

What is AI?

IBM Research defines Artificial Intelligence (AI) as Augmented Intelligence, helping experts scale their capabilities as machines do the time-consuming work.

AI learns by creating machine learning models based on provided inputs and desired outputs.

AI should not attempt to replace human experts.

Exists three types of AI:

- **Weak or narrow AI:** It's applied to a specific domain, they cannot learn new tasks, making decision based on programmed algorithms or training data. They only can do a specific task. Examples: language translators, recommendation engines...
- **Strong or Generalized AI:** This AI can operate with independent/unrelated tasks. It can learn new tasks to solve new problems, can perform at a human level intelligence.
- **Super IA or Conscious IA:** Is AI with human-level consciousness, it's the future of IA.

AI-powered advances in speech-to-text technology have made real time transcription a reality.

What is Cognitive technology?

Cognitive computing systems differ from conventional computing systems in that they can:

- Read and interpret unstructured data, understanding not just the meaning of words but also the intent and context in which they are used.
- Reason about problems in a way that humans reason and make decisions.
- Learn over time from their interactions with humans and keep getting smarter

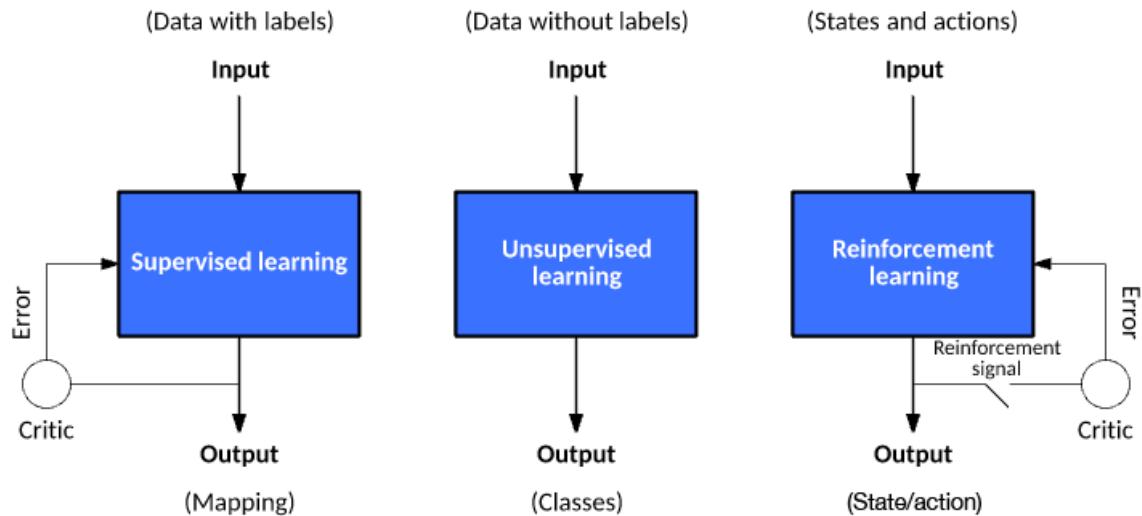
First, **cognitive technology** is often used to **enable innovation and discovery** by understanding new patterns, insightsⁱ and opportunities. Second, it is often used to **optimize operations** to provide better awareness, continuous learning, better forecasting and optimization. Third, to **augment and scale expertise** by capturing and sharing the collective knowledge of the organization. Finally, to **create adaptive, personalized experiences**, including individualized products and services, to better engage customers and meet their needs.

Terminology and related concepts

Artificial intelligence is a branch of computer science dealing with a simulation of intelligent behavior. AI systems will typically demonstrate behaviors associated with **human intelligence** such as planning, learning, reasoning, problem-solving, knowledge representation, perception, motion, and manipulation.

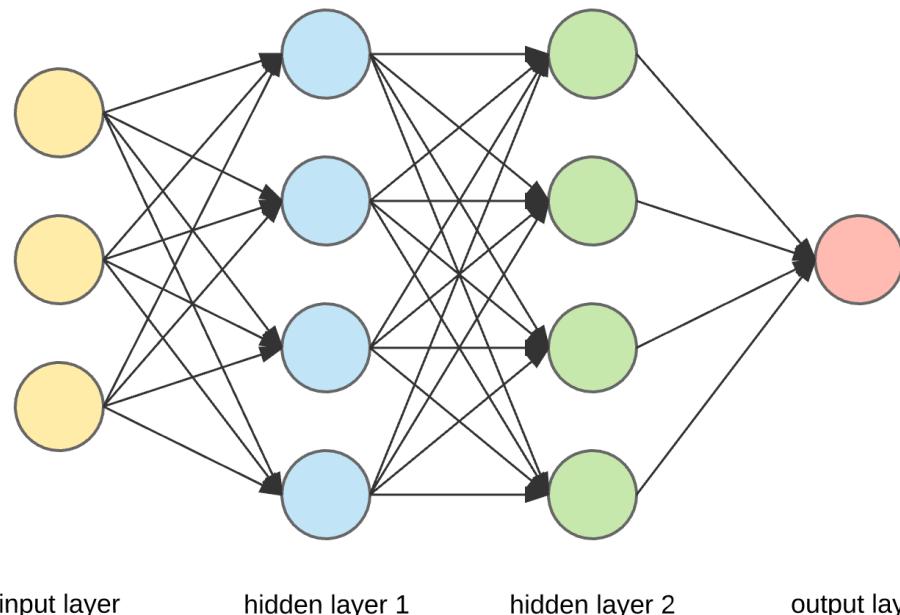
Machine learning is what enables machines to solve problems on their own and make accurate predictions using the provided data.

- **Supervised learning:** Is trained on human-labeled data. You provide input and labels.
- **Unsupervised learning:** Giving unlabeled data and letting it find patterns by itself. You provide input.
- **Reinforcement learning:** Providing with a set of rules and constraints and letting it learn how to achieve its goals. Uses a reward function to penalize bad actions or reward good actions.



Deep learning is a specialized subsetⁱⁱ of Machine Learning that uses layered neural networks to simulate human decision-making. Deep learning algorithms can label and categorize information and identify patterns. It is what enables AI systems to continuously learn on the job, and improve the quality and accuracy of results by determining whether decisions were correct.

A **neural network** in AI is a collection of small computing units called neurons that take incoming data and learn to make decisions over time. Neural networks are often layered deep and are the reason deep learning algorithms become more efficient as the datasets increase in volume, as opposed to other machine learning algorithms that may plateau as data increases.



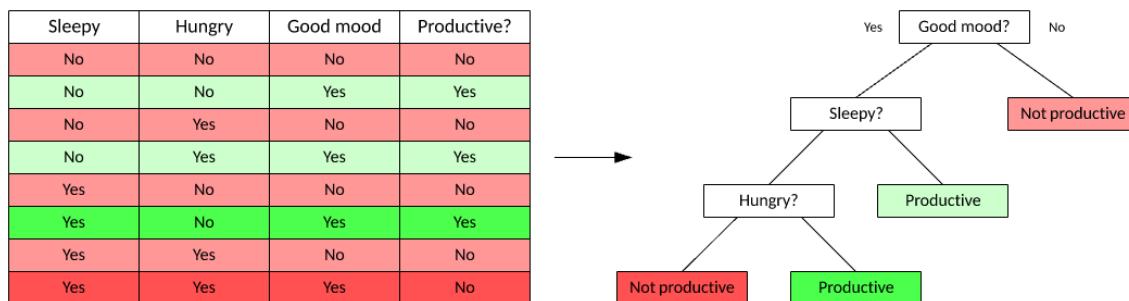
Neural networks are trained from examples rather than being explicitly programmed. **Neural networks** learn through a process called **backpropagation**. **Backpropagation** uses a set of training data that match known inputs to desired outputs. First, the inputs are plugged into the network and outputs are determined. Then, an error function determines how far the given output is from the desired output. Finally, adjustments are made in order to reduce errors.

Types of neural networks:

- **Perceptrons** are the simplest and oldest types of neural networks. They are single-layered neural networks consisting of input nodes connected directly to an output node
- **Convolutional neural networks or CNNs** are multilayer neural networks. **CNNs** are useful in applications such as image processing, video recognition, and natural language processing. A **convolution** is a mathematical operation, where a function is applied to another function and the result is a mixture of the two functions. Convolutions are good at detecting simple structures in an image, and putting those simple features together to construct more complex features.
- **Recurrent neural networks or RNNs**, are recurrent because they perform the same task for every element of a sequence, with prior outputs feeding subsequent stage inputs.

Decision trees

A decision tree is a supervised learning method for classification. Create trees that predict the result of an input vector based on decision rules.



Two types of models exist for decision trees: **classification trees**, where the target variable is a discrete value and the leaves represent class labels (as shown in the example), and **regression trees(search it)**, where the target variable can take continuous values. You use a data set to train the tree, which then builds a model from the data. You can then use the tree for decision-making with unseen data

Types of dataset in machine learning

- The **Training subset** is the data used to train the algorithm.
- The **Validation subset** is used to validate our results and fine-tune the algorithm's parameters.
- The **Testing data** is the data the model has never seen before and used to evaluate how good our model is. We can then indicate how good the model is using terms like, accuracy, precision and recall.

Application areas

- **Natural Language Processing (NLP)** is a subset of **artificial intelligence** that enables computers to understand the meaning of human language, including the intent and context of use.
- **Speech-to-text** enables machines to convert speech to text by identifying **common patterns** in the different pronunciations of a word, mapping new voice samples to corresponding words.
- **Speech Synthesis** enables machines to create natural sounding voice models.

- **Computer Vision** enables machines to identify and differentiate objects in images the same way humans do.

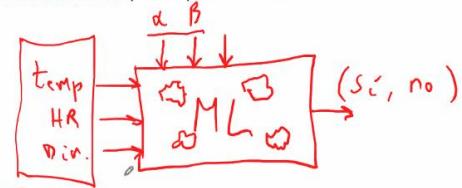
Machine learning

Aprendizaje supervisado

Machine Learning (4/9)

Parámetros:

- ▶ Aprendidos de los datos mediante entrenamiento, **no se establecen manualmente** por el profesional.
- ▶ Contienen el patrón de datos.
- ▶ **Ej)** Coeficientes de regresión lineal.
- ▶ **Ej)** Pesos de la red neuronal.



Entrenamiento.

Hiperparámetros:

- ▶ **Pueden ser configurados manualmente** por el profesional.
- ▶ Se puede ajustar para optimizar el rendimiento del aprendizaje automático.
- ▶ **Ej)** k en el algoritmo KNN.
- ▶ **Ej)** Tasa de aprendizaje en red neuronal.
- ▶ **Ej)** Profundidad máxima en un árbol.

En la salida, si es una variable discreta (es decir que se pueden contar los resultados posibles) se llama **Clasificación**, si se trata de una variable real es decir continua se llama **Regresión**.

Dentro de la caja se consideran parámetros.

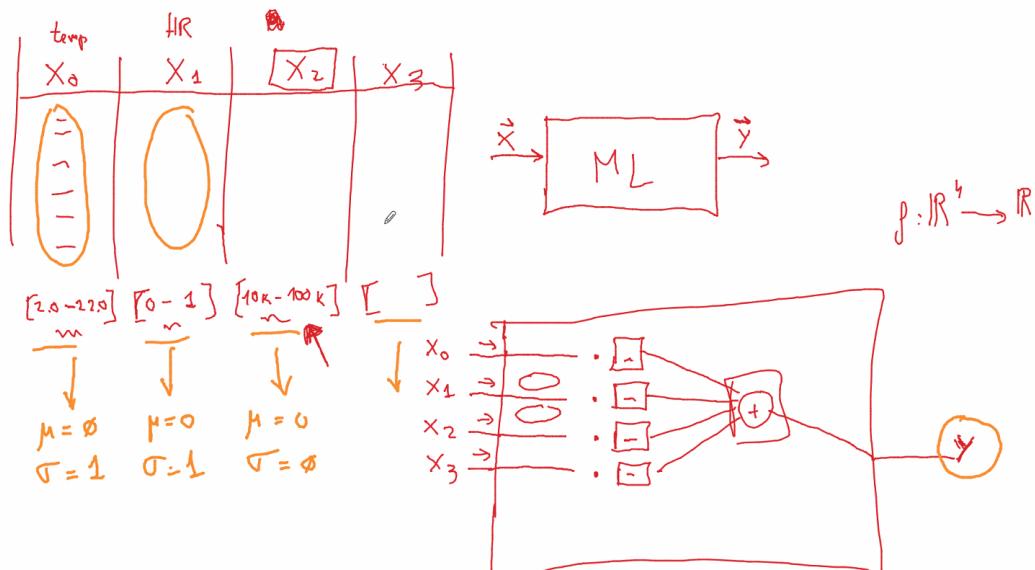
Las features son los valores constantes.

Flujo de aprendizaje

La **Ingeniería de características**, es la fase donde creamos nuestros propios datos.

Regresión lineal con variables continuas

El problema sería cuando los rangos son muy diferentes, en el caso de tener variables continuas.



Entonces sería necesario realizar un estándar o un escalado de las mismas. Solo se puede realizar una **estandarización** si las variables se distribuyen en una normal o si tenemos un dataset gigante, debido al **teorema central del límite**.

Para detectar si nuestras variables son **normales** debemos usar el test de Kolmogorov-Smirnov.

Si no cumplimos estas características para realizar la estandarización, se puede realizar un reescalado, es decir una regla de tres (**minmaxscaler**)

Mientras estén en el mismo **orden de magnitud** funcionará, teniendo en cuenta que por ejemplo podemos poner la variable 1 y la 2 en estandarización y las demás en minmaxscaler.

Los problemas que tiene el minmaxscaler, es que, si nuestros datos no son lineales, podemos tener un mínimo con mucha desviación típica de los datos centrales y con los máximos iguales, para eso tenemos el **robustscaler**.

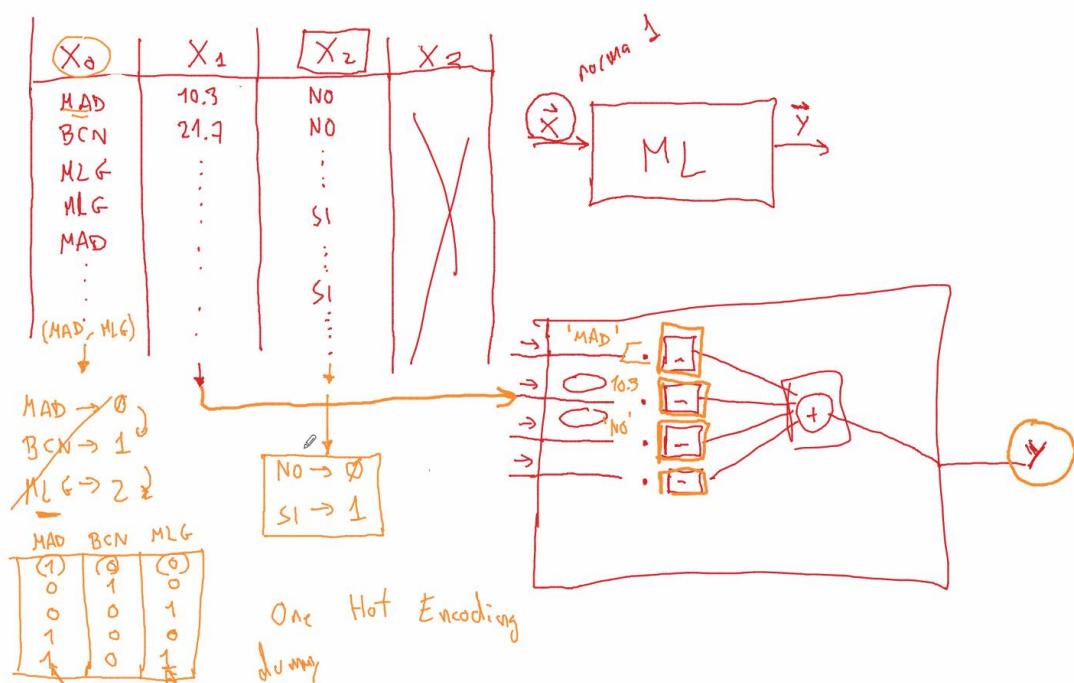
Discretización

Debe ser binaria, de un valor pasamos a dos.

Un ejemplo es la edad, que puede ser una variable continua (es decir 37.5) y pasarla a por ejemplo adolescente, joven adulto, adulto...

Es decir, clasificamos los datos en ciertas labels.

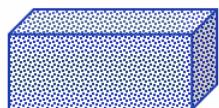
Regresión lineal con variables discretas



Procesamiento de datos

Se divide en distintas fases:

Operaciones:



Limpieza



Transformación/Remodelación



Integración/Unión

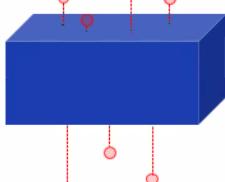
Operaciones:



Escalado/Normalización



Imputación de valores perdidos



Tratamiento atípico

La imputación de valores perdidos, tenemos tres formas de enfrentarnos a los huecos en los datos:

- Si falta algún valor de alguna de sus variables, eliminar la fila, esto obviamente es demasiado drástico, debido a que cada dato cuenta y puede ser un dato mayor
- En imputación de valores perdidos, intentaremos recomponer los datos para dejar de tener el hueco.

En el tratamiento de datos atípicos se debe tomar la decisión de eliminarlos o dejarlos, si decidimos dejarlos el modelo no va a ser capaz de generalizar, el cuál es el objetivo final de ML.

Ingeniería de características

Es el proceso de inventarnos nuevas variables, nuevas columnas, con el dataset que ya tenemos, para así enriquecerlo, teniendo en cuenta que las transformaciones lineales no suelen aportar mucho, como por ejemplo tener, unidades y precio por unidad y saber cual es el costo total.

One hot encoding

Se basa en transformar una serie de variables en características, por una serie de 0s y 1s para evitar dar más peso a unas variables que a otras.

Ingeniería de características (9/9)

Convertir variables numéricas en categóricas :

- ▶ De una variable numérica podemos derivar una categórica que contiene intervalos como valores.
 - ▶ Estos intervalos pueden ser cuantiles o personalizados.
 - ▶ A partir de la variable categórica, las variables ficticias correspondientes se pueden generar como de costumbre.
- Ej) Una variable numérica "Edad" se puede convertir en "Edad_10", "Edad_20", etc.

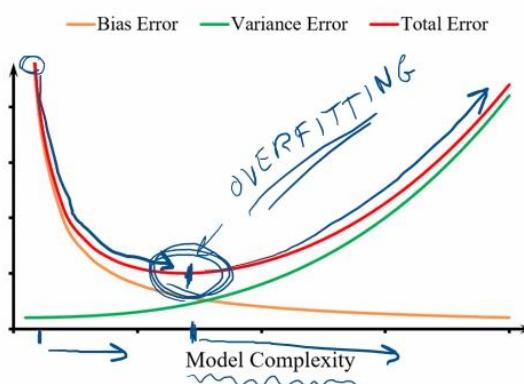
Edad	Edad_10	Edad_20	Edad_30	Edad_40	Edad_50	Edad_60	Edad_70	Edad_80
8	0	0	0	0	0	0	0	0
17	1	0	0	0	0	0	0	0
26	0	1	0	0	0	0	0	0
63	0	0	0	0	0	1	0	0

Overfitting

El overfitting es lo que sucede cuando nuestro proceso se empieza a aprender de memoria el dataset, a poco que le preguntes una cosa ligeramente distinta a lo aprendido, no sabrá solucionarlo.

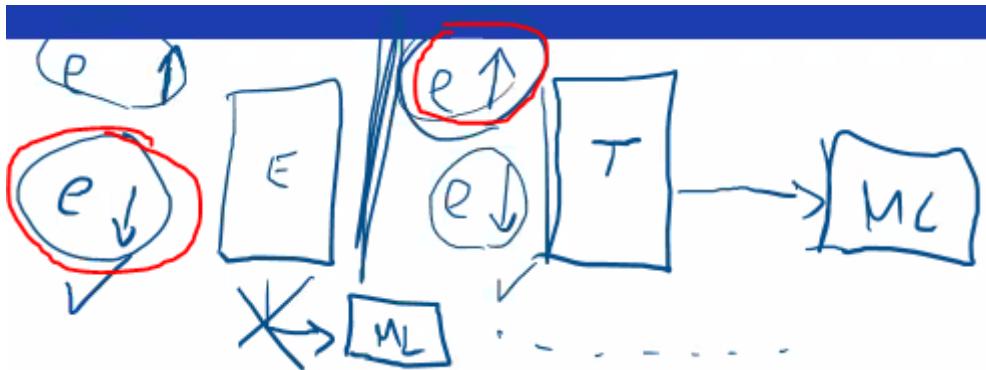
Tipos de error (3/3)

Error total:



- ▶ El objetivo es minimizar el error total = error de sesgo + error de varianza.
- ▶ Se requiere la complejidad suficiente para "optimizar" el modelo.

Podremos detectarlo porque, el modelo ya entrenado obviamente tendrá un error muy bajo si le preguntamos sobre el dataset que ha aprendido, para identificar el overfitting lo pasaremos a un conjunto de datos tipo test, para ver si está generalizando o no. Si el error es bajo en entrenamiento y alto en test, tenemos problemas de overfitting.



Regresión lineal

De manera sencilla, intenta pasar con una línea por la mayoría de puntos.

Sirve para determinar si existe una correlación entre los datos.

Fortalezas es que es un modelo de caja blanca, y es sencillo.

Inconvenientes:

- **Sobreajuste**, Es el problema que surge cuando sobreentrenas un modelo, porque se “memoriza” los datos, por lo tanto, cada vez que sale de su zona de confort no sabrá responder correctamente. – Se arregla con la normalización
- **Outlayers**, son los datos que tenemos que están o muy por encima de la media o muy por debajo, este tipo de modelo es muy sensible a estos datos, por lo tanto es mejor no tenerlos en cuenta en la Regresión lineal – Se arregla con la normalización
- **Escalabilidad**, cuando tenemos datasets más grandes tenemos un problema de coste computacional. – Se arregla con la sgd
- **Linealidad**, este modelo asume que hay una correlación entre la variable objetivo y el resto de variables, casi nunca nos encontraremos con este tipo de relación, se puede comprobar si existe una linealidad entre variables usando – Se arregla con la polinómica

Diagnóstico de regresión lineal

Diagnóstico en regresión lineal (2/14)

Métricas de error:

m features (columnas)

$\underset{\sim}{\text{ML: } \mathbb{R}^m} \rightarrow \underset{\sim}{\mathbb{R}}$

$$\underline{MSE} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

$$\underline{RMSE} = \sqrt{MSE}$$

$$\underline{MAE} = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}$$

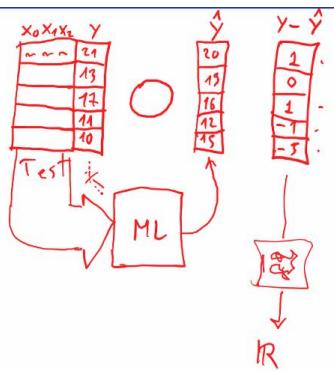
$$\underline{MAPE} = \frac{100}{n} \times \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

$$\cdot MAE = \frac{8}{5} = 1.000 \$$$

$$MSE = \frac{28}{5} \neq \text{unidad}$$

$$\cdot RMSE = \sqrt{MSE} = 1000 \$$$

- ¡Cuanto más pequeño, mejor!

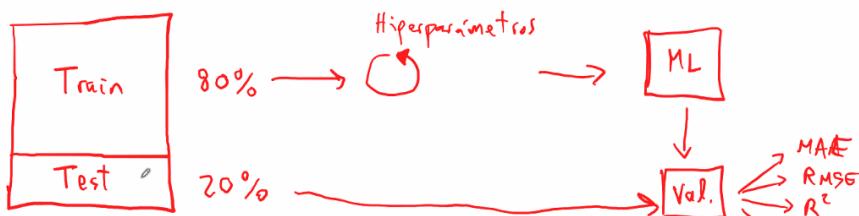


Validaciones

Machine Learning (5/9)

Validación cruzada:

- Los datos deben dividirse en un conjunto de entrenamiento y un conjunto de prueba.
- En principio, el conjunto de prueba debe usarse **solo una vez**. ¡No debe ser reutilizado!
- Si el conjunto de entrenamiento se usa también para la evaluación, los errores pueden ser irrealmente pequeños.
- Para evaluar errores realistas durante el entrenamiento se dividen los datos del entrenamiento en dos.



Validación cruzada y optimización de hiperparámetros:

- Como podemos evaluar repetidamente los errores durante el entrenamiento, también es posible ajustar los hiperparámetros.

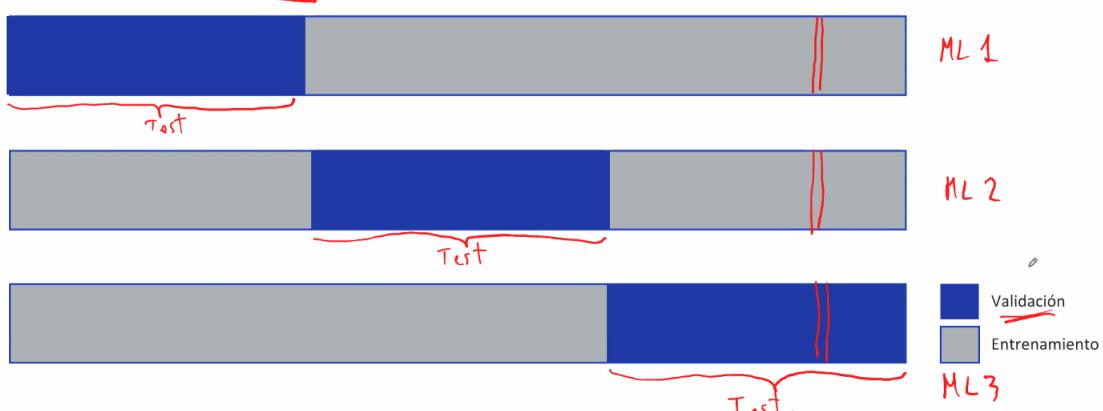
Validación cruzada

k-fold

Machine Learning (7/9)

■ Validación cruzada, método: k-Fold

$K=3$



- Subdividir el conjunto de datos de entrenamiento en k partes iguales. Luego, aplicar secuencialmente.

Son tres modelos iguales desentrenados cada uno independiente del otro, en el cual generaremos de manera aleatoria los sets de entrenamiento, para mitigar así la “buena o mala suerte”.

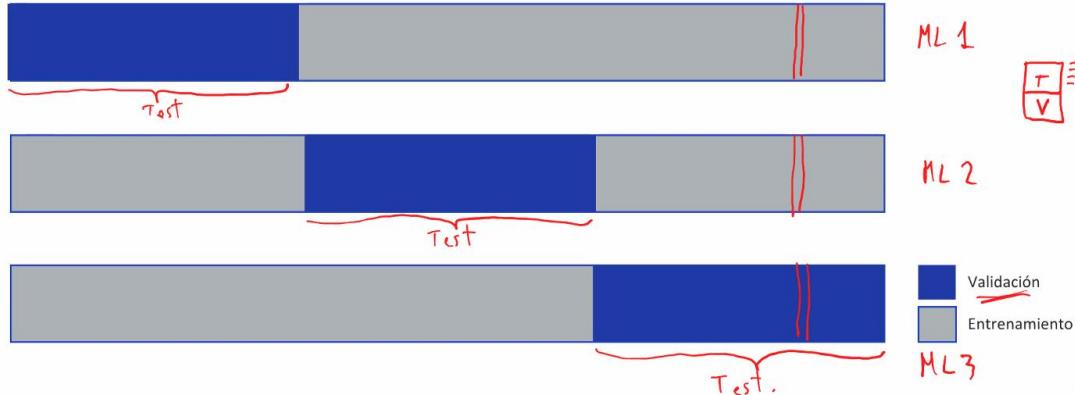
Leave one out

Haces tantos experimentos como observaciones tengas, nos lleva un tiempo de computo elevadísimo.

Machine Learning (7/9)

■ Validación cruzada, método: k-Fold

$K=3$ $K=5$



- Subdividir el conjunto de datos de entrenamiento en k partes iguales. Luego, aplicar secuencialmente.

Ridge regression

Es una variante de la regresión lineal que evita el sobreajuste o el overfitting añadiendo una penalización para menguarla.

Se basa en el alfa, que deberemos de calcular el valor optimo realizando experimentos, y quedándonos con el valor que sea más optimo.

O en inglés **loss function**.

Clasificación

Regresión logística

Conceptos básicos de regresión logística (2/17)

Sobre regresión logística:

- Hay una o más variables explicativas: X_1, X_2, \dots, X_k
- Hay una variable de respuesta: Y
- La variable de respuesta Y es binaria donde los valores posibles son {0,1}, {False, True}, {No, Si}, etc.
- Uno de los algoritmos de clasificación más básicos.

$$\text{regresión: } m: \mathbb{R}^d \rightarrow \mathbb{R}$$

$$\text{claseficación: } n: \mathbb{R}^d \rightarrow \{c_0, \dots, c_K\}$$

Aplicaremos la regresión logística para convertir nuestro modelo de regresión lineal en uno de clasificación, la predicción dará un número entre 0 – 1 que será la probabilidad (p)

$$\text{claseficación Binaria} \quad \{c_0, c_1\} \quad 1-p$$

Conceptos básicos de regresión logística (4/17)

Sobre la regresión logística:

- Las combinaciones lineales de variables X_i son las denominadas "logit" que se indican aquí como S

$$S = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k \quad \text{Reg. lineal.}$$

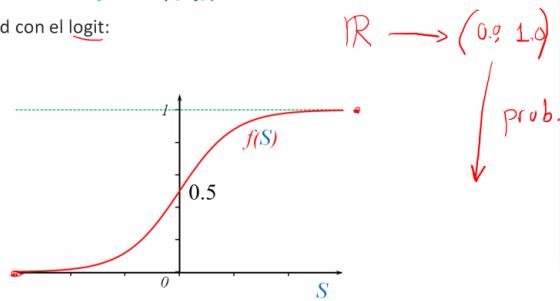
- La probabilidad condicional de que Y sea igual a 1 se denota como $P(Y = 1 | \{x_i\})$.

- La función "Sigmoid" o "Logística" conecta la probabilidad con el logit:

$$m(\vec{x}_i) = \frac{0.8}{1.8}$$

c_0

$$f(S) = \frac{e^S}{1+e^S}$$



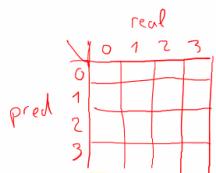
Medidas de rendimiento de regresión logística (1/14)

Matriz de confusión:

Ej)

		Real 0	Real 1
Predicho 0	120	5	FN
Predicho 1	15 → FP	20	

- La matriz de confusión es una tabla de contingencia que cuenta las frecuencias de lo real frente a lo previsto.



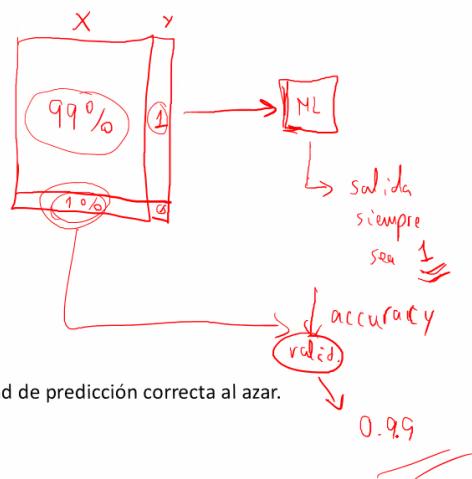
160 muestras en total

Es necesario tener todas las medidas de rendimiento, porque el dataset puede tener un bias importante, como por ejemplo:

Medidas de rendimiento de regresión logística (7/14)

Terminología:

- Exactitud (Accuracy) = $\frac{\text{Número de predicciones correctas}}{\text{Número Total de predicciones}}$
- Sensibilidad = $\frac{\text{Número de 1s correctamente predichos}}{\text{Número Total de 1s}}$
- Especificidad = $\frac{\text{Número de 0s correctamente predichos}}{\text{Número Total de 0s}}$
- Precisión = $\frac{\text{Número de 1s correctamente predicho}}{\text{Número Total de 1s predichos}}$
- Cohen's kappa $\kappa = \frac{\text{Exactitud} - p_e}{1 - p_e}$ donde p_e es la probabilidad de predicción correcta al azar.



Donde el accuracy sería del 99%, pero eso no significa que nuestro modelo sea válido, hay que comprobar todas las demás medidas.

Es importante en clasificación, tener datos iguales de cada tipo de clasificación, es decir si tengo un dataset de 150 datos y tengo una clasificación con 3 opciones, lo ideal sería que fuesen 50,50,50.

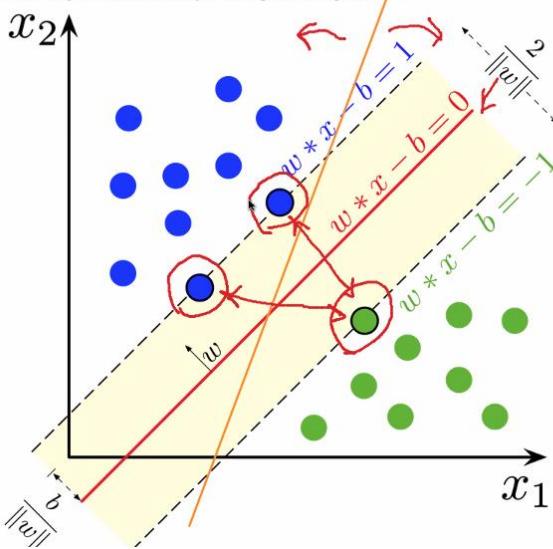
Support vector machines (SVM)

Es un modelo configurable.

No solo encuentra la recta que separa las clases, sino que además maximiza la distancia, es decir busca las observaciones de cada clase que están más cerca una de las otras, y realiza así una frontera.

LinearSVC

Las Máquinas de Vectores Soporte (Support Vector Machines) realizan la clasificación encontrando el hiperplano que maximiza el margen entre las clases presentes en un conjunto de datos. Este concepto, que de entrada puede parecer confuso, es en realidad una idea bastante intuitiva y viene descrita por la siguiente figura:



Las observaciones más cercanas entre sí se llaman **vectores soporte**.

Matemáticamente podemos definir las funciones anteriores con las siguientes expresiones:

$$\hat{y}_i = b + \vec{w} \cdot \vec{x}_i$$

$$cost_0(\hat{y}_i) = \max(0, 1 - \hat{y}_i)$$

$$cost_1(\hat{y}_i) = \max(0, 1 + \hat{y}_i)$$

Y unificarlas del siguiente modo:

$$cost(\hat{y}_i) = y_i \cdot cost_1(\hat{y}_i) + (1 - y_i) \cdot cost_0(\hat{y}_i)$$

$$cost(\hat{y}_i) = y_i \cdot \max(0, 1 + \hat{y}_i) + (1 - y_i) \cdot \max(0, 1 - \hat{y}_i)$$

Si tenemos en cuenta los datos de entrada obtenemos:

$$cost(\vec{x}_i) = y_i \cdot \max(0, 1 + b + \vec{w} \cdot \vec{x}_i) + (1 - y_i) \cdot \max(0, 1 - b - \vec{w} \cdot \vec{x}_i)$$

Y finalmente podemos definir la función de coste de las máquinas de vector soporte añadiendo un término de regularización como:

$$\text{minimizar } loss \left[C \sum_{i=1}^n y_i \cdot \max(0, 1 + b + \vec{w} \cdot \vec{x}_i) + (1 - y_i) \cdot \max(0, 1 - b - \vec{w} \cdot \vec{x}_i) \right] + \frac{1}{2} \sum_{j=1}^m w_j^2$$

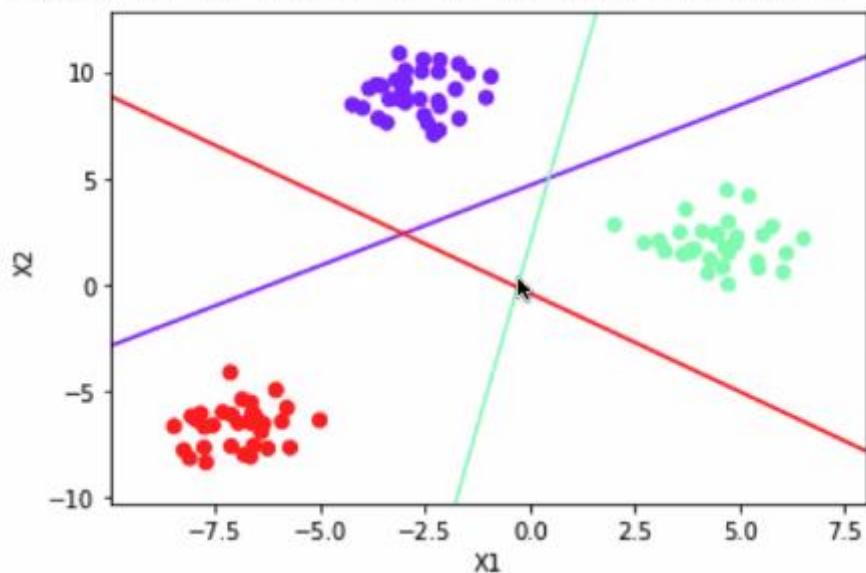
Donde n representa el número de muestras y m el número de características o *features*. Observamos, además, que se ha añadido un hiperparámetro C al modelo. C , al que podemos llamar no-regularización, permite controlar el proceso de aprendizaje: valores grandes de C harán el modelo muy sensible a los outliers mientras que valores bajos de C harán el método muy generalista.

Las Máquinas de Vector Soporte (lineales) se encuentran definidas en `sklearn` en la clase `sklearn.svm.LinearSVC`. Como se observa en su

El parámetro C si lo ponemos muy alto vamos a ser muy sensibles a los outliers, y muy bajo aprenderá peor.

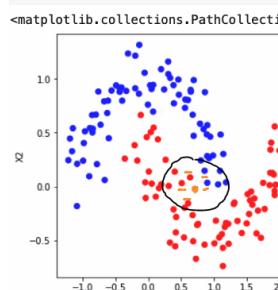
Si existe un punto entre las fronteras, que es un poco terreno de nadie, calculará la probabilidad de que pertenezca a una clase con la cercanía de las fronteras. Si se encuentra solo entre dos fronteras, seguirá calculando la probabilidad de la tercera, pero dará un porcentaje cercano a 0.

[<matplotlib.lines.Line2D at 0x7ff027e1dc10>]



KNN

Calcula los 5 vecinos del punto a predecir, y según la mayoría clasificará ese nuevo punto de una manera o de otra.



K es un kip
K es 5
5 vecinos = [3 rojos y 2 azules]

El método como tal no hace un proceso de entrenar, ya que solo se guarda la matriz de puntos del entrenamiento, cuando le pidamos predecir un punto se fijará en los de alrededor.

Naïve Bayes Classifier

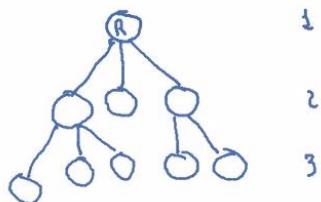
Se hace la suposición de que las variables implicadas no están relacionadas entre sí, es lo que se llama variables independientes.

Árboles de decisión

Hasta ahora todos los modelos que hemos visto necesitaban que las variables estuvieran estandarizadas, normalizadas, que fuesen numéricas...

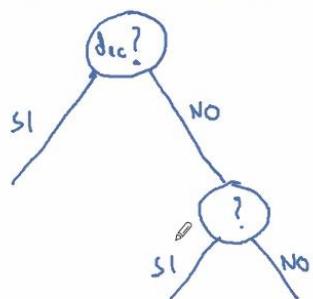
Este modelo NO necesita cumplir con estas restricciones.

Árbol (comput)



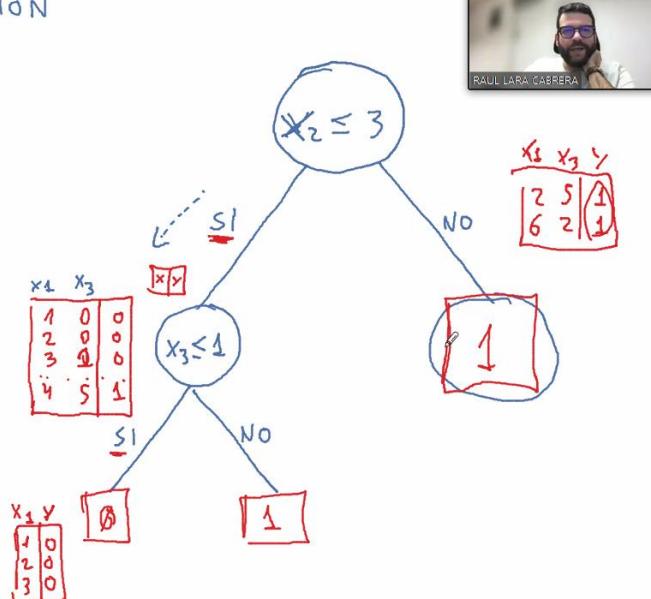
1
2
3

Árbol de decisión



ARBOLES DE DECISIÓN

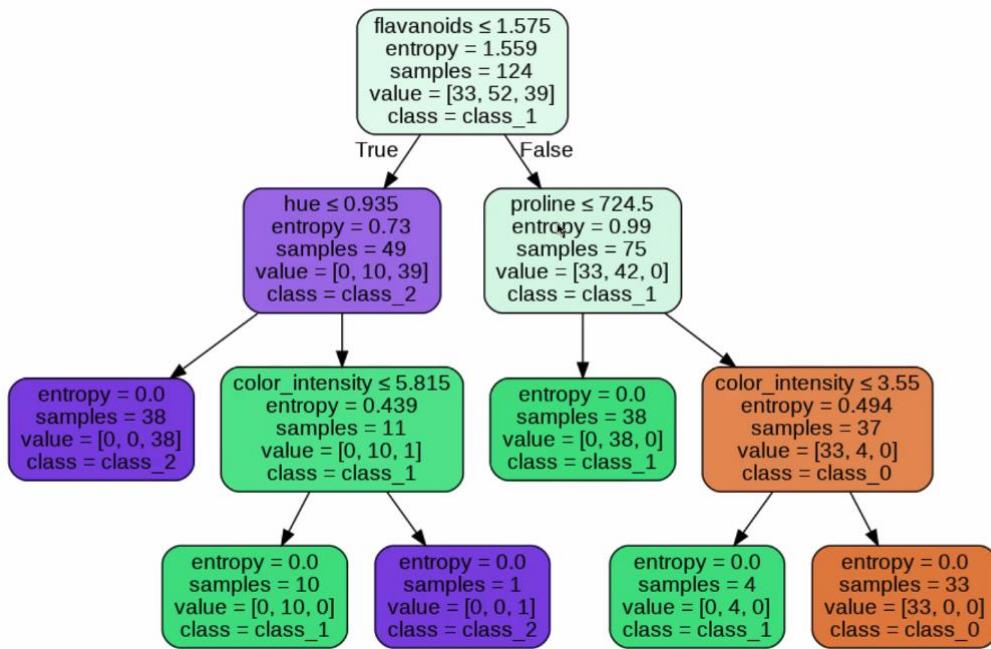
X_1	X_2	X_3	Y
1	2	0	0
2	3	0	0
3	3	1	0
2	7	5	1
4	1	5	1
6	4	2	1



Este modelo es extremadamente propenso a cometer overfitting.

Es un proceso altamente explicativo:

⇒



La **entropía** es la incertidumbre digamos, perdemos entropía cada vez que obtenemos información.

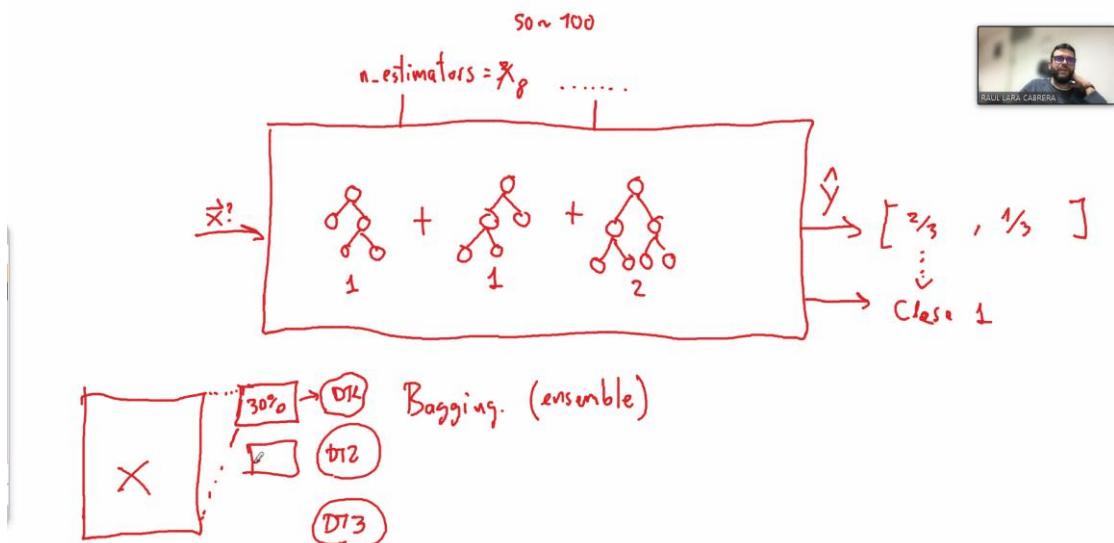
Samples nos indica el número de total filas en nuestro dataset.

Value[clase_0, clase_1, clase_2] → Es decir que ese array nos demuestra cuantas samples de cada tipo hay en ese fragmento del dataset.

Random fotest

Examina muchos árboles de decisión que tengan overfitting, se realiza un agrupamiento final por mayoría en el resto de árboles de decisión, es decir si quiero calcular que x parámetro sean que clase, si el 80% decide que es clase_1, obtendremos ese valor como correcto.

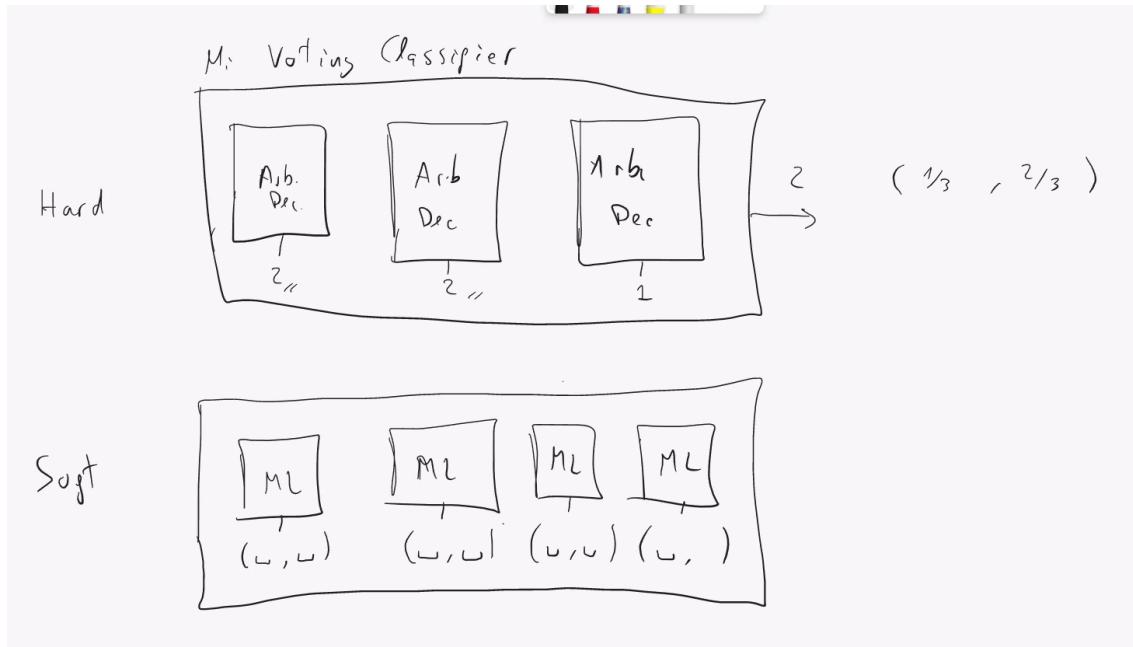
Cada árbol se entrena con un conjunto aleatorio de nuestro dataset de entrenamiento.



Ensembles

Están formados por N algoritmos de clasificación.

Voting Classifier

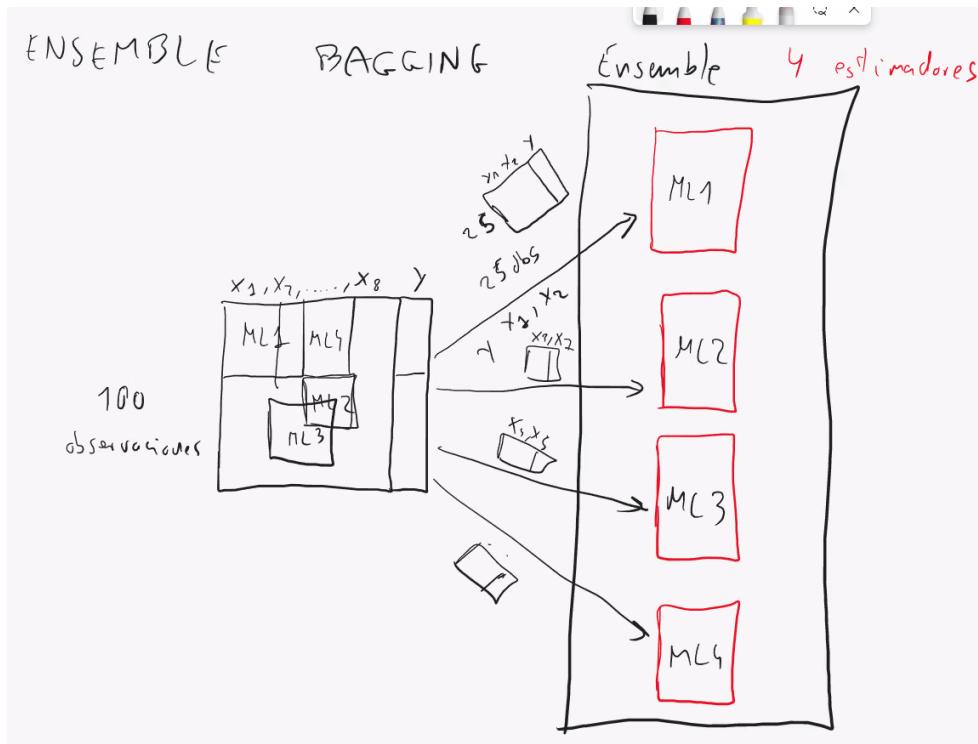


El **soft** se pueden combinar distintos métodos de ML (Regresión Lineal, Random forest..)

Normalmente es más interesante utilizar **soft** voting más que hard voting.

Bagging

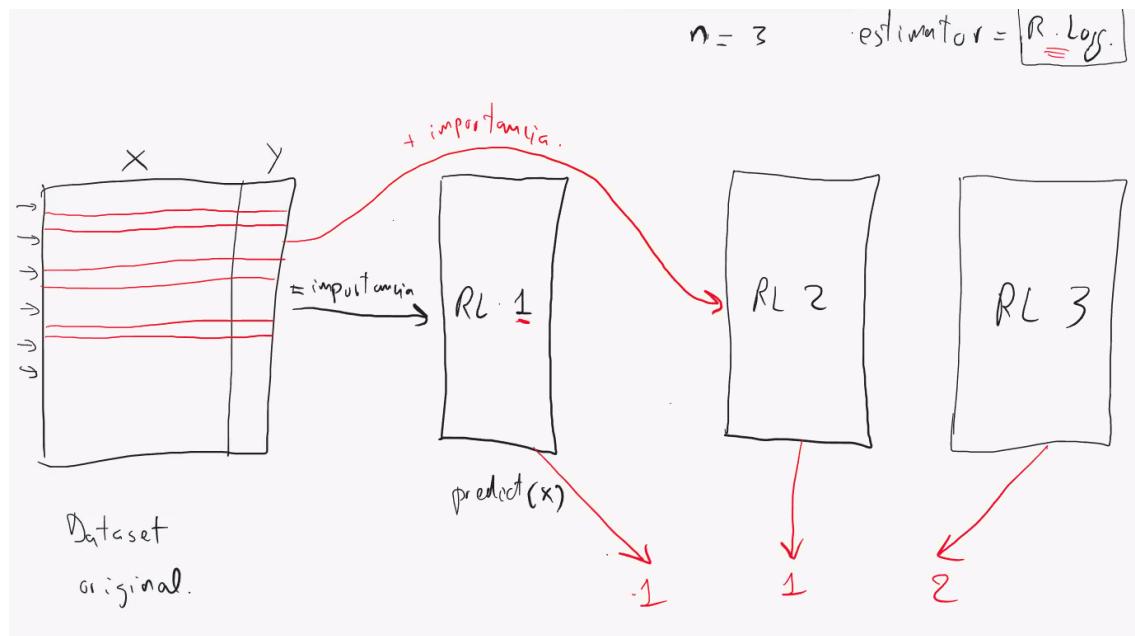
Desordena las filas y para cada algoritmo de clasificación dividirá los datos que tenemos entre ellos, además selecciona aleatoriamente las variables que se les pasa.



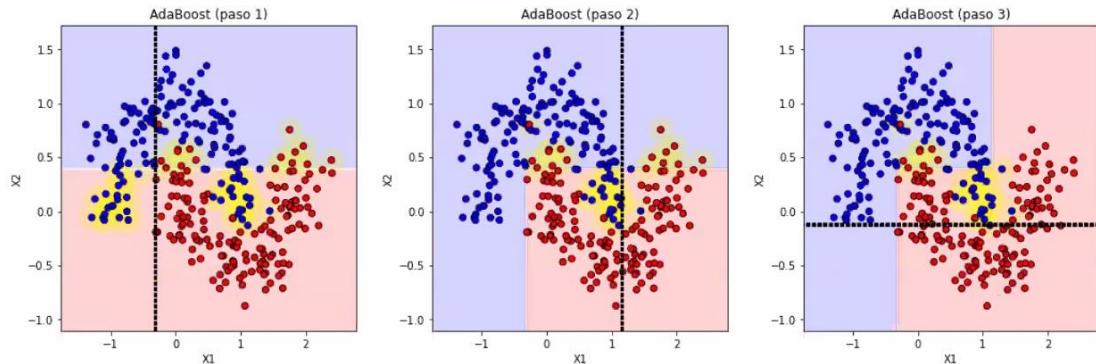
El modelo interno tiene que ser el mismo repetido por el número de estimadores.

Boosting

El modelo aprende de lo que falla el anterior

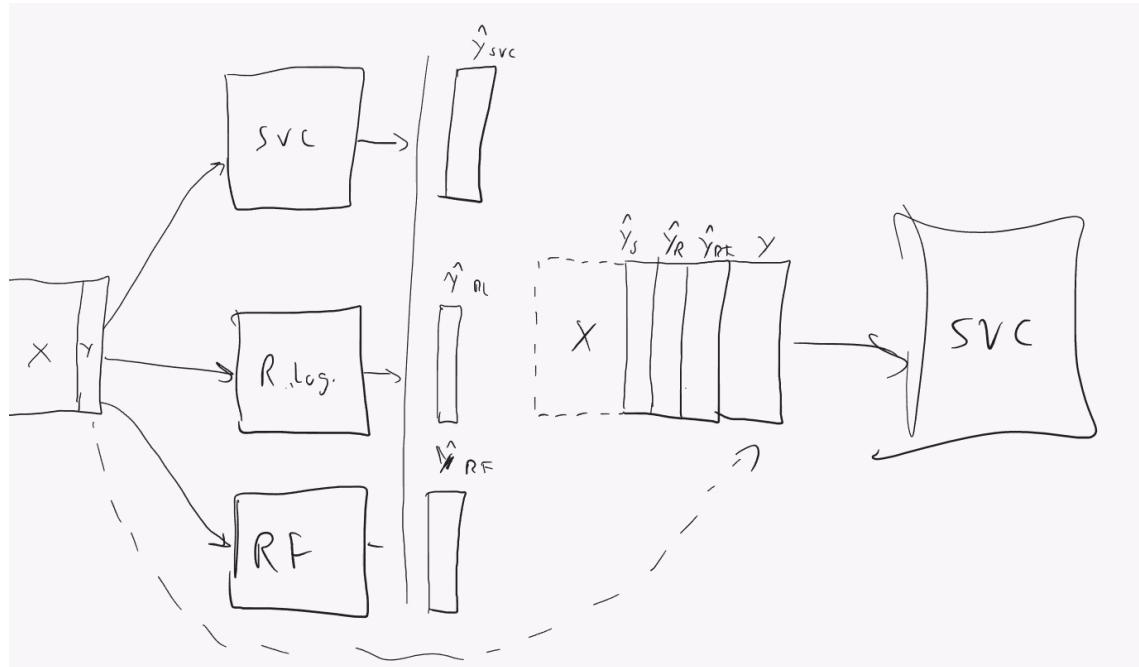


Aquí podemos ver por ejemplo que se va nutriendo de cada modelo a partir de lo que ha fallado en los anteriores.



Stacked

Entrenas un clasificador cuya entrada son las salidas de otros clasificadores entrenados previamente.



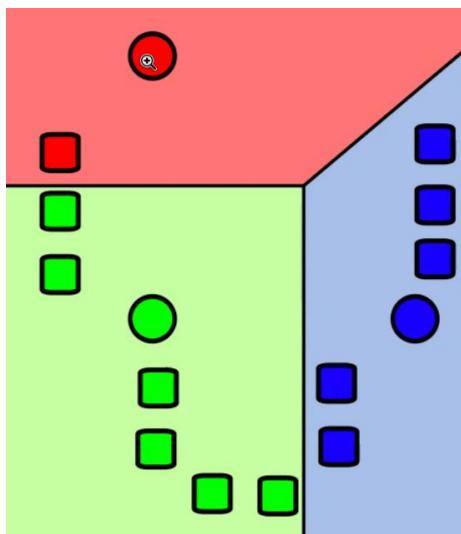
Unsupervised learning

K-Means

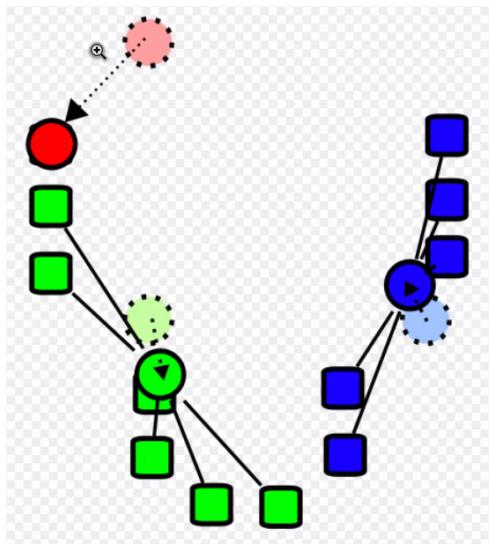
Es un algoritmo lineal, separa por un plano. Es el algoritmo más básico y el que mejores resultados da.

En K-Means se tiene que especificar de antemano la cantidad de grupos que tenemos que dividir el dataset. Este sería nuestro **hiperparametro**, el cual se le llama K.

Primero este método lo que hace es asociar cada observación con su media más cercana, siendo las medias los círculos. Estos centroides (es así como se llaman los puntos), se colocan aleatoriamente escogiendo datos de nuestro dataset.

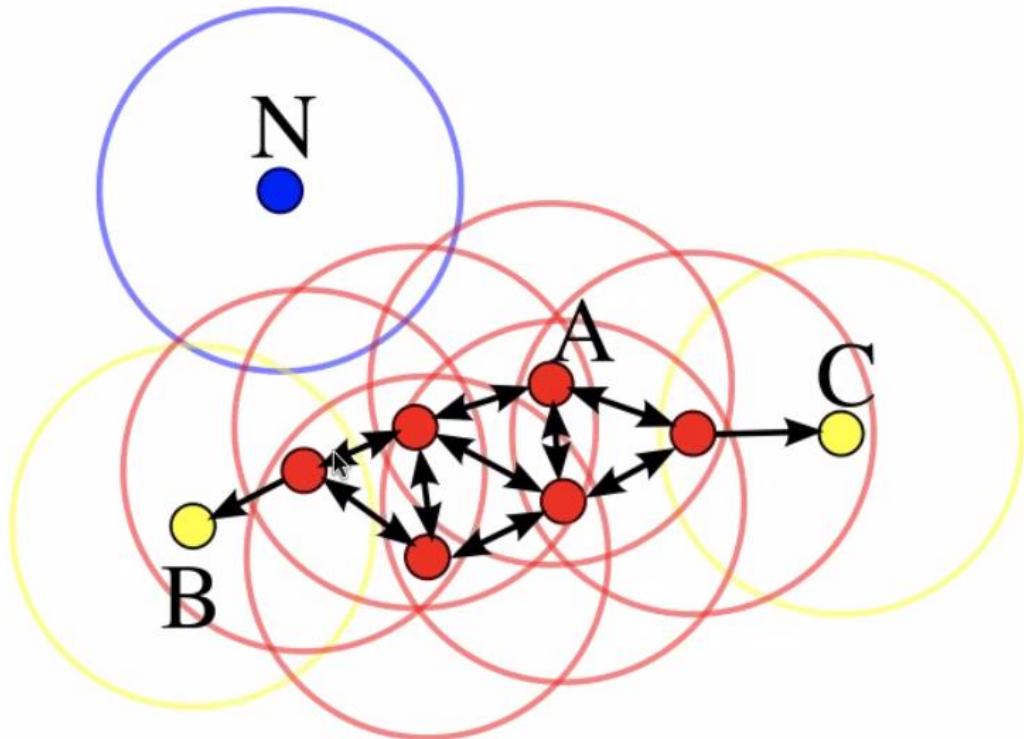


Después de esto, las medias se mueven a una media de las observaciones asociadas a ellas. Colocándose así en un sitio más acertado cada vez. Se repite este proceso n veces.



DBSCAN

Este algoritmo tiene dos hiperparametros, el radio del círculo y el número mínimo de puntos que debe de haber dentro de ese radio para considerarse de la misma clase.



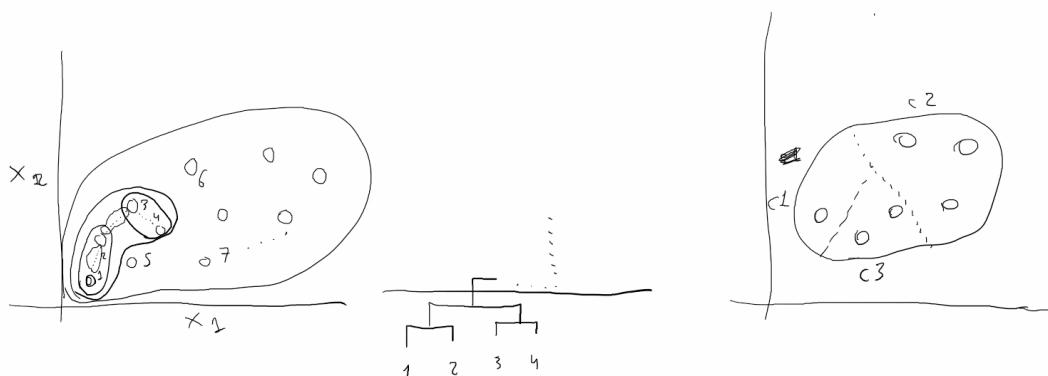
En este caso N sería un outlayer. Es decir un punto descartado normalmente porque se encuentra muy desplazado del computo global de puntos.

Aglomerativo y divisivo



AGLOMERATIVO (sklearn)

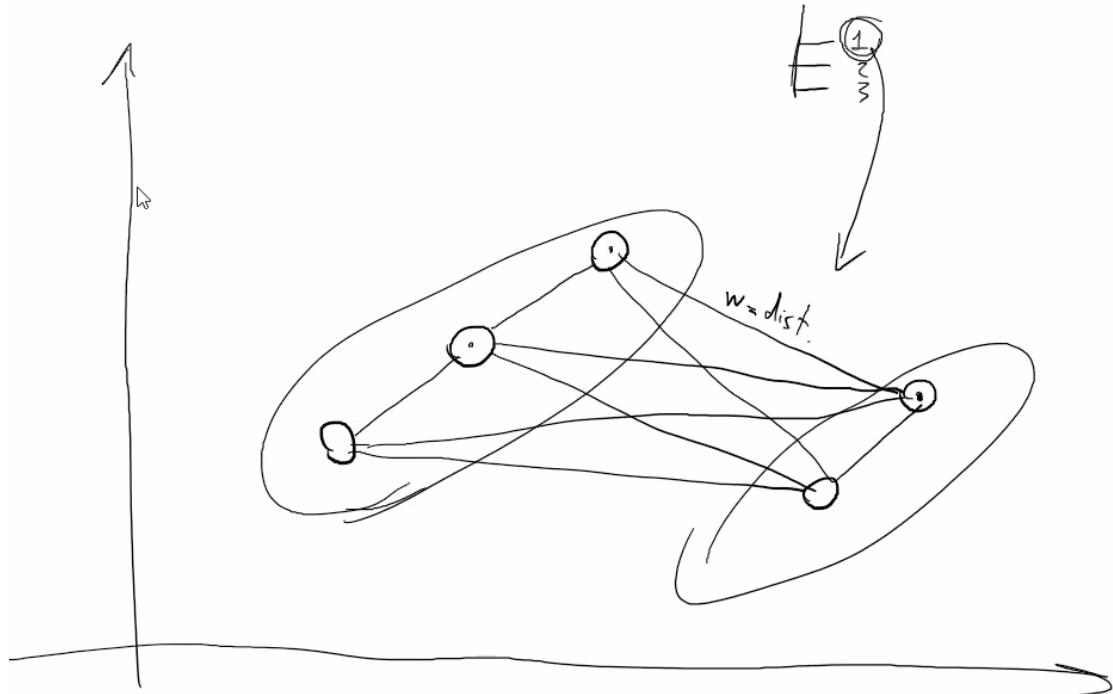
DIVISIVO (m0)



Aglomerativo también se llama Bottom-up, aglutina aquellos puntos que estén más cercanos en clusters.

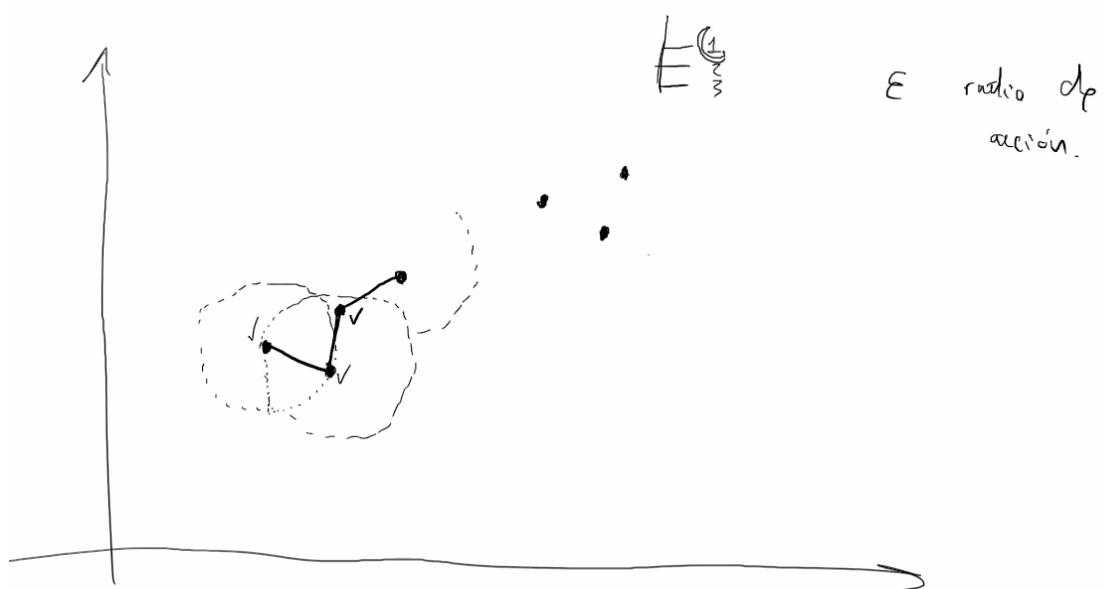
Grafos

CLUSTERING \rightarrow PARTICIONADO
DE
GRAFOS



CLUSTERING \rightarrow PARTICIONADO

DE
GRAFOS



PNL (Procesamiento de lenguaje natural)

Corpus, dentro del PNL es el conjunto de datos. En texto. En esta última parte se diferencia del dataset, ya que el dataset es numérico.

Corpus

Conjunto de documentos = observaciones

Corpus:

- Se refiere al conjunto de datos de texto sujetos a análisis.
- a) Cuerpo sin procesar: datos de texto almacenados en una base de datos.
- b) Cuerpo etiquetado: datos de texto donde las palabras y frases han sido etiquetadas de acuerdo con un modelo.



Stopwords, son aquellas palabras vacías que no contienen información semántica de uso general.

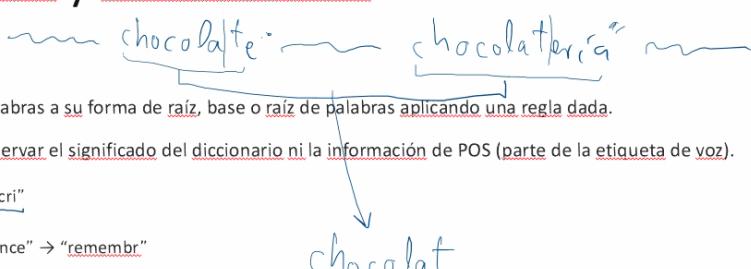
Stemming, reduce las palabras a la raíz.

Stemming.

NLTK

Derivación y Lematización

Derivación:



- Reduce las palabras a su forma de raíz, base o raíz de palabras aplicando una regla dada.

- Puede no preservar el significado del diccionario ni la información de POS (parte de la etiqueta de voz).

Ej) "cried" → "cri"

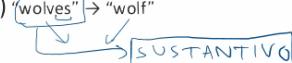
Ej) "remembrance" → "remembr"

Lematización:

- Reduce las palabras a su "lema" al eliminar las terminaciones de inflexión.

- Conserva el significado del diccionario y la información POS (parte de la etiqueta de voz).

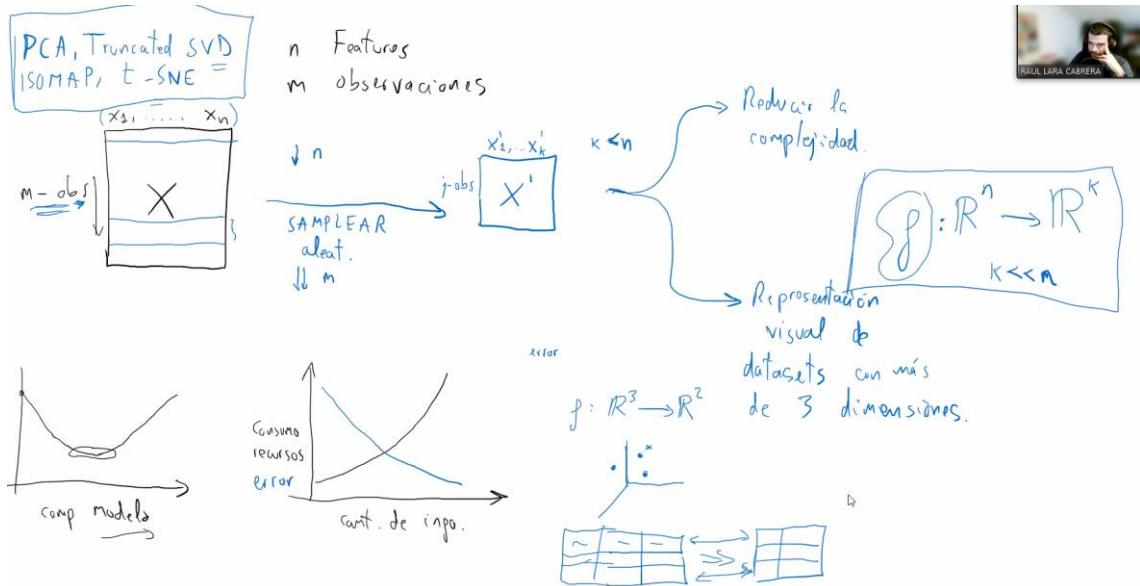
Ej) "wolves" → "wolf"



P

wolf

Reducción de dimensiones

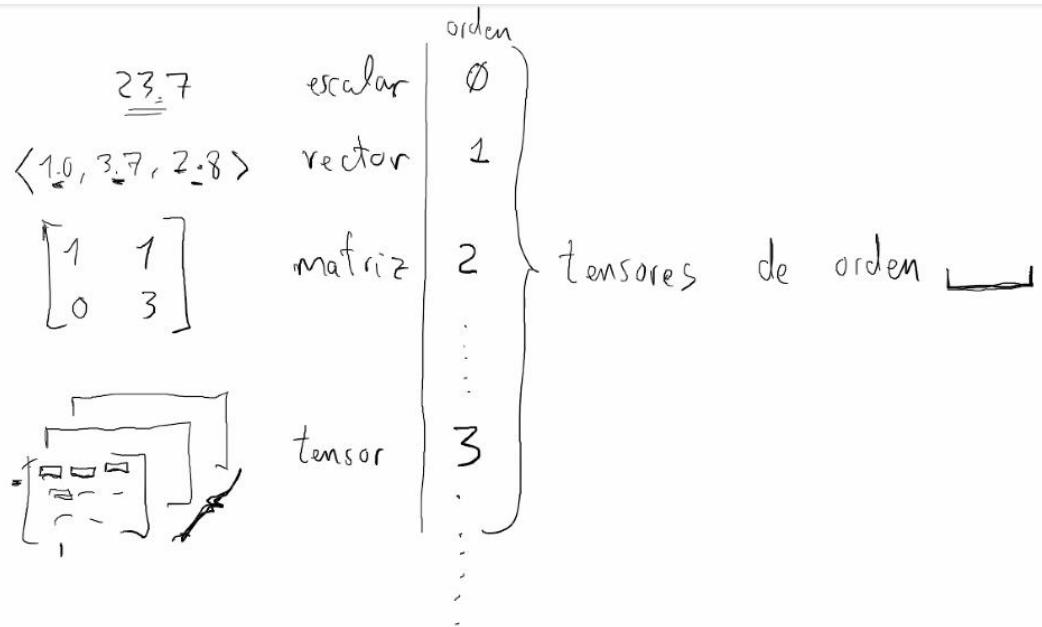


Se consideran aprendizaje no supervisado ya que con el dataset que vamos a trabajar no está etiquetado, aunque lo este, esa etiqueta no se podrá tocar, ya que nunca querremos reducir la salida.

Deep learning

Tensores

Es una estructura de datos donde se pueden tener n dimensiones, igual que los arrays.



Los tensores tienen un orden y unas dimensiones.

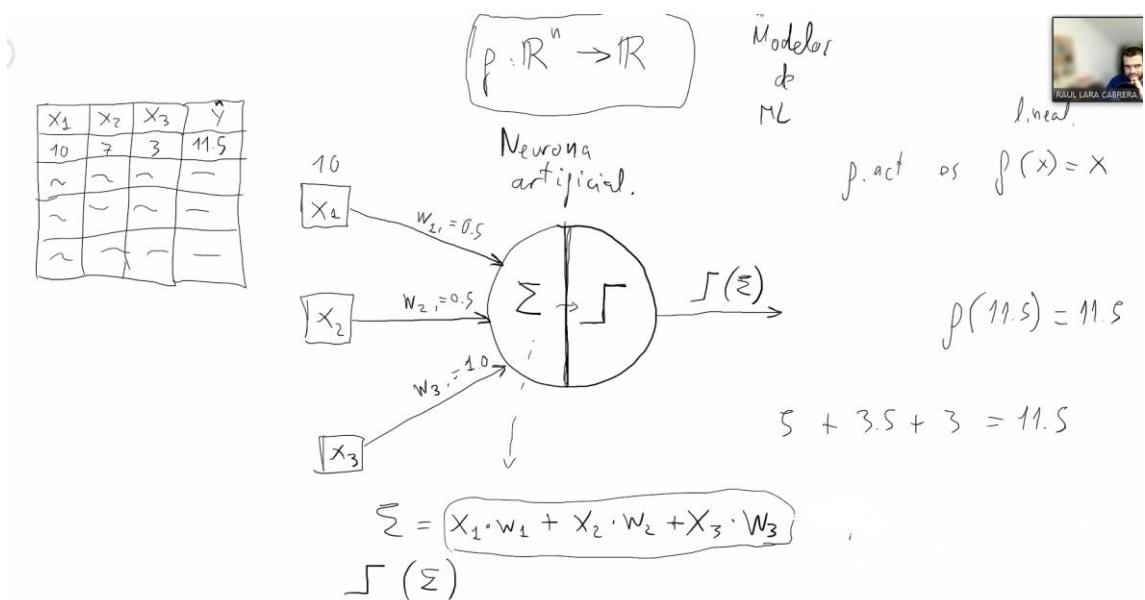
Variables de Tensorflow

Se corresponden a los parámetros que se introducen cuando entrenas un modelo. Estas variables se ajustan automáticamente con el entrenamiento.

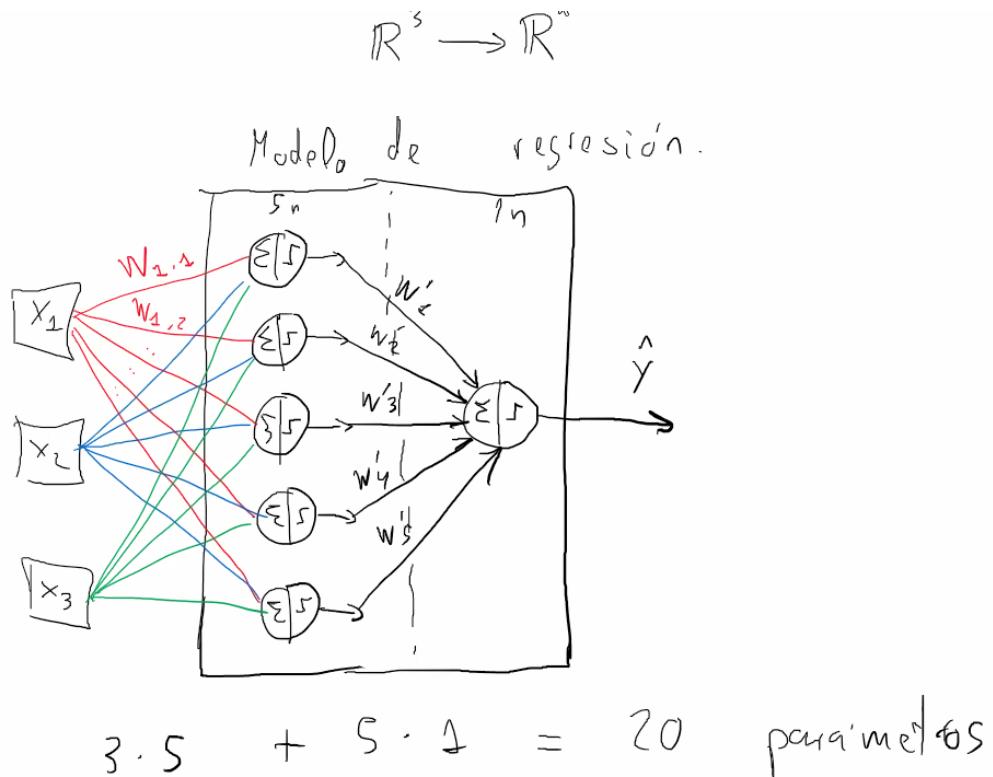
Redes neuronales

Una neurona artificial es una unidad de computo.

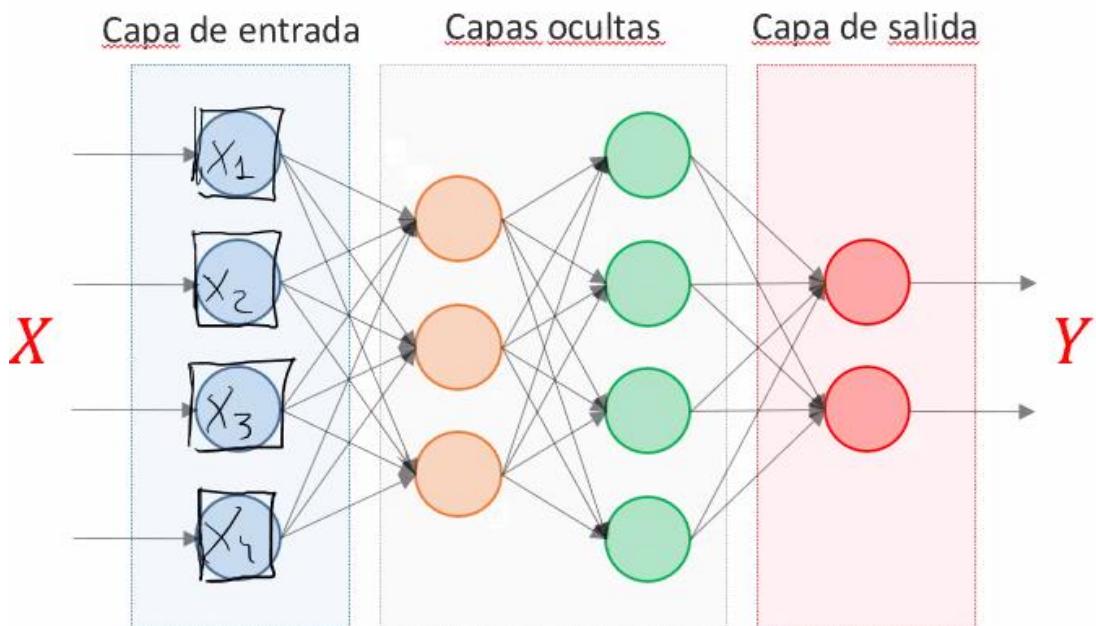
Una única neurona representa una regresión lineal:



Aquí tenemos una red neuronal Feed-forward, porque de la entrada a la salida va todo en el mismo sentido.



Son 20 parámetros debido a que multiplicamos el numero de entradas por el numero de neuronas en cada nivel.



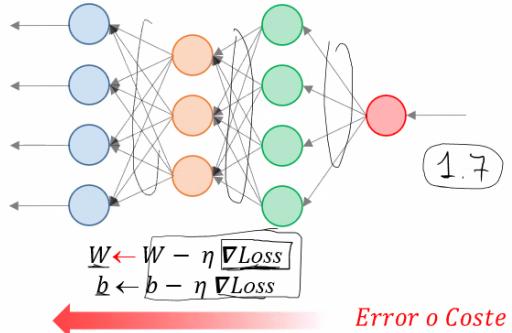
En el caso de problemas de clasificación, tendremos una neurona de salida por cada tipo posible de clasificación, es decir si es un binario (0,1) habrá dos de salida.

A todas las neuronas de la misma capa se le establece la misma función de activación.

Las redes neuronales se entrenan con **backpropagation** a través de las reglas de actualización del descenso de gradiente.

- | Entrenamiento ANN: propagación hacia atrás, backpropagation.
4) Propagar el error hacia atrás y actualizar los parámetros mediante el algoritmo de descenso de gradiente. Repetir desde 1).

Ej)



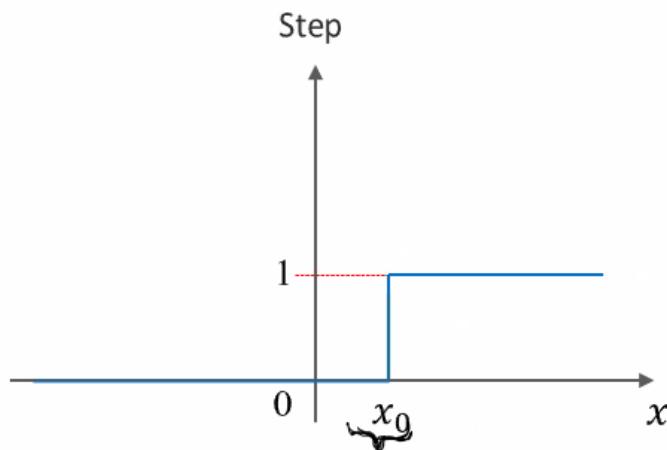
Lo hace la librería por nosotros.

Funciones de activación

Capas distintas pueden tener funciones de activación distintas.

Umbral lienal o paso

Lo que hace es que la neurona solo tenga dos posibles salidas, es decir se discretiza en 0 o 1.



Se debe especificar como hipermámetro el umbral al cual pasa de 0 a 1.

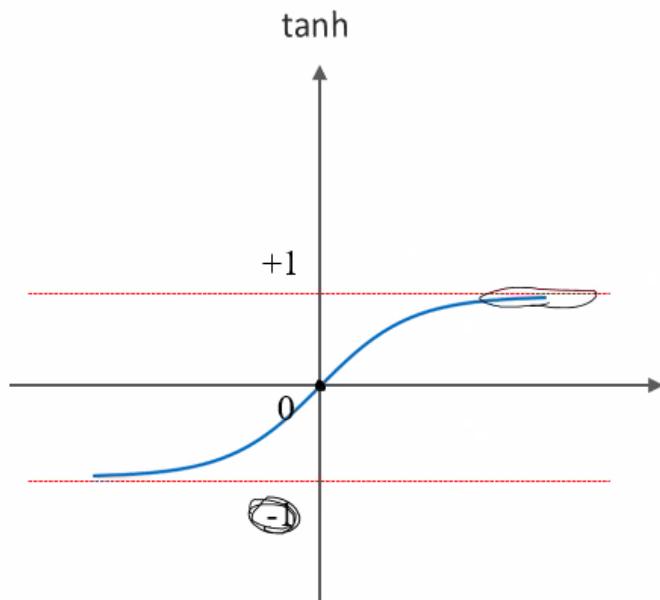
Sigmoide

Cualquier valor de entrada lo vamos a comprimir entre los valores 0 – 1, es decir se convierte a una salida probabilística.

Se debe evitar usar esta función de activación para capas intermedias, aunque en capas finales se recomienda usar en casos de clasificación binaria.

Tangente hiperbólica o tanh

El rango esta entre -1 y +1, utilizaremos



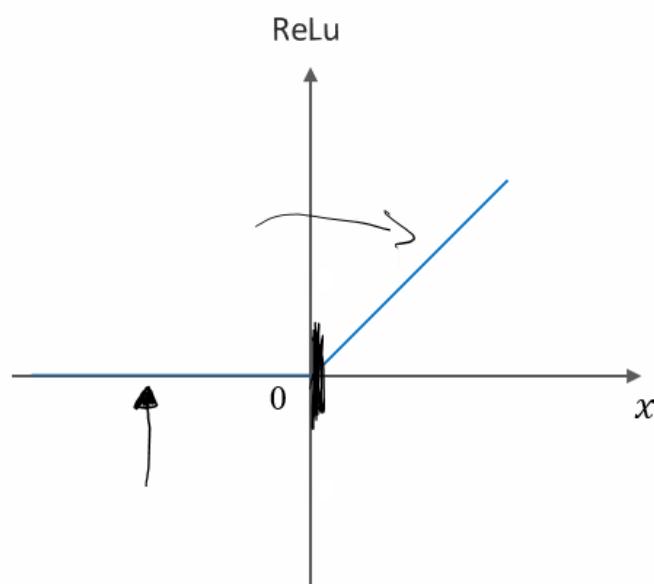
Lineal

Es como no poner nada, el valor de salida es ese valor.

ReLU (Unidad lineal rectificadora)

Si la entrada es negativo, siempre dará 0. Si la entrada es un valor positivo lo va a propagar a la salida tal cual, en este caso funciona como la lineal.

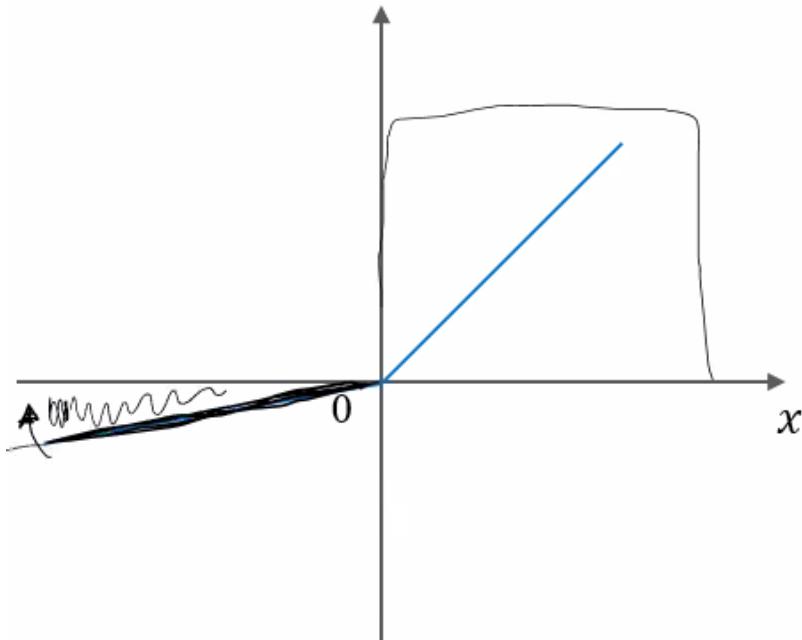
Es una función de las más utilizadas, la pega que tiene es que las entradas con valores negativos las estamos rechazando, para solucionar esto está Leaky ReLU



Leaky ReLU

Los valores negativos son cercanos a 0. No descarta de manera indiscriminada todas las entradas negativas.

Leaky ReLu



Fórmulas de todas

Nombre	Fórmula	TensorFlow
Step	$Step(x) = \theta(x - x_0)$	<code>(tf.sign(x-x0)+1)/2</code>
Sigmoid	$\sigma(x) = \frac{1}{1 + e^{-x}}$	<code>tf.sigmoid(x)</code>
Tanh	$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	<code>tf.tanh(x)</code>
ReLU	$ReLU(x) = \max(0, x)$	<code>tf.nn.relu(x)</code>
Softmax	$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}}$	<code>tf.nn.softmax(x)</code>

Si tengo que utilizar la sigmoide para más de dos clases, se usa la softmax ya que es multiclase.

Red Neuronal Convolucional (CNN)

Principalmente de una imagen se sacan datos importantes que detecta la propia red neuronal para luego alimentar al clasificador.

Nosotros no especificamos los datos que necesitamos, simplemente vamos aplicando filtros a dicha imagen, que deja de ser una imagen en el momento que entra a la red neuronal para pasar a ser datos que puede interpretar un clasificador.

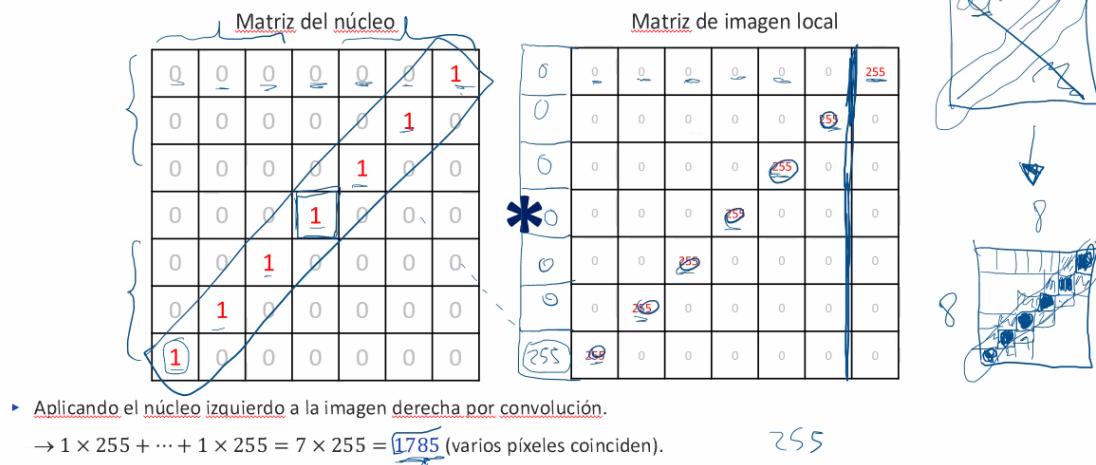
Se usa para datasets con imágenes, realiza filtros a las mismas para el data augmentation, mientras entrena a la vez.

Una de las prácticas más comunes es ir desplazando la imagen, resume la información centrándose en el pixel central, si realizamos esto con la imagen completa perderemos información de los bordes, para no perder esa información existe una técnica llamada **stride**.

Estos filtros también se pueden usar para detectar patrones locales:

Red Neuronal Convolucional (10/17)

Los filtros de convolución se pueden usar para detectar patrones locales:



El número de filas y columnas además del pixel central del filtro, son las que cogemos como muestra de la imagen original. Los filtros son impares.

Si nuestro filtro es de 3x3 y la imagen original es de 5x5, el filtro quitando siempre el punto del centro tiene un pixel a cada lado, que debemos restar a nuestra imagen original de 5x5, lo que deja como resultado del filtro una imagen 3x3, en este caso coincide con el filtro, pero no siempre será así.

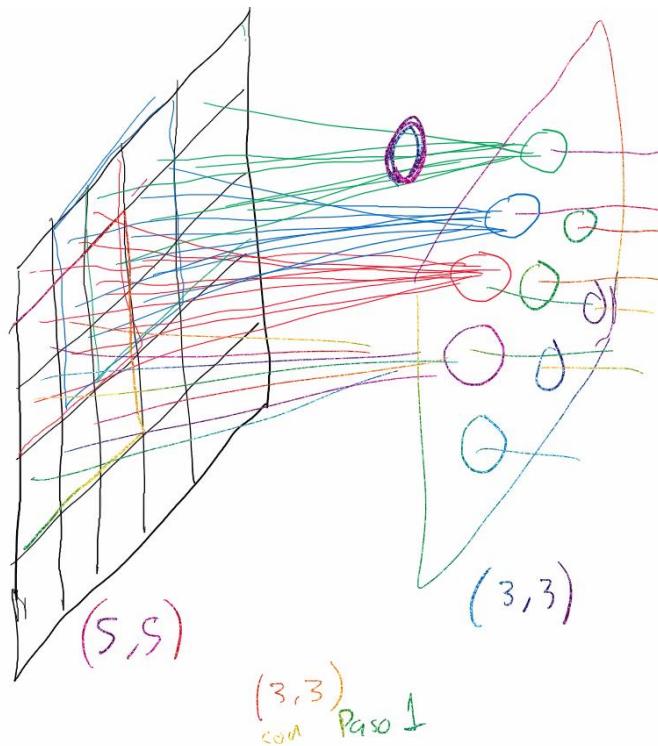
0	0	1
0	0	1
1	0	0

$$0 \cdot 1 + 0 \cdot 1 \cdot 1 \cdot 2$$

1	1	2	0	0
0	1	3	0	2
1	5	4	0	0
4	1	3	0	2
7	1	0	2	2

6	5	6
11	1	5
14	1	2

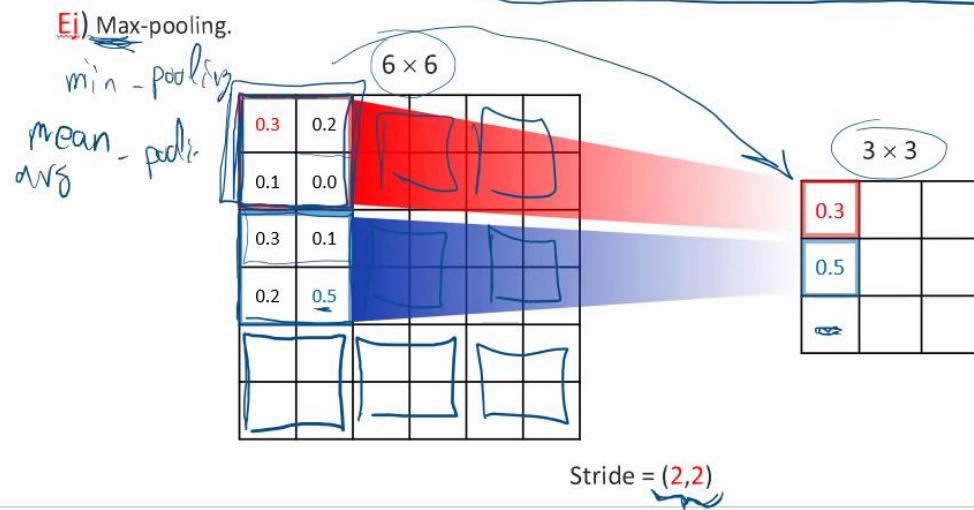
Ese resultado no se calcula de manera secuencial metiendo los datos en nuestra red neuronal uno a uno, sino que cada desplazamiento es una neurona independiente.



Max-pooling

Reduce dimensionalmente para prevenir el problema del sobreajuste, pierde información pero ayuda al entrenamiento de la red.

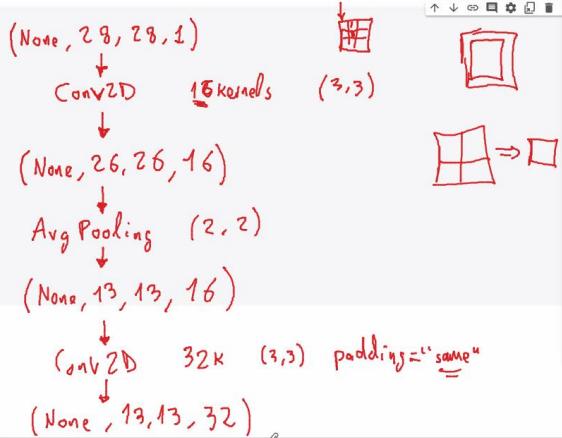
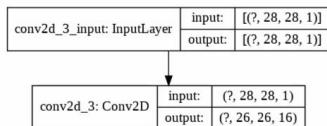
- Mueve el núcleo y reduce la información. La agrupación ayuda a prevenir el problema de sobreajuste.



Cambios en shape

Vamos a definir la función generadora del modelo. Usaremos un parámetro para indicar si queremos el modelo con capas Dropout o no, de forma que con la misma función generadora obtenemos distintos modelos.

```
1 def create_fashion(dropout=False):
2     model = Sequential()
3     model.add(layers.Conv2D(16, (3,3), activation='relu', input_shape=(image_height,image_width,1)))
4     if (dropout):
5         model.add(layers.Dropout(0.3))
6     model.add(layers.AveragePooling2D((2,2)))
7     model.add(layers.Conv2D(32, (3,3), activation='relu', padding="same"))
8     if (dropout):
9         model.add(layers.Dropout(0.3))
10    model.add(layers.AveragePooling2D((2,2)))
11   model.add(layers.Conv2D(64, (3,3), activation='relu', padding="same"))
12   model.add(layers.Flatten())
13   if (dropout):
14       model.add(layers.Dense(32, activation='relu'))
15   model.add(layers.Dense(32, activation='relu'))
16   model.add(layers.Dense(num_classes, activation='softmax'))
17
18   return model
21 plot_model(create_fashion(), show_layer_names=True, show_shapes=True)
```

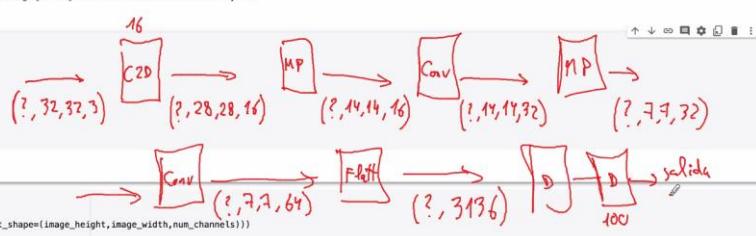


[17] Muestras de validación: 10000
 Resolución: 32x32
 Número de canales: 3

Aplicamos una transformación a los pixeles para dejar sus valores en float32 en el rango [0 ... 1]. También convertimos la variable objetivo en salida múltiple con one-hot encoding.

```
1 X_train = X_train / 255.0 # values [0..1] improve results
2 X_test = X_test / 255.0
3
4 y_train = np_utils.to_categorical(y_train)
5 y_test = np_utils.to_categorical(y_test)
6
7 print("y_train shape: " + str(y_train.shape))
8 num_classes = y_test.shape[1]
9 print("Número de clases: " + str(num_classes))
y_train shape: (50000, 100)
Número de clases: 100
```

```
1 def create_cnn_cifar():
2     model = Sequential()
3     model.add(layers.Conv2D(16, (5,5), activation='relu', input_shape=(image_height,image_width,num_channels)))
4     model.add(layers.MaxPooling2D((2,2)))
5     model.add(layers.Conv2D(32, (3,3), activation='relu', padding="same"))
6     model.add(layers.MaxPooling2D((2,2)))
7     model.add(layers.Conv2D(64, (3,3), activation='relu', padding="same"))
8     model.add(layers.Flatten())
9     model.add(layers.Dense(64, activation='relu'))
10    model.add(layers.Dense(100, activation='softmax'))
11
12 plot_model(create_cnn_cifar(), show_layer_names=True, show_shapes=True)
```

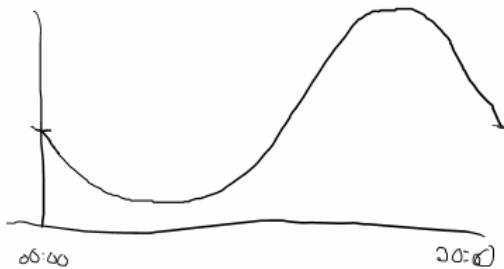


Red Neuronal Recurrente (RNN)

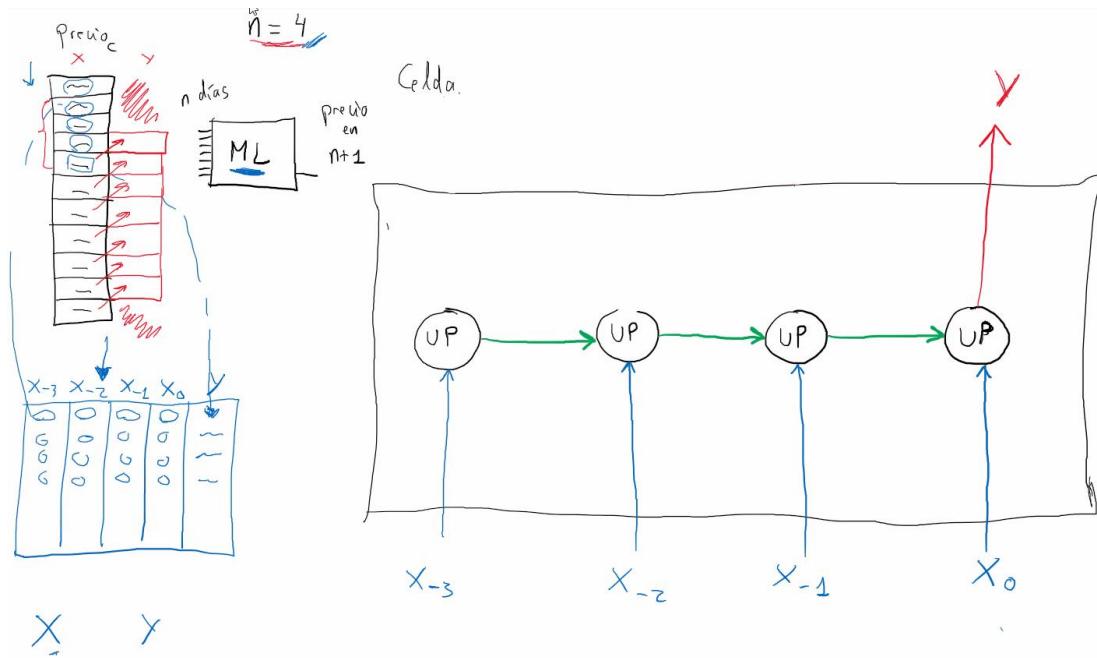
Red Neuronal Recurrente (1/7)

Acerca de la Red Neuronal Recurrente (RNN):

- ▶ RNN es una red neuronal profunda comúnmente utilizada con datos de secuencia.
- ▶ En una secuencia, hay un orden implícito; Los pasos son ordenados por el tiempo.
- ▶ Necesitamos un modelo que pueda tener en cuenta la autocorrelación.



Función de transición de estados



Cada unidad de procesamiento (que no es una neurona) se le pasa una única entrada del dataset (no todo, esto es importante) y se le pasa el estado anterior de la unidad de procesamiento.

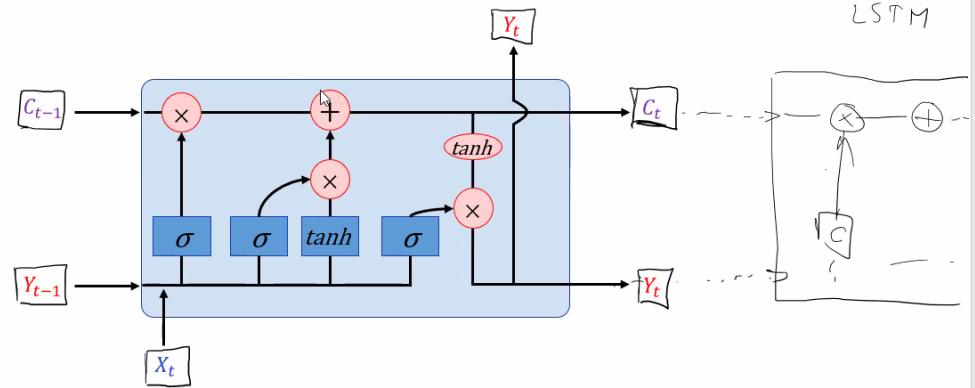
Con esto aprendemos distintos patrones secuenciales en nuestros datos.

Celda de memoria a corto y largo plazo (LSTM)

Esta celda son las unidades de procesamiento que hemos visto antes.

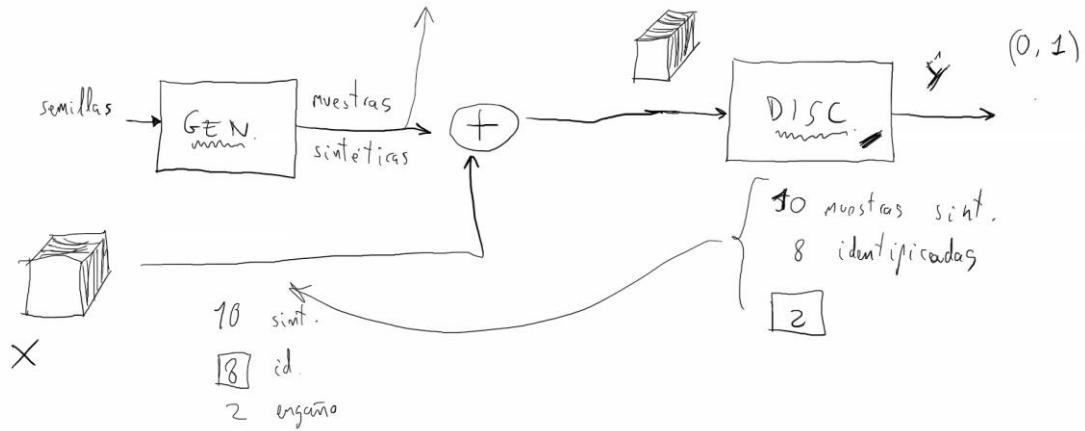
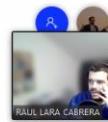
Celda de memoria a corto y largo plazo (LSTM):

- Mejora las células RNN regulares y puede mantener una correlación de larga duración.



Red generativa antagónica (GAN)

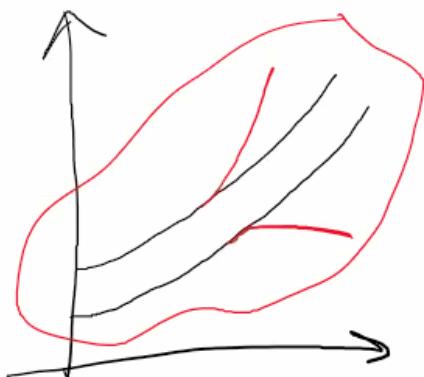
$GAN = \text{Generadora} + \text{Discriminadora}$.



La parte **discriminadora** quiere conseguir identificar que partes son sintéticas.

La parte **generadora** quiere conseguir realizar datos reales.

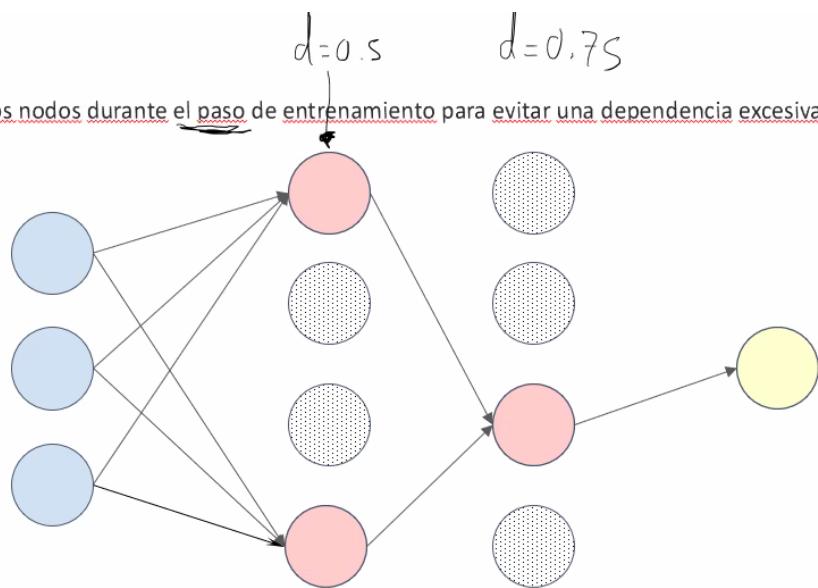
Lo normal es que haya una **divergencia**, y llegue un punto en que ninguna de las dos aprende. Es importante saber que ambas redes deben estar sincronizadas y aprender al mismo tiempo, para que este tipo de cosas no ocurran.



Previsión de errores

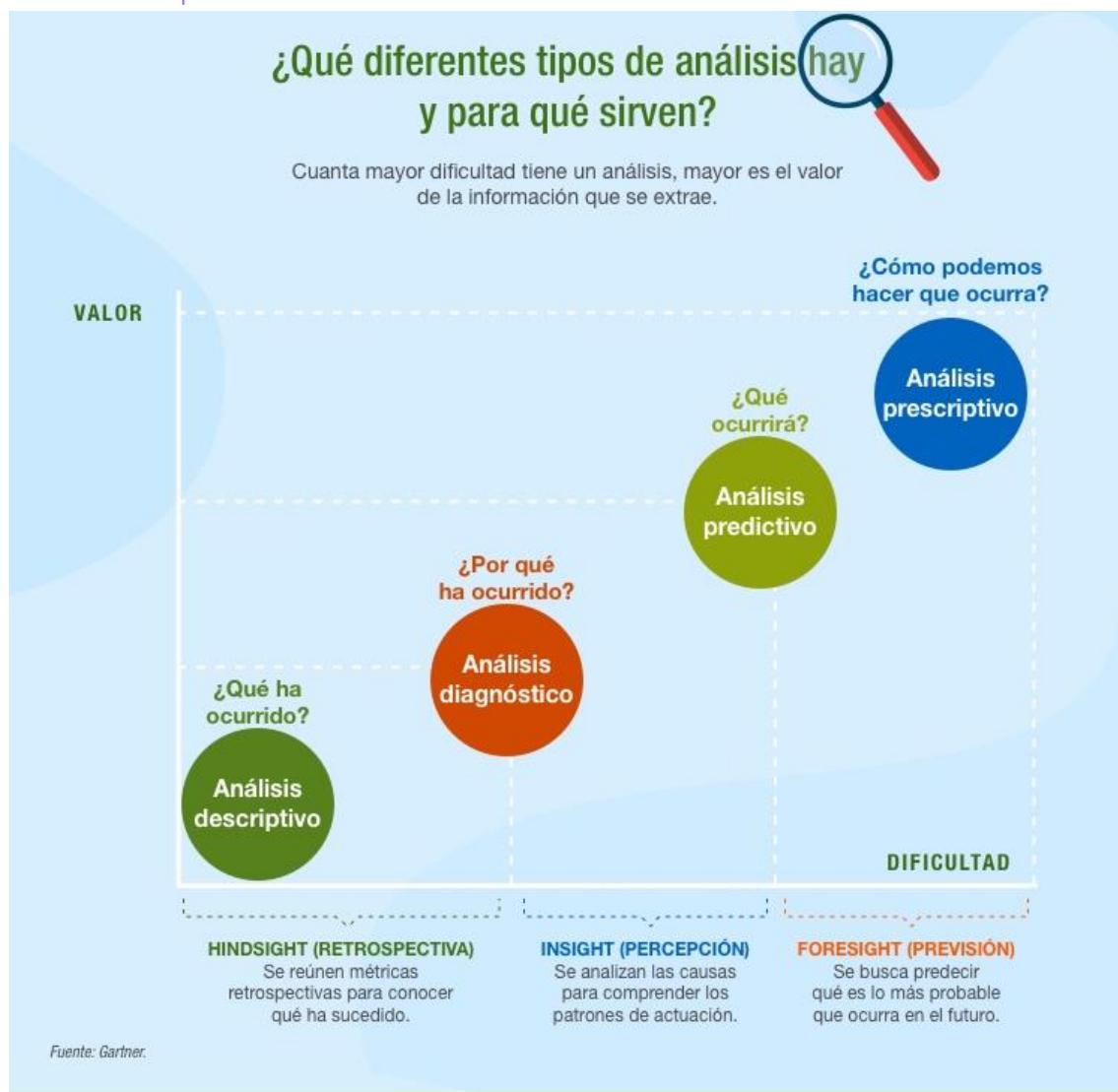
Dropout:

- Excluir al azar ciertos nodos durante el paso de entrenamiento para evitar una dependencia excesiva.



Conceptos generales

Diferentes tipos de análisis



Ejemplos de entramiento

cómputo de la regresión.

```
1 import math
2 from sklearn.metrics import mean_absolute_error, mean_squared_error
3
4 mi_sgd = SGDRegressor(max_iter=500, random_state=17)
5 mi_sgd.fit(X, y)
6 print(f"R^2: {mi_sgd.score(X, y)}\nMAE: {mean_absolute_error(y, mi_sgd.predict(X))}\nRMSE: {math.sqrt(mean_squared_error(y, mi_sgd.predict(X)))}"
```

```
1 from sklearn.model_selection import cross_val_score
2
3 n_samples = 500
4 np.random.seed(43)
5
6 X = np.expand_dims(np.linspace(-4, 4, n_samples), axis=1) + np.random.rand(n_samples, 1)
7 y = - 3 - X + 4*X*X - 7*X*X*X + 35 * np.random.rand(n_samples, 1)
8
9 pipe = Pipeline([('poly', PolynomialFeatures()),
10 | | | | ('ridge', Ridge())])
11
12 # Calidad del modelo con polinomio de grado 2
13 pipe.set_params(poly_degree=2, ridge_alpha=0.01, ridge_fit_intercept=False)
14
15 grado2 = cross_val_score(pipe, X, y, cv=5, scoring='neg_root_mean_squared_error')
16
```

```
○
9 pipe = Pipeline([('poly', PolynomialFeatures()),
10 | | | | ('ridge', Ridge())])
11
12 # Calidad del modelo con polinomio de grado 2
13 pipe.set_params(poly_degree=2, ridge_alpha=0.01, ridge_fit_intercept=False)
14
15 grado2 = cross_val_score(pipe, X, y, cv=5, scoring='neg_root_mean_squared_error')
16
17 pipe.set_params(poly_degree=3, ridge_alpha=0.01, ridge_fit_intercept=False)
18
19 grado3 = cross_val_score(pipe, X, y, cv=5, scoring='neg_root_mean_squared_error')
20
21 pipe.set_params(poly_degree=4, ridge_alpha=0.01, ridge_fit_intercept=False)
22
23 grado4 = cross_val_score(pipe, X, y, cv=5, scoring='neg_root_mean_squared_error')
```

```
✓ 1 print('Error medio del modelo de orden 2: ', np.mean(grado2))
ls 2 print('Error medio del modelo de orden 3: ', np.mean(grado3))
3 print('Error medio del modelo de orden 4: ', np.mean(grado4))
```

```
⇨ Error medio del modelo de orden 2: -191.19137909066512
Error medio del modelo de orden 3: -10.29097400525359
Error medio del modelo de orden 4: -13.752873105086987
```

La mejor opción seria de orden 3

```
[ ] 5 |     # La parte del sumatorio  
[ ] 6 |     for observacion in X:  
[ ] 7 |
```

2). Defina la clase 'LogisticRegression' para producir el resultado que se muestra a continuación:

```
1 class LogisticRegression:  
2     def __init__(self, learn_rate):  
3         # <Tu código va aquí>  
4         self.beta = # vector aleatorio  
5         self.learning_rate = learn_rate  
6  
7     def train(self, input_X, input_Y, n_epochs):  
8         # <Tu código va aquí>  
9         for i in n_epochs:  
10             g = gradient(input_X, input_Y, self.beta)  
11             self.beta = self.beta - self.learning_rate * g  
12  
13     def query(self, input_X, prob=True, cutoff=0.5):  
14         # <Tu código va aquí>
```

Bibliography

[Cognitive neural networks deep dive](#)

[Models machine learning](#)

References

ⁱ Insight: profound perception.

ⁱⁱ Subset: a part of a larger group of related things.