

Validación de modelos y control de sobreajuste

**Licenciatura en Inteligencia Artificial y Ciencia de Datos, CUGDL,
Universidad de Guadalajara.**

Guadalajara, Jal., agosto de 2025

Validación de modelos y control de sobreajuste

Introducción

- Esta unidad se ubica después de estudiar la **regresión lineal simple y múltiple**, y antes de abordar los modelos de clasificación como la regresión logística. Representa un puente clave entre la construcción del modelo y su evaluación robusta.

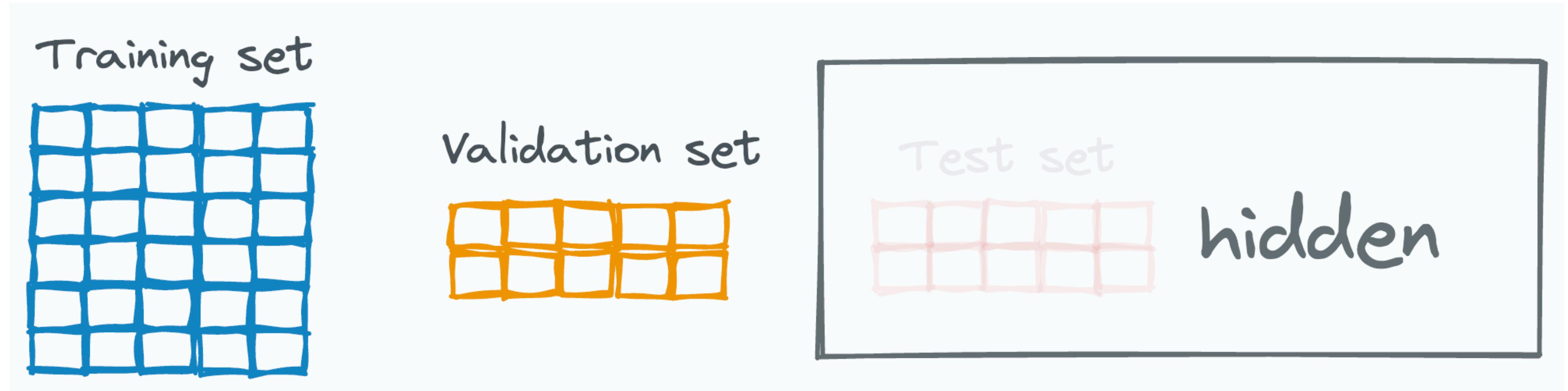
Objetivos

- Reconocer el problema del sobreajuste en modelos de Machine Learning.
- Evaluar el rendimiento de un modelo utilizando división de en conjunto datos de entrenamiento y de prueba, junto con validación cruzada.
- Aplicar técnicas de regularización como Rige y Lasso para mejorar la capacidad de generalización del modelo.

Conjuntos de datos

Entrenamiento, validación y prueba

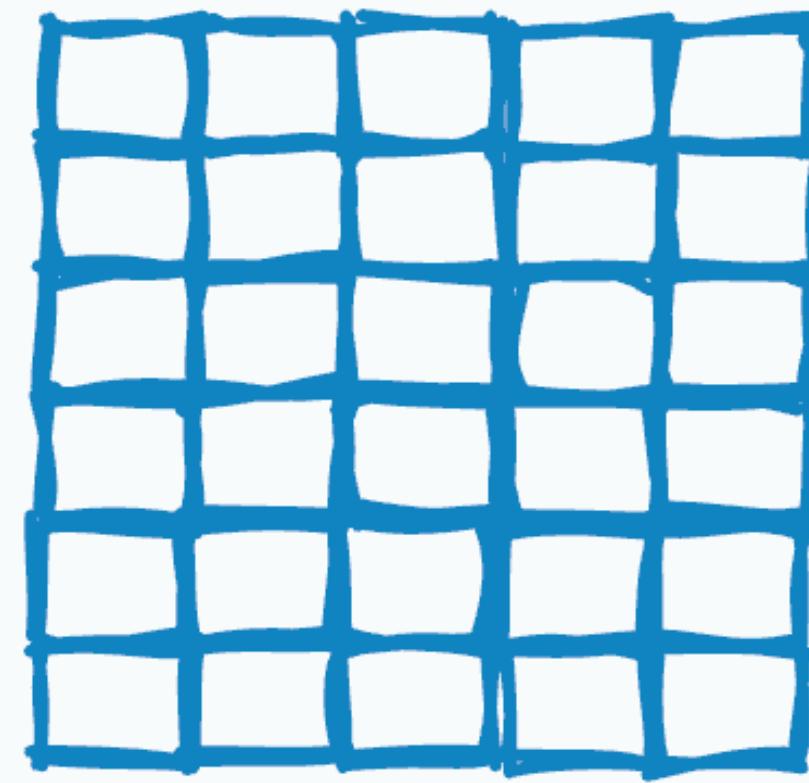
- Evaluar el modelo con los mismos datos con el cual se entrenó **no es una buena práctica**.
- El modelo podría "memorizar" relaciones particulares del conjunto de entrenamiento.
- Esto puede producir un desempeño artificialmente algo que no se replica en datos nuevos.
- El objetivo no es que el modelo memorice, sino que **generalice**.



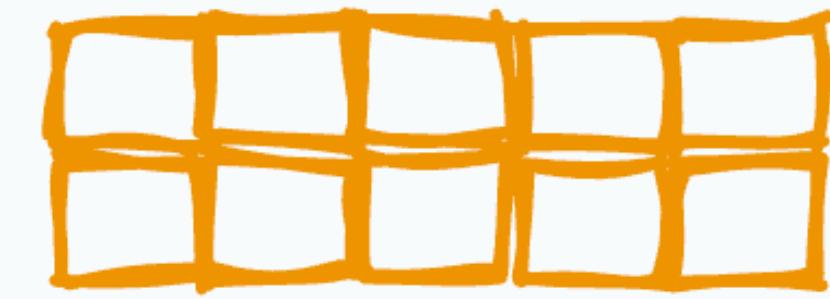
Conjuntos de datos

Entrenamiento, validación y prueba

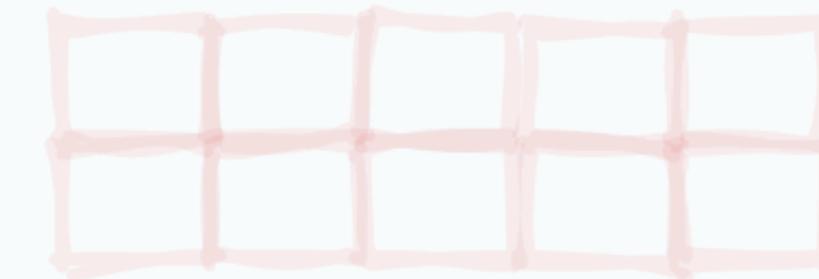
Training set



Validation set



Test set



hidden

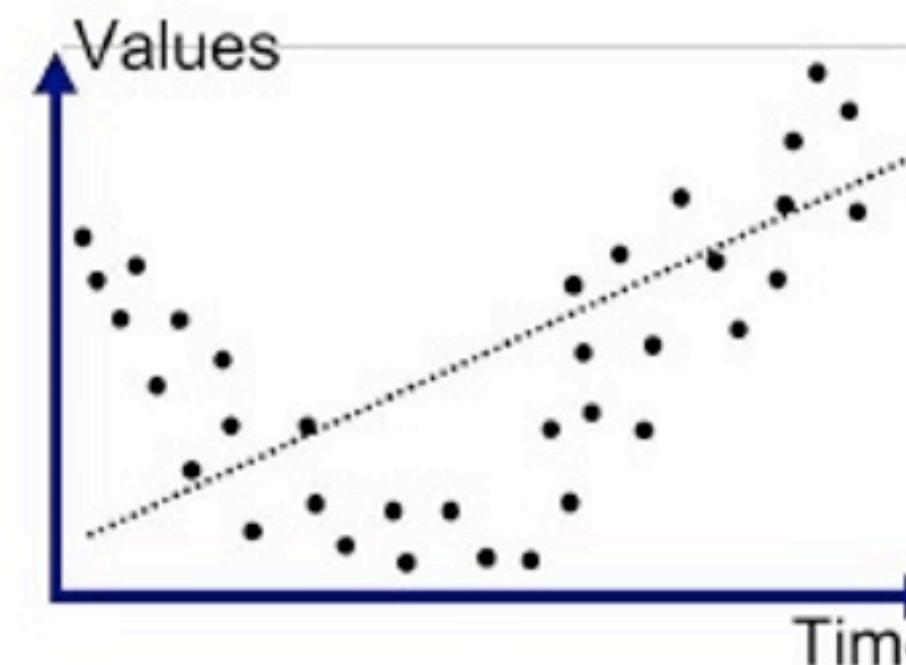
- **Conjunto de entrenamiento:** Es el subconjunto más grande de los datos y se utiliza para entrenar el modelo. Durante esta etapa, el modelo ajusta sus parámetros internos al identificar patrones y relaciones en los datos proporcionados. Este conjunto es esencial para que el modelo "aprenda" las reglas básicas del problema que debe resolver.

- **Conjunto de validación:** Es un subconjunto separado que se usa durante el entrenamiento para evaluar el rendimiento del modelo. Sirve para ajustar los hiperparámetros, entre otras cosas, para monitorear si el modelo tiene un buen rendimiento. Ayuda a garantizar que el modelo funcione bien con datos que no ha visto durante el entrenamiento.

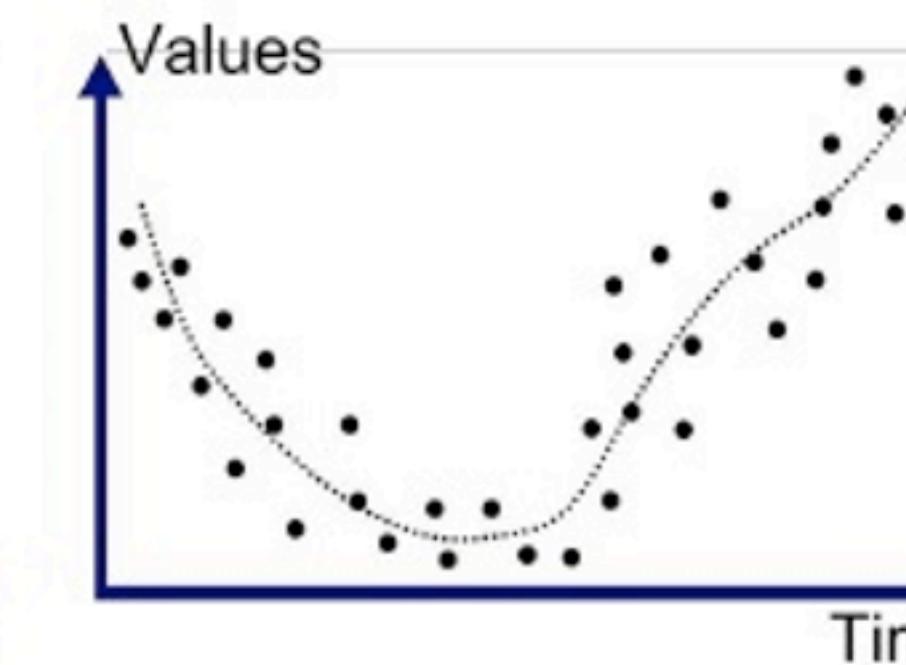
- **Conjunto de prueba:** Es un conjunto completamente independiente que se utiliza después de finalizar el entrenamiento. Este conjunto nunca se utiliza durante el proceso de entrenamiento o validación y tiene como objetivo medir el rendimiento real del modelo en datos no vistos, proporcionando una evaluación imparcial de su capacidad para generalizar.

El problema del sobreajuste

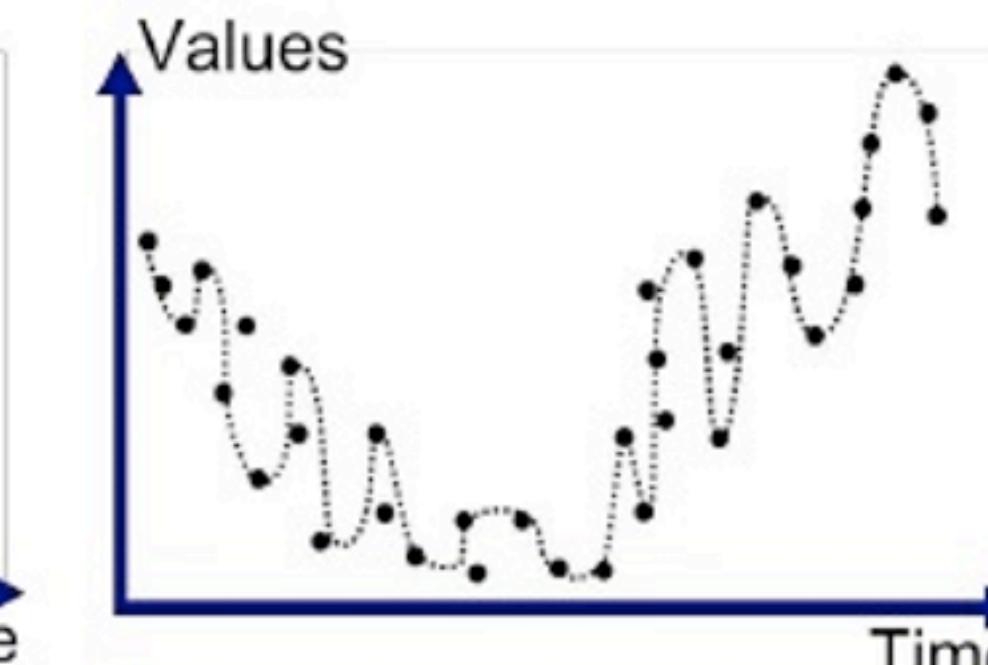
- Un modelo está **sobreajustado** cuando aprende demasiado bien el conjunto de entrenamiento, incluyendo ruido o patrones no generalizables.
- Tiene un **error muy bajo** en entrenamiento, pero alto en prueba o nuevos datos.
- El modelo memoriza en lugar de aprender patrones útiles.
- Lo contrario sería el **subajuste**, cuando el modelo es demasiado simple y no capta ni los patrones relevantes.



Underfitted



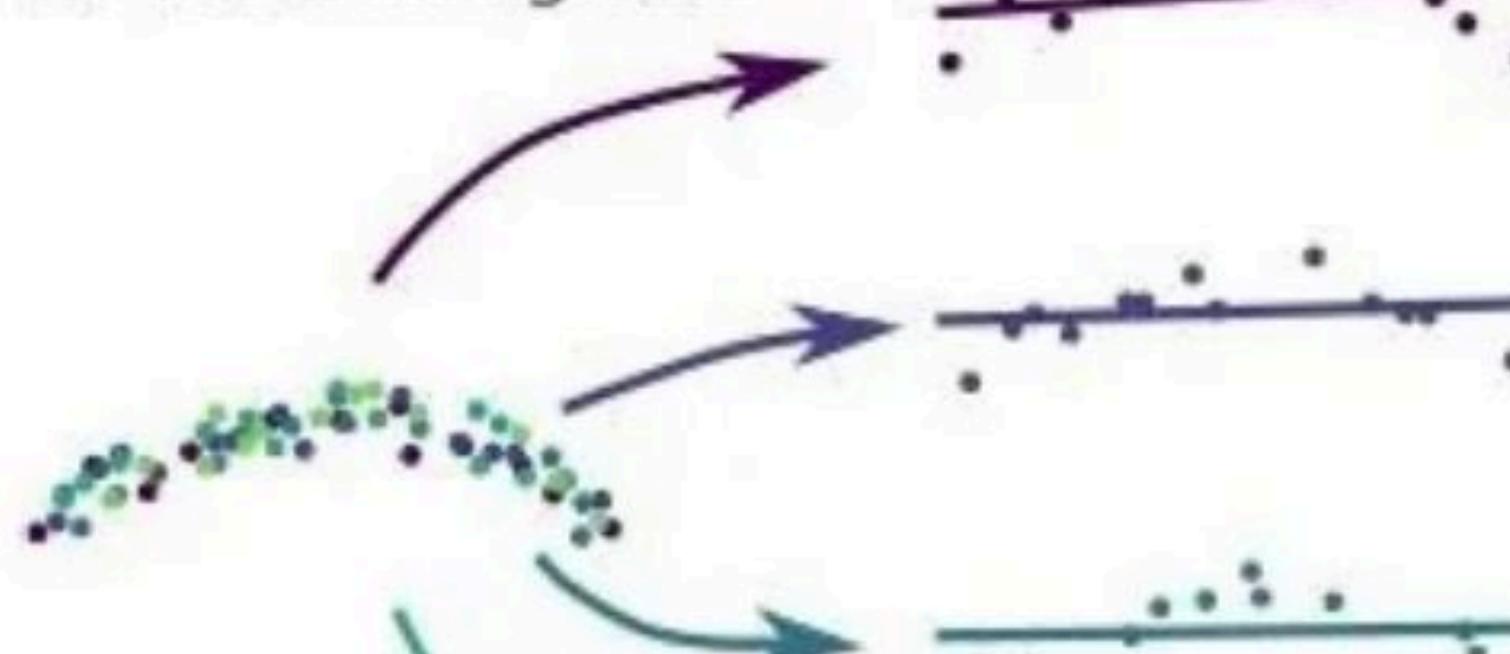
Good Fit/Robust



Overfitted

Bias and Variance in ML models

Train the same model
on different subsets
of the training data



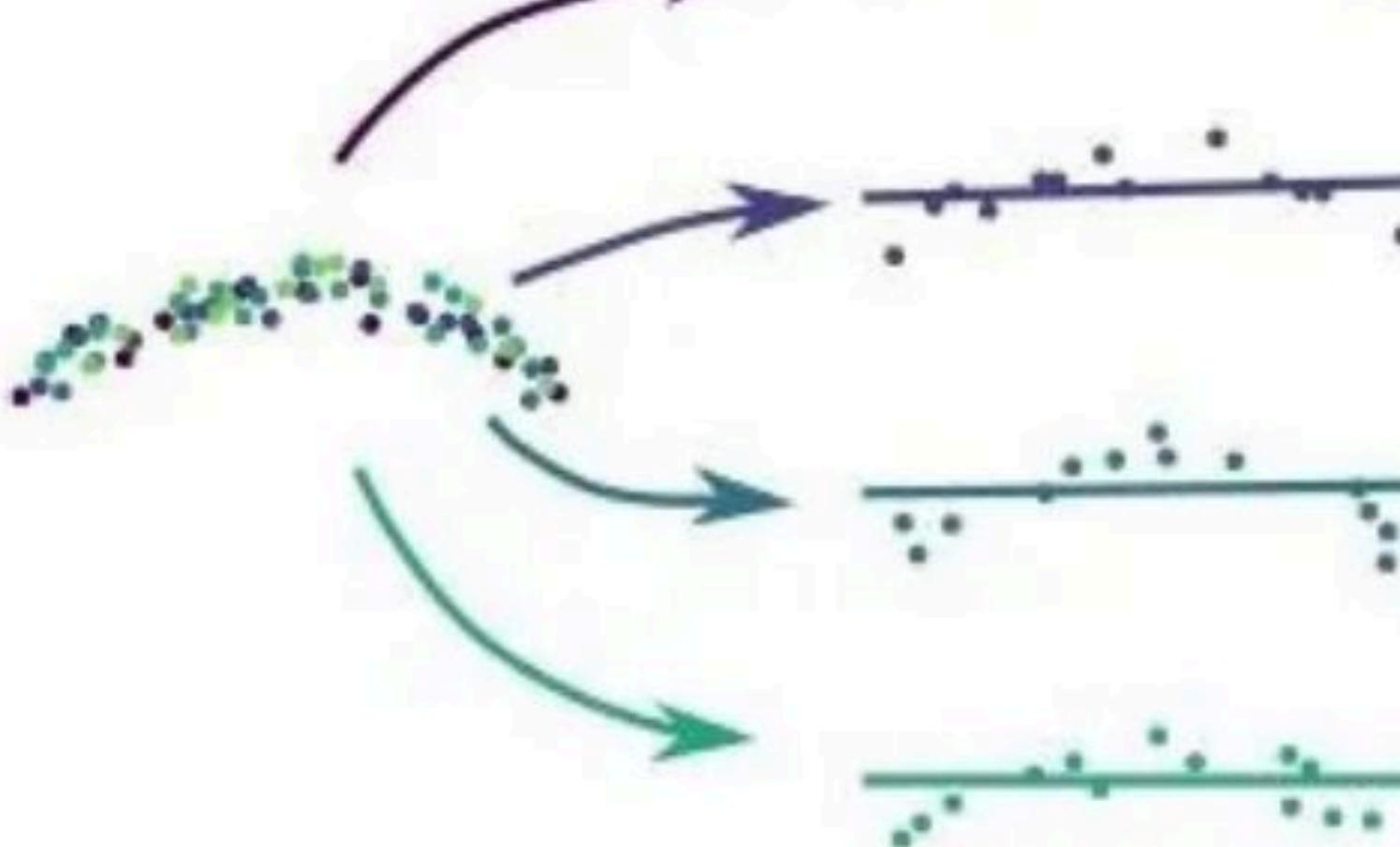
Underfit model



Overfit model



"Just right" model ✓



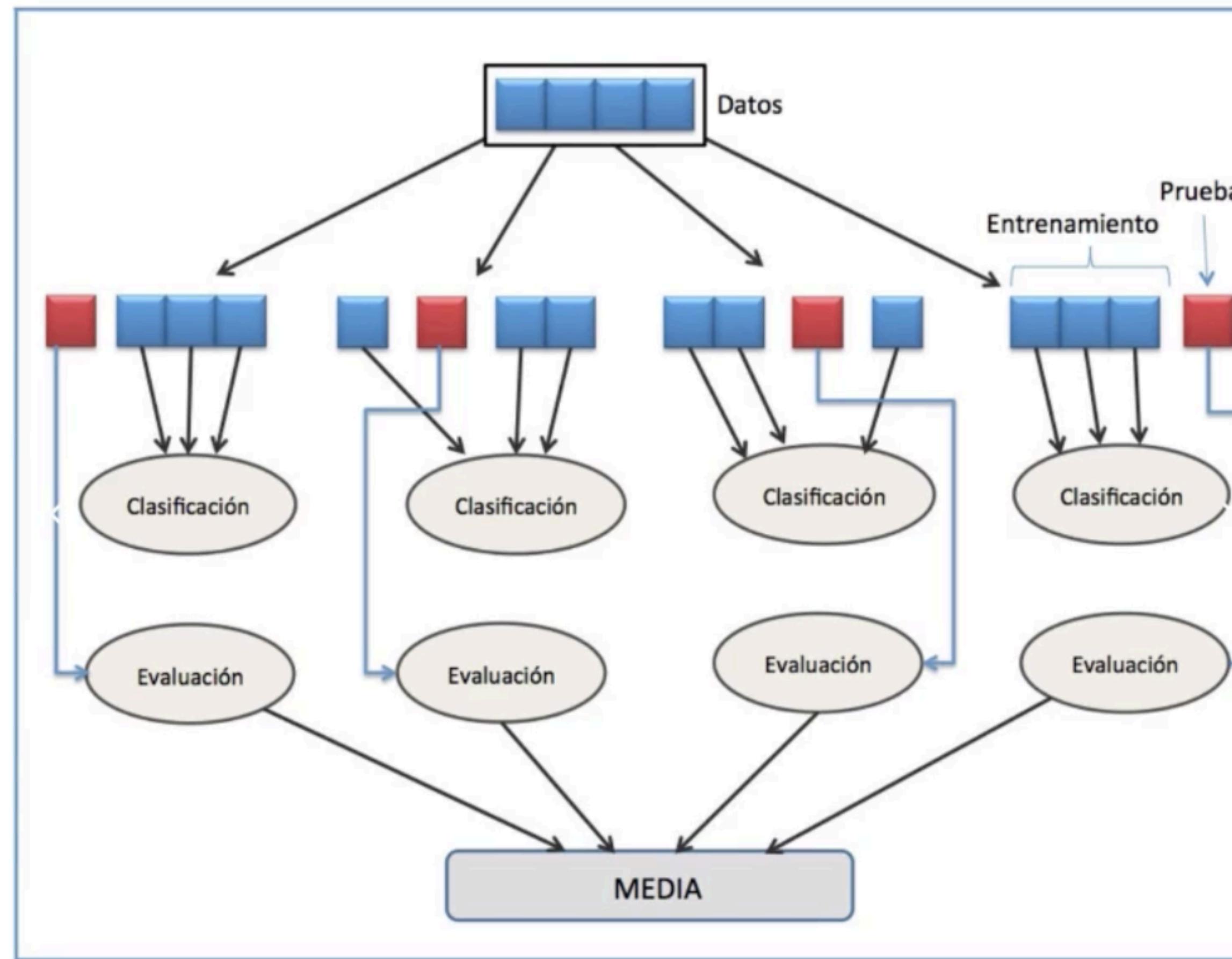
High bias, Low variance

Low bias, High variance

Low bias, Low variance



Validación cruzada



- La **validación cruzada** es una técnica muy utilizada para evaluar los resultados del análisis estadístico y garantizar que son independientes de la partición que se ha hecho entre el conjunto de entrenamiento y el de prueba.
- Se repite el experimento varias veces, partiendo los datos tantas veces como sean necesarias, para calcular la media aritmética obtenida de las medias de evaluación sobre diferentes particiones llevadas a cabo.

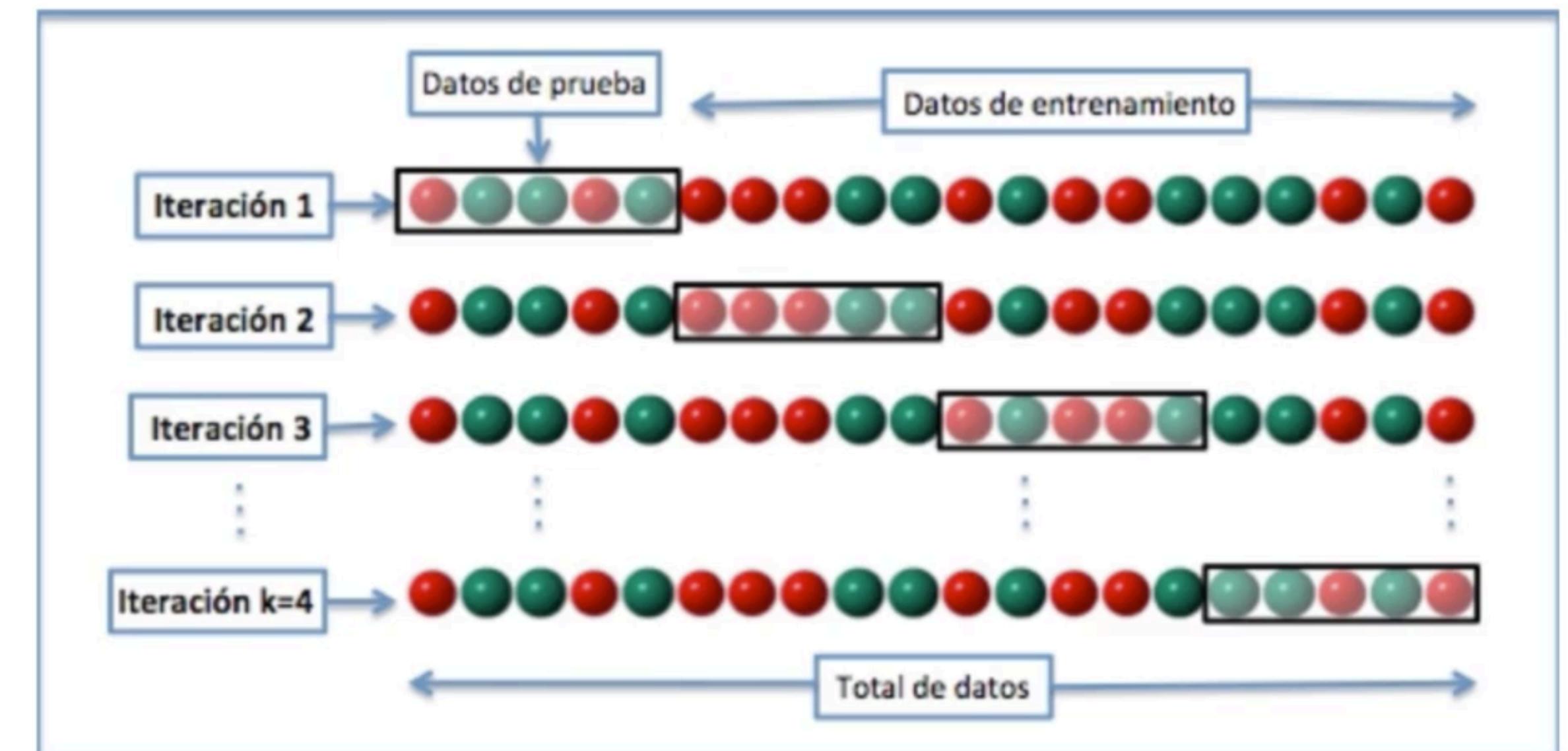
Validación cruzada

KFold

- Los datos de muestras se dividen en k subconjuntos. De forma que uno de los subconjuntos se utiliza para la prueba y $k-1$ para el modelo.
- Se vuelve a repetir pero intercambiando los subconjuntos. Hasta tener k iteraciones.
- Se realiza la media de la métrica de evaluación (e.g., mse, r^2 , accuracy, etc.) de cada iteración

$$E = \frac{1}{K} \sum_{i=1}^k E_i$$

Validación cruzada de K iteraciones K-fold Cross Validation

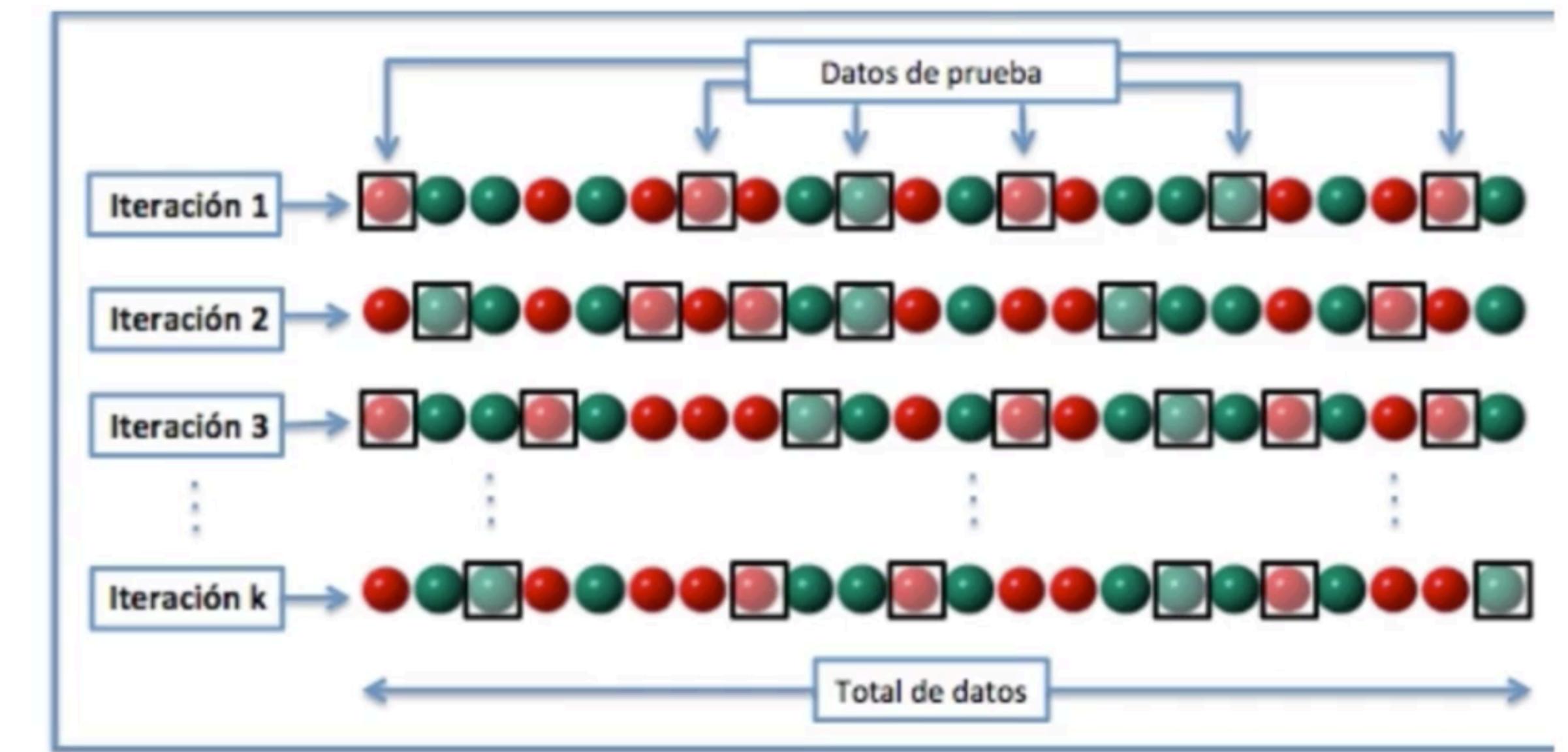


Validación cruzada

Aleatoria

- Se divide aleatoriamente el conjunto de datos de entrenamiento y de prueba.

Validación cruzada aleatoria Random Cross Validation



Evaluación de un modelo

Métricas

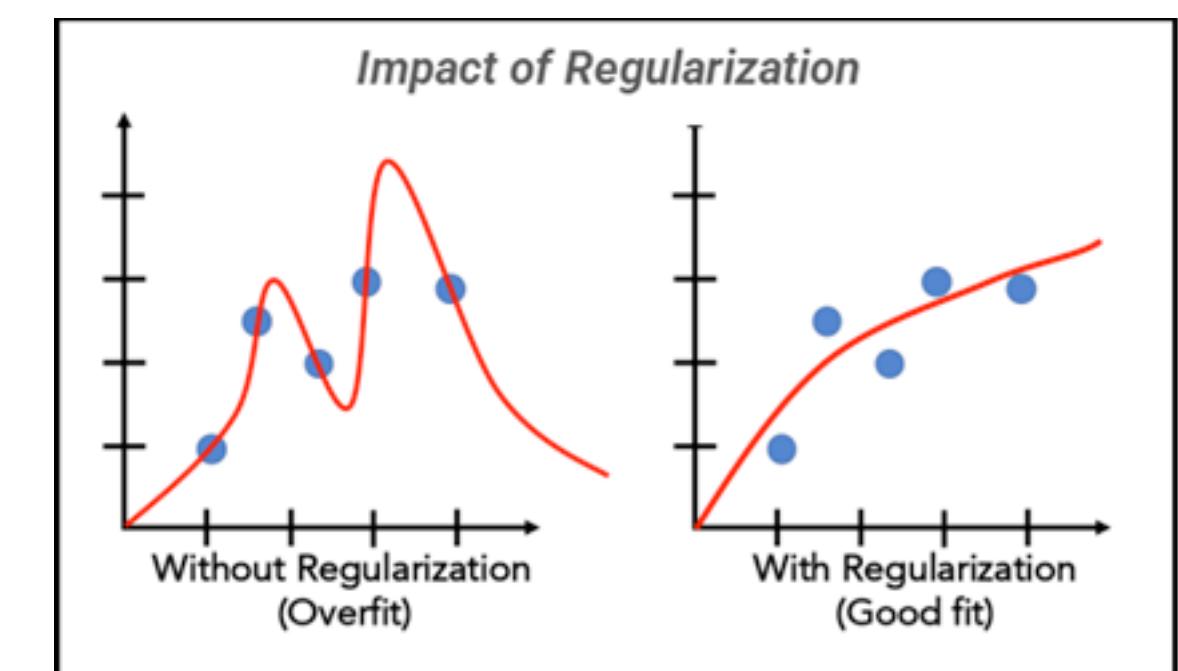
- Una vez que separamos datos o aplicamos validación cruzada, necesitamos medir el **error de predicción** del modelo.
- Las métricas más utilizadas en una **regresión** son:
 - MAE (Mean Absolute Error): $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$
 - MSE (Mean Squared Error): $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
 - R^2 (coeficiente de determinación)

Sobreajuste

Regularización

A veces, aunque validamos correctamente, el modelo **tiene a sobreajustar** cuando hay:

- Muchas variables predictoras.
- Variables altamente correlacionadas.
- Pocos datos en comparación con el número de variables.



La **regularización** modifica la llamada **función de costo** del modelo, penalizando la complejidad de éste. Su objetivo es encontrar un equilibrio entre el ajuste y la simplicidad del modelo.

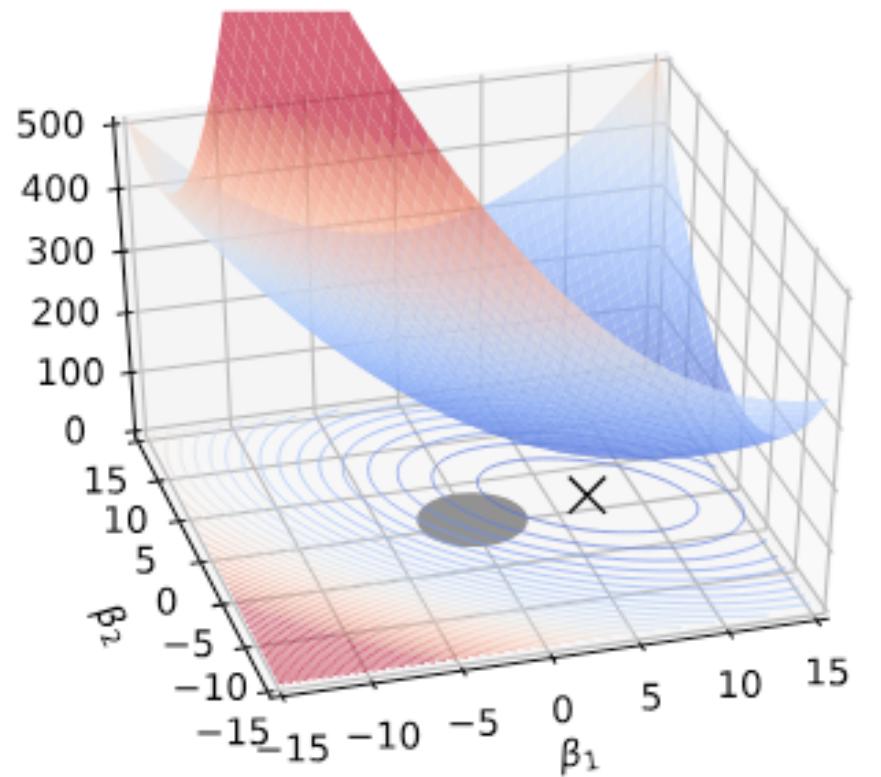
Sobreajuste

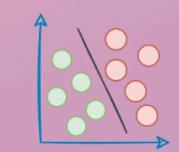
Función de costo

- Una función de costo (cost function/loss function) es una expresión matemática que **cuantifica el error** que comete un modelo al realizar predicciones.
- Por ejemplo, en **regresión lineal**, nuestra función de costo sería MSE

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

- Es el criterio que el modelo intenta minimizar (o maximizar en otros casos) durante el entrenamiento.
- Nos guía a ajustar los **parámetros** del modelo. Por ejemplo, en este caso los coeficientes β_i .
- Igualmente, nos da una idea del rendimiento, por ejemplo, a menor pérdida = mejor modelo.





10 Most Common Loss Functions in Machine Learning

by Avi Chawla

Loss Function Name	Description	Function
--------------------	-------------	----------

Regression Losses

Mean Bias Error	Captures average bias in prediction. But is rarely used for training.	$\mathcal{L}_{MBE} = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))$
Mean Absolute Error	Measures absolute average bias in prediction. Also called L1 Loss.	$\mathcal{L}_{MAE} = \frac{1}{N} \sum_{i=1}^N y_i - f(x_i) $
Mean Squared Error	Average squared distance between actual and predicted. Also called L2 Loss.	$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2$
Root Mean Squared Error	Square root of MSE. Loss and dependent variable have same units.	$\mathcal{L}_{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2}$
Huber Loss	A combination of MSE and MAE. It is parametric loss function.	$\mathcal{L}_{\text{Huberloss}} = \begin{cases} \frac{1}{2}(y_i - f(x_i))^2 & : y_i - f(x_i) \leq \delta \\ \delta(y_i - f(x_i) - \frac{1}{2}\delta) & : \text{otherwise} \end{cases}$
Log Cosh Loss	Similar to Huber Loss + non-parametric. But computationally expensive.	$\mathcal{L}_{\text{LogCosh}} = \frac{1}{N} \sum_{i=1}^N \log(\cosh(f(x_i) - y_i))$

Classification Losses (Binary + Multi-class)

Binary Cross Entropy (BCE)	Loss function for binary classification tasks.	$\mathcal{L}_{BCE} = \frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i))$
Hinge Loss	Penalizes wrong and right (but less confident) predictions. Commonly used in SVMs.	$\mathcal{L}_{\text{Hinge}} = \max(0, 1 - (f(x) \cdot y))$
Cross Entropy Loss	Extension of BCE loss to multi-class classification.	$\mathcal{L}_{CE} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \cdot \log(f(x_{ij}))$ N : samples; M : classes
KL Divergence	Minimizes the divergence between predicted and true probability distribution	$\mathcal{L}_{KL} = \sum_{i=1}^N y_i \cdot \log\left(\frac{y_i}{f(x_i)}\right)$

Regularización

Regresión lineal tradicional

- Recordemos antes nuestra **regresión lineal múltiple**, donde para p variables predictoras,

$$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}$$

- Este modelo lineal es "entrenado" minimizando una función de costo:

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

Problema:

- Si las variables están altamente correlacionadas (colineales), estos coeficientes β pueden crecer mucho y ser inestables. Si hay muchas variables, también puede haber problemas de sobreajuste.

Regularización

Regresión lineal tradicional

- Esto problema tiene una solución analítica.
- Si utilizamos notación matricial para el modelo $Y = XB + \varepsilon$, podemos aplicar una derivada y encontrar un punto crítico (el mínimo en este caso).
- Lo que queremos minimizar son los residuos al cuadrado

$$\varepsilon^T \varepsilon = (Y - XB)^T (Y - XB) = Y^T Y + B^T X^T X B - 2Y^T X B$$

$$\frac{\partial \varepsilon^T \varepsilon}{\partial B} = 0 = 2X^T X \beta - 2X^T Y$$

$$\beta = (X^T X)^{-1} X^T Y$$

Regularización L2

Ridge

- Ridge es una forma de regresión lineal que busca reducir el sobreajuste penalizando los coeficientes grandes del modelo. Esto lo hace añadiendo una penalización al tamaño de estos coeficientes.

- $$\text{Loss}_{\text{Ridge}} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p \beta_j^2$$
- λ es el **hiperparámetro** de la regularización (en scikit-learn es denotado como α).
- Se penaliza la suma de los cuadrados de los coeficientes (sin incluir β_0).
- Cuanto mayor es λ , más se castigan los coeficientes grandes.

Regularización L2

Ridge

- Esto problema **también** tiene una solución analítica.

$$\varepsilon^T \varepsilon = (Y - XB)^T(Y - XB) + \lambda B^T B = Y^T Y + B^T X^T X B - 2Y^T X B + \lambda B^T B$$

$$\frac{\partial \varepsilon^T \varepsilon}{\partial B} = 0 = 2X^T X \beta - 2X^T Y + 2\lambda \beta$$

$$\beta = (X^T X + I\lambda)^{-1} X^T Y$$

- Esta modificación tiene un efecto importante. Si $X^T X$ tuviera problemas de multicolinealidad, esta se hace difícil de invertir, lo que lo hace inestable y numéricamente insegura. Pero si agregamos el término $I\lambda$, se agrega un valor positivo a la diagonal y se garantiza esta inversión, lo que se traduce en un cálculo más estable, numéricamente hablando, y con parámetros β más pequeños.

Regularización L1

Lasso

- Lasso es una técnica de regresión lineal regularizada que, además de reducir el sobreajuste, puede eliminar completamente algunas variables al forzar coeficientes exactamente a cero.

$$\text{Loss}_{\text{Lasso}} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

- λ es el **hiperparámetro** de la regularización (en scikit-learn es denotado como α).
- Se penaliza a los coeficientes grandes, pero **no suavemente como Ridge**, sino con una penalización constante.
- A diferencia de Ridge, que solo reduce los coeficientes, Lasso puede forzarlos exactamente a cero.
- Por lo mismo, esto ayuda también a hacer una selección de variables relevantes en el modelo.
- A diferencia de Ridge, la minimización no tiene una solución analítica, haciendo un poco más difícil el cómputo.

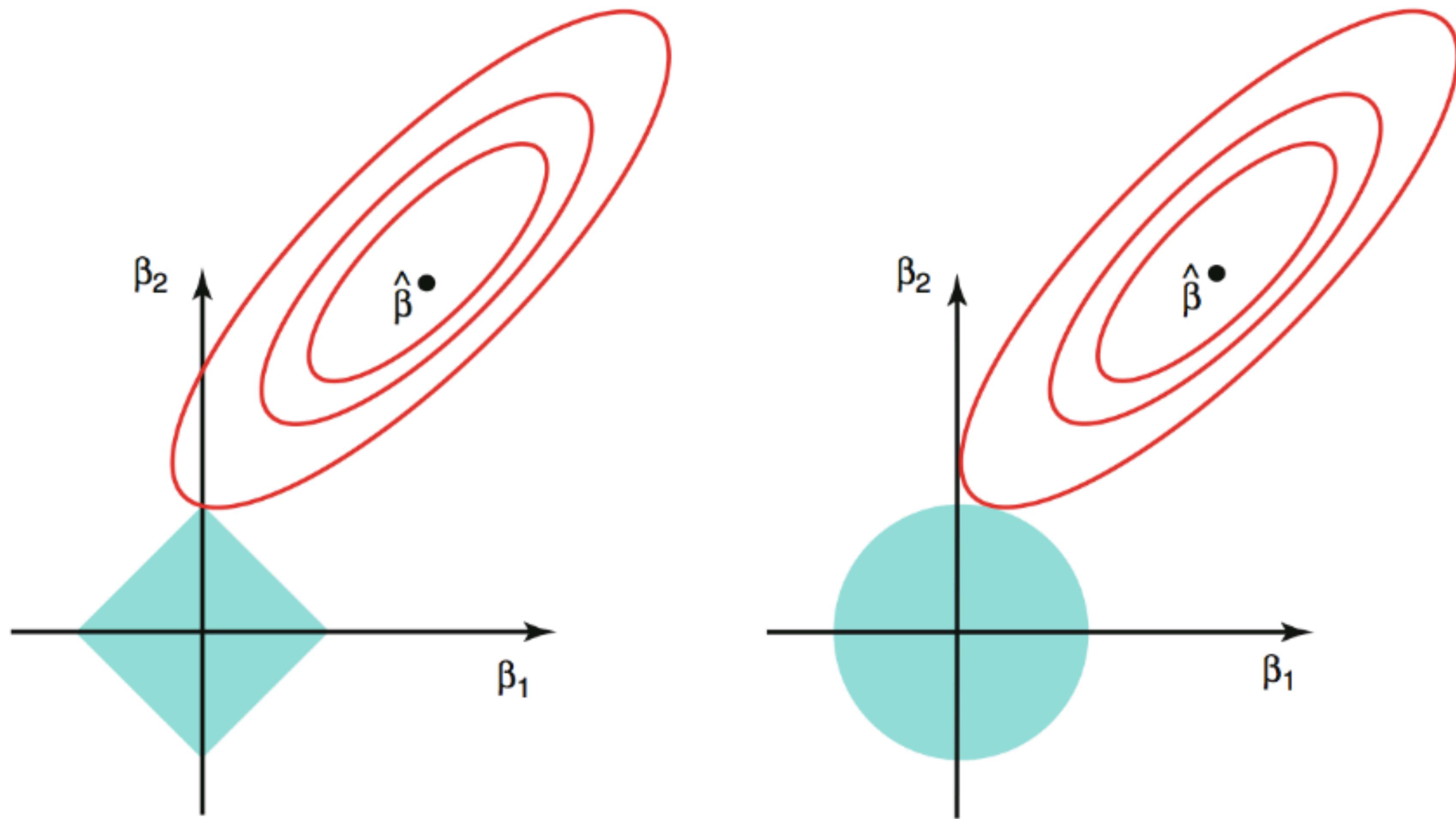


FIGURE 6.7. Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.

Regularización

Ridge vs Lasso

Característica	Ridge	Lasso
Penalización	$\sum \beta_j^2$	$\sum \beta_j $
Reduce coeficientes	Si	Si
Puede hacer coeficientes = 0	No	Si
Selección de variables	No	Si
Solución analítica	Si	No (requiere optimización iterativa)