

Modulo 1 (I)

Introducción al Machine Learning (ML)

Mercedes Riveira-Martin
Instituto de Investigación Sanitaria Galicia Sur (IISGS)

INTELIGENCIA ARTIFICIAL

Roadmap

1. ¿Qué es el ML y por qué nos gusta tanto?
2. Breve introducción al dato
3. Fundamentos de un modelo de ML
 1. ¿Qué es un modelo de ML
 2. ¿Qué significa entrenar un modelo?
 3. Hiperparámetros
 4. La función de coste
4. Datos de un modelo de Machine Learning
 1. División de nuestro conjunto de datos
 2. Cross-validation
5. Evaluación del modelo
 1. Métricas de evaluación
 2. Challenges
 1. Overfitting
 2. Underfitting
 3. ¿Cómo evitamos el overfitting?: Regularización
6. Tipos de algoritmos de ML
 1. Supervisado (labeled):
 1. Clasificación
 2. Regresión
 2. No-Supervisado (unlabeled) (solo ejemplos)
 1. Clustering
 2. Dimensionality reduction
 3. Otros tipos
7. El DATO en Machine Learning
 1. Preparando los datos: EDA y Feature Engineering
 2. Mejorar el modelo a través del dato
 1. Data Augmentation
 2. Transfer Learning
8. Recapitulación

¿Qué es el Machine Learning (ML)?

¿Y por qué nos gusta tanto?

Artificial Intelligence

Cualquier técnica que permita a los ordenadores imitar la inteligencia humana. Incluye machine learning

Machine Learning
Subconjunto de IA cuyas técnicas permiten a las máquinas aprender sin ser programadas. Incluye Deep Learning

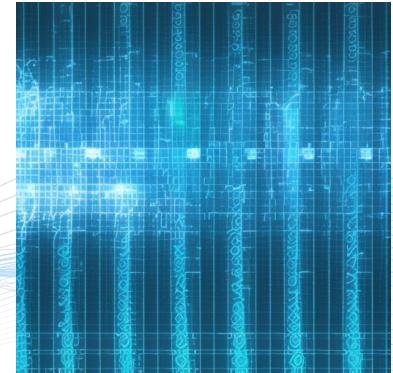
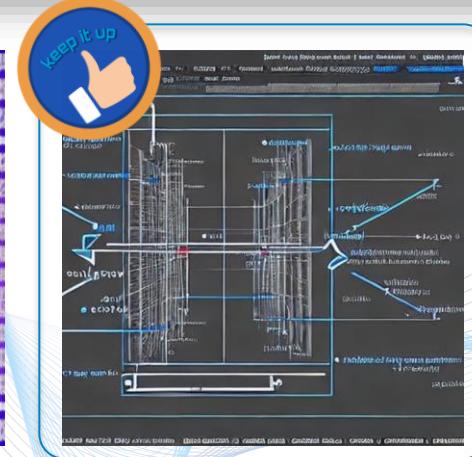
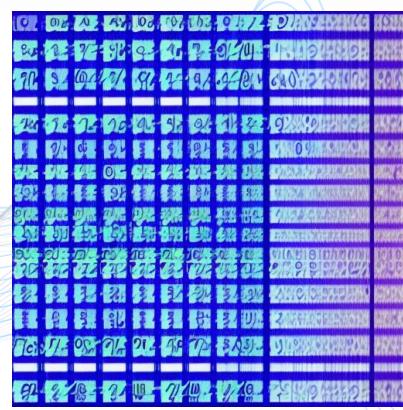
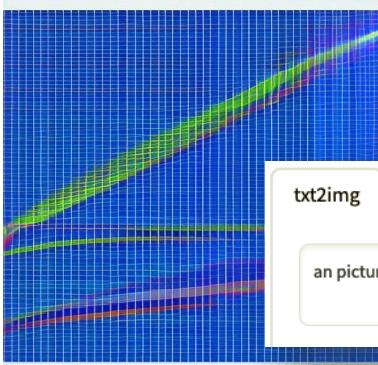
Deep Learning

Subconjunto del ML basado en redes neuronales que permite a una máquina entrenarse para realizar una tarea aprendiendo con ejemplos

stable-diffusion

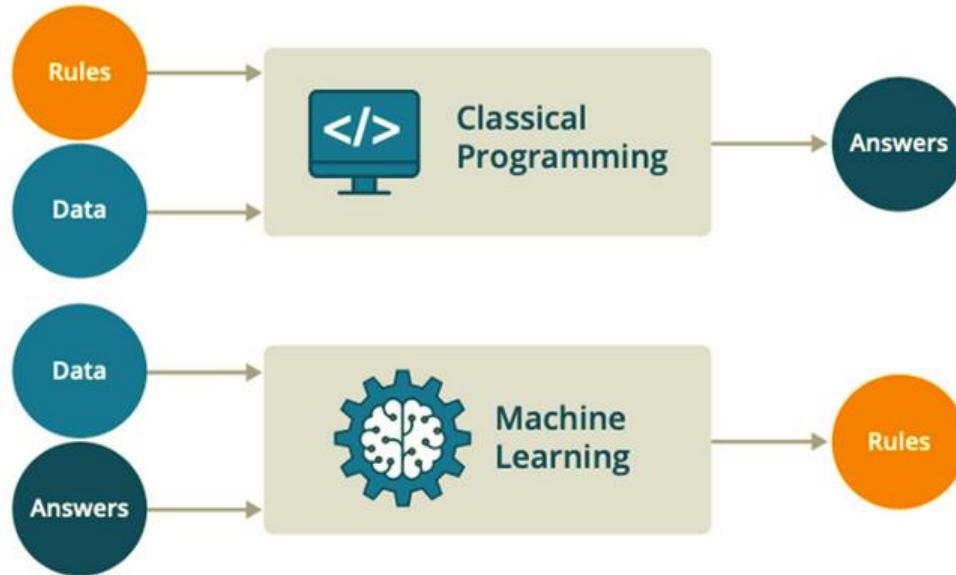
¿Qué es el Machine Learning (ML)?

¿Y por qué nos gusta tanto?



¿Qué es el Machine Learning (ML)?

¿Y por qué nos gusta tanto?

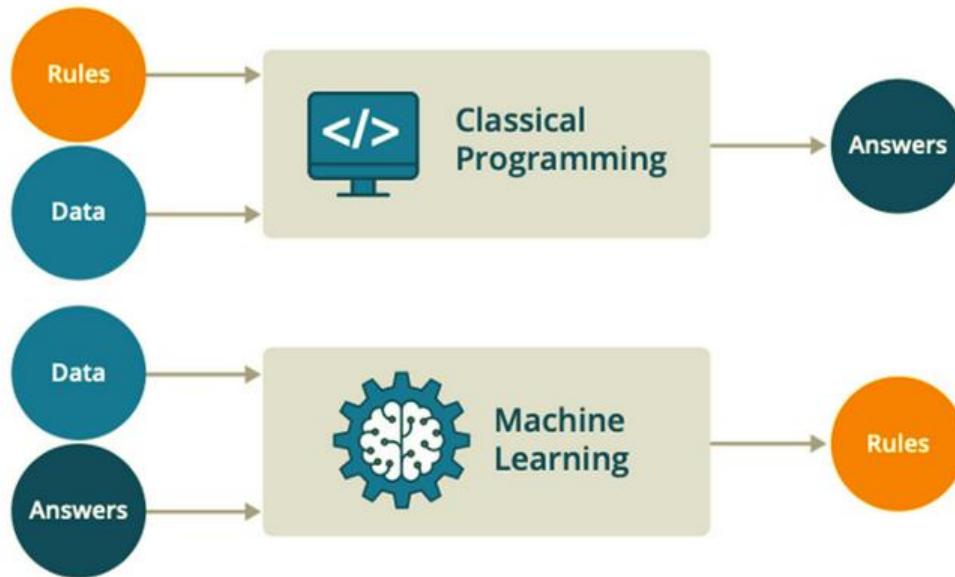


En la programación tradicional **se escribe un programa con una serie de reglas** específicas a seguir.

En ML **se escribe un programa capaz de aprender a través de ejemplos** para obtener unas reglas

¿Qué es el Machine Learning (ML)?

¿Y por qué nos gusta tanto?



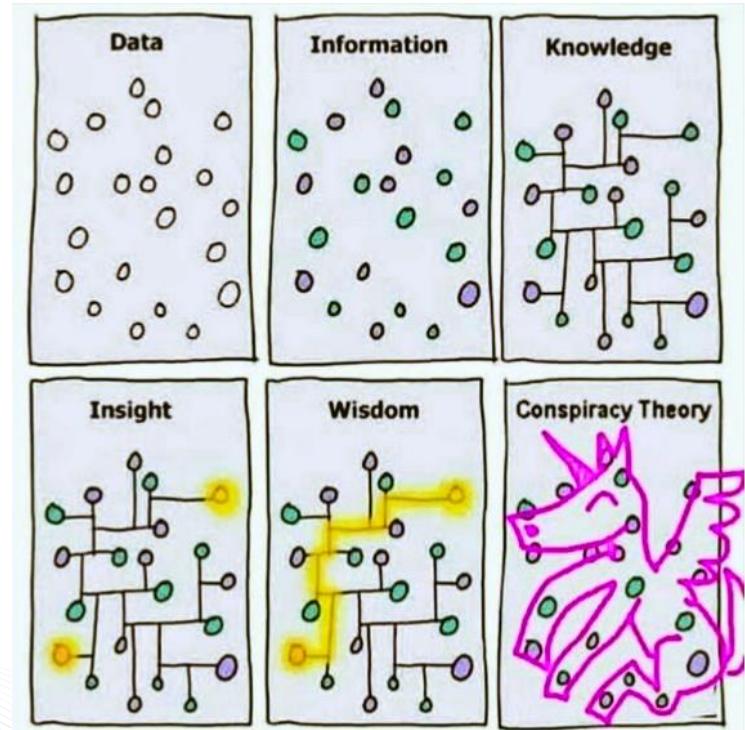
Lo interesante de esto es que:

1. Nuestro trabajo buscando esas reglas se **reduce**
2. Los modelos se pueden utilizar para otro conjunto de datos, **sin reescribir el código**
3. **Nos permite resolver problemas demasiado complejos para los métodos de cálculo tradicionales y/o sin solución (reglas) conocida**
4. También nos ayuda a **aprender (data mining)**

Breve introducción al dato

Datos vs. Información vs. Conocimiento

- Los **datos** son observaciones de un fenómeno, que pos si solos carecen de sentido
- Cuando los ponemos en **contexto** les damos sentido. Hablamos de **información**.
- Cuando estructuramos esta información y la juntamos con nuestra experiencia previa, hablamos de **conocimiento**, el cual nos da la capacidad de **predecir** o **inferir sucesos**



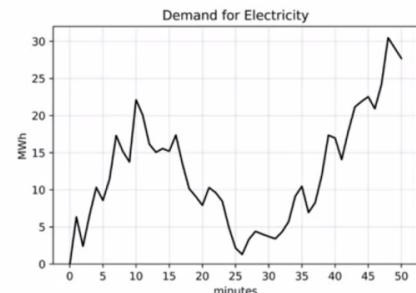
Breve introducción al dato

¿Cómo serán los datos en ML?

- Imágenes
- Texto
- Audio
- Series temporales
- Datos tabulares
- ...



“Esto es un perro”



Breve introducción al dato

¿Cómo serán los datos en ML?

- Imágenes
- Texto
- Audio
- Series temporales
- **Datos tabulares**
- ...

spreadsheet = dataset

Filas = Registro / obsevación
X = variable independiente

Columnas = atributos de esas obsevaciones (feature)

Y = variable dependiente

| Player | Minutes | Points | Rebounds | Assists |
|--------|---------|--------|----------|---------|
| A | 41 | 20 | 6 | 5 |
| B | 30 | 29 | 7 | 6 |
| C | 22 | 7 | 7 | 2 |
| D | 26 | 3 | 3 | 9 |
| E | 20 | 19 | 8 | 0 |
| F | 9 | 6 | 14 | 14 |
| G | 14 | 22 | 8 | 3 |
| I | 22 | 36 | 0 | 9 |
| J | 34 | 8 | 1 | 3 |

Fundamentos de un modelo de Machine Learning

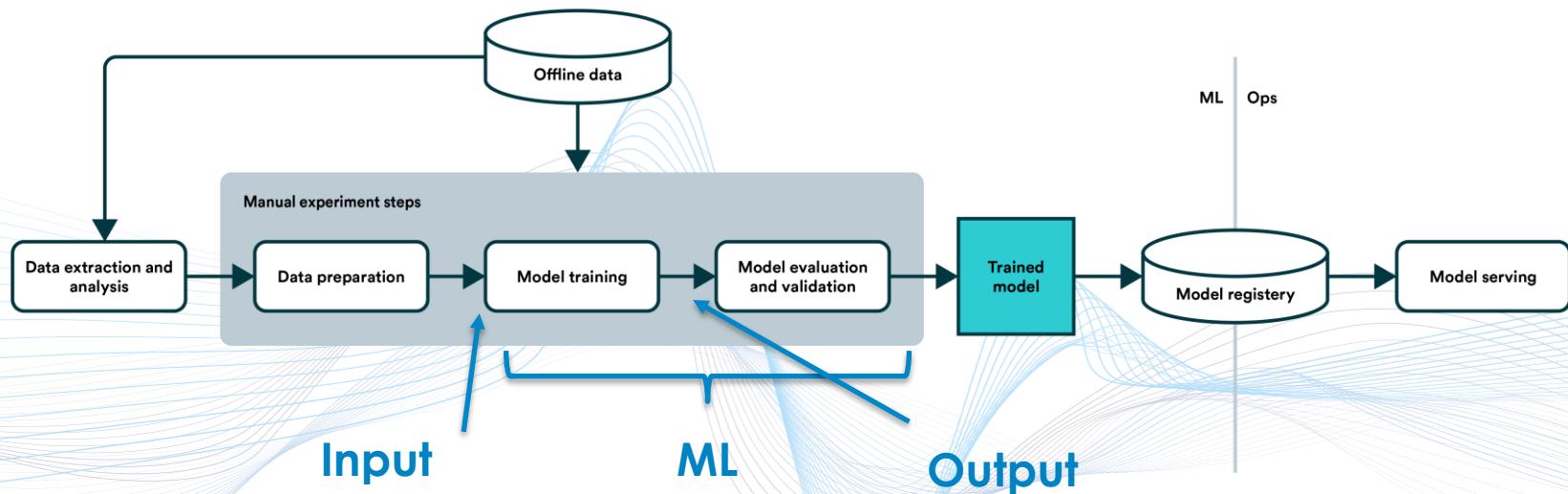
1. ¿Qué es un modelo de ML?
2. ¿Qué significa entrenar un modelo?
3. Hiperparámetros
4. La función de coste
 - 4.1. Descenso del gradiente



Fundamentos de un modelo de Machine Learning

1. ¿Qué es un modelo de ML?

Un modelo de ML es una **función** definida por unos parámetros, unos datos de **entrada** (input, datos **independientes**) y **salida** (output, datos **dependientes** que queremos predecir)



Fundamentos de un modelo de Machine Learning

2. ¿Qué significa entrenar un modelo?

Entrenar un modelo no es más que encontrar los **parámetros** de la función (del modelo) que **minimizan el error de salida**.

Este error es la diferencia entre el valor estimado y el valor real (predicción vs. etiqueta real), y se analiza con la **función de coste**



Fundamentos de un modelo de Machine Learning

2. ¿Qué significa entrenar un modelo?

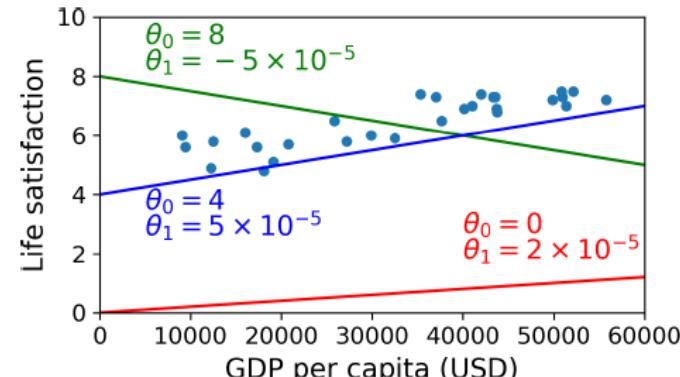
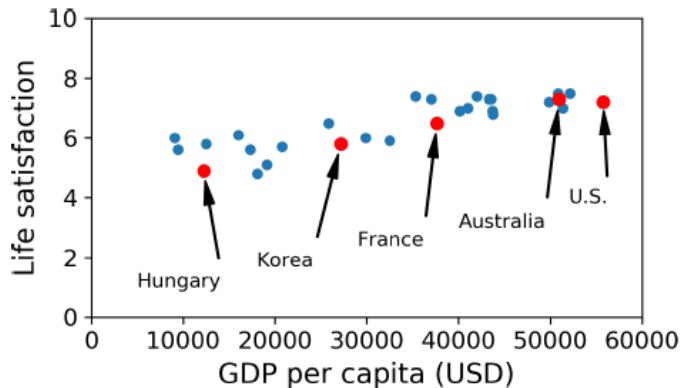
Ejemplo: queremos predecir el nivel de satisfacción vital de una población en función de su gdp per cápita

- 1) Vemos los datos, observamos una tendencia
- 2) Seleccionamos una función de regresión lineal simple con dos **parámetros**

$$\text{life_satisfaction} = \theta_0 + \theta_1 \times \text{GDP_per_capita}$$

- 3) La función de coste (**RMSE**) optimiza estos valores para minimizar el error

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



Fundamentos de un modelo de Machine Learning

2. ¿Qué significa entrenar un modelo?

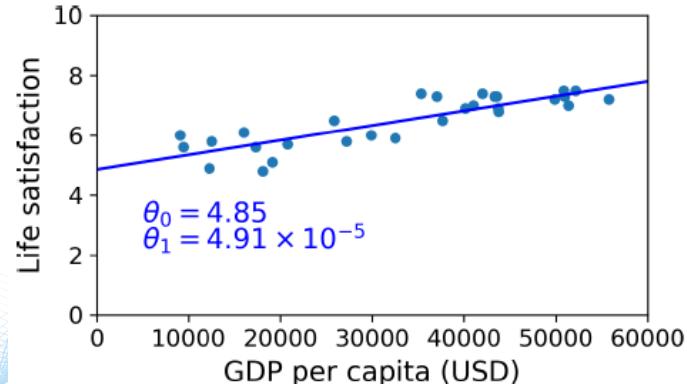
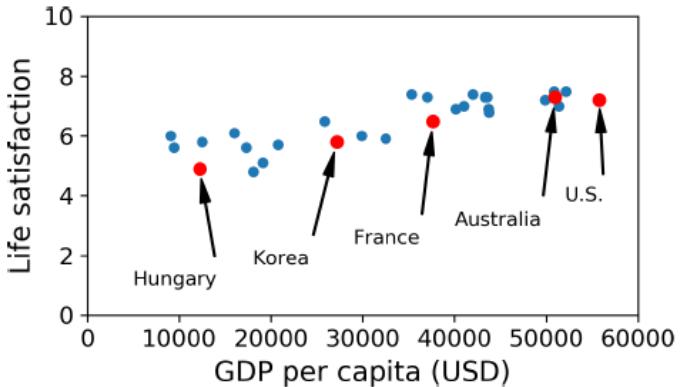
Ejemplo: queremos predecir el nivel de satisfacción vital de una población en función de su pib per cápita

- 1) Vemos los datos, observamos una tendencia
- 2) Seleccionamos una función de regresión lineal simple con dos **parámetros**

$$\text{life_satisfaction} = \theta_0 + \theta_1 \times \text{GDP_per_capita}$$

- 3) La función de coste (**RMSE**) optimiza estos valores para minimizar el error

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



Fundamentos de un modelo de Machine Learning

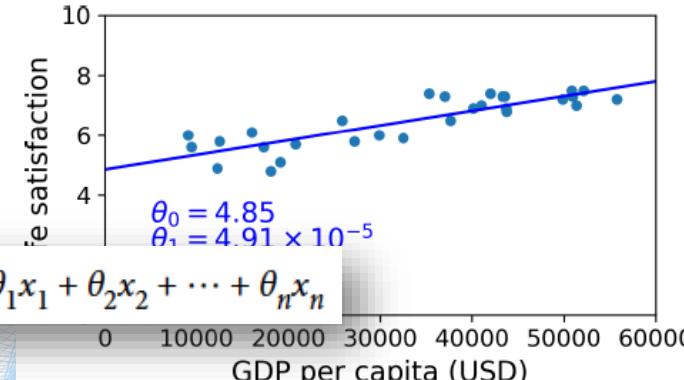
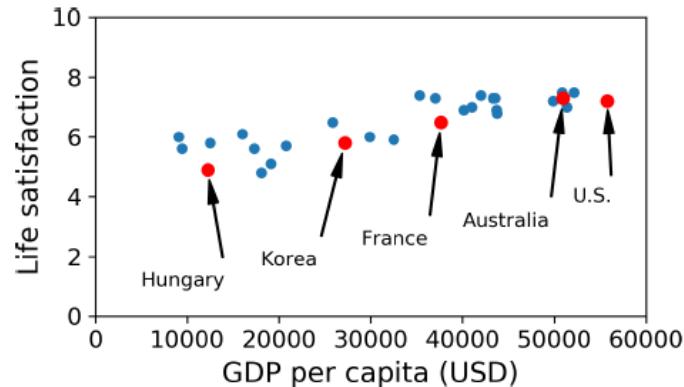
2. ¿Qué significa entrenar un modelo?

Ejemplo: queremos predecir el nivel de satisfacción vital de una población en función de su pib per cápita

Si nuestro modelo se ha entrenado bien, podremos hacer predicciones (**inferir**)

Si no, tendremos que:

- Usar más **atributos** (tasa de empleo, salud...)
- Mejorar nuestros **datos** (outliers?)
- Seleccionar un **modelo mejor** (regresión polinómica por ejemplo) o modificar los **hiperparámetros**



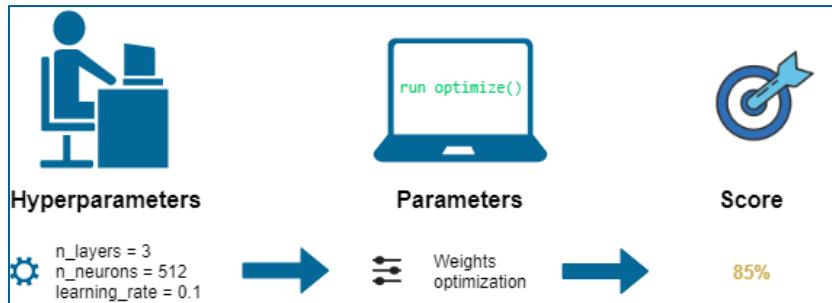
Fundamentos de un modelo de Machine Learning

3. Hiperparámetros

Los hiperparámetros afectan a la **forma de aprender** que tiene el modelo, pero no afectan a la función. Se definen **antes** del entrenamiento

Es importante no confundir los **parámetros** de una función con los **hiperparámetros** de entrenamiento

- **Learning rate**: velocidad a la que itera el modelo
- **Número de épocas (epochs)**: numero de veces que el modelo "ve" el conjunto completo de los datos de entrenamiento
- **Número de batch**: cantidad de datos que el modelo "ve" en una iteración
- **Early stopping** ...



$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Fundamentos de un modelo de Machine Learning

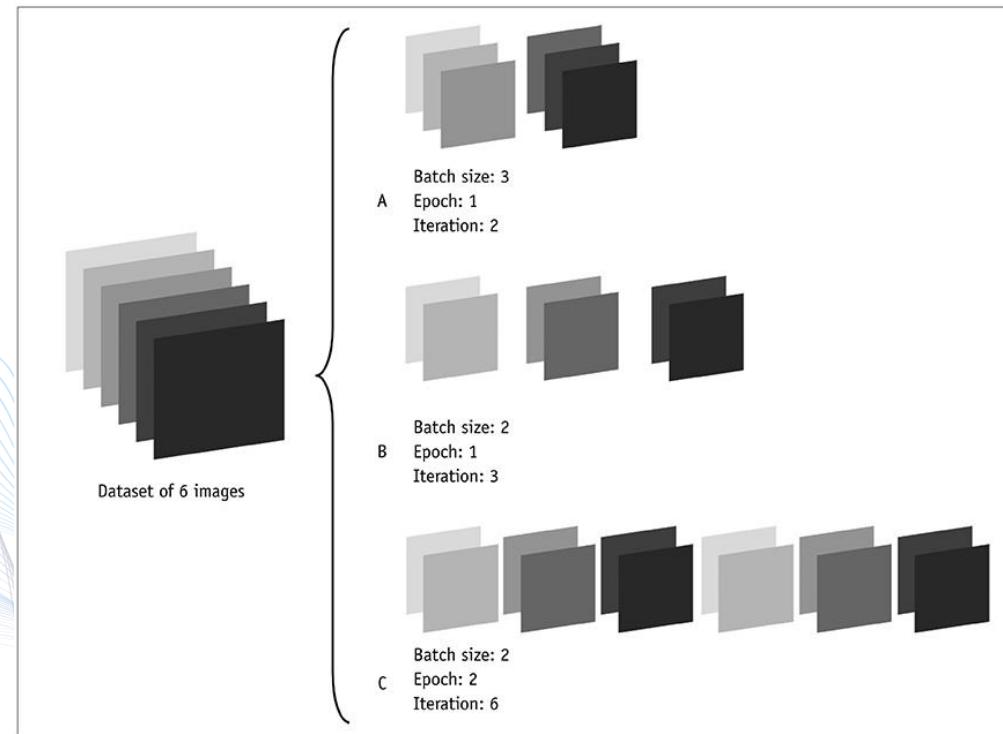
3. Hiperparámetros

Ejemplo:

Tengo un dataset de 6 imágenes.

Si el batch size es 2 y la epoch 1, el modelo tendrá que iterar 3 veces sobre los datos ($6/2 = 3$).

Si la epoch es dos, esto lo tendrá que hacer dos veces enteras ($6/2 = 3$; $3*2 = 6$).



Fundamentos de un modelo de Machine Learning

4. La función de coste

La función de coste analiza la diferencia (error) entre **el valor real y la predicción**

1) Problemas de regresión:

- 1) RMSE (Root Mean Squared Error)
- 2) Mean Squared Error (MSE) (L2)
- 3) Mean Absolute Error (MAE) (L1)

2) Problemas de clasificación

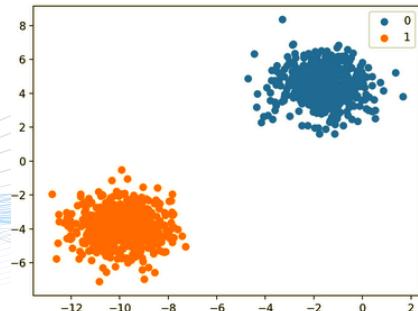
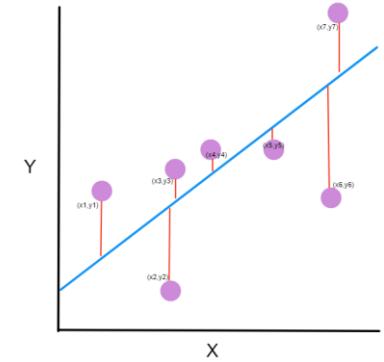
- 1) Cross-Entropy Loss
 - Binary Cross-Entropy
- 2) Hinge loss
- 3) ...

$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2}$$

$$\text{MSE}(\mathbf{X}, h_{\theta}) = \frac{1}{m} \sum_{i=1}^m (\theta^T \mathbf{x}^{(i)} - y^{(i)})^2$$

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = h_{\theta}(\mathbf{x}) = \theta \cdot \mathbf{x}$$

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$



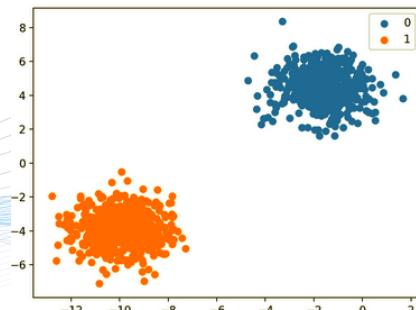
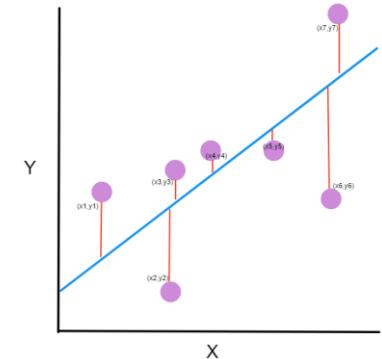
Fundamentos de un modelo de Machine Learning

4. La función de coste

La función de coste hay que optimizarla (generalmente, minimizarla).

Métodos de optimización (entre otros):

- **Método de mínimos cuadrados (least-squared, OSL)**
 - Funciona bien con pocas features y observaciones
 - Bueno para soluciones analíticas
- **Descenso del gradiente (Gradient Descent)**
 - Funciona bien con muchas features y observaciones
 - Nos vale para cualquier tipo de función de coste (derivable)
 - Tipos:
 - Stochastic Gradient Descent (SGD)
 - Batch Gradient-Descent (BGD)



Fundamentos de un modelo de Machine Learning

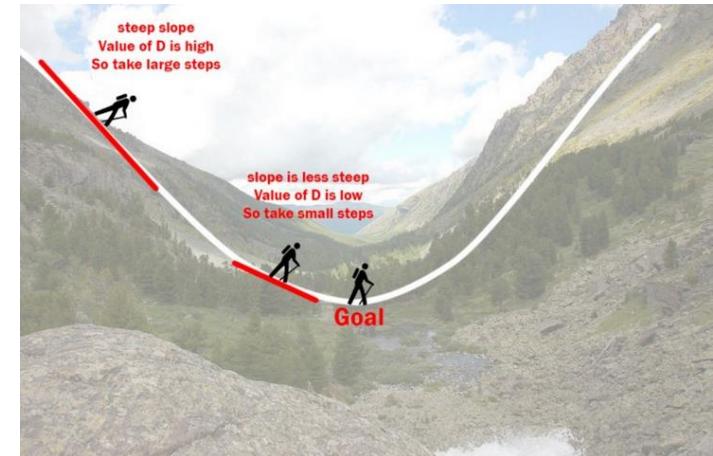
4. La función de coste

Descenso del gradiente (Gradient Descent)

- Proceso matemático **iterativo** para encontrar el **mínimo** de una función (la función de coste)

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

α = learning rate



$$\Theta^1 = \Theta^0 - \alpha \nabla J(\Theta)$$

evaluated at Θ^0

current position
opposite direction
next position
small step
direction of fastest increase

Fundamentos de un modelo de Machine Learning

4. La función de coste

Descenso del gradiente (Gradient Descent)

- Ejemplo: descenso del gradiente en regresión lineal simple

$$Y' = mX + b$$

Coste =

$$J_{m,b} = \frac{1}{N} \sum_{i=1}^N (Y'_i - Y_i)^2$$

$$J_{m,b} = \frac{1}{N} \sum_{i=1}^N (\text{Error}_i)^2$$

$\alpha = \text{learning rate}$

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Fórmula para el
descenso del
gradiente

$$\frac{\partial J}{\partial m} = 2 \cdot \text{Error} \cdot \frac{\partial}{\partial m} \text{Error}$$

$$\frac{\partial J}{\partial b} = 2 \cdot \text{Error} \cdot \frac{\partial}{\partial b} \text{Error}$$

$$\frac{\partial}{\partial m} \text{Error} = \frac{\partial}{\partial m} (Y' - Y) = x$$

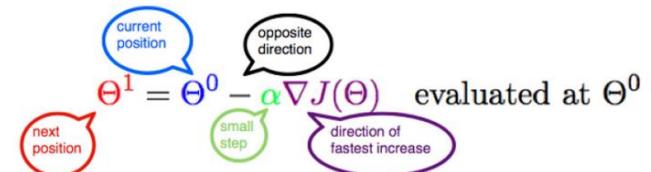
$$\frac{\partial}{\partial b} \text{Error} = \frac{\partial}{\partial b} (Y' - Y) = 1$$

$$\frac{\partial J}{\partial m} = 2 \cdot \text{Error} \cdot X$$

$$\frac{\partial J}{\partial b} = 2 \cdot \text{Error}$$

$$m' = m_0 - \alpha \cdot 2 \cdot (Y'_i - Y) \cdot X$$

$$b' = b_0 - \alpha \cdot 2 \cdot (Y'_i - Y)$$

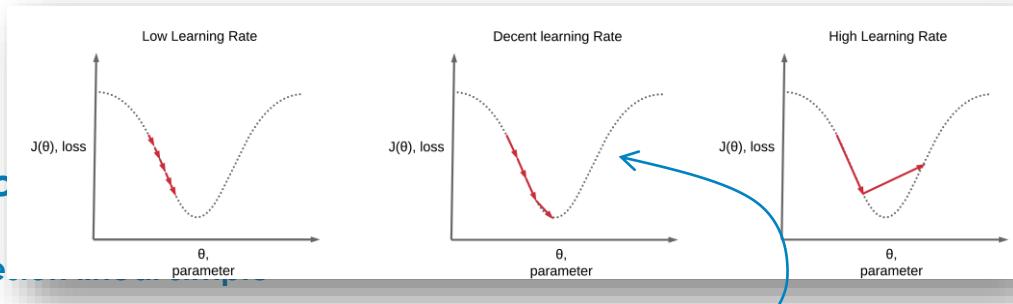


Fundamentos de un modelo de Machine Learning

4. La función de coste

Descenso del gradiente (Gradient Descent)

- Ejemplo: descenso del gradiente en regresión lineal



$$Y' = mX + b$$

Coste =

$$J_{m,b} = \frac{1}{N} \sum_{i=1}^N (Y'_i - Y_i)^2$$

$$J_{m,b} = \frac{1}{N} \sum_{i=1}^N (\text{Error}_i)^2$$

$\alpha = \text{learning rate}$

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\frac{\partial J}{\partial m} = 2 \cdot \text{Error} \cdot \frac{\partial}{\partial m} \text{Error}$$

$$\frac{\partial J}{\partial b} = 2 \cdot \text{Error} \cdot \frac{\partial}{\partial b} \text{Error}$$

$$\frac{\partial}{\partial m} \text{Error} = \frac{\partial}{\partial m} (Y' - Y) = x$$

$$\frac{\partial}{\partial b} \text{Error} = \frac{\partial}{\partial b} (Y' - Y) = 1$$

$$\frac{\partial J}{\partial m} = 2 \cdot \text{Error} \cdot X$$

$$\frac{\partial J}{\partial b} = 2 \cdot \text{Error}$$

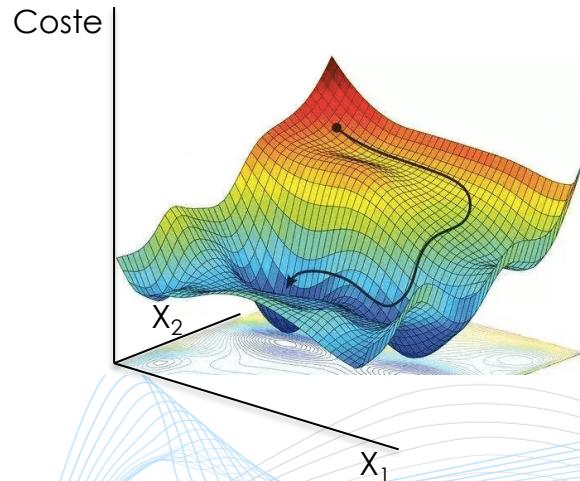
$$m' = m_0 - \alpha \cdot 2 \cdot (Y'_i - Y) \cdot X$$

$$b' = b_0 - \alpha \cdot 2 \cdot (Y'_i - Y)$$

Fundamentos de un modelo de Machine Learning

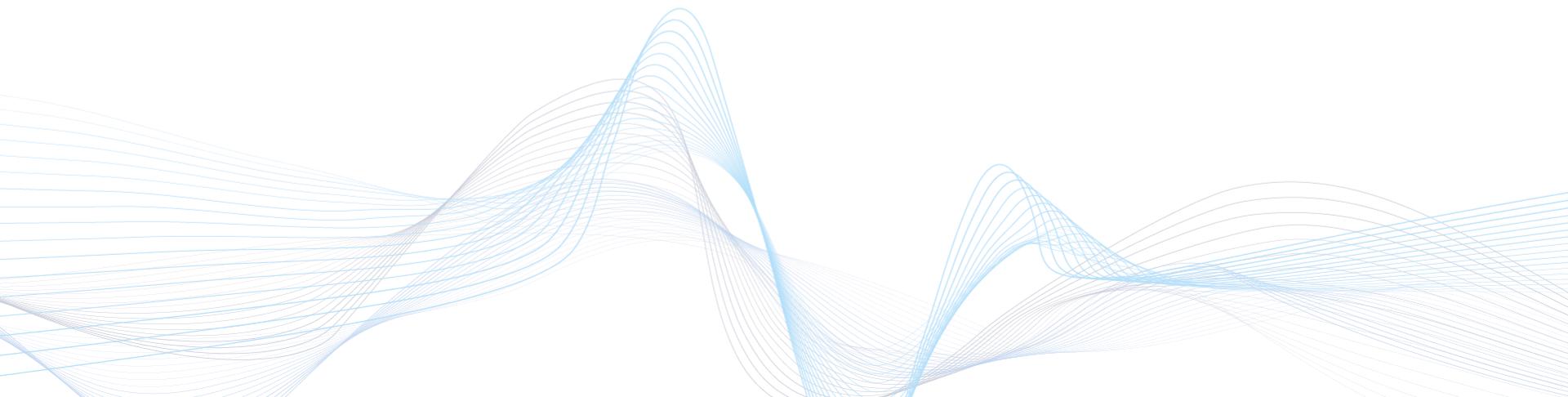
4. La función de coste

Avance a redes neuronales: veremos que la función de coste de una red neuronal no es tan sencilla, y que por tanto el algoritmo del descenso del gradiente es más complejo, pero en definitiva el método es el mismo



Datos de un modelo de Machine Learning

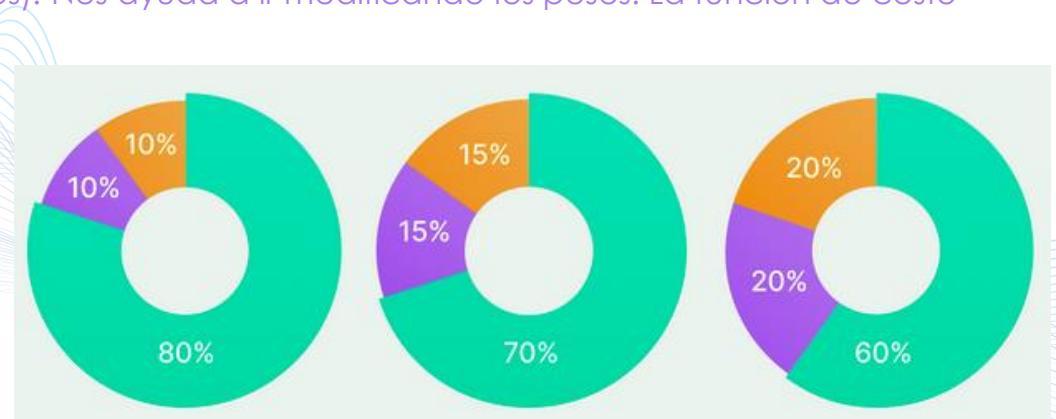
1. División de nuestro conjunto de datos
2. *Cross-validation*



Datos de un modelo de Machine Learning

1. División de nuestro conjunto de datos

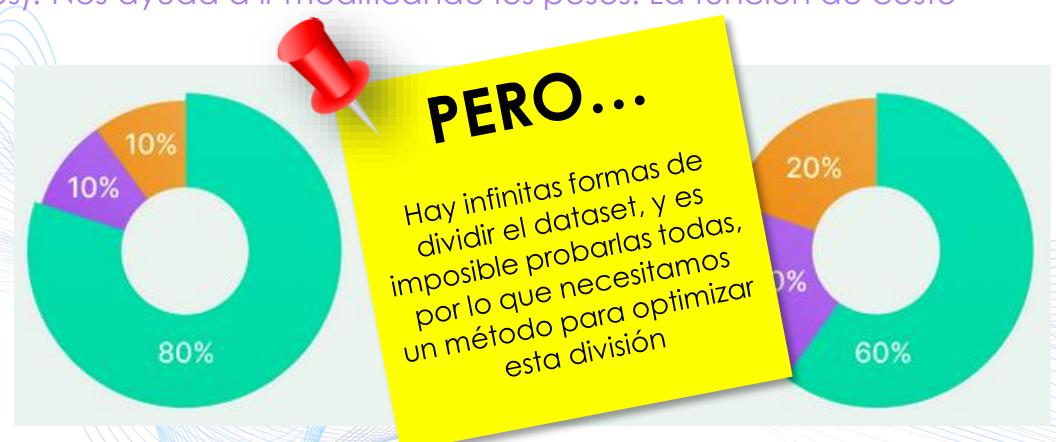
- **Train set**: conjunto de datos con los que entrenamos el algoritmo. En cada iteración del modelo (epoch) los datos son los mismos. La función de coste se evalúa con este set.
- **Validation set**: conjunto de datos con los que se validar el modelo después de cada epoch (durante el entrenamiento, pero sin haberlos visto antes). Nos ayuda a ir modificando los pesos. La función de coste **también** se evalúa con este set.
- **Test set**: este set es el que se introduce en el modelo **una vez entrenado**. Puesto que contiene datos que el modelo nunca ha visto antes, nos indica cómo de bien **generaliza** el modelo.



Datos de un modelo de Machine Learning

1. División de nuestro conjunto de datos

- **Train set**: conjunto de datos con los que entrenamos el algoritmo. En cada iteración del modelo (epoch) los datos son los mismos. La función de coste se evalúa con este set.
- **Validation set**: conjunto de datos con los que se validar el modelo después de cada epoch (durante el entrenamiento, pero sin haberlos visto antes). Nos ayuda a ir modificando los pesos. La función de coste **también** se evalúa con este set.
- **Test set**: este set es el que se introduce en el modelo **una vez entrenado**. Puesto que contiene datos que el modelo nunca ha visto antes, nos indica cómo de bien **generaliza** el modelo.



Datos de un modelo de Machine Learning

2. Cross-Validation (Validación cruzada)

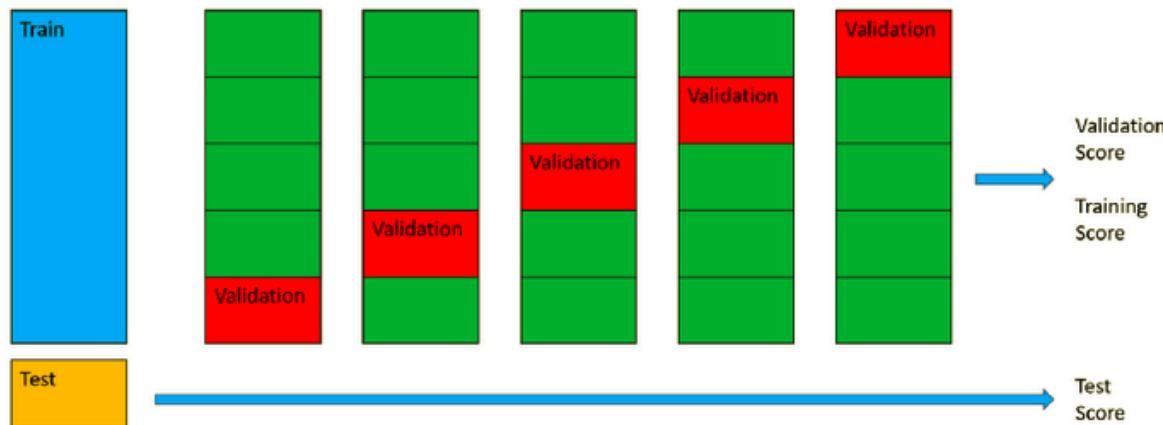
K-Fold Cross-Validation

- Se dividen los datos en train y test
- El train se divide en **k subconjuntos** de manera aleatoria:
 - $k-1$ conjuntos serán de entrenamiento
 - 1 conjunto será de validación
- Se realiza el mismo proceso k veces, pero cambiando los subconjuntos, para la siguiente epoch
- Al finalizar las k iteraciones se promedia el error y la precision para cada subconjunto



Datos de un modelo de Machine Learning

2. Cross-Validation (Validación cruzada)



Ejemplo:

Epochs: 20
K = 5 (k-fold cross-validation)

Mi modelo va separar k (5) veces un conjunto de datos formado por k-1 (4) subconjuntos de train y uno de validation.

Sobre cada conjunto de datos el modelo pasa 20 veces (epochs)

i.e.: El modelo pasa por el mismo set de datos $5 \times 20 = 100$ veces pero dividido de formas distintas

Evaluación de un modelo

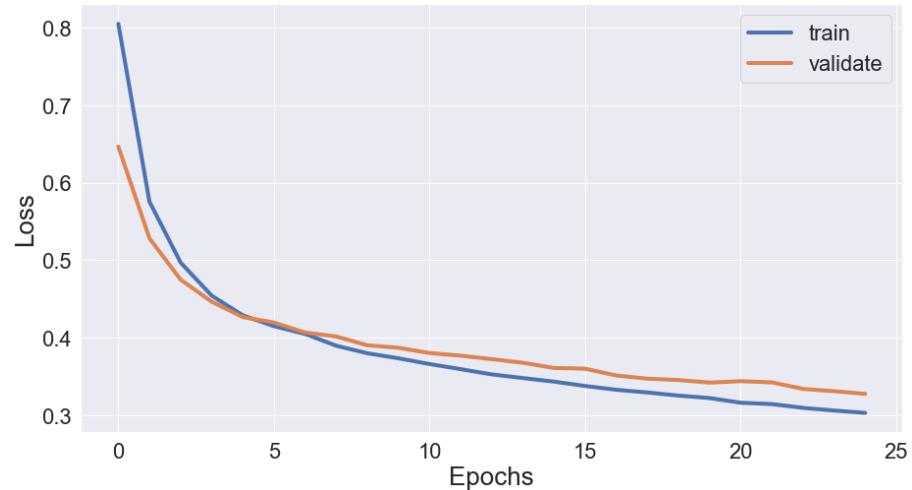
1. Métricas de evaluación
2. Challenges:
 - 2.1. Overfitting
 - 2.2. Underfitting
3. ¿Cómo evitamos el overfitting?: Regularización

Evaluación de un modelo

1. Métricas de evaluación

Learning curves (curvas de aprendizaje)

- **Training loss:** nos indica si el modelo se ajusta bien o mal a los datos de entrenamiento
- **Validation loss:** nos indica el rendimiento de un modelo en el set de validación
- Otras métricas en función del problema

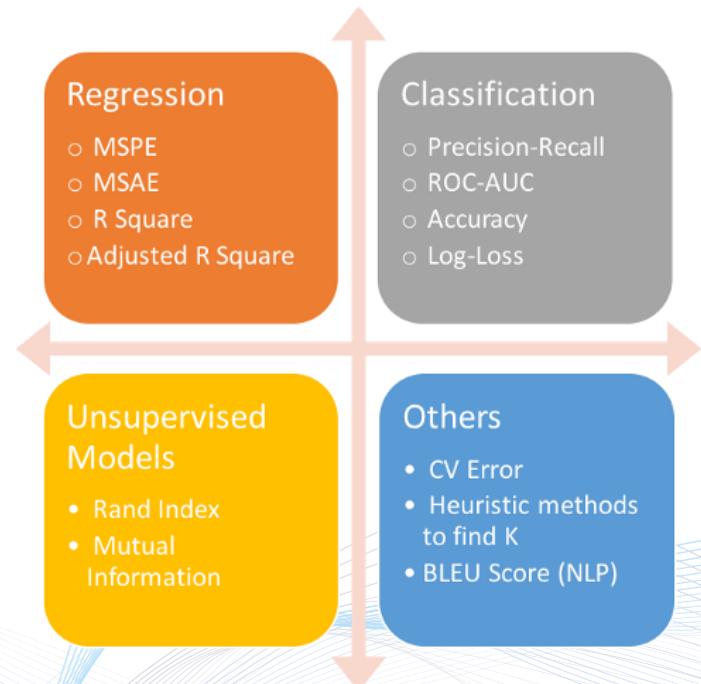


Evaluación de un modelo

1. Métricas de evaluación

Learning curves (curvas de aprendizaje)

- **Training loss:** nos indica si el modelo se ajusta bien o mal a los datos de entrenamiento
- **Validation loss:** nos indica el rendimiento de un modelo en el set de validación
- Otras métricas en función del problema

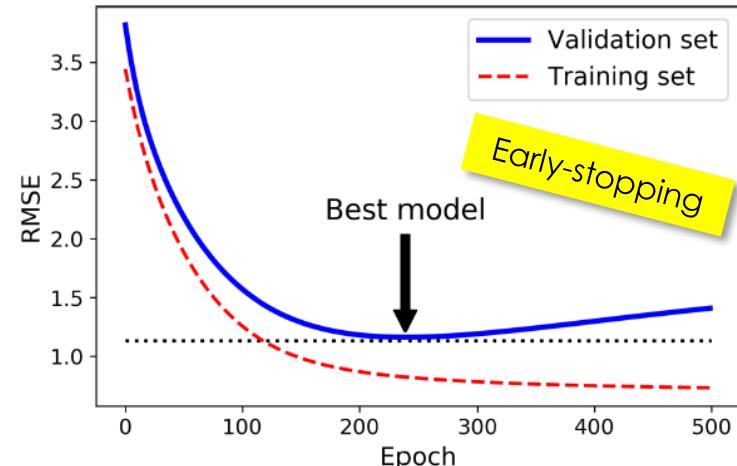
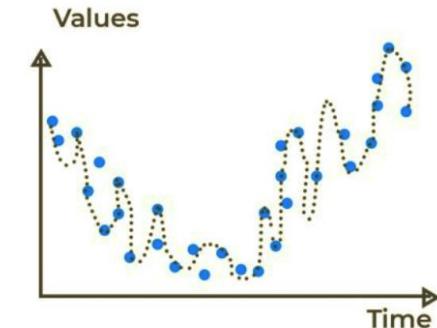


Evaluación de un modelo

2. Challenges

Overfitting

- Nuestra predicción se ajusta “demasiado bien” a nuestros datos, por lo que el training loss decrece
- Sin embargo, no es capaz de generalizar, por lo que la validation loss crece
- El momento en el que nuestra training loss baje, pero la validation loss crezca, estamos haciendo **overfitting**

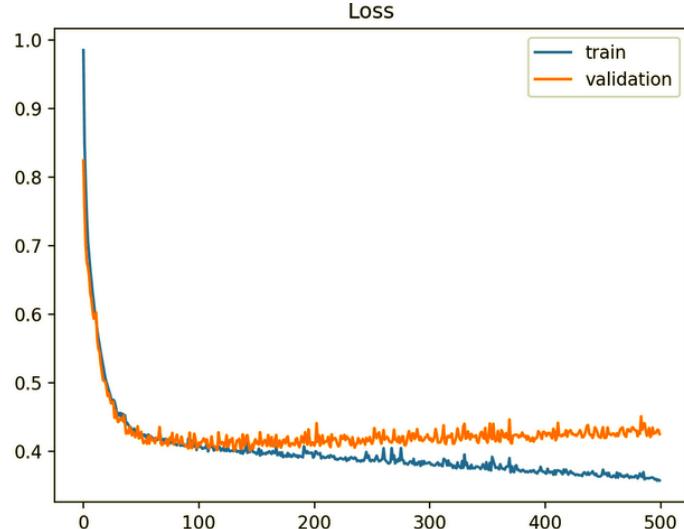
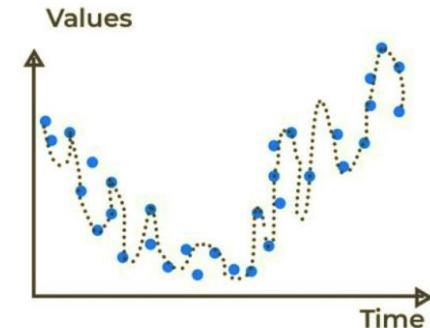


Evaluación de un modelo

2. Challenges

Overfitting

- Nuestra predicción se ajusta “demasiado bien” a nuestros datos, por lo que el training loss decrece
- Sin embargo, no es capaz de generalizar, por lo que el validation loss crece
- El momento en el que nuestra training loss baje, pero la validation loss crezca, estamos haciendo **overfitting**

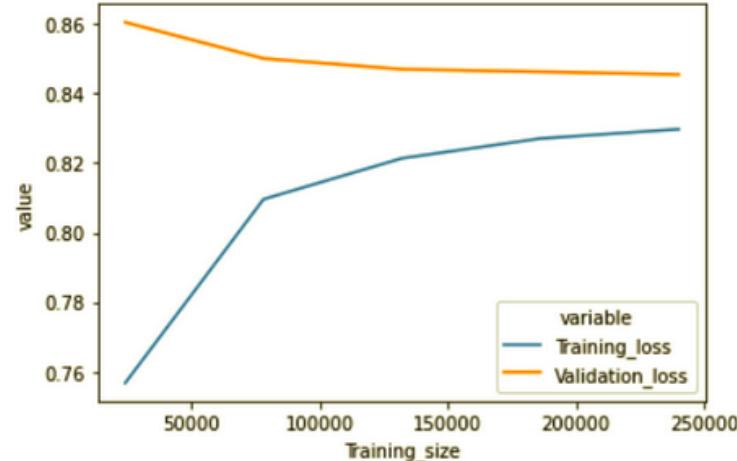
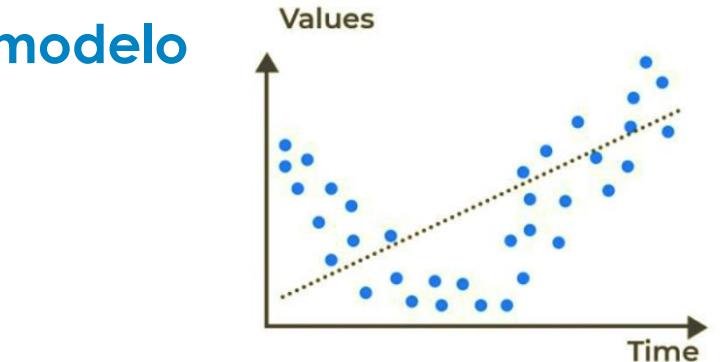


Evaluación de un modelo

2. Challenges

Underfitting

- Nuestra predicción no es capaz de aprender los patrones de nuestros datos y de ajustarse a ellos
- Tanto el validation como el training loss son bajos
- Esto puede deberse a que nuestro modelo es demasiado sencillo

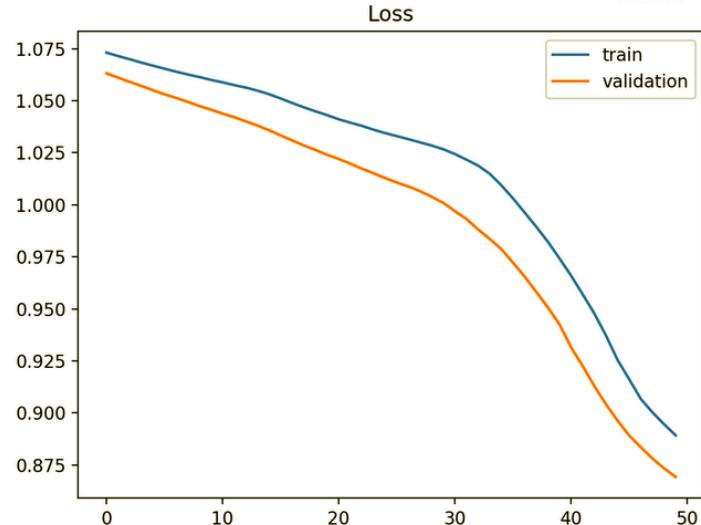
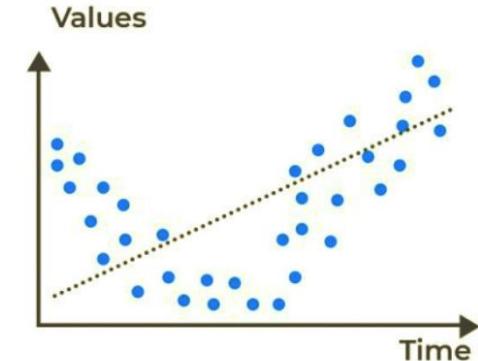


Evaluación de un modelo

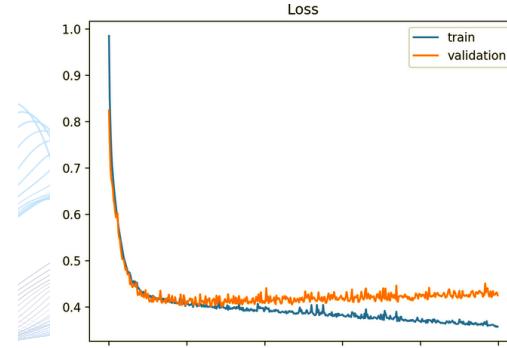
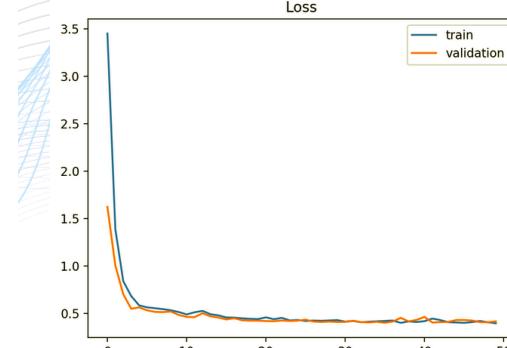
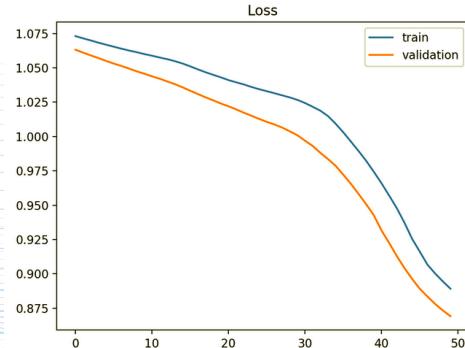
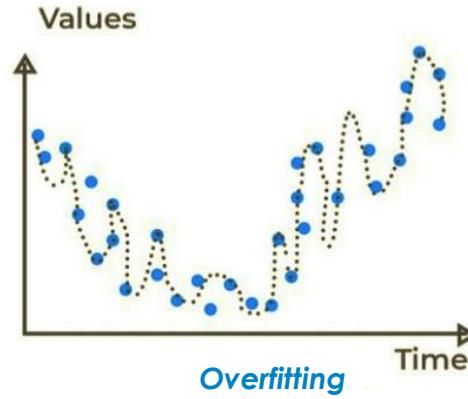
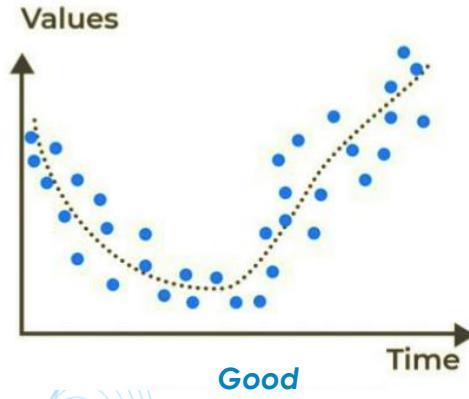
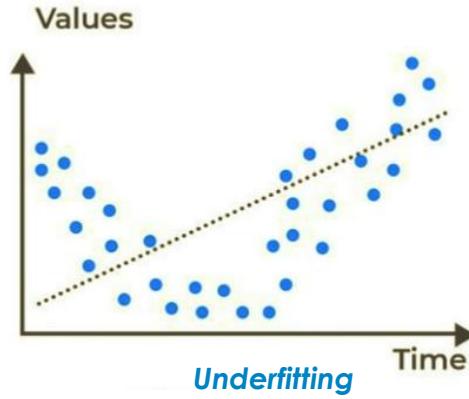
2. Challenges

Underfitting

- Nuestra predicción no es capaz de aprender los patrones de nuestros datos y de ajustarse a ellos
- Tanto el validation como el training loss son bajos
- Esto puede deberse a que nuestro modelo es demasiado sencillo

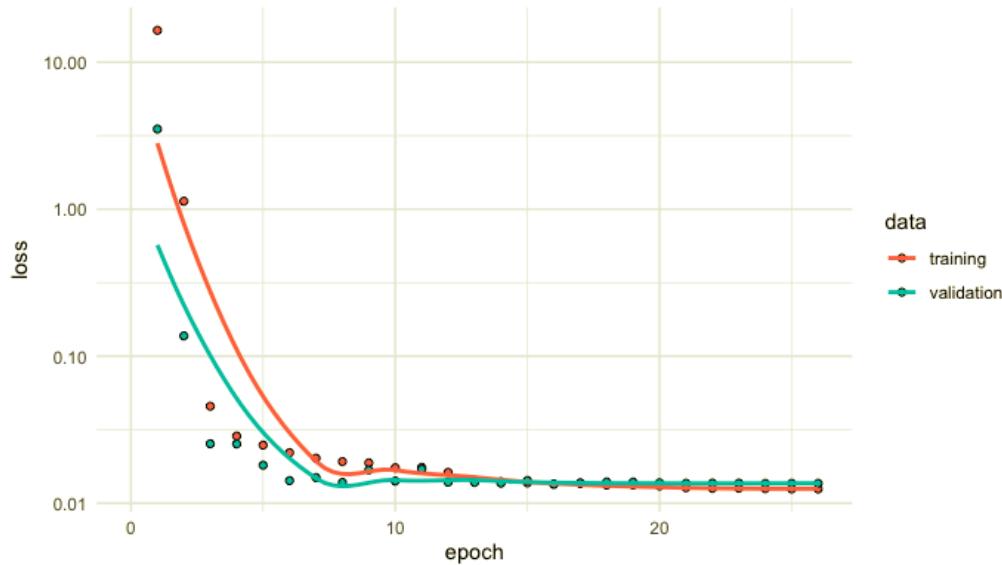


Evaluación de un modelo

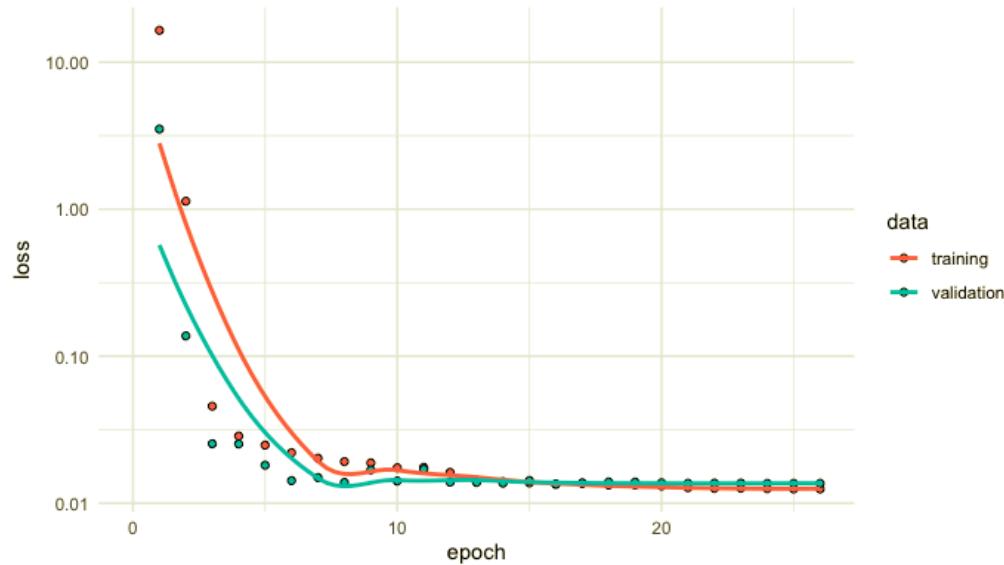


PAUSE II

Evaluación de un modelo



Evaluación de un modelo

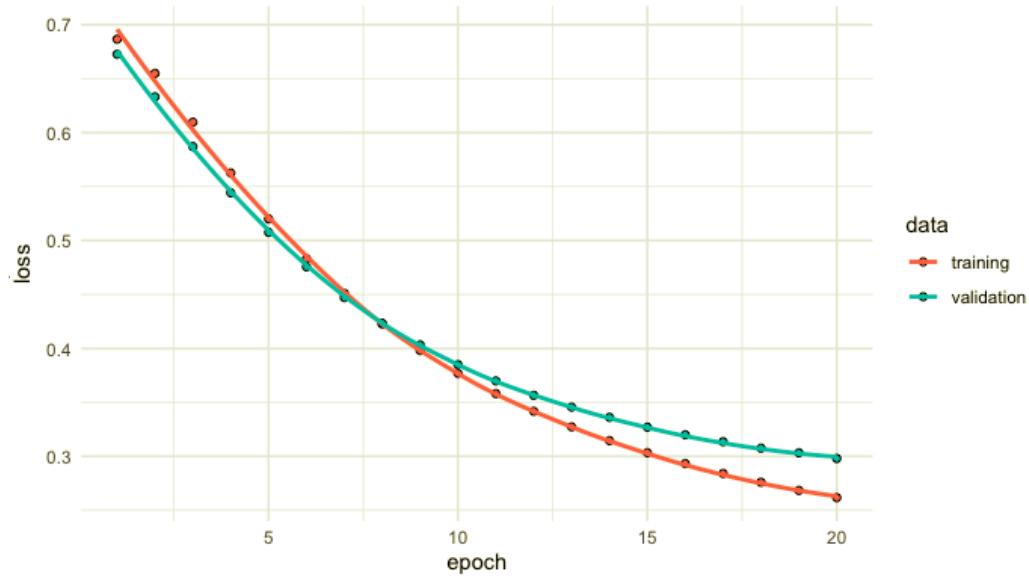


Buen entrenamiento

La training loss decrece y se establece, y la de validación hace lo mismo, sin crecer a partir de un epoch.

PAUSE II

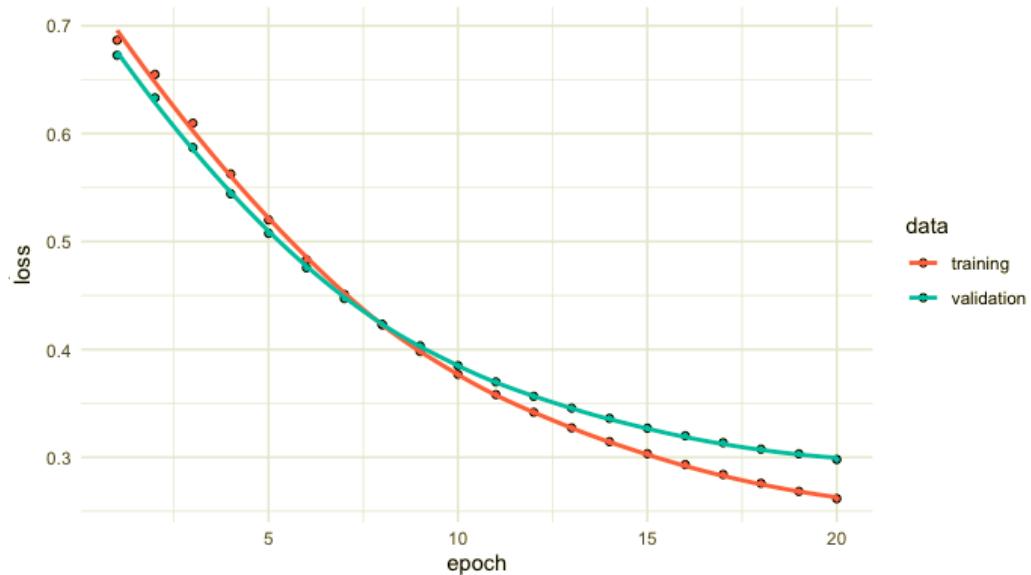
Evaluación de un modelo



data
—●— training
—●— validation

?

Evaluación de un modelo



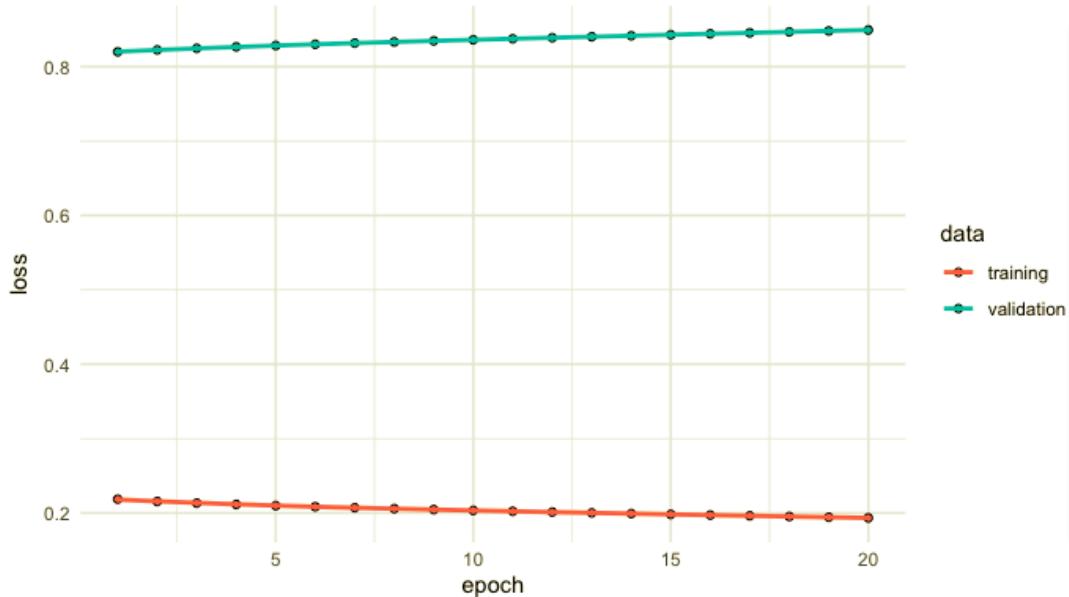
Underfitting

Tanto la training como la validation loss decrecen continuamente, lo que indica que el modelo no es capaz de aprender de los datos.

Podemos aumentar el número de epochs, incrementar el learning rate...

PAUSE II

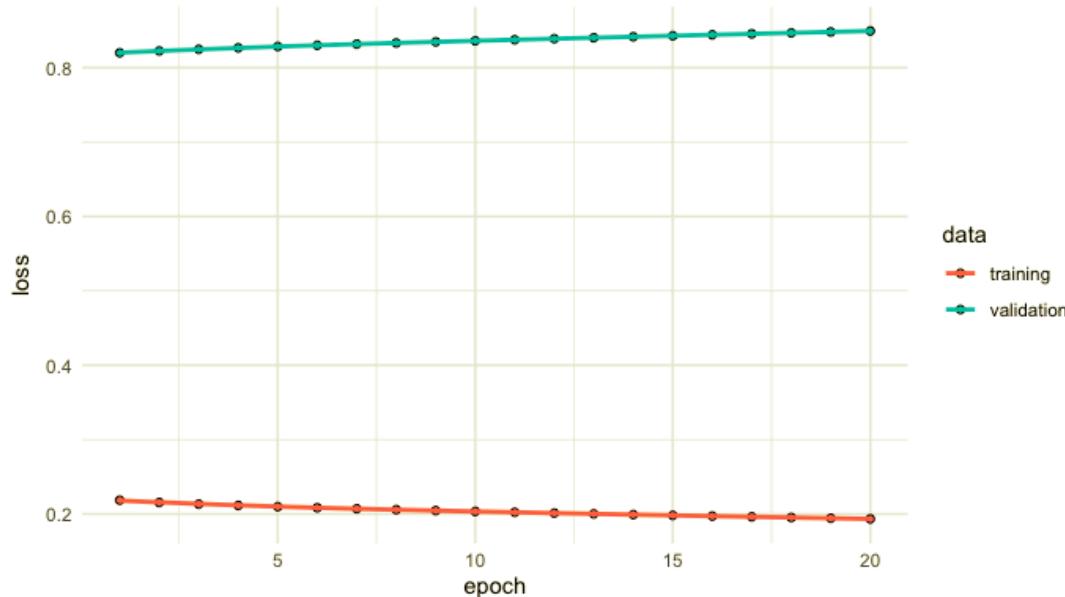
Evaluación de un modelo



data
—●— training
—●— validation



Evaluación de un modelo



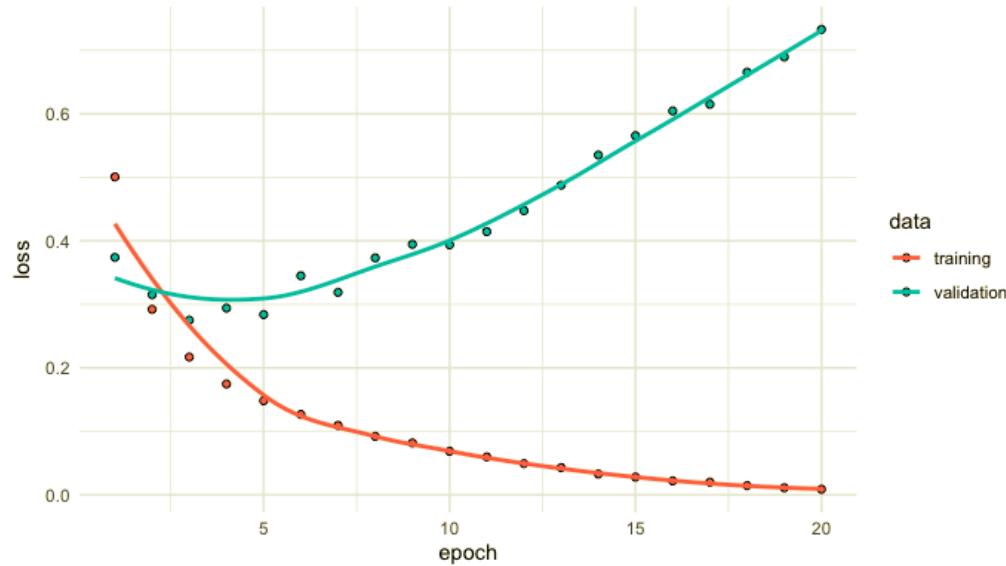
Underfitting

La training loss muestra una curva plana (no hay aprendizaje porque no hay una reducción del error). A la validation loss le ocurre lo mismo, pero además el error es mayor porque los datos no son los de entrenamiento.

Podemos incrementar el dataset, complicar el modelo...

PAUSE II

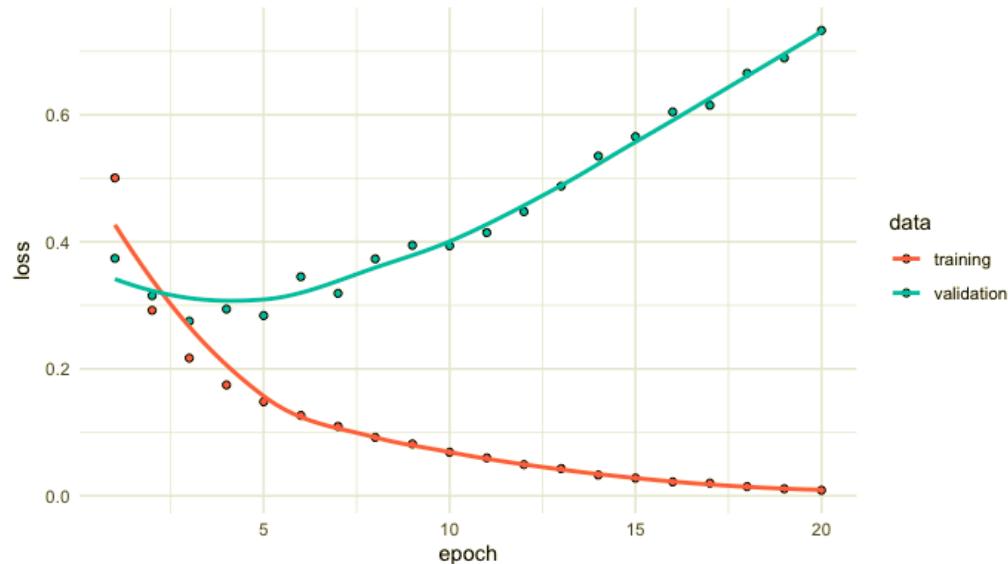
Evaluación de un modelo



data
—●— training
—●— validation



Evaluación de un modelo



Overfitting

La training loss decrece, pero la validation loss crece, por lo que el modelo se ajusta bien a los datos de entrenamiento, pero no es capaz de generalizar.

Podemos simplificar el modelo, reducir el learning rate,

Evaluación de un modelo

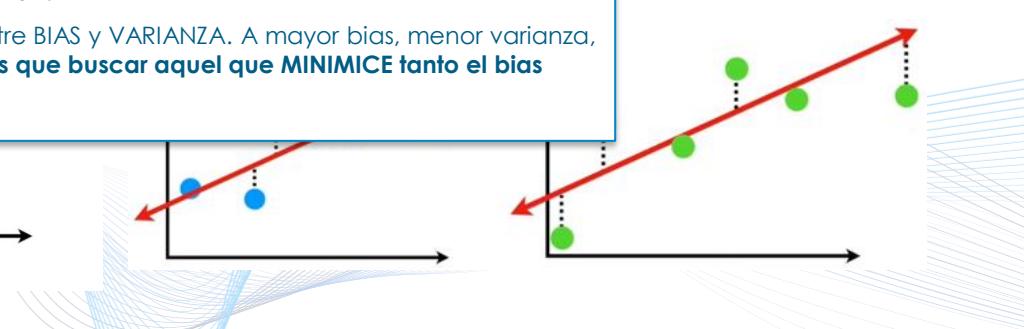
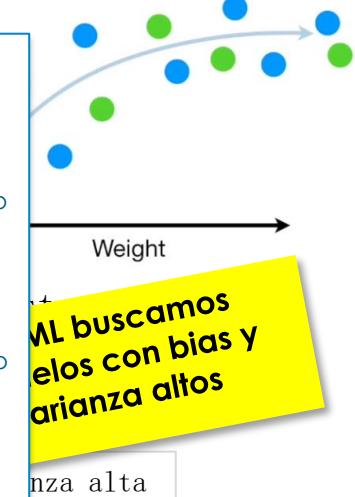
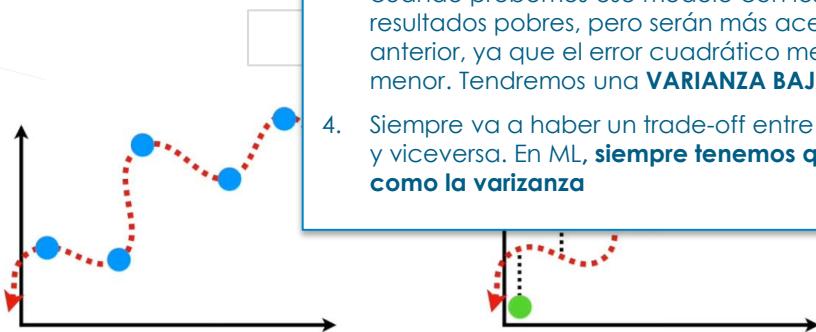
3. ¿Cómo evaluar un modelo?

Antes tenemos los datos

- **Bias:** inabilidad del modelo de ajustarse a la tasa de variación de los datos
- **Varianza:** diferencia entre las predicciones del modelo para los datos de entrenamiento y los datos de prueba

¡ATENCIÓN, DIAPOSITIVA Y VÍDEO INCORRECTO!

1. Los datos **azules** son **train/validation** y los **verdes**, **test**
2. Si el modelo se ajusta muy bien a los datos de entrenamiento, tendremos **BAJO BIAS**. Sin embargo, si evaluamos el modelo con los datos de test, veremos que el modelo no es capaz de generalizar, y por tanto la diferencia (error cuadrático medio) entre la predicción del modelo y el punto de test, será muy alta. Hablamos de que tendremos una **VARIANZA ALTA**. Esto pasará en el caso de modelos complejos, como K-vecinos, random forest...
3. Por el contrario, si el modelo no se ajusta tan bien a los datos de entrenamiento (como una regression lineal por ejemplo), el error de **BIAS** será **más ALTO**. Sin embargo, cuando probemos ese modelo con los datos de entrenamiento, también dará resultados pobres, pero serán más acertados que los que obteníamos en el caso anterior, ya que el error cuadrático medio entre cada predicción y el set test será menor. Tendremos una **VARIANZA BAJA**.
4. Siempre va a haber un trade-off entre **BIAS** y **VARIANZA**. A mayor bias, menor varianza, y viceversa. En ML, **siempre tenemos que buscar aquel que MINIMICE tanto el bias como la varianza**

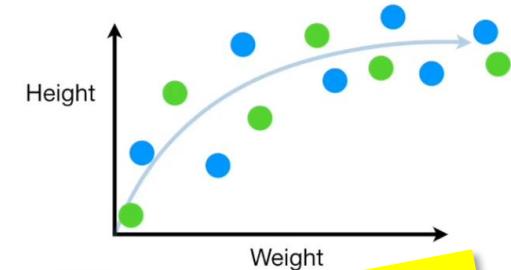


Evaluación de un modelo

3. ¿Cómo evitamos el overfitting?

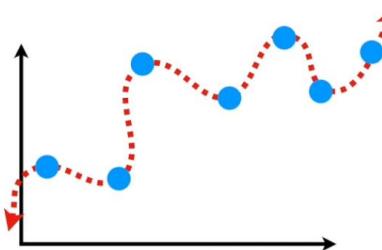
Antes tenemos que entender dos conceptos: bias y varianza

- **Bias:** inhabilidad de un modelo (como regresión lineal) de ajustarse a la tendencia de unos datos
- **Varianza:** diferencia de ajuste entre el set de test y de validación

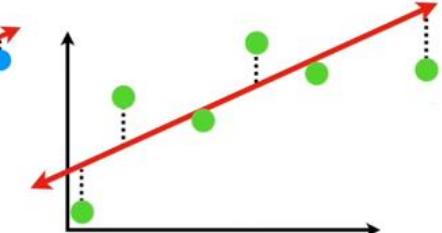
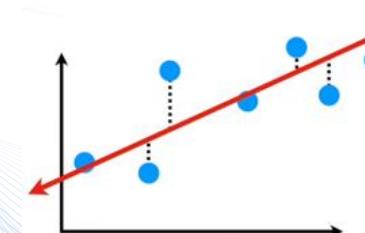
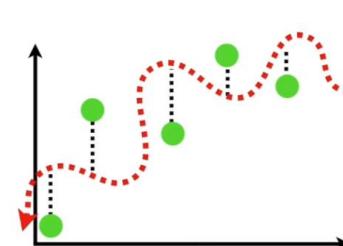


En ML buscamos
modelos con bias y
varianza bajos

Bias bajo, varianza alta



Bias alto, varianza baja



Evaluación de un modelo

3. ¿Cómo evitamos el overfitting?

Regularización: La regularización es una técnica de optimización que nos ayuda a evitar el overfitting

Consiste en añadir un término (penalización) en la función de coste para que nuestra predicción se aleje (nosotros decidimos cuánto) de los datos de entrenamiento. También para eliminar features “inútiles”

Tipos:

- **Ridge (L2) regression**
- **Lasso (L1) regression**
- Otros:

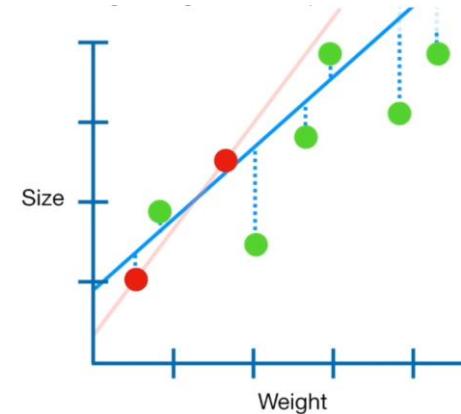
- **Elastic Net, Early stopping...**

$$L_{ridge}(\hat{\beta}) = \sum_{i=1}^n (y_i - x_i^T \hat{\beta})^2 + \lambda \sum_{j=1}^m \hat{\beta}_j^2 = \|y - X\hat{\beta}\|^2 + \lambda \|\hat{\beta}\|^2.$$

$= \text{mínimos cuadrados} + \lambda \cdot \text{slope}^2$

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^n (y_i - x_i^T \hat{\beta})^2 + \lambda \sum_{j=1}^m |\hat{\beta}_j|.$$

$= \text{mínimos cuadrados} + \lambda \cdot |\text{slope}|$



Penalización

Evaluación de un modelo

3. ¿Cómo evitamos el overfitting?

Regularización: La regularización es una técnica de optimización que nos ayuda a evitar el overfitting

Consiste en añadir un término (penalización) en la función de coste para que nuestra predicción se aleje (nosotros decidimos cuánto) de los datos de entrenamiento. También para eliminar features “inútiles”

Tipos:

- **Ridge (L2) regression**
- **Lasso (L1) regression**
- Otros:

- **Elastic Net, Early stopping...**

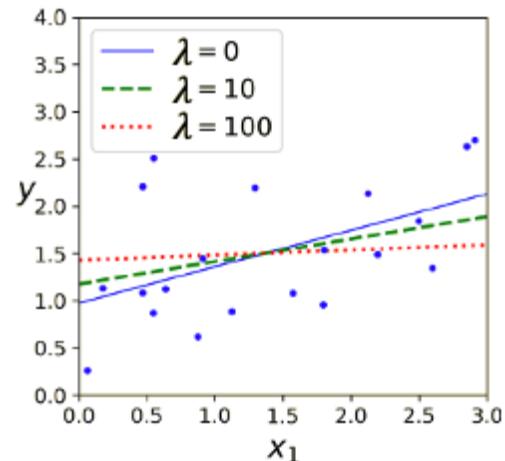
$$L_{ridge}(\hat{\beta}) = \sum_{i=1}^n (y_i - x_i^T \hat{\beta})^2 + \lambda \sum_{j=1}^m \hat{\beta}_j^2 = \|y - X\hat{\beta}\|^2 + \lambda \|\hat{\beta}\|^2.$$

$= \text{mínimos cuadrados} + \lambda \cdot \text{slope}^2$

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^n (y_i - x_i^T \hat{\beta})^2 + \lambda \sum_{j=1}^m |\hat{\beta}_j|.$$

$= \text{mínimos cuadrados} + \lambda \cdot |\text{slope}|$

Penalización



Evaluación de un modelo

3. ¿Cómo evitamos el overfitting?

Regularización: La regularización es una técnica de optimización que nos ayuda a evitar el overfitting

Consiste en añadir un término (penalización) en la función de coste para que nuestra predicción se aleje (nosotros decidimos cuánto) de los datos de entrenamiento. También para eliminar features “inútiles”

Tipos:

- **Ridge (L2) regression**
- **Lasso (L1) regression**
- Otros:

- **Elastic Net, Early stopping...**

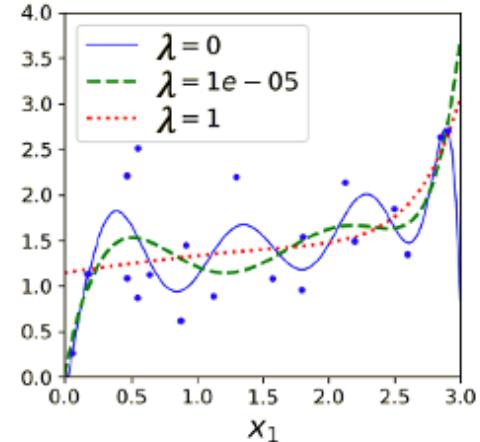
$$L_{ridge}(\hat{\beta}) = \sum_{i=1}^n (y_i - x_i^T \hat{\beta})^2 + \lambda \sum_{j=1}^m \hat{\beta}_j^2 = \|y - X\hat{\beta}\|^2 + \lambda \|\hat{\beta}\|^2.$$

$= \text{mínimos cuadrados} + \lambda \cdot \text{slope}^2$

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^n (y_i - x_i^T \hat{\beta})^2 + \lambda \sum_{j=1}^m |\hat{\beta}_j|.$$

$= \text{mínimos cuadrados} + \lambda \cdot |\text{slope}|$

Penalización



Tipos de algoritmos de Machine Learning

1. Supervisado:

- 1.1. **Clasificación:** k-NN, regresión logística, Decision trees y Random forest, SVM
- 1.2. **Regresión:** reg. Lineal y polinómica, SVM

2. No-Supervisado (unlabeled)

- 2.1. **Clustering** (K-means)
- 2.2. **Reducción de dimensionalidad**

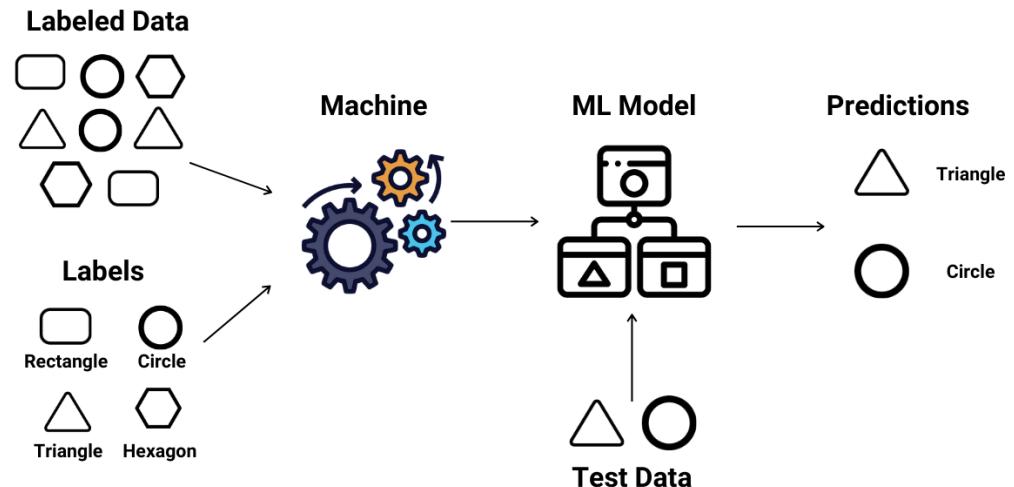
3. Semi-supervisado

Tipos de algoritmos de Machine Learning

1. Supervisado

- **Datos etiquetados**
- Tipos de aprendizaje supervisado:
 - **Clasificación:** se obtienen datos por categorías (clases)
 - **Regresión:** se obtienen números (no clases)

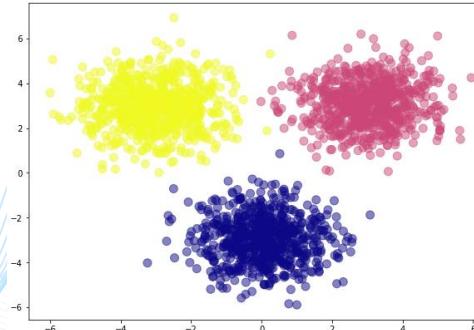
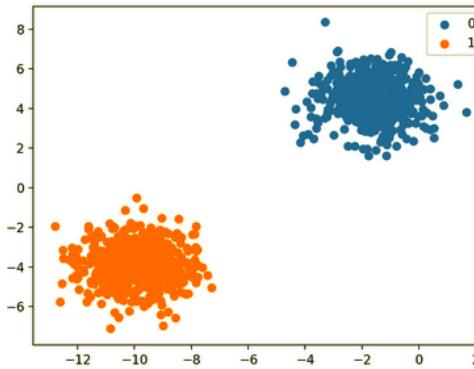
Supervised Learning



Tipos de algoritmos de Machine Learning

1.1. Clasificación

- **Predicen a qué clase (categoría) pertenece un objeto**
- Puede ser:
 - **Binaria** (dos clases)
 - **Multi-clase** (más de dos clases)
- Los algoritmos más utilizados son:
 - k-Nearest Neighbor (KNN)
 - Regresión logística
 - Support Vector Machines (SVM)
 - Árboles de decisión

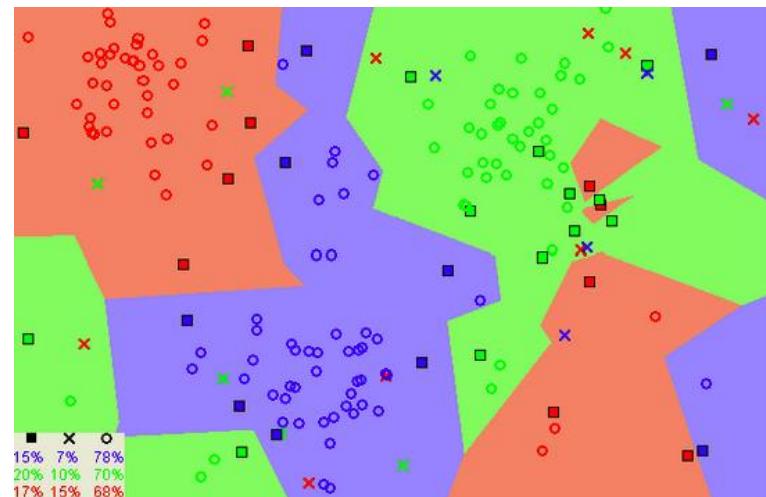


Tipos de algoritmos de Machine Learning

1.1. Clasificación

K-Nearest Neighbour (KNN)

- Aprovecha que puntos pertenecientes a la misma categoría pueden ser cercanos
- Se seleccionan **K** puntos vecinos a un punto (el que se quiere clasificar). Se calculan las distancias (L2) de cada uno de los K vecinos a ese punto. La clase que tenga más vecinos cercanos, será a la que pertenezca ese punto (**votación**).

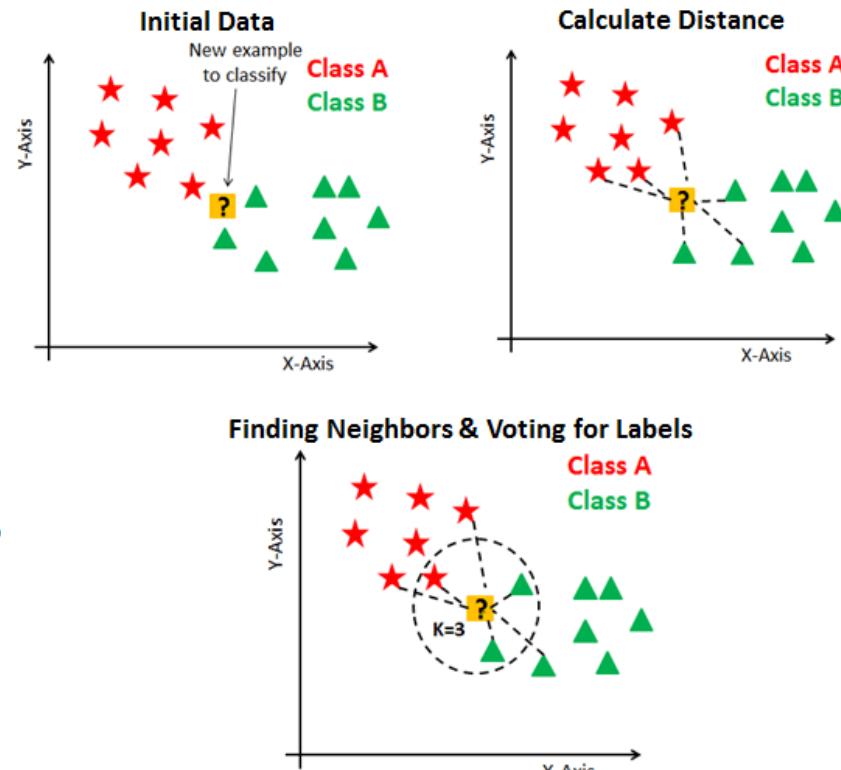


Tipos de algoritmos de Machine Learning

1.1. Clasificación

K-Nearest Neighbour (KNN)

- Aprovecha que puntos pertenecientes a la misma categoría pueden ser cercanos
- Se seleccionan **K** puntos vecinos a un punto (el que se quiere clasificar). Se calculan las distancias (L2) de cada uno de los **K** vecinos a ese punto. La clase que tenga más vecinos cercanos, será a la que pertenezca ese punto (**votación**).

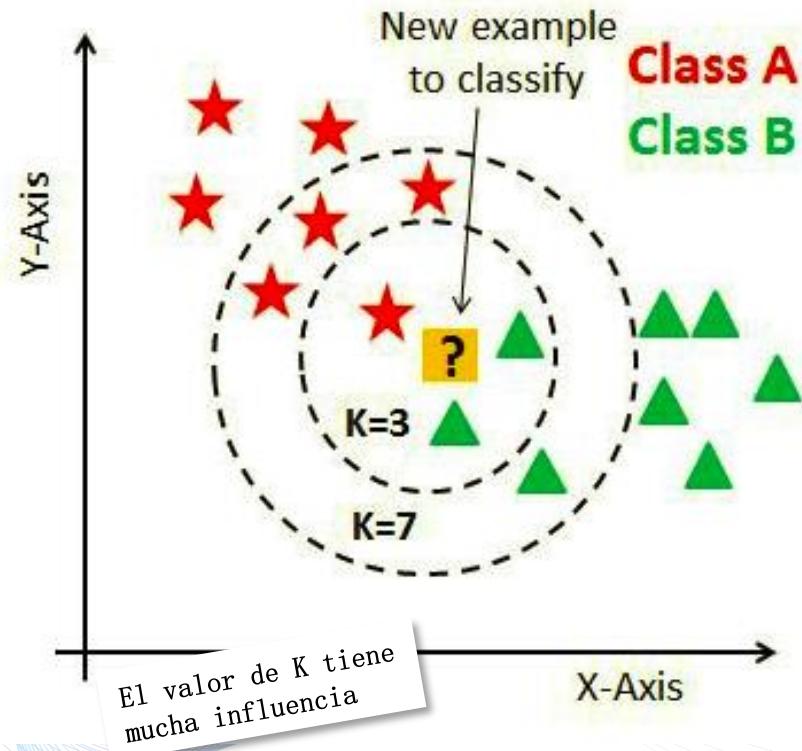


Tipos de algoritmos de Machine Learning

1.1. Clasificación

K-Nearest Neighbour (KNN)

- Aprovecha que puntos pertenecientes a la misma categoría pueden ser cercanos
- Se seleccionan **K** puntos vecinos a un punto (el que se quiere clasificar). Se calculan las distancias (L2) de cada uno de los **K** vecinos a ese punto. La clase que tenga más vecinos cercanos, será a la que pertenezca ese punto (**votación**).

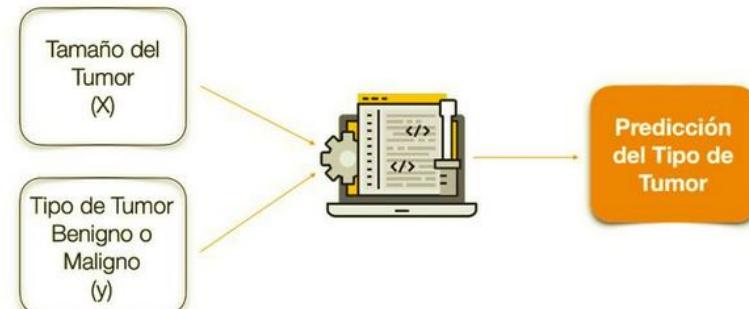


Tipos de algoritmos de Machine Learning

1.1. Clasificación

Regresión logística

- Se utiliza en **clasificación binaria**, sobre datos que pueden ser 1 o 0
- Identifica la probabilidad de ocurrencia de un evento con una de las dos categorías
- Describe y estima la relación entre una **variable binaria dependiente** y las **variables independientes**.
- Ejemplo: spam detector, cancer/no-cancer, etc.

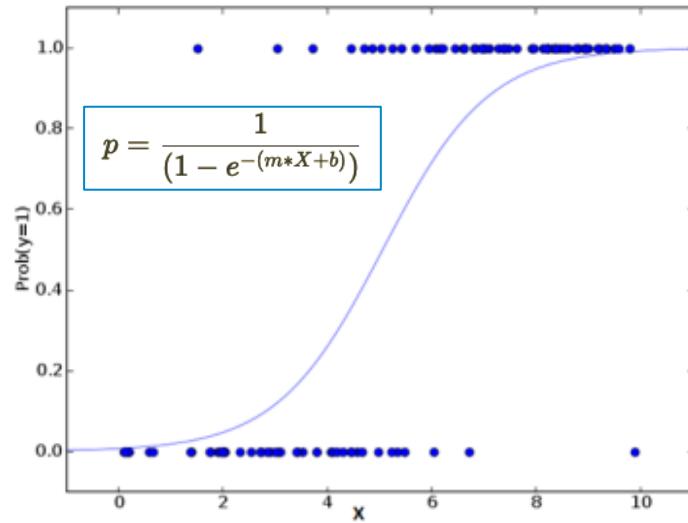


Tipos de algoritmos de Machine Learning

1.1. Clasificación

Regresión logística

- Es una regresión lineal que está modelada por la función sigmoide
- Tenemos que parametrizar nuestra función con los parámetros que optimicen la función de coste
- En este caso la más utilizada es la función **log loss**



$$\hat{y} = \sigma(w^T x + b) = \sigma(z)$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Tipos de algoritmos de Machine Learning

1.1. Clasificación

Regresión logística

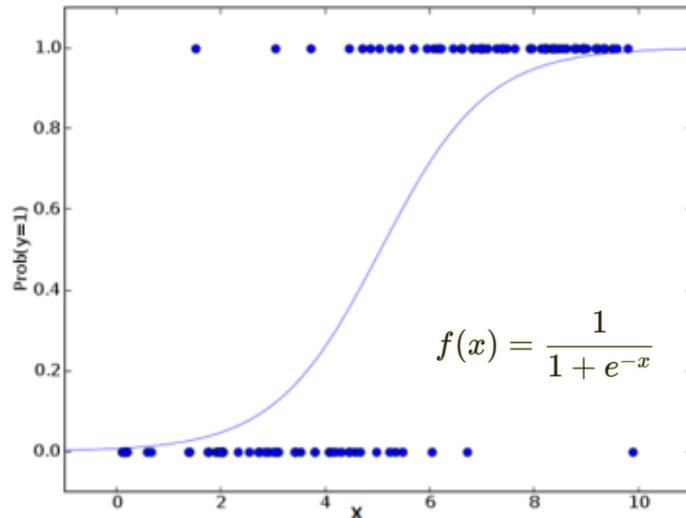
- Fución de coste: *log loss (Binary Cross-Entropy)*

Busca el valor que optimice el número de clases acertada.

Equation 4-17. Logistic Regression cost function (log loss)

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)}) \right]$$

Clase positiva ($y = 1$) Clase negativa ($y = 0$)



$$\hat{y} = \sigma(w^T x + b) = \sigma(z)$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

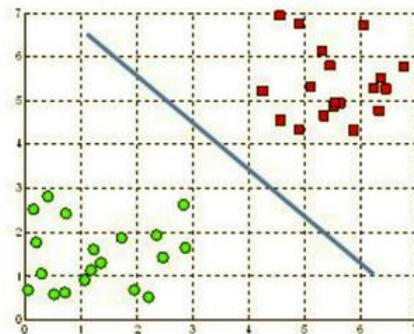
Tipos de algoritmos de Machine Learning

1.1. Clasificación

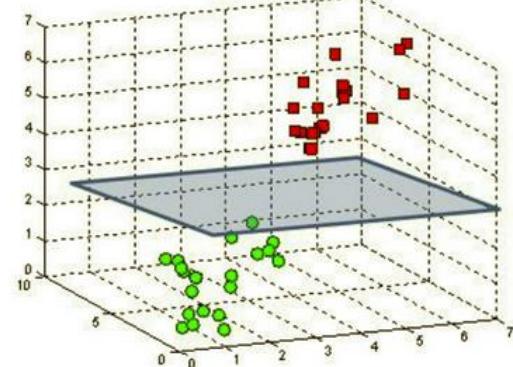
Support Vector Machines (SVM)

- Máquinas de vectores de soporte
- Se puede utilizar tanto en regresión como clasificación
- Objetivo: encontrar la mejor separación posible entre clases buscando el **hiperplano** que maximize el margen de separación entre ambas.

A hyperplane in \mathbb{R}^2 is a line



A hyperplane in \mathbb{R}^3 is a plane

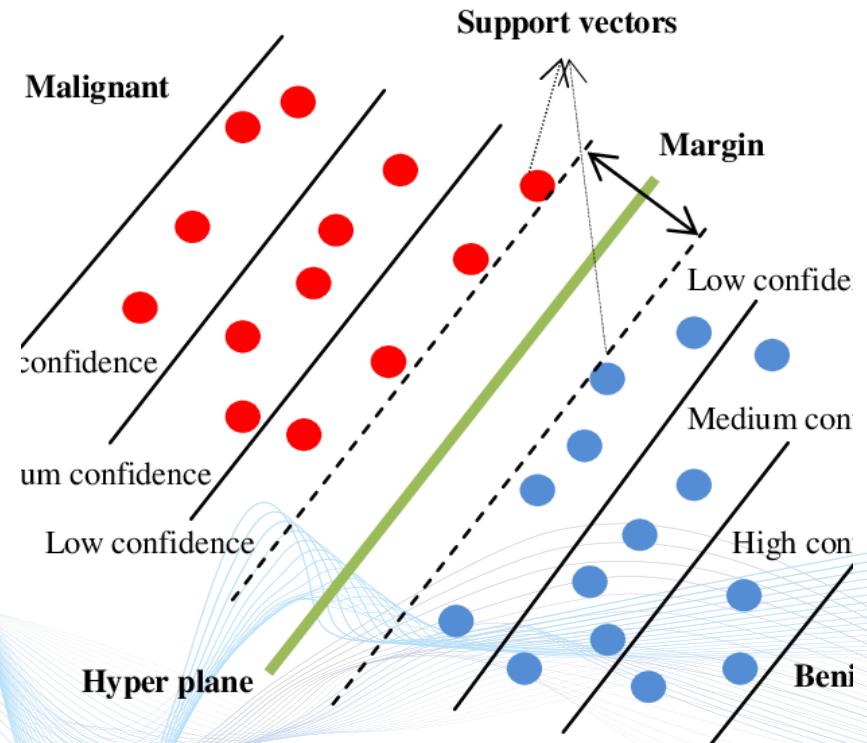


Tipos de algoritmos de Machine Learning

1.1. Clasificación

Support Vector Machines (SVM)

- Máquinas de vectores de soporte
- Se puede utilizar tanto en regresión como clasificación
- Objetivo: encontrar la mejor separación posible entre clases buscando el **hiperplano** que maximice el margen de separación entre ambas.

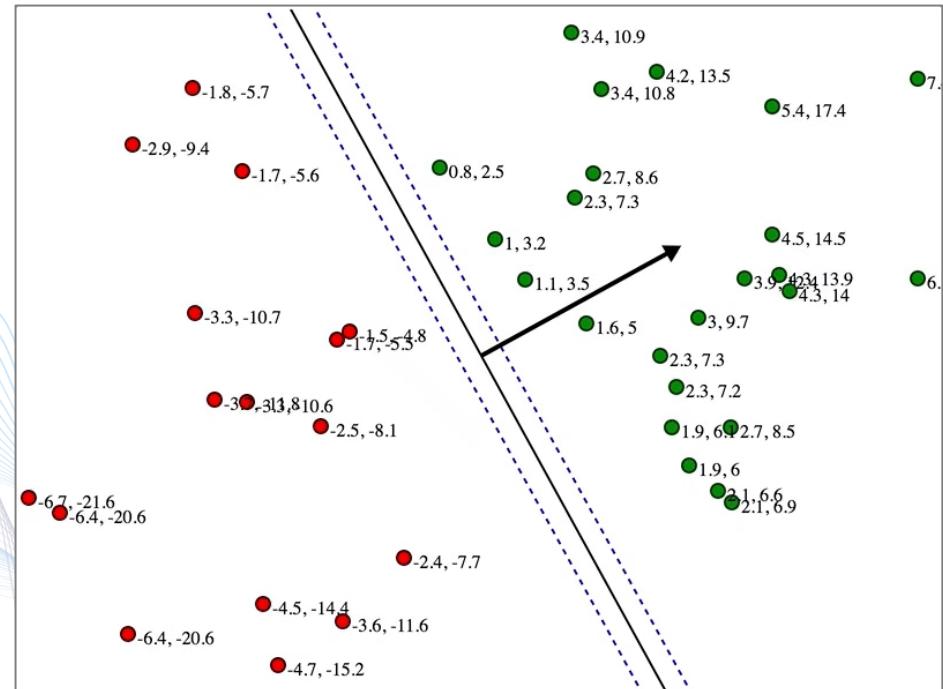


Tipos de algoritmos de Machine Learning

1.1. Clasificación

Support Vector Machines (SVM)

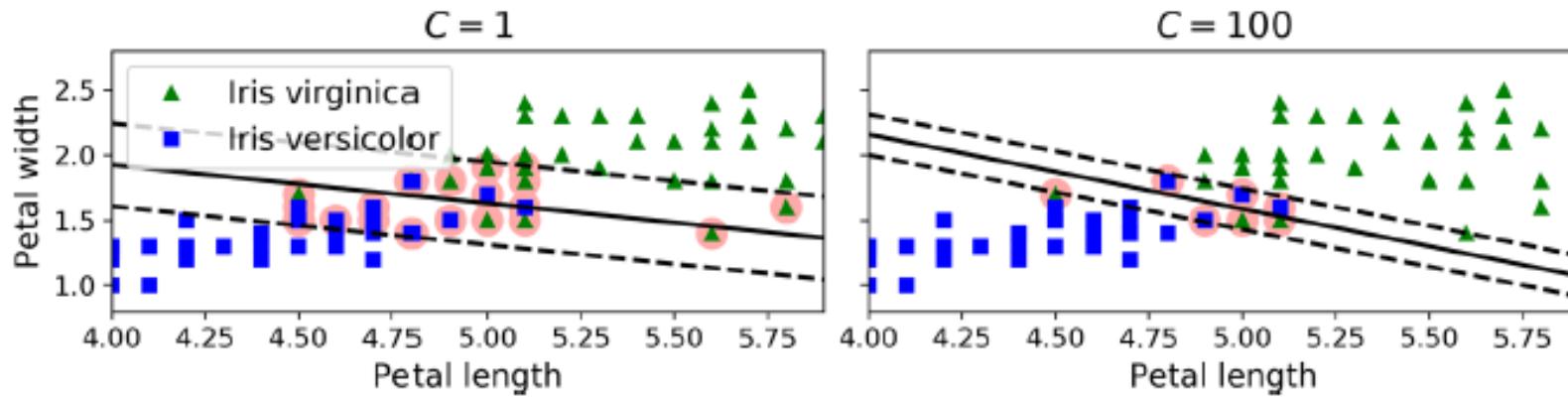
- Máquinas de vectores de soporte
- Se puede utilizar tanto en regresión como clasificación
- Objetivo: encontrar la mejor separación posible entre clases buscando el **hiperplano** que maximize el margen de separación entre ambas.



Tipos de algoritmos de Machine Learning

1.1. Clasificación

Support Vector Machines (SVM)



El hiperparámetro "c" nos ayuda a establecer un balance entre cuánto abrimos los márgenes y cuantos puntos "erroneos" abarcamos. Es un parámetro de **regularización**

Tipos de algoritmos de Machine Learning

1.1. Clasificación

Support Vector Machines (SVM)

- Fución de coste: **Hinge loss**

$$l(y) = \max(0, 1 - t \cdot y)$$

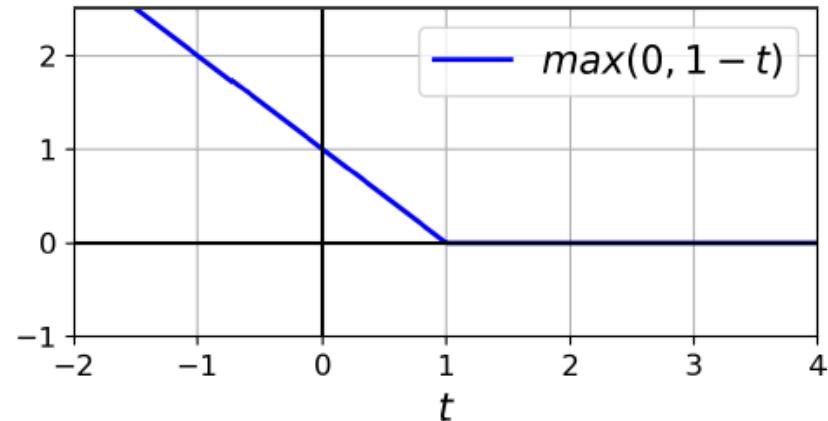
Ejemplo:

- observación = 1 (t)
- Predicción = 0.5 (y)
- Loss = 0.5

$$l(y) = \max(0, 1 - 1 \cdot 0.5) = 0.5$$

- observación = -1 (t)
- Predicción = 0.5 (y)
- Loss = 0.5

$$l(y) = \max(0, 1 - (-1) \cdot 0.5) = 1.5$$



Tipos de algoritmos de Machine Learning

1.1. Clasificación

Support Vector Machines (SVM)

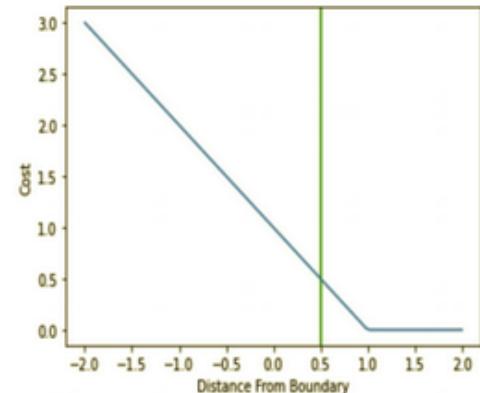
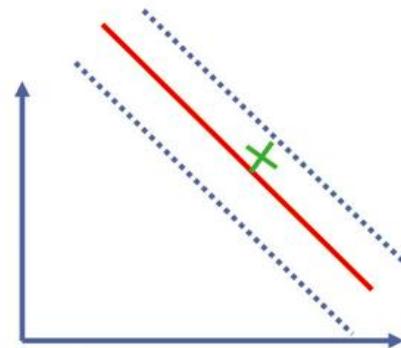
- Fución de coste: *Hinge loss*

$$l(y) = \max(0, 1 - t \cdot y)$$

Ejemplo:

- observación = 1 (t)
- Predicción = 0.5 (y)
- Loss = 0.5

$$l(y) = \max(0, 1 - 1 \cdot 0.5) = 0.5$$



Tipos de algoritmos de Machine Learning

1.1. Clasificación

Support Vector Machines (SVM)

- Fución de coste: **Hinge loss**

$$l(y) = \max(0, 1 - t \cdot y)$$

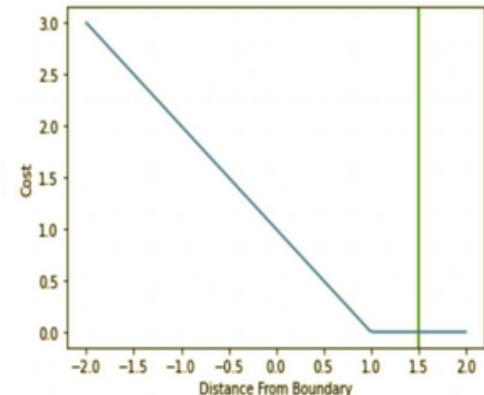
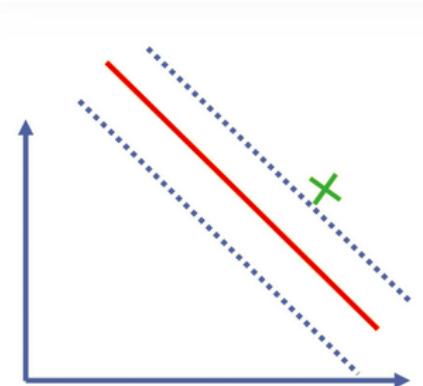
Ejemplo:

- observación = 1 (t)
- Predicción = 0.5 (y)
- Loss = 0.5

$$l(y) = \max(0, 1 - 1 \cdot 0.5) = 0.5$$

- observación = -1 (t)
- Predicción = 0.5 (y)
- Loss = 0.5

$$l(y) = \max(0, 1 - (-1) \cdot 0.5) = 1.5$$

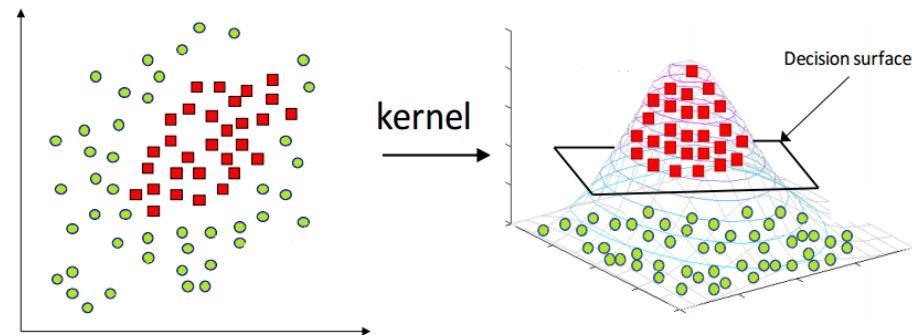


Tipos de algoritmos de Machine Learning

1.1. Clasificación

Support Vector Machines (SVM)

- Si los datos no son separables por un hiperplano de 2 dimensiones se utiliza un **kernel** para aumentar de dimensión
- Un kernel es una serie de funciones que toman un espacio de entrada de baja dimensión y **lo transforma en un espacio de mayor dimensión**, de modo que al realizar esta transformación convierten un problema no separable en un problema separable linealmente.
- Algunos kernel son: Linear kernel, Polynomial kernel, Radial Basis Function (RBF) kernel...

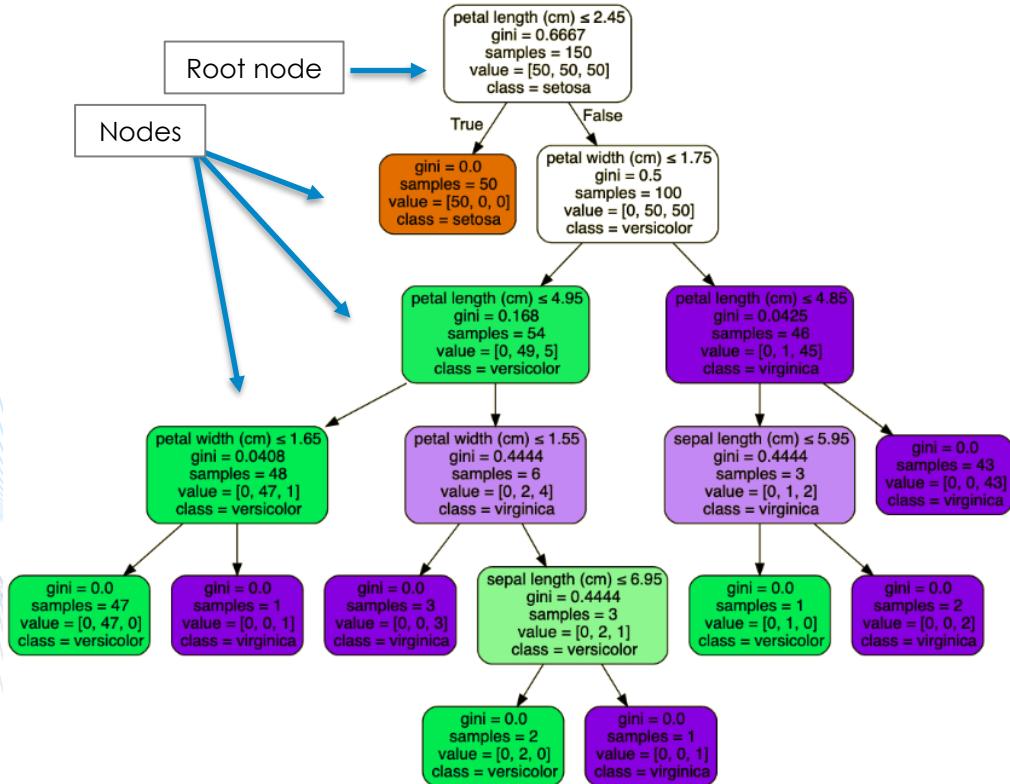


Tipos de algoritmos de Machine Learning

1.1. Clasificación

Decision Trees y Random Forest

- Un **árbol de decisión** (decision tree) se puede usar tanto para clasificación como para regression
- **Objetivo:** crear un modelo que prediga el valor de una variable target mediante el aprendizaje de reglas de decisión simples inferidas a partir de las características de los datos
- El **índice de gini** nos indica si un nodo es puro ($\text{gini} = 0$) o impuro ($\text{gini} > 0$). En el primer caso, no se puede separar más y la clase obtenida es la predicción



Tipos de algoritmos de Machine Learning

1.1. Clasificación

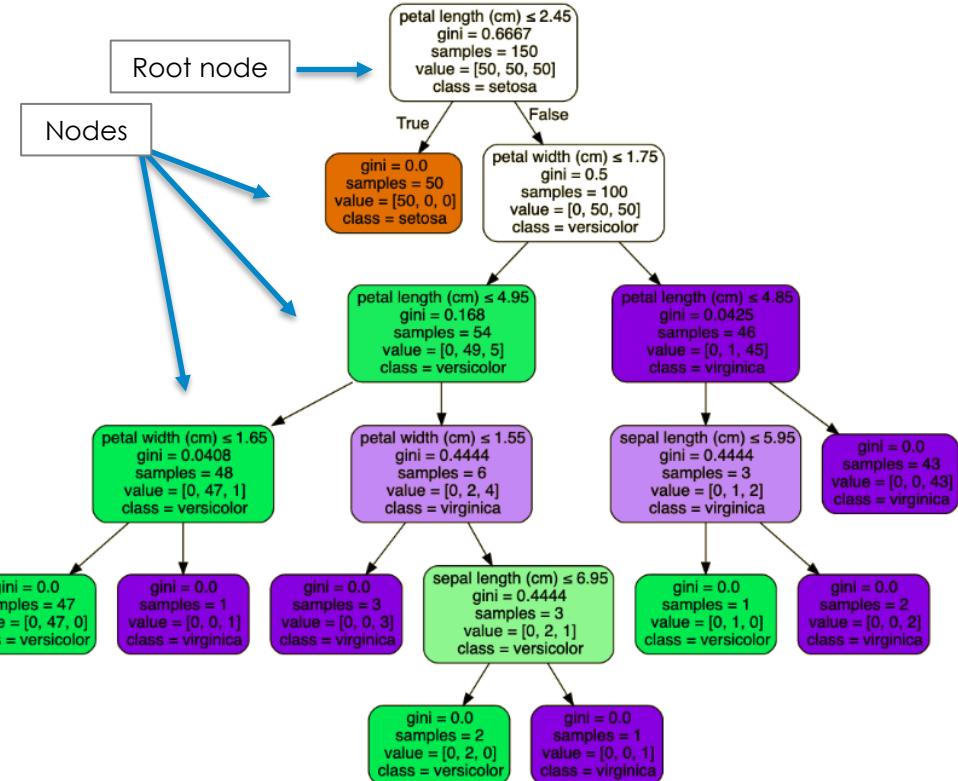
Decision Trees y Random Forest

- Función de coste: **CART (Classification and Regression Tree)**

Equation 6-2. CART cost function for classification

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where $\begin{cases} G_{\text{left/right}} \end{cases}$ measures the impurity of the left/right subset,
 $\begin{cases} m_{\text{left/right}} \end{cases}$ is the number of instances in the left/right subset.

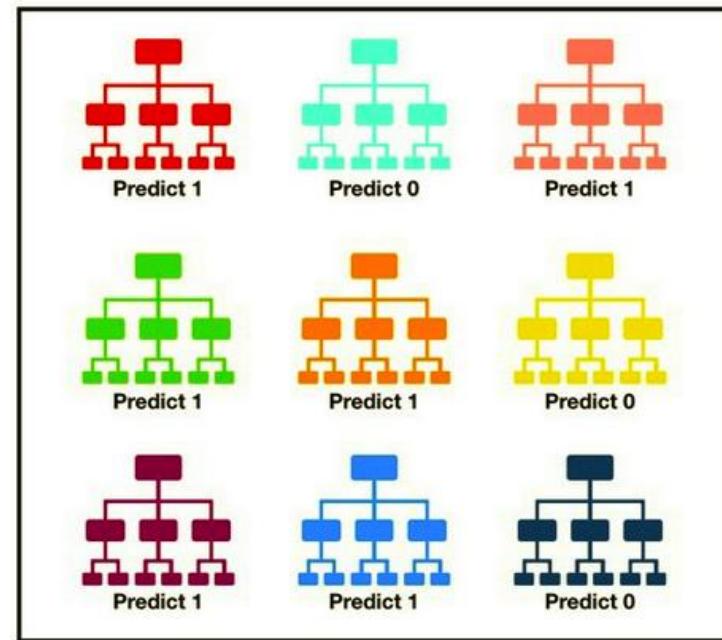


Tipos de algoritmos de Machine Learning

1.1. Clasificación

Decision Trees y Random Forest

- Un **random forest** está formado por un conjunto de árboles de decisión, cada uno encargado de analizar una clase
- Cada árbol individual arroja una predicción de clase y la clase con más **votos** se convierte en la predicción de nuestro modelo



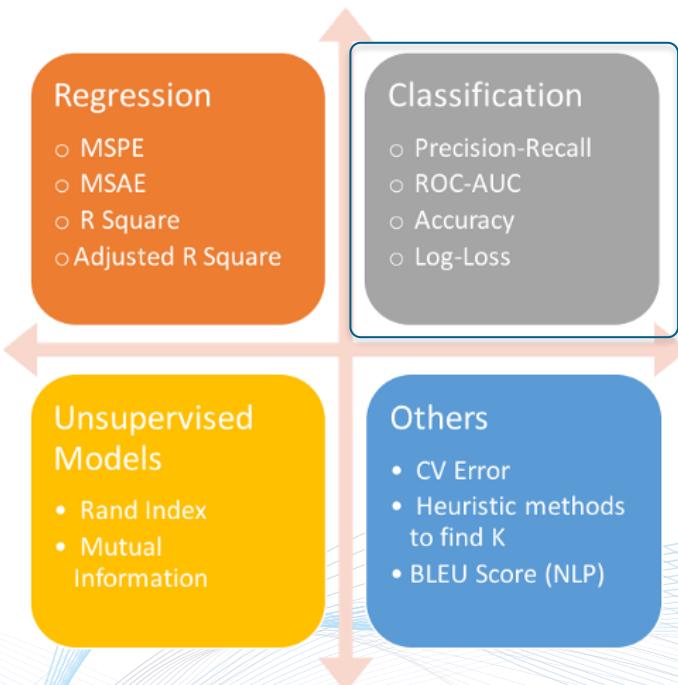
Prediction 1: 6

Prediction 0: 3

Tipos de algoritmos de Machine Learning

1.1. Clasificación

Métricas de evaluación (Clasificación)



Tipos de algoritmos de Machine Learning

1.1. Clasificación

Métricas de evaluación (Clasificación)

La matriz de confusión

- **Precisión**
- **Sensibilidad** (o *recall*): ratio de verdaderos positivos
- **Especificidad**: ratio de verdaderos negativos

| | | Predicted Class | | |
|--------------|----------|--|---|---|
| | | Positive | Negative | |
| Actual Class | Positive | True Positive (TP) | False Negative (FN) <i>Type II Error</i> | Sensitivity $\frac{TP}{(TP + FN)}$ |
| | Negative | False Positive (FP) <i>Type I Error</i> | True Negative (TN) | Specificity $\frac{TN}{(TN + FP)}$ |
| | | Precision $\frac{TP}{(TP + FP)}$ | Negative Predictive Value $\frac{TN}{(TN + FN)}$ | Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$ |

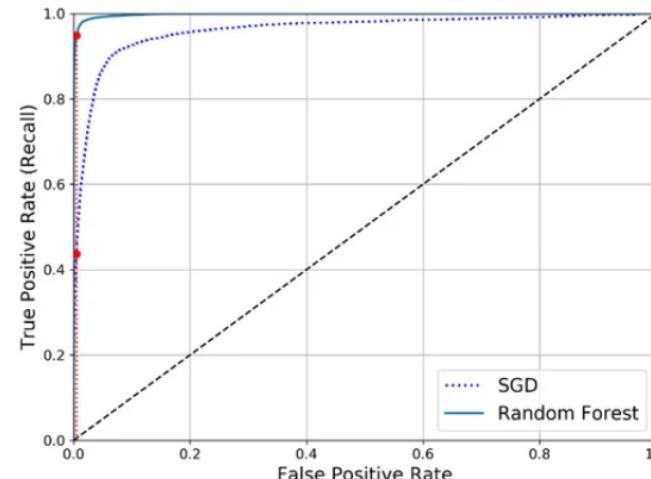
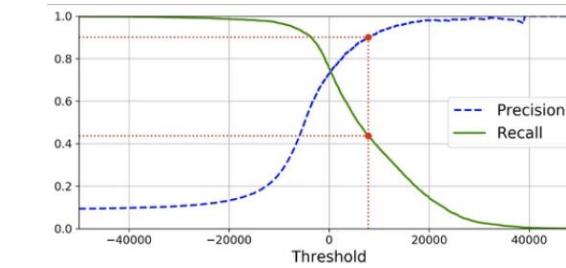
Tipos de algoritmos de Machine Learning

1.1. Clasificación

Métricas de evaluación (Clasificación)

La matriz de confusión

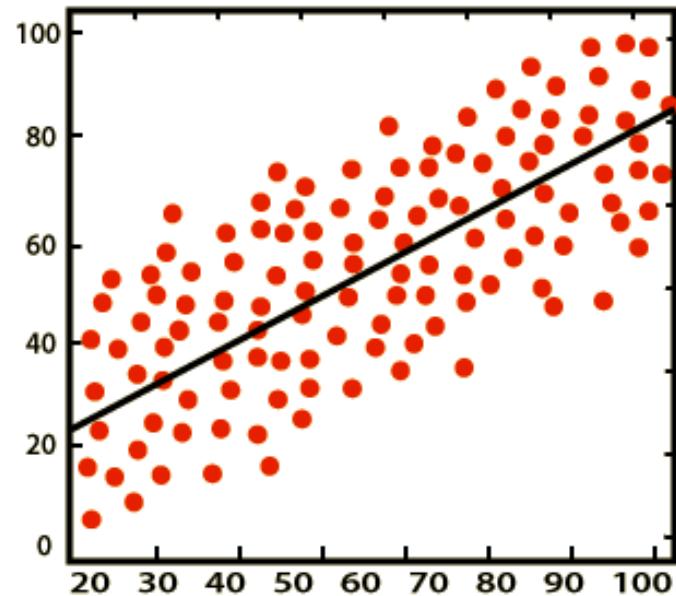
- **Precisión**
- **Sensibilidad** (o *recall*): ratio de verdaderos positivos
- **Especificidad**: ratio de verdaderos negativos
- Cada punto generado con la matriz de confusión para distintos umbrales genera la **Curva ROC** (Receiver Operating Characteristic curve)
- La curva ROC nos valdrá para comparar entre distintos modelos



Tipos de algoritmos de Machine Learning

1.2. Regresión

- Establecen una relación entre un cierto número de features y una **variable objetivo continua**.
- Los algoritmos más utilizados son:
 - Regresión lineal simple
 - Regresión polinómica
 - Support Vector Machines (SVM)
 - Árboles de decisión
- Ejemplo: predicción del precio del pan un día determinado de la semana



Tipos de algoritmos de Machine Learning

1.2. Regresión

$$Y = mX + b$$

Regresión lineal simple (aka *least squares*)

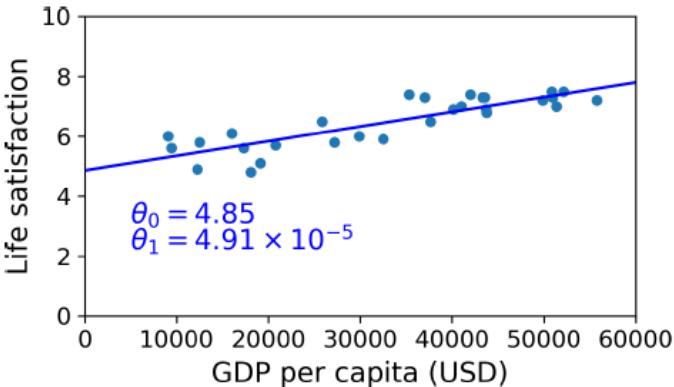
- Objetivo: modelar la relación entre una variable escalar **dependiente** "y" y una variable **independientes (o explicativas)** "X"
- Función de coste: RMSE, MSE, MAE...
- Optimizamos con el **descenso del gradiente**

$$\text{RMSE}(X, h) = \sqrt{\frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2}$$

$$\text{MSE}(X, h_{\theta}) = \frac{1}{m} \sum_{i=1}^m (\theta^T x^{(i)} - y^{(i)})^2$$

$$E = \frac{1}{n} \sum_{i=0}^n (y_i - \bar{y}_i)^2$$

$$\text{life_satisfaction} = \theta_0 + \theta_1 \times \text{GDP_per_capita}$$



MSE

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$m' = m_0 - \alpha \cdot 2 \cdot (Y'_i - Y) \cdot X$$

$$b' = b_0 - \alpha \cdot 2 \cdot (Y'_i - Y)$$

Tipos de algoritmos de Machine Learning

1.2. Regresión

Regresión lineal multiple (MLR)

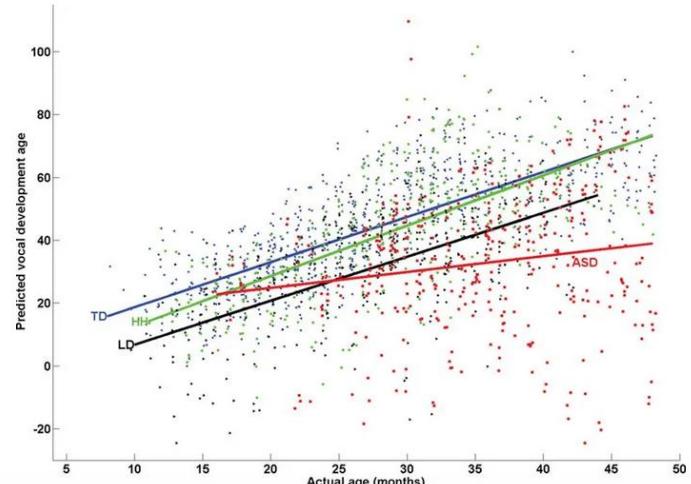
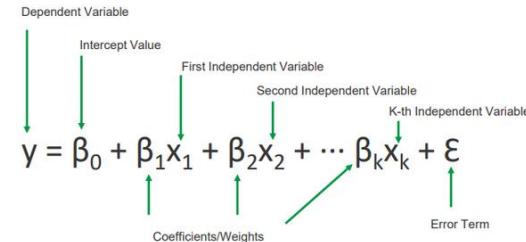
- Es una generalización del modelo de regresión lineal simple.
- Se relaciona una variable dependiente (Y') con k variables explicativas (x_1, x_2, \dots, x_k)
- Podemos tener n observaciones

$$y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{21} + \dots + \beta_k x_{k1} + \epsilon_1$$

$$y_2 = \beta_0 + \beta_1 x_{12} + \beta_2 x_{22} + \dots + \beta_k x_{k2} + \epsilon_2$$

...

$$y_n = \beta_0 + \beta_1 x_{1n} + \beta_2 x_{2n} + \dots + \beta_k x_{kn} + \epsilon_n$$



Tipos de algoritmos de Machine Learning

1.2. Regresión

Regresión lineal multiple (MLR)

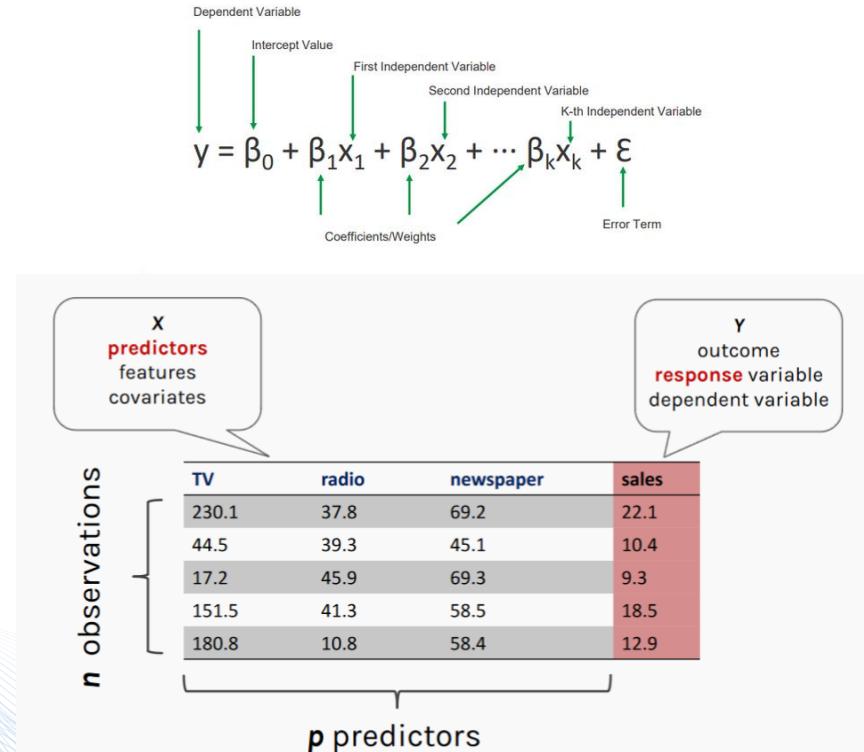
- Es una generalización del modelo de regresión lineal simple.
- Se relaciona una variable dependiente (Y') con k variables explicativas (x_1, x_2, \dots, x_k)
- Podemos tener n observaciones

$$y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{21} + \dots + \beta_k x_{k1} + \epsilon_1$$

$$y_2 = \beta_0 + \beta_1 x_{12} + \beta_2 x_{22} + \dots + \beta_k x_{k2} + \epsilon_2$$

...

$$y_n = \beta_0 + \beta_1 x_{1n} + \beta_2 x_{2n} + \dots + \beta_k x_{kn} + \epsilon_n$$



Tipos de algoritmos de Machine Learning

1.2. Regresión

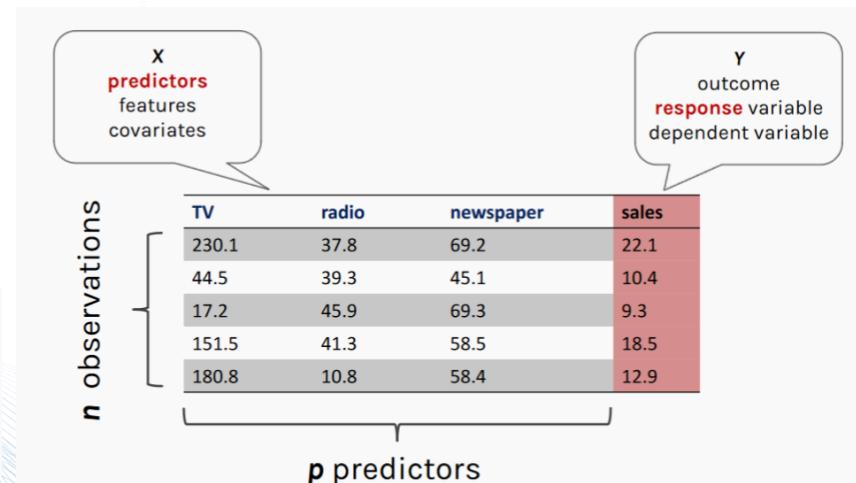
Regresión lineal multiple (MLR)

- Es una generalización del modelo de regresión lineal simple.
- Se relaciona una variable dependiente (Y') con k variables explicativas (x_1, x_2, \dots, x_k)
- Podemos tener n observaciones

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,J} \\ 1 & x_{2,1} & \dots & x_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,J} \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_J \end{pmatrix}$$

Dependent Variable
y = $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \epsilon$

Intercept Value
First Independent Variable
Second Independent Variable
K-th Independent Variable
Coefficients/Weights
Error Term

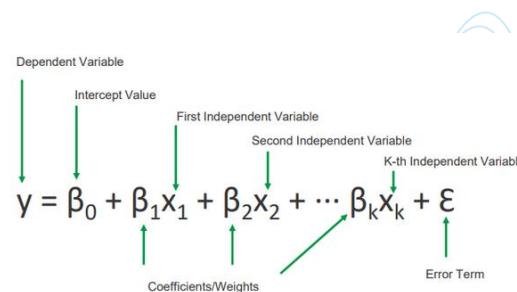


Tipos de algoritmos de Machine Learning

1.2. Regresión

Regresión lineal multiple (MLR)

- Función de coste: **RSS (Residual Sum of Squares)**


$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \varepsilon$$

→
$$\text{RSS} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2$$

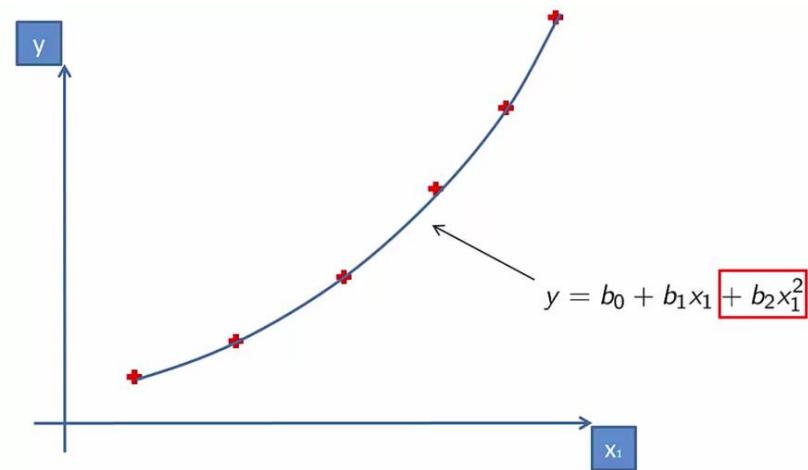
Tipos de algoritmos de Machine Learning

1.2. Regresión

Regresión polinómica

- Variación de la regression lineal simple con relación curvilinea entre el valor independiente (x) y el dependiente (y)

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \dots + \beta_M x^M + \epsilon.$$

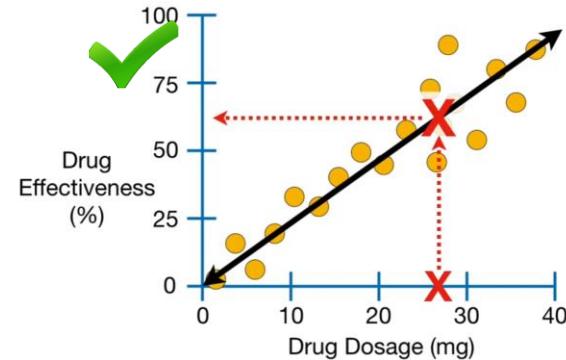


Tipos de algoritmos de Machine Learning

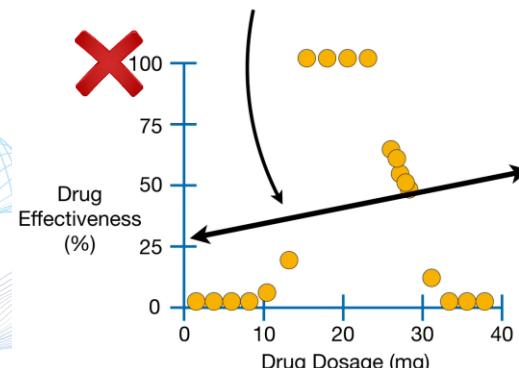
1.2. Regresión

Árboles de decisión (“Regression trees”)

- Igual que en clasificación, pero predicen variables continuas en lugar de discretas (categóricas).
- Cada nodo representa un valor numérico, no una categoría
- En cada “split” se busca el valor umbral que divide los datos de manera que la suma de los errores cuadráticos medios sea mínima (MSE)
- Función de coste: CART



In this case, fitting a straight line to the data will not be very useful.

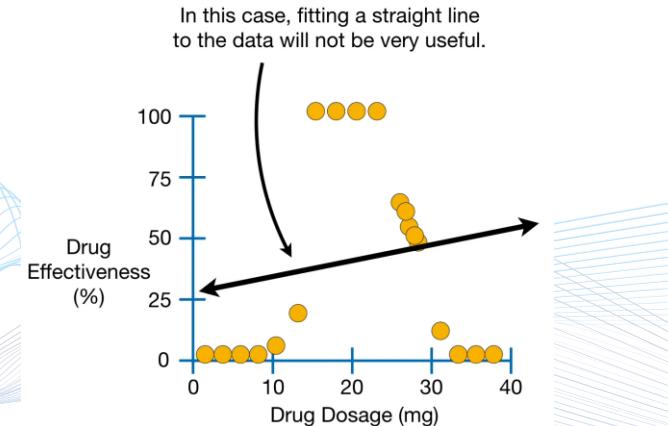
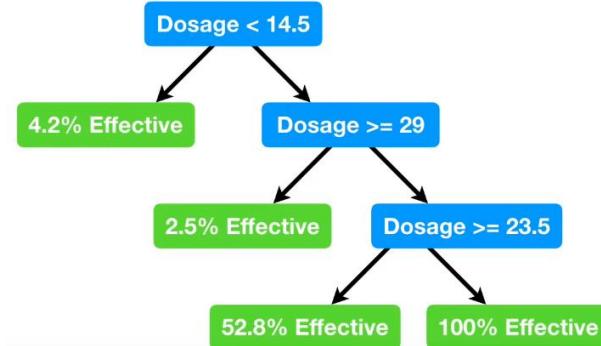


Tipos de algoritmos de Machine Learning

1.2. Regresión

Árboles de decisión (“Regression trees”)

- Igual que en clasificación, pero predicen variables continuas en lugar de discretas (categóricas).
- Cada nodo representa un valor numérico, no una categoría
- En cada “split” se busca el valor umbral que divide los datos de manera que la suma de los errores cuadráticos medios sea mínima (MSE)
- Función de coste: CART

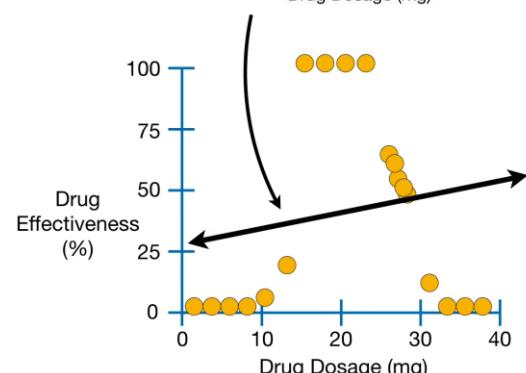
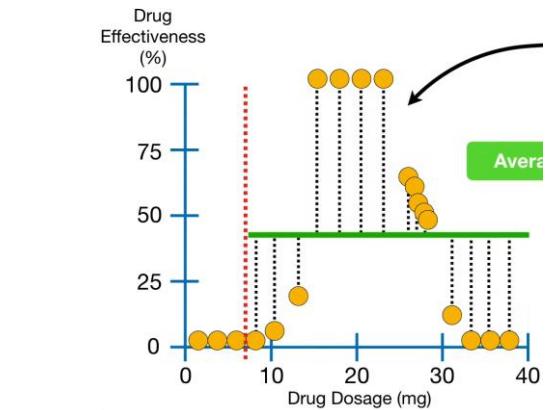
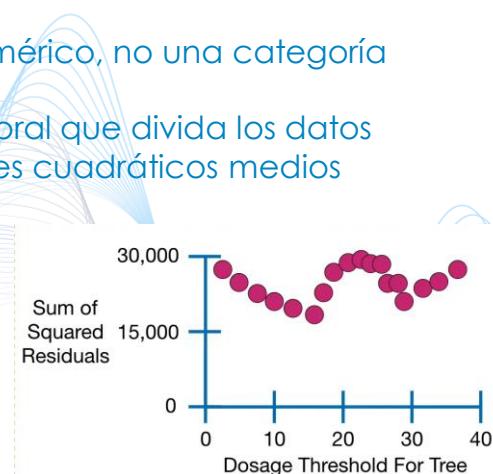


Tipos de algoritmos de Machine Learning

1.2. Regresión

Árboles de decisión (“Regression trees”)

- Igual que en clasificación, pero predicen variables continuas en lugar de discretas (categóricas).
- Cada nodo representa un valor numérico, no una categoría
- En cada “split” se busca el valor umbral que divide los datos de manera que la suma de los errores cuadráticos medios sea mínima (MSE)
- Función de coste: CART

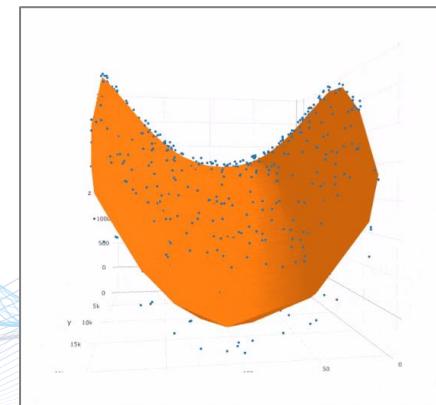
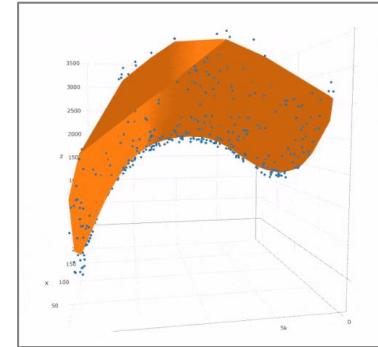
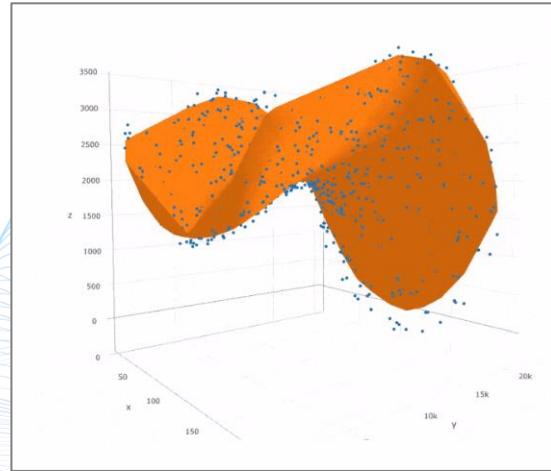


Tipos de algoritmos de Machine Learning

1.2. Regresión

Support Vector Machines (support vector regression)

- Predicen valores discretos
- El principio es el mismo que para clasificación
- Buscan el hiperplano que tenga el máximo número de puntos



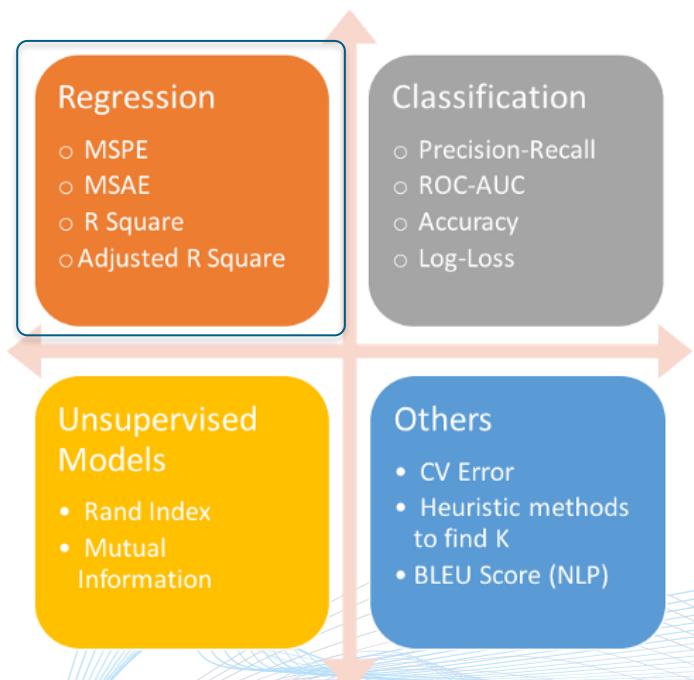
Tipos de algoritmos de Machine Learning

1.2. Regresión

Métricas de evaluación (Regresión)

- ¿Cómo calculamos la precision de mi modelo de regression? → **No se puede!!! La precision es una métrica para la clasificación**
- Queremos saber cuán cerca está la predicción de los valores esperados
- Podemos usar:
 - R^2
 - RMSE o MSE
 - MSPE
 - ...

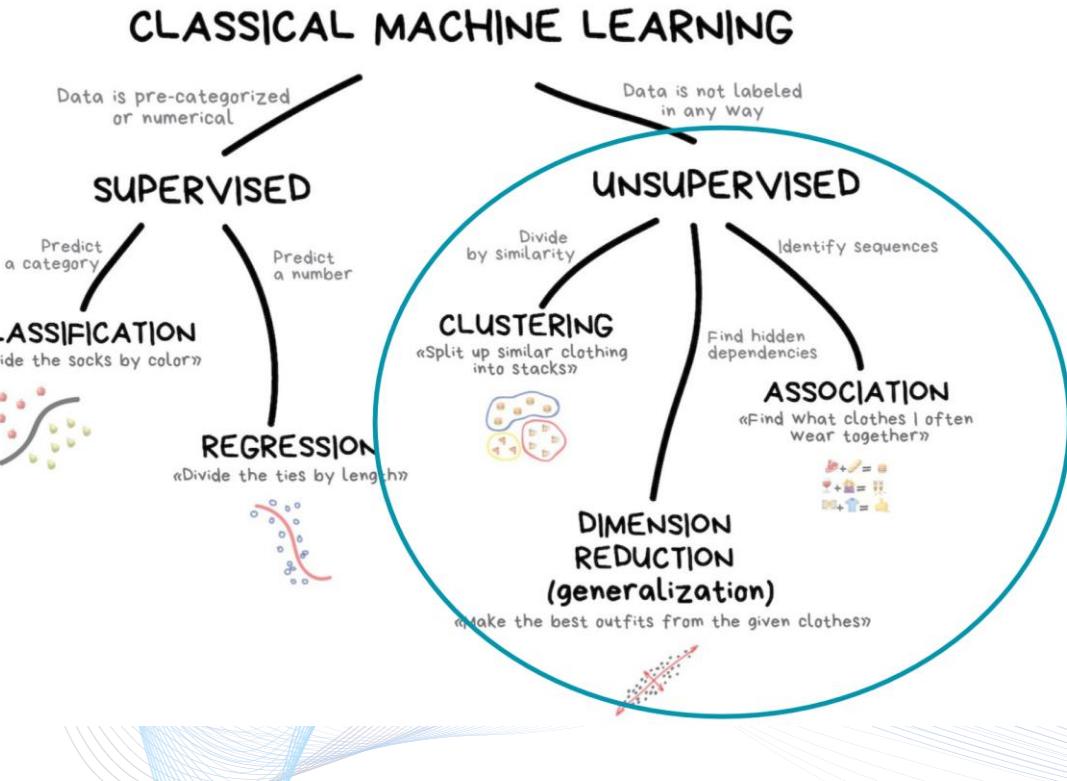
No confundir con la función de coste!



Tipos de algoritmos de Machine Learning

2. No supervisado

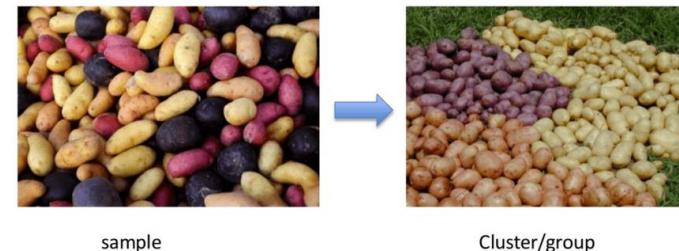
- **Datos NO etiquetados**
- Objetivo: encontrar la estructura subyacente del conjunto de datos, agrupar esos datos según sus similitudes y representar ese conjunto de datos en un formato comprimido.
- Tipos de aprendizaje no supervisado:
 - **Clustering (agrupamiento)**
 - **Reducción de dimensionalidad**
 - Detección de anomalías, estimación de densidad...



Tipos de algoritmos de Machine Learning

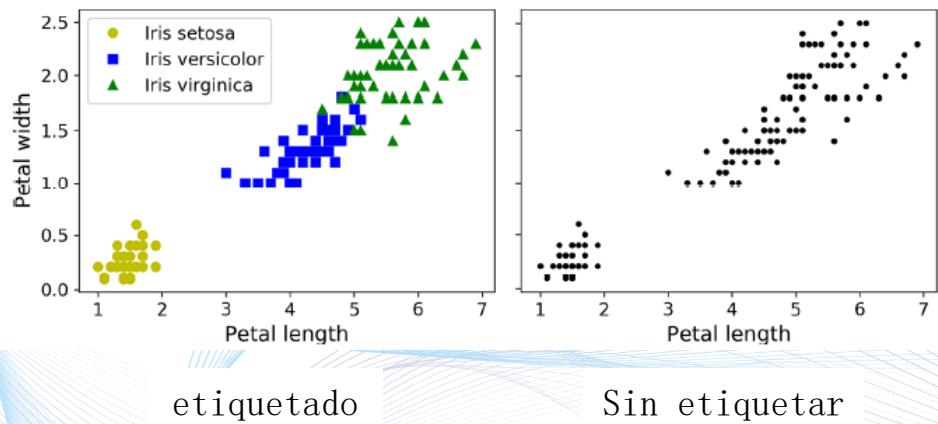
2.1. Clustering (agrupamiento)

- El objetivo es agrupar datos que poseen características similares en “clusters”
- La definición de este “cluster” es lo que determinará el tipo de algoritmo que usaremos:
 - **K-Means**
 - DBSCAN
 - Agrupamiento jerárquico
 - “Gaussian Mixtures”
 - ...
- Aplicaciones: análisis de datos (data mining), segmentación de clientes (customer segmentation), segmentación de imágenes, etc.



sample

Cluster/group



etiquetado

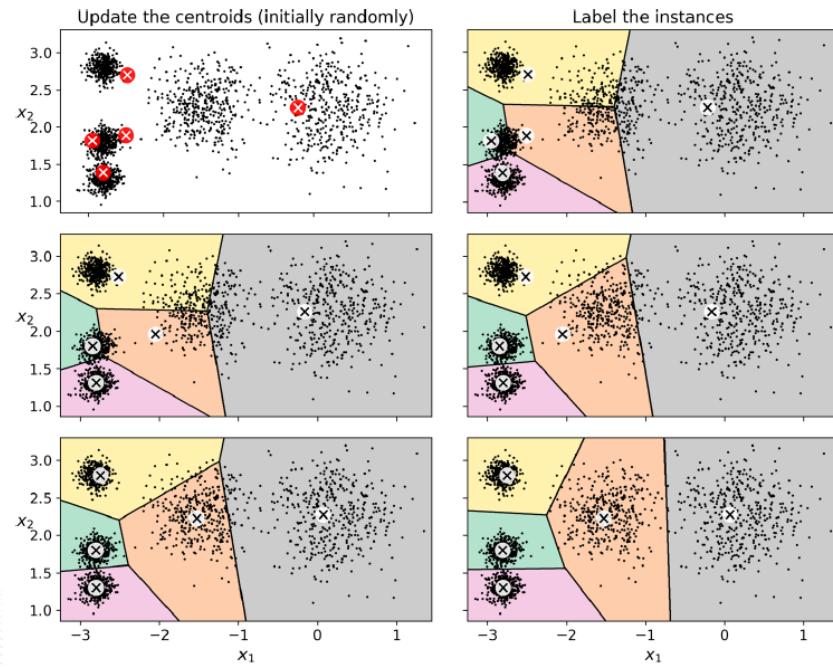
Sin etiquetar

Tipos de algoritmos de Machine Learning

2.1. Clustering (agrupamiento)

K-means

- Objetivo: agrupar objetos en **K clusters** basándose en sus características.
- Se realiza minimizando la suma de distancias entre cada objeto y el **centroide** de su cluster.
 1. Se indica el número de clusters (K)
 2. Se definen K centroides (puntos aleatorios)
 3. Se mide la distancia de cada punto a cada centroide
 4. Se asigna cada punto al grupo del centroide más cercano
 5. Se recalcula el nuevo centroide de cada grupo, y se mueve el anterior a ese punto.
 6. Se repite hasta que los centroides “no se mueven”.



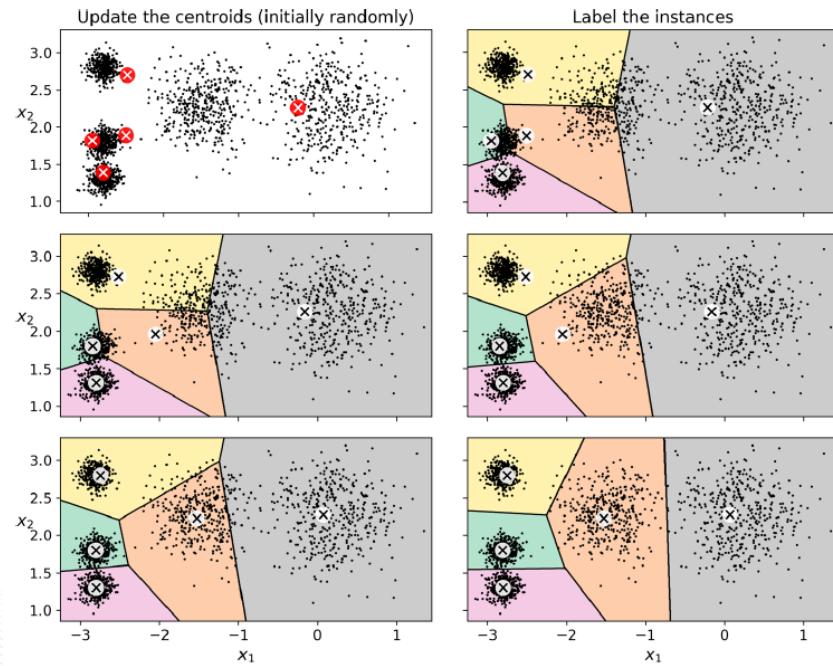
Tipos de algoritmos de Machine Learning

2.1. Clustering (agrupamiento)

K-means

- Objetivo: agrupar objetos en **K clusters** basándose en sus características.
- Se realiza minimizando la suma de distancias entre cada objeto y el **centroide** de su cluster.
- 7. Se calcula la **inercia** (suma de las varianzas de cada cluster). También llamada *within-cluster sum-of-squares (SSE)*
- 8. Este proceso se repite n veces con distintas posiciones de los centroides iniciales
- 9. La inicialización que minimice la inercia es el resultado final.

$$\sum_{i=0}^n \min_{\mu_K \in C} (\| x_i - \mu_K \|^2)$$



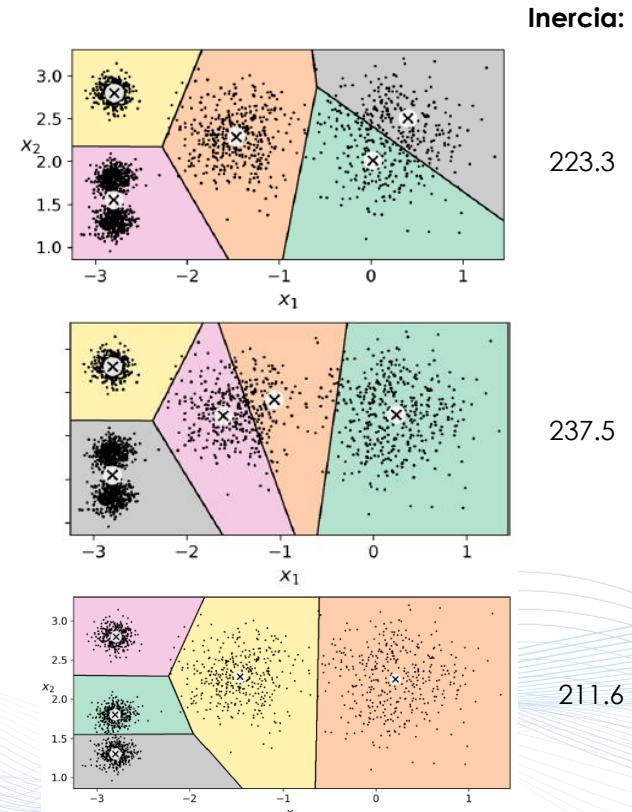
Tipos de algoritmos de Machine Learning

2.1. Clustering (agrupamiento)

K-means

- Objetivo: agrupar objetos en **K clusters** basándose en sus características.
 - Se realiza minimizando la suma de distancias entre cada objeto y el **centroide** de su cluster.
7. Se calcula la **inercia** (suma de las varianzas de cada cluster). También llamada *within-cluster sum-of-squares (SSE)*
 8. Este proceso se repite n veces con distintas posiciones de los centroides iniciales
 9. La inicialización que minimice la inercia es el resultado final.

$$\sum_{i=0}^n \min_{\mu_K \in C} (\| x_i - \mu_K \|^2)$$



Tipos de algoritmos de Machine Learning

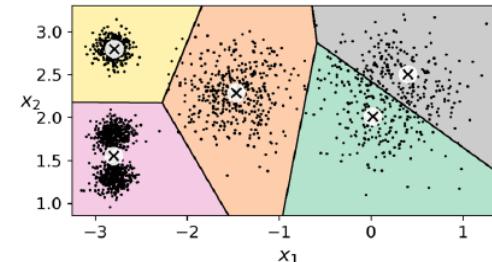
2.1. Clustering (agrupamiento)

K-means

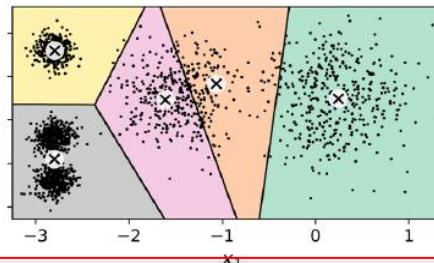
- Objetivo: agrupar objetos en **K clusters** basándose en sus características.
 - Se realiza minimizando la suma de distancias entre cada objeto y el **centroide** de su cluster.
7. Se calcula la **inercia** (suma de las varianzas de cada cluster). También llamada *within-cluster sum-of-squares (SSE)*
 8. Este proceso se repite n veces con distintas posiciones de los centroides iniciales
 9. La inicialización que minimice la inercia es el resultado final.

$$\sum_{i=0}^n \min_{\mu_K \in C} (\| x_i - \mu_K \|^2)$$

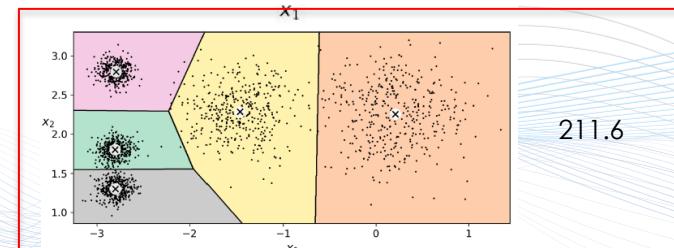
Inercia:



223.3



237.5



211.6

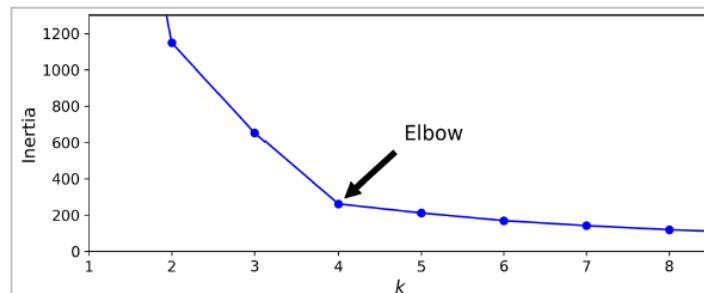
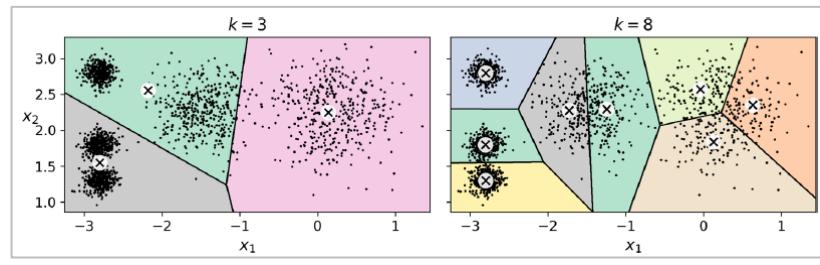
Tipos de algoritmos de Machine Learning

2.1. Clustering (agrupamiento)

K-means

Cómo se decide el número de K's?

- Repetimos el proceso muchas veces, desde $K = 1$ hasta $K = \dots 10$ (¿?) buscando la inercia más baja?
 - **No!** La inercia siempre disminuye a mayor número de centroides (K).
 - **Representamos inercia vs. K :** hay un punto en el que el cambio no es tan drástico, reflejado por un **codo** en esta función. **Este es el número ideal de K**
 - Esto es el **método del codo** (elbow method)



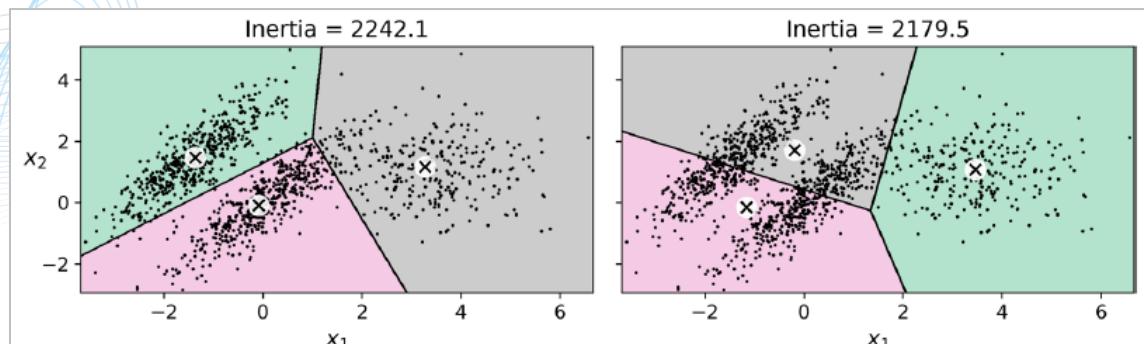
Tipos de algoritmos de Machine Learning

2.1. Clustering (agrupamiento)

K-means

Limitaciones de este algoritmo

- Hay que ejecutarlo muchas veces para asegurarse de que no converge en mínimos locales (suele pasar mucho)
- Hay que especificar de antemano el número K (superado por el algoritmo de **agrupamiento jerárquico**)
- No funciona bien con clústeres que no son esféricos (superado por el algoritmo **DBSCAN** o *Density-Based Spatial Clustering of Applications with Noise*)



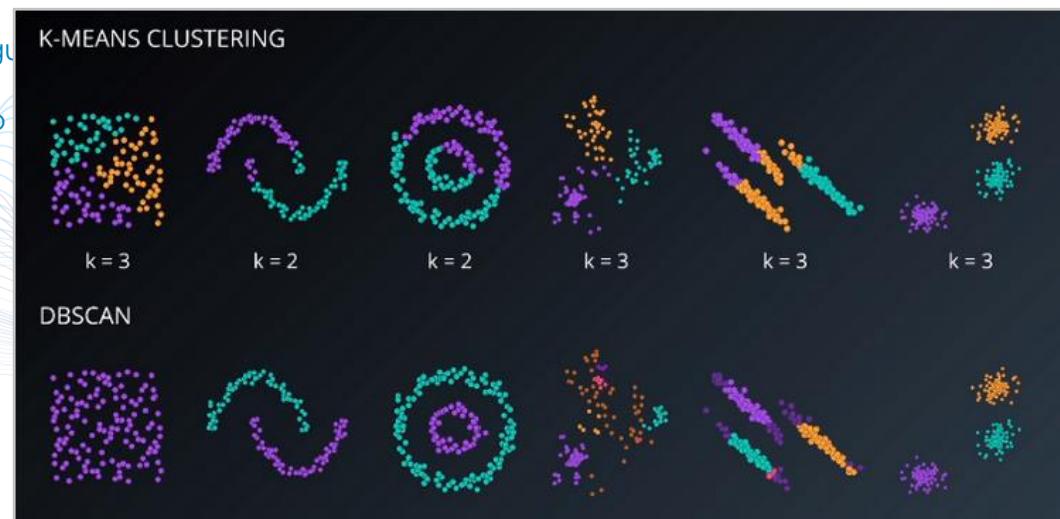
Tipos de algoritmos de Machine Learning

2.1. Clustering (agrupamiento)

K-means

Limitaciones de este algoritmo

- Hay que ejecutarlo muchas veces para asegurarse de que ha收敛ido.
- Hay que especificar de antemano el número de clústeres (k).
- No funciona bien con clústeres que no son esféricos (superado por el algoritmo **DBSCAN** o *Density-Based Spatial Clustering of Applications with Noise*)



Tipos de algoritmos de Machine Learning

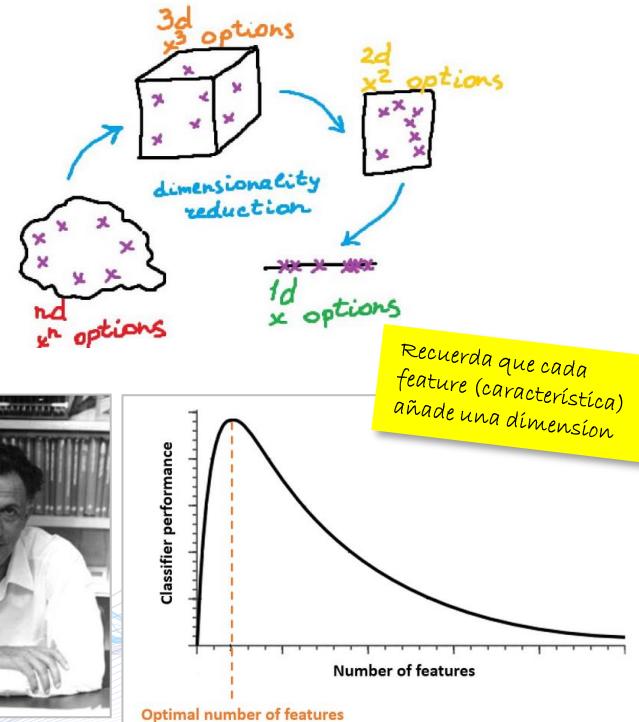
2.2. Reducción de dimensionalidad

- El objetivo de estos métodos es **reducir** el número de features de un dataset (ojo, características, no muestras!!)
- Queremos evitar la “maldición de la dimensionalidad” (*Curse of Dimensionality*)
- Tipos de algoritmos:
 - **PCA** (Principal Component Analysis)
 - Manifold Learning
 - LLE (Locally Linear Embedding)
 - Isomapas
 - t-SNE (*t*-distributed Stochastic Neighbor Embedding)
 - Autoencoders
 - Others...

1957



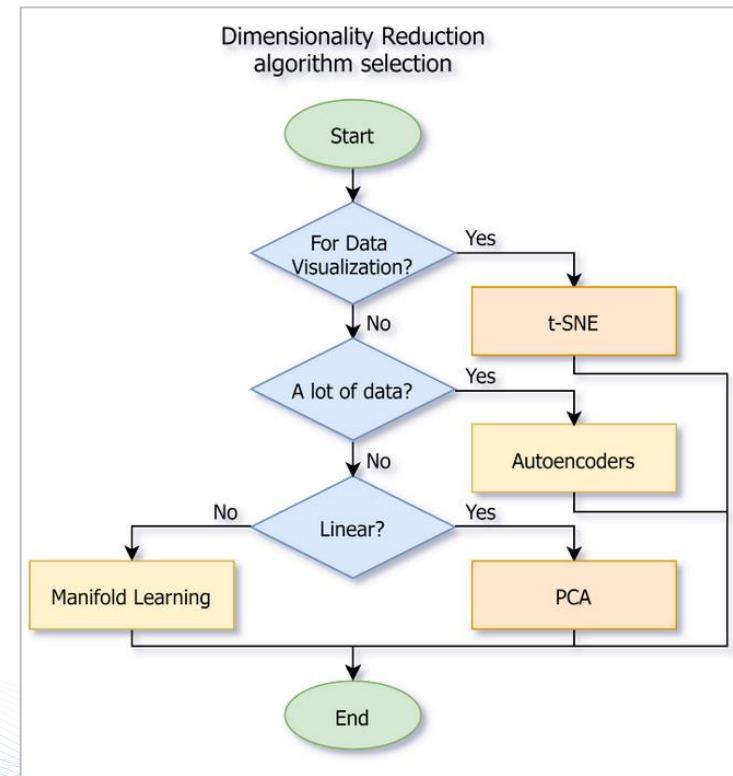
Richard E. Bellman



Tipos de algoritmos de Machine Learning

2.2. Reducción de dimensionalidad

- El objetivo de estos métodos es **reducir** el número de features de un dataset (ojo, características, no muestras!!)
- Queremos evitar la “maldición de la dimensionalidad” (**Curse of Dimensionality**)
- Tipos de algoritmos:
 - **PCA** (Principal Component Analysis)
 - Manifold Learning
 - LLE (Locally Linear Embedding)
 - Isomapas
 - t-SNE (*t*-distributed Stochastic Neighbor Embedding)
 - Autoencoders
 - Others...

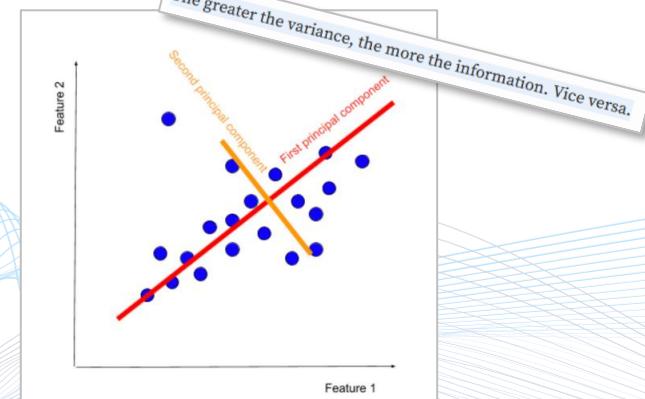
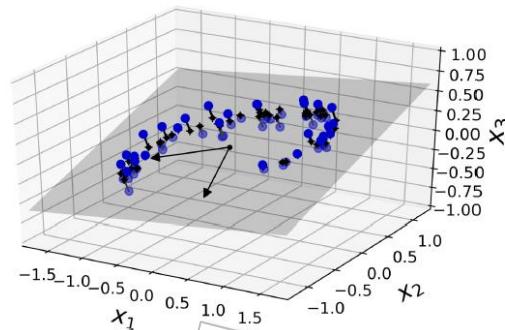


Tipos de algoritmos de Machine Learning

2.2. Reducción de dimensionalidad

PCA (Análisis de Componentes Principales)

- Es el método más conocido para reducción de dimensionalidad. Hay muchos algoritmos de PCA
- Se busca el hiperplano que mantenga la máxima **varianza** de los datos. Sobre ese plano, se proyectan los puntos
 - Se proyectan los puntos sobre el hiperplano que minimice la distancia a todos los puntos. Esta es la primera componente (PC1). **Explica la máxima varianza (máxima información).**
 - La segunda componente (PC2) será ortogonal a PC1. **Explicará la “segunda” máxima varianza**
 - Y así... (máximo n dimensiones donde n es el numero total de features. Lógico, ¿no?
 - Tendremos una combinación lineal de features, cada una “pesada” con su importancia



Tipos de algoritmos de Machine Learning

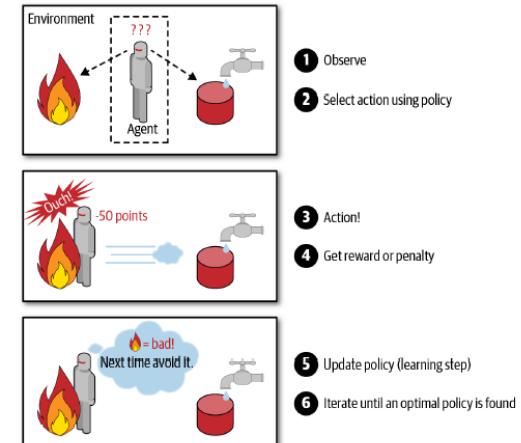
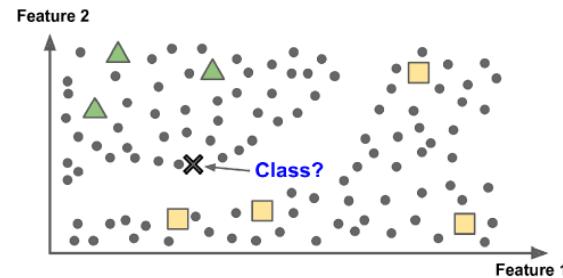
3. Otros tipos

Semi supervisado

- Trata con datos que están parcialmente etiquetados

Reinforcement learning

- Es un aprendizaje por refuerzo en el que el sistema que aprende (agente) observa el entorno, ejecuta acciones y obtiene recompensas o penalizaciones.



El DATO en Machine Learning

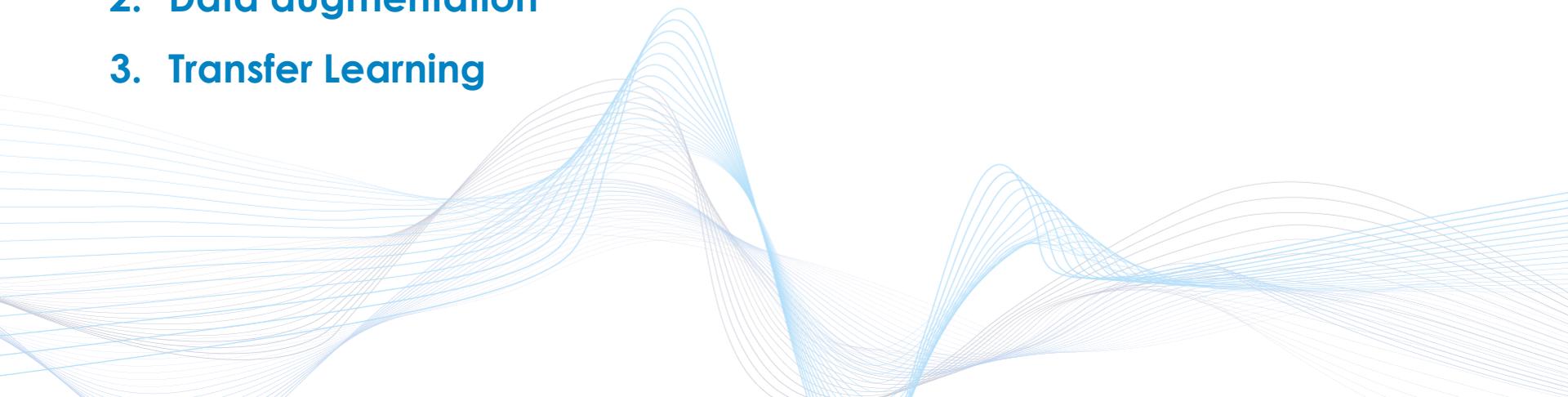
1. Preparando los datos

1.1. Exploratory Data Analysis (EDA)

1.2. Feature engineering

2. Data augmentation

3. Transfer Learning



El DATO en Machine Learning

1. Preparando los datos

1.1. Exploratory Data Analysis (EDA)

El Análisis de datos exploratorio (EDA) es el **primer proceso** que debemos hacer antes de construir ningún modelo.

Nos ayuda, entre otras cosas, a reducir posibles **sesgos** (*bias*)

Necesitamos preguntarnos:

- Cómo y de qué tipo son mis datos? (tabulares, ficheros, imágenes, palabras...)
- Cuál es la complejidad de mis datos?
- Necesito más datos?
- Faltan datos? (i.e. hay NaN's?)
- Hay datos duplicados?
- Cuántas variables y cuantas muestras tengo?
- ...



El DATO en Machine Learning

1. Preparando los datos

1.1. Exploratory Data Analysis (EDA)

El Análisis de datos exploratorio (EDA) es el **primer proceso** que debemos hacer antes de construir ningún modelo.

Nos ayuda, entre otras cosas, a reducir posibles **sesgos** (*bias*)

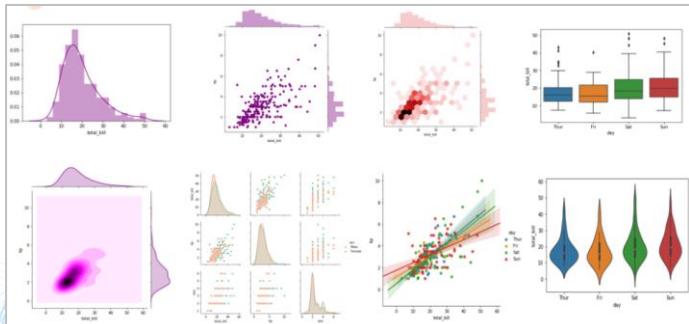
Una vez hayamos obtenido las respuestas a estas preguntas, procedemos a preprocessar los datos: **Data Preprocessing**

- Si las muestras de una variable deben ser datos numéricos y hay alguno que no lo sea, transformarlo
- Si hay datos nulos (NaN) eliminarlos
- Si hay features que ya se vean innecesarias, quitarlas
- Si hay datos duplicados, eliminarlos
- ...

In [5]: `frame = pd.read_csv('ratings.csv')
frame.describe()`

Out[5]:

| | placeID | rating | food_rating | service_rating |
|-------|---------------|-------------|-------------|----------------|
| count | 1161.000000 | 1161.000000 | 1161.000000 | 1161.000000 |
| mean | 134192.041344 | 1.199828 | 1.215332 | 1.090439 |
| std | 1100.916275 | 0.773282 | 0.792294 | 0.790844 |
| min | 132560.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 132856.000000 | 1.000000 | 1.000000 | 0.000000 |
| 50% | 135030.000000 | 1.000000 | 1.000000 | 1.000000 |
| 75% | 135059.000000 | 2.000000 | 2.000000 | 2.000000 |
| max | 135109.000000 | 2.000000 | 2.000000 | 2.000000 |



El DATO en Machine Learning

1. Preparando los datos

1.1. El análisis exploratorio de datos (EDA)

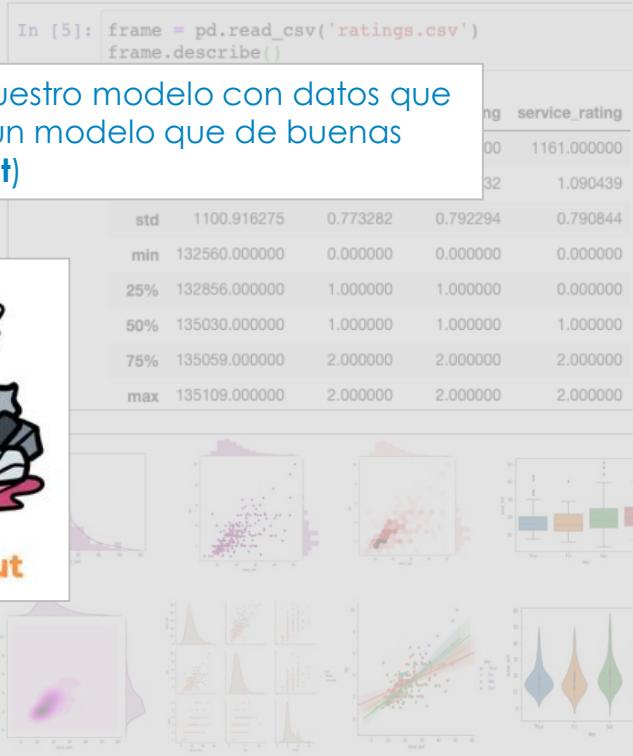
Este paso es muy importante, ya que si entrenamos nuestro modelo con datos que no son buenos (**garbage in**) no podemos esperar un modelo que de buenas predicciones (**garbage out**)

El Análisis de datos exploratorio (EDA) es el **primer proceso** que debemos hacer antes de construir ningún modelo.

Nos ayuda, entre otras cosas:

Una vez hayamos obtenido los datos procedemos a preprocesarlos:

- Si las muestras de una variable tienen errores, hay alguno que no lo sea
- Si hay datos nulos (NaN)
- Si hay features que ya se vean innecesarias, quitarlas
- Si hay datos duplicados, eliminarlos
- ...



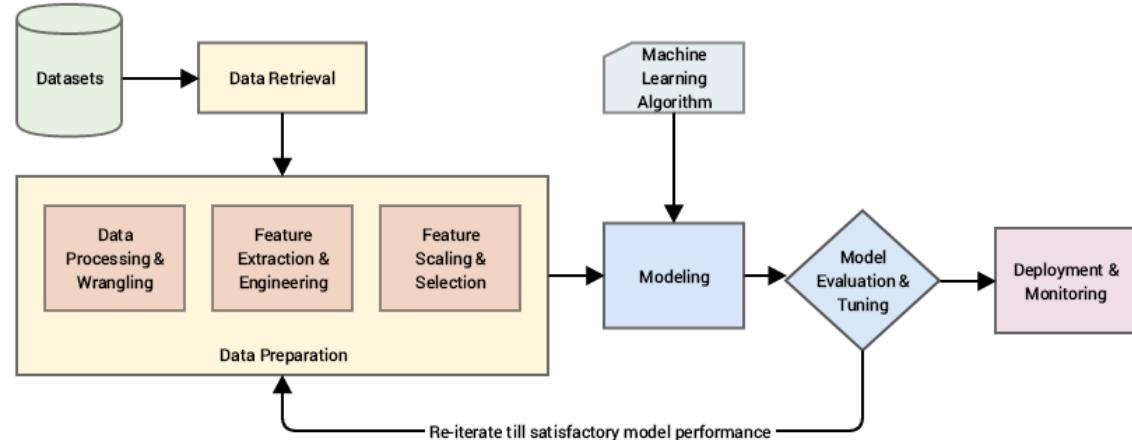
El DATO en Machine Learning

1. Preparando los datos

1.2. Feature Engineering (Ingeniería de datos)

El objetivo de este paso es crear un **set óptimo de features** de entrenamiento. Consiste en varios procesos:

- Feature creation (añadiendo o quitando features)
- Transformación de features
- Extracción de features relevantes (información útil)
- ...



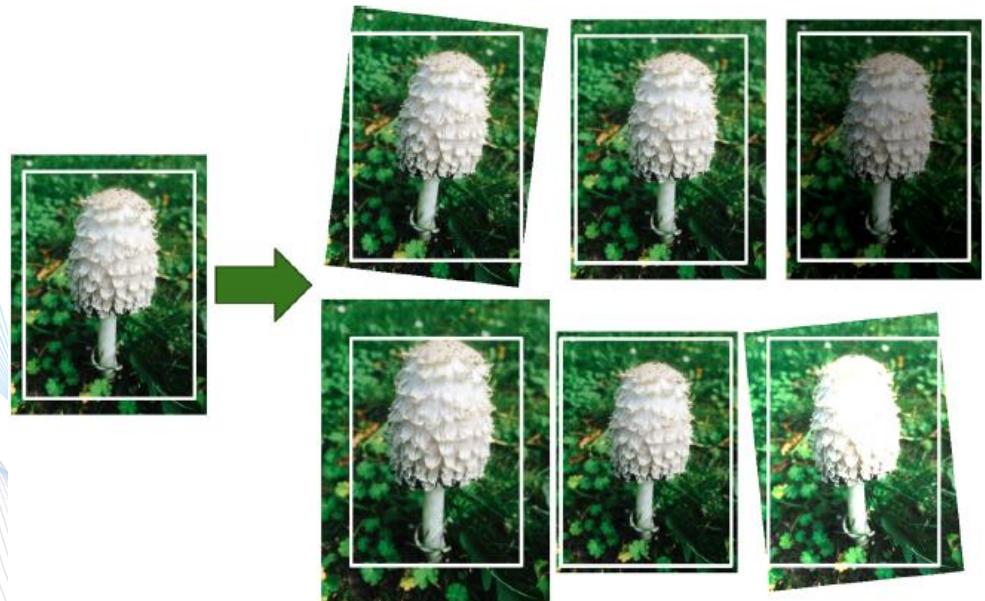
El DATO en Machine Learning

2. Mejorar el modelo a través del dato

Data augmentation

- Generación de nuevas instancias a partir de los datos que ya tenemos
- Ejemplos de variaciones en una imagen:

- Flip (volteo)
- Rotación
- Escalado / Zoom
- Recorte
- Translación
- Añadir ruido gaussiano
- ...



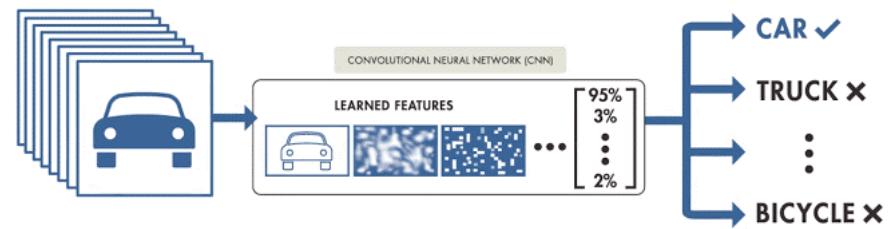
El DATO en Machine Learning

2. Mejorar el modelo a través del dato

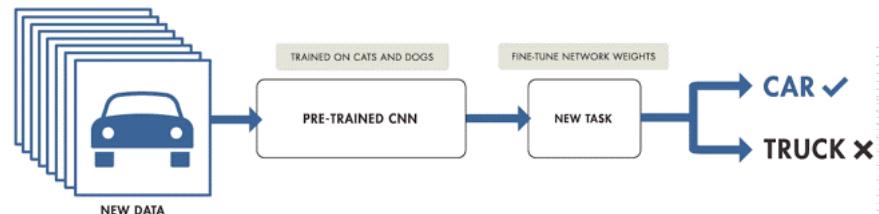
Transfer Learning

- Podemos evitar la necesidad de tanta cantidad de datos para entrenar un modelo...
- ...haciendo uso de modelos preentrenados
- Avance de redes neuronales: en una red neuronal solo tendremos que entrenar las últimas capas de nuestro modelo, que serán las dedicadas a nuestro problema.

TRAINING FROM SCRATCH



TRANSFER LEARNING



To take home...

- A diferencia de la programación tradicional (datos → reglas → modelo → resultado), en ML se escribe un **programa capaz de aprender a través de ejemplos para obtener unas reglas** (datos → modelo → resultado → reglas)
- Un modelo es una función matemática definida por unos parámetros. La medida del error entre los valores de salida (output = predicción) y los datos reales se mide con la **función de coste**. Entrenar un modelo significa buscar los parámetros de la función que **minimizan la función de coste**.
- Queremos que nuestro modelo se ajuste bien a los datos, pero ni demasiado (**overfitting**) ni muy poco (**underfitting**). Existen técnicas para evitar estos escenarios (**regularización**)
- **Pipeline:**
 1. Visualización y limpieza de datos (garbage in → garbage out)
 2. Feature Engineering
 3. Selección del modelo (clasificación, regresión, no supervisado, etc.)
 4. División de los datos en sets: train,test. Set de validación con cross-validation
 5. Train!
 6. Evaluación del modelo (curvas de aprendizaje)
 7. Observamos y evaluamos nuestras predicciones con el test set
 8. 1,200 × 818

Me: *uses machine learning*

Machine: *learns*

Me:

