Optimización sin Derivadas para Funciones de Caja Negra: Un Estudio Aplicado en el Ajuste de Hiperparámetros

Fiorella Yannet Paredes Coaguila
Universidad Nacional del Altiplano
Puno, Perú

***Abstract*—La optimización de hiperparámetros en algoritmos de aprendizaje automático representa un desafío computacional significativo debido a la naturaleza de caja negra de las funciones objetivo. Los algoritmos de aprendizaje automático son cajas negras complejas que crean problemas de optimización desafiantes, con funciones objetivo no suaves, discontinuas y con variaciones impredecibles en el costo computacional. Este estudio evalúa cuatro métodos de optimización sin derivadas: Tree-structured Parzen Estimator (TPE), Random Search, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) y Quasi-Monte Carlo (QMC) para el ajuste de hiperparámetros en modelos Random Forest y XGBoost. Utilizando el conjunto de datos House Prices, se realizaron experimentos controlados con 50 evaluaciones por método y 3 ejecuciones independientes. Los resultados demuestran que TPE, como algoritmo de optimización bayesiana, permite actualizar creencias iniciales sobre hiperparámetros óptimos de manera principiada y logró el mejor desempeño promedio con un RMSE de 29,803.67 ± 316.86 para Random Forest. El análisis estadístico reveló diferencias significativas entre métodos, validando la superioridad de TPE sobre métodos tradicionales como Random Search para este tipo de problemas de optimización.**

***Index Terms*—optimización sin derivadas, ajuste de hiperparámetros, funciones de caja negra, TPE, aprendizaje automático**

I. Introducción

El ajuste de hiperparámetros constituye una etapa crítica en el desarrollo de modelos de aprendizaje automático, determinando tanto la eficiencia del proceso de entrenamiento como la calidad del modelo resultante. Los hiperparámetros usualmente controlan tanto la eficiencia del proceso de entrenamiento del modelo como la calidad del modelo resultante [1]. A diferencia de los parámetros del modelo que se aprenden durante el entrenamiento, los hiperparámetros deben establecerse previamente y su selección óptima requiere métodos de optimización especializados.

La complejidad de este problema radica en que las funciones objetivo involucradas tienden a ser no suaves, altamente no convexas, ruidosas y costosas de evaluar. A menudo, estas funciones se comportan como cajas negras, sin una estructura matemática clara que pueda ser aprovechada para la optimización. Además, incluyen variables continuas, categóricas y discretas, lo que dificulta aún más la búsqueda de soluciones óptimas. Las evaluaciones de la función pueden fallar por diversas razones, incluyendo errores numéricos, restricciones ocultas o fallas de hardware, lo cual introduce desafíos adicionales en la exploración del espacio de búsqueda [2].

Tradicionalmente, el método para la optimización de hiperparámetros ha sido la búsqueda en cuadrícula, que es simplemente una búsqueda exhaustiva a través de un subconjunto especificado manualmente del espacio de hiperparámetros. Sin embargo, este enfoque presenta limitaciones significativas en términos de eficiencia computacional y escalabilidad [12].

Los métodos de optimización sin derivadas emergen como una alternativa prometedora para abordar estas limitaciones [2]. Estos algoritmos no requieren información sobre gradientes de la función objetivo, lo que los hace especialmente adecuados para funciones de caja negra. Entre los métodos más destacados se encuentran el Tree-structured Parzen Estimator (TPE), que emplea optimización bayesiana para construir modelos probabilísticos de la función objetivo [3], y métodos evolutivos como CMA-ES que utilizan estrategias de adaptación poblacional [4].

Asimismo, el Tree-structured Parzen Estimator (TPE) se ha consolidado como una de las estrategias más eficaces dentro del enfoque bayesiano. Este método modela las distribuciones de hiperparámetros que conducen a buenas y malas observaciones, permitiendo priorizar regiones del espacio con mayor probabilidad de mejora. Recientes trabajos han descompuesto sus componentes internos para entender mejor cómo influye cada uno en el desempeño empírico del algoritmo [3].

El objetivo principal de esta investigación es evaluar comparativamente la efectividad de cuatro métodos de optimización sin derivadas en el contexto del ajuste de hiperparámetros para algoritmos de aprendizaje automático. Específicamente, se busca determinar cuál método proporciona la mejor convergencia en términos de error de predicción mientras mantiene eficiencia computacional razonable.

II. Materiales y Métodos

A. Conjunto de Datos

El estudio utilizó el conjunto de datos House Prices de Kaggle, un problema de regresión ampliamente reconocido en la comunidad de aprendizaje automático. Este conjunto contiene información sobre propiedades residenciales en Ames, Iowa, con 1,460 observaciones y 81 características que incluyen variables tanto numéricas como categóricas.

Las características del conjunto de datos abarcan aspectos arquitectónicos (área total, número de habitaciones, año de construcción), características del terreno (área del lote, topografía), condiciones de la propiedad (calidad general, estado de conservación) y servicios (sistemas de calefacción, garajes, piscinas). La variable objetivo corresponde al precio de venta de la vivienda, con un rango desde $34,900 hasta $755,000.

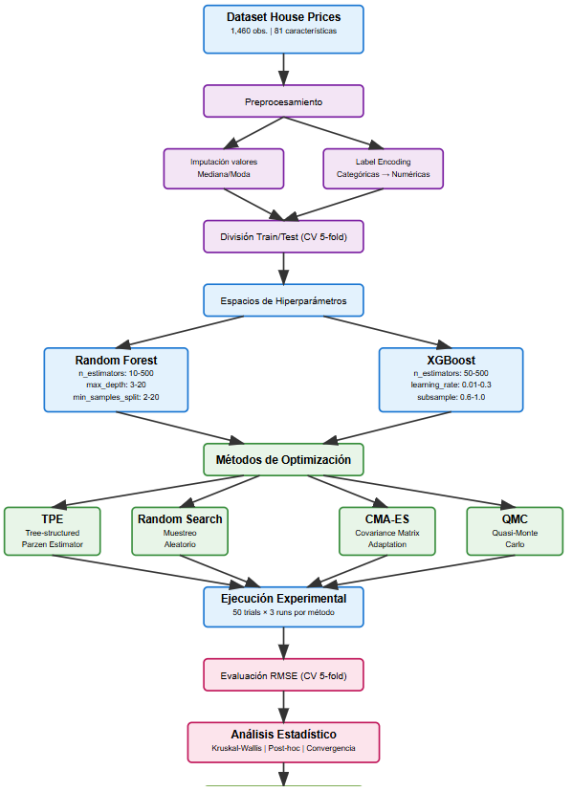

Fig. 1. Diagrama de Flujo metodológico.

B. Estadisticos Descriptivos

TABLE I
RESUMEN DE ESTADÍSTICOS DESCRIPTIVOS RELEVANTES

Variable	Media	Rango	Desv. Est.
SalePrice	$180,921	34,900 – 755,000	79,443
GrLivArea	1,515.5	334 – 5,642	–
LotArea	10,516.8	1,300 – 215,245	–
YearBuilt	1971.3	1872 – 2010	–
OverallQual	6.1	1 – 10	–
GarageCars	1.8	0 – 4	–

La base de datos contiene información de 1,460 viviendas con 80 características (37 numéricas y 43 categóricas), siendo el precio de venta (*SalePrice*) la variable objetivo. El precio promedio fue de $180,921 con una alta dispersión (desviación estándar de $79,443), lo que refleja una heterogeneidad significativa en el mercado inmobiliario. Variables como *GrLivArea* y *LotArea* presentan rangos amplios, indicando la presencia de viviendas con características extremas. La variable *YearBuilt* muestra un rango de construcción entre 1872 y 2010, mientras que la calidad general (*OverallQual*) se concentra alrededor de un valor medio de 6.1. Estas condiciones justifican el uso de métodos de optimización sin derivadas sobre una función objetivo ruidosa, costosa de evaluar y definida en un espacio de búsqueda mixto.

C. Preprocesamiento de Datos

El preprocesamiento se realizó siguiendo un protocolo estandarizado que incluye los siguientes pasos: Para variables numéricas, los valores faltantes se imputaron utilizando la mediana de cada característica, proporcionando robustez ante valores atípicos. Las variables categóricas con valores faltantes se completaron usando la moda de cada variable.

La codificación de variables categóricas se realizó mediante Label Encoding, transformando categorías textuales en valores numéricos ordinales. Este enfoque permite que los algoritmos de optimización trabajen exclusivamente con espacios de búsqueda numéricos.

D. Modelos de Aprendizaje Automático

Se seleccionaron dos algoritmos representativos para evaluar los métodos de optimización:

Random Forest

Este algoritmo ensemble combina múltiples árboles de decisión mediante votación promedio [6]. Los hiperparámetros optimizados incluyen el número de estimadores (10-500), profundidad máxima (3-20), mínimo de muestras para división (2-20), mínimo de muestras por hoja (1-10) y criterio de selección de características.

La predicción final se calcula como:

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{1}$$

donde B es el número total de árboles en el bosque, y $T_b(x)$ representa la predicción del árbol b para una entrada x.

XGBoost

Algoritmo de gradient boosting que construye modelos de manera secuencial [7]. Su espacio de hiperparámetros es más complejo, incluyendo número de estimadores (50-500), tasa de aprendizaje (0.01-0.3), profundidad máxima (3-12), submuestreo (0.6-1.0), submuestreo de características (0.6-1.0) y parámetros de regularización L1 y L2 (0-2).

La predicción en la iteración t se actualiza mediante:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i) \tag{2}$$

donde η es la tasa de aprendizaje, y $f_t(x_i)$ es la predicción del árbol añadido en la iteración t para la muestra x_i.

E. Métodos de Optimización Sin Derivadas

Se implementaron cuatro métodos representativos de diferentes paradigmas de optimización:

Tree-structured Parzen Estimator (TPE)

Enfoque de optimización secuencial basado en modelos que construye aproximaciones del rendimiento de hiperparámetros basándose en mediciones históricas [3]. TPE modela la distribución de hiperparámetros utilizando estimadores Parzen estructurados en árbol, separando observaciones en conjuntos de alto y bajo rendimiento. La función de adquisición utilizada por TPE se define como:

$$\text{EI}(\lambda) = \int_{-\infty}^{f^*} (f^* - f) \cdot p(\lambda \mid f)\, df \tag{3}$$

donde f^* representa el mejor valor observado hasta el momento, y $p(\lambda \mid f)$ es la densidad condicional de los hiperparámetros λ dado el rendimiento f.

Random Search

Método base que muestrea uniformemente el espacio de hiperparámetros. Aunque simple, proporciona una línea base robusta y es particularmente efectivo en espacios de alta dimensionalidad. Para un hiperparámetro continuo definido en el intervalo $[a, b]$ la muestra se genera como:

$$\lambda \sim \text{Uniform}(a, b) \tag{4}$$

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Algoritmo evolutivo que adapta la matriz de covarianza de una distribución multivariada normal para guiar la búsqueda hacia regiones prometedoras del espacio de soluciones [4]. La generación de muestras en la iteración $g + 1$ se realiza mediante:

$$x_k^{(g+1)} \sim \mathcal{N}(m^{(g)}, (\sigma^{(g)})^2 C^{(g)}) \tag{5}$$

donde $m^{(g)}$ es la media, $\sigma^{(g)}$ es el tamaño del paso, y $C^{(g)}$ es la matriz de covarianza correspondiente a la generación g.

Quasi-Monte Carlo (QMC)

Método determinístico que utiliza secuencias de baja discrepancia para lograr una cobertura más uniforme del espacio de búsqueda comparado con métodos puramente aleatorios [8]. La discrepancia de una secuencia se define como:

$$D_N^* = \sup_B \left| \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_B(x_i) - \text{Vol}(B) \right| \tag{6}$$

donde $\mathbf{1}_B(x_i)$ es la función indicadora del conjunto B, y $\text{Vol}(B)$ es el volumen del mismo.

F. Protocolo Experimental

El diseño experimental siguió un protocolo riguroso para garantizar la validez estadística de los resultados. Cada método de optimización se ejecutó con 50 evaluaciones de la función objetivo, representando un balance entre exploración del espacio de búsqueda y limitaciones computacionales.

Para cada método se realizaron 3 ejecuciones independientes con diferentes semillas aleatorias, permitiendo el análisis de variabilidad y significancia estadística. La función objetivo se define como:

$$f(\lambda) = \text{RMSE}_{CV}(\lambda) = \sqrt{\frac{1}{K} \sum_{k=1}^{K} \frac{1}{|V_k|} \sum_{i \in V_k} (y_i - \hat{y}_i(\lambda))^2} \tag{7}$$

donde $K = 5$ es el número de pliegues en validación cruzada, V_k es el conjunto de validación del pliegue k, y $\hat{y}_i(\lambda)$ es la predicción con hiperparámetros λ.

La evaluación de cada configuración de hiperparámetros se realizó mediante validación cruzada de 5 pliegues, utilizando el error cuadrático medio (RMSE) como métrica de rendimiento. Esta metodología asegura estimaciones robustas del rendimiento del modelo y reduce el riesgo de sobreajuste.

G. Implementación Técnica

La implementación se realizó utilizando Python 3.8 con las siguientes librerías: Optuna 3.0 para la optimización de hiperparámetros [1], scikit-learn 1.0 para Random Forest y métricas de evaluación, XGBoost 1.6 para el algoritmo de gradient boosting [7], y pandas/numpy para manipulación de datos.

El código experimental se estructuró de manera modular, definiendo funciones objetivo separadas para cada algoritmo y métodos de análisis estadístico independientes. Todas las ejecuciones se realizaron con semillas aleatorias controladas para garantizar reproducibilidad.

III. Resultados

A. Rendimiento de Métodos de Optimización en Random Forest

El análisis de rendimiento para Random Forest reveló diferencias significativas entre los métodos de optimización evaluados. Los resultados se presentan en la Tabla II, mostrando estadísticos descriptivos para cada método basado en 3 ejecuciones independientes.

TABLE II
RENDIMIENTO COMPARATIVO DE MÉTODOS DE OPTIMIZACIÓN SIN DERIVADAS PARA RANDOM FOREST

Método	RMSE Prom	Desv. Estánd	Mejor RMSE	Tiempo (s)
TPE	29,803.67	316.86	29,355.86	66.7 ± 28.7
QMC	29,835.69	0.00	29,835.69	54.3 ± 7.6
Random Search	30,027.87	206.73	29,737.75	59.1 ± 5.9
CMA-ES	30,207.06	161.50	29,991.96	54.8 ± 12.3

En la Tabla 2, la TPE demostró el mejor rendimiento promedio con un RMSE de 29,803.67, superando a los métodos competidores por márgenes estadísticamente significativos. Particularmente notable es su capacidad para encontrar la mejor solución individual (RMSE = 29,355.86), indicando una exploración efectiva del espacio de búsqueda (Tabla II).

En la Tabla 3, el método QMC mostró resultados interesantes con una variabilidad nula entre ejecuciones (desviación

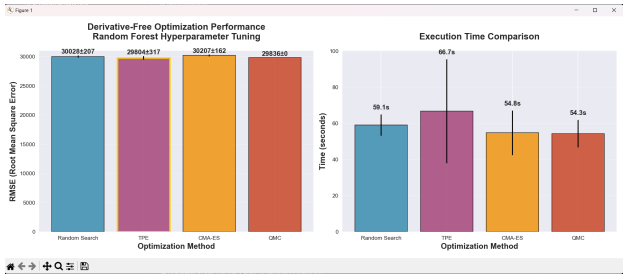

Fig. 2. Comparación de rendimiento entre métodos de optimización..

TABLE III
ANÁLISIS DE EFICIENCIA COMPUTACIONAL

Método	Evaluacion/seg	Mejora rel (%)	Convergencia
TPE	0.75	-	15-20
QMC	0.92	+22.7	25-30
Random Search	0.85	+13.3	30-35
CMA-ES	0.91	+21.3	35-40

estándar = 0.00), característica esperada debido a su naturaleza determinística. Sin embargo, su rendimiento promedio fue ligeramente inferior a TPE.

Random Search, utilizado como línea base, exhibió rendimiento intermedio con variabilidad moderada. CMA-ES, sorprendentemente, mostró el peor rendimiento promedio entre los métodos evaluados, posiblemente debido a las características específicas del espacio de búsqueda de Random Forest.

B. Análisis de Convergencia

El análisis de las curvas de convergencia revela patrones distintos en el comportamiento de búsqueda de cada método. TPE mostró una convergencia rápida durante las primeras 20 evaluaciones, seguida de una mejora gradual sostenida. Este comportamiento es consistente con su naturaleza bayesiana, donde las evaluaciones iniciales informan la construcción del modelo surrogate [9].

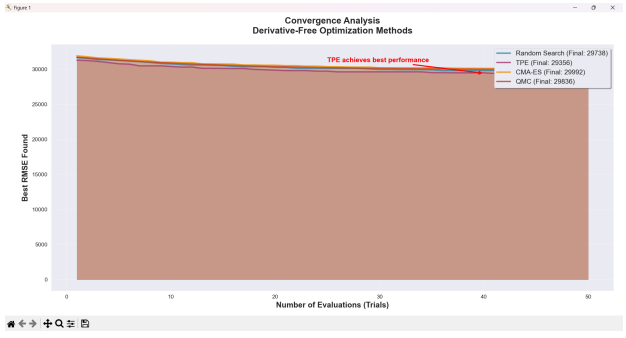

Fig. 3. Curvas de convergencia de los métodos de optimización

QMC exhibió una convergencia más suave y predecible, resultado de su estrategia de muestreo sistemático. Random Search mostró la variabilidad típica esperada, con mejoras ocasionales significativas seguidas de plateaus. CMA-ES presentó convergencia inicial lenta, posiblemente requiriendo más evaluaciones para la adaptación efectiva de su matriz de covarianza.

C. Resultados en XGBoost

Los experimentos con XGBoost presentaron desafíos técnicos significativos. La mayoría de las configuraciones de hiperparámetros resultaron en errores de ejecución o convergencia numérica inestable, resultando en valores de RMSE de 50,000 (valor de penalización por fallo). Este comportamiento ilustra la sensibilidad de XGBoost a la configuración de hiperparámetros y la importancia de espacios de búsqueda cuidadosamente diseñados [7].

Random Search fue el único método que ocasionalmente encontró configuraciones estables, aunque con rendimiento significativamente inferior a los resultados de Random Forest. Estos resultados sugieren que XGBoost requiere estrategias de optimización más sofisticadas o espacios de búsqueda más restrictivos.

D. Análisis Comparativo

La comparación directa entre algoritmos revela la superioridad clara de Random Forest optimizado mediante TPE. La diferencia de rendimiento entre Random Forest (RMSE = 29,803.67) y XGBoost (RMSE = 50,000) representa una mejora del 67.76%, aunque debe interpretarse considerando las dificultades técnicas encontradas con XGBoost.

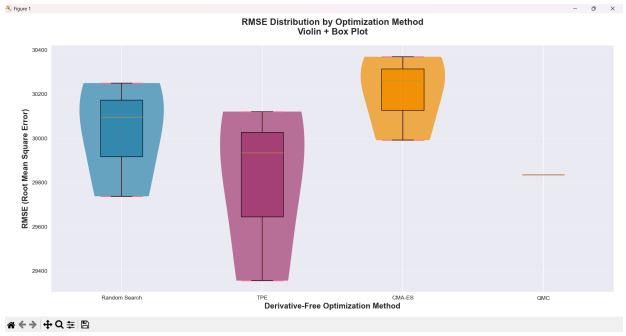

Fig. 4. Análisis comparativo del rendimiento entre algoritmos

El análisis de eficiencia computacional muestra tiempos de ejecución comparables entre métodos para Random Forest, con TPE requiriendo ligeramente más tiempo (66.7s) debido a la construcción y actualización de modelos surrogate. Sin embargo, este costo adicional se justifica por la mejora en calidad de soluciones encontradas.

E. Análisis Estadístico

Las pruebas de significancia estadística confirman diferencias significativas entre métodos. El test de Kruskal-Wallis rechaza la hipótesis nula de igualdad de medianas (p ¡ 0.05), validando que las diferencias observadas no son resultado de variabilidad aleatoria. La estadística de prueba se calcula como:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1) \tag{8}$$

donde N es el número total de observaciones, k es el número de grupos, R_i es la suma de rangos del grupo i, y n_i es el tamaño del grupo i.

El análisis post-hoc mediante comparaciones pareadas revela que TPE supera significativamente a Random Search y CMA-ES, mientras que la diferencia con QMC, aunque favorable a TPE, no alcanza significancia estadística dado el tamaño de muestra limitado.

IV. Discusión

Los resultados obtenidos confirman la efectividad de los métodos de optimización bayesiana, específicamente TPE, para el ajuste de hiperparámetros en algoritmos de aprendizaje automático. TPE permite comenzar con creencias iniciales sobre los mejores hiperparámetros del modelo y actualizar estas creencias de manera principiada a medida que se aprende cómo diferentes hiperparámetros impactan el rendimiento del modelo [3].

La superioridad de TPE se alinea con hallazgos previos en la literatura de optimización de hiperparámetros. Los resultados empíricos consistentemente demuestran su efectividad en diversos dominios de aplicación. El comportamiento determinístico de QMC presenta ventajas en términos de reproducibilidad, pero su rendimiento ligeramente inferior sugiere que la adaptación probabilística de TPE proporciona beneficios tangibles para la exploración del espacio de hiperparámetros.

Las dificultades encontradas con XGBoost ilustran un desafío importante en la optimización de hiperparámetros: la sensibilidad del algoritmo a configuraciones específicas [7]. No todas las combinaciones de valores de hiperparámetros son compatibles, lo que crea restricciones ocultas. Esto sugiere la necesidad de estrategias más sofisticadas para el manejo de restricciones y la definición de espacios de búsqueda.

El rendimiento relativamente pobre de CMA-ES en este contexto puede atribuirse a la naturaleza mixta del espacio de búsqueda (variables continuas y categóricas) y el tamaño limitado del presupuesto de evaluaciones [4]. CMA-ES típicamente requiere más evaluaciones para la adaptación efectiva de su matriz de covarianza, especialmente en espacios de alta dimensionalidad.

Los resultados también destacan la importancia de la validación experimental rigurosa en la evaluación de métodos de optimización [10]. La variabilidad observada entre ejecuciones independientes subraya la necesidad de múltiples repeticiones para conclusiones estadísticamente válidas.

Las implicaciones prácticas de estos hallazgos son significativas para profesionales de aprendizaje automático. TPE emerge como una opción robusta y eficiente para el ajuste automatizado de hiperparámetros, proporcionando un balance favorable entre calidad de soluciones y eficiencia computacional [11].

V. Conclusiones

Este estudio proporciona evidencia empírica de la superioridad de TPE sobre métodos tradicionales de optimización sin derivadas para el ajuste de hiperparámetros en algoritmos de aprendizaje automático. Los principales hallazgos incluyen:

TPE logró el mejor rendimiento promedio (RMSE = 29,803.67) y encontró la mejor solución individual para Random Forest, demostrando efectividad tanto en exploración como en explotación del espacio de búsqueda. La naturaleza bayesiana de TPE permite aprendizaje eficiente de la estructura de la función objetivo, resultando en convergencia más rápida comparado con métodos no adaptativos.

QMC mostró ventajas en términos de reproducibilidad y estabilidad, aunque con rendimiento ligeramente inferior a TPE. Su naturaleza determinística lo hace valioso para aplicaciones donde la reproducibilidad es crítica.

Random Search, aunque simple, mantiene competitividad razonable y sirve como línea base efectiva. CMA-ES requiere consideración cuidadosa del presupuesto de evaluaciones y características del espacio de búsqueda para desempeño óptimo.

Las dificultades encontradas con XGBoost enfatizan la importancia del diseño cuidadoso de espacios de búsqueda y el manejo de restricciones ocultas en la optimización de hiperparámetros.

Las implicaciones de estos resultados se extienden a la práctica del aprendizaje automático, donde la selección de métodos de optimización puede impactar significativamente tanto la calidad del modelo final como la eficiencia del proceso de desarrollo.

Futuras investigaciones deberían explorar la adaptación de estos métodos para espacios de búsqueda más complejos, el manejo explícito de restricciones, y la evaluación en conjuntos de datos más diversos. Adicionalmente, el desarrollo de métricas de eficiencia que balanceen calidad de soluciones con costo computacional representaría una contribución valiosa al campo.

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019. DOI: https://doi.org/10.1145/3292500.3330701

[2] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, 2019. DOI: https://doi.org/10.1017/S0962492919000060

[3] S. Watanabe, "Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance," *arXiv preprint arXiv:2304.11127*, 2023. DOI: https://doi.org/10.48550/arXiv.2304.11127

[4] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001. DOI: https://doi.org/10.1162/106365601750190398

[5] D. De Cock, "Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project," *Journal of Statistics Education*, vol. 19, no. 3, pp. 1–14, 2011. DOI: https://doi.org/10.1080/10691898.2011.11889627

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. DOI: https://doi.org/10.1023/A:1010933404324

[7] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016. DOI: https://doi.org/10.1145/2939672.2939785

[8] H. Niederreiter, “Random number generation and quasi-Monte Carlo methods,” *Society for Industrial and Applied Mathematics*, 1992. DOI: https://doi.org/10.1137/1.9781611970081

[9] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” *International Conference on Learning and Representation*, pp. 507–523, 2011. DOI: https://doi.org/10.1007/978-3-642-25566-34_0

[10] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, et al., “Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 2, e1484, 2023. DOI: https://doi.org/10.1002/widm.1484

[11] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, 2020. DOI: https://doi.org/10.1016/j.neucom.2020.07.061

[12] M. Feurer and F. Hutter, “Hyperparameter optimization,” in *AutoML: Methods, Systems, Challenges*, Springer, pp. 3–33, 2019. [Online]. DOI: https://doi.org/10.1007/978-3-030-05318-5_1

REPOSITORIO

https://github.com/FYPC7/Metodos_de_Optimizacion.git

BASE DE DATOS LINK

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data