

**DigitalHouse** >  
Coding School

# DATA SCIENCE

MÓDULO 3

Regularización

1

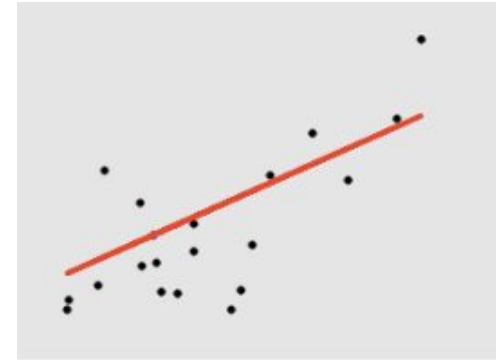
**Entender la regularización como técnica para evitar el sobreajuste**

2

**Aplicar regularización usando scikit-learn**

3

**Aprender a hacer validación cruzada para ajustar los hiper-parámetros de regularización**



# Sesgo - Varianza



Dado el modelo:

$$Y = f(X) + \epsilon.$$

Donde epsilon es un término aleatorio con distribución:

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

Podemos obtener una estimación de  $f$  para hacer predicciones sobre  $Y$ :

$$\hat{Y} = \hat{f}(X),$$

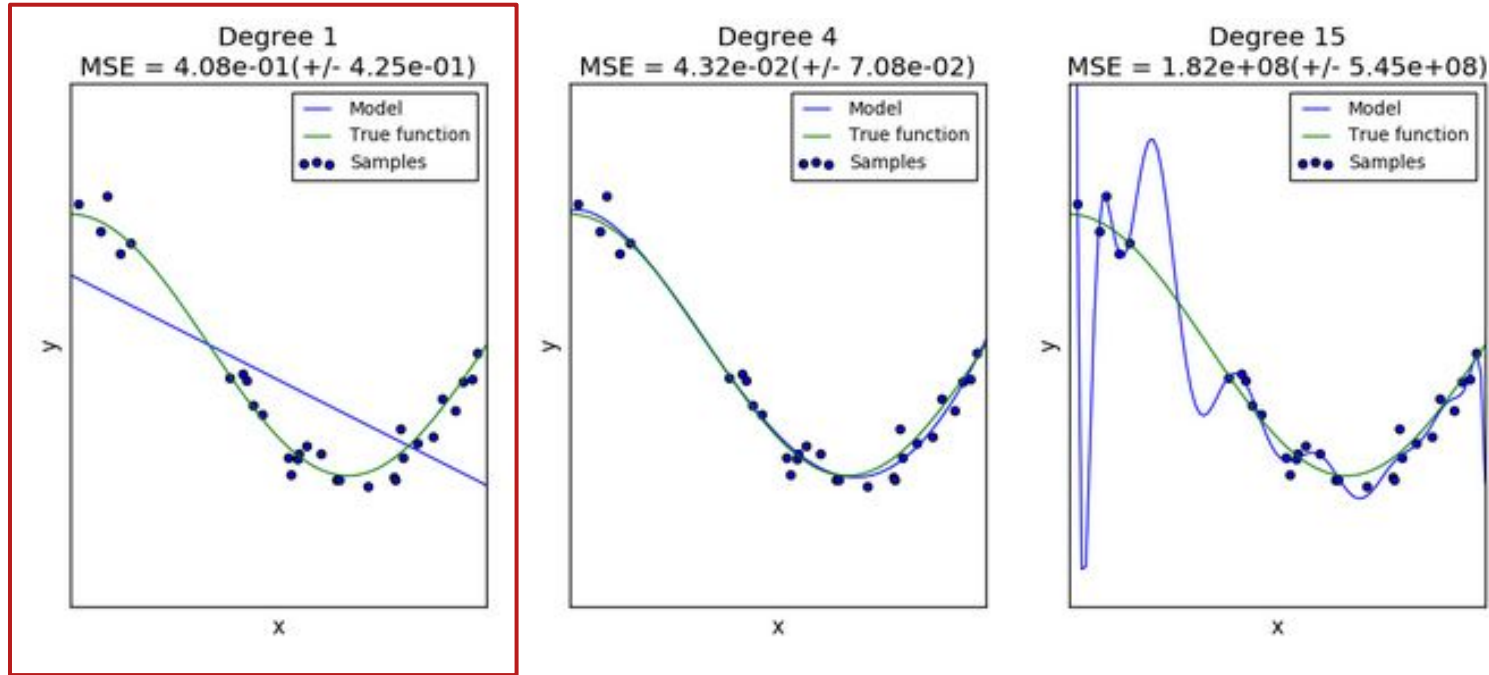
La esperanza del error de predicción al cuadrado será la siguiente:

$$Err(x) = E \left[ (Y - \hat{f}(x))^2 \right]$$

Podemos descomponer la esperanza del error al cuadrado de la siguiente manera:

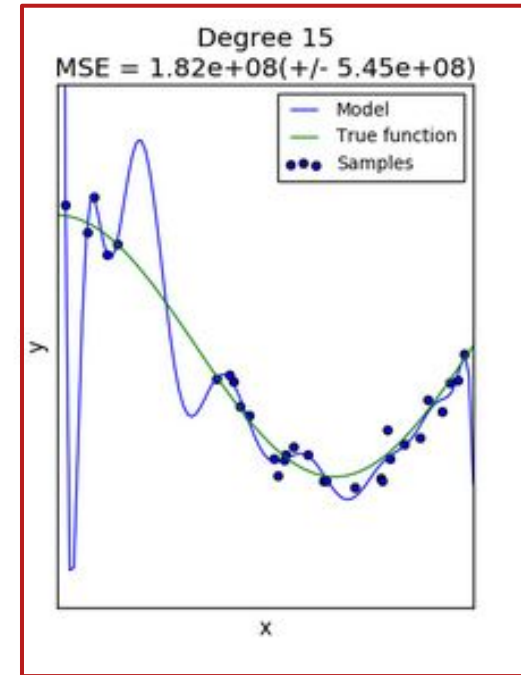
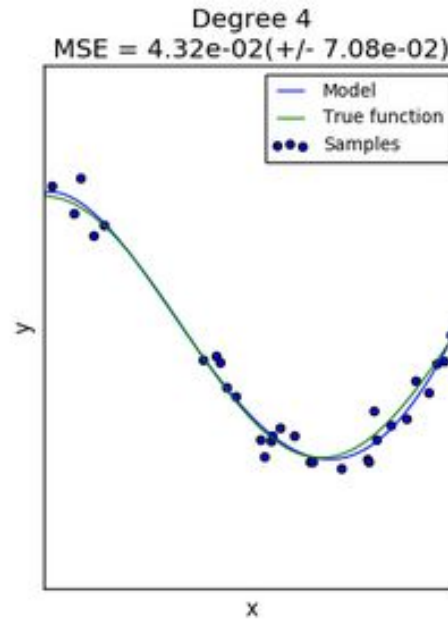
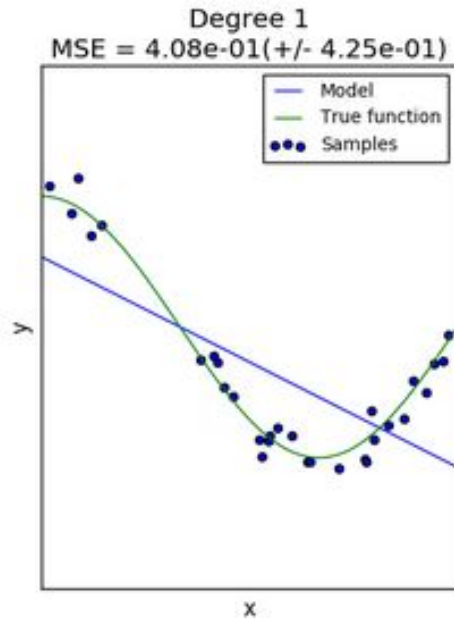
$$Err(x) = \left( E[\hat{f}(x)] - f(x) \right)^2 + E \left[ \left( \hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

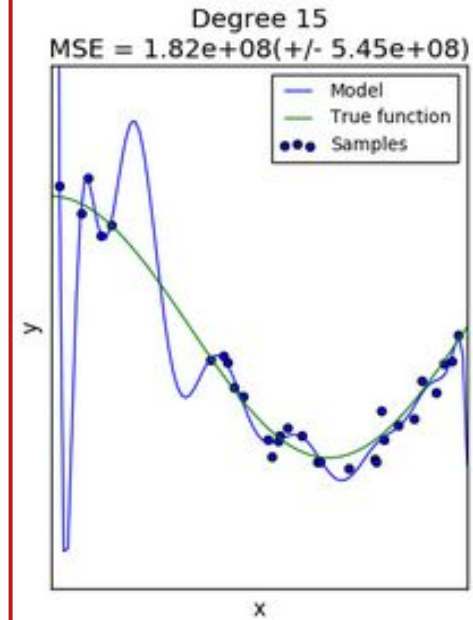
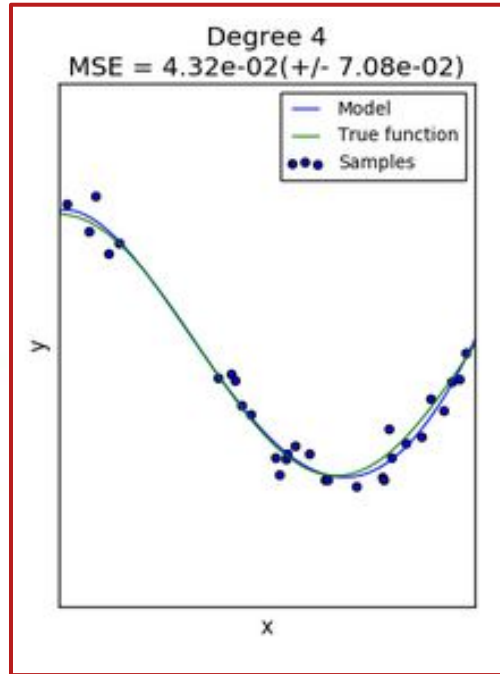
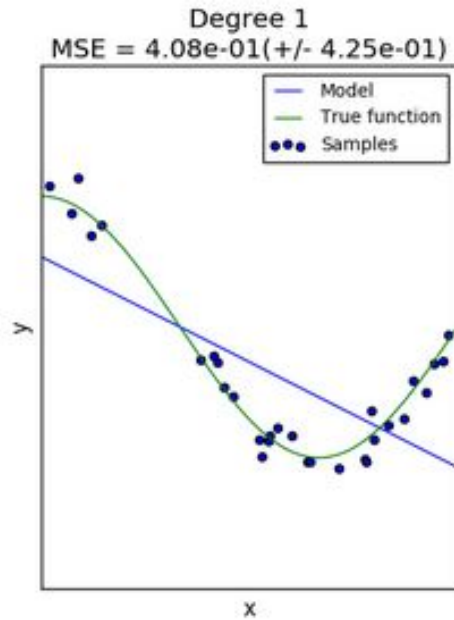


Si el modelo es **demasiado simple** (tiene pocos grados de libertad), entonces no importa cuán grande sea la muestra: tenemos **sesgo o error sistemático**:

$$E(\hat{f}(x)) \neq E(f(x))$$

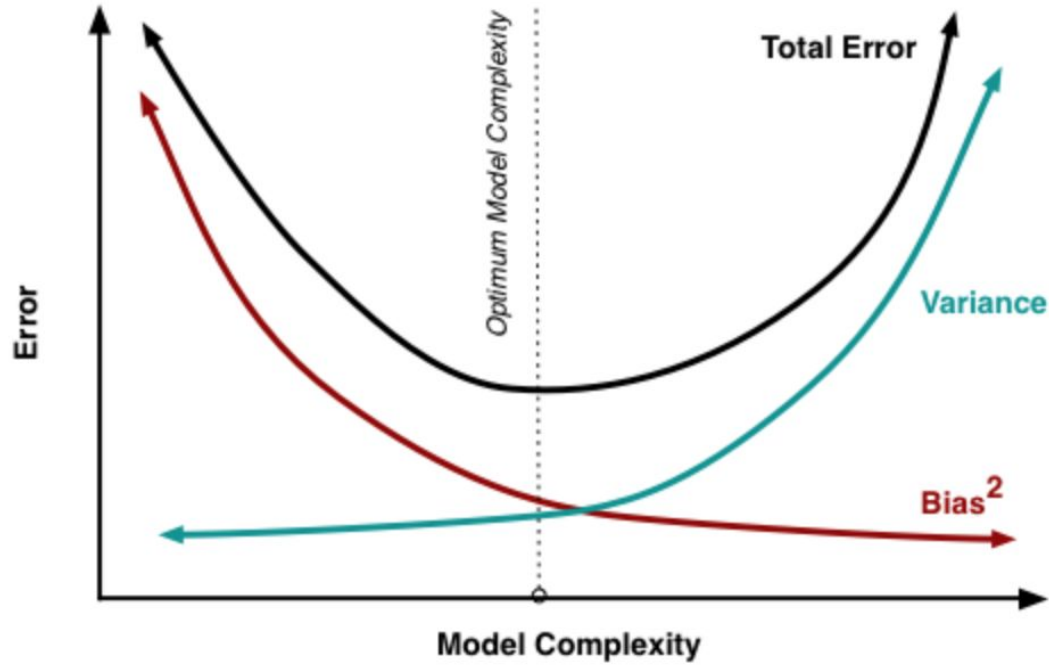


Si el modelo es **demasiado complejo** (ie. tiene demasiados grados de libertad), entonces el estimador puede ajustarse regularidades espurias de la muestra generando un **sobre-ajuste**.



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





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

# 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:

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

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.

Vamos a ver a continuación dos técnicas de regularización:

- **Regresión Ridge**
- **Regresión Lasso.**

Vamos a ver a continuación dos técnicas de regularización:

- **Regresión Ridge**
- **Regresión Lasso.**

Estas técnicas proponen cambiar ligeramente el problema de optimización de mínimos cuadrados, para intentar **“achicar” (*shrink*) el valor absoluto de los estimadores de los Betas.**

Vamos a ver a continuación dos técnicas de regularización:

- **Regresión Ridge**
- **Regresión Lasso.**

Estas técnicas proponen cambiar ligeramente el problema de optimización de mínimos cuadrados, para intentar **“achicar” (*shrink*) el valor absoluto de los estimadores de los Betas.**

**¿Por qué esto mejoraría la estimación?** Vamos a ver de qué forma este método introduce un sesgo pero reduce la varianza.



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

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

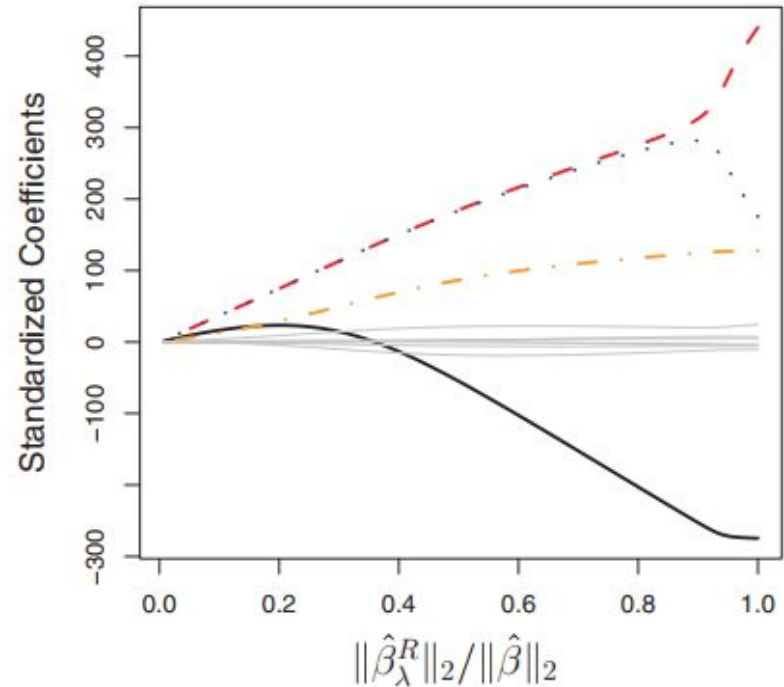
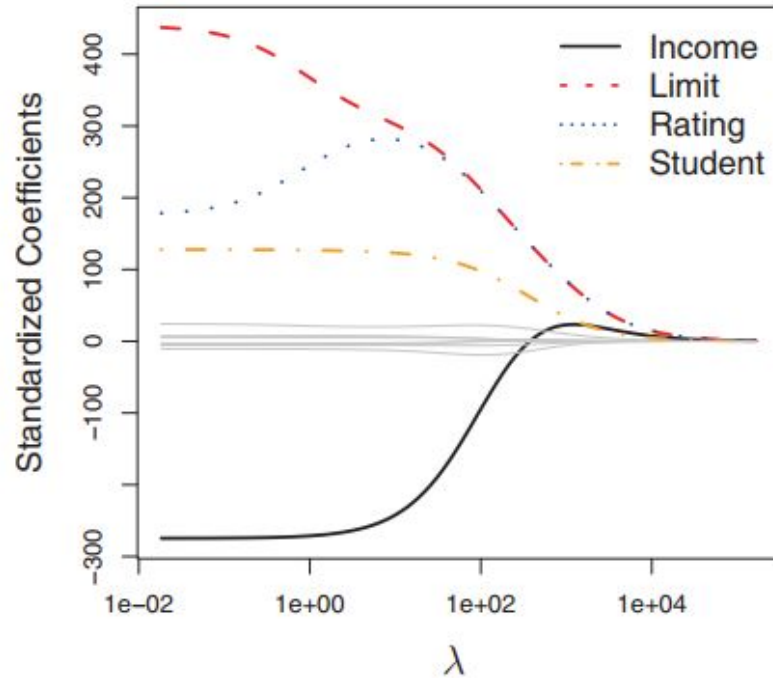
- La función que se minimiza en Regresión Ridge es:

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

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  $\beta_j^2$  sea demasiado grande

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

- Al igual que MCO, buscamos achicar el RSS.
- Sin embargo, existe un **término de penalización**, que es menor cuando los Betas se acercan a cero, por lo tanto tiene el efecto de achicar los mismos hacia cero (tanto si son negativos como positivos)
- El **hiperparámetro lambda**, maneja la ponderación de cada término.
- ¿Cuál es el mejor valor para lambda? ¿Cómo elegíamos el valor óptimo de un hiperparámetro?  
Como siempre, lo hacemos a través de **CROSS VALIDATION**



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

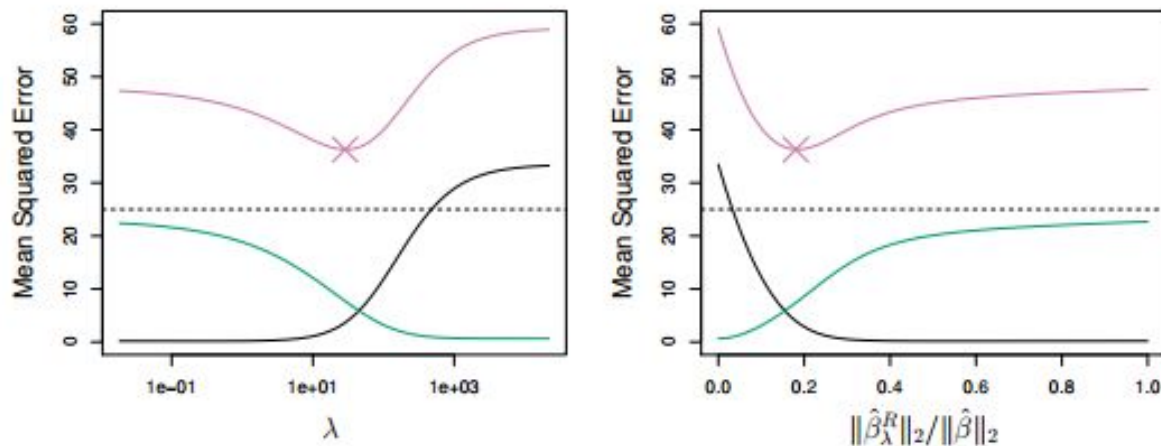
- En Regresión **Ridge** tanto la estimación de los coeficientes como la predicción son **sensibles a la escala**.
- Recordemos el problema de optimización que resuelve Ridge:

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

- Si una variable se encuentra en una escala que le da un valor absoluto mayor, esto **va a afectar el cálculo de la suma de cuadrados** del vector de coeficientes.
- Por esta razón **es importante estandarizar (dividir por el desvío estándar)** todos los regresores antes de ejecutar una regresión Ridge. Así ya no están en unidades físicas sino en unidades de su propio desvío estándar.

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2}}$$

## El tradeoff entre Bias-Variance



Aquí se pueden ver  $n=50$  simulaciones,  $p=45$  predictores, todos con coeficientes no nulos.

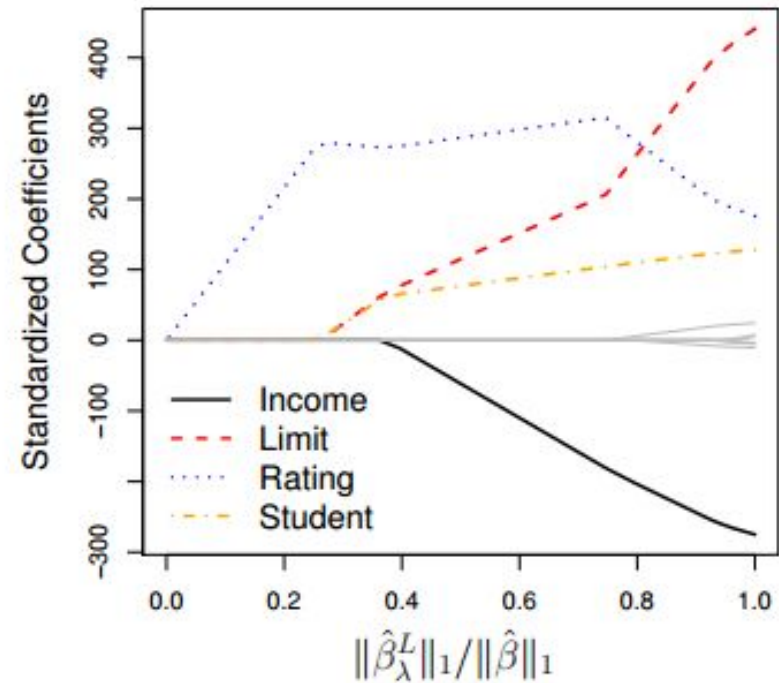
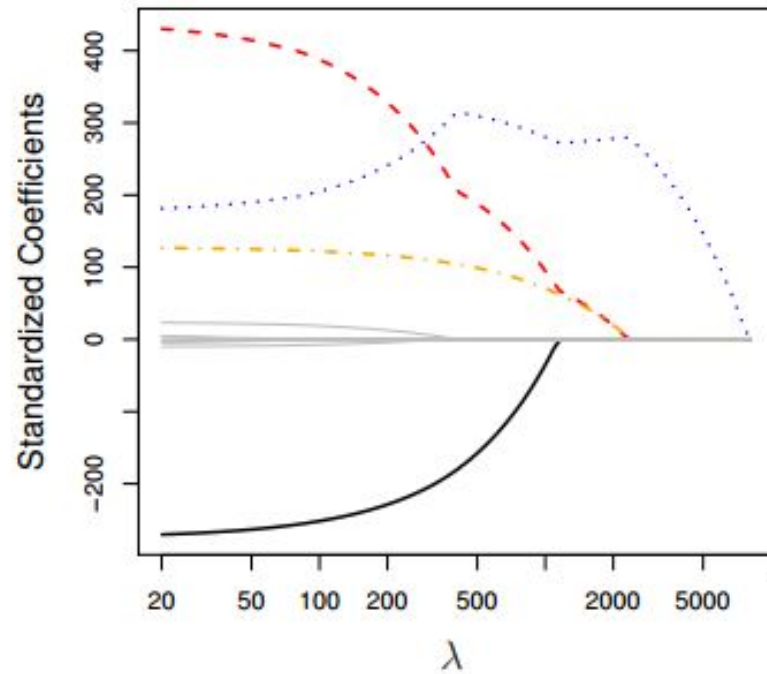
El gráfico expresa el sesgo cuadrado (negro), varianza (verde) y el MSE del test (violeta), para una regresión ridge en los datos simulados, como una función de  $\lambda$  y de  $\|\hat{\beta}_\lambda^R\|_2 / \|\hat{\beta}\|_2$

- 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 una alternativa relativamente nueva a Ridge, que corrige esta desventaja. Los coeficientes  $\hat{\beta}_\lambda^L$ , minimizan el número de variables

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

- Lasso utiliza penaliza con  $l_1$ , y no con  $l_2$ . La norma de  $l_1$  de un vector de coeficientes  $\beta$  está dada por  $\|\beta\|_1 = \sum |\beta_j|$ .

- Como en la regresión ridge, **Lasso “achica” los coeficiente** estimados hacia el zero.
- Sin embargo, en el caso de Lasso, el  **$\lambda$  fuerza los coeficientes a valer exactamente cero**, en el caso de que  $\lambda$  sea lo suficientemente grande.
- Por lo tanto, el **Lasso hace selección de variables**
- Entonces, decimos que Lasso **genera modelos dispersos**, es decir, modelos con una selección de variables
- Al igual que en Ridge, la elección de un buen valor  $\lambda$  es crítico en Lasso; nuevamente, **cross-validation** es el método para su elección



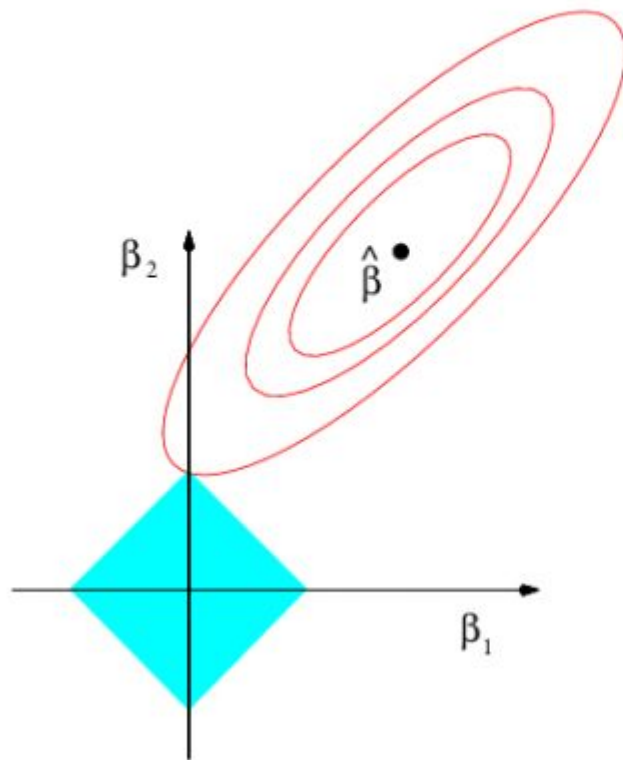


- ¿Por qué Lasso, a diferencia de Ridge, resulta en coeficientes estimados exactamente igual a cero?
- Uno puede mostrar que la estimación de coeficientes de las regresión Lasso y Ridge resuelve estos problemas, respectivamente.

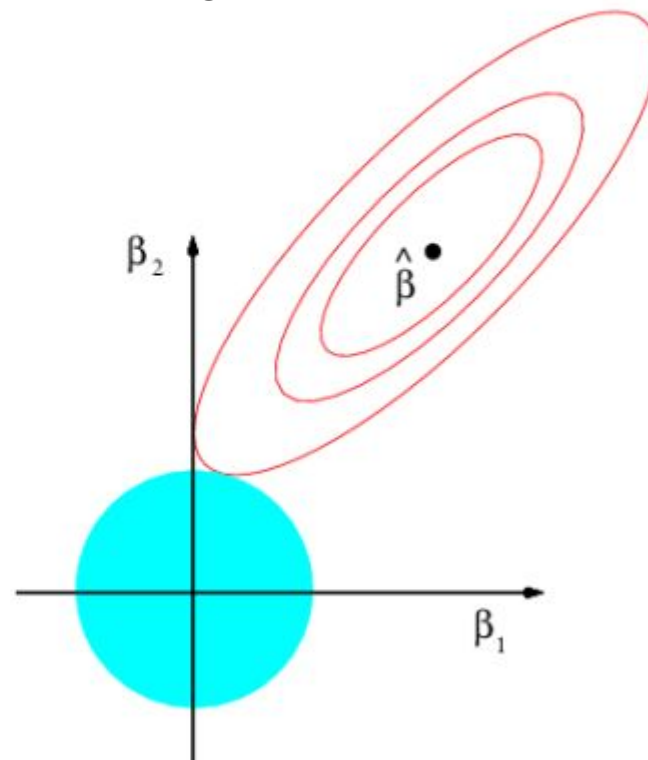
$$\underset{\beta}{\text{minimize}} \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^p |\beta_j| \leq s$$

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^p \beta_j^2 \leq s,$$

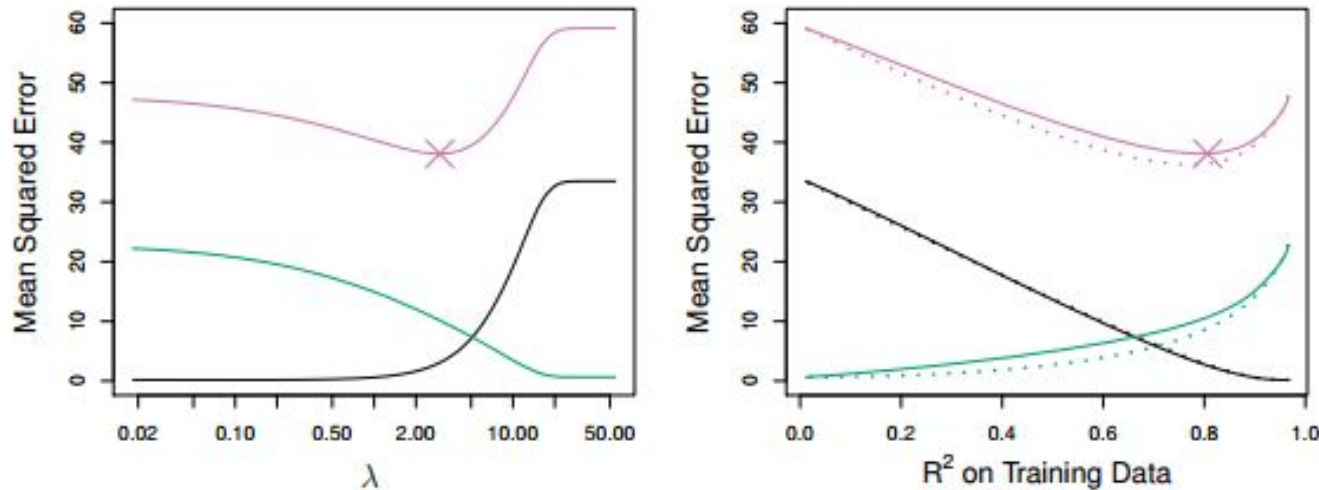
Lasso



Ridge



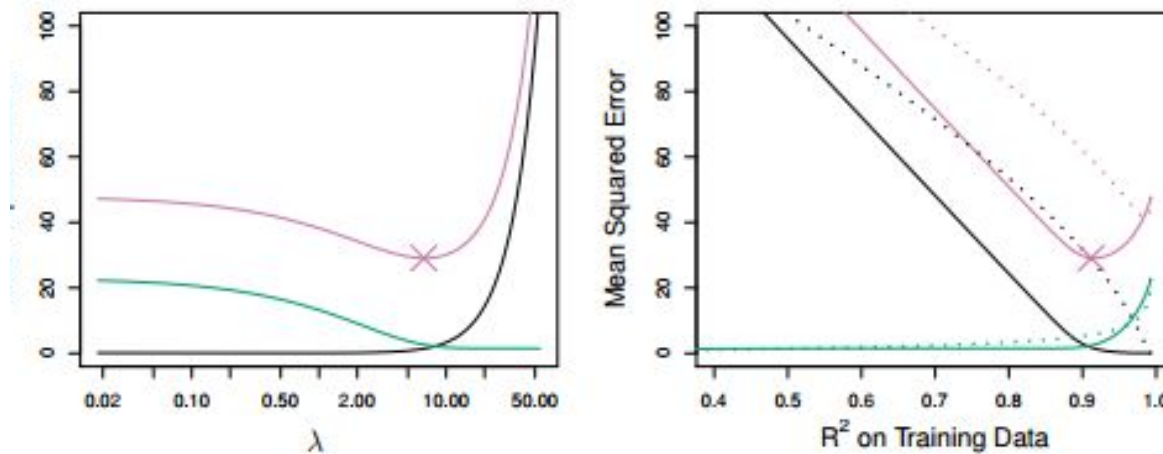
Ejemplo simulado, haciendo que todos los coeficientes fueran diferentes a cero. En este caso, los dos modelos tienden a performar prácticamente igual. Ridge tiene una menor varianza y por eso parece mejorar respecto a Lasso



**Izquierda:** Sesgo cuadrado (negro), la varianza (verde) y el MSE del test (violeta) para Lasso en set simulado.

**Derecha:** Comparación del Sesgo cuadrado, la varianza y el MSE del test, entre Lasso (llena) y Ridge (punteada).

Estos datos se generaron haciendo que solamente dos coeficientes fueran diferentes a cero. De esta forma, vemos cómo Lasso mejora claramente la performance con respecto a Ridge, tanto en lo referido a variancia como a MSE.



**Izquierda:** Sesgo cuadrado (negro), la variancia (verde) y el MSE del test (violeta) para Lasso en set simulado.

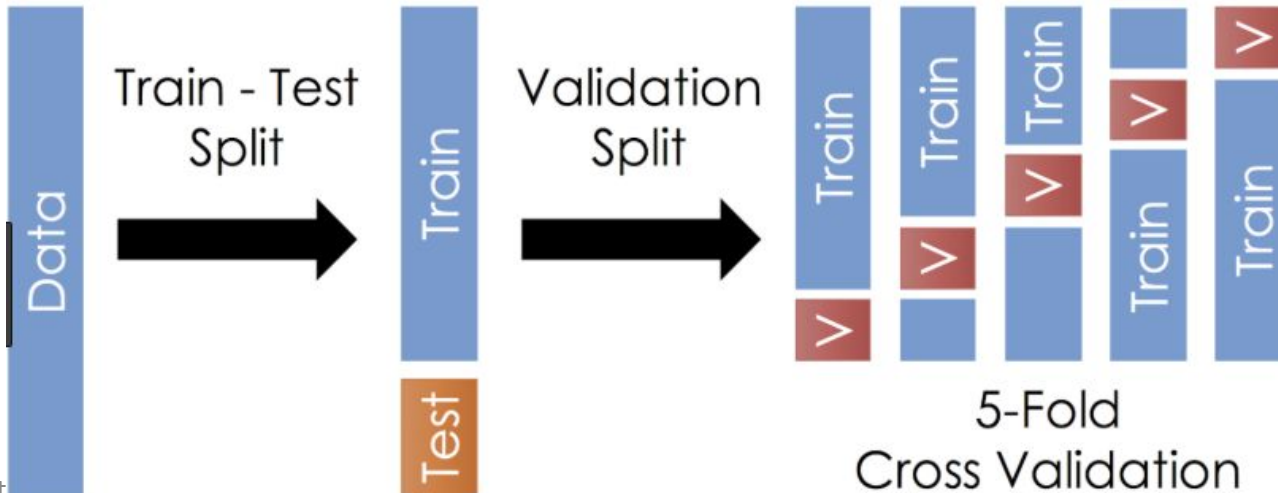
**Derecha:** Comparación del Sesgo cuadrado, la variancia y el MSE del test, entre Lasso (llena) y Ridge (punteada).

- Se necesita un método para poder **ajustar el hiperparámetro  $\lambda$**
- **Cross-validation** es una manera simple de atacar este problema. Se elige un rango de valores que puede tomar el hiperparámetro, y luego se computan los errores que devuelve cross-validation, para cada caso.
- Se elige el hiperparámetro asociado al menor error computado.
- Finalmente, “re-fiteamos” el modelo con el hiperparámetro elegido.

$$\lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2 = \lambda (\|\beta\|_1 + \alpha \|\beta\|_2^2)$$

- ElasticNet combina (linealmente) lo mejor de ambos mundos.
- El parámetro  $\lambda$  regula la complejidad del modelo. El parámetro  $\alpha$  regula la importancia relativa de Lasso vs. Ridge.
- Es posible obtener soluciones parsimoniosas y bien condicionadas.
- No free lunch!: ahora hay que calibrar dos hiperparámetros.

- ¿Cómo funciona?
  - Hacemos el split train/validación y test
  - Empezamos dividiendo el dataset de train/validación en k grupos (generalmente, 5 o 10 suele ser la medida convencional) del mismo tamaño.
  - En la primera iteración, el primer grupo generado pasa a ser un test test; el resto, pasa a ser el training set
    - Entrenamos un modelo sobre el training data
    - Hacemos las predicciones sobre el test set y calculamos el error sobre este test-set
  - Repetimos k veces, variando el test set en cada iteración.
  - Al final, promediamos los errores en cada una de las iteraciones



# Conclusión





- La **regularización nos ayuda a evitar el sobreajuste** limitando la complejidad del modelo
- Matemáticamente lo logra **penalizando la complejidad** dentro de la función de costo
- Modelos con regularización suelen tener **mayor poder de generalización**
- Para determinar el valor de los **hiper-parámetros** usados para regularizar, usamos **validación cruzada**