datarebel

Regresión Regularizada

Data Translator

Objetivos

- Deficiencias de una Regresión Lineal Ordinaria
- Regresión Ridge
- Regresión Lasso
- Cuándo usar cada una

Revisión: Regresión Lineal

Asumimos que el mundo está construido sobre una relación lineal. Bajo esa suposición, podemos modelar esa relación entre *features* y el *target* de la siguiente manera:

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

target
(aka, resultado)

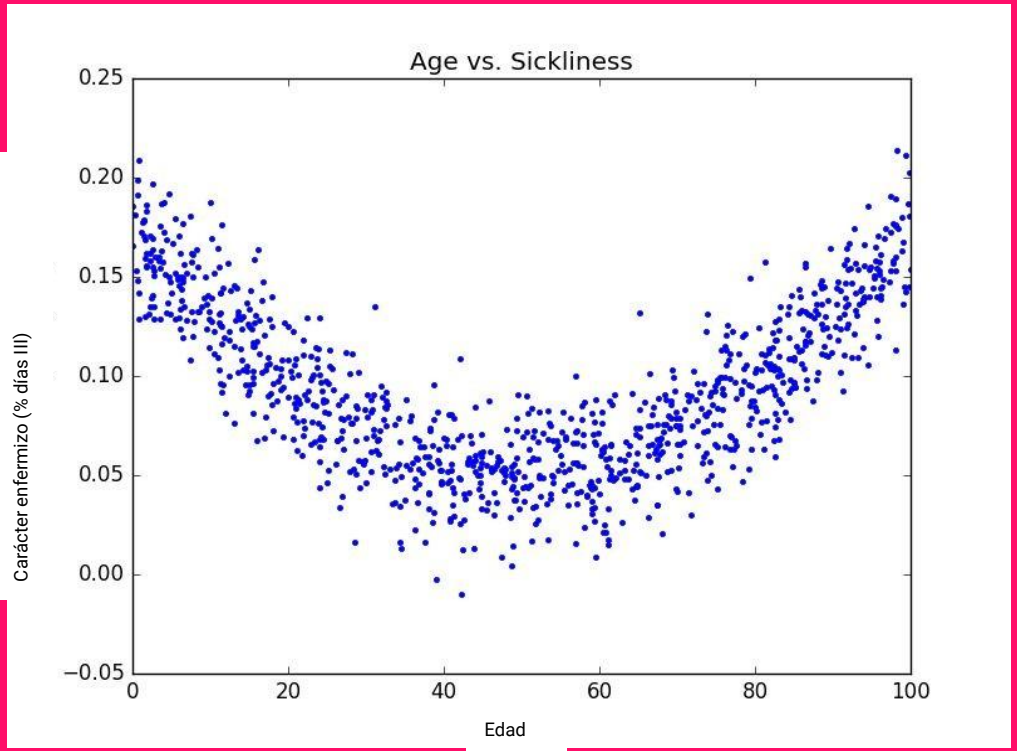
parámetros del modelo (aka coeficientes)

features
(aka predictores)

error de muestra

Revisión: Regresión Lineal

Podemos hacer una regresión lineal, no-lineal insertando features de interacción extra o features de orden superior.

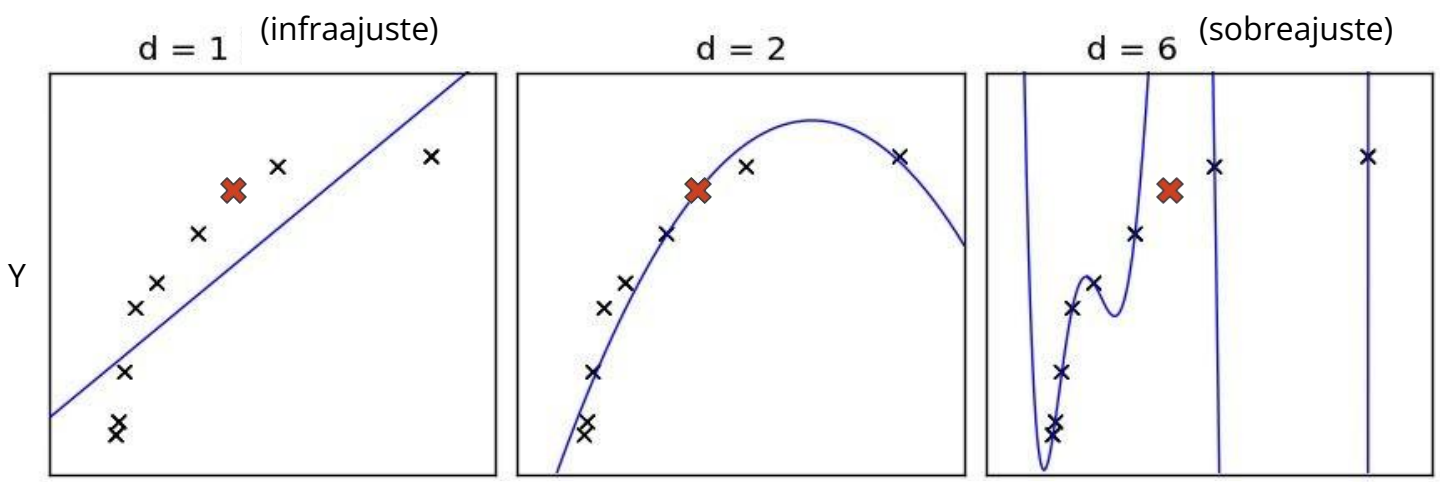


Ejemplo

$$Y = \beta_0 + \beta_1 * \text{edad}$$

$$Y = \beta_0 + \beta_1 * \text{edad} + \beta_2 * \text{edad}^2$$

Las “maravillas” del sobreajuste...



¿Qué está mal en el primer modelo?

¿Qué está mal en el segundo modelo?

¿Qué está mal en el tercer modelo?

Infrajuste y Sobreajuste

Infrajuste (Underfitting): El modelo no captura *del todo* la relación entre predictores y el target. El modelo aún *no ha aprendido* la indicación de los datos.

→ ¿Qué debemos hacer si el modelo infrajusta los datos? (asume utilizando reg. lin.)

Sobreajuste (Overfitting): El modelo ha tratado de capturar el error de muestra. El modelo ha aprendido la indicación de los datos *y* el ruido.

→ ¿Qué hacemos si el modelo sobreajusta los datos? (más complicado... ¿alguna idea?)

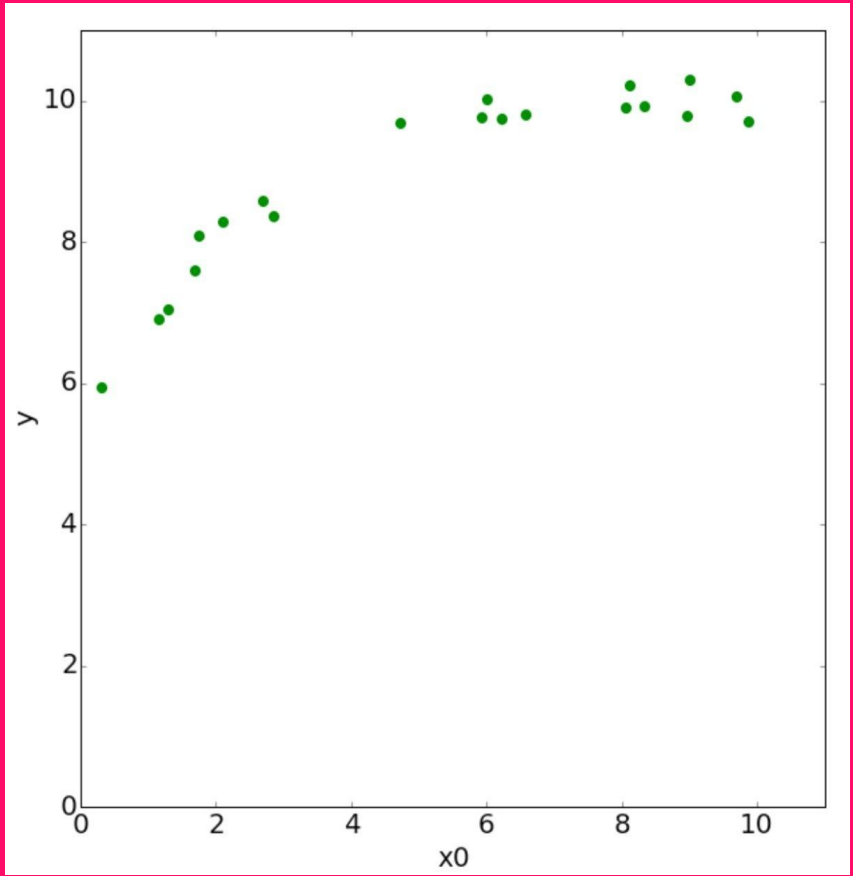
Ejemplo de Regresión Lineal

Datos: 20 ejemplos x 10 features

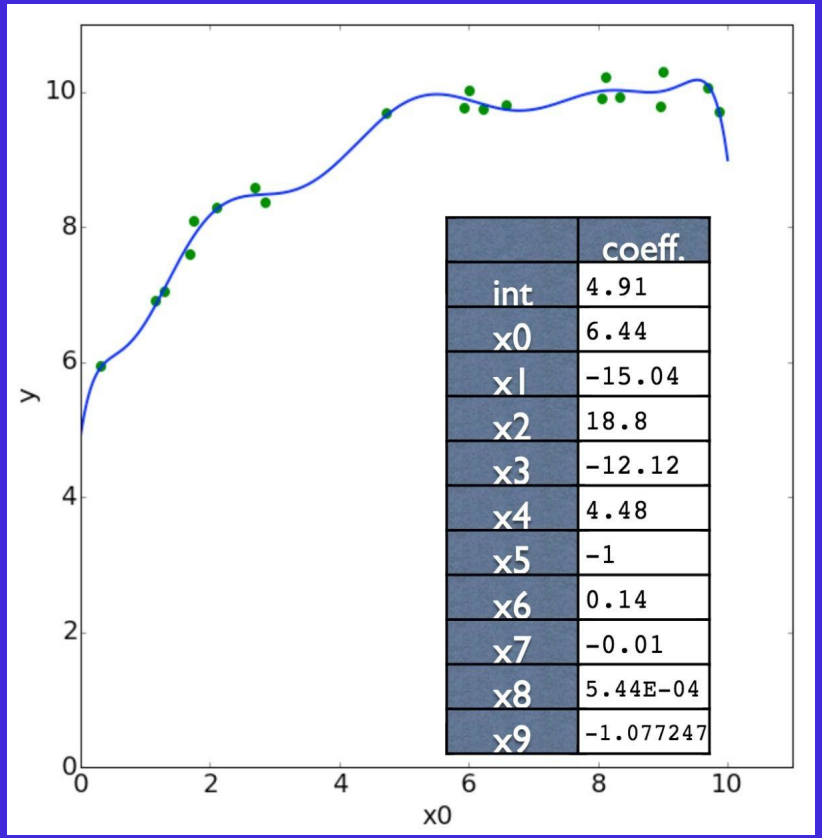
Predictor: *y*

y	x0	x1	x2	x3	...
9.92	8.33	69.39	578.0	4815.	...
8.58	2.69	7.26	19.54	52.64	...
8.07	1.75	3.06	5.35	9.36	...
8.29	2.11	4.46	9.41	19.86	...
...	...	...	...	...	...

Ejemplo Regresión Lineal (x0 vs y)

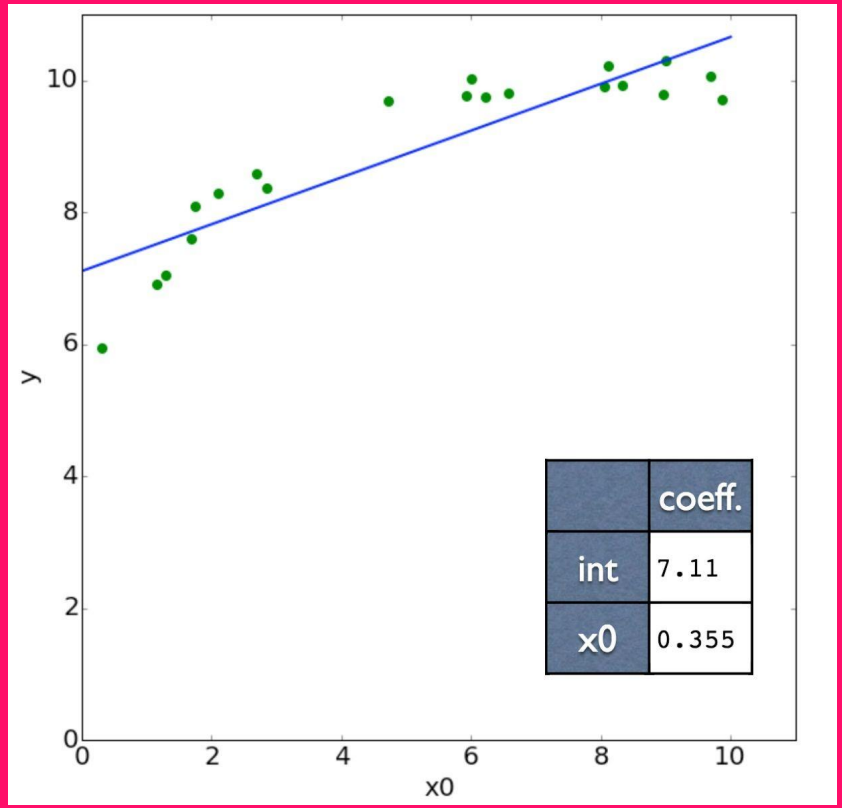


Ejemplo Regresión Lineal (x0 vs y, modelo sobre todos los features)



	coeff.
int	4.91
x0	6.44
x1	-15.04
x2	18.8
x3	-12.12
x4	4.48
x5	-1
x6	0.14
x7	-0.01
x8	5.44E-04
x9	-1.077247

Ejemplo Regresión Lineal (x0 vs y, modelo solamente sobre x0 features)



En altas dimensiones, los datos están (usualmente) dispersos.

datarebel

De nuevo, la **maldición de la dimensionalidad**.
(hablaremos de esto más adelante en el curso)

La regresión lineal tiene una alta variación (i.e tiende a sobreajustar) en altas dimensiones de datos.
Nos gustaría restringir (“normalizar” o “regularizar”) el modelo para que tenga menor variación.

Por ejemplo el grupo de datos 20 ejemplos x 10 features, cuando se ajusta a todos los features, la complejidad del modelo crece dramáticamente.
(Ten en mente , algunos grupos de datos tienen miles de features)

Regresión Lineal (otra revisión)

Se modela el mundo según:

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

Se estiman los parámetros del modelo minimizando:

$$\sum_{i=1}^{N} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij} \hat{\beta}_j)^2$$

Regresión Ridge (Regresión Lineal con Regularización Tikhonov (L2))

datarebel

Se modela el mundo según:

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

(se mantiene igual)

Se estiman los parámetros del modelo minimizando:

(el parámetro de “regularización”)

$$\sum_{i=1}^{N}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2 + \lambda \sum_{i=1}^{p} \hat{\beta}_i^2$$

(¡nuevo término!)

Regresión Ridge

$$\sum_{i=1}^{N}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2 + \lambda \sum_{i=1}^{p} \hat{\beta}_i^2$$

¿Qué sucede si se ajusta el valor lambda igual a cero?

¿Qué logra el nuevo término?

¿Qué efecto hace en un feature cuyo valor coeficiente correspondiente (beta) es cero?

Regresión Ridge

$$\sum_{i=1}^{N}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2 + \lambda \sum_{i=1}^{p} \hat{\beta}_i^2$$

Nota que no se penaliza B_0.

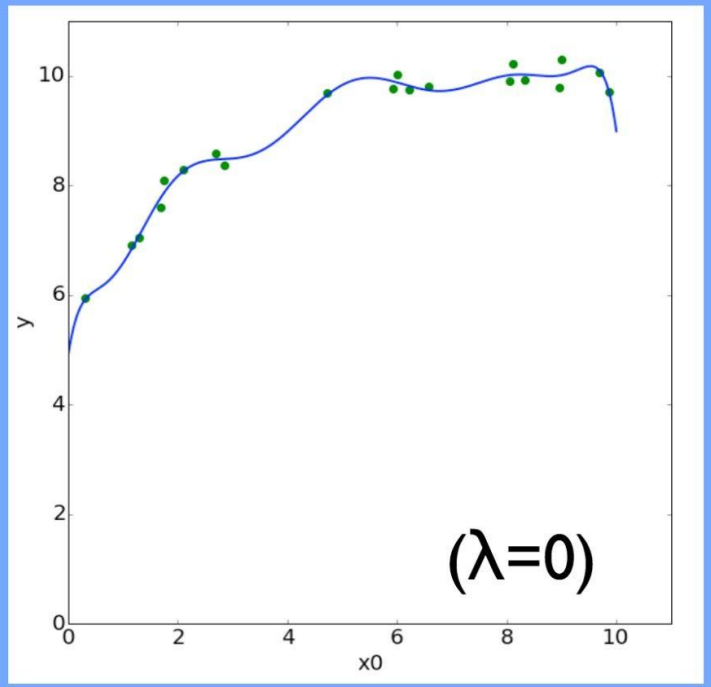
Cambiar lambda cambia la cantidad en que los coeficientes amplios son penalizados.

Incrementar la lambda incrementa los sesgos del modelo y decrece su variación. ← esto es bueno!

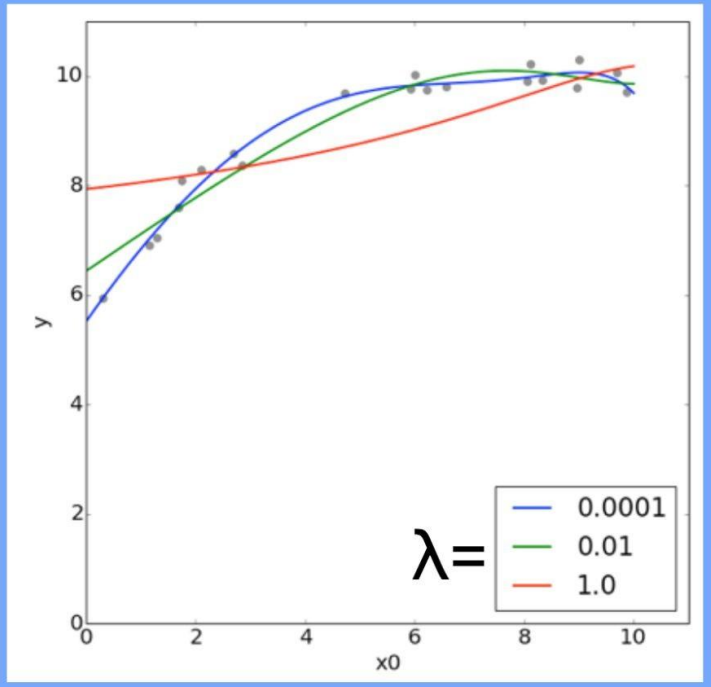
Regresión Ridge

datarebel

Regresión Lineal



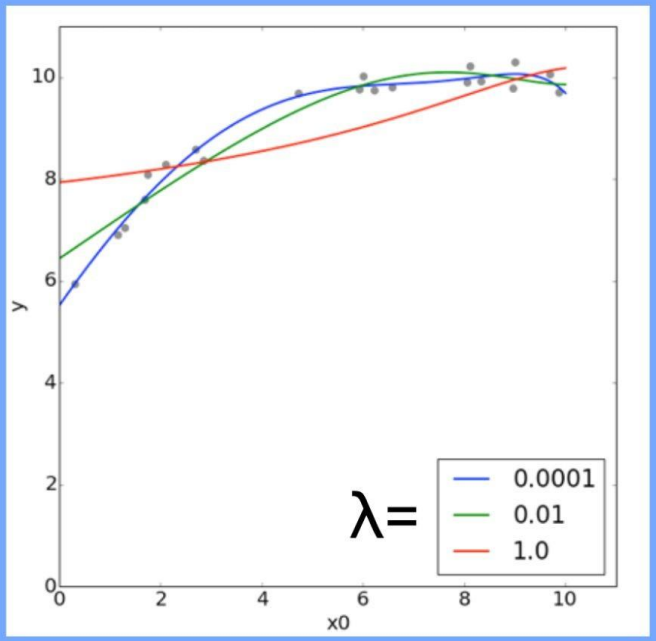
Regresión Ridge



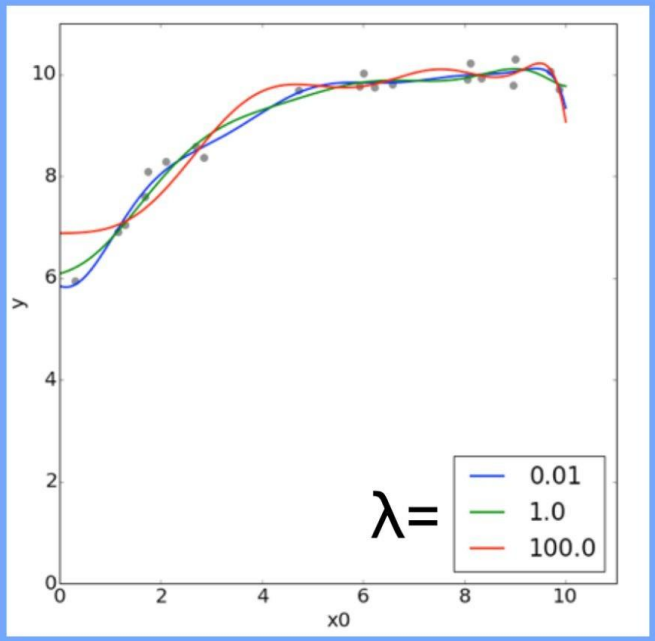
Regresión Ridge

datarebel

Datos Normalizados



Datos No-Normalizados



Valor único de λ asume que los features están en la misma escala

Regresión LASSO (Regresión Lineal con Regularización LASSO (L1))

datarebel

Se modela el mundo según:

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

(se mantiene igual)

Se estiman los parámetros del modelo minimizando:

$$\sum_{i=1}^{N} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij} \hat{\beta}_j)^2 + \lambda \sum_{i=1}^{p} |\hat{\beta}_i|$$

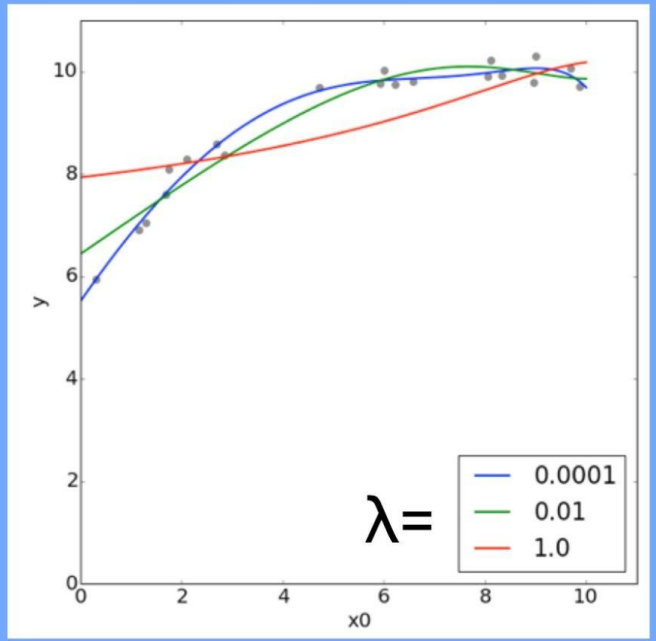
(el parámetro de “regularización”)

(valor absoluto en vez de al cuadrado)

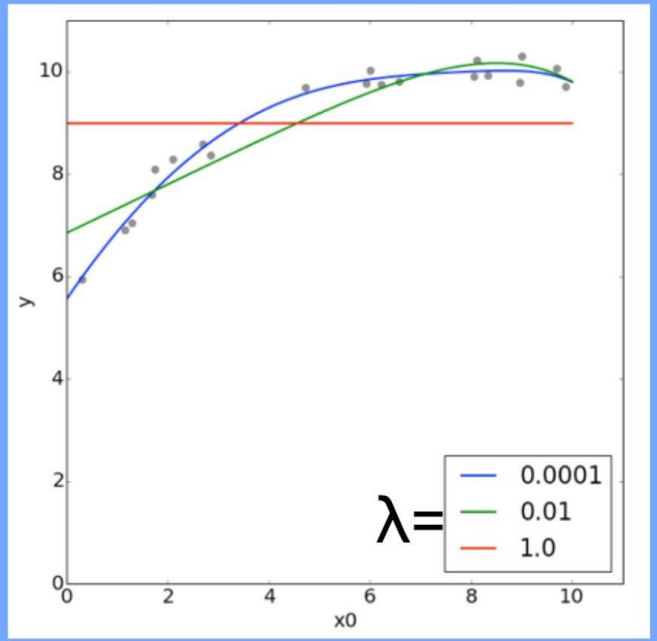
Regresión LASSO

datarebel

Regresión Ridge

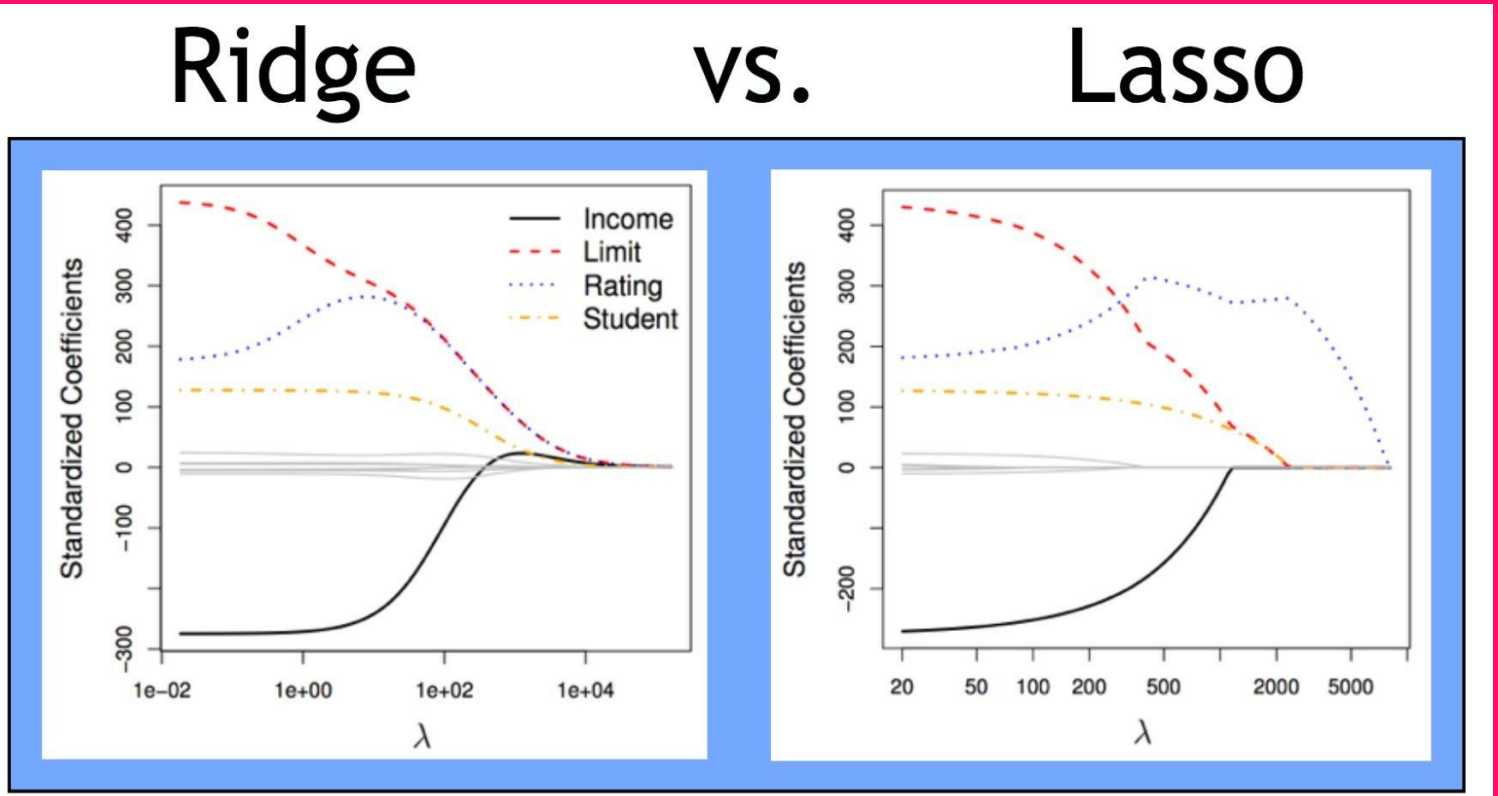


Regresión Lasso



Ridge vs LASSO

datarebel

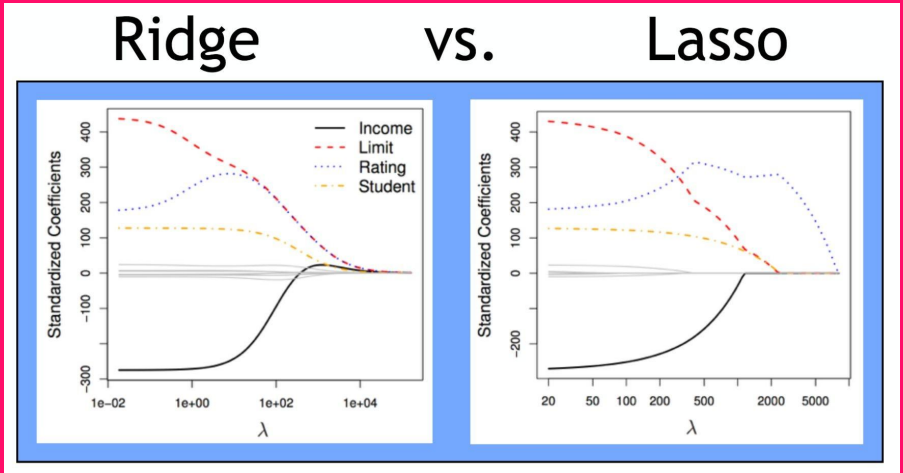


Ridge vs LASSO

- Ridge fuerza a que los parámetros sean pequeños + Ridge es computacionalmente más fácil porque es diferenciable.
- Lasso tiende a agrupar los coeficientes exactamente iguales a cero.
  - Esto es útil como un mecanismo “selección de feature automático”
  - lleva a modelos dispersos
  - sirve un propósito similar a una selección de features gradual.

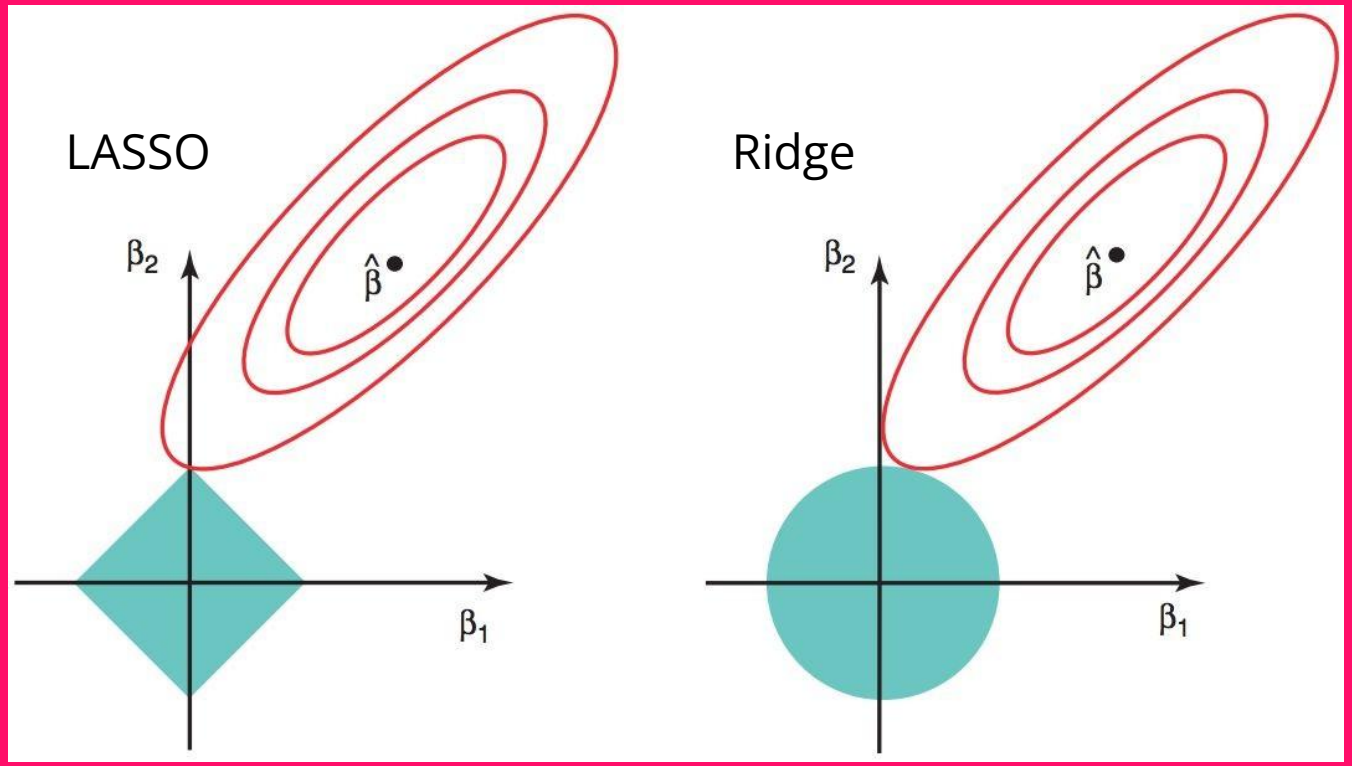
Cuál es mejor depende de tu grupo de datos

Modelos dispersos se beneficiarían de Lasso; modelos densos se beneficiarían de Ridge

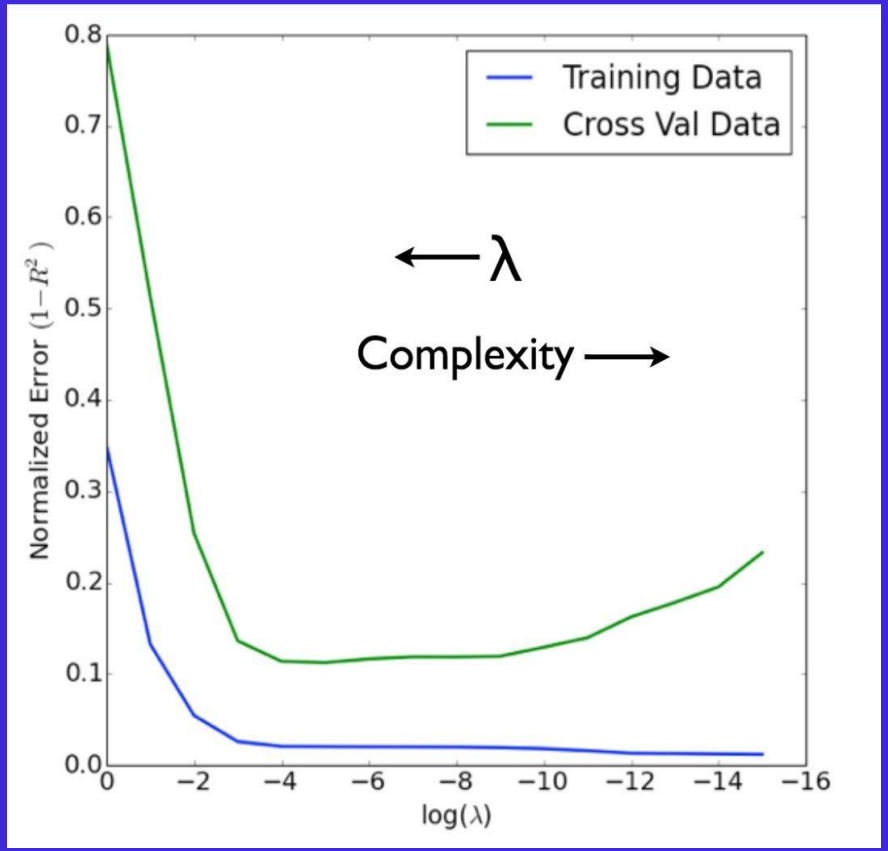


Ridge vs LASSO

datarebel



Escoger lambda via Validación Cruzada



scikit-learn

datarebel

Clases:

- sklearn.linear_model.**LinearRegression**(...)
- sklearn.linear_model.**Ridge**(alpha=my_alpha, ...)
- sklearn.linear_model.**Lasso**(alpha=my_alpha, ...)

Todas tienen estos métodos:

- fit(X, y)
- predict(X)
- score(X, y)

scikit-learn

datarebel

Clases::

- sklearn.linear_model.**LinearRegression**(...)
- sklearn.linear_model.**Ridge**(alpha=my_alpha, ...)
- sklearn.linear_model.**Lasso**(alpha=my_alpha, ...)
- sklearn.linear_model.**ElasticNet**(alpha=my_alpha, l1_ratio = !!!!, ...)
  Wow!

(En sklearn alpha = lambda) Todos tienen estos métodos:

- fit(X, y)
- predict(X)
- score(X, y)

Recuerda:

datarebel

1) ¡Usa regularización!
   a) Ayuda a prevenir el sobreajuste
   b) Ayuda con la colinealidad
   c) Facilita una perilla para ajustar el intercambio sesgo/variación
2) ¡No olvides estandarizar los datos!
   a) Columna a columna, reduce la media y divide por la desviación estándar
3) Lambdas controlan el tamaño (L1 & L2) y existencia (L1) de los coeficientes feature.
   a) Lambdas grandes significan mayor regularización (coeficientes menores/más pequeños) y menor complejidad del modelo.
4) ¡Puedes tenerlo todo! (ElasticNet)

datarebel

¡Gracias!

Data Translator