

Machine Learning Supervisado: De la Econometría a la Predicción

HE2: Consultoría Económica con IA Responsable

Santiago Neira & Catalina Bernal

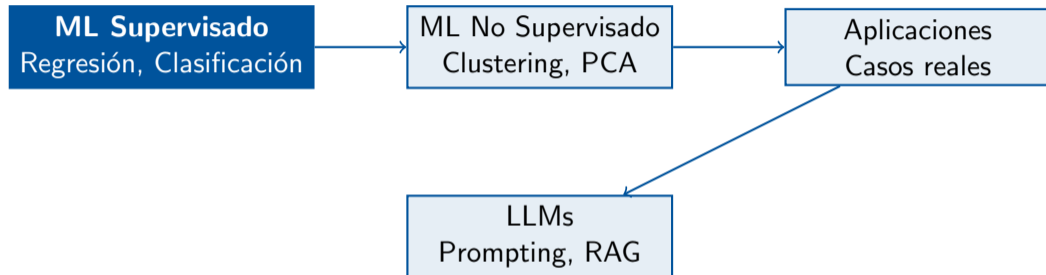
Universidad de los Andes
Departamento de Economía

Febrero 2026

Agenda de hoy

- 1 Roadmap del Módulo Técnico
- 2 Machine Learning: El Marco General
- 3 El Puente: Regresión Lineal en Lenguaje ML
- 4 Repaso Técnico: OLS, Supuestos y Métricas
- 5 Bias-Variance Tradeoff: El Corazón del ML
- 6 Train/Test Split: ¿Qué Significa “Predecir”?
- 7 Cross-Validation
- 8 Regresiones Polinomiales
- 9 Overfitting y Underfitting: La U-Curve

¿Hacia dónde vamos? Panorama técnico prox 6 semanas



Hoy: Regresión lineal como puente entre econometría y ML

Próxima clase: Regularización (Ridge, Lasso) y tuning de hiperparámetros

¿Qué es Machine Learning?

Definición operativa: Algoritmos que aprenden patrones de los datos para hacer predicciones o tomar decisiones.

El problema fundamental:

$$Y = f(X) + \varepsilon \quad (1)$$

donde:

- Y es la variable objetivo (lo que queremos predecir)
- X es el vector de características (features)
- $f(\cdot)$ es la función **desconocida** que relaciona X con Y
- ε es el error irreducible

Objetivo del ML: Encontrar \hat{f} tal que $\hat{Y} = \hat{f}(X)$ sea una buena aproximación.

La Caja Negra: $f(X)$



Inputs



BLACK BOX

$\rightarrow f(X) \rightarrow$

\tilde{Y} Output

Implicación clave: En ML, nos importa menos *qué hay dentro* de la caja y más *qué tan bien predice*.

Esto contrasta con econometría, donde la especificación f es el objeto de interés.

Supervisado vs. No Supervisado

Aprendizaje Supervisado

- Tenemos (X_i, Y_i) para entrenar
- El modelo aprende la relación $X \rightarrow Y$
- **Regresión:** Y continua
- **Clasificación:** Y categórica

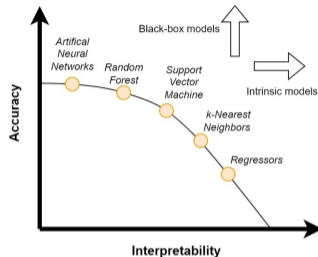
Ejemplos: Predecir ingreso, detectar fraude, pronóstico de ventas

Aprendizaje No Supervisado

- Solo tenemos X_i (sin etiquetas)
- El modelo busca estructura
- **Clustering:** Agrupar similares
- **Reducción de dimensionalidad**

Ejemplos: Segmentación de clientes, detección de anomalías

Interpretabilidad y IA Responsable



¿Por qué importa la interpretabilidad?

- **Accountability:** ¿Quién es responsable de una decisión algorítmica?
- **Fairness:** ¿El modelo discrimina grupos protegidos?
- **Comunicación:** Explicar a stakeholders no técnicos
- **Debugging:** Entender por qué el modelo falla

Trade-off: Modelos más complejos → mejor predicción, peor interpretabilidad

Ya conocen este modelo

Regresión Lineal Múltiple:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_p X_{pi} + \varepsilon_i \quad (2)$$

En notación matricial:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \quad (3)$$

En lenguaje de ML:

- Y es el *target* o *label*
- X_1, \dots, X_p son las *features*
- $\boldsymbol{\beta}$ son los *parámetros* o *pesos*
- El modelo $\hat{f}(X) = X\hat{\boldsymbol{\beta}}$ es el *learner*

Dos perspectivas, un modelo

Perspectiva Econométrica

- Interés en $\hat{\beta}_j$
- ¿Cuál es el efecto de X_j sobre Y ?
- Inferencia causal
- Supuestos: exogeneidad, homocedasticidad
- Tests de hipótesis sobre β

Perspectiva ML

- Interés en \hat{Y}
- ¿Qué tan bien predecimos?
- Generalización a datos nuevos
- Supuestos: menos restrictivos
- Métricas de predicción

El cambio de paradigma

Econometría: “¿ X causa Y ?”

ML: “Dado X , ¿cuánto vale Y ?”

Mínimos Cuadrados Ordinarios (OLS)

Objetivo: Minimizar la suma de errores cuadráticos

$$\min_{\beta} \sum_{i=1}^n (Y_i - \mathbf{x}'_i \beta)^2 = \min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|^2 \quad (4)$$

Solución analítica:

$$\hat{\beta}_{OLS} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \quad (5)$$

Predicciones:

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} = \mathbf{H}\mathbf{Y} \quad (6)$$

donde **H** es la matriz “hat” (proyección).

Supuestos del Modelo Lineal Clásico

- ❶ **Linealidad en parámetros:** $Y = X\beta + \varepsilon$
- ❷ **Exogeneidad estricta:** $\mathbb{E}[\varepsilon|X] = 0$
- ❸ **Rango completo:** $\text{rank}(\mathbf{X}) = p + 1$ (no multicolinealidad perfecta)
- ❹ **Homocedasticidad:** $\text{Var}(\varepsilon|X) = \sigma^2$
- ❺ **No autocorrelación:** $\text{Cov}(\varepsilon_i, \varepsilon_j|X) = 0$ para $i \neq j$
- ❻ **(Para inferencia) Normalidad:** $\varepsilon|X \sim N(0, \sigma^2)$

Nota ML: Para predicción, nos importa principalmente (1) y (3). Los demás afectan inferencia sobre β , no necesariamente predicción.

Coeficiente de Determinación:

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2} \quad (7)$$

- SSR = Suma de cuadrados de residuos (error no explicado)
- SST = Suma de cuadrados total (variación total de Y)
- $R^2 \in [0, 1]$: proporción de varianza explicada

R^2 **Ajustado:** Penaliza por número de variables

$$R_{adj}^2 = 1 - \frac{SSR/(n - p - 1)}{SST/(n - 1)} = 1 - (1 - R^2) \frac{n - 1}{n - p - 1} \quad (8)$$

Error Cuadrático Medio (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (9)$$

Raíz del Error Cuadrático Medio (RMSE):

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2} \quad (10)$$

¿Por qué RMSE?

- Está en las mismas unidades que Y
- Interpretación directa: “en promedio, nos equivocamos por RMSE unidades”

Reglas del Pulgar

Para R^2 (¡con cuidado!)

- $R^2 > 0,7$: Ajuste “bueno” en muchos contextos
- $R^2 \approx 0,3 - 0,5$: Común en ciencias sociales
- R^2 alto no implica causalidad ni buen modelo

Para RMSE

- Comparar contra σ_Y (desviación estándar de Y)
- $RMSE < \sigma_Y$ implica que el modelo aporta información
- Útil para comparar modelos *en el mismo problema*

Advertencia: Estas métricas calculadas sobre los datos de entrenamiento son **optimistas**. Veremos por qué.

Descomposición del Error de Predicción

Para un punto nuevo x_0 , el error esperado de predicción es:

$$\mathbb{E} \left[(Y_0 - \hat{f}(x_0))^2 \right] = \underbrace{\text{Var}(\hat{f}(x_0))}_{\text{Varianza}} + \underbrace{[\text{Bias}(\hat{f}(x_0))]^2}_{\text{Sesgo}^2} + \underbrace{\text{Var}(\varepsilon)}_{\text{Error irreducible}} \quad (11)$$

- **Bias (Sesgo):** Error sistemático por simplificar demasiado

$$\text{Bias}(\hat{f}(x_0)) = \mathbb{E}[\hat{f}(x_0)] - f(x_0) \quad (12)$$

- **Varianza:** Sensibilidad del modelo a la muestra específica

$$\text{Var}(\hat{f}(x_0)) = \mathbb{E} \left[(\hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)])^2 \right] \quad (13)$$

Intuición del Tradeoff

Low Bias, Low Variance
MSE= 0.11



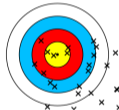
Low Bias, High Variance
MSE= 0.79



High Bias, Low Variance
MSE= 0.62



High Bias, High Variance
MSE= 0.99



Alto Bias (Underfitting)

- Modelo muy simple
- No captura la señal
- Error sistemático

Alta Varianza (Overfitting)

- Modelo muy complejo
- Se aprende el ruido
- Inestable entre muestras

Conexión con Econometría

En econometría clásica:

- Nos obsesionamos con el **bias** porque queremos $\hat{\beta}$ insesgados
- Un modelo con variables omitidas \rightarrow sesgo en $\hat{\beta} \rightarrow$ inferencia causal incorrecta
- Preferimos especificación correcta aunque tenga varianza alta

En ML:

- El objetivo es minimizar el **error total de predicción**
- A veces conviene **introducir sesgo deliberadamente** si reduce mucho la varianza
- Ridge y Lasso hacen exactamente esto (próxima clase)

El cambio de paradigma, formalmente

Econometría: Minimizar Bias^2

ML: Minimizar $\text{Bias}^2 + \text{Varianza}$

El Problema Fundamental

Pregunta: Si ajusto un modelo y calculo R^2 o MSE , ¿tengo una buena medida de qué tan bien predice?

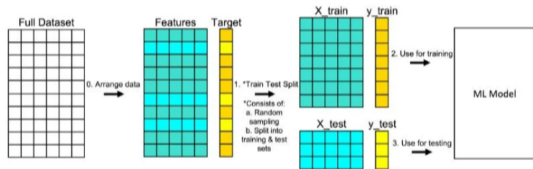
Respuesta: No necesariamente.

¿Por qué?

- El modelo fue optimizado para esos datos específicos
- Puede haber memorizado el ruido (overfitting)
- El error de entrenamiento es **optimista**

Lo que realmente nos importa: ¿Cómo se desempeña en *datos que nunca ha visto*?

La Solución: Train/Test Split



Procedimiento:

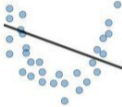

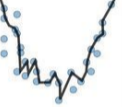
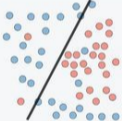

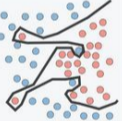



- 1 Dividir datos: típicamente 70–80 % train, 20–30 % test
- 2 Entrenar el modelo **solo** con training
- 3 Evaluar en datos nunca vistos

Métricas:

$$MSE_{train} = \frac{1}{n_{train}} \sum_{i \in train} (Y_i - \hat{Y}_i)^2$$

$$MSE_{test} = \frac{1}{n_{test}} \sum_{i \in test} (Y_i - \hat{Y}_i)^2$$

El Tradeoff en Acción

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none"> • High training error • Training error close to test error • High bias 	<ul style="list-style-type: none"> • Training error slightly lower than test error 	<ul style="list-style-type: none"> • Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	<ul style="list-style-type: none"> • Complexify model • Add more features • Train longer 		<ul style="list-style-type: none"> • Perform regularization • Get more data

	Under.	Óptimo	Over.
Bias	Alto	Moderado	Bajo
Varianza	Baja	Moderada	Alta
Error Train	Alto	Bajo	Muy bajo
Error Test	Alto	Bajo	Alto

Escenario	MSE_{train}	MSE_{test}
Underfitting	Alto	Alto
Buen modelo	Bajo	Bajo
Overfitting	Muy bajo	Alto

Señales de overfitting:

- $MSE_{test} \gg MSE_{train}$
- R^2_{train} muy alto, R^2_{test} bajo

Regla práctica: Si MSE_{test} es más de 10-20 % mayor que MSE_{train} , probablemente hay overfitting.

Data Leakage: El Pecado Capital

Definición

Data leakage ocurre cuando información del conjunto de test “contamina” el entrenamiento.

Ejemplos comunes:

- 1 Normalizar/estandarizar usando *todos* los datos antes de dividir
- 2 Seleccionar variables basándose en correlaciones con *todos* los datos
- 3 En series de tiempo: usar datos futuros para predecir pasado
- 4 Incluir variables que “vienen del futuro” respecto a la predicción

Consecuencia: Métricas de test optimistas → modelo falla en producción

Regla de oro: El test set debe simular **exactamente** las condiciones de uso real.

Ejemplo: Estandarización Correcta

¿Cómo estandarizar sin data leakage?

× **INCORRECTO**

```
# Calcula con TODOS los datos
mean = X.mean()
std = X.std()
X_scaled = (X - mean) / std

# Luego divide
X_train, X_test = split(X_scaled)
```

El test “ve” información del futuro

✓ **CORRECTO**

```
# Primero divide
X_train, X_test = split(X)

# Calcula SOLO con train
mean_tr = X_train.mean()
std_tr = X_train.std()

# Aplica a ambos
X_train_sc = (X_train-mean_tr)/std_tr
X_test_sc = (X_test-mean_tr)/std_tr
```

Regla: fit en train, transform en ambos

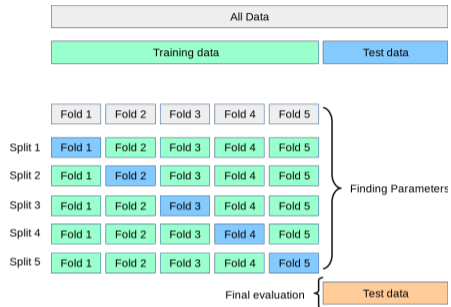
Problemas:

- El MSE_{test} depende de *qué* observaciones caen en test
- Con muestras pequeñas, un solo split puede ser muy variable
- Desperdiciamos datos que podrían usarse para entrenar

Solución: K-Fold Cross-Validation

- 1 Dividir datos en K “folds” (típicamente $K = 5$ o 10)
- 2 Para cada fold k :
 - Entrenar con los otros $K - 1$ folds
 - Evaluar en fold k
- 3 Promediar las K estimaciones de error

K-Fold Cross-Validation: Visualización



Error de Cross-Validation:

$$CV_{(K)} = \frac{1}{K} \sum_{k=1}^K MSE_k \quad (14)$$

donde MSE_k es el error en el fold k cuando se usa como test.

Estimadores Promedio y Variabilidad

Además del promedio, reportamos la desviación estándar:

$$SE(CV) = \sqrt{\frac{1}{K} \sum_{k=1}^K (MSE_k - \overline{MSE})^2} \quad (15)$$

¿Por qué importa?

- Alta variabilidad \rightarrow el modelo es inestable
- Comparar modelos: si los intervalos se solapan, no hay diferencia clara

Reglas prácticas:

- $K = 5$: Buen balance entre sesgo y varianza del estimador CV
- $K = 10$: Estándar en la práctica
- $K = n$ (Leave-One-Out): Bajo sesgo, alta varianza, computacionalmente costoso

Extendiendo el Modelo Lineal

Problema: La relación entre X e Y puede ser no lineal.

Solución: Regresión polinomial

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \dots + \beta_d X_i^d + \varepsilon_i \quad (16)$$

Observación clave: Esto sigue siendo un modelo *lineal en los parámetros*.

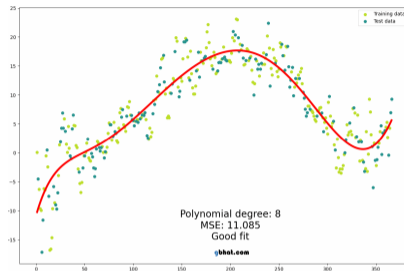
Si definimos:

$$\tilde{X}_i = [1, X_i, X_i^2, \dots, X_i^d]' \quad (17)$$

Entonces:

$$Y = \tilde{X}\beta + \varepsilon \quad (\text{jOLS aplica!}) \quad (18)$$

Grado del Polinomio y Complejidad



(Clic para ver animación)

El grado d controla la complejidad:

- d bajo \rightarrow modelo simple \rightarrow alto bias, baja varianza
- d alto \rightarrow modelo complejo \rightarrow bajo bias, alta varianza

Pregunta: ¿Cómo elegimos d ? \rightarrow Cross-validation

Selección del Grado Óptimo

Procedimiento:

- 1 Para cada $d \in \{1, 2, 3, \dots, d_{max}\}$:
 - Calcular $CV_{(K)}$ usando K-fold cross-validation
- 2 Seleccionar d^* que minimiza $CV_{(K)}$

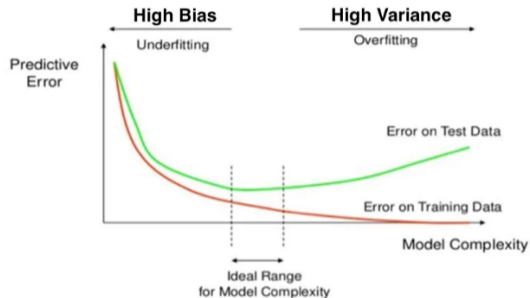
Alternativa: Regla de “1 desviación estándar”

- Seleccionar el modelo más simple cuyo CV esté dentro de 1 SE del mínimo
- Favorece parsimonia cuando la diferencia no es significativa

Nota

Este mismo procedimiento aplica para seleccionar *cualquier* hiperparámetro (próxima clase: λ en Ridge/Lasso).

La Curva en U del Error



Observaciones:

- Error de training **siempre** decrece con complejidad
- Error de test tiene un **mínimo** en complejidad óptima
- La brecha entre ambos indica overfitting

Diagnóstico: ¿Underfitting u Overfitting?

Diagnóstico	Error _{train}	Error _{test}
Underfitting	Alto	Alto
Buen ajuste	Bajo	Bajo (similar)
Overfitting	Muy bajo	Alto

Soluciones:

Si hay underfitting:

- Aumentar complejidad
- Añadir features
- Usar modelo más flexible

Si hay overfitting:

- Reducir complejidad
- Regularización (Ridge, Lasso)
- Más datos de entrenamiento

Preparando la Clase

La próxima clase veremos:

Regularización: Métodos que introducen sesgo deliberadamente para reducir varianza

- **Ridge:** Penaliza $\|\beta\|^2$ (norma L2)
- **Lasso:** Penaliza $\|\beta\|_1$ (norma L1) \rightarrow selección de variables
- **Elastic Net:** Combinación de ambas

Tuning de hiperparámetros:

- ¿Cómo elegir λ (parámetro de regularización)?
- Grid search + Cross-validation

Implementación en Python: `scikit-learn`

- ➊ **Cambio de paradigma:** De “¿ X causa Y ?” a “¿Qué tan bien predecimos Y ?”
- ➋ **Bias-Variance Tradeoff:** No podemos minimizar ambos simultáneamente
- ➌ **Train/Test Split:** Medir performance en datos no vistos es esencial
- ➍ **Cross-Validation:** Estimación más robusta del error de generalización
- ➎ **Complejidad del modelo:** El grado del polinomio (o cualquier hiperparámetro) debe elegirse por CV
- ➏ **Overfitting:** El enemigo principal; regularización es la solución (próxima clase)

Referencias y Lecturas Recomendadas

Textos fundamentales:

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning* (2nd ed.). Springer. [\[Disponible gratis online\]](#)
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer. [\[Disponible gratis online\]](#)

Para economistas:

- Mullainathan, S., & Spiess, J. (2017). Machine Learning: An Applied Econometric Approach. *Journal of Economic Perspectives*, 31(2), 87-106.
- Athey, S., & Imbens, G. W. (2019). Machine Learning Methods That Economists Should Know About. *Annual Review of Economics*, 11, 685-725.

¡Gracias!

`s.neira10@uniandes.edu.co`