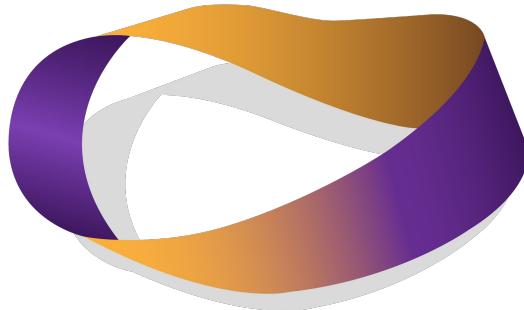


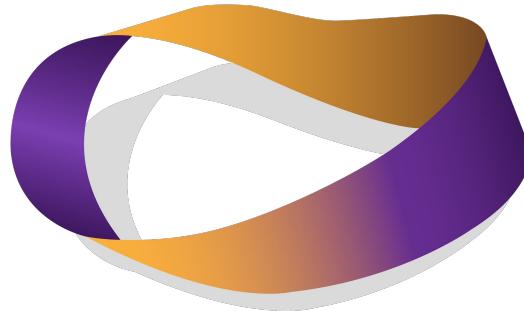
Machine Learning en producción



ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Machine Learning Pipelines



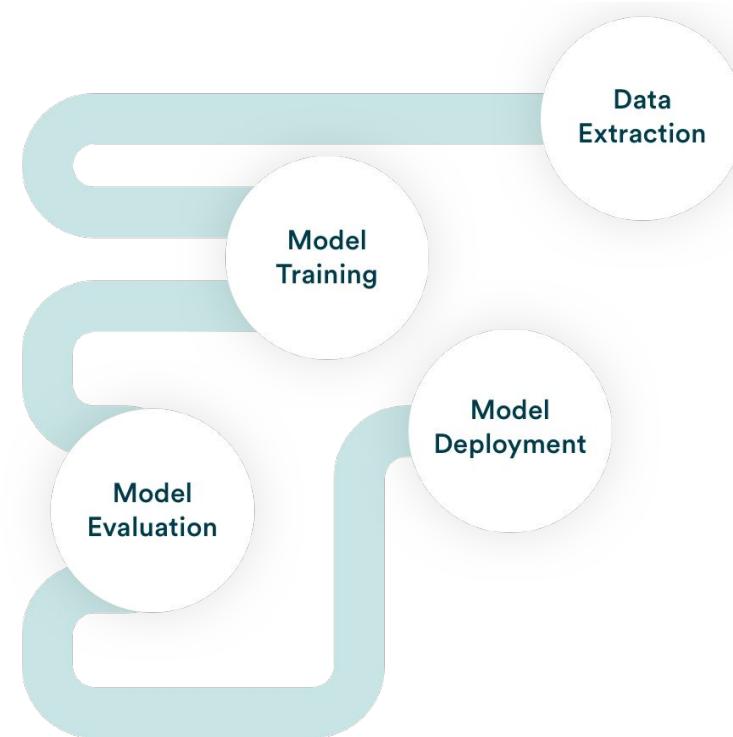
ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

¿Qué es un Pipeline?

Cuando se necesita usar el machine learning para resolver un problema real, es necesario seguir cierto procedimiento que nos asegure que la solución final es confiable

Nuestro trabajo es verificar cada uno de los pasos, para después unirlos como un sistema total



¿Por qué es necesario un Pipeline?

En retos pasados de este curso hemos observado formas de afrontar distintos problemas, desde el procesamiento de datos, imágenes y texto, sin embargo hemos parado en el momento que estos modelos presentan resultados

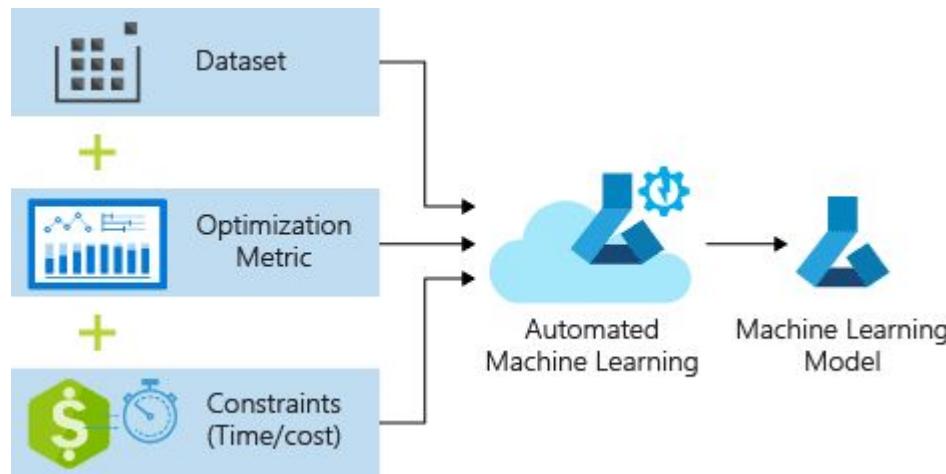
- ¿Qué pasa si los datos ya no son como los de entrenamiento?
- ¿Puede nuestro modelo usarse para un trabajo general o específico?



¿Por qué es necesario un Pipeline?

Los pipelines en machine learning implementan y formalizan los procesos para acelerar, reusar, controlar y llevar a producción los modelos de machine learning.

Ayudando a simplificar las herramientas y conceptos necesarios para construir un buen modelo en producción



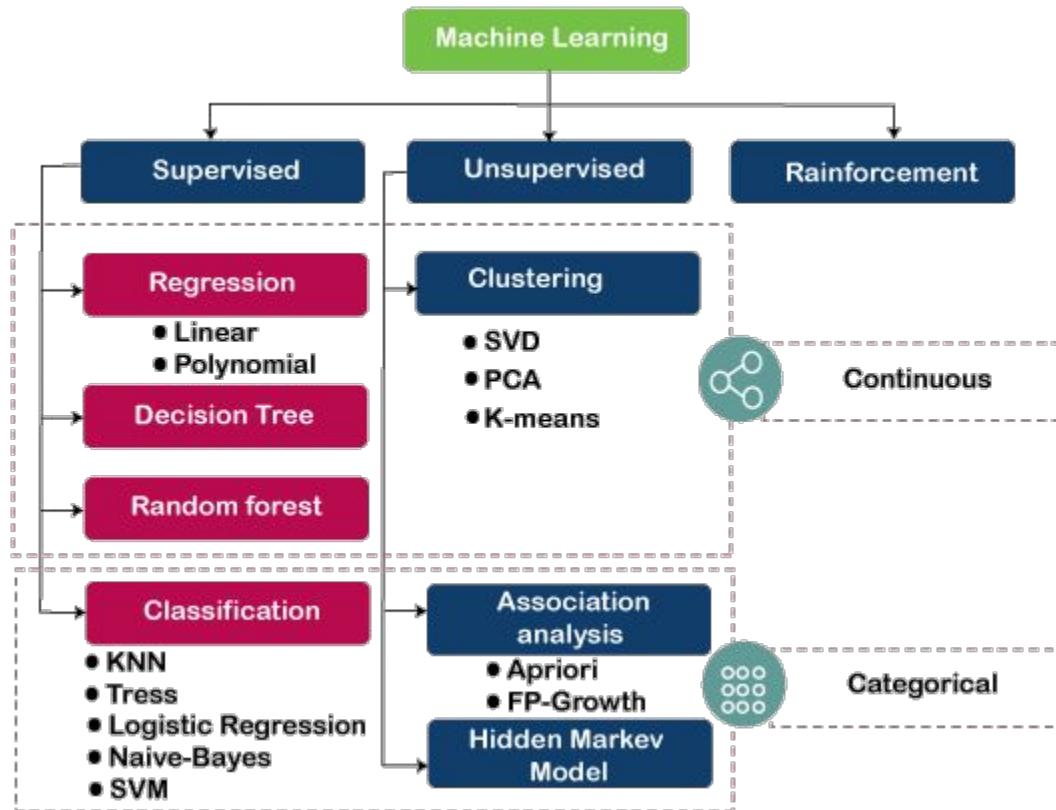
¿Qué es necesario en un Pipeline?

Un pipeline correcto debe asegurar ciertos pasos:

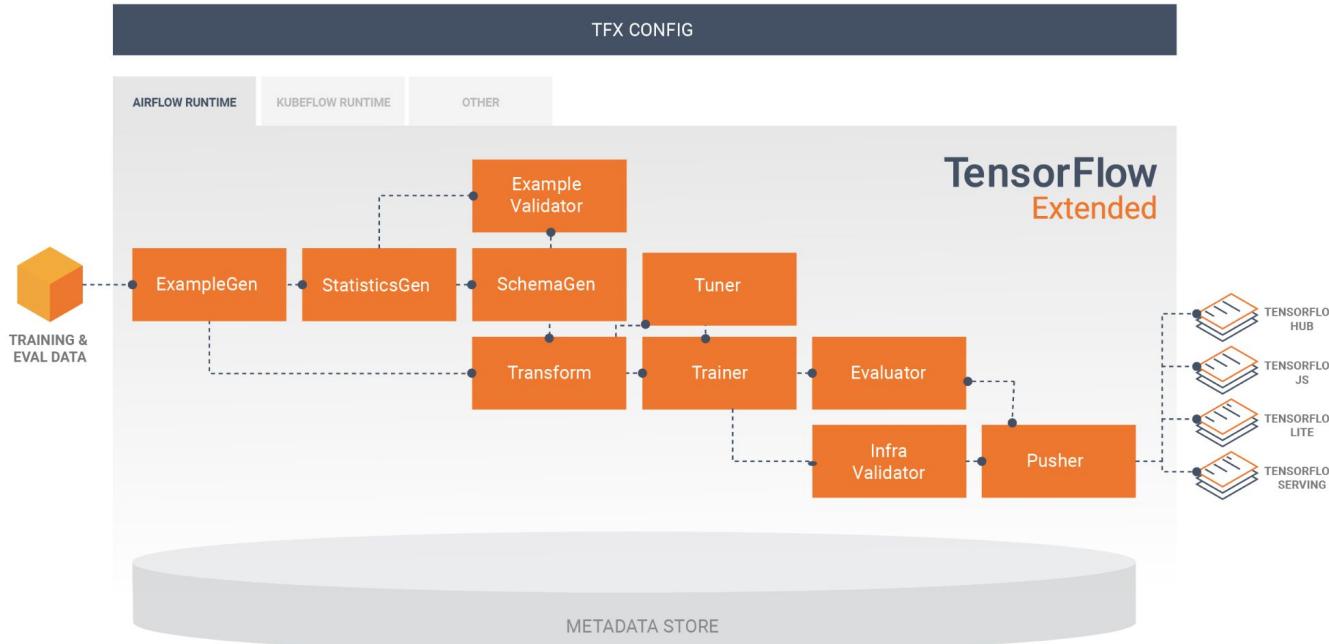
- Versionar los datos eficientemente y acelerar el entrenamiento de un nuevo modelo
- Validar los datos recibidos y prevenir pérdida de datos
- Pre-procesar eficientemente los datos para el entrenamiento y validación de los modelos
- Entrenar eficientemente los modelos de machine learning
- Dar seguimiento al entrenamiento de los modelos
- Analizar y validar los modelos entrenados
- Desplegar el modelo validado
- Escalar el modelo desplegado
- Capturar nuevos datos de entrenamiento y analizar las métricas de rendimiento de forma regular

¿Qué paso nos falta?

Elegir el mejor modelo



¿Con qué puedo crear un pipeline?



Tensorflow y Tensorflow Extended

Tensorflow como hemos revisado en una de las herramientas más usadas para machine learning y deep learning de la cual nos podemos apoyar de su extensión Extended para crear nuestros pipelines

- El ecosistema de Tensorflow es de los más usados para machine learning. Incluye múltiples proyectos y librerías como son Tensorflow Privacy y Tensorflow Probability
- Es muy popular y usado ampliamente en pequeños y largos setups de producción, con una comunidad muy activa de usuarios
- Soporta su uso en la academia y en producción. TFX es integrado completamente para su uso en producción
- Tensorflow y TFX son open source, sin restricciones de uso



¿Por qué es necesario un Pipeline?

El mayor beneficio que tenemos de los pipelines en machine learning, es que podemos automatizar el ciclo de vida de un modelo. Esto es, cuando tenemos datos nuevos, es necesario un flujo de trabajo que incluye validación de datos, preprocessamiento, entrenamiento del modelo, análisis, despliegue. Todo esto debemos de poder hacerlo de forma automática



Veamos los detalles de un buen pipeline

Enfocarse en nuevos modelos, no en mantener modelos existentes

La automatización de pipelines de machine learning, libera un gran tiempo usado en el mantenimiento de los procesos existentes. Muchos proyectos cuando son creados se hacen de forma manual donde un experto es quien debe hacer todos los pasos necesarios hasta desplegar el proyecto.

El problema es que si existen nuevos datos, el proceso debe realizarse nuevamente costando tiempo



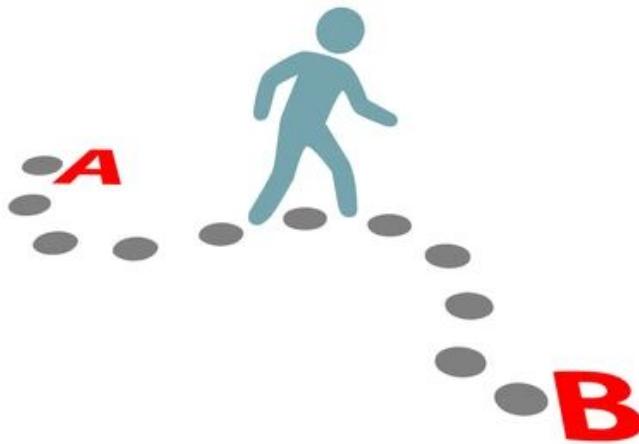
Prevención de Bugs

Pipelines automáticos pueden prevenir bugs, cuando estamos creando un nuevo modelo, debemos de verificar que los datos puedan ser ajustado nuevamente, se puede dar el caso que se cambie el procesamiento de los datos pero el modelo no necesite necesariamente ese procesamiento



Rastreo del camino

Cuando seguimos un pipeline, podemos crear un histórico de los pasos seguidos, modificaciones y resultados, útiles cuando queremos rehacer un modelo o modificarlo, esto nos ayuda a saber que posibles modificaciones pueden resultar en mejoras sin probarlas nuevamente



Estandarización

Dado que los pasos han sido marcados, es fácil seguirlos nuevamente cuando se recrea un sistema, muchas veces solo es necesario el nuevo dataset para poder ponerlo a trabajar

Casos de negocio de pipelines

Denotaremos los logros más importantes de los pipelines en tres:

- Más tiempo para producir nuevos modelos
- Proceso simple para mejorar los modelos existentes
- Menos tiempo gastado en reproducir los modelos

Esto nos ayuda a reducir el costo de nuestro proyecto y además:

- Ayuda a detectar posibles sesgos en los datasets de entrenamiento - (sesgos en poblaciones de uso)
- Crear los pasos a seguir para conseguir buenos resultados
- Liberar tiempo de los desarrolladores sin pérdida de rendimiento



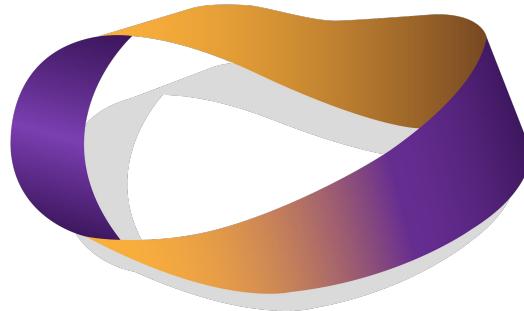
¿Siempre es bueno un Pipeline?

Aunque hemos dado grandes beneficios del uso de pipelines, también es cierto que estos no necesariamente deben realizarse para cada uno de los proyectos que realicemos

- Sistemas de prueba mínimos
- Sistemas únicos
- Modelos de aprendizaje



Tensorflow Extended (Introducción)



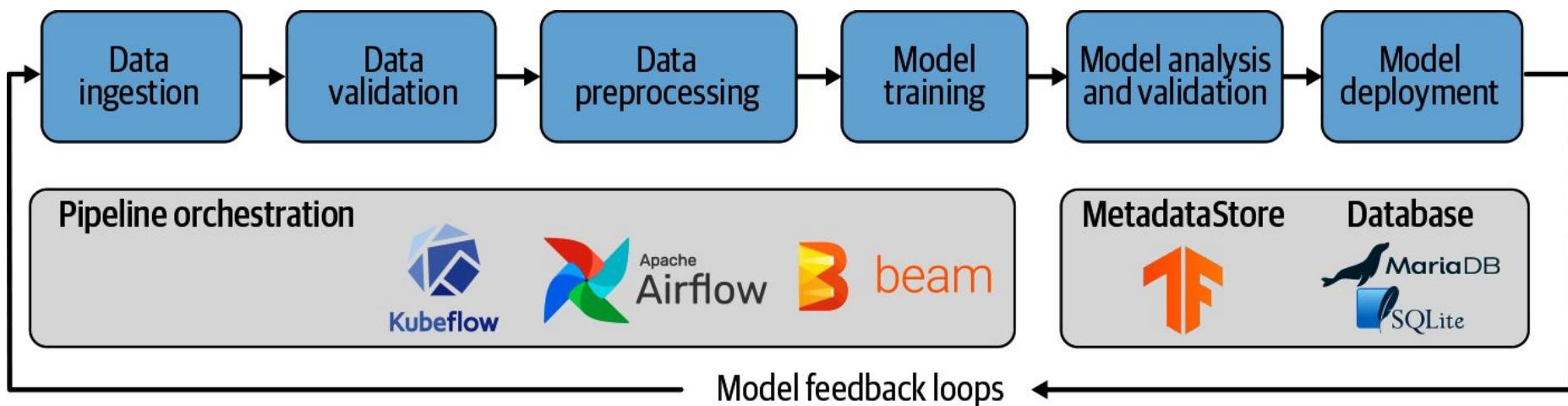
ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Tensorflow Extender - Pipelines de machine learning

Como hemos explicado, el proceso completo para llevar a cabo un modelo de machine learning en producción contiene un gran número de pasos, por lo que asegurarnos que cada uno de estos es una tarea agotadora si se trata de varios modelos o modelos que necesitan cambiar continuamente

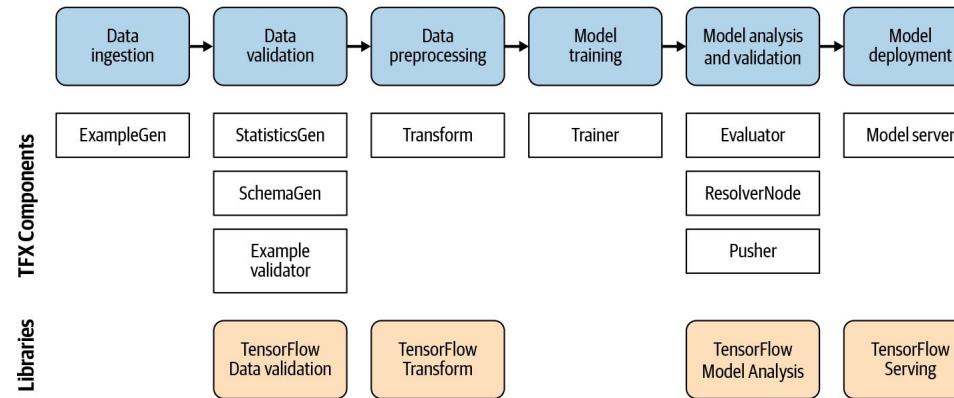
Los desarrolladores de Google al enfrentarse a esta tarea, crean TFX el cual nos ayuda a simplificar el proceso de la creación de los pipelines



Tensorflow Extender - Pipelines de machine learning

Esto simplifica los procesos que anteriormente hemos tenido que realizar manualmente con la ayuda de sus componentes:

- Ingesta de datos - ExampleGen
- Validación de datos - StatisticsGen, SchemaGen y ExampleValidator
- Pre-procesamiento de datos - Transform
- Entrenamiento de modelos - Trainer
- Checar modelos entrenados previamente - ResolverNode
- Análisis y evaluación de modelos - Evaluator
- Despliegue de modelos - Pusher

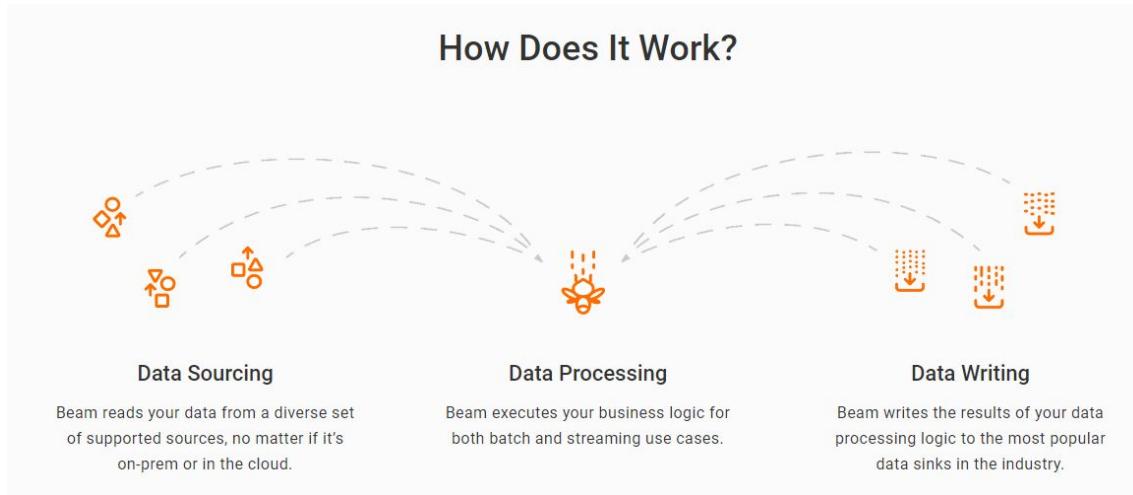


Apache Beam

Además de Tensorflow Extended debemos usar una herramienta que nos servirá para “pegar” todos los componentes y realizar la función de orquestador



How Does It Work?



Reto: Inicializa un ambiente de Tensorflow Extended, prueba la carga correcta de los diferentes módulos

Resultado Esperado:

- En caso necesario instala las librerías necesarias

```
  Downloading tfx-1.12.0-py3-none-any.whl (2.7 MB)
[██████████| 2.7 MB 4.3 MB/s]
Requirement already satisfied: jinja2<4,>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from tfx) (2.11.3)
Collecting google-cloud-aiplatform<1.18,>=1.6.2
  Downloading google_cloud_aiplatform-1.17.1-py2.py3-none-any.whl (2.3 MB)
[██████████| 2.3 MB 36.8 MB/s]
Requirement already satisfied: absl-py<2.0.0,>=0.9 in /usr/local/lib/python3.8/dist-packages (from tfx) (1.3.0)
Collecting apache-beam[gcp]<3,>=2.40
  Downloading apache_beam-2.43.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (14.5 MB)
[██████████| 14.5 MB 40.5 MB/s]
Collecting attrs<22,>~19.3.0
  Downloading attrs-21.4.0-py2.py3-none-any.whl (60 kB)
[██████████| 60 kB 6.5 MB/s]
```

Tips:

- Verifica como se instala una librería en COLAB

```
Successfully uninstalled PyYAML-6.0
ERROR: pip's dependency resolver does not currently take into account all
xarray 2022.12.0 requires packaging>=21.3, but you have packaging 20.9 whi
Successfully installed apache-beam-2.43.0 attrs-21.4.0 cachetools-4.2.4 cl
WARNING: The following packages were previously imported in this runtime:
[google]
You must restart the runtime in order to use newly installed versions.
```

Reto_01_TFX_intro.ipynb

Solución:

```
# Instalación de Tensorflow Extended  
!pip install tfx
```

Funcionamiento de los componentes

Como hemos visto, la forma de construir el pipeline es mediante bloques los cuales necesitan realizar una tarea específica, pero todos tienen una forma similar de trabajar

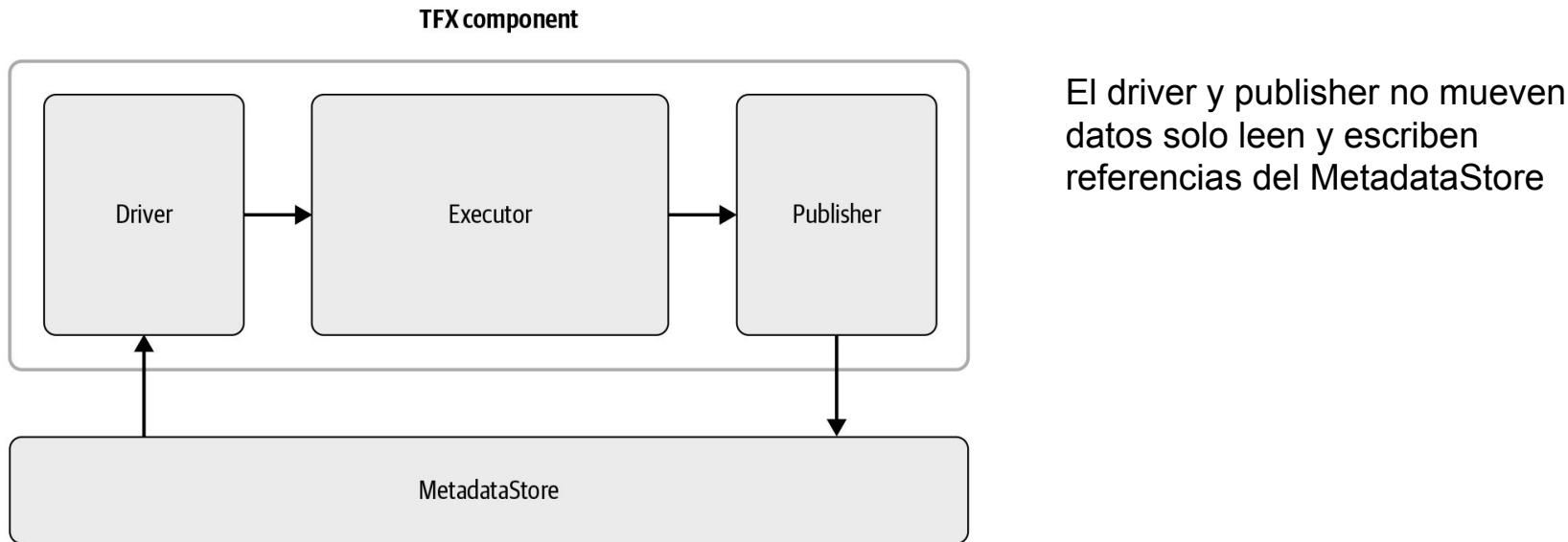
- Recibir una entrada
- Realizar una acción
- Guardar el resultado



Funcionamiento de los componentes

Como hemos visto, la forma de construir el pipeline es mediante bloques los cuales necesitan realizar una tarea específica, pero todos tienen una forma similar de trabajar

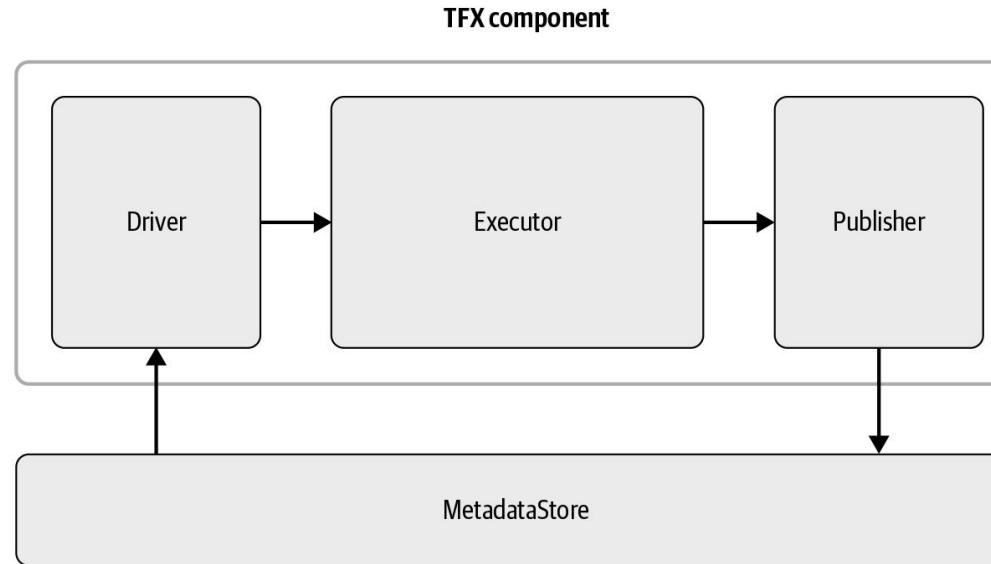
- Driver - maneja el orden de los datos guardados
- Executor - realiza la acción de los componentes
- Publisher - maneja el guardado de los datos



Más componentes

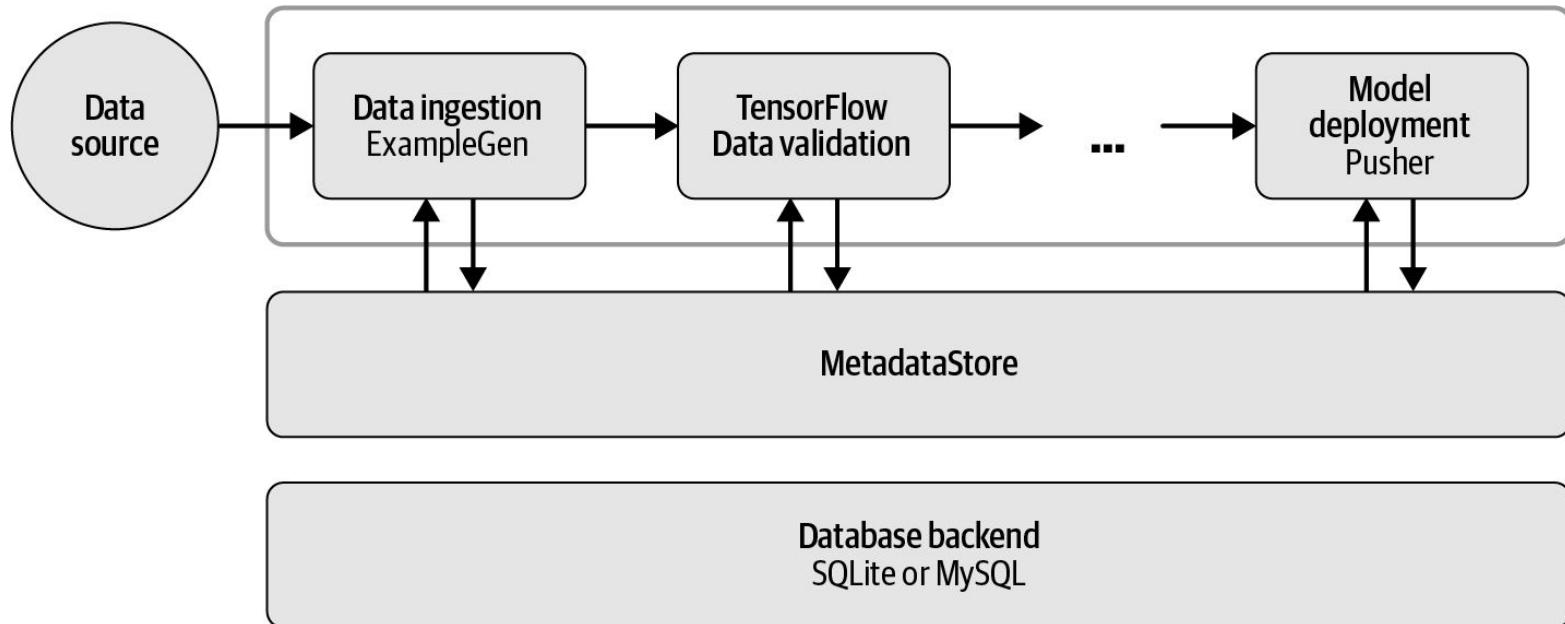
A las entradas y salidas de cada ‘component’ le llamaremos ‘artifacts’

- Raw input data
- Preprocessed data
- Trained models
- Etc



Forma de comunicarse

Para este tipo de pipeline los componentes no interactúan directamente mandando información si no que hacen llamados y escriben sobre el metadata, de esta forma podemos tener los datos centrados



Pipelines interactivos

Como podemos ver existen muchos conceptos y acciones que realizar para tener un pipeline, sin embargo, TFX nos proporciona una forma interactiva de crearlos y evitar la mayor cantidad de bugs. Al inicio realizaremos todos los pasos en nuestro ambiente local (COLAB) y en los usos avanzados podremos verlos trabajar en sistemas de producción



Alternativas a TFX

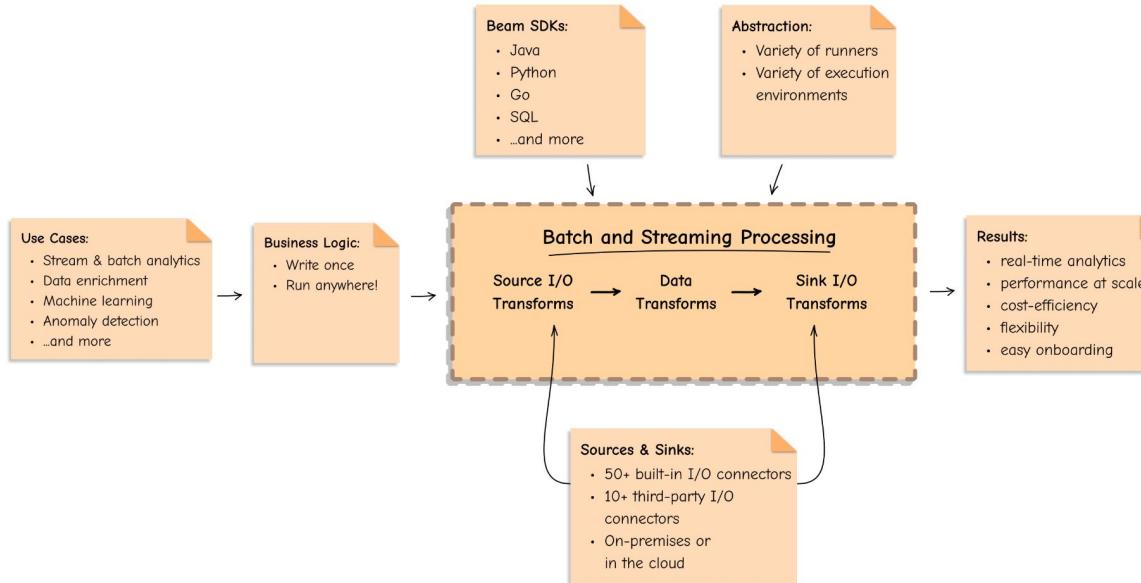
Así como Tensorflow tiene a TFX más compañías se han interesado en la generación de Pipelines de forma eficiente, donde podemos distinguir a:

Company	Framework	Link
AirBnb	AeroSolve	https://github.com/airbnb/aerosolve
Stripe	Railyard	https://stripe.com/blog/railyard-training-models
Spotify	Luigi	https://github.com/spotify/luigi
Uber	Michelangelo	https://eng.uber.com/michelangelo-machine-learning-platform/
Netflix	Metaflow	https://metaflow.org/

Apache Beam

Apache Beam es un modelo open-source unificado para el procesamiento de pipelines que simplifica el proceso para datos a gran escala

En nuestro caso lo usaremos para realizar algunas de las tareas de TFX como son Tensorflow Transform o Tensorflow Data Validation



Apache Beam

Una de las ventajas más grandes que nos ofrece este sistema es que podemos trabajar con varios tipos de servicios con pocos o cero comandos

- pip install apache-beam - Sistema integro de apache beam
- pip install 'apache-beam[gcp]' - Para trabajar con Google Cloud Platform
- pip install 'apache-beam[boto]' - Para trabajar con Amazon Web Services



Construyendo un pipeline (Modo manual)



ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Almacenamiento e ingestá de datos



ACTUMLOGOS

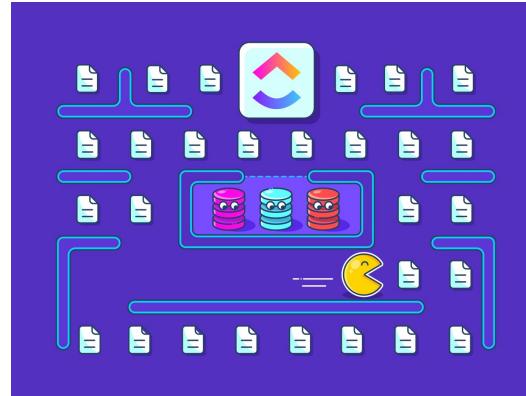
DESARROLLANDO HABILIDADES TECNOLÓGICAS

Almacenamiento e ingesta de datos

El primer paso de todo uso de machine learning es conseguir los datos sobre la problemática, la adquisición se puede generalmente por bases de datos que contienen grandes cantidades de datos relacionadas a muchos sistemas.

- Bases de datos en internet
- Bases de datos del trabajo
- Bases de datos de universidades

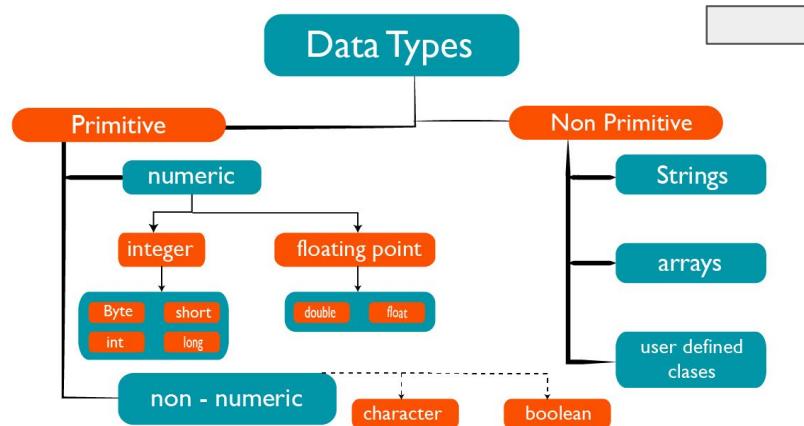
¿Qué pasa cuando no hay datos de mi problema?



Almacenamiento e ingesta de datos

Además del acceso a la información, debemos de conocer el tipo de datos y cantidad que tenemos a nuestra disposición

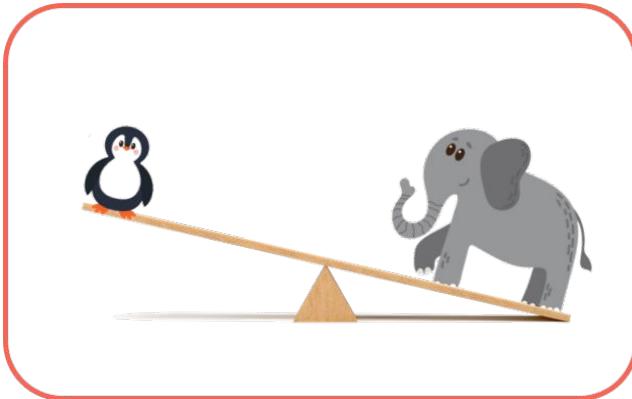
- Datos numéricos - Presentados como tablas con filas y columnas (archivos csv)
- Imágenes - Conocer su tipo y tamaño (PNG, JPG, Web, etc) (píxeles X píxeles)
- Texto - Presentado mediante tablas de filas y columnas o como texto plano (archivos csv o txt)
- Sonido - archivos del tipo MIDI, MP3, WAV y en caso necesario dividió por instancias (t tiempo de grabación)
- Series de tiempo - presentado como una secuencia de datos en N dimensiones



Almacenamiento e ingesta de datos

Uno de los grandes problemas del Big Data es el almacenamiento, ya que mientras más datos se tienen más tiempo de transferencia se necesita para moverlos de una localidad a otra.

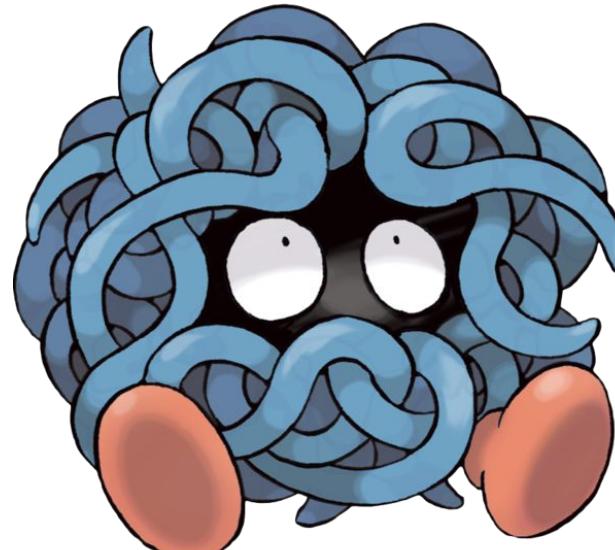
- Base de datos pequeña - Su uso generalmente es cargado en memoria y usado directamente sobre el problema
- Base de datos mediana - Es necesario usar un generador aleatorio que cargara y descargara la información en paquetes de la memoria para ser usados
- Base de datos grandes - La información se guarda en sistemas privados o externos a los cuales se accede mediante la nube, todo el procesamiento se realiza mediante comando y se cargan lotes de información mediante consultas



Preparación de los datos

Cuando tenemos acceso a los datos, es necesario poder entregarlos a los sistemas siguientes de una forma entendible, ordenada y simplificada

Cada tipo de dato necesita un procesamiento distinto, además que es necesario verificar el tipo de dataset en general que se presenta



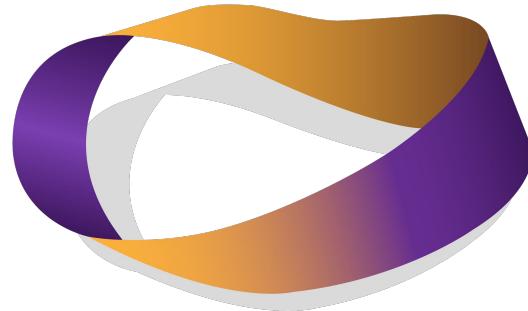
Más datos, más entrenamiento

En el caso de los sistemas de producción, cuando un modelo de ML o DL se ejecuta podemos utilizar estas nuevas entradas como una oportunidad de mejora, esto es, los nuevos datos nos servirán para refinarn el modelo

- El versionado es una de las formas de poder tener el control de qué tipo de datos son útiles y cuáles no, ya que algunas veces podemos encontrarnos con problemas de desbalanceo que anteriormente no teníamos



Validación de datos

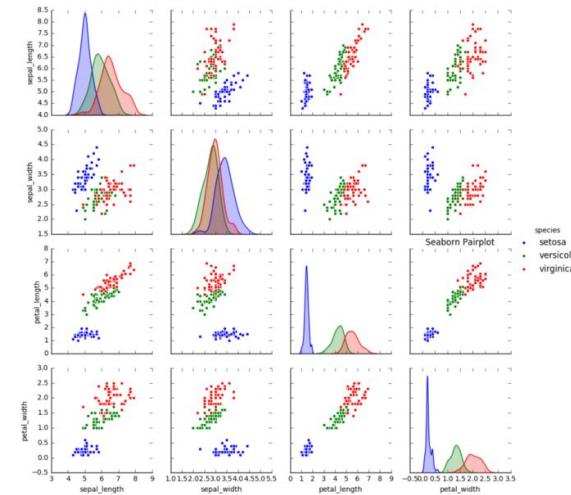
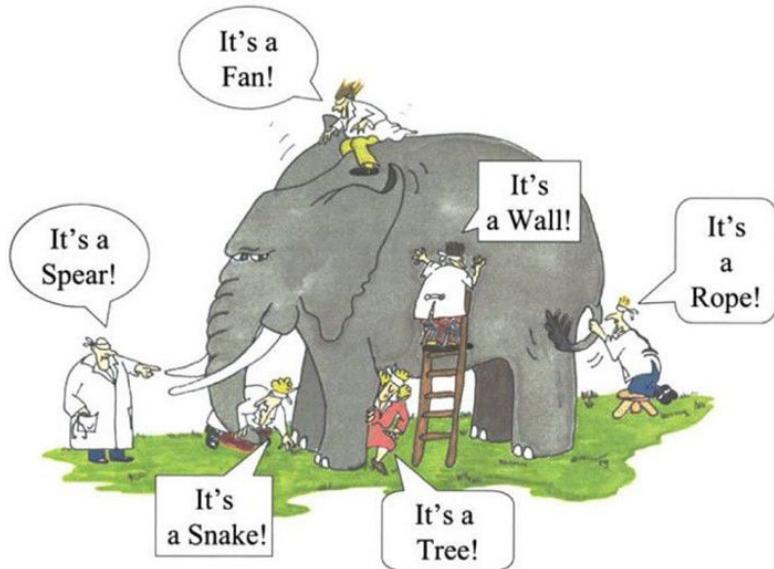


ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Explorar los datos y entenderlos

El primer paso después de obtener los datos para nuestros modelos, es verificar si estos datos realmente son útiles y que sean correctos para nuestro caso, esto lo podemos realizar mediante una exploración que podemos definir como encontrar las propiedades que hacen a este dataset este dataset



Reto: Complete el Reto_02_validacion para observar los datos descritos dentro del archivo, verifique que tipo de datos es y utilice las librerías para observar sus estadísticas

Resultado Esperado:

	0	1	2	3
count	404.000000	404.000000	404.000000	404.000000
mean	3.745111	11.480198	11.104431	0.061881
std	9.240734	23.767711	6.811308	0.241238
min	0.006320	0.000000	0.460000	0.000000
25%	0.081437	0.000000	5.130000	0.000000
50%	0.268880	0.000000	9.690000	0.000000
75%	3.674808	12.500000	18.100000	0.000000
max	88.976200	100.000000	27.740000	1.000000

Tips:

- Analice si es posible otro tipo de análisis
- Verifique otra base de datos y observe qué tipo de análisis necesita

Solución:

```
# Librerias
import tensorflow as tf
import pandas as pd

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.boston_housing.load_data(
    path="boston_housing.npz", test_split=0.2, seed=113
)

# Convertirlo a un dataframe
x_train = pd.DataFrame(x_train)
```

```
x_train.info()
```

```
x_train.describe()
```

Validación en imagen

En el reto anterior observamos cómo validar los datos cuando estos son del tipo ‘valor’ sin embargo, cuando los datos que necesitamos cambian de tipo es necesario poder validarlos mediante distintas observaciones

En el caso de las imágenes necesitamos observar:

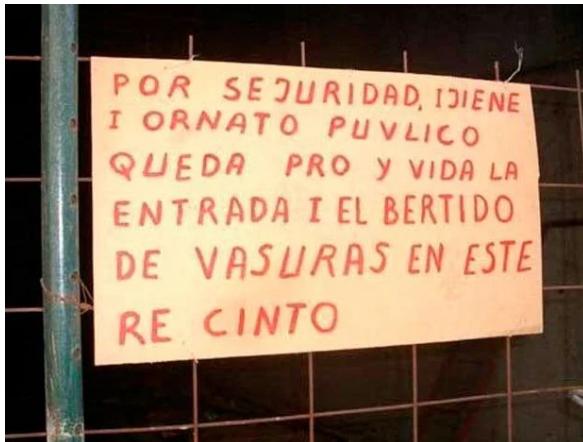
- Resolución de la imagen
- Tamaño de la imagen
- Formato
 - Escala de grises
 - RGB
 - YMC
- Nitidez



Validación en texto

Para el caso de texto, también debemos de tener en cuenta ciertos puntos:

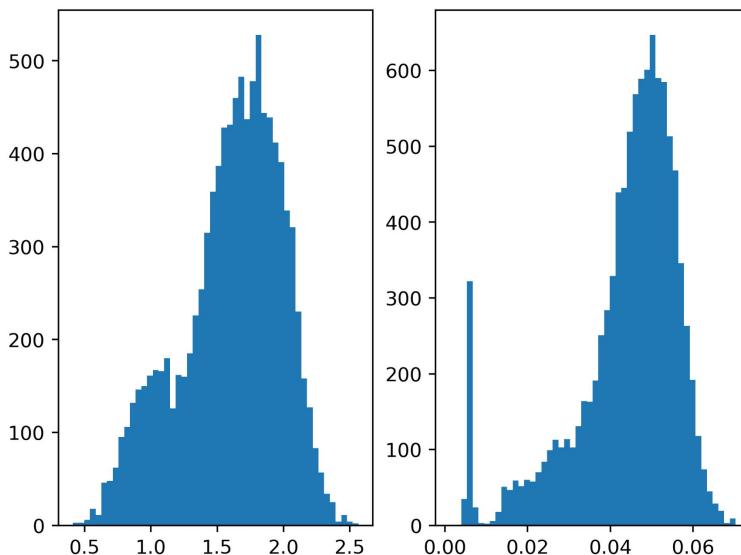
- Idioma
- Errores gramaticales
- Tipo de texto
 - Poemas
 - Histórico
 - Etc
- Sentido



Aumento de datos

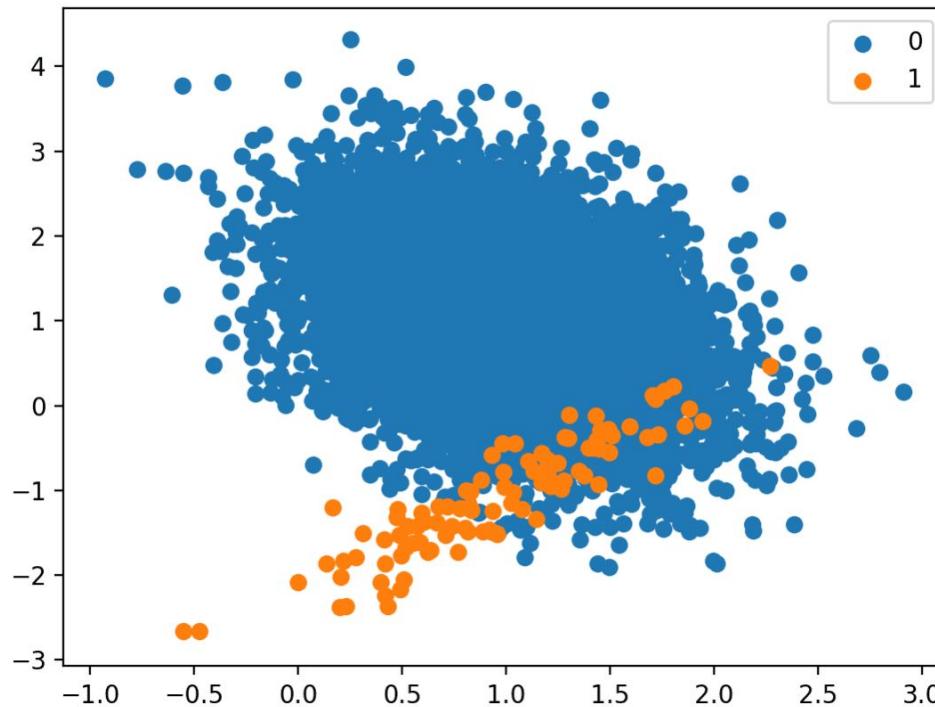
Como se indicaba anteriormente, los nuevos datos que entran de forma natural en producción pueden ser utilizados para mejorar los modelos de ML, sin embargo, es necesario tener algunas consideraciones:

1. Los datos deben ser ‘similares’ a los datos anteriores



Aumento de datos - Problema

Otro punto a considerar, es que los datos regularmente para modelos de ML y en algunos casos para los modelos de DL, se espera que sean balanceados



Preprocesamiento de los datos



ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

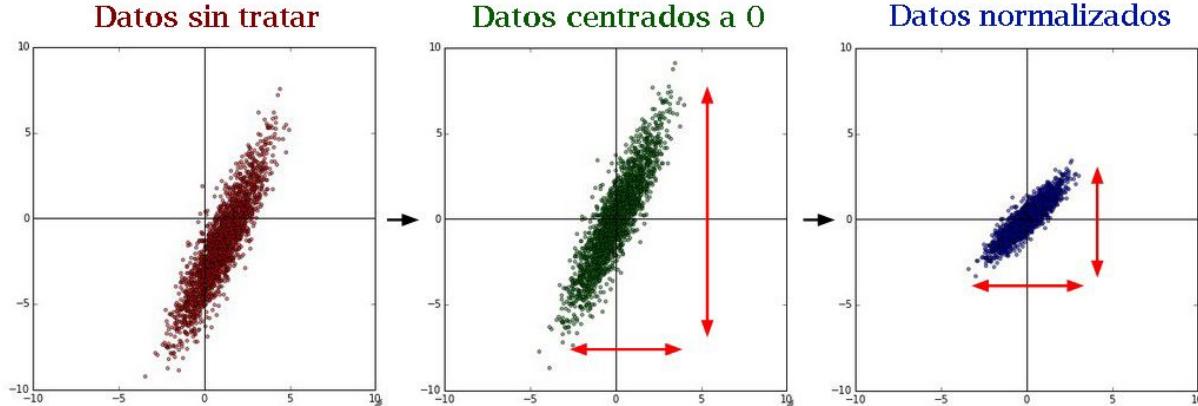
Los datos antes de los modelos

Después de recolectar nuestros datos y de verificar que estos tengan la información adecuada, es necesario transformarlos para que la computadora los entienda.

- En el caso de datos no numéricos, los datos del tipo ‘string’ deben ser convertidos mediante técnicas de vectorización
 - True = 1, False = 0
 - Amarillo = 1, Rojo = 2, Verde = 3
- También es necesario realizar una limpieza para que los datos no afecten a los modelos



Procesar los datos



Estandarización

Proceso de re-escalamiento de características para obtener una distribución gaussiana de $\mu=0$ y $\sigma=1$

$$z = \frac{x - \mu}{\sigma}$$

μ es la media, σ desviación estándar

Feature Scaling: El **escalado de características** es usado para hacer que todos los atributos tengan la misma escala

Normalización

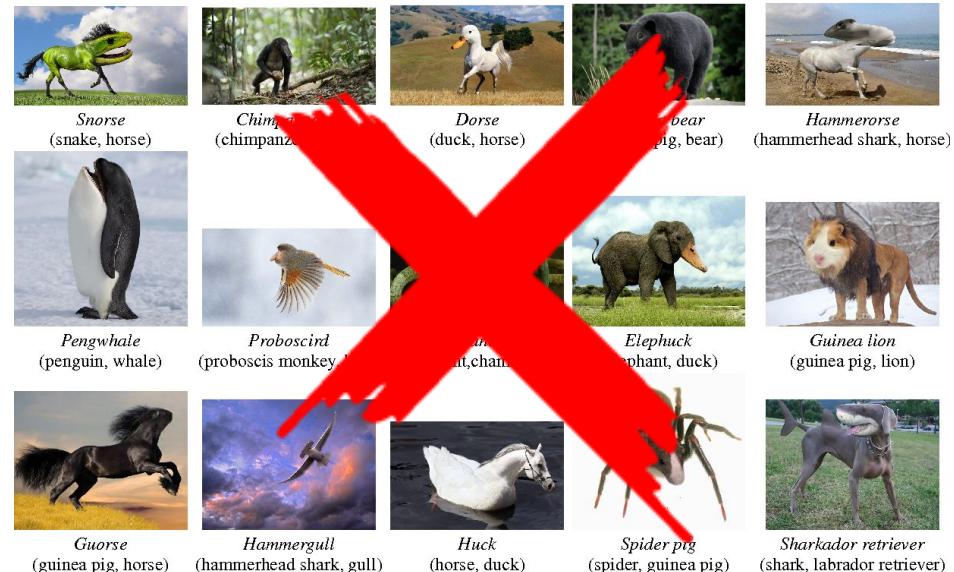
Reduce el rango de las características entre $[0, 1]$

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Limpiar los datos

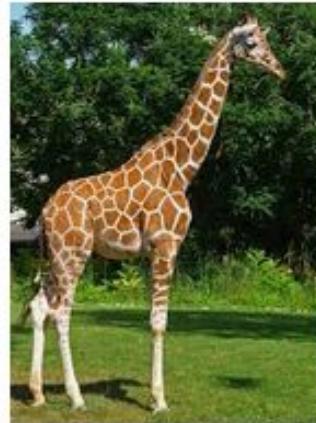


Eliminar redundancia



Eliminar datos corruptos

Limpiar los datos



$Y = \text{Jirafa}$

$Y = \text{Elefante}$

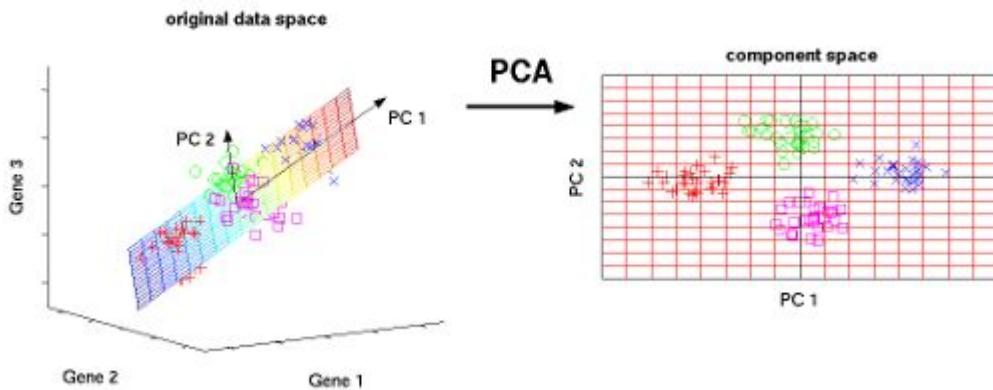
$Y = \text{Elefante}$

$Y = \text{Jirafa}$

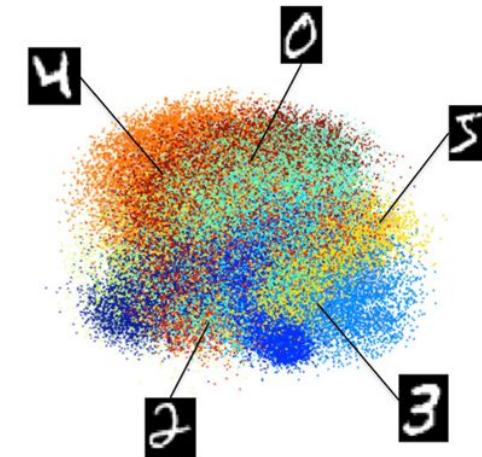
Corregir errores

Muchos datos, pocos recursos

Existen ocasiones cuando la cantidad de entrada de datos puede ser muy grande y si estamos utilizando un modelo de ML clásico o nuestros recursos en computación son bajos es recomendable reducir la entrada de datos, pero, **sin perder información**



5	0	4	1	9	2
1	3	1	4	3	5
3	6	1	7	2	8
6	9	4	0	9	1

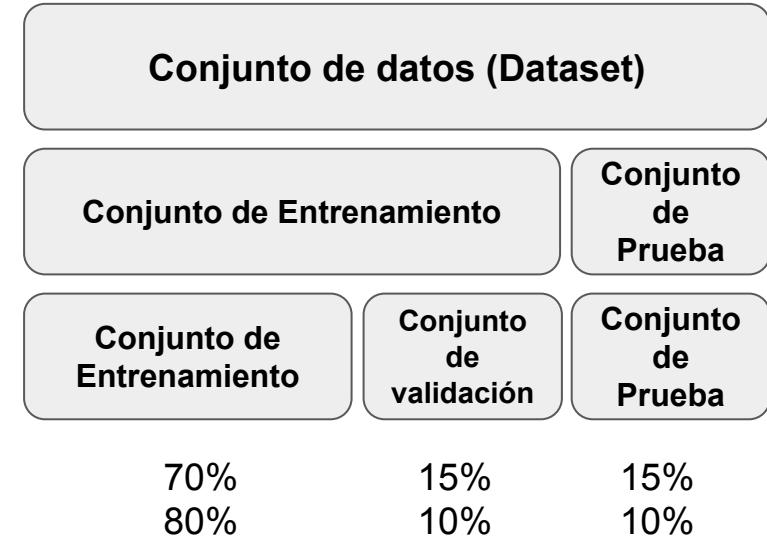


Separación de los datos en sets

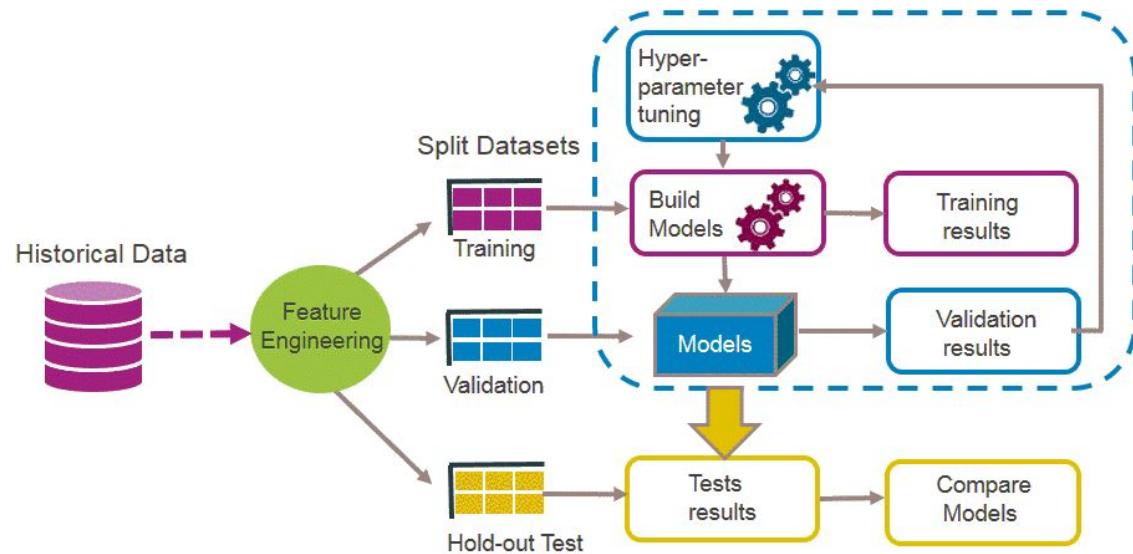
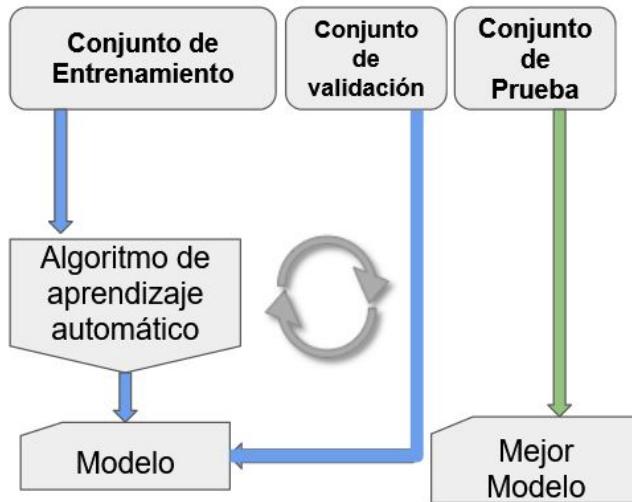
Representación de los datos

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.

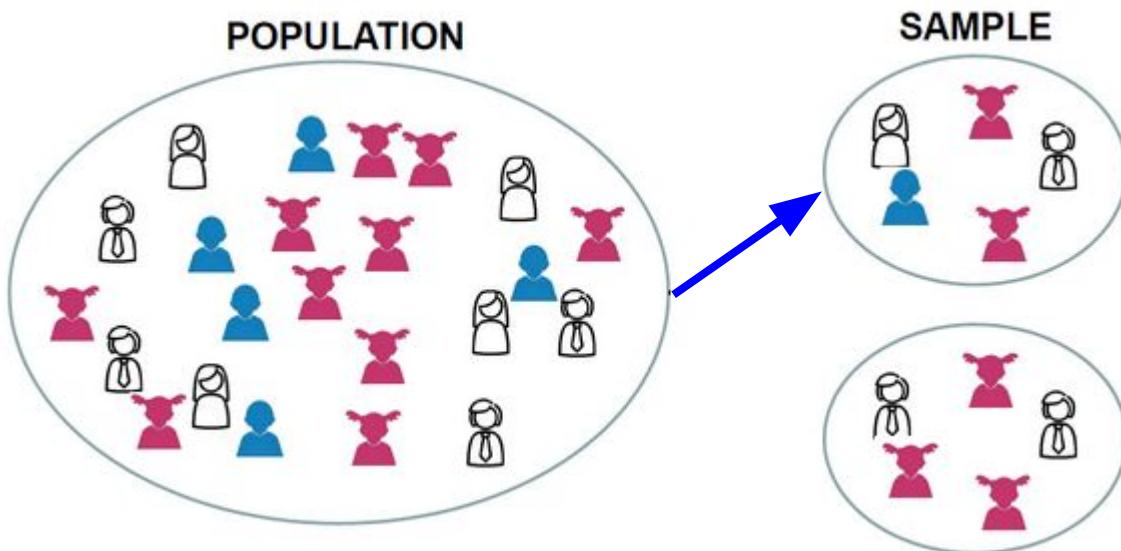


Separación de los datos en sets



Separación de los datos en sets

Aplicar un método aleatorio de muestreo es bueno si el conjunto de datos es muy grande, de lo contrario se corre el riesgo de introducir sesgo (Sampling Bias)



Solución:

Asegurarse de tomar muestras representativas para el conjunto de prueba

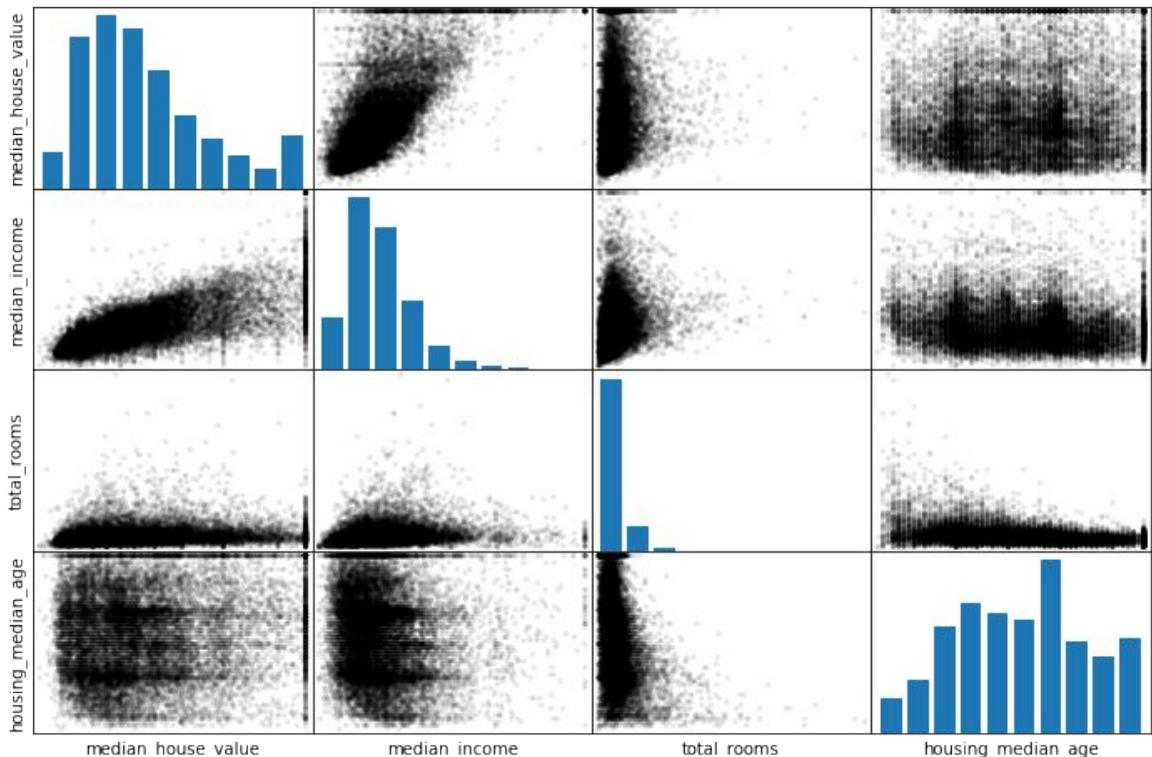
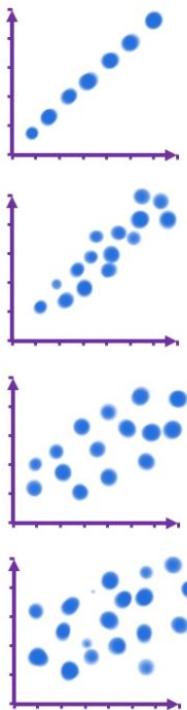
Muestreo aleatorio estratificado



- Los individuos se dividen en grupos o estratos.
- La muestra se elige escogiendo en cada estrato un número representativo de individuos.

Es importante tener una cantidad suficiente de muestras representativas en el conjunto de datos, de lo contrario podría generar un sesgo

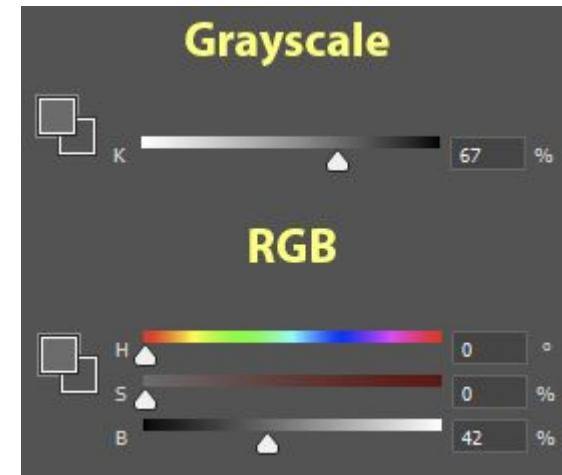
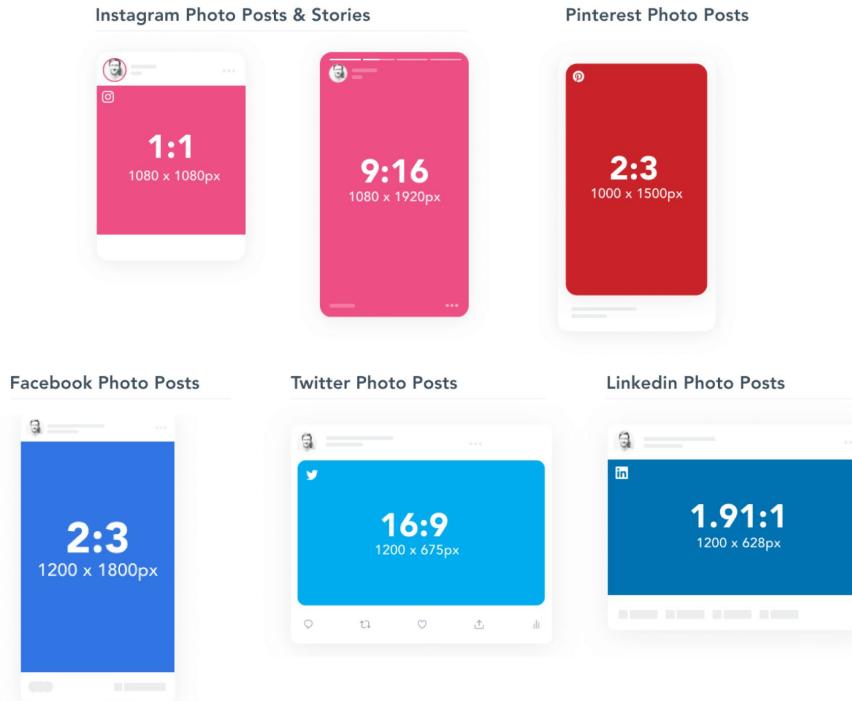
Correlación de los datos



Nota: La diagonal es el histograma de cada rasgo

Preprocesamiento en las imágenes

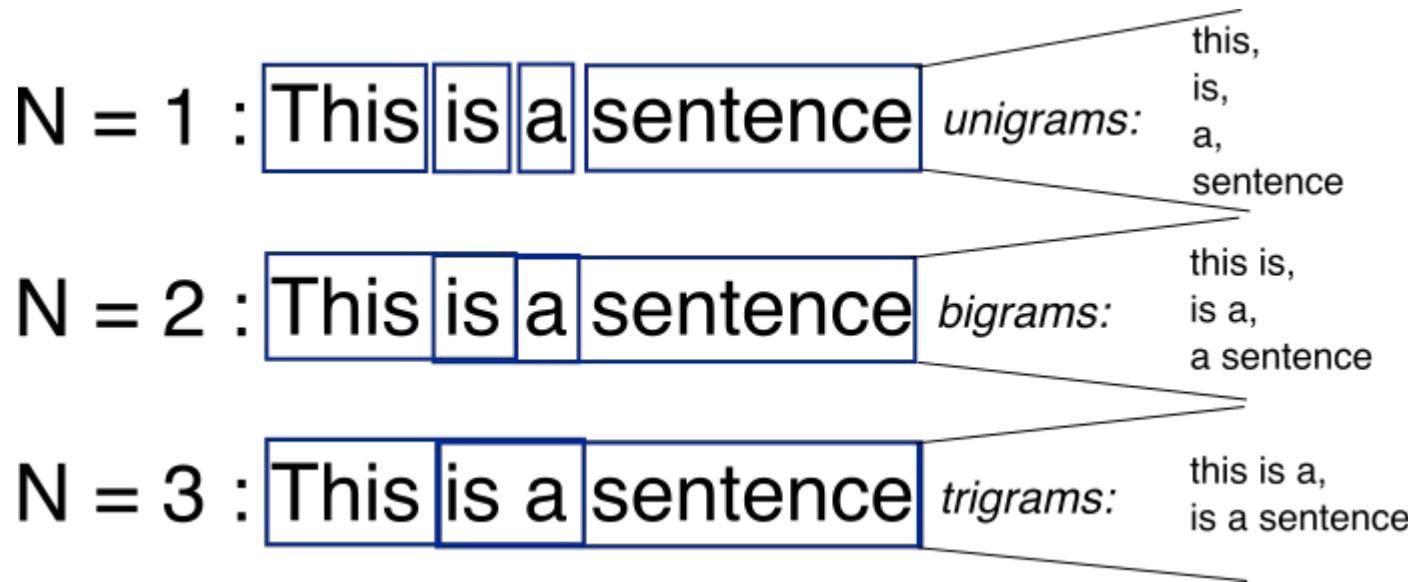
Para el caso de bases de datos de imágenes, el preprocesamiento solo debe identificar qué rasgos deben tener en común las imágenes



Preprocesamiento en texto

El texto, como sabemos, es tratado como una cadena de caracteres, la cual debe ser procesada para entenderla mediante los modelos necesarios, algunos pasos que se siguen para esto es:

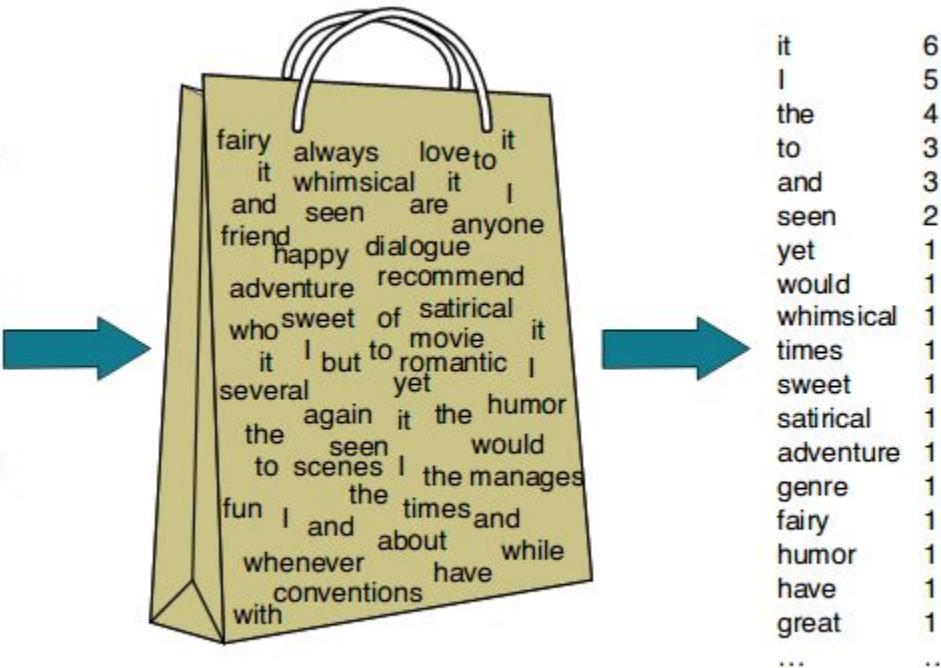
- Separación por palabras



Preprocesamiento en texto

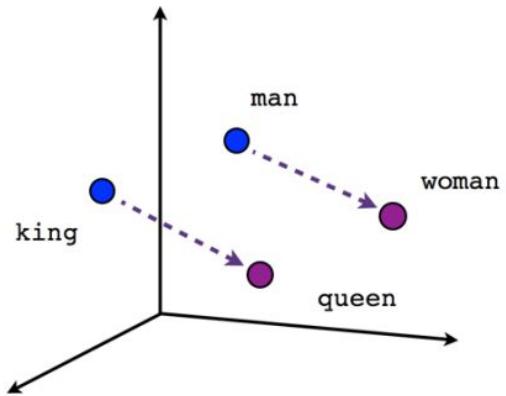
- Bolsas de palabras

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

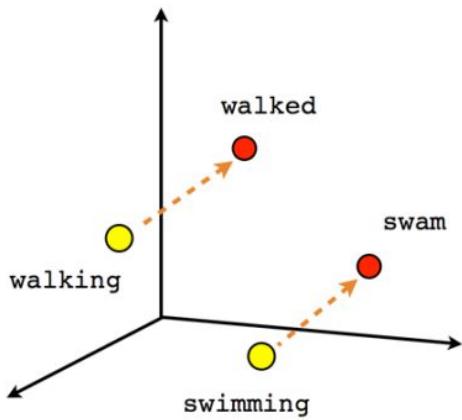


Preprocesamiento en texto

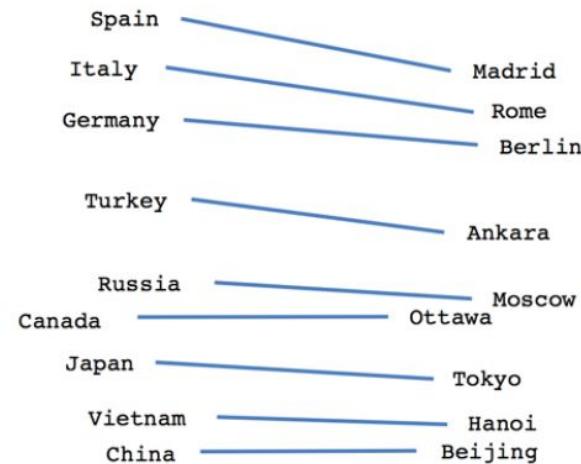
- Embeddings



Male-Female

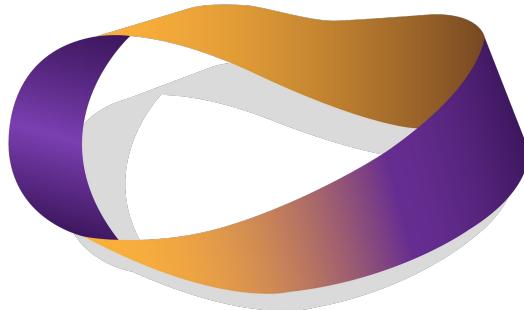


Verb tense



Country-Capital

Entrenamiento de modelos de ML

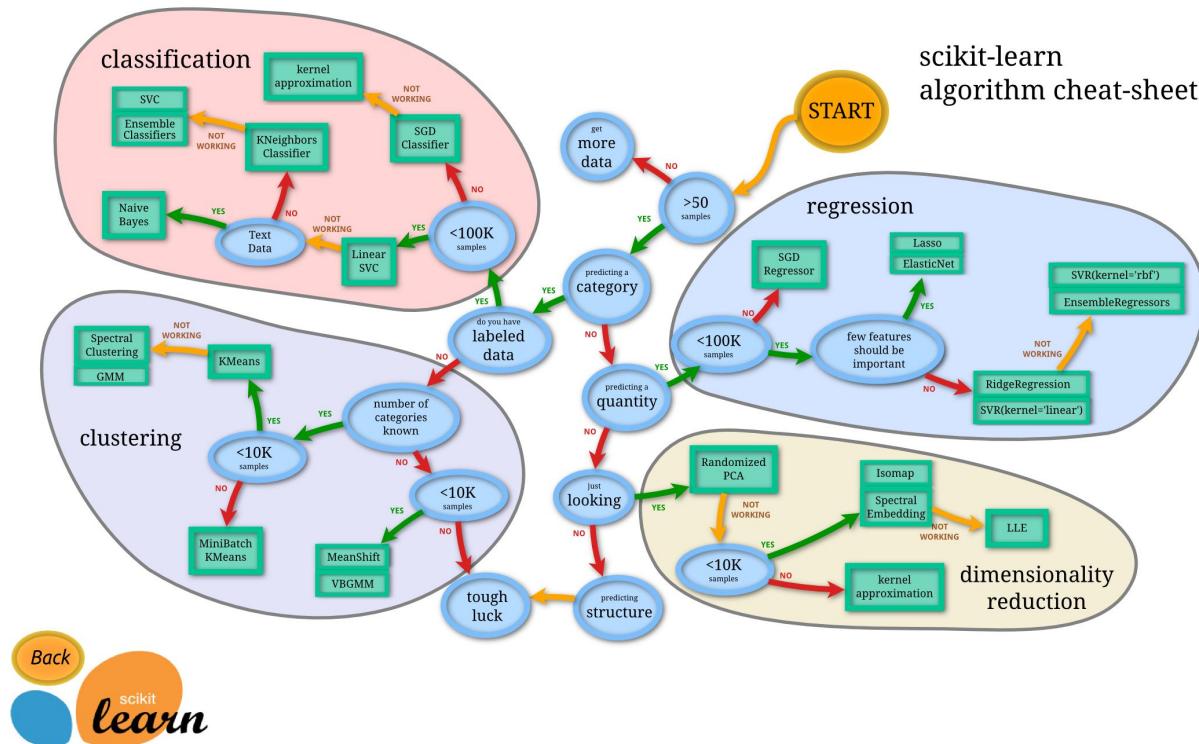


ACTUMLOGOS

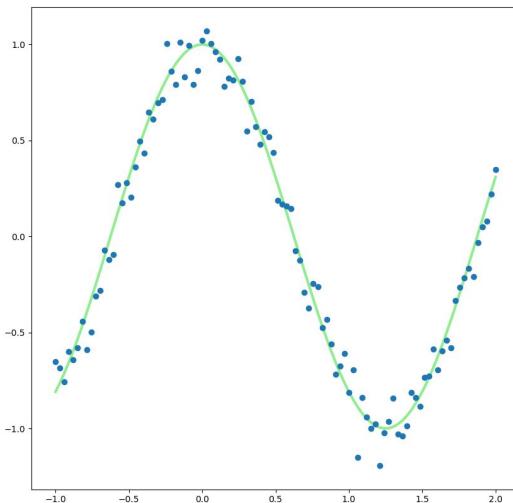
DESARROLLANDO HABILIDADES TECNOLÓGICAS

Modelos de ML y DL

Como hemos visto, existen múltiples modelos de ML y DL, cada uno de estos enfocados para un tipo de problema y saber cual utilizar nos puede ayudar a reducir el tiempo necesario para resolver el problema

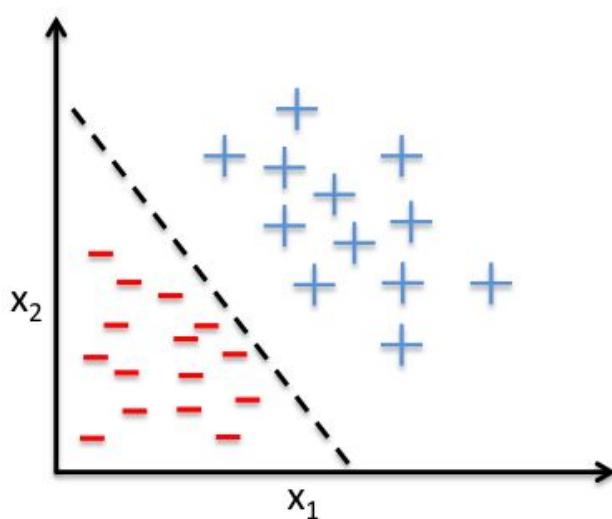


Regresión



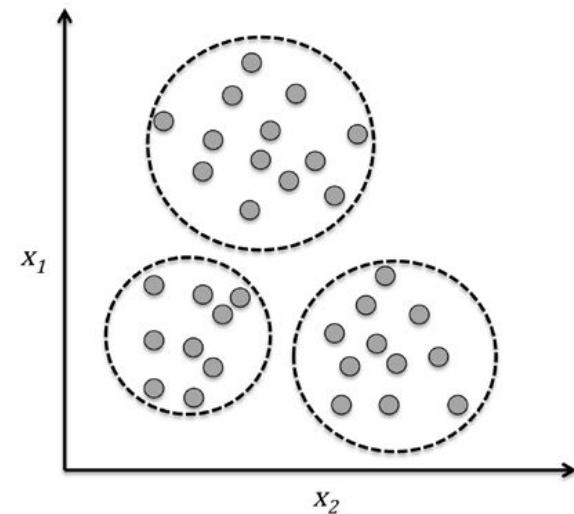
Tomamos el **conjunto de entrenamiento** para tratar de generar una función que se ajuste a los datos (predecir un valor numérico)

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



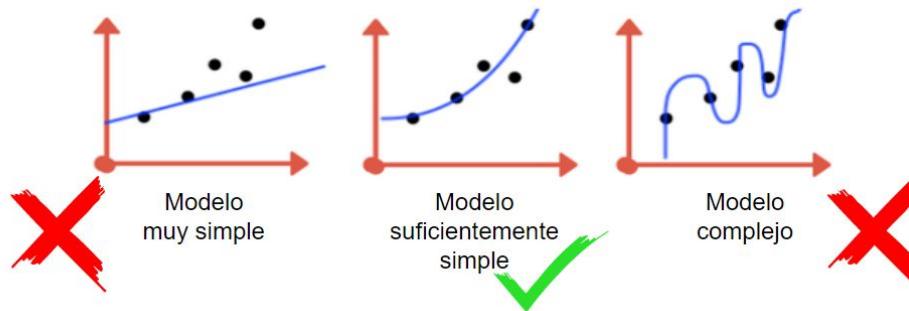
Es la división de los datos en grupos con **rasgos** similares (sin conocer las clases).

Modelo correcto para el problema

- En los datos - Los rasgos que describen los datos deben ser lo más simple posible, pero no los más simples.

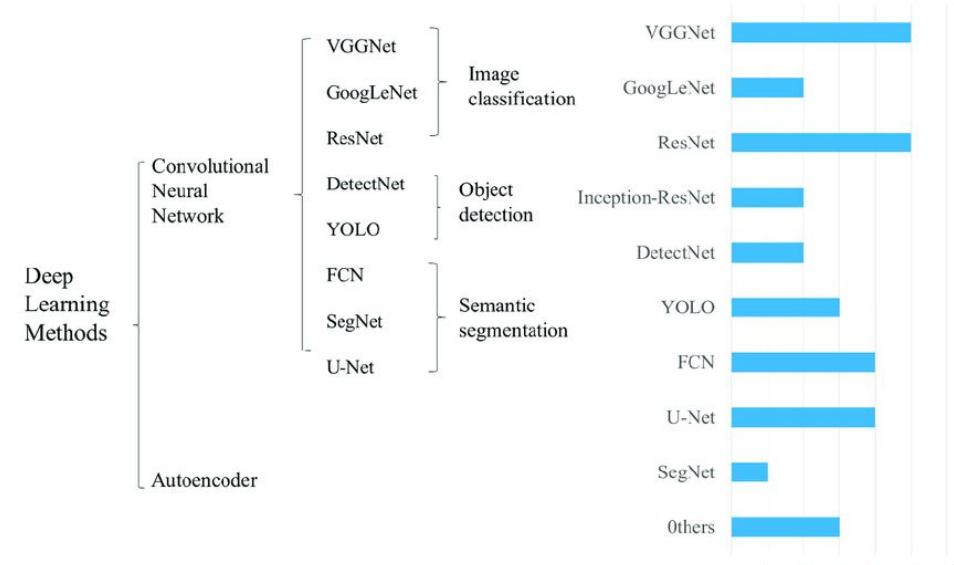


- En los modelos - El modelo más simple que se ajusta a los datos, también es el más plausible.



Modelos dependiendo del tipo de datos

- Modelos para datos
 - MLP
 - SVM
 - Árboles de decisión
 - Bosques aleatorios
 - KNN
 - PCA
- Modelos de imagen
 - Redes neuronales convolucionales
 - Redes profundas tipo perceptrón
- Modelos de texto
 - Redes profundas tipo perceptrón
 - Redes neuronales recurrentes
 - Transformers
- Modelos generativos
 - GANS
 - Encoder - Decoder
 - Style tranfer

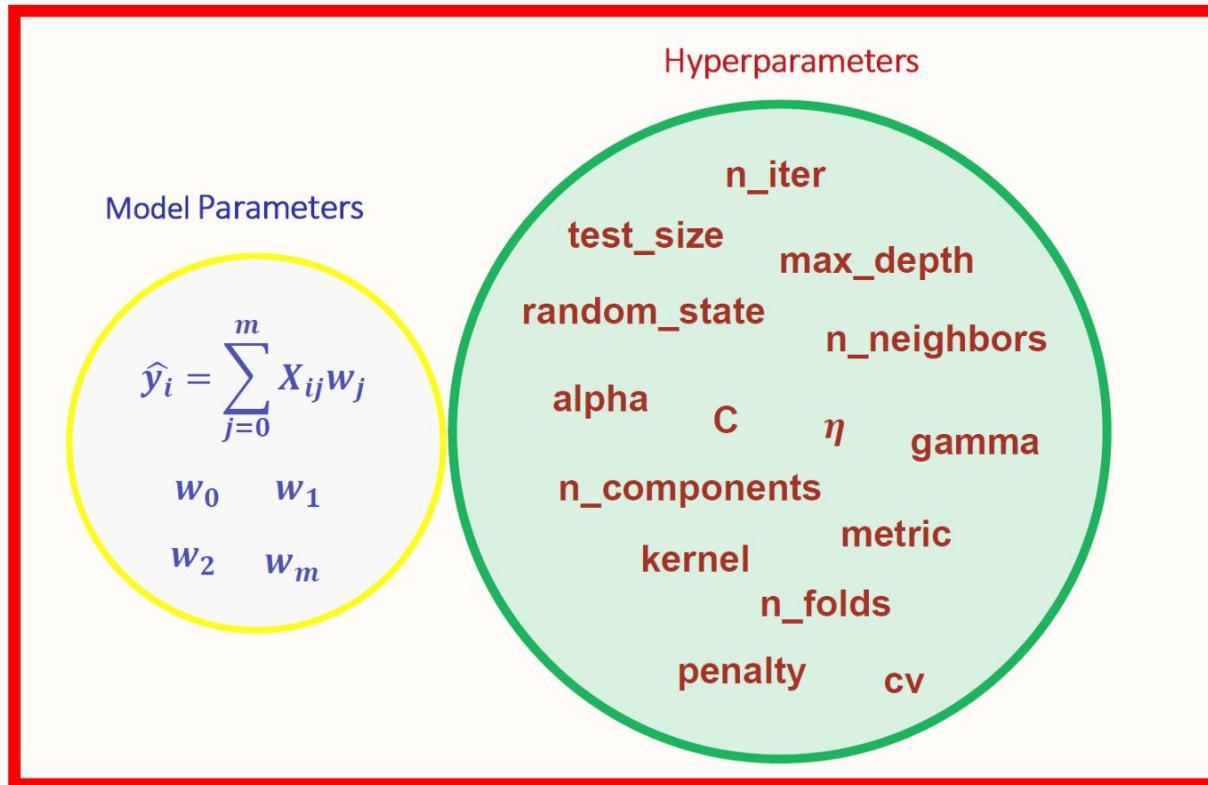


a

b

Parámetros de entrenamiento

Cada uno de los modelos de ML y DL contienen hiperparámetros únicos e hiperparámetros en común



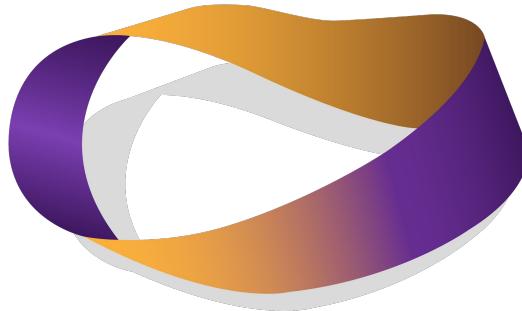
Parámetros de entrenamiento en DL

En el caso de DL, podemos definir varios hiperparámetros en común que debemos de especificar antes del entrenamiento

- Número de épocas
 - Veces que se entrenará el modelo con el set total de entrenamiento
- Tamaño del batch
 - Cantidad de datos que se usarán en ‘paralelo’ para un determinado paso de entrenamiento
- Función de optimización
 - Función que mide los resultados del modelo con los datos reales (y_{train})
- Set de validación
 - Cantidad de datos que se usan para medir el rendimiento durante el entrenamiento

Hyperparameter	Approximate sensitivity
Learning rate	High
Optimizer choice	Low
Other optimizer params (e.g., Adam beta1)	Low
Batch size	Low
Weight initialization	Medium
Loss function	High
Model depth	Medium
Layer size	High
Layer params (e.g., kernel size)	Medium
Weight of regularization	Medium
Nonlinearity	Low

Monitoreando procesos de entrenamiento

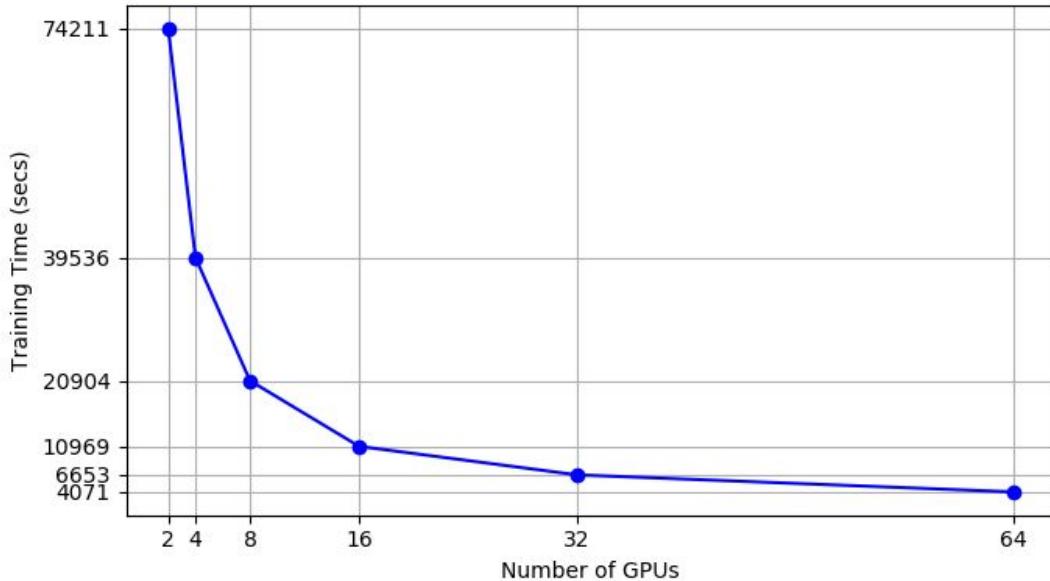


ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Monitoreando el entrenamiento

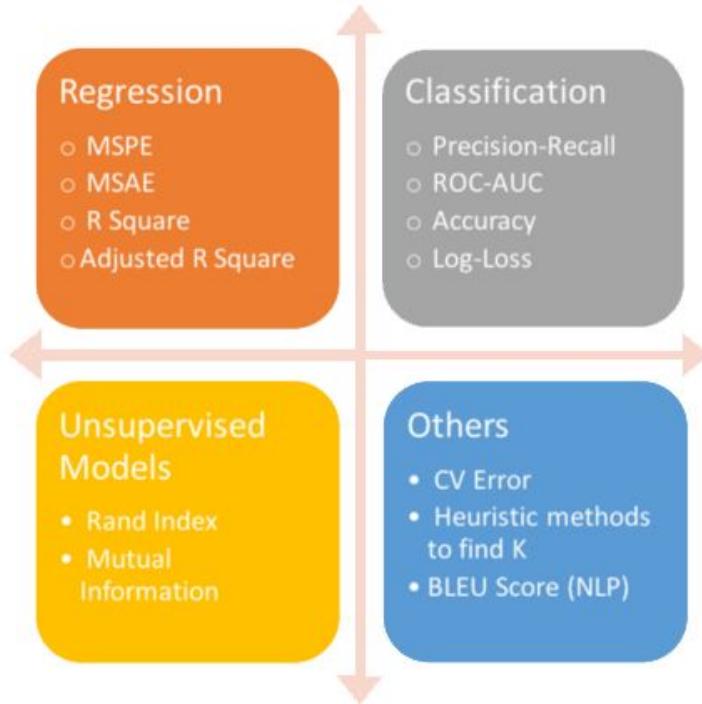
De forma general, cuando estamos entrenando un modelo de DL tendremos tiempo para ir analizando el proceso, ya que dependiendo del número de hiperparámetros y de datos en el set de entrenamiento esto puede durar desde minutos hasta meses.



Una fase importante es poder analizar este entrenamiento aun cuando no se ha finalizado el proceso y así reducir tiempos

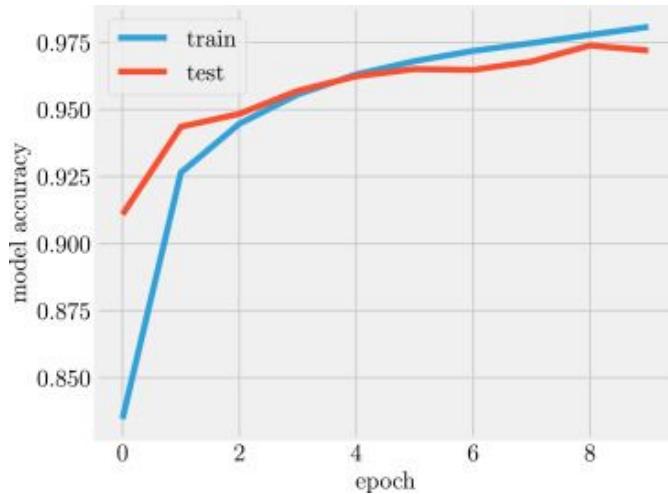
Medidas de desempeño

El primer paso es poder elegir la medida de desempeño correcta, esta está totalmente ligada al tipo de problema y tipo de dato a usar



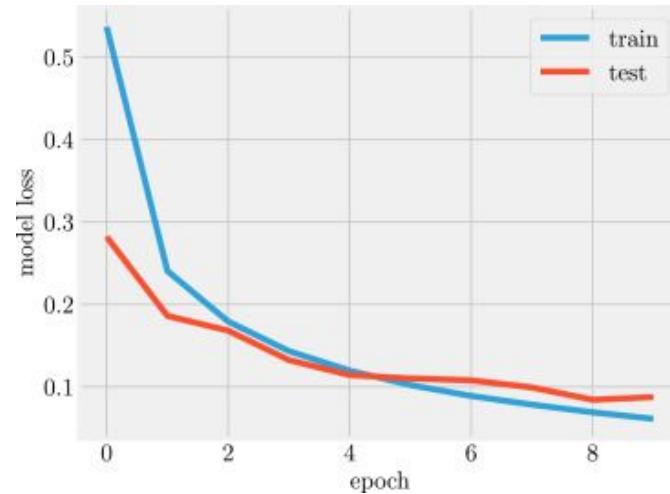
Curvas de aprendizaje

Útiles para monitorear cómo evoluciona el aprendizaje de un modelo, nos permite averiguar si hay subajuste, sobreajuste o un buen ajuste y actuar en consecuencia.



Exactitud (Accuracy)

$$\frac{\text{Predicciones correctas}}{\text{Total de predicciones}}$$

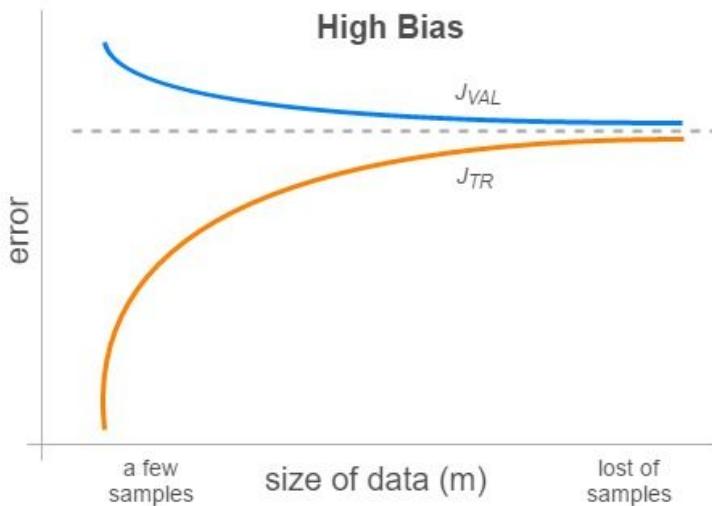


Pérdida (loss)

La diferencia entre el valor predicho por el modelo y el valor verdadero
Depende del algoritmo: MSE, entropía cruzada, IOU, etc

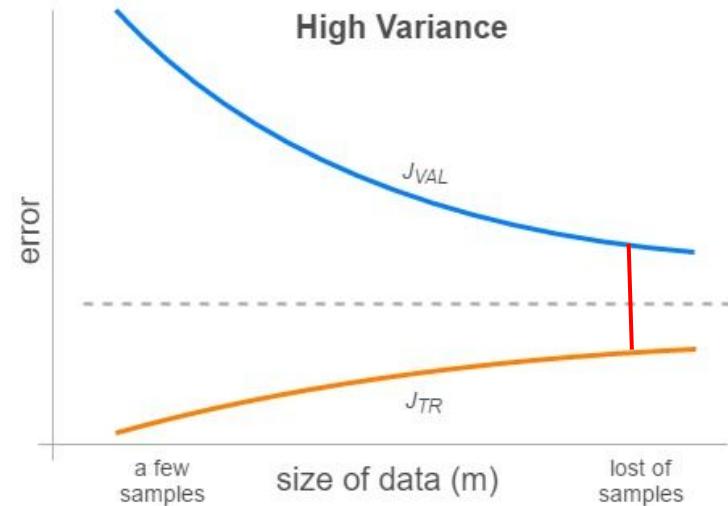
Curvas de aprendizaje

Curvas de aprendizaje: También, nos pueden ser útiles para mostrar la relación entre el error del modelo respecto al tamaño del conjunto de datos.



Sesgo alto:

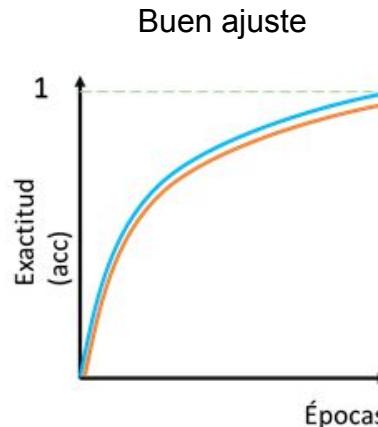
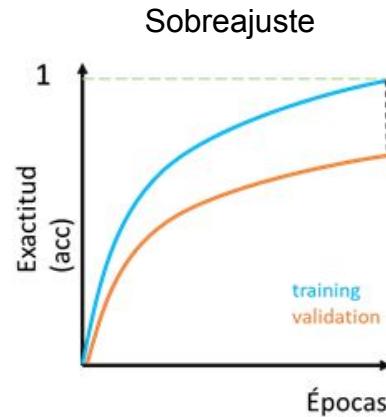
El modelo no se ha ajustado lo suficiente
(subajuste)



Alta varianza:

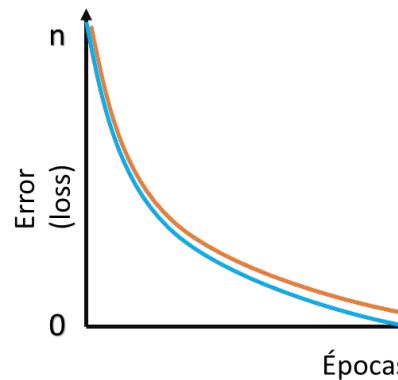
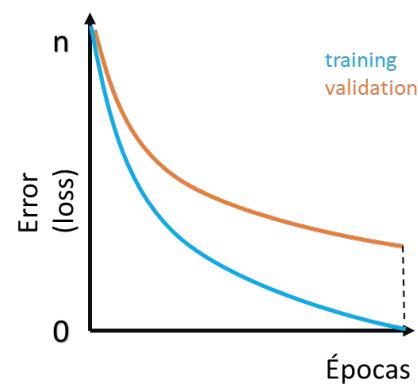
El modelo muestra signos de sobreajuste

Regularización



Son técnicas para reducir el sobreajuste:

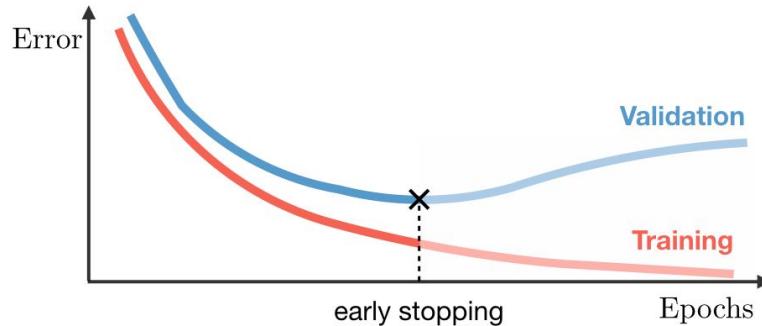
- Early stopping
- Dropout
- L1 y L2
- Elastic net



Regularización

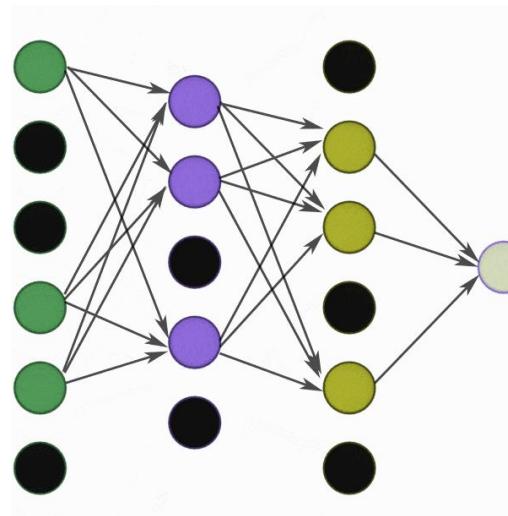
Detención temprana (Early stopping)

Detener el ajuste del modelo cuando empiece a presentar sobreajuste



Dropout (apagado aleatorio de neuronas)

Durante el entrenamiento de una red neuronal artificial, apagar aleatoriamente neuronas para simplificar el modelo



Regularización

En descenso por gradiente, se utilizó MSE como la medida del error (función de costo) $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$

$$J = Err + \lambda C$$

Al agregar términos de regularización a la función de costo, penalizamos que los parámetros del modelo (pesos) tengan valores grandes, es decir; la regularización restringe la capacidad del modelo para ajustarse, evitando el sobreajuste

$$J = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda C$$

λ Parámetro de regularización (hiperparámetro)

$$C = \|\mathbf{w}\|_p = \left(\sum_{i=1}^n |\mathbf{w}_i|^p \right)^{\frac{1}{p}}$$

p-nom:

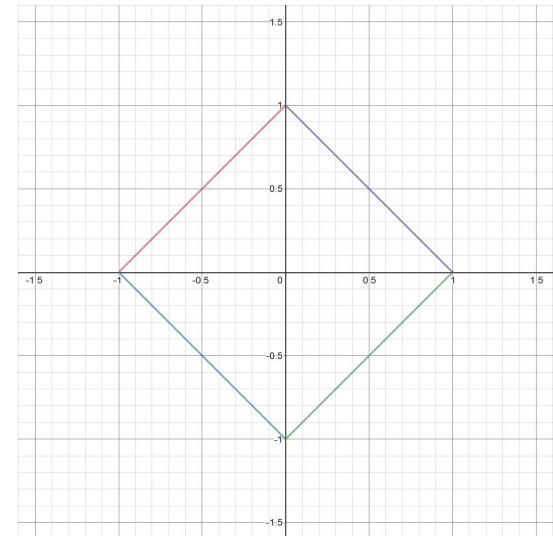
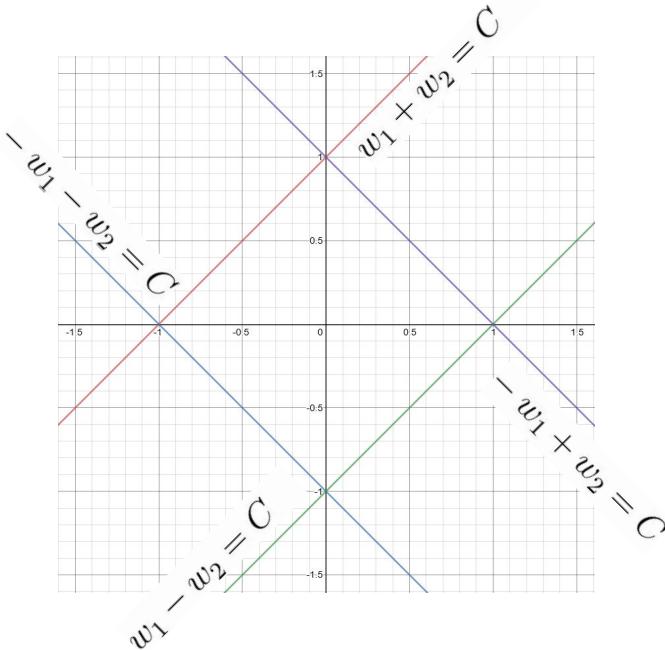
Existen muchos tipos de normas, las más comunes son **L1**(LASSO) y **L2** (Ridge)

La regularización sólo se aplica en el ajuste (entrenamiento), la predicción se hace sin regularización

En un espacio de **2 dimensiones**, dibujamos geométricamente algunas líneas para ilustrar la ecuación

$$C = \begin{cases} w_1 + w_2 \\ w_1 - w_2 \\ -w_1 + w_2 \\ -w_1 - w_2 \end{cases}$$

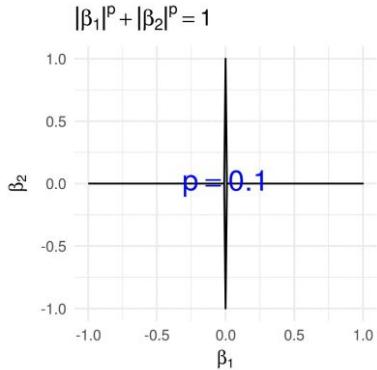
$$s.t. \begin{cases} |w_1| \leq C \\ |w_2| \leq C \end{cases}$$



L1 LASSO (Least Absolute Shrinkage and Selection Operator): se puede aplicar a regresiones lineales, polinómicas, regresión logística, redes neuronales, SVM

Regularización

$$\|\mathbf{w}\|_p = \left(\sum_{i=1}^n \mathbf{w}_i^p \right)^{\frac{1}{p}}$$



La norma L1: LASSO

$$\lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=1}^n |\mathbf{w}_i|$$

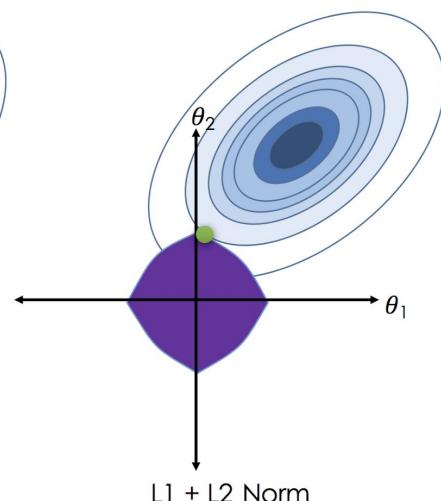
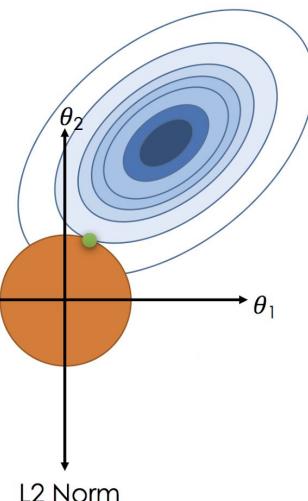
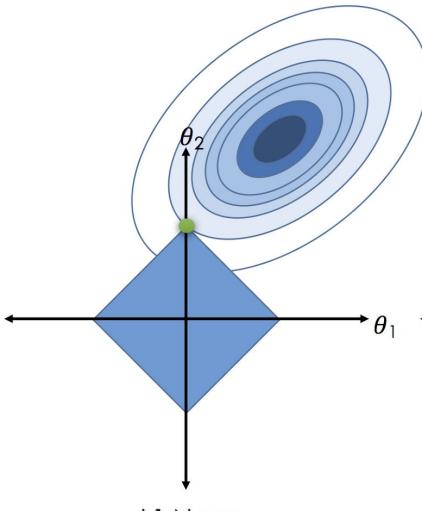
La norma L2: Ridge

$$\lambda \|\mathbf{w}\|_2^2 = \lambda \sum_{i=1}^n \mathbf{w}_i^2$$

L1 + L2: Elastic Net

$$\lambda \left[(1 - \alpha) \|\mathbf{w}\|_1 + \alpha \|\mathbf{w}\|_2^2 \right]$$

$\alpha \in [0, 1]$



$$\sum_{i=1}^n |\mathbf{w}_i| = |w_1| + |w_2| + \cdots + |w_n|$$

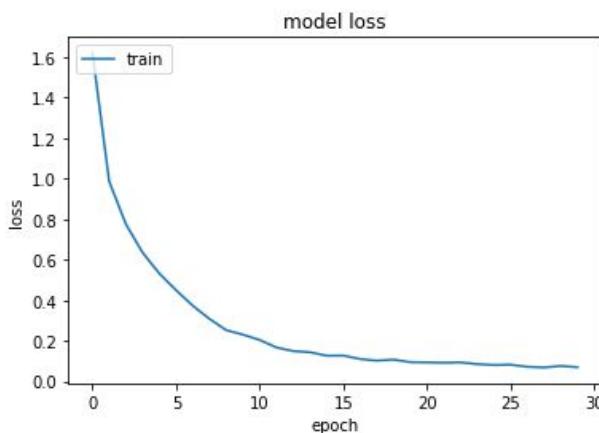
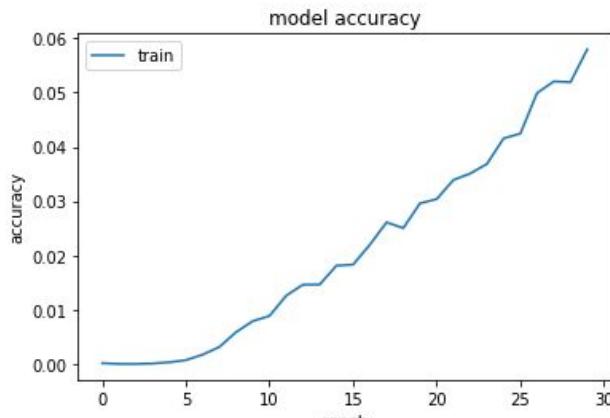
$$\sum_{i=1}^n \mathbf{w}_i^2 = w_1^2 + w_2^2 + \cdots + w_n^2$$

Reto: Complete el Reto_03_entrenamiento para entrenar un modelo simple de DL, verifique los parámetros y composición del modelo, identifique posibles mejoras

Tips:

- Revise nuevamente las técnicas de mejora
- Verifique la estructura del modelo y las posibles ayudas integradas

Resultado Esperado:



Solución:

```
# Librerias

import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.utils import to_categorical

# Mas librerias si fueran necesarias

from tensorflow.keras import Model
from tensorflow.keras.layers import Dropout, Dense, Input, Flatten, Conv2D, MaxPooling2D
```

```
# Observar ejemplos de la base de datos

plt.figure(figsize=(10,5))

for i in range(10):
    plt.subplot(2,5,i+1)
    plt.axis('off')
    plt.title('label: '+str(y_train[i].argmax()))
    plt.imshow(x_train[i])
```

Solución:

```
# Modelo de Deep Learning

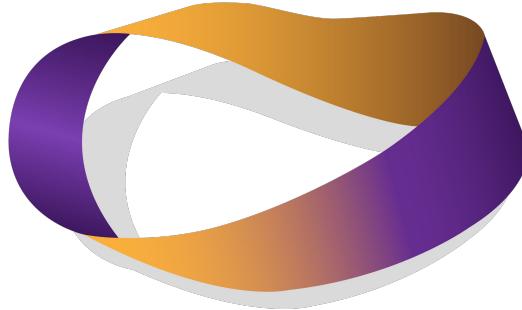
inputs = Input(shape=x_train[0].shape)

x = Conv2D(64, (3,3), strides=(1,1), padding='same', activation='relu')(inputs)
x = Conv2D(128, (3,3), strides=(1,1), padding='same', activation='relu')(x)
x = MaxPooling2D((2,2))(x)
x = Conv2D(256, (3,3), strides=(1,1), padding='same', activation='relu')(x)
x = Conv2D(256, (3,3), strides=(1,1), padding='same', activation='relu')(x)
x = MaxPooling2D((2,2))(x)
x = Flatten()(x)
x = tf.keras.layers.Dense(128, activation=tf.nn.relu)(x)
x = tf.keras.layers.Dense(256, activation=tf.nn.relu)(x)
x = tf.keras.layers.Dense(512, activation=tf.nn.relu)(x)
outputs = tf.keras.layers.Dense(10, activation=tf.nn.softmax)(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

```
model.evaluate(x_test,y_test)
```

Análisis de modelos y validación



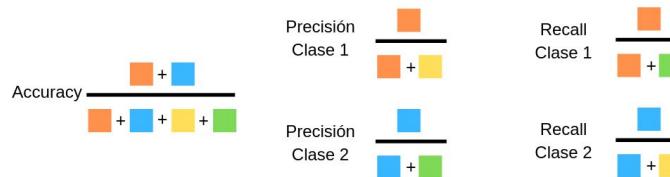
ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Validación de los modelos

Cuando ya hemos seleccionado los modelos, debemos de tener una forma de interpretar sus resultados como buenos o malos, para esto existen diferentes métricas dependiendo del tipo de problema

	Predicción Clase 1	Predicción Clase 2
Valor real Clase 1	Aciertos True Positive Clase 1	Fallos False Positive Clase 2
Valor real Clase 2	Fallos False Positive Clase 1	Aciertos True Positive Clase 2

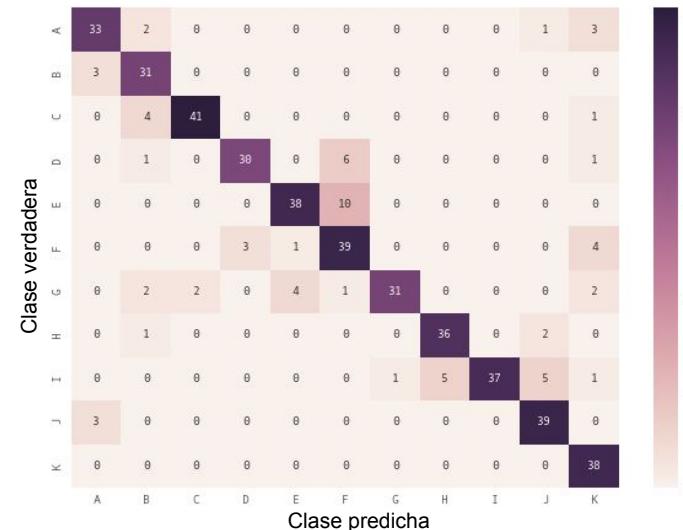


Matriz de confusión: Una matriz de confusión es una matriz de NxN donde N es el número de clases del problema, muestra los falsos positivos y falsos negativos.

- Nos da información acerca del número de veces que las instancias de una clase A son clasificadas como otra clase B.

		Dato predicho													
		Rechazar	Aceptar												
Dato real	Rechazar	No error	Falso positivo												
	Aceptar	Falso negativo	No error												

Biclase



Multiclas

$$\text{Exactitud} = \frac{\text{predicciones correctas}}{\text{Total}}$$

		predictions (output)			
		A	B	C	D
actual class (input)	A	9	1	0	0
	B	1	15	3	1
	C	5	0	24	1
	D	0	4	1	15

TP(True Positive)

A	B	C	D	
A	9	1	0	0
B	1	15	3	1
C	5	0	24	1
D	0	4	1	15

TN_A(True Negative for A)

A	B	C	D	
A	9	1	0	0
B	1	15	3	1
C	5	0	24	1
D	0	4	1	15

TN_D(True Negative for D)

A	B	C	D	
A	9	1	0	0
B	1	15	3	1
C	5	0	24	1
D	0	4	1	15

FP_A(False Positive for A)

A	B	C	D	
A	9	1	0	0
B	1	15	3	1
C	5	0	24	1
D	0	4	1	15

FP_B(False Positive for B)

A	B	C	D	
A	9	1	0	0
B	1	15	3	1
C	5	0	24	1
D	0	4	1	15

FN_A(False Negative for A)

A	B	C	D	
A	9	1	0	0
B	1	15	3	1
C	5	0	24	1
D	0	4	1	15

FN_D(False Negative for D)

Precisión y sensibilidad

- **Precisión (PPV)**: Proporción de las predicciones positivas que son correctas
- **Sensibilidad (Recall o TPR)**: Proporción de las predicciones correctas para la clase positiva

		Dato predicho	
		Rechazar	Aceptar
Dato real	Rechazar	TN	FP
	Aceptar	FN	TP

$$\text{Especificidad} = \frac{TN}{TN + FP}$$

$$\text{Sensibilidad} = \frac{TP}{TP + FN}$$

$$\text{Precisión} = \frac{TP}{TP + FP}$$

$$NPV = \frac{TN}{TN + FN}$$

TPR: True Positive Rate
NPV: Negative Predictive Value
PPV: Positive Predictive Value
TN: True Negative
TP: True Positive
FP: False Positive
FN: False Negative

Precisión y sensibilidad

- F_1 : Es conveniente combinar precisión y recall en una métrica llamada F1-score (Como resultado, el clasificador solo obtendrá un puntaje alto si tanto el recall como la precisión son altos)

		Dato predicho	
		Rechazar	Aceptar
Dato real	Rechazar	TN	FP
	Aceptar	FN	TP

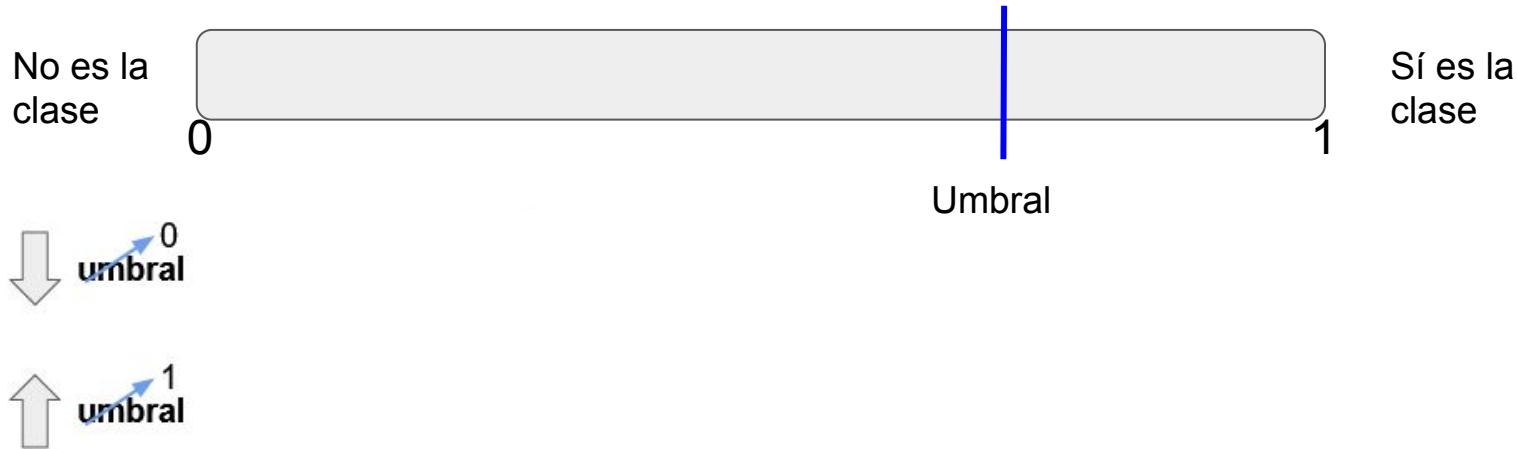
$$F_1 = \frac{2}{\frac{1}{pres} + \frac{1}{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

$$\frac{FN+FP}{2} = FR: \text{False Ratio}$$

Curva ROC

En clasificadores bayesianos o redes neuronales, la salida es una probabilidad de pertenencia a una de las dos clases.

Por lo que se debe fijar un valor de **umbral** para distinguir entre clases



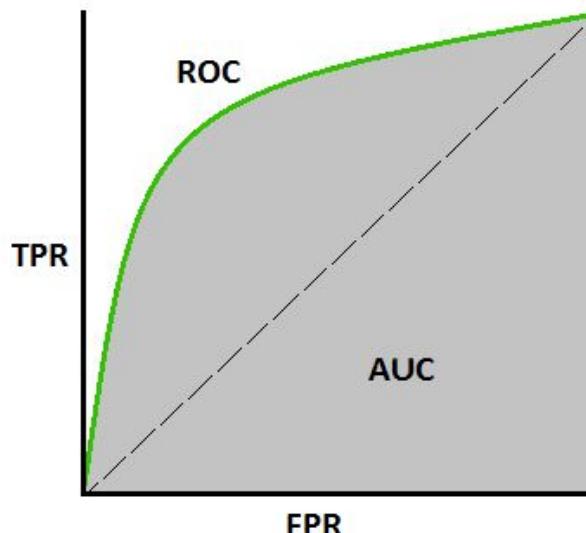
La curva ROC corresponde a los TPR vs FPR para todos los valores de los umbrales

SGDClassifier usa un valor de umbral predeterminado de 0

Curva ROC (Receiver Operating Characteristic): Es una herramienta utilizada comúnmente en clasificadores binarios.

Es la gráfica de la **sensibilidad** (Recall o True Positive Rate) contra el **FPR** (False positive rate) dado por $1 - \text{especificidad}$

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \quad \text{VS} \quad FPR = 1 - \frac{TN}{TN + FP}$$



Sensibilidad= Recall = TPR
TPR: True Positive Rate
FPR: False Positive Rate
AUC: Area Under The Curve

Las curvas ROC son apropiadas cuando se tiene balance entre clases. Da una imagen optimista del modelo en conjuntos de datos desbalanceados.

Curva ROC

El AUC (Área bajo la curva) indica la capacidad del modelo para distinguir entre clases.

A mayor área, mejor es el desempeño el modelo

Interpretación numérica de AUC:

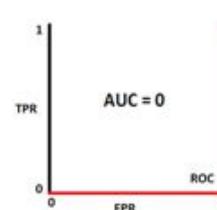
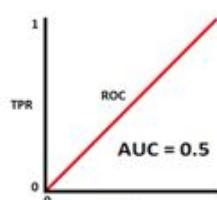
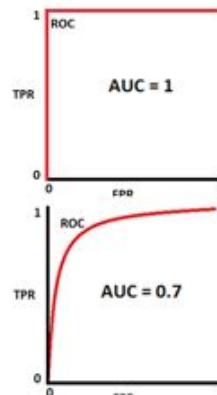
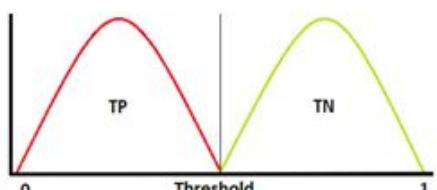
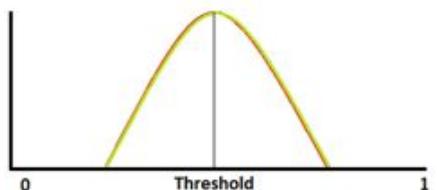
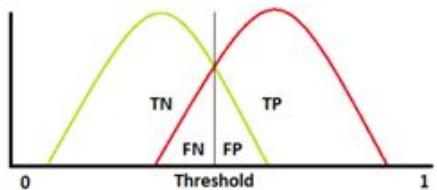
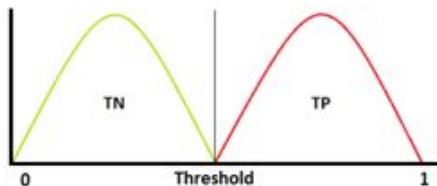
0.9 - 1 = Excelente

0.8 - 0.9 = Bueno

0.7 - 0.8 = Aceptable

0.6 - 0.7 = Pobre

0.5 - 0.6 = Malo



No se equivoca

Confunde algunos

Decide igual que tirar un volado

Lo clasifica todo al revés

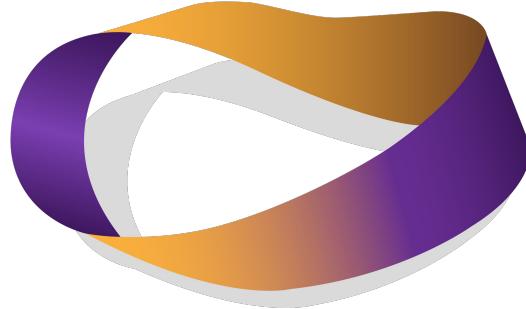
La verdadera validación

Ahora que sabemos algunas métricas que nos permiten validar nuestros modelos necesitamos realizar la verdadera validación



El set de testing es una prueba de datos jamas usada para entrenar que nos mostrará los resultados esperados en producción

Después del entrenamiento



ACTUMLOGOS

DESARROLLANDO HABILIDADES TECNOLÓGICAS

Después de la validación

Como hemos visto durante este módulo, los pasos necesarios para asegurarnos que un sistema que puede trabajar con ML o DL deben ser cuidadosos y llevan una gran cantidad de selecciones, esto puede llegar a ser abrumador si no tenemos el tiempo para probar todas las combinaciones deseadas.

Afortunadamente como hemos visto existen herramientas que nos permitirán automatizar este proceso



Transcripción de la lengua de señas a español

Como una forma de verificar todos los pasos que hemos realizado, trabajaremos en un sistema que nos permita identificar las principales señales realizadas con las manos a un texto



Transcripción de la lengua de señas a español

Como una forma de verificar todos los pasos que necesitamos para un pipeline, trabajaremos en un sistema que nos permita identificar las principales señales realizadas con las manos a un texto

≡ kaggle

+ Create

Home

Competitions

Datasets

Code

Discussions

Learn

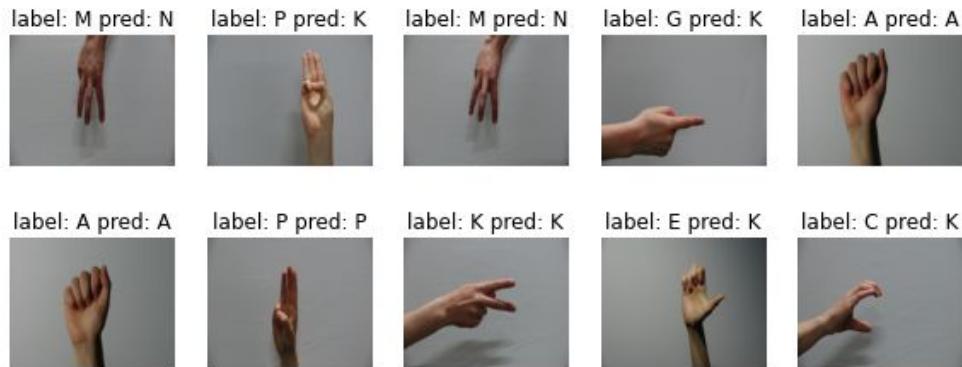
More

Spanish Sign Language Alphabet (Static)

18 inversed static letters from LSE over a white background

Reto: Complete el Reto_04_señas para entrenar un modelo simple de que permita identificar la lengua de señas

Resultado Esperado:



Tips:

- Siga los paso para un pipeline
- Verifique que tipo de dato es
- Verifique el tipo de red correcto para este problema
- Implemente mejoras
- Guarde el modelo

Solución:

```
# Ver un solo dato

import matplotlib.pyplot as plt
from PIL import Image
import numpy as np

dato = Image.open('/content/lenguaje/fondo_blanco/A/DSC00882.JPG')

plt.imshow(dato)
plt.show()

print('Tamaño de la imagen: ', np.array(dato).shape)

# Iterate directory
for path in os.listdir(dir_path):
    letra = path
    letra1 = r'/content/lenguaje/fondo_blanco/' + letra
    for path1 in os.listdir(letra1):
        if os.path.isfile(os.path.join(letra1, path1)):
            count += 1
            img_dir = os.path.join(letra1, path1)
            labels.append(letra)
            data = Image.open(img_dir)
            data1 = data.resize((250,200))
            data2 = np.array(data1)
            x_data.append(data2)
    print('total de imagenes: ', count)
```

Solución:

```
from sklearn import preprocessing  
  
le = preprocessing.LabelEncoder()  
le.fit(labels)  
labels1 = le.transform(labels)
```

```
print('training: ', np.array(x_train).shape)  
print('validation: ', np.array(x_val).shape)  
print('test: ', np.array(x_test).shape)
```

```
# Modelo preentrenado  
  
import tensorflow as tf  
  
base = tf.keras.applications.EfficientNetB7(  
    include_top=False,  
    weights="imagenet",  
    input_tensor=None,  
    input_shape=(200,250,3),  
    pooling=None,  
    classes=len(y_train[0]),  
    classifier_activation="softmax")  
`
```

Solución:

```
modelo = keras.models.Sequential()

modelo.add(base)
modelo.add(layers.Flatten())
modelo.add(layers.Dense(100))
modelo.add(layers.Dense(200))
modelo.add(layers.Dense(len(y_train[0])), activation = 'softmax'))
```

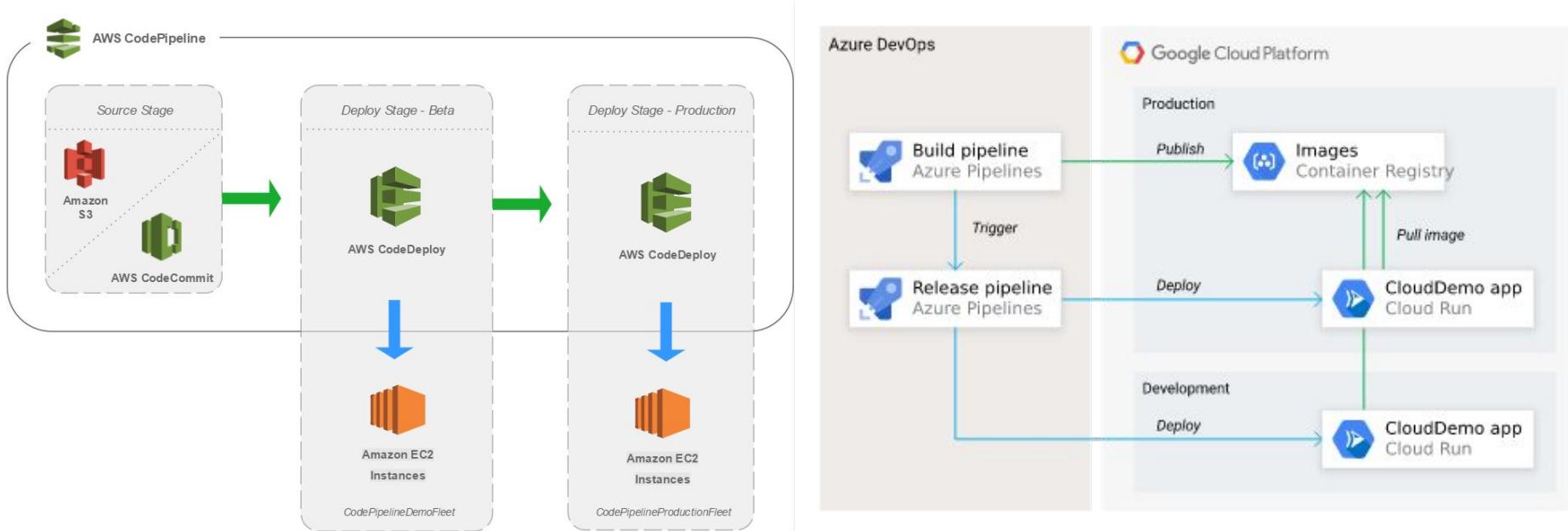
```
modelo.save('modelo_10_epocas.h5')
```

```
resultado = modelo.predict(np.array(x_test))

# Ahora con letras

plt.figure(figsize=(10, 4))
for i in range(10):
    plt.subplot(2,5,i+1)
    plt.imshow(x_test[i])
    yy = le.inverse_transform(np.expand_dims(y_test[i].argmax(),0))[0]
    y_r = le.inverse_transform(np.expand_dims(resultado[i].argmax(),0))[0]
    plt.title('label: '+yy+' pred: '+ y_r)
    plt.axis('off')
```

iPlus ultra! ...ir más allá



Conclusión Final

- La selección correcta de datos, modelos e hiperparámetros nos permite encontrar la mejor propuesta para nuestro problema
- De forma general, debemos de probar varias combinaciones dentro de nuestro sistema para encontrar el mejor resultado posible
- Cada modelo de ML y DL tiene particularidades y campos donde funciona mejor
- La creación de un pipeline nos permite organizar los pasos necesarios para un producto final

Glosario