Optimización de Hiperparámetros en Algoritmos de Aprendizaje Automático

July Andrea Acero Lozada y Kevin Stiveen Rojas Ramírez

Trabajo de Grado para Optar por el Título de Ingeniero Industrial

Director
Henry Lamos Díaz
PhD. Física-Matemática

Codirector
David Esteban Puentes
MSc. Ingeniería Industrial

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Estudios Industriales y Empresariales
Ingeniería Industrial
Bucaramanga
2023

Dedicatoria

Es con gran emoción que dedico este triunfo especialmente a mis amados padres, Carmen Elisa y Luis Eduardo, que con su inmenso afecto y lucha constante han estado a lo largo de mi trayecto académico y personal, brindándome refugio, resaltando mis capacidades e impulsándome cada día a ser mejor en todos los ámbitos. Agradecida con la vida por otorgarme padres tan excepcionales, por ser mi soporte, por creer en mí en cada instante, apoyarme incansablemente y dar todo de sí para mi formación profesional. De igual manera, este momento de culminación lo dedico a mi hermana, Karen Dayana, por ser una de mis motivaciones para avanzar y lograr grandes propósitos en la vida.

Y sin lugar a duda, a mis adorados abuelos, Rosalía Jiménez y Humberto Lozada, por todos sus consejos y palabras llenas de sabiduría, por encomendarme en sus oraciones y cuidar de mí. Gracias por todo el amor que me han brindado desde pequeña, por creer en mí en todo momento. Son mi ejemplo de tenacidad y perseverancia.

July Andrea Acero Lozada

Dedicatoria

A mis padres, quienes me brindaron las herramientas para poder dar lo mejor de mí a lo largo de la carrera universitaria. Su dedicación y amor fueron el motor que me impulsaba cada día a alcanzar mis metas. Cada logro en el ámbito académico y social es un reflejo de los valores que han inculcado en mí, y gracias a ello he podido enfrentar las diferentes problemáticas con determinación y perseverancia. Estaré siempre agradecido por su apoyo incondicional, los amo.

A mis hermanos por ser mi inspiración para superar los retos que conlleva la vida. Su presencia ha sido un constante estímulo para seguir adelante, incluso cuando las dificultades parecían insuperables. El lazo fuerte que hemos creado nos brinda la energía necesaria para superar cualquier eventualidad juntos, los amo y agradezco tenerlos a mi lado.

A mi familia cercana por apoyarme siempre en todos mis proyectos. La confianza que me han brindado me ha dado fuerzas para seguir adelante, sin importar los desafíos que se presenten en el camino. Su alegría y amor incondicional han sido un regalo preciado que me ha dado la fortaleza necesaria para enfrentar cada día con entusiasmo y determinación. Me siento bendecido por contar con una familia tan hermosa.

A mis amigos quienes han estado siempre que los necesitaba y fueron ese apoyo que requería en momentos de dificultad. Su presencia constante fue muy necesaria en aquellos momentos cuando dudaba de mis habilidades, muchas gracias.

Kevin Stiveen Rojas Ramirez

Agradecimientos

Agradecemos a nuestros padres por brindarnos la oportunidad de pertenecer a esta prestigiosa universidad y programa académico, así como también por su confianza y sacrificio.

Al profesor David Esteban Puentes Garzón, por la confianza depositada, por los conocimientos y la guía que nos brindó para dar inicio a la realización de nuestra investigación. Mucha admiración y respeto como profesional y persona. Sin duda alguna, una inspiración y ejemplo a seguir.

A Juan David Márquez González y a la profesora Yuly Andrea Ramírez Sierra, quienes fueron un gran apoyo para la culminación de esta investigación. Estamos muy agradecidos por la ayuda brindada.

A nuestro director, Henry Lamos, por su tiempo y conocimientos transmitidos durante el desarrollo de este proyecto.

Al grupo de investigación ÓPALO, por los espacios de orientación y conocimiento.

Contenido

Lista figuras

Lista de Tablas

Lista de apéndices

(Ver apéndices adjuntos en la carpeta)

Resumen

Título: Optimización de Hiperparámetros en Algoritmos de Aprendizaje Automático.

Autores: July Andrea Acero Lozada y Kevin Stiveen Rojas Ramírez.

Palabras Clave: HPO, AutoML, espacios de búsqueda, hiperparámetros, estrategia de espacios de búsqueda, Ingeniería de características.

Descripción:

En la actualidad, el *Automated Machine Learning* (AutoML) ha sido ampliamente aplicado debido al alto potencial benéfico que aporta a los distintos sectores de la industria, particularmente en la mejora de los flujos de trabajo, el rendimiento de los procesos y la efectividad empresarial. Dentro del AutoML es importante la adecuada elección de los valores de hiperparámetros mediante técnicas de optimización, ya que los algoritmos de ML dependen de los valores de hiperparámetros (HPs) elegidos debido a que estos influyen en el rendimiento de la máquina. Por ello, en esta investigación se diseña metodológicamente el proceso de construcción de los espacios de búsqueda y se propone una técnica para la optimización hiperparámetros de una máquina de *Random Forest* (RF) mediante la adaptación de la metaheurística de *Particle Swarm Optimization* (PSO). Esta técnica se utiliza para el análisis de conjunto de datos equilibrados y desequilibrados, mediante un proceso de *benchmarking*. Para validar el rendimiento del método, inicialmente se ejecutó el algoritmo de *Random Forest* sin aplicar la técnica de optimización, en donde se encontró la existencia de sobreajuste. Para reducir este comportamiento se establece una comparación entre PSO y técnicas de mayor usabilidad como *Grid Search* y *Random Search* a fin de analizar el comportamiento del modelo e identificar la técnica más efectiva en términos de funcionalidad y rendimiento. Los resultados demuestran que el modelo de clasificación Random Forest junto con la técnica de optimización de hiperparámetros *Particle Swarm Optimization* (PSO) mejoró la eficacia general del modelo, posibilitando obtener valores óptimos de hiperparámetros que mejoraron el rendimiento y el sobreajuste del modelo.

*Trabajo de grado.

**Facultad de Ingenierías Fisicomecánicas. Escuela de Estudios Industriales y Empresariales. Director: Henry Lamos Díaz, Ph.D en Física-Matemática. Codirector: David Esteban Puentes Garzón, M.Sc en ingeniería industrial

Abstract

Title: Hyperparameter Optimization in Machine Learning Algorithms.

Autor's: July Andrea Acero Lozada y Kevin Stiveen Rojas Ramírez.

Keywords: HPO, AutoML, search spaces, hyperparameters, search space strategy, feature engineering.

Description:

Nowadays, Automated Machine Learning (AutoML) has been widely applied due to the high beneficial potential it brings to different industry sectors, particularly in the improvement of workflows, process performance and business effectiveness. Within AutoML, the proper choice of hyperparameter values through optimization techniques is important, since ML algorithms depend on the chosen hyperparameter values (HPs) due to the fact that they influence machine performance. Therefore, in this research we methodologically design the process of constructing the search spaces and propose a technique for hyperparameter optimization of a Random Forest machine by adapting the particle swarm metaheuristic (PSO). This technique is used for the analysis of balanced and unbalanced dataset through a benchmarking process. To validate the performance of the method, the Random Forest algorithm was initially run without applying the optimization technique, where the existence of overfitting was found. To reduce this behavior, a comparison between PSO and more usable techniques such as Grid Search and Random Search is established in order to analyze the behavior of the model and identify the most effective technique in terms of functionality and performance. The results show that the Random Forest classification model together with the hyperparameter optimization technique Particle Swarm Optimization (PSO) improved the overall efficiency of the model, making it possible to obtain optimal hyperparameter values that improved the performance and overfitting of the model.

*Trabajo de grado.

**Facultad de Ingenierías Fisicomecánicas. Escuela de Estudios Industriales y Empresariales. Director: Henry Lamos Díaz, Ph.D en Física-Matemática. Codirector: David Esteban Puentes Garzón, M.Sc en ingeniería industrial

Introducción

Los distintos sectores económicos a nivel mundial se han visto enfrentados a nuevos retos para analizar, asegurar y cuidar sus datos (Xu & Shi, 2015). Sectores como el educativo, financiero, energético, salud, industrial, tecnológico, manufacturero y de servicios, generan gran cantidad de datos en múltiples formatos. Estos conjuntos de datos conformados por mayor tamaño y complejidad son conocidos como *Big Data*, caracterizados por poseer un alto volumen, velocidad y variedad en su estructura que, al ser utilizados en la toma de decisiones, requieren de velocidad y validez. Sin embargo, los hardware y software convencionales no tienen la capacidad para gestionarlos o procesarlos por ser tan masivos y complejos de examinar (Rawat & Yadav, 2021). Por lo tanto, con el fin de abordar tales desafíos, han emergido nuevas técnicas de análisis más robustas con la capacidad de soportar las propiedades de los volúmenes de datos (Rawat & Yadav, 2021).

Técnicas innovadoras como el procesamiento del lenguaje natural, el aprendizaje automático y el aprendizaje profundo, han surgido en el campo de la inteligencia artificial (IA) (Prudius et al., 2019). En los últimos años, el *Machine Learning* (ML) ha sido uno de los métodos con mayor trascendencia en investigación y parte integral del trabajo eficiente. Esto debido a que soluciona los problemas asociados al *Big Data*, beneficiándose de las diversas características de los datos (alta heterogeneidad, la falta estructura, el carácter incompleto, los errores manuales, etc.) a raíz de un eficiente uso de recursos computacionales y humanos. El ML trae consigo ventajas competitivas en las organizaciones mejorando la toma de decisiones en el ámbito estratégico, de una manera efectiva, informada y fiable, al basarse en un análisis holístico de datos propios de la organización. Este proceso se realiza mediante la adquisición, gestión, procesamiento, extracción, interpretación de dichos datos. Por tanto, el uso de modelos y técnicas

de ML es más que necesario para cualquier organización actualmente con el propósito de facilitar el proceso de toma de decisiones (Prudius et al., 2019).

El *Machine Learning* (ML) es uno de los métodos de análisis más relevantes en la actualidad, debido a que son genéricos y demuestran alto rendimiento en problemas de análisis de datos con grandes volúmenes. Sin embargo, el proceso de construcción dificulta en algunos casos su aplicación, debido a que requiere grandes cantidades de tiempo en la definición de la arquitectura modelo y elección del tipo de algoritmo (L. Yang & Shami, 2020). Por esta razón, se implementa el *Automated Machine Learning* (AutoML) como método para automatizar las diversas decisiones asociadas a su construcción, aplicando técnicas de Optimización Hiperparamétrica (HPO) a modelos de ML. De esta manera, se permite la configuración óptima de la arquitectura, reduciendo el esfuerzo e interacción humana necesario y aumentando el rendimiento (Wistuba et al., 2021).

Por tanto, el uso del AutoML mejora el rendimiento en los modelos de ML para lograr el cumplimiento de los objetivos planteados mediante la utilización de menos recursos beneficiando en gran medida a las organizaciones. Por ello, el presente trabajo se enfoca en la aplicación del AutoML mediante técnicas de Optimización de Hiperparámetros (HPO) aplicadas a un modelo de ML. A partir de la identificación, selección y validación se definirán las técnicas de HPO más relevantes para solucionar las problemáticas de un algoritmo de ML.

En consecuencia, dada la importancia primordial del proceso de AutoML en la ejecución de modelos de *Machine Learning*, a lo largo del presente proyecto se implementa la técnica *Particle Swarm Optimization (*PSO) en un algoritmo de *Random Forest* con el fin de ajustar los hiperparámetros del modelo y a través de un proceso de *benchmarking* evaluar el desempeño del modelo en distintos escenarios, en este caso, en conjunto de datos equilibrados y desequilibrados.

Esta investigación busca verificar que el rendimiento y efectividad de una máquina de *Machine Learning* depende en gran medida de los valores óptimos de hiperparámetros encontrados mediante la técnica de optimización elegida.

Tabla 1

Cumplimiento de objetivos

Objetivo	Cumplimento
Identificar las técnicas de construcción de espacios de búsqueda a través de la revisión de literatura.	Capítulo 1
Seleccionar las estrategias de construcción de espacios de búsqueda para la optimización de hiperparámetros en los modelos de aprendizaje automático.	Capítulo 9
Validar las estrategias de construcción de espacio de búsqueda mediante conjuntos de datos disponibles para benchmarking.	Capítulo 16
Elaborar un artículo de carácter publicable a partir de los resultados obtenidos en la investigación.	Artículo

1. Revisión de Literatura

1.1 Análisis Bibliométrico

La revisión de literatura se inicia con una búsqueda direccionada hacia la optimización de hiperparámetros en algoritmos de aprendizaje automático. Inicialmente, se utilizan las bases de datos disponibles en la plataforma de la Universidad Industrial de Santander (UIS). Dentro de ellas se elige la base de datos *Scopus*, que permite hacer una búsqueda multidisciplinaria, analizando los problemas que han sido solucionados mediante la optimización de hiperparámetros en otro tipo de investigaciones.

Se plantea la ecuación de búsqueda, definida en la Ecuación 1, la cual incluye términos claves tales como: AutoML, Optimización de hiperparámetros y aprendizaje automático. Estos aspectos son relevantes para esta investigación.

$$ALL = (("automl" \ OR \ automated \ OR \ "Automated \ machine \ learning") \ AND \ ("HPO" \ OR \ "Hyperparameter \ optimization") \ AND \ ("machine \ learning" \ OR \ "ml")) \quad (1)$$

Inicialmente, se obtienen 3075 documentos. Con el fin de disminuir e identificar aquellos de mayor trascendencia para el trabajo, se procede a depurar la información por años (2013-2022) obteniendo los documentos más recientes, con el propósito de conocer la importancia que ha tenido el *Automated Machine Learning* (AutoML) y la aplicación de la optimización de hiperparámetros (HPO) en temas de investigación respecto a los últimos años.

Finalmente, se consideraron los índices de impacto *CiteScore Percentil*, como criterios de exclusión e inclusión para mejorar la calidad de la información. Se tienen en cuenta aquellas publicaciones ubicadas del percentil 85 en adelante. Todos los resultados de las depuraciones se encuentran en la Tabla 2.

Tabla 2

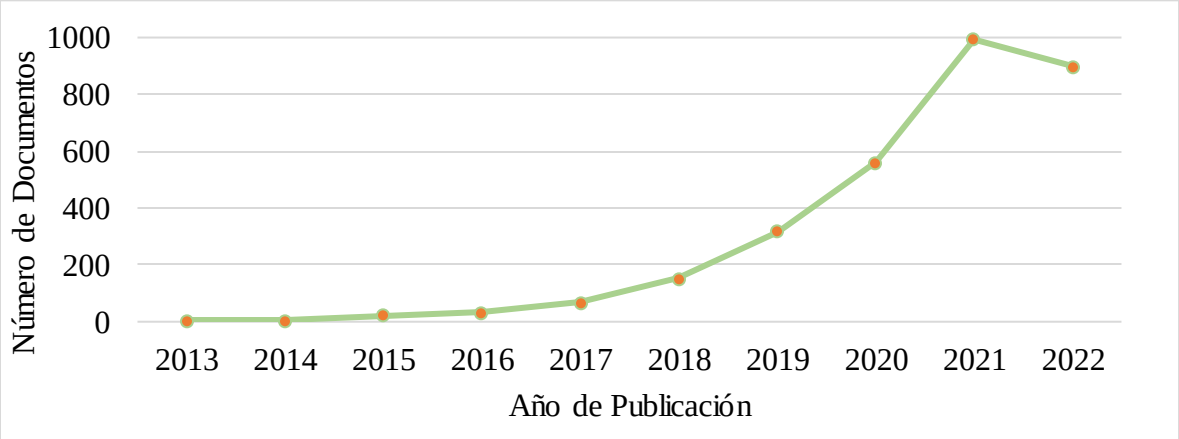
Criterios de inclusión y exclusión en el proceso de revisión bibliográfica

Criterio	Resultados
Todos los archivos	3075
Title-Abs-Key	163
Últimos 10 años	161
Categoría de Journal	33

De acuerdo con la información encontrada por medio de *Scopus* y el uso del software *VOSviewer*, se procede a un análisis general tomando en cuenta los siguientes factores: año de publicación, países, palabras claves, autores. Para ello, se analiza las publicaciones de los artículos encontrados desde 2013 hasta 2022, como se visualiza en la Figura 1.

Figura 1

Total de publicaciones por año (todos los documentos encontrados)



Nota. Adaptado de Scopus (2022)

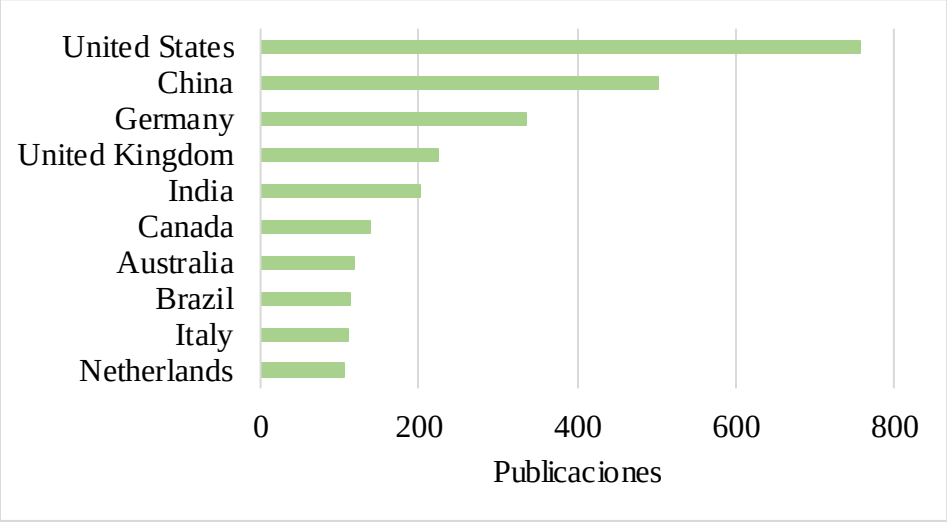
Se observa un comportamiento ascendente en los temas de AutoML y Optimización de Hiperparámetros (HPO) estudiados en esta investigación. Se espera que en el futuro se presente una tendencia creciente, en cuanto al interés de desarrollo, aplicación y potencialización de estas

técnicas para la solución de distintos problemas en varias áreas como informática, ingeniería, medicina, contabilidad, negocios y gestión.

A su vez, en la Figura 2 se puede observar las publicaciones de documentos por países. En dicha gráfica están los primeros diez países con mayor cantidad de publicaciones, siendo Estados Unidos el mayor contribuyente de producciones científicas en este tema, seguido por China y Alemania. Si bien Colombia no está entre los 10 primeros de este ranking, está en el top 50 con 13 publicaciones.

Figura 2

Países con mayor número de publicaciones (total de documentos encontrados)

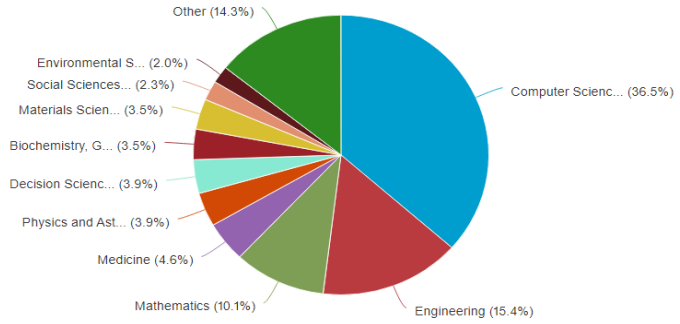


Nota. Adaptado de *Scopus* (2022)

La Figura 3 muestra las principales categorías o áreas de investigación en las que se clasifican los documentos de revisión, es de señalar que un documento puede pertenecer a más de una categoría al mismo tiempo. En consecuencia, las categorías más relevantes son: *Computer Science, Engineering* y *Business, Management and Accounting*; estando en línea con los temas de investigación, pues hacen parte de las tendencias en la rama de Ingeniería Industrial.

Figura 3

Categorías de las publicaciones

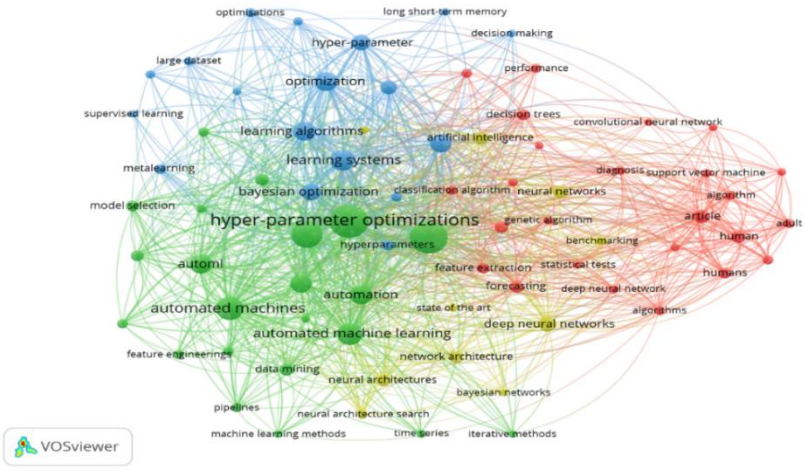


Nota. Tomado de *Scopus* (2022)

En la Figura 4, se presenta la correlación de las palabras claves incluidas en el estudio, concretamente de aquellos datos obtenidos en los últimos 10 años; por ende, se evidencia la conexión entre Optimización de hiperparámetros *(Hyperparameter Optimization)* y *AutoML*, así mismo, están interrelacionadas con otros temas que forman parte del estudio.

Figura 4

Conexión entre palabras clave

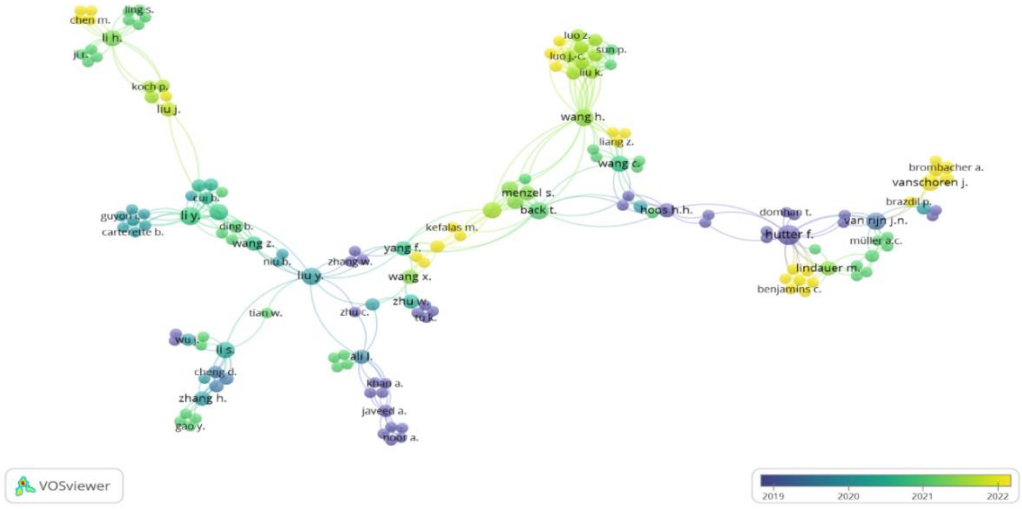


Nota. Tomado de *VOSviewer* (2022)

Por último, en la Figura 5 se visualiza la conexión entre autores, lo cual demuestra la relevancia y la colaboración investigativa para el crecimiento, construcción y desarrollo del tema de estudio en más de una producción científica.

Figura 5

Conexión de autores en colaboración para producciones científicas



Nota. Tomado de *VOSviewer* (2022)

Teniendo en cuenta el análisis bibliométrico realizado, es evidente que el *AutoML* y *Hyperparameter optimization* (HPO) han sido objeto de una creciente investigación en los últimos 10 años. Por ello, se considera analizar este tema bajo el enfoque de Ingeniería Industrial, mediante un nuevo caso de estudio; además, los autores piensan que no hay suficiente investigación a nivel nacional que demuestre la factibilidad de implementar dicho concepto en las organizaciones. En consecuencia, se pretende complementar la investigación existente o servir como base para futuros estudios.

1.2 Análisis preliminar de la literatura

En general, el *Automated Machine Learning* (AutoML), la Optimización de Hiperparámetros (HPO) y el *Machine Learning* (ML) son temas actuales y altamente relacionados que han tomado rumbos variados en sus aplicaciones, debido al alto potencial benéfico en los distintos sectores de la industria. Lo anterior se corrobora en las investigaciones más relevantes relacionadas con los temas mencionados. En la Tabla 3 se muestra la aplicabilidad del AutoML, ML y HPO en diversidad de sectores. Dentro de los sectores más destacados se encuentran el educativo, financiero, energético, salud, industrial, tecnológico, manufacturero y de servicios. Estas investigaciones demuestran los beneficios percibidos en sus procesos, como la mejora en flujos de trabajo, rendimiento y efectividad. A continuación, se presentan y discuten los principales desarrollos y aportes encontrados, que sirven como referencia para la comprensión de estos importantes temas.

Tabla 3

Artículos relevantes obtenidos de la ecuación de búsqueda

TÍTULO	AUTOR	AÑO
Hyperparameter optimization via sequential uniform designs	Yang, Z., Zhang, A.	2021
Efficient AutoML via Combinational Sampling	Nguyen, D.A., Kononova, A.V., Menzel, S., Sendhoff, B., Back, T.	2021
Automated Machine Learning: The New Wave of Machine Learning	Chauhan, K., Jani, S., Thakkar, D., (...), Tanwar, S., Obaidat, M.S.	2020

Continuación Tabla 3

Artículos relevantes obtenidos de la ecuación de búsqueda

An Efficient Contesting Procedure for AutoML Optimization	Nguyen, D.A., Kononova, A.V., Menzel, S., Sendhoff, B., Back, T.	2022
Autofhm: A Python Library for Automated Machine Learning	Viji Rajendran, V., Verghese, D.C., Mohammed Arshu, P.T., Randheer Ramesh, K., Subin, T.G.	2021
SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization	Lindauer, M., Eggensperger, K., Feurer, M., (...), Sass, R., Hutter, F.	2022
Benchmark of automated machine learning with state-of-the-art image segmentation algorithms for tool condition monitoring	Lutz, B., Reisch, R., Kisskalt, D., (...), Knoll, A., Franke, J.	2020
Auto-CASH: A meta-learning embedding approach for autonomous classification algorithm selection	Mu, T., Wang, H., Wang, C., Liang, Z., Shao, X.	2022
Development of AMES: Automated ML Expert System	Patil, P.S., Kappuram, K., Rumao, R., Bari, P.	2022
Hyperparameters search methods for machine learning linear workflows	Peskova, K., Neruda, R.	2019

Continuación Tabla 3

Artículos relevantes obtenidos de la ecuación de búsqueda

Implementing autoML in educational data mining for prediction tasks	Tsiakmaki, M., Kostopoulos, G., Kotsiantis, S., Ragos, O.	2020
Interpretable Assessment of ST-Segment Deviation in ECG Time Series	Jurado, I.C., Fedjajevs, A., Vanschoren, J., Brombacher, A.	2020
AutoTinyML for microcontrollers: Dealing with black-box deployability	Perego, R., Candelieri, A., Archetti, F., Pau, D.	2022
Auto-REP: An Automated Regression Pipeline Approach for High-efficiency Earthquake Prediction Using LANL Data	Yang, F., Kefalas, M., Koch, M., (...), Qiao, Y., Back, T.	2022
Ultron-AutoML: An open-source, distributed, scalable framework for efficient hyper-parameter optimization	Narayan, S., Krishna, C.S., Mishra, V., (...), Gupta, A., Singh, N.	2020
Bayesian Contextual Bandits for Hyper Parameter Optimization	Sui, G., Yu, Y.	2020
BiLSTM-CNN Hyperparameter Optimization for Speech Emotion and Stress	Gumelar, A.B., Yuniarno, E.M., Adi, D.P., (...), Sugiarto, I., Purnomo, M.H.	2021
Automated hyper-parameter tuning for machine learning models in machine health prognostics	Cheung, W.-C., Zhang, W., Liu, Y., Yang, F., Rick-Siow-Mong, G.	2018

Nota. Adaptado de *Scopus* (2022).

Para empezar, Hutter et al., (2019) y Das & Mert Cakmak (2018) indican que la necesidad de implementar el AutoML, radica en la influencia que tienen las decisiones de diseño en el rendimiento de los modelos ML, debido a que, aún para expertos, la cantidad de tiempo requerido para tomar correctamente decisiones relacionadas con el preprocesamiento de características, elección de modelos u algoritmos, y optimización de hiperparámetros, es extremadamente grande. Para este fin, el campo del AutoML tiene por objetivo tomar esas decisiones de manera automatizada y objetiva basándose en datos propios de la problemática, se apoya en tres métodos: la Optimización hiperparamétrica (HPO), el Meta-aprendizaje (*Meta-Learning*) y la Búsqueda de Arquitectura Neuronal (*Neural Architecture Search*, NAS).

El primer método, el Meta-aprendizaje (*Meta-Learning*) hace referencia a observar sistemáticamente el rendimiento de diferentes enfoques de ML en distintas tareas de aprendizaje y, aprender de esta experiencia (Metadatos). Con esto se logra aprender nuevas tareas mucho más rápido, es decir, recoge de manera sistemática la experiencia de modelos de ML y con ella aprender más fácilmente. El siguiente procedimiento, corresponde a la Búsqueda de Arquitectura Neuronal (*Neural Architecture Search*, NAS) que engloba a las arquitecturas neuronales que actualmente han sido desarrolladas de modo automático. De esta forma, se evita el desarrollo manual por expertos humanos, que supone un proceso lento y propenso a errores. En resumen, es el proceso de automatización de la ingeniería arquitectónica.

Además, los métodos de la NAS se clasifican según tres dimensiones: el espacio de búsqueda, la estrategia de búsqueda y la estrategia de estimación del rendimiento. Según Wistuba et al. (2021), NAS es un pilar importante del AutoML debido a su interpretabilidad, reproducibilidad y robustez. También, muestra el proceso general llevado a cabo por NAS en el

flujo de trabajo del AutoML, que consta de varios pasos: preparación de datos (*Collection, cleaning y argumentation*), ingeniería de características (*selection, extraction y construction*), generación de modelos (*hyperparameter optimization (HPO*) and *architecture optimization (AO))* y evaluación de modelos (*Low-fidelity, Early-stopping, Surrogate Model y Weight-sharing*), así como la metodología a seguir en este tipo de aplicaciones.

Finalmente, la optimización hiperparamétrica (HPO) es el interés central de la presente investigación, esto respecta a que cada algoritmo de ML tiene hiperparámetros asociados a su estructura y proceso de entrenamiento; en consecuencia, es tarea básica del AutoML establecerlos automáticamente, logrando reducir el esfuerzo humano, mejorar el rendimiento y aumentar la reproducibilidad e imparcialidad (Meisenbacher et al., 2022). Dentro de este orden de ideas, los autores Yang & Shami, (2020) dan contextualización exhaustiva de la HPO aplicada al ML. Primeramente, corroboran la problemática principal de ML, pues atribuyen a la elección del algoritmo de ML y su construcción grandes cantidades de tiempo, que imposibilitan en algunos casos su aplicación. Adicionalmente, señalan que las técnicas tradicionales como el ajuste manual y métodos basados en gradientes, son inadecuadas debido a gran cantidad de tiempo requerida y la no convexidad para las métricas de algunos hiperparámetros.

Según lo anterior, para problemas actuales de ML, enfoques teóricos de decisión (*Grid Search (GS), Random Search (RS) y el grad student descent (GSD)*), los modelos de optimización bayesianos (según su función sustituta: *Gaussian process (GP), Random forest (RF) y Tree-Structured Parzen Estimator (TPE)*), las técnicas de optimización Multifidelidad (*Successive Halving(Exhaustive methods), Hyperban y Bayesian Optimization Hyperband (BOHB)*) y Metaheurísticos (*Genetic Algorithm (GA) y Particle Swarm Optimization (PSO)*) son más adecuados para abordar las diferentes problemáticas que enfrenta la HPO en el ML. Finalmente,

los autores citados proponen un uso óptimo de técnicas de HPO según la técnica de *Machine Learning,* además de sus medios de aplicación.

A partir de lo mencionado, se puede señalar que la flexibilidad de diversidad de técnicas de AutoML en HPO da lugar a diversas aplicaciones. Por ejemplo, en el sector manufacturero, Lutz et al., (2020), abordan el problema asociado al tiempo de reemplazo de la herramienta de corte industrial, que surge a raíz de que este instrumento se debe utilizar el mayor tiempo posible, pero sin afectar la calidad de corte. Por ello, a fin de prevenir las afectaciones de esta problemática, los sistemas *Tool Condition Monitoring* (TCM) controlan el estado de la herramienta. Sin embargo, los métodos tradicionalmente utilizados para procesar señales y detectar los defectos, requieren mucho tiempo y son limitados en cuanto a variaciones. En este último problema, a modo de solución aplicaron tres métodos recientes: *Sliding Window AutoML*, redes de segmentación de un paso (*FCN,U-Net SegNet, LinkNet, and PSPNet*) y *Sliding Window personalizado*, y a través de una comparación de criterios de *benchmarking* (*pixel-wise accuracy, intersection over union* (IoU), *and mean intersection over union* (mIoU)) se recibió un mayor rendimiento de la técnica combinada con AutoML, lo cual, se le atribuye a su mayor precisión, flexibilidad, interpretabilidad, menor complejidad y configuración que los demás modelos. Debido a todo lo anterior, el AutoML es una opción prometedora, fiable y adecuada para la segmentación de imágenes en el monitoreo del estado de la herramienta de corte industrial.

Por otro lado, aplicado en el campo educativo Peskova & Neruda, (2019) implementan el HPO para optimizar la configuración de los métodos y flujos de trabajo de ML. El autor da solución al problema principal de HPO, teniendo en cuenta los hiperparámetros propios del modelo ML y de técnicas de preprocesamiento. Inicialmente, desarrollaron dos variaciones de modelos de AutoML. La primera, consiste en la aplicación de dos técnicas de HPO (*Simulated Annealing*

Search (SA) y *Evolutionary Algorithm Search* (EA)) a once algoritmos de ML usados para clasificación y regresión, entre los cuales se encuentran: *Bernoulli NB, Decision Tree, AdaBoost, Gradient Boostin y Random Forest*; y la segunda, consistió en adicionar a la primera variante una cadena de cinco metodologías de preprocesamiento de datos: *Standard Scaler, Quantile Transformer, Power Transformer, Normalizer y Principal component analysis* (PCA). Además, para ambas variaciones, definieron sus hiperparámetros según el tipo (Categórico, entero, booleano, flotante). Después de su desarrollo, evaluaron dos aspectos de los modelos mencionados: las curvas de aprendizaje según el tipo de metodología de preprocesamiento y los valores de *Benchmarking* (*Minimum error, Improvement y Best and median values*) generados por *Randomized Search* y por *Randomized Search* combinado preprocesamiento. En general el resultado que se obtuvo fue que la cadena de métodos de preprocesamiento, el recocido simulado y la búsqueda evolutiva, funcionan significativamente mejor en comparación al *benchmark* generado por *Randomized Search*. De igual modo, encontró que al añadir más parámetros aumenta la legitimidad al utilizar algoritmos de búsqueda fuerte.

Por otro lado, en el sector productivo, Motz et al., (2022) abordan, mediante diferentes técnicas de HPO a las grandes cantidades de datos generados por los procesos de producción digitalizados y conectados. Específicamente, aplicaron técnicas de HPO como: *Grid search*, *Random search*, *CMA-ES, hyperband*, *BOHB, FABOLAS*, *GPBO, SMAC* y *TPE*, a cuatro modelos de ML: *Decision tree*, *Random Forest, XGBoost* y *Multilayer perceptron*. Las cuales cuentan con mayor popularidad, viabilidad técnica y beneficios potenciales, y se aplican a tres diferentes áreas y procesos (predicción de parámetros de producción), máquinas y activos (Mantenimiento predictivo), y productos (Calidad predictiva). Igualmente, mediante un enfoque estructurado de *Benchmarking* (*Final performance, anytime performance, robustness, and parallelization*

efficiency) evaluaron y validaron las diferentes configuraciones destacando aquellas que son relevantes en los criterios de evaluación.

También en el sector de la salud, Javeed et al., (2019) muestran que la aplicación de modelos de ML optimizados en preprocesamiento e hiperparámetros, pueden solucionar la problemática relacionada con diagnósticos de cardiopatía (*heart disease*). Lo anterior, se justifica al analizar los tipos más comunes de cardiopatía: la insuficiencia cardíaca (*heart failure* (IC)) y enfermedad arterial coronaria (*Coronary Artery Disease* (CAD)). Ambos tipos están altamente relacionados y con alta representación en tasas de mortalidad a lo largo del mundo. Por ello, a fin de prevenir, se han aplicado métodos tradicionales de diagnóstico como la Angiografía (*Angiography*), diagnósticos de expertos y algoritmo de ML no optimizados. Sin embargo, dichos métodos generan diversidad de aspectos negativos que dificultan su aplicación, como el elevado costo, efectos secundarios, gastos grandes de tiempo, errores humanos y resultados erróneos (Debidos al sobreajuste). Por tanto, para la solución final, los autores aplicaron a una base de datos de insuficiencia cardíaca, un sistema híbrido entre dos algoritmos: Búsqueda aleatoria (Encargado del preprocesamiento de características) y Bosque aleatorio (Encargado de la predicción del diagnóstico a partir del subconjunto de características arrojado por la búsqueda aleatoria), utilizando un esquema de validación de 70-30% y métricas de evaluación (*Accuracy, Sensitivity, Specificity* and *MCC*) se comprobó que el sistema propuesto mejora el rendimiento y reduce la complejidad, respecto a modelos tradicionales usados en diagnóstico.

En términos generales, se puede decir que, existen varios desarrollos e investigaciones donde los sistemas de AutoML han tomado bastante rigor, desarrollando modelos que reducen el tiempo de ejecución, los errores de aplicación y los costes. Por ello, en el ámbito empresarial el AutoML es implementado como sistema para manejar la eficacia de los procesos involucrados en

diversas áreas. Así, se contribuye a dar solución a diversas problemáticas que exceden la capacidad humana, a través de la optimización de los espacios de búsqueda de los hiperparámetros asociados a las técnicas de regresión, clasificación y *clustering*, con el fin último de encontrar su mejor configuración, de modo que el desempeño del modelo sea el óptimo.

2. Planteamiento del problema

En los últimos años, el aprendizaje automático (*Machine learning* (ML)) se ha presentado como una manera efectiva para proporcionar una respuesta apropiada a diferentes campos de la ciencia, principalmente, en investigaciones relacionadas con la clasificación de imágenes, detección de objetos y modelamiento del lenguaje natural. Por otra parte, el rendimiento de los algoritmos de ML se ve afectado por factores asociados a su construcción como la experticia del diseñador, el tipo de algoritmo y la arquitectura del modelo. Los cuales imposibilitan en algunos casos la aplicación de algoritmos de ML, debido a las grandes cantidades de tiempo, recursos económicos y tecnológicos que demandan (Wistuba et al., 2021).

Por tanto, una solución a estas problemáticas es la optimización o ajuste de hiperparámetros (HPO), que permite definir el diseño y aprendizaje del modelo a entrenar, de manera que su desempeño sea el mejor posible. Así, el adecuado ajuste de hiperparámetros reduce el número de configuraciones a realizar, lo que incrementa la eficiencia de sistemas automáticos robustos, capaces de entrenar y evaluar algoritmos de ML. En este sentido, todo sistema de ML tiene hiperparámetros y en aras de simplicidad para el usuario, se validan de forma automática para que haya exactitud en las estimaciones realizadas. Con el objeto de realizar HPO se destacan los siguientes enfoques: la búsqueda de cuadrícula, la búsqueda aleatoria y la optimización bayesiana (Wen et al., 2020). De igual forma, existen métodos tradicionales de optimización de

hiperparámetros como la búsqueda exhaustiva manual, que no está disponible para espacios de búsqueda de alta dimensión y grandes conjuntos de datos (Li et al., 2022).

Es de señalar que, la optimización de hiperparámetros se lleva a cabo dentro de un espacio de búsqueda. Siendo necesario definirlo lo más correctamente posible y utilizarlo con el objetivo de encontrar el conjunto de las posibles configuraciones a un problema en concreto, y con su optimización, obtener aquella solución que mejor se ajuste al resultado esperado (W. Zhu & Wang, 2021). Aunque existen una gran variedad de estructuras o morfologías de selección de estos espacios, tales como completamente estructurados, basados en celdas, jerárquicos, basados en morfismos (He et al., 2021); sigue existiendo restricciones en las posibilidades de diseño debido al sesgo humano (W. Zhu & Wang, 2021). Para evitar este tipo de limitantes, recientemente se ha incursionado al campo de la automatización del aprendizaje automático (*Auto Machine Learning*-AutoML). Por ser un tema relativamente nuevo, algunos autores lo definen como: "El AutoML tiene como objetivo tomar estas decisiones de forma objetiva, automatizada y basada en los datos" (Hutter et al., 2019). Por otro lado, Zöller & Huber (2021) señalan que AutoML está diseñado para reducir la demanda de científicos de datos y permitir que los expertos en el dominio creen automáticamente aplicaciones de aprendizaje automático sin muchos requisitos de conocimientos estadísticos.

De acuerdo con lo mencionado, una inadecuada optimización y elección de hiperparámetros imposibilitaría el funcionamiento de un modelo de *Machine Learning*, causando errores de entrenamiento, minimización en la calidad de los datos e inconvenientes en el rendimiento de los algoritmos usados. El proceso de ajuste de hiperparámetros es diferente entre los algoritmos de ML debido a los distintos tipos de hiperparámetros, incluidos los categóricos, discretos y continuos (L. Yang & Shami, 2020). Sin embargo, el ajuste manual es ineficaz para

muchos problemas debido a ciertos factores que incluyen una gran cantidad de hiperparámetros, modelos complejos, evaluaciones de modelos que consumen mucho tiempo e interacciones de hiperparámetros no lineales (L. Yang & Shami, 2020). Los cuales han inspirado una mayor investigación en técnicas para la optimización automática de hiperparámetros.

En resumen, AutoML mejora el rendimiento y ahorra cantidades sustanciales de tiempo y dinero. Por ello, el interés investigativo y comercial ha aumentado considerablemente en la última década, siendo además relevante a nivel empresarial, debido a que las empresas se encuentran implementando nuevas herramientas y tecnologías ofrecidas por la inteligencia como estrategia para la toma de decisiones y direccionamiento empresarial. Finalmente, AutoML hace accesibles los enfoques de aprendizaje automático de última generación a los científicos del sector interesados en aplicar estos conocimientos, pero que presentan como obstáculo la falta de recursos tecnológicos para la aplicación de los algoritmos de búsqueda. La calidad de las soluciones no depende de la técnica de automatización, sino que también depende del correcto acotamiento o definición del espacio de búsqueda además de la optimización de hiperparámetros. Por lo tanto, a lo largo de este trabajo, se exploran y se definen los espacios de búsqueda de modelos de ML, mediante la validación de distintas técnicas de optimización, logrando como resultado la elección de hiperparámetros más viables para la solución de un problema de ML en concreto.

3. Objetivos

3.1 Objetivo general

Diseñar los espacios de búsqueda para la optimización de los hiperparámetros en algoritmos de aprendizaje automático.

3.2 Objetivos específicos

- Identificar las técnicas de construcción de espacios de búsqueda a través de la revisión de literatura.
- Seleccionar las estrategias de construcción de espacios de búsqueda para la optimización de hiperparámetros en los modelos de aprendizaje automático.
- Validar las estrategias de construcción de espacio de búsqueda mediante conjuntos de datos disponibles para benchmarking.
- Elaborar un artículo de carácter publicable a partir de los resultados obtenidos en la investigación.

4. Resultados Esperados

- Definir las técnicas de espacios de búsqueda más relevante en el estudio de hiperparámetros en aprendizaje automático.
- Concretar el espacio de búsqueda para la optimización de hiperparámetros.
- Artículo de carácter publicable.

5. Marco de Referencia

5.1 Marco de Antecedentes

En referencia al tema principal de investigación, se obtiene como resultado diferentes investigaciones y proyectos de grado a nivel local, nacional e internacional que tratan la importancia que ha tenido el AutoML, mediante la adecuada elección de distintos modelos de ML, llevando a solucionar óptimamente un problema en específico.

A nivel local, en la Universidad industrial de Santander, en 2021, Camelo García, D, y Suárez Suárez, P. en su tesis titulada "*un algoritmo genético para la detección de comunidades en redes sociales mejorado mediante una técnica de clustering*", da solución al problema de detección de comunidades (CD) en una red de colaboradores en la Universidad Industrial de Santander (UIS), esto mediante un algoritmo genético mejorado con *clustering*. Los aportes importantes al presente proyecto se dan en términos metodológicos en aspectos relacionados con la revisión y propios de la construcción y aplicación del algoritmo, también, adiciona una comprensión de dos conceptos específicos: 1. el algoritmo genético en términos de su definición, caracterización y construcción y 2. el *benchmarking* en términos de las diferentes técnicas y su posterior aplicación (Camelo García & Suárez Suárez, 2021).

Igualmente, en el mismo año, Gómez Mercado, P. en su investigación "*Modelo de gestión para determinar plan de cambio de bastidores y orugas aplicando machine learning en la flota de tractores de orugas de cerrejón*" aborda problemas relacionados con ineficiencia en compras de mantenimiento en el cerrejón, a partir de un modelo de ML, específicamente, una máquina de soporte vectorial, cuyo fin es clasificar las condiciones de dos componentes de tractores de orugas: el bastidor y la oruga (Desgaste y normal). Este trabajo aporta un refuerzo del concepto de ML en

la inteligencia artificial, igualmente da una descripción de su clasificación, detallando el aprendizaje no supervisado y el método de *Machine Learning* usado (Gómez Mercado, 2021).

Además, en 2020, Carrillo Tarazona, C. en su tesis titulada "*Revisión de estrategias basadas en inteligencia artificial y machine learning para el control de la operación de sistemas de distribución eléctrica*" aborda la transición del sector eléctrico mediante la inteligencia artificial. El autor indaga sobre los diferentes métodos basados en ML usados para el diseño y aplicación de los sistemas de generación, trasmisión y distribución eléctrica inteligentes, estos desarrollos se enfocan en el apoyo a decisiones de la misma red, específicamente, en su diagnóstico, pronóstico de fallas, restauración, mantenimiento, gestión e integración. Lo anterior, apoya las definiciones previamente adquiridas sobre el *Machine Learning* como herramienta de la inteligencia artificial, además de incluir las diferentes técnicas asociadas a las diversas aplicaciones en el sector eléctrico (Carrillo tarazona, 2020).

Adicional a lo anterior, en 2018, Martínez Lizarazo, S. en su tesis titulada "*Sistema de recomendación para productos bancarios con técnicas de machine learning*", se enfoca en el sector bancario, dando solución a problemas usuales en las organizaciones, en marketing y ventas, mediante un sistema de recomendación personalizado desarrollado con técnicas de ML. Igualmente, las aportaciones obtenidas se fundamentan en aspectos como la exploración y procesamiento de los datos, las definiciones relacionadas con ciertos algoritmos de *Machine Learning*, específicamente, los árboles de decisión y bosques aleatorios (*Random forest*) y las métricas de desempeño para evaluar los algoritmos (Lizarazo, 2018).

En cuanto a investigaciones realizadas a nivel nacional, en el 2020, Muñoz N. y Romero J. en la investigación denominada "*optimización de los hiperparámetros de una máquina de regresión de soporte vectorial utilizando enjambre de partículas para el pronóstico de casos de*

COVID-19", aplicaron la optimización a los hiperparámetros de una máquina de regresión de soporte vectorial (SVR), mediante la adaptación de algoritmo metaheurístico *Particle Swarm Optimization* (PSO) para estudiar las series de tiempo del total de casos positivos acumulados par el COVID-19 en la ciudad de Bogotá. Para validar dicho pronóstico, se comparó SVR con optimización de hiperparámetros y SVR con hiperparámetros por defecto, con lo cual, se pudo comprobar la efectividad y el desempeño de los métodos de optimización hiperparamétrica. En relación con los aportes al tema en cuestión, adiciona la optimización de hiperparámetros; además, da las directrices y pasos para aplicar el PSO a las máquinas de *Machine Learning*, señalando la importancia de la optimización de hiperparámetros en estos modelos (Muñoz Cañón & Romero Triana, 2021).

Por otro lado, a nivel internacional, la investigación realizada por Javeed et al. en 2019 titulada "*An Intelligent Learning System Based on Random Search Algorithm and Optimized Random ForestModel for Improved Heart Disease Detection"* se propone un sistema de aprendizaje híbrido denominado (RSA-FA) que combina dos algoritmos: búsqueda aleatoria (RSA) y bosque aleatorio (RF). Los autores concluyeron que el método propuesto es eficiente y menos complejo que el modelo de bosque aleatorio convencional, debido a que produce un 3,3 % más de precisión y usa solo siete características. En conclusión, como aspectos importantes para tener en cuenta para el proyecto actual, se destaca el problema del sobreajuste, que se relaciona con el problema principal como el aumento en la precisión de detección cardiaca en datos de prueba y disminución en los datos de entrenamiento, también, incluye la optimización hiperparamétrica dando las pautas para aplicar el RSA y el RF(Javeed et al., 2019).

Finalmente, en 2022, Mohan & Badra en su investigación "*A novel automated SuperLearner using a genetic algorithm-based hyperparameter optimization"* desarrollan un

nuevo modelo automatizado de *SuperLearner* (SL) que utiliza un algoritmo genético (AutoSL-GA) basado en la optimización de hiperparámetros (HPO), debido a dos principales retos ocasionados de la industria 4.0 a raíz de la utilización de modelos de *Machine Learning*: la falta de científicos y el alto costo de obtener datos etiquetados. A partir de las pruebas y comparaciones realizadas, obtuvieron un resultado más eficiente en cuanto a velocidad, rendimiento y capacidad (Mohan & Badra,2022). Como principales aportes se tiene la descripción de hiperparámetros para máquinas de ML, específicamente el ANN, AVM, ENR, KRR, LGB y CBR, también detalla un proceso metodológico para la optimización hiperparamétrica de estas máquinas (Mohan & Badra, 2023).

Como conclusión, se evidencia que a nivel nacional e internacional el tema de investigación ha tomado importancia. Sin embargo, internacionalmente se encuentra más desarrollado, en términos de la optimización de hiperparámetros. Por ello, es fundamental indagar profundamente a nivel local estos temas, lo cual, es fundamento del presente trabajo.

5.2 Marco Teórico

A continuación, se presentan los principales conceptos asociados a la optimización de hiperparámetros dentro del campo de AutoML. Luego, se indica la manera cómo se desarrolla la búsqueda de la mejor combinación de hiperparámetros para un modelo de *Machine Learning* en aprendizaje supervisado.

5.2.1 Auto Machine Learning

Un sistema de *Automated Machine Learning* (AutoML) es un método que posibilita la construcción y desarrollo de modelos de *Machine Learning*, construyendo un conjunto de modelos ordenados en función de su desempeño. Con el fin de garantizar un resultado con un nivel predictivo óptimo, el modelo de un sistema de AutoML, en primera instancia, se debe centrar en

la elección de los datos a estudiar e identificación de sus características. Posteriormente, se debe optar por la selección de los criterios de validación a través de la revisión y elección de las métricas que se utilizarán para evaluar los modelos y por último se establecen las restricciones a partir del problema a manejar. Lo cual determina el espacio de búsqueda, generándose un conjunto de posibles opciones o configuraciones que el algoritmo evalúa a partir de las condiciones planteadas (Management solutions, 2020).

El objetivo del AutoML no solo es automatizar las tareas en las que los procesos heurísticos son limitados, sino también generar técnicas de búsqueda de patrones y de algoritmos más automáticos, con una mejor trazabilidad. Los sistemas de *Automated Machine Learning* constituyen diversas herramientas para ejecutar modelos, reduciendo costes, tiempo de desarrollo y posibles errores en la implementación de dichos sistemas (Management solutions, 2020).

Según Management solutions (2020), para el desarrollo de los componentes de un sistema de AutoML se debe:

1. Automatizar los aspectos más relevantes del análisis, teniendo en cuenta el tratamiento de los datos, la transformación de las variables y la optimización de sus hiperparámetros.

2. Definir un espacio de búsqueda que incluya posibles modelos de *Machine learning* y parámetros, en donde se pueda comparar y elegir los mejores modelos, mediante criterios establecidos durante su construcción.

3. Por último, automatizar las técnicas de análisis para evaluar el modelo elegido, con el fin de permitir la correcta interpretación por parte de los distintos usuarios.

Finalmente, se busca obtener un sistema que de manera automatizada posibilite hallar determinadas tendencias en los datos con la mínima intervención humana, dando solución a tareas de mayor complejidad. En donde cada proceso sea más eficiente y robusto, teniendo en cuenta

restricciones computacionales y tiempos de ejecución. En consecuencia, las ventajas del AutoML provienen del adecuado proceso de HPO, pertenecientes o referentes a su canalización; Preparación de los datos, elección de los atributos, la mejor selección de espacio de búsqueda, el proceso de HPO y finalmente su evaluación (Wistuba et al., 2021).

5.2.2 Hiperparámetros

Dentro de los algoritmos de *Machine Learning* en el contexto del AutoML, se encuentran dos tipos de parámetros: los parámetros del modelo, que son aquellos que se inicializan y actualizan en el proceso de aprendizaje, como por ejemplo los pesos de las neuronas en las redes neuronales, y aquellos de los cuales depende el modelo cuando se instancia, denominados hiperparámetros. Estos últimos definen la estructura del modelo de ML, y debido a esto no se pueden estimar en el aprendizaje de datos. Por ejemplo, en una máquina de regresión de soporte vectorial (SVR) un hiperparámetro sería el tipo de Kernel, el cual, a valores muy bajos ocasiona un sobreajuste y a valores altos ocasiona una pérdida de la captura de la complejidad de los datos. Por tanto, los hiperparámetros que configuran la arquitectura del modelo de ML, definen en gran medida su rendimiento, por ello, el AutoML busca optimizarlos, mediante el uso de técnicas de optimización aplicadas al espacio de búsqueda (L. Yang & Shami, 2020).

Por consiguiente, el conjunto de valores que pueden asumir los hiperparámetros se denomina espacio de búsqueda, también conocido como espacio de configuración, siendo el conjunto de todas las posibles configuraciones de hiperparámetros, en donde se busca el óptimo para realizar la mejor predicción posible (Management solutions, 2020). Todas las configuraciones posibles de un método crean un espacio de hiperparámetros que puede ser discreto, continuo o mixto, dependiendo del tipo de hiperparámetros que puedan establecerse. Los algoritmos de

búsqueda van desde los estocásticos, pasando por la búsqueda sistemática exhaustiva hasta métodos más complejos como los algoritmos evolutivos.

Por esta razón, con el fin de ajustar determinado conjunto de hiperparámetros y lograr un impacto significativo en la fase de entrenamiento de un modelo de ML, la optimización matemática puede ayudar a este propósito. La optimización matemática es definida por algunos autores como el proceso de minimizar/maximizar uno o más objetivos sin infringir las restricciones especificadas mediante la regulación de un conjunto de variables de decisión que influyen tanto en los objetivos como en las restricciones (Vagaská et al., 2022). Es decir, es aquel proceso donde se busca el valor mínimo o máximo de uno o varios objetivos, que se encuentran sujeta a ciertas restricciones, cabe aclarar que ambos se encuentran definidos en base a variables de decisión que deben determinarse con el modelo mismo. El objetivo final que persigue la optimización es minimizar el esfuerzo, la energía y los costes consumidos o maximizar el beneficio deseado, que puede expresarse como una función objetivo de determinadas variables de decisión" (Vagaská et al., 2022). Por tanto, la optimización apoya en cierta manera a la toma de decisiones, ya que proporciona una manera de obtener una solución a cierta problemática de manera óptima. La optimización matemática puede ser aplicada a distintas situaciones en el ámbito empresarial, en ingeniería hace posible el predecir el comportamiento de ciertos sistemas o procesos para poder controlarlos; en la ciencia de los materiales optimiza el rendimiento; en tecnología es aplicada en optimización de procesos tecnológicos reduciendo la energía consumida. En general, la optimización proporciona las diferentes herramientas sin importar el campo para encontrar la mejor solución a un problema en específico a partir del modelado matemático (Vagaská et al., 2022).

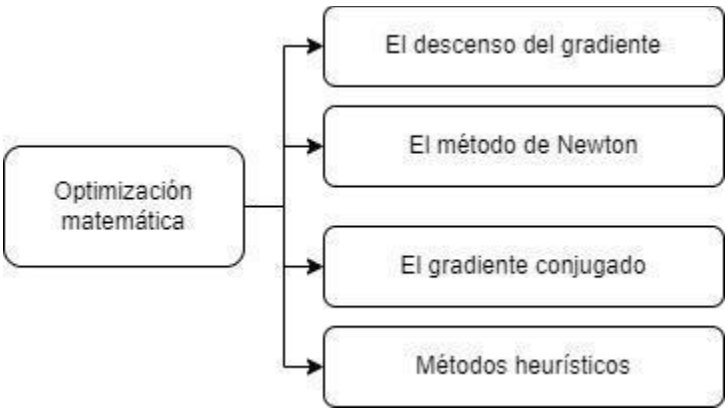
Los métodos tradicionales usados para solucionar un problema de optimización matemática se presentan en la Figura 6. Sin embargo, para casos en que la función objetivo no es

convexa se presentan ciertos inconvenientes como, el punto crítico podría ser un óptimo local en lugar del global lo que ocasiona que no se llegue a la solución óptima. Otro factor es que solo funcionan para variables numéricas o continuas y finalmente no son óptimos en cuanto a los recursos que se deben invertir para su ejecución. Por tanto, es importante conocer otros métodos de optimización que mejoran los inconvenientes presentados (L. Yang & Shami, 2020).

En síntesis, la optimización mediante aprendizaje automático no solo trae mejoras en desarrollos científicos sino también en la toma de decisiones y crecimiento empresarial a partir de la optimización de procesos, aportando ventajas tales como el mejor uso de los recursos para aumentar la eficiencia, avance en los flujos de trabajo, reducción de costos, mejoras en la comunicación y predicción. En concordancia, gestionar la optimización de los procesos es fundamental para la transformación digital en todas las empresas, con la optimización se puede mitigar el riesgo mediante el mapeo de actividades que facilitan la estandarización de procesos y su formalización, reduciendo los errores, repeticiones y dudas en los procedimientos, lo que reduce notablemente los riesgos (Muñoz Cañón & Romero Triana, 2021).

Figura 6

Tipología métodos de optimización matemática tradicionales.



Nota. Adaptado de Yang & Shami (2020)

5.2.3 Optimización de hiperparámetros (HPO)

La optimización de hiperparámetros se usa con el fin de mejorar el rendimiento de los métodos de ML de forma eficaz, ya que, de cierta manera, ajusta de manera ideal la arquitectura de estos métodos. Su desempeño depende del *tunning*(ajuste) de los hiperparámetros. Por lo tanto, al ajustar mediante técnicas de optimización se reducen errores manuales; además, se mejora el rendimiento de los algoritmos de ML y se generan modelos e investigaciones óptimas.

La mayoría de los procesos de optimización de hiperparámetros tiene cuatro componentes según Yang & Shami (2020):

1. El estimador (para una regresión o para una clasificación) con su función objetivo.
2. El espacio de búsqueda o también llamado espacio de configuración de parámetros.
3. Un método de optimización que se usa para hallar la combinación de los hiperparámetros.
4. Una función de evaluación que permite comparar el desempeño para diferentes configuraciones de los hiperparámetros.

Teniendo en cuenta los componentes anteriores, los procesos de optimización de hiperparámetros tienen que seguir la siguiente metodología propuesta por Yang & Shami (2020) :

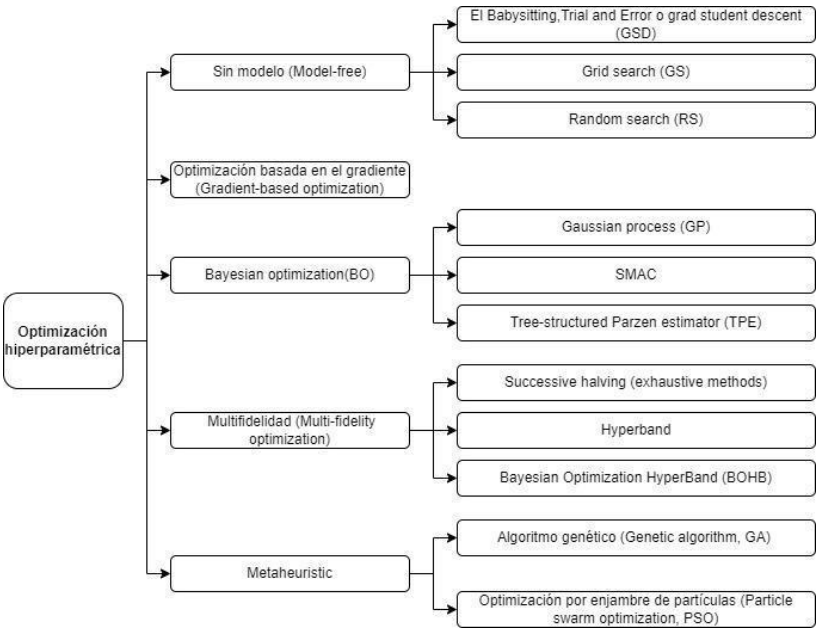
- Seleccionar la función objetivo y las métricas de rendimiento. (Algoritmo de ML y función métrica de rendimiento).
- Seleccionar los hiperparámetros que requieren ajuste, resumir sus tipos y determinar la técnica de optimización adecuada. (Definir la técnica de optimización como los hiperparámetros a optimizar).
- Entrenar el modelo ML utilizando la configuración de hiperparámetros por defecto o los valores comunes como modelo de referencia.

- Comenzar el proceso de optimización con un gran espacio de búsqueda como el dominio factible de hiperparámetros determinado por las pruebas manuales y/o el conocimiento del dominio. (Definir parte del espacio de búsqueda de acuerdo con la técnica de HPO).
- Reducir el espacio de búsqueda basándose en las regiones de valores de hiperparámetros actualmente probados con buen rendimiento, o explorar nuevos espacios de búsqueda si es necesario.
- Devolver la configuración de hiperparámetros de mejor rendimiento como solución final.

Los métodos de optimización de hiperparámetros más comunes aplicados a distintos modelos de ML se muestran en la Figura 7. La tipología se basa en función de las distintas características propias de los hiperparámetros (L. Yang & Shami, 2020).

Figura 7

Tipología métodos de optimización hiperparamétrica



Nota. Adaptado de Yang & Shami (2020)

5.2.4 Machine Learning

En general, los modelos de *Machine Learning* tienen una de las direcciones más relevantes en el desarrollo de la tecnología moderna (Prudius et al., 2019). Además, representa una de las técnicas más aplicadas dentro del campo de la inteligencia artificial debido a que son genéricos y demuestran un alto rendimiento en problemas de análisis de datos (Yang & Shami, 2020). El *Machine Learning* tiene gran cantidad de aplicaciones tales como la publicidad, los sistemas de recomendación, la visión por ordenador, el procesamiento del lenguaje natural, análisis de comportamiento, entre otras. (L. Yang & Shami, 2020). De igual modo, esta amplia gama de aplicaciones seguirá creciendo principalmente debido a la acumulación de enormes cantidades de datos en los sectores científico, industrial, de transporte y empresarial (Carrillo Tarazona, 2020).

Por lo general, las técnicas de ML, principalmente, estudian tres problemas fundamentales según Prudius et al. (2019):

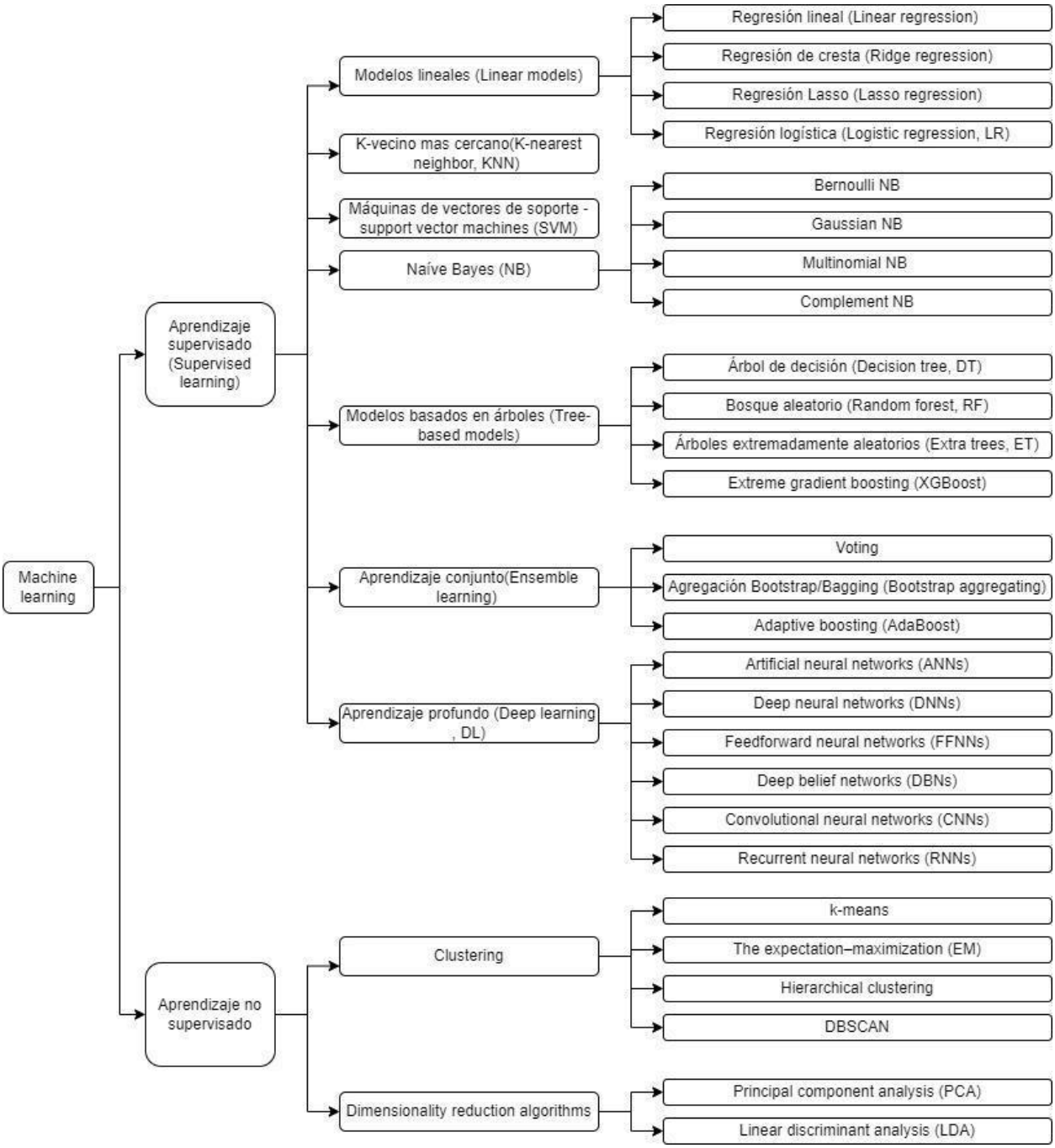
- El *clustering* cuyo objetivo es dividir una gran cantidad de individuos, objetos, en grupos (llamados *"clusters")* que sean los más similares en el grupo, pero con grandes diferencias entre sí.
- La clasificación que divide un número determinado de objetos en clases predefinidas mediante uno o más parámetros.
- La regresión, que estudia el impacto de una o distintas variables independientes sobre la variable dependiente.

En cuanto a su tipología, la Figura 8 resume las principales técnicas utilizadas. Dichos métodos resuelven uno o más problemas formulados anteriormente; por ejemplo, el SMV sirve tanto para clasificación como para regresión; el KNN está enfocado en la clasificación y los

modelos lineales de regresión, se enfocan en la predicción, entre otras aplicaciones (L. Yang & Shami, 2020).

Figura 8

Tipología métodos de Machine learning



Nota. Adaptado de Yang & Shami (2020)

5.2.5 Aprendizaje supervisado

Es aquel aprendizaje que usa datos etiquetados en determinado modelo a entrenar. En donde el algoritmo genera una función de entrada con la salida deseada. Se pueden abordar problemas de clasificación y regresión. Los modelos de clasificación predicen valores discretos o categóricos y los modelos de regresión predicen valores continuos (L. Yang & Shami, 2020).

En concordancia, según Motz et al. (2022) en el aprendizaje supervisado un algoritmo de ML intenta modelar las relaciones entre pares de entradas dados y sus etiquetas. El objetivo principal es lograr la capacidad de generalización, es decir, se busca que el modelo o algoritmo de ML construido, entrenado y probado, tenga la capacidad de pronosticar correctamente la etiqueta de una instancia no vista.

5.2.6 Aprendizaje no supervisado

Es una forma en la que los algoritmos basan su proceso de entrenamiento mediante datos no etiquetados. El algoritmo aprende de los datos de entrada, sin disponer la información sobre las salidas para reconocer patrones. El aprendizaje no supervisado principalmente, permite abordar los problemas con mediana o ninguna idea de visualización de los resultados (L. Yang & Shami, 2020). Se puede derivar la estructura de los datos en los que no necesariamente se conoce el efecto de las variables, estos se agrupan según la relación entre las variables usadas. Este tipo de aprendizaje no se fundamenta en resultados de predicción (Le, 2013).

5.2.7 Aprendizaje Reforzado

El aprendizaje reforzado se implementa en situaciones donde no se dispone de datos pre-etiquetados o clasificados. Es usado para resolver problemas en escenarios dinámicos e inciertos, en donde el sistema aprende en un entorno donde no hay información sobre la posible salida (Rojas, 2020).

5.2.8 Metaheurística

Las metaheurísticas se han aplicado para mejorar el aprendizaje de *Machine Learning*, específicamente en la optimización de sus parámetros o de configuraciones asociadas a su diseño (Muñoz Cañón & Romero Triana, 2021). Algunas técnicas de este método se basan en la población, concretamente en sus comportamientos. Algunos ejemplos son los algoritmos PSO y GA, que se derivan de teorías biológicas, los cuales, fueron usados inicialmente para estudiar el comportamiento de diferentes poblaciones, y actualmente se ha comprobado su amplia aplicabilidad para diferentes ramas de investigación y problemas de optimización complejos (L. Yang & Shami, 2020). A continuación, se evidencian en la Figura 9 los métodos más usados.

Tabla 4

Métodos metaheurísticos de optimización

Stochastic Optimization Algorithms
Evolution Strategy (ES)
Differential Evolution (DE)
Violation Learning Differential Evolution (VLDE)
Genetic Algorithm (GA)
Simulated Annealing (SA)
Particle Swarm Optimization (PSO)
Predator-prey Pigeon-inspired Optimization (PPPIO)
Ant Colony (AC)
Artificial bee colony (ABC)
Tabu Search (TS)
Harmony search (HS)

Nota. Tomado de Vagaská et al. (2022)

5.2.9 Árboles de decisión

Son métodos de aprendizaje estadístico, en donde los árboles de decisión parten de una raíz o *root* y acaban en diferentes nodos. En cada decisión, el árbol se divide en dos nodos en función de la variable de regresión o clasificadora más importante. Dependiendo del tipo de la variable de estudio, esta se puede clasificar como categórica o continua, esto cambia el tipo de árbol, siendo el objetivo de los primeros árboles la clasificación y de los segundos la regresión (Alai et al., 2021).

Su uso en el manejo de datos brinda facilidad a cualquier usuario en cuanto a la comprensión de sus resultados. Cuando se obtiene un árbol, este determina una regla de decisión. Según Alai et al., (2021) esta técnica permite:

- Segmentación, establece los grupos importantes para agrupar.
- Clasificación, asigna a cada partición de una población los ítems correspondientes.
- Predicción, se establecen reglas para pronosticar eventos.
- Reducción de la dimensión de los datos, identifica información relevante para el diseño de un modelo.
- Identificación-interrelación, identificar variables y relaciones significativas para grupos identificados a partir de análisis de datos.
- Recodificación, transformar variables cualitativamente sin perder información relevante.

5.2.10 Random Forest

Random Forest es una técnica basada en la combinación de múltiples árboles de decisión, tiene la ventaja de utilizar muchas variables de distintos tipos como categóricas, numéricas y regresoras (Alai et al., 2021). *Random Forest* son árboles independientes uno de otro, es decir, no

se necesita saber cómo le fue a un árbol para entrenar el siguiente, según Alai et al., (2021) los componentes más importantes de RF son:

5.2.10.1 Propiedades y características.

Algunas características de RF son, su amplia aplicación en el campo investigativo, debido a la capacidad que tienen de soportar grandes conjuntos de datos y diversidad de hiperparámetros, tales como categóricos, numéricos y discretos, en donde se suele combinar distintas variables sin el requerimiento de un modelo paramétrico (Alai et al., 2021).

5.2.10.2 Medida de proximidad.

Se presenta entre dos observaciones, dada mediante la proporción de árboles, por lo cual existe la posibilidad de que se encuentren o se separen respecto al mismo nodo final, en donde la imputación y visualización del RF depende del alcance de la proximidad. Existe la posibilidad que, dos unidades de los árboles existentes acaben en el mismo nodo, en tal caso, la proximidad será de 1, y en caso contrario, 0. Siendo así, puede obtener un resultado de distancia ajustada que otorga mayor peso a los predictores más relevantes y menor peso a aquellos que no aportan al estudio (Alai et al., 2021).

5.2.10.3 Validación Out Of Bag(OOB).

En cada árbol de RF los individuos que no participan en su construcción se denominan *Out of Bag observations*, estas aportan en el cálculo del error de estimación y la relevancia de los predictores. El error de generalización se halla con árboles en los que, los individuos no han participado para cada uno de ellos, es decir, aquellos *Out of Bag observations*. A nivel computacional, es más eficiente que otros tipos de validación como el *bootstrap* o la validación cruzada, ya que, el método por sí mismo calcula el error de generalización OOB automáticamente. El error hallado generalmente se usa para encontrar la cantidad de árboles adecuada para minimizar

el error de generalización y reducir el tiempo de procesamiento y así establecer el MTRY (Número de regresores que tendrá cada división) óptimo (Alai et al., 2021).

6. Metodología

Para dar cumplimiento a los objetivos propuestos, se tomó como referencia la metodología de AutoML planteada en Management solutions (2020). A continuación, se enuncian las etapas que se llevan a cabo:

1. Identificación y planificación, se realiza una revisión del tema principal a estudiar, y posteriormente una determinación o planteamiento de los objetivos a cumplir.

2. Preparación de los datos, realizando una selección y definición del dataset junto con la generación variables a incluir el modelo.

3. Desarrollo del modelo, se selecciona el modelo a utilizar, teniendo en cuenta los criterios y el ajuste de los hiperparámetros. Se pretende una adecuada combinación que sea ajustable al modelo seleccionado.

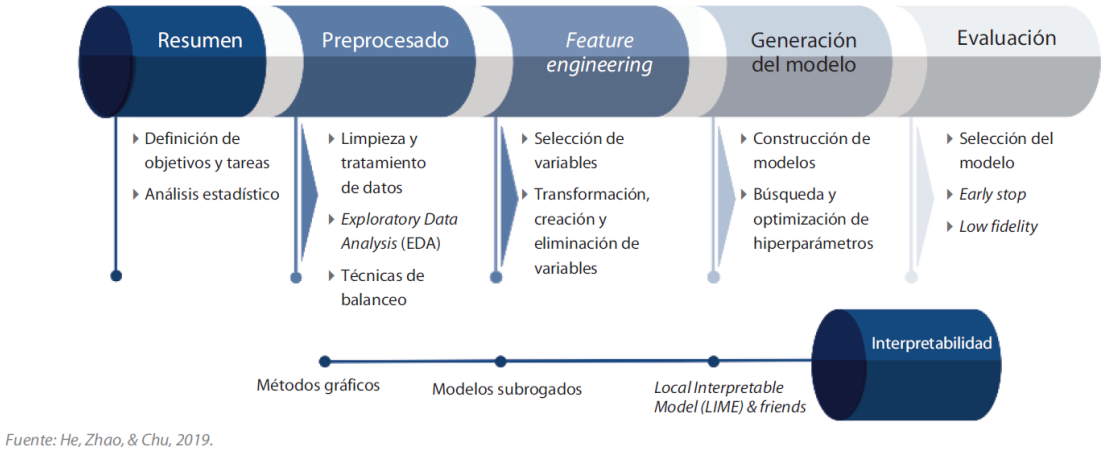
4. Evaluación, validación y aprobación, para verificar la consecución de los objetivos planteados, junto con el rendimiento del modelo seleccionado.

5. Aplicación del modelo aprobado al proceso o problema a solucionar, a partir de los estándares requeridos por las partes interesadas en la investigación. Así mismo, se lleva a cabo un seguimiento y monitorización de los resultados.

Otro factor para considerar fueron los componentes del sistema de AutoML. Los cuales se muestran en Figura 9.

Figura 9

Componentes de un sistema de AutoML



Nota. Tomado de Management solutions (2020).

Para el presente proyecto, a continuación, se plantea cada una de las fases y actividades referentes a la metodología usada para su desarrollo:

6.1 Fase 1: Conceptualización.

- Selección de *Scopus* como base de datos, para realizar la búsqueda de documentación científica, encontrando de esta manera diferentes desarrollos investigativos que hasta la fecha han tenido gran relevancia.
- Construcción de la ecuación de búsqueda que permita abordar desde la conceptualización del tema principal de investigación hasta la aplicación de técnicas de optimización de hiperparámetros en problemas de *Automated Machine Learning*.
- Realizar una revisión de literatura sobre las diferentes técnicas de espacios de búsqueda, mediante: 1. Un análisis bibliométrico a partir de los resultados obtenidos en Scopus cuyo fin será examinar ciertos aspectos relevantes entorno a la optimización de hiperparámetros, *Machine Learning* y *Automated Machine Learning*. Para identificar las palabras clave, principales autores y sectores de aplicación, se utiliza el *software VOSviewer*. 2. Una revisión sistemática de

los diferentes artículos asociados a la ecuación de búsqueda para sintetizar la evidencia científica referente al tema central de investigación.

6.2 Fase 2. Modelamiento.

- Categorización de los modelos de espacios de búsqueda definiendo sus principales características, metodología e implicaciones técnicas.
- Selección de los métodos de espacios de búsqueda que mejor se adecúen al modelo de clasificación *Random Forest*, encontrando la combinación óptima respecto al conjunto de valores, que permita la adecuada inicialización de este. Por ende, se realiza un análisis de investigaciones enfocadas en AutoML y HPO, con el objetivo de hallar la frecuencia e importancia de las técnicas de espacio de búsqueda que mejor se han adecuado a la máquina de *Random Forest*. Teniendo en cuenta las técnicas de espacios de búsqueda categorizadas, tales como Optimización Bayesiana (BO), Algoritmo Genético (GA), Tree-structured Parzen estimator(TPE), Sequential Model-based Algorithm configuration(SMAC) y PSO (Particle Swarm Optimization) o más conocido como optimización por enjambre de partículas, para determinar cuál de ellas permiten realizar una mejor búsqueda dentro del espacio.
- Selección de los hiperparámetros involucrados en el modelo, a partir de las técnicas de optimización encontradas en el estudio de frecuencia de aplicación.

6.3 Fase 3. Ejecución y análisis.

- Construcción de los espacios de búsqueda, mediante la definición de las variables de estudio en el dataset obtenido en la plataforma de OpenML y características propias del algoritmo de optimización hiperparamétrica.
- Elección del método de optimización de hiperparámetros que mejor se ajuste a la máquina de *Random Forest*, evaluando el rendimiento de las técnicas elegidas, a partir de: 1.

Métricas de error que son utilizadas para validar los algoritmos de clasificación de ML, como lo son, Sensibilidad y Precisión y 2. Función de aptitud, a fin de establecer una comparación.

- Validación de la estrategia de construcción mediante el dataset asignado para esta etapa.

7. Formulación del modelo de AutoML

Los algoritmos de aprendizaje automático dependen de los valores de hiperparámetros (HPs) elegidos para el aprendizaje del modelo. Estos, influyen sustancialmente en la complejidad, el comportamiento y la velocidad de los algoritmos de ML, por lo cual, los valores deben seleccionarse cuidadosamente para lograr un rendimiento óptimo. Por ello, surgen las técnicas de HPO o de espacios de búsqueda, entre las cuales se encuentran metodologías sencillas como la búsqueda en cuadrícula y aleatoria; y métodos más avanzados como estrategias de evolución (*Evolution Strategies*), Optimización Bayesiana (*Bayesian Optimization*), Hiperbanda(*Hyperband*) y Carreras(*Racing)* (Bischl et al., 2023;Andonie, 2019). Estas se implementan como método para ajustar los espacios de búsqueda y así detectar la mejor configuración de hiperparámetros (HPC).

Las técnicas suelen caracterizarse a través de dos aspectos cruciales: primero, la forma de gestionar el equilibrio entre exploración y explotación en relación con zonas del espacio de búsqueda y segundo, el manejo de la inferencia y búsqueda, respecto a la cantidad de tiempo y gastos generales. En concordancia, un aspecto a considerar en los espacios de configuración, son las dependencias existentes entre hiperparámetros, un ejemplo se da en una máquina de soporte vectorial (*Support Vector Machine*-SVM) donde el tipo de Kernel influye en el valor óptimo de los demás hiperparámetros. En el caso mencionado, se dice que el espacio de búsqueda es jerárquico (Bischl et al., 2023;Andonie, 2019)

En relación con lo anterior, es de resaltar las siguientes características o funcionalidades, en las que los algoritmos de HPO difieren según Bischl et al. (2023):

- Paralelización: Ejecutar evaluaciones de configuraciones de hiperparámetros al mismo tiempo.
- Comportamiento del optimizador: Revisar actualizaciones de proximidad en las configuraciones previamente evaluadas.
- Gestión del ruido: Verificar que el error de generalización estimado no presente sesgos.
- Multifidelidad: Ejecutar modelos con evaluaciones que impliquen menor tiempo y recursos, por ejemplo, subconjuntos más pequeños de los datos, para inferir en el rendimiento del dataset total.
- Complejidad del espacio de búsqueda: Manejar diferentes tipos de espacio de búsqueda de parte del optimizador.

A su vez, Bischl et al. (2023) menciona que en los algoritmos de HPO se pueden presentar algunos inconvenientes de aplicación, como lo son:

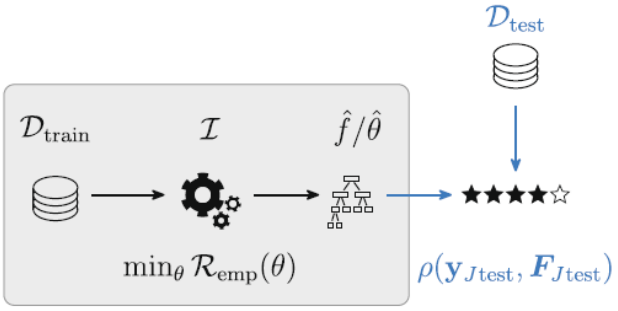
- Escasa interpretación de los métodos de HPO por parte de los usuarios, debido a que pueden tomarlas como cajas negras demasiado complejas.
- Incertidumbre en la implementación de HPO como método de solución, puesto que, se espera que los resultados obtenidos sean proporcionales a la inversión de recursos económicos, tecnológicos y de tiempo.
- Falta de orientación sobre la elección y configuración de los métodos de HPO según el problema a solucionar.

- Dificultad para definir adecuadamente el espacio de búsqueda de un proceso de HPO.

Dada la aplicación de un modelo de aprendizaje supervisado, como lo es en este caso, es importante destacar que para ajustar un modelo y reducir los riesgos de ajuste, se utiliza el principio de minimización del riesgo empírico (ERM), un trade-off entre el error de aprendizaje y el error previsto. En la Figura 13, se muestra el proceso explicativo del principio de ERM, en donde, se busca obtener un modelo ajustado para minimizar el riesgo empírico, iniciando con la toma los datos de entrada, posteriormente se realiza el proceso de minimización y se devuelve el modelo que se ajustó junto con los parámetros definidos para el modelo (Bischl et al., 2023).

Figura 10

Principio de minimización del riesgo empírico



Nota. Tomado de Bischl et al. (2023)

Adicionalmente, si en el proceso anterior el equilibrio estadístico depende de los tamaños relativos de la división y el tamaño completo de un conjunto de datos, los métodos de remuestreo deben ser usados para mejorar el rendimiento de los valores obtenidos. Lo cual permite solucionar parcialmente el sesgo generado por el conjunto de datos, dividiéndolos periódicamente en conjuntos de entrenamiento y prueba aplicando el estimador para cada uno de ellos y finalmente, agregando todos los valores de rendimiento obtenidos. Por consiguiente, el remuestreo es un

aspecto importante a tener en cuenta, ya que utiliza los datos con mayor efectividad, al revisar la estimación repetida y el promedio de múltiples resultados de la estimación del error de generalización con menor varianza (Bischl et al., 2023).

En relación con las discusiones anteriores sobre aspectos relevantes en la HPO, posteriormente se describen las técnicas de ajuste de hiperparámetros que han sido utilizadas frecuentemente, con el fin de solucionar problemas de diversos sectores económicos que necesitan optimizar sus procesos, como lo han sido principalmente *Model-free, Gradient-based, Bayesian Optimization (BO), Multi-fidelity* y *Evolution strategies* (ES).

8. Técnicas de hyperparameter optimization (HPO)

Dada la necesidad de obtener los mejores resultados del proceso de optimización de hiperparámetros de una máquina de *Random Forest*, se lleva a cabo la siguiente categorización de las técnicas de hiperparámetros en función de su idoneidad, relacionada a la variabilidad de las características de las máquinas y el problema que se precisa solucionar; esta clasificación se realiza mediante el análisis de diversas aplicaciones y desarrollos, incluyendo modelos sencillos y robustos. De esta manera, se busca identificar y seleccionar la mejor técnica para garantizar una optimización efectiva a partir de las características y metodologías que mejor se adecúen a la construcción del espacio de búsqueda, el cual permitirá hallar la combinación de hiperparámetros óptima**.**

8.1 Model -free

8.1.1 Babysitting

La técnica de ensayo y error *(Babysitting o Trial and Error)* es una técnica tradicional usada en la optimización de hiperparámetros (H*yperparameter optimization*). Esta evalúa cada configuración del hiperparámetro del espacio de búsqueda de manera manual. Por consiguiente,

este método requiere buen conocimiento y experiencia del investigador, de ello dependerá la eficiencia en utilización de tiempo y recursos. Sin embargo, en algunos casos no es factible su aplicación, debido a diversas limitantes como espacios de configuración complejos, elevado número de tipos y dependencias de hiperparámetros, evaluaciones costosas en tiempo y recursos (L. Yang & Shami, 2020).

Su funcionamiento es sencillo comparado con otras técnicas de HPO. Inicialmente, al finalizar la construcción del modelo de *Machine Learning*, se evalúa una configuración de hiperparámetros dada, fundamentada en el criterio del investigador. Posteriormente, se prueban diversas configuraciones de hiperparámetros buscando encontrar aquella que más se adecúe a los objetivos y limitantes. Comúnmente, los tiempos y recursos para encontrar la mejor configuración en el proceso manual tienden a ser insuficientes, entonces, a lo largo del presente documento se categorizan y caracterizan los principales y diversos métodos automáticos de HPO que solucionan este inconveniente (L. Yang & Shami, 2020).

8.1.2 Grid search (GS)

La búsqueda en cuadrícula o más conocido como *Grid search*, es un proceso ampliamente aplicado en el campo de ajuste de hiperparámetros. Este explora de manera exhaustiva el espacio de búsqueda de un modelo de ML, evaluando todas las configuraciones de hiperparámetros en una red construida dentro de dicho espacio. Esta red se constituye por el producto cartesiano de distintos valores de hiperparámetros cuyos rangos están discretizados. En la práctica, GS es un método sencillo de aplicar, sin embargo, su alto tiempo de convergencia impide su aplicación a problemas con presupuestos limitados (Bischl et al., 2023).

8.1.2.1 Metodología

GS no puede explorar de manera óptima el espacio de búsqueda por sí solo, por tanto, la metodología necesaria para explorar regiones óptimas es: Primero, definir el espacio de búsqueda total a explorar referente a sus hiperparámetros y valores, así como el tamaño de la red, garantizando que sean significativos en tamaño. Segundo, realizar las evaluaciones de las configuraciones de hiperparámetros en la red y espacio de búsqueda definido. Posteriormente reducir el espacio de búsqueda a explorar y el tamaño del paso, mediante la evaluación anterior e iterar hasta alcanzar la configuración del hiperparámetro óptima (L. Yang & Shami, 2020).

8.1.2.2 Propiedades

Las características primordiales al aplicar GS en la HPO son:

- Paralelización: Fácilmente paralelizable, debido a sus evaluaciones independientes.
- Comportamiento: El comportamiento de las diferentes evaluaciones en las configuraciones de hiperparámetros no es considerado. *Grid Search* evalúa uniformemente el espacio de búsqueda sin tener en cuenta evaluaciones de configuraciones o combinaciones realizadas.
- Complejidad computacional: Complejidad polinómica $O(n^k)$, dada por k (Número de HPs) y n (Valores de HPs).
- Multifidelidad: *Grid Search* utiliza una red para explorar una parte del espacio de búsqueda, y no su totalidad.
- Espacio de búsqueda: GS maneja espacios de búsqueda pequeños y jerárquicos. Es afectada por la maldición de la dimensionalidad dada por su complejidad.
- Hiperparámetros: Ineficiencia en hiperparámetros continuos, condicionales y enteros. Y eficiente en hiperparámetros categóricos.

- Algoritmos de *Machine Learning***:** Aplicable de manera óptima a los algoritmos *Voting y Bagging*. De igual forma, es aplicable a todos los algoritmos de ML como punto de referencia.

8.1.3 Random Search (RS)

Búsqueda aleatoria o *Random Search,* supera la limitación de GS asociada al espacio de búsqueda. Este puede detectar el óptimo global o aproximaciones en grandes espacios de búsqueda con recursos limitados. Para lograrlo, por medio de una distribución preestablecida, se elige una muestra de HPCs de un espacio de configuración determinado y las evalúa hasta alcanzar el criterio de terminación dado mediante el tiempo de ejecución. En problemas de HPO, proporciona una manera de explorar ampliamente el espacio de búsqueda, por ello es comúnmente usado como punto de referencia. No obstante, su eficiencia se reduce debido a que muchas de estas evaluaciones son innecesarias, al no relacionar comportamientos globales de evaluaciones en la configuración de hiperparámetros (Bischl et al., 2023).

Del mismo modo, *Random Search* suele tener un mejor rendimiento y aplicabilidad que *Grid Search* en entornos de mayor dimensión. Otra ventaja es que puede ampliarse fácilmente con más muestras; en cambio, el número de puntos en una red de *Grid Search* debe especificarse previamente.

8.1.3.1 Metodología

Random Search se rige por el siguiente procedimiento:

- Definir la distribución, el espacio de búsqueda de HPs y el número de HPCs a muestrear.
- Especificar el criterio de finalización dado por el tiempo y recursos disponibles.
- Elegir el conjunto de muestras de HPCs de la distribución especificada.

- Entrenar la primera HPCs y evaluar su rendimiento.
- Iterar hasta el criterio de terminación y elegir la HPC óptima según la evaluación.

8.1.3.2 Propiedades

Las principales propiedades para tener en cuenta para su aplicación son:

- Paralelización: Fácilmente paralelizable, debido a sus evaluaciones independientes.
- Comportamiento: RS no considera las evaluaciones a nivel conjunto, el método evalúa de manera aleatoria e independiente el espacio de búsqueda sin asociar sus precedentes.
- Complejidad computacional: Complejidad de orden lineal $O(n)$, dada por n valores de HPs.
- Multifidelidad: RS reduce el número de evaluaciones, mediante una muestra del espacio de búsqueda dados recursos y tiempo limitados.
- Espacio de búsqueda: RS maneja grandes espacios de búsqueda y jerárquicos.
- Hiperparámetros: Eficiente en hiperparámetros categóricos, continuos y enteros e ineficiente en hiperparámetros condicionales.
- Algoritmos de *Machine Learning*: Aplicable de manera óptima a todos los algoritmos de ML como punto de referencia.

8.2 Gradient-based (GB)

La optimización basada en el gradiente*(Gradient-based)*, es una técnica de HPO tradicional. Se mueve hacia un óptimo identificando la dirección prometedora del hiperparámetro a optimizar mediante un gradiente. A razón de su naturaleza, solo es eficiente para optimizar funciones convexas. Por consiguiente, los métodos *Gradient-based* se aplican para optimizar determinados hiperparámetros en un algoritmo de ML y no a la totalidad de hiperparámetros ya sean discretos, categóricos o dependientes (L. Yang & Shami, 2020).

8.2.1 Metodología

Al aplicar *Gradient-based* se debe tener en cuenta los siguientes pasos: en primera medida se debe seleccionar aleatoriamente una configuración de hiperparámetro. Luego, calcular el gradiente de las distintas direcciones en la configuración y localizar la siguiente configuración moviéndose en dirección opuesta al hiperparámetro con mayor gradiente y finalmente iterar hasta encontrar el óptimo local (Global para para funciones convexas).

8.2.2 Propiedades

Las principales propiedades para tener en cuenta para su aplicación son:

- Paralelización: Difícil de paralelizar, al tener iteraciones dependientes.
- Comportamiento: Gradient-based considera evaluaciones anteriores del gradiente en iteraciones presentes para encontrar los demás puntos a evaluar.
- Complejidad computacional: Complejidad polinómica $O(n^k)$, dada por k (Número de HPs) y n (Valores de HPs).
- Multifidelidad: Gradient-based reduce el número de evaluaciones, al explorar el espacio de búsqueda mediante iteraciones anteriores.
- Espacio de búsqueda: Opera en espacios de búsqueda no jerárquicos.
- Hiperparámetros: Eficiente en hiperparámetros continuos e ineficiente en hiperparámetros enteros, condicionales y categóricos.
- Algoritmos de *Machine Learning*: Aplicable de manera óptima a hiperparámetros continuos en ciertos algoritmos de ML.
- Otros: Puede solo detectar un óptimo local, en lugar de uno global en funciones no convexas.

8.3 Bayesian Optimization (BO)

Los algoritmos de optimización bayesiana son popularmente los más usados. Esta técnica permite seleccionar los hiperparámetros próximos a evaluar, basándose en resultados de iteraciones pasadas. Para tal fin, BO se basa en un modelo sustituto y una función de adquisición. El modelo sustituto ajusta las configuraciones observadas a la función objetivo generando una distribución predictiva, esta es usada en la función de adquisición para encontrarlos, balanceando la exploración y explotación. La exploración consiste en muestreo de configuración de hiperparámetros en nuevas regiones del espacio de búsqueda y explotación toma configuraciones de hiperparámetros en áreas prometedoras ya conocidas. Por tanto, la optimización bayesiana al equilibrar la explotación y exploración evita que regiones óptimas conocidas y desconocidas del espacio de búsqueda (donde probablemente se encuentre el óptimo global) no se detecten. En resumen, BO generalmente detecta configuraciones casi óptimas en pocas iteraciones, lo que proporciona un rendimiento superior a las técnicas anteriormente mencionadas (L. Yang & Shami, 2020).

8.3.1 Metodología

El funcionamiento general de las técnicas de BO es descrito seguidamente: se empieza por elegir y construir un modelo sustituto probabilístico de la función objetivo. Luego, descubrir las configuraciones de hiperparámetros óptimas en el modelo sustituto mediante la función de adquisición. Por último, aplicar y evaluar las configuraciones encontradas por el modelo sustituto en la función objetivo y actualizar el modelo sustituto. Tener en cuenta que se repite la secuencia según el máximo de iteraciones.

8.3.2 Propiedades

Las propiedades para fundamentarse al aplicar los tipos de BO según sus modelos sustitutos a HPO son:

- Paralelización: Difícil de paralelizar, debido a sus resultados dependientes.
- Comportamiento: BO considera resultados previos de configuraciones de hiperparámetros, encontrando aquellas que se evaluarán a futuro, mediante la actualización de la función sustituta.
- Complejidad computacional: Complejidad cúbica para BO-GP $O(n^3)$ y cuasi-lineal para SMAC y BO-TPE $O(nlogn)$.
- Multifidelidad: BO considera evaluaciones menos costosas y eficientes, al basarse en un modelo sustituto cuya ejecución requiere menos recursos que la función objetivo y una función de adquisición que cumple la función de equilibrar el muestreo en zonas óptimas exploradas e inexploradas)
- Espacio de búsqueda: Espacios de búsqueda grandes y jerárquicos. BO-GP es la excepción, puesto que no puede manejar espacios jerárquicos.
- Hiperparámetros: Eficiente en todo tipo de hiperparámetros, ya sean continuos, discretos, categóricos y condicionales. Cabe destacar que BO-GP no es eficiente con hiperparámetros condicionales.
- Algoritmos de *Machine Learning*: Aplicado a cualquier tipo de algoritmo de ML asociado a su flexibilidad, teniendo como principales exponentes SMAC y BO-TPE.
- Otros: Más eficiente que GS y RS al detectar HPCs con valores previamente probados. Así mismo, BO es eficaz para funciones objetivo f estocásticas, no convexas o no continuas.

8.3.3 Modelos sustitutos

Debido a que la elección del modelo sustituto tiene gran influencia en el rendimiento de BO, próximamente se discutirán los tres principales algoritmos de BO: BO-GP, SMAC y BO-TPE.

8.3.3.1 Bayesian Optimization y Gaussian Process (BO-GP)

El proceso Gaussiano *(Gaussian Process-GP*) es un modelo sustituto utilizado comúnmente por BO cuando el espacio de búsqueda es puramente de valor real. En general, el algoritmo BO-GP tiene como asunción que la función objetivo *f* de media μ y covarianza σ^2 es una realización de GP. Por tanto, las predicciones siguen la distribución normal descrita en la Ecuación 2.

$$p(x, D) = N(\hat{\mu}, \hat{\sigma}^2) \quad (2)$$

Donde, *D* es el espacio de búsqueda y *y = f(x)* es el resultado de la evaluación de cada *x* HPC. El proceso llevado a cabo por este algoritmo se describe a continuación:

1. Obtener el conjunto de predicciones del modelo sustituto.
2. Elegir las configuraciones de hiperparámetros a evaluar mediante intervalos de confianza generados.
3. Evaluar las configuraciones elegidas.
4. Añadir las configuraciones o combinaciones probadas a los registros de muestra.
5. Corregir el algoritmo BO-GP mediante los registros de muestra.
6. Finalmente, la metodología se repite hasta el criterio de terminación.

Por otra parte, es importante destacar que BO-GP no puede manejar hiperparámetros condicionales. Además, en la práctica es utilizado para optimizar pocos hiperparámetros continuos con el fin de obtener un buen rendimiento.

8.3.3.2 Bayesian Optimization y Random Forest (BO-RF)

RF es otra función sustituta para modelar *f*. La BO con RF es llamada comúnmente SMAC. Esta tiene dos asunciones relacionadas, la primera, es la existencia de un modelo Gaussiano $N(\hat{\mu}, \hat{\sigma}^2)$ y la segunda que la función de regresión *r(x)* tiene media $\hat{\mu}$ y varianza $\hat{\sigma}^2$. Si ambos se cumplen, entonces para un conjunto de *B* árboles de regresión en el bosque el valor de $\hat{\mu}$ y $\hat{\sigma}^2$ está dado por la Ecuación 3 y Ecuación 4.

$$\hat{\mu} = \frac{1}{|B|} * \sum_{r \epsilon B} r(x) \quad (3)$$

$$\hat{\sigma}^2 = \frac{1}{|B| - 1} * \sum_{r \epsilon \underline{B}} (r(x) - \hat{\mu})^2 \quad (4)$$

La principal metodología utilizada en SMAC como modelo sustituto se muestra a continuación: inicialmente se realiza un muestreo con reemplazo de n instancias del dataset de entrenamiento, para construir *B* árboles de regresión. Posteriormente, se selecciona el nodo de división mediante el número de HPs asignado a cada árbol; se fija el número de instancias y el número de árboles a crecer. Finalmente, $\hat{\mu}$ y $\hat{\sigma}^2$ de cada HPC es estimada.

Adicionalmente, las principales funciones de SMAC se dan en cuanto al manejo de las distintas clases de hiperparámetros (Continuos, discretos, categóricos y condicionales) con menor complejidad que BO-GP. Sin embargo, su rendimiento es inferior a BO-GP al tratar con espacios de búsqueda exclusivamente numéricos.

8.3.3.3 Bayesian Optimization y Tree Parzen Estimator (BO-TPE).

El Árbol Estimador de Parzen *(Tree Parzen Estimator*-TPE) igualmente se maneja como modelo sustituto en optimización bayesiana. Sin embargo, a diferencia de los demás, inicialmente

BO-TPE crea dos distribuciones de densidad *l(x)* y *g(x)* como modelos generativos mostrados en la Ecuación 5. Después, TPE divide las observaciones mediante un percentil *y** en buenos y malos valores, y los dos conjuntos resultantes modelan la Ecuación 5. Posteriormente, estos conjuntos son utilizados por la función de adquisición, la cual refleja la mejora esperada en la relación entre *l(x)* y *g(x)*, y con ella se determinan nuevas configuraciones de hiperparámetros para la evaluación.

$$p(x, D) = \{l(x), si\ y < y*\ \ g(x), si\ y < y* \quad (5)$$

Las ventajas de BO-TPE se dan en que admite naturalmente todo tipo de HPs especialmente los condicionales. También, la complejidad de BO-TPE es menor que la complejidad de BO-GP.

8.3.4 Funciones de adquisición.

El modelo sustituto equilibra el pronóstico y su incertidumbre mediante la función de adquisición. Elegir la función adecuada es un proceso importante en los algoritmos de BO. Las principales funciones de adquisición encontradas en la literatura al aplicar BO son: mejora prevista (Expected *Improvement*-EI) que encuentra la configuración con una probabilidad de mejora mayor a un umbral determinado o mejor a los valores que se han encontrado y límite inferior de confianza (*Lower Confidence Bound*-LCB) cuyo objetivo es reducir la incertidumbre de la función objetivo en lugar de buscar la mejora de la predicción (Bischl et al., 2023).

8.4 Multifidelidad (Multi-fidelity)

Los enfoques multifidelidad se orientan a problemas relacionados a recursos limitados. En general, el problema más común en la HPO es el asociado al largo tiempo de ejecución. Este varía dependiendo de factores como el tamaño del dataset y el espacio de configuración formado por los hiperparámetros a optimizar. Según el tamaño de estos, el tiempo de ejecución en algunos casos puede llegar a ser extremadamente alto, por ello, los enfoques multifidelidad toman un subconjunto

de los observaciones o características del dataset original con el fin de reducir el tiempo de ejecución. El tamaño del subconjunto varía en función del valor de fidelidad asignado, en concreto, cuando se asigna una fidelidad mínima los subconjuntos son más pequeños, lo que implica menor costo y una mayor capacidad de generalización; mientras que en altos valores de fidelidad los subconjuntos son grandes implicando mayor costo y generalización. Igualmente, en estos enfoques las HPCs que impliquen un bajo rendimiento dentro de los subconjuntos se descartan, evaluando así las de mejor rendimiento en el entrenamiento, utilizando de manera óptima los recursos asignados (L. Yang & Shami, 2020). A continuación, se discuten los principales métodos multifidelidad que han tenido éxito en diferentes problemas de HPO:

8.4.1 Successive halving (SH)

Este algoritmo basado en *bandit* (*bandit-based*) supera aspectos negativos presentes en RS y GS que impiden su aplicación, al tener en cuenta restricciones referentes a tiempo y recursos. Teniendo en cuenta lo mencionado, el aspecto principal a tener en cuenta en iteraciones sucesivas de reducción sucesiva a la mitad (*Successive halving),* es la asignación de estos factores (denominados presupuesto(*B*)) a cada configuración de hiperparámetro. Por tanto, el problema central se encuentra en escoger probar menos configuraciones con un *B* mayor o más configuraciones con un *B* menor (L. Yang & Shami, 2020).

8.4.1.1 Metodología

El proceso general de SH empieza por evaluar los n conjuntos de HPCs con presupuestos igualmente determinados, definidos como $b = B/n$. Luego, se elimina la mitad de las HPCs con el rendimiento más bajo según las evaluaciones en cada iteración y se utiliza la mitad restante de HPCs en la próxima iteración con 2 veces el presupuesto del primer paso. ($b_{i+1} = 2 * b_i$). Y finalmente, se itera hasta encontrar la HPC óptima.

8.4.1.2 Propiedades

Los principales beneficios de SH se atribuyen a las siguientes propiedades:

- Paralelización: Paralelización disponible, debido a sus evaluaciones independientes
- Comportamiento: SH tiene en cuenta evaluaciones de iteraciones anteriores y presupuestos asignados, al escoger las mejores configuraciones de hiperparámetros y evaluarlos nuevamente con presupuestos mayores.
- Complejidad computacional: Al utilizarse comúnmente como un subproceso de hiperbanda, la complejidad computacional no se encuentra debidamente especificada.
- Multifidelidad: Aumenta la eficiencia y disminuye el costo de las evaluaciones, al restringir las valoraciones de HPCs en términos de tiempo y recursos.
- Espacio de búsqueda: Maneja espacios de búsqueda no jerárquicos.
- Hiperparámetros: Eficiente en cuanto a hiperparámetros continuos, discretos, categóricos e ineficiente en HPs condicionales.
- Algoritmos de *Machine Learning*: Aplicable de manera óptima a KNN, K-means, clustering, PCA y LDA.
- Otros: Su eficiencia es afectada por el equilibrio entre el número de HPCs y los b de cada configuración.

8.4.2 Hyperband

Hyperband es ampliamente aplicada en el campo de HPO, fundamentado en bandit(*bandit-based*) se considera una versión mejorada de *Random Search*. Su funcionamiento difiere de otras técnicas de multifidelidad al solucionar el problema de eficiencia de *Successive halving*. *Hyperband* elige dinámicamente un número de HPCs, de manera que el número de configuraciones y sus presupuestos se equilibren adecuadamente. Cabe aclarar, que *Successive*

halving es utilizado como una función que asigna el presupuesto total (*B*) al número de configuraciones definidas (L. Yang & Shami, 2020).

8.4.2.1 Metodología

La metodología del algoritmo se describe a continuación:

1. Calcular *bmín* y *bmáx que son* restricciones presupuestarias. Estas se determinan según: 1. Puntos de datos totales, 2. Número de instancias requerido y 3. Presupuestos disponibles.

2. Encontrar el número de configuraciones (n) y el presupuesto asignado a cada configuración (*b)* basándose en *bmin* y *bmax* (pasos 2 y 3).

3. Muestrear las configuraciones de hiperparámetros basándose en *n* y *b* definidos. Luego, procesarlas para muestrearlas en el *Successive halving*, eliminando configuraciones de peor rendimiento (pasos 3 y 4).

4. Finalmente, se itera hasta identificar la HPC óptima.

8.4.2.2 Propiedades

Las principales propiedades de *Hyperband* se discuten seguidamente:

- Paralelización: Paralelización disponible, debido a que trata cada HPC de forma independiente.
- Comportamiento: Hyperband tiene en cuenta de manera equilibrada y dinámica la cantidad de HPCs y sus presupuestos.
- Complejidad computacional. Complejidad cuasi-lineal $O(nlogn)$, la cual hace referencia a una estimación del tiempo de ejecución de un algoritmo en función del tamaño del conjunto de datos que maneja, expresada en términos de la notación "O grande" de la complejidad algorítmica. En particular, la expresión “O(nlogn)" indica que el tiempo de ejecución del algoritmo

aumentará de manera proporcional al producto de n y log n, donde n representa el tamaño del conjunto de datos y log n es el logaritmo en base 2 de n.

- Multifidelidad: Reduce el número de evaluaciones de HPCs innecesarias, al definir de manera óptima el presupuesto necesario.
- Espacio de búsqueda: Maneja de manera eficiente espacios de búsqueda no jerárquicos.
- Hiperparámetros: Eficiente para hiperparámetros continuos, discretos, categóricos e ineficiente en hiperparámetros condicionales.
- Algoritmos de *Machine Learning*: Aplicable de manera óptima a KNN, K-means, clustering, PCA y LDA.
- Otros. Su complejidad es mayor que Successive halving al implicarlo en el algoritmo como subrutina**.** Así mismo, utiliza *Random search* para investigar el espacio de configuración de HPs.

8.4.3 Bayesian Optimization HyperBand (BOHB)

Es una metodología que combina la Optimización Bayesiana (BO) e hiperbanda. Generalmente, *Random search* (RS) es utilizado por hiperbanda para explorar el espacio de búsqueda, sin embargo, este hecho limita el rendimiento y el tiempo de ejecución. BOHB sustituye Random Search por BO, y con esto elimina las limitantes mencionadas aumentando las capacidades del algoritmo (L. Yang & Shami, 2020).

8.4.3.1 Propiedades

A continuación, se discuten las principales propiedades de BOHB:

- Paralelización: Paralelización disponible, debido a que trata cada HP de forma independiente.

- Comportamiento: BOHB cambia el método búsqueda en el espacio de configuración, y con ello proporciona beneficios asociados a BO dentro de cada iteración.
- Complejidad computacional. Complejidad cuasi-lineal $O(nlogn)$.
- Multifidelidad: Al incluir BO en el proceso de búsqueda en cada iteración, se reduce el tiempo de cómputo, además de converger de manera más rápida a la solución.
- Espacio de búsqueda: Trabaja con grandes espacios de búsqueda y jerárquicos.
- Hiperparámetros: Eficiente con todos los tipos de HPs continuos, enteros, condicionales y categóricos.
- Algoritmos de *Machine Learning*: Aplicable de manera óptima a *Support Vector Machine* (SVM) y *Deep Learning* (DL).
- Otros: En BOHB, el modelo sustituto de *Bayesian optimization* (BO) es el árbol estimador de Parzen (*Tree-structured Parzen Estimators*-TPE). Además, es importante tener en cuenta que las evaluaciones con bajos presupuestos en subconjuntos deben representar las evaluaciones de *training set* (conjunto de datos que se utiliza para entrenar un modelo de aprendizaje automático) si se quiere una velocidad de convergencia óptima.

8.5 Evolution strategies (ES)

Las estrategias de evolución *(Evolution strategies*-ES) o metaheurísticas se inspiran en teorías de evolución biológica y poblaciones estocásticas. Las cuales crean, evalúan, actualizan y seleccionan de manera iterada una población (imitando su comportamiento) hasta encontrar el individuo adecuado u óptimo global. El modo de realización de dichas actividades diferencia el tipo de estrategia o algoritmo. Comúnmente, sus principales características se dan en su capacidad de admitir cualquier tipo de hiperparámetros y abordar espacios de búsqueda grandes, por ello son

usados en problemas complejos de HPO. Finalmente, su limitación está asociada a una mayor complejidad respecto a otro algoritmo de HPO (Bischl et al., 2023).

8.5.1 Genetic Algorithm (GA)

El algoritmo genético, es muy utilizado en la optimización dentro de las metaheurísticas, se basa en la teoría evolutiva, al asignar una mayor probabilidad de supervivencia a aquellos individuos de características y adaptabilidad superiores. Estos individuos, transfieren capacidades a sus descendientes, creando así, nuevas generaciones en cada iteración. Durante este proceso, los individuos de mejor rendimiento prevalecen y aquellos de rendimiento inferior se descartan gradualmente. Últimamente, con el pasar de las iteraciones se encuentra aquel con mejor adaptabilidad u óptimo global. Un aspecto a tener en cuenta es que el propio método incluye hiperparámetros a ajustar lo que aumenta su complejidad y produce una baja velocidad de convergencia comparada con otras ES (L. Yang & Shami, 2020) .

De igual forma, los componentes de un GA son los siguientes: Cromosoma/individuo el cual representa (*Chromosome/individual)*: Representa una HPC.

- Genes: Valores de hiperparámetros pertenecientes a cada cromosoma o individuo (representados por dígitos binarios).
- Población (*population*): Todas las configuraciones de hiperparámetros actualmente registradas.
- Función de aptitud (*fitness function*): Métrica de evaluación de las configuraciones de hiperparámetros.

Las operaciones necesarias en GA posterior a su inicialización son:

- Selección (*selection*): Consiste en elegir los mejores individuos a partir de las evaluaciones de la función aptitud.

- Cruce (*crossover*): Consiste en el intercambio de genes entre cromosomas.
- Mutación (*mutation*): Consiste en alterar aleatoriamente uno o más genes de un individuo.

8.5.1.1 Metodología

El procedimiento en general se muestra a continuación:

1. Inicializar la población (espacio de búsqueda), los individuos(hiperparámetros) y genes (valores de hiperparámetros). Usualmente la inicialización se da de manera aleatoria.
2. Evaluar mediante la función de aptitud el rendimiento de cada cromosoma en la generación.
3. Realizar la selección, cruce y mutación en los cromosomas, creando una nueva generación con las HPCs que se evaluarán en la próxima iteración.
4. Iterar desde el paso dos, hasta encontrar la configuración óptima sujeta a ciertos criterios de terminación.

8.5.1.2 Propiedades

Las principales características se presentan a continuación:

- Paralelización: difícil de ejecutar debido a que es secuencial.
- Comportamiento: Las HPCs de buen rendimiento en cada generación sobreviven pasando a la siguiente de manera iterada, por tanto, las evaluaciones en cada iteración son comparadas.
- Complejidad computacional: Complejidad de orden cuadrático $O(n^2)$.
- Multifidelidad: Reduce el número de evaluaciones y costos de tiempo, al explorar el espacio de búsqueda mediante la evaluación de los mejores HPCs en cada generación.
- Espacio de búsqueda: Trabaja con grandes espacios de búsqueda y jerárquicos.

- Hiperparámetros: Eficiente con todos los tipos de hiperparámetros continuos, enteros, condicionales y categóricos.
- Algoritmos de *Machine Learning*: Aplicable de manera óptima a máquinas de Machine Learning con gran número, tipo y valores de hiperparámetros.
- Otros. El método de inicialización puede aumentar la eficiencia de GA, sin embargo, no es estrictamente necesario, ya que, el método por sí solo converge a la configuración óptima. Por tanto, es ideal cuando no se tiene experiencia en la exploración del espacio de búsqueda.

8.5.2 Enjambre de partículas (Particle Swarm Optimization-PSO)

El enjambre de partículas es otro algoritmo evolutivo utilizado en la HPO. Al igual que el algoritmo genético, se fundamenta en poblaciones, concretamente en comportamientos de un individuo de manera particular o conjunta. En PSO, el espacio de búsqueda es recorrido de manera semi-aleatoria por un enjambre, este es formado por un grupo de partículas que cooperan e intercambian cierta información para encontrar la HPC óptima. Cada partícula *(Si)* es representada por un vector con la posición actual (*xi*), la velocidad(*vi*) y la mejor posición observada(*pi*), como se demuestra en la Ecuación 6(L. Yang & Shami, 2020).

$$Si = \langle \overrightarrow{xi}, \overrightarrow{vi}, \overrightarrow{pi} \rangle \quad (6)$$

Cabe destacar, que el proceso de intercambio de información difiere del algoritmo genético. En GA todos los individuos comparten información entre sí, y se mueven de manera uniforme hacia la región encontrada, en cambio en PSO la información que se transmite es solo del mejor individuo local y global, lo que ocasiona que el proceso de búsqueda siga la dirección del óptimo actual global.

8.5.2.1 Metodología

PSO se define por el siguiente proceso según Muñoz Cañón & Romero Triana (2021):

1. Definir *n* partículas en un enjambre *S* (Tamaño del enjambre).

$$S = (S_1, S_2, \dots, S_n)$$

2. Inicializar las partículas en el enjambre, estableciendo la posición inicial y velocidad.

3. Evaluar cada partícula registrando el *Pbest* (Mejor solución de cada partícula) y *Gbest* (Mejor solución global).

4. Modificar la velocidad y posición en función en función de dos factores, la posición anterior (*Pbest*) y el óptimo global (*Gbest*) encontrado por el enjambre.

1. Iterar hasta alcanzar la convergencia o criterios de terminación.

8.5.2.2 Propiedades

Los factores para tener en cuenta al aplicar PSO se discuten a continuación:

- Paralelización: Paralelización disponible.
- Comportamiento: PSO tiene mayor velocidad y convergencia que GA. En PSO, cada partícula detecta y actualiza el óptimo mediante la comunicación de este a las demás.
- Complejidad computacional: Complejidad cuasi-lineal $O(nlogn)$.
- Multifidelidad: Reduce el número de evaluaciones, al tener en cuenta el óptimo local y global de cada partícula para moverse a la siguiente HPC y evaluarla.
- Espacio de búsqueda: Trabaja con grandes espacios de búsqueda y jerárquicos.
- Hiperparámetros: Eficiente con todos los tipos de HPs continuos, enteros, condicionales y categóricos.

- Algoritmos de *Machine Learning:* Aplicable de manera óptima a máquinas de *Machine Learning* con gran número, tipo y valores de hiperparámetros.
- Otros: El método de inicialización es fundamental en PSO. Adicionalmente, la complejidad del PSO es menor que el algoritmo genético al no existir cruce o mutación.

Dado el detalle de las diferentes técnicas de optimización de hiperparámetros, la Tabla 5, muestra la clasificación de los métodos aplicables para cada tipo de algoritmo de *Machine Learning*. Para una mejor visualización dirigirse al Apéndice B.

Tabla 5

Aplicación de técnicas de HPO en máquinas de ML

TÉCNICAS DE OPTIMIZACIÓN DE HIPERPARÁMETROS (HPO)														
TIPOS DE APRENDIZAJE AUTOMÁTICO	ALGORITMOS DE APRENDIZAJE AUTOMÁTICO	Sin modelo *(Model-free)*				Bayesian optimization(BO)			Multifidelidad (Multi-fidelity optimization)			Metaheuristic	
		Babysitting	Búsqueda cuadrícula (GS)	Búsqueda aleatoria (RS)	Optimización basada en gradiente	Proceso gaussiano (GP)	Configuración de algoritmos basada en modelos secuenciales (SMAC)	Estimador de Parzen estructurado en árbol (TPE)	Reducción sucesiva a la mitad (Successive halving)	Hyperband	Optimización bayesiana (BOHB)	Algoritmo genético (GA)	Optimización por enjambre de partículas (PSO)
Aprendizaje supervisado	Modelos lineales *(Linear models)*					✓	✓						
	K-vecino mas cercano *(K-Nearest Neighbor, KNN)*					✓	✓	✓		✓			
	Máquinas de vectores de soporte *(Support Vector Machines, SVM)*					✓	✓	✓			✓		
	Naïve Bayes *(NB)*					✓							
	Modelos basados en árboles *(Tree-based models)*						✓	✓			✓	✓	✓
	Aprendizaje conjunto *(Ensemble learning)*		✓			✓	✓	✓					
	Aprendizaje profundo *(Deep learning, DL)*										✓		✓
Aprendizaje no supervisado	*Clustering*					✓	✓	✓		✓			
	Algoritmos de reducción de dimensionalidad *(Dimensionality reduction algorithms)*					✓	✓	✓		✓			

Nota. Adaptado de (Yang & Shami, 2020)

Sin embargo, debido a desventajas de determinados algoritmos o variantes de optimización, en donde se presenta que, los algoritmos suelen hacer pruebas solo en conjuntos limitados de instancias, explorando pocas alternativas de diseño y configuración de parámetros. Surgen otro tipo de variantes que implementan técnicas de diseño experimental más automatizadas, con el fin de encontrar una configuración aceptable de hiperparámetros.

8.6 Técnicas de HPO adicionales

A raíz del análisis de las diferentes técnicas de HPO, algunos autores han propuesto algunas variantes de los métodos anteriormente caracterizados. Estos, se discuten brevemente a continuación:

8.6.1 Técnicas de muestreo

Inicialmente como métodos de muestreo adicionales a *Random Search y Grid Search* se encuentran *Latin Hypercube Sampling, Sobol sequences* y *naive i.i.d. sampling*. Estos han demostrado amplia aplicabilidad y rendimiento en HPO dada su flexibilidad. De igual forma, en BO las redes neuronales (NN) han demostrado ser un adecuado modelo sustituto, al permitir tres aspectos cruciales: 1. Manejar entradas discretas mediante codificación. 2. Implementar gradientes para la optimización de la función de adquisición y 3. obtener límites de incertidumbre en las predicciones.

8.6.2 Multifidelidad

De igual forma, en multifidelidad algunas variantes similares a BOHB como la búsqueda de entropía *y FABOLAS* se consideran buenas opciones para abordar problemas de BO. La búsqueda de entropía evalúa las configuraciones de baja fidelidad solo cuando aporten gran cantidad de información en comparación con los recursos asignados. *FABOLAS* decide dinámicamente el tamaño del dataset de entrenamiento para cada evaluación equilibrando el coste computacional y la ganancia de información relacionada a la configuración óptima. Ambos métodos introducen mejoras adicionales a los métodos multifidelidad incluyendo aspectos nuevos en la optimización de hiperparámetros.

8.6.3 Carreras iteradas (Iterated racing)

Dado que los algoritmos modernos o evolutivos requieren configurar gran cantidad de parámetros, las carreras iteradas surgen como variante para seleccionar una configuración entre un número de candidatos, mediante pruebas estadísticas secuenciales. Estas clasificaciones pueden seleccionarse mediante técnicas de diseño de experimentos (DOE) o al azar. En el caso de carreras iteradas, un modelo de muestreo se refina iterativamente de acuerdo con el resultado de carreras anteriores.

La iteración de carreras es un método para la configuración automática que consta de tres pasos: (1) muestrear nuevas configuraciones de acuerdo con una distribución particular, (2) seleccionar las mejores configuraciones de las recién muestreadas mediante carreras, y (3) actualizar el muestreo, con distribución para sesgar el muestreo hacia las mejores configuraciones. Estos tres pasos se repiten hasta que se cumple un criterio de terminación. En cada paso de racing, las configuraciones candidatas se evalúan en una sola instancia. Después, se descartan aquellas configuraciones candidatas que presentan un rendimiento estadísticamente inferior a al menos una de las otras, y la carrera continúa con las configuraciones supervivientes restantes.

Hay que tener en cuenta los siguientes aspectos en la ejecución de carreras iteradas:

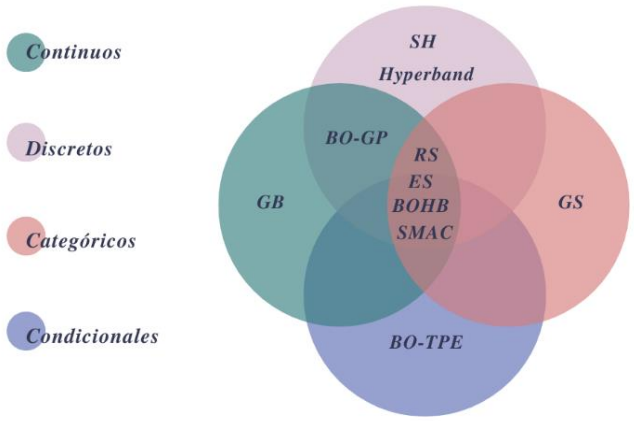
- Al considerar la solución de un problema mediante este algoritmo parametrizado, el conjunto de posibles instancias del problema puede verse como una variable aleatoria.
- Cada parámetro configurable tiene asociada una distribución de muestreo que es independiente de las distribuciones de muestreo de los otros parámetros, aparte de las restricciones y condiciones entre parámetros.

- En caso de detectar una convergencia temprana, se puede aplicar una técnica de reinicio suave, la cual implica un reinicio parcial de la distribución de muestreo. Esta reinicialización solo afecta a las configuraciones destacadas que se utilizaron para generar las configuraciones candidatas con distancia cero.
- Si el parámetro numérico es de tipo entero, se redondea el valor muestreado al entero más cercano. El muestreo se ajusta para evitar el sesgo contra los extremos introducido por el redondeo después del muestreo de una distribución truncada.

En la Figura 11 se muestra la clasificación de las distintas técnicas vistas, dividiéndolas según su capacidad de procesamiento de tipos de hiperparámetros.

Figura 11

Clasificación de técnicas según tipología de HP



Nota. Adaptado de (Yang & Shami, 2020)

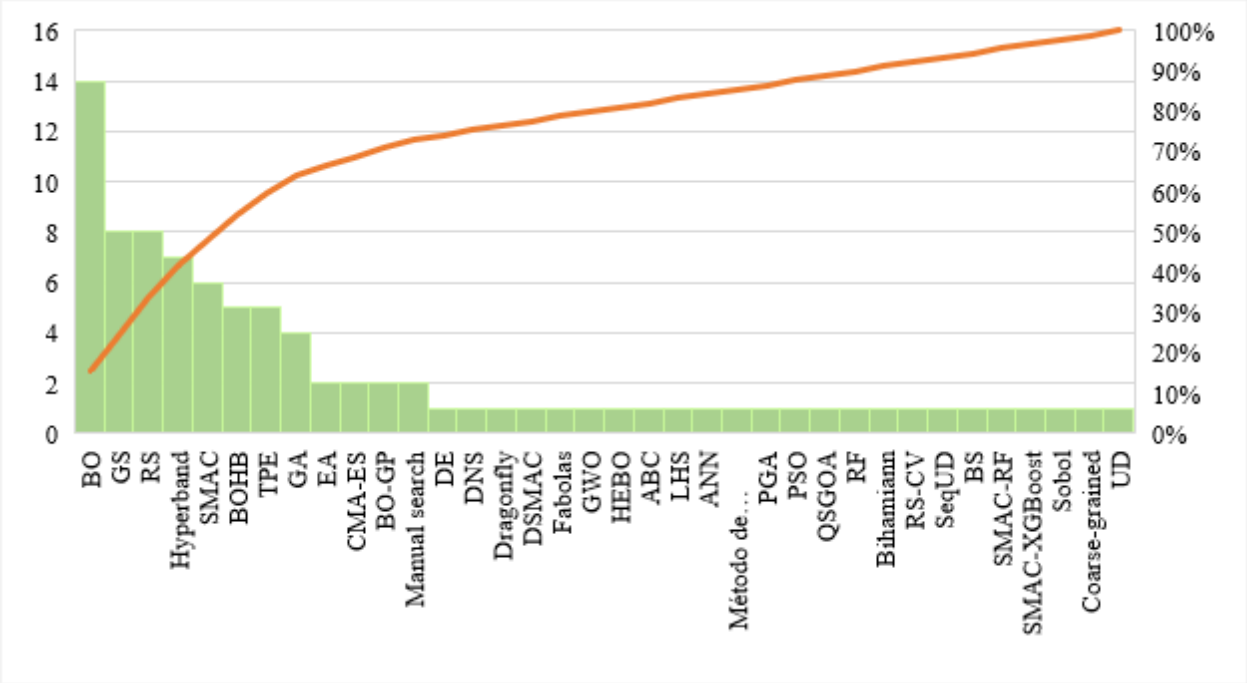
9. Selección de los métodos de optimización

Ahora bien, a partir de las distintas investigaciones analizadas, se busca encontrar el método de ajuste de hiperparámetros más adecuado en el rendimiento de un algoritmo de *Machine Learning*. Este método es elegido basado en las frecuencias de aplicabilidad en otras investigaciones, así como a sus propiedades, flexibilidad en el manejo de espacios de búsqueda y aspectos de desarrollo que lo hacen elegible entre otras técnicas de ML.

Lo anterior, se realizó mediante el análisis de elementos como el conjunto de datos, los métodos y optimización utilizados, las métricas, el objetivo de aprendizaje automático y el lenguaje de programación identificados en cada desarrollo investigativo. Por tal motivo, se ha determinado la relevancia de las técnicas de optimización de hiperparámetros (HPO) y los algoritmos de aprendizaje automático. Las siguientes gráficas muestran la frecuencia de uso de estas técnicas y algoritmos en los estudios analizados.

Figura 12

Frecuencia por técnica de HPO



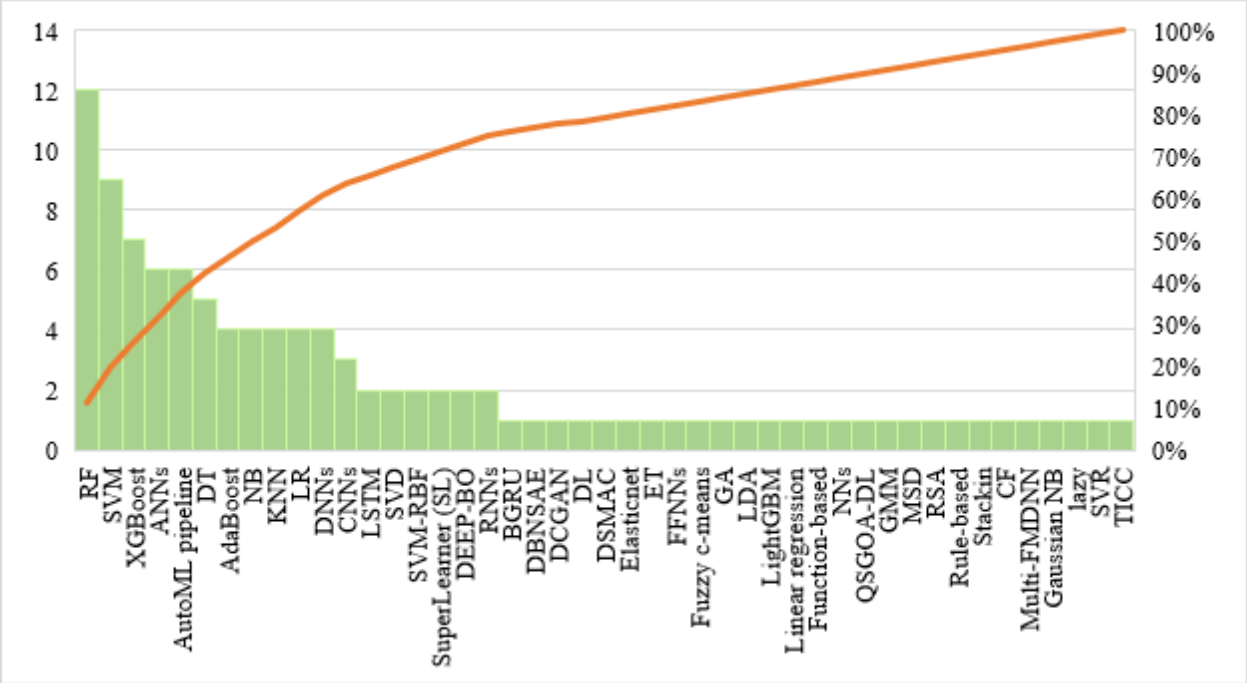
En ese orden de ideas, se evidencia las distintas formas de aplicación de las técnicas de HPO para algoritmos de ML. En la Figura 12, se puede señalar que las técnicas con más tendencia han sido Optimización Bayesiana (BO), *Grid Search* (GS) y *Random Search* (RS), esto dado principalmente por su sencillez y capacidad de superar limitaciones de recursos y tiempo.

De la misma forma en la Figura 13, los algoritmos de ML con más usabilidad son *Random Forest*, Máquinas de soporte vectorial (SVM) y *XGBoost*, debido a su capacidad de manejar

diversos tipos de variables. Estos resultados corroboran la información previamente categorizada, afirmando la funcionalidad y desempeño que tienen estos métodos para abordar diversos problemas. De igual forma, es de destacar que los algoritmos de optimización de hiperparámetros han tenido importancia para aportar soluciones a distintos sectores económicos como el de construcción, salud, financiero, educativo, energía, hidrocarburos, entre otros. Su versatilidad los convierte en una herramienta valiosa para la resolución de problemas en diferentes ámbitos.

Figura 13

Frecuencia por algoritmo de ML



A raíz de lo anterior, se encuentra que, aunque de 35 artículos analizados, 3 de ellos no incluían información acerca del proceso de HPO, estos analizaban aspectos relevantes y, por tanto, se toman a consideración en algunos casos. Igualmente, según lo revisado se destacan algunos aspectos cruciales y definitivos para tener en cuenta:

- Se ha optado por utilizar *Python* como lenguaje de programación. Debido a que es ampliamente implementado en el manejo de los algoritmos de *Machine Learning* y optimización,

siendo empleado en 28 de los 35 artículos usados. Además, cuenta con una amplia gama de entorno de trabajo y bibliotecas que permiten la correcta funcionalidad de los algoritmos de clasificación y regresión en el aprendizaje automático.

- Teniendo en cuenta que los algoritmos de *Machine Learning* se basan en sistemas de clasificación y regresión, se evidencia que, de las 35 investigaciones revisadas, 8 implementaron algoritmos de clasificación, 15 de regresión y 12 ambos sistemas simultáneamente. Basándonos en este análisis, para este estudio se optó por manejar algoritmos de clasificación, ya que, al iniciar con una clasificación, esta permite obtener una mejor precisión en las características de los datos, debido al nivel de sensibilidad que presenta en la etapa inicial de clasificación.
- Las métricas con las cuales se evalúan el rendimiento de las técnicas de HPO son variadas dado el objetivo. No obstante, algunas de ellas son utilizadas frecuentemente, como el caso de Precisión(*precision*), sensibilidad (*Recall*), Exactitud(*Accuracy*) y Puntuación F1(*F1 Score*). Por tanto, para este estudio son tomadas en consideración como métricas complementarias para evaluar detalladamente el proceso de ajuste y rendimiento dentro del modelo.
- Por último, se tomaron diferentes archivos de OpenML y de Kaggle para el análisis de datos supervisados, esta es una plataforma que maneja una amplia gama de repositorios aplicados en AutoML. Por tanto, se toma como referencia para la futura evaluación de *benchmarking*.

Adicionalmente, aspectos cruciales correspondientes a la siguiente fase de análisis se da en la elección de la técnica de ML y de HPO. Siendo así, a continuación, se presenta el análisis correspondiente:

En primer lugar, respecto a la máquina de ML, se optó por elegir *Random Forest,* dada su amplia aplicabilidad demostrada. Esto dado a su flexibilidad que le permite: 1). manejar un gran

número de variables e identificar las más significativas, 2). Mejorar como algoritmo los modelos de árboles de decisión al incluir un factor aleatorio, obteniendo como resultado un modelo más refinado y 3). Participar como optimizador en diversos procesos de HPO.

En segundo lugar, la técnica de optimización escogida para el presente proyecto como resultado del análisis de frecuencia y caracterización es Enjambre de partículas (*Particle Swarm Optimization- PSO).* Esta selección se justifica principalmente por la adecuación que tiene a la máquina de RF, que maneja diversos hiperparámetros y un espacio de búsqueda con alta dimensión. Si bien el análisis de frecuencia arroja que métodos de optimización bayesiana (BO) y multifidelidad son los más aplicados; sin embargo, estos basan su aplicabilidad en restricciones de tiempo y recursos, lo cual no es el objetivo primordial del presente estudio. Por tanto, se elige PSO para llevar a cabo una exploración de manera más detallada de los espacios de búsqueda y encontrar los valores óptimos de un conjunto de configuraciones en dicho espacio.

10. Definición de los hiperparámetros a optimizar

La selección de los hiperparámetros es parte fundamental en la construcción de modelos de ML. La selección adecuada de hiperparámetros permitirán que una máquina de *Random Forest* presente una mayor precisión, óptima generalización y eficiencia en cuanto a tiempo de entrenamiento y recursos computacionales; minimizando los riesgos de un posible sobreajuste. Siendo así, a continuación, se describen los hiperparámetros comúnmente ajustados en un algoritmo como *Random Forest*:

- n estimators: Es de los hiperparámetros importantes en el algoritmo *Random Forest*, representa el número de árboles que se construyen o se incluyen en el modelo. Aunque generalmente, cuanto mayor sea el número de árboles se considera que mejor será la precisión del modelo, esto no siempre sucede, ya que puede haber un punto de disminución en la mejora de la

precisión después de cierto número de árboles. Por tanto, algunas características que lo hacen importante como hiperparámetro son, el impacto en la precisión, impacto en el tiempo de entrenamiento, regularización, impacto en la interpretación, impacto en la variabilidad, impacto en la complejidad del modelo (Alai et al., 2021; Fernández et al., 2018).

Es un hiperparámetro importante que se debe ajustar cuidadosamente durante el proceso de modelado, ya que, al ajustar el número de árboles se puede encontrar el equilibrio entre la precisión y el tiempo de entrenamiento, y a su vez evitar el sobreajuste en el modelo.

- *Maximum tree depth* (*Max_depth):* se refiere a la profundidad máxima permitida para cada árbol en el bosque aleatorio, controlando la complejidad del árbol. La profundidad del árbol se refiere a la cantidad de ramificaciones que se pueden hacer en un árbol de decisión antes de llegar a las hojas. Es un límite para detener la división adicional de nodos (sobreajuste del modelo) cuando se alcanza la profundidad del árbol especificada durante la construcción del árbol de decisión inicial. La profundidad máxima de un árbol binario es el número de nodos desde la raíz hasta el nodo hoja más alejado, siendo la altura de un árbol binario (Alai et al., 2021; Fernández et al., 2018).

Algunas características importantes de *Max depth* en *Random Forest* son: Control de la complejidad del modelo, reducción de la varianza e impacto en el rendimiento.

- *Measuring function* (Criterion): Utilizada para medir la cantidad y calidad de las divisiones en cada nodo del árbol de decisión. Las funciones de medición más comunes y utilizadas son el índice de la impureza de Gini y la entropía, las cuales miden la homogeneidad de la clase de los datos en un nodo. Los criterios admitidos son MSE para el error cuadrático medio, que es igual a la reducción de la varianza como criterio de selección de características, y MAE para el error absoluto medio (Alai et al., 2021; Fernández et al., 2018).

Algunas características importantes de la función de medición en *Random Forest* son: Selección de la función de medición, impacto en el tiempo de entrenamiento e interpretación de la salida.

- *The maximum number of features* ('max_features'): Se refiere al número máximo de características para generar la mejor división en cada nodo del árbol de decisión. Algunas características que hacen de este hiperparámetro importante son la reducción de la varianza, el control de la complejidad del modelo, el impacto en el rendimiento y la elección de las características (Alai et al., 2021; Fernández et al., 2018).

El ajuste del número máximo de características dependerá de la complejidad del problema que se está abordando, junto con la cantidad y calidad de los datos de entrenamiento disponibles.

- *The minimum number of data points* (min samples split y min samples leaf): Representa el número mínimo de muestras requeridas para decidir si un nodo se dividirá en una o más ramas adicionales. Este hiperparámetro establece la cantidad mínima de puntos de datos necesarios en un nodo hoja del árbol, su importancia radica en la capacidad para evitar el sobreajuste (*overfitting*) del modelo, que es un problema común en el aprendizaje automático. El número mínimo de muestras requeridas usa funciones como, *min samples split* la cual especifica la cantidad mínima de muestras necesarias para dividir un nodo interno, mientras que *min samples leaf* especifica la cantidad mínima de muestras necesarias para estar en un nodo hoja. Estos hiperparámetros son útiles para controlar la complejidad del modelo y evitar que se ajuste demasiado a los datos de entrenamiento, lo que puede llevar a una reducción en la capacidad de generalización del modelo (Alai et al., 2021; Fernández et al., 2018). Este hiperparámetro tiene características como:

1. Si se establece un número mínimo de puntos de datos demasiado bajo, el modelo puede dividir el árbol demasiado y ajustarse en exceso a los datos de entrenamiento, ya que, este hiperparámetro afecta directamente la profundidad del árbol de decisión utilizado en el modelo.

2. Define el número mínimo de puntos de datos necesarios en un nodo de hoja del árbol para que se siga dividiendo en ramas adicionales.

3. Ajustar adecuadamente este hiperparámetro es importante para lograr un equilibrio balanceado entre la capacidad del modelo para ajustarse a los datos de entrenamiento y su capacidad para generalizar y predecir nuevos datos. Si se establece un número mínimo de puntos de datos demasiado alto, puede darse la posibilidad que el modelo no interprete adecuadamente la complejidad del conjunto de datos.

Según lo anterior la Tabla 6 muestra la influencia que tienen los diferentes hiperparámetros al modelo de *Random Forest*.

Tabla 6

Influencia hiperparámetros

Hiperparámetros	Precisión	Rendimiento	Minimización de sobreajuste	Óptimo tiempo de ejecución	Controlar complejidad
n estimators	✓			✓	✓
Maximum tree depth		✓			
Measuring function	✓		✓		
Max features		✓	✓		✓
The minimum number of data points		✓	✓		
Split selection method	✓				
min weight fraction leaf	✓		✓		✓
Max leaf nodes	✓	✓	✓		

11. Construcción de Espacios de Búsqueda

Una vez revisados y analizados los hiperparámetros pertenecientes a la máquina de RF, es necesario establecer los rangos a optimizar. Para ello, basándose en investigaciones enfocadas en optimización de hiperparámetros realizada por Motz et al. (2022), Tayebi & El Kafhali (2022), L. Yang & Shami (2020), Zheng et al. (2021) y Zhu et al. (2022) se construye la Tabla 7. En la cual, se muestran distintos rangos de valores de HPs comúnmente optimizados en RF. Cabe aclarar que, los rangos mostrados generalizan las diversas investigaciones de optimización de *Random Forest*, por lo tanto, fueron escogidos como principales exponentes.

Tabla 7

Influencia hiperparámetros

Hiperparámetro	Tipo	Espacios de búsqueda				
		No.1	No.2	Otros espacios		
n_estimators	Entero	[10,100]	[50,200]	[100,300]	[10,510]	[10,200]
max_depth	Entero	[5, 50]	[350,400, 450]	[10,100]	[1,300]	
criterion	Categórico	[Entropy, Gini]		[Entropy, Gini]		
max_features	Entero /Real	[1,64]	[12,16]	-	[8,128]	[0.1, 1]
min_samples_split	Entero	[2,11]	[2,3]	[10,100]	[2,18]	-
min_samples_leaf	Entero	[1,11]	[1,5]	[1,10]	[1,21]	[1, 20]

Nota. Adaptado de Motz et al. (2022), Tayebi & El Kafhali (2022), L. Yang & Shami (2020), Zheng et al. (2021) y Zhu et al. (2022)

11.1 Descripción de las características del dataset

Una vez definido el espacio de búsqueda, se deben determinar los conjuntos de datos que se utilizarán en el proceso de *benchmarking*. Para ello, se toman como referencia el Bank

Marketing Dataset y Heart Failure Dataset, los cuales provienen del repositorio UCI de *Machine Learning* y la plataforma Kaggle respectivamente. Estos son elegidos debido a las características convenientes para realizar el proceso de HPO, así como también por su relevancia en el sector financiero y salud. Seguidamente se describen más a detalle:

11.1.1 Heart Failure Prediction Dataset

Este dataset se asocia al sector médico, concretamente en la insuficiencia cardíaca. Consta de 11 características y 918 instancias originalmente fragmentadas de acuerdo con el lugar de muestreo, específicamente Cleveland (303 observaciones), Hungría (294 observaciones), Suiza (123 observaciones), Long Beach VA (200 observaciones) y Stalog (270 observaciones). Debido a que de las 1190 observaciones totales 272 son duplicaciones, el conjunto de datos fue recortado hasta 918.

El objetivo de clasificación es identificar la presencia de cardiopatía en el paciente. Esto, mediante la predicción de valores según el grado, específicamente de cero (sin presencia) a 4 (presencia de alto riesgo). Se debe agregar que en las diversas investigaciones se reemplazó y simplificó el objetivo a clasificación binaria, enfocándose en presencia (valores 1,2,3,4) y no presencia (valores 0) de suficiencia cardíaca.

11.1.2 Bank Marketing Dataset

El dataset cuenta con 41.188 instancias y 20 atributos. Este se relaciona con campañas de marketing de una entidad bancaria de Portugal. El objetivo de la clasificación es predecir si el cliente suscribirá (sí/no) a un depósito bancario a plazo. También, es importante considerar que las observaciones se basan en marketing directo, esto mediante varias llamadas telefónicas según los requerimientos.

11.2 Definición de las variables de estudio respecto al dataset

Una vez finalizada la selección de los dataset, es importante conocer sus propiedades, a fin de identificar aspectos relevantes para el procesamiento y análisis. En concordancia, la Tabla 8 resume la información general de los conjuntos de datos. En ambos casos, el problema a abordar es multivariado; no obstante, al examinar las categorías de la etiqueta objetivo, se evidencia que para el conjunto de datos Bank Marketing existe un desequilibrio de clases (36548 instancias negativas y 4640 instancias positivas), mientras que para el conjunto de datos Heart Failure este desequilibrio es casi nulo (508 instancias positivas y 410 instancias negativas).

Tabla 8

Características de los dataset en estudio

Dataset	Características	Variables tipo	Número
Heart Failure	Multivariado -Equilibrado	Entera	5
		Categórica	6
		Real	1
Bank Marketing	Multivariado - Desequilibrado	Entera	5
		Categórica	10
		Real	5

Ahora bien, además de aspectos generales se debe de analizar los valores específicos de cada variable predictora con el propósito de encontrar su método de procesamiento. Dicho lo anterior, para el procesamiento de las variables categóricas se seleccionó el "*one hot encoding"* basándose en Z. Yang & Zhang (2021). Por su parte, las variables numéricas no serán normalizadas, dado que el *Random Forest* no es sensible a escalas (Zheng et al., 2021). Además, en cuanto a los valores perdidos se tratarán por medio de imputación, utilizando la mediana y moda según la tipología del valor nulo (Z. Yang & Zhang, 2021).

11.3 Elección del rango de hiperparámetros

Una vez estudiada la complejidad de los dataset en términos de su extensión y características, se debe definir el rango de los hiperparámetros que se optimizará. Debido a que se quiere abarcar un gran espacio de búsqueda se seleccionan los seis hiperparámetros a optimizar: *n estimators, máximum tree depth, criterion, max features, min simples split y min simples leaf*. De igual modo, se seleccionaron los rangos de valores o espacios de búsqueda No.1 y No.2, los cuales son mostrados en la Tabla 7. Se aclara que se seleccionaron dos rangos con el fin de realizar una comparación de rendimiento y de abarcar un mayor espacio de búsqueda (L. Yang & Shami, 2020; Zheng et al., 2021).

12. Aplicación del algoritmo de aprendizaje automático

Para dar inicio al procesamiento de los conjuntos de datos para la validación del algoritmo a analizar en el lenguaje de programación *Python*, se opta por inicializar la máquina de *Random Forest* sin aplicar las técnicas de optimización, con el fin de comparar el rendimiento del modelo. El procesamiento de la máquina se ejecuta con dos conjuntos de datos disponibles para realizar pruebas de rendimiento: un dataset con datos equilibrados (Heart Failure) y otro con datos desequilibrados (Bank Marketing). Al ejecutar el código en RF, los resultados ilustrados en la Tabla 9 demuestran que el modelo se ajusta perfectamente a los datos de entrenamiento, sin embargo, en datos no vistos la capacidad predictiva disminuye, por tanto, se genera un sobreajuste en la totalidad de las métricas, el cual afecta negativamente el modelo, al no proporcionar un valor de estimación de rendimiento confiable en datos no vistos.

Tabla 9

Resultados modelo básico de Random Forest

Métricas	Bank Marketing		Heart Failure	
	Train	*Test*	*Train*	*Test*
Recall	1	0.56630	1	0.92125
Accuracy	1	0.91045	1	0.86086
Precision	1	0.59065	1	0.84172
F1- score	1	0.57822	1	0.87969

En consecuencia, se aplica la técnica de optimización de hiperparámetros PSO para reducir el sobreajuste y aumentar el rendimiento en el problema de clasificación. El objetivo es crear un espacio de búsqueda adecuado para seleccionar los mejores valores de los hiperparámetros de RF, logrando como resultado un modelo óptimo, aceptable y capaz de soportar el análisis de sistemas robustos.

13. Algoritmo de Machine Learning

El método propuesto consiste en aplicar el algoritmo de optimización *Particle Swarm Optimization* (PSO) a *Random Forest* (RF), con el fin de encontrar la configuración óptima de sus hiperparámetros y reducir el sobreajuste presentado anteriormente. En este sentido, a continuación, se discuten factores importantes en la aplicación de PSO y RF.

13.1 Inicialización

En primer lugar, un aspecto crucial en la aplicación de PSO es la inicialización de sus parámetros, dado que afectan su velocidad de convergencia y rendimiento general (Chand Bansal et al., 2019;L. Yang & Shami, 2020) . Por ende, los valores adecuados de los parámetros de PSO se establecen de la siguiente manera:

13.1.1 Tamaño de enjambre

Su valor no se ve sujeto a una regla específica, dado que comúnmente es estimado como resultado de pruebas y errores (Muñoz Cañón & Romero Triana, 2021). En concordancia, Tambouratzis (2022) sugiere que los tamaños de enjambre de menos de 20 partículas tienen una capacidad reducida para encontrar soluciones óptimas en espacios de optimización; sin embargo, aumentar en un número considerable las partículas podría llevar a un rendimiento subóptimo. En vista de lo mencionado, el proceso de optimización actual se utilizó un tamaño de enjambre de 50 partículas, ya que, se considera que este valor puede proporcionar mejores soluciones con un rendimiento adecuado (Chand Bansal et al., 2019).

13.1.2 Posiciones iniciales partículas

La inicialización de las partículas comúnmente varía en función de la información disponible del espacio de búsqueda. Si se dispone de una posible región del espacio que contenga el óptimo global, las partículas deben distribuirse a lo largo de la región. En cambio, cuando el espacio de búsqueda es inexplorado en su totalidad, las regiones deben tomarse equivalentemente (Parsopoulos & Vrahatis, 2010). Por esta razón, el presente estudio se considera equivalente, al considerar que no se dispone de información adicional sobre el óptimo global, ya que solo se cuenta con los rangos de HPs especificados.

Por otro lado, según el grado de uniformidad deseado, la generación de las posiciones de las partículas se puede realizar mediante la inicialización uniforme pseudo-aleatoria o cuasi-aleatoria (Parsopoulos & Vrahatis, 2010). Esta última, garantiza una mejor uniformidad o menor discrepancia. En función de lo anterior se implementó las secuencias de Sobol como inicializador de posición (Z. Yang & Zhang, 2021).

13.1.3 Velocidades iniciales partículas

El valor de la velocidad suele inicializarse en cero o de manera aleatoria. En el primer caso, las partículas obtienen aceleración basándose en su distancia a las mejores posiciones; sin embargo, esto puede ocasionar que el enjambre quede atrapado en óptimos locales, debido a la propiedad que asigna velocidades menores alrededor del óptimo global sospechado (Parsopoulos & Vrahatis, 2010; Simon, 2013). Por consiguiente, se decidió que la velocidad inicial sea establecida de manera aleatoria.

13.1.4 Iteraciones o generaciones

En PSO es necesario establecer un criterio para finalizar el algoritmo. Frecuentemente el número de generaciones es usado como principal limitador. De acuerdo con esto, Simon (2013) sugiere que para problemas abordados con enjambres de 50 partículas es recomendable usar 1000 evaluaciones de la función, lo que corresponden a 20 generaciones de la población en el desarrollo actual, pero dada la complejidad abordada en este documento se tomarán 10 generaciones de optimización.

13.2 Parámetros de velocidad y posición.

Debido a que las velocidades y posiciones se actualizan conforme a las iteraciones, es necesario definir algunas constantes que rigen dicho cambio (Parsopoulos & Vrahatis, 2010; Simon, 2013). En las Ecuaciones (7) y (8) podemos observar respectivamente, cómo se realizan los cambios de la velocidad (*V*) y posición (*X*) de una partícula *j* en la generación *i*.

$$V_j(i) = V_j(i-1) + C_1 r_1 * \left[Pbest - x_j(i-1)\right] + C_2 r_2 * [Gbest - x_j(i-1)] \qquad (7)$$

En donde: j es una partícula del enjambre (j =1, 2, 3, …, n), i es la iteración o generación actual (i = 1, 2, 3, …, m), C_1 y C_2 son los ritmos de aprendizaje cognitivo y social de las partículas, $r_1 y\ r_2$ son valores de probabilidad aleatorios.

$$X_j(i) = X_j(i-1) + V_j(i) \tag{8}$$

En donde, *j* es una partícula del enjambre (j = 1, 2, 3, …, n), *i* es la iteración o generación actual (i = 1, 2, 3, …, m).

Teniendo en cuenta lo anterior, es fácil notar que C_1 y C_2 ,al establecer el tamaño paso influyen significativamente en las nuevas posiciones y velocidades de las partículas en cada generación. Así pues, en el proceso de HPO, se ha definido un valor de 2, para equilibrar la velocidad de convergencia rápida y la importancia del óptimo local y global para cada partícula. (Chand Bansal et al., 2019).

Los valores definidos anteriormente proporcionan una correcta inicialización de PSO. Es importante señalar que actualmente algunas variantes de PSO, como las mencionadas por Tambouratzis (2022) y Parsopoulos & Vrahatis (2010) han mejorado aspectos relacionados con la convergencia (Inercia) y la inicialización (algoritmos previos a las iteraciones). No obstante, el estudio de las variantes queda a consideración de desarrollos futuros.

13.3 Límites de velocidad y posición

Durante el funcionamiento del algoritmo *Particle Swarm Optimization* (PSO) suelen ocurrir dos eventos claves que se deben abordar antes de su ejecución. El primero es denominado explosión del enjambre, que se refiere al aumento incontrolado de la velocidad que ocasiona la divergencia del enjambre en la solución global. El segundo se da como consecuencia del carácter vectorial en las actualizaciones de la velocidad y la posición, pues con el transcurso de las generaciones, algunas partículas pueden abandonar el espacio de búsqueda, y con ello impiden el carácter factible de las soluciones encontradas (Parsopoulos & Vrahatis, 2010; Simon, 2013). Considerando lo anterior, es posible abordar los inconvenientes señalados al limitar tanto la

velocidad máxima alcanzada como los valores de posición de las partículas en cada ejecución del algoritmo de optimización, a continuación, discutiremos las principales formas de llevarlo a cabo.

13.4 Velocidad máxima.

Al limitar la velocidad máxima, las partículas convergen con mayor facilidad hacia el óptimo deseado, dado que su tamaño de paso se vuelve más pequeño. En general, la velocidad máxima se puede limitar por un valor escalar o por un vector de velocidad máxima. En el primer caso, cada componente del vector velocidad estará sujeto a un valor escalar de velocidad predefinido y en caso de no cumplirse se deberá ajustar a ese valor establecido. En el caso vectorial, se debe limitar cada componente de la velocidad a un valor específico, y de manera similar ajustarlo a su límite en caso de sobrepasarlo (Parsopoulos & Vrahatis, 2010). Debido a que al limitar cada componente de velocidad la sensibilidad de las dimensiones es considerada, el caso vectorial mostrado en la Ecuación 9 es usado como limitante de velocidad.

$$V_j(i) = \{ \; Vmax, si \;\; V_j(i) > Vmax \;\; -Vmax, si \;\; V_j(i) > Vmax \;\; donde, Vmax > 0 \qquad (9)$$

Es importante destacar que los valores que sujetan los componentes de la velocidad máxima afectan la exploración y explotación del algoritmo. Valores altos de Vmax permiten cambios más grandes en las posiciones de las partículas Xi, lo que fomenta la exploración de nuevas áreas del espacio de búsqueda. Mientras que, valores bajos de Vmax aumentan la explotación de las regiones conocidas al limitar los cambios en Xi. En concordancia, su valor se calcula basado en la proporción de la longitud de los espacios de búsqueda, Ecuación 10. Los valores límites recomendados para PR por Simon (2013), sugieren que las velocidades de las partículas no deben exceder el 20%, 30% y 100% de la longitud de sus respectivas dimensiones. En consecuencia, en el actual documento la velocidad será limitada con un PR de 30% para asegurar el equilibrio entre explotación y exploración en el algoritmo.

$$Vmax = (b_{d-} a_d) * PR \quad (10)$$

En donde, d es la dimensión d = 1, 2, 3, …, D, b es el límite superior de posición, a es el límite inferior de posición y PR es la proporción asignada.

13.5 Posición factible

Al realizar los cálculos vectoriales de las actualizaciones, es común que las partículas tengan valores que excedan los límites de posición especificados. Generalmente, se soluciona por medio de técnicas que restringen el comportamiento de las partículas, estas son: 1. Sujeción, consiste en fijar las partículas que abandonan el espacio al límite rebasado, 2. Movimiento de rebote, que ocasiona que la partícula cambie de dirección regresando al espacio de búsqueda. 3. Inicialización, que asigna los valores iniciales a las partículas del espacio de manera cuasi – aleatoria o pseudo -aleatoria (Chand Bansal et al., 2019; Parsopoulos & Vrahatis, 2010). En este proceso, es usado como referencia el movimiento de sujeción Ecuación 11, porque este permite un comportamiento libre de las partículas, además da indicios de comportamiento óptimos que en el espacio no se están teniendo en cuenta.

$$X_j(i) = \{ \; Xmax, si \quad X_j(i) > Xmax \quad -Xmax, si \quad X_j(i) > Xmax \quad donde, \quad Xmax \; > 0 \quad (11)$$

Donde, *Xmax* es la posición máxima que puede alcanzar la partícula.

14. Algoritmo de optimización

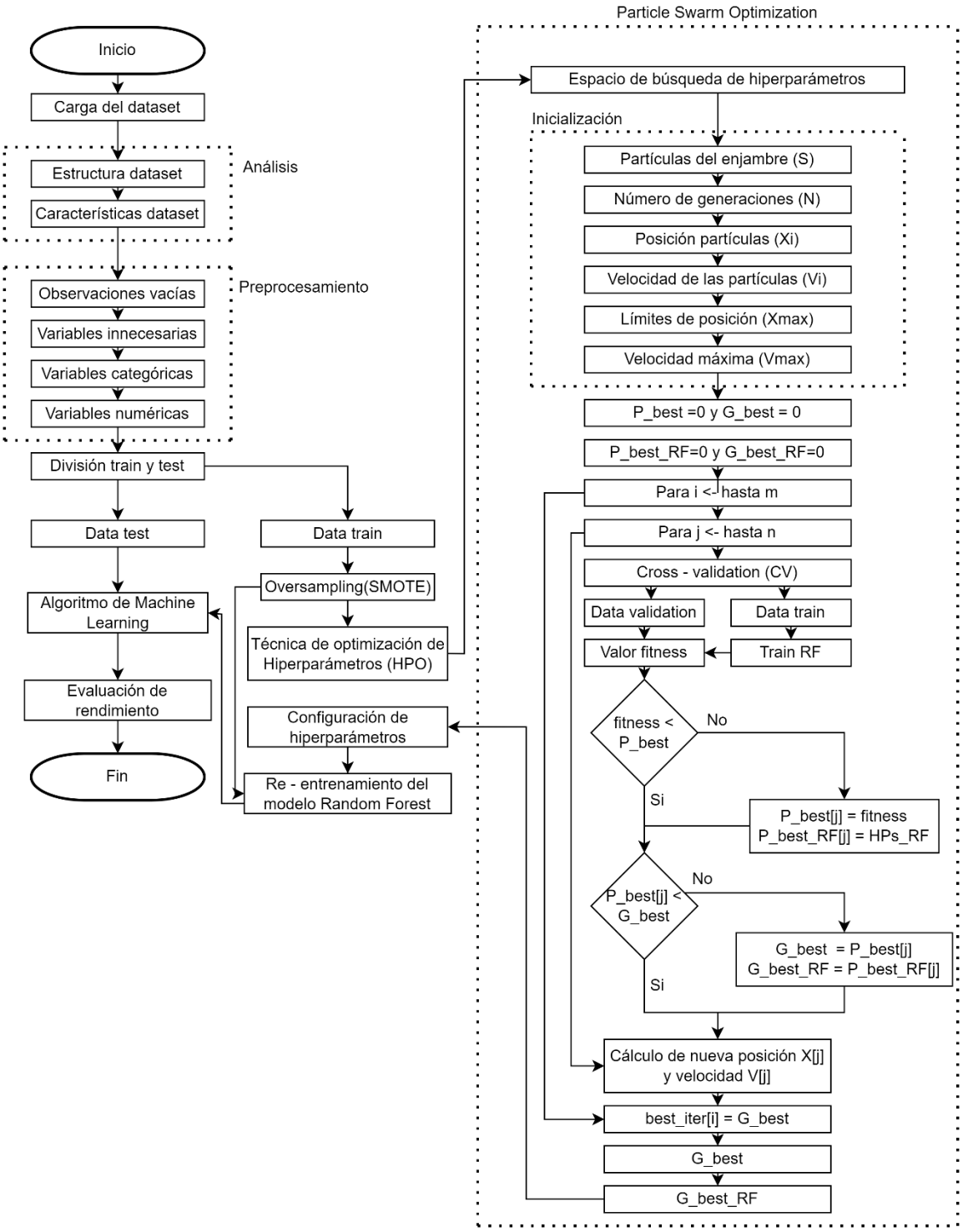
Basado en lo anteriormente descrito, la Figura 14 muestra el algoritmo de optimización aplicado en el presente documento. El algoritmo inicia con la carga y análisis del conjunto de datos, este paso busca encontrar aquellas modificaciones en términos de características y estructuras que son necesarias de realizar en el dataset, con el objetivo de utilizarlos posteriormente en el algoritmo. Luego, el preprocesamiento es el encargado de realizar los cambios identificados

previamente, tales como el reemplazo de observaciones vacías, eliminación de características innecesarias, codificación de variables categóricas y normalización de variables numéricas.

Luego de la etapa de preprocesamiento el conjunto de datos es dividido en entrenamiento-validación y prueba mediante muestreo estratificado en una proporción de 0.75 y 0.25 respectivamente (Owen, 2022). El conjunto de prueba (test) es guardado para la evaluación final, mientras que el conjunto de entrenamiento y validación es sobremuestreado por la técnica SMOTE en conjunto de datos desequilibrados, a fin de equilibrar las instancias positivas mediante datos sintéticos (Fernández et al., 2018). Ahora, el conjunto resultante es ingresado en el algoritmo de optimización PSO + RF, el cual utiliza una validación cruzada de 5 folds, la cual es recomendada para tamaños de datasets medianos (Bischl et al., 2023).

Figura 14

Algoritmo de optimización



El algoritmo *Particle Swarm Optimization* (PSO) combinado con *Random Forest* se aplica como se muestra a continuación:

1. Definir el espacio de búsqueda que se quiere optimizar, específicamente el rango de hiperparámetros, en donde el algoritmo PSO buscará la mejor solución (Tabla 7).

2. Inicializar las partículas del enjambre (S), el número de generaciones (N), la posición de las partículas (Xj), la velocidad de las partículas (Vj) y los demás criterios de inicialización.

3. Establecer los límites de velocidad (Vmax) y posición (Xmax).

4. Definir la función de aptitud, f. (Que en nuestro caso es el resultado arrojado por el *Random Forest* utilizando una validación cruzada de 5 folds).

5. Evaluar las partículas del enjambre con base en la función de fitness, f.

6. Establecer el vector de posición actual de cada partícula como su mejor vector de posición (Primera iteración).

7. Establecer la mejor posición global actual, seleccionando el mejor valor de las partículas del enjambre (Primera iteración).

8. Actualizar los mejores vectores de posición de cada partícula, empezando por comparar la puntuación de aptitud actual de cada partícula con la mejor puntuación de aptitud en las generaciones transcurridas. Si la puntuación de aptitud actual es mejor que la puntuación de aptitud en las generaciones transcurridas, actualizar P best[j] y P best RF[j] con el vector de posición actual.

9. Actualizar el vector de mejor posición global de la siguiente manera: empezando por comparar la puntuación de aptitud actual de cada partícula con la puntuación de aptitud global.

Luego, si la puntuación de aptitud actual es mejor que la puntuación de aptitud global, actualizar G best y G best RF.

10. Actualizar la posición y el vector velocidad de cada partícula en base a las fórmulas de actualización.

11. Repetir los pasos de 7 a 10 hasta alcanzar el número máximo de ensayos.

12. Devuelve la mejor posición global final.

Finalmente, el modelo se entrena con la configuración de HPs establecida, y se reevalúa en un conjunto de datos de prueba, para encontrar el valor de la métrica e identificar el rendimiento real del modelo en escenarios desconocidos.

15. Evaluación a través de las métricas de medición.

Una vez entendido el algoritmo de optimización es importante analizar las métricas que serán utilizadas en la evaluación y comparación de rendimiento. La Figura 15 identifica la matriz de confusión, que almacena de manera matricial los resultados reales y predictivos que permiten sencillez en el análisis de la clasificación (N. Zhu et al., 2022).

Figura 15

Matriz de confusión

Real \ *Predicción*		*Negativo*	*Positivo*
		0 (No presencia – No depósito)	*1 (Presencia – Depósito)*
Negativo	*0(No presencia – No depósito)*	*Verdadero Negativo (VN)*	*Falso Positivo (FP) Error tipo I*
Positivo	*1(Presencia – Depósito)*	*Falso Negativo (FN) (Error tipo II)*	*Verdadero Positivo (VP)*

Nota. Adaptada de Zhu et al. (2022).

Igualmente, en esta matriz el valor de "1" es tomado como etiqueta positiva y el valor de "0" es tomado como etiqueta negativa, además se aclara que en caso de Bank marketing dataset el valor "1" significa la realización del depósito a término fijo y el valor "0" es la no realización. Por otro lado, en el dataset Heart Failure "1" es la presencia de insuficiencia cardiaca y "0" es la normalidad del paciente. Por consiguiente, el análisis asociado a cada uno de los componentes de esta matriz y su relación con las métricas de evaluación se discuten seguidamente.

15.1 Dataset Desequilibrado Bank Marketing

15.1.1 Recall

El *recall* o sensibilidad es una métrica fundamental para el dataset Bank Marketing al centrarse en minimizar los falsos negativos, en otras palabras, aquellos casos en los que el modelo predice incorrectamente que un cliente no realizará un depósito cuando en realidad sí lo hará (Error tipo II). En el sector bancario, esto podría tener como consecuencia perder oportunidades de negocio beneficiosas al no identificar correctamente a los clientes potenciales que están dispuestos a invertir en depósitos. Por tanto, es escogida como métrica central para la evaluación y ejecución del algoritmo, la Fórmula 12 muestra la ecuación en términos de la matriz de confusión.

$$Recall = \frac{VP}{VP + FN} \tag{12}$$

15.1.2 Accuracy

Accuracy o exactitud en este caso aborda la capacidad del modelo *Random Forest* para clasificar asertivamente a los clientes en sus decisiones de depósito. Una alta exactitud indica que el modelo está realizando un buen trabajo en la clasificación de los clientes en las categorías correctas, ya sea que realicen o no un depósito. La Fórmula 13 indica el cálculo de esta.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \quad (13)$$

15.1.3 Precisión

En el caso de Bank Marketing, la precisión, presentada en la Fórmula 14 brindará una medida de qué tan preciso es el modelo al identificar los falsos positivos correctamente, es decir los clientes que el modelo predice que realizarán un depósito a plazo fijo, pero en realidad no lo harán (Error tipo I). Esto proporciona una medida que ayuda a evaluar la calidad de las predicciones en comparación con los clientes reales que no harán dichos depósitos.

$$Precision = \frac{VP}{VP + FP} \quad (14)$$

15.1.4 F1 Score

Proporciona una medida del equilibrio entre la *precision* y el *recall* del modelo. Es decir, tiene en cuenta tanto los falsos positivos (Error tipo I) y falsos negativos (Error tipo II). Al evaluar esta métrica se enfatiza en las predicciones incorrectas en el modelo (FP y FN), pues, debe utilizarse cuando la importancia de las predicciones incorrectas sea similar. Como se muestra en la Fórmula 15.

$$F1 = 2 * \frac{precision * recall}{precision + recall} = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \quad (15)$$

15.2 Dataset equilibrado Heart Failure

15.2.1 Recall

Un alto valor de *recall* o sensibilidad, indicaría que el modelo es capaz de encontrar de manera precisa aquellos pacientes que se encuentran en riesgo de padecer insuficiencia cardíaca. Esto es importante porque se busca minimizar los casos de falsos negativos, es decir, los casos en

los que el modelo no logra detectar a pacientes que realmente necesitan atención médica. Dada su importancia esta métrica se escoge como eje central de la optimización en el presente dataset.

15.2.2 Accuracy

Un alto valor de *accuracy* o exactitud significaría que el modelo está logrando clasificar la mayoría de las muestras de manera correcta, lo cual es importante para obtener resultados confiables y precisos en la detección de insuficiencia cardíaca y saber los clientes exactos que necesitan tratamiento.

15.2.3 Precisión

Se usa para evaluar la clasificación de un paciente cuya predicción es que sufrirá insuficiencia cardíaca, pero en la realidad no sucede (Error tipo I). Así pues, una alta precisión garantiza la clasificación correcta de los pacientes que no sufrirán insuficiencia cardíaca. Por lo tanto, un modelo con alto valor de precisión será competente para identificar con mayor rigor a los pacientes que no están en riesgo de padecer dicha enfermedad.

15.2.4 F1 score

Enfocado en la detección de insuficiencia cardíaca, es importante encontrar un equilibrio entre la capacidad de identificar correctamente los casos positivos (*recall*) y asegurarse de que las predicciones positivas sean precisas (precisión). Indicando el equilibrio entre ambas métricas.

16. Validación del modelo

Una vez planteado el algoritmo de optimización, se validó el rendimiento que le proporciona al modelo de *Random Forest (RF).* Para ello, en el proceso de *benchmarking* se comparó la técnica principal *Particle Swarm Optimization (PSO)* con dos métodos comúnmente utilizados en investigaciones, específicamente *Grid Search (GS)* y *Random Search (RS),* para analizar el comportamiento del modelo e identificar la técnica más efectiva en términos de

funcionalidad y rendimiento en conjuntos de datos equilibrados y desequilibrados. Las condiciones experimentales se garantizaron de forma equitativa para su comparación: preprocesamiento de variables (*One hot encoding*, imputación, y reemplazo), número de ejecuciones (500 ejecuciones dadas las 10 generaciones del PSO) y recursos computacionales (Colab notebook). A su vez, con el fin de realizar un ajuste adicional se consideró examinar dos tipos diferentes de combinaciones de espacios de búsqueda para los cuales se analiza el comportamiento de los hiperparámetros y los resultados de las métricas. Posteriormente, se presentan los resultados obtenidos en los conjuntos de datos previamente presentados, de acuerdo con la experimentación realizada:

16.1 Heart Failure Dataset

Inicialmente se ejecutó el algoritmo sin ajustar el equilibrio de clases (Sin SMOTE) dado, que en caso del conjunto de datos Heart Failure las clases objetivas tiene proporciones similares. De igual manera, se realiza la optimización mediante los métodos usualmente aplicados a fin de su comparación con PSO, es decir, *Random Search y Grid search.* La Tabla 10 muestra los resultados pertenecientes a los valores óptimos de hiperparámetros, según la técnica de optimización y espacio de búsqueda utilizado.

Tabla 10

Resultados valores de hiperparámetros de Random Forest (RF) en Heart Failure

Hiperparámetro	Espacio de Búsqueda	Particle Swarm Optimization	Random Search	Grid Search
n_ estimators	[10,100]	71	40	10
max_depth	[5,50]	16	17	5
criterion	[Entropy, Gini]	Gini	Gini	Gini
max_features	[1,64]	1	1	1

Continuación Tabla 10

Resultados valores de hiperparámetros de Random Forest (RF) en Heart Failure.

min_samples_split	[2,11]	11	8	3
min_samples_leaf	[1,11]	4	8	1
n_ estimators	[50,200]	89	117	50
max_depth	[350,450]	359	350	350
criterion	[Entropy, Gini]	Entropy	Gini	Entropy
max_features	[12,16]	13	12	12
min_samples_split	[2,3]	3	3	2
min_samples_leaf	[1,5]	5	3	4

De igual manera, la Tabla 11 muestra el rendimiento obtenido por las configuraciones de hiperparámetros anteriormente presentadas. La evaluación demuestra que en el caso del espacio de búsqueda 1, PSO redujo el sobreajuste del modelo de manera significativa respecto al modelo básico de *Random Forest.* El rendimiento del modelo es mayor respecto a las demás técnicas de optimización. En concordancia, un aspecto mejorable se relaciona con el alto tiempo de ejecución, el cual puede verse ocasionado por la convergencia de las partículas hacia regiones del espacio de búsqueda que son robustas computacionalmente.

Por otro lado, el espacio de búsqueda número 2 obtuvo buenos resultados de sensibilidad (*Recall*) para el conjunto tratado, siendo esta la métrica objetivo, la cual indicó que el modelo encontró de manera precisa los pacientes que están en riesgo de padecer insuficiencia cardíaca. Igualmente, para las demás métricas el rendimiento fue superior con promedio de 0.9279 en entrenamiento y 0.8882 en pruebas, reduciendo el sobreajuste y aumentando el desempeño en comparación con RF y GS. Un aspecto a tener en cuenta para el *Random search,* es el mayor

tiempo de ejecución respecto a las demás técnicas, esto se debe a evaluaciones con alto precio computacional que ocasionan tal diferencia.

Hay que tener en cuenta que la elección de un espacio de búsqueda respecto a otro depende del problema que se quiera abordar. Los mejores valores que se elijan en cuanto a rendimiento, eficacia y optimización dependerá del problema que se quiera solucionar. En este caso, para manejar la eficacia de ejecución, el tiempo se puede utilizar como parámetro de comparación dado la igualdad de resultados en *Recall* en ambos espacios de búsqueda, es decir, si se requiere saber en menor tiempo cuáles pacientes necesitan ser tratados con urgencia, el que brinda una mejor solución es el espacio de búsqueda No.1, debido a que este tiene menor costo computacional en comparación con el espacio de búsqueda No.2. En esta situación específica el tiempo es un factor indispensable para el hallazgo eficaz de usuarios que padecen la enfermedad.

Tabla 11

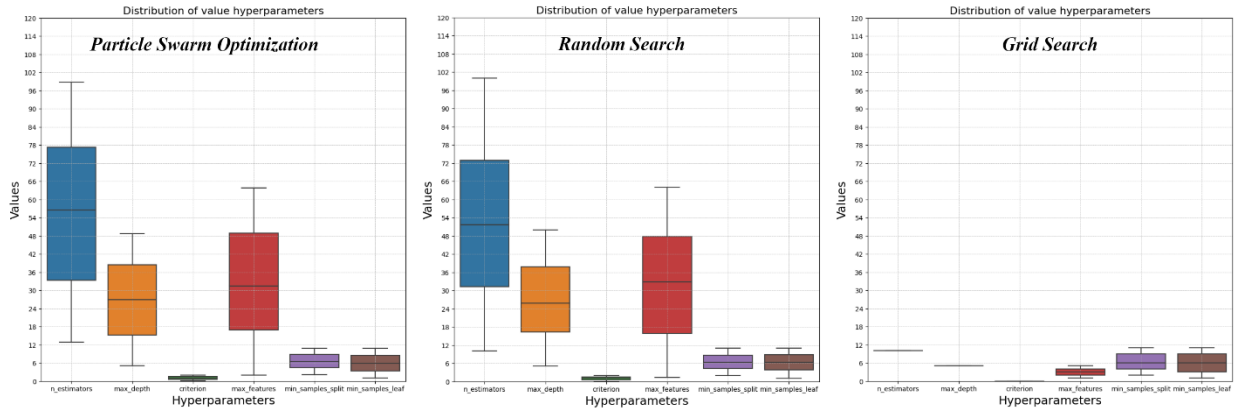
Métricas dataset Heart Failure

Métricas	Espacio de Búsqueda	Particle Swarm Optimization		Random Search		Grid Search	
		Train	*Test*	*Train*	*Test*	*Train*	*Test*
Accuracy	No.1	0.9026	0.8608	0.8720	0.8391	0.8851	0.8217
Precision		0.8886	0.8321	0.8581	0.8169	0.8813	0.8161
Recall		0.9422	0.9370	0.9212	0.9133	0.9160	0.8740
F1-score		0.9146	0.8814	0.8886	0.8624	0.8983	0.8441
Tiempo		6 min, 57 s		7 min, 42 s		1 min, 40 s	
Accuracy	No.2	0.9215	0.8739	0.9520	0.8391	0.94912	0.85217
Precision		0.9181	0.8500	0.9507	0.8169	0.95052	0.82978
Recall		0.9422	0.9370	0.9632	0.9133	0.95800	0.92125
F1- score		0.9300	0.8913	0.9569	0.8624	0.95424	0.87313
Tiempo		14 min, 33 s		17 min, 6 s		7 min, 14 s	

Ahora bien, una vez confirmada la mejora de rendimiento proporcionada por el espacio de búsqueda No.1 y espacio de búsqueda No. 2, es importante analizar el comportamiento del algoritmo PSO. Para ello, la Figura 16 muestra la inicialización de las partículas del enjambre en la primera generación, así como también los valores evaluados por los demás métodos de optimización.

Figura 16

Inicialización de los métodos Particle Swarm Optimization, Random Search y Grid Search



Se observa que, en la primera generación los valores de las 50 partículas de *Particle Swarm Optimization* (PSO) se asimilan a los 500 valores aleatorios de *Random search* (RS), la simetría de los diagramas de caja indica uniformidad en la generación de las posiciones iniciales. A su vez, el método de optimización *Grid search* (GS) no alcanza en las 500 iteraciones a evaluar combinaciones con valores diferentes de algunos hiperparámetros, por lo que se necesitaría mayor cantidad de evaluaciones para alcanzar la uniformidad de PSO y RF. Lo anterior, corrobora que el rendimiento de 50 iteraciones (50 partículas) de PSO es comparable con 500 iteraciones pseudo aleatorias de RS, en cuanto a explotación del espacio; lo que ocasiona que al momento de buscar

el óptimo el enjambre de partículas tenga la capacidad de converger hacia regiones óptimas después de la primera iteración, mejorando el rendimiento respecto a RS y GS.

Igualmente, es importante analizar el comportamiento de las partículas a lo largo de las generaciones, en la Figura 17 y la Figura 18 se ilustran las 6 dimensiones que conforman el espacio de búsqueda.

Figura 17

Comportamiento de las partículas en el espacio criterion, max depth y n estimators

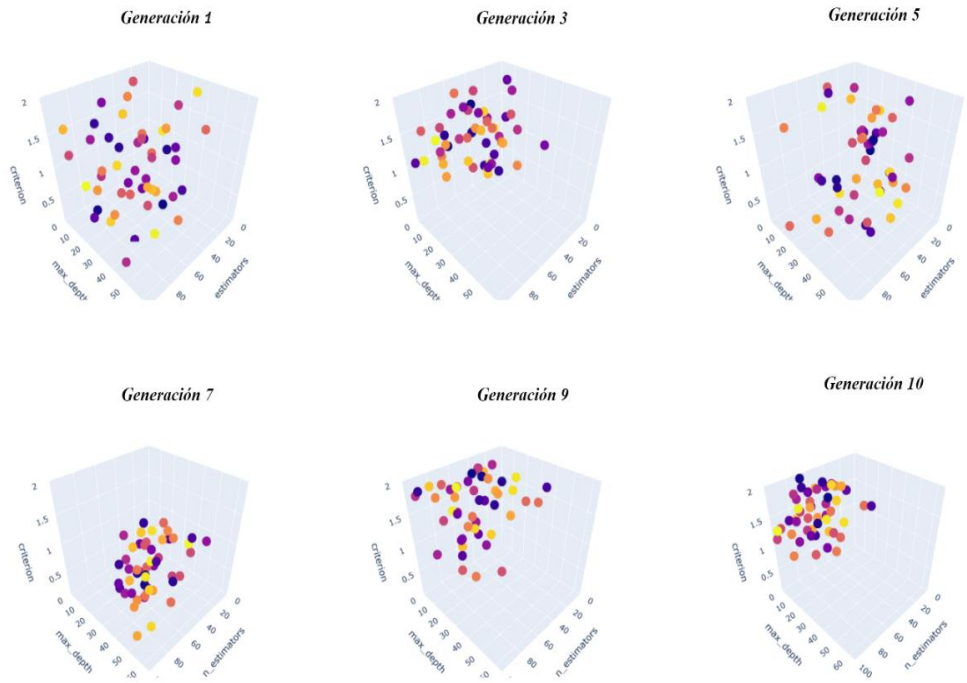
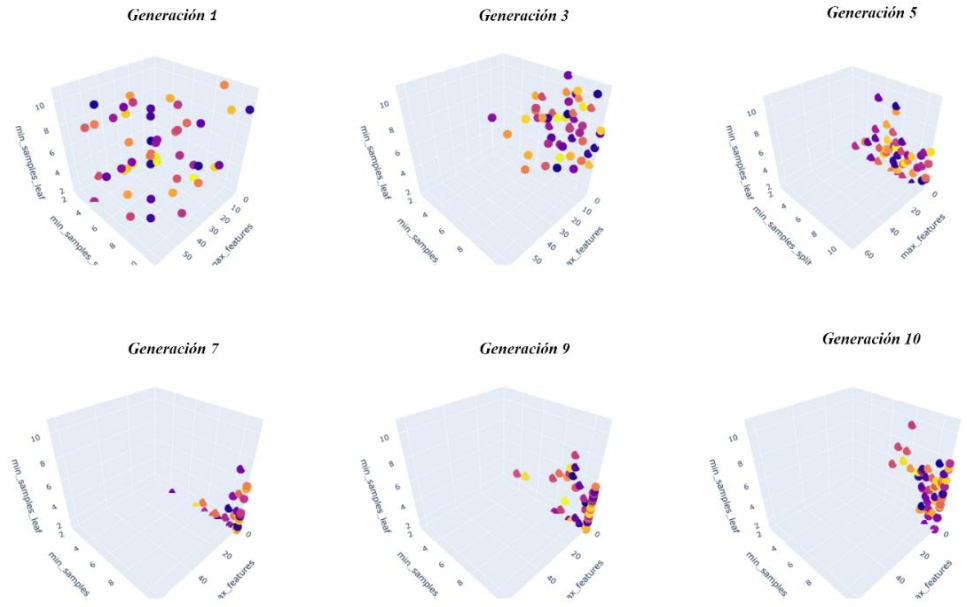


Figura 18

Comportamiento de las partículas en min simples leaf, min simples Split y max features



Al analizar el comportamiento de las partículas a lo largo de las generaciones, se puede evidenciar el factor continuo propio del algoritmo de optimización. Al analizar las posiciones se encuentran indicativos de regiones óptimas del espacio, en relación con los valores de hiperparámetros. Estos aspectos se discuten a continuación:

- criterion: Las partículas se mantuvieron en valores altos (>1) lo que sugiere que *Entropy*(entropía) debe ser utilizada para medir la calidad de las divisiones. Sin embargo, el valor óptimo se encontró como Gini. Esta situación puede estar dada por la sujeción y codificación necesaria para hiperparámetros categóricos, por tanto, sus límites deben mantenerse de manera similar.
- max depth: Los valores de este hiperparámetros se mantuvieron uniformes en el rango de optimización dado, por tanto, no se tiene indicios de una región óptima.

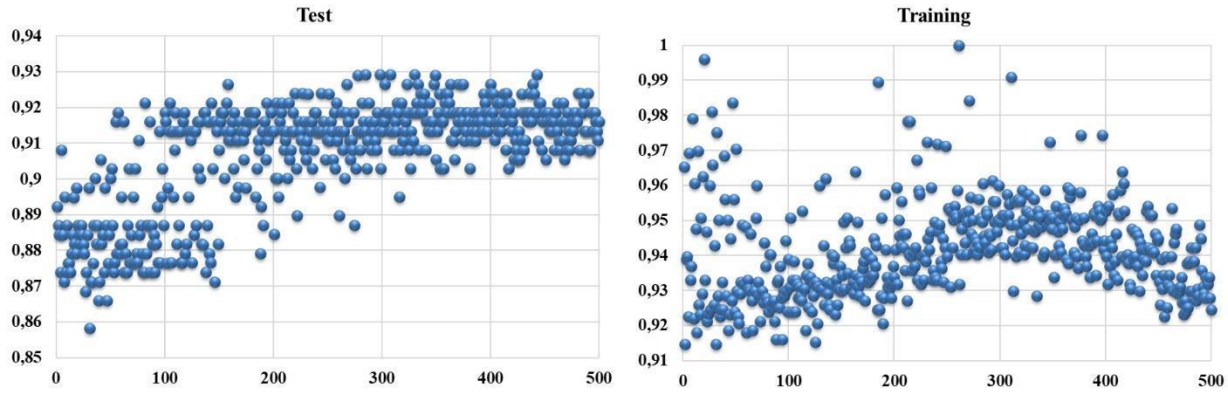
- n estimators: El número de árboles que se construyen en el modelo parece indicar valores adecuados mayores a treinta. Además, dado que no se abandona el espacio de búsqueda se considera que el límite superior es el óptimo.
- min simples leaf: Valores menores a ocho indican rendimiento superior para el número menor de muestras en un nodo hoja. Igualmente, los valores del límite superior no son rebasados al no encontrarse valores óptimos fuera del espacio de optimización.
- min simples Split: Valores mayores a seis para el número mínimo de muestras en la división de un nodo parecen ser los indicados, y dado que el espacio de búsqueda es rebasado podría pensarse en explorar valores superiores.
- max features: Valores superiores a sesenta y cuatro parecen ser reconocidos como óptimos por las partículas. Por tanto, si dentro de un contexto similar se necesita evitar que las partículas se sujeten al espacio, se recomienda aumentar el límite superior a los espacios de búsqueda proporcionados.

Los análisis anteriores sugieren regiones óptimas a explorar que podrían usarse para modificar el espacio de búsqueda inicial, a fin de mejorar la convergencia de las partículas a la solución óptima.

Ahora bien, una vez analizado el comportamiento de las partículas es importante examinar los valores óptimos a lo largo de las generaciones a fin de evidenciar la convergencia de las partículas. Para ello, la Figura 19 muestra los valores en la prueba (Test) y entrenamiento (Train) en relación con las evaluaciones del modelo. Al pasar las generaciones, la optimización encuentra valores más altos en datos no vistos, el sobreajuste en el entrenamiento reduce su valor acercándose a los valores reales predictivos. Por ende, la técnica de optimización convergió hacia el espacio óptimo y logró encontrar la región del espacio que contenía los mejores valores de rendimiento.

Figura 19

Valores del entrenamiento y test.



De acuerdo con los análisis anteriores, el algoritmo de optimización aumentó el rendimiento del modelo logrando converger a regiones óptimas del espacio. También, aspectos asociados a sus valores indican que la elección del espacio de búsqueda puede influir directamente en los resultados y en el tiempo de ejecución, por tanto, es importante la construcción de espacios computacionalmente óptimos y que a su vez tengan valores adecuados de rendimiento.

16.2 Bank marketing

Inicialmente, se aplicó el remuestreo estratificado para el desarrollo del modelo, para lo cual, *Recall* se eligió como métrica clave del dataset; en el primer espacio de búsqueda el modelo logró capturar correctamente 66,1% de las muestras positivas y en los datos de prueba solo 54,39%. De manera similar, para el segundo espacio de búsqueda capturó el 71,4% de las muestras positivas, indicando adecuadamente las instancias positivas durante el entrenamiento. Sin embargo, en el conjunto de prueba se capturó solo el 54% de las muestras positivas reales, revelando que el modelo presentó dificultades al generalizar adecuadamente nuevos datos. Una razón de este inconveniente se debe al sesgo o desequilibrio aún presente en el dataset, lo que causó una disminución en la sensibilidad para detectar instancias positivas.

En consecuencia, se aplicó la técnica de SMOTE a fin de equilibrar las clases. Es importante destacar que esta técnica no interfiere en el proceso de optimización de PSO, sino que se usa exclusivamente para abordar el desequilibrio presente en el conjunto de datos. Por esta razón, se aplicaron las distintas técnicas de optimización de hiperparámetros para encontrar aquella con mejor rendimiento. La Tabla 12 y Tabla 13 muestra la configuración de hiperparámetros óptima y el valor de las métricas respectivamente.

Tabla 12

Valores de hiperparámetros de Random Forest (RF) según la técnica de optimización

Métricas	Espacio de Búsqueda	Particle Swarm Optimization	Random Search	Grid Search
n_ estimators	[10,100]	36	57	10
max_depth	[5,50]	5	6	5
criterion	[Entropy, Gini]	Entropy	Entropy	Gini
max_features	[1,64]	25	18	1
min_samples_split	[2,11]	5	5	2
min_samples_leaf	[1,11]	5	2	5
n_ estimators	[50,200]	50	68	50
max_depth	[350,450]	433	411	350
criterion	[Entropy, Gini]	Entropy	Entropy	Entropy
max_features	[12,16]	13	13	14
min_samples_split	[2,3]	2	2	2
min_samples_leaf	[1,5]	5	5	4

Los resultados para el espacio de búsqueda No.1. indican que la métrica objetivo obtiene su valor más alto con el algoritmo PSO. Las demás métricas que complementan el estudio arrojaron mejores resultados para *Random search* con un costo computacional más alto. Por su parte GS no obtuvo resultados favorables en el dataset, debido principalmente a la evaluación de solo una parte de las configuraciones de hiperparámetros en la cuadrícula combinatoria.

De igual forma, el espacio de búsqueda No.2 obtuvo mejor rendimiento en las métricas para PSO, RS Y GS respecto al modelo *Random Forest* con HPs predeterminados. Además, comparando PSO con RS, no se encontró un rendimiento superior. No obstante, el tiempo de ejecución se redujo considerablemente, por lo cual se obtuvieron desempeños similares con presupuestos más bajos. Adicionalmente, PSO superó a GS en el objetivo de optimización indicando ser la mejor de las técnicas de optimización.

Tabla 13

Métricas dataset Bank Marketing

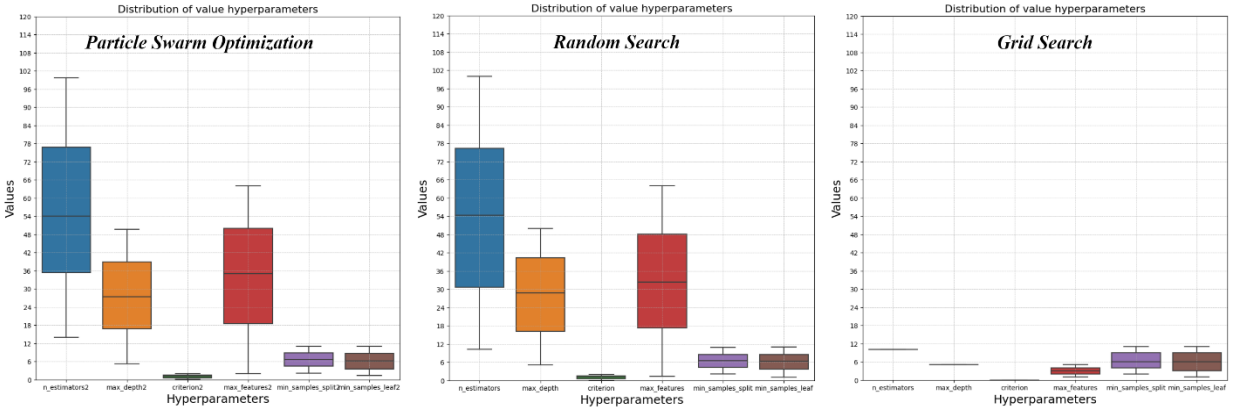
Métricas	Espacio de Búsqueda	Particle Swarm Optimization		Random Search		Grid Search	
		Train	*Test*	*Train*	*Test*	*Train*	*Test*
Accuracy	No.1	0.9008	0.8568	0.9116	0.8676	0.7455	0.7839
Precision		0.8568	0.4352	0.8698	0.4555	0.7777	0.2793
Recall		0.9624	0.9094	0.9682	0.8965	0.6875	0.5810
F1-score		0.9065	0.5887	0.9164	0.6041	0.7298	0.3772
Tiempo		3h,16 min		6 h,13 min		13 min	
Accuracy	No.2	0.9733	0.9096	0.9731	0.9117	0.9100	0.9789
Precision		0.9656	0.5849	0.9658	0.5915	0.9739	0.5877
Recall		0.9815	0.6827	0.9808	0.6991	0.9841	0.6758
F1- score		0.9735	0.6300	0.9733	0.6408	0.9790	0.6287
Tiempo		5h, 44 min		10 h, 49 min		3 h,64 min	

En síntesis, analizando ambos espacios de búsqueda se encontró que el No.1 aportó mayor capacidad predictiva y redujo el sobreajuste en la combinación con PSO. Por su parte, el No.2

aunque logró mejoras, fueron menores al espacio No. 1 en términos de desempeño. Por ende, se muestra el análisis de inicialización en la Figura 20, en el que se analiza las distribuciones en el espacio con mayor rendimiento.

Figura 20

Inicialización de los métodos Particle Swarm Optimization, Random Search y Grid Search



Dicho lo anterior, es importante considerar el espacio de búsqueda abarcado por la primera iteración de PSO respecto a RS y GS analizando las distribuciones. De la misma manera que para el conjunto de datos Heart Failure, el espacio de búsqueda abarcado por la técnica de optimización PSO es similar a RS, lo que confirma la uniformidad lograda en la primera generación. Por otro lado, GS no tuvo evaluaciones suficientes para abarcar un mayor espacio, por tanto, su bajo rendimiento en comparación con RS y PSO es evidente**.** A partir de lo anterior se infiere que independiente del conjunto de datos estudiado, el rendimiento de PSO en cuanto a la explotación y exploración del espacio es mayor que para las demás técnicas, dada su capacidad de convergencia. Igualmente, es importante analizar a lo largo de las generaciones el comportamiento del enjambre, para ello la Figura 21 y la Figura 22 representan el comportamiento de las partículas en las 6 dimensiones que conforman el espacio.

Figura 21

Comportamiento de las partículas en el espacio criterion, max depth y n estimators

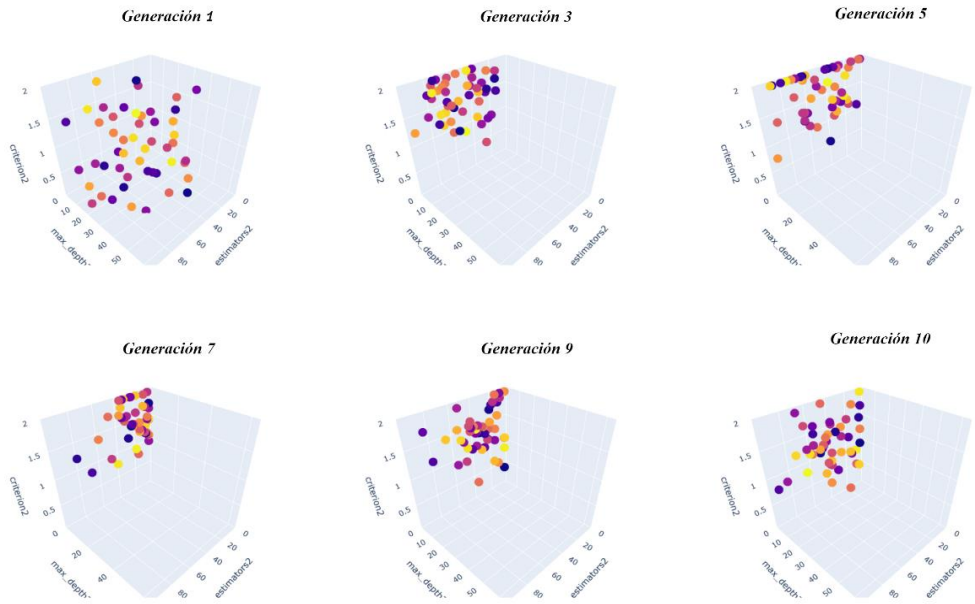
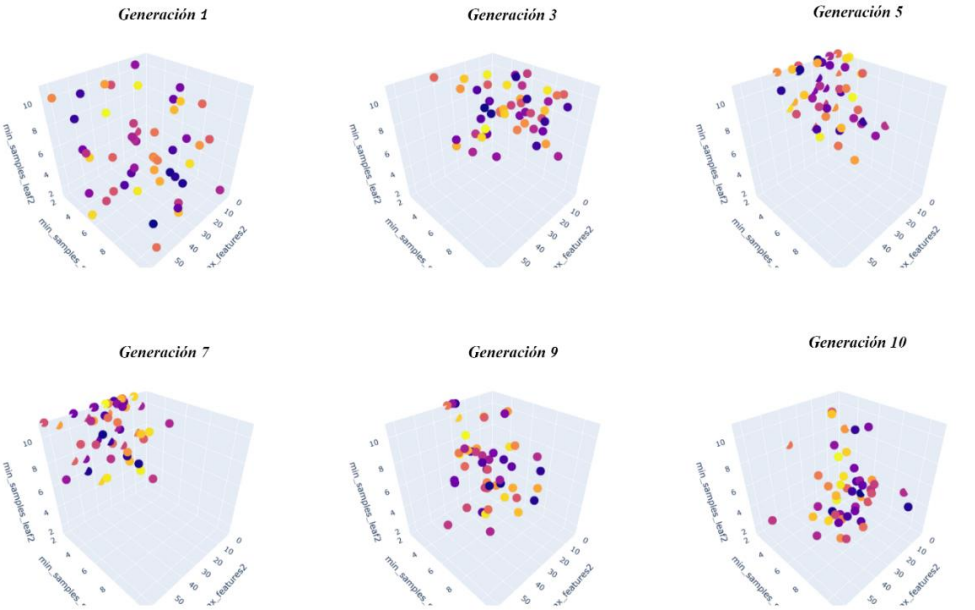


Figura 22

Comportamiento de las partículas en min simples leaf, min simples Split y max features



Acorde con lo anterior, se debe analizar el comportamiento de las partículas a lo largo de las generaciones. Al analizar las posiciones se encontraron aspectos que pueden ayudar a inferir en la elección de espacio de búsqueda con el fin de mejorar los rendimientos del modelo.

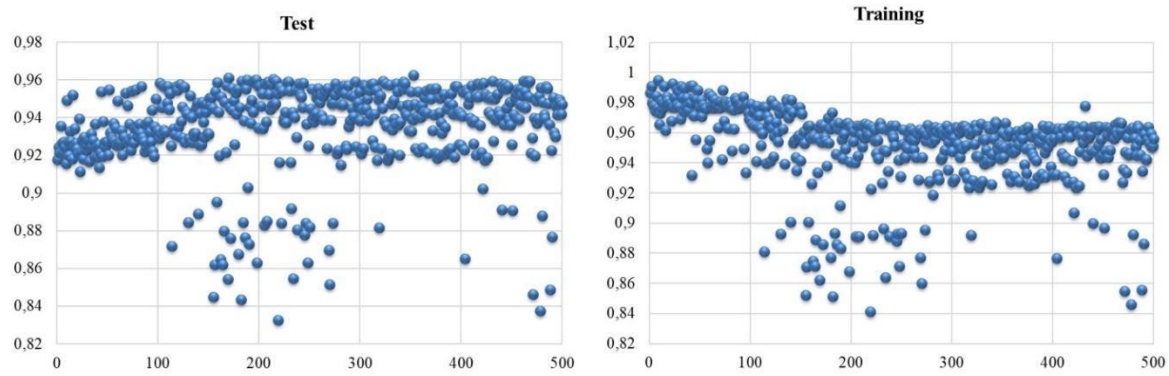
- criterion: Valores mayores a 1 se siguen manteniendo como objetivo de las partículas, sin embargo, con el pasar de las generaciones estas regresan a valores bajos, que en este caso es acorde con los resultados obtenidos. Por lo tanto, se recomienda mantener los valores de criterion para la optimización dado que los valores óptimos obtenidos varían sobre estos.
- max depth: Para la profundidad las cifras indican valores menores a 30 y dado que las partículas no abandonan el espacio de búsqueda su límite superior se puede mantener.
- n estimators: El número de árboles en el modelo no indica la ubicación de una región óptima, dado por la complejidad del dataset.
- min simples leaf: El valor de este hiperparámetros no sugiere regiones del espacio óptimas, por lo cual las partículas no convergen a un valor específico.
- min simples Split: La cifra varía con el pasar de las iteraciones. Sin embargo, en las últimas iteraciones se presenta que valores mayores a 4 son tomados como óptimos.
- max features: Para este hiperparámetro las partículas parecen explorar valores entre diez y cuarenta para las últimas generaciones lo que indicaría la región óptima del espacio.

Los análisis sugieren que las regiones óptimas a explorar podrían modificar la convergencia proporcionando mayor facilidad para encontrar la solución óptima. Adicionalmente, contrario al anterior conjunto de datos, las partículas tuvieron mayor dificultad para encontrar el óptimo, esto puede estar dado por la complejidad del problema, la calidad de los datos y los parámetros seleccionados para la técnica SMOTE.

Posteriormente, al analizar los valores de entrenamiento y prueba indicados en la Figura 23, podemos encontrar que las soluciones óptimas halladas por parte del modelo obtuvieron mejoras de rendimiento moderadas en comparación con las generaciones iniciales. De igual forma, con el pasar del tiempo, lo que es evidente es la reducción del sobreajuste que proporciona un modelo más confiable a la hora de evaluar datos no vistos.

Figura 23

Valores del entrenamiento y test



De manera general, los resultados de las métricas son satisfactorias y mejoraron significativamente la capacidad predictiva en comparación a los valores arrojados por el modelo de *Random Forest* ejecutado sin técnica de optimización. Demostrando la efectividad de los hiperparámetros ajustados dado el óptimo desempeño en los problemas de clasificación tratados.

Comparando PSO con otras técnicas de optimización de hiperparámetros, como *Random Search y Grid Search* que han sido las más usadas y sencillas de manejar. Es notable que de *Random Forest* con PSO se obtienen resultados superiores a diferencia de las otras técnicas, siendo un indicativo de que los hiperparámetros ajustados son adecuados.

Conclusiones

Se destaca la relevancia significativa que tuvo para el modelo de clasificación *Random Forest* el uso de la técnica de optimización de hiperparámetros *Particle Swarm Optimization* (PSO), dado que no solo mejoró la eficacia general del modelo, sino que también posibilitó obtener valores óptimos de hiperparámetros que mejoraron el rendimiento métricas y redujeron el sobreajuste.

Se evidenció que el uso de *Particle Swarm Optimization* (PSO) junto con *Synthetic Minority Oversampling Technique* (SMOTE) aportaron de manera significativamente en el rendimiento y sobreajuste general del modelo de clasificación en el conjunto de datos desequilibrado. Esta combinación no solo redujo el tiempo de ejecución, sino que también incrementó significativamente el valor de las métricas evaluadas, mejorando la capacidad predictiva del algoritmo en general.

En la aplicación de PSO para la optimización del modelo de clasificación *Random Forest*, se encontró una limitación respecto a la falta de información, específicamente la menor proporción de etiquetas de instancias positivas. Esto ocasiona que, si el modelo no tiene la información necesaria para ajustarse en el proceso de optimización, la mejora de rendimiento es sustancial, siendo necesario la aplicación de técnicas adicionales para aumentar la información proporcionada a *Random* Forest y generar con ello gran capacidad predictiva.

El uso de la validación cruzada (CV) como técnica adicional proporciona una evaluación más realista en la ejecución de la optimización de hiperparámetros, pues posibilitó encontrar la mejor configuración dentro del espacio de búsqueda, garantizando una mejora en el rendimiento del modelo en datos no vistos.

El algoritmo de PSO mejora significativamente el rendimiento y reduce el sobreajuste de los algoritmos de *Machine Learning*. Sin embargo, dado su naturaleza vectorial, PSO maneja principalmente hiperparámetros continuos, por lo que es necesario un proceso de codificación y decodificación en el caso de hiperparámetros discretos y categóricos. Esto ocasiona evaluaciones de la función fitness innecesarias, que podrían eliminarse del algoritmo con alguna modificación a sus fórmulas de actualización de velocidad.

Aún sin usar la técnica de SMOTE para manejar el desequilibrio de clases en el dataset desequilibrado, se observó que la aplicación de la técnica de PSO por sí solo tiene un impacto favorable en los resultados de optimización del modelo de *Random Forest*. PSO lleva a cabo una búsqueda exhaustiva en el espacio de búsqueda de los hiperparámetros, ajustando y depurando aquellos valores óptimos para obtener un rendimiento superior del modelo.

El proceso de optimización de hiperparámetros es un proceso largo y complejo, ya que cualquier cambio en los parámetros o implementación incorrecta puede llevar a un descenso del rendimiento, principalmente generado por tiempos largos y resultados bajos respecto al modelo básico. Por ende, el correcto uso de la metodología descrita a lo largo de este estudio permitirá la definición de los espacios de búsqueda en cualquier algoritmo de *Machine Learning*.

Recomendaciones

Antes de entrenar un modelo de *Random Forest* o cualquier algoritmo de *Machine Learning* y combinarlo con PSO, se recomienda que se realice una validación minuciosa de las etiquetas, garantizando que sean precisas, confiables y con la cantidad de información necesaria. Evitando que se introduzcan sesgos o errores que puedan llegar a comprometer la eficiencia de la técnica de optimización.

Se recomienda que, para conjuntos de datos desequilibrados sea usado el remuestreo estratificado como método para reducir el sesgo, el cual es dado por la falta de representación de algunas etiquetas objetivo en el conjunto de datos. Esta técnica mejora la capacidad predictiva del modelo al proporcionar información referente a las etiquetas con menor proporción al modelo *Random Forest.*

Aunque SMOTE es una técnica práctica para abordar el balance de los grupos desequilibrados en un modelo de clasificación, se recomienda tener precaución al seleccionar las instancias usadas como vecinos, porque la calidad de las nuevas muestras generadas por la técnica depende de estos. Hay que tener en cuenta que, si seleccionan instancias muy agrupadas las nuevas observaciones sintéticas generadas pueden llegar a ser redundantes y no aportar valor al modelo. Mientras que, si las instancias seleccionadas son muy diferentes entre sí, puede afectar negativamente el desempeño del modelo de clasificación, reduciendo su rendimiento al contar con información no representativa.

El tratamiento de las partículas del enjambre que abandonan el espacio de búsqueda puede modificarse por uno más adecuado. Debido principalmente al hecho de que, al sujetar los límites, las partículas pueden seguir dirigiendo el enjambre a espacios de búsqueda no factibles, hecho que genera que las partículas abandonen en mayor medida el espacio de búsqueda.

La inicialización del modelo de PSO debe realizarse cuidadosamente, debido a que gran parte del rendimiento del modelo está sujeto a los parámetros del método de optimización. Aún con lo anterior, estudios de parámetros de PSO dirigidos especialmente a algoritmos de ML son escasos. Por lo tanto, es recomendable estudiar a detalle estos parámetros en entornos de optimización a fin de encontrar la mejor combinación de parámetros en PSO.

PSO limita tanto la velocidad como el espacio de búsqueda a fin de evitar el fenómeno conocido como explosión del enjambre. Sin embargo, aunque el problema se soluciona parcialmente al incluir un tamaño de paso pequeño, aún se puede ver un rendimiento sub-óptimo de optimización. Por ende, variaciones como Lbest PSO y Gbest PSO pueden ser implementadas a fin de comparar su rendimiento con el modelo básico.

Al momento de analizar los conjuntos de datos un aspecto principal a abordar es la variación de rendimiento de los algoritmos de *Machine Learning*. Esto ocasiona que para diferentes problemas los algoritmos de ML necesiten cambiar para obtener rendimientos óptimos. Por consiguiente, se podría considerar la elección de diferentes algoritmos como un hiperparámetros adicional, a fin de encontrar la solución óptima del problema abordado.

En la aplicación empresarial de la optimización de hiperparámetros se debe de centrar especial interés en aspectos relacionados con la complejidad computacional del espacio de búsqueda de optimización. Esto es cuando los tiempos de las evaluaciones de cada configuración son altos dada una gran extensión en el conjunto de entrenamiento y alto número de combinaciones de hiperparámetros. Para solucionarlo, es posible aplicar técnicas que reducen la complejidad, ajustando el conjunto de entrenamiento en términos de características e instancias, los hiperparámetros optimizados y los objetivos de optimización. Estas técnicas podrían ser exploradas en futuros trabajos como complemento en la definición de los espacios de búsqueda.

Para la realización de futuras investigaciones enfocadas en la optimización de hiperparámetros, es importante considerar la posibilidad de encontrarse con escenarios donde los rangos del espacio de búsqueda no sean óptimos para configurar la cantidad de variables o hiperparámetros existentes. Para dichos casos, se sugiere acotar el espacio de configuración mediante técnicas de reducción de la complejidad del espacio. Lo cual puede ayudar a limitar las opciones y enfocar la búsqueda en zonas más prometedoras, facilitando encontrar la configuración óptima de hiperparámetros.

Referencias Bibliográficas

Alai, L., Taleb, B., Salgado, D., Rosa, E., & Alonso, R. (2021). *Imputación de datos mediante Random Forest.*

Andonie, R. (2019). Hyperparameter optimization in learning systems. *Journal of Membrane Computing*, *1*(4), 279–291. https://doi.org/10.1007/s41965-019-00023-0

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D., & Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. https://doi.org/10.1002/widm.1484

Camelo García, D. N., & Suárez Suárez, P. C. (2021). Un algoritmo genético para la detección de comunidades en redes sociales mejorado mediante una técnica de clustering. In *Proyecto de grado*. Universidad Industrial de Santander.

Carrillo tarazona, C. (2020). *Revisión de estrategias basadas en inteligencia artificial y machine learning para el control de la operación de sistemas de distribución de energía eléctrica*. Universidad Industrial de Santander.

Chand Bansal, J., Kumar Singh, P., & R.pal, N. (2019). *Evolutionary and Swarm Intelligence Algorithms* (J. Chand Bansal, P. Kumar Singh, & N. R. Pal, Eds.; 1st ed., Vol. 779). Springer Cham. https://doi.org/https://doi-org.bibliotecavirtual.uis.edu.co/10.1007/978-3-319-91341-4

Das, Sibanjan., & Mert Cakmak, Umit. (2018). *Hands-On Automated Machine Learning : a beginner's guide to building automated machine learning systems using AutoML and Python.* (Vol. 1). Packt Publishing.

Fernández, A., García, S., Galar, M., C.Pratii, R., Herrera, F., & krawczykk, B. (2018). *Learning from Imbalanced Data Sets* (1st ed., Vol. 1). Springer Nature Switzerland SG. https://doi.org/https://doi.org/10.1007/978-3-319-98074-4

Gómez Mercado, P. (2021). *Modelo de gestión para determinar plan de cambio de bastidores y orugas aplicando machine learning en la flota de tractores de orugas de cerrejón*. Universidad Industrial de Santander.

He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, *212*. https://doi.org/10.1016/j.knosys.2020.106622

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning; Methods, Systems, Challenges* (cham: Springer International Publishing AG, Ed.; Vol. 1). Springer Nature. https://doi.org/10.1007/978-3-030-05318-5

Javeed, A., Zhou, S., Yongjian, L., Qasim, I., Noor, A., & Nour, R. (2019). An Intelligent Learning System Based on Random Search Algorithm and Optimized Random Forest Model for Improved Heart Disease Detection. *IEEE Access*, *7*, 180235–180243. https://doi.org/10.1109/ACCESS.2019.2952107

Le, Q. V. (2013). Building high-level features using large scale unsupervised learning. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 8595–8598. https://doi.org/10.1109/ICASSP.2013.6639343

Li, H., Liang, Q., Chen, M., Dai, Z., Li, H., & Zhu, M. (2022). Pruning SMAC search space based on key hyperparameters. *Concurrency and Computation: Practice and Experience*, *34*(9). https://doi.org/10.1002/cpe.5805

Lizarazo, S. M. (2018). *Sistema de recomendación para productos bancarios con técnicas de machine learning*. Universidad Industrial de Santader (UIS).

Lutz, B., Reisch, R., Kisskalt, D., Avci, B., Regulin, D., Knoll, A., & Franke, J. (2020). Benchmark of Automated Machine Learning with State-of-the-Art Image Segmentation Algorithms for Tool Condition Monitoring. *Procedia Manufacturing*, *51*, 215–221. https://doi.org/10.1016/J.PROMFG.2020.10.031

Management solutions. (2020). *Auto Machine Learning, hacia la automatización de los modelos.* www.managementsolutions.com

Meisenbacher, S., Turowski, M., Phipps, K., Rätz, M., Müller, D., Hagenmeyer, V., & Mikut, R. (2022). Review of automated time series forecasting pipelines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *12*(6). https://doi.org/10.1002/widm.1475

Mohan, B., & Badra, J. (2023). A novel automated SuperLearner using a genetic algorithm-based hyperparameter optimization. *Advances in Engineering Software*, *175*. https://doi.org/10.1016/j.advengsoft.2022.103358

Motz, M., Krauß, J., & Schmitt, R. H. (2022). Benchmarking of hyperparameter optimization techniques for machine learning applications in production. *Advances in Industrial and Manufacturing Engineering*, *5*. https://doi.org/10.1016/j.aime.2022.100099

Muñoz Cañón, N. D., & Romero Triana, J. A. (2021). Optimización de los hiperparámetros de una máquina de regresión de soporte vectorial utilizando enjambre de partículas para el pronóstico de casos de COVID-19. *Revista UIS Ingenierías*, *20*(2), 181–196. https://doi.org/10.18273/revuin.v20n2-2021015

Owen, Louis. (2022). *Hyperparameter tuning with Python : boost your machine learning model's performance via hyperparameter tuning.* (D. Sugarman & D. Ayare, Eds.; 1st ed., Vol. 1). Packt Publishing Ltd.

Parsopoulos, K. E., & Vrahatis, M. N. (2010). *Particle swarm optimization and intelligence : advances and applications* (1st ed., Vol. 1). Information Science Reference.

Peskova, K., & Neruda, R. (2019). Hyperparameters search methods for machine learning linear workflows. *Proceedings - 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1205–1210. https://doi.org/10.1109/ICMLA.2019.00199

Prudius, A. A., Karpunin, A. A., & Vlasov, A. I. (2019). Analysis of machine learning methods to improve efficiency of big data processing in Industry 4.0. *Journal of Physics: Conference Series*, *1333*(3). https://doi.org/10.1088/1742-6596/1333/3/032065

Rawat, R., & Yadav, R. (2021). Big Data: Big data analysis, issues and challenges and technologies. *IOP Conference Series: Materials Science and Engineering*, *1022*(1). https://doi.org/10.1088/1757-899X/1022/1/012014

Rojas, E. M. (2020). Machine learning: Analysis of programming languages and development tools | Machine learning: Análisis de lenguajes de programación y herramientas para desarrollo. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, *2020*(E28), 586–599.

Simon, D. (2013). *Evolutionary optimization algorithms* (1st ed., Vol. 1). John Wiley & Sons, Inc.

Tambouratzis, G. (2022a). Investigating the Effect of Hyperparameter Values and Size on Swarm Optimization Effectiveness. *Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence, SSCI 2022*, 1636–1643. https://doi.org/10.1109/SSCI51031.2022.10022218

Tambouratzis, G. (2022b). Investigating the Effect of Hyperparameter Values and Size on Swarm Optimization Effectiveness. *Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence, SSCI 2022*, 1636–1643. https://doi.org/10.1109/SSCI51031.2022.10022218

Tayebi, M., & El Kafhali, S. (2022). Performance analysis of metaheuristics based hyperparameters optimization for fraud transactions detection. *Evolutionary Intelligence*. https://doi.org/10.1007/s12065-022-00764-5

Vagaská, A., Gombár, M., & Straka, Ľ. (2022). Selected Mathematical Optimization Methods for Solving Problems of Engineering Practice. *Energies*, *15*(6). https://doi.org/10.3390/en15062205

Wen, L., Ye, X., & Gao, L. (2020). A new automatic machine learning based hyperparameter optimization for workpiece quality prediction. *Measurement and Control (United Kingdom)*, *53*(7–8), 1088–1098. https://doi.org/10.1177/0020294020932347

Wistuba, M., Rawat, A., & Pedapati, T. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, *1*, 1–27. http://arxiv.org/abs/1905.01392

Xu, Z., & Shi, Y. (2015). Exploring Big Data Analysis: Fundamental Scientific Problems. *Annals of Data Science*, *2*(4), 363–372. https://doi.org/10.1007/s40745-015-0063-7

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061

Yang, Z., & Zhang, A. (2021). Hyperparameter optimization via sequential uniform designs. *Journal of Machine Learning Research*, *22*.

Zheng, D., Qin, C., & Liu, P. (2021). Adaptive Particle Swarm Optimization Algorithm Ensemble Model Applied to Classification of Unbalanced Data. *Scientific Programming*, *2021*. https://doi.org/10.1155/2021/7589756

Zhu, N., Zhu, C., Zhou, L., Zhu, Y., & Zhang, X. (2022). Optimization of the Random Forest Hyperparameters for Power Industrial Control Systems Intrusion Detection Using an Improved

Grid Search Algorithm. *Applied Sciences (Switzerland), 12*(20). https://doi.org/10.3390/app122010456

Zhu, W., & Wang, X. (2021). Automated Machine Learning and Meta-Learning for Multimedia. In *Automated Machine Learning and Meta-Learning for Multimedia*. Springer International Publishing. https://doi.org/10.1007/978-3-030-88132-0

Zöller, M.-A., & Huber, M. F. (2021). Benchmark and Survey of Automated Machine Learning Frameworks. *Journal of Artificial Intelligence Research*, *70*, 409–472. https://doi.org/10.1613/JAIR.1.11854