Introducción:

En este proyecto hemos trabajado con un algoritmo de *ensamble secuencial*, una técnica que combina varios modelos simples también llamados estimadores débiles para crear un modelo más robusto y preciso.

A diferencia de otros enfoques que entrenan todos los modelos al mismo tiempo, el ensamble secuencial lo hace de forma progresiva: cada nuevo modelo intenta corregir los errores que cometieron los anteriores.

Lo interesante es que nosotros no usamos ninguna librería avanzada de *boosting* como XGBoost o LightGBM, sino que implementamos todo desde cero, para entender cómo funcionan este tipo de algoritmos por dentro.

Probamos nuestro modelo en dos conjuntos de datos reales: uno para predecir precios de viviendas y otro que intenta predecir la progresión de la enfermedad de Parkinson a partir de datos médicos.

La idea fue ver cómo se comporta el ensamble con distintos modelos base, como árboles de decisión o regresión lineal, y comparar sus resultados."

Objetivo del proyecto:

El objetivo principal de este proyecto fue entender y construir un algoritmo de ensamble secuencial desde cero, sin depender de librerías externas que ya lo tengan implementado.

Queríamos aprender cómo se combinan modelos simples para mejorar el rendimiento, cómo se calcula el error en cada iteración, y cómo se ajustan los pesos de cada modelo dentro del ensamble.

Además, buscamos comprobar si realmente mejora los resultados respecto a entrenar un único modelo, especialmente en conjuntos de datos reales, donde los errores pueden tener impacto en decisiones importantes.

Conceptos clave:

Un algoritmo de ensamble secuencial es un tipo de modelo de aprendizaje automático que no entrena un solo modelo, sino una secuencia de modelos.

Cada modelo intenta corregir los errores que cometió el modelo anterior.

Es decir, se entrena uno, luego otro que mejora los fallos del anterior, y así sucesivamente.

Al final, todos los modelos se combinan para hacer una predicción conjunta, que normalmente es mucho mejor que la de un solo modelo.

Implementación:

La implementación del meta-modelo se basa en el principio de boosting: entrenar secuencialmente varios modelos sobre los errores que van dejando los anteriores. Empezamos con una predicción inicial igual a la media de la variable objetivo, y en cada iteración añadimos un estimador entrenado sobre los modelos actuales. Además, incluimos un mecanismo opcional de parada temprana: si durante varias iteraciones seguidas no mejora el rendimiento en un conjunto de validación, se detiene el entrenamiento y se guardan solo los mejores modelos.

Hiperparámetos y entrenamiento:

El entrenamiento se hizo con validación cruzada 5-fold para evaluar el rendimiento de cada combinación de hiperparámetros. En los árboles de decisión, ajustamos el número de estimadores, la tasa de aprendizaje, el tamaño de muestreo y la profundidad máxima. En la regresión lineal, el proceso fue similar, pero ajustando si se incluía término independiente. Usamos una búsqueda manual sencilla, probando distintas combinaciones dentro de un rango definido en los requisitos del pdf.

Resultados R^2:

- House:
  - En DecisionTreeRegressor tiene un rendimiento bajo debido al que el dataset es más pequeño y el modelo tiende a sobreajustar.
  - En LinearRegression tiene un comportamiento más estable ya que el precio de la vivienda se interpreta de mejor forma linealmente.
- Parkinson:
  - En DecisionTreeRegressor es muy eficaz captando relaciones lineales no complejas.
  - En LinearRegression no es nada eficaz ya que las relaciones del conjunto no son lineales.

En resumen, los árboles funcionan mejor en datos complejos y grandes, mientras que la regresión lineal se adapta mejor a relaciones más simples y datasets más pequeños.