



**FLACSO**  
ARGENTINA

Facultad  
Latinoamericana de  
Ciencias Sociales.  
Sede Argentina.

Área Comunicación  
y Cultura.

# CURSO DE POSGRADO **BIG DATA E INTELIGENCIA TERRITORIAL**

## Clase 1. Introducción al Aprendizaje Automático

# Quién soy

- Lic. en Química (2007-2012)
- Doctorado en Recursos Naturales (2013-2018) - CONICET
- PostDoc CONICET *Laboratorio de Estructura Molecular y Propiedades.* (2019-2022)
- Diplomatura en *ciencia de datos, aprendizaje automático y sus aplicaciones.* FaMAF - UNC (2021)
- *Data Scientist* - Navent i+D (2022-actualidad)



Twitter: [https://twitter.com/data\\_datum](https://twitter.com/data_datum)

LinkedIn: <https://www.linkedin.com/in/roxana-noelia-v/>

# Introducción Generalidades de ML

# *Machine learning en la vida cotidiana*



# *Machine Learning - definición*

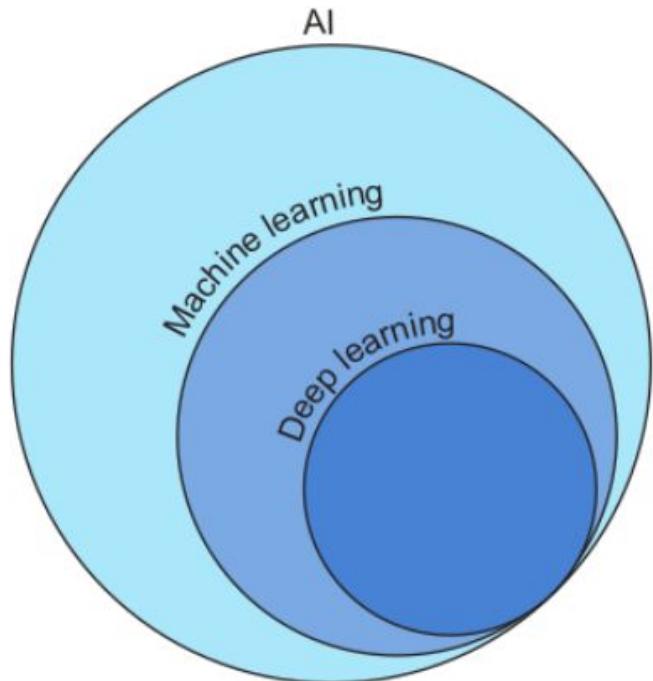
El aprendizaje automático se trata de **extraer conocimiento de los datos** (*Andreas Müller - Introduction to Machine Learning with Python*)

El aprendizaje automático se ocupa de la **construcción de algoritmos que, para ser útiles, se basan en una colección de ejemplos de algún fenómeno**. Los ejemplos pueden provenir de la naturaleza, ser recolectados por humanos o generados por otro algoritmo.

El aprendizaje automático también se puede definir como el proceso de resolución de un **problema práctico** mediante, 1) recopilar un conjunto de datos y 2) entrenar algorítmicamente un modelo estadístico basado en ese conjunto de datos (*Andriy Burkov - Machine Learning Engineering*)

# *Machine Learning - Definición*

- Subcampo de la inteligencia artificial.
- Machine Learning vs Deep Learning
- Ciencia de datos



# *Aprendizaje automático vs aprendizaje profundo*

- Costo computacional.
- Dificultad en el entrenamiento.
- Ingeniería de features vs Ingeniería de arquitecturas.

# *Aprendizaje automático vs ciencia de datos*

- En ML se hace énfasis en el modelado de los datos (algoritmo, error, división de los datos), en la ciencia de datos, hacemos foco en todo el proceso desde la delimitación del problema (problem framing) hasta su resolución y despliegue\*.

# *Cuando es apropiado ML para abordar un problema*

- Cuando el problema es complejo para ser programado (exceso de reglas para programar).
- Cuando el problema está cambiando constantemente.
- Cuando es un problema relacionado a la percepción (visión, lenguaje).
- Cuando es un fenómeno no estudiado.
- Cuando el problema tiene un objetivo simple.
- Cuando es eficiente con respecto al costo.\*

# *Conceptos Fundamentales*

- **Muestra, punto, observación, instancia** se refiere a una unidad de análisis.
- **Set de entrenamiento** son los datos utilizados para el modelado.
- **Set de prueba** son los datos utilizados para medir el desempeño del modelo, entre un conjunto de candidatos.
- **Atributos, predictores, features, variables independientes o descriptores** son los datos de entrada para la ecuación de predicción.
- **Salida, variable dependiente, variable respuesta, clase, o "target"** es la cantidad/clase a ser predicha.

# Conceptos Fundamentales

observación o muestra								
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

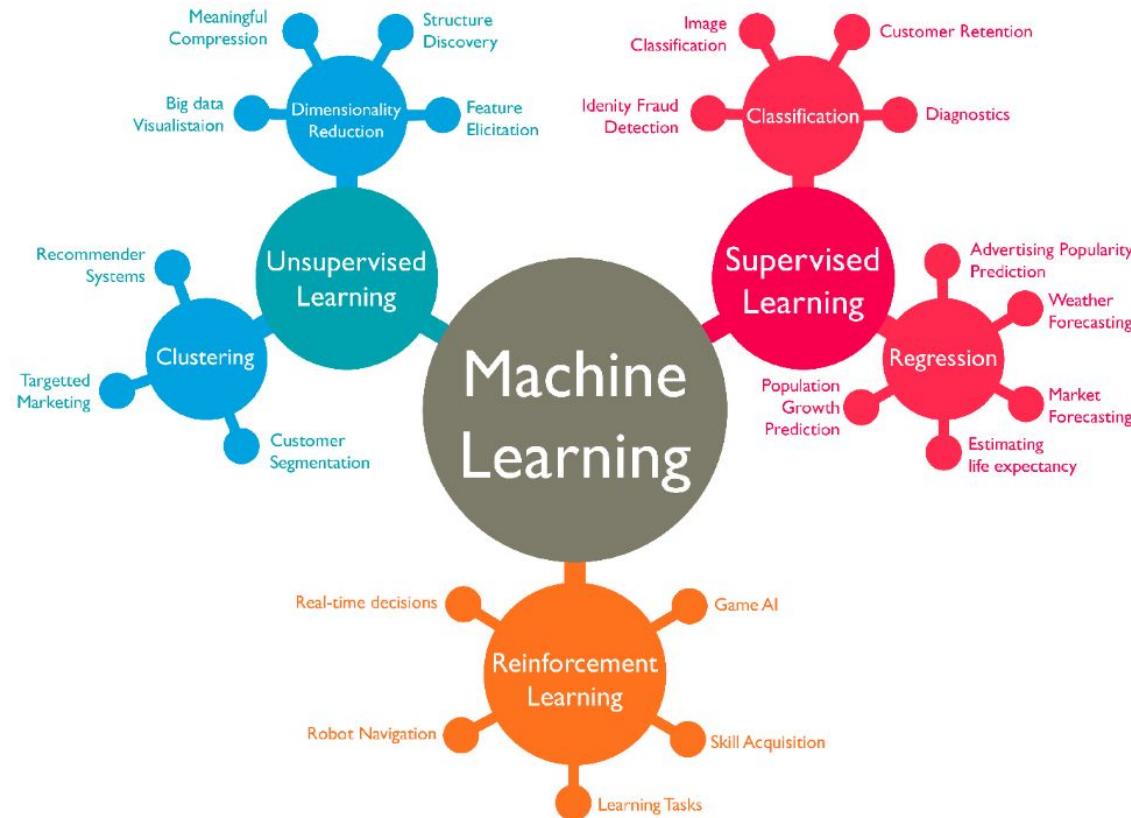
features / variables: columnas

target:  
variable a  
predecir

# Tipos de aprendizaje

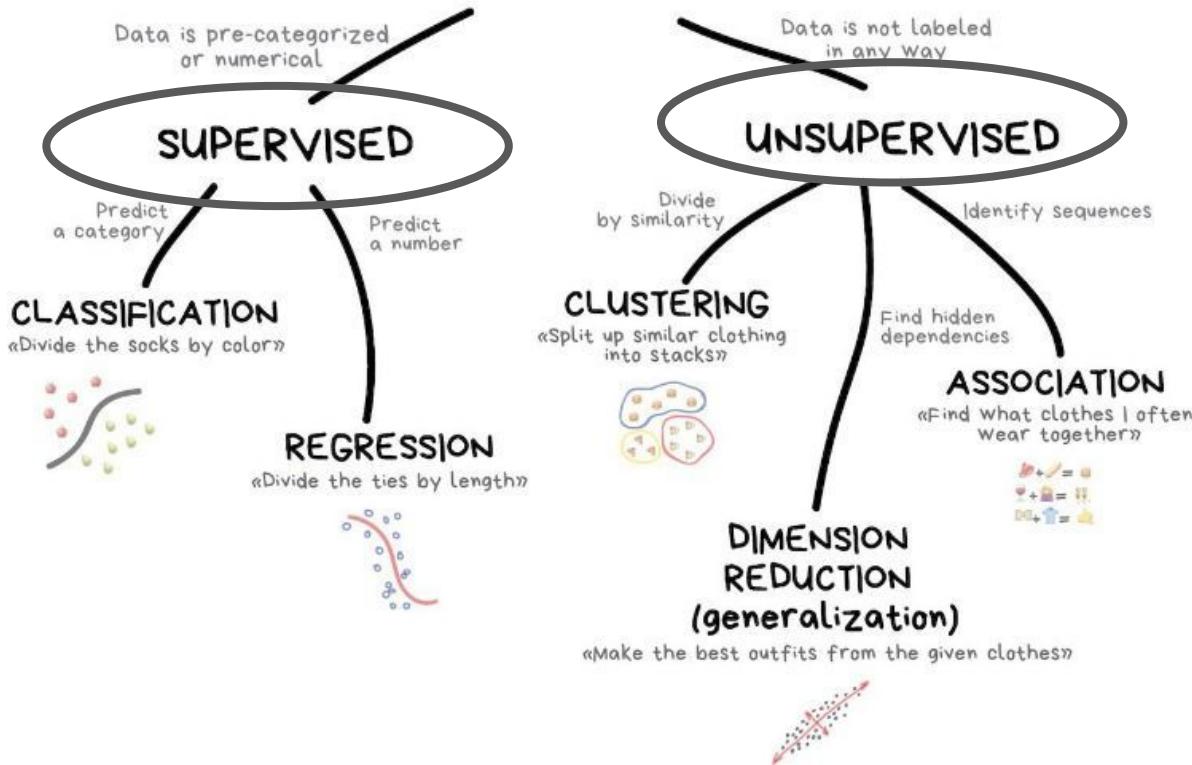
Análisis de  
redes sociales

Embeddings



# Tipos de aprendizaje

## CLASSICAL MACHINE LEARNING

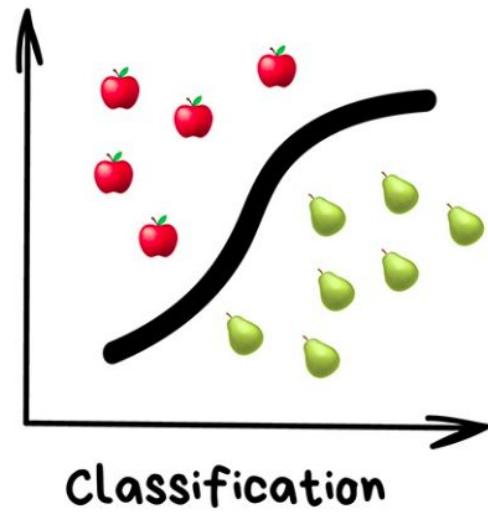


# Clasificación (ejemplos típicos)

Usado para:

- Filtros de spam (spam vs jam)
- Detección de lenguaje (español, inglés, etc)
- Análisis de sentimientos (positivo / negativo)
- Reconocer dígitos escritos a mano (de 0 a 9).

Variable a predecir: categoría (o clase)

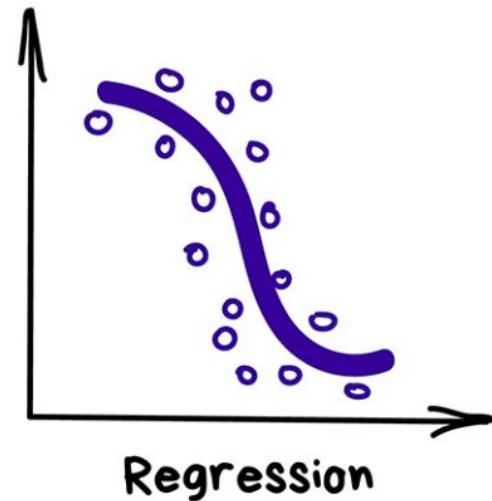


# *Regresión (ejemplos típicos)*

Usado para:

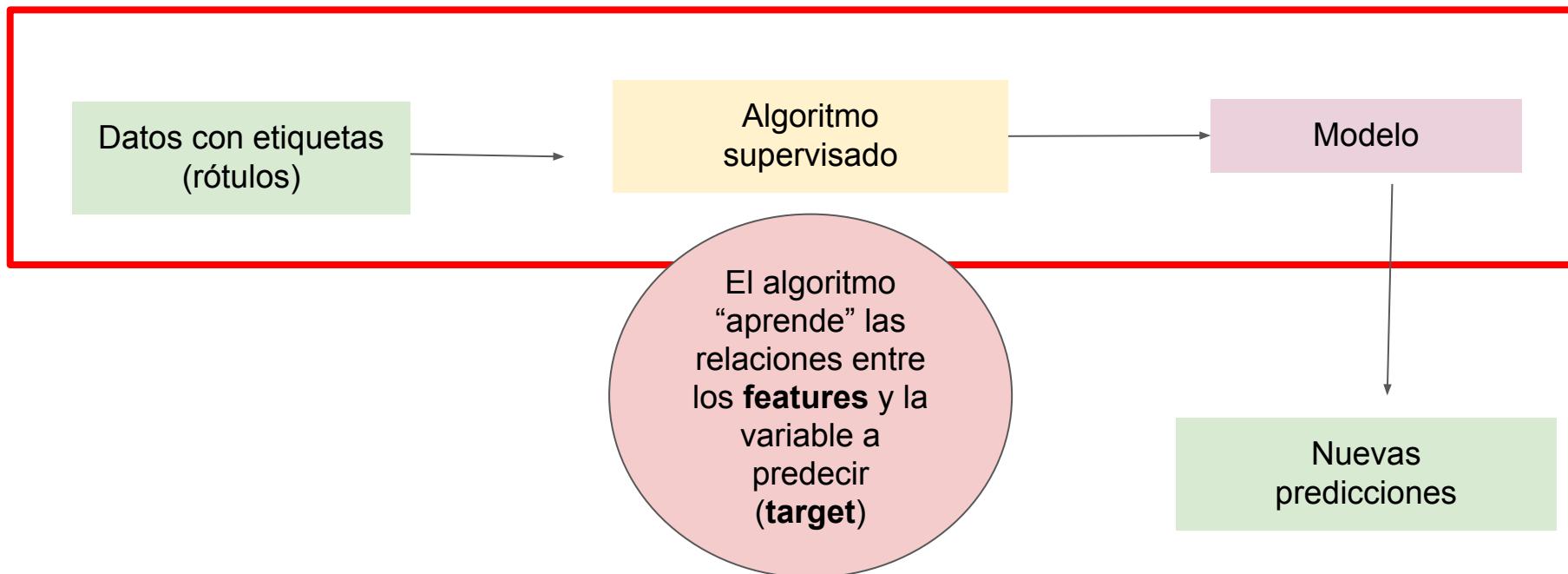
- Predicción de precios de casas según variables geográficas, metros cuadrados, etc.
- Establecer relaciones entre ingreso y variables demográficas.
- Predicción de accidentes automovilísticos.

Variable a predecir: numérica

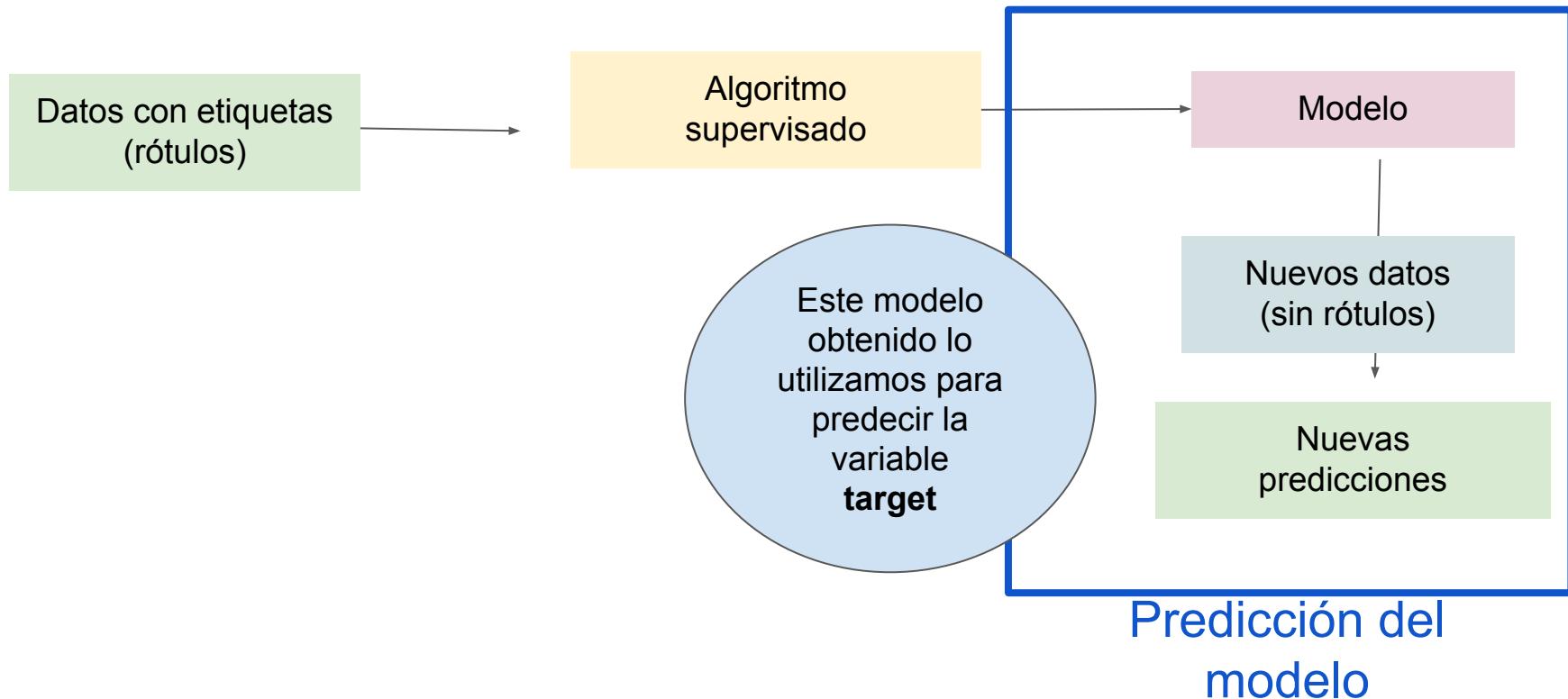


# Aprendizaje Supervisado

## Entrenamiento del modelo



# Aprendizaje Supervisado

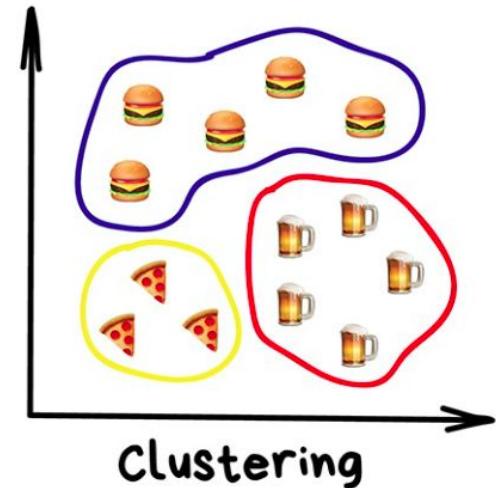
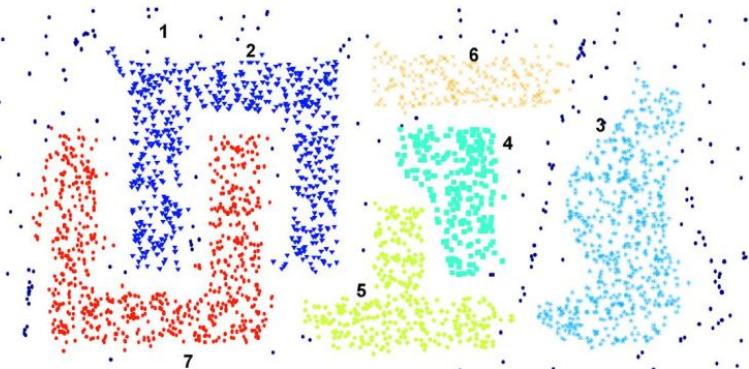


# *Clustering -clusterizado- (ejemplos típicos)*

Usado para:

- Segmentación de mercado (identificar tipo de consumidores, consumidores más o menos leales, etc).
- Agrupamiento de productos.

Nuestro objetivo va a ser encontrar posibles agrupamientos en los datos.

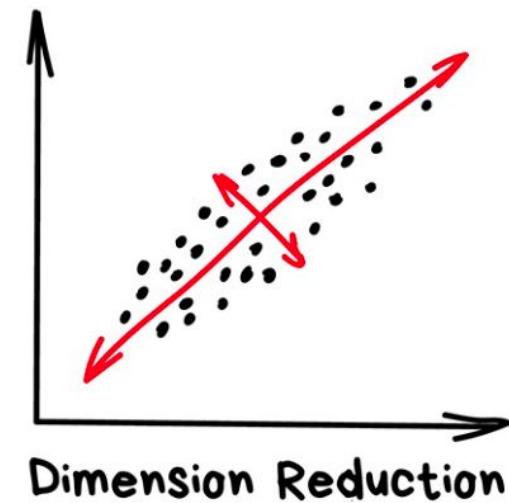


# *Reducción de dimensiones (ejemplos típicos)*

Usado para:

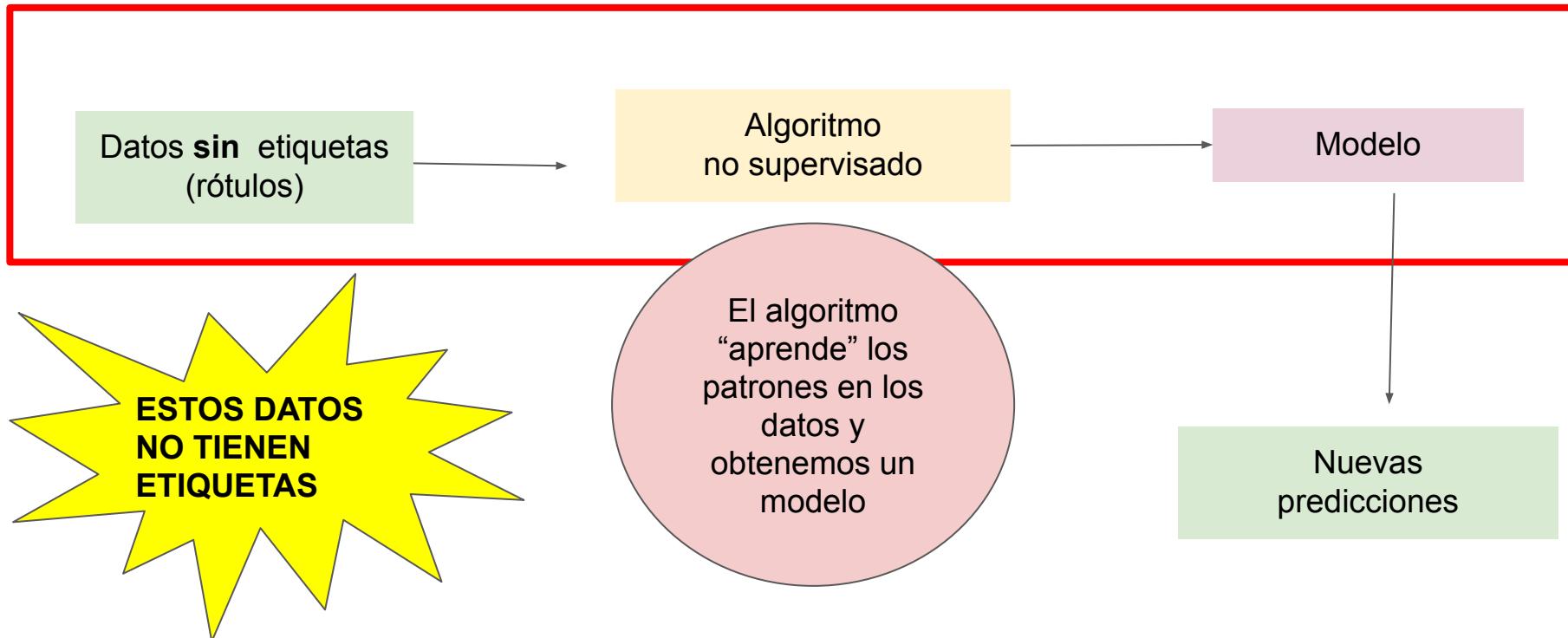
- Modelado de tópicos (análisis de texto).
- Visualización del problema que nos interesa, posibles relaciones entre variables.

Cuando hacemos reducción de dimensiones, en vez, de utilizar las variables que nos proporcionan, vamos a crear variables nuevas que conserven la mayor variabilidad posible.

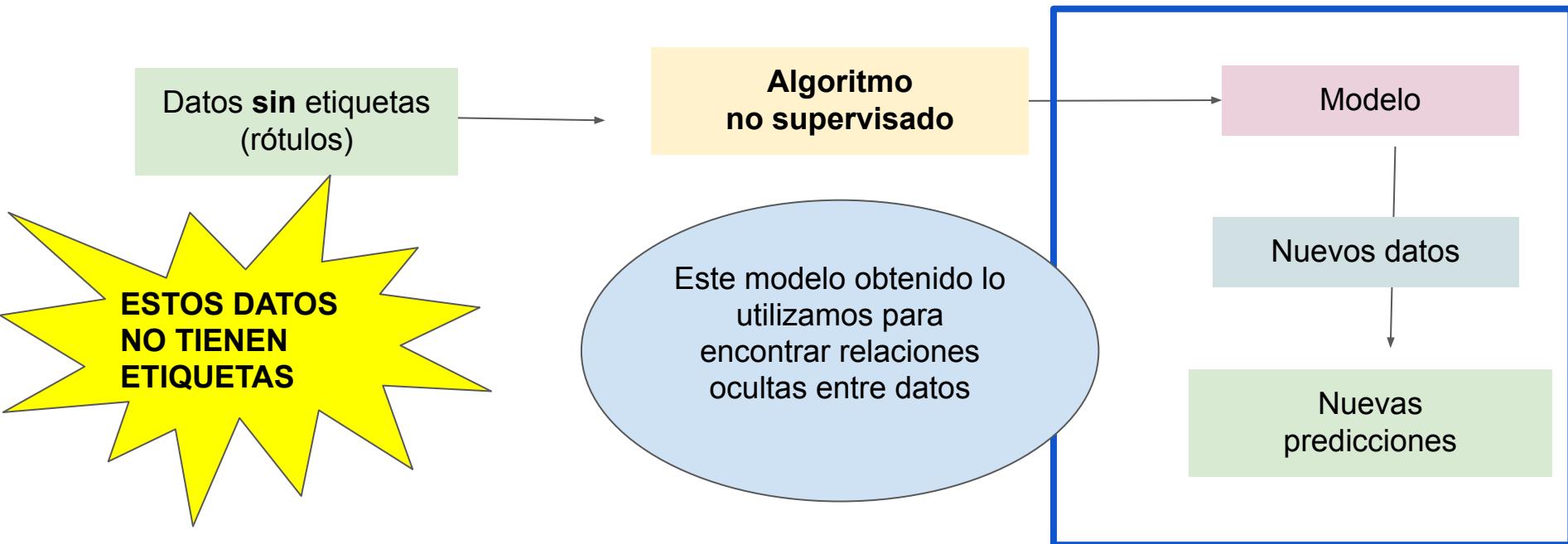


# Aprendizaje no supervisado

## Entrenamiento del modelo



# Aprendizaje no supervisado



# *Parámetros vs hiperparámetros*

- **Parámetros** son variables que definen el modelo entrenado mediante aprendizaje. Los parámetros son directamente modificados por el algoritmo de aprendizaje basado en los datos de entrenamiento. Ej, la pendiente y la ordenada de una recta en regresión lineal simple.
- **Hiperparámetros** son *variables internas* de un algoritmo de ML que influencian la performance del modelo. No pueden ser aprendidos mediante el algoritmo de ML, ya que son definidos por el usuario, aunque lo más común es realizar una búsqueda de hiperparámetros (en grilla o de manera aleatoria son las formas de búsqueda más comunes). Ej: profundidad de un árbol en árboles de decisión, el K óptima en KNN.

# Modelado Estadístico. Las dos culturas

*Statistical Science*  
2001, Vol. 16, No. 3, 199–231

## Statistical Modeling: The Two Cultures

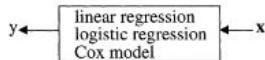
Leo Breiman

*Abstract.* There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems. Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools.

### 1. INTRODUCTION

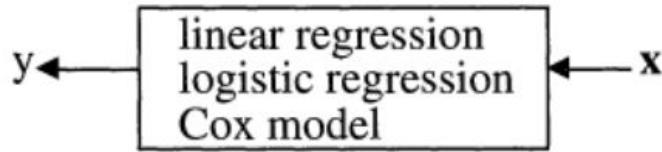
Statistics starts with data. Think of the data as being generated by a black box in which a vector of input variables  $\mathbf{x}$  (independent variables) go in one side, and on the other side the response variables  $\mathbf{y}$  come out. Inside the black box, nature functions to associate the predictor variables with the response variables, so the picture is like this:

The values of the parameters are estimated from the data and the model then used for information and/or prediction. Thus the black box is filled in like this:



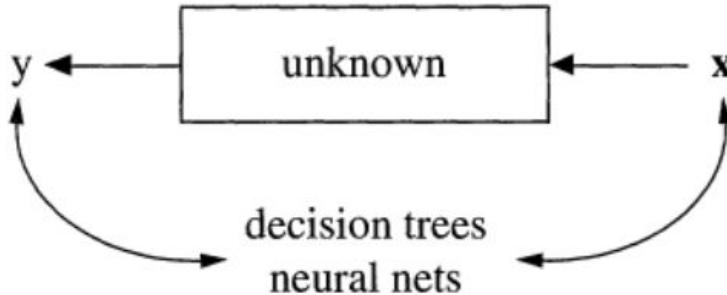
*Model validation.* Yes–no using goodness-of-fit

# Modelado Estadístico



- Énfasis en  $f(x)$ . El modelo se postula en base a supuestos sobre  $f(x)$ .
- Conocimiento acumulado, teoría, diseño de experimentos.
- Los parámetros son estimados con los datos y luego se realizan las predicciones.
- Evaluación del modelo: estimadores insesgados, robustos, mínima varianza.

# *Modelado Algorítmico*



- Énfasis en la predicción.
- El enfoque es encontrar una  $f(x)$  - un algoritmo - que opera sobre las  $x$  (las muestras) para producir las  $y$  (la predicción).
- El modelo se “aprende” de los datos.
- Evaluación del modelo: performance predictiva

# *Etapas de un problema de ML (supervisado)*

- **Definir el problema** ¿Qué se pretende predecir? ¿De qué datos se dispone? o ¿Qué datos es necesario conseguir?
- **Explorar y entender los datos** que se van a emplear para crear el modelo.
- **Preprocesar los datos** aplicar las transformaciones necesarias para que los datos puedan ser interpretados por el algoritmo de machine learning seleccionado.
- **Métrica de éxito** definir una forma apropiada de cuantificar cómo de buenos son los resultados obtenidos.
- **Preparar la estrategia para evaluar el modelo** separar las observaciones en un conjunto de entrenamiento, un conjunto de validación (o validación cruzada) y un conjunto de test. Es muy importante asegurar que ninguna información del conjunto de test participa en el proceso de entrenamiento del modelo.

# *Etapas de un problema de ML (supervisado)*

- **Ajustar un primer modelo** capaz de superar unos resultados mínimos. Por ejemplo, en problemas de clasificación, el mínimo a superar es el porcentaje de la clase mayoritaria (la moda).
- Gradualmente, **mejorar el modelo** incorporando-creando nuevas variables u optimizando los hiperparámetros.
- **Evaluar la capacidad del modelo final** con el conjunto de test para tener una estimación de la capacidad que tiene el modelo cuando predice nuevas observaciones.

# *Análisis Exploratorio de Datos*

## **EDA o Análisis Exploratorio de Datos**

**es un ciclo iterativo y un proceso creativo en donde,**

- Generas preguntas acerca de tus datos.**
- Buscas respuestas mediante la visualización y transformación de los mismos.**
- En base a lo aprendido, refinas tus preguntas e incluso, generas nuevas.**

# *Es importante en esta etapa (EDA)*

**Estudio de la distribución de las variables**

**Presencia de valores perdidos**

**Desbalance de las clases o grupos en estudio**

**Presencia de datos extremos o outliers**

**Covariación/correlación de variables**

# *¿Por qué es importante EDA?*

Porque te permite conocer y entender tus datos.

# *Curación de datos*

## **Curación de Datos o Ingeniería de Features**

es una etapa en la cual **seleccionamos y transformamos las variables**

- Puede haber combinación de datasets (enriquecimiento)
- Vemos que variables son relevantes (feature selection)
- Limpieza de datos (inconsistencias)

# *Es importante en esta etapa (curación de datos)*

**Imputación de datos ruidosos, perdidos o erróneos.**

**Codificación de variables categóricas.**

**Transformación de variables.**

**Ingeniería de outliers.**

**Escalado de Features**

**Discretización de variables (continuas ---> discretas)**

# *¿Por qué es importante la curación de datos?*

Porque permite transformar los datos para que tengan una forma adecuada para modelarlos.

Tener en cuenta que en esta etapa siempre se introduce SESGO.

# *Modelar con aprendizaje automático*

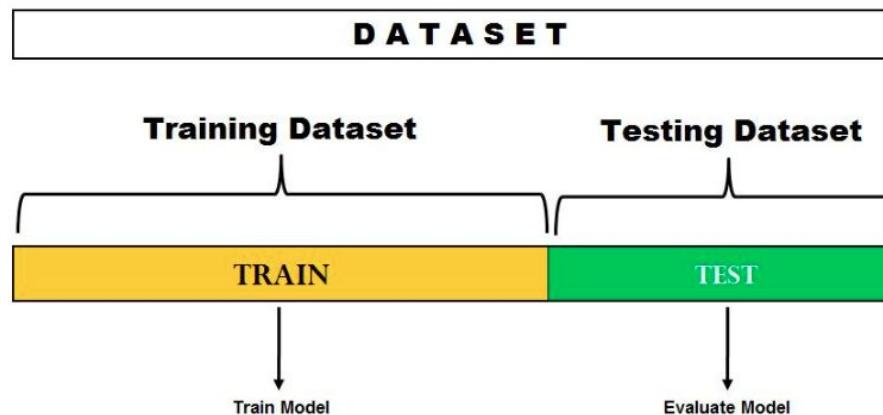
Podemos decir que modelar con Machine learning implica estos aspectos:

- Train / test split (¿cómo vamos a dividir los datos?)
- Complejidad / Interpretabilidad del modelo
- Error de entrenamiento / error de generalización
- Métricas
- Underfitting vs overfitting

# *Train - test split*

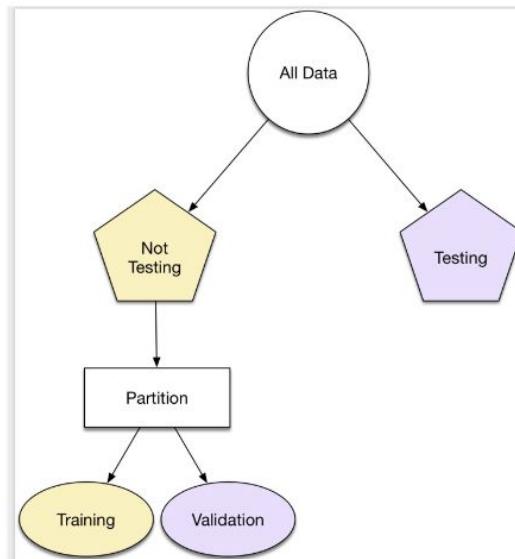
## Opción 1: HOLDOUT

La manera más simple de dividir un dataset consiste en dividir en datos de train (entrenamiento) y test (testeo)



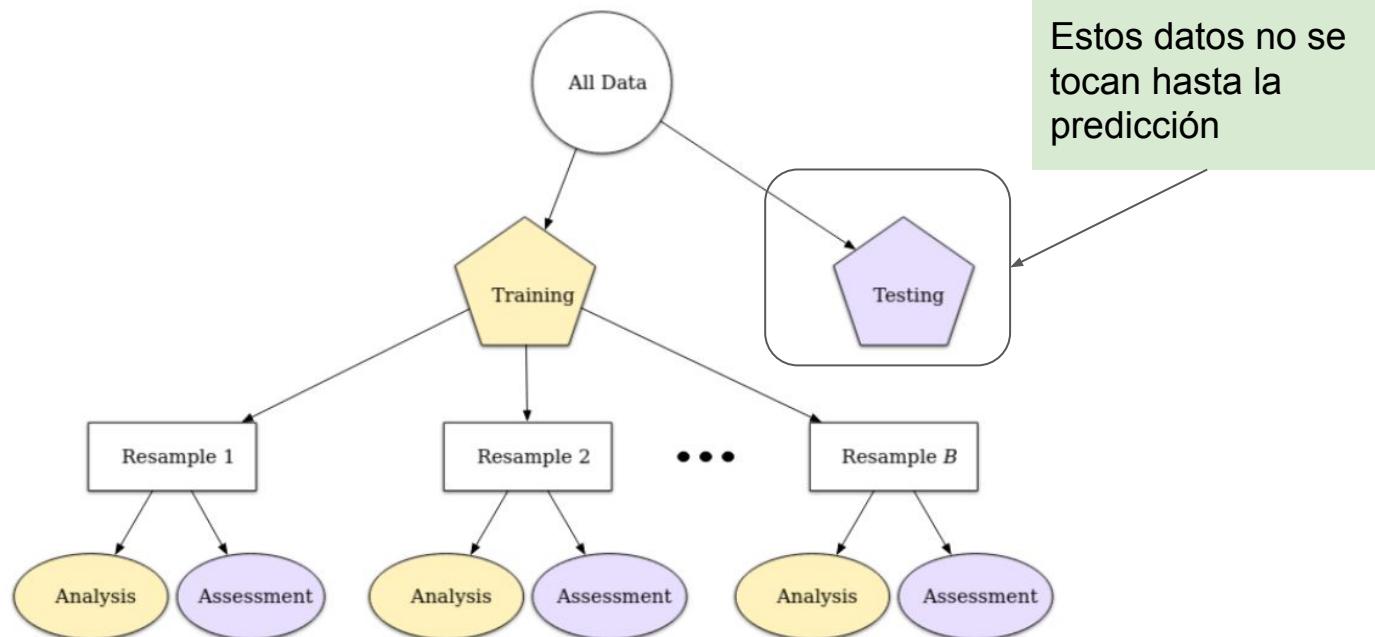
# *Entrenamiento - validación - testeo*

Opción 2: Dividir en 3 partes el dataset:

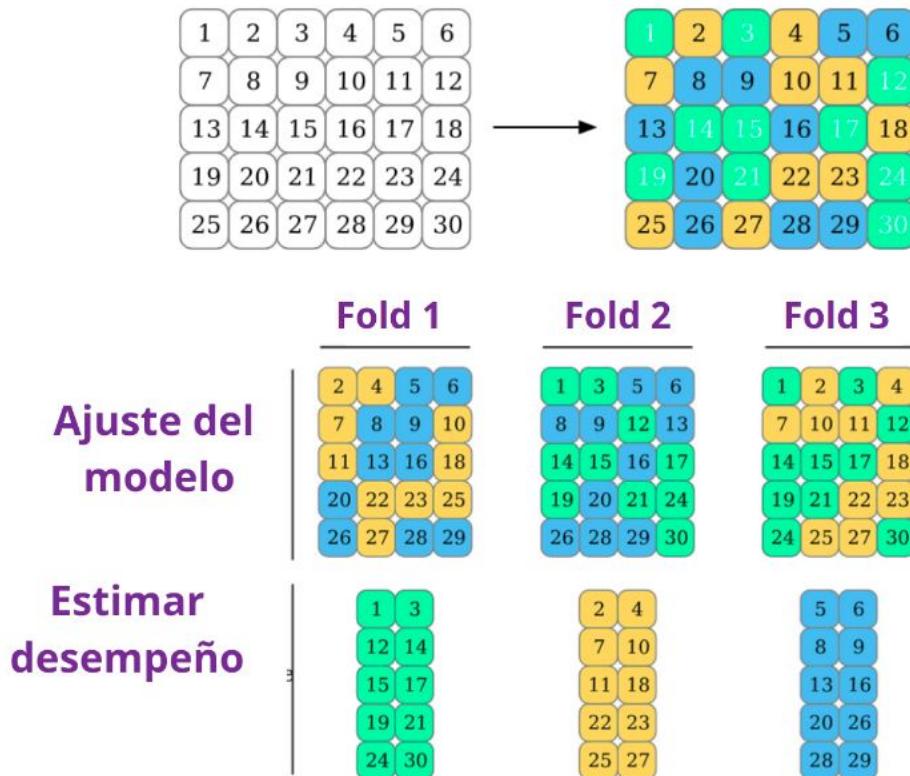


# Validación cruzada

## Opción 3: Cross - validation



# Validación cruzada (un poco más en detalle)



# *¿Para qué usamos la validación cruzada?*

La validación cruzada se utiliza para estimar el error de generalización de un modelo y también para hacer optimización de hiperparámetros.

# Bootstraping

Ajuste  
del modelo

Bootstrap  
iter 1

1	1	4	7	8	8
10	13	13	13	14	15
16	16	16	17	19	19
21	22	23	23	24	23
25	25	25	27	28	29

Bootstrap  
iter 2

2	2	3	3	3	4
4	4	6	6	7	10
11	12	12	14	14	15
17	17	18	21	22	22
23	23	28	27	28	30

Bootstrap  
iter 3

2	2	3	3	4	5
5	5	6	7	10	11
12	15	16	18	18	19
19	20	20	20	21	21
21	21	22	22	29	30

Implica repetir  
la misma  
muestra varias  
veces

Estimar  
desempeño

2	3	5	6	9	11
12	18	20	24	26	28
			30		

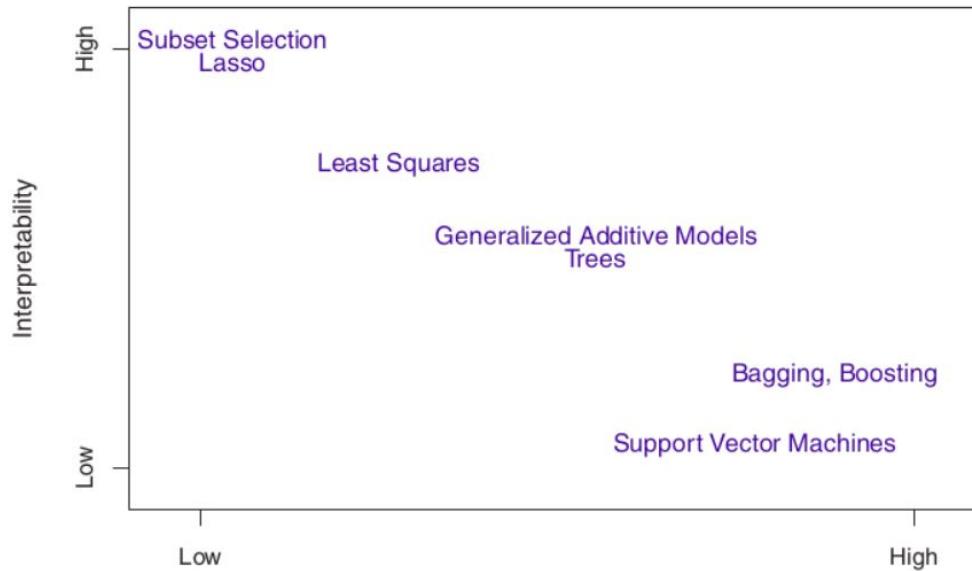
1	5	8	9	13	16
19	20	24	26	29	

1	8	9	13	14	17
23	24	25	26	27	28

Tanto validación cruzada como bootstrap son técnicas de resampleo (resampling).

La diferencia radica en que la validación cruzada no es con reemplazo mientras que bootstrap sí lo es.

# *Interpretabilidad vs Flexibilidad*



# Métricas

Durante el modelado de datos es probable que no hagamos un solo modelo, sino varios.

La manera de saber qué tan buenos son, es evaluar esos algoritmos mediante métricas.

Tenemos métricas de regresión y clasificación.

# Métricas para regresión

- Mean Square Error (MSE)

$$MSE = \frac{1}{n} \sum \underbrace{\left( y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

- Root Mean Square Error (RMSE)

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

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{n} \sum \underbrace{|y - \hat{y}|}_{\text{The absolute value of the residual}}$$

Divide by the total number of data points

Predicted output value

Actual output value

Sum of

The absolute value of the residual

# Métricas para clasificación

- Supongamos que se desea realizar una clasificación en la que disponemos de dos clases: una clasificada como Positivo y otra, como Negativo.
- A su vez, disponemos de un modelo en el cual las predicciones no coinciden 100% con las clases verdaderas.



# Positivos verdaderos



VP

Positivos predichos  
como positivos

# Falsos negativos



FN

Positivos predichos  
como negativos

# Negativos verdaderos

Clases verdaderas      Predicciones



P = Positivo N= Negativo

Negativos predichos como negativos

VN

# Falsos positivos

Clases verdaderas      Predicciones



P = Positivo N= Negativo

Negativos predichos como positivos

FP

# Matriz de confusión

Todo lo anterior puede resumirse en la Matriz de Confusión



# *Precisión vs sensibilidad*

- Uno podria elegir trabajar con las métricas de Precision o Recall para un problema desbalanceado. Maximinar la precisión minimizará los FALSOS POSITIVOS, mientras que el Recall minimizará los FALSOS NEGATIVOS.
- Entonces, podria ser adecuado trabajar con:

**Precisión: Cuando el objetivo es minimizar los falsos positivos.**

**Sensibilidad (Recall): Cuando el objetivo es minimizar los falsos negativos.**

# Curva ROC

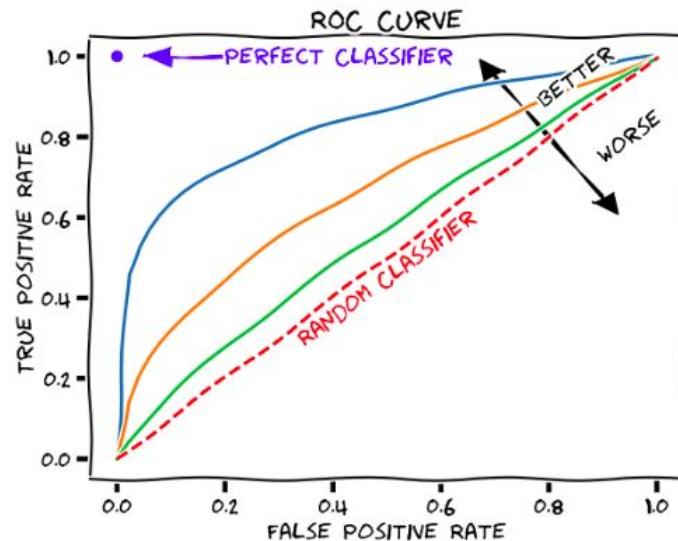
La curva ROC relaciona el recall con el ratio de falsos positivos. Es decir, relaciona la sensibilidad de nuestro modelo con los fallos optimistas (clasificar los negativos como positivos). Tiene sentido, ya que generalmente si aumentamos el recall, nuestro modelo tenderá a ser más optimista e introducirá más falsos positivos en la clasificación.

El AUC es conveniente por dos razones:

- *Es invariable con respecto a la escala.* Mide que tan bien se clasifican las predicciones, en lugar de sus valores absolutos.
- *El AUC es invariable con respecto al umbral de clasificación.*

# Curvas ROC

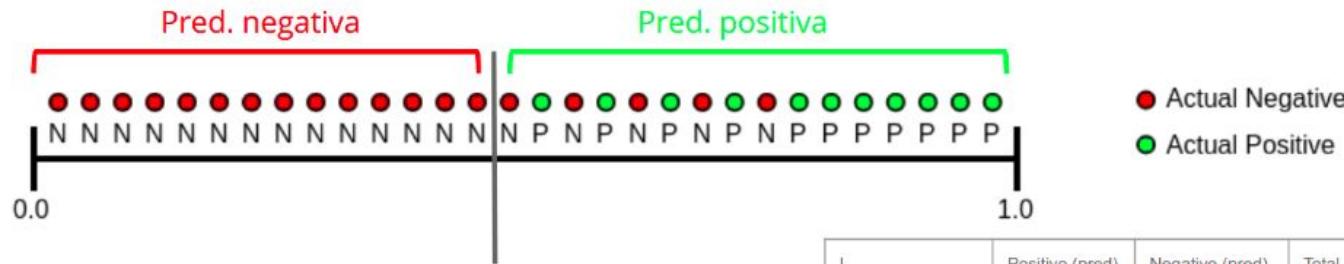
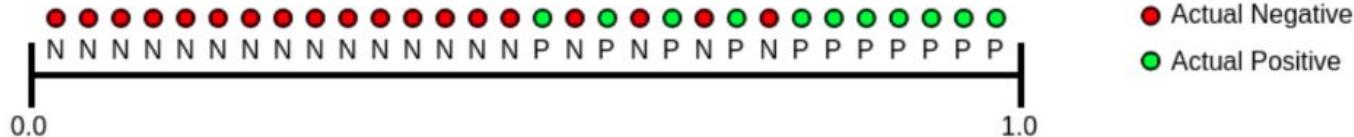
El objetivo es que la curva se acerque lo más posible a la esquina superior izquierda, de manera que aumente el recall



La curva ROC nos permite obtener AUC o área bajo la curva. El AUC oscila entre 0 y 1. Un modelo cuyas predicciones son un 100% incorrectas tiene un AUC de 0; otro cuyas predicciones son un 100% correctas tiene un AUC de 1.

# Independiente del umbral seleccionado

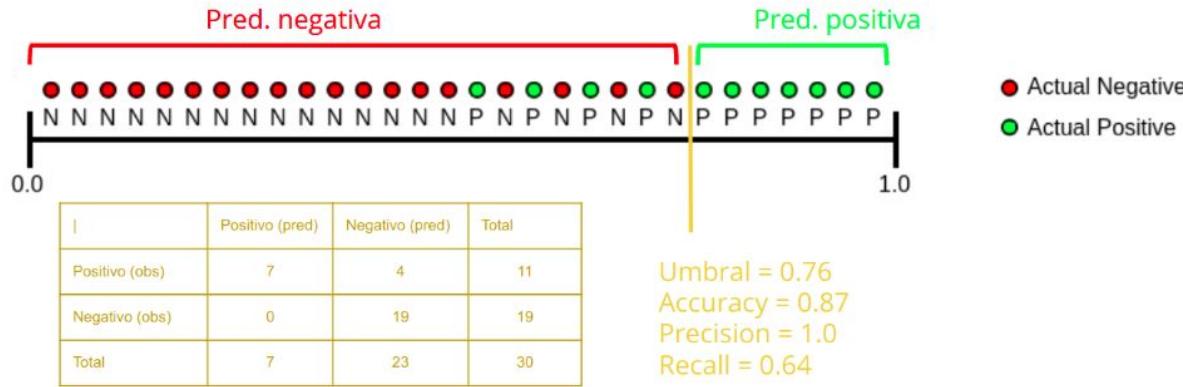
Supongamos que tenemos un modelo de clasificación binaria



Umbral = 0.5  
Accuracy = 0.83  
Precision = 0.69  
Recall = 1.0

	Positivo (pred)	Negativo (pred)	Total
Positivo (obs)	11	0	11
Negativo (obs)	5	14	19
Total	16	14	30

# Independiente del umbral seleccionado



Vamos a buscar clasificadores que rankeen:

- **alto en Y**, es decir que tengan una alta sensibilidad (recall), o alta tasa de verdaderos positivos, y
- **bajo en X**, es decir, valores bajos en falsos positivos.

# *Underfitting vs Overfitting*

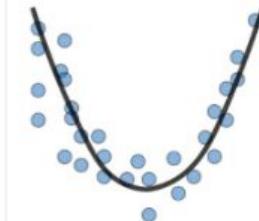
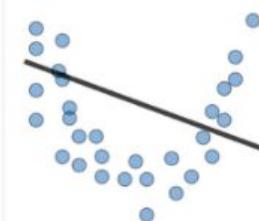
Ajuste óptimo

Ajuste pobre  
de los datos

El modelo  
sobreajusta los  
datos

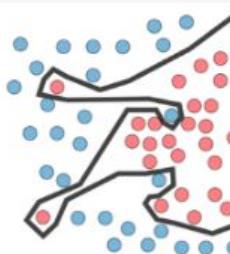
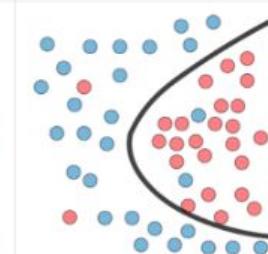
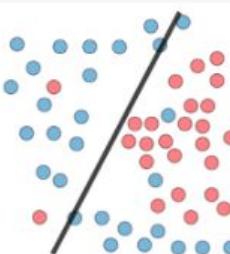
## **Underfitting**

Regression  
illustration

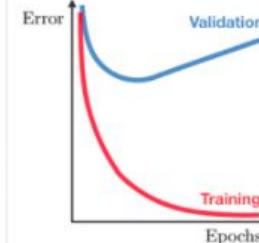
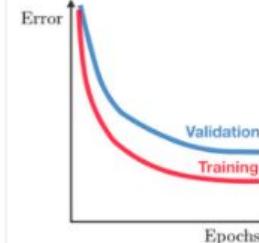


## **Overfitting**

Classification  
illustration



Deep learning  
illustration



# *Underfitting*

Cuando un modelo comete muchos errores en el entrenamiento, decimos que tienen un **alto bias**, que underfittea o que el ajuste es pobre. Esto se puede presentar debido a:

- El modelo es muy simple para los datos.
- Los features, variables (columnas) son poco informativos sobre el problema.

*Posibles soluciones:*

- Probar un modelo más complejo.
- Mejores Features
- Conseguir más datos.

# Overfitting

Cuando un modelo tiene una buena performance en el set de validación pero mala performance en test, se dice que el modelo **sobreajusta u overfittea**. También se dice, que el modelo *memoriza* los datos de entrenamiento por eso no generalizará bien frente a nuevos datos. Un modelo que sobreajusta, tiene una **alta varianza**

- El modelo es muy complejo para los datos.
- Los features, variables (columnas) son muchas para la cantidad de filas.

*Posibles soluciones:*

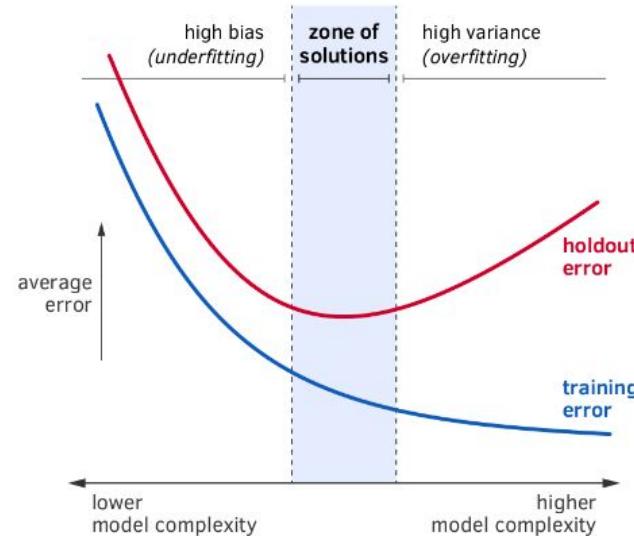
- Probar un modelo más simple.
- Menos Features
- Conseguir más datos.

## Objetivo principal del aprendizaje automático

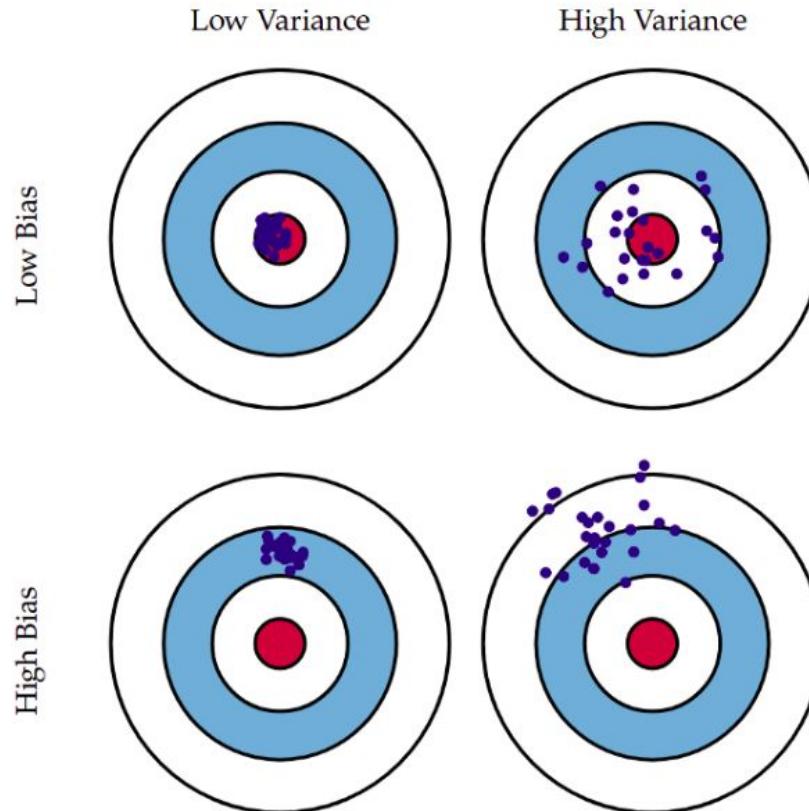
El objetivo de un modelo de ML siempre es que pueda generalizar bien frente a nuevos datos o datos no vistos por el modelo.

# *Bias variance trade-off*

En la práctica, por reducir la varianza, se reduce el bias y viceversa. En otras palabras, por reducir el overfitting, se aumenta el underfitting y viceversa. A esto llamamos bias variance trade-off. Por tener un modelo que ajusta de manera perfecta en el set de training, terminamos con un modelo pobre para los datos de test.



# Sesgo - varianza

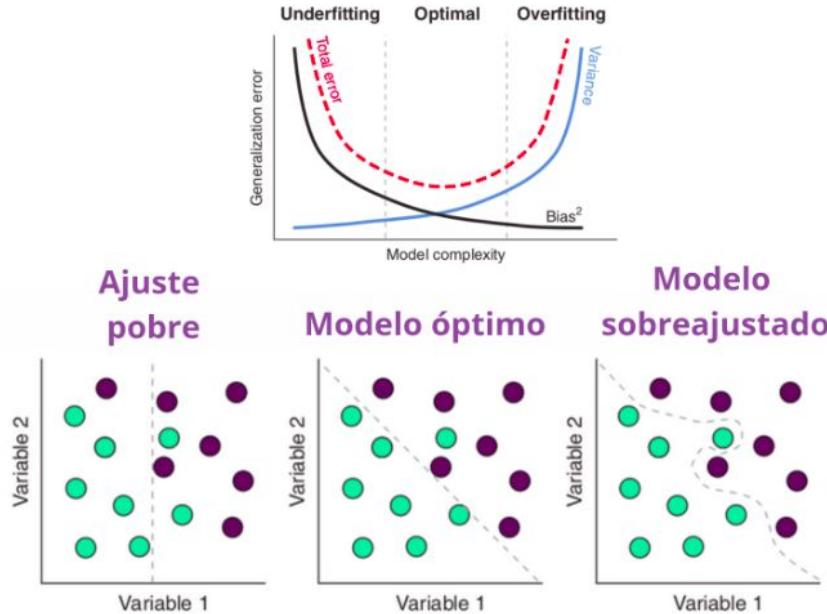


Cuando tenemos un modelo de ML, es deseable que tenga:

- baja varianza
- bajo bias.

# Bias variance trade-off

(Balance sesgo-varianza)



# Software

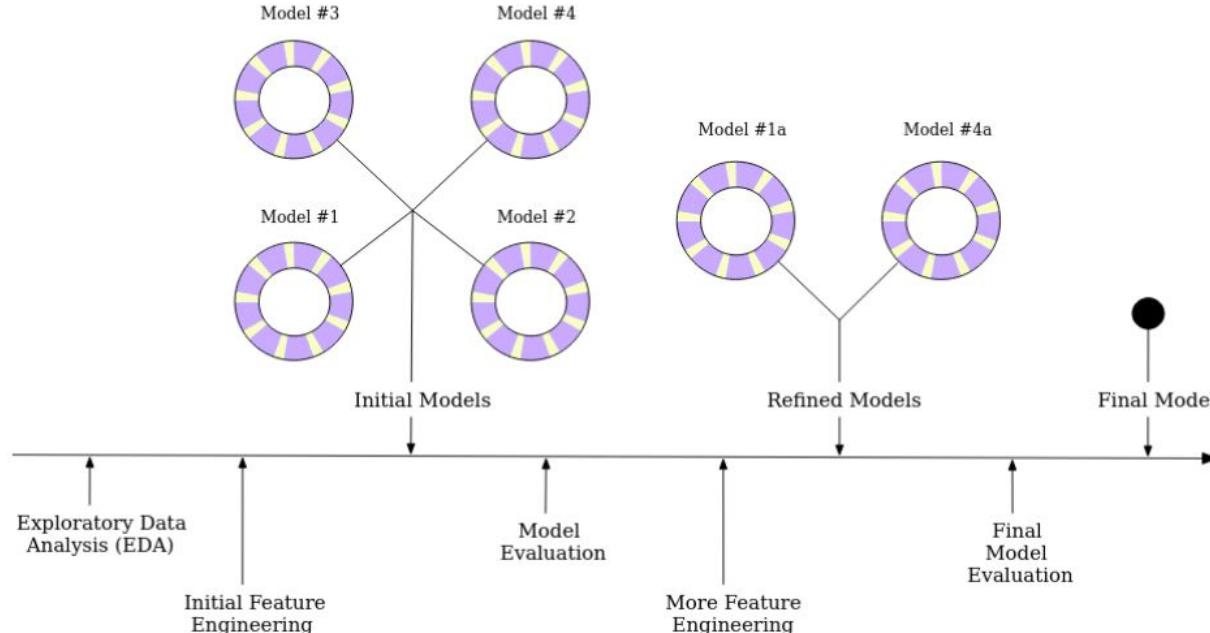
# *De caret a tidymodels en R*



- El objetivo principal de *caret* era **unificar la sintaxis** para modelar utilizando de base distintas librerías de R.
- El objetivo además de *tidymodels* es realizarlo de **manera ordenada**



# *Etapas del modelado de datos*



# *tidymodels*

