EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CULIACÁN

Desarrollo e implementación de un AutoML en múltiples configuraciones de hardware para el análisis de ambientes distribuidos en tareas de clasificación y regresión

TESIS

PRESENTADA ANTE EL DEPARTAMENTO ACADÉMICO DE ESTUDIOS DE POSGRADO DEL INSTITUTO TECNOLÓGICO DE CULIACÁN EN CUMPLIMIENTO PARCIAL DE LOS REQUISITOS PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

POR:

ING. OSCAR FRANCISCO URIAS FAVELA
INGENIERO EN BIOMÉDICA

DIRECTOR DE TESIS:

DR. HÉCTOR RODRÍGUEZ RANGEL

CULIACÁN, SINALOA septiembre, 2022

Dedicatoria

Dedico la presente tesis a mis padres Jose Oscar Urias Gutierrez y Gemma Guadalupe Favela Ismerio quienes me han ayudado en cada una de las etapas de mi vida aconsejándome y siendo un apoyo emocional durante toda mi vida, junto a ellos he podido lograr muchos de mis objetivos. A Dios que me ha acompañado en durante todo este trayecto de mi vida, quien me otorgo inspiración y calma en los momentos difíciles. Junto a todos ellos se los agradezco desde el fondo de mi corazón.

Agradecimientos

Agradezco a mis **padres** por el apoyo que he recibido durante toda mi vida en cada una de mis decisiones.

A mis **amigos** que estuvieron en cada momento para animarme en los momentos difíciles.

Al **Dr. Héctor Rodríguez Rangel** por su asesoramiento y guía durante la realización de este trabajo.

Al **Lic. Jose Luis López Audeves** por ser un docente apasionado por el aprendizaje y su asesoramiento en temas relacionados con este trabajo.

A la **Dra. Lucía Barrón** por ser un ejemplo de como debe ser una investigadora por excelencia llena de conocimiento y pasión.

Al **Dr. Ramón Zatarain** por compartir su experiencia en la investigación de proyectos.

Al **Dr. Ricardo Quintero** por demostrar una gran pasión por la docencia al momento de compartir su conocimiento.

A las maestras **Ekaterine Peralta y Lucy López** por su gran gestión dentro del departamento de posgrado, siempre llevando un excelente control de trámites.

Al **Tecnológico Nacional De México Campus Culiacán** por ofrecer un espacio de aprendizaje durante mis estudios de maestría.

Al **Consejo Nacional de Ciencia y Tecnología (CONACyT)** por su apoyo financiero que me permitió permanecer enfocado en mis estudios de maestría.

Declaración de autenticidad

Por la presente, declaro que, salvo cuando se haga referencia específica al trabajo realizado por otras personas, el contenido del presente trabajo de tesis es original y no se ha presentado total o parcialmente para su consideración para cualquier otro título o grado en esta o en cualquier otra universidad. Esta tesis es el resultado de mi propio trabajo y esfuerzo, y no incluye nada que sea resultado de algún trabajo realizado en colaboración, salvo que se indique específicamente en el texto.

Oscar Francisco Urias Favela. Culiacán, Sinaloa, México, 2022.

Resumen

En la actualidad el aprendizaje máquina es parte fundamental en el desarrollo tecnológico, debido al amplio campo de aplicaciones que tiene en la industria. Pero el proceso de modelos de aprendizaje máquina requiere de expertos capaces de producir algoritmos de alta complejidad, que sean iterativos, así como una experiencia sólida en el ajuste de hiperparámetros. Como una solución a la falta de expertos en la creación de modelos de machine learning se desarrollaron nuevos sistemas llamados *Automated Machine Learning* (AutoML). Pero recientemente el gran aumento de datos, así como la falta de un análisis sobre las distintas configuraciones de hardware y como estos afectan al AutoML han hecho que estos sistemas requieran de un equipo de cómputo cada vez más costoso, lo que hace que no sean accesible para la organización e investigadores de países en desarrollo y sub desarrollados.

En este proyecto se diseñaron múltiples configuraciones de hardware, las cuales puedan trabajar de manera distribuida para posteriormente implementar en cada una de ellas un sistema de AutoML para la realización de tareas de clasificación y regresión. Cada configuración cuenta con un modelo específico de GPU, las cuales son: RTX 3060ti, RTX 2080ti, GTX 1080 y Quadro M4000. Para el desarrollo del ambiente distribuido, se determinó que el modelo arquitectónico maestro-esclavo sería implementado, ya que permite dividir las cargas del AutoML entre múltiples computadoras. En modelo maestro-esclavo se designa una computadora como el nodo maestro, mientras que las otras trabajaran como nodo esclavo. Para realizar una conexión estable, se buscó implementar la comunicación mediante el *bróker* RabbitMQ, un sistema de gestión de mensajes el cual permite tener mejor control de los datos enviados entre nodos debido a su sistema de colas, organizando de mejor manera los canales de comunicación. Se utilizaron un total de cinco conjuntos de datos, de los cuales se utilizaron dos para tareas de clasificación, mientras que los otros tres fueron para tareas de regresión. Cada ambiente distribuido que se diseñó trabajó con diez experimentos por cada uno de los conjuntos de datos. Con base en los resultados obtenidos durante la experimentación, se realizó un análisis sobre el costo/beneficio que presentan cada ambiente distribuido. De esta manera se puede determinar cuál es la configuración de hardware óptima para la persona u organización que esté desarrollando su propio sistema de AutoML.

Palabras clave

- Aprendizaje máquina
- Aprendizaje profundo
- Auto Aprendizaje Máquina (AutoML)
- Bróker RabbitMQ
- Búsqueda de arquitectura neuronal (NAS)
- Clasificación de imágenes
- Inteligencia artificial
- Modelo Maestro-Esclavo
- Optimización bayesiana
- Optimización de hiperparámetros
- Regresión vectorial
- Sistemas distribuidos

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En los tiempos actuales, la aplicación de técnicas de inteligencia artificial, como las redes neuronales y el aprendizaje profundo, han tenido cada vez más impacto en la tecnología y la sociedad. El aumento de dispositivos inteligentes combinado con lo accesible que se han vuelto para el público general ha generado un mayor flujo de información. Esto ha hecho que el campo de investigación en la inteligencia artificial se ampliara más allá del área informática. Otras áreas de investigación empezaron a presentar una demanda en los sistemas de inteligencia artificial, como lo pueden ser la medicina, con su implementación en tratamientos especializados a pacientes, el apoyo de equipo al personal médico en la toma de decisiones mediante técnicas de aprendizaje profundo durante los pronósticos clínicos, la prevención de enfermedades mediante el reconocimiento de patrones y análisis de imágenes médicas, también ha mejorado el estudio de desarrollo del genoma humano, entre otras aportaciones (Magoulas & Prentza, 1999). Aunado a lo antes mencionado, se han logrado aplicar técnicas de aprendizaje profundo a estudios de mercados que observan tendencias en la bolsa de acciones, herramientas para inversionistas (Bose & Mahapatra, 2001). El desarrollo de las redes sociales ha presentado también un campo de oportunidad en la inteligencia artificial con el análisis de usuarios, tendencias, publicidad dirigida, mejor manejo del flujo de usuarios para empresas como Instagram, Facebook, Twitter, WhatsApp, etc (Balaji et al., 2021). Existen otros campos diversos como lo son: astronomía en la clasificación de estrellas, filtros de mensajería al detectar correos, basura, la biología, ciencia de datos, entre otros (Das et al., 2015).

El aprendizaje máquina es un área que se encuentra dentro del campo de la inteligencia artificial. Tiene como objetivo el reproducir el comportamiento del aprendizaje humano en las máquinas, permitiendo que estas aprendan mediante experiencia (El Naqa & Murphy, 2015). Esto con el fin de eliminar toda necesidad de intuición o conocimiento experto de en los procesos, aunque otros buscan mejorar la colaboración entre el experto y computadora.

Dentro de aprendizaje máquina, se tiene los métodos de aprendizaje profundo, que a diferencia de los métodos estadísticos no pueden utilizar de manera satisfactoria múltiples tipos de información de manera simultánea Boulesteix & Schmid, 2014. Las técnicas de aprendizaje profundo han demostrado la capacidad de aprender patrones a partir de diferentes tipos de datos (Fahle et al., 2020). Por lo tanto, se han presentado múltiples ventajas con el uso de la inteligencia artificial en la industria, tales como realizar tareas de alta complejidad en un menor tiempo en el que lo realizaría un ser humano de manera normal. Concretar tareas de un alto nivel de estrés y complejidad de manera sencilla, tiene un alto índice de éxito en cumplir las tareas específicas, requiere una menor cantidad de espacio y peso para realizar ciertas tareas que requieran de varias personas, como también puede encontrar patrones que no son detectables a simple vista por una persona (Khanzode & Sarode, 2020).

Ahora bien, se deben señalar algunos de los principales problemas a los que se enfrenta el aprendizaje máquina. Por ejemplo, la cantidad de poder computacional necesaria para ejecutar los algoritmos son demasiado altos. Para su desarrollo es necesario de un experto de aprendizaje máquina. Además, se tiene la limitante de que la cantidad de expertos del área de la inteligencia artificial es baja comparada con otros campos de la informática. Por lo que es difícil encontrar personas capacitadas con una experiencia sólida en el diseño de algoritmos de optimización y el ajuste de hiperparámetros (Bhbosale et al., 2020). Otro de los problemas que se encuentran es que cada área de investigación es única, presentando sus propios retos y desafíos a resolver, por lo que la solución a sus problemas es particular (Zoph & Le, 2016).

Una de las soluciones propuestas a la problemática de la falta de expertos y sus múltiples aplicaciones, es el desarrollo de sistemas Auto Aprendizaje Máquina (AutoML). Estas técnicas permiten la obtención de configuraciones de arquitecturas de Aprendizaje Máquina con mejor rendimiento que los métodos tradicionales. A su vez, implementan sistemas de búsqueda donde se realiza una exploración sistemática de hiperparámetros y/o arquitecturas

que permitan la obtención de modelos óptimos de aprendizaje máquina para dar solución a diferentes tareas o problemas.

Algunas de las empresas más importantes de la industria tecnológica han trabajado en el desarrollo de este tipo de sistemas (AutoML) son Google, Amazon y Microsoft. Donde, Google con su servicio de Google Cloud tienen su sistema llamado *Cloud AutoML*. Otras grandes empresas tecnológicas también utilizan sus servicios en la nube para ofrecer estas herramientas como lo son Amazon con Amazon Web Service (AWS) *AutoML solutions*, *Azure AutoML* de la compañía Microsoft o *H20 AutoML cloud* de la empresa H2O.ia. Sin embargo, el uso de estas tecnologías tiene un alto costo, lo cual puede no ser accesible para personas no profesionales, universidades u organizaciones provenientes de países en desarrollo o subdesarrollados.

Los sistemas de AutoML son una herramienta que ayuda a los expertos en el desarrollo de mejores modelos de aprendizaje máquina y hacen accesible a que personas con una menor expertis puedan crear sus propios modelos de inteligencia artificial. Por otro lado, debido a la cantidad de datos que se producen día con día, ocasionan algunos problemas adicionales. Estos son, la limitación de hardware que tienen los equipos de cómputo menos especializados hace que los costos para adquirir el equipo necesario sean demasiado altos y sea inaccesible para muchas personas u organizaciones, este combinado con la cantidad de procesos complejos que requiere un sistema de AutoML para generar el modelo óptimo, hacen que se vea afectada la precisión y el tiempo que tarda en ejecutar el sistema.

Debido al problema antes presentado, la aplicación de ambientes distribuidos permite resolver problemas de computación masiva. Mediante la aplicación de múltiples computadoras organizadas en forma de clústers, se puede implementar una infraestructura distribuida para dar solución al problema anterior. Entre los beneficios de la aplicación de cómputo distribuidos se encuentran: la distribución de procesos en distintas ubicaciones, la optimización de equipos y mejora de balance. La ejecución de aplicaciones en una arquitectura centralizada hace que los sistemas se vuelvan más ineficientes y requieran de más tiempo, esto por la gran carga de trabajo que requieren las aplicaciones.

Durante la elaboración de este proyecto se diseñó e implemento múltiples configuraciones de hardware distribuidos, los cuales fueron utilizados para repartir los procesos de un

sistema de AutoML en tareas de clasificación y regresión. Así pues, se utilizó como métricas de evaluación el tiempo, rendimiento y costos del sistema. Esto con el objetivo de realizar un análisis con base a los resultados que permitirá determinar cuál configuración es la que ofrece un resultado adecuado a unos costos accesibles. Se trabajó una arquitectura maestro-esclavo, la cual permite organizar distintos equipos en nodos esclavos y un nodo maestro. Esto combinado al uso de un Bróker para la gestión de colas, las cuales permiten un mejor flujo de información entre los nodos, permite dividir las cargas de trabajo, mejorar los tiempos de trabajo con respecto a una única computadora, así como presentar costos más accesibles para grupos que no cuenten con los recursos necesarios para la implementación de sistemas de Machine learning.

Las distintas configuraciones se basan principalmente en el uso de múltiples GPU, así como el diseño de la configuración. El primer diseño consiste en utilizar una computadora con múltiples GPU, las cuales harán la función de nodos esclavos realizando las tareas asignadas, mientras que el procesador actuar como Nodo Maestro, gestionando las tareas y administrando los resultados, este diseño se implementa con los modelos RTX 2080Ti y GTX 1080Ti en distintas configuraciones. Para el caso del segundo diseño, donde se utilizaron las GPU dedicadas RTX 3060Ti y la serie Quadro M4000, se implementó múltiples computadoras las cuales estarán conectadas mediante una conexión LAN, cuatro actuarán como Nodo Esclavo y uno como Nodo Maestro.

1.1. Definición del problema

En la época actual, existe cada vez más el uso de sistemas inteligentes para resolver problemas de alta complejidad, así como las aplicaciones y el uso de la tecnología han estado formando cada vez parte más fundamental de nuestra vida diaria, el uso de modelos inteligentes que faciliten la vida ha estado en auge en la mayor parte del sector industrial. Esto se debe también a la gran cantidad de datos que se requieren procesar, creando nuevos retos para la industria, los cuales la tecnología convencional no puede solucionar.

De manera más formal, la resolución de problemas se realizaba de manera estructurada con base en que un especialista en cierto campo de la industria aplicaba sus conocimientos a

las máquinas, para que estas pudieran resolver los inconvenientes que se pudieran presentar mediante modelos de aprendizaje automático. En la actualidad, la gran cantidad de datos que se pueden obtener de los sistemas computacionales convencionales hace que un especialista se le dificulte cada vez más obtener el mejor resultado posible mediante el método tradicional de búsqueda de hiperparámetros, esto en parte a la incertidumbre de la gran cantidad de cálculos matemáticos que esto llevó consigo realizar.

Como respuesta a estos inconvenientes se empezaron a desarrollar nuevos sistemas de aprendizaje automático denominados AutoML (Automated Machine Learning). Los modelos de autoaprendizaje automático vinieron a revolucionar el campo de la inteligencia artificial, debido a que presentan la capacidad de que el sistema sea capaz de determinar la mejor solución posible al problema que se ha planteado resolver. Quitando de esta manera gran parte de la carga de trabajo de crear el modelo más efectivo para la tarea a realizar al especialista responsable.

Ahora lo siguiente en cuestión, es que la gran cantidad de datos que los nuevos sistemas de aprendizaje automático pueden procesar, está limitado por el hardware con el cual se está trabajando, aumentando cada vez más los costos de las supercomputadoras para el desarrollo de los nuevos modelos de inteligencia artificial. Creando así el problema más reciente en estos sistemas y es su capacidad de procesamiento limitada, resultando en aumento de los costes y el tiempo en que se generan los modelos más precisos.

Ahora que se han planteado las distintas problemáticas que se presentan, surge la necesidad que se desarrolle un sistema distribuido que permita repartir los procesos de los modelos de manera local. Para así mejorar el tiempo de desarrollo de estos modelos y optimizar los resultados que se pueden obtener, así como la reducción de costos. Actualmente, aunque se cuente con microservicios en la nube que permiten el desarrollo de estos ambientes distribuidos. Así como también permite rentar y personalizar el diseño de supercomputadoras especializadas en el campo de la inteligencia artificial, no se tiene un análisis conciso de cuál podría ser la mejor configuración según el problema que se esté trabajando. Por tanto, suele ser recurrente que los costos de producción se eleven demasiado a pesar de que inicialmente ya son altos. Esto presenta un problema para la mayoría de los investigadores y organizaciones en el mundo, sobre todo en los países en desarrollo o subdesarrollados.

1.2. Hipótesis

El diseño e implementación de diferentes configuraciones de ambientes distribuidos en un sistema de AutoML, para la solución de problemas de clasificación y regresión, permitirá desarrollar un análisis que determinará la configuración de hardware distribuida con mejor costo beneficio.

1.3. Objetivo

En esta sección se describen los objetivos deseados durante el proyecto, así como también una serie de pasos específicos a seguir, los cuales serán necesarios para obtener los resultados deseados.

1.3.1. Objetivo general

Diseñar, implementar y analizar el rendimiento de un sistema de autoaprendizaje máquina en diferentes configuraciones distribuidas para tareas de clasificación y regresión.

1.3.2. Objetivos específicos

- Diseñar e implementar distintas configuraciones de hardware en ambientes distribuidos para la resolución de problemas de regresión y clasificación.
- Realizar la ejecución de un sistema de AutoML en distintas configuraciones de hardware distribuidas aplicando múltiples conjuntos de datos.
- Realizar un análisis de los resultados obtenidos del sistema de AutoML implementado en las distintas configuraciones de hardware.
- Redactar la documentación correspondiente al análisis de resultados con base en los experimentos realizados en distintas configuraciones distribuidas.

1.4. Justificación

Las soluciones actualmente desarrolladas siguen siendo un impedimento que utilizar para algunos equipos de trabajo por su mismo costo. Además, otros equipos que tengan un capital suficiente pueden presentar limitaciones por algún tipo de política y seguridad que no permitan que proveedores externos tengan algún control sobre la información. Aquí es donde entran las soluciones de código abierto, debido a que ayudan a estos equipos con su desarrollo, manteniendo su sistema cerrado y acercándose al área de la inteligencia artificial.

Muchos equipos o personas, ya sea de ámbito académico y empresarial, suelen tener a su disposición equipos de cómputo que no estén a tiempo completo en uso. Por lo tanto, con el objetivo de aprovechar el hardware existente, se implementa un procesamiento distribuido entre múltiples computadores para repartir la carga. De tal manera que, se puedan encontrar modelos con alta aptitud utilizando un amplio espacio de búsqueda de arquitecturas para problemas de clasificación y regresión.

Los sistemas de AutoML son herramientas que se pueden utilizar casi en cualquier tipo de hardware computacional. Como consecuencia, su desempeño estará limitado a la capacidad que tenga el hardware que se esté utilizando, así como el diseño que se esté implementando. Los componentes especializados para el desarrollo de estos sistemas suelen tener costos muy elevados como para que cualquier persona u organización los adquiera. Un correcto análisis sobre los diferentes tipos de diseños que puede presentar el hardware, así como determinar un modelo centralizado o distribuido, puede determinar el hardware óptimo para cada proyecto. Haciendo de esta manera que se reduzca aún más los costos de utilizar un sistema de AutoML mejorando su rendimiento.

La optimización de modelos de aprendizaje máquina son una de las principales áreas dentro de la inteligencia artificial, pues se encarga de la mejora de dichos modelos. De tal manera que, se encuentran soluciones óptimas a los diferentes problemas que se enfrentan. Aunque en la actualidad se cuentan con soluciones a estos problemas, algunas de estas soluciones son privadas y requieren una subscripción de paga para ser utilizados, pero cuentan con excelentes resultados y un cierto grado de facilidad de uso. Un ejemplo es AutoML de Google, el cual maneja distintos tipos de tarifa dependiendo de la configuración del hardware y la tarea

que se realice. En el caso de clasificación de imágenes se maneja el precio de $3.465 dólares por hora en la capacitación de uso del sistema, después se cobra $18 dólares por el número de horas que dure en entrenamiento, también se cobra por cada implementación del modelo generado en $1.375 dólares por implementación. Si se realiza un cálculo se puede estimar que se puede llegar a pagar $2880 dólares en promedio mensuales, aunque si los datos que utilizamos son muy pesados, aumentando así las horas de entrenamiento se puede llegar a pagar $13,544 dólares mensuales. También existe la opción de personalizar la configuración de hardware que se utilizara, aumentando los costos de producción al agregar hardware más especializado, lo cual no es accesible para todas las personas.

Gran parte del éxito de las técnicas de aprendizaje máquina se debe a la colaboración que son capaces de aplicar con otras áreas de conocimiento. Estas áreas son capaces de generar un gran conjunto de datos propios, los cuales pueden obtener información valiosa para ser implementada. Esta información en conjunto con el aprendizaje máquina es capaz de generar conocimiento valioso para la ciencia. Mediante la aplicación de la solución planteada se ampliará el espectro de personas expertas en otras áreas, teniendo la posibilidad de acceder a estas nuevas características, y aumentar así, su área de conocimiento.

1.5. Estructura de la tesis

El presente trabajo está dividido en 6 capítulos, incluyendo este capítulo de introducción, y se organiza de la siguiente forma:

- Capítulo 1.- Introducción: Se presentó y abordo la definición del problema al que se enfrenta este trabajo, seguido de la hipótesis y objetivos que esperan resolver el problema presentado, y finalmente, se presenta la justificación del “porque” se realizó este trabajo.

- Capítulo 2.- Marco Teórico: Se describe la teoría fundamental de los temas principales que conforman el proyecto, como la inteligencia artificial, aprendizaje máquina, aprendizaje profundo, tipos de aprendizaje, algoritmos de optimización y sistemas distribuidos.

- Capítulo 3.- Estado del arte: Se presentan las investigaciones y propuestas actuales que se encuentran en desarrollo de temas relacionados, como el aprendizaje máquina automatizado y optimización de hiperparámetros, presentando un resumen de la metodología que implementan y sus resultados que servirán a modo de comparación.

- Capítulo 4.- Metodología: Se describe el proceso de desarrollo que se tuvo para la creación del sistema y los argumentos lógicos que se tomaron para las decisiones de diseño, como el preprocesamiento de la información, el procedimiento de optimización de modelos y el diseño de los espacios de búsqueda, así como la implementación de las herramientas tecnológicas utilizadas.

- Capítulo 5.- Análisis de resultados: Se muestran los resultados obtenidos del sistema en los diferentes módulos implementados junto con los diferentes conjuntos de datos utilizados, se analiza los resultados de la perdida de información y el tiempo de optimización tomado para obtenerlo.

- Capítulo 6.- Conclusiones: Se describe un resumen de la metodología implementada en el proyecto y de la aportación que ofrece al área del aprendizaje máquina automatizado. Este capítulo es complementado con algunas ideas de mejoras que podrían implementarse, en este proyecto o similares, a futuro.

Capítulo 2

Marco teórico

En este capítulo se describen una serie de conceptos, los cuales presentan la información necesaria en la cual está basado el presente trabajo. El capítulo comienza con conceptos básicos y generales para después introducir conceptos específicos a este trabajo. Inicialmente, se comienza con el concepto de la inteligencia artificial y su historia. Después, se definen una serie de conceptos sobre el aprendizaje máquina. Se avanza a una serie de conceptos que definen parte del aprendizaje profundo y posteriormente se definirán algunos conceptos sobre el aprendizaje máquina automatizado. Finalmente, se incluye el tema de sistemas distribuidos y las aplicaciones utilizadas.

2.1. Inteligencia artificial

Para llegar a la definición de inteligencia artificial, es necesario primero definir que es la inteligencia. La inteligencia es la capacidad que tienen las personas o animales para resolver problemas o alcanzar metas específicas (McCarthy, 1998). Basado en lo anterior, podemos decir que la inteligencia artificial se basa en crear máquinas capaces de resolver problemas como lo haría una ser humano (Negnevitsky & Intelligence, 2005).

La forma en la que actualmente se habla sobre la inteligencia artificial, se refieren a máquinas que son capaces de realizar una o más tareas. Algunas de estas son: Entender el lenguaje humano, realizar tareas mecánicas que involucren maniobras complejas, resolver problemas complejos basados en computadora que posiblemente involucren grandes volúmenes de datos en un tiempo corto y que retorne respuestas de manera humana (Joshi, 2020).

2.2. Historia de la inteligencia artificial

En 1943 se plantea el primer trabajo reconocido en el área de Inteligencia Artificial (IA) por Warren McCulloch y Walter Pitts. Mediante una investigación en el sistema nervioso central, realizaron el primer modelo de neuronas cerebrales. El modelo propuesto consiste en que cada neurona inicia con un estado binario, en cualquier condición encendido o apagado. De esta manera se demostró que cualquier función cerebral puede ser computada por alguna red de neuronas conectadas (Negnevitsky & Intelligence, 2005).

Otro de los fundadores de la IA es el matemático Húngaro John Von Neumann. Mediante el apoyo y financiamiento a dos estudiantes graduados del departamento de matemáticas de Princeton, Marvin Minsky y Dean Edmons, construyeron la primera computadora de redes neuronales en el año de 1951 (Joshi, 2020).

Otro de los investigadores de la primera generación fue Claude Shannon. En 1950 pública un artículo sobre máquinas que juegan ajedrez. Demostrando que, la computadora de Von Neumann, aunque pudiera examinar cada una de los posibles movimientos a una velocidad de microsegundo por jugada, tomaría $3x10^{106}$ años para realizar el primer movimiento. De esta manera se demuestra la necesidad de heurísticas para la búsqueda de soluciones (Benko & Lányi, 2009).

En 1956 John McCarthy, Martin Minsky y Claude Shannon se encargarían de organizar el primer verano de investigación de Inteligencia Artificial en el Dartmouth College. Esto trajo consigo que muchos investigadores estuvieran interesados en el estudio de la inteligencia máquina, redes neuronales artificiales y teoría de autómatas. Un total de 10 investigadores se reunieron iniciando una nueva ciencia llamada inteligencia artificial (McCarthy, 1998).

En los años de 1960, los investigadores intentaron simular el proceso de pensamiento complejo, mediante la invención de métodos generales para resolver amplias clases de problemas. Utilizando el mecanismo de búsqueda de propósito general para buscar una solución al problema. Tales acercamientos, se les conoce ahora como métodos débiles, aplicando información débil acerca del dominio del problema, el resultado obtenido fue un rendimiento débil de los programas desarrollados (Stuart R, 2010).

Sin embargo, para los años 1970 la euforia inicial de la IA se fue, los gobiernos cancela-

ron las fundaciones a proyectos de IA debido a que no tenían una aplicación fuera de jugar juegos. Existen múltiples razones para que esto pasara, siendo las principales: los investigadores de la IA desarrollaban métodos generales para una gran gama de diferentes problemas; también muchos problemas que se intentaban resolver fueron demasiado amplios y difíciles; Por último, en 1971 el gobierno británico suspendió el apoyo a las investigaciones de la IA.

Durante los años 1970 se desarrolló una tecnología llamada sistemas expertos. Estos sistemas son capaces de hacer inferencias simples dada una base de hechos y un conjunto de reglas definidas por un experto humano. Estos sistemas tuvieron un número creciente de aplicaciones a finales de los 1970, demostrando que la tecnología de IA podría pasar con éxito de los laboratorios de investigación a un ambiente comercial. Aunque al inicio la necesidad de hardware costoso y lenguajes de programación complicados se dejó en manos de desarrolladores expertos. En los años 1980, con la llegada de las computadoras personales (PC) y el desarrollo de herramientas de sistemas expertos fáciles de usar, permitieron que ingenieros e investigadores de todas las disciplinas tuvieran la oportunidad de desarrollar sistemas expertos (Stuart R, 2010).

En el año de 1986 se desarrolla el algoritmo de aprendizaje de propagación hacia atrás. Este fue el algoritmo que se convirtió en la técnica más popular para el entrenamiento de perceptrones multicapa (redes neuronales). Lo anterior combinado con las limitaciones de los sistemas expertos, sobre que no todo conocimiento puede ser expresado con base en reglas, permitió que en los años 1990 se empezaron a utilizar las redes neuronales como alternativas para la extracción de conocimiento escondido en largos conjuntos de datos (Mitchell et al., 1997).

2.3. Aprendizaje máquina

El término aprendizaje máquina tiene como definición el estudio de modelos y algoritmos estadísticos que utilizan los sistemas informáticos para realizar tareas sin necesidad de ser programado de manera explícita por la persona que desarrollo el sistema (Mahesh, 2020).

El aprendizaje máquina involucra una serie de mecanismos adaptativos que permite a las computadoras a aprender de la experiencia, ejemplo y analogía. Las capacidades de aprendi-

zaje pueden ir mejorando en el rendimiento de un sistema inteligente sobre el tiempo (Negnevitsky & Intelligence, 2005). Dicho de una manera más formal se encuentra la siguiente definición: "Un programa de computadora se dice que aprende de la experiencia *E* con respecto a una clase de tareas *T* y medida de rendimiento *P*, si su rendimiento con respecto a las tareas *T*, medidas por *P*, mejoran con la experiencia *E*" (Mitchell et al., 1997).

En el aprendizaje máquina, el aprendizaje ocurre extrayendo la mayor cantidad de información del conjunto de datos, también llamado comúnmente conjunto de datos. A través de algoritmos que analizan la estructura base de los datos y distinguen las señales del ruido, encuentran la señal, o el patrón que definirá su salida (Conway & White, 2012).

Los algoritmos de aprendizaje máquina pueden ser clasificados en varios tipos, cada uno de estos se puede dividir en función de la naturaleza de los datos utilizados, estos son (El Naqa & Murphy, 2015):

- Aprendizaje supervisado: El objetivo del aprendizaje supervisado es utilizar un conjunto de datos conocidos para estimar un conjunto de datos desconocidos, los cuales al final nos entregan un conjunto de datos etiquetados, pueden ser unos ejemplos la clasificación y regresión.

- Aprendizaje no supervisado: En este aprendizaje el conjunto de datos es una colección de muestras no etiquetadas. Donde x es un vector de características, y la meta del algoritmo consiste en crear un modelo que tome el vector de características x como entrada y lo transforma en otro vector, o si no, a un valor que puede ser utilizado para resolver problemas prácticos.

- Aprendizaje semi-supervisado: En este tipo de aprendizaje el conjunto de datos contiene tanto muestras etiquetadas como no etiquetadas. Usualmente, la cantidad de muestras no etiquetadas es mucho mayor que el número de muestras etiquetadas. Su objetivo es el mismo que el aprendizaje supervisado con la esperanza que al utilizar muestras no etiquetadas se produzca un mejor modelo.

- Aprendizaje por refuerzo: Es un subcampo del aprendizaje máquina donde la máquina "vive" en un ambiente y es capaz de percibir el estado de ese mismo ambiente como un

vector de características. La máquina realiza acciones en cada uno de los estados del ambiente. Su objetivo consiste en aprender una política, definida como una función f que toma el vector de características de un estado como entrada, obteniendo de salida una acción óptima a ejecutar.

Cada uno de estos tipos de aprendizaje ofrecen diferentes soluciones dependiendo de las características del problema. En este trabajo se enfocará en el aprendizaje supervisado, el cual será profundizado a continuación.

2.3.1. Aprendizaje supervisado

Los algoritmos de aprendizaje supervisado consisten en un par de vectores los cuales realizan la función de entrenamiento, en el cual un vector representa los valores de entrada con los cuales entrenaras la red y el otro son los valores que deseamos obtener a la salida. El objetivo del aprendizaje supervisado es crear una función que pueda predecir el valor correspondiente a cualquier objeto de entrada válido después de ver el vector con los valores deseados. De manera más resumida podemos decir que el aprendizaje supervisado se caracteriza por aprender a identificar un objeto en función de un conjunto de objetos previamente seleccionados y que cuentan una característica específica (Bengio & Courville, 2017). Con el fin de resolver un determinado problema de aprendizaje supervisado, uno tiene que considerar varios pasos:

- Determinar el tipo de datos que se utilizará para entrenar el modelo.
- Reunir un conjunto de objetos y entrada que se recopila y salidas correspondientes se recogen también.
- Determinar la función de ingreso de la representación de la función aprendido. La precisión aprendida depende en gran medida de como el objeto de entrada está representado.
- Determinar la estructura de la función adecuada para resolver y el problema y la técnica de aprendizaje correspondiente.

- Completar el diseño, el ingeniero ejecuta el algoritmo de aprendizaje en el conjunto de la formación obtenida. Parámetros del algoritmo de aprendizaje pueden ser ajustados mediante la optimización de rendimiento en un subconjunto de ellas (conjunto de validación).

2.3.1.1. Clasificación

En aprendizaje automático, la clasificación estadística es el problema de identificar una etiqueta de clase y colocarla en una nueva observación, esto sobre la base ya establecida previamente de un conjunto de datos que cuentan cada uno con su etiqueta correspondiente a su clase. Dependiendo del número de etiquetas que se tenga asignado, se puede afirmar que una clasificación es binaria o multiclase (Burkov, 2019).

La clasificación tiene como objetivo predecir las etiquetas de un nuevo objeto observado en función de los registros pasados. Dependiendo del número de etiquetas que se tenga asignado, se puede afirmar que una clasificación es binaria en caso de tener solo dos clases, las cuales suelen ser representadas con 1 y 0, aunque también hay casos donde se puede obtener un valor flotante, caso el cual necesitara definirse un umbral. Por otro lado, si tenemos más de dos etiquetas de clase, podemos definir que la clasificación que estamos realizando es multiclase (Patterson & Gibson, 2017). En la Figura 2.1 se observa de mejor manera el algoritmo de Clasificación.

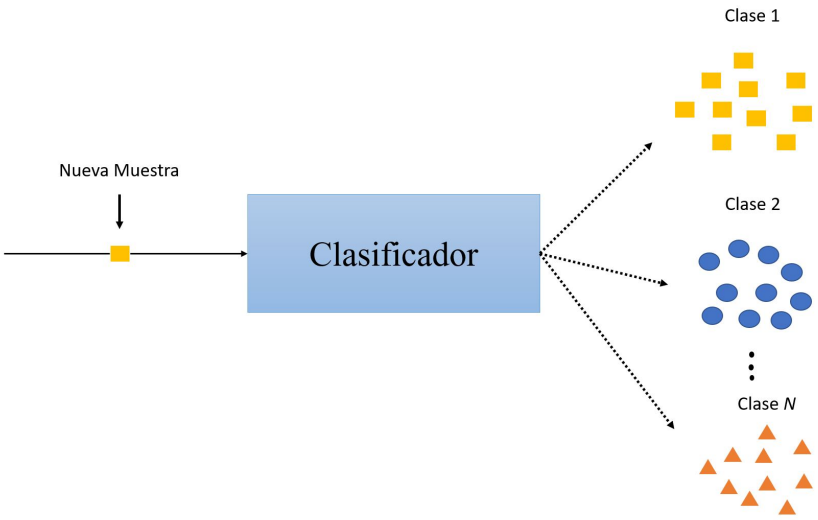

Figura 2.1: Proceso de clasificación

2.3.1.2. Regresión

La regresión es un proceso estadístico predictivo en el que un modelo intenta predecir valores continuos (i.e. Ventas, precios, reseñas) a través de relaciones entre variables dependientes e independientes. Es decir, encontrar una ecuación que sustituya los valores de las variables, obteniendo así los valores a predecir.

El algoritmo más utilizado para varios problemas es la regresión lineal simple. Esta es una regresión en la que se forma una línea recta cuando se satisface la ecuación. El método de los mínimos cuadrados se utiliza para obtener tales ecuaciones.

Dado un conjunto de puntos, el algoritmo de regresión construye un modelo que permite ajustar la relación de dependencia entre un objeto independiente particular (un valor de variable independiente x) y el valor "resultado" respectivamente (un valor de variable independiente y). Observe la Figura 2.2 para tener una idea más clara de esto.

Se establece una línea como referencia, basado en la relación establecida por el cálculo realizado entre la distancia de la recta con respecto a los puntos de datos correspondientes a los valores (x, y). Esta distancia, las líneas verticales, son el "residual" o el "error". El algoritmo de regresión recalculará (y moverá) la línea en cada iteración, encontrando la línea que mejor coincida con los puntos de datos (x, y), o en otras palabras, la línea con el mínimo error (más cercano a la mayoría de puntos).

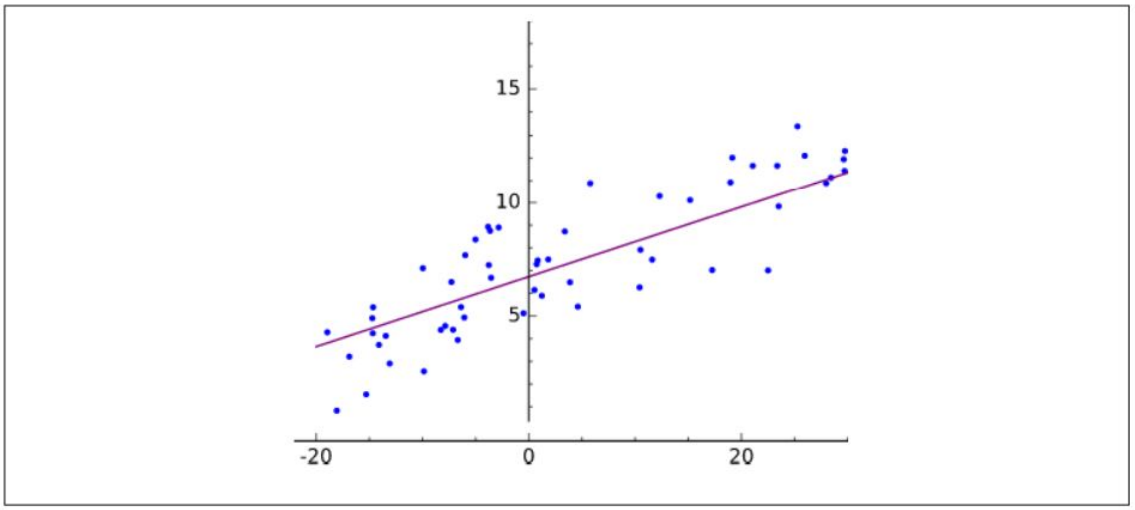

Figura 2.2: Modelo de regresión Lineal. Fuente Patterson & Gibson, 2017

2.3.2. Redes neuronales

Las redes neuronales son representaciones computacionales basadas en el sistema nervioso. Las redes neuronales tienen como su unidad básica las neuronas, las cuales se suelen organizar en capas, como se muestra en la Figura 2.3.

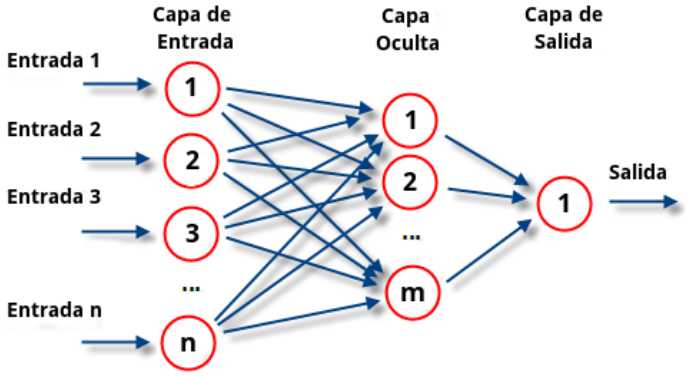

Figura 2.3: Redes_neuronales_esquema

Una red neuronal es un modelo simple que simula cómo el cerebro humano procesa la información: Funciona combinando simultáneamente una gran cantidad de unidades de procesamiento interconectadas que parecen sesiones de una versión abstracta de una neurona.

La red aprende relacionando los registros individuales, generando predicciones para cada registro y ajustando los pesos cuando hace predicciones incorrectas. Este proceso se repite varias veces y la red está mejorando continuamente sus predicciones hasta que se alcanza uno o más criterios de parada.

Una red neuronal es un modelo de computadora que comparte algunas propiedades con el cerebro, en donde muchas unidades simples trabajan en paralelo con una unidad de control no centralizada. Donde los pesos entre las unidades son los medios principales de almacenamiento de información a largo plazo en las redes neuronales. La actualización de esos pesos son la forma principal de que la red neuronal aprende nueva información (Patterson & Gibson, 2017).

2.4. Aprendizaje profundo

La inteligencia artificial es un campo muy amplio, el cual existe desde hace tiempo. Este campo engloba el área del aprendizaje máquina, que a su vez engloba el campo del aprendizaje profundo. En otras palabras, el aprendizaje profundo es un subcampo del aprendizaje máquina.

El concepto del aprendizaje profundo ha ido cambiando en la década pasada, y es un término que sigue a discusión. Sin embargo, una definición que puede servir es que el aprendizaje profundo trata con ”Redes neuronales con más de dos capas” (Patterson & Gibson, 2017). Aunque no es un concepto propiamente dicho, esta definición nos dice que el aprendizaje profundo engloba técnicas de aprendizaje máquina con un nivel de complejidad alto.

Los aspectos que diferencian una red de aprendizaje profundo con redes multicapa son las siguientes:

- Más neuronas que las anteriores redes.
- Formas más complejas de conectar las capas.
- Una explosión en la cantidad de poder de cómputo disponible para entrenar.
- Extracción de características automática.

El concepto de más neuronas, quiere decir que la cantidad de neuronas utilizadas aumento con respecto a años pasados. En cambio, el concepto de más conexiones significa que las redes cuentan con más parámetros a optimizar, y esto requiere una explosión en el poder computacional necesario. De esta forma, las redes profundas pueden modelar espacios del problema más complejos. En otras palabras, el aprendizaje profundo son redes neuronales con un gran número de parámetros y capas en alguna de las cuatro arquitecturas de redes fundamentales:

- Redes pre-entrenadas no supervisadas.
- Redes neuronales convolucionales.

- Redes neuronales recurrentes.
- Redes neuronales recursivas.

2.4.1. Hiperparámetros

Los hiperparámetros son típicamente un conjunto de parámetros que pueden ser ilimitados. El usuario teóricamente puede escoger un número entre 1 e infinito, pero no puede simplemente utilizar los datos de entrenamiento para encontrar el número correcto de nodos. Por lo que, la definición de los límites de los hiperparámetros es especificada por el experto, donde los límites son creados con base en múltiples restricciones, tales como: requisitos de cálculo, dimensionalidad, tamaño de los datos, etc. Típicamente, sería necesario crear los límites para más de un hiperparámetro, por lo que, se obtendría una cuadrícula n-dimensional de los hiperparámetros para elegir (Joshi, 2020).

En el aprendizaje máquina existen los parámetros de modelo y parámetros que se ajustan para hacer que las redes se entrenen mejor y más rápido. A estos parámetros de ajuste se les denominan hiperparámetros, y su función principal consiste en controlar las funciones de optimización y selección de modelos durante el entrenamiento del algoritmo de aprendizaje. La selección de hiperparámetros se enfoca en asegurar que el modelo no tenga un subajuste o sobreajuste el conjunto de datos, mientras que aprende la estructura de los datos. Aunque los hiperparámetros son diferentes para cada tipo de modelo de aprendizaje, los que comúnmente se encuentran en las redes neuronales son los siguientes (Patterson & Gibson, 2017):

- Número de capas
- Número de neuronas por capa
- Funciones de activación
- Funciones de perdida
- Función de optimización
- Épocas de entrenamiento
- Inicializador de pesos.

- Tamaño de lotes.
- Tasa de aprendizaje
- Regularización
- Impulso
- Escasez

2.4.2. Tasa de aprendizaje

El rol de la tasa de aprendizaje consiste en moderar el grado en el cual los pesos del modelo son cambiados. Usualmente, se establece un valor bastante pequeño (por ejemplo, 0.1) y comúnmente se configura para decaer a medida que el número de iteraciones de ajustes de pesos aumenta (Mitchell et al., 1997).

La tasa de aprendizaje afecta la cantidad en la que se ajustan los parámetros durante la optimización, esto para minimizar el error en las conjeturas de la red neuronal. Un coeficiente de tasa de aprendizaje grande (por ejemplo, 1) hará que los parámetros den saltos, y los coeficientes pequeños (por ejemplo, 0.0001) hará que los parámetros avancen lentamente. Si los saltos son grandes, aunque ahorraran tiempo inicialmente, pueden llegar a ser desastrosos si sobrepasa el mínimo establecido. En cambio, una tasa de aprendizaje pequeña eventualmente llevará al modelo a un error mínimo, pero puede llegar a tomar una mayor cantidad de tiempo (Patterson & Gibson, 2017).

2.4.3. Optimizador

Los algoritmos de aprendizaje profundo implican optimización en muchos contextos. La optimización se usa a menudo para escribir pruebas o diseñar algoritmos. El problema de optimización más difícil en el aprendizaje profundo es el entrenamiento de redes neuronales. Un caso en particular de optimización de una red neuronal es el siguiente: Encontrar los parámetros θ de una red neuronal que significativamente reduzca el costo de la función $j(\theta)$, que comúnmente incluye una medida de rendimiento evaluada en todo el conjunto de datos (Goodfellow et al., 2016).

Otra forma de ver el aprendizaje automático es como un problema de optimización, donde buscamos reducir la función de pérdida en relación con los parámetros de la función de predicción. El algoritmo *Stochastic Gradient Descent* (SGD) y sus variantes son uno de los algoritmos más utilizados para mejorar los modelos de aprendizaje automático. Sus puntos fuertes son la facilidad de implementación y el procesamiento rápido de grandes conjuntos de datos. (Patterson & Gibson, 2017).

Otro de sus puntos es que el SGD puede ser ajustado mediante la adaptación de la tasa de aprendizaje o utilizando información de segundo orden. La versión original del SGD utiliza el gradiente de manera directa, esto deriva en que el gradiente puede ser muy cercano a cero para cualquier parámetro, así el SGD toma pasos muy pequeños o grandes en algunos casos. Para solucionar esto existen los algoritmos adaptativos, estos algoritmos son variantes del SGD y pueden adaptar una tasa de aprendizaje diferente a cada parámetro. Alguno de los algoritmos adaptativos más populares son:

- AdaGrad
- RMSProp
- Adam
- AdaDelta

Cada uno de estos algoritmos presentan sus ventajas y desventajas. Ninguno de estos es mejor que los demás en todos los casos. El uso de cada uno de ellos depende en gran medida de las necesidades del modelo y las características de los datos que se manejan para el entrenamiento.

2.5. Auto Aprendizaje Máquina (AutoML)

En años recientes, las técnicas de aprendizaje profundo han invadido todos los aspectos de la vida de la sociedad y han traído una gran conveniencia para la misma. Sin embargo, la creación de sistemas de aprendizaje profundo de alta calidad para una tarea específica depende mucho de la experiencia del humano, lo anterior hace que la aplicación del aprendizaje

profundo se vea limitado a pocas áreas. Aquí es donde entra el Auto Aprendizaje Máquina (AutoML por sus siglas en inglés), esta área se vuelve una solución prometedora para la construcción de sistemas de aprendizaje profundo sin la necesidad de asistencia humana (He et al., 2021).

El interés reciente por modelos de aprendizaje máquina complejos y computacionalmente caros con una gran cantidad de hiperparámetros, como marcos de trabajo de AutoML y redes neuronales profundas, ha resultado en el resurgir de la investigación de Optimización de Hiperparámetros (HPO por sus siglas en inglés) (Hutter et al., 2019).

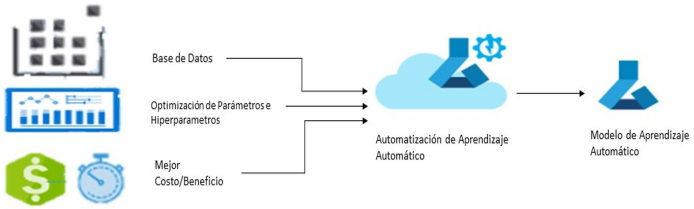

Figura 2.4: Modelo de AutoML

Un sistema de aprendizaje máquina contiene hiperparámetros, estos son modificados de manera automática para optimizar un modelo de aprendizaje máquina. Los modelos de redes neuronales profundas dependen crucialmente de un rango de opciones de hiperparámetros, entre estos se encuentran: Arquitectura de la red neuronal, regularización, y optimización.

2.5.1. Optimización de hiperparámetros

La optimización de hiperparámetros (HPO por sus siglas en inglés) es uno de las principales acciones del AutoML. Este problema nace en los años 90, donde Kohavi & John encontraron y establecieron que diferentes configuraciones de hiperparámetros tienden a trabajar de mejor manera para diferentes conjuntos de datos (Kohavi & John, 1995). Por lo tanto, en la optimización se busca obtener los mejores hiperparámetros posibles

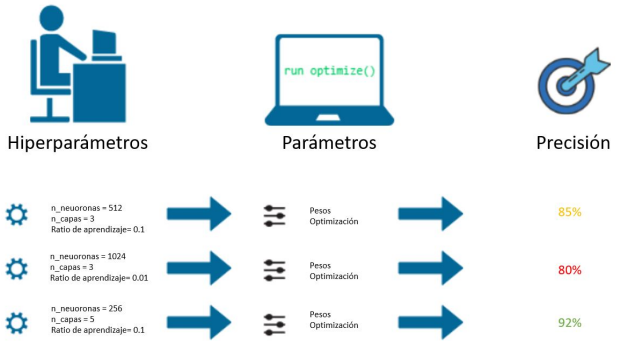

Figura 2.5: Optimizador de hiperparámetros

2.5.2. Búsqueda de un modelo de arquitectura neuronal

NAS es un proceso de ingeniería de arquitectura automática, que se considera un subcampo del AutoML y contiene una superposición significativa con la HPO. Los métodos de NAS son categorizados de acuerdo con tres dimensiones, las cuales son: Espacio de búsqueda, Estrategia de búsqueda, y estrategia de estimación de rendimiento (Hutter et al., 2019). La Figura 2.6 muestra el proceso de búsqueda NAS.

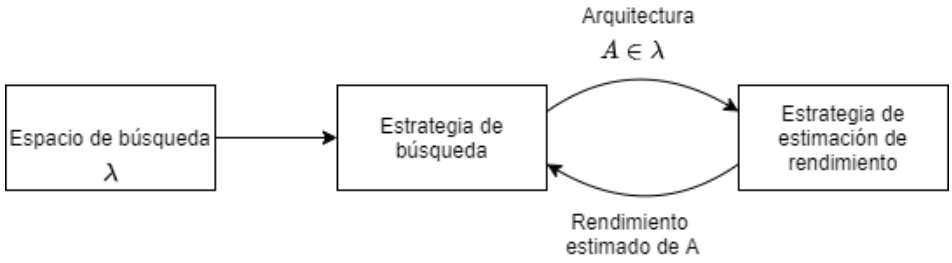

Figura 2.6: Proceso de búsqueda del método NAS. Fuente Hutter et al., 2019

2.5.3. Optimización bayesiana

El método de optimización bayesiana es una propuesta de marco de trabajo de optimización para la optimización global de funciones de caja negra costosas. La optimización bayesiana se caracteriza por ser un algoritmo iterativo que consta de dos componentes principales: Un modelo sustituto probabilístico y una función de adquisición para decidir cuál será el siguiente punto para evaluar. Durante cada iteración, el modelo sustituto es ajustado a

todas las observaciones de la función objetivo que se han realizado hasta el momento. Posteriormente, la función de adquisición determina la utilidad de los diferentes puntos candidatos, utilizando la distribución predictiva del modelo probabilístico, mediante un intercambio entre la exploración y la explotación (Hutter et al., 2019).

2.6. Computación distribuida

Durante la realización de este proyecto se implementó el uso de tecnologías distribuidas, esto debido a que es necesario la implementación de múltiples nodos que funcionen en conjunto, que sean flexibles y sencillos en su desarrollo.

La computación distribuida es un modelo para resolver grandes problemas computacionales utilizando una gran cantidad de computadoras organizadas en grupos integrados en una infraestructura de telecomunicaciones distribuida Coulouris et al., 2012. La ventaja de la computación distribuida es que la operación de procesamiento se puede dividir en distintas ubicaciones donde se permita realizar de manera más eficiente. En la Figura 2.7 se observa el funcionamiento de computación distribuida.

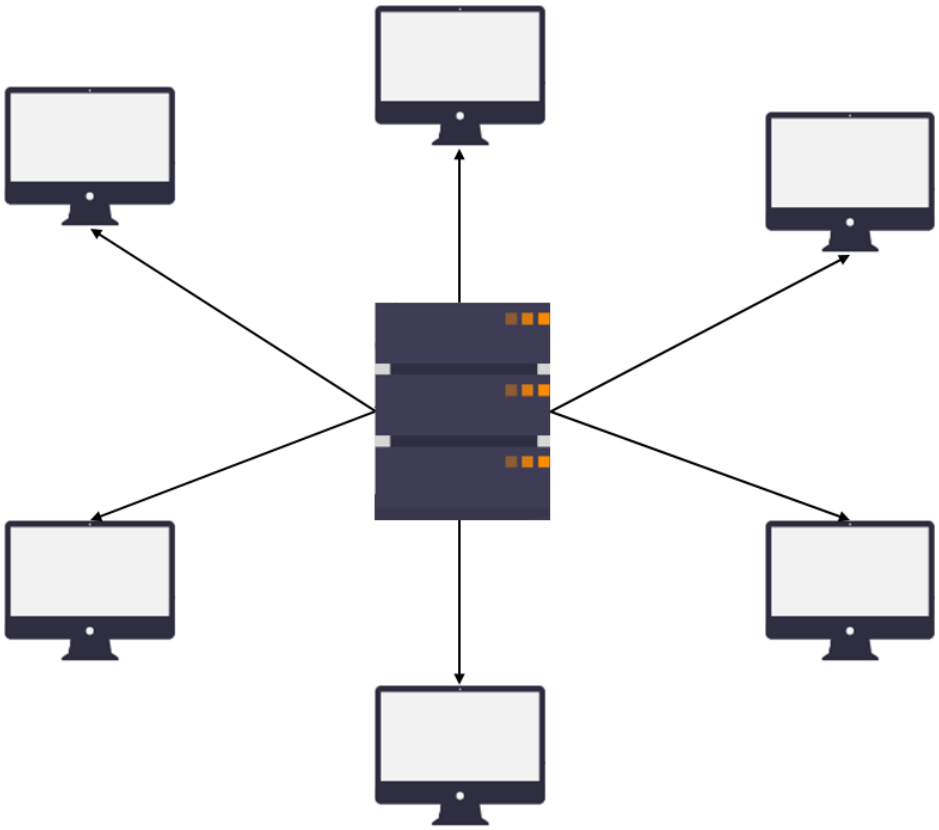

Figura 2.7: Computación distribuida

2.6.1. Historia de la computación distribuida

Los primeros sistemas distribuidos fueron diseñados en los años 70 mediante el uso de redes locales (LAN) como Ethernet (Andrews, 2020). Esta innovación permitió el desarrollo de múltiples lenguajes, algoritmos y aplicaciones para la distribución de procesos. No obstante, hasta que los precios de las redes LAN bajaran, se logró el desarrollo de la computación cliente-servidor (Banks, 2008). Las últimas décadas se han concretado muchos avances en el desarrollo de algoritmos distribuidos, elevando cada vez más la importancia de la computación distribuida. Existen múltiples ejemplos de sistemas distribuidos, los cuales son aplicados en el día a día en la sociedad. Comúnmente se han aplicado que la mayoría de los sistemas están estructurados con un modelo cliente-servidor, en donde el servidor es el encargado de almacenar los datos y manejar los recursos, proporcionando un servicio a varios clientes que

se encuentran en distintas ubicaciones (Ghosh, 2006). Sin embargo, algunas aplicaciones no dependen de un servidor central, sino que son sistemas peer-to-peer los cuales han presentado un aumento en su popularidad en tiempos modernos (Liben-Nowell et al., 2002).

2.6.2. Sistemas distribuidos

Los sistemas destinados están destinados a ser utilizados en un entorno del mundo real, por lo que deben estar diseñados para funcionar correctamente en la gama más amplia posible de circunstancias ante posibles dificultades y amenazas. Las propiedades y los problemas de diseño de sistemas distribuidos pueden ser capturados y discutidos mediante el uso de modelos descriptivos. Cada modelo tiene la intención de proporcionar una descripción abstracta y simplificada pero consistente de un aspecto relevante del diseño del sistema distribuido.

Algunos aspectos relevantes pueden ser: el tipo de nodo y de red, el número de nodos y la responsabilidad de estos y posibles fallos tanto en comunicación como entre los nodos. Se puede definir tantos modelos como características queramos considerar en el sistema, pero suele atender a esta clasificación: modelos físicos, modelos arquitectónicos, modelos de interacción, modelos de fallos, modelos fundamentales y modelos de seguridad,

- Modelos físicos: Representan la forma más explícita para describir un sistema, identifican la composición física del sistema en términos computacionales. Principalmente atendiendo la heterogeneidad y escalabilidad.

- Modelo arquitectónico: Describen el sistema en términos de las tareas computacionales y de comunicación realizada por los elementos. La principal preocupación es determinar la relación entre procesos y hacer el sistema confiable, adaptable y rentable.

- Modelos fundamentales: Toman una perspectiva abstracta de acuerdo con el análisis de un aspecto individual de un sistema distribuido. Se debe obtener solo lo esencial del sistema para poder comprender el comportamiento del sistema.

▷

2.6.2.1. Modelos de interacción

Analizan la estructura y secuencia de la comunicación entre los elementos del sistema. Cobran importancia las prestaciones del canal de comunicación (Latencia, AB, fluctuación), haciendo imposible predecir el retraso con el que puede llegar el mensaje. En otras palabras, no hay un tiempo global a todo el sistema, la ejecución es "No determinista". Cada computadora tiene su propio reloj interno, lo que conlleva tener que sincronizar los relojes locales de todas las máquinas que componen el sistema distribuido. Existen dos tipos de modelos de interacción síncrono, asíncrono.

- Modelo síncrono: Este modelo se enfoca en establecer una comunicación en tiempo real entre los dos dispositivos.
- Modelo asíncrono: Es un modelo que permite establecer una comunicación en la que no es necesario obtener una respuesta inmediata a una petición.

2.6.2.2. Modelo de fallos

Estudio e identificación de las posibles causas de fallos. Pueden clasificarse según su entidad, dando lugar a fallo de procesos o fallos de comunicación, o según el problema, dando lugar a fallos por omisión o arbitrarios: fallos por omisión en procesos, fallos arbitrarios o bizantinos, enmascaramiento de fallos.

2.6.2.3. Modelos de seguridad

La seguridad de un sistema distribuido se puede lograr asegurando los procesos y canales utilizados para sus interacciones y protegiendo los objetos que es necesario encapsular contra el acceso no autorizado. Estos modelos proporcionan la base para construir un sistema seguro atendiendo a recursos de todo tipo. Para ellos, es clave postular un enemigo que es capaz de enviar cualquier mensaje a cualquier proceso y leer o copiar cualquier mensaje enviado entre un par de procesos.

2.6.3. Cliente-Servidor

Es un modelo arquitectónico el cual permite dividir las tareas que está realizando un software entre distintos clientes, llamamos servidor al encargado de administrar y gestionar los recursos entre los llamados clientes, los cuales se encarga en realizar las peticiones del servidor y retornar una respuesta, el servidor puede comportarse como cliente también si este response a otro servidor (Blancarte, 2020). En la Figura 2.8 se muestra un ejemplo de cómo se aplica en este modelo.

- Procesos: Son las tareas que realizar por los clientes que fueron ordenados por el servidor.
- Almacenamiento de la información: Es el lugar donde se almacenan los datos con los que se está trabajando
- Comunicaciones: Es el proceso de comunicación que existe entre los distintos nodos.

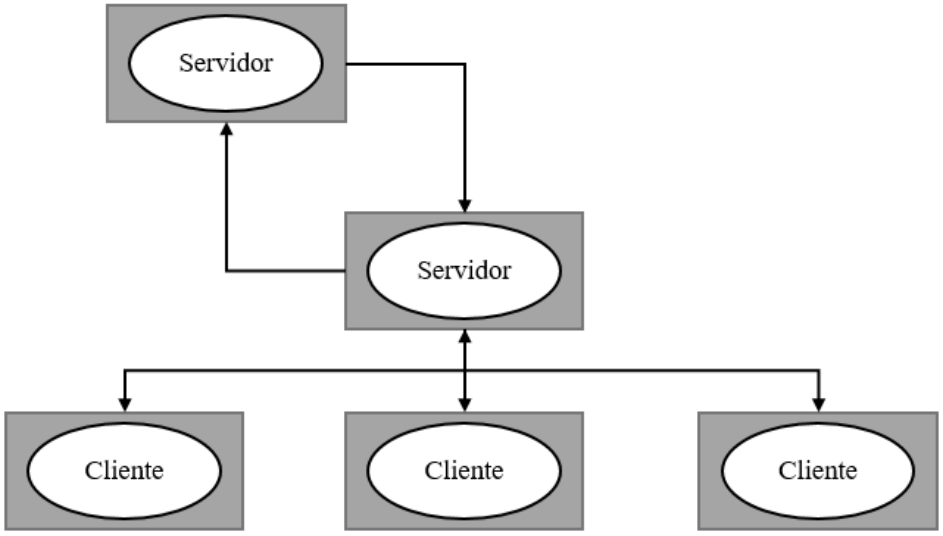

Figura 2.8: Modelo cliente-servidor

2.6.4. Redes peer-to-peer

También se les conocen como sistemas de igual a igual, debido a que todos sus nodos están conectados entre sí, como se observa en la Figura 2.9. Es un modelo totalmente descentralizado, el cual se organiza de manera dinámica, permitiendo un mejor equilibrio entre las cargas de trabajo asignadas a cada computadora del sistema.

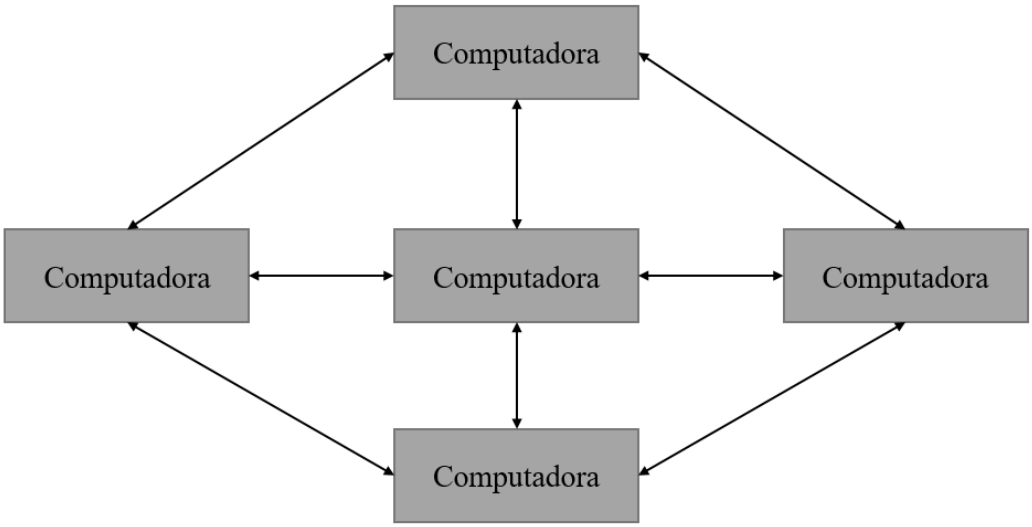

Figura 2.9: Modelo de Peer-to-Peer

2.6.5. Grid computing

Se trata de un modelo el cual permite que se distribuyan los procesos computacionales en una red de computadoras de tamaño variable, permitiendo la programación paralela. Permite tener una conexión a una supercomputadora, la cual realizara cálculos complejos, mientras que otra de sus conexiones realiza trabajos más sencillos como manejo de datos, esto distribuido en múltiples ubicaciones así como se muestra en la Figura 2.10. Tiene como objetivo resolver problemas computaciones con la mayor velocidad posible al menor costo posible utilizando todos los recursos computacionales disponibles.

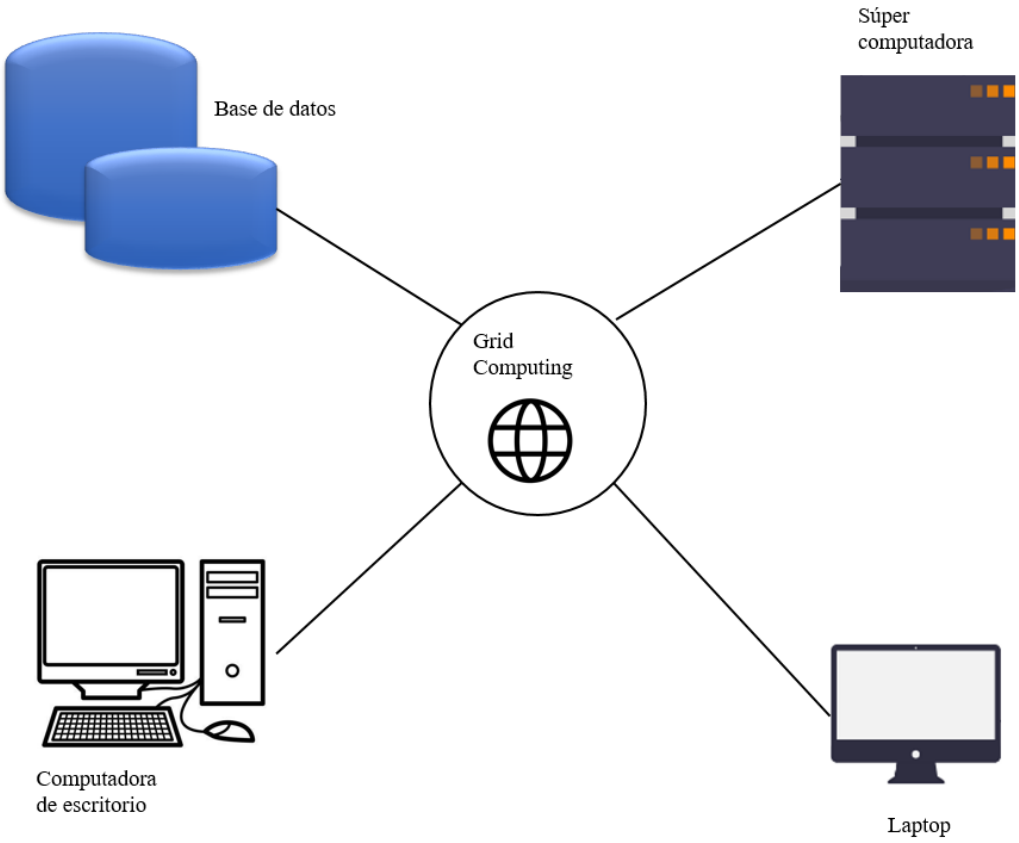

Figura 2.10: Modelo de Grid Computing

2.6.6. Red centralizada

Las redes centralizadas son sistemas que utilizan una arquitectura/cliente-servidor donde los nodos clientes están conectados directamente al servidor. La popularidad de estos sistemas reside en que permite a las empresas tener múltiples clientes, lo cuales recibirán un resultado por parte de la empresa. En la Figura 2.11 se muestran los distintos tipos de redes en los que se puede configurar un sistema distribuido.

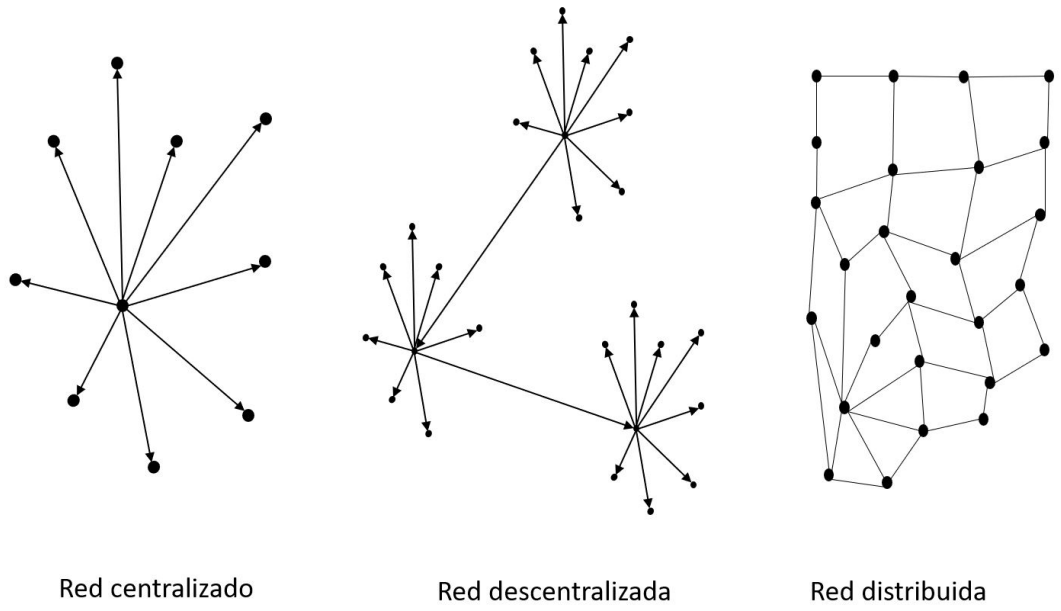

Figura 2.11: Topología de red centralizada

2.6.7. Procesamiento por lotes

El procesamiento por lotes está diseñado para que los procesos por lotes solo se ejecuten durante un período breve de tiempo. Entre los ejemplos de este proceso se encuentran: generación de agregación de datos de telemetría de usuario, análisis de datos de ventas para informes diarios o semanales, o la trans-codificación de archivos de vídeo. Los procesos por lotes generalmente son utilizados porque permiten el procesamiento de grandes volúmenes de datos rápidamente, mediante la utilización del paralelismo para acelerar el procesamiento (Burns, 2018).

El modelo más simple de procesamiento por lotes es por medio de una cola de trabajo. En un sistema de cola de trabajos, existe un lote de trabajo por realizar. Cada trabajo es independiente de otro trabajo y puede ser procesado sin interacciones entre ellos. Generalmente, el objetivo de las colas de trabajo son garantizar que cada trabajo se procese dentro de un cierto período de tiempo. La Figura 2.12 muestra un modelo de cola de trabajo genérica.

Figura 2.12: Modelo de cola de trabajo genérica

2.6.8. Intercambio de mensajes

La comunicación entre una red de computadoras puede realizarse de diferentes maneras en un sistema distribuido. Esta comunicación tiene una serie de características de rendimiento que se relacionan con la latencia, el ancho de banda y la fluctuación (Coulouris et al., 2012):

- **Latencia:** Es la demora entre el inicio de la transmisión del lenguaje de un proceso y el comienzo de su receptor por otro.
- **Ancho de banda:** Es el monto total de información que puede ser transmitida sobre una red de computadoras en un tiempo dado. Cuando se utiliza un gran número de canales de comunicación en la misma red, estos tienen que compartir el ancho de banda disponible.
- **Fluctuación:** Es la variación en el tiempo tomado para entregar una serie de mensajes. La fluctuación es relevante en cuestiones de datos multimedia.

Una manera de realizar la implementación de intercambio de mensajes es mediante un intermediario de mensajes. El Protocolo Avanzado de Cola de Mensajes (AMQP por sus siglas en inglés) es un protocolo Máquina a Máquina (M2M) ligero, el cual está diseñado para ofrecer confiabilidad, seguridad, aprovisionamiento e interoperabilidad. Además, el protocolo AMQP soporta arquitecturas de petición/respuesta y publicación/subscriptor (Naik, 2017, October).

Para iniciar una comunicación AMQP, este requiere el publicador o el consumidor creen un intercambio (*exchange* en inglés) con un nombre para después mandar una retransmisión

(*broadcast* en Inglés) con ese nombre. De esta manera, el publicador y el consumidor puede descubrirse unos a otros. Posteriormente, el consumidor se encarga de crear una cola (*queue* en inglés) y esta es adjuntada al intercambio al mismo tiempo. Finalmente, el publicador manda mensajes al *exchange*, y este se encarga de rutear a la cola correspondiente, a este proceso se le denomina *binding*. El protocolo AMQP realiza el intercambio de mensajes de diferentes maneras: directa, en forma de abanico, por tema o basado en encabezados.

2.7. Tecnologías de Machine Learning

El proyecto principal está desarrollado en lenguaje Python, mediante la aplicación de librerías como TensorFlow 2.2, para la implementación de modelos de redes neuronales, Keras, para la construcción de arquitecturas de redes neuronales, y Optuna, para la implementación de la optimización bayesiana. La gestión de librerías por parte de Python se realizó con un plugin del mismo Python. A su vez, el intercambio de mensajes entre nodos se realiza mediante la aplicación RabbitMQ.

2.7.1. Python

Es un lenguaje que se caracteriza por ser sencillo y práctico a la hora de desarrollar en la plataforma. Es un lenguaje de propósito general, y ofrece un ambiente cómodo para aplicaciones científicas, lo cual lo hace una buena opción para los científicos (Oliphant, 2007). Entre las características que ofrece Python se encuentran:

- Licencia de código abierto que permite la venta, utilización o distribución de aplicaciones escritas en Python.
- Permite la disponibilidad de ejecutar aplicaciones en múltiples plataformas.
- Permite la construcción de aplicaciones sofisticadas de manera orientada a objetos.
- Tiene la habilidad de interactuar con una gran variedad de otros softwares.
- Contiene una gran cantidad de módulos de librerías, esto significa que se puede construir programas más sofisticados.

- Tiene una comunidad famosa para recibir respuestas rápidas y útiles a las consultas de usuarios.

2.7.2. TensorFlow

TensorFlow Es una interfaz creada para desarrollar y ejecutar algoritmos de aprendizaje máquina. Un programa que utilice TensorFlow puede ejecutar, con poco o ningún cambio, en una amplia variedad de sistemas, desde máquinas pequeñas, como móviles, hasta sistemas distribuidos a gran escala de cientos de máquinas y dispositivos computacionales como tarjetas GPU. Entre las aplicaciones que puede tener se incluyen el entrenamiento y la inferencia de algoritmos para modelos de redes neuronales profundas. La API de TensorFlow fue desarrollada en noviembre de 2015 por Google y fue lanzado como paquete de código abierto bajo la licencia de Apache 2.0 (Abadi et al., 2016).

2.7.3. Rabbit MQ

RabbitMQ es una *broker* de mensajes de código abierto que implementa un estándar AMQP. Fue lanzado en el año 2007, hasta que la última versión más estable apareció en 2015. Esta aplicación fue desarrollada en el lenguaje de programación Erlang y está basado en el framework de "Plataforma de telecomunicaciones abierta". De entre las características más importantes de RabbitMQ están: muchos servidores RabbitMQ de una red local pueden ser agrupadas juntas, formando un solo *bróker* lógico, permitiendo la implementación de características como el balance de carga y la tolerancia a fallos; Otra característica es el protocolo AMQP que utiliza para aceptar las conexiones entre las diferentes plataformas (Ionescu, 2015).

Capítulo 3

Estado del arte

Actualmente, el mundo de la tecnología está en constante desarrollo, lo que lleva a que las empresas necesiten equipos cada vez más especializados. Esto con el objetivo que se puedan realizar tareas complejas a una velocidad aceptable para los estándares de la sociedad de hoy en día. Otra problemática común es el manejo de grandes volúmenes de datos y el obtener un buen rendimiento generando bajos costos de producción. Por lo tanto, el interés por un sistema de AutoML como solución a esta problemática ha aumentado considerablemente en los últimos años.

3.1. Principales sistemas de AutoML

Dentro de los primeros sistemas de AutoML desarrollados se encuentran Auto-WEKA Thornton et al., 2013. Este sistema fue el primero en abordar el área de la automatización de procesos de aprendizaje como un campo de estudio formal *"Combined Algorith Selection and Hyperparameter optimization"* (CASH). Utilizaron la librería WEKA de Java la cual selecciona un algoritmo mientras que los hiperparámetros son dados por el *dataset*. Utilizan técnicas de optimización bayesiana basada en modelos secuenciales, esto les permite trabajar con hiperparámetros continuos y categóricos. De esta manera, cada iteración que hace calcula la función de perdida mediante un método de validación cruzada, el algoritmo seleccionaba los mejores hiperparámetros y actualizaba el modelo hasta la última iteración donde retornaba el mejor modelo. En el año 2017 se desarrolló Auto-WEKA 2.0 el cual cuenta con cambios que le representan una mejoría en comparación con su predecesor, se agregó soporte para

modelos de regresión y cambiaron el algoritmo de optimización por un árbol de decisión bayesiana obteniendo mejores resultados (Kotthoff et al., 2019).

Hyperopt-Sklearn es un sistema de AutoML desarrollado en 2014 inspirado en Auto-WEKA (Komer et al., 2014), el cual buscaba compartir sus funcionalidades con el lenguaje de programación Python. Se basa en el mismo concepto de configurar hiperparámetros como un espacio de búsqueda de variables aleatorias. En el caso de este trabajo, utiliza como base para los algoritmos la librería *scikit-learn* la cual se encarga de entrenar y validar los modelos basados en la función objetivo. Utiliza un método de validación cruzada mediante la implementación de una librería de optimización de hiperparametros *hyperopt* el cual presento una precisión aceptable con respecto a los anteriores modelos, incluso teniendo algunos conjuntos de datos donde presentaba mejor precisión que los modelos ya existentes. Una escalabilidad más fiable fue una de las ganancias que obtuvo al corregir el problema encontrado en Auto-WEKA.

El siguiente año se siguió trabajando en los conceptos ya existentes en los sistemas de AutoML, por lo que en 2015 salió a la luz AUTO-SKLEARN (Feurer et al., 2019). Este nuevo sistema resolvió alguno de los problemas que venía persistiendo en los primeros modelos de AutoML. Los cuales fueron causados principalmente por la forma en que se implementó el espacio de búsqueda con anterioridad. AUTO-SKLEARN implemento un algoritmo de optimización bayesiana, así como también expandió el número de algoritmos de búsqueda, lo que llevo a obtener resultados más precisos y suaves. Los pocos inconvenientes que se encuentran en el sistema es la carencia de soporte para algoritmos de regresión y tiene un rendimiento en grandes conjuntos de datos.

Año 2016 fue uno donde se introdujo una nueva ruta en el desarrollo de Sistemas de AutoML con TPOP (*Tree-based Pipelin Optimization Tool*) (Olson & Moore, 2016). El sistema basó su construcción en el uso de algoritmos genéticos, aunque su primer enfoque fue el de clasificación y precisión, logro obtener resultado significativo como herramienta de aprendizaje automático. El proceso de búsqueda que utiliza es una herramienta de optimización de tuberías basada en árboles, la cual cuenta con tres procesos principales: preprocesamiento de características, selección de características y clasificación supervisada de características, creando así árboles de algoritmos genéticos. Esto permitió tener una mejor escalabilidad y

flexibilidad debido a que se puede agregar o eliminar nodos en las tuberías con facilidad.

3.2. Segunda generación de sistemas de AutoML

Se presentaron otros trabajos con esta metodología basada en algoritmos genéticos como es el Caso de GAMA: *Genetic Automated Machine learnign Asistant* presentado en 2019 (Gijsbers & Vanschoren, 2019). Este sistema de AutoML fue desarrollado en el lenguaje de programación Python, uso algoritmos genéticos usando tuberías mediante librería *scikit-learn* para los algoritmos de inteligencia artificial, como principal característica se destaca que usa los algoritmos de manera asíncrona a diferencia de su antecesor TPOP, el cual hacia manejo de manera síncrona, también se debe mencionar que solo soporta algoritmos de regresión y clasificación.

En 2018 se presentó un trabajo llamado ML-Plan: *Automated machine learninig via hierarchical plannig (Mohr et al., 2018)*. Este cuenta con una característica principal que lo diferencia de los demás AutoML y es que su espacio de búsqueda no se basa en cambiar la solución durante la búsqueda. En general, cuenta con una jerarquización que le hace probar cada una de las soluciones de manera global hasta obtener el resultado deseado. El mismo año también se presentó TransmogrifIA (Kevin Moore, 2018). Una librería de AutoML la cual fue diseñada en escala, con la capacidad de ejecutarse sobre el FrameWork de Apache Spark, su desarrollo está pensado para que no se necesite un experto en aprendizaje máquina. Si no que de manera estricta maneja una separación entre el flujo de trabajo del modelo de machine learning y la manipulación de datos para que pueda ser reutilizable y modular. Otro trabajo a destacar el mismo año fue el de OBOE: *Collaborative Filtering for AutoML Initialization* donde maneja de los principales retos en el machine learning que es la selección de algoritmos y el ajuste de hiperparámetros (Yang et al., 2018). OBOE predice cuál algoritmo tendrá el mejor rendimiento en un tiempo limitado. El sistema predice el tiempo de ejecución de cada algoritmo, por lo que cuantifica la información que se obtendrá sobre un conjunto de datos. A diferencia de otros sistemas de AutoML donde dedican todo el tiempo en explorar cada una de las meta-características del modelo.

Hoy en día se observa como los modelos de aprendizaje automático se han estado desa-

rrollando de gran manera y con esto han presentado nuevas situaciones que requirieron comenzar a trabajar en estos sistemas de AutoML para dar un acceso más sencillo a las personas al mundo de la inteligencia artificial. Al tener estos sistemas que se basan principalmente en librerías ya establecidas para el uso de un único nodo, haciendo todo el proceso de manera centralizada. Luego se comenzaron a buscar maneras de hacer que fueran escalables, para así lidiar con la gran demanda de datos que se está dando en la actualidad.

Ultron-AutoML es una herramienta de aprendizaje automático implementada en un ambiente distribuido tolerante a fallas (Narayan et al., 2020). El cual presenta una mejora de rendimiento, la creación de modelos de aprendizaje automático mediante el uso de la optimización de hiperpárametros. Este *framework* permite un buen manejo del trabajo a realizar, los datos que se utilizan tienen un mejor control y también ha demostrado que la carga que recibe la CPU ha bajado su carga hasta un $30\ \%$ al ser dividida en distintos nodos. Al manejar una gran cantidad de nodos también permite asegurar una buena escalabilidad en el sistema, así como una sencilla duplicidad de los resultados obtenidos.

D-SmartML es un sistema que busca crear su propia base sobre el *framework* de distribución Apache-Spark el cual ayuda a que se solucione el problema de escalabilidad y el manejo de un gran volumen de datos (Abd Elrahman et al., 2020). D-smartML cuenta con un mecanismo meta aprendizaje que le ayuda automatizar la selección de algoritmos. Admite tres procesos distintos, los cuales son ajustes de hiperparametros, búsqueda aleatoria distribuida y la optimización de hiperbanda distribuida.

Las tecnologías que implementan sistemas de AutoML han aportado grandes soluciones a problemas, lo cual aporta la posible implementación de distintas funcionalidades en distintos campos que antes no se habían contemplado. Algunos son sistemas de atención al cliente, dispositivos inteligentes que se encargan de obtener grandes cantidades de datos y mientras que la herramienta de aprendizaje automático ayuda en el proceso de análisis de datos reduciendo los costos (Li et al., 2019).

En el área de la salud se han desarrollado también propuestas interesantes que manejan este tipo de herramientas. En el trabajo de (Alaa & Schaar, 2018) se implementa AutoPrognosis, que es un modelo que se diseñó especialmente en pronósticos clínicos, utilizando un conjunto eficiente de tuberías gracias a un moderno sistema de optimización bayesiana por

lotes. Debido a que los datos clínicos presentan una mayor complejidad, se buscan también opciones como Atseer (Wang et al., 2019) que brinda una mejor interfaz al usuario, brindando la capacidad de seleccionar mejor el campo de búsqueda en los sistemas de AutoML.

Katib es una plataforma escalable de AutoML el cual encuentra su base en Kubernetes, esto le permite tener una variedad de algoritmos, también realizar ajuste de hiperparámetros y búsqueda de arquitectura neuronal. Este sistema divide todos sus componentes y los encapsula en microservicios, opera desde la API de Kubernetes. Esto le permite tener una buena comunicación entre sus componentes, una gestión flexible y una implementación escalable a un costo reducido. Contiene una buena interfaz de usuario, haciéndolo una plataforma universal para cualquier tipo de usuario (Zhou et al., 2019).

3.3. Sistemas de AutoML comerciales

La popularidad de estos sistemas de AutoML en la sociedad actual ha llevado a empresas a desarrollar sus propios sistemas, los cuales son implementados como modelo de negocio, algunas de estas empresas son H20, Amazon, Google, Microsoft, Datarobot y SparkCognition.

La empresa H20.io cuenta con múltiples servicios relacionados con la inteligencia artificial, entre los cuales se destaca su propio sistema llamado H20 AutoML, el cual está construido sobre la base de su propio *Framework* H20 el cual le permite distribuir los procesos de manera centralizada entre múltiples nodos y utilizar un método de optimización *random forest* (LeDell & Poirier, 2020).

También hay empresas como Datarobot, la cual nos permite pagar por sus servicios de AutoML, cuenta con múltiples características, como la de realizar tareas de clasificación, regresión, series de tiempo, entre otros. También cuenta con una interfaz amigable para el usuario y características que te permiten tener un mejor manejo de los datos con los que se esté trabajando durante su utilización (*DataRobot's AI Cloud Platform,* 2012).

Otra empresa que se mencionó fue SparkCognition, la cual se encargó de desarrollar su propio sistema de AutoML llamado Darwin, el cual recientemente fue actualizado a su versión 2.0. Se cuenta con soporte para múltiples tareas como algoritmos supervisados, no

supervisados, series de tiempo, etc. La nueva versión Darwin 2.0 le brinda al usuario mayor control sobre los datos, la capacidad de brindar soluciones a problemas críticos, se agregaron algoritmos genéticos y realizar predicciones en tiempo real, todo mediante un sistema en la nube (*Darwin™ 2.0,* 2022).

Existen muchos modelos que trabajan en la nube, como es el caso de Cloud AutoML, el cual es un sistema creado por la compañía Google. La empresa cuenta con su propia plataforma, la cual le permite tener un mejor control de los usuarios, ya que es necesario que cada usuario tenga su cuenta propia. El objetivo de este sistema es facilitar la creación de modelos de inteligencia artificial, se puede contratar en función de las siguientes categorías, visión, video inteligente, procesamiento natural de lenguaje, traducción y datos estructurados (Bisong, 2019).

La empresa Amazon, por otro lado, cuenta con su propio servicio llamado Amazon Sagamaker el cual sirve como un AutoML permitiendo desarrollar, crear y entrenar sus propios modelos de manera sencilla. De la misma manera que Google implementan diferentes módulos tanto de visión artificial, procesamiento de lenguaje natural, regresión, entre otros con el objetivo de reducir costos operativos (Scheppat, 2022). Cuenta con una versión *open source* llamada AutoGluon, se ha utilizado como apoyo para la creación de modelos de inteligencia artificial menos complejos que en su versión de comercial, sin mencionar que tienes que contar con los conocimientos de manejo, ya que es parte de una librería de Python la cual se puede importar a diferencia de SageMaker el cual cuenta con su propia interfaz aparte de ser multiplataforma es más amigable para el usuario (Erickson et al., 2020).

Por último, tenemos la empresa Microsoft, el cual cuenta con su propio sistema de servicios basados en la nube como las anteriores empresas. El sistema que manejan se llama Azure Automated Machine learning, este sistema cuenta con varios apartados donde se puede seleccionar el tipo de modelo que se desea generar, sea aprendizaje supervisado, no supervisado, aprendizaje por refuerzo, entre otros. Al tener su manejo en a nube, cuenta con múltiples opciones de configuraciones, en las cuales podemos implementar el sistema de manera distribuida o completamente centralizada en un solo nodo (Mukunthu et al., 2019).

Tabla 3.1: Resumen del estado del arte (Trabajos de AutoML)

Autor	Sistema	Lenguaje	Optimización	Tipo de Sistema
Thornton et al., 2013	Auto-Weka	Java	Bayesiana	No distribuido
Komer et al., 2014	Hyperopt-Sklearn	Python	*Hyperopt*	No distribuido
Feurer et al., 2019	Auto-Sklearn	Python	Bayesiana+Meta-learn	No distribuido
Olson & Moore, 2016	TPOT	Python	Programacion Genetica	No distribuido
Narayan et al., 2020	Ultron-AutoML	Python	HPO Random, TPE, ReMAADE	Distribuido-Descentralizado
Abd Elrahman et al., 2020	D-SmartML	Python,	*Distributed Grid Search*, *Distributed Random Search*, *Distributed Hyperband*	Distribuido-Descentralizado
Alaa & Schaar, 2018	AutoPrognosis	Python	Bayesiana	Distribuido-Descentralizado
LeDell & Poirier, 2020	H_2O AutoML	H_2O	*Random Forest*	Distribuido-Centralizado

Los sistemas de AutoML han evolucionado de manera constante desde sus inicios, Auto-Weka siendo el primero en ser desarrollado bajo el lenguaje de programación java, abrió el camino para otras organizaciones comenzará el desarrollo de sus propios sistemas. A diferencia de Auto-Weka los sistemas de AutoML fueron implementados en su mayoría en el lenguaje de programación Python, tales como Hyperopt-Skleanr, Auto-Skleanr, TPOT, Ultron-AutoML, AutoPrognosis, D-SmartML. Se buscaron también múltiples sistemas de optimización, el más común de todos se puede determinar que fuel el algoritmo de optimización bayesiana. Inicialmente, los AutoML no estaban contemplados para que trabajaran de manera distribuida, por lo que solo los sistemas más modernos cuentan con un enfoque distribuido que les permitió descentralizar los procesos del AutoML. Se puede concluir que en su mayoría los sistemas están diseñados para ser *open source* aunque también existe algunas organizaciones que realizan estos sistemas con el objetivo de obtener ganancias, como Google, Amazon, H20.io, entre otros.

Capítulo 4

Metodología

En este capítulo se presentará la metodología propuesta para el diseño e implementación de este proyecto. Esta sección será dividida en varias partes en donde se explicara de manera más detallada cada una de las etapas de elaboración del trabajo. El trabajo cuenta con distintos componentes, los cuales puede considerarse de manera general como un conjunto de todo el trabajo que se realizó. En la Figura 4.1 se puede observar cada una de las distintas fases del funcionamiento del sistema, los cuales son: Inicialización del sistema, Nodo Maestro, Bróker RabbitMQ y Nodo Esclavo.

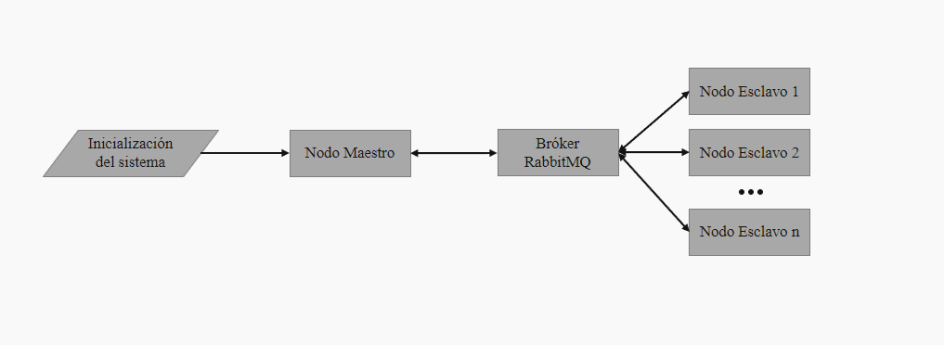

Figura 4.1: Diagrama general simplificado

Primero empieza la etapa de inicialización de datos, en esta etapa se enfoca principalmente en los parámetros iniciales que tendrá el sistema, también se seleccionó el conjunto de datos y el tipo de tarea que se va a realizar como uno de los puntos claves para que el sistema funcione. Continuando, se levantó un servidor de comunicación con el sistema de mensajería RabbitMQ.

Después la segunda etapa del sistema consiste en lo que el Nodo maestro realizara como

funciones principales del sistema, el Nodo Maestro se encargó de generar diccionarios en formato JavaScript Object Notation (JSON) en los cuales estará almacenado los parámetros del modelo a construir y entrenar por el Nodo Esclavo.

Después, con los resultados que se obtuvieron de los primeros modelos generados, se aplicara un algoritmo de optimización bayesiana, el cual permite mejorar los modelos que posteriormente se generen. También el Nodo Maestro se encarga de determinar el tipo de entrenamiento que realizaran los nodos esclavos. Así como también selecciona los mejores modelos generados para ingresarlos a un salón de la fama y finalmente determinar cuál es el modelo más eficaz.

En la etapa de comunicación mediante el *bróker* RabbitMQ se determina la gestión de mensajes entre los nodos esclavos y el Nodo Maestro, se debe generar dos colas llamadas "Parámetros" y "Resultados" las cuales serán usadas como canales de comunicación, distribuye de manera equitativa entre todos los nodos que estén escuchando las tareas que esté mandando el nodo maestro. Por último, el Nodo Maestro es el componente encargado de escuchar el canal de "Parámetros"donde recibirá los diccionarios en formato JSON con los modelos que tendrá que generar y entrenar, una vez termine el entrenamiento regresara los resultados mediante el gestor de mensajería RabbitMQ y serán entregados al Nodo Maestro.

4.1. Inicialización del sistema

En la primera etapa de inicialización del sistema consiste principalmente en los datos que están ingresando para inicializar el programa, el AutoML que se estará utilizando, inicia a partir de un conjunto de características los cuales determinan el comportamiento principal de los modelos generados, los parámetros que se utilizan son: el conjunto de datos, el tipo de conjunto que se estará utilizando, tipos de optimizadores, la función de activación, función de perdida, métricas de rendimiento, número de modelos generados en la fase de exploración y número de modelos seleccionados en el salón de la fama. En la Figura 4.2 se observa el proceso en el que los datos son ingresados al sistema.

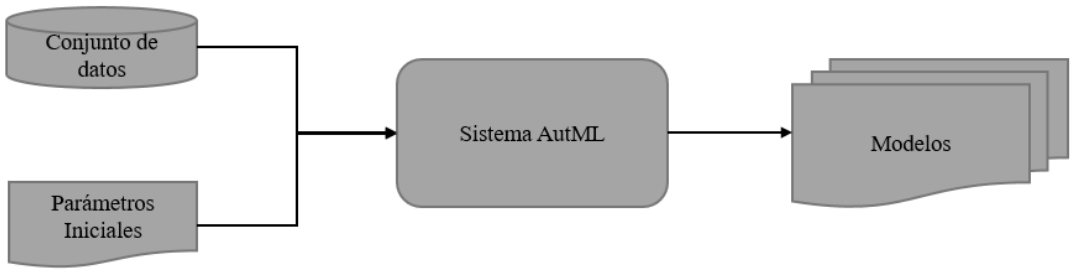

Figura 4.2: Diagrama representativo del ingreso de datos al sistema durante la inicialización del sistema.

La siguiente parte de esta etapa de inicialización consiste en levantar un servidor de comunicación, para eso se utilizó un sistema de gestión de mensajería, también conocido como bróker, mediante la tecnología de código abierto RabbitMQ. También se utilizó como complemento Docker, una herramienta que permite levantar el servidor mediante contenedores. Esto con el objetivo de mantener una conexión estable, manejo amigable de interfaz y una sencilla implementación. Primero se seleccionó el nombre del Host para tener un control del servicio, luego los puertos que se estarán utilizando como canales de comunicación entre el servidor y los clientes, también se agregaron los usuarios que se pueden comunicar con RabbitMQ, así como sus respectivos permisos de acceso dependiendo del nivel de usuario. Cada uno de estos parámetros fueron generados en un archivo Docker llamados *Dockerfile*, el cual al momento de querer montar la conexión se creara una imagen Docker con las características del *Dockerfile*. Por último, una vez creada la conexión al montar la imagen de Docker con las características que ingresamos, se establecerá una conexión con RabbitMQ generando un contenedor RabbitMQ, permitiendo gestionar desde su interfaz el proceso de comunicación siempre que el contenedor este activo. En la Figura 4.3 se observa el proceso de creación del servidor.

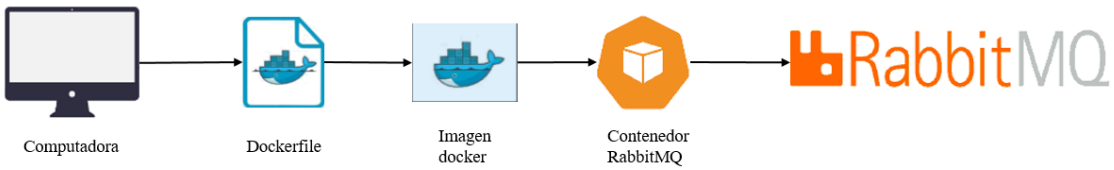

Figura 4.3: Proceso de levantar servidor RabbitMQ

4.2. Nodo Maestro

El Nodo Maestro es el encargado principal de la mayor parte de los procesos que realiza la aplicación durante la ejecución. Este nodo se encarga de definir los distintos parámetros necesarios para la creación de modelos, como la arquitectura de modelos y los parámetros de entrenamiento, también tiene como trabajo el coordinar el proceso de optimización. Una vez recibe los resultados obtenidos por los nodos esclavos, se encargará de ajustar los parámetros mediante un algoritmo de optimización bayesiana. También selecciona los mejores modelos en función al rendimiento que tuvieron durante la fase de exploración para así poder crear el salón de la fama. Una vez el Nodo Maestro haya creado el salón de la fama, pasa a realizar un entrenamiento profundo, una vez más manda estos modelos a través de RabbitMQ para que los distribuya entre los esclavos y estos sean entrenados. Una vez finalizado el entrenamiento selecciona el modelo que genero mejor rendimiento y es retornado al usuario como el mejor modelo, en la Figura 4.4 se puede observar mejor el funcionamiento interno del Nodo maestro.

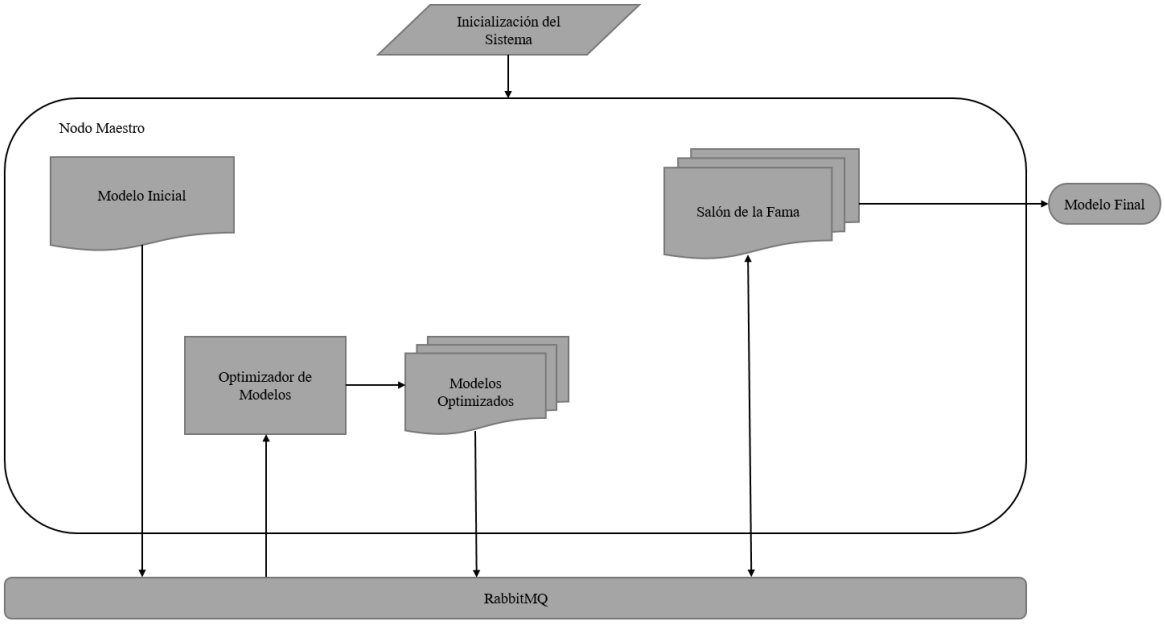

Figura 4.4: Diagrama de procesos en el nodo maestro

La primera tarea que se encarga de realizar el Nodo Maestro es la de generación del modelo inicial, el cual será implementado como diccionario en formato JSON, el número de modelos iniciales que se generara irán en función del número de nodos esclavos que se encuentren conectados escuchando al *bróker* de RabbitMQ. Normalmente, se solicita un modelo extra del total de nodos esclavos conectados con el propósito de que ningún nodo esclavo se encuentre sin realizar ninguna tarea en algún momento. Una vez se tienen los modelos generados en el formato correspondiente, pasa a realizar una petición de entrenamiento. Este proceso ocurre cuando el nodo maestro manda una solicitud a los nodos esclavos, a través de RabbitMQ, el *bróker* recibe los modelos generados, los modelos cuentan un ID de identificación para que se conozca en todo momento con cuál modelo se está trabajando. El modelo cuenta con ciertas características las cuales se muestran en la Tabla(4.1).

Tabla 4.1: Conjunto de características del modelo

Característica	Definición
ID	Numero único que se le otorga a cada modelo de manera que este pueda ser identificado en todo momento
Arquitectura	Conjunto de capas e hiperparametros generados a partir del espacio de búsqueda para la construcción del modelo.
Épocas	Numero de épocas que realizara el modelo, esto en función del tipo de entrenamiento
Earling Stopping	Numero de épocas limite que tiene el modelo para mejorar que utiliza para ver mejoría en el modelo
Espacio de Búsqueda	Define el espacio de búsqueda que sera utilizado
Hash de espacio de búsqueda	Es un valor que se genera por la función hash para asegurar un valor único que permite comprobar la compatibilidad entre el nodo y el espacio de búsqueda
Etiqueta del conjunto de datos	Etiqueta que se utiliza para identificar el conjunto de datos el cual tiene que cargar y que sera utilizado en el entrenamiento
Tipo de Entrenamiento	Es un valor booleano que utiliza para identificar si el entrenamiento que realizara es parcial o profundo.

Después de que el nodo trabajador retorna una respuesta, el nodo maestro maneja esa respuesta, interpreta los resultados y los envía al optimizador de modelos. El optimizador de modelos se basa en un algoritmo de optimización, el cual se encargará de generar nuevos modelos con base en los resultados obtenidos. Cuando se recibe una respuesta, el nodo maestro realiza el siguiente conjunto de acciones:

- General modelo: Es el proceso principal donde se realiza una llamada a la estrategia de búsqueda a través del algoritmo de optimización para generar un nuevo modelo.
- Esperar: Es un proceso que se basa en no generar modelos de búsqueda mientras espera que los nodos esclavos terminen su entrenamiento. En caso de llegar al modelo final por entrenar, este queda esperando hasta el último de los trabajadores.
- Comenzar una nueva fase: Proceso que se encarga de avisar que la fase exploración ha terminado y comienza la fase de entrenamiento profundo.
- Finalizar: Notifica al proceso principal que se ha completado cada uno de los pasos especificados con anterioridad, también se encarga de desplegar la información sobre

el mejor modelo generado, la arquitectura que utilizo, el tiempo que tardo en realizarse todo el proceso y los detalles del rendimiento del modelo.

4.2.1. Optimizador de modelos

El proceso de optimización se realiza mediante el uso de un *framework* llamado *optuna*. Se aplica un algoritmo llamado *bayesian optimzation based on kernel fitting* (TPE) en los modelos que genera el Nodo Maestro, esto con el fin de mejorar los hiperparámetros seleccionados. *Optuna* cuenta con 5 componentes principales para la optimización.

- *Study*: Este componente se basa en gestionar la información sobre la tarea de optimización, como el algoritmo a utilizar (*sampler*) y donde se almacenan los resultados de la prueba (almacenamiento).
- *Trial*: Componente que corresponde a cada prueba, donde la función objetiva muestra los parámetros que pasaron el objetivo de prueba e informa de los valores intermedios para realizar un filtro.
- *Storage*: Es el componente que almacena los resultados del trial, gracias a este componente permite realizar un filtrado persistente.
- *FrozenTrial*: Es una representación de cada trial en la capa de almacenamiento, guarda la función objetivo y los parámetros utilizados para evaluar cada trial.
- *Sampler*: Es un componente para implementar un algoritmo que se encargara de evaluar los siguientes parámetros seleccionados para obtener un mejor valor objetivo. El muestreo de hiperparámetros con estrategia de optimización o evolución bayesiana se define en el componente *Sampler*.

El Nodo Maestro durante ese proceso se encarga de tomar los valores iniciales para generar el primer modelo. Una vez que los primeros *n* modelos sean retornados desde el *bróker* RabbitMQ, el proceso de optimización mediante el uso de la librería *optuna* toma esos valores como el nuevo objetivo. Dentro del algoritmo de optimización, en cada uno de los *Trial*

que pasa van mejorando los resultados obtenidos con base en lo ya se registró con anterioridad. En la Figura 4.5 se puede observar un comportamiento de la taza de aprendizaje que va realizando *optuna* cada que busca optimizar el resultado.

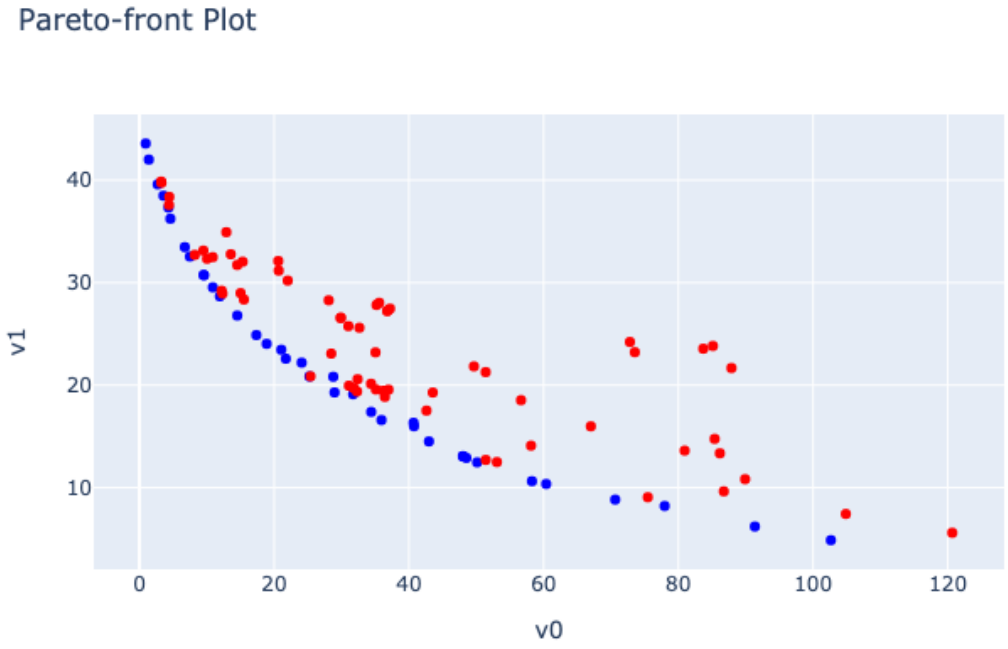

Figura 4.5: Gráfica de aprendizaje del algoritmo de optimización

4.2.2. Salón de la fama

A continuación, se hablará sobre el proceso de selección del salón de la fama, anteriormente se mencionó que durante cada iteración el Nodo Maestro aplicaba un proceso de optimización que buscaba mejorar los modelos generados, pero esto no asegura que siempre el siguiente modelo será mejor que el anterior. Al inicio, cuando se ingresan los datos iniciales se seleccionó el tamaño que tendrá el salón de la fama, en ese lugar se guardaran los n mejores modelos que allá generado el sistema a lo largo de la fase de exploración. Cada nuevo modelo generado durante la fase de exploración es comparado con los rendimientos resultantes del salón de la fama. En caso de que el nuevo modelo generado sea mejor que alguno de los modelos ya existentes en el salón de la fama, el de menor rendimiento es sustituido por el nuevo. En la Figura 4.6 se muestra de manera gráfica el proceso de selección de modelos.

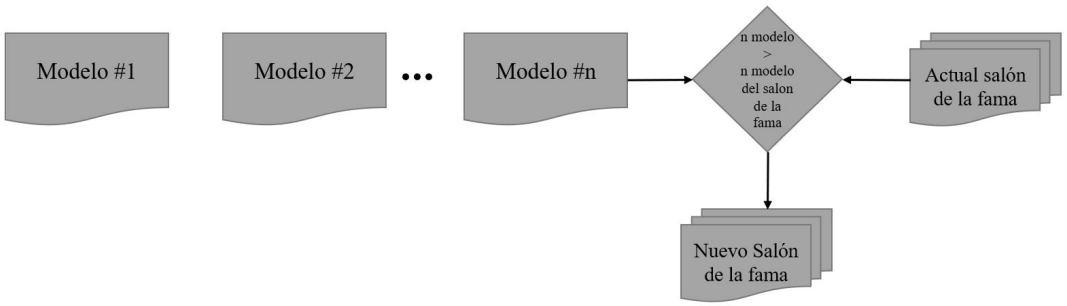

Figura 4.6: Diagrama de selección de modelo en el salón de la fama

Una vez terminada la fase de exploración, el salón de la fama estará completo con los *n* mejores modelos, por lo que el siguiente paso consiste en realizar un entrenamiento más exhaustivo para ver cuál de los modelos en el salón de la fama será el modelo final. Para esto se inició una nueva fase llamada entrenamiento profundo, en la cual se enviaran los modelos atreves del *bróker* RabbitMQ hacia los *n* número de nodos esclavos que estén escuchando en el momento. Esta vez el número de épocas por entrenamiento será ampliado con base en el número que fue ingresado durante la etapa de inicialización del sistema. Finalmente, una vez se terminen de entrenar los modelos del salón de la fama una vez más, el nodo maestro se encarga de seleccionar cuál tuvo el mejor rendimiento, se lo muestra al usuario y lo guarda en formato JSON para que pueda crear el modelo Final generado por el sistema.

4.3. Comunicación RabbitMQ

La comunicación desarrollada en este proyecto se maneja utilizando un gestor de mensajería de código abierto llamado RabbitMQ, mediante este *broker* se utilizó el protocolo de comunicación AMQP 0-9-1. Este protocolo permite que se puedan implementar una conexión a múltiples salidas, a estas conexiones se les conoce comúnmente como canales por los cuales se realizaran las operaciones de consumir, administrar y publicar información. Para que los clientes puedan interactuar con RabbitMQ se realizó ciertas configuraciones previas

por parte de cada uno de los clientes. Primero se seleccionó el tipo de puerto llamado TCP, el cual se utilizó en este proyecto, después se determinó el número de puertos que estarán abiertos para que la información fluya por ese medio, también se necesitó registrar el nombre del Host y su dirección IP las cuales corresponderán a los datos del servidor/maestro. El servidor y cliente se comunicarán para determinar si los datos son correctos, en caso de que alguno de estos datos sea incorrecto no permitirá establecer una conexión.

Ahora, una vez se tiene la conexión establecida tanto por parte del Nodo Maestro como por el Nodo Esclavo con RabbitMQ, el Nodo Maestro se encarga de enviar los modelos generados a través de RabbitMQ, el cual recibirá los archivos y los gestiona mediante dos colas. La primera cola generada por RabbitMQ fue llamada "Parámetros" esta cola será la encargada de redireccionar la información del Nodo Maestro hacia los nodos esclavos que estén conectados manteniendo en espera un modelo, para así tener un mejor flujo de información entre los nodos. La otra cola generada fue "Resultados" mediante este canal se buscará recibir todos los archivos con los resultados obtenidos por parte de cada uno de los nodos esclavos y enviarlos al Nodo Maestro para que pueda así realizar el siguiente proceso por parte del Nodo Maestro. En la Figura 4.7 se observa de manera gráfica como trabaja RabbitMQ entre los nodos Esclavo-Maestro.

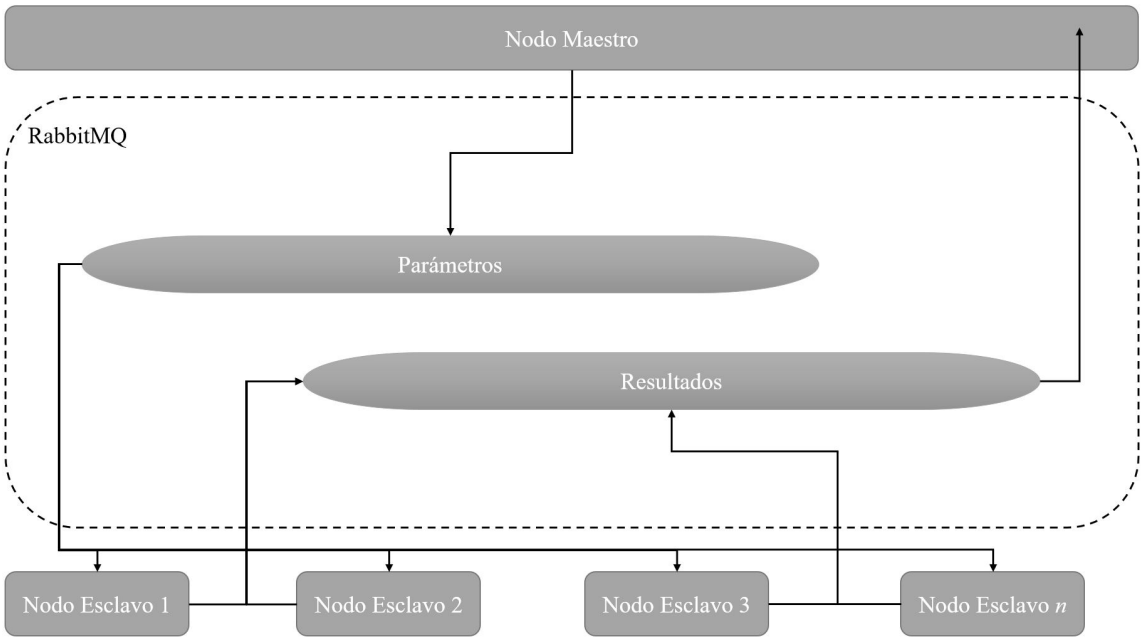

Figura 4.7: Diagrama de comunicación entre nodo maestro y nodos esclavos

El trabajo implementó un modelo arquitectónico maestro-esclavo, este permite la distribución de los entrenamientos realizados por el sistema de AutoML entre *n* número de nodos esclavos que estén escuchando en el momento al bróker RabbitMQ. Se implementó una topología de red centralizada en la cual se cuenta con un servidor central, en este caso el Nodo Maestro y los clientes representados por los nodos esclavos. También se creó una red local la cual funciona de manera exclusiva para la distribución de información, esta red se puede acceder con cable Ethernet o Wifi. Este modelo arquitectónico se basa en el funcionamiento de dos partes principales:

- Nodo Maestro: Este nodo es implementado en una computadora para la generación de modelos, verificación de resultados y el manejo del algoritmo bayesiano para la optimización de modelos.
- Nodo Esclavo: Este nodo se encarga de interpretar las arquitecturas, construir los modelos que recibe del nodo maestro, realizar los entrenamientos y retornar los resultados.

La Figura 4.8 Muestra la interacción que tienen los componentes de la arquitectura propuesta con anterioridad con el gestor de mensajería RabbitMQ.

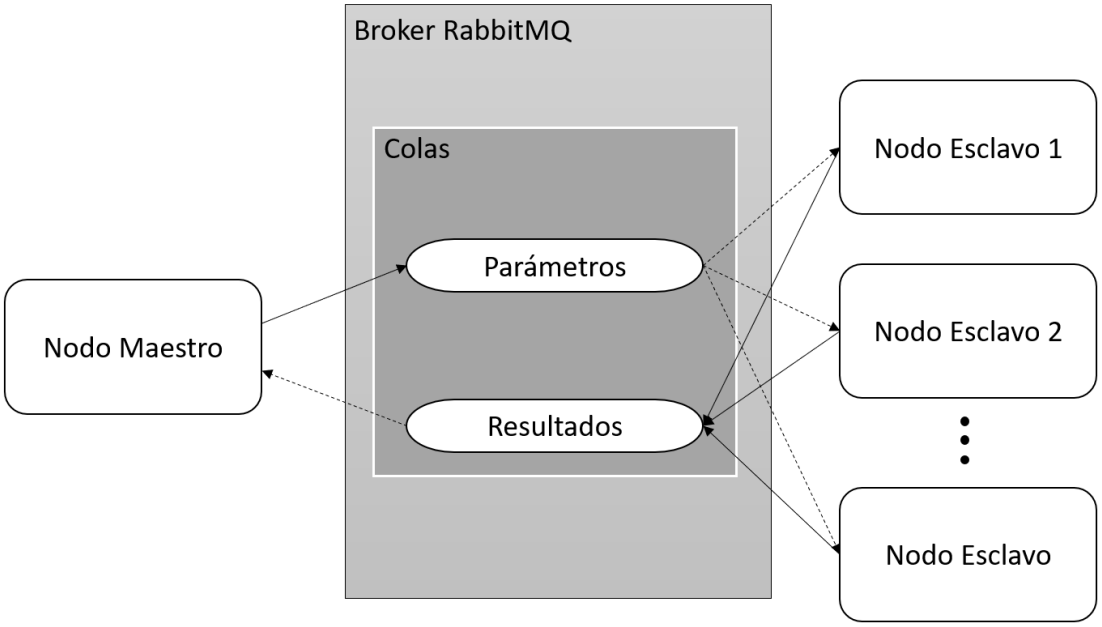

Figura 4.8: Diagrama de Maestro-Esclavo mediante RabbitMQ

4.4. Nodo Esclavo

Todos los nodos esclavos manejan el mismo proceso, este proceso consiste en recibir la petición por parte del nodo maestro a través del *Broker* y ejecutar estas tareas. El Nodo Maestro se encarga principalmente del análisis del rendimiento y de la generación de modelos. Por otro lado, el Nodo Esclavo mediante una cola ”Parámetros”gestionada por RabbitMQ, obtiene un modelo, el cual es construido en función de los parámetros obtenidos a través del archivo JSON que recibió. Después, realiza un entrenamiento, el cual se especificara previamente si se trata de un entrenamiento de exploración o un entrenamiento profundo. Una vez este haya finalizado, regresa los resultados atreves de una cola de “Resultados” que está conectada al *broker* RabbitMQ y este se encarga de dirigirlo hacia el Nodo Maestro. En la Figura 4.9 se muestra de mejor manera el proceso del nodo esclavo.

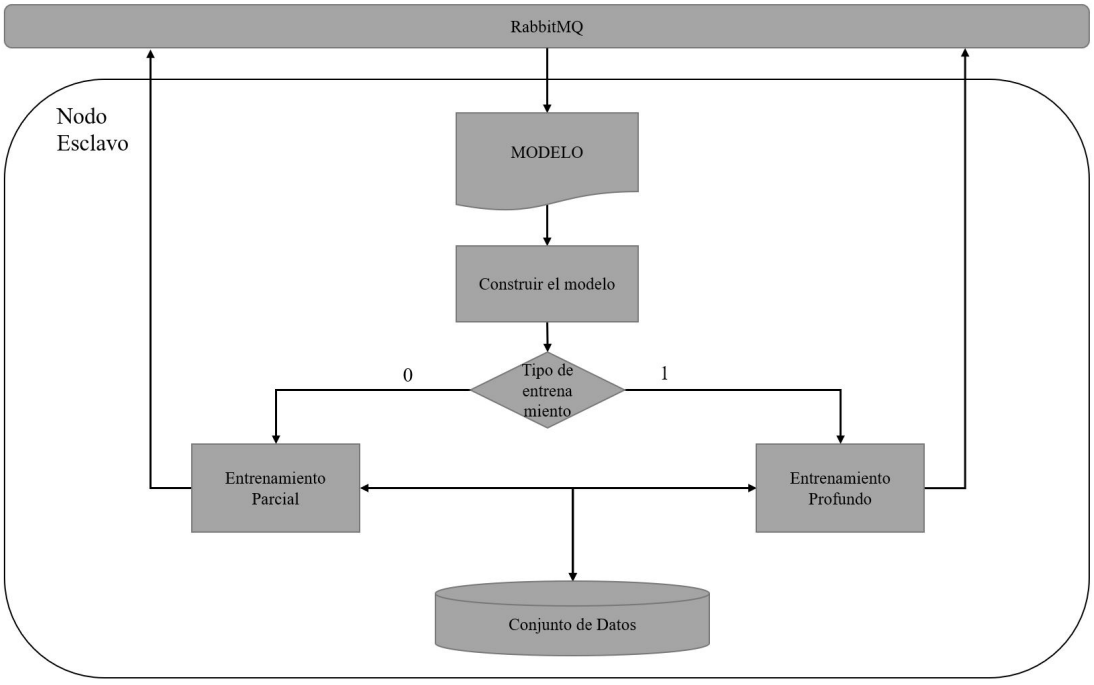

Figura 4.9: Proceso de nodo esclavo

El Nodo Esclavo consiste en varios procesos, los cuales están conectados con RabbitMQ, el cual inicia enviando el archivo JSON a través de la cola “Parámetros”. Una vez se recibe el siguiente archivo pasará a realizar procesos, los cuales se detallan mejor en la Tabla 4.2.

Tabla 4.2: Conjunto de procesos realizados por el nodo esclavo

Características	Definición
Interpretar mensaje	Recibe el mensaje por parte del Bróker en un formato JSON, el nodo esclavo, se encarga de interpretar el mensaje, construir el modelo y entrenarlo a partir de los hiperparámetros especificados
Validar espacio de búsqueda	Válida la compatibilidad de los diccionarios mediante el valor del Hash, en caso de que el trabajador especifique un espacio de búsqueda diferente retornará un mensaje de error.
Cargar el conjunto de datos	Busca el conjunto de datos dentro de la memoria y lo carga para entrenar el modelo. Después en caso de no encontrar el conjunto de datos buscará descargarlo de la fuente en caso de ser posible.
Construir modelo	Construye un modelo de aprendizaje profundo a partir de las especificaciones de entretenimiento.
Entrenar modelo	Se empieza el entrenamiento del modelo creado con el conjunto de datos y los hiperparámetros especificados en la petición.
Enviar resultados	Crea un mensaje en formato JSON con las medidas de rendimiento del entrenamiento y lo manda al Bróker a la cola de resultados.

Por último el Nodo Esclavo regresa un resultado en respuesta del entrenamiento realizado con el modelo, esto sucede una vez termine con la tarea que le fue asignada el nodo Maestro. El archivo resultante es retornado por la cola “Resultados” gestionada por el *broker* RabbitMQ. Los mensajes se codifican en formato JSON y contienen la información descrita en la Tabla 4.3.

Tabla 4.3: Características del archivo resultante

Característica	Definición
ID	Identificador del modelo entrenado, corresponde con el ID de la petición.
Rendimiento	Valor de rendimiento del modelo entrenado parcialmente y de manera profunda, se genera mediante un experimento con el conjunto de datos de prueba.
Épocas terminadas	Valor booleano que indica si el modelo termino de manera satisfactoria las épicas especificadas o si se activó él early Stopping.

La metodología desarrollada en este capítulo se ha implementado en distintas configuraciones para así obtener un panorama más amplio de como interactúa el AutoML en varios ambientes distribuidos. Los cuales, al final generan una respuesta distinta a la misma implementación mostrada con anterioridad. En la Figura 4.10 se muestra un diagrama final de como funciona el AutoML aplicado a un ambiente distribuido.

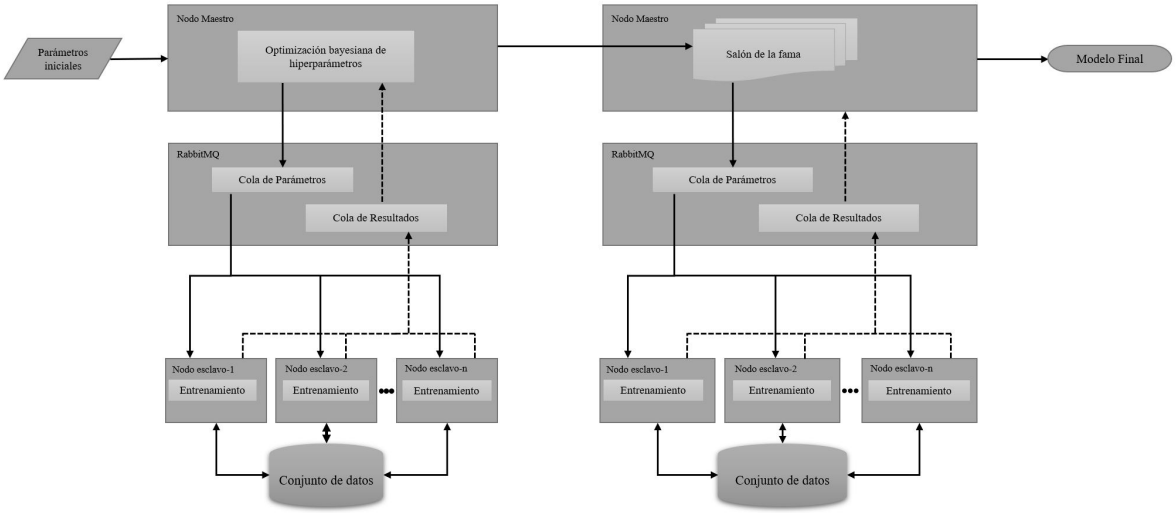

Figura 4.10: Diagrama general del sistema

4.5. Diseño e implementación del hardware en ambientes distribuidos

En esta sección se estará abordando el diseño e implementación utilizado durante el proyecto, se diseñaron dos distintos tipos de configuraciones de hardware. El primero se basa en una conexión LAN donde se creó una red privada como medio de comunicación y el segundo diseño utilizó una comunicación de manera interna, ya que todos los componentes de hardware están conectados de manera física. El tipo de diseño utilizado en cada ambiente distribuido fue aplicado en función al tipo de hardware específico con el que cuentan los equipos de cómputo utilizados. La primera configuración se trata de un diseño de configuración LAN como el que se observa en la Figura (4.11). Para la implementación del experimento se toma en cuenta cinco computadoras, las cuales una será utilizada como Nodo Maestro y las otras

cuatro, serán definidas como los nodos esclavos.

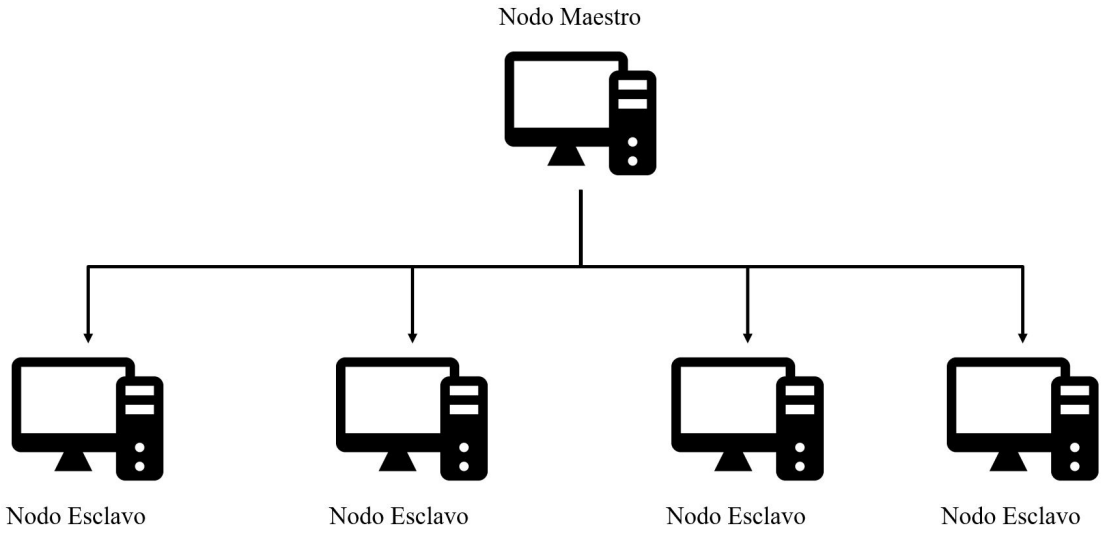

Figura 4.11: Diseño de configuración distribuida con RTX 3060Ti

Cada una de las computadoras cuenta con un hardware homogéneo, por lo que cada computadora cuenta con los mismos componentes, Por tanto, se destaca los componentes claves a la hora de desarrollo de modelos de inteligencia artificial como lo es, la GPU RTX 3060Ti la cual cuenta con 112 *Tensor Cores* y 3,584 **CUDA cores** los cuales, son tecnologías diseñadas especialmente para mejorar el rendimiento a la hora de realizar cálculos matriciales de alta complejidad, también cuenta con un par de memorias DDR4 de 16 GB y una GPU Intel i7 de 8.ª generación.

El segundo ambiente se basa en una configuración con el diseño de un ambiente de cómputo de alto rendimiento, como se observa en la Figura 4.12. Para la implementación del experimento se utilizó una computadora, la cual cuenta dos GPU de manera interna, las cuales harán la función del Nodo Esclavo, mientras que la función de Nodo Maestro será realizada por la CPU.

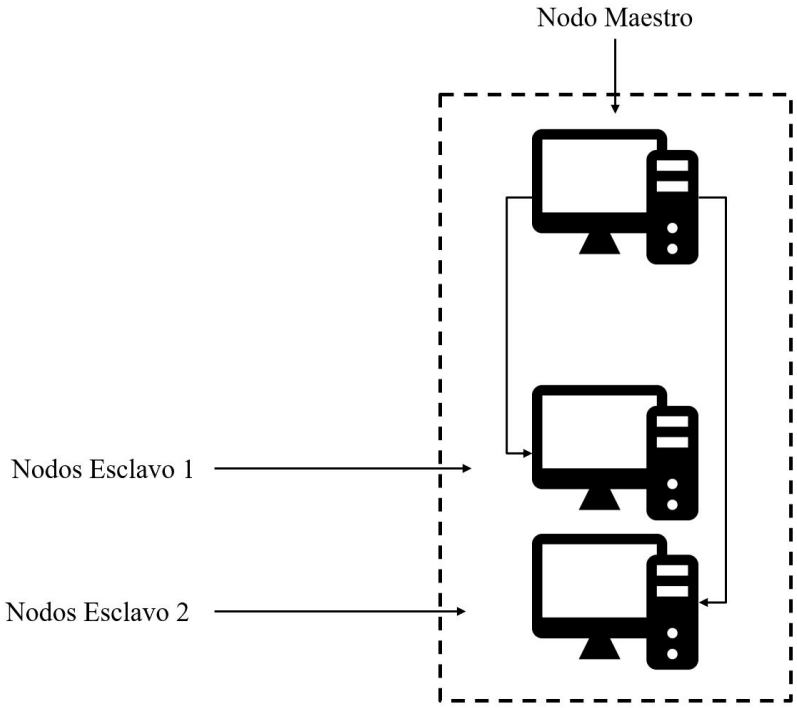

Figura 4.12: Diseño de configuración distribuida con RTX 2080Ti

La computadora cuenta con un único y propio hardware, el cual será utilizado por ambos nodos esclavos durante su ejecución. Los componentes que contienen esta computadora son dos GPU RTX 2080Ti las cuales cuentan con 544 *tesor cores* y 4352 *CUDA cores*, los cuales son la tecnología que les permite acelerar el proceso de cálculos matriciales de alta complejidad. También cuenta con un par de memorias RAM DDR4 de 16 GB a 3200 MHz y una GPU inter i7 de 8.ª generación.

El tercer ambiente utiliza una distinta configuración, pero presenta el mismo diseño que el segundo ambiente, esto se puede observar en la Figura 4.13. La implementación utilizó dos GPU de manera interna y de igual manera que el anterior ambiente utiliza sus GPU como nodos esclavos, mientras que la CPU lo selecciona para realizar la función del Nodo Maestro.

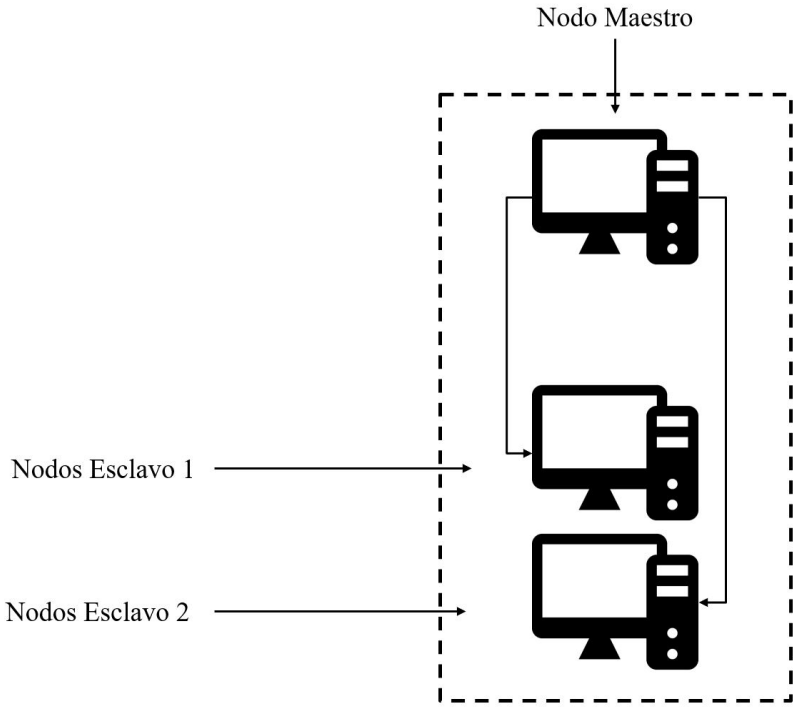

Figura 4.13: Diseño de configuración distribuida con RTX 1080Ti

La computadora cuenta con dos GPU RTX 1080Ti las cuales cuentan con 3584 *CUDA cores* este le permite realizar cálculos de manera más eficiente debido a que realiza los trabajos de manera paralela, aunque a diferencia de las GPU utilizada en los anteriores ambientes, esta no utiliza *Tesor cores* los cuales son microprocesadores dedicados especialmente para operaciones matemáticas. Se utilizaron un par de memorias RAM DDR4 de 16 GB a 3200 MHz y una GPU Intel i7 de 8.ª generación.

Esta cuarta configuración utiliza el diseño de configuración LAN como se observa en la Figura 4.14. Durante la implementación del experimento se utilizaron cuatro computadoras, las cuales se definieron como nodos esclavos, mientras que una computadora fue la encargada de actuar como Nodo Maestro, haciendo un total de cinco computadoras

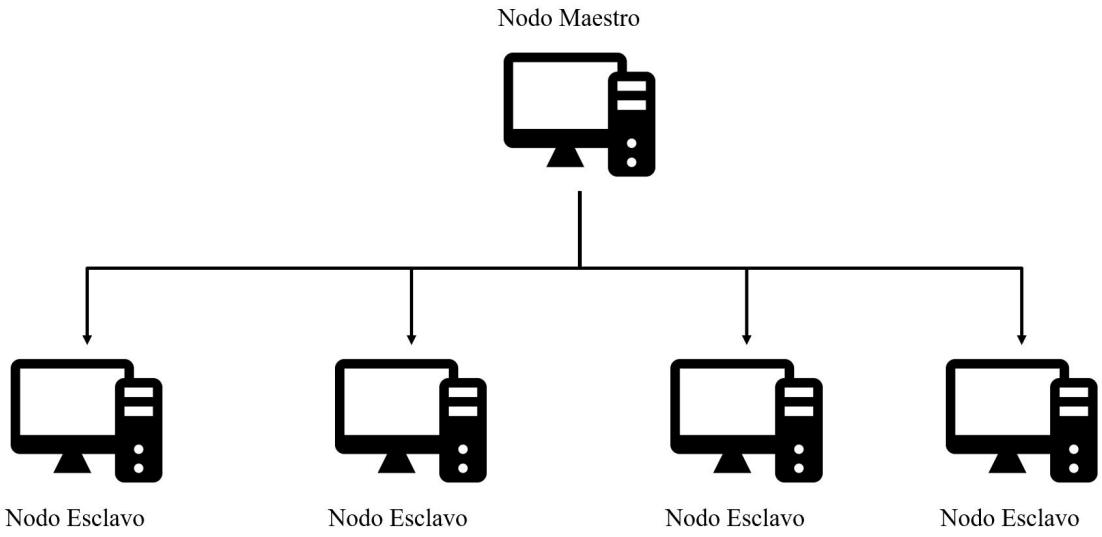

Figura 4.14: Diseño de configuración distribuida con Quadro M4000

Cada una de las computadoras que actuaron como nodos esclavos cuentan con el mismo Hardware. La GPU que estuvieron manejando es la Nvidia Quadro M4000, la cual cuenta con 1664 *CUDA cores*, esta tecnología le permite trabajar de manera más eficaz los cálculos complejos, ya que realiza de manera paralela los procesos que esté ejecutando en el momento. También cuenta con un par de memorias RAM DDR3 de 8 GB a 2400 MHz y un procesador inter i7 de 8.ª generación.

Capítulo 5

Análisis de resultados

En este capítulo se hará un análisis de los resultados con base en las distintas configuraciones distribuidas, las cuales se plantean durante la metodología. Las distintas configuraciones realizadas consisten en dos diferentes tipos de diseño, el primero en una red de distribución local, el cual se manejan distintas computadoras las cuales están interconectadas a través de una red de conexión local (LAN) la cual servirá como medio de transferencia de información y el segundo diseño implementado en este trabajo es el de un ambiente de cómputo de alto rendimiento el cual consiste en tener una computadora con ciertos componentes que lo hacen realizar procesos computacionales de alta complejidad, estas computadoras cuentan con dos GPU en su diseño, por lo que cada GPU haciendo que el trabajo se distribuya de manera interna en cada una de las GPU en representación de los nodos esclavos, mientras que la CPU actuara como Nodo Maestro.

Durante la experimentación se utilizaron distintos conjuntos de datos para la realización de tareas de clasificación y regresión. En el primer caso, donde se utilizó el método de clasificación mediante el uso del AutoML, se utilizaron los conjuntos de datos *fashion_mnist* y *cifar10*. Para el caso de las tareas de regresión se utilizó el conjunto de datos *AMZ_Datos* el cual nos proporciona el valor por acción de la empresa Amazon desde su primer año de cotización en la bolsa hasta la fecha actual, el segundo conjunto de datos es *BTC_datos_historicos* este conjunto de datos igual representa un valor en el tiempo igual que el anterior, pero en este caso en vez de una empresa utiliza la moneda digital y descentralizado Bitcoin. Por último, se experimentó también con el conjunto de datos *Clima_culiacan* donde nos muestra como han sido las temperaturas mínimas y máximas, así como la temperatura promedio del municipio

de Culiacán desde 1985.

El sistema de AutoML fue ejecutado 10 veces con cada uno de los distintos conjuntos de datos mencionados con anterioridad, esto con el objetivo de tener unos resultados más concluyentes, así como ver que tan variable puede ser el comportamiento del sistema. Para los conjuntos de datos de clasificación se utilizó una fase de exploración de 500 modelos y 10 épocas en cada uno de los modelos. Durante la fase de entrenamiento profundo, se seleccionaron 10 modelos que se guardaron anteriormente en el salón de la fama, cada modelo aplicó un total de 100 épocas. Ahora, en el caso de utilizar los conjuntos de datos para la tarea de regresión, se utilizó una fase de exploración de 100 modelos a 10 épocas cada una y en la parte de entrenamiento profundo se utilizaron los 10 modelos seleccionados en el salón de la fama, pero esta vez utilizando 100 épocas durante el entrenamiento.

5.1. Primer ambiente

El primer ambiente distribuido fue implementado mediante un diseño LAN, el cual maneja cuatro computadoras independientes entre sí, que funcionaran como nodos esclavos y una computadora que realizara exclusivamente la tarea de Nodo Maestro. Cada computadora maneja una GPU RTX 3060Ti el cual permite manejar cálculos matriciales de alta complejidad de manera más eficiente debido al uso de su tecnología “Tensor Cores”. Para la experimentación sobre tareas de clasificación se tomaron en cuenta los conjuntos de datos *fashion_mnist* y *cifar10*. En la Tabla 5.1 se pueden observar los resultados de los experimentos durante la implementación del AutoML en el primer ambiente distribuido para la tarea de clasificación. Fueron determinados los parámetros de rendimiento y tiempo, para valorar la eficiencia del sistema. El rendimiento se valoró en función de la precisión que tenía el modelo generado a la hora de clasificar, mientras que el tiempo tomaba en cuenta, cuanto se tardaba en generar el mejor modelo.

Tabla 5.1: Resultados de los experimentos del primer ambiente en tareas de clasificación usando los conjuntos de datos *fasion_mnist* y *cifar10*

Primer ambiente.				
	fashion_mnist		*cifar10*	
#	Rendimiento	Tiempo	Rendimiento	Tiempo
1	0.9237	03:56:33	0.8543	03:28:33
2	0.9244	02:15:02	0.8407	03:04:06
3	**0.9318**	04:15:29	0.8550	03:12:19
4	0.9226	**02:13:17**	0.8424	03:42:52
5	0.9215	03:17:31	0.8518	02:08:58
6	0.9228	02:13:29	0.8440	**02:13:14**
7	0.9229	02:14:16	0.8543	03:05:35
8	0.9215	03:36:33	0.8495	02:55:33
9	0.9221	03:34:17	**0.8557**	03:39:59
10	0.9223	03:26:08	0.8507	03:12:14

Con base en los resultados anteriores se realizó un resumen estadístico, con el cual se puede realizar un mejor análisis. Como se puede observar en la Tabla 5.2 se toman en cuenta los mejores resultados de cada conjunto de datos, su peor resultado, la desviación estándar y el promedio. Esto con el objetivo de poder hacer mejor una comparativa con otros ambientes.

En el caso del primer ambiente se puede observar que para el conjunto de datos *fashion_mnist* el mejor rendimiento lo tuvo en el tercer experimento, a su vez fue el experimento que más tardo en generar el modelo, por lo que necesita determinar que es lo que el usuario considere el factor más importante. Por otro lado, el experimento número cuatro, es el que cuenta con el mejor tiempo en la generación del modelo, con un rendimiento de 0.9226, el cual es un número por debajo del promedio observador durante la experimentación, el cual es de 0.9235.

Por otro lado, cuando se observa el conjunto de datos *cifar10* se puede concluir que el experimento número nueve fue el que presento un mejor rendimiento con respecto a los demás resultados, ahora bien en este caso el tiempo que tardo en generarse no fue el peor, pero aun así fue superior al promedio, por lo que se puede observar una tendencia en que los modelos con una mayor precisión suelen presentar tiempos mayores a la media.

Aunque hay casos en los que un mayor tiempo también puede significar entorpecer el modelo, como en el caso del cuarto experimento, donde genera en la duración más larga del experimento, pero también obtiene un rendimiento más bajo que el promedio obtenido

durante la experimentación.

Tabla 5.2: Resumen estadístico del primer ambiente en tareas de clasificación usando los conjuntos de datos *fashion_mnist* y *cifar10*.

Resumen estadístico del primer ambiente en clasificación				
fashion_mnist	Promedio	Mejor resultado	Peor resultado	Desv. Estandar
Rendimiento	0.9235	0.9318	0.9215	0.0030
Tiempo	03:06:16	02:13:17	04:15:29	00:47:41
cifar10	Promedio	Mejor resultado	Peor resultado	Desv. Estandar
Rendimiento	0.8498	0.8557	0.8407	0.0055
Tiempo	03:04:20	02:08:58	03:42:52	00:32:01

Para las tareas de regresión se utilizó como métricas de medición el error cuadrático medio (MSE) a la hora de evaluar los modelos que genera el sistema, así como también el tiempo en el cual el sistema de AutoML tardar en general el modelo más eficiente. En la Tabla 5.3 se muestran los resultados obtenidos durante la fase de experimentación con cada uno de los conjuntos de datos utilizados, los cuales son: AMZ_Datos, BTC_datos_historicos y Clima_culiacan.

Tabla 5.3: Resultados de los experimentos del primer ambiente en tareas de regresión usando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*.

Primer ambiente.						
	AMZ_Datos		*BTC_datos_historicos*		*Clima_culiacan*	
#	Rendimiento	Tiempo	Rendimiento	Tiempo	Rendimiento	Tiempo
1	0.1750	00:05:30	0.0372	00:04:49	0.0043	00:37:19
2	0.0318	01:01:04	0.0091	**00:03:35**	0.0041	01:11:34
3	0.1315	01:00:32	**0.0067**	00:04:19	0.0043	01:05:10
4	0.3098	01:00:54	0.0089	00:04:22	0.0041	00:31:43
5	0.2487	00:45:36	0.0119	00:05:27	0.0045	01:07:49
6	0.0294	01:00:32	0.0069	01:00:37	0.0042	01:08:55
7	0.0682	**00:05:04**	0.0110	00:45:55	**0.0041**	**00:26:46**
8	0.4192	00:31:47	0.0113	01:00:22	0.0043	01:11:01
9	0.0924	00:07:14	0.0074	00:44:38	0.0043	01:05:10
10	**0.0241**	00:05:35	0.0079	01:00:26	0.0041	00:31:47

A continuación se presentan un resumen estadístico obtenido de los experimentos anteriormente mostrados. Observe en la Tabla 5.4 que se distribuye cada una de las bases de datos con sus respectivos puntos claves para la realización de un análisis comparativo con

otros ambientes distribuidos. Cada conjunto de datos que se está utilizando cuenta con características únicas, esto hará que los resultados puedan no ser muy similares entre sí debido a estas características. El primer conjunto de datos AMZ_Datos muestra una variación muy grande en los tiempos que tarda en general el modelo más óptimo, siendo el tiempo de una hora aproximadamente el peor obtenido en el segundo experimento, contra el experimento número siete, el cual ofrece cinco minutos en generar el mejor resultado. Esto se debe principalmente a la aleatoriedad presentada en el algoritmo de optimización, donde puede ocurrir que un modelo complejo nos brinde una buena solución, así como un modelo más sencillo, el cual no requiera un esfuerzo computacional tan grande por parte del sistema.

Tabla 5.4: Resumen estadístico del primer ambiente en tareas de regresión usando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*.

Resumen estadístico del primer ambiente en regresión.				
AMZ_Datos	Promedio	Mejor resultado	Peor resultado	Desv. Estándar
Rendimiento	0.1530	0.0241	0.4192	0.1345
Tiempo	00:34:22	00:05:04	01:01:04	00:26:12
BTC_datos_historicos	Promedio	Mejor resultado	Peor resultado	Desv. Estándar
Rendimiento	0.0118	0.0067	0.0372	0.0091
Tiempo	00:29:27	00:03:35	01:00:37	00:26:53
Clima_culiacan	Promedio	Mejor resultado	Peor resultado	Dev. Estándar
Rendimiento	0.0042	0.0041	0.0045	0.0001
Tiempo	00:53:43	00:26:46	01:11:34	00:19:34

El segundo conjunto de datos *BTC_datos_historicos* presentó resultados mejores con respecto los resultados recientemente mostrados. Las características de cada conjunto de datos influyen en como se comportan los modelos, aunque en cuestión de rendimiento se ve una notable mejora, esto puede deberse a que hay cambios menos bruscos en el conjunto de datos de *AMZ_Datos*, mismo caso mostrados si observamos los resultados en rendimiento obtenidos por el tercer conjunto de datos donde el clima al ser un dato más constante, permite que exista menos margen de error en los modelos de regresión.

En el caso del tiempo, hay una similitud entre los tres conjuntos de datos y es que todos parecen superar la hora en los tiempos de ejecución cuando se trata de su peor resultado, mientras que el clima parece tener los tiempos más largos, ya que su mejor resultado es de 26 minutos aproximadamente, el hecho de que sea un conjunto de datos más grande que los

demás juega un factor importante a la hora de obtener el modelo.

5.2. Segundo ambiente

La configuración de hardware del segundo ambiente distribuido, cuenta con un diseño de torre, lo que permitió tener dos GPU trabajando como nodos esclavos en la misma computadora. Utiliza dos GPU RTX 2080ti las cuales cuentan con la tecnología de "Tesor Cores", por se permitirá analizar si esto mejora el rendimiento del sistema o reduce los tiempos en los que genera el modelo. Los resultados de utilizar los conjuntos de datos *fasion_mnist* y *cifar10* para la tarea de clasificación se pueden observar en la Tabla 5.5. Para realizar un análisis se determinaron los parámetros de rendimiento, donde se obtiene la precisión con la que el modelo realiza tareas de clasificación y el tiempo que tarda el AutoML en generar el modelo que se utilizó.

Tabla 5.5: Resultados de los experimentos del segundo ambiente en tareas de clasificación usando los conjuntos de datos *fasion_mnist* y *cifar10*.

Segundo ambiente.				
	fashion_mnist		*cifar10*	
#	Rendimiento	Tiempo	Rendimiento	Tiempo
1	0.9293	06:47:47	0.8436	04:29:41
2	**0.9398**	03:54:03	0.8216	04:27:25
3	0.9330	03:32:12	0.8492	04:27:48
4	0.9312	03:29:48	**0.8651**	07:11:07
5	0.9326	03:20:41	0.8461	**04:00:13**
6	0.9243	03:18:48	0.8513	04:34:11
7	0.9201	03:20:35	0.8501	04:44:01
8	0.9265	03:40:35	0.8496	04:44:15
9	0.9276	10:36:01	0.8535	04:35:09
10	0.9253	**03:15:29**	0.8533	05:49:36

Tomando en cuenta el resultado anterior se realizaron los siguientes resúmenes estadísticos. El tener acomodado de mejor manera los puntos claves de los resultados de los experimentos, permite que se pueda realizar un mejor análisis comparativo con respecto a otros ambientes. Como se puede observar en la Tabla 5.6 los puntos más importantes a destacar son el mejor resultado, el peor resultado, el promedio y la desviación estándar de cada uno de

los conjuntos de datos. Se puede destacar que la diferencia de tiempo entre los dos conjuntos de datos en el apartado de promedio es muy similar, siendo apenas una diferencia de 23 minutos. En el apartado del mejor resultado ya empieza a notar una mayor diferencia, pero donde de verdad se puede observar un cambio en el tiempo, es en el peor resultado, donde el conjunto de *fashion_mnits* tarda 3 horas más aproximadamente en comparación al conjunto de *cifar10*. Esto se puede deber principalmente a las características que presentan los datos ingresados. También se debe destacar que el mejor rendimiento, ofrece un mejor resultado comparado con el promedio en caso del primer conjunto de datos. En cambio, con *cifar10* no ocurre esto, sino que el mejor modelo en rendimiento, también es el que presenta un mayor tiempo, por lo que se puede decir que genero un modelo más complejo.

Tabla 5.6: Resumen estadístico del segundo ambiente en tareas de clasificación usando los conjuntos de datos *fashion_mnist* y *cifar10*.

Resumen estadístico del segundo ambiente en clasificación.				
fashion_mnist	Promedio	Mejor resultado	Peor resultado	Desv. Estandar
Rendimiento	0.9289	0.9398	0.9201	0.0055
Tiempo	04:31:36	03:15:29	10:36:01	02:22:56
cifar10	Promedio	Mejor resultado	Peor resultado	Desv. Estandar
Rendimiento	0.8483	0.8651	0.8216	0.0010
Tiempo	04:54:21	04:00:13	07:11:07	00:55:29

Continuando con los experimentos, en la Tabla 5.7 se muestran los resultados obtenidos durante la implementación de un AutoML para las tareas de regresión. Se utilizó el MSE como métrica de medición para el rendimiento junto al tiempo en los conjuntos de datos de AMZ_Datos, BTC_datos_historicos y Clima_culiacan.

En cada uno de los conjuntos de datos se muestra una tendencia sobre la constancia que tiene cada conjunto con respecto al rendimiento y tiempo. Al manejar datos metereológicos en el conjunto de datos Clima_culiacan, sus datos más estables permiten que el margen de error sea menor con respecto a los demás.

Tabla 5.7: Resultados de los experimentos del segundo ambiente en tareas de regresión usando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*.

Segundo ambiente.						
	AMZ_Datos		*BTC_datos_historicos*		*Clima_culiacan*	
#	Rendimiento	Tiempo	Rendimiento	Tiempo	Rendimiento	Tiempo
1	0.1794	00:08:52	0.0380	00:03:31	**0.0051**	00:58:16
2	0.0981	00:09:59	0.0425	00:08:22	0.0055	00:48:05
3	0.0107	00:07:40	0.0098	00:03:35	0.0052	01:17:21
4	0.0703	00:12:43	0.0210	00:03:30	0.0054	01:28:19
5	**0.0071**	00:07:14	0.0320	00:03:43	0.0052	02:00:56
6	0.0388	00:15:43	0.0337	00:03:35	0.0053	03:37:33
7	0.0358	**00:06:22**	0.0235	**00:03:20**	0.0053	00:47:37
8	0.0298	00:08:58	**0.0058**	00:03:40	0.0055	01:37:08
9	0.0195	00:08:05	0.0370	00:03:48	0.0052	00:55:54
10	0.1597	00:09:46	0.0258	00:03:57	0.0054	**00:33:03**

Por último presentamos los datos en manera de resumen con los conjuntos de datos de regresión. Observe la Tabla 5.8 para tener mejor comprensión de los resultados mediante el análisis de los parámetros previamente establecidos, los cuales fueron el mejor resultado, pero resultado, desviación estándar y promedio. En cuestión de tiempo, se observa que el segundo conjunto de datos BTC_datos_historicos tiene tiempos más bajos con respecto a los otros conjuntos de datos en cada uno de los parámetros que se toman en cuenta. En el caso del rendimiento se determinó que previamente que los datos del tercer conjunto Clima_culiacan tienen un menor error con respeto a los demás. Aun así, se observa que los mejores resultados obtenidos en cada base de datos no muestran una diferencia mayor a 0.002, por lo que se muestrearon buenos resultados en rendimiento.

Aunque también se puede observar el rendimiento desde la perspectiva de cuanto es el rendimiento promedio que está ofreciendo cada base de datos. Al tomar en cuenta tal parámetro se observa que la diferencia aumenta considerablemente, pasando a ser el mejor promedio en rendimiento el tercer conjunto de datos *Clima_culiacan* con 0.0053, mientras que el primer conjunto de datos *AMZ_Datos* con un rendimiento de 0.0649 paso a ser el peor promedio tendiendo una diferencia de 0.0596 entre ambos conjuntos.

Tabla 5.8: Resumen estadístico del segundo ambiente en tareas de regresión usando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*.

Resumen estadístico del segundo ambiente en regresión.				
AMZ_Datos	Promedio	Mejor resultado	Peor resultado	Desv. Estándar
Rendimiento	0.0649	0.0071	0.1794	0.0617
Tiempo	00:09:32	00:06:22	00:15:43	00:02:48
BTC_datos_historicos	Promedio	Mejor resultado	Peor resultado	Desv. Estándar
Rendimiento	0.0269	0.0058	0.0425	0.0121
Tiempo	00:04:06	00:03:20	00:08:22	00:01:30
Clima_culiacan	Promedio	Mejor resultado	Peor resultado	Dev. Estándar
Rendimiento	0.0053	0.0051	0.0055	0.0001
Tiempo	01:24:25	00:33:03	03:37:33	00:53:51

Por último, es adecuado señalar que aunque la base de datos de Clima_culiacan está presentado buenos resultados con base al rendimiento, el tiempo que tarda en general los modelos es demasiado alto en comparación a los demás conjuntos de datos por un espectro muy amplio de tiempo. Esto se puede deber a varios factores como la complejidad del modelo, ya que la desviación estándar de 53 minutos, lo que indica una dispersión de los datos demasiado amplia en el tiempo. También se debe puntualizar el tamaño del conjunto de datos, el cual es más grande con respecto a los primeros dos, siendo un factor importante en la duracion de los entrenamientos.

5.3. Tercer ambiente

La configuración de hardware del tercer ambiente distribuido, cuenta con un diseño de torre, lo que permitió tener dos GPU trabajando como nodos esclavos en la misma computadora. Utiliza dos GPU GTX 1080ti las cuales no cuentan con la tecnología de “Tesor Cores”, por lo que al realizar los experimentos se observa de manera más clara la falta de potencia al momento de realizar cálculos matriciales complejos. Los resultados de utilizar los conjuntos de datos *fasion_mnist* y *cifar10* para la tarea de clasificación se pueden observar en la Tabla 5.9. Para realizar un análisis se determinaron los parámetros de rendimiento, donde se obtiene la precisión con la que el modelo realiza tareas de clasificación y el tiempo que tarda el AutoML en generar el modelo que se utilizó.

Tabla 5.9: Resultados de los experimentos del tercer ambiente en tareas de clasificación usando los conjuntos de datos *fasion_mnist* y *cifar10*.

Tercer ambiente.				
	fashion_mnist		*cifar10*	
#	Rendimiento	Tiempo	Rendimiento	Tiempo
1	0.9229	05:51:27	**0.8629**	07:48:25
2	0.9247	06:00:22	0.8406	07:51:55
3	**0.9318**	06:59:29	0.8544	08:33:56
4	0.9244	05:56:06	0.8396	07:38:02
5	0.9250	06:17:31	0.8579	08:14:24
6	0.9243	05:27:09	0.8277	07:30:57
7	0.9261	**05:13:23**	0.8389	**06:27:41**
8	0.9240	05:57:37	0.8485	07:36:55
9	0.9265	06:03:42	0.8457	07:31:54
10	0.9247	06:09:38	0.8504	08:04:53

Ahora bien, con base en los resultados anteriores, se realizó un resumen estadístico que permite realizar un análisis entre los distintos ambientes de manera más sencilla. La Tabla 5.10 toma como puntos claves, el mejor resultado, el peor resultado y la desviación estándar con respecto al rendimiento y tiempo obtenido con respecto a los resultados obtenidos.

Se puede observar que los conjuntos de datos *fashion_mnist* y *cifar10* han aumentado considerablemente los tiempos en los que se genera el modelo con respecto a los demás ambientes. Pero aun así, se sigue observando la tendencia de que el conjunto de datos *fashion_mnist* presenta tiempos más cortos de ejecución en comparación al segundo conjunto de datos. El mejor resultado obtenido por *cifar10* fue en el primer experimento con un valor de 0.8629, con respecto a cuanto tiempo tardo en obtener ese modelo fue un tiempo de 07:48:25, lo que comparado con el tiempo promedio de los experimentos solo es superado por cinco minutos aproximadamente haciéndolo un muy buen resultado.

Continuando ahora con el conjunto de datos fashion_mnist su mejor resultado es de 0.9318, el cual se obtuvo durante el tercer experimento, tomándole un tiempo de casi siete horas. Haciendo la comparativa con los valores promedios, se determina que en el apartado de tiempo, no hay una diferencia tan grande siendo de tan solo 46 minutos. Al tener esta diferencia con tiempos elevados de más de seis horas, se puede determinar que los tiempos promedios están muy cerca del mejor resultado, haciendo un factor muy positivo para el

conjunto de datos de *fashion_mnist*.

Tabla 5.10: Resumen estadístico del tercer ambiente en tareas de clasificación usando los conjuntos de datos *fashion_mnist* y *cifar10*.

Resumen estadístico del tercero ambiente en clasificación.				
fashion_mnist	Promedio	Mejor resultado	Peor resultado	Desv. Estandar
Rendimiento	0.9254	0.9318	0.9229	0.0024
Tiempo	05:59:38	05:13:23	06:56:29	00:28:28
cifar10	Promedio	Mejor resultado	Peor resultado	Desv. Estandar
Rendimiento	0.8466	0.8629	0.8277	0.0104
Tiempo	07:43:54	06:27:41	08:33:51	00:33:36

Los siguientes experimentos fueron obtenidos al implementar un AutoML en una configuración de hardware distribuida para las tareas de regresión. Como métricas de evaluación se utilizó MSE para medir el rendimiento y el tiempo que se tardó en generar el modelo. En la Tabla 5.11 se observan los siguientes resultados de los experimentos utilizando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*. Esto con el objetivo de tener un orden con respecto a los resultados obtenidos y así realizar un análisis sobre cada uno de los ambientes.

Tabla 5.11: Resultados de los experimentos del tercer ambiente en tareas de regresión usando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*.

Tercer ambiente.						
	AMZ_Datos		*BTC_datos_historicos*		*Clima_culiacan*	
#	Rendimiento	Tiempo	Rendimiento	Tiempo	Rendimiento	Tiempo
1	0.1542	00:13:44	0.0139	00:38:12	0.0042	01:28:06
2	0.3253	00:17:06	0.0097	00:39:29	0.0040	01:31:25
3	0.1338	00:14:03	0.0931	00:54:13	0.0040	00:56:37
4	0.0548	00:18:34	**0.0087**	00:43:29	0.0041	00:57:50
5	0.1081	00:18:20	0.0357	**00:29:53**	0.0042	01:16:28
6	0.0902	00:15:34	0.2408	00:44:30	0.0041	01:11:10
7	0.1961	00:17:15	0.0352	00:57:20	0.0043	01:23:30
8	0.0449	00:19:08	0.0312	00:45:55	0.0043	01:30:45
9	**0.0135**	**00:12:24**	0.0464	00:43:38	0.0041	00:55:54
10	0.2684	00:15:09	0.0150	00:42:06	**0.0040**	**00:52:03**

Teniendo en cuenta los resultados obtenidos anteriormente, se resumieron los datos obtenidos con base en algunos puntos claves como lo son el mejor resultado, el peor resultado,

la desviación estándar y el promedio. En la Tabla 5.12 se pueden observar los resultados organizados de manera que permita realizar un análisis conciso de los experimentos realizados en regresión. Los conjuntos de datos de regresión, han mostrado que necesitan menos tiempo que los modelos de clasificación. Pero aun así, se observa que siguen la tendencia de que necesitan cada vez más tiempo para generar los modelos con respecto a los ambientes que cuentan con una GPU moderna en los nodos esclavos. Se logra mantener una regularidad en los resultados obtenidos, esto se puede puntualizar debido a que si se observa la desviación estándar, no está siendo demasiado alta con base en resultados.

Continúa la misma línea en que el conjunto de datos *Clima_culiacan* obtiene el mejor rendimiento en cada uno de los puntos claves. Aunque en cuestión de tiempo mantiene las altas duraciones en generar el modelo. En el conjunto de datos *AMZ_Datos* se observa que el experimento número obtuvo el mejor resultado en el menor tiempo de todos los experimentos. Por lo tanto, se puede concluir que los buenos resultados también se pueden obtener en pequeñas cantidades de tiempo, aunque siga influyendo el factor de aleatoriedad en el algoritmo de optimización.

Tabla 5.12: Resumen estadístico del tercer ambiente en tareas de regresión usando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*.

Resumen estadístico del tercer ambiente en regresión.				
AMZ_Datos	Promedio	Mejor resultado	Peor resultado	Desv. Estándar
Rendimiento	0.1389	0.0135	0.3253	0.1
Tiempo	00:16:08	00:12:24	00:19:08	00:02:18
BTC_datos_historicos	Promedio	Mejor resultado	Peor resultado	Desv. Estándar
Rendimiento	0.0529	0.0087	0.2408	0.0705
Tiempo	00:43:52	00:29:53	00:57:20	00:07:46
Clima_culiacan	Promedio	Mejor resultado	Peor resultado	Dev. Estándar
Rendimiento	0.0041	0.0040	0.0043	0.0001
Tiempo	01:12:23	00:52:03	01:31:25	00:15:45

También se debe destacar que el conjunto de datos *BTC_datos_historicos* presenta buenos resultados con respecto a su mejor rendimiento, el cual fue obtenido en el cuarto experimento con un valor de 0.0087 en un tiempo aproximado de 43 minutos. Concluyendo de esta manera que se obtienen resultados constantes, aunque no se determinen como los mejores, están en un punto intermedio con los otros conjuntos de datos.

5.4. Cuarto ambiente

La configuración de hardware del cuarto ambiente distribuido cuenta con un diseño de conexión LAN. Esto permitió que se tengan cuatro computadoras independientes actuando como nodos esclavos y una computadora dedicada especialmente para el Nodo Maestro. Cada computadora maneja una GPU Quadro M4000, los cuales no cuentan con la tecnología "Tensor Cores", por lo que se vera reflejada una disminución en la capacidad para resolver problemas matriciales de alta complejidad. Los resultados mostrados en la Tabla 5.13 fueron obtenidos mediante la implementación del sistema de AutoML utilizando los conjuntos de datos *fashion_mnist* y *cifar10* para la tarea de clasificación en el ambiente distribuido. Esto con el objetivo de realizar un análisis sobre el comportamiento que tienen el sistema de AutoML en el comportamiento del hardware.

Tabla 5.13: Resultados de los experimentos del cuarto ambiente en tareas de clasificación usando los conjuntos de datos *fasion_mnist* y *cifar10*.

Cuarto ambiente.				
	fashion_mnist		*cifar10*	
#	Rendimiento	Tiempo	Rendimiento	Tiempo
1	0.9319	18:30:44	0.8436	13:25:36
2	0.9193	10:05:28	0.8216	12:06:45
3	0.9261	14:38:06	0.8492	11:17:25
4	0.9254	10:08:56	**0.8651**	19:05:25
5	0.9200	15:05:30	0.8461	10:00:15
6	0.9269	23:14:03	0.8235	12:15:43
7	0.9194	15:34:54	0.8510	15:14:50
8	0.9239	07:40:33	0.8536	14:31:16
9	0.9261	**05:46:33**	0.8325	**09:45:56**
10	**0.9333**	14:28:41	0.8610	18:36:12

Continuando con el análisis de resultados, se realizó un resumen estadístico que permite manejar de manera más eficiente ciertos puntos claves de los experimentos como lo son él mejore resultado, el peor resultado, la desviación estándar y el promedio. En la Tabla 5.14 se pueden observar con detalle cada uno de estos puntos mencionados con anterioridad. Destacando como primer punto que los tiempos en los que tardo el sistema en generar el mejor modelo son elevados, más si se comparan con los ambientes ya implementados anteriormen-

te.

El conjunto de datos *fashion_mnist* presenta el experimento con mayor tiempo en generar su mejor modelo, con un tiempo exacto de 24:14:13. Este número es tres veces más grande que se compara con el mejor resultado, el cual se obtuvo en el noveno experimento con un tiempo de 5:46:33. Ahora bien, si hace la comparación con el conjunto de datos *cifar10* se observa que la tendencia no cambia mucho, el mejor rendimiento obtuvo unos resultados de 9 horas aproximadamente, mientras que el peor resultado tardo 19 horas aproximadamente, siendo poco más del doble de tiempo en generar el modelo final.

Tabla 5.14: Resumen estadístico del cuarto ambiente en tareas de clasificación usando los conjuntos de datos *fashion_mnist* y *cifar10*.

Resumen estadístico del cuarto ambiente en clasificación.				
fashion_mnist	Promedio	Mejor resultado	Peor resultado	Desv. Estandar
Rendimiento	0.9252	0.9333	0.9193	0.0048
Tiempo	13:31:22	05:46:33	23:14:13	05:13:17
cifar10	Promedio	Mejor resultado	Peor resultado	Desv. Estandar
Rendimiento	0.8447	0.8651	0.8216	0.0147
Tiempo	13:37:56	09:45:56	19:05:25	03:15:41

También se debe de tomar en cuenta que resultados en el apartado de rendimiento está ofreciendo el ambiente distribuido. Si se observa con cuidado, se puede ver que los resultados son aceptables comparados con los anteriores ambientes, también señalar que la desviación estándar es muy pequeña, por lo que se puede determinar que han obtenido resultados constantes en lo que a precisión se refiere. Por último, también se observa que el conjunto de datos *fashion_mnist* obtuvo su mejor resultado en el décimo experimento, mientras que en el conjunto de datos de *cifar10* se generó en el cuarto experimento. En ambos casos presentaron un tiempo que supera al promedio, por lo que no fueron precisamente rápidos en ninguno de estos casos.

Para la implementación de las tareas de regresión se utilizaron los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*. Se utilizó el error cuadrático medio(MSE) para medir el rendimiento de los modelos generados, así como también se registró el tiempo que tardaba en generarse ese modelo. Esto con la finalidad de organizar los datos de tal manera que permita realizar un análisis adecuado de los resultados de los experimentos.

En la Tabla 5.15 se pueden observar los resultados de los distintos experimentos realizados con cada uno de los conjuntos de datos

Tabla 5.15: Resultados de los experimentos del cuarto ambiente en tareas de regresión usando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*.

Cuarto ambiente.						
	AMZ_Datos		*BTC_datos_historicos*		*Clima_culiacan*	
#	Rendimiento	Tiempo	Rendimiento	Tiempo	Rendimiento	Tiempo
1	0.0880	00:06:45	0.0080	00:06:07	**0.0040**	00:22:36
2	0.1093	00:06:48	0.0116	00:06:08	0.0042	00:17:13
3	0.1428	00:10:17	0.0098	00:12:01	0.0042	00:33:13
4	**0.0359**	**00:06:41**	0.0136	00:07:25	0.0041	**00:13:26**
5	0.0713	00:07:03	**0.0076**	00:05:31	0.0041	00:19:57
6	0.2205	00:12:04	0.0110	**00:04:41**	0.0043	00:20:41
7	0.0762	01:03:28	0.0096	00:08:20	0.0043	00:21:27
8	0.2103	00:08:47	0.0104	00:06:13	0.0045	00:19:34
9	0.1550	00:09:41	0.0207	00:09:39	0.0041	00:19:35
10	0.0993	00:10:47	0.0079	00:09:08	0.0042	00:19:13

Ahora los siguientes datos presentados se obtuvieron al realizar el resumen estadístico con base en los conjuntos de datos de regresión. En la Tabla 5.16 se pueden observar cada uno de los puntos claves obtenidos durante la experimentación, siendo estos el promedio, mejor resultado, peor resultado y desviación estándar. Al observar cada uno de los conjuntos de datos, se puede observar que los tiempos de ejecución suelen ser muy bajos, destacando el segundo ambiente, el cual tiene como su mejor resultado cuatro minutos aproximadamente. También se debe puntualizar que el conjunto de datos *Clima_culiacan* tiene unos tiempos de ejecución más bajos en comparación con los otros ambientes, manteniendo sus buenos resultados en rendimiento y una desviación estándar baja, lo que hace tener los mejores resultados durante la implementación del cuarto ambiente.

Tabla 5.16: Resumen estadístico del cuarto ambiente en tareas de regresión usando los conjuntos de datos *AMZ_Datos*, *BTC_datos_historicos* y *Clima_culiacan*.

Resumen estadístico del cuarto ambiente en regresión.				
AMZ_Datos	Promedio	Mejor resultado	Peor resultado	Desv. Estándar
Rendimiento	0.1208	0.0359	0.2205	0.0604
Tiempo	00:14:14	00:06:41	01:03:28	00:17:24
BTC_datos_historicos	Promedio	Mejor resultado	Peor resultado	Desv. Estándar
Rendimiento	0.0110	0.0076	0.0207	0.0038
Tiempo	00:07:31	00:04:41	00:12:01	00:02:15
Clima_culiacan	Promedio	Mejor resultado	Peor resultado	Dev. Estándar
Rendimiento	0.0042	0.0040	0.0045	0.0001
Tiempo	00:20:41	00:13:26	00:33:13	00:05:04

Ahora bien, otro punto importante a señalar es que en el conjunto de datos *AMZ_Datos*, los resultados obtenidos durante la séptima experimentación se obtuvo el peor tiempo entre los tres conjuntos de datos, obteniendo un tiempo de poco más de una hora en generar el modelo aproximadamente alejándose demasiado del mejor resultado y tiempo promedio. La media de este conjunto de datos está en los 14 minutos, por lo que el peor resultado se puede considerar una medida atípica dentro de los resultados obtenidos en la experimentación. Por último, el conjunto de datos *BTC_datos_historicos* ha tenido el comportamiento más estable entre los tres conjuntos de datos utilizados, con una desviación estándar relativamente baja en comparación a sus resultados, manteniendo tiempos de ejecución eficaces con respecto a los demás, así como un rendimiento bueno si lo consideramos con los demás ambientes.

5.5. Análisis comparativo

En esta sección se estará realizando una comparación del rendimiento, así como en los tiempos de ejecución realizada por los distintos conjuntos de datos en cada uno de las configuraciones en las cuales fueron aplicadas. Se utilizaron distintos modelos de gráficas para tener un mejor entendimiento de los valores anteriormente mostrados y como estos se relacionan para así poder determinar cuál será la opción más conveniente comando en cuenta los resultados generales. Debido a que el componente clave en cada una de las distintas configuraciones es la GPU, se utiliza como referencia dentro de este análisis.

En la Figura 5.1 mostrada a continuación podemos observar como se representa a través de un gráfico de barras las comparativas dentro del peor rendimiento, el mejor rendimiento y el rendimiento promedio ofrecido por cada una de las configuraciones. Analizando primero los resultados promedios mostrados en la primera imagen, podemos observar que en cuestión de constancia, la precisión obtenida en las tareas de clasificación no varía demasiado en cada uno de los siguientes modelos, siendo el mejor la configuración con la RTX 3060Ti con 0.8498 en cuestión de rendimiento, comparado con las computadoras que tienen Quadro M4000 las cuales ofrecieron un promedio de 0.8447 no necesariamente se obtuvo una diferencia relevante aunque aun así existe esa diferencia. Cabe destacar que el mejor rendimiento obtenido entre todas las configuraciones fue un empate entre las configuraciones RTX 2080Ti y las Quadro M4000, de esta manera se puede determinar que todas las configuraciones son propicias a dar buenos resultados, aunque, por otro lado, dentro de los peores rendimientos también fueron obtenidos por estas mismas configuraciones con un valor de 0.8216 en precisión. Como punto final, si analizamos un espectro amplio de regularidad, podremos decir que la configuración con las RTX 3060Ti presentan una mejor fiabilidad a la hora de obtener resultados, ya que aunque no nos dio el mejor rendimiento, si muestra ser más constante obteniendo mejores modelos que las otras configuraciones.

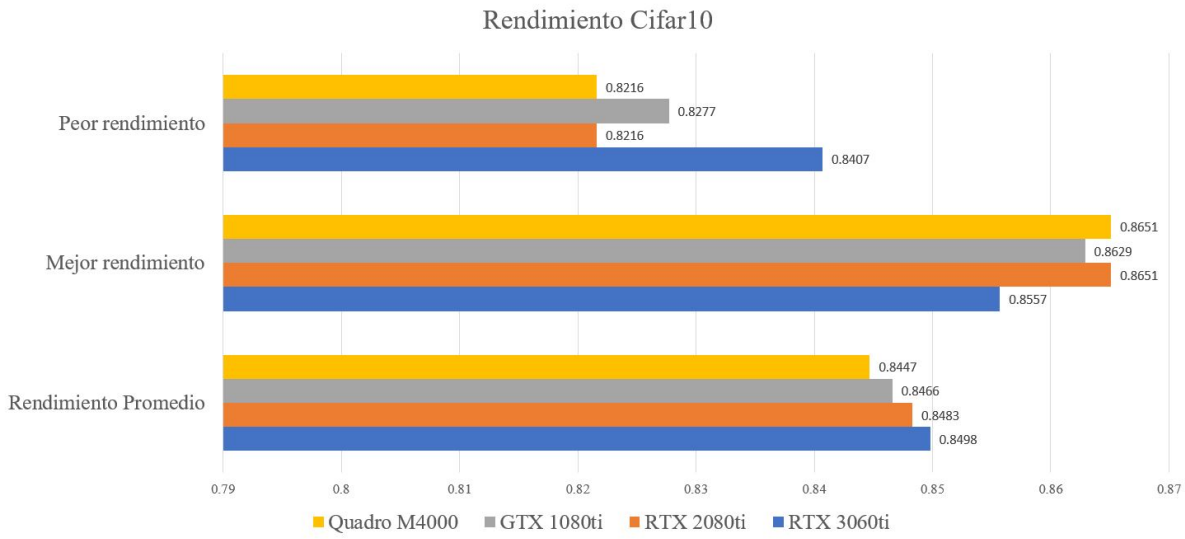

Figura 5.1: Análisis comparativo en rendimiento del conjunto de datos *cifar10* para la tarea de clasificación de imagenes

Ahora se mostrara una breve observación a la Figura 5.2 donde se observa las configuraciones en función del tiempo de ejecución del sistema. La manera en la que se representan los datos es mucho más simple que en la cuestión de rendimiento, aquí se tomara como punto clave cuál es la configuración que realiza más rápido el proceso total de obtención del modelo de machine learning. La configuración con RTX 3060Ti se destaca mucho en comparación con los otros ambientes, esto en la gráfica se hace demostrar de manera muy clara, también se puede determinar que tiempo más largo de obtención es de la configuración con las Quadro M4000, por lo que a pesar de tener el modelo con el mejor rendimiento se puede llegar a obtener en un tiempo de 13 horas aproximadamente en promedio, esto en contraste con la RTX 3060Ti que solo requiere 3 horas promedio para generar un modelo.

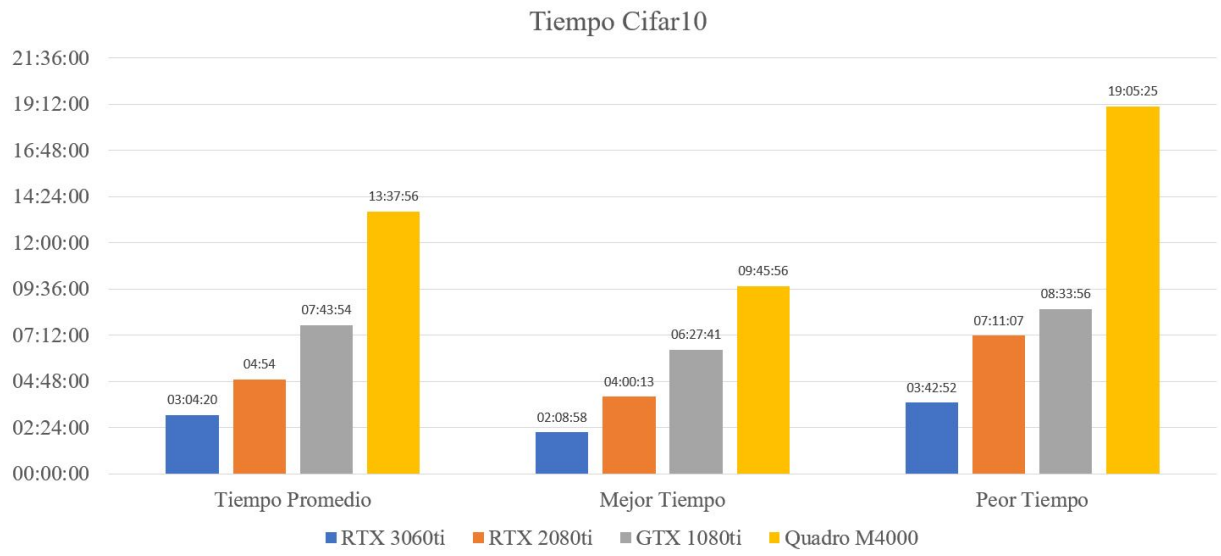

Figura 5.2: Análisis comparativo en tiempo del conjunto de datos *cifar10* para la tarea de clasificación de imagenes

La Figura 5.3 muestra los resultados más significativos obtenidos por el conjunto de datos Fashion_mnist. Para medir el rendimiento se utilizó la precisión como medida de evaluación del rendimiento, entre más cerca esten los resultados de las pruebas generadas por los modelos del 1 se pueden considerar que más precisos. El peor rendimiento observado durante los experimentos fue obtenido por el cuarto ambiente distribuido, el cual se caracteriza por tener Nvidia Quadro M4000 como componente clave de procesamiento, esto con un valor

de 0.9193. En cambio, en contraste tenemos que la configuración que genero el mejor rendimiento obtenido es el que utiliza las Nvidia RTX 2080Ti con un valor de 0.9398. Se puede observar que fue una gran diferencia, partiendo de este punto, podemos determinar que la mejor opción es el uso del segundo ambiente, ya que también cuenta con la mayor estabilidad en la generación de modelos, esto también se puede observar en los resultados mostrados a continuación.

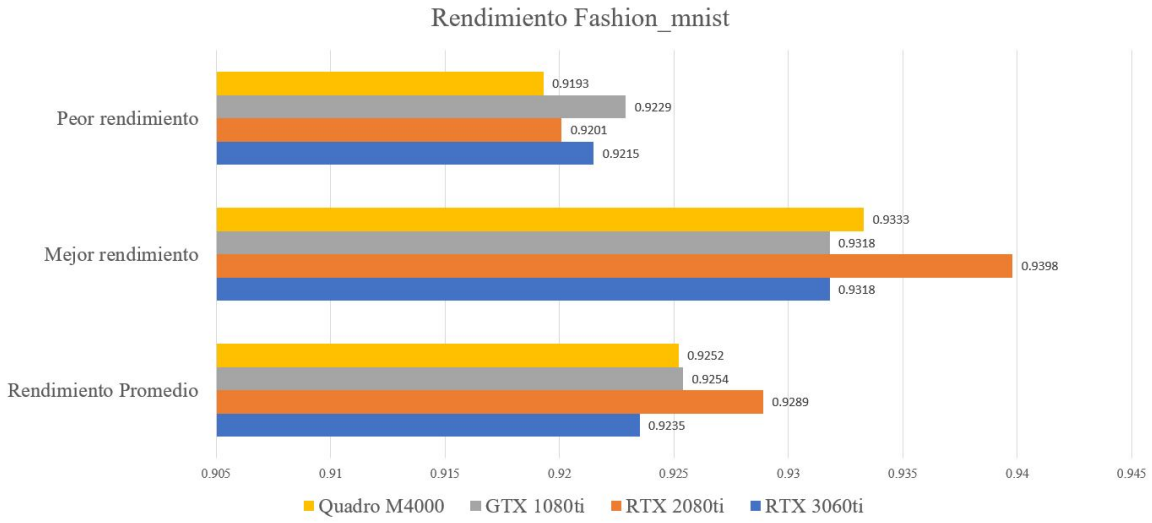

Figura 5.3: Análisis comparativo en rendimiento del conjunto de datos *fashion_mnist* para la tarea de clasificación de imágenes

El rendimiento no es el único parámetro que se evaluando en nuestros experimentos, aunque ciertamente es un punto clave, también se debe tomar en cuenta el tiempo que tarda el AutoML en generar el modelo. En la gráfica anterior se mencionó que el segundo ambiente con la RTX2080Ti nos daba el mejor rendimiento al hacer una contraparte con el tiempo de ejecución, se puede observar que en tiempo promedio se encuentra en segundo lugar con 4 horas y medias, mientras que el primer lugar lo obtiene en este caso el primer ambiente el cual cuenta con RTX 3060Ti. Esta misma tendencia también se puede observar los mejores tiempos obtenidos, por lo que teniendo una diferencia de 1 hora aproximada entre ambos ambientes. Ahora, si bien se tiene que los resultados en rendimiento promedio no son los mejores, se debe hacer una distinción que en caso de que el trabajo que se realiza necesita de una alta precisión se determina como mejor opción el segundo ambiente, mientras que en

caso de que lo que busque el proyecto es obtener los modelos a la mayor brevedad posible, el primer ambiente será la elección correcta, de esta manera el punto clave para determinar la mejor opción en este caso parte del tipo de proyecto que realizara el usuario.

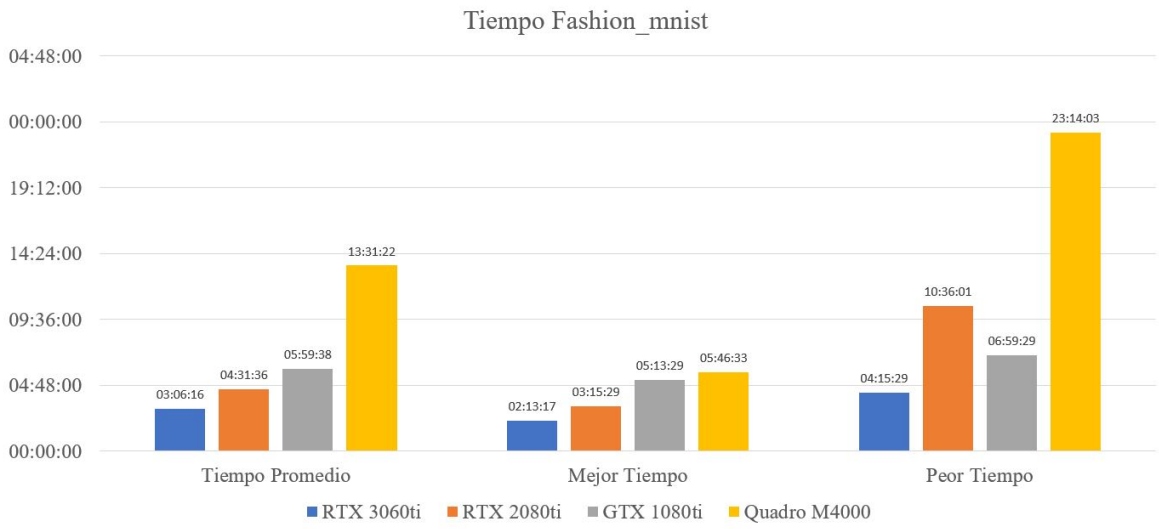

Figura 5.4: Análisis comparativo en tiempo del conjunto de datos *fashion_mnist* para la tarea de clasificación de imágenes

A continuación se muestra el análisis del los conjuntos de datos para las tareas de regresión, primero se debe puntualizar que las métricas de evaluación durante los experimentos fue el error obtenido por los modelos durante las pruebas, por lo que los valores que estaremos tomando en cuenta son con base en esa medición buscando obtener el valor más cercano a cero. También se sigue utilizando el tiempo como una de las medidas claves en las pruebas realizadas. La Figura 5.5 muestra los resultados claves obtenidos en los experimentos de manera más gráfica, por lo que se destaca principalmente con las pruebas con el mejor rendimiento, obteniendo la segunda configuración el mejor resultado con 0.0071. En cuestión de rendimiento promedio también sale victorioso con un valor de 0.0649, por lo que nos asegura que constantemente tendrá buenos resultados y por último si se observa también los peores resultados durante la experimentación se puntualiza que aunque el valor de error del segundo ambiente basado en RTX 2080Ti puede ser muy grande, sigue siendo un mejor resultado que el proporcionado por los otros ambientes haciendo de este un claro vencedor en cuestión de

rendimiento.

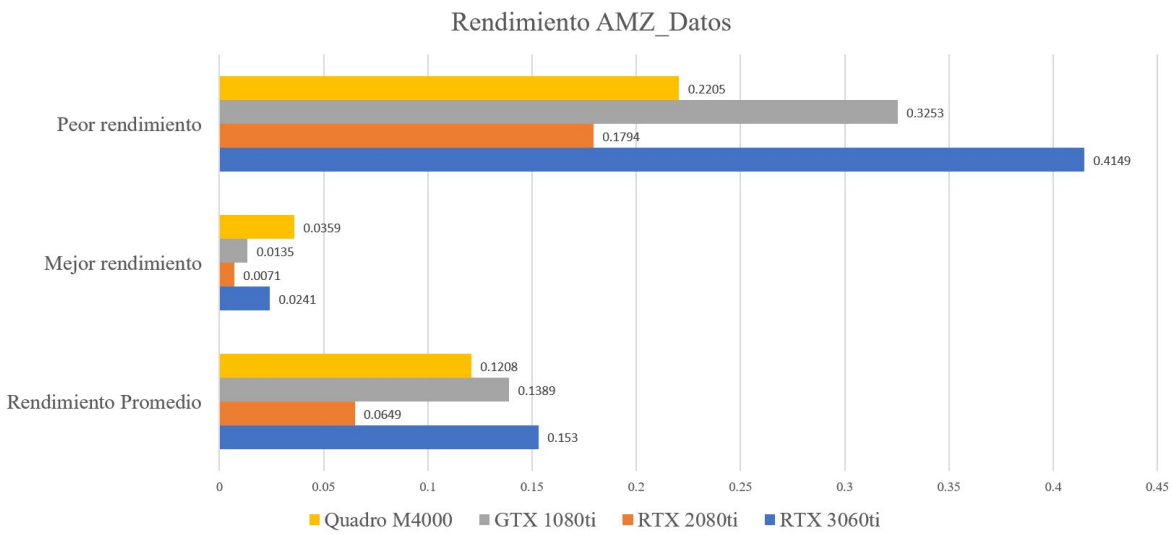

Figura 5.5: Análisis comparativo en rendimiento del conjunto de datos AMZ_Datos para la tarea de regresión

Como se ha mencionado ya anteriormente, no se debe descuidar los tiempos de ejecución. Para tener un panorama más amplio de los resultados obtenidos se realiza también esta comparativa con los tiempos de ejecución del AutoML. Teniendo como base anterior la RTX 2080Ti siendo el mejor ambiente en cuestión de rendimiento, en la Figura 5.6 se observa que los resultados obtenidos son mucho más rápido en comparación a las tareas de clasificación de imágenes. Esto se debe principalmente a la complejidad de los cálculos matriciales realizados en las imágenes, mientras que en las tareas de regresión suelen ser vectores de información los que se están procesando, siendo estos en general más rápidos. Ahora si bien se puede decir que las tareas regresión son más rápidas, se puede observar una tendencia entre los distintos ambientes, el segundo ambiente por ejemplo cuenta con un tiempo de 6 minutos obteniendo un segundo lugar con respecto a los 5 minutos que tarda en ejecutar todo el proceso, el primer ambiente, aclarando que se están tomando los mejores tiempos. Pero en caso de que se observe la frecuencia con la que se obtienen de manera rápida estos modelos se puede obtener que con 9 minutos de valor promedio la RTX 2080Ti cuenta con mejor tiempo que el

segundo y tercer ambiente, mientras que el primer ambiente cuenta con una irregularidad que le lleva a 34 minutos aproximados de media en obtener sus modelos. Así mismo, se puede concluir que la RTX 2080Ti proporciona la mejor opción Rendimiento/Tiempo.

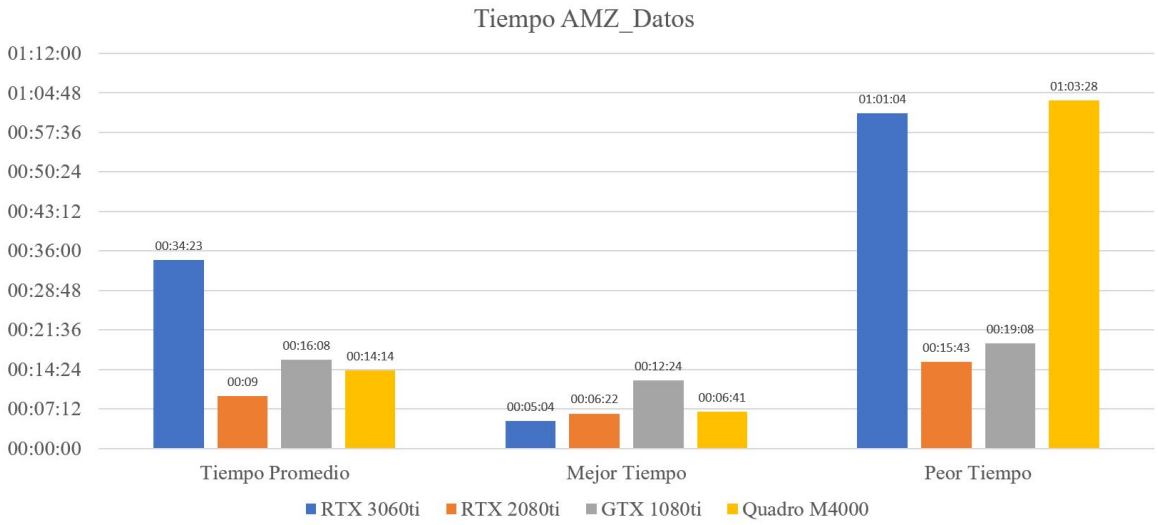

Figura 5.6: Análisis comparativo en tiempo del conjunto de datos AMZ_Datos para la tarea de regresión

La figura 5.7 se muestra la gráfica sobre el rendimiento obtenido en los experimentos realizados con el conjunto de datos BTC_datos_historicos. Los puntos claves que se están analizando son el peor rendimiento, el mejor y el rendimiento promedio. Teniendo una relación de 0.0118 de error promedio, se determinó que el primer ambiente, el cual utilizo RTX 3060Ti genera los resultados más estables. Así también se puede observar que el mejor rendimiento se obtiene con el segundo ambiente, el cual utiliza RTX 2080Ti. Aunque se puntualiza que el segundo lugar sé encuentra el primer ambiente, el cual ya nos está ofreciendo los resultados más estables proporcionados por los modelos generados. Por lo que el primer ambiente está ofreciendo buenos resultados, como para determinar que está siendo el mejor de los distintos ambientes.

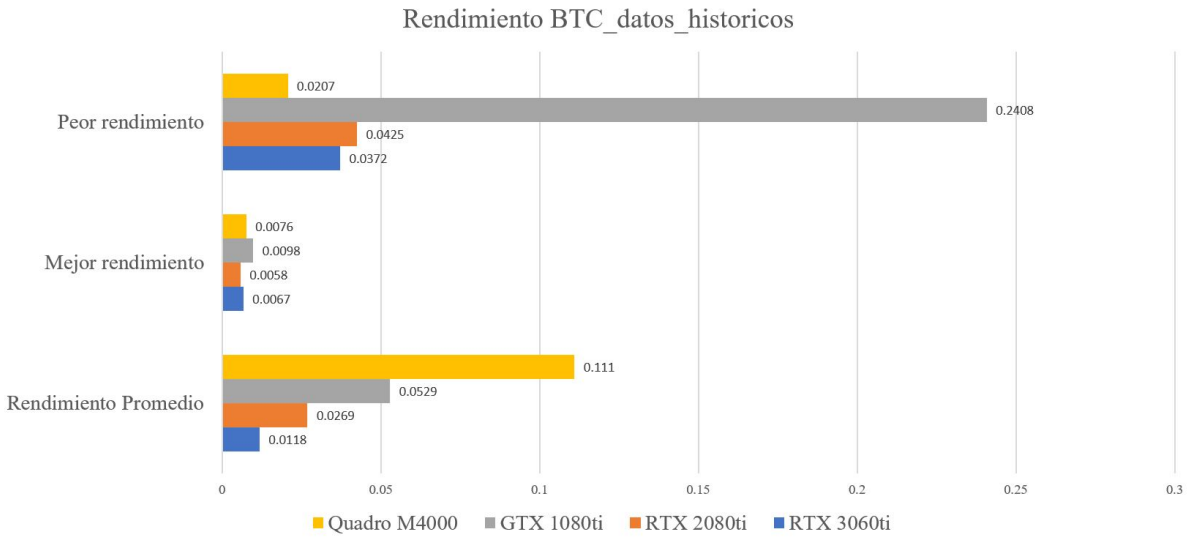

Figura 5.7: Análisis comparativo en rendimiento del conjunto de datos BTC_datos_historicos para la tarea de regresión

Ahora, si bien, se tiene que los dos mejores ambientes fueron el primero y el segundo en cuestión de rendimiento, el factor que determinara cuál ofrece los mejores beneficios es el tiempo, esta información estará representada en la Figura 5.8. El segundo ambiente distribuido, ofrece tiempos de ejecución más bajos y estables con un registro de 4 minutos aproximadamente. En cambio, el primer ambiente ofrece unos resultados más elevados, siendo el tercer ambiente más rápido en generar el modelo optimizado con un registro de 29 minutos en promedio. Por tanto, aunque se observe el mejor tiempo y se reduzca el tiempo registrado a 3 minutos y medio aproximados, esto no compensa la gran diferencia de los tiempos promedios, además de observar en la gráfica que el peor tiempo fue obtenido por el primer ambiente con una hora promedio. Con esto se determina finalmente que la mejor opción en este caso es el segundo ambiente.

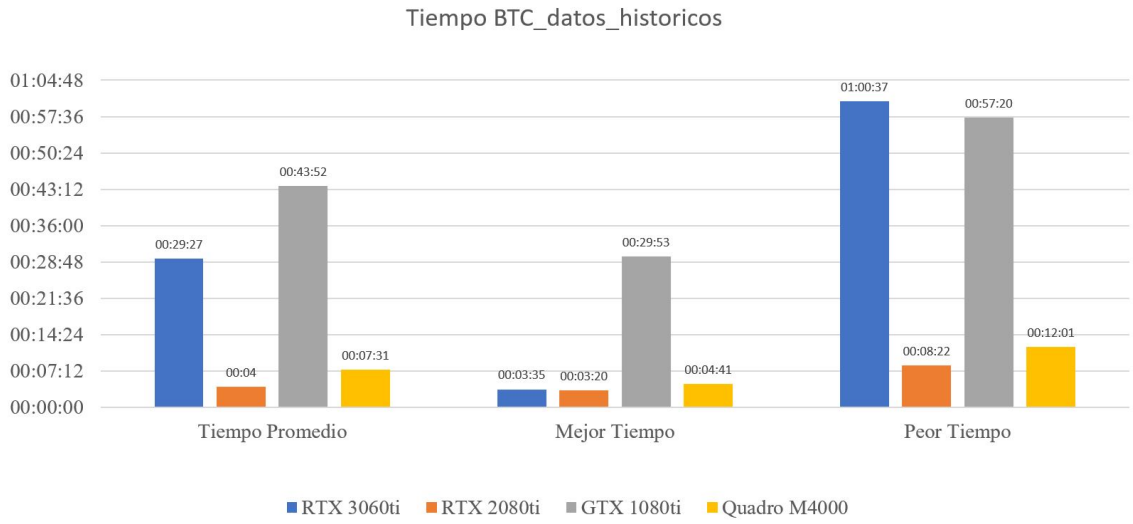

Figura 5.8: Análisis comparativo en tiempo del conjunto de datos BTC_datos_historicos para la tarea de regresión

Finalmente, analizaremos el último conjunto de datos para la tarea de regresión Clima_culiacan. La Figura 5.9 nos ilustra los resultados de la experimentación agrupados de manera que se pueda identificar de mejor manera el rendimiento obtenido para este conjunto de datos. Primero se hace hincapié en el rendimiento promedio, porque esto determina cuál es la configuración que genera un rendimiento estable más positivo que sus competidores, en este caso hay una variación mínima entre la primera, tercera y cuarta configuración. En cambio, el segundo ambiente da unos resultados muy poco favorables en comparación a los demás ambientes. Ahora se debe analizar también otros factores como el mejor y peor rendimiento, donde se puede observar que también tenemos una variación demasiado pequeña de 0.0001 entre los ambientes anteriormente mencionados. Por lo tanto, se concluyo de esta parte que el determinar el mejor modelo será decidido por los tiempos.

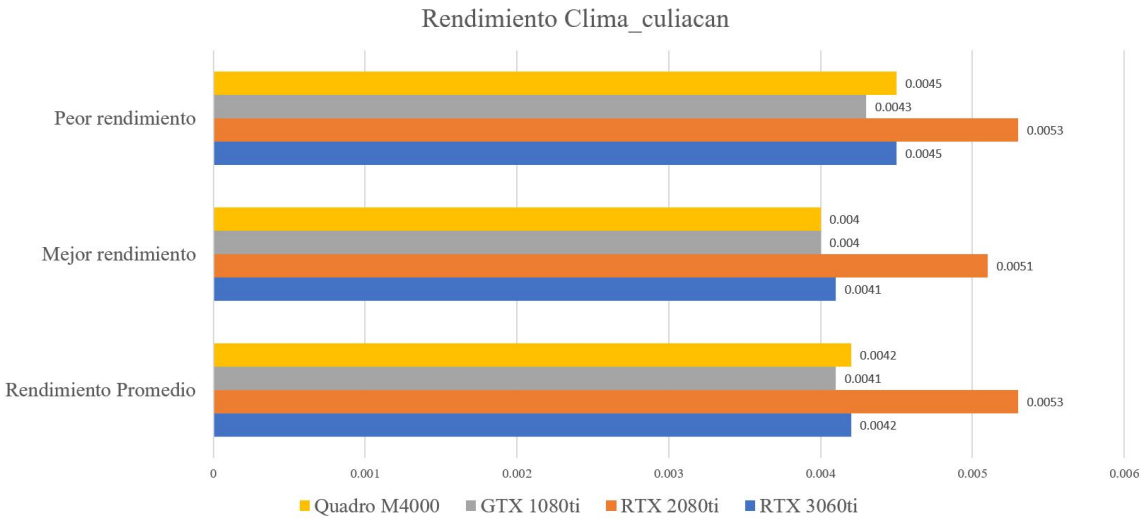

Figura 5.9: Análisis comparativo en rendimiento del conjunto de datos Clima_Culiacan para la tarea de regresión

En la Figura 5.10 se observa la gráfica que muestra los tiempos que tardan en generar los modelos. Analizando cada uno de las partes de la gráfica, se observa que los tiempos resultantes, determinan que el cuarto ambiente, el cual utiliza tarjetas de video Series Quadro M4000, tiene los mejores resultados en tiempo. En tiempo promedio se ofrece un tiempo de 20 minutos aproximados, su mejor tiempo es de 13 minutos y un resultado negativo máximo de 33 minutos aproximado. La ventaja clara, con respecto al siguiente puesto, permite seleccionarlo como la mejor opción para este conjunto de datos. Si bien el primer ambiente es el claro ganador, el segundo lugar es el primer ambiente el cual proporciona unos resultados de 53 minutos promedio, su mejor tiempo siendo de 26 minutos y el peor de los tiempos 1 hora con 11 minutos, siendo estos tiempos la segunda mejor opción en cuestión de tiempo y rendimiento ofrecido por cada ambiente distribuido.

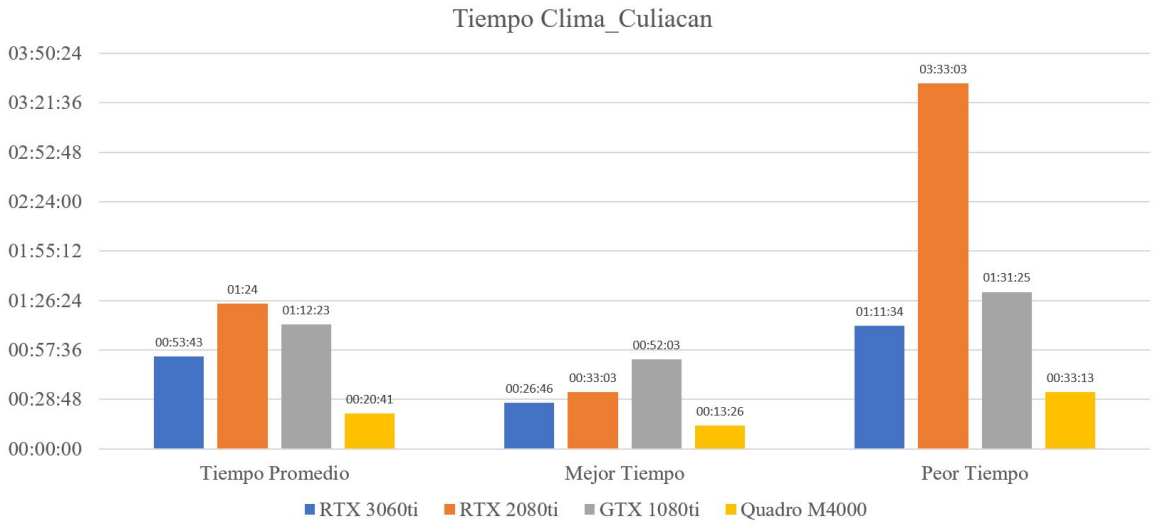

Figura 5.10: Análisis comparativo en tiempo del conjunto de datos Clima_Culiacan para la tarea de regresión

5.6. Evaluación de costos

Durante esta sección se mostrará los cálculos realizados sobre el dinero necesario en la inversión de cada una de las configuraciones. Los componentes seleccionados fueron valorados en función de la importancia que tienen a la hora de realizar los procesos de modelos de machine learning, estos componentes son la GPU, CPU y memoria RAM. Al final se hará un cálculo de la suma total de los costos de cada configuración para tener una valoración general y específica sobre los costos generados. La finalidad con la que se realizó este análisis es proporcionar a las organizaciones mejorar la decisión que tomen en función del hardware con el que cuenten o el presupuesto que manejen con respecto a sus necesidades. La Tabla 5.17 se muestran los presupuestos obtenidos con base en los precios de lista en el mercado actual.

Análisis de costos de cada ambiente				
Ambiente distribuido	Costo Total	Costo GPU/cu	Memoria RAM 2x8GB	CPU Intel I7
Configuración 1	$78,036	$7,980	$2,499	$8,500
Configuración 2	$48,960	$19,980	$2,499	$8,500
Configuración 3	$36,960	$13,980	$2,050	$7,650
Configuración 4	$98,908	$15,820	$2,325	$6,500

Tabla 5.17: Tabla comparativa de costos

En la tabla 5.11 se ilustra la relación que tiene el costo total de cada ambiente con respecto a cada uno de las distintas configuraciones implementadas. De esta manera se puede observar que la configuración 4 presenta los costos más elevados con un precio de $98,908, siendo esto último considerablemente mayor que el segundo y tercer ambiente. La tercera configuración presenta los costos más bajos con un precio de $36,960 en el costo total durante la implementación, la configuración que más cercana a este rango de precio es la que presenta la configuración 1 con un coste total de $48,960.

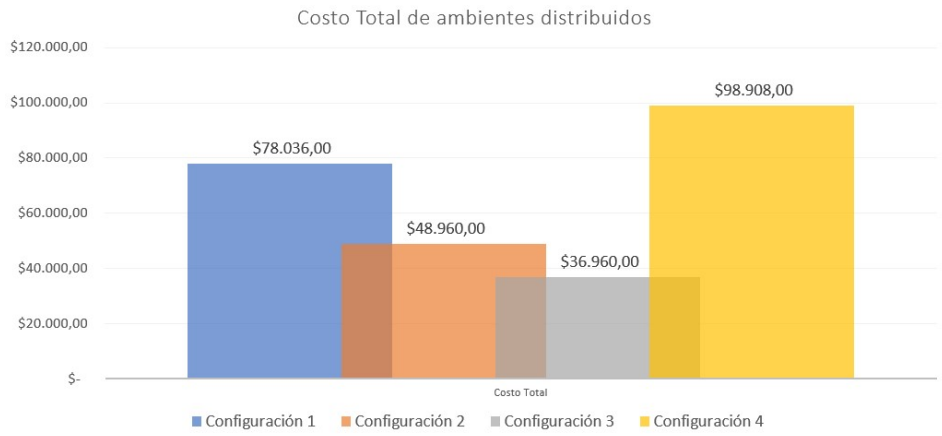

Figura 5.11: Análisis comparativo del costo total de las distintas configuraciones distribuidas

Capítulo 6

Conclusiones

En esta sección se describen una serie de conclusiones obtenidas con base en los resultados del proyecto. A su vez, se desarrollarán también una serie de aportaciones realizadas y los trabajos a futuro que se pueden implementar para mejorar este proyecto.

6.1. Conclusiones del trabajo

El área de aprendizaje máquina es un área de interés para las distintas organizaciones como lo son: académicas, investigación, empresas enfocadas en el desarrollo e innovación, así como la implementación industrial. Esto se debe principalmente a los beneficios que se han obtenido al implementar estas técnicas en múltiples áreas de estudios tales como el procesamiento de imágenes, inteligencia de negocios, la aplicación de series de tiempo para la predicción meteorológicas, ciencia de datos, procesamiento de lenguaje natural.

Aun teniendo los logros obtenidos mediante el aprendizaje máquina, estos necesitan de una gran cantidad de esfuerzo en especialistas y una inversión en hardware demasiado elevado, debido a esto se desarrollaron los sistemas de AutoML los cuales permiten que personas con baja expertis puedan resolver problemas de aprendizaje máquina, aun así solo soluciono uno de los dos problemas que se plantean, ya que los costos de un hardware especializados para implementar este tipo de sistemas sigue siendo elevando haciéndolo inaccesibles para el usuario común además de muchas otras organizaciones que no cuentan con un alto presupuesto.

Sin embargo, se comenzó a buscar soluciones a estos aspectos de los costos y la necesidad

de hardware especializados. Mediante la aplicación de sistemas distribuidos en los sistemas de AutoML se consiguió reducir los costos considerablemente, ya que es un sistema que originalmente se diseñó para funcionar de manera centralizada, comenzaron a aparecer diseños de sistemas de AutoML que se pueden implementar de manera distribuida reduciendo los tiempos y costos de implementación. También permitió que se pudieran utilizar equipos menos especializados, haciéndolos más accesibles para empresas u organizaciones pequeñas, así como al público en general.

Los diseños utilizados durante el proyecto fueron implementados en 5 distintos conjuntos de datos, 2 para la tarea de clasificación y 3 para regresión. Se obtuvieron resultados satisfactorios si se comparan con el estado del arte. La implementación adecuada de un diseño arquitectónico de hardware ha creado un flujo adecuado de trabajo durante la comunicación del ambiente distribuido. Esto tuvo como consecuencia que los resultados se obtuvieran en una cantidad de tiempo adecuada para cada tarea realizada. Así mismo, se puede determinar que la implementación del modelo arquitectónico maestro-esclavo, combinado con un correcto diseño e implementación del hardware, permite que los beneficios obtenidos por los sistemas de AutoML crezcan con respecto al uso del equipo de cómputo sin considerar sus características.

El proyecto presenta un análisis sobre la relación costo/beneficio que puede tener cada una de las configuraciones, con respecto a tres puntos principales que son el tiempo, dinero y rendimiento. Ahora bien, se puede concluir que las necesidades del proyecto que se realiza determinara cuál será la mejor configuración con base a costos/beneficio.

En el caso de las tareas de clasificación, se puede determinar mediante las Tablas 5.2 y 5.4 que si nuestro objetivo es tener los resultados en el menor tiempo posible, sin tomar otros aspectos como los costos de hardware, la primera configuración será definitivamente la mejor opción. En cambio, si se opta por rendimiento, al observar las Tablas 5.1 y 5.3 se selecciona al segundo ambiente. Finalmente, si se determina como punto clave los costos, se puede hacer una selección de la configuración con base al presupuesto del proyecto, de esta manera se puede buscar en un rango específico de dinero cuál es la que ofrece mejor rendimiento y/o tiempo.

Continuando ahora con las tareas de Regresión, se concluyó, basado en las Tablas 5.5, 5.7

y 5.9 que los resultados obtenidos pueden ser un poco más variables al utilizar conjuntos de datos de regresión. En cuestión de rendimiento se determina que el segundo ambiente tiene unos resultados satisfactorios, sin embargo, si se observan los resultados del conjunto de datos Clima_culiacan se debe puntualizar que no obtuvo el mejor resultado en comparación con las otras configuraciones, aun así la diferencia puede ser tan poco significativa que la segunda configuración sigue siendo la mejor opción de manera general.

Ahora, si bien ya se habló del rendimiento, el tiempo mostrado en las Tablas 5.6, 5.8 y 5.10 sugiere que hay variedad en los resultados, por lo que determinar cuál es la mejor opción puede ser un poco más complejo si solo se utiliza este parámetro como única referencia, las dos configuraciones que entregaron los mejores resultados fueron la segunda y el cuarto ambiente. Para llegar a una conclusión satisfactoria se puede observar en la Tabla 5.11 que el cuarto ambiente cuenta con un presupuesto de casi el doble de lo que se necesita para el segundo ambiente, por lo que se puede concluir que la configuración 2 es la mejor opción de gracias al análisis realizado.

De manera general, el desarrollo de este proyecto permite que se optimice el presupuesto que se designa para los proyectos de AutoML, el generar un análisis de resultados extenso en función de más de una configuración, permite obtener múltiples opciones a la hora de elegir que camino se debe tomar para solucionar un problema de machine learning. Como es en el caso de este proyecto, donde se vio que el cuarto ambiente tenía unos altos costos y unos resultados muy pobres de tiempo y rendimiento, por lo que se volvió una opción poco viable en su desarrollo, así como el tercer ambiente, el cual se destaca por ser el que menos recursos necesita para su implementación, generando buenos resultados, aunque no se destaque realmente en ninguna de las pruebas realizadas.

Finalmente, se hace mención de como, mediante la exploración de los distintos diseños de hardware y los múltiples experimentos realizados con el AutoML en cada una de las configuraciones, se pudo concluir que las configuraciones que dedicaban un hardware específico al nodo maestro para gestionar el flujo de trabajo, tenían una mayor constancia en los tiempos que tardaba en ejecutar el sistema.

Esto se debe principalmente que en los diseños donde el hardware estaba todo conectado dentro de una misma computadora, los recursos de procesamientos necesarios para cada nodo

esclavo eran compartidos entre sí junto al nodo maestro, teniendo que la memoria cache almacenada durante cada experimento se fuera acumulando, teniendo que esforzarse la CPU cada vez más en realizar los procesos y teniendo resultados más inconsistentes en comparación al diseño de cinco computadoras donde una era dedicada exclusivamente al nodo maestro y las otras cuatro a cada uno de los nodos esclavos utilizando su propia memoria RAM y GPU durante el proceso. De esta manera, al obtener esta información podemos determinar de mejor manera cuál será la mejor configuración de hardware en función del costo/beneficio.

6.2. Aportaciones

Una de las aportaciones principales de este proyecto fue el desarrollo e implementación de más de una configuración de hardware, permitiendo al usuario tener un panorama más amplio del comportamiento de distintos conjuntos de datos en un entorno de ambiente distribuido. Por otro lado, se puede destaca que el sistema de AutoML fue capaz de ser ejecutados en distintos entornos físicos, los cuales van de configuraciones especializadas a equipos de uso general. Puntualizando que la distribución de procesos permite que la aplicación de estos sistemas sea más accesible a todo tipo de organizaciones y publico en general. Además, otra de los aportes fue el determinar que personas con poco conocimiento sobre el manejo de machine learning, es capaz de ejecutar sistema de AutoML, para proporcionar soluciones a las distintas problemáticas que aborde de machine learning. También durante el análisis de los resultados obtenidos mediante la aplicación de distintas configuraciones se logró obtener un panorama más amplio sobre el diseño de hardware y como este ayuda a plantear mejores ambientes distribuidos, los cuales junto a la utilización del AutoML presentan un mayor costo/beneficio y una reducción del tiempo con respecto a los sistemas clásicos los cuales se basan en métodos centralizados como los mostrados en la literatura.

6.3. Trabajo futuro

El presente trabajo está realizado de manera que se puedan realizar cambio de manera sencilla. Una de las mejoras, es el aumento de múltiples configuraciones, teniendo de esta manera un mayor rango de opciones en la implementación del hardware, así como la aplicación del AutoML en servicios de la nube para tener un análisis más amplio sobre los resultados, como también una comparación entre un entorno físico y la nube.

Para la parte de comunicación se podrían explorar la implementación de otros tipos de sistemas distribuidos tales como clustering y/o peer-to-peer. Además, como parte importante de mejorar a la comunicación, se podría crear una un punto de datos específico, con cada uno de los conjuntos de datos que se van a utilizar, con la capacidad de ir ampliando para así tener un mejor control sobre los datos que se estén utilizando, minimizando errores entre los nodos.

También se puede explorar distintos diseños de hardware, en los cuales se contemple la implementación de múltiples GPU en de manera que trabajen en paralelo o de manera serial, observando el resultado que estos pueden ofrecer con respecto al funcionamiento normal que se les dan en una computadora.

Bibliografía

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (page 34).

Abd Elrahman, A., El Helw, M., Elshawi, R. & Sakr, S. (2020). D-SmartML: A Distributed Automated Machine Learning Framework. *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 1215-1218 (pages 38, 41).

Alaa, A. & Schaar, M. (2018). Autoprognosis: Automated clinical prognostic modeling via bayesian optimization with structured kernel learning. *International conference on machine learning*, 139-148 (pages 38, 41).

Andrews, G. R. (2020). *Foundations of multithreaded, parallel, and distributed programming*. Addison-Wesley. (Page 25).

Balaji, T., Annavarapu, C. S. R. & Bablani, A. (2021). Machine learning algorithms for social media analysis: A survey. *Computer Science Review*, *40*, 100395 (page 1).

Banks, M. (2008). *On the way to the web: The secret history of the internet and its founders*. Springer. (Page 25).

Bengio, I., Y.and Goodfellow & Courville, A. (2017). *Deep learning (Vol. 1)*. Massachusetts, USA: MIT press. (Page 14).

Benko, A. & Lányi, C. S. (2009). History of artificial intelligence. *Encyclopedia of Information Science and Technology, Second Edition* (pp. 1759-1762). IGI Global. (Page 11).

Bhbosale, S., Pujari, V. & Multani, Z. (2020). Advantages And Disadvantages Of Artificial Intellegence. *Aayushi International Interdisciplinary Research Journal*, 227-230 (page 2).

Bisong, E. (2019). Google AutoML: cloud vision. *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 581-598). Springer. (Page 40).

Blancarte, O. (2020). Introducción a la Arquitectura de Software: Un enfoque práctico. *Retrieved*, *11*(29), 2020 (page 28).

Bose, I. & Mahapatra, R. K. (2001). Business data mining—a machine learning perspective. *Information & management*, *39*(3), 211-225 (page 1).

Boulesteix, A.-L. & Schmid, M. (2014). Machine learning versus statistical modeling. *Biometrical Journal*, *56*(4), 588-593 (page 2).

Burkov, A. (2019). The hundred-page machine learning book (Vol. 1). (Page 15).

Burns, B. (2018). *Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services*. O'Reilly Media, Inc. (Page 31).

Conway, D. & White, J. (2012). *Machine learning for hackers*. "O'Reilly Media, Inc." (Page 13).

Coulouris, G., Dollimore, J., Kindberg, T. D. & G., B. (2012). *Distributed Systems: Concepts and Design Edition 5*. System, 2(11), 15. (Pages 24, 32).

Darwin™ 2.0, [Darwin™ 2.0,]. (2022). https://www.sparkcognition.com/product/Darwin. (Page 40)

Das, S., Dey, A., Pal, A. & Roy, N. (2015). Applications of artificial intelligence in machine learning: review and prospect. *International Journal of Computer Applications*, *115*(9) (page 1).

DataRobot's AI Cloud Platform, [DataRobot's AI Cloud Platform,]. (2012). https://www.datarobot.com/resources/end-to-end-ai-guide/thank-you/. (Page 39)

El Naqa, I. & Murphy, M. J. (2015). What is machine learning? *machine learning in radiation oncology* (pp. 3-11). Springer. (Pages 2, 13).

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M. & Smola, A. (2020). AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. (Page 40).

Fahle, S., Prinz, C. & Kuhlenkötter, B. (2020). Systematic review on machine learning (ML) methods for manufacturing processes – Identifying artificial intelligence (AI) methods for field application [53rd CIRP Conference on Manufacturing Systems 2020]. *Procedia CIRP*, *93*, 413-418. https://doi.org/https://doi.org/10.1016/j.procir.2020.04.109 (page 2)

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M. & Hutter, F. (2019). Auto-sklearn: efficient and robust automated machine learning. *Automated Machine Learning* (pp. 113-134). Springer, Cham. (Pages 36, 41).

Ghosh, S. (2006). *Distributed systems: an algorithmic approach*. Chapman; Hall/CRC. (Page 26).

Gijsbers, P. & Vanschoren, J. (2019). GAMA: genetic automated machine learning assistant. *Journal of Open Source Software*, *4*(33), 1132 (page 37).

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). (Page 20).

He, X., Zhao, K. & Chu, X. (2021). AutoML: A Survey of the State-of-the-Art. *Knowledge-Based Systems*, *212*, 106622 (page 22).

Hutter, F., Kotthoff, L. & Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature. (Pages 22-24).

Ionescu, V. M. (2015). The analysis of the performance of RabbitMQ and ActiveMQ. *2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER)*, 132-137 (page 34).

Joshi, A. V. (2020). *Machine learning and artificial intelligence*. Springer. (Pages 10, 11, 19).

Kevin Moore, L. M., Kin Fai Kan. (2018). TransmogrifAI [[[Web; accedio el 03-06-2022]]. (Page 37).

Khanzode, K. C. A. & Sarode, R. D. (2020). Advantages and Disadvantages of Artificial Intelligence and Machine Learning: A Literature Review. *International Journal of Library & Information Science (IJLIS)*, *9*(1), 3 (page 2).

Kohavi, R. & John, G. H. (1995). Automatic parameter selection by minimizing estimated error. *In Machine Learning Proceedings 1995 (pp. 304-312). Morgan Kaufmann.* (page 22).

Komer, B., Bergstra, J. & Eliasmith, C. (2014). Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. *ICML workshop on AutoML*, *9*, 50 (pages 36, 41).

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F. & Leyton-Brown, K. (2019). Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA. *Automated Machine Learning* (pp. 81-95). Springer, Cham. (Page 36).

LeDell, E. & Poirier, S. (2020). H2o automl: Scalable automatic machine learning. *Proceedings of the AutoML Workshop at ICML*, *2020* (pages 39, 41).

Li, Z., Guo, H., Wang, W. M., Guan, Y., Barenji, A. V., Huang, G. Q., McFall, K. S. & Chen, X. (2019). A blockchain and automl approach for open and automated customer service. *IEEE Transactions on Industrial Informatics*, *15*(6), 3642-3651 (page 38).

Liben-Nowell, D., Balakrishnan, H. & Karger, D. (2002). Analysis of the evolution of peer-to-peer systems. *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, 233-242 (page 26).

Magoulas, G. D. & Prentza, A. (1999). Machine learning in medical applications. *Advanced course on artificial intelligence*, 300-307 (page 1).

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, *9*, 381-386 (page 12).

McCarthy, J. (1998). What is artificial intelligence? (Pages 10, 11).

Mitchell, T. M. et al. (1997). Machine learning (pages 12, 13, 20).

Mohr, F., Wever, M. & Hüllermeier, E. (2018). ML-Plan: Automated machine learning via hierarchical planning. *Machine Learning*, *107*(8), 1495-1515 (page 37).

Mukunthu, D., Shah, P. & Tok, W. H. (2019). *Practical Automated Machine Learning on Azure: Using Azure Machine Learning to Quickly Build AI Solutions*. O'Reilly Media. (Page 40).

Naik, N. (2017, October). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *2017 IEEE international systems engineering symposium (ISSE) (pp. 1-7). IEEE* (page 32).

Narayan, S., Krishna, C. S., Mishra, V., Rai, A., Rai, H., Bharti, C., Sodhi, G. S., Gupta, A. & Singh, N. (2020). Ultron-AutoML: an open-source, distributed, scalable framework for efficient hyper-parameter optimization. *2020 IEEE International Conference on Big Data (Big Data)*, 1584-1593 (pages 38, 41).

Negnevitsky, M. & Intelligence, A. (2005). A guide to intelligent systems. *Artificial Intelligence* (pages 10, 11, 13).

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science Engineering, 9(3), 10-20.* (page 33).

Olson, R. S. & Moore, J. H. (2016). TPOT: A tree-based pipeline optimization tool for automating machine learning. *Workshop on automatic machine learning*, 66-74 (pages 36, 41).

Patterson, J. & Gibson, A. (2017). *Deep learning: A practitioner's approach*. .ºReilly Media, Inc." (Pages 15-21).

Scheppat, G. D. (2022). *A comparison of AutoML solutions ATM and AWS Sagemaker Autopilot*, Informatik. (Page 40).

Stuart R, P. N. (2010). *Artificial Intelligent A modern Aproach*. Pearson Inc. (Pages 11, 12).

Thornton, C., Hutter, F., Hoos, H. H. & Leyton-Brown, K. (2013). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 847-855 (pages 35, 41).

Wang, Q., Ming, Y., Jin, Z., Shen, Q., Liu, D., Smith, M. J., Veeramachaneni, K. & Qu, H. (2019). Atmseer: Increasing transparency and controllability in automated machine

learning. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1-12 (page 39).

Yang, C., Akimoto, Y., Kim, D. & Udell, M. (2018). Oboe: Collaborative filtering for automl initialization. *arXiv preprint arXiv:1808.03233* (page 37).

Zhou, J., Velichkevich, A., Prosvirov, K., Garg, A., Oshima, Y. & Dutta, D. (2019). Katib: A Distributed General AutoML Platform on Kubernetes. *2019 USENIX Conference on Operational Machine Learning (OpML 19)*, 55-57. https://www.usenix.org/conference/opml19/presentation/zhou (page 39)

Zoph, B. & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (page 2).