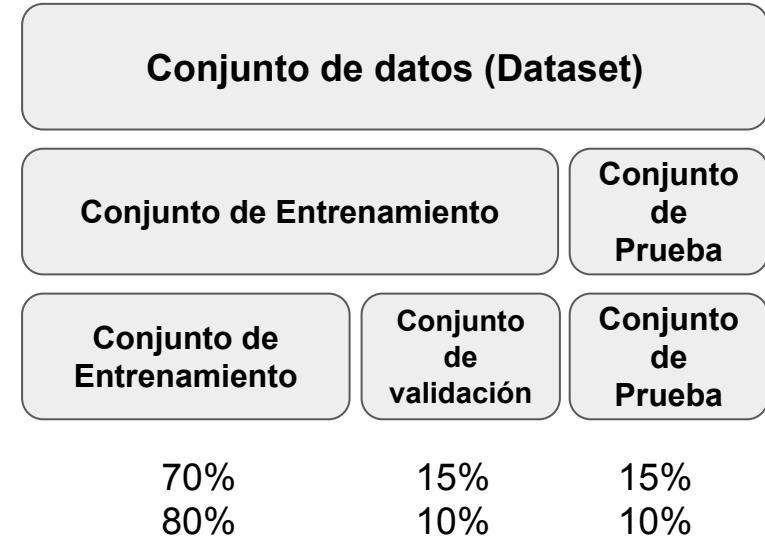
Podemos aplicar técnicas para quedarnos sólo con los rasgos relevantes

Conjunto de datos y cómo manejarlos

Representación de los datos

Los conjuntos de datos se separan en tres grupos:

- Conjunto de entrenamiento: Utilizados para determinar los parámetros del estimador (modelo).
- Conjunto de validación: Estima el error del modelo, sirve para ajustar hiper-parámetros
- Conjunto de prueba: Utilizado para medir que tan buena es la generalización del modelo.



Formas de aprender

Aprendizaje por lotes (Batch learning): El sistema aprende de los datos mediante lotes (grupos de muestras), hasta presentarle todo el conjunto de entrenamiento. El aprendizaje se realiza fuera de línea.

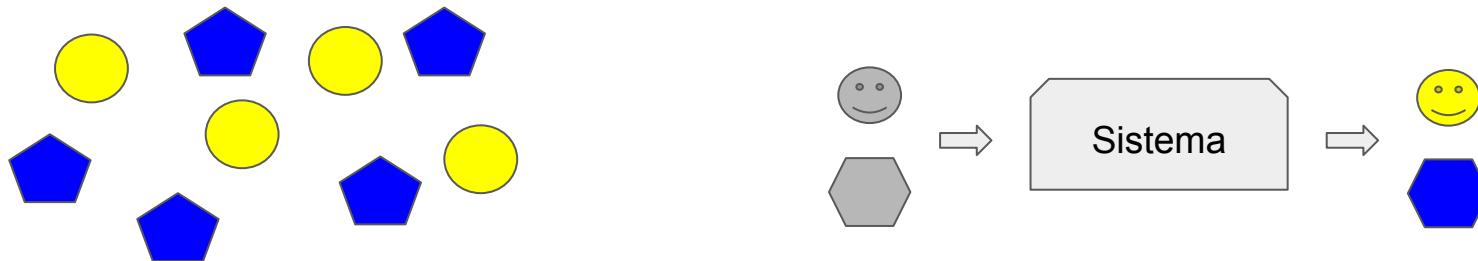


Aprendizaje en línea (Online learning): El sistema va aprendiendo incrementalmente al momento que se le presenta la información (en línea), pero suele hacerse cuando el sistema no está en producción (fuera de servicio), aprendizaje en línea puede ser un nombre confuso.

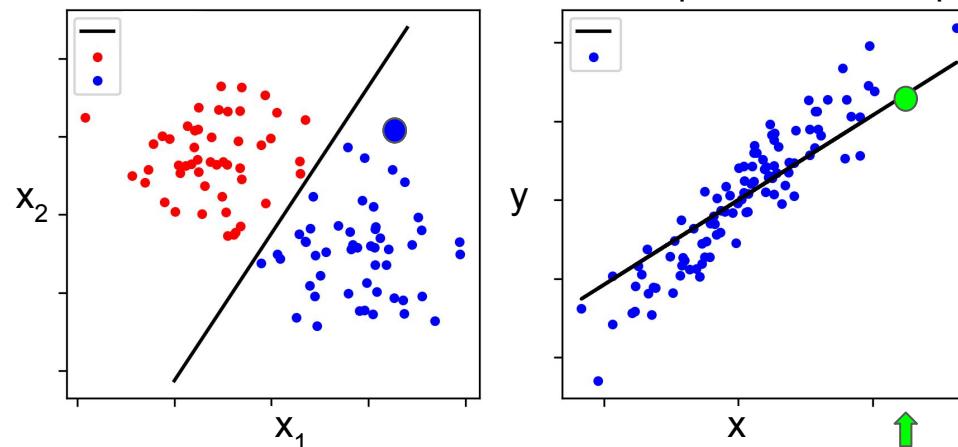


Formas de generalizar

Aprendizaje basado en instancia: El sistema memoriza los ejemplos y generaliza utilizando una medida de similitud



Aprendizaje basado en modelo: El sistema crea un modelo que es utilizado para hacer predicciones



Reto: En 10 minutos complete la **celda 6**, averigüe cómo cargar un archivo csv y abra el archivo “data.csv”. Lea cada fila y cree las listas para X_data y y_data. Asegúrese que los datos leídos estén en tipo de dato entero. Separe el conjunto de datos en entrenamiento y de prueba 70% y 30% respectivamente.

Tips: Use las bibliotecas csv, sklearn.model_selection

- Use las funciones ->
train_test_split , de la biblioteca sklearn.model_selection
reader, csv_file de la biblioteca csv
- Consulte en “San Google”

Datos:

```
[32 3 46 17 12 14 15 8 16 30 37 18 25 44 41 39 21 20 19 42 1 48 34 11 2 0 5 24 29 9 10 36 31 6 28 35 45 49 7 23 4 47 33 38 43 13 22 27 40 26]  
[24 8 31 11 10 12 12 12 14 21 25 12 15 29 23 24 16 13 14 30 6 31 20 11 10 8 6 15 17 11 9 24 18 7 20 21 29 30 9 16 8 28 19 27 25 9 16 20 26 22]
```

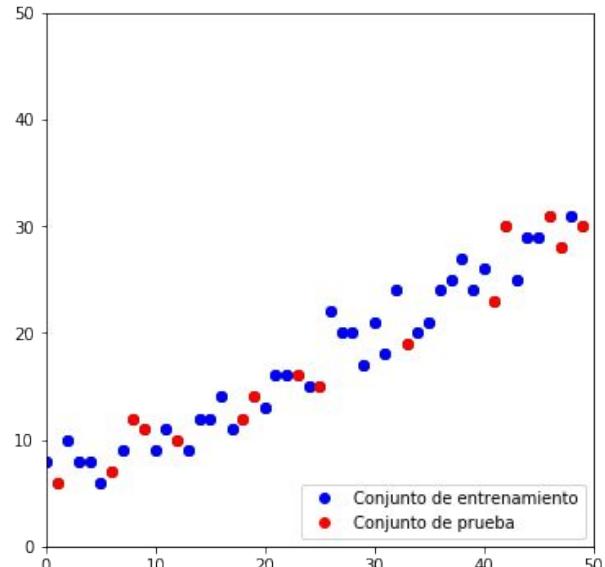
Entrenamiento:

```
[13 27 43 11 38 2 29 24 36 34 28 3 4 48 31 26 40 14 44 20 22 10 17 5 45 15 21 37 0 30 32 35 39 7 16]  
[ 9 20 25 11 27 10 17 15 24 20 20 8 8 31 18 22 26 12 29 13 16 9 11 6 29 12 16 25 8 21 24 21 24 9 14]
```

Prueba:

```
[49 18 25 42 47 33 9 19 46 41 12 1 23 8 6]  
[30 12 15 30 28 19 11 14 31 23 10 6 16 12 7]
```

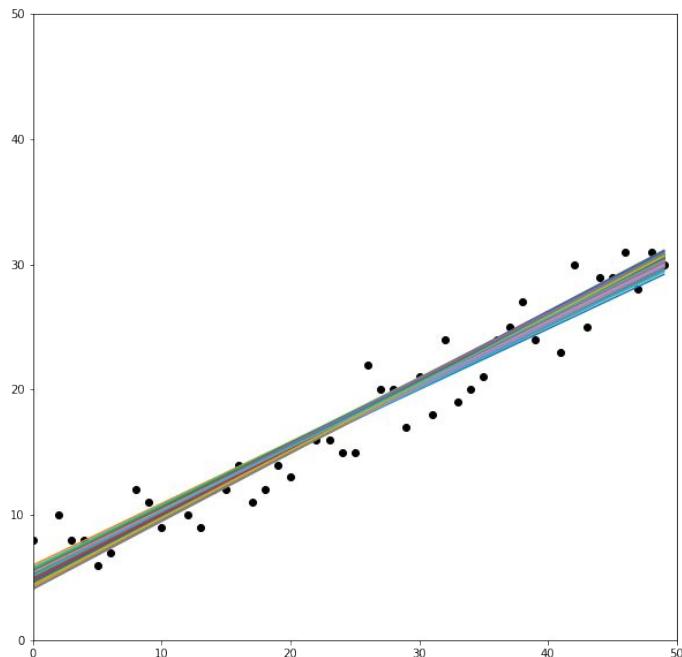
Resultado Esperado:



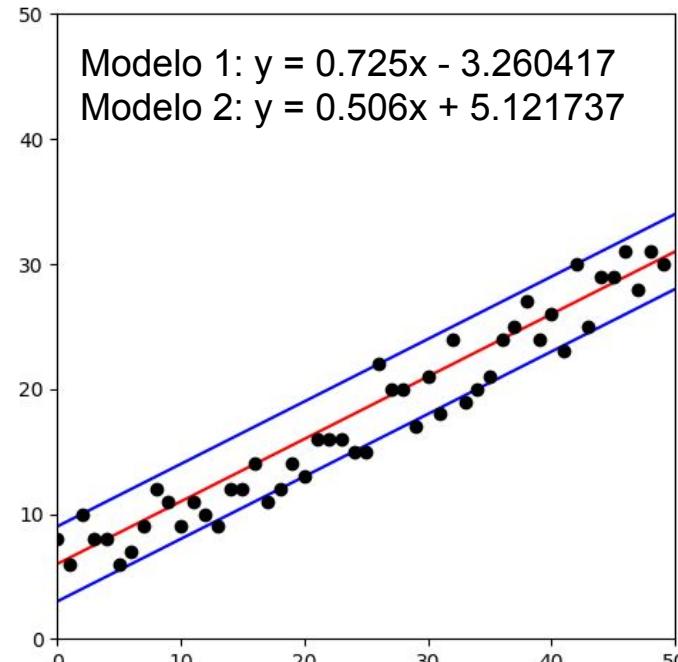
Solución:

```
1 from sklearn.model_selection import train_test_split
2 import csv
3
4 path = 'Datasets/data.csv'
5 diccionario = {}
6 X_data = []
7 y_data = []
8
9 with open(path, "r") as csv_file: # Abrir el archivo
10     csv_reader = csv.reader(csv_file)
11     next(csv_reader) # le la primer linea (header)
12     for fila in csv_reader: # por cada fila en el archivo
13         X_data.append(int(fila[0]))
14         y_data.append(int(fila[1]))
15
16
17 X_data = np.asarray(X_data)
18 y_data = np.asarray(y_data)
19 X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, train_size=0.7)
20
21 plt.figure(figsize=(6, 6))
22 plt.scatter(X_data, y_data, color='black')
23 plt.plot(X_train, y_train, "bo", label="Conjunto de entrenamiento")
24 plt.plot(X_test, y_test, "ro", label="Conjunto de prueba")
25 plt.legend(loc="lower right")
26 plt.xlim(0, 50) # Límite del eje x
27 plt.ylim(0,50) # Límite del eje y
28 plt.show()
29
30 print("Datos:")
31 print(X_data)
32 print(y_data)
33
34 print("\nEntrenamiento:")
35 print(X_train)
36 print(y_train)
37
38 print("\nPrueba:")
39 print(X_test)
40 print(y_test)
```

Ejecute la última **celda**, observe y compare el comportamiento del predictor acorde al modelo real que se intenta aprender

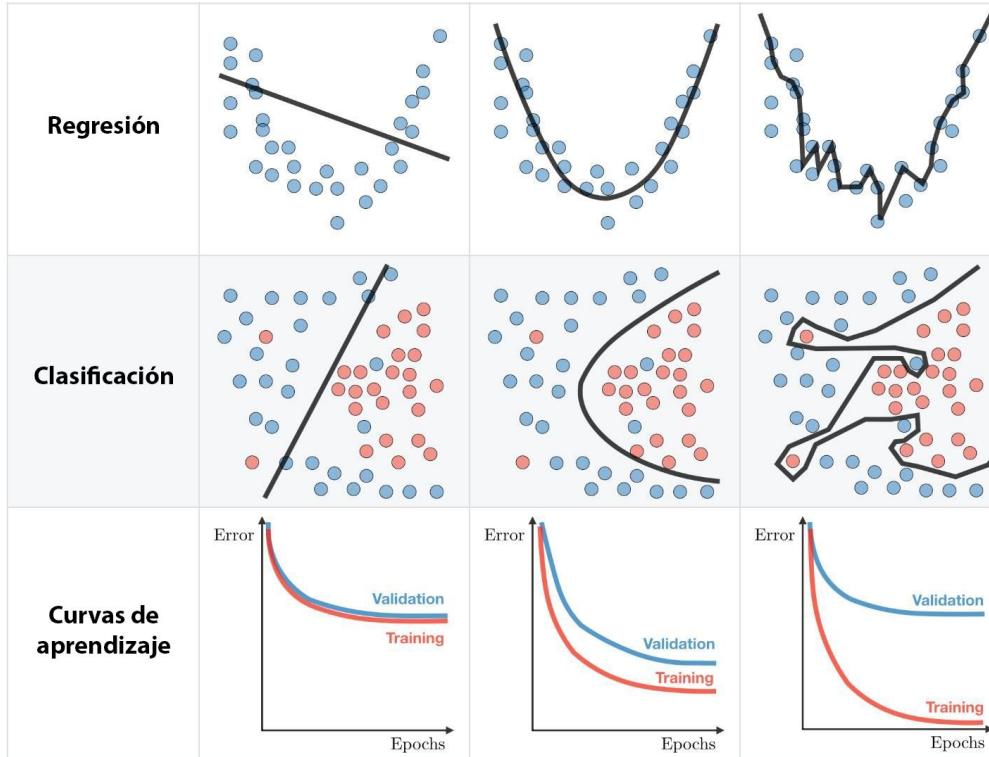


Modelo lineal 100 hipótesis



Función generadora $y = 0.5m + 6$
Con ruido aleatorio uniforme de [-3, +3]

Sobreajuste y subajuste



Sklearn PolinomialFeatures: Genera una matriz o vector de todas las combinaciones de rasgos con grado menor igual al grado especificado.

Ejemplo de su cálculo para grado 2 y grado 3:

$$p(a, b, \text{grad} = 2) = (1, a, b, a^2, ab, b^2)$$

$$p(a, b, \text{grad} = 3) = (1, a, b, a^2, ab, b^2, a^3, a^2b, ab^2, b^3)$$

$$p(3, 4, \text{grad} = 2) =$$

$$p(5, 6, \text{grad} = 2) =$$

Resultado Esperado:

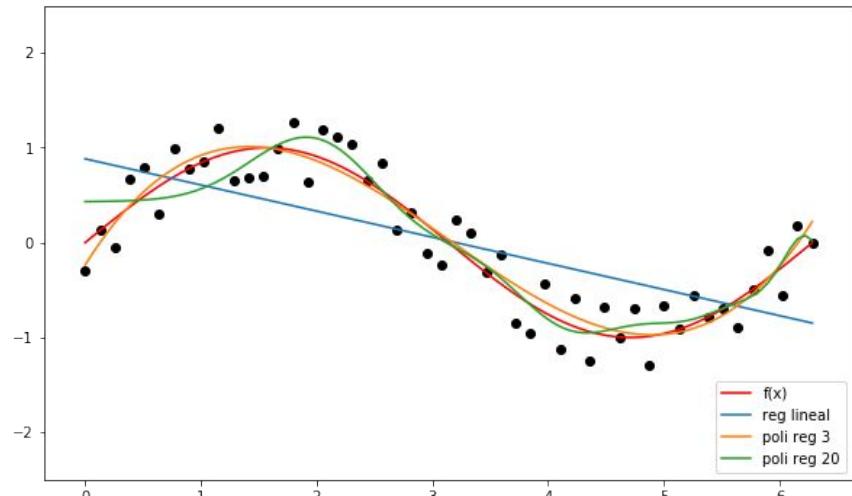
Reto: En 15 minutos, En la **celda 1** identifique los datos de entrenamiento y prepáralos para su utilización.

En la **celda 3**, grafique el modelo lineal y el polinomial, pruebe valores diferentes de n (entre 2 y 50)

Tips: Use las bibliotecas: sklearn, matplotlib y numpy

- Use las funciones ->
LinearRegression, PolynomialFeatures, np.linspace,
plt.plot, plt.scatter
- Consulte en “San Google”

x_train shape (50,)
X_train shape (50, 1)



Solución:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.linear_model import LinearRegression
4 from sklearn.preprocessing import PolynomialFeatures
5
6 # Función original
7 x = np.linspace(0, 2*np.pi, 100)
8 y = np.sin(x)
9 x = x[:, np.newaxis]
10
11 # Datos de la función
12 np.random.seed(2) # Semilla para replicar los resultados
13 x_train = np.linspace(0, 2*np.pi, 50)
14 y_train = np.sin(x_train) + np.random.randint(-1, 2, 50) * 0.3
15 X_train = x_train[:, np.newaxis]
16
17 print("x_train shape", x_train.shape)
18 print("X_train shape", X_train.shape)
```

Solución:

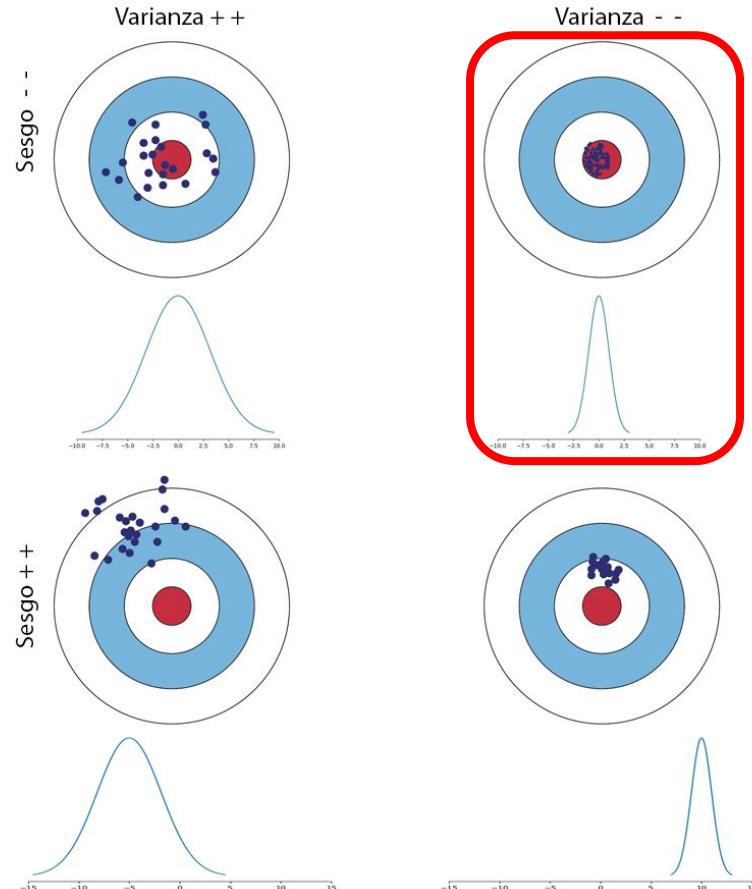
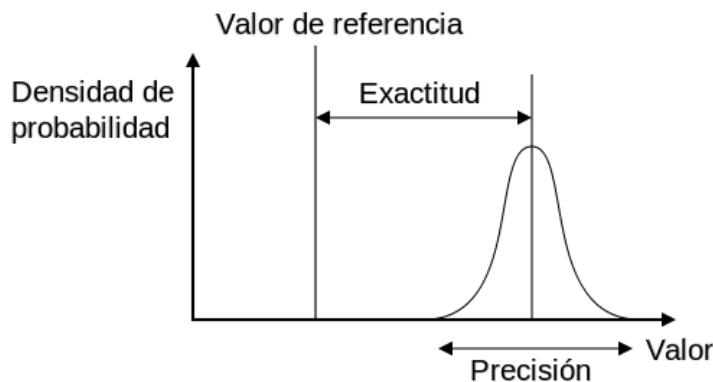
```
1 # Modelo lineal
2 lin_r = LinearRegression()
3 lin_r.fit(X_train, y_train) # aprende de las muestras
4 y_lin = lin_r.predict(x) # ajusta a todos los valores de x
5
6 # Modelo polinomial
7 y_pol3 = polinom_n(3, X_train, y_train, x)
8 # Probar valores de 2 a 50
9 n = 20
10 y_poln = polinom_n(n, X_train, y_train, x)
11
12
13 # Plot de la figura
14 plt.figure(figsize=(10, 6))
15 plt.plot(x, y, "r", label="f(x)") # Grafica la función original
16 plt.scatter(X_train, y_train, color='black') # Grafica las muestras
17 plt.plot(x, y_lin, label="reg lineal") # Grafica la regresión lineal
18 plt.plot(x, y_pol3, label="poli reg 3") # Grafica la regresión polinomial
19 plt.plot(x, y_poln, label="poli reg %d"%(n))
20 plt.legend(loc="lower right")
21 plt.ylim(-2.5, 2.5)
22 plt.show()
```

¿Qué es sesgo (bias)?

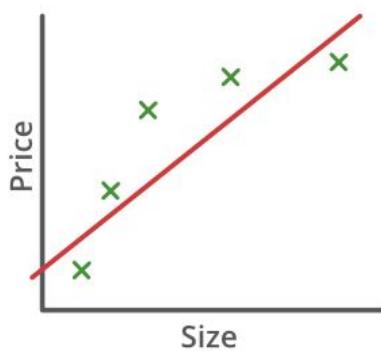
Lo podemos entender como la **exactitud** del estimador (accuracy)

¿Qué es varianza (variance)?

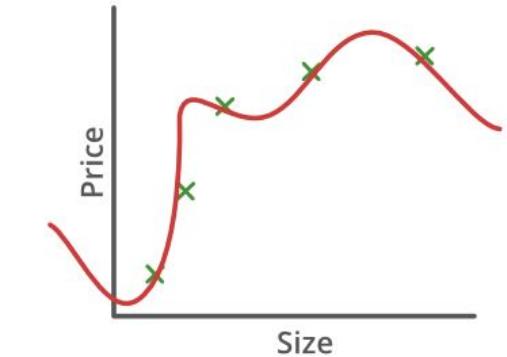
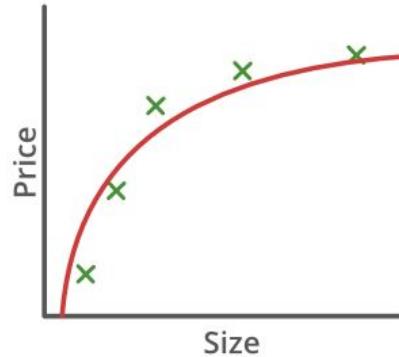
La **precisión** del estimador (precision)



Balance sesgo - varianza (trade-off)

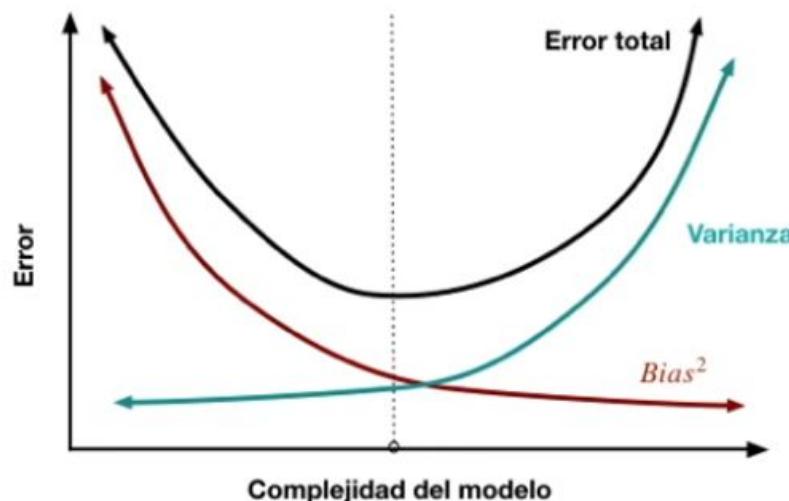


High bias (underfit)



High variance (overfit)

Balance sesgo - varianza (trade-off)



Un buen modelo se encuentra en el punto de equilibrio entre el sesgo y la varianza de tal manera que minimice el error total

Sesgo Alto	Sesgo Bajo
Menos flexibilidad para ajustarse a los datos	Más flexibilidad para ajustarse a los datos
Baja Varianza	Alta Varianza
Baja sensibilidad de la estimación a los cambios en la entrada	Alta sensibilidad de la estimación a los cambios en la entrada

Error irreducible: por variables desconocidas, características desconocidas, problema mal enmarcado (ruido irreducible en los datos)

Explorando un conjunto de datos: Iris Dataset

Objetivo:

Analizar un archivo con múltiples características, que se entienda cómo se manipula un conjunto de datos de sklearn.

Ir familiarizándose con multidimensionalidad 3D, 4D...



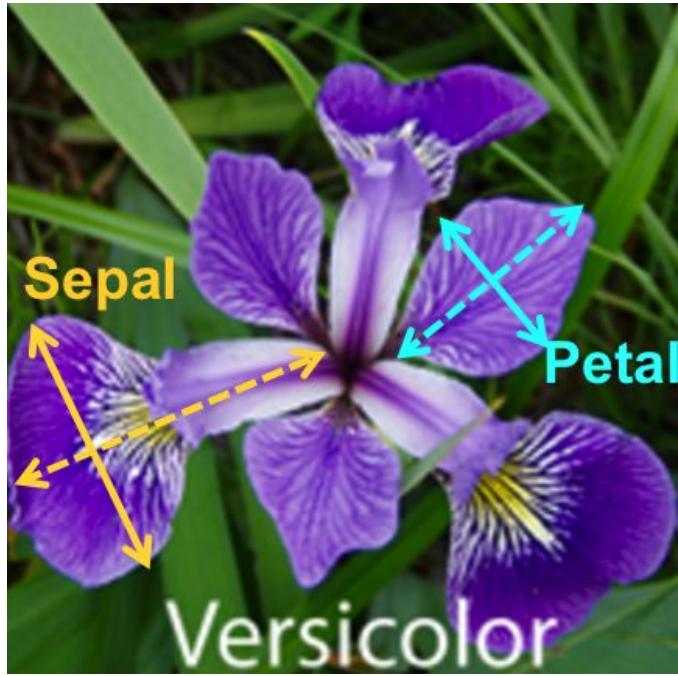
Iris setosa



Iris versicolor



Iris virginica



Ver el conjunto de datos:

https://en.wikipedia.org/wiki/Iris_flower_data_set

<https://gist.github.com/netj/8836201>

Reto: En 5 minutos, cargue el conjunto de datos iris de sklearn en la **celda 1** e imprima el objeto iris, entienda cómo está conformado. Con un ciclo for obtenga las claves del diccionario iris.

Escuche la explicación del instructor sobre la **celda 2 y 3**, analice las diferentes propiedades del objeto iris

Tips: Use la biblioteca: sklearn.datasets

- Use las funciones ->
load_iris
- Consulte en “San Google”

Resultado Esperado:

```
{'data': array([[5.1, 3.5, 1.4, 0.2],  
[4.9, 3. , 1.4, 0.2],  
[4.7, 3.2, 1.3, 0.2],  
[4.6, 3.1, 1.5, 0.2],  
...  
[5.9, 3. , 5.1, 1.8]]), 'target': array([0,  
0, 0, 0, 0,  
...  
'target_names': array(['setosa',  
'versicolor', 'virginica'],  
...  
sklearn/datasets/data/iris.csv'}
```

data
target
target_names
DESCR
feature_names
filename

03_iris_dataset_reto.ipynb

Solución:

```
1 # Importar módulos
2 from sklearn import datasets
3
4 # Carga el conjunto de datos
5 iris = datasets.load_iris()
6
7 # Acceso a las propiedades con el operador "."
8 X = iris.data
9 y = iris.target
10
11 print(iris)
12
13 # Imprime las llaves del diccionario
14 for dato in iris:
15     print(dato) #Es un string
```

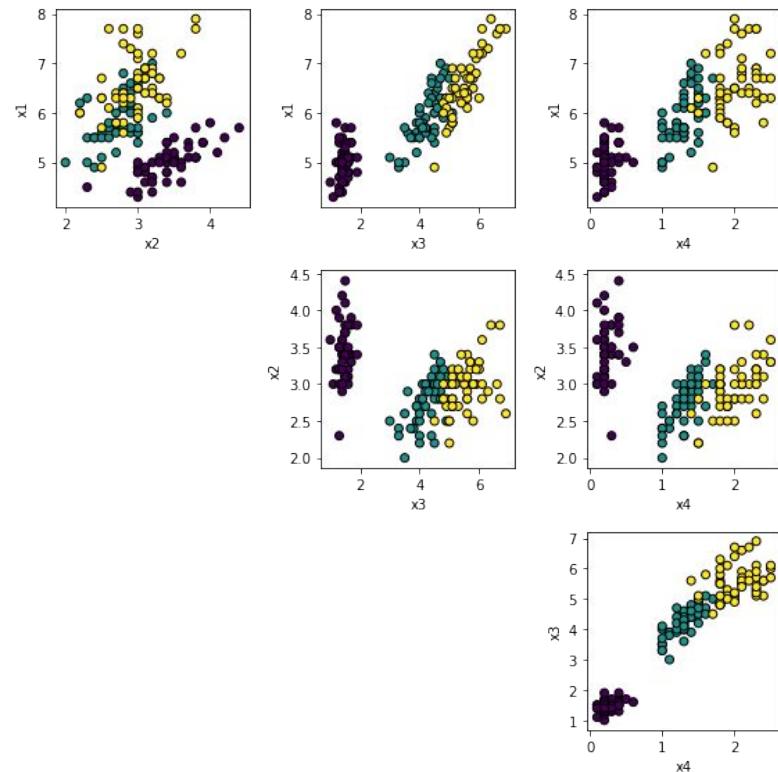
Reto: En 10 minutos complete la **celda 4**, obtenga los 4 vectores de características y guardelos en variables. Imprima las diferentes combinaciones de características para hacer plots 2D entre ellas x_1 con x_2 , x_2 con x_3 y así sucesivamente.

Utilice subplots para dar forma de matriz diagonal a la salida

Tips: Use la biblioteca: `matplotlib.pyplot`

- Use las funciones ->
`scatter`, `xlabel`, `ylabel`, `tight_layout`, `show`, `subplot`
- Consulte en “San Google”

Resultado Esperado:



Solución:

```
1 import matplotlib.pyplot as plt
2
3 x1_sepal_length = iris.data[:, 0]
4 x2_sepal_width = iris.data[:, 1]
5 x3_petal_length = iris.data[:, 2]
6 x4_petal_width = iris.data[:, 3]
7
8
9 plt.figure(figsize=(8, 8))
10 plt.subplot(3, 3, 1)
11 plt.scatter(x2_sepal_width, x1_sepal_length, edgecolor='k', c=y)
12 plt.xlabel("x2")
13 plt.ylabel("x1")
14 plt.subplot(3, 3, 2)
15 plt.scatter(x3_petal_length, x1_sepal_length, edgecolor='k', c=y)
16 plt.xlabel("x3")
17 plt.ylabel("x1")
18 plt.subplot(3, 3, 3)
19 plt.scatter(x4_petal_width, x1_sepal_length, edgecolor='k', c=y)
20 plt.xlabel("x4")
21 plt.ylabel("x1")
22 plt.subplot(3, 3, 5)
23 plt.scatter(x3_petal_length, x2_sepal_width, edgecolor='k', c=y)
24 plt.xlabel("x3")
25 plt.ylabel("x2")
```

Solución:

```
26  
27 plt.subplot(3, 3, 6)  
28 plt.scatter(x4_petal_width, x2_sepal_width, edgecolor='k', c=y)  
29 plt.xlabel("x4")  
30 plt.ylabel("x2")  
31 plt.subplot(3, 3, 9)  
32 plt.scatter(x4_petal_width, x3_petal_length, edgecolor='k', c=y)  
33 plt.xlabel("x4")  
34 plt.ylabel("x3")  
35 plt.tight_layout() # Ajusta el plot para que no se ensime  
36 plt.show() # Instrucción para mostrar el plot
```

Evaluar el desempeño de los modelos



ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

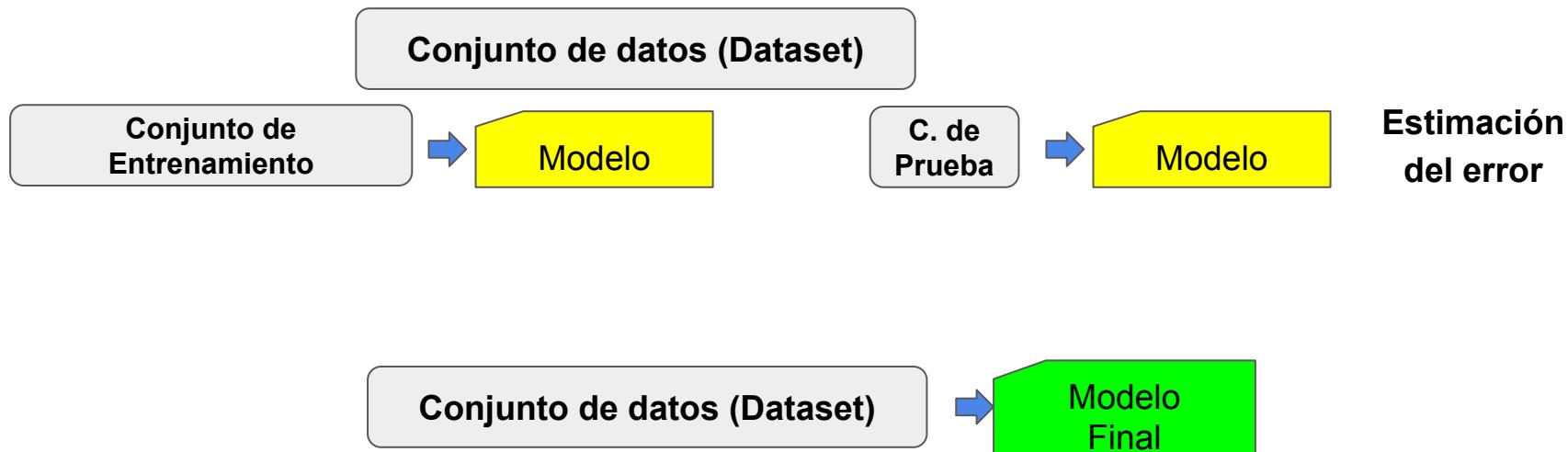
Proceso para la selección de modelos

- Estimar el desempeño del modelo en datos no vistos por el modelo
- Aumentar el desempeño ajustando el algoritmo ([hiper-parámetros](#))
- Comparar diferentes algoritmos e identificar el modelo que mejor resuelve el problema

Formas de evaluar el desempeño de los modelos

Validación sostenida (holdout):

1. Dividir el conjunto en dos partes; entrenamiento y prueba
2. Utilizar solo el conjunto de entrenamiento para ajustar el modelo e hiper-parámetros
3. Usar el conjunto de prueba para estimar el error
4. Entrenar el modelo final con todo el conjunto



Reto: En 5 minutos complete la **celda 1**, separe el conjunto de datos para entrenamiento y prueba con una proporción 60-40

Entrenar un clasificador SVC y mostrar su desempeño en ambos conjuntos

Resultado Esperado:

Celda 1

Acc entrenamiento: 98.889 %
Acc estimado: 96.667 %

Tips: Use la biblioteca: sklearn

- Use las funciones ->
Train_test_split, datasets, svm
- Consulte en “San Google”

04_Evaluacion_validacion_reto.ipynb

Sección “Holdout”

57

Solución:

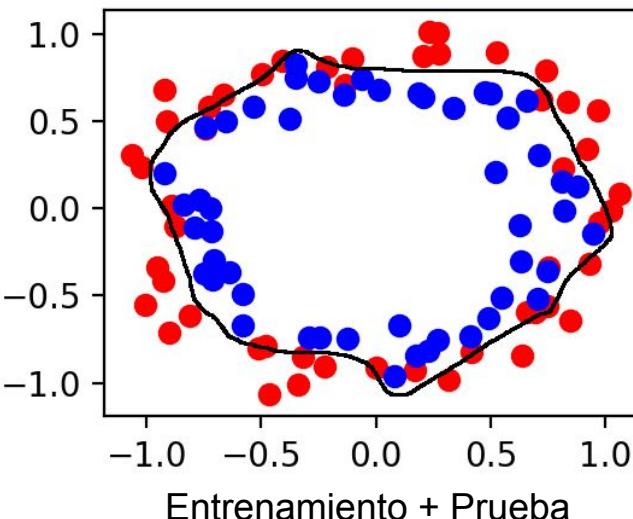
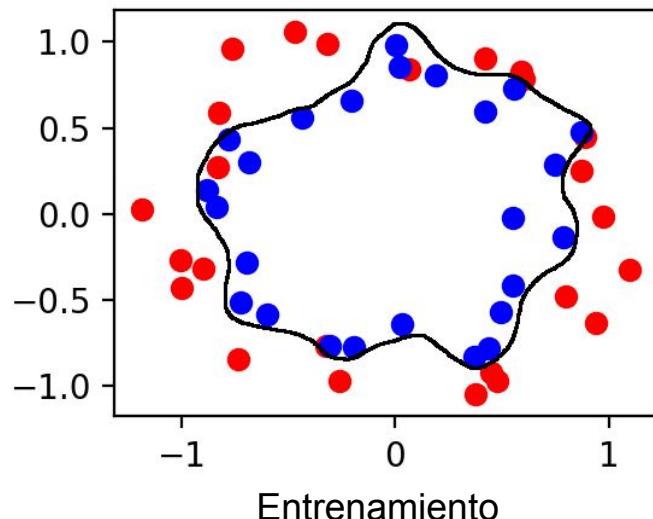
```
1 from sklearn.model_selection import train_test_split
2 from sklearn import datasets
3 from sklearn import svm
4
5 iris = datasets.load_iris()
6 X_train, X_test, y_train, y_test = \
    train_test_split(iris.data, iris.target, test_size=0.4, random_state=0)
7
8
9 svm_clasif = svm.SVC(kernel='linear', C=1)
10 svm_clasif.fit(X_train, y_train)
11 print("Acc entrenamiento: {:.3f} %".format(svm_clasif.score(X_train, y_train)*100))
12 print("Acc estimado: {:.3f} %".format(svm_clasif.score(X_test, y_test)*100))
```

Validación sostenida (holdout)

Problemática:

Cuando se tiene un conjunto de datos reducido, este puede generar sobreajuste.

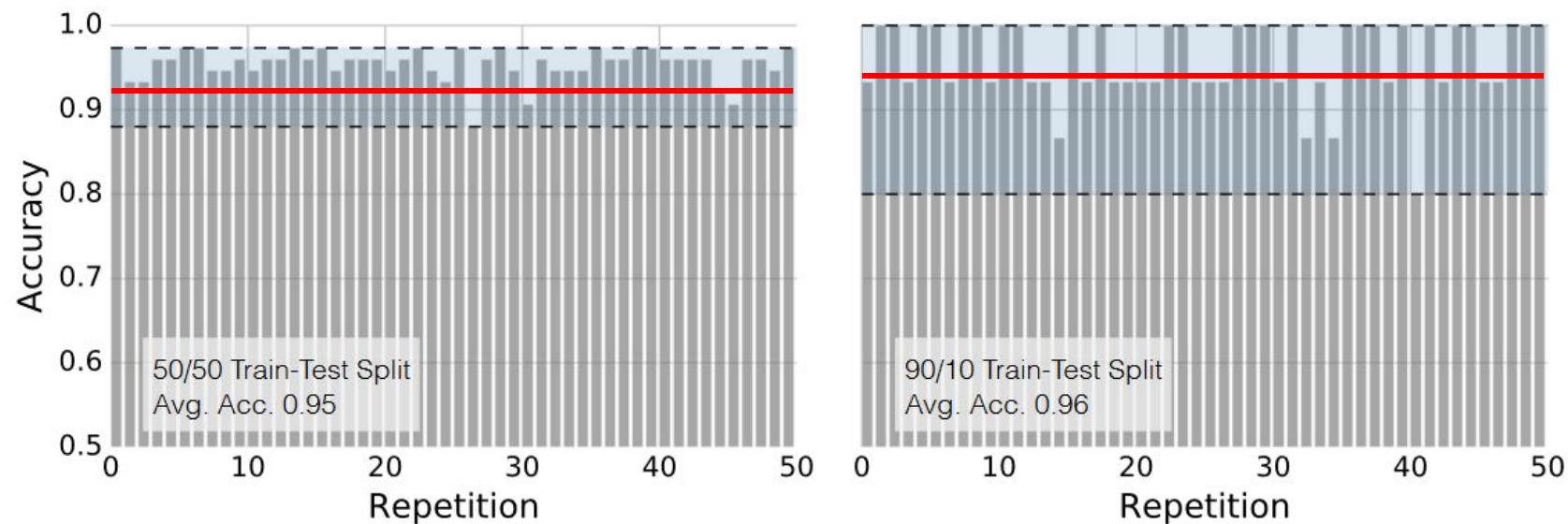
Aun si se entrena con todos los datos, el modelo seguirá teniendo sobreajuste.



Formas de evaluar el desempeño de los modelos

Validación sostenida repetida (holdout):

Si lo repetimos múltiples veces, nos da información de la estabilidad del modelo



Reto: En 15 minutos complete la **celda 2**, probar con diferentes porcentajes para el conjunto de prueba:
50, 40, 30, 20 y 10

Repita el siguiente proceso para cada uno de los porcentajes:

Realice 50 entrenamientos, imprima solamente la media y desviación estándar del acumulado de los 50 entrenamientos

Tips: Use la biblioteca: numpy

- Use las funciones ->
`np.mean, np.std`
- Consulte en “San Google”

Resultado Esperado:

Celda 2

Holdout 0:

Proporción: 0.5 - 0.5

media: 0.973 Desviación std: 0.017

Holdout 1:

Proporción: 0.6 - 0.4

media: 0.973 Desviación std: 0.020

Holdout 2:

Proporción: 0.7 - 0.3

media: 0.975 Desviación std: 0.020

Holdout 3:

Proporción: 0.8 - 0.2

media: 0.977 Desviación std: 0.028

Holdout 4:

Proporción: 0.9 - 0.1

media: 0.980 Desviación std: 0.033

04_Evaluacion_validacion_reto.ipynb

Sección “Repeated Holdout”

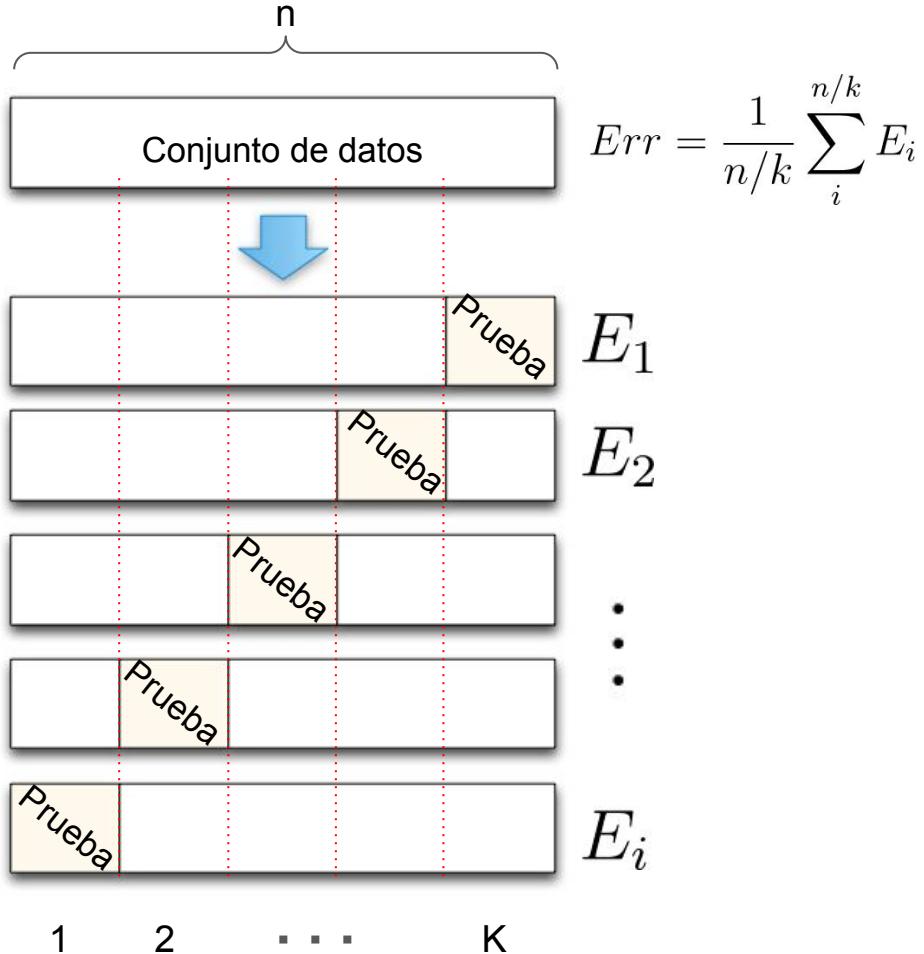
Repeated Holdout

```
1 from sklearn.model_selection import train_test_split
2 from sklearn import datasets
3 from sklearn import svm
4 import numpy as np
5
6 iris = datasets.load_iris()
7 prop_splits = [0.5, 0.4, 0.3, 0.2, 0.1]
8 repeticiones = 50
9 for i, proporcion in enumerate(prop_splits):
10     print("Holdout %d:" % (i))
11     acc = []
12     for j in range(repeticiones):
13         # Random splits
14         X_train, X_test, y_train, y_test = \
15             train_test_split(iris.data, iris.target, test_size=proporcion)
16
17         svm_clasif = svm.SVC(kernel='linear', C=1)
18         svm_clasif.fit(X_train, y_train)
19         acc.append(svm_clasif.score(X_test, y_test))
20
21     print("Proporción: %.1f - %.1f" % (1-proporcion, proporcion))
22     acc = np.array(acc)
23     print("media: %.3f    Desviación std: %.3f" % (acc.mean(), acc.std()))
24     print("")
```

Validación cruzada (k-Fold): $k > 1$

El conjunto se divide aleatoriamente en k grupos, donde uno es usado para prueba y el resto para entrenamiento

se hace repetidamente hasta que se haya hecho prueba sobre cada grupo.



Reto: En 10 minutos complete las **celdas 3 y 4**

Celda 3: Separe el conjunto X en 4 k-fold

Celda 4:

Aplique validación cruzada con un k-fold de 5 sobre el conjunto de datos iris. Calcule la media y desviación estándar

Tips: Use la biblioteca: `sklearn.model_selection`

- Use las funciones ->
`KFold, cross_val_score`
- Reuse conceptos previos si es necesario
- Consulte en “San Google”

Resultado Esperado:

Celda 3

```
TRAIN: [3 4 5 6 7 8] TEST: [[1 2]]  
TRAIN: [1 2 5 6 7 8] TEST: [[3 4]]  
TRAIN: [1 2 3 4 7 8] TEST: [[5 6]]  
TRAIN: [1 2 3 4 5 6] TEST: [[7 8]]
```

Celda 4

```
cros_val: [0.96666667 1.           0.96666667 0.96666667 1.  
]  
media: 0.980   Desviación std: 0.016
```

04_Evaluacion_validacion_reto.ipynb

Sección “K-fold validation”

K-fold validation

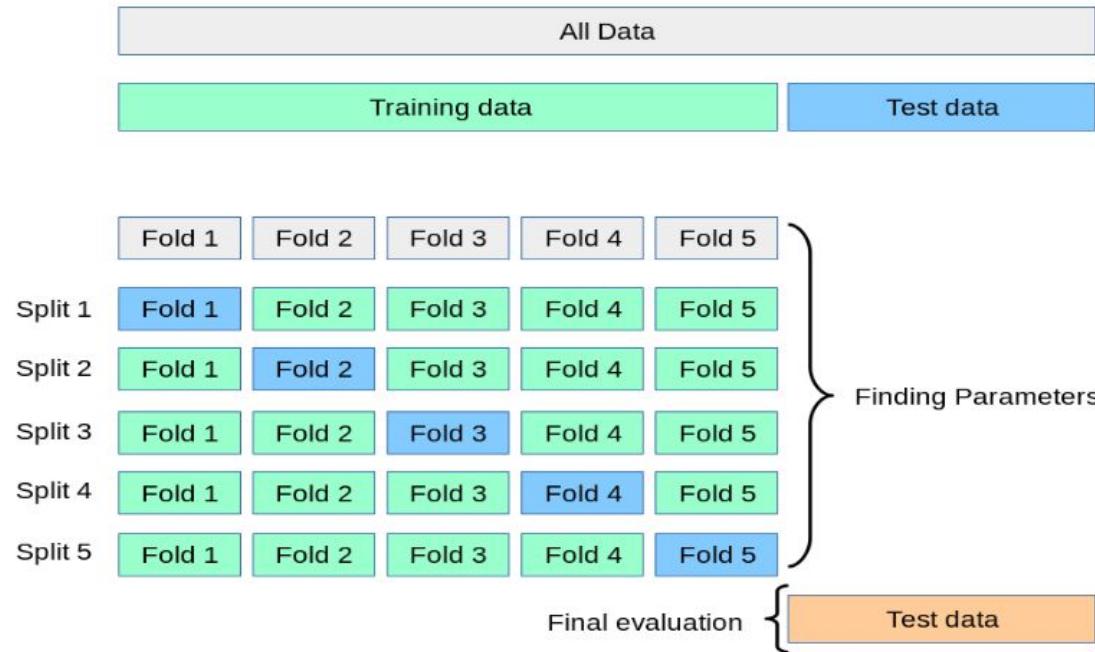
```
1 import numpy as np
2 from sklearn.model_selection import KFold
3
4 X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
5 kf = KFold(n_splits=4)
6
7 for train_index, test_index in kf.split(X):
8     print("TRAIN:", X[train_index].reshape(-1), "TEST:", X[test_index])
```

```
1 from sklearn.model_selection import cross_val_score
2
3 svm_clasif = svm.SVC(kernel='linear', C=1)
4 validacion_cruzada = cross_val_score(svm_clasif, iris.data, iris.target, cv=5)
5 print("cros_val:", validacion_cruzada)
6 print("media: %.3f Desviación std: %.3f" % (validacion_cruzada.mean(), validacion_cruzada.std()))
```

Si K = número de datos

Versión extrema donde el grupo es de tamaño 1, también se le conoce como:

Dejar uno fuera, Validación cruzada (**Leave-one-out cross validation**)



Reto: En 10 minutos complete las **celdas 5 y 6**

Celda 5: Aplique leave one out sobre el conjunto X

Celda 6: Aplique validación cruzada como en el reto anterior, en esta ocasión el k-fold es igual que el número muestras por clase

Tips: Use la biblioteca: `sklearn.model_selection`

- Use las funciones ->
`LeaveOneOut`, `cross_val_score`, `shape`
- Reuse conceptos previos si es necesario
- Consulte en “San Google”

Resultado Esperado:

Celda 5

```
TRAIN: [1 2 3] TEST: [0]
TRAIN: [0 2 3] TEST: [1]
TRAIN: [0 1 3] TEST: [2]
TRAIN: [0 1 2] TEST: [3]
```

Celda 6

Iris, muestras por clase:

```
2      50
1      50
0      50
dtype: int64
```

media: 0.973 Desviación std: 0.090

04_Evaluacion_validacion_reto.ipynb

Sección “Leave One Out”

Leave One Out

```
1 from sklearn.model_selection import LeaveOneOut  
2  
3 X = [1, 2, 3, 4]  
4 loo = LeaveOneOut()  
5 for train, test in loo.split(X):  
6     print("TRAIN:%s TEST:%s" % (train, test))
```

```
1 from sklearn.model_selection import cross_val_score  
2 import pandas as pd  
3  
4 pd_data = pd.Index(iris.target)  
5 muestras_clase = pd_data.value_counts()[0]  
6 print("Iris, muestras por clase:")  
7 print(pd_data.value_counts())  
8  
9 print()  
10 svm_clasif = svm.SVC(kernel='linear', C=1)  
11 validacion_cruzada = cross_val_score(svm_clasif, iris.data, iris.target, cv=muestras_clase)  
12 print("leave one out shape:", validacion_cruzada.shape)  
13 print(validacion_cruzada)  
14 print("media: %.3f Desviación std: %.3f" % (validacion_cruzada.mean(), validacion_cruzada.std()))
```

Formas de evaluar el desempeño de los modelos

Problema:

Medir múltiples veces el **error** en el conjunto de prueba, adaptando hiper-parámetros. Se termina creando un modelo ajustado a **ese conjunto de prueba**

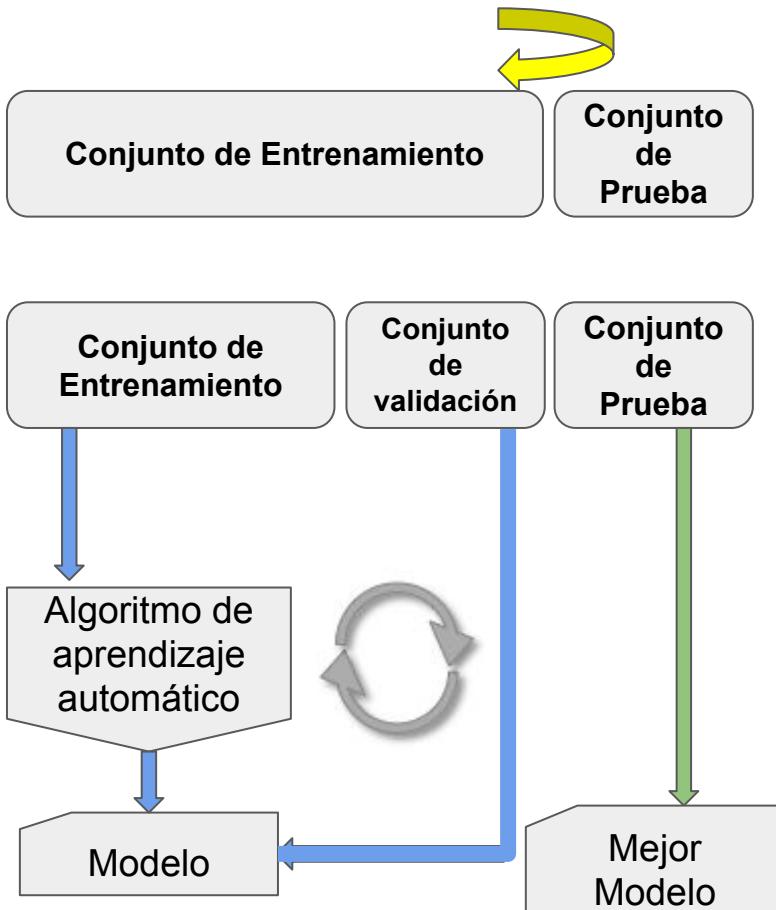
Solución:

Conjunto de validación y de prueba:

Utilizar un conjunto llamado “validación”

Ajustar hiper-parámetros, lograr un buen desempeño del modelo en el conjunto “validación”

Tomar el mejor modelo y **estimar el error** en el conjunto de prueba



Reto: En 5 minutos complete la **celda 7**, separe correctamente los conjuntos entrenamiento, validación y prueba con proporción 60-20-20 por ciento respectivamente

Modifique el parámetro C del clasificador de 0.1 a 1 identifique cuál es el mejor valor

Finalmente evalúe el modelo en el conjunto de prueba solo con el mejor parámetro encontrado en la **celda 8**

Tips: Use la biblioteca: `sklearn.model_selection`

- Use las funciones ->
`train_test_split`
- Consulte en “San Google”

Resultado Esperado:

Celda 7

Entrenamiento: 90 muestras
Acc: 0.978

Validación: 30 muestras
Val_acc: 0.967

Celda 8

Prueba: 30 muestras
test_acc: 0.967

Train, validation, test

```

1 from sklearn.model_selection import train_test_split
2 from sklearn import datasets
3 from sklearn import svm
4
5 iris = datasets.load_iris()
6 X_train, X_test, y_train, y_test = \
7     train_test_split(iris.data, iris.target, test_size=0.4, random_state=0)
8 X_val, X_test, y_val, y_test = \
9     train_test_split(X_test, y_test, test_size=0.5, random_state=0)
10
11 ## Variar el parametro C
12 svm_clasif = svm.SVC(kernel='linear', C=0.3)
13 svm_clasif.fit(X_train, y_train)
14 print("Entrenamiento: %d muestras"% len(y_train))
15 print("Acc:", svm_clasif.score(X_train, y_train))
16 print("")
17 print("Validación: %d muestras"% len(y_val))
18 print("Val_acc:", svm_clasif.score(X_val, y_val))
19 print("")
```

```

1 # por último, evaluar el mejor modelo en test
2 print("Prueba: %d muestras"% len(y_test))
3 print("test_acc:", svm_clasif.score(X_test, y_test))
```

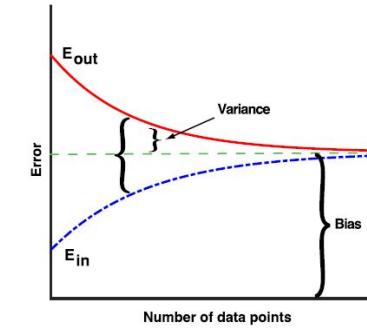
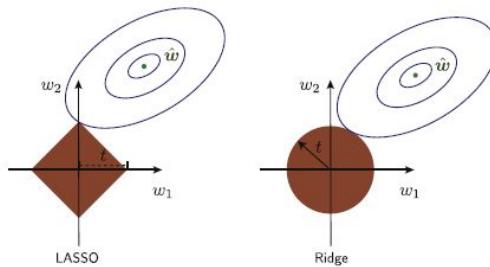
Resumen

Machine Learning es útil para:

- En aprendizaje automático existen tres principales ramas: supervisado, no supervisado y por refuerzo
- Los principales tipos de problemas son: regresión, clasificación y agrupamiento
- En aprendizaje automático es importante tratar los datos: eliminar ruido, corregir errores, normalizar, quitar rasgos irrelevantes y más.
- Se busca que los modelos tengan un balance entre sesgo y varianza
- Existen métodos para validar el aprendizaje de los modelos: holdout, k-fold, leave one out, train-validation-test

A high-bias, low-variance introduction to Machine Learning for physicists

P. Mehta, M. Bükov, C.-H. Wang et al. / Physics Reports 810 (2019) 1–124



Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning

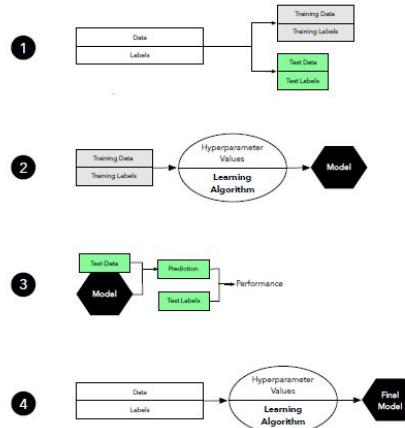


Figure 2: Visual summary of the holdout validation method.

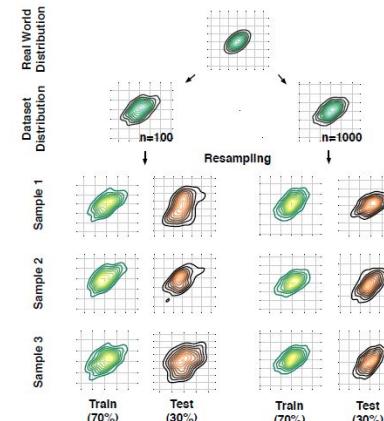


Figure 5: Repeated subsampling from a two-dimensional Gaussian distribution.

Conclusión Final

- Aplicar técnicas de Aprendizaje Automático para profundizar en grandes cantidades de datos puede ayudar a descubrir patrones que no son inmediatamente visibles. Esto se llama minería de datos.
- Los tipos de datos con los que se trabaja y los datos que se recopilan, dependen del problema que esté tratando de resolver:
 - Sonido (reconocimiento de voz)
 - Texto (Clasificación de comentarios)
 - Imágenes (visión artificial)
 - Series temporales (datos del sensor, actividad web)
 - Video (detección de movimiento)

Libros del curso

- Aurélien Géron, Hands-On Machine Learning with Scikit-Learn & Tensorflow, O'Reilly, 1st edition, 2017. (**libro práctico**)
- Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin, Learning from data: a short course, AMLbook.com, 2012. (**libro teórico**)
- Sebastian Raschka, Python Machine Learning, PACKT, 2015. (**libro práctico complementario**)

Los pueden descargar desde la página rusa o los pueden comprar en Amazon.

Glosario

Aprendizaje automático (Machine Learning): es una área del conocimiento que busca crear modelos matemáticos y algoritmos efectivos que aprendan a realizar tareas a partir la experiencia representada por datos numéricos. Dichas tareas no deben ser programadas directamente.

Aprendizaje supervisado (Supervised Learning): el modelo matemático debe aprender a asociar las entradas con las salidas. Los ejemplos (entradas, salidas) son dados por un ser humano o algún medio externo, por eso se llama supervisado.

Aprendizaje NO supervisado (Unsupervised Learning): el modelo matemático debe aprender a agrupar los datos de entrada o a representarlos basado en similitudes y diferencias entre ellos. Es deseable que los datos sean clasificados o representados en grupos con el menor traslape y en el menor número de grupos posible. No se conocen las clases ni los grupos al inicio.

Aprendizaje por refuerzo (Reinforcement Learning): el modelo matemático debe aprender a elegir las acciones más convenientes de un agente de acuerdo al estado del ambiente y el estado del agente, con el fin de maximizar una función de recompensa. Por ejemplo, aprender a jugar fútbol, donde quieres maximizar el número de goles anotados y minimizar los recibidos, el ambiente es el campo de fútbol y los agentes son los jugadores.

Regresión: Tarea de aproximar una función para mapear las variables de entrada X a salidas continuas Y, que son valores reales asociado a cantidades (La función predice un valor numérico real)

Glosario

Clasificación: Tarea de aproximar una función para mapear las variables de entrada X a salidas discretas categóricas Y conocidas como etiquetas o clases. (La función predice clases)

Agrupamiento: Tarea de encontrar similitudes en los datos y particionarlos (agruparlos) en clusters cuando se tienen datos sin etiquetar (no se sabe la respuesta correcta).

Precisión (precision): La dispersión del conjunto de valores obtenidos de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión

Exactitud (accuracy): Grado de concordancia entre los resultados de una medición y el valor verdadero

Estimador: Un estimador es cualquier objeto que aprende de los datos; Puede ser un algoritmo de clasificación, regresión o agrupamiento o un transformador que extrae / filtra características útiles de los datos sin procesar

Conjunto de entrenamiento (training set): Los ejemplos que utiliza el sistema para aprender se denominan conjunto de entrenamiento. Cada ejemplo de entrenamiento se llama una instancia de entrenamiento (o muestra)

Rasgos (features): Un rasgo es una propiedad medible individual o **característica** de un fenómeno que se está observando.

Glosario

Sobreajuste (overfitting): Significa que el modelo funciona bien en los datos de entrenamiento, pero no generaliza bien

Subajuste (underfitting): Es lo opuesto al sobreajuste, cuando el modelo es demasiado simple para aprender la estructura de los datos

Hipótesis (hypothesis): Es una determinada función que esperamos sea similar a la función verdadera (la función objetivo) que queremos modelar

Hiperparámetro (hyperparameter): Es un parámetro del algoritmo de aprendizaje y no del modelo, el ajuste de hiperparámetros es realizado la persona

Generalización (generalization): Es la capacidad de un modelo matemático para desempeñarse igual con ejemplos diferentes a los ejemplos de entrenamiento (o validación). Comúnmente, los ejemplos diferentes deben tener la misma naturaleza de origen que los ejemplos de entrenamiento (o validación).