

# Machine Learning

Conceptos  
Aprendizaje Supervisado

Data Science Bootcamp

The Bridge



Un modelo de Machine Learning...

¿Qué es un modelo?

# Modelo

## *Concepto de modelo*

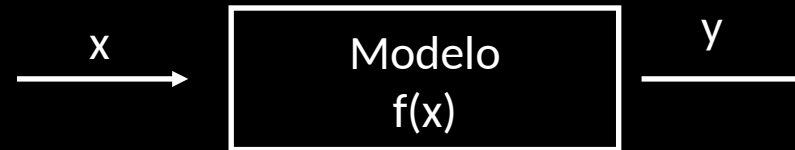
Un modelo es un conjunto de fórmulas matemáticas que expresan relaciones y patrones entre variables, utilizados para estudiar comportamientos de sistemas complejos ante situaciones difíciles de observar en la realidad.

Es una manera de traducir a la matemática relaciones entre datos de la vida real.

Variables que me digan la predisposición que tienen los pacientes a contraer cáncer de pulmón

Variables que me indiquen si un cliente es buen o mal pagador para un banco.

Qué producto estaría dispuesto a comprar un usuario X de Facebook, en función de sus gustos y búsquedas.







Con un modelo puedo predecir cierto output, como por ejemplo si el paciente contrae o no cáncer de pulmón, a partir de un conjunto de inputs, que podrían ser edad, fumador, enfermedades previas...

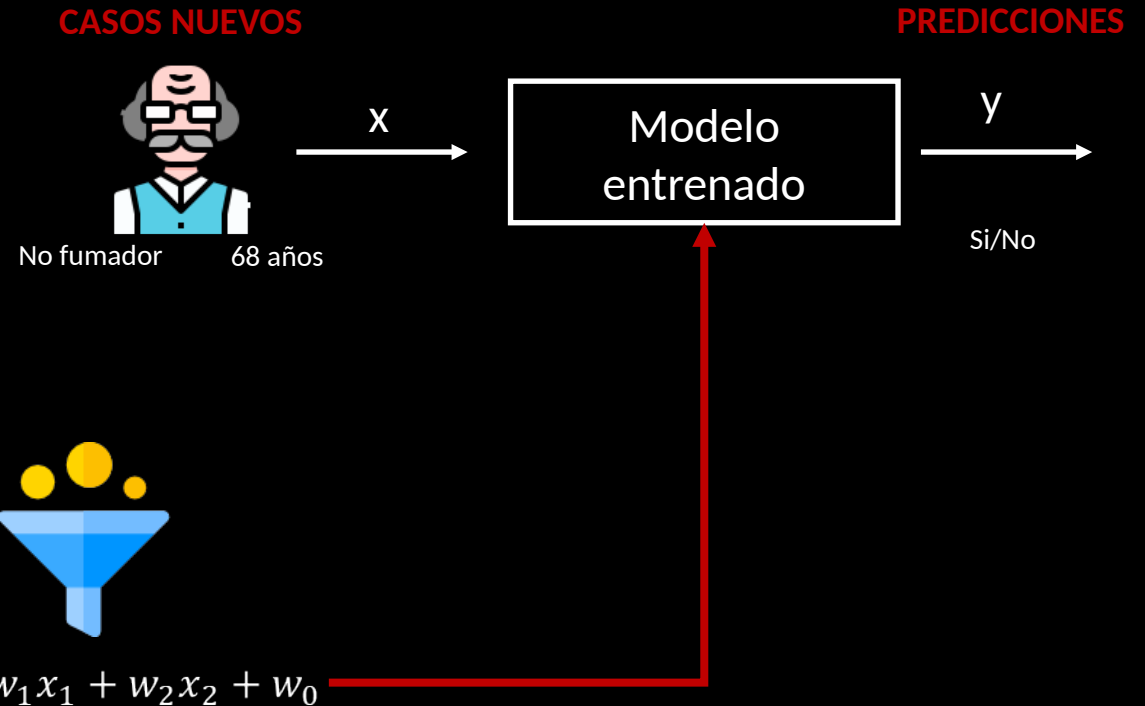
Parece que un modelo es algo que me sirve para hacer predicciones a partir de ciertos inputs. Ahora bien, ¿cómo se crea uno de estos?

# Modelo

*¿Cómo se crea un modelo?*

Un modelo necesita un histórico de datos que le sirvan de muestra, al igual que el aprendizaje de un ser humano. Con esos datos aprende patrones para predecir el output (tiene o no cáncer), de manera que cuando vengan inputs nuevos, sabrá dar un output.

	X		Y
	Fumador	Edad	¿Ha tenido cáncer?
	Si	57	Si
	No	32	No
	Si	39	No
	Si	60	No



# Target & Features

$X$  e  $Y$

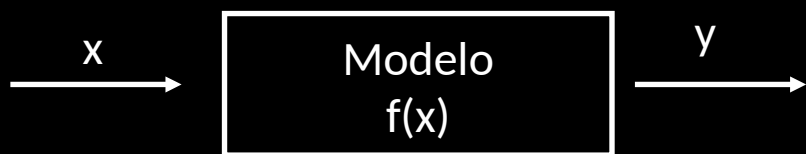
Esto es pura terminología de Machine Learning





## Target

Se trata de la variable objetivo, la variable a predecir, el output del modelo. Por norma general es una única variable, ya sea numérica o categórica

## Features

Todo el resto de variables utilizadas como input del modelo



	X (features)		Y (target)
	Fumador	Edad	¿Ha tenido cáncer?
	Si	57	Si
	No	32	No
	Si	39	No
	Si	60	No

Existen varios tipos de modelos...

¿Cómo se que he elegido el que da mejores resultados?

# Modelo

*¿Es bueno el modelo elegido?*

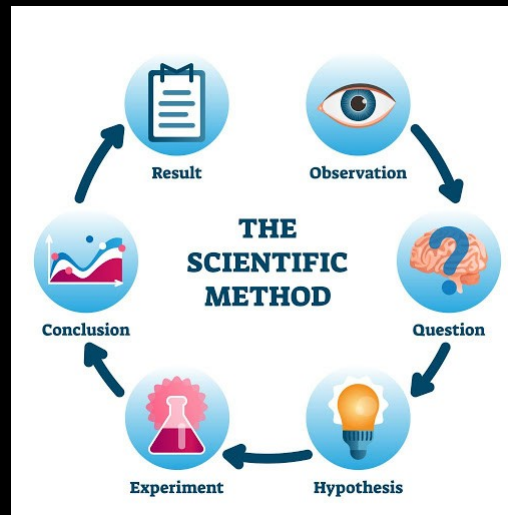
¿Cuál es el mejor modelo para nuestros datos?

A priori no lo sabemos



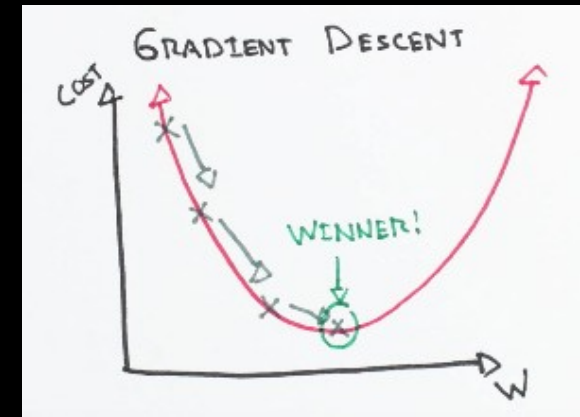
¿Cómo solucionamos esto?

Método científico



¿Qué objetivo perseguimos?

De momento...Minimizar errores.  
Encontrar un modelo que prediga con la menor cantidad de errores posible





Entonces hay que probar varios  
modelos.

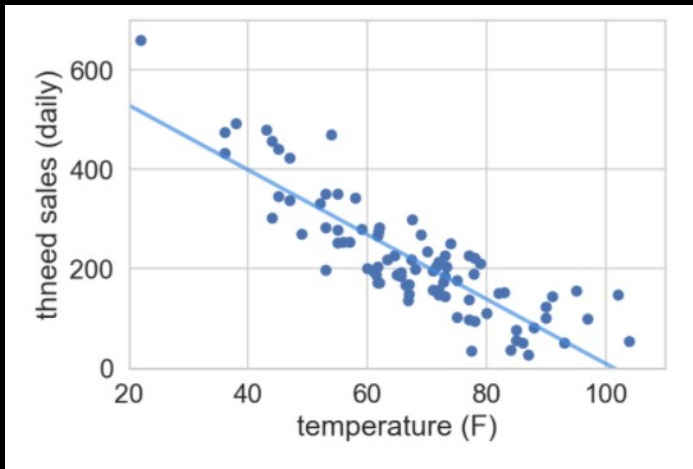
Nos quedaremos con el mejor.  
Vamos a probar

# Modelo

*¿Cuál es el mejor modelo?*

**Modelo 1**

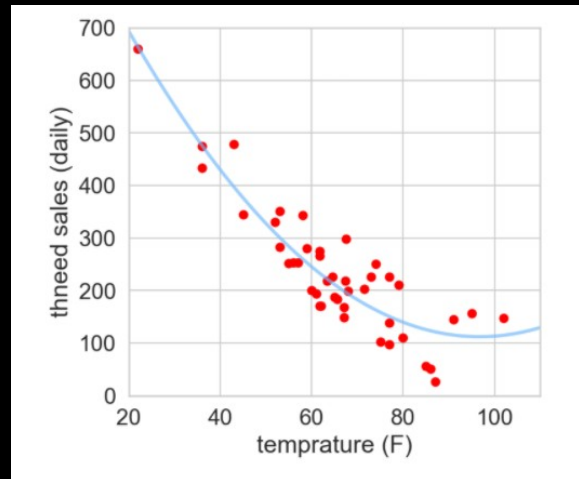
*Regresión lineal*



Error = 63.0

**Modelo 2**

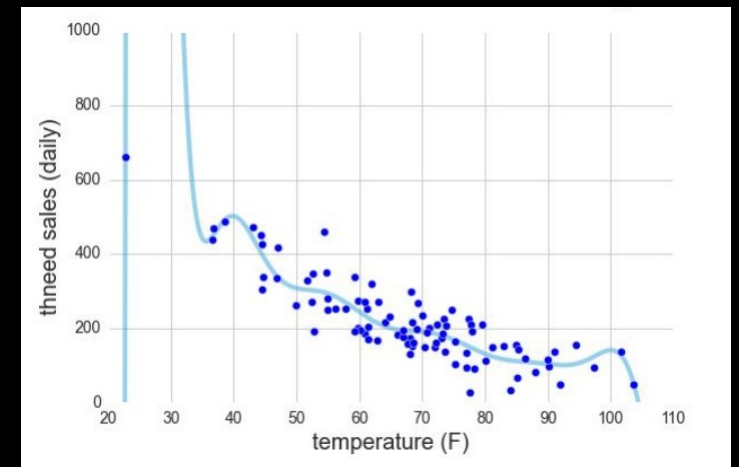
*Regresión polinómica grado 2*



Error = 51.5

**Modelo 3**

*Regresión polinómica grado > 2*



Error = 49.2

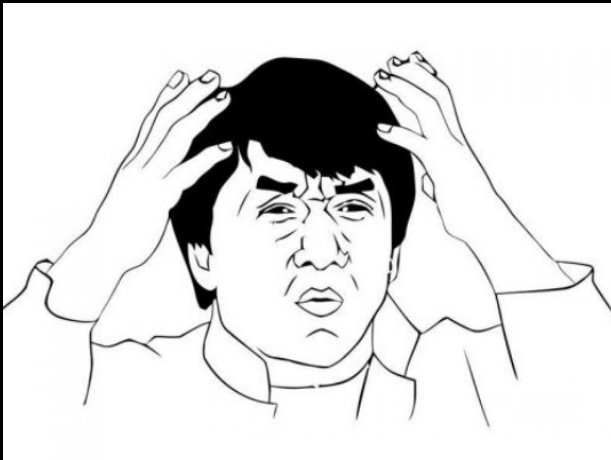
¡Bingo! Parece que he encontrado el mejor modelo

¡Ya tenemos el modelo! Vamos a  
ponerlo en producción

# Objetivo de un modelo

*Generalizar ante datos nuevos*

Error = 87.9

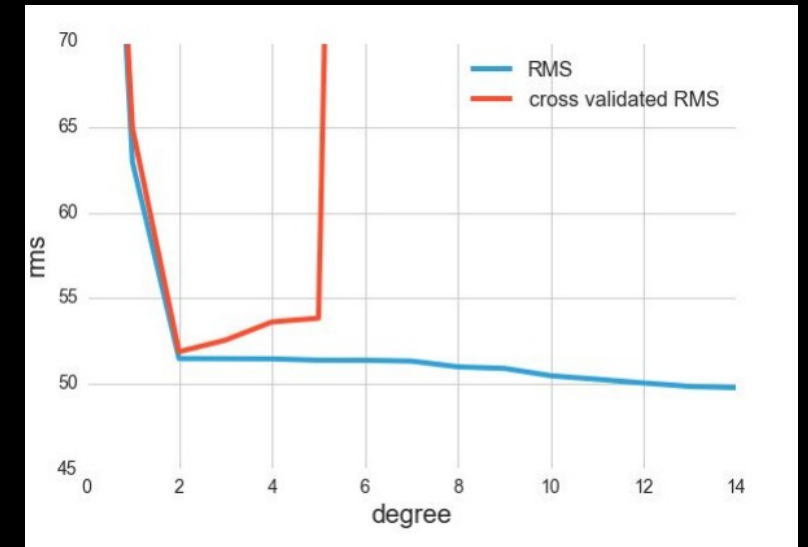


**¿Problema?**

*El modelo no **generaliza** bien*

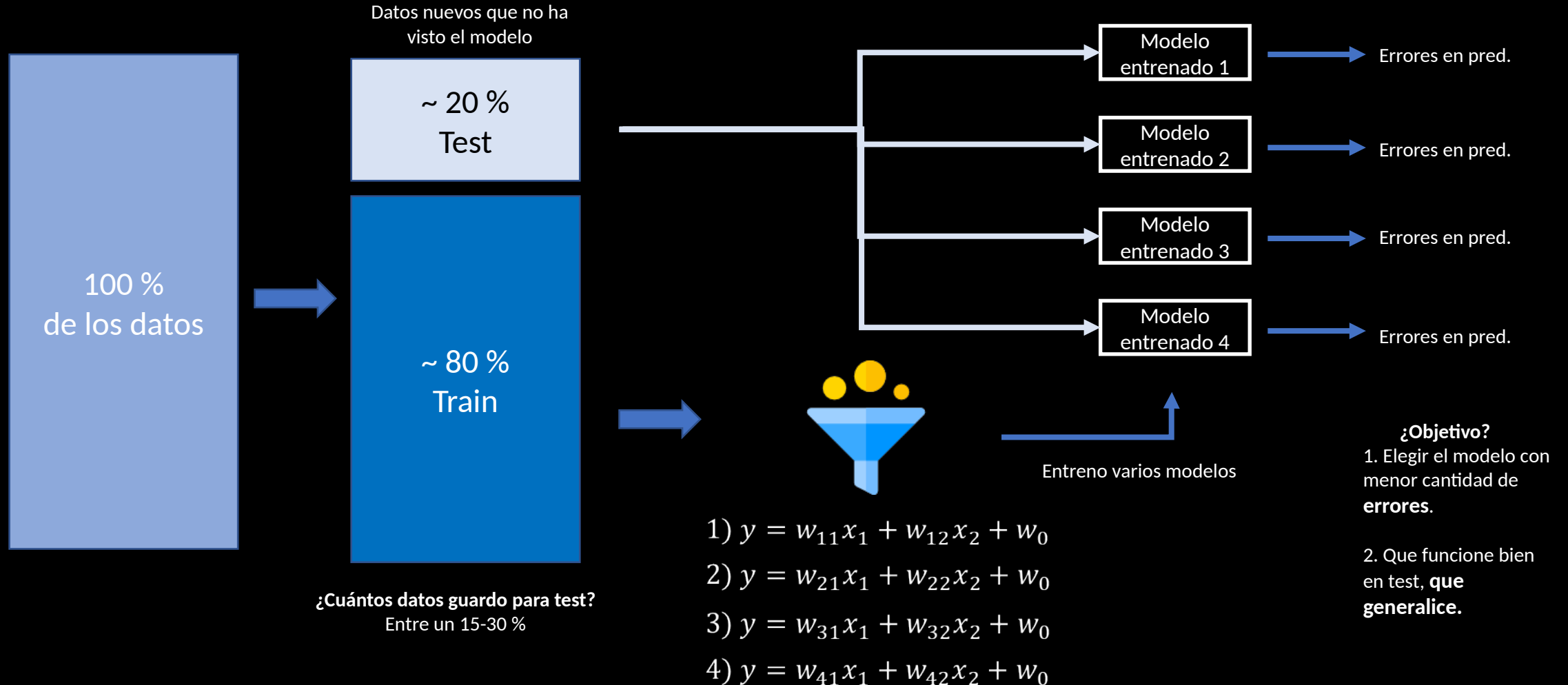
**¿Solución?**

*Dividir en train/test. Guardo unos datos de test, ajenos completamente al entrenamiento. Y pruebo el modelo entrenado, a ver qué tal generaliza con datos nuevos*



# Train - test

*División del dataset*



# Examen de matemáticas

*¿Cómo aprendo matemáticas?*

## 1. Entreno para el examen

*¿Cómo? Con ejercicios*

*Aprendo a resolver problemas*



Resolución de problemas con  
solución

## 2. ¿Cómo se que he aprendido la lección?

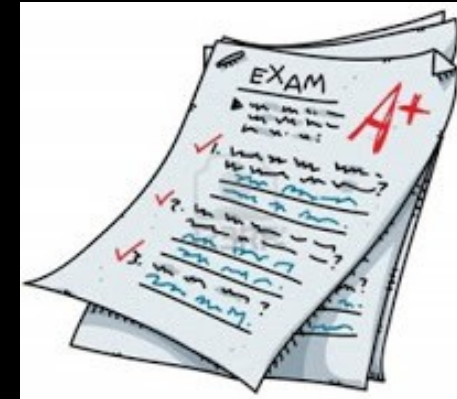
*Con un examen*



Resolución de problemas sin  
solución

## 3. ¿Cómo se si el examen ha sido bueno?

*Contando errores y aciertos en el examen*



Comparo ejercicios resueltos por el  
alumno vs soluciones.

**Métrica empleada para medir  
errores.**



# Overfitting

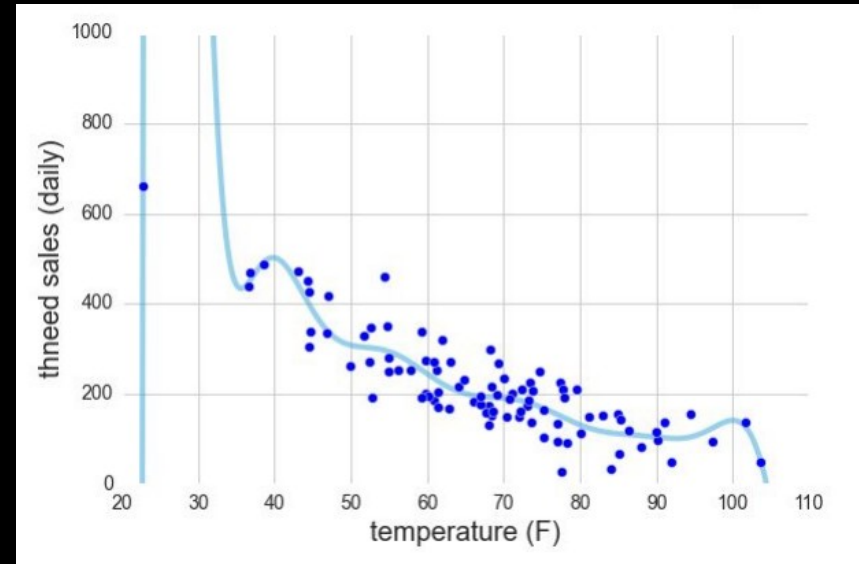
Si estudiamos los problemas de memoria y nos cambian una coma en el examen, no vamos a saber resolverlo.

Estoy **ajustándome demasiado a mis problemas** de entrenamiento y **no voy a poder generalizar**/resolver nuevos problemas, desconocidos para mí.

Tengo que aprender los patrones de los problemas, y la manera de resolverlos, no aprendérmelos de memoria

## Overfitting o sobreajuste

*Sobreajuste de un modelo a los datos de entrenamiento. Como consecuencia el modelo no va a generalizar bien. No se va a comportar como debería cuando le entren nuevos datos*



## ¿Solución?

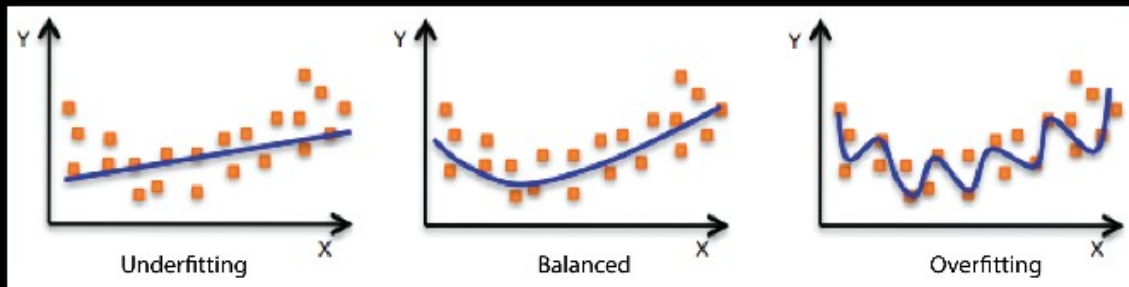
1. Utilizar más datos en training
2. Reducir la complejidad de los modelos
3. Cross Validation!!



La complejidad del modelo me perjudica en test. Habrá que probar modelos más simples

# Underfitting

Se produce cuando entrenamos con pocos datos un modelo, y este no es capaz de identificar los patrones que hay en los datos para sus predicciones. Por tanto este modelo **NO puede generalizar bien**, para cuando le lleguen datos nuevos.



## Underfitting & Overfitting Machine Learning

### Underfitting

Entreno al modelo con  
1 sólo raza de perro



Muestra nueva:  
¿Es perro?



NO  
FALLO

La máquina fallará en reconocer al perro por falta de suficientes muestras. No puede generalizar el conocimiento.

### Overfitting

Entreno al modelo con  
10 razas de perro color marrón



Muestra nueva:  
¿Es perro?



NO  
FALLO

La máquina fallará en reconocer un perro nuevo porque no tiene estrictamente los mismos valores de las muestras de entrenamiento.

¡Qué lío!...¿Hago modelos simples?  
¿Hago modelos complejos?

# Bias vs Variance

*Hay que encontrar el equilibrio*

## Bias

Capacidad de un modelo de ajustarse a los datos de entrenamiento. Si se ajusta de más, tendremos Overfitting, y por tanto un Variance alto.

## Variance

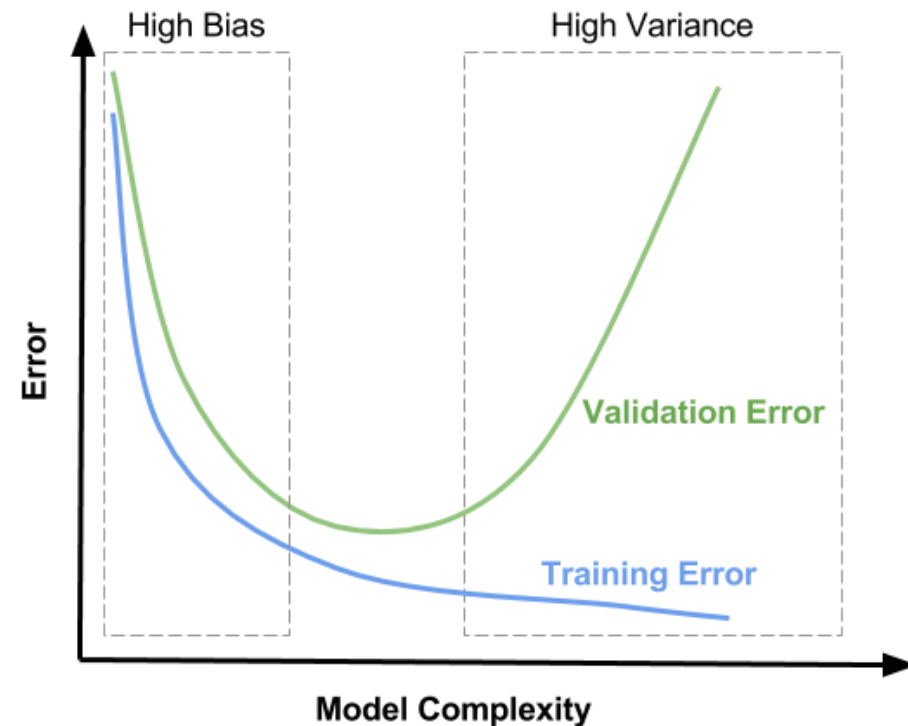
Capacidad de un modelo para mantener sus regiones de decisión ante pequeñas variaciones de los datos.

## ¿Objetivo?

Reducir el bias y el variance al máximo. Con esto conseguimos tener un modelo que se ajuste al patrón de los datos y al mismo tiempo generalizar con datos futuros

Underfitting

Overfitting



[Demo interesante](#)

¿Hay alguna técnica para solventar esto?

Regularization  
Cross Validation

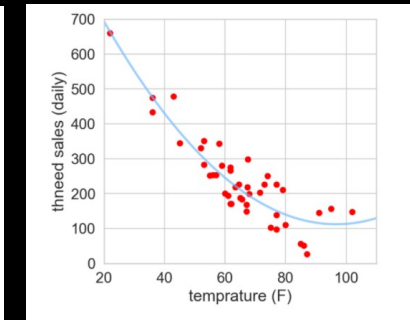
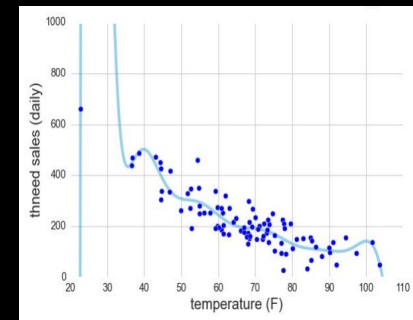
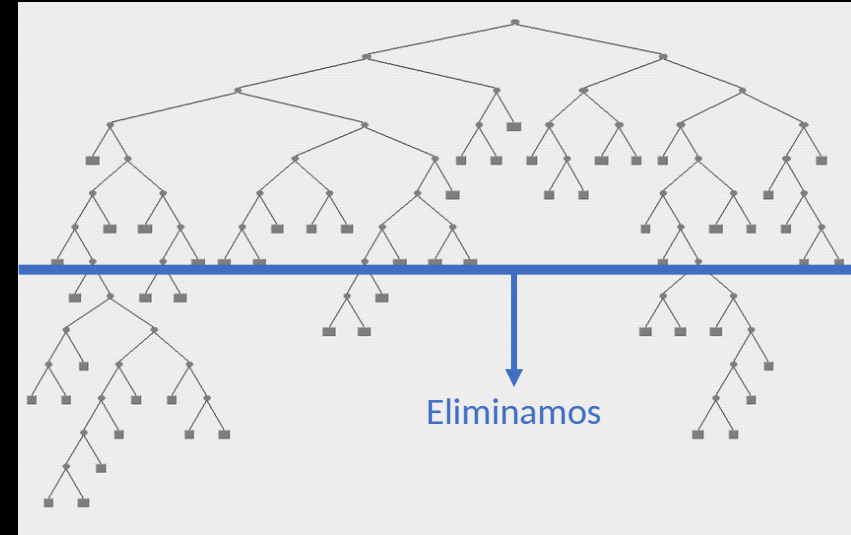
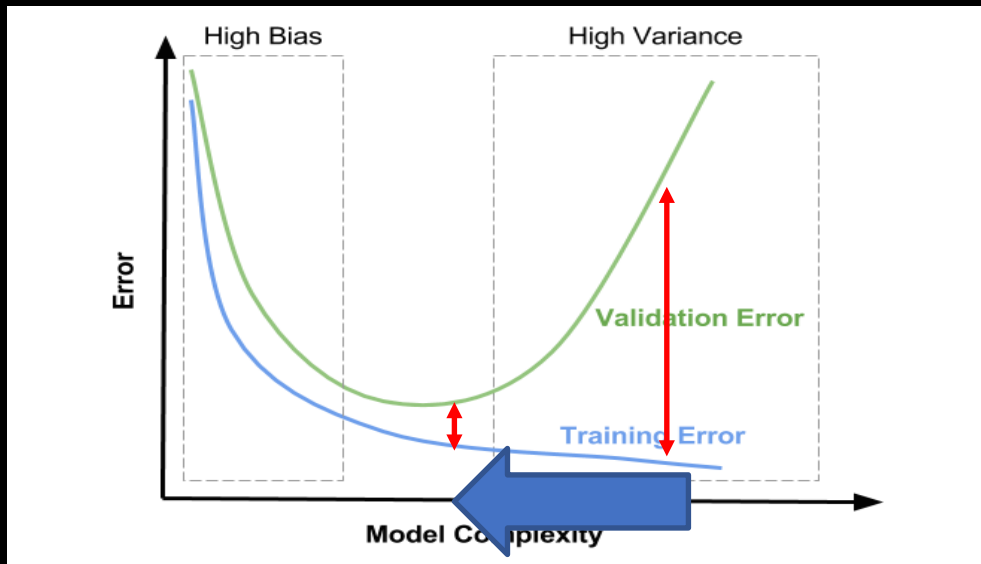
# Regularization

*Simplificación del modelo para que generalice bien*

**¿Qué conseguimos con esto?**

Aumentamos los errores en train a cambio de disminuirlos en test.

Reducimos ese gap entre errores



# Cross Validation

*Utilizamos los datos de train en test y viceversa*

Ejemplo para K=5

1. Mezclar datos
2. Divide tus datos en 5 conjuntos (A, B, C, D, E)
3. Guarda 1/5 de los datos para test. Entrena con B, C, D, E y utiliza A para test
4. Te da un error
5. Ahora guárdate B para test y utiliza A, C, D, E de entrenamiento.
6. Da otro error
7. Cuando acabes de hacer eso con todos los *folds*, calcula la media de todos los erros obtenidos y será el error cometido por el modelo.

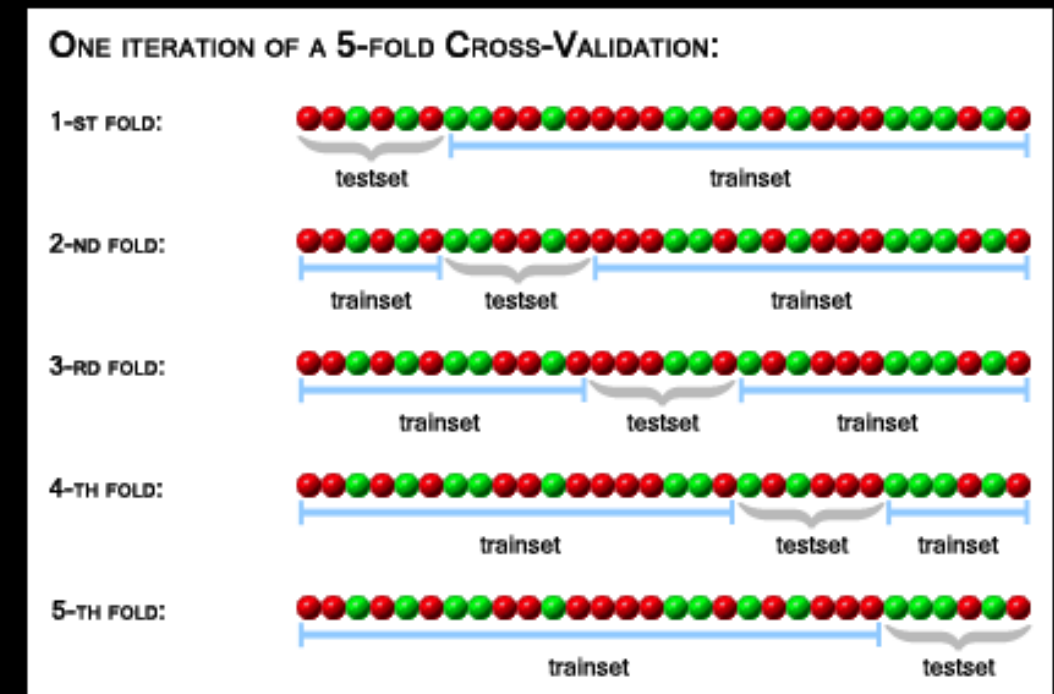
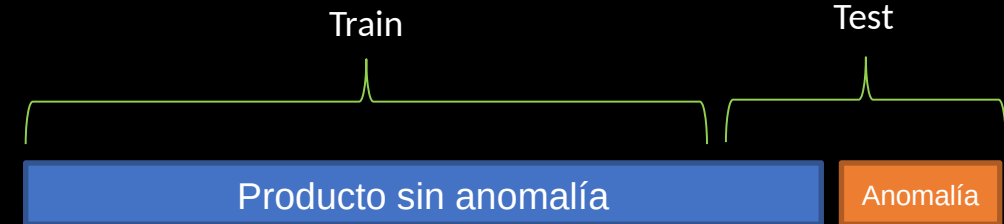
## ¿Qué conseguimos con esto?

Garantizar que la evaluación del modelo sea independiente de la partición.

Al utilizar todos los datos de train, en test, y los datos de test a su vez en train también, conseguiré que mi modelo generalice bien, y por tanto no me produzca overfitting.

## ¿En cuántos k-folds hay que dividirlo?

Se suelen usar 10, aunque depende del volumen del dataset. Si es muy grande, dividir en 5 k-folds.

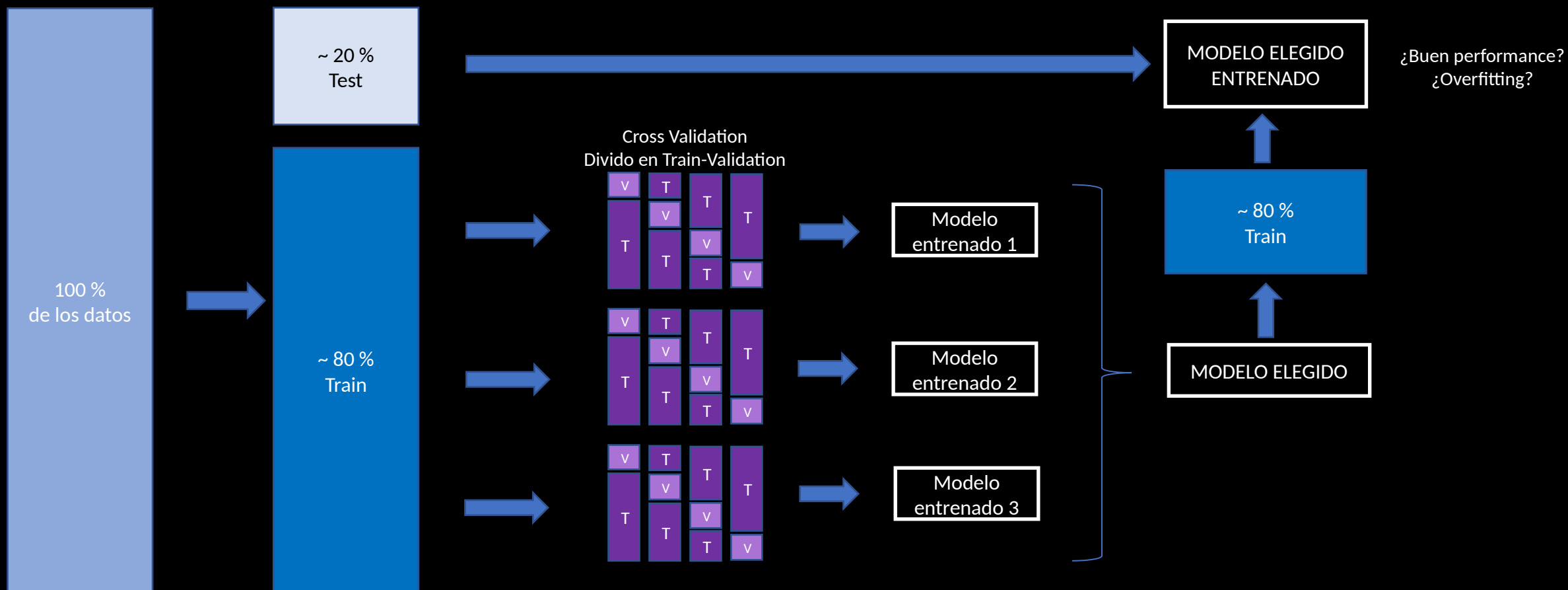


Entonces con Cross Validation  
ya no necesito dividir en train  
test...**No exactamente**



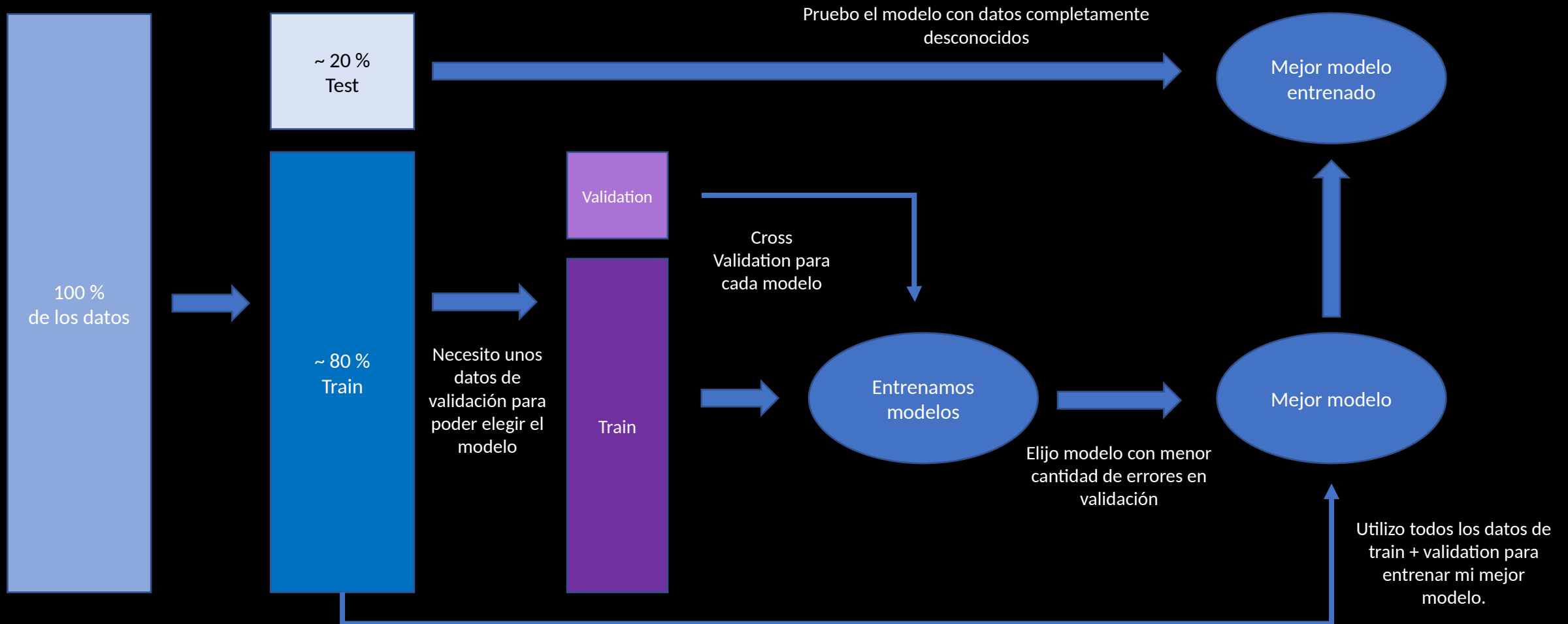
# Train – Test – Validation

*División del dataset*



# Train - Test - Validation

*División del dataset*

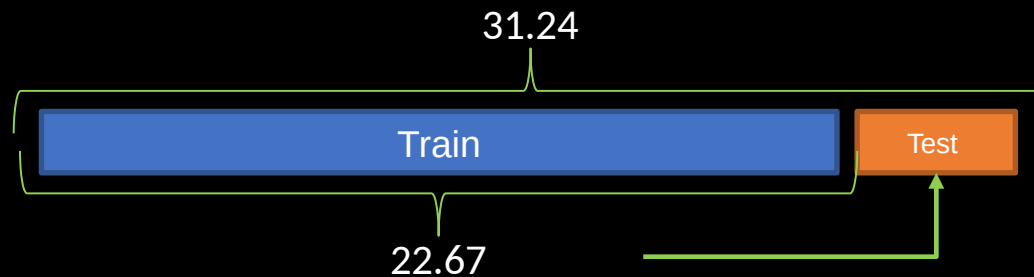


# Train – Test – Validation

## *Algunas consideraciones*

Cuidado al aplicar modificaciones en los conjuntos de train/test. **No podemos contaminar ambos conjuntos con datos del otro.** El conjunto de test se separa de los datos y no se toca hasta el final. Todas las transformaciones que apliquemos en train, tiene que salir únicamente del conjunto de train.

Ejemplo: si vamos a imputar missings en una columna por su media (imaginemos 31.24 con todos los datos), y sustituimos en train, estaríamos utilizando los datos de test para sustituir ese missing en train, cuando en realidad **para los datos de train, los de test son completamente desconocidos**. La media del conjunto de train podría ser de 22.67, muy lejos de la media de train + test.



## ¿Cómo aplicar las transformaciones/limpieza obtenidas hasta ahora?

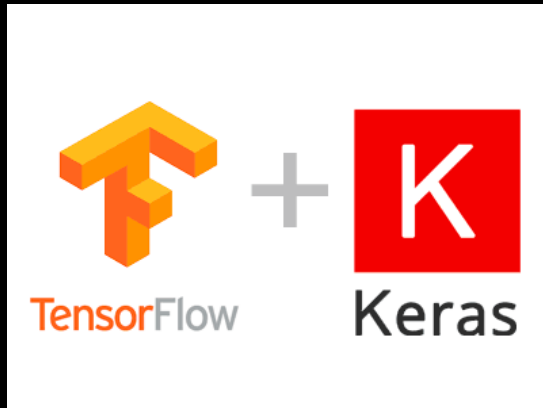
- **Scalers:** por ejemplo, si tenemos un StandardScaler en train, usar ese mismo StandardScaler en test.
- **Missings:** si aplicábamos la media/mediana/moda en train, aplicar esa misma métrica en test.
- **Feature reduction:** eliminar las features que quitábamos en train (y añadir los dummies vacíos en test)
- **Feature engineering:** mismos cálculos que en train.

Esto lo veo muy complicado  
¿Cómo voy a programar un  
modelo?

# Machine Learning en Python



sklearn



keras





Mucha info  
Vamos a resumir

# Resumen

## Modelo

A partir de una serie de inputs (features), podré realizar predicciones sobre una variable objetivo (target)

## ¿Qué necesito?

Datos, y un buen lenguaje de programación con librerías de machine learning (sklearn y keras de python)

## ¿Cómo construyo un modelo?

Entrenándolo con datos y buscando su error mínimo. Para ello hay que aplicar el método científico.

Divido los datos en train + test. El conjunto de train a su vez lo divido en train + validation que utilizo para escoger el mejor modelo.

Pruebo el mejor modelo en test. Siempre busco que tenga pocos errores y generalice bien.

## Objetivo del modelo

Pocos errores y que generalice bien cuando se enfrente a datos nuevos. Tiene que ser capaz de predecir bien ante datos futuros.

## Overfitting y Underfitting

Underfitting es que el modelo no encuentra bien las asociaciones entre los datos, y overfitting que se ajusta demasiado al entrenamiento. En ambos casos no va a generalizar bien el modelo. Hay que buscar el equilibrio (bias vs variance)