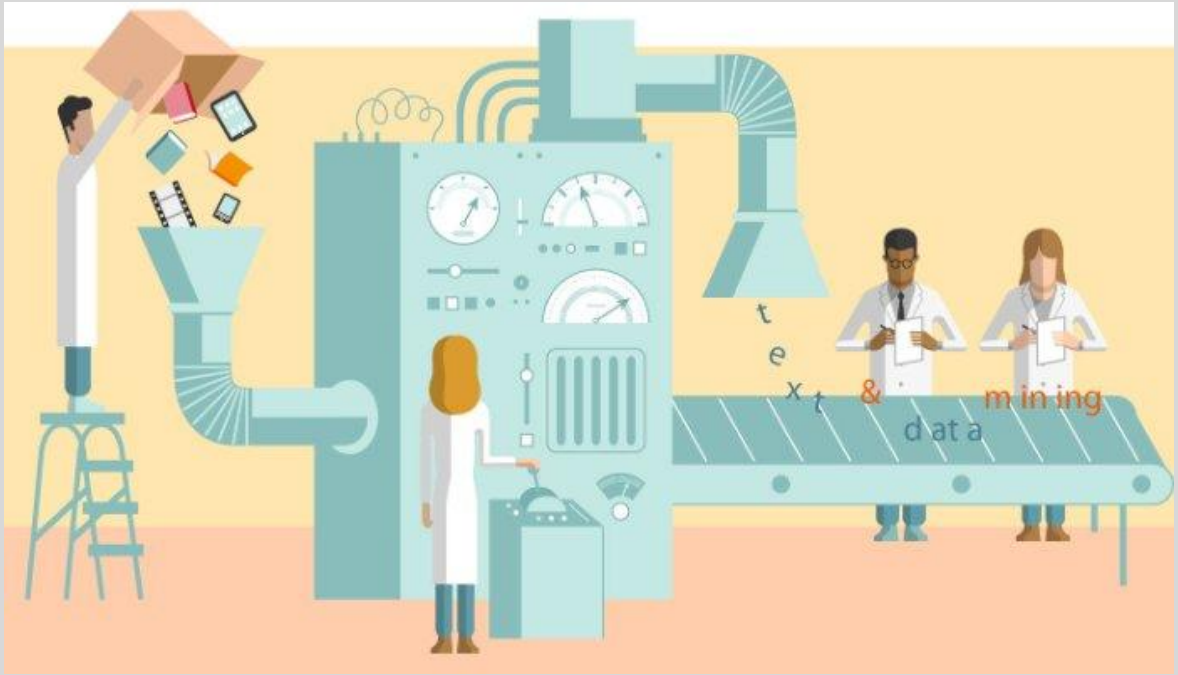


DigitalHouse > Coding School

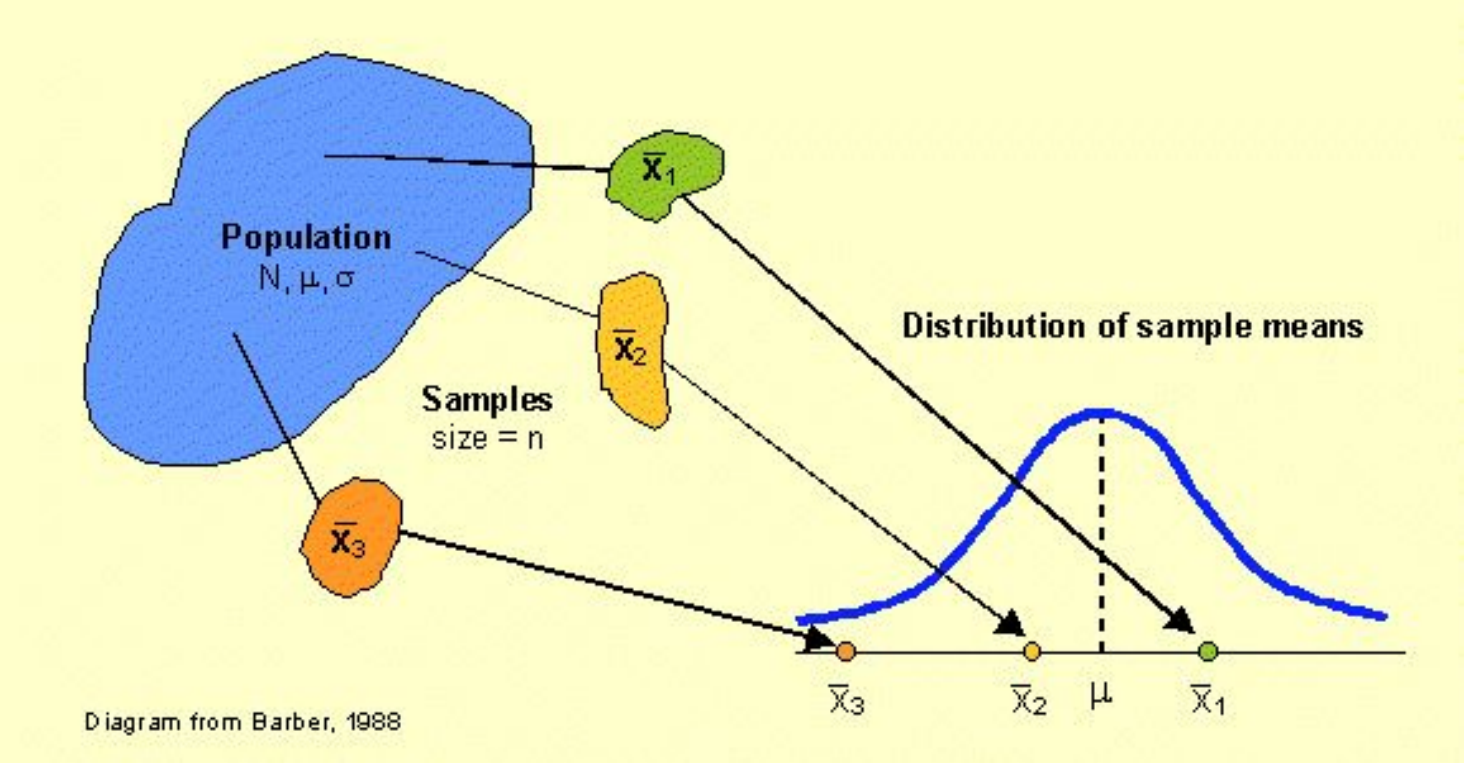
DATA SCIENCE

Repaso para la Evaluación Individual

Estadística Inferencial

Distribución de las medias muestrales:

Simulación:
https://gallery.shinyapps.io/CLT_mean/



Teorema (del límite central): Sea $X_1, X_2, ..., X_n$ un conjunto de variables aleatorias, independientes e idénticamente distribuidas de una distribución con media **μ** y varianza $\sigma^2 \neq 0$. Entonces, si **n** es suficientemente grande, la variable aleatoria

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

tiene aproximadamente una distribución normal con $\mu_{\bar{X}} = \mu$ y $\sigma^2_{\bar{X}} = \frac{\sigma^2}{n}$.

Condiciones del TCL

- **Independencia**: las observaciones tienen que provenir de un **muestreo aleatorio** y deben ser **independientes**. Si el muestreo es sin reemplazo, entonces n < 10% de la población.
- Si la distribución de una población es **muy asimétrica**, **n deberá ser muy grande** (a veces se menciona un n=30 como aproximación, pero en realidad depende de cuán asimétrica es la población). Si la población tiene distribución normal, no hay condiciones sobre n.

1. Las hipótesis: siempre sobre un **parámetro poblacional.**

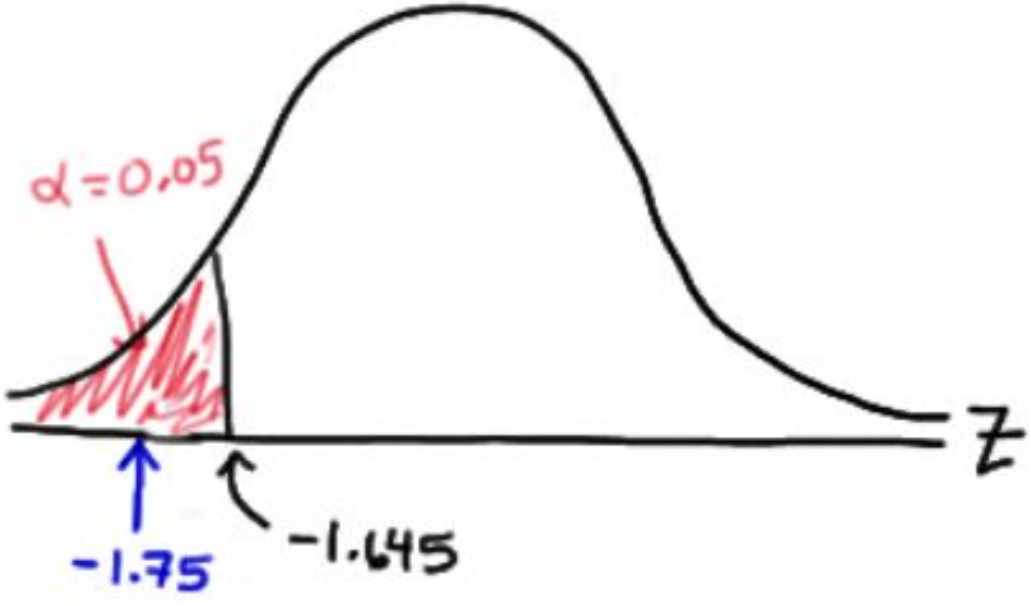
2. Hipótesis **nula** vs. hipótesis **alternativa.**

3. Nivel de significación (**alpha**).

Definición de **p-value**:

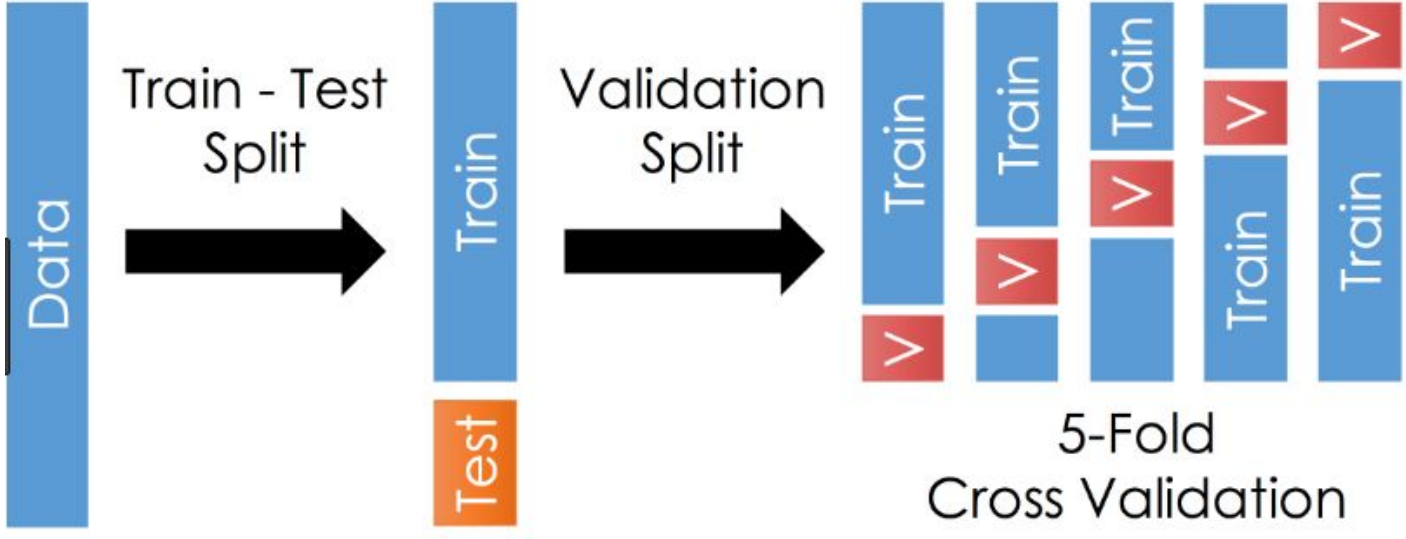
- "La probabilidad de obtener el valor observado o más extremos del estadístico de prueba si la hipótesis nula fuese cierta"

Si p-value < nivel de significación, entonces rechazo H_0. Forma alternativa de fijar la regla de decisión. No necesito buscar valores críticos en una tabla.


α = 0,05
Z
-1.75
-1.645

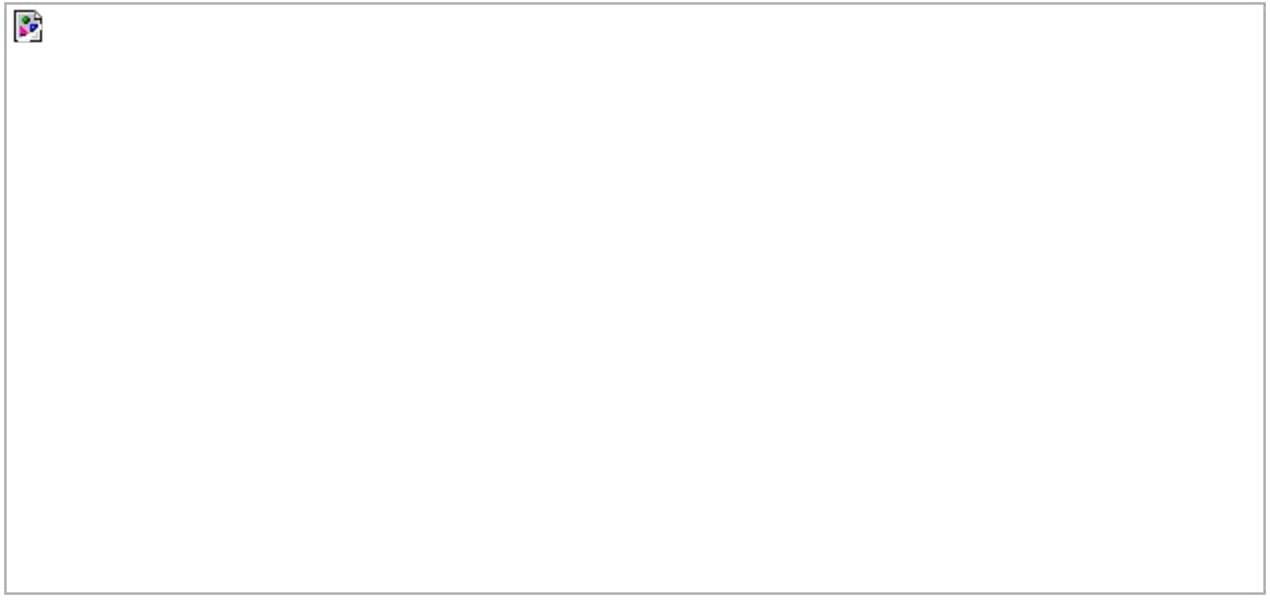
Pensando en Model Validation

El esquema completo sería el siguiente. La cantidad de **folds** es otro aspecto a probar (en general, **entre 5 y 10** particiones)



1) Time Series Split: dividir train y test, con la condición de dejar en test sólo observaciones del futuro.

2) **CV** de series de tiempo (**rolling forecasting origin)**: se construyen una serie de pruebas donde se parte el set en train y test pero incrementando la cantidad de información que se usa.



Seleccionar el mejor modelo

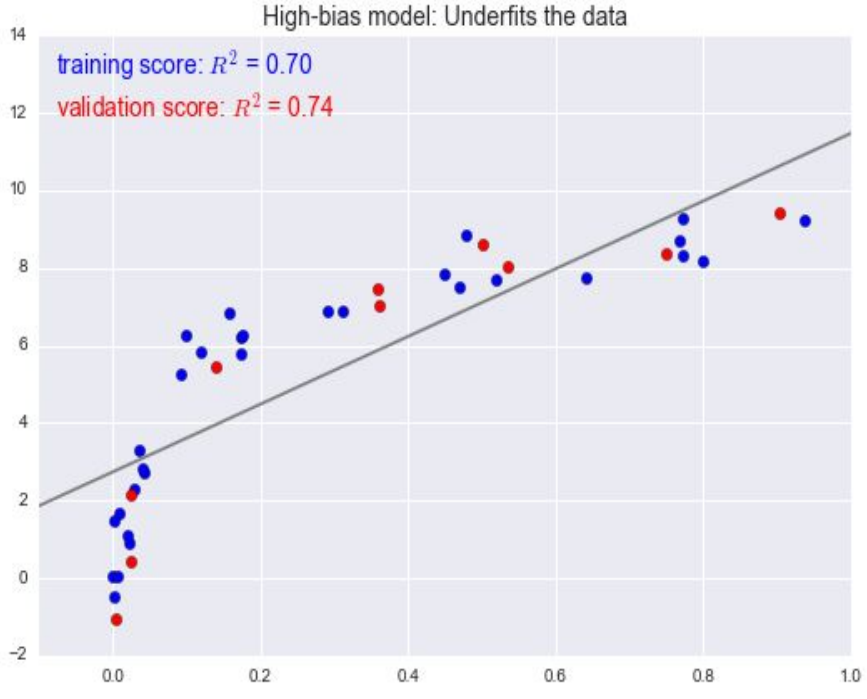
Si nuestro estimador tiene un rendimiento bajo, ¿cómo deberíamos avanzar?

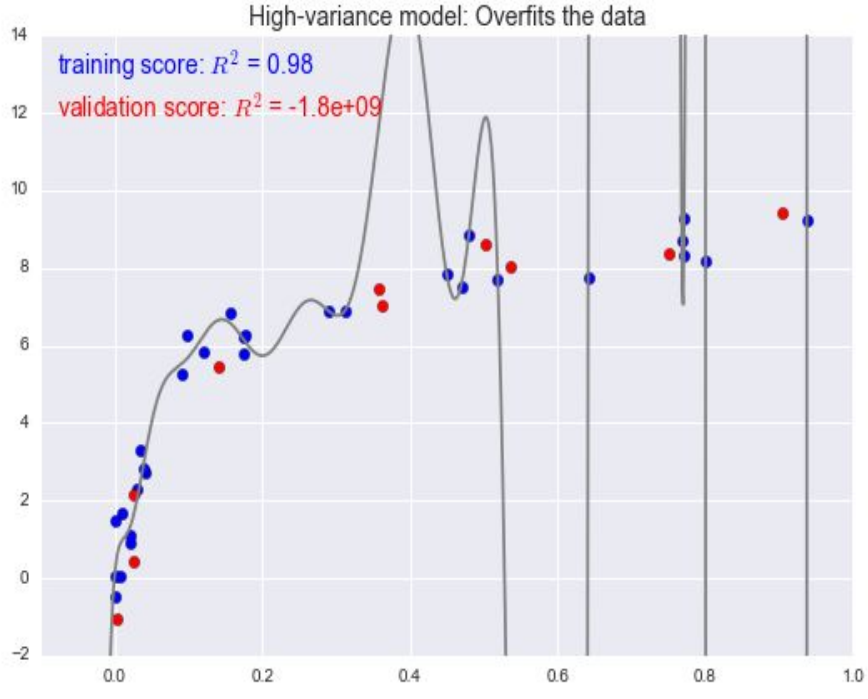
La respuesta a esta pregunta es frecuentemente contraintuitiva.

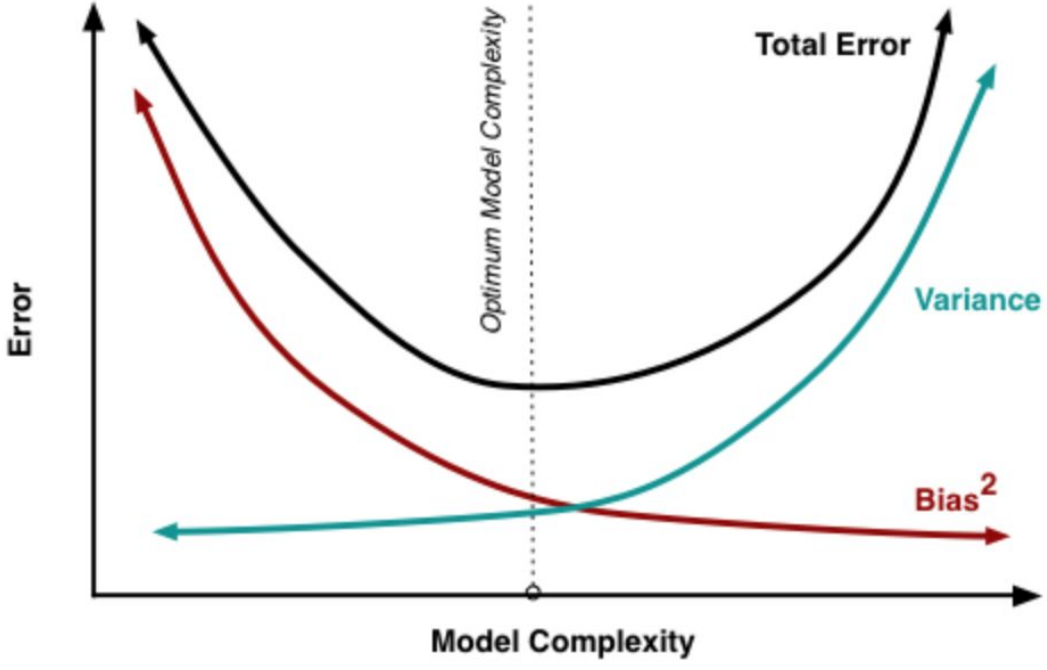
Por ejemplo:

- A veces, usar un modelo más complicado nos dará peores resultados
- Agregar más muestras de training puede no mejorar tus resultados

Fundamentalmente, la búsqueda del "mejor modelo" se trata en encontrar un punto óptimo en lo que se conoce como el dilema entre el sesgo y la varianza.

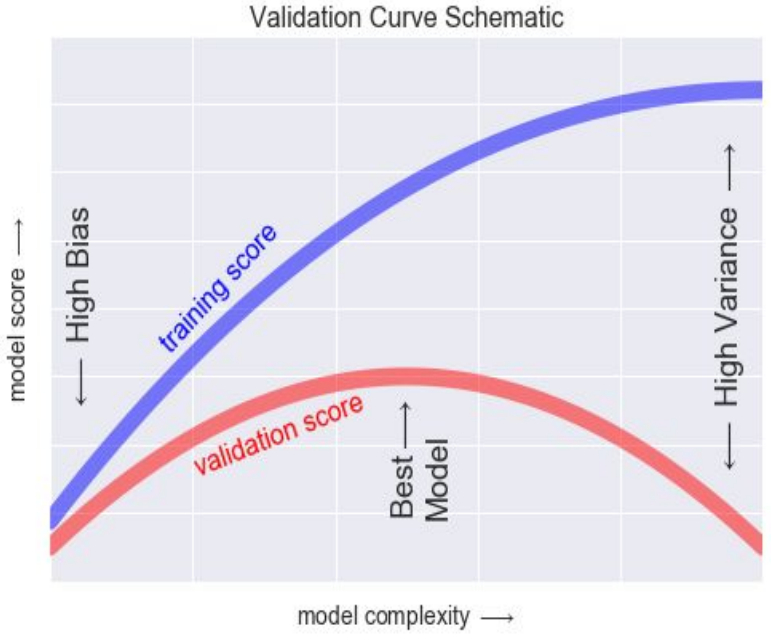

High-bias model: Underfits the data
training score: R^2 = 0.70
validation score: R^2 = 0.74
14
12
10
8
6
4
2
0
−2
0.0
0.2
0.4
0.6
0.8
1.0


High-variance model: Overfits the data
training score: R^2 = 0.98
validation score: R^2 = -1.8e+09
14
12
10
8
6
4
2
0
−2
0.0
0.2
0.4
0.6
0.8
1.0



Por lo tanto, el modelo no debe ser ni muy simple ni muy complejo

- Número de observaciones constante
- El score de entrenamiento es mayor que el score de validación.
- Para un modelo de muy baja complejidad (sesgo alto), el set de entrenamiento se **underfitea**
- Para modelos con muy alta complejidad (modelos con alta varianza), los datos de entrenamiento son **overfiteados**
- Para algún modelo intermedio, la validation curve tiene un máximo. Este nivel de complejidad indica un **compromiso apropiado entre sesgo y varianza**.

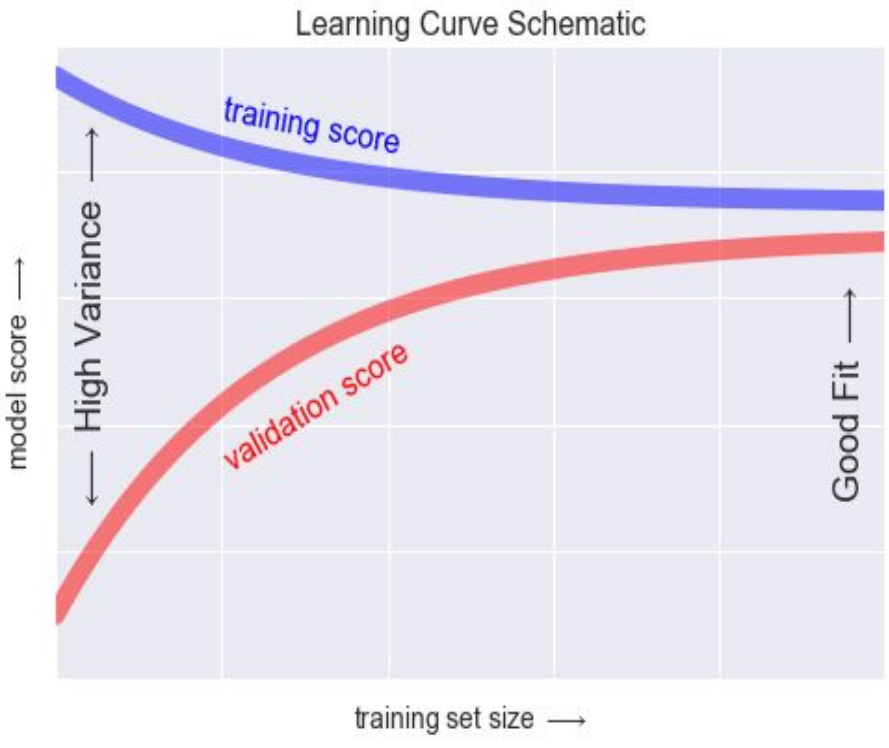


La complejidad del modelo es constante

La característica notable de la learning curve es **la convergencia a un score particular a medida que el número de muestras de entrenamiento crece.**

En particular, una vez que tenemos puntos suficientes para que el modelo haya convergido, **agregar más puntos de entrenamiento no ayudará a mejorar el score.**

La única forma de incrementar la performance del modelo en este caso es usar otro modelo, generalmente más complejo.



Regresión Lineal: supuestos de Gauss-Markov

1. El modelo es **lineal en los parámetros**.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

2. Los estimadores de los parámetros poblacionales se estiman a partir de una **muestra aleatoria**.

3. **No hay colinealidad perfecta** entre las variables explicativas.

4. El valor **esperado del error es 0** para cualquier valor de la variable explicativa.
5. Para cualquier valor de la variable explicativa, el error tienen la misma varianza (**homocedasticidad**).
6. El error es independiente de las variables explicativas y se distribuye normalmente.

$$\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$$

7. No existe autocorrelación entre los errores de dos observaciones diferentes condicionadas a X.

$$Cov(\varepsilon_i, \varepsilon_h | X) = 0$$

Regularización

Función de pérdida de una regresión lineal:

$$CF = \sum_{i}^{N} (\hat{y}_i - y_i)^2$$

Las técnicas de regularización agregan una “penalidad” a esa función de costo:

$$CF = \sum_{i}^{N} (\hat{y}_i - y_i)^2 + \alpha\vec{\theta}$$

Las técnicas de regularización agregan una “penalidad” a esa función de costo:

$$CF = \sum_{i}^{N} (\hat{y}_i - y_i)^2 + \alpha\vec{\theta}$$

Theta es el vector que corresponde a los parámetros del modelo (en una regresión lineal, los betas) y alpha es un parámetro que “regula” la fuerza de la penalización: cuanto más grande es, mayor es la penalización.

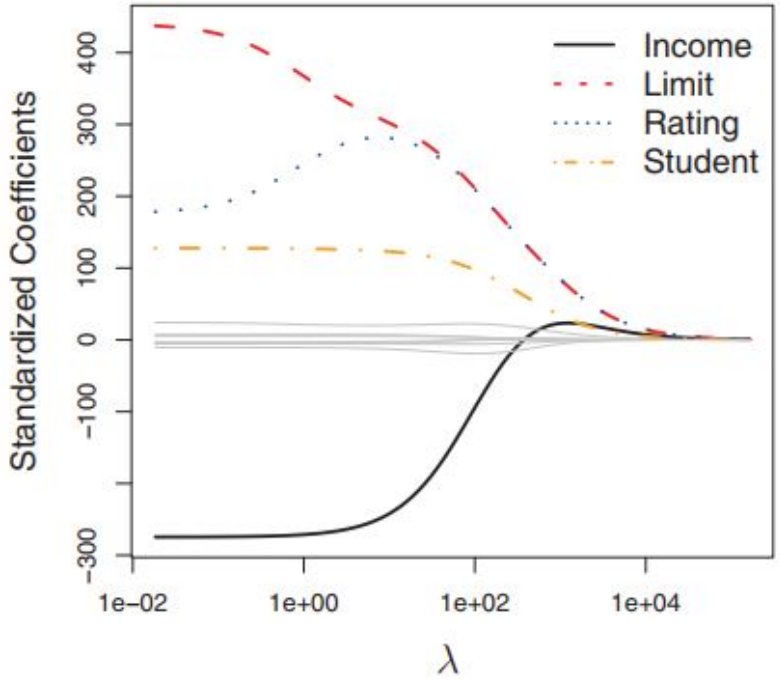
- Recordemos la función que se minimiza en la estimación de mínimos cuadrados:

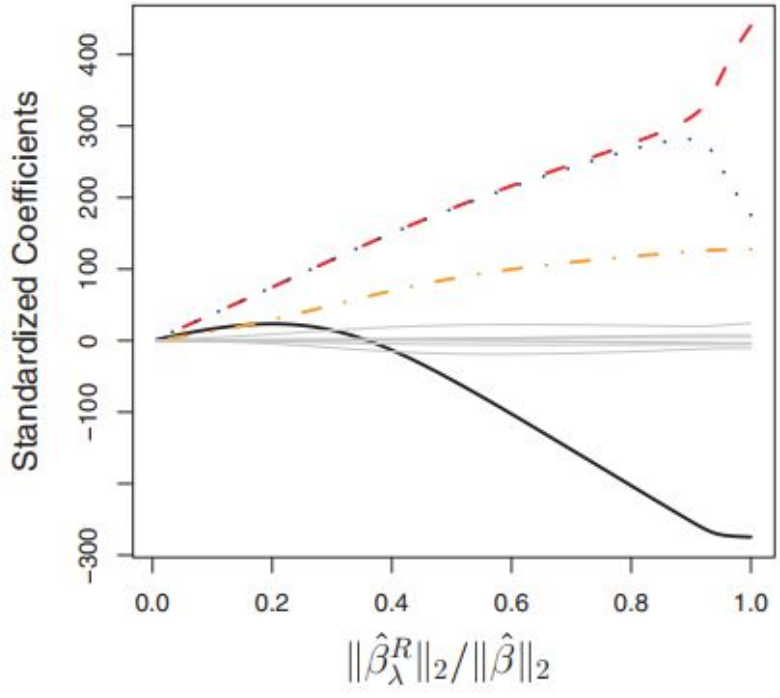
$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

- La función que se minimiza en Regresión Ridge es:

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

La diferencia es que agregamos un término nuevo. En este término, un **hiperparámetro lambda** penaliza el valor de los coeficientes al cuadrado.
Entonces, tengo que minimizar el cuadrado de los errores, intentando que ningún β_j^2 sea demasiado grande



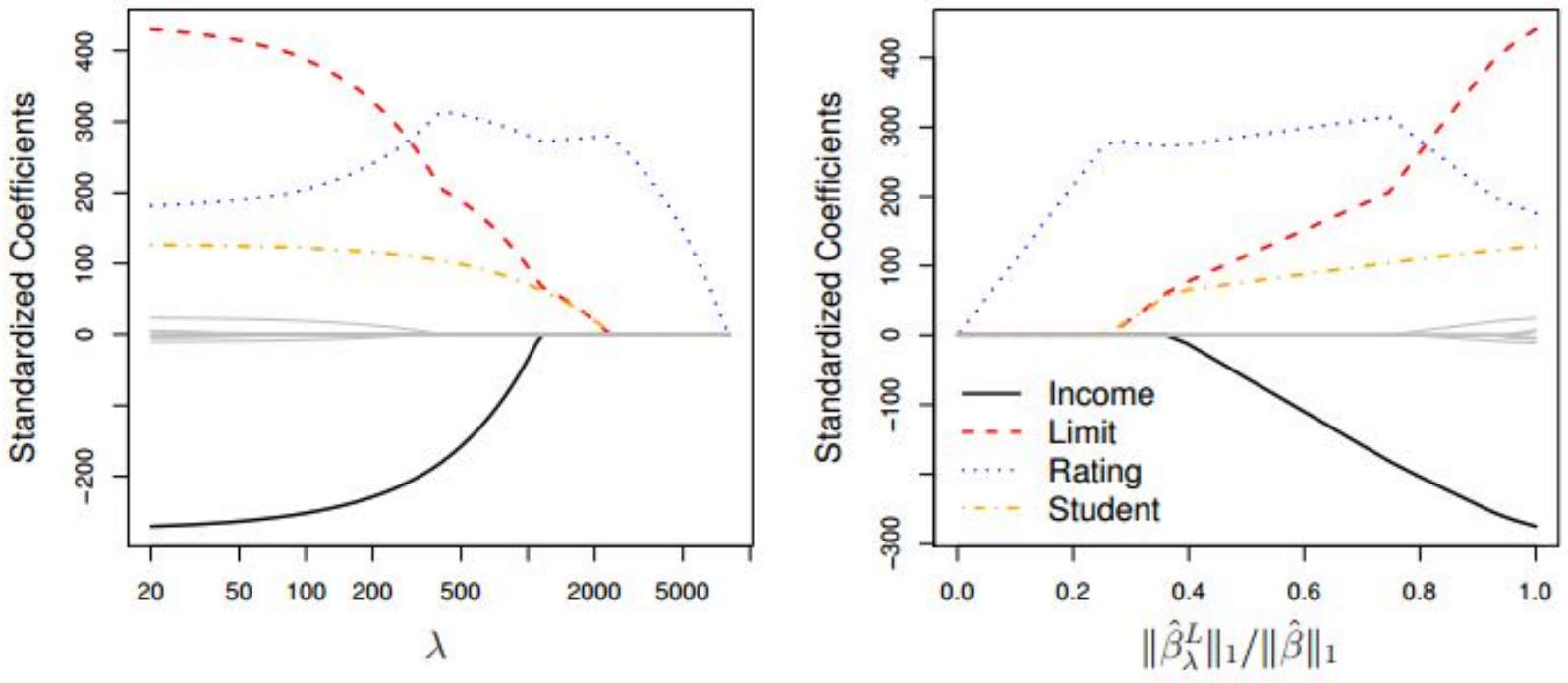


$$\|\beta\|_2 = \sqrt{\sum_{j=1}^{p} \beta_j{}^2}.$$

- La regresión Ridge tiene una clara desventaja: incluye todos los predictores p en el modelo final, a diferencia de aquellos modelos que eligen un conjunto de variables.
- La regresión Lasso es un alternativa relativamente nueva a Ridge, que corrige esta desventaja. Los coeficientes $\hat{\beta}_\lambda^L$, minimizan el número de variables

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

- Lasso utiliza penaliza con l1, y no con l2. La norma de l1 de un un vector de coeficientes β está dada por $\|\beta\|_1=\sum|\beta_j|.$


Standardized Coefficients
400
300
200
100
0
−200
20
50
100
200
500
2000
5000
λ
Standardized Coefficients
400
300
200
100
0
−100
−300
0.0
0.2
0.4
0.6
0.8
1.0
$\|\hat{\beta}_\lambda^L\|_1/\|\hat{\beta}\|_1$
Income
Limit
Rating
Student

NORMALIZACIÓN

¿Por qué normalizar?

- Manejo de cantidades en **diferentes unidades o escalas**
- Muchos **algoritmos** de machine learning toman la normalización como **requerimiento**
- Existen distintas razones por las cuales un algoritmo de ML requiere estandarizar los datos.

Muchos algoritmos de ML se basan en el cálculo de medidas de distancia que se calculan entre todos los puntos en base a distintos features.

La medida de distancia que viene implementada por default, es la distancia euclídea que requiere matemáticamente que todos los features sean numéricos.

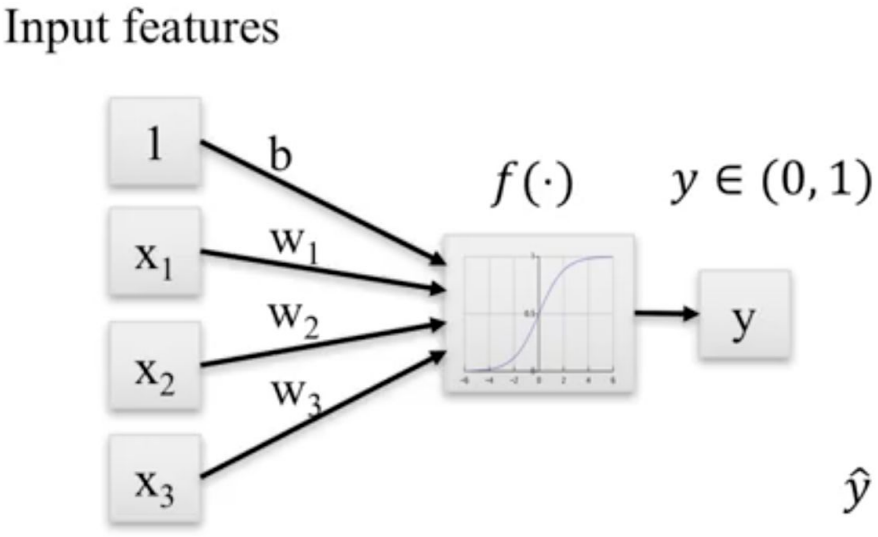
Además, para no favorecer a ningún feature en particular a la hora de explicar la distancia, tenemos que estandarizar y deshacernos de las unidades.

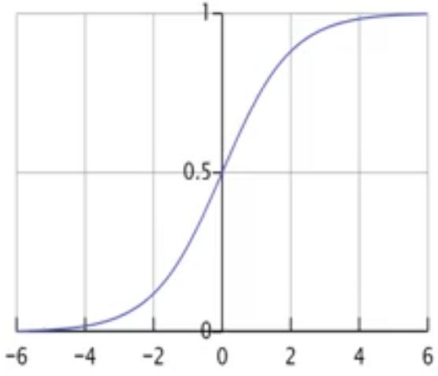
¿Cómo normalizar?

Existen algunas formas típicas de normalizar:

- La ***estandarización***: $x_{norm} = (x - \mu) / \sigma$
- La normalización ***min-max:*** $x_{norm} = (x - min) / (max - min)$

Modelos de Clasificación: Regresión Logística





$$\hat{y} = \text{logistic}(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)$$

$$= \frac{1}{1 + \exp\left[-(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)\right]}$$

Es posible aplicar **regularización** a la regresión logística, por ejemplo aplicando regularización con norma L2:

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left[\left(y^{(i)}\log\left(\hat{y}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right] + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Hay que tener en cuenta que en Scikit-Learn, la regresión logística aplica regularización por default. El parámetro *lambda* se reemplaza por su inversa, el parámetro *C*. A mayor valor de C, menos penalización. El valor por default de Scikit-Learn es C=1.

Modelos de Clasificación: explorando métricas de evaluación

Accuracy

- Proporción de **clases correctamente predichas**.
- Notar que esta métrica no "alerta" si no tengo casos positivos bien predichos.

	Predice que está saludable (y=0)	Predice que está enfermo (y=1)
Está saludable (y=0)	46 **(TN)**	85 **(FP)**
Está enfermo (y=1)	168 **(FN)**	31 **(TP)**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Los datasets del mundo real suelen presentar **clases desbalanceadas**.

- ¿Qué pasa en un **dataset desbalanceado** con el **accuracy**?
  El accuracy suele ser muy alto, pero esto generalmente es porque el modelo responde la **"solución trivial"** o algo muy cercano.

- Algunos casos comunes de datasets desbalanceados son:
  - Estudios sobre fraude
  - Diagnóstico de enfermedades

Recall (Sensibilidad o True Positive Rate):

- Proporción de **positivos correctamente predichos**.
- Si su valor es bajo, es porque hay presencia de falsos negativos. Por eso, esta medida es **sensible a los FN**.

	Predice que está saludable (y=0)	Predice que está enfermo (y=1)
Está saludable (y=0)	46 **(TN)**	85 **(FP)**
Está enfermo (y=1)	168 **(FN)**	31 **(TP)**

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN}$$

Precisión:

- Cantidad de **verdaderos positivos sobre el total de predicciones positivas**.
- Si su valor es bajo, es porque hay presencia de falsos positivos. Por eso, esta medida es **sensible a los FP**.

	Predice que está saludable (y=0)	Predice que está enfermo (y=1)
Está saludable (y=0)	46 **(TN)**	85 **(FP)**
Está enfermo (y=1)	168 **(FN)**	31 **(TP)**

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

F1:

- Es la **media armónica de los scores de recall y precisión**.
- Como regla general, **cuanto mayor es esta métrica, mejor es el modelo**.

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * (precision * recall)}{precision + recall}$$

Fβ:

- Es la **media armónica ponderada** de los scores de recall y precisión**.**
- Con el parámetro β se puede regular la **importancia relativa de cada término**.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

- Es una forma de representar los métricas en un modelo de clasificación binaria:
  - True Positive Ratio = **TP / (TP + FN)**
  - False Positive Ratio = **FP / (FP + TN)**
- Mundo ideal: mi modelo debería tener una Sensibilidad (TPR) de 100% y una FPR de 0%.

- Teniendo en cuenta esto…
    - ¿Cuál de los siguientes modelos es mejor?
- Una buena medida es el **área debajo de la curva ROC**:
    - Cuanto mayor sea el área… mejor será el modelo. ¿Por qué?