
# Técnicas de Remuestreo

## Máster en Data Science

**Mario Encinar, PhD, MAPFRE**
encinar@ucm.es

En estadística, se considera **remuestreo** a toda una variedad de métodos para

- **Estimar** la precisión de **estadísticos muestrales** utilizando **subconjuntos** de la muestra disponible (***jacknife***) o tomando sub-muestras con **reemplazamiento** de tal muestra (***bootstrap***).
- **Validar modelos** empleando **subconjuntos aleatorios** de la muestra disponible (*bootstrap* y ***cross-validation***).

El **bootstrap** se emplea para estimar la distribución muestral de un estimador mediante *remuestreos* con reemplazo de la muestra original.

El objetivo, habitualmente, es conseguir estimaciones robustas o intervalos de confianza para parámetros cuando no se puede emplear la inferencia paramétrica clásica

- porque no se dispone de hipótesis sobre la población (o, disponiendo de ellas, su veracidad es dudosa).
- porque el cálculo de las desviaciones estándar es muy complicado.

El **jacknife** se emplea para estimar el sesgo y la varianza de un cierto estadístico a partir de una muestra, recalculando la estimación de tal estadístico dejando fuera una o varias observaciones de la muestra cada vez.

En multitud de ocasiones, la estimación jacknife del estadístico converge hacia el verdadero valor del estadístico en algún sentido, lo que justifica su utilidad.

La validación cruzada (**cross-validation**) es una variación del jacknife que se emplea para validar el ajuste de un modelo predictivo.

Un subconjunto de la muestra de datos disponible (el conjunto de *training*) se emplea para ajustar un modelo y el resto de los datos (el conjunto de *validación*) se emplea para calcular valores predichos. La comparación de valores reales y valores predichos sirve para estimar el rendimiento *real* del modelo (*error de test*).

**Nota:** Para fijar ideas, nos centramos en problemas de regresión, aunque todas las técnicas que vamos a ver son aplicables en problemas de clasificación.

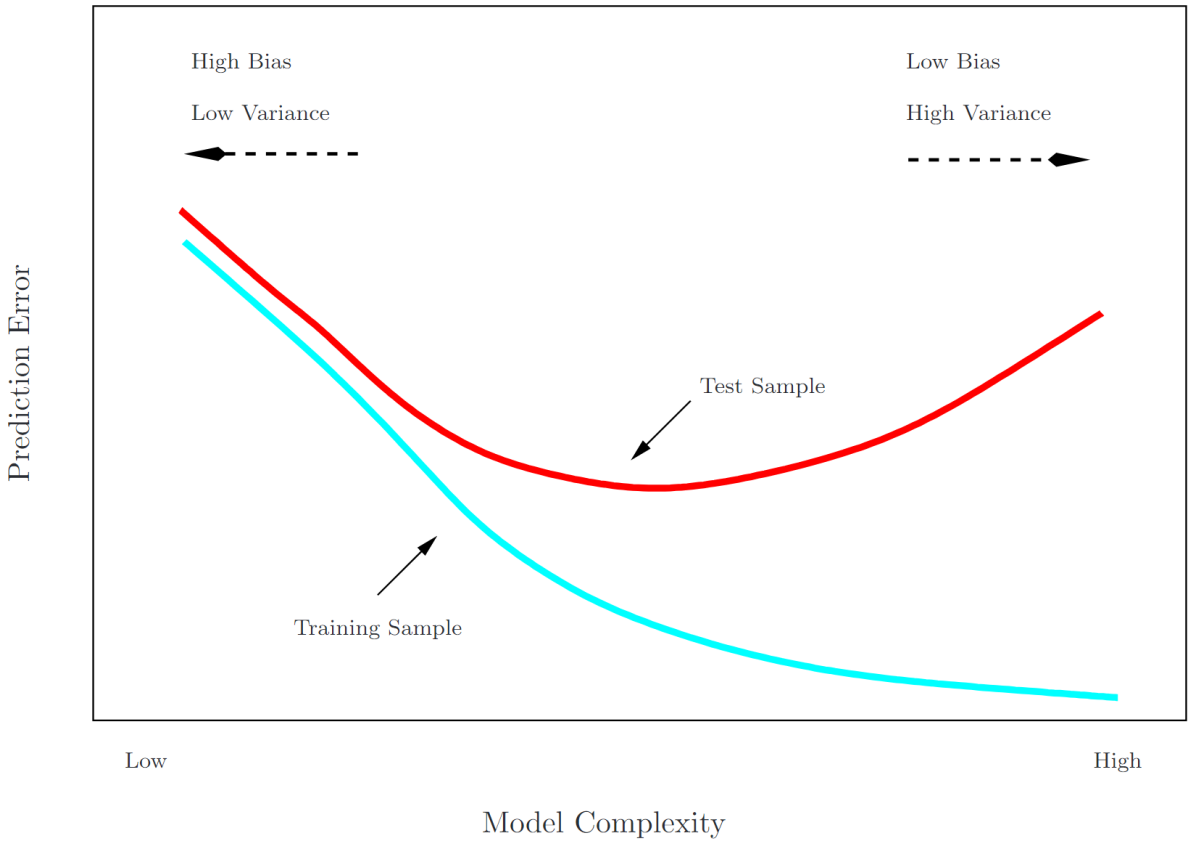
Cuando se construye un modelo $\hat{f}$ para predecir una cierta variable $Y$ en función de un conjunto de predictores $X = (X_1, \ldots, X_p)$, el objetivo es, casi siempre, minimizar una cierta función de error

$$Err(Y, \hat{Y}) = Err(Y, h(X)).$$

Más precisamente, lo que guía la construcción del modelo (elección de parámetros, estructura, etc.) es la búsqueda de un mínimo de tal función **sobre la muestra**.

Al error que produce el modelo sobre la muestra de datos disponible se le llama **error de training**.

Naturalmente, el fin último de la construcción de un modelo no es (o no debe ser) minimizar el error de training, sino cometer el mínimo error posible sobre la población (es decir, sus **predicciones** sobre datos *nuevos* deben ser todo lo precisas que sea posible). A este error sobre datos nuevos se le llama **error de test**.




Minimizar el error de training no garantiza que el error de test vaya a ser pequeño. De hecho, alcanzar errores de training muy pequeños suele ser señal de sobreajuste (**overfitting**): el modelo puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación con la función objetivo. El rendimiento del modelo sobre la muestra mejora mientras que su actuación con muestras nuevas empeora.

Para estimar el error de test, como sabemos, hay dos alternativas posibles:

- Ajustar el error de training, asumiendo que la muestra es representativa y suficientemente grande y que tiene determinadas propiedades estadísticas, para estimar el error de test.
- *Reservar* un cierto subconjunto de los datos de training, que no intervengan en la construcción del modelo, y aplicar el modelo sobre ese subconjunto.

Hoy nos centramos en la segunda alternativa (*cross-validation*).

La primera forma (o, más bien, un antecedente) de *cross-validation* es el uso de un **único conjunto de validación**.

1. Se divide, de forma aleatoria, el conjunto de datos en un conjunto de training y un conjunto de validación.
2. Se ajusta el modelo sobre el conjunto de training y se mide su rendimiento sobre el conjunto de validación.

Un parámetro a tener en cuenta es el tamaño relativo de las muestras de entrenamiento y test, que dependerá de la aplicación (y de factores externos).

Esta forma de validación es fácil de entender y sencilla de implementar, pero presenta dos problemas potenciales:

1. Las estimaciones del error de test pueden variar mucho dependiendo, precisamente, de qué observaciones *caigan* en el conjunto de training y cuáles en el de test.
2. Dado que sólo se emplea un subconjunto de los datos disponibles para entrenar el modelo, el error de test estimado será, probablemente, una sobre-estimación del verdadero error de test del modelo.

La ***leave-one-out cross-validation*** (en adelante, `LOOCV`) intenta resolver algunos de los problemas que presenta la forma de validación anterior.

Si $(x_1, y_1)$, $(x_2, y_2)$, $\ldots$, $(x_n, y_n)$ es la muestra disponible, podemos incluir **una sola observación** $(x_i, y_i)$ en el conjunto de validación, y emplear el resto de observaciones como conjunto de training. Si ajustamos un modelo $\hat{f}_i$ utilizando ese conjunto de training y, con él, se predice $\hat{y}_i = \hat{f}(x_i)$, se puede considerar que

$$\texttt{MSE}_i = (y_i - \hat{y}_i)^2$$

es una aproximación del `MSE` de test del modelo.

De hecho, se puede probar (bajo hipótesis razonables) que $\texttt{MSE}_i$ es un estimador insesgado del $\texttt{MSE}$ de test del modelo, pero tiene una varianza muy alta (ya que depende de una única observación).

La solución natural en este caso es tomar medias: **repetimos el proceso anterior para todas las observaciones** $(x_i, y_i)$, y estimamos el error de test mediante

$$\texttt{MSE}_{\texttt{LOOCV}} = \frac{1}{n}\sum_{i=1}^{n} \texttt{MSE}_i.$$

Con esto resolvemos los problemas de la primera estrategia de validación:

- La estimación del error de test no es estocástica.
- Los modelos se ajustan con un conjunto de datos de tamaño muy similar al total, así que el error estimado no deber sobre-estimar (mucho) el error de test real.

El problema fundamental que se presenta en este caso es que esta forma de validación es muy costosa desde el punto de vista computacional.

Una generalización sencilla de la LOOCV que es menos costosa consiste en tomar, en cada iteración, un conjunto más grande de observaciones en lugar de una única observación en el conjunto de test.

En $k$***-fold cross-validation*** (en adelante, $k$FCV)

1. Se reparten los datos disponibles en $k$ submuestras (*folds*) de tamaño comparable.
2. Para cada fold $i$, se entrena el modelo utilizando las $k-1$ submuestras restantes y se valida con ese fold como conjunto de test, y se anota el error $\texttt{MSE}_i$.
3. Se estima el error de test mediante

$$\texttt{MSE}_{k\texttt{FCV}} = \frac{1}{k}\sum_{i=1}^{k}\texttt{MSE}_i.$$

Una última alternativa, muy similar a la anterior, es la ***leave-group-out cross-validation*** (en adelante, `LGOCV`): fijados $p$ (la proporción de datos de training) y $k$ (el número de iteraciones):

1. Para cada $i = 1, 2, \ldots, k$, separamos de forma **aleatoria** los datos disponibles en una muestra de training de tamaño relativo $p$ frente al total, y utilizamos el resto como muestra de validación.
2. Ajustamos un modelo utilizando la muestra de training y validamos con los datos restantes, anotando el error $\texttt{MSE}_i$.
3. Se estima el error de test mediante

$$\texttt{MSE}_{\texttt{LGOCV}} = \frac{1}{k} \sum_{i=1}^{k} \texttt{MSE}_i.$$

En estadística, *bootstraping* se refiere a cualquier técnica o métrica que se base en muestreo aleatorio con reemplazo. Más precisamente, para estimar propiedades de un cierto estadístico dada una cierta muestra de la población (de la que no se conoce *gran cosa*):

1. Se consideran $B$ muestras tomadas de la muestra original, del mismo tamaño que ella, y con reemplazo.
2. Se calcula el valor del estadístico sobre cada una de esas $B$ muestras.
3. Se *estima* la distribución muestral del estadístico mediante la distribución del estadístico de las muestras de bootstrap.

**Recomendación:** Leer esto y esto.

**Referencias:**


- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R.* Springer. ISBN: 978-1-4614-7138-7.
- Hastie, T., Tibshirani, R., Friedman, J. (2009). *The elements of Statistical Learning. Data Mining, Inference and Prediction.* Springer. ISBN: 978-0-387-84858-7.