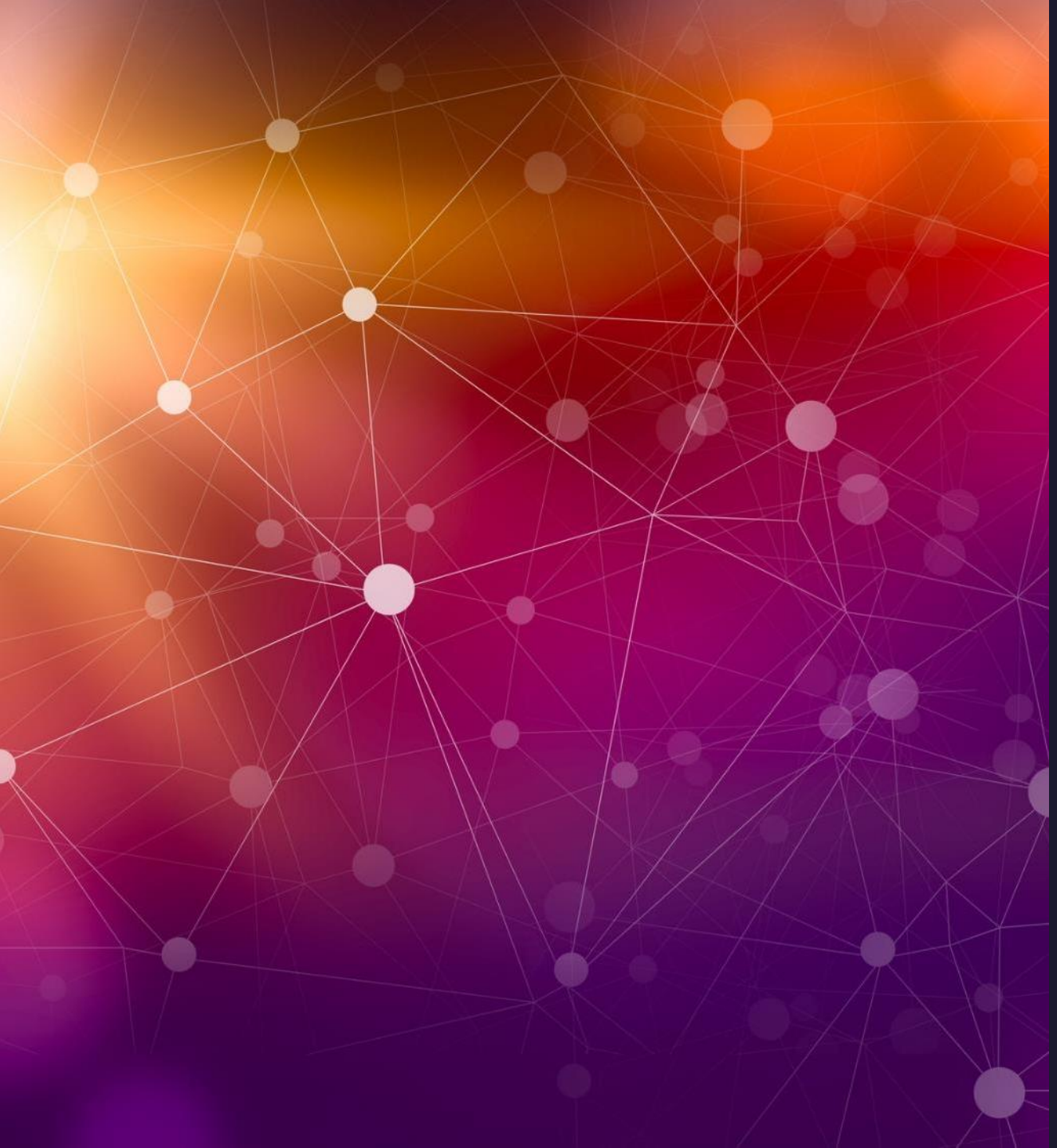


Universidad Tecnológica de Panamá
Facultad de Ingeniería Eléctrica
Maestría en Ingeniería Eléctrica

Semana 4

Introducción a Machine Learning

El problema del Data Split en Machine Learning

Cuando se entrena y evalúa un modelo de *machine learning*, es común dividir el conjunto de datos en dos partes: una para el **entrenamiento** del modelo y otra para su **evaluación**.

Una práctica habitual es utilizar un **80% de los datos para entrenamiento** y el **20% restante para prueba**, aunque también se emplean proporciones como **70/30** o **90/10**, dependiendo del caso.

El problema del Data Split en Machine Learning

Sin embargo, la forma en que se realiza esta división puede introducir un **sesgo significativo** (*bias*) en los resultados.

La selección específica de los datos que se usan para entrenar y probar puede afectar el rendimiento observado del modelo, especialmente si la muestra no es representativa o si las clases no están equilibradas.

Por ello, es fundamental aplicar técnicas de validación más robustas, como la **validación cruzada**, para obtener evaluaciones más confiables y generalizables.

Validación cruzada

Definición

Es un método que divide los datos en partes para entrenar y probar el modelo varias veces, asegurando que no dependa demasiado de un solo conjunto de datos.

Tipos de validación cruzada

- Hold-Out (Validación simple)
- K-Fold Cross-Validation
- Stratified K-Fold
- Leave-One-Out Cross-Validation (LOOCV)
- Leave-P-Out Cross-Validation
- Repeated K-Fold
- Nested Cross-Validation

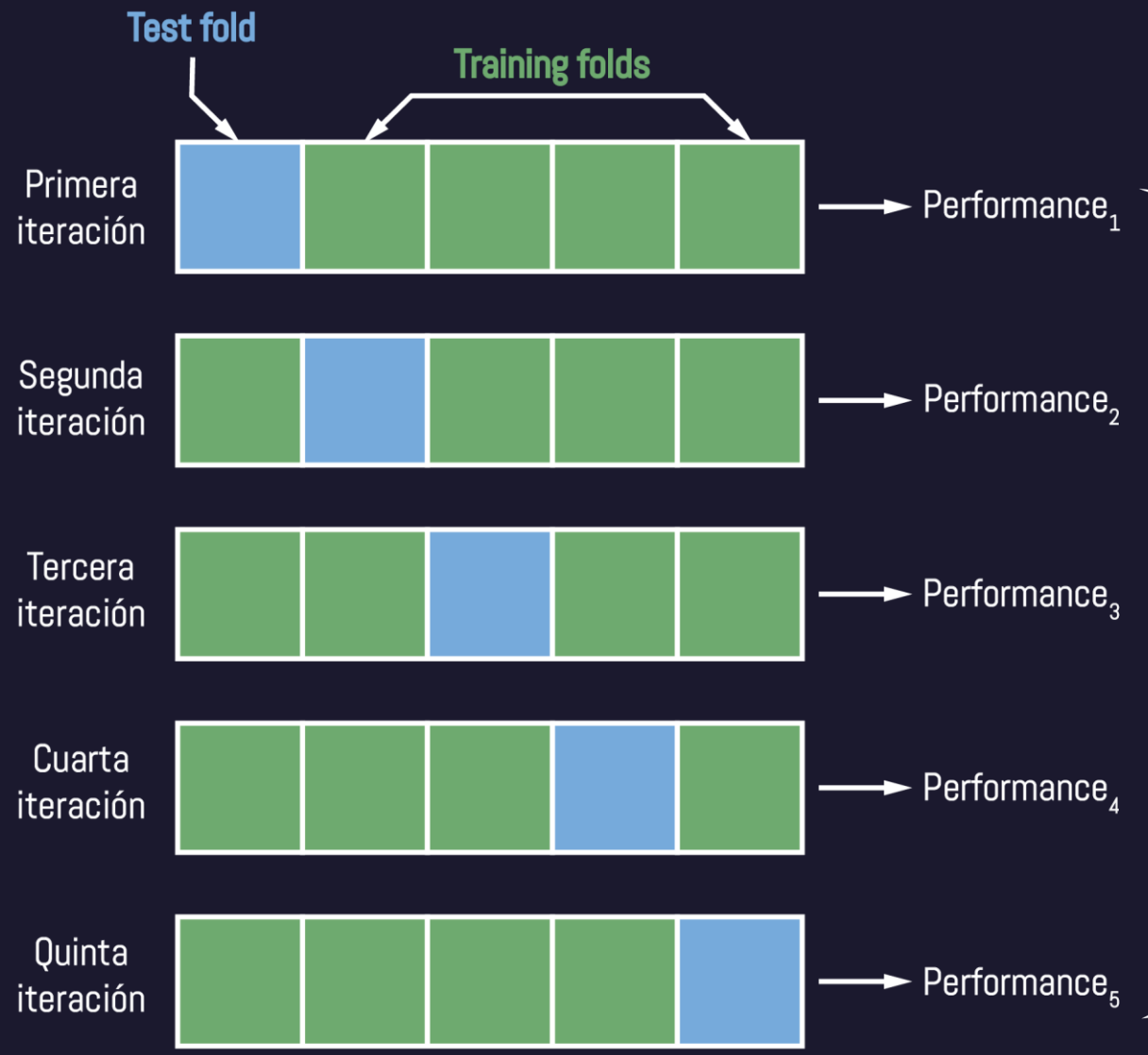
Hold-Out (Validación simple)

- Se divide el dataset una sola vez (por ejemplo 80%/20%).
- Muy común y rápida, pero sufre de **alta varianza**: el resultado puede depender mucho de la partición aleatoria.

K-Fold Cross-Validation

- Divide los datos en **K partes (folds)**.
- El modelo se entrena en K−1 partes y se prueba en la restante.
- Se repite K veces cambiando la parte de prueba.
- Se promedian los resultados finales.
- **Ventaja**: Balance entre sesgo y varianza.

K-Fold Cross Validation



$$\text{Overall Performance} = \frac{1}{5}\sum_{i=1}^{5} \text{Performance}_i$$

Stratified K-Fold

- Igual que K-Fold, pero asegura que **cada fold mantenga la proporción de clases** (en clasificación).
- Útil cuando hay **desbalance de clases** (por ejemplo, 90% clase A y 10% clase B).

Leave-One-Out Cross-Validation (LOOCV)

- Cada instancia es usada una vez como conjunto de prueba.
- El modelo se entrena con todos los demás datos.
- Muy preciso, pero **computacionalmente costoso** con grandes datasets.

Leave-P-Out Cross-Validation

- Similar al LOOCV, pero se dejan P muestras fuera para prueba en cada iteración.
- Poco usado por su alto costo computacional.

Repeated K-Fold

- Repite la validación K-Fold varias veces con particiones distintas.
- Proporciona una evaluación más estable.
- Se promedian todos los resultados de todas las repeticiones.

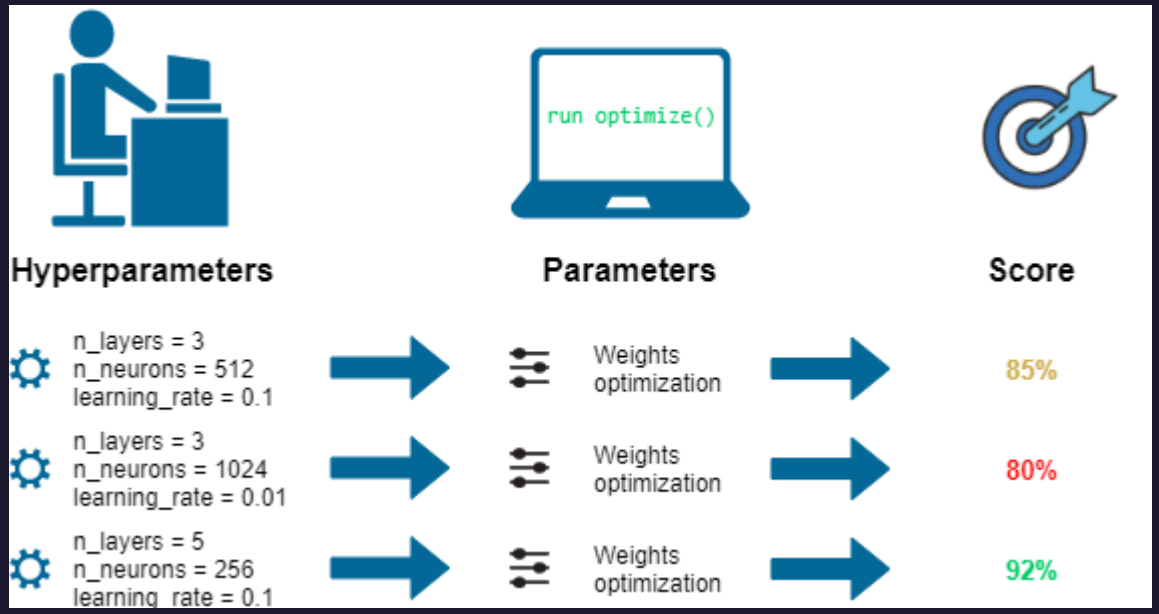
Nested Cross-Validation

- Usa dos niveles de validación cruzada: uno para **evaluar el modelo** y otro para **seleccionar hiperparámetros**.
- Evita el sobreajuste en la selección de modelos.
- Ideal para **comparar modelos o pipelines complejos**.

Optimización de hiperparámetros

Optimización de hiperparámetros en machine learning

La optimización de hiperparámetros es el proceso de buscar automáticamente los mejores valores posibles para los hiperparámetros de un modelo, con el fin de mejorar su rendimiento.



¿Qué son los hiperparámetros?

Son **configuraciones externas** al modelo que no se aprenden directamente de los datos.

Ejemplos:

- Número de vecinos en KNN (n_neighbors)
- Tipo de regularización en modelos lineales (penalty, C)
- Valor de suavizado en Naive Bayes (alpha)
- Profundidad máxima de un árbol de decisión (max_depth)

Técnicas de optimización de hyperparámetros

- Grid Search
- Random Search
- Validación cruzada (Cross-Validation)
- Búsqueda Bayesiana (Bayesian Optimization)

Grid Search

Consiste en definir un conjunto de valores posibles para cada hiperparámetro y evaluar **todas las combinaciones posibles**. Es una búsqueda exhaustiva y sistemática que garantiza probar cada opción dentro de la grilla, pero puede volverse muy costosa cuando hay muchos hiperparámetros o valores posibles.

Random Search

Selecciona combinaciones de hiperparámetros de forma **aleatoria** dentro de los rangos definidos. No explora todas las combinaciones, pero es más eficiente en tiempo y puede encontrar buenas soluciones con menos pruebas, especialmente cuando solo unos pocos hiperparámetros influyen fuertemente en el rendimiento.

Validación cruzada (Cross-Validation)

No es una técnica de búsqueda por sí sola, sino una forma de **evaluar** cada combinación de hiperparámetros. Divide los datos en varios subconjuntos (folds), entrena el modelo en algunos y lo evalúa en los restantes. Se repite este proceso para cada combinación y se promedian los resultados para obtener una evaluación robusta.

Búsqueda Bayesiana (Bayesian Optimization)

Utiliza modelos probabilísticos para predecir qué combinaciones de hiperparámetros podrían funcionar mejor basándose en los resultados anteriores. A diferencia de Grid o Random Search, no elige combinaciones al azar, sino que aprende de las pruebas anteriores para dirigir la búsqueda de manera más eficiente e inteligente.

Optimización Bayesiana

¿Qué es la optimización bayesiana?

- La optimización bayesiana es un proceso que:
- Construye un **modelo probabilístico** (surrogate model) que aproxima la relación entre los hiperparámetros y el rendimiento del modelo.
- Usa ese modelo para predecir qué combinaciones de hiperparámetros podrían dar el mejor resultado en el futuro.
- Balancea:
  - ✓**Exploración** (probar combinaciones nuevas).
  - ✓**Explotación** (refinar combinaciones que ya dieron buenos resultados).
- Esto permite encontrar **buenos hiperparámetros con menos intentos** que Grid Search o Random Search.

Ventajas

- Encuentra **buenos hiperparámetros** con menos evaluaciones que Grid Search o Random Search.
- Es **eficiente** cuando el costo de entrenar el modelo es alto.
- Aprende de los resultados previos para **dirigir la búsqueda** de forma más inteligente.
- Balancea de forma natural la **exploración** y la **explotación**.
- Permite trabajar con espacios de búsqueda **continuos y discretos** a la vez.
- Puede optimizar cualquier métrica (AUC, precisión, MSE, etc.).

Desventajas

- Puede ser difícil de implementar o configurar correctamente en problemas complejos.
- Su rendimiento disminuye cuando hay muchos hiperparámetros (alta dimensión).
- Requiere más memoria y procesamiento que métodos más simples.
- No es tan efectiva cuando los hiperparámetros son puramente categóricos o tienen pocas combinaciones posibles.

Herramientas disponibles

- **scikit-optimize (skopt)**

  Librería ligera y fácil de usar; incluye BayesSearchCV para integración directa con Scikit-learn.

- **Optuna**

  Framework flexible y rápido; permite definir espacios de búsqueda complejos y optimizar funciones arbitrarias.

- **Hyperopt**

  Permite búsqueda bayesiana con TPE (Tree-structured Parzen Estimator); buena integración con frameworks de deep learning.

- **GPyOpt**

  Basado en Gaussian Processes; enfocado en optimización bayesiana pura.

- **Ray Tune**

  Framework escalable que combina búsqueda bayesiana con otras estrategias y permite paralelización en clusters.

Otros algoritmos de optimización

Algoritmos genéticos (Genetic Algorithms)

- Simulan la evolución natural: selección, cruce y mutación de soluciones.
- Muy útiles para espacios grandes, no lineales o con múltiples óptimos.

PSO – Particle Swarm Optimization

- Basado en el comportamiento de enjambres (pájaros o peces).
- Cada “partícula” representa una solución que se mueve por el espacio buscando el mejor resultado.
- Rápido y eficiente para funciones continuas.

Otros algoritmos de optimización

- Simulated Annealing
- Differential Evolution
- Hill Climbing / Random Walk
- Hyperband / Successive Halving

¿Cuándo usar estos algoritmos?

- Cuando el espacio de búsqueda es **complejo o no diferenciable**
- Cuando hay **muchas variables o restricciones**
- Cuando se busca **robustez** y no solo velocidad