HENRY

Data Science

M4L6 | Optimización de modelos

soyhenry.com

Objetivos

- Evaluar la importancia de la validación cruzada, explorando métodos como K-Fold para mejorar la robustez y generalización de los modelos.
- Distinguir el concepto de sobreajuste y subajuste, analizando su impacto en el rendimiento de modelos de aprendizaje automático.
- Aplicar técnicas de optimización de hiperparámetros como Grid Search, Random Search y Optuna, integrando estrategias de regularización L1 y L2 para mejorar la precisión.

#TEMAS

Agenda

COMENCEMOS →

.01 Validación cruzada

.02 Importancia de la validación cruzada

.03 Selección de hiperparámetros y fine-tuning

.04 Avance de PI

→

<-->

¿Qué vimos en la lecture?

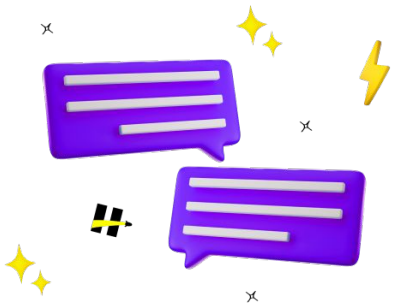
<01>

Validación cruzada

El objetivo de entrenar un modelo

- Entrenamos modelos para que acierten con datos que todavía no vieron.
- Ese objetivo —**generalizar**— guía todo el trabajo de métricas, validación y tuning.
- Un modelo que solo acierta en entrenamiento no sirve: debe funcionar en casos nuevos.
- La clave es evaluar correctamente su capacidad de generalización.

Underfitting y Overfitting



- **Underfitting:** el modelo es rígido, pasa por alto patrones y tiene alto sesgo.
- **Overfitting:** el modelo es tan flexible que memoriza ejemplos y no generaliza.
- En el primer caso, falla por simplicidad; en el segundo, por exceso de complejidad.
- Ambos extremos son problemáticos: el desafío es lograr el punto medio.

Curvas de error y equilibrio sesgo-varianza

- **Underfitting:** errores altos tanto en entrenamiento como en validación.
- **Overfitting:** error bajo en train pero alto en validación.
- El punto ideal se logra cuando el modelo tiene capacidad justa para explicar sin perseguir ruido.
- Ese equilibrio minimiza el error total de generalización.

Cómo evitar el overfitting

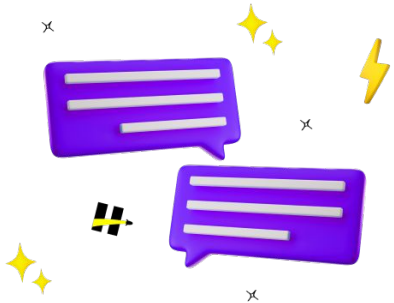
- **Más datos:** más volumen y diversidad reducen la varianza del modelo.
- Los nuevos ejemplos ayudan a distinguir patrones del ruido.
- En datos tabulares, “más datos” implica nuevas fuentes o mejor etiquetado.
- También puede incluir depurar outliers o registros erróneos.
- En datos no tabulares, se puede aplicar augmentation controlado.

- **Menos características:** cada feature irrelevante aumenta el riesgo de perseguir ruido.
- Quitar columnas redundantes o de baja importancia mejora la generalización.
- Técnicas típicas: eliminar nulos, detectar colinealidad extrema o usar selección basada en modelos.
- **Reducir la complejidad:** limitar profundidad o tamaño de hojas en árboles.
- El modelo debe capturar las tendencias, no cada detalle menor.

Cómo evitar el underfitting

- A veces el problema no es el algoritmo, sino la falta de variables informativas.
- Si el modelo no tiene datos suficientes para explicar el fenómeno, no puede aprender.
- **Ingeniería de variables:** crear nuevas señales a partir de las existentes.
- Ejemplos: interacciones (edad×ingreso), transformaciones logarítmicas o cuadráticas.
- También ratios, agregaciones por entidad o tiempo, y lags en series temporales.
- **Aumentar la complejidad del modelo:** pasar de lineales a árboles, RF o boosting.
- **Relajar la regularización:** un C demasiado bajo restringe el aprendizaje.
- **Aumentar parámetros de ensambles:** más árboles o profundidad permiten mayor expresividad.
- El ajuste debe ser incremental: subir capacidad y observar si la validación mejora.
- La estabilidad entre pliegues es el criterio final de confianza.

Identificar overfitting vs underfitting



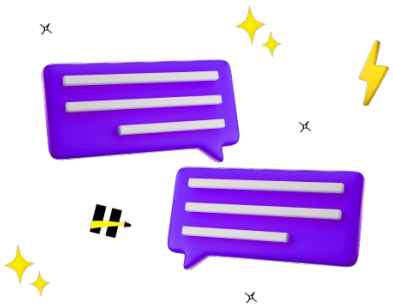
- Modelo A: Accuracy(train)=0.95, test=0.72 → **overfitting**, rinde bien solo en train.
- Modelo B: Accuracy(train)=0.81, test=0.79 → más equilibrado.
- Modelo A pierde capacidad de generalización por exceso de flexibilidad.
- Modelo B muestra métricas más estables entre particiones.
- Respuesta correcta: **b)** el modelo A muestra signos de overfitting.

Importancia de la validación cruzada

- Un modelo puede parecer bueno solo por azar si la partición no es representativa.
- El esquema clásico (70/15/15) tiene alta varianza en estimación.
- Además, puede sobreajustarse al set de validación si se usa siempre el mismo.
- La validación cruzada promedia resultados y ofrece una estimación más robusta.
- Permite aprovechar mejor los datos disponibles.

Split clásico vs validación cruzada

- **Split clásico:** útil con datasets grandes o para prototipos rápidos.
- **Validación cruzada:** divide en K pliegues y alterna el rol de validación.
- Reduce la sensibilidad al azar de una sola partición.
- Mejora la comparación entre configuraciones de modelos.
- Cada configuración se mide bajo el mismo protocolo.



Elección de K y forma de reportar resultados

- K=5 → balance entre precisión y costo computacional.
- K=10 → preferible con pocos datos, más precisión pero mayor tiempo.
- Cada modelo entrena con (K−1)/K del total de datos.
- Reportar **media ± desvío estándar** de las métricas.
- Ejemplo: F1 = 0.82 ± 0.03 muestra estabilidad entre pliegues.

Validación cruzada K-Fold estándar

- Divide el dataset en K bloques aleatorios de tamaño similar.
- Cada pliegue actúa una vez como validación y las demás como entrenamiento.
- Útil en regresión o clasificación balanceada sin estructuras especiales.
- Limitación: si hay desbalance o correlación por entidad, la estimación se sesga.
- Puede generar pliegues no representativos del conjunto total.

Validación cruzada estratificada (Stratified K-Fold)

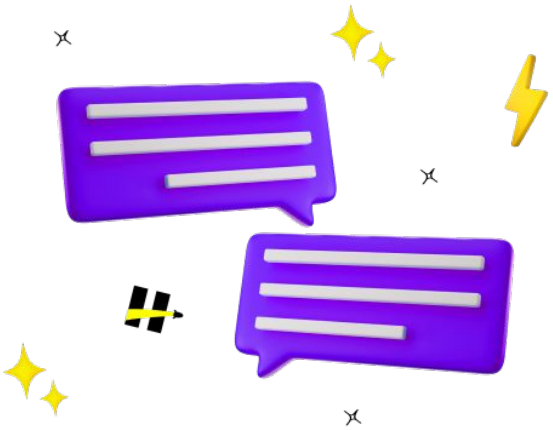
- Mantiene proporciones de clase similares en cada pliegue.
- Evita que algún pliegue se quede sin ejemplos positivos.
- Reduce la variabilidad de las métricas respecto a K-Fold estándar.
- Es la opción por defecto en clasificación con clases desbalanceadas.
- Garantiza que la evaluación sea más representativa.

Validación cruzada agrupada (Grouped K-Fold)

- Diseñada para datos con grupos naturales (usuarios, pacientes, empresas).
- Un mismo grupo **no puede** aparecer en train y validación a la vez.
- Evita fugas de información y métricas artificialmente infladas.
- Permite estimar si el modelo generaliza a entidades no vistas.
- Es fundamental cuando existen múltiples observaciones por grupo.

Validación cruzada para series temporales (Time Series Split)

- Respeta el orden temporal: no mezcla pasado y futuro.
- Usa ventanas expansivas o deslizantes para entrenar y validar.
- Evalúa si el modelo puede extrapolar hacia adelante.
- Útil en contextos con estacionalidad o cambios de tendencia.
- Asegura una validación coherente con la realidad temporal.



<02>

Importancia de la validación cruzada

Selección de métricas de evaluación

- La métrica define qué significa “mejor modelo”.
- Accuracy puede ser engañosa si el dataset está desbalanceado.
- Se necesitan métricas específicas: Precision, Recall, F1, ROC-AUC, PR-AUC.
- La elección depende del costo de los errores y del objetivo del negocio.
- Elegir bien la métrica evita optimizar hacia un resultado irrelevante.

Threshold o umbral de decisión

- Los modelos clasificadores devuelven probabilidades, no etiquetas.
- El valor de corte (por defecto 0.5) define cuándo predecir “sí” o “no”.
- Ajustar el threshold cambia el equilibrio entre Precision y Recall.
- Se elige según costos, restricciones o curvas ROC y PR.
- Es la conexión entre las predicciones y las decisiones reales.

Métricas offline vs online

- **Offline:** se calculan en laboratorio (F1, RMSE, AUC) con train/valid/test.
- **Online:** miden impacto real en negocio (conversión, churn, fraude).
- Las offline son más baratas y rápidas, pero aproximadas.
- Las online requieren experimentos (A/B tests) y confirman valor real.
- Una mejora offline debería anticipar una mejora online.

<03>

Selección de hiperparámetros y fine-tuning

Selección de hiperparámetros y fine-tuning

- Los modelos tienen **hiperparámetros** que definen su comportamiento (profundidad, regularización, cantidad de árboles, etc.).
- No se aprenden automáticamente: debemos elegirlos manualmente.
- El objetivo es encontrar la combinación que maximice el rendimiento del modelo.
- La validación cruzada permite comparar configuraciones de forma confiable.
- Optimizar hiperparámetros mejora la capacidad de generalización.

Grid Search: exploración exhaustiva

- Construye una “grilla” con todas las combinaciones posibles de hiperparámetros.
- Prueba cada combinación y selecciona la que logra mejor resultado.
- Ventaja: asegura revisar todas las configuraciones definidas.
- Desventaja: es costosa si el número de parámetros o valores crece.
- Útil cuando el espacio de búsqueda es acotado y el entrenamiento es rápido.

Random Search y Optuna

- **Random Search:** elige combinaciones al azar dentro de rangos definidos.
- Es más eficiente que Grid Search cuando hay muchos hiperparámetros.
- **Optuna:** optimizador adaptativo que aprende de intentos previos.
- Propone nuevas configuraciones prometedoras y descarta las malas tempranamente (*pruning*).
- Encuentra resultados de alta calidad con menos pruebas y menor costo.

Regularización L1 y L2 en el tuning

- Regularizar evita **overfitting** penalizando coeficientes grandes.
- **L1 (Lasso):** fuerza algunos coeficientes a cero → selección automática de variables.
- **L2 (Ridge):** suaviza coeficientes sin anularlos → reduce varianza.
- El grado de penalización se controla con hiperparámetros (α o C).
- Ajustar estos valores en Grid, Random o Optuna mejora el equilibrio entre sesgo y varianza.

<DATA SCIENCE/>

Vayamos a la práctica

Avance de PI

Consigna

En esta segunda etapa del proyecto, **FinanceGuard** busca avanzar hacia modelos de mayor complejidad y rendimiento para la predicción de churn. El objetivo es comparar el desempeño de distintos algoritmos de **ensemble learning**, aplicando **Random Forest** y métodos de **Gradient Boosting** como **XGBoost**, **LightGBM** y **CatBoost**. El estudiante deberá ajustar parámetros clave (*learning_rate, max_depth, n_estimators*), aplicar estrategias de **validación cruzada** —como *StratifiedKFold* o *StratifiedGroupKFold*— y evaluar los modelos con métricas robustas ante desbalanceo, como **ROC-AUC**, **PR-AUC** y **F1-score**.

Además, se desarrollará un **ensamble por Stacking**, combinando modelos base de boosting con una **Regresión Logística regularizada** como meta-learner, y comparando su desempeño con métodos individuales. La **optimización de hiperparámetros** se aplicará específicamente al modelo **XGBoost**, utilizando **Grid Search**, mientras que la optimización bayesiana con **Optuna** será opcional. El entregable final, **2_GradientBoosting_Optimizacion.ipynb**, deberá incluir la comparación de resultados, la interpretación de la importancia de variables y las conclusiones sobre el modelo con mejor capacidad predictiva para reducir la tasa de abandono de clientes.

Tareas a realizar

1. **Random Forest y Gradient Boosting:**
   - **Random Forest y XGBoost:**
     - Parámetros clave: learning_rate, max_depth, n_estimators
     - Early stopping y otros parámetros para evitar overfitting
     - Feature importance y gain
     - Regularización alpha y lambda
   - **LightGBM:**
     - Optimización para velocidad
     - Parámetros específicos: num_leaves, min_data_in_leaf
     - Categorical feature handling
   - **CatBoost:**
     - Manejo automático de categóricas

2. **Validación cruzada y métricas especializadas:**
   - **Estrategias de CV:**
     - StratifiedKFold para datos desbalanceados
     - GroupKFold para datos agrupados
     - StratifiedGroupKFold para datos desbalanceados y agrupados
     - TimeSeriesSplit para datos temporales

Métricas de evaluación:

- Accuracy, Recall, y Precision
- PR-AUC para datos desbalanceados
- ROC-AUC para datos desbalanceados
- Business metrics personalizada

Tareas a realizar

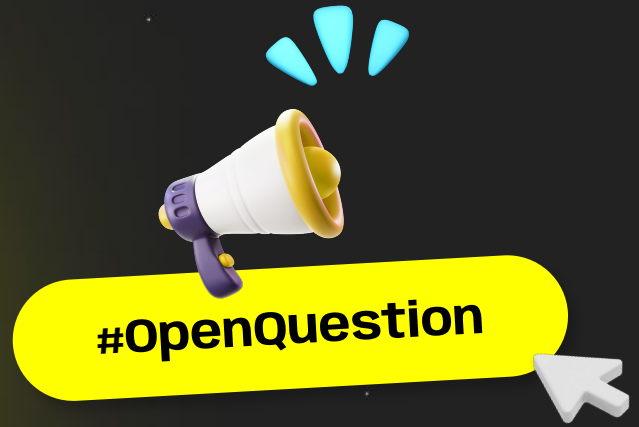
3. **Stacking y Blending:**

   - **Stacking:**
     - Nivel 1: XGBoost, LightGBM, CatBoost
     - Meta-learner: Regresión Logística regularizada
     - Cross-validation para evitar overfitting
   - **Blending:**
     - Holdout set para meta-learner
     - Ponderación óptima de modelos

4. **Optimización de hiperparámetros avanzada:**

   - **Bayesian Optimization con Optuna:**
     - Objective function personalizada
     - Visualización de importancia de hiperparámetros
   - **Grid Search y Random Search:**
     - Comparación de eficiencia
     - Nested cross-validation

HENRY



¿Preguntas?

HENRY

¡Muchas gracias!

→ soyhenry.com