Análisis de herramientas de soporte al ciclo de vida de los sistemas de aprendizaje automático en tareas de clasificación

Alumno: Gianluca Sparvoli

Tutora técnica: Dra. Claudia Pons

Profesora de trabajo final: Dra. Marcela Samela

Trabajo Final de Carrera presentado para obtener el título de Licenciado en Gestión de Tecnología informática

Marzo 2023

Resumen

La inteligencia artificial es una tecnología que ha evolucionado de manera abrupta en los últimos años. Esta herramienta contiene un método especifico, llamado aprendizaje automático (o ML), que se lleva a cabo a través de varias tareas complejas, siendo uno de los factores más importantes de su ciclo la selección de un algoritmo especifico.

El presente trabajo sirvió de guía a la hora de seleccionar una herramienta de código abierto eficaz y precisa enfocada en el problema planteado de aprendizaje automático orientado en algoritmos de clasificación. En este sentido, se investigaron y seleccionaron las herramientas más relevantes, para luego adoptar un caso de estudio especifico y desarrollar todo su ciclo de vida. Una vez concluida esta fase, se analizaron las consecuencias y virtudes de cada una, para así confeccionar un manual detallado de recomendaciones y buenas prácticas asociado al tema expuesto, cumpliendo con el objetivo general del trabajo: comparar las herramientas más utilizadas de soporte al proceso de desarrollo de aplicaciones de aprendizaje automático automatizado en tareas de selección.

Los resultados obtenidos en esta investigación permitieron agregar valor a la tecnología desarrollada, dilucidando una de las primordiales dificultades cuando se enfrenta un problema de aprendizaje automático a la hora de seleccionar una herramienta gratuita automatizable para este tipo de obstáculos.

Palabras clave: algoritmos de clasificación, aprendizaje automático automatizado, herramienta de código abierto, inteligencia artificial

Abstract

Artificial intelligence is a technology that has evolved abruptly in recent years. This tool contains a specific method, called machine learning (or ML), which is carried out through various complex tasks, one of the most important factors in its cycle being the selection of a specific algorithm.

The present work served as a guide when selecting an efficient and precise open source tool focused on the problem of automatic learning oriented classification algorithms. In search of this, the most relevant ones were investigated and selected, to then adopt a specific case study and develop its entire life cycle. Once this phase was concluded, the consequences and virtues of each one were analyzed, in order to prepare a detailed manual of recommendations and good practices associated with the exposed topic, accomplish the general objective of the work: compare the most used tools to support the process of development of automated machine learning applications in selection tasks.

The results obtained in this research made it possible to add value to the developed technology, elucidating one of the main difficulties when facing an automatic learning problem when selecting a free tool that can be automated for this type of obstacle.

Keywords: artificial intelligence, automated machine learning, classification algorithms, open source tool

Reconocimientos

El presente trabajo y mi camino por esta carrera universitaria, si bien ha requerido de esfuerzo y mucha dedicación, no hubiese sido posible sin la cooperación desinteresada de cada una de las personas que me acompañaron en su recorrido y que han sido un soporte importante en momentos de angustia.

Antes que todo, agradezco a mis tutores, que con su amplia experiencia y conocimientos me orientaron al correcto desarrollo y culminación con éxito de este trabajo para la obtención de la Licenciatura en Gestión de Tecnología Informática, y, a través de ellos a la Universidad Abierta Interamericana: autoridades y docentes.

El trabajo realizado lo dedico con mucho cariño a mis padres, que fueron el sustento en todo momento para continuar pese a todos los obstáculos que se presentaron en el trayecto de esta investigación. También se la dedico a mis familiares, ya que, con su ejemplo y amor siempre me dieron esperanza y tuvieron fe en mí, y me encaminaron seguir con la propuesta investigativa; ellos han sido el sostén fundamental para lograr mis objetivos. A mis amigos que gracias a su apoyo moral me permitieron perseverar con empeño, dedicación y cariño, y a todos quienes contribuyeron con un granito de arena apoyándome para culminar con éxito la meta propuesta.

Luego de todas las frustraciones, éxitos, fracasos y años requeridos, este título es una meta personal que me planteé a mí mismo hace mucho tiempo atrás, no solo para crecer y dar un paso hacia adelante, sino para demostrar que con optimismo y perseverancia todo es posible. A riesgo de parecer orgulloso o egocéntrico, también me gustaría agradecerme a mi mismo, por no bajar los brazos, por seguir intentando sin mediar la circunstancia y por no ceder. Por esto y todo lo mencionado, gracias a todas esas personas que me apoyaron en los momentos en los que estaba demasiado cansado como para recordar mi objetivo.

Mil veces gracias.

Índice General

Índice de gráficos

Índice de tablas

1 – Introducción

1.1 Problemas y Soluciones

1.1.1 Identificación del Problema.

El mundo de la inteligencia artificial es muy amplio y es posible diferenciarlo dentro de cuatro categorizaciones, según Russell y Norvig (2004): sistemas que piensan como humanos, sistemas que actúan como humanos, sistemas que piensan racionalmente y sistemas que actúan racionalmente. Estas categorizaciones contienen dos escuelas de conocimiento, llamadas inteligencia artificial simbólica e inteligencia artificial no simbólica. Dentro de las escuelas de conocimiento no simbólicas, existe una rama fundamental, llamada aprendizaje automático. Esta se ha convertido en uno de los principales tópicos de la actualidad debido a que permite procesar enormes cantidades de datos de entrada y predecir su salida. Es decir que, distintas organizaciones, por ejemplo, podrán continuamente anteponerse a cambios en la industria de forma casi inmediata.

Crear una aplicación de aprendizaje automático o poner en funcionamiento un algoritmo de aprendizaje automático es un proceso continuamente iterativo y complejo. No se puede simplemente entrenar un modelo una vez y dejarlo inmutable, habrá cambios en los datos, las preferencias de los clientes evolucionan y surgirán competidores. Por lo tanto, el modelo necesita mantenerse actualizado cuando entre en producción. Nuestra finalidad principal es hacer a este modelo lo más automatizable posible.

El ciclo de vida del aprendizaje automático es continuo y elegir el algoritmo correcto de aprendizaje automático es solo uno de los pasos.

Por lo expuesto, se ha propuesto un trabajo de investigación, indagando acerca de estos tópicos y tratando de responder la siguiente pregunta:

¿Cuál es la herramienta de soporte más eficiente para el proceso de desarrollo de aplicaciones de aprendizaje automático automatizable en tareas de clasificación?

1.1.2 Planteamiento del tema de investigación.

Este trabajo, en el marco del trabajo final de carrera para la Licenciatura en Gestión de Tecnología Informática, de la Universidad Abierta Interamericana, perteneciente a la ciudad de Buenos Aires, Argentina, comprendido en el proyecto de investigación llamado “Técnicas de Inteligencia Artificial basadas en una integración de la lógica simbólica y no-simbólica”, dirigido por la Dra. Claudia Pons y organizado por el CAETI (Centro de Altos Estudios en Tecnología Informática) de la mencionada universidad, se centra en la búsqueda, implementación y análisis de las herramientas de código abierto más pertinentes para el proceso de perfeccionamiento y automatización de aplicaciones de aprendizaje automático en tareas de clasificación.

Se plantea indagar sobre las principales herramientas de aprendizaje automático automatizable (o AutoML) de código abierto en tareas de clasificación y seleccionar las tres más relevantes, para a continuación adoptar un caso de estudio específico y desarrollar todo su ciclo de vida utilizando los diferentes repertorios que están a disposición y que fueron seleccionados. Una vez concluida esta etapa de indagación e implementación, se procederá a analizar las consecuencias, para así confeccionar un manual detallado de recomendaciones y buenas prácticas asociado al tema expuesto.

Ahora bien, se ha realizado una investigación acerca de antecedentes y trabajos de este tipo de investigaciones, y se ha podido establecer que existen una infinidad de bibliotecas y herramientas disponibles para la implementación de modelos de AutoML.

En el presente informe, observaremos aquellas herramientas más famosas y utilizadas en el lenguaje de programación Python. Nuestra elección se justifica en que además de ser muy robusto y fácil de utilizar, Python es de muy rápida implementación y gratuito. Otra de sus ventajas, relacionadas al ámbito que nos compete, es que las mejores y más eficientes herramientas están desarrolladas en este lenguaje, por lo que lo consideramos el óptimo para el tipo de aplicación que estamos indagando.

1.2 Justificación del Tema

En la actualidad, existen una gran cantidad de herramientas de código abierto destinadas a ayudar a los científicos de datos a recorrer el ciclo de vida de las aplicaciones de aprendizaje automático. Estas a menudo están vinculadas directamente al lenguaje de programación que se está utilizando para dotar a la aplicación (Python, R, Java, etc.).

Los científicos de datos necesitan herramientas para la selección de algoritmos ya que varios modelos diferentes de aprendizaje automático pueden ser útiles para resolver problemas. La posibilidad de experimentar con diferentes algoritmos permitirá crear modelos que escalen y que posean mayor capacidad para predecir resultados. Estas herramientas también ofrecen soporte en otras actividades del ciclo de vida, tal como la evaluación del modelo y la preparación de los datos (Hurwitz y Kirsch, 2018).

Como se mencionó en el apartado anterior, el ciclo de aprendizaje automático es continuo y elegir el algoritmo correcto de aprendizaje automático es solo uno de los pasos. Los pasos del ciclo de aprendizaje automático son los detallados a continuación.

Un primer paso en el ciclo de aprendizaje automático es identificar qué datos vamos a usar y donde se encuentran las fuentes relevantes de estos datos. Además, a medida que se desarrolle el algoritmo de aprendizaje automático, los datos deberían aumentar para perfeccionar el sistema. A continuación, como segundo paso, los datos identificados deben ser preparados para resultar limpios, seguros y normalizados. Si se crea una aplicación de aprendizaje automático basada en datos inexactos, la misma fallará. Posteriormente, como tercer paso, se seleccionará el algoritmo de aprendizaje automático. Hay muchos algoritmos prediseñados disponibles y es posible que varios algoritmos de aprendizaje automático sean aplicables a los mismos datos. En nuestro caso en específico, nos enfocaremos en algoritmos de clasificación. El siguiente paso implica entrenar el algoritmo elegido para crear el modelo. Dependiendo del tipo de datos y del algoritmo, el entrenamiento puede ser de aprendizaje supervisado, no supervisado o semi supervisado. En nuestro caso en específico, será supervisado. A continuación, como quinto paso, se evaluarán los modelos entrenados para encontrar el algoritmo que tenga mejor rendimiento. Seguidamente, si se desea se podrán implementar los modelos resultantes en aplicaciones locales y en la nube. Por último, podrán hacerse predicciones basadas en nuevos datos entrantes con los modelos resultantes y evaluar

la validez de esas predicciones. La información que se recopila al analizar la validez de las predicciones luego retroalimenta el ciclo de aprendizaje automático para ayudar a mejorar su precisión (Hurwitz y Kirsch, 2018).

En otro orden de ideas, existen una gran cantidad de librerías destinadas a la aplicación de modelos de AutoML, pero sólo algunas son lo necesariamente fiables y buenas a la hora de su implementación. Esto trae aparejado el problema de la elección de una específica, lo que demandará una enorme cantidad de tiempo y de recursos que podrían destinarse a otros fines. En consecuencia, el presente trabajo ayudará a aclarar uno de los principales inconvenientes a la hora de encarar un problema de aprendizaje automático automatizado en tareas de clasificación, facilitando buenas prácticas, guías y modelos acordes a la hora de seleccionar una herramienta gratuita para este tipo de complicaciones según el tipo de conjunto de datos seleccionado.

1.3 Objetivo del Trabajo Final

1.3.1 Objetivos generales.

En el presente trabajo, el objetivo general es comparar las herramientas más utilizadas de soporte al proceso de desarrollo de aplicaciones de aprendizaje automático automatizado en tareas de clasificación.

1.3.2 Objetivos específicos.

Los objetivos específicos de este trabajo se detallan a continuación.

Investigar el ciclo de vida de las aplicaciones de aprendizaje automático automatizado.

Analizar herramientas de aprendizaje automático automatizado en tareas de clasificación y seleccionar las tres más utilizadas y beneficiosas.

Entrenar modelos de aprendizaje automático automatizado utilizando cada una de las herramientas seleccionadas.

Elaborar un estudio comparativo de las herramientas con una lista de recomendaciones y buenas prácticas.

1.4 Nuestra Propuesta - Hipótesis

En el presente trabajo, la hipótesis es la siguiente: La utilización del software de código abierto AutoKeras, con Python como lenguaje de programación, permite automatizar un modelo de aprendizaje automático en tareas de clasificación de forma rápida, segura y eficiente.

1.5 Estructura General del Trabajo Final

En el capítulo 1, se desarrollará una breve introducción acerca de todo el trabajo, especificando la identificación del problema, planteamiento del tema de investigación, justificación del tema, objetivos (generales y específicos) e hipótesis.

En el capítulo 2, se llevará a cabo el marco teórico del trabajo, detallando cada uno de los apartados y conceptos necesarios para la realización del mismo

En el capítulo 3, se desarrollará la sección de sistemáticas, detallando categorías en que se sustenta el trabajo, población, técnicas de recolección de datos a emplear, técnicas de análisis de datos relevados y características del informe final

En el capítulo 4, se llevará a cabo el cronograma, medido en tiempos, y los recursos necesarios para realizar el trabajo

En el capítulo 5, se desarrollará la forma y los resultados a la hora de recolectar y analizar datos

En el capítulo 6, se llevará a cabo el grueso del trabajo, detallando que y como se realizó, así como los resultados puntuales pertinentes

En el capítulo 7, se desarrollarán las conclusiones finales del trabajo.

2 – Marco teórico

2.1 Introducción e historia

El concepto de inteligencia artificial general no es actual como se cree comúnmente, sino que el primer trabajo surgió en 1943 con Warren McCulloch y Walter Pitts. Estos autores indicaron que era posible que redes neuronales definidas de forma adecuada tuvieran la capacidad de aprender, y propusieron así "un modelo constituido por neuronas artificiales, en el que cada una de ellas se caracterizaba por estar «activada» o «desactivada»" (Russell y Norvig, 2004, p. 19), en el cual el estado de una neurona "se daba como respuesta a la estimulación producida por una cantidad suficiente de neuronas vecinas" (Ídem). Asimismo, Alan Turing, en su artículo publicado en 1950, fue el primero en modular una visión sobre esta tecnología, ya que "introdujo la prueba de Turing, el aprendizaje automático, los algoritmos genéricos y el aprendizaje por refuerzo" (Russell y Norvig, 2004, p. 20). Posteriormente, John McCarthy en 1956 fue quien le dio forma a este concepto, definiéndolo como "«Inteligencia Artificial. Quizá «racionalidad computacional» hubiese sido más adecuado, pero «IA» se ha mantenido" (Ídem). Sin embargo, no fue hasta 1987 que la IA se convirtió en una ciencia.

La utilización de metodologías mejoradas y marcos teóricos ha autorizado que este campo alcance un grado de conocimiento que ha permitido que ahora las redes neuronales se puedan comparar con otras técnicas similares de campos como la estadística, el reconocimiento de patrones y el aprendizaje automático, de forma que las técnicas más prometedoras pueden aplicarse a cualquier problema. Como resultado de estos desarrollos, la tecnología denominada "minería de datos" ha generado una nueva y vigorosa industria (Russell y Norvig, 2004, p. 30).

Como queda en evidencia, desde sus inicios a la actualidad, la inteligencia artificial ha avanzado a pasos agigantados. Algunos de los desarrollos más conocidos son, por ejemplo, los utilizados diariamente por los asistentes virtuales de los teléfonos celulares. Cada una de las aplicaciones de la tecnología ya mencionada genera una gran cantidad de datos que irán poco a poco delimitando las preferencias de cada usuario. Teniendo en cuenta esto, las empresas utilizan estos datos para conocer esas preferencias y poder predecir, por ejemplo, qué servicios son los que más se consumen. Todo esto es proporcionado gracias a la integración de dos tecnologías llamadas "ciencia de datos" y "aprendizaje automático". Esta última, tratada en

este trabajo, es la encargada de establecer patrones que, con una probabilidad muy alta, seguirán dándose en el futuro.

2.2 Antecedentes de trabajos similares

La investigación en el campo de aprendizaje automático automatizado (AutoML) se remonta a una década atrás e, incluso a nivel mundial, hay pocas publicaciones que abordan esta materia. Pese a ello, los trabajos publicados produjeron un cuerpo importante de conocimientos que le han brindado validez y reconocimiento como campo de investigación y desarrollo. Hasta ahora, la mayoría de estas investigaciones pueden caracterizarse como reseñas en las que se comparan estas herramientas, y se explicita un paso a paso de esta confrontación. Sin embargo, también hay antecedentes acerca de algunas herramientas de soporte comparando su eficiencia para el proceso de desarrollo de aplicaciones de AutoML en tareas de clasificación.

A lo largo de estos años, varias líneas de investigación han surgido en este campo, algunas más desarrolladas que otras. A continuación, se mencionan algunos de estos trabajos, que se vinculan estrechamente con la problemática que aquí abordamos.

Por una parte, situándonos en Argentina, un primer y único trabajo corresponde a Bender y Nicolet (2020), quienes realizaron una "Evaluación Comparativa de Herramientas AutoML de Código Abierto en Tareas de Regresión" para la Universidad del Salvador. En este trabajo se investigó acerca de aplicaciones y herramientas para referencias de AutoML de código abierto y se propuso un análisis comparativo de su eficacia utilizando conjuntos de datos públicos para algoritmos de regresión.

Por otra parte, en el ámbito internacional, se han encontrado varios artículos relevantes relacionados a la materia en cuestión.

El primer trabajo pertenece a Gijsbers et al. (2019). En él, se presenta un marco de referencia abierto, continuo y extensible que sigue las mejores prácticas y evita errores comunes a la hora de comparar sistemas de AutoML para tareas de clasificación.

Un segundo trabajo corresponde a Milutinovic et al. (2020), quienes llevaron a cabo un desarrollo de un sistema informático de código abierto en base a una descripción de prácticas fundamentales de evaluación de AutoML, permitiéndole a la comunidad probar hipótesis sobre el funcionamiento interno de los sistemas de esta tecnología.

Un tercer trabajo pertenece a Hanussek et al. (2020), quienes compararon y comprobaron como diversas tecnologías de AutoML con diferentes conjuntos de datos pueden obtener un resultado casi igual o, en algunos casos superador, al de los mejores científicos de datos humanos del mundo.

Un cuarto trabajo pertenece a Ferreira et al. (2021), quienes llevaron a cabo una comparación de rendimiento y efectividad de algunas de las diferentes herramientas de código abierto y de código cerrado de AutoML basadas en tres escenarios, aprendizaje automático, aprendizaje profundo y la librería XGBoost.

Estos antecedentes se relacionan con la investigación en curso de manera directa, ya que esta propone el descubrimiento de las más eficaces herramientas de aprendizaje automático automatizado, pero sobre el otro tipo de algoritmos que permite el aprendizaje supervisado (explicado en el apartado 2.4.1 del presente documento). Además, estos trabajos explican pasos sistemáticos a llevar a cabo para la evaluación de los mismos y algunas comparaciones similares para tomar como ejemplo.

2.3 Inteligencia Artificial

A lo largo del tiempo, la humanidad ha tratado de descifrar la manera en la que se piensa, se razona y se aprende, dando lugar a nuevas disciplinas llamadas ciencias cognitivas. Entre ellas se encuentran la inteligencia artificial y las ciencias computacionales que comparten un enfoque conocido como “paradigma del procesamiento de información” según el cual la mente humana es un sistema procesador de símbolos.

La inteligencia artificial está diseñada para imitar la mente humana a la hora de razonar y tomar decisiones, es decir, intenta seleccionar las mejores opciones para solucionar un problema complejo, mediante algoritmos y datos. A partir de aquí, se puede concluir, entonces, que la inteligencia artificial depende de dos pilares, un agente definido como cualquier ente

capaz de recibir entradas de su entorno, procesarlas y proporcionar salidas al respecto, y una racionalidad comprendida como la capacidad para actuar y pensar, mejorando continuamente, para alcanzar un objetivo. Esta combinación de agente y racionalidad fue definida por Russell y Norvig (2004) como agentes racionales computacionales.

En otro orden de ideas, la mencionada inteligencia artificial puede ser diferenciada dentro de 4 categorizaciones fundamentales interconectadas entre sí, descriptas a continuación.

Figura 1

Enfoques de la inteligencia artificial

A. Sistemas que piensan como humanos	B. Sistemas que piensan racionalmente
C. Sistemas que actúan como humanos	D. Sistemas que actúan racionalmente

Nota. Adaptado de *Inteligencia artificial. Un enfoque moderno. Segunda edición* (p. 30), por Russell y Norvig, 2004, Pearson Prentice Hall.

En la figura 1, podemos observar una diferenciación en torno a dos comportamientos, donde la A y la B orientan al razonamiento, en tanto las de la demás indican la conducta. Se evidencia que, además de los comportamientos, hay una diferenciación hacia referencias humanas y racionales.

> El enfoque centrado en el comportamiento humano debe ser una ciencia empírica, que incluya hipótesis y confirmaciones mediante experimentos. El enfoque racional implica una combinación de matemáticas e ingeniería. Cada grupo al mismo tiempo ha ignorado y ha ayudado al otro (Russell y Norvig, 2004, p. 30).

Una vez definida esta concepción, se nos hace preciso aclarar dos niveles de conocimiento acerca de la inteligencia artificial, conocidos como inteligencia artificial simbólica y no simbólica.

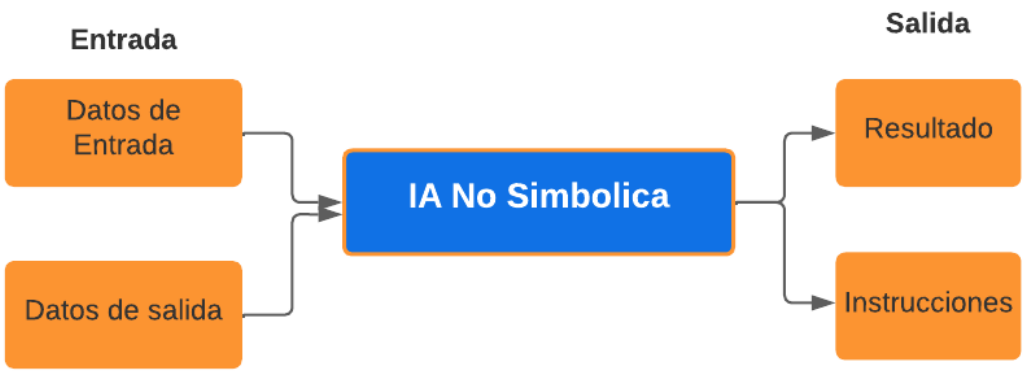
En el primero, se da un paradigma orientado a la interpretación, donde un sistema, sigue e interpreta instrucciones ya programadas para poder obtener resultados basados en el razonamiento lógico. Este sistema contará con datos y un mecanismo de deducción ya implementado. En la figura que se presenta a continuación, se puede observar el flujo de trabajo de este paradigma.

Figura 2
Inteligencia artificial simbólica

En el segundo nivel, la inteligencia artificial no simbólica, el modelo de conocimiento está orientado al aprendizaje. Se trata de un sistema al cual se le otorgan datos de entrada y una enorme cantidad de datos de salida (también llamados ejemplos), pero no las reglas a seguir. Este, construirá su propio algoritmo (o instrucciones), dando un resultado acorde a los datos otorgados. Un ejemplo de esto es el paradigma llamado aprendizaje automático (que se verá más en profundidad en el apartado 2.4). En la siguiente figura, se puede observar el flujo de trabajo de este paradigma.

Figura 3

Inteligencia artificial no simbólica



2.4 Aprendizaje Automático

El aprendizaje automático se ha convertido, en los últimos tiempos, en uno de los tópicos más importantes en organizaciones con necesidades de innovación. De hecho, se considera que, si en este contexto se utilizan las fuentes de datos más precisas y constantes, se conseguirá predecir el futuro de ese sistema. Según Hurwitz y Kirsch (2018), el aprendizaje automático se puede definir como una forma de inteligencia artificial que permite que un sistema aprenda a partir de datos en lugar de programación secuencial. Este utiliza una variedad de algoritmos que aprende de forma repetida e incansable de los datos para mejorar, describirlos y predecir resultados. En otras palabras, a través de algoritmos, concede la capacidad de localizar patrones en datos masivos y elaborar predicciones.

Ahora bien, a la hora de analizar datos existen dos universos diferentes: el del análisis de texto y el del análisis de imágenes. En el primer caso, se utilizan algoritmos que procesan datos de texto a escala, categorizando un documento de texto en uno de un conjunto predefinido de grupos. En muchos problemas de clasificación de grupos, esta categorización se basa principalmente en las palabras clave del texto (Google Developers, s.f.-e, 2022). Por otra parte, en el caso del análisis de las imágenes, en su gran mayoría, se necesita una enorme cantidad de procesamiento y transformación antes de que este modelo de inteligencia artificial pueda ser siquiera entrenado. Las imágenes, por ejemplo, deben ser normalizadas en cuanto a su tamaño

y dimensiones o modificadas en cuanto a su color. Todo ello es sumamente importante y, si no se logra de manera correcta, el procesamiento se volverá mucho más arduo y tomará más tiempo, recursos y costos (Google Developers, s.f.-f, 2022).

Un modelo se representa mediante sus parámetros e hiperparámetros relacionados de manera directa. No obstante, según Elgendy (2020) no se deben confundir. Los hiperparámetros son las variables que establece y ajusta una persona de forma manual. Estos definen la estructura de la red y determinan cómo esta se entrena. Básicamente, cualquier cosa sobre la que un individuo decida valores o elija su configuración antes de que comience el entrenamiento y cuyos valores o configuración permanezcan iguales cuando este finalice es un hiperparámetro. Por otra parte, los parámetros son las variables ajustables por la red que se optimizan automáticamente durante el proceso para producir el mínimo error. Es decir, el proceso de entrenamiento involucra elegir los hiperparámetros óptimos que el algoritmo de aprendizaje pueda usar para perfeccionar los parámetros que orienten correctamente las entradas de datos con sus respectivos grupos.

En su obra Goodfellow et al. (2016) formularon que el entrenamiento del modelo generalmente comienza con la inicialización de los parámetros (valores aleatorios). A medida que este avanza, los valores iniciales se actualizan continuamente, pero los valores de los hiperparámetros establecidos por el usuario permanecen sin cambios. Al final del proceso de aprendizaje, los parámetros son los que constituyen el modelo mismo.

Para optimizar y predecir resultados, el aprendizaje automático utiliza tres tipos de técnicas llamadas aprendizaje supervisado, no supervisado y semi supervisado (secciones 2.4.1, 2.4.2 y 2.4.3 del presente documento).

2.4.1 Aprendizaje Supervisado

El aprendizaje supervisado es una técnica particular del aprendizaje automático (sección 2.4). Este se lleva a cabo cuando a un algoritmo de aprendizaje automático se le brindan, por un lado, los datos de entrada y de salida, y, por otro, los llamados “datos de entrenamiento”, con la finalidad de que este pueda encontrar conexiones entre los elementos y predecir esos resultados. El aprendizaje supervisado soluciona dificultades familiares y utiliza

un conjunto de datos etiquetados para entrenar un algoritmo en tareas específicas (Russell y Norvig, 2004). Google Developers (s.f.-a, 2022) brinda en su página una analogía con un escenario cotidiano: un estudiante aprende material nuevo al estudiar exámenes de prueba con preguntas y respuestas, seguidamente, una vez que se capacitó de forma correcta, estará preparado para tomar un nuevo examen con una probabilidad muy alta de obtener una calificación satisfactoria.

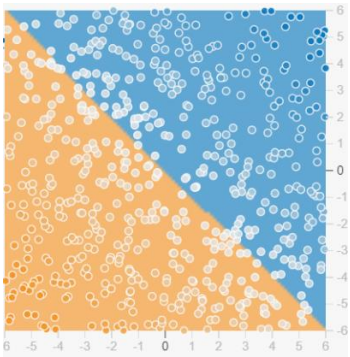
En este sentido es fundamental mencionar que en este tipo de aprendizaje se destacan dos algoritmos o tipos entrenamientos llamados clasificación y regresión (secciones 2.4.1.1 y 2.4.1.2).

2.4.1.1 Algoritmo de regresión

Dentro de la categorización de aprendizaje supervisado, existe el algoritmo de regresión. Este se logra mediante la determinación de una recta para proporcionar la tendencia de un conjunto de datos. Su objetivo primordial es establecer una regla para la relación entre un cierto número de características y una variable continua. Lo que se espera como resultado es un número. No se ubica en un grupo, sino que retorna un valor específico (Google Developers, s.f.-a, 2022). En la figura a continuación, se puede observar un ejemplo del resultado de este algoritmo.

Figura 4

Algoritmo de regresión



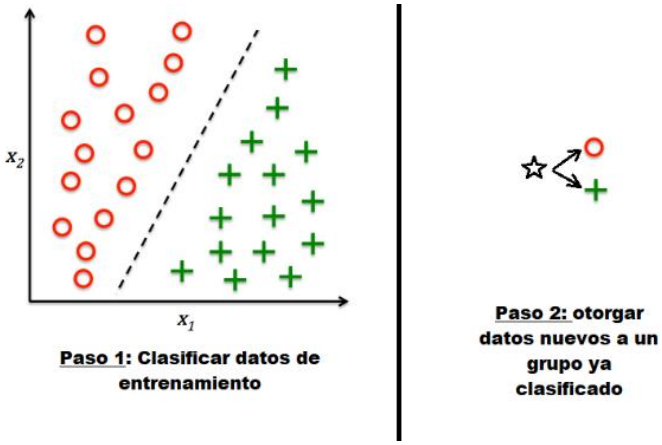
Nota. Ejemplo de Algoritmo de regresión.

2.4.1.2 Algoritmo de clasificación

Otro tipo de algoritmo dentro del aprendizaje supervisado es el que se denomina algoritmo de clasificación. Este se logra cuando el algoritmo nos predice a qué grupo pertenece el elemento en estudio. Este algoritmo encuentra patrones en los datos que le brindamos de entrenamiento y los clasifica en grupos, luego, compara los nuevos datos otorgados y los sitúa en uno de esos grupos. La variable por predecir es un conjunto de estados discretos o categóricos. De esta manera, el algoritmo de clasificación puede ser binario (por ejemplo: si / no; azul / rojo, etc.) o múltiple (por ejemplo: producto 1/ producto 2, producto 3, etc.) (Google Developers, s.f.-a, 2022). En la figura a continuación, se puede observar el flujo de trabajo de este algoritmo.

Figura 5

Algoritmo de clasificación



2.4.2 Aprendizaje No Supervisado

En contra posición del aprendizaje supervisado del apartado 2.4.1, existe el aprendizaje no supervisado, que también se presenta como técnica particular del aprendizaje automático (sección 2.4). Este radica en aprender a partir de patrones de entradas para los que no se especifican los valores de sus salidas. Se utiliza para identificar nuevos patrones y detectar anomalías. Los datos que se introducen en los algoritmos de aprendizaje no supervisados no están etiquetados. El algoritmo (o

modelo) intenta dar sentido a los datos por sí mismos mediante la búsqueda de características y patrones; “consiste en aprender a partir de patrones de entradas para los que no se especifican los valores de sus salidas” (Russell y Norvig, 2004, p. 740).

El objetivo de un modelo de aprendizaje no supervisado es identificar patrones significativos entre los datos. En otras palabras, el modelo no tiene pistas sobre cómo categorizar cada dato, sino que debe inferir sus propias reglas. Además, debe definir cada uno de los grupos a clasificar (Google Developers, s.f.-a, 2022).

2.4.3 Aprendizaje Semi-Supervisado o por refuerzo

El aprendizaje semi-supervisado se presenta como caso intermedio entre el aprendizaje supervisado y el no supervisado. Este aprendizaje utiliza datos de entrenamiento etiquetados y no etiquetados. Usualmente se emplea como una adaptación del aprendizaje no supervisado, dándole una pequeña cantidad de datos etiquetados para que luego se detecten patrones mediante algoritmos (Google Developers, s.f.-a, 2022).

2.5 Modelos de aprendizaje automático

Un modelo de aprendizaje automático permite ser entrenado para reconocer ciertos tipos de patrones. Entrena un modelo sobre un conjunto de datos, proporcionándole un algoritmo que puede usar para razonar y aprender de esos datos. Luego, con el modelo, se pueden realizar predicciones sobre los datos introducidos. Dicho esto, los modelos de aprendizaje automático orientados en algoritmos de clasificación pueden estar incluidos por tres grandes categorizaciones, que se detallan a continuación.

2.5.1 Regresión logística

Se reconoce con el nombre de regresión logística a un modelo de aprendizaje que trabaja por aproximación, relacionando una variable escalar dependiente (llamada Y) y una o más variables explicativas (llamada X). Su principal diferencia con otros

modelos de aprendizaje, que se detallaran en los apartados subsiguientes (árbol de decisión y redes neuronales), es que antes de mirar a los datos, ya sabemos cuántos parámetros necesitamos. Trata de encontrar una línea que se ajuste bien a la nube de puntos que se disponen. La regresión logística se usa para modelar la probabilidad de un número finito de resultados, siendo dos los más usuales. De manera que este modelo consiste en hallar cuáles son los mejores parámetros para los datos que poseamos y se enfoca en la distribución de probabilidad condicional de la respuesta dados los valores de los predictores (Goodfellow et al., 2016).

2.5.2 Árbol de decisión

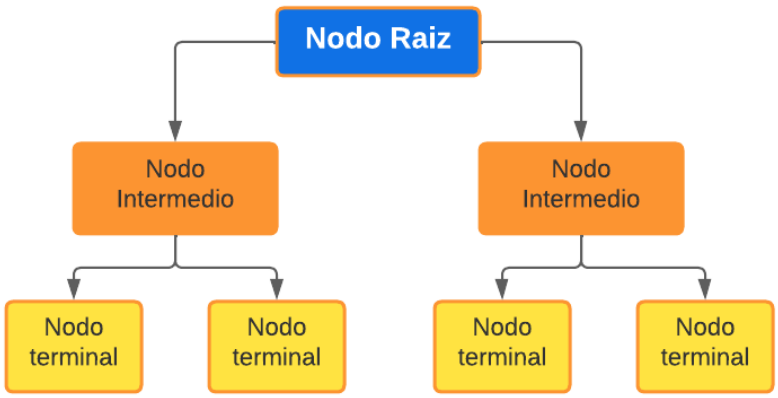
Otro modelo de aprendizaje diferente a la regresión logística es el árbol de decisión. Este consta de nodos que forman el llamado "árbol raíz". Todos los nodos tienen exactamente un borde entrante. El nodo que tiene salida se denomina nodo interno o nodo de prueba, el resto se denominan hojas. En un árbol de decisión, cada nodo de prueba divide el espacio de la instancia en dos o más subespacios de acuerdo con una cierta función discreta de los valores de entrada. En el caso más simple, cada prueba considera un solo atributo, de modo que el espacio de la instancia se divide de acuerdo con el valor del atributo (Russell y Norvig, 2004).

> Un árbol de decisión toma como entrada un objeto o una situación descrita a través de un conjunto de atributos y devuelve una «decisión»: el valor previsto de la salida dada la entrada. Los atributos de entrada pueden ser discretos o continuos. A partir de ahora, asumiremos entradas discretas. El valor de la salida puede ser a su vez discreto o continuo; aprender una función de valores discretos se denomina clasificación; aprender una función continua se denomina regresión. Nos concentraremos en clasificaciones booleanas, en las cuales cada ejemplo se clasifica como verdadero (positivo) o falso (negativo) (Russell y Norvig, 2004, p. 744).

En la figura a continuación, se puede observar el flujo de trabajo de este modelo.

Figura 6

Árbol de decisión



2.5.3 Redes neuronales

Una gran cantidad de las tecnologías recientes que utilizamos diariamente proviene de la imitación del cuerpo humano. Tanto es así, que en el caso del aprendizaje automático se representó al cerebro para tratar de lograr un modelo computacional de aprendizaje biológico. Como resultado, nacieron las redes neuronales artificiales (Goodfellow et al., 2016).

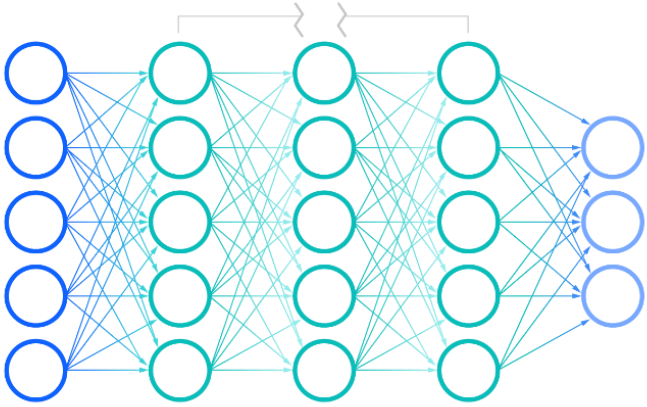
> Una neurona es una célula del cerebro cuya función principal es la recogida, procesamiento y emisión de señales eléctricas… Se piensa que la capacidad de procesamiento de información del cerebro proviene principalmente de redes de este tipo de neuronas. Por esta razón, algunos de los primeros trabajos en IA pretendían crear redes neuronales artificiales (Russell y Norvig, 2004, p. 838).

Cada neurona tiene conexiones (o enlaces) a las demás, pudiendo recibir, procesar y transmitir señales con información. Estas señales tienen un peso que se ajusta a medida que avanza el aprendizaje, que aumenta o disminuye la fuerza de la señal en una conexión. Por lo general, las neuronas se agregan en capas; diferentes capas pueden realizar diferentes transformaciones en sus entradas. Las señales viajan desde la capa de entrada hasta la capa de salida, posiblemente después de atravesar las

capas varias veces (IBM Cloud Education, 2020). En la figura a continuación, se puede observar el flujo de trabajo de este modelo.

Figura 7

Redes neuronales



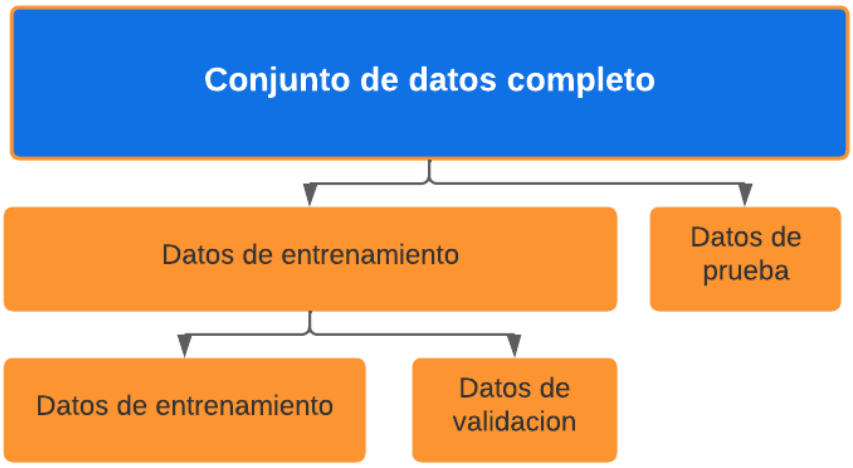
Nota. Adaptado de *Neural Networks* [Fotografía], IBM Cloud Education, 2020.

2.6 Fases de desarrollo – Conjunto de datos

Un sistema de aprendizaje automático basa su efectividad en la información con la cual se nutre. Para ser desarrollado correctamente, se deben tener varios conjuntos de datos. Estos se clasifican en tres tipos: de entrenamiento, de validación y de prueba. En la mayor parte de los casos, los datos son defectuosos o tienen errores. En este sentido, la preparación de datos supone un conjunto de procedimientos de ayuda para que estos sean más adecuados para el aprendizaje automático. En muchas ocasiones, incluso, estos procedimientos consumen la gran parte del tiempo dedicado a estos sistemas (Google Developers, s.f.-b, 2022). Los tipos de conjuntos de datos y pasos para descartar errores se detallan a continuación.

Figura 8

Ciclo de vida de un conjunto de datos

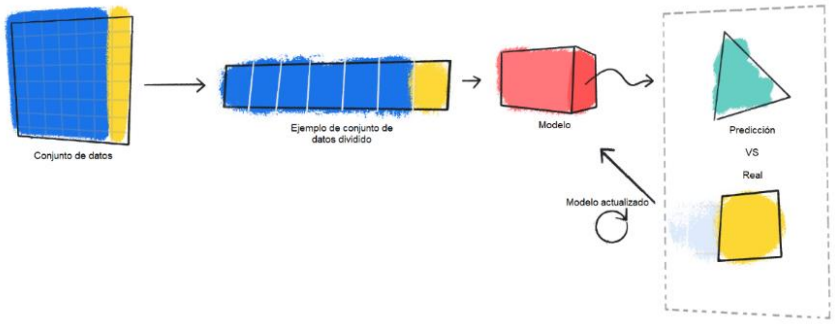


2.6.1 Entrenamiento

Antes de que un modelo pueda realizar predicciones, debe ser entrenado. Para lograrlo, como primer paso, le brindamos un conjunto de datos con ejemplos. El modelo tomará un ejemplo particular aleatoriamente y dará una predicción. Seguidamente, se comparará el valor predicho con el real y se actualizará la solución. Este proceso se repite para cada uno de los elementos que tenga el conjunto de datos. Es por ello que mientras más datos se consigan, más robusto será el modelo y realizará mejores predicciones, cumpliendo con su objetivo principal. De esta forma, el modelo aprende gradualmente la relación correcta entre las características y la etiqueta (Google Developers, s.f.-c, 2022). En la figura a continuación, se puede observar el ciclo de entrenamiento de un conjunto de datos.

Figura 9

Ciclo de entrenamiento de un conjunto de datos



Nota. Adaptado de Supervised Learning [Fotografía], por Google Developers, s.f.-c, 2022.

2.6.2 Validación

Una vez que se concluye con el entrenamiento de un modelo sobre un conjunto de datos, se continúa con su validación. Esta se utiliza para evaluar frecuentemente el algoritmo durante la fase de entrenamiento. Su principal función es ayudar a proteger a los modelos de ajustes innecesarios y eliminar errores (Google Developers, s.f.-d, 2022).

2.6.3 Prueba

La prueba es la fase que le sigue a la validación de los datos necesaria para evaluar el rendimiento o precisión del modelo una vez finalizada la fase de entrenamiento. Es una muestra de datos utilizada para hacer una evaluación imparcial del modelo final que se ajusta a los datos de entrenamiento. Se lleva a cabo introduciendo un conjunto de datos representativo con respecto al global. En otras palabras, este conjunto de datos valida si el modelo generado a partir de datos de entrenamiento realmente funciona (Google Developers, s.f.-d, 2022). En la figura presentada a continuación, se puede observar el ciclo de validación y prueba de un conjunto de datos.

Figura 10

Ciclo de validación y prueba de un conjunto de datos

2.7 Aprendizaje Automático Automatizado (AutoML)

Ante todo, es necesario mencionar que llevar a cabo la práctica de un modelo de aprendizaje automático no es una tarea sencilla. Sus principales dificultades, entre otras, radican en que se deben tener conocimientos profundos de los conceptos para poder desarrollar uno de forma satisfactoria y en que presentan un costo elevado. Para combatir estos problemas, surge AutoML, cuyo objetivo es proporcionar herramientas de aprendizaje fácilmente accesibles para personas con experiencia limitada, permitiendo que una máquina imite cómo humanos diseñan, ajustan y aplican algoritmos.

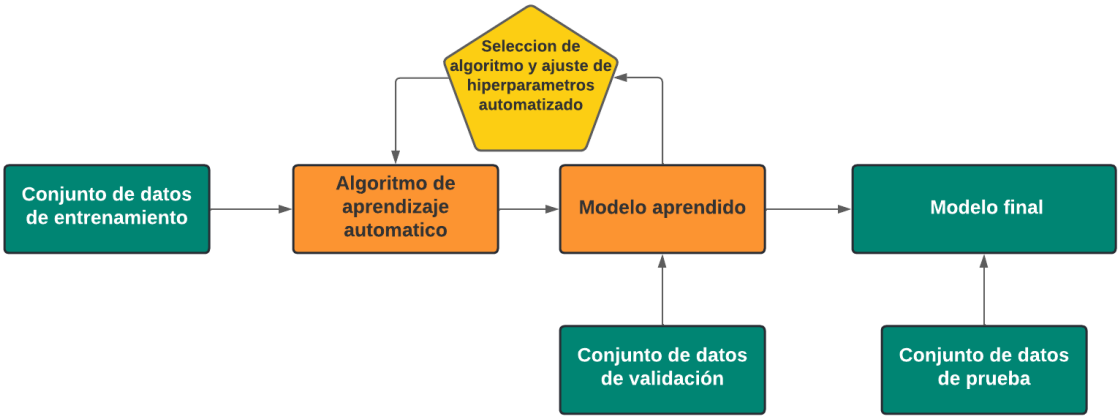
Como lo denota su nombre, el aprendizaje automático automatizado es el proceso de realizar tareas repetitivas sin intervención humana en un sistema, empleando el aprendizaje automático para resolver problemas. En otras palabras, se puede definir como “el proceso de automatizar las tareas lentas e iterativas del desarrollo de modelos de aprendizaje automático” (Microsoft, 2022). Su principal función es crear “modelos de aprendizaje con un escalado, eficiencia y productividad altos, al mismo tiempo que mantiene la calidad del modelo” (Microsoft, 2022).

Cada algoritmo de AutoML consiste en tres elementos principales llamados espacio de búsqueda, estrategia de búsqueda y estrategia de evaluación del desempeño. El primero puede ser definido como un conjunto de hiperparámetros junto con sus rangos. El segundo, como una estrategia para seleccionar el conjunto óptimo de hiperparámetros del espacio de búsqueda que, a menudo, selecciona secuencialmente los hiperparámetros en el espacio de búsqueda y evalúa su rendimiento. Por último, la evaluación de desempeño se define como una forma de calificar el rendimiento de un algoritmo de aprendizaje automático específico instanciado por los

hiperparámetros seleccionados (Song et. al, 2022). En la figura 11, se observa un adaptado del ciclo de vida de esta tecnología.

Figura 11

Ciclo de vida de modelo de AutoML



Para poder efectuar esta tecnología existe una importante cantidad de herramientas. Las principales diferencias entre estas radican mayormente en características como el precio, soporte y fidelidad. Seguidamente, se detallan las tres más utilizadas en torno a su categorización dentro del apartado llamado código abierto.

2.7.1 AutoKeras

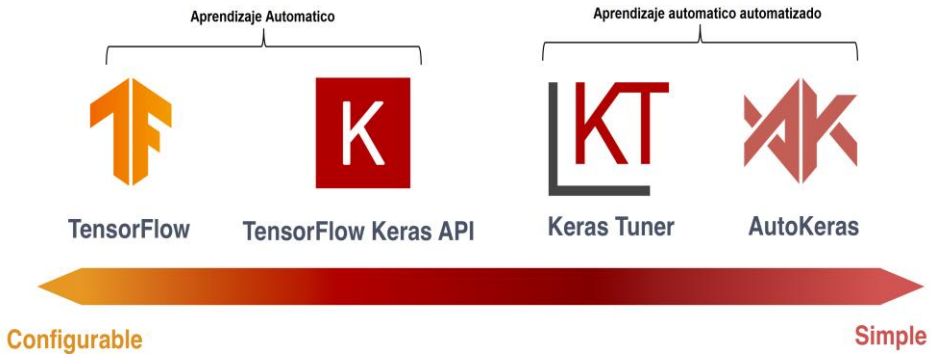
AutoKeras se representa como un sistema de código abierto de aprendizaje automático automatizado basado en TensorFlow Keras, una biblioteca de redes neuronales artificiales escrita en el lenguaje de programación Python. Este sistema proporciona funciones para buscar automáticamente arquitectura e hiperparámetros de modelos de aprendizaje, y presenta las ventajas de poder crear modelos sin mucho conocimiento sobre todo lo que rodea al aprendizaje automático y de ser compatible

con los tres principales sistemas operativos para computadoras (MacOs, Linux y Windows) (AutoKeras, 2022).

Otro punto fuerte de AutoKeras es que permite la exportación fácil de los modelos, y al estar basada en Tensorflow, una de las librerías más utilizadas para redes neuronales, estos son fácilmente adaptables a un entorno de producción con pocas dependencias y en casi cualquier tipo de arquitectura de máquina (Murillo Gonzalez, 2021). A continuación, se observa un adaptado de una comparación entre las tecnologías TensorFlow Keras y AutoKeras.

Figura 12

Comparación de TensorFlow Keras y AutoKeras



Nota. Adaptado de *Automated Machine Learning in Action* (p. 74), por Song et. al, 2022, Manning Publications.

2.7.2 Auto-Sklearn

Auto-Sklearn es un conjunto de herramientas de aprendizaje automático basado en la librería del lenguaje de programación Python llamada Scikit-Learn. Entre sus principales ventajas, se destaca que libera al usuario de la selección de algoritmos y el ajuste de hiperparámetros. También ofrece una amplia gama de algoritmos de aprendizaje automático implementados y es de fácil uso tanto para expertos como para principiantes.

Otra de sus virtudes es que “no hay parámetros necesarios que aportar para las tareas, aunque sí que hay parámetros opcionales como el tiempo de procesamiento o el error máximo admitido a la hora del entrenamiento del modelo” (Murillo Gonzalez, 2021, p. 17). Sin embargo, entre sus principales desventajas, se menciona que no es compatible con otro sistema operativo que no sea Linux (AutoML org, s.f.-a, 2022).

2.7.3 Auto-Pytorch

Auto-Pytorch es un sistema de aprendizaje automático automatizado llevado a cabo por Facebook. Está basado en Pytorch, una librería de código abierto desarrollada en el lenguaje de programación Python. Auto-Pytorch es capaz de implementar y ajustar automáticamente el ciclo entero de vida de una aplicación de aprendizaje automático, desde el procesamiento de datos hasta las técnicas de entrenamiento (AutoML org, s.f.-b, 2022). Zimmer et al. (2021) concluyeron que este sistema optimiza de forma sólida la arquitectura de una red y los hiperparámetros para entrenarla, permitiendo un ciclo totalmente automatizado. Además, logra un rendimiento de vanguardia, aseguran los autores.

3 – Metodología

3.1 Categorías en que se sustenta el trabajo de investigación

Esta investigación, desde el punto de vista de los objetivos del trabajo, puede ser categorizada como descriptiva y, a la vez, como explicativa. La primera categorización se logró alcanzar porque nuestro estudio presenta una reseña exhaustiva de las cualidades y características de la población en estudio. En este caso, se proporciona una descripción de los más eficientes sistemas de aprendizaje automático automatizado para algoritmos de clasificación, permitiendo poner de manifiesto su comportamiento, así como ofrecer también un manual detallado de recomendaciones y buenas prácticas. Nuestro trabajo alcanza también la categorización de investigación explicativa puesto que mediante pruebas prácticas se logró automatizar modelos de aprendizaje automático en tareas de clasificación de forma rápida y segura pudiendo determinar la eficiencia de cada uno de los sistemas probados y comprobando con datos reales que AutoKeras tiene una ventaja holgada con respecto a las demás librerías.

Finalizando la categorización de las metodologías alcanzadas, desde el punto de vista del propósito de investigación se adoptó por su clasificación como investigación aplicada. Esto se debe a que los conocimientos desarrollados son extremadamente necesarios para obtener resultados, conclusiones y poder determinar si los objetivos de esta investigación fueron cumplidos de manera correcta.

3.2 Determinación de la población de trabajo

Dentro del universo del aprendizaje automático, existen tres cuestiones fundamentales a la hora de analizar datos. Estas son llamadas análisis de texto, análisis de imágenes y análisis de datos numéricos, donde cada una cuenta con sus respectivas ventajas y desventajas. En el caso del primer elemento mencionado, se utilizan algoritmos que procesan datos de texto a escala, categorizando un documento de texto en uno de un conjunto predefinido de grupos. En muchos problemas de clasificación de grupos, esta categorización se basa principalmente en las palabras clave del texto (Google Developers, s.f.-e, 2022).

En el caso de las imágenes, en su gran mayoría necesitan una enorme cantidad de procesamiento y transformación antes de que pueda ser siquiera entrenada. Por ejemplo, estas

deben ser normalizadas en cuanto a su tamaño y dimensiones o modificadas en cuanto a su color, transformándolas en imágenes en diferentes tonos de blanco y negro (siempre que no sea un factor clave para su entrenamiento). Todo esto es sumamente significativo, y si no se logra de manera correcta, el procesamiento se volverá mucho más arduo y tomará más tiempo, recursos y costos (Google Developers, s.f.-f, 2022).

Por otra parte, en el caso de datos numéricos, se trabaja con datos mucho más sencillos y sin necesidad de una transformación. Este tipo de datos no es visual ni está organizado en elementos lingüísticos y está formado por cifras y medidas recopiladas por máquinas, sensores o personas. Uno de los pasos en el flujo de trabajo usual de un sistema de aprendizaje automático es transformar los datos de entrada en datos numéricos para luego procesarlos. Esto se da ya que las herramientas de procesamiento de estos tipos de sistemas exigen trabajar con datos numéricos, cosa que con este tipo de datos no es necesaria (Google Developers, s.f.-g, 2022).

Debido a los puntos presentados en anterioridad, la elección de la población de trabajo en el presente documento se encuentra enfocada hacia el análisis y clasificación de datos numéricos tanto binarios como no binarios (secciones explicadas en el apartado 2.4.1.1), dentro de su categorización como algoritmo de clasificación de aprendizaje supervisado. Debido a esto, se indagaron y utilizaron muestras compatibles con este tipo de conjunto de datos.

Para el caso de la población de datos numéricos binarios, se utilizó como muestra un conjunto de datos obtenidos del sitio web Kaggle (un banco de datos público) llamado Breast Cancer [Cáncer de mama] (Namdari, 2022). Este conjunto se compone de pacientes con cáncer de mama durante 2017, obtenidos del programa SEER del NCI, que brinda información sobre estadísticas de cáncer basadas en la población. El objetivo final para lograr un modelo binario con este conjunto de datos es poder comprobar si el cáncer es benigno o maligno.

Por otro lado, para el caso de la población de datos numéricos no binarios (o múltiples), se utilizaron como muestra dos conjuntos de datos, obtenidos del sitio web Kaggle. El primero, llamado Mobile Price Classification [Clasificación de precios de celulares móviles] (Sharma, 2017), contiene información acerca de miles de teléfonos celulares con sus respectivas características. El objetivo de este conjunto de datos es poder realizar un modelo que pueda predecir un rango de precios de los diferentes teléfonos celulares según sus características. El

segundo, llamado Milk Quality Prediction [Predicción de la calidad de la leche] (Rajendran, 2022), se compone de miles de datos orientados a características propias de la leche, como son temperatura, olor, color, etc. El objetivo final para lograr un modelo binario con este conjunto de datos es poder predecir el nivel de calidad de la leche clasificada.

3.3 Determinación de las técnicas de recolección de datos a emplear

Para recolectar datos, se determinó la utilización de diferentes técnicas para evaluar el modelo de AutoML elegido. Entre ellos se destacan los explicados a continuación.

- Velocidad de entrenamiento: uno de los factores más críticos en las plataformas de aprendizaje automático es el tiempo que necesitan para entrenar los modelos. La velocidad de entrenamiento generalmente se mide como la cantidad de muestras por segundo que la plataforma procesa durante el entrenamiento (Elgendy, 2020).
- Tasa de error: luego de entrenar el modelo, el próximo paso será determinar su desempeño. Para lograr esto, debemos determinar si el modelo tiene cuellos de botella que están afectando ese rendimiento ideal, siendo el sobreajuste y el ajuste insuficiente del conjunto de datos de entrenamiento sus principales causas. Si al modelo le está yendo muy bien en el conjunto de entrenamiento, pero relativamente mal en el conjunto de validación, entonces se está dando un sobreajuste. En caso contrario, si el modelo se desempeña mal en el conjunto de datos de entrenamiento, se da un ajuste insuficiente. El valor ideal es 0, por lo que mientras más cerca de ese número se encuentre, menos error tendrá. (Elgendy, 2020).
- Precisión: es la proporción de predicciones que coinciden exactamente con las etiquetas de clase verdaderas. Mientras más cerca de 1 (o 100%) se encuentre, más eficiente y preciso será el modelo (Microsoft Learn, 2022).

3.4 Determinación de las técnicas de análisis de datos relevados

Tomando en cuenta los puntos detallados para recolectar datos, las técnicas empleadas para poder analizarlos se especifican a continuación.

- Curva de aprendizaje: en lugar de observar los números resultantes de precisión de entrenamiento y de validación en cada uno de los modelos, podemos llevar a cabo un gráfico comparable sobre estos datos para que sean más fácilmente legibles. Esta técnica nos indica un porcentaje en el que el modelo entrenado predice correctamente la clasificación de los conjuntos de datos de entrenamiento y prueba otorgados. Se utilizarán las mismas técnicas de evaluación para cada librería de AutoML elegida, tanto para los datos de entrenamiento como para los datos de prueba. Vale aclarar, que mientras más altos resulten estos porcentajes, mejor entrenado y definido estará nuestro modelo (Elgendy, 2020).
- Evaluación de eficiencia: se compara la tasa de error con la cantidad de modelos alcanzados, pudiendo visualizar una curva que refleja cómo su error se hace cada vez más pequeño (idealmente).
- Algoritmo elegido por la herramienta de AutoML: en este trabajo, nos enfocaremos únicamente en algoritmos de aprendizaje automático supervisado, en su categorización de algoritmos de clasificación. Siguiendo este tema, se definen como algoritmos que aprenden a asociar alguna entrada con alguna salida, dado un conjunto de ejemplos de entrenamiento otorgados (Goodfellow et al., 2016).

4 – Cronograma operativo y recursos necesarios

4.1 Recursos necesarios

Los recursos necesarios para llevar a cabo este trabajo fueron:

- Recursos Humanos: alumno (Gianluca Sparvoli), tutora técnica (Dra. Claudia Pons) y profesora de trabajo final (Dra. Marcela Samela).
- Recursos Logísticos: este apartado puede ser dividido en cuatro categorías:
  - ✓ Software: Google Colab, como entrono de desarrollo, Python, como lenguaje de programación general y AutoKeras, Auto-Sklearn y Auto-Pytorch como librerías para entrenar modelos de AutoML.
  - ✓ Hardware: computadora con acceso a internet.
  - ✓ Recursos bibliográficos: todos los recursos detallados en la sección de referencias del presente trabajo.
  - ✓ Conjuntos de datos: todos los conjuntos de datos gratuitos detallados en el punto 3.2 del presente documento.

4.2 Cronograma

El trabajo planteado se llevó a cabo durante todo el año 2022. En líneas generales hubo tres momentos clave detallados a continuación.

- Buceo bibliográfico y planteo del problema
- Composición del marco teórico y metodologías
- Redacción del informe final, resultados y conclusiones.

Seguidamente, se detalla un cronograma operativo del presente trabajo final de carrera.

Tabla 1

Cronograma operativo del presente trabajo final de carrera.

	2022											
	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sept	Oct	Nov	Dic
Selección y delimitación del tema de investigación												
Recopilación de datos												
Revisión de fuentes primarias y secundarias												
Estructura del trabajo												
Introducción												
Revisión parcial del trabajo con tutor												
Marco teórico												
Metodologías												
Recolección y análisis de datos												
Elaboración del informe												

Conclusiones y líneas futuras de investigación												
Revisión final del trabajo con tutor												

4.3 Presupuesto – Recursos humanos

Desde el punto de vista de presupuesto, si bien no se tuvieron costos fijos ya que el desarrollo fue llevado a cabo con lenguajes de programación, entornos de desarrollo y librerías gratuitas y de código abierto, se incluyeron costos variables tales como horas de desarrollo de software. Siguiendo este punto, en Argentina, a la fecha de septiembre de 2022 según Glassdoor (2022), un desarrollador de aprendizaje automático posee un sueldo promedio de 232534 pesos por mes, por cuarenta horas de trabajo semanales, lo que da un promedio de 1453,33 pesos por hora trabajada. En el desarrollo de este trabajo, se calculó que según experiencias previas aproximadamente se empleó un total de cinco horas por herramienta con cada uno de los conjuntos de datos seleccionados. Siendo cuatro conjuntos de datos diferentes, con tres herramientas seleccionadas, multiplicando por el precio de hora de sueldo promedio, se obtiene un resultado final de 87199,80 pesos de costo variable.

5 – Recolección y análisis de datos

5.1 Recolección de datos

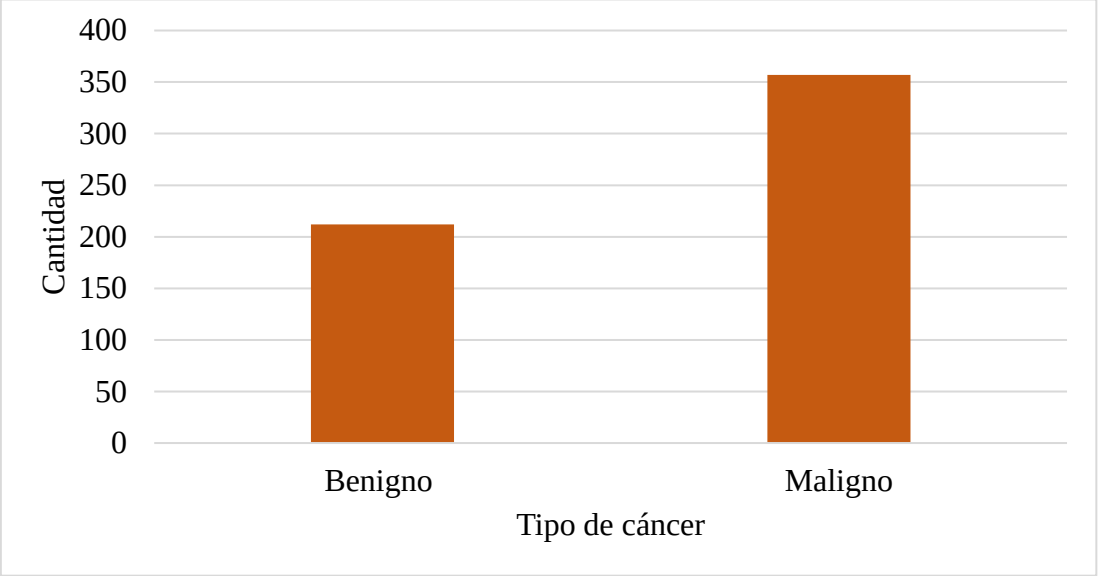
A continuación, se contemplan todos los requerimientos necesarios para poder llevar a cabo las pruebas y así demostrar el objetivo específico número tres del presente trabajo, siendo este, entrenar modelos de aprendizaje automático automatizado utilizando cada una de las herramientas seleccionadas.

A la hora de recolección de datos e implementación de las pruebas requeridas para contemplar el análisis y clasificación de texto binario, se realizaron pruebas con las herramientas de aprendizaje automático automatizado AutoKeras, Auto-Pytorch y Auto-Sklearn con el conjunto de datos llamado Breast Cancer [Cáncer de mama] (explicados en el apartado 3.2). Para una comparación justa, se utilizaron los mismos procedimientos de entrenamiento. Un elemento en común a todos estos conjuntos es que el su total fue dividido en un 80% en datos de entrenamiento y un 20% en datos de prueba. También, luego de normalizar un poco los datos, se procedió en todos los conjuntos a separarlos en datos de valor (simbolizado con el valor “x”) y de resultado (simbolizado con el valor “y”) y, posteriormente, en entrenamiento y prueba a cada uno de estos.

Comenzando por la herramienta AutoKeras, con el conjunto de datos llamado Breast Cancer, se cumplió con el objetivo de implementar y predecir si el cáncer representado en cada una de las filas es benigno o maligno, representado con un 0 o un 1 respectivamente en la columna llamada target [objetivo] (disponible en la sección de anexo de código 1 del presente trabajo). Como se observa en la figura 13, el conjunto cuenta con 569 filas de datos, en las que 212 tienen un valor en el campo target de 0 y 357 de 1. El modelo pudo ser llevado a cabo de forma satisfactoria pudiendo obtener un 95,61% de precisión en las pruebas con sólo un 2,94% de error de entrenamiento y validación. A su vez, sólo tardó 7 segundos en realizar el entrenamiento, lo que nos indica que esta herramienta es mucho más rápida que muchas otras de su misma categoría.

Figura 13

Comparación entre tipos de cáncer del conjunto de datos Breast Cancer.



Siguiendo con la implementación del conjunto de datos Breast Cancer, se continuó con el entrenamiento de su respectivo modelo con la herramienta Auto-Sklearn. Como conclusión del ejercicio, presente en el apartado de anexo de código 2, se observó que el modelo mostró un 93,5% de precisión en las pruebas con sólo un 4,3% de error de entrenamiento y validación. A su vez, tardó 96 segundos en realizar el entrenamiento.

Finalizando con el conjunto mencionado anteriormente, se realizó su entrenamiento con la herramienta Auto-Pytorch. En este se cumplió con el objetivo de implementar y predecir si el cáncer representado en cada una de las filas es benigno o maligno (disponible en la sección de anexo de código 3 del presente trabajo). El modelo pudo ser llevado a cabo de forma satisfactoria pudiendo obtener un 97,3% de precisión en las pruebas con 2,6% de error de entrenamiento y validación. A su vez, tardó 120 segundos en realizar el entrenamiento, lo que nos indica que esta herramienta es más lenta en comparación a AutoKeras.

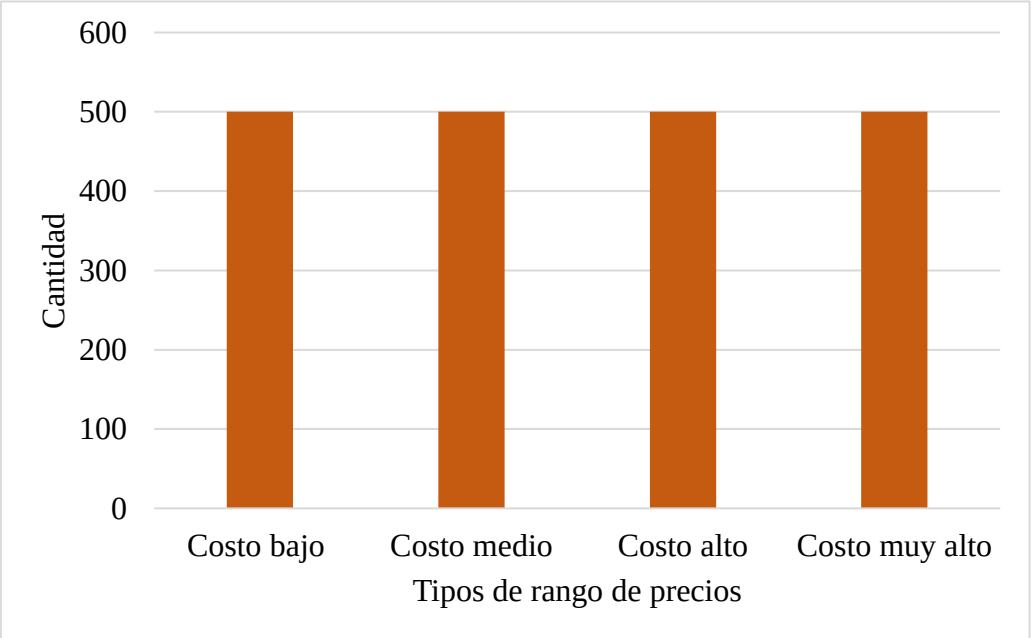
A la hora de recolección de datos de los experimentos requeridos para contemplar el análisis y clasificación de texto no binario, se realizaron pruebas con las herramientas de aprendizaje automático automatizado ya mencionadas con anterioridad, con los conjuntos de datos llamados Mobile Price Classification [Clasificación de precios de celulares móviles] y

Milk Quality Prediction [Predicción de la calidad de la leche]. Como ya fue mencionado, el total del conjunto de datos fue dividido en un 80% en datos de entrenamiento y un 20% en datos de prueba.

Continuando con los conjuntos de datos seleccionados, el próximo a evaluar fue el llamado Mobile Price Classification con la herramienta AutoKeras. El objetivo de este problema era poder predecir un rango de precios simbolizados con cuatro valores, siendo estos 0 (costo bajo), 1 (costo medio), 2 (costo alto) y 3 (costo muy alto), dentro de la variable llamada price_range [rango de precio]. Siguiendo esta idea, en la figura 14 se observa que el conjunto cuenta con 2000 filas de datos, con una división de 500 filas de datos para cada categoría. Seguidamente, se entrenó el modelo satisfactoriamente con un total de tres intentos con diez épocas cada uno, obteniendo como resultado un 88,25 % de prueba de precisión, un 3,17 % de error de entrenamiento y de validación y un promedio de 28 segundos de velocidad de entrenamiento. La implementación práctica está disponible en la sección de anexo de código 4 del presente trabajo.

Figura 14

Comparación entre los tipos de rango de precios simbolizados en el conjunto de datos Mobile Price Classification.



Siguiendo con la implementación del conjunto de datos Mobile Price Classification, se continuó con el entrenamiento de su respectivo modelo con la herramienta Auto-Sklearn. El modelo, presente en el apartado anexo de código 5, logró ser llevado a cabo de forma satisfactoria pudiendo obtener un 95,2% de precisión en las pruebas con sólo un 2,6% de error de entrenamiento y validación. A su vez, estuvo 95 segundos realizando el entrenamiento ya mencionado.

Consumando los entrenamientos con el conjunto mencionado anteriormente, se lo realizó con la herramienta Auto-Pytorch. En este se cumplió con el objetivo de implementar y predecir el rango de precios (disponible en la sección de anexo de código 6 del presente trabajo). El modelo pudo ser llevado a cabo de forma satisfactoria pudiendo obtener un 93,2% de precisión en las pruebas con 6,7% de error de entrenamiento y validación. A su vez, tardó 115 segundos en realizar el entrenamiento.

Finalizando con las implementaciones, el último elegido es el llamado Milk Quality Prediction [Predicción de la calidad de la leche]. En el caso de la herramienta AutoKeras, se logró llevar a cabo una implementación de forma satisfactoria (disponible en la sección de anexo de código 7 del presente trabajo). El objetivo de este problema era poder predecir un rango de precios simbolizados con tres valores alfanuméricos, siendo estos Low (malo), Medium (medio), High (bueno) dentro de la variable llamada Grade [grado] (disponible en la sección de anexo de código 4 del presente trabajo). El conjunto cuenta con 1059 filas de datos, en las que 256 tienen un valor en el campo Grade de High, 429 con un valor Low y 374 con valor Medium. El modelo pudo ser llevado a cabo de forma satisfactoria pudiendo obtener un 97,64% de precisión en las pruebas con sólo un 1,63% de error de entrenamiento y validación. A su vez, sólo tardó 20 segundos en realizar el entrenamiento.

A continuación, se llevó a cabo el modelo de Milk Quality Prediction con la herramienta Auto-Sklearn. En este se cumplió con el objetivo de implementar y predecir el rango de precios (disponible en la sección de anexo de código 8 del presente trabajo). El modelo pudo ser llevado a cabo de forma satisfactoria pudiendo obtener un 100% de precisión en las pruebas con 0% de error de entrenamiento y validación. A su vez, tardó 96 segundos en realizar el entrenamiento.

Concluyendo con los entrenamientos de los modelos, se continuó con el ejercicio con la herramienta Auto-Pytorch y el conjunto Milk Quality Prediction. El modelo, presente en el apartado anexo de código 9, logró ser llevado a cabo de forma satisfactoria pudiendo obtener un 100% de precisión en las pruebas con sólo un 0% de error de entrenamiento y validación. A su vez, estuvo 113 segundos realizando el entrenamiento ya mencionado.

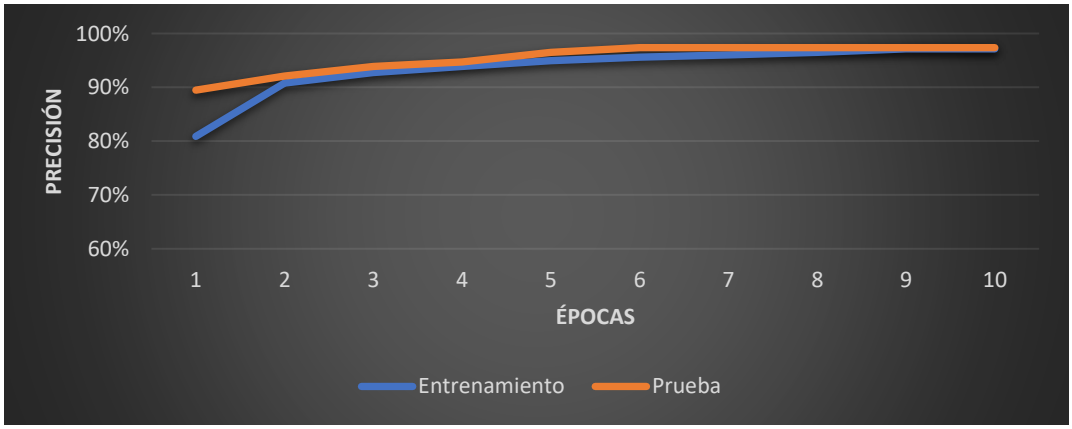
5.2 Análisis de datos

Una vez concluida la recolección de datos, se procedió con el análisis de los mismos. En cuanto a las técnicas, las mismas fueron definidas en la sección 3.4 del presente trabajo, siendo llamadas curva de aprendizaje, evaluación de eficiencia y algoritmos elegidos por cada herramienta de AutoML. Normalizando un poco estas cuestiones, en el caso de la primera mencionada en el eje X están representadas la cantidad de épocas de entrenamiento, y en el eje Y su precisión. Por otra parte, en el caso de la evaluación de eficiencia, se normalizo una referencia en el eje X de cantidad de épocas de entrenamiento y, en el eje Y, de tasa de error.

Comenzando con la herramienta de aprendizaje automático automatizado AutoKeras, se realizaron implementaciones de manera satisfactoria. En el caso del primer conjunto de datos binario, llamado Breast Cancer, logramos visualizar de manera correcta su curva de aprendizaje, representada en la figura 15. En ella se puede dilucidar que tanto las curvas de entrenamiento como las de prueba se ajustan de una forma similar, salvo en la primera época.

Figura 15

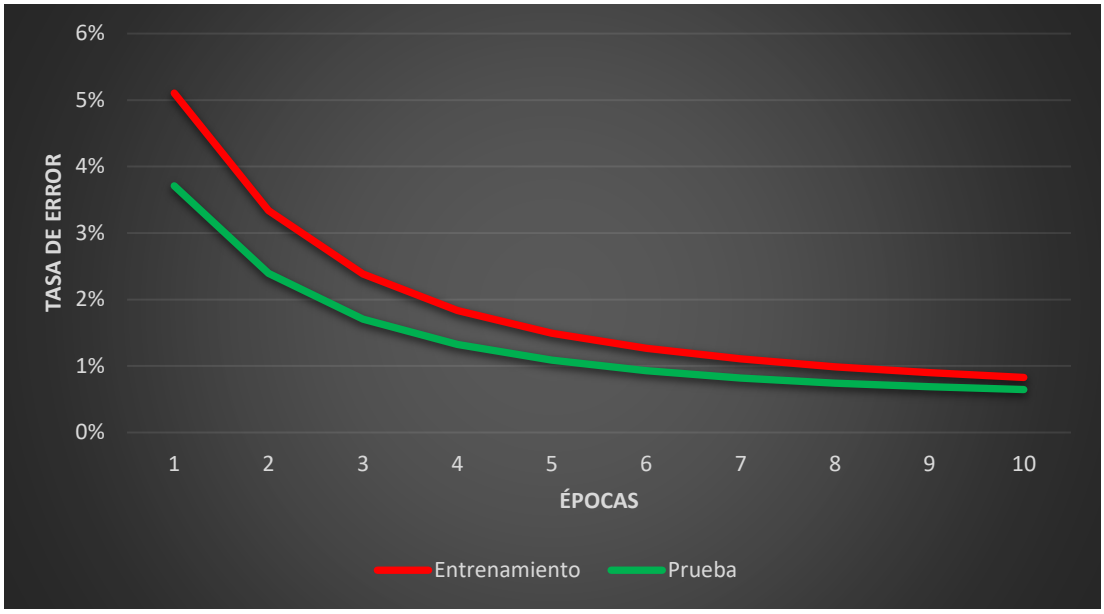
Curva de aprendizaje de AutoKeras con datos de Breast Cancer.



Continuando con AutoKeras y Breast Cancer, se graficó su evaluación de eficiencia, presente en la figura 16. En este caso, se puede denotar que el valor de error de entrenamiento en la primera época es un tanto elevado, siendo este casi un 5%, sin embargo, a medida que el modelo es entrenado este valor se va haciendo cada vez más pequeño para así lograr que en la última época sea casi nulo, mejorando su desempeño y disminuyendo su tasa de error.

Figura 16

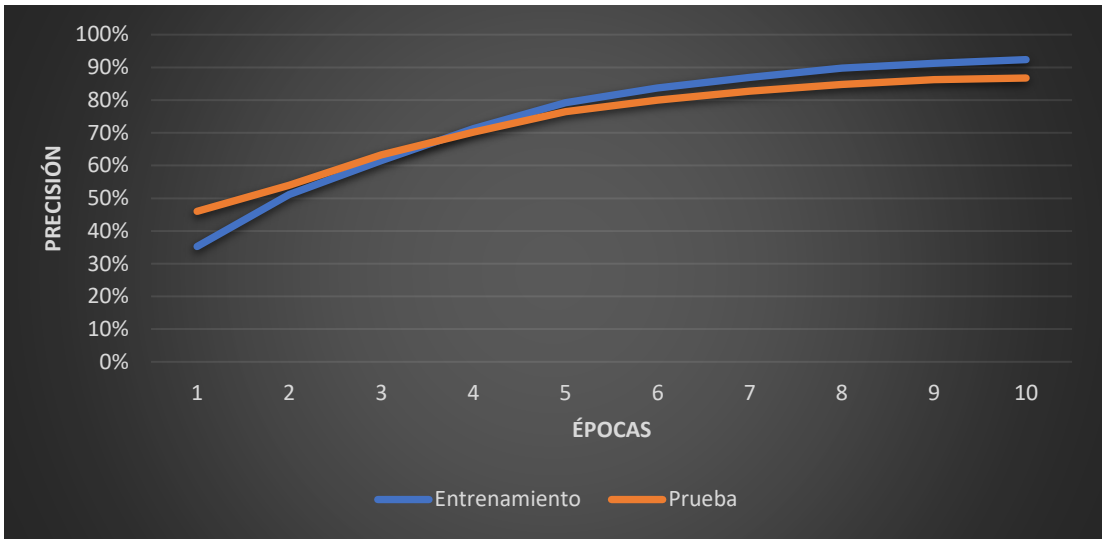
Evaluación de eficiencia de AutoKeras con datos de Breast Cancer.



Siguiendo con los conjuntos de datos seleccionados, el próximo a evaluar fue el llamado Mobile Price Classification con la herramienta AutoKeras. En el caso de la curva de aprendizaje, representada en la figura 17, se puede observar que tanto las curvas de entrenamiento como las de prueba se ajustan a los datos con un comportamiento similar. También, se observó que durante las épocas iniciales su valor de precisión es muy bajo (alrededor del 50%), pero a medida que el modelo se entrena, va incrementando cada vez más.

Figura 17

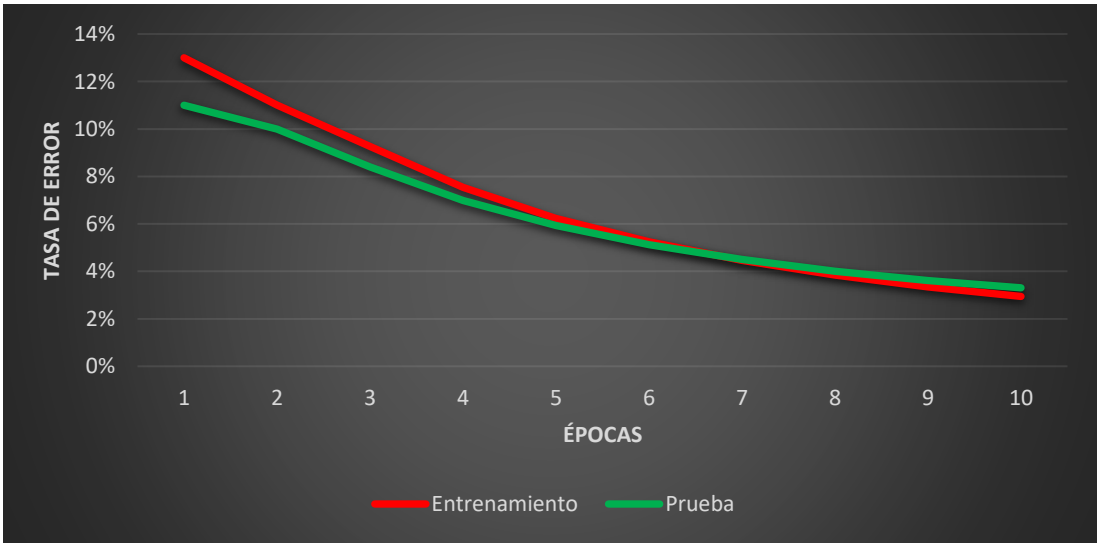
Curva de aprendizaje de AutoKeras con datos de Mobile Price Classification.



Siguiendo con este conjunto de datos y esta tecnología, se optó por delinear su evaluación de eficiencia, presente en la figura 18. En ella, se puede observar que sus valores son cada vez menores, mejorando época por época su desempeño y disminuyendo su tasa de error.

Figura 18

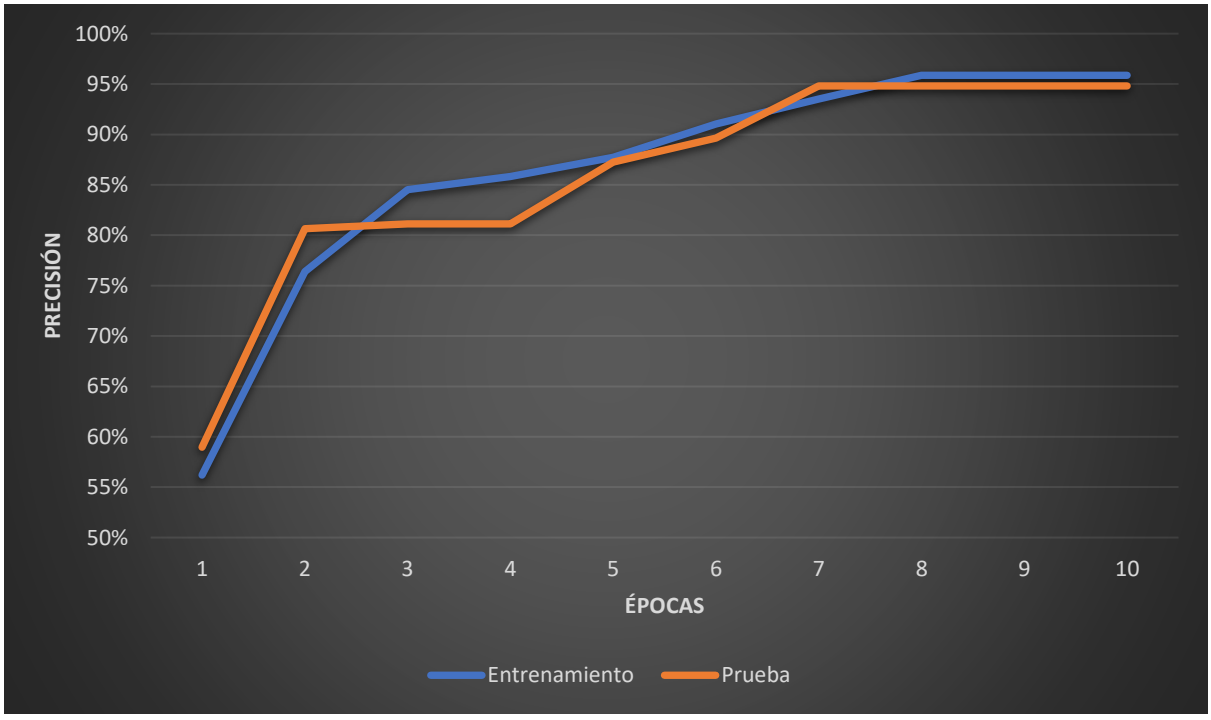
Evaluación de eficiencia de AutoKeras con datos de Mobile Price Classification.



Finalizando con las implementaciones de los diferentes conjuntos de datos seleccionados, el ultimo elegido es el llamado Milk Quality Prediction [Predicción de la calidad de la leche]. En el caso de la herramienta AutoKeras, la curva de aprendizaje, representada en la figura 19, mostró qué tanto las curvas de entrenamiento como las de prueba se ajustan de una forma similar, comenzando con una primera época un poco baja.

Figura 19

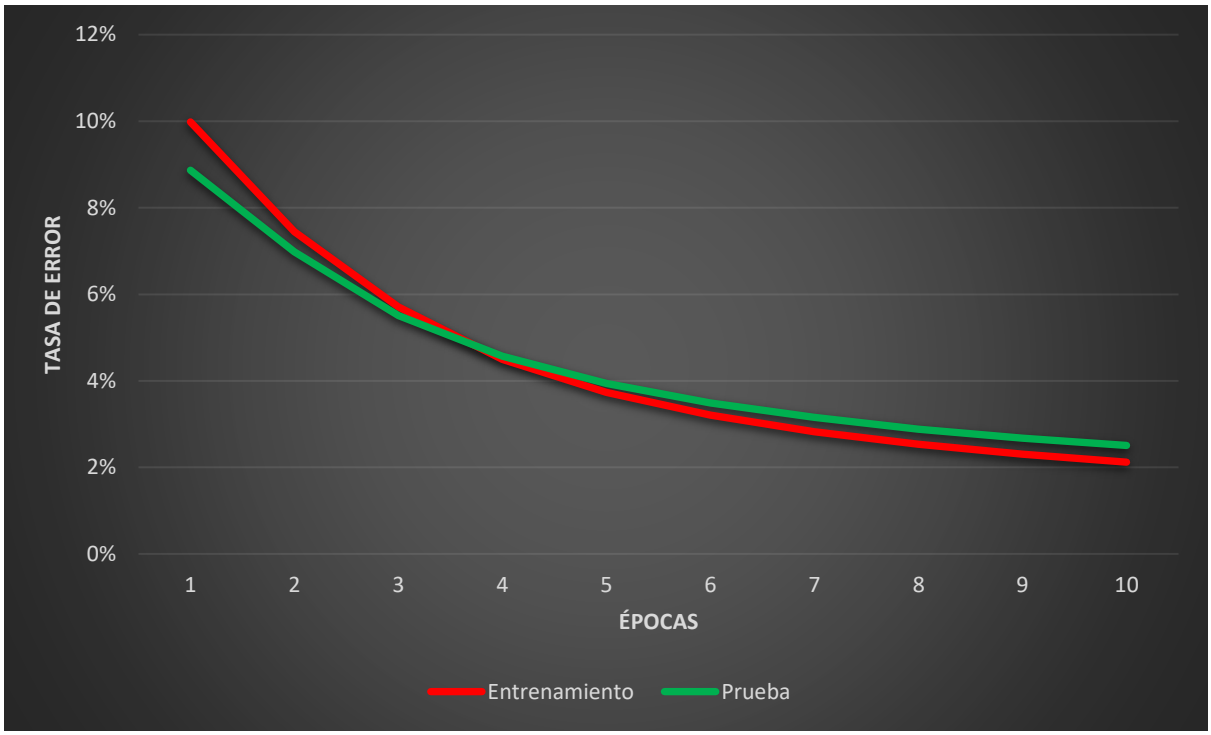
Curva de aprendizaje de AutoKeras con datos de Milk Quality Prediction.



Continuando con AutoKeras y Milk Quality Prediction, se graficó su evaluación de eficiencia, presente en la figura 20. En este caso, el valor de error de entrenamiento en la primera época es elevado, llegando casi al 10%, sin embargo, a medida que el modelo es entrenado este valor se va haciendo cada vez más pequeño para así lograr que en la última época sea de un valor aproximado al 2%, mejorando su desempeño y disminuyendo su tasa de error.

Figura 20

Evaluación de eficiencia de AutoKeras con datos de Milk Quality Prediction.

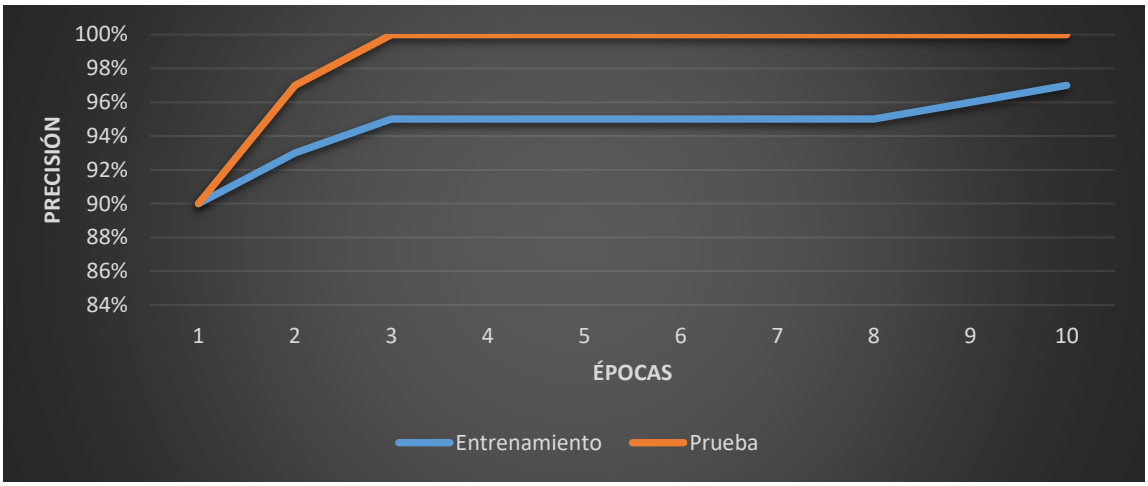


Por otro lado, siguiendo con la segunda herramienta utilizada para la recolección y análisis de datos, se llevaron pruebas de modelos con Auto-Pytorch. Antes de exponer los resultados, debemos indicar que no fue posible conseguir los valores resultantes de la tasa de error de cada uno de los modelos entrenados, por lo que los gráficos de evaluación de eficiencia no pudieron ser obtenidos.

Comenzando con el conjunto de datos Breast Cancer, se pudo graficar su curva de aprendizaje, representada en la figura 21, mostró qué tanto las curvas de entrenamiento como las de prueba se ajustan de una forma un tanto similar, comenzando con una primera época con un valor muy alto, y finalizando con una precisión del 98,5% promedio.

Figura 21

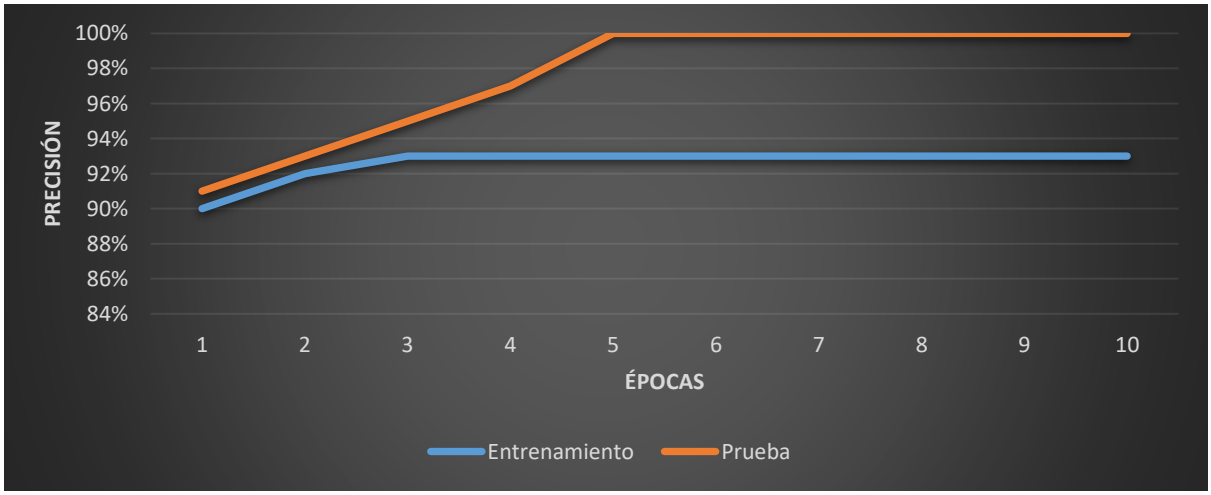
Evaluación de eficiencia de Auto-Pytorch con datos de Breast Cancer.



Siguiendo con los conjuntos de datos seleccionados, el próximo a evaluar fue el llamado Mobile Price Classification con la herramienta Auto-Pytorch. En el caso de la curva de aprendizaje, representada en la figura 22, se puede observar que tanto las curvas de entrenamiento como las de prueba se ajustan a los datos con un comportamiento similar. También, se dilucidó que durante las primeras épocas su valor de precisión es moderadamente alto, y a medida que se entrena el modelo va incrementando de manera escasa.

Figura 22

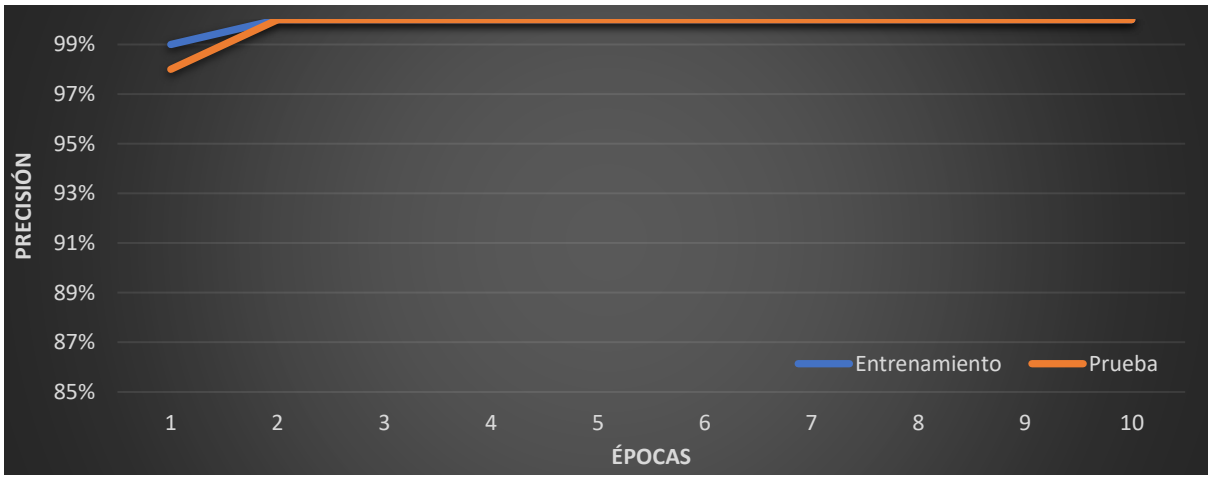
Evaluación de eficiencia de Auto-Pytorch con datos de Mobile Price Classification.



Finalizando con las implementaciones de los diferentes conjuntos de datos seleccionados, el ultimo elegido es el llamado Milk Quality Prediction [Predicción de la calidad de la leche]. En el caso de la herramienta Auto-Pytorch, la curva de aprendizaje está representada en la figura 23 a continuación.

Figura 23

Evaluación de eficiencia de Auto-Pytorch con datos de Milk Quality Prediction.

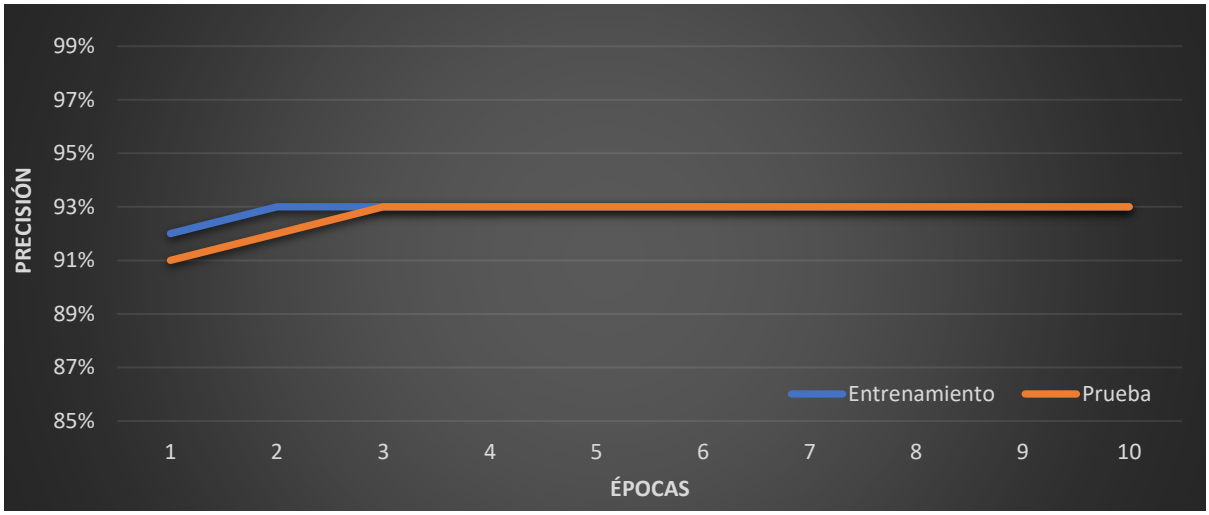


Por último, finalizando con la tercera y última herramienta utilizada para la recolección y análisis de datos, se llevaron a cabo pruebas de modelos con Auto-Sklearn. Antes de exponer los resultados, debemos indicar que, al igual que con la herramienta Auto-Pytorch, no fue posible conseguir los valores resultantes de la tasa de error de cada una de las épocas entrenadas debido a que la herramienta no lo permite, de manera que los gráficos de evaluación de eficiencia no pudieron ser alcanzados.

Comenzando con el conjunto de datos Breast Cancer, se pudo graficar su curva de aprendizaje de manera satisfactoria, presente en la figura 24. En ella se muestra que durante las primeras épocas su grado de precisión comienza con un valor bueno, pero a medida que se entrena el modelo, no tiene una incrementación notable.

Figura 24

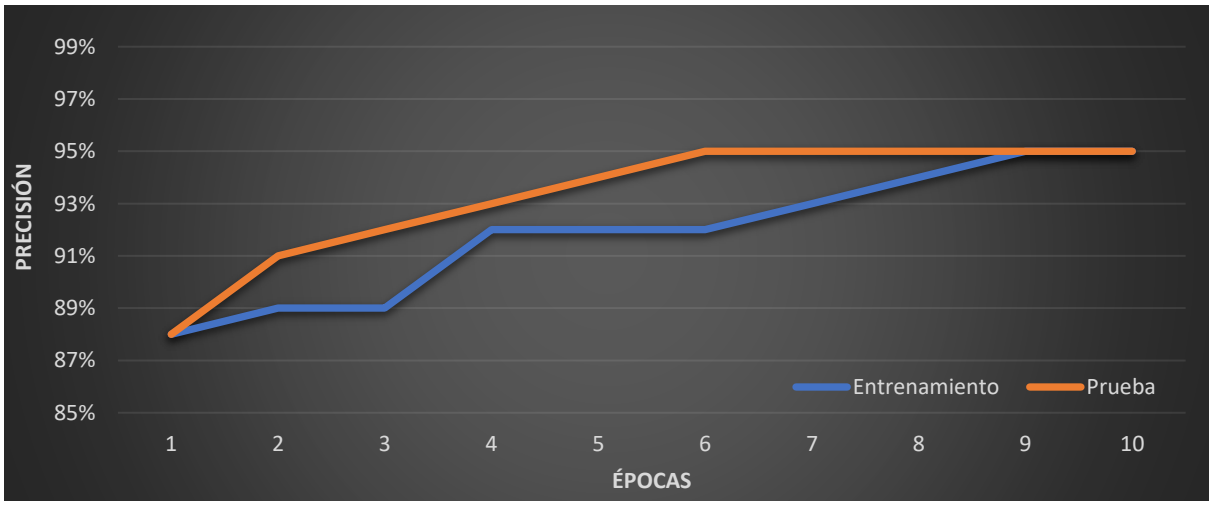
Evaluación de eficiencia de Auto-Sklearn con datos de Breast Cancer.



Siguiendo con los conjuntos de datos seleccionados, el próximo a evaluar fue el llamado Mobile Price Classification con la herramienta Auto-Sklearn. En el caso de la curva de aprendizaje, representada en la figura 25, se puede observar que tanto las curvas de entrenamiento como las de prueba se ajustan a los datos con un comportamiento similar. También, se verificó que durante las primeras épocas su valor de precisión es un poco bajo, pero a medida que se entrena el modelo, va incrementando cada vez más.

Figura 25

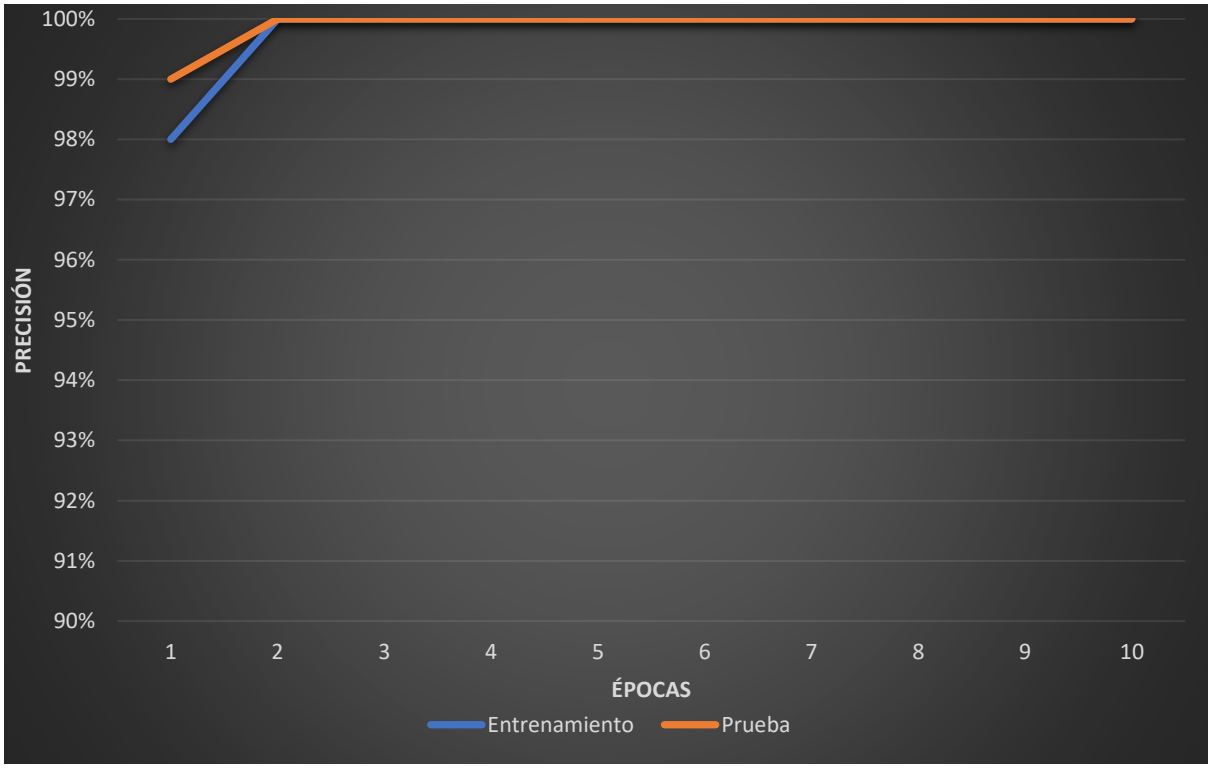
Evaluación de eficiencia de Auto-Sklearn con datos de Mobile Price Classification.



Finalizando con las implementaciones de los diferentes conjuntos de datos seleccionados, el último elegido es el llamado Milk Quality Prediction. En el caso de la herramienta Auto-Sklearn, la curva de aprendizaje, representada en la figura 26, mostró qué tanto las curvas de entrenamiento como las de prueba se ajustan de una forma similar, con un valor de precisión absoluto.

Figura 26

Evaluación de eficiencia de Auto-Sklearn con datos de Milk Quality Prediction



Como último paso de análisis de datos, se detalla a continuación los algoritmos elegidos para cada herramienta de AutoML. Sin embargo, vale aclarar que tanto la herramienta AutoKeras como la herramienta Auto-Pytorch no nos brindan esta opción de forma nativa dentro de su entorno, por lo que no se pudo obtener esta información. Siguiendo con Auto-Sklearn con los diferentes conjuntos de datos utilizados en este trabajo, se denota en la tabla 2 que la herramienta detectó como mejor algoritmo de AutoML es el llamado Random Forest o bosque aleatorio en el caso de Breast Cancer y Milk Quality Prediction. Sin embargo, en el

caso del conjunto de datos Mobile price Classification, el algoritmo por excelencia es MLP (perceptrón multicapa).

El algoritmo Random Forest o Bosque aleatorio ajusta una serie de clasificadores de árboles de decisión en varias submuestras del conjunto de datos y utiliza el promedio para mejorar la precisión y controlar el sobreajuste (Scikit Learn, 2022).

Por otro lado, el algoritmo MLP o perceptrón multicapa, es una clase de red neuronal artificial utilizada para abordar problemas de clasificación. Este tipo de algoritmo consta de al menos tres capas de nodos: una capa de entrada, una capa oculta y una capa de salida. En un algoritmo de perceptrón multicapa, múltiples capas densas están conectadas de tal manera que las salidas de una capa están completamente conectadas a las entradas de la siguiente (TensorFlow, 2022).

Tabla 2

Algoritmos elegidos por la herramienta Auto-Sklearn con los diferentes conjuntos de datos.

Conjunto de datos	**Tipo de algoritmo**	**Perdida en el conjunto de validación**	**Duración (segundos)**
Breast Cancer	Random Forest	0,006623	1,438864
Mobile Price Classification	MLP (Multi-layer Perceptron)	0,060606	3,799774
Milk Quality Prediction	Random Forest	0,003571	2,261252

6 – Elaboración del informe

6.1 Estudio comparativo de las herramientas

Durante el presente trabajo, el objetivo general planteado fue poder comparar las herramientas más utilizadas de soporte al proceso de desarrollo de aplicaciones de aprendizaje automático automatizado en tareas de clasificación. Estas herramientas, llamadas AutoKeras, Auto-Sklearn y Auto-Pytorch, resultaron un tanto amistosas en algunos casos a la hora de ser implementadas. Como se percibió anteriormente, se llevó a cabo en el capítulo 5 del presente trabajo su recolección y análisis de los datos planteados en las metodologías del capítulo 3. A continuación, se realizó un estudio comparativo de las herramientas ya mencionadas, resumiendo detalladamente los datos fundamentales obtenidos a lo largo del trabajo.

Siguiendo un poco lo ya mencionado en este capítulo, dentro de la recolección de datos con los métodos definidos en el apartado 3.3, se presenta una tabla comparativa de los resultados obtenidos en los diferentes entrenamientos.

Tabla 3

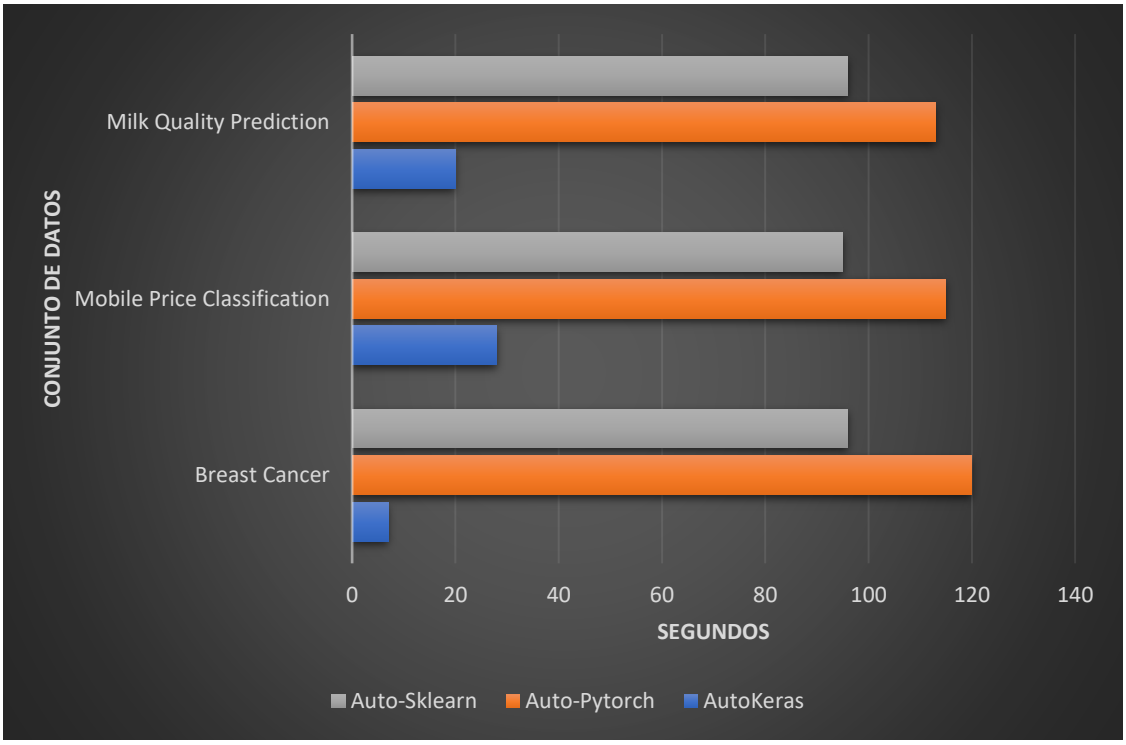
Comparación de modelos entrenados.

		Binario	No Binario	
		Breast Cancer	Mobile Price Classification	Milk Quality Prediction
AutoKeras	Velocidad de entrenamiento	7 seg.	28 seg.	20 seg.
	Tasa de error	2,94 %	3,17 %	1,63 %
	Precisión	95,61 %	88,25 %	97,64 %
Auto-Pytorch	Velocidad de entrenamiento	120 seg.	115 seg.	113 seg.
	Tasa de error	2,6 %	6,7 %	0 %
	Precisión	97,3 %	93,2 %	100 %
Auto-Sklearn	Velocidad de entrenamiento	96 seg.	95 seg.	96 seg.
	Tasa de error	4,3 %	2,6 %	0 %
	Precisión	93,5 %	95,2 %	100 %

Observando la tabla 3, que compara los resultados obtenidos en los diferentes entrenamientos, se puede establecer relaciones acerca de las diferentes herramientas en orden de velocidad de entrenamiento, tasa de error y precisión. A continuación, se establece una comparación entre las velocidades de entrenamiento de los diferentes modelos (presente en la figura 27).

Figura 27

Comparación entre las velocidades de entrenamiento de los diferentes modelos.

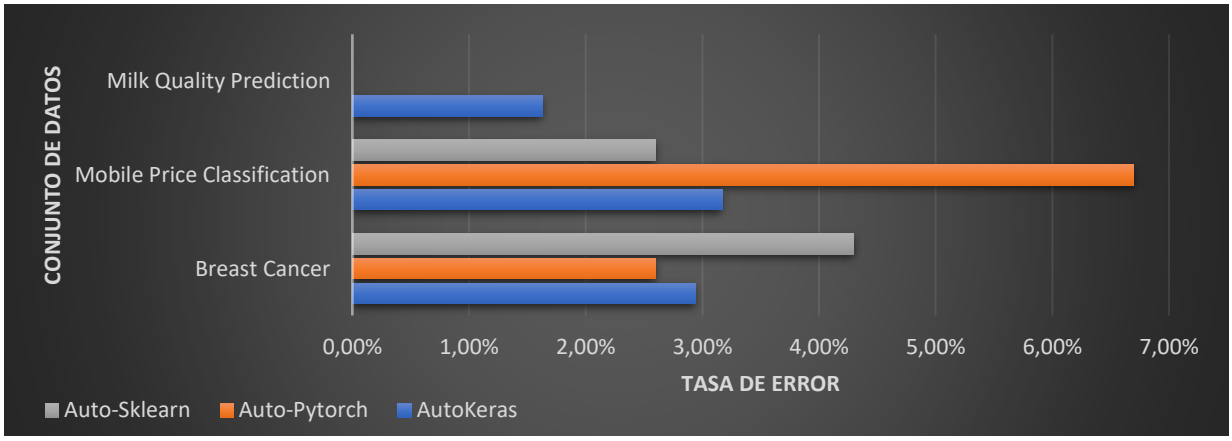


En la figura 27, se puede denotar que claramente AutoKeras tiene una velocidad de entrenamiento mucho mayor a la del resto de las herramientas, completando el entrenamiento en casi 5 veces menos en algunos casos.

Siguiendo con las comparaciones, la próxima a hacer fue la de las tasas de errores obtenidas de cada uno de los modelos, presentada a continuación.

Figura 28

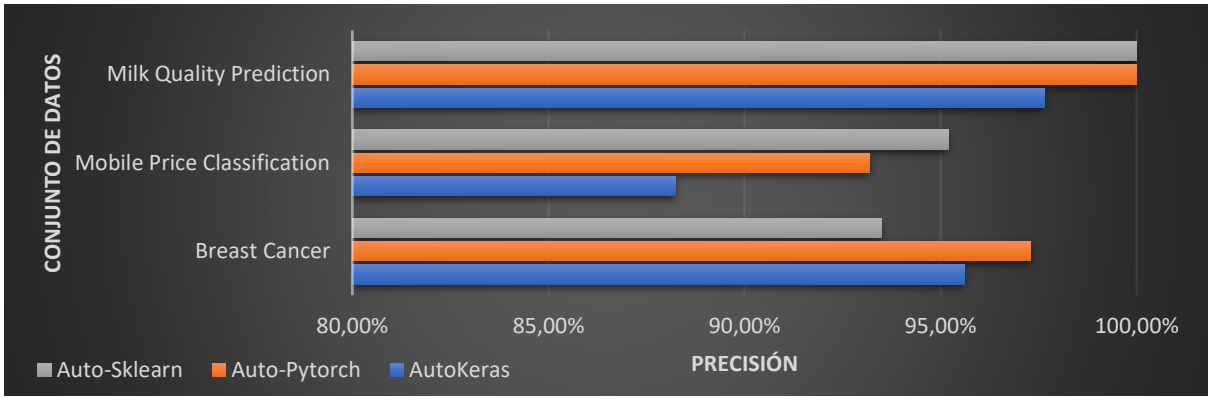
Comparación entre las tasas de errores de los diferentes modelos.



En la figura 28, se puede observar claramente que la menor tasa de error para los modelos binarios corresponde a Auto-Pytorch, pero que, en el caso de los no binarios, no hay un favorito, todas las herramientas están parejas en promedio. Una aclaración: en este caso las barras que no aparecen en la imagen corresponden a una tasa de error de la herramienta del 0%, lo que es un dato a resaltar. Finalizando con estas comparaciones, la última a realizar fue la de la precisión de cada uno de los modelos entrenados. Esta comparación está representada en la figura 29.

Figura 29

Comparación del nivel de precisión de los diferentes modelos.



En la figura 29, en promedio el modelo que mejores resultados mostró fue Auto-Sklearn, pero de muy cerca le sigue Auto-Pytorch. Sin embargo, algo a remarcar es que todos los modelos obtuvieron un valor de precisión muy elevado en todos los entrenamientos.

Luego de finalizar con las comparaciones acerca de las técnicas de recolección de datos, se prosiguió con su equivalente sobre las técnicas de análisis de datos. Antes de comenzar, se hizo imposible poder obtener los valores resultantes de la tasa de error de cada uno de los modelos entrenados con las herramientas Auto-Sklearn y Auto-Pytorch, por lo que los gráficos de evaluación de eficiencia no pudieron ser obtenidos. A continuación, se presentan comparaciones sobre las diferentes curvas de precisiones promedio obtenidas en conjuntos de datos binarios (dentro de la figura 30) y de no binarios (dentro de la figura 31). Concluyendo con las comparaciones, queda en evidencia que el modelo más preciso al finalizar los entrenamientos es en los modelos binarios es Auto-Pytorch, mientras que en los no binarios lo es Auto-Sklearn, pero con un valor casi idéntico a Auto-Pytorch.

Figura 30

Comparación entre las curvas de precisiones promedio de los modelos binarios.

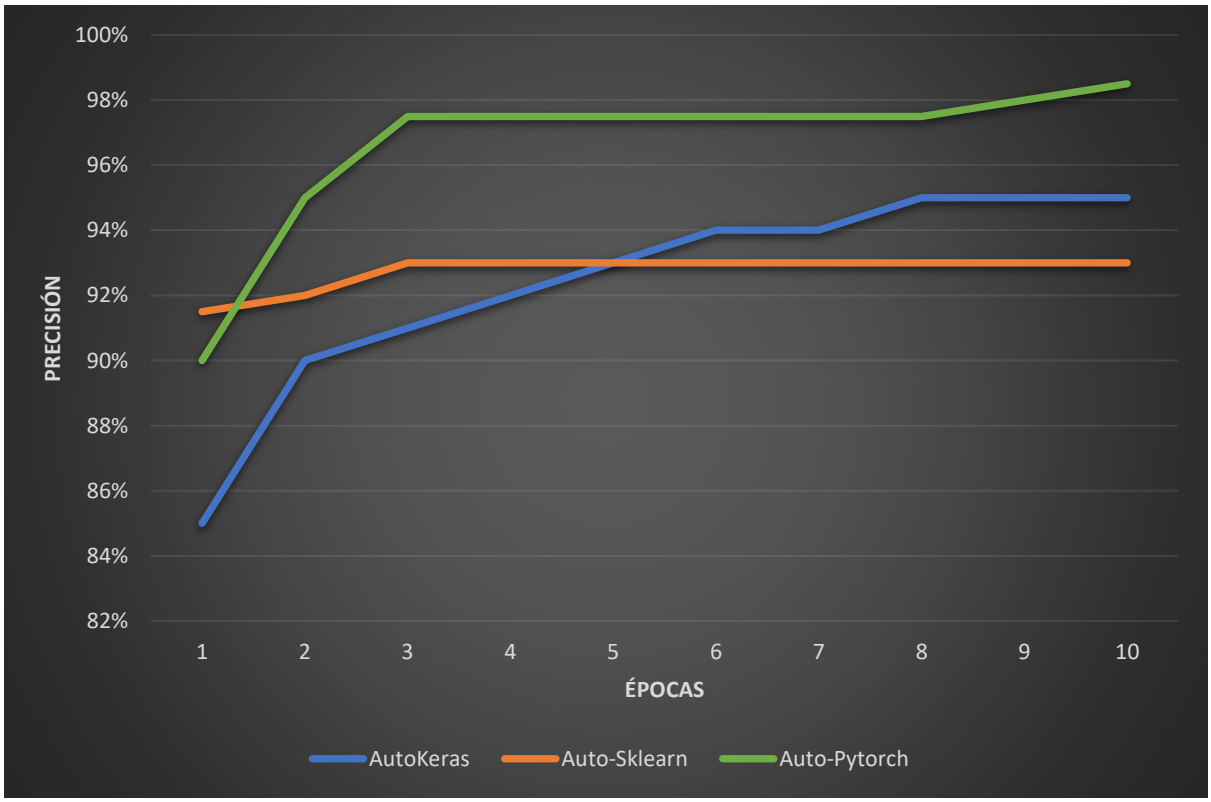
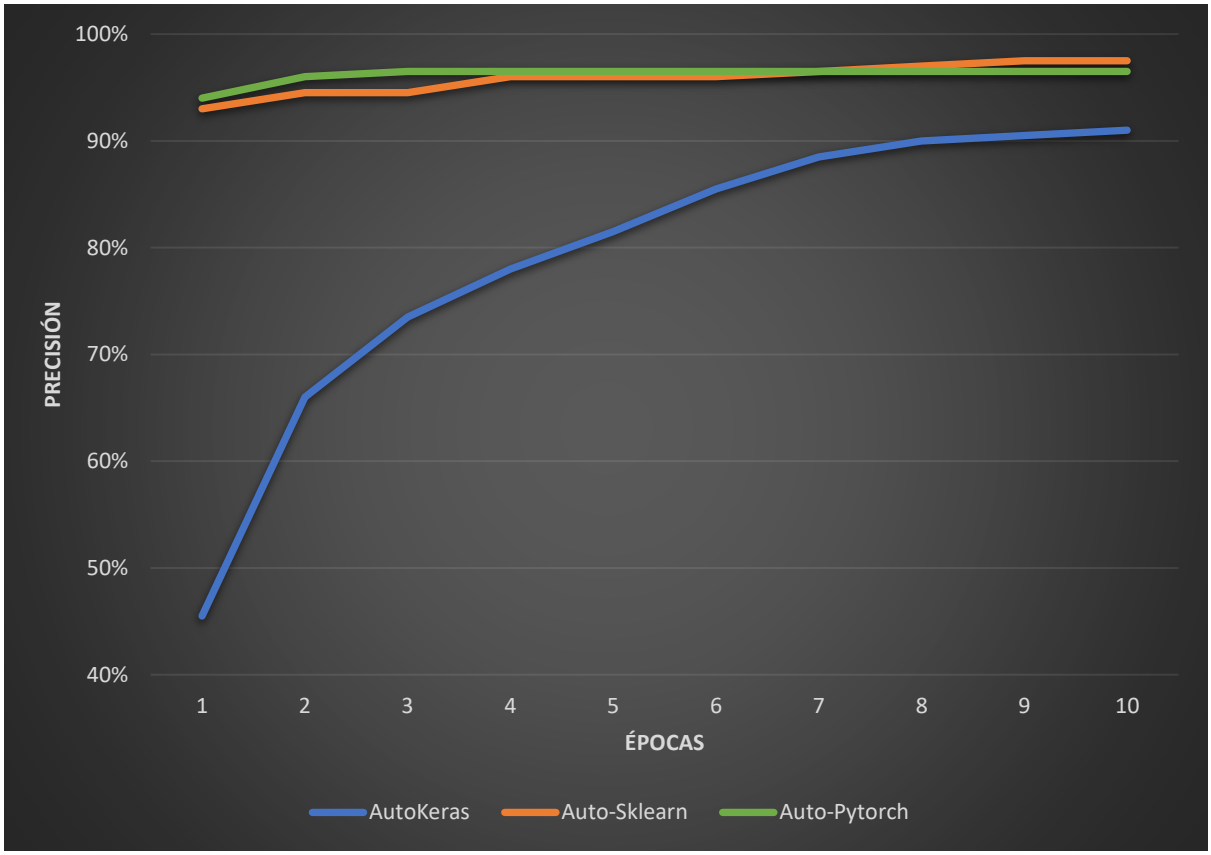


Figura 31

Comparación entre las curvas de precisiones promedio de los diferentes modelos no binarios.



6.2 Guía de buenas prácticas

Todas las herramientas que se utilizan sin previo conocimiento son difíciles de entender. Por esta razón, se llevó a cabo una guía de buenas prácticas, comprendiéndola con un conjunto de recomendaciones acerca de los cuales nos basamos para llevar a cabo las implementaciones y poder así evitar errores acerca de las herramientas en cuestión en este trabajo.

Tanto AutoKeras (autokeras.com) como Auto-Sklearn (automl.github.io/auto-sklearn) y Auto-Pytorch (automl.github.io/Auto-PyTorch) cuentan con páginas webs oficiales para poder consultar. Sin embargo, solo el sitio web de la primera herramienta está desarrollado de manera amigable, con ejemplos y una explicación más en profundidad sobre las diferentes

opciones para implementar. Algunas buenas prácticas y recomendaciones generales para todas las herramientas son:

- Diferenciar que tipo de implementación se desea y observar un tutorial en la página oficial;
- Probar el ejemplo entero sin modificaciones para ver su funcionamiento;
- Modificar los datos de entrada con el conjunto de datos que se desee;
- Predecir nuevos resultados con el modelo obtenido y observar su desempeño;

A continuación, las implementaciones comienzan a variar, es por eso que diferenciaremos cada una de las herramientas. En el caso específico de la herramienta AutoKeras, las buenas prácticas y recomendaciones que se lograron diferenciar son:

- Luego de separar los datos en entrenamiento y prueba, se deben convertir estos datos en array, un tipo de datos de lista separados por comas. Una forma muy sencilla de hacerlo es utilizar la herramienta NumPy (numpy.org) de la siguiente manera:

```
import numpy as np
x_train = np.array(X_train)
y_train = np.array(y_train)
x_test = np.array(X_test)
y_test = np.array(y_test)
```

- A la hora de entrenar el modelo, guardarlo en una variable para poder saber sus valores históricos, como se muestra a continuación:
  `history = modeloEntrenado.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=10)`
- Evaluar la red del siguiente modo:

```
modeloEntrenado.evaluate(x_test,y_test)
```

- Implementar su curva de aprendizaje con la libreria de diseño Matplotlib (matplotlib.org):

```
from matplotlib import pyplot
f, ax = pyplot.subplots()
ax.set_ylim(bottom=0.75)
ax.plot(history.history['accuracy'], label='entrenamiento')
ax.plot(history.history['val_accuracy'], label='prueba')
pyplot.legend()
pyplot.show(f)
```

Seguidamente, las recomendaciones para implementar Auto-Sklearn son las siguientes:

- A la hora de obtener el tiempo empleado para el entrenamiento, deberemos realizarlo con una herramienta extra ya que Auto-Sklearn no nos lo brinda. Para esto, se utilizó la librería nativa de Python llamada Time (docs.python.org/3/library/time) de la siguiente manera:

```
start = time.time()
modeloEntrenado.fit(X_train, y_train)
stop = time.time()
print(f"Tiempo empleado: {stop - start} segundos")
```

- Evaluar la red del siguiente modo:

```
predictions = modeloEntrenado.predict(X_test)
print(sklearn.metrics.accuracy_score(y_test, predictions))
```

- Debido a que Auto-Sklearn no cuenta con un historial sobre cada uno de los valores de los entrenamientos a los cuales podamos acceder, se debió utilizar la opción que provee la herramienta para obtener su curva de aprendizaje, ayudado por la librería de diseño Matplotlib (matplotlib.org):

```
from matplotlib import pyplot
poT = modeloEntrenado.performance_over_time_
poT.plot(
    kind="line",
    legend=True,
    grid=True,
)
pyplot.show()
```

Por último, las buenas prácticas y recomendaciones que se lograron diferenciar para la herramienta Auto-Pytorch son:

- Al igual que con Auto-Sklearn, no se puede obtener de forma nativa el tiempo de entrenamiento total, por lo que se debió utilizar la librería nativa de Python llamada Time (docs.python.org/3/library/time) de la siguiente manera:

```
start = time.time()
modeloEntrenado.search(
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    optimize_metric='accuracy',
    total_walltime_limit=100,
    func_eval_time_limit_secs=10
)
stop = time.time()
print(f"Tiempo empleado: {stop - start} segundos")
```

- Evaluar la red del siguiente modo:

```
score = api.score(y_test, predictions)
print(score)
```

- Debido a que Auto- Pytorch no cuenta con un historial sobre cada uno de los valores de los entrenamientos de forma nativa, se correspondió utilizar la opción que suministra la herramienta para obtener su curva de aprendizaje, ayudado por la librería de diseño Matplotlib (matplotlib.org):

```
from autoPyTorch.utils.results_visualizer import
PlotSettingParams,ColorLabelSettings
metric_name = 'accuracy'
params = PlotSettingParams(
    xlabel='Segundos',
    ylabel='Accuracy',
    legend=True,
    figname= None,
    show=True)
```

```
modeloEntrenado.plot_perf_over_time(
    metric_name=metric_name,
    plot_setting_params=params,
    color_label_settings= ColorLabelSettings(single_train=('red',
None), single_opt=('white', None), single_test=('green', None),
ensemble_train=('white', None), ensemble_test=('white', None))
)
```

7 – Conclusiones

La motivación para la realización de este trabajo surge desde la necesidad de resolver la problemática que tienen las personas sin experiencia previa en el mundo del aprendizaje automático a la hora de comenzar con una implementación de este estilo. Con el universo de herramientas disponibles actualmente, se hace imposible elegir una si no se tiene una base de conocimiento sobre ellas, incluso muchas veces dejando esto al azar. Debido a esto, nos propusimos determinar cuál es la herramienta de soporte más eficiente para el proceso de desarrollo de aplicaciones de aprendizaje automático automatizable en tareas de clasificación.

En primer lugar, podemos concluir que todas las herramientas tratadas en este trabajo poseen un muy buen nivel de precisión a la hora de entrenar modelos con los datos brindados. Si bien difieren en los valores finales de precisión, tasa de error y velocidades de entrenamiento, la que se elija va a dar como resultado un valor con una precisión elevada.

En segundo lugar, tanto Auto-Sklearn como Auto-Pytorch son mucho menos amigables y ofrecen menos opciones personalizables comparándolos con AutoKeras. En los primeros dos, no es posible ver los resultados de cada uno de los entrenamientos, siendo su tasa de error inconseguible. Sin embargo, esta no es una complicación importante, ya que los datos finales de error de la mejor época entrenada pudieron ser conseguidos, quedando este método de análisis de datos más como un estudio comparativo e histórico acerca de las diferentes épocas de cada uno de los entrenamientos dentro del respectivo modelo. Otro punto a recalcar es que Auto-Pytorch nos obliga a convertir todos los valores a entrenar en numéricos, ya que no acepta entradas con valores de otro tipo, mientras que AutoKeras y AutoSklearn proveen esta conversión de manera transparente al programador, lo cual facilita la tarea de entrenamiento cuando los datos disponibles no son numéricos.

En tercer lugar, consideramos que el objetivo general de comparar las herramientas más utilizadas de soporte al proceso de desarrollo de aplicaciones de aprendizaje automático automatizado en tareas de clasificación fue cumplido. Este fue llevado a cabo investigando el ciclo de vida de las mismas para luego analizar las diferentes herramientas de este estilo en tareas de clasificación y seleccionar las tres más utilizadas y beneficiosas. Seguidamente, se entrenaron modelos de aprendizaje automático automatizado utilizando cada una de las

herramientas seleccionadas y por último, se elaboraron estudios comparativos de las herramientas con una lista de recomendaciones y buenas prácticas de los resultados obtenidos.

En cuarto lugar, se pudo determinar que la utilización de las herramientas de software de código abierto AutoKeras, Auto-Pytorch y Auto-Sklearn, con Python como lenguaje de programación, permiten automatizar diferentes modelos de aprendizaje automático en tareas de clasificación de forma rápida, segura y eficiente.

Por último, podemos concluir que, si bien todas las herramientas son eficaces para poder entrenar modelos de aprendizaje automático automatizado, la que mayor eficacia tuvo en promedio tanto en modelos binarios como no binarios fue Auto-Pytorch. Asimismo, no podemos dejar de recalcar que, dependiendo lo que busque el desarrollador, puede optar por una u otra según sus necesidades debido a que con todas se obtendrán muy buenos resultados. Por ejemplo, en el caso que se necesite mayor velocidad, la más rápida resultó ser AutoKeras.

En trabajos futuros, nos proponemos comprobar si en el caso del universo de datos no numéricos (análisis de texto y análisis de imágenes) las conclusiones halladas en esta investigación se mantienen.

Acrónimos

ANN	Artificial neural networks [Redes neuronales artificiales]
ML	Machine Learning [aprendizaje automático]
AutoML	Automated Machine Learning [aprendizaje automático automatizado]

Referencias

AutoKeras. (2022). *AutoKeras: Home*. https://autokeras.com/

AutoML org. (s.f.-a). (2022). *Auto-Sklearn*. https://www.automl.org/automl/auto-sklearn/

AutoML org. (s.f.-b). (2022). *Auto-PyTorch*. https://www.automl.org/ automl/autopytorch/

Bender, A. y Nicolet, S. (26-30 octubre de 2020). *Evaluación Comparativa de Herramientas AutoML de Código Abierto en Tareas de Regresión*. XXI Simposio Argentino de Inteligencia Artificial, Buenos Aires, Argentina.

Elgendy, M. (2020). *Deep Learning for Vision Systems*. Manning Publications.

Ferreira, L., Pilastri, A., Martins, C. M., Pires, P. M. y Cortez, P. (28-22 de julio de 2021). *A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost* [Resumen de presentación de la conferencia]. Conferencia Internacional Conjunta sobre Redes Neuronales 2021 (IJCNN). http://repositorium.uminho.pt/bitstream/1822/74125/1/automl_ijcnn.pdf

Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B. y Vanschoren, J. (14 de junio de 2019). *An Open Source AutoML Benchmark* [Resumen de presentación de la conferencia]. Sexto taller de Conferencia Internacional sobre Aprendizaje Automático sobre aprendizaje automático automatizado, Long Beach, Estados Unidos. https://sites.google.com/view/automl2019icml/?pli=1

Glassdoor. (2022). *Sueldos de Machine Learning Engineer*. https://www.glassdoor.com.ar/Sueldos/buenos-aires-machine-learning-engineer-sueldo-SRCH_IL.0,12_IM963_KO13,38.htm

Goodfellow, I., Bengio, Y. y Courville, A. (2016). *Deep Learning*. The MIT Press.

Google Developers. (s.f.-a). (2022). *What is Machine Learning?* https://developers.google.com/machine-learning/intro-to-ml/what-is-ml

Google Developers. (s.f.-b). (2022). *Training and Test Sets: Splitting Data*. https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data

Google Developers. (s.f.-c). (2022). *Supervised Learning.* https://developers.google.com/machine-learning/intro-to-ml/supervised

Google Developers. (s.f.-d). (2022). *Validation Set: Another Partition.* https://developers.google.com/machine-learning/crash-course/validation/another-partition

Google Developers. (s.f.-e). (2022). *Text Classification: Introduction.* https://developers.google.com/machine-learning/guides/text-classification

Google Developers. (s.f.-f). (2022). *ML Practicum: Image Classification.* https://developers.google.com/machine-learning/practica/image-classification

Google Developers. (s.f.-g). (2022). *Glosario sobre aprendizaje automático.* https://developers.google.com/machine-learning/glossary/#numerical_data

Hanussek, M., Blohm, M. y Kintz, M. (26-28 de diciembre de 2020). *Can AutoML outperform humans? An evaluation on popular OpenML datasets using AutoML Benchmark* [Resumen de presentación del congreso]. Segundo Congreso Internacional de Inteligencia Artificial, Robótica y Control 2020 (AIRC 2020). https://arxiv.org/ftp/arxiv/papers/2009/2009.01564.pdf

Hurwitz, J. y Kirsch, D. (2018). *Machine Learning for dummies*. IBM.

IBM Cloud Education (2020). *Neural Networks*. https://www.ibm.com/cloud/learn/neural-networks

Microsoft. (2022). *¿Qué es el aprendizaje automático automatizado (AutoML)?* https://docs.microsoft.com/es-es/azure/machine-learning/concept-automated-ml

Microsoft Learn. (2022). *Evaluate automated machine learning experiment results*. https://learn.microsoft.com/en-us/azure/machine-learning/how-to-understand-automated-ml#classification-metrics

Milutinovic, M., Schoenfeld, B., Martinez-Garcia, D., Ray, S., Shah, S. y Yan, D. (18 de Julio de 2020). *On Evaluation of AutoML Systems* [Resumen de presentación de la conferencia]. Séptimo taller de Conferencia Internacional sobre Aprendizaje Automático sobre aprendizaje automático automatizado. https://sites.google.com/view/automl2020/home

Murillo Gonzalez, M. (2021). *Creación de librería para predicción de series temporales de señales industriales*. [Tesis de máster, Universidad Internacional Menéndez Pelayo]. Repositorio abierto de la universidad de Cantabria. https://repositorio.unican.es/xmlui/handle/10902/24930

Namdari, R. (2022). *Breast Cancer*. Kaggle. https://www.kaggle.com/datasets/reihanenamdari/breast-cancer

Rajendran, S. (2022). *Milk Quality Prediction*. Kaggle. https://www.kaggle.com/datasets/cpluzshrijayan/milkquality

Russell, S. y Norvig, P. (2004). *Inteligencia artificial. Un enfoque moderno*. *Segunda edición*. Pearson Prentice Hall.

Scikit Learn. (2022). *sklearn ensemble Random Forest Classifier*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Sharma, A. (2017). *Mobile Price Classification*. Kaggle. https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification

Song, Q., Jin, H. y Hu, X. (2022). *Automated Machine Learning in Action*. Manning Publications.

TensorFlow. (2022). *Multilayer perceptrons for digit recognition with Core APIs*. https://www.tensorflow.org/guide/core/mlp_core

Zimmer, L., Lindauer, M. y Hutter, F. (2021). Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9), 3079-3090. https://doi.org/10.1109/tpami.2021.3067763

Anexo

Anexo A. Código Fuente

Anexo de código 1

```
# Importacion de librerias

import numpy as np

import pandas as pd

import plotly.express as px

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

import os

from sklearn.datasets import load_files

import autokeras as ak

from sklearn.datasets import load_breast_cancer

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split


# Cargando datos

TrainDataSet= load_breast_cancer()

df = pd.DataFrame(data=TrainDataSet.data,
columns=TrainDataSet.feature_names)

df['target'] = TrainDataSet.target

df.head()

df[df.columns].describe()


# Analisis exploratorio de datos

# Cantidad de elementos por categoría del elemento target (indicador
de cancer)
```

```
fig = px.histogram(df,
                   x='target',
                   color = 'target',
                   text_auto=True,
                   color_discrete_sequence=px.colors.qualitative.G10,
                   template='simple_white',
                   labels={"target": "Target"},
                   title='Target Value Count')
fig.update_layout(bargap=0.1,
xaxis_title="Target",yaxis_title="Count")
fig.show()


#Limpieza de datos
input_cols = df.columns[:-1]
input_cols
target_col =  df.columns[-1]
target_col
inputs_df = df[list(input_cols)].copy()
targets = df[(target_col)]
df.head(20)


#Modelo de AutoML
#Separo datps de las etiquetas de los resultados
x = df.drop(['target'],axis=1)
y = np.array(df['target'])
#Separo datos en trainig y test
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size =
0.2)
x_train = np.array(X_train)
```

```
y_train = np.array(y_train)

x_test = np.array(X_test)

y_test = np.array(y_test)

# Iniciación del Clasificador de datos estructurados de AutoKeras

clf = ak.StructuredDataClassifier(

    overwrite=True, max_trials=3

)


# Entrenamiento de datos

history = clf.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=10)


# Predicción del mejor modelo resultante

predicted_y = clf.predict(x_test)

res = clf.evaluate(x_test,y_test)


# Predicción de nuevos resultados estimados

y_pred = clf.predict(x_test)

for i in range(50):

  print("valor real: ", y_test[i])

  print("valor estimado: ", y_pred[i])

  print("---------------------------------- ")

# Curva de aprendizaje

from matplotlib import pyplot

f, ax = pyplot.subplots()

ax.set_ylim(bottom=0.75)

ax.plot(history.history['accuracy'], label=' entrenamiento')

ax.plot(history.history['val_accuracy'], label='prueba')

pyplot.legend()
```

```
pyplot.show(f)


# Evaluación de eficiencia

fun, axis = pyplot.subplots()

axis.set_ylim(bottom=0)

axis.plot(history.history['loss'], color='red', label='entrenamiento'
 )

axis.plot(history.history['val_loss'],  color='green', label='prueba'
)

pyplot.legend()

pyplot.show(fun)
```

Anexo de código 2

```
import numpy as np

import pandas as pd

import plotly.express as px

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load_breast_cancer

import sklearn.metrics

import autosklearn.classification

import autosklearn.metrics

import time


#LOADING DATA

TrainDataSet= load_breast_cancer()

df = pd.DataFrame(data=TrainDataSet.data,
columns=TrainDataSet.feature_names)

df['target'] = TrainDataSet.target
```

```
df.head()


#EXPLORATORY DATA ANALYSIS OF BC
fig = px.histogram(df,
                   x='target',
                   color = 'target',
                   text_auto=True,
                   color_discrete_sequence=px.colors.qualitative.G10,
                   template='simple_white',
                   labels={"target": "Target"},
                   title='Target Value Count')


fig.update_layout(bargap=0.1,
xaxis_title="Target",yaxis_title="Count")
fig.show()


#DATA CLEANING
input_cols = df.columns[:-1]
input_cols
target_col =  df.columns[-1]
target_col


inputs_df = df[list(input_cols)].copy()
inputs_df
targets = df[(target_col)]
targets


#AUTOML MODEL
X = inputs_df
```

```
y = targets

X_train, X_test, y_train, y_test =
sklearn.model_selection.train_test_split(X,y, test_size = 0.2,
random_state=1)


automl = autosklearn.classification.AutoSklearnClassifier(

    time_left_for_this_task=100,

    per_run_time_limit=10

)


start = time.time()

history = automl.fit(X_train, y_train, dataset_name="Breast Cancer")

stop = time.time()

print(f"Training time: {stop - start}s")


#Evaluar red

predictions = api.predict(X_test)

print("Precision", sklearn.metrics.precision_score(y_test,
predictions))

print("Recall", sklearn.metrics.recall_score(y_test, predictions))

print("Loss", sklearn.metrics.log_loss(y_test, predictions))


print(api.show_models())

print(api.leaderboard())

print(pd.DataFrame(automl.cv_results_))


#Predecir nuevos resultados

y_pred = automl.predict(X_test)

y_test = np.array(y_test)
```

```
error = 0;

for i in range(len(y_test)):
  print("valor real: ", y_test[i])
  print("valor estimado: ", y_pred[i])
  if(y_test[i] != y_pred[i]):
    error = error + 1;
  print("---------------------------------- ")

error_rate = error  / len(y_test)
print("Error: ",error_rate)

#Curva de precision
poT = automl.performance_over_time_
poT.plot(
    x="Timestamp",
    kind="line",
    legend=True,
    title="Auto-sklearn accuracy over time",
    grid=True,
)
plt.show()
```

Anexo de código 3

```
import numpy as np
import pandas as pd
import plotly.express as px
```

```
import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load_breast_cancer

import sklearn.metrics

import time

from autoPyTorch.api.tabular_classification import
TabularClassificationTask


#LOADING DATA

TrainDataSet= load_breast_cancer()

df = pd.DataFrame(data=TrainDataSet.data,
columns=TrainDataSet.feature_names)

df['target'] = TrainDataSet.target

df.head()


#EXPLORATORY DATA ANALYSIS OF BC

fig = px.histogram(df,
                   x='target',
                   color = 'target',
                   text_auto=True,
                   color_discrete_sequence=px.colors.qualitative.G10,
                   template='simple_white',
                   labels={"target": "Target"},
                   title='Target Value Count')


fig.update_layout(bargap=0.1,
xaxis_title="Target",yaxis_title="Count")

fig.show()
```

```
#DATA CLEANING

input_cols = df.columns[:-1]

input_cols

target_col =  df.columns[-1]

target_col


inputs_df = df[list(input_cols)].copy()

inputs_df

targets = df[(target_col)]

targets


#AUTOML MODEL

X = inputs_df

y = targets

X_train, X_test, y_train, y_test =
sklearn.model_selection.train_test_split(X,y, test_size = 0.2,
random_state=1)


X_train = X_train.astype(np.float32)

X_test = X_test.astype(np.float32)

y_train = y_train.astype(np.float32)

y_test = y_test.astype(np.float32)


api = TabularClassificationTask()


start = time.time()

history = api.search(

    X_train=X_train,

    y_train=y_train,
```

```
    X_test=X_test,
    y_test=y_test,
    optimize_metric='accuracy',
    total_walltime_limit=100,
    func_eval_time_limit_secs=10
)
stop = time.time()
print(f"Training time: {stop - start}s")

#Evaluar red
predictions = api.predict(X_test)
print("Precision", sklearn.metrics.precision_score(y_test,
predictions))
print("Recall", sklearn.metrics.recall_score(y_test, predictions))
print("Loss", sklearn.metrics.log_loss(y_test, predictions))

score = api.score(y_test, predictions)
print(score)
print(api.show_models())

#Predecir nuevos resultados
y_pred = api.predict(X_test)
y_test = np.array(y_test)

error = 0;

for i in range(len(y_test)):
  print("valor real: ", y_test[i])
  print("valor estimado: ", y_pred[i])
```

```
  if(y_test[i] != y_pred[i]):
    error = error + 1;
  print("---------------------------------- ")

error_rate = error  / len(y_test)
print("Error: ",error_rate)

#Curva de precision
from autoPyTorch.utils.results_visualizer import
PlotSettingParams,ColorLabelSettings
metric_name = 'accuracy'

params = PlotSettingParams(
    xlabel='Segundos',
    ylabel='Accuracy',
    legend=True,
    figname= None,
    show=True

)

api.plot_perf_over_time(
    metric_name=metric_name,
    plot_setting_params=params,
    color_label_settings= ColorLabelSettings(single_train=('red',
None), single_opt=('white', None), single_test=('green', None),
ensemble_train=('white', None), ensemble_test=('white', None))
)
```

Anexo de código 4

```
# Importacion de librerias
from pprint import pprint
import sklearn.datasets
import sklearn.metrics
import pandas as pd
import numpy as np
import plotly.express as px
import autokeras as ak
import tensorflow as tf
import os
from sklearn.datasets import load_files

# Cargando datos
TrainDataSet= pd.read_csv('../content/train.csv')
TrainDataSet.head()

# Analisis exploratorio de datos
TrainDataSet.describe()
# Cantidad de elementos por categoría del elemento price_range
fig = px.histogram(TrainDataSet,
                   x='price_range',
                   color = 'price_range',
                   text_auto=True,
                   color_discrete_sequence=px.colors.qualitative.G10,
                   template='simple_white',
                   labels={"Price Range": "Price Range"},
                   title='Price Range Value Count')
```

```
fig.update_layout(bargap=0.1, xaxis_title="Price
Range",yaxis_title="Quantity")

fig.show()


#Limpieza de datos

input_cols = TrainDataSet.columns[:-1]

input_cols

target_col =  TrainDataSet.columns[-1]

target_col

inputs_df = TrainDataSet[list(input_cols)].copy()

targets = TrainDataSet[(target_col)]

df.head(20)


#Modelo de AutoML

#Separo datos de las etiquetas de los resultados

x = inputs_df

y = np.array(targets)

#Separo datos en trainig y test

X_train, X_test, y_train, y_test = train_test_split(x,y, test_size =
0.2)

x_train = np.array(X_train)

y_train = np.array(y_train)

x_test = np.array(X_test)

y_test = np.array(y_test)


# Iniciación del Clasificador de datos estructurados de AutoKeras

clf = ak.StructuredDataClassifier(

    overwrite=True, max_trials=3

)
```

```
# Entrenamiento de datos
history = clf.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=10)


# Predicción del mejor modelo resultante
predicted_y = clf.predict(x_test)
res = clf.evaluate(x_test,y_test)


# Predicción de nuevos resultados estimados
y_pred = clf.predict(x_test)
for i in range(50):
  print("valor real: ", y_test[i])
  print("valor estimado: ", y_pred[i])
  print("---------------------------------- ")


# Curva de aprendizaje
from matplotlib import pyplot
f, ax = pyplot.subplots()
ax.set_ylim(bottom=0.75)
ax.plot(history.history['accuracy'], label=' entrenamiento')
ax.plot(history.history['val_accuracy'], label='prueba')
pyplot.legend()
pyplot.show(f)


# Evaluación de eficiencia
fun, axis = pyplot.subplots()
axis.set_ylim(bottom=0)
axis.plot(history.history['loss'], color='red', label='entrenamiento'
 )
```

```
axis.plot(history.history['val_loss'],  color='green', label='prueba'
)
pyplot.legend()
pyplot.show(fun)
```

Anexo de código 5

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
import sklearn.metrics
import autosklearn.classification
import autosklearn.metrics
import time

#LOADING DATA
df= pd.read_csv('../content/train.csv')
df.head()



#DATA CLEANING
input_cols = df.columns[:-1]
input_cols
```

```
target_col =  df.columns[-1]
target_col


inputs_df = df[list(input_cols)].copy()
inputs_df
targets = df[(target_col)]
targets


#AUTOML MODEL
X = inputs_df
y = targets
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X,y, test_size = 0.2, random_state=1)


automl = autosklearn.classification.AutoSklearnClassifier(
    time_left_for_this_task=100,
    per_run_time_limit=10
)


start = time.time()
history = automl.fit(X_train, y_train, dataset_name="Mobile-Price-Classification")
stop = time.time()
print(f"Training time: {stop - start}s")
```

```
#Evaluar red
predictions = automl.predict(X_test)
print("Accuracy score", sklearn.metrics.accuracy_score(y_test, predictions))

automl.show_models()
automl.leaderboard()

#Predecir nuevos resultados
y_pred = automl.predict(X_test)
y_test = np.array(y_test)

error = 0;

for i in range(len(y_test)):
  print("valor real: ", y_test[i])
  print("valor estimado: ", y_pred[i])
  if(y_test[i] != y_pred[i]):
    error = error + 1;
  print("---------------------------------- ")

error_rate = error  / len(y_test)
print("Error: ",error_rate)
```

```
#Curva de precision
poT = automl.performance_over_time_
poT.plot(
    x="Timestamp",
    kind="line",
    legend=True,
    title="Auto-sklearn accuracy over time",
    grid=True,
)
plt.show()
```

Anexo de código 6

```
import numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
import sklearn.metrics
import time
from autoPyTorch.api.tabular_classification import TabularClassificationTask
```

```
#LOADING DATA

TrainDataSet= pd.read_csv('../content/train.csv')

TrainDataSet.head()



#DATA CLEANING

input_cols = TrainDataSet.columns[:-1]

input_cols

target_col =  TrainDataSet.columns[-1]

target_col


inputs_df = TrainDataSet[list(input_cols)].copy()

inputs_df

targets = TrainDataSet[(target_col)]

targets


#AUTOML MODEL

X = inputs_df

y = targets

X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X,y, test_size = 0.2, random_state=1)


X_train = X_train.astype(np.float32)
```

```
X_test = X_test.astype(np.float32)
y_train = y_train.astype(np.float32)
y_test = y_test.astype(np.float32)

api = TabularClassificationTask()

start = time.time()
history = api.search(
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    optimize_metric='accuracy',
    total_walltime_limit=100,
    func_eval_time_limit_secs=10
)
stop = time.time()
print(f"Training time: {stop - start}s")

#Evaluar red
predictions = api.predict(X_test)
print("Accuracy score", sklearn.metrics.accuracy_score(y_test, predictions))
```

```
print(api.sprint_statistics())
score = api.score(y_test, predictions)
print(score)
print(api.show_models())

#Predecir nuevos resultados
y_pred = api.predict(X_test)
y_test = np.array(y_test)

error = 0;

for i in range(len(y_test)):
  print("valor real: ", y_test[i])
  print("valor estimado: ", y_pred[i])
  if(y_test[i] != y_pred[i]):
    error = error + 1;
  print("---------------------------------- ")

error_rate = error  / len(y_test)
print("Error: ",error_rate)

#Curva de precision
```

```
from autoPyTorch.utils.results_visualizer import PlotSettingParams,ColorLabelSettings

metric_name = 'accuracy'


params = PlotSettingParams(
    xlabel='Segundos',
    ylabel='Accuracy'
    legend=True,
    figname= None,
    show=True,


)


api.plot_perf_over_time(
    metric_name=metric_name,
    plot_setting_params=params,
    color_label_settings= ColorLabelSettings(single_train=('red', None), single_opt=('white', None), single_test=('green', None), ensemble_train=('white', None), ensemble_test=('white', None))
)
```

Anexo de código 7

```
# Importacion de librerias
from pprint import pprint
```

```
import sklearn.datasets
import sklearn.metrics
import pandas as pd
import numpy as np
import plotly.express as px
import autokeras as ak
import tensorflow as tf
import os
from sklearn.datasets import load_files

# Cargando datos
TrainDataSet= pd.read_csv('../content/milknew.csv')
TrainDataSet.head()

# Analisis exploratorio de datos
TrainDataSet.describe()
# Cantidad de elementos por categoría del elemento Grade
fig = px.histogram(TrainDataSet,
                   x='Grade',
                   color = 'Grade',
                   text_auto=True,
                   color_discrete_sequence=px.colors.qualitative.G10,
                   template='simple_white',
                   labels={"Grade": "Grade"},
                   title='Grade Value Count')

fig.update_layout(bargap=0.1,
xaxis_title="Grade",yaxis_title="Quantity")
fig.show()
```

```
#Limpieza de datos
input_cols = TrainDataSet.columns[:-1]
input_cols
target_col = TrainDataSet.columns[-1]
target_col
inputs_df = TrainDataSet[list(input_cols)].copy()
targets = TrainDataSet[(target_col)]
df.head(20)

#Modelo de AutoML
#Separo datps de las etiquetas de los resultados
x = inputs_df
y = np.array(targets)
#Separo datos en trainig y test
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size =
0.2)
x_train = np.array(X_train)
y_train = np.array(y_train)
x_test = np.array(X_test)
y_test = np.array(y_test)

# Iniciación del Clasificador de datos estructurados de AutoKeras
clf = ak.StructuredDataClassifier(
    overwrite=True, max_trials=3
)

# Entrenamiento de datos
```

```
history = clf.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=10)


# Predicción del mejor modelo resultante

predicted_y = clf.predict(x_test)

res = clf.evaluate(x_test,y_test)


# Predicción de nuevos resultados estimados

y_pred = clf.predict(x_test)

for i in range(50):

  print("valor real: ", y_test[i])

  print("valor estimado: ", y_pred[i])

  print("---------------------------------- ")


# Curva de aprendizaje

from matplotlib import pyplot

f, ax = pyplot.subplots()

ax.set_ylim(bottom=0.5)

ax.plot(history.history['accuracy'], label=' entrenamiento')

ax.plot(history.history['val_accuracy'], label='prueba')

pyplot.legend()

pyplot.show(f)


# Evaluación de eficiencia

fun, axis = pyplot.subplots()

axis.set_ylim(bottom=0)

axis.plot(history.history['loss'], color='red', label='entrenamiento'
 )
```

```
axis.plot(history.history['val_loss'],  color='green', label='prueba'
)
pyplot.legend()
pyplot.show(fun)
```

Anexo de código 8

```
import numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
import sklearn.metrics
import autosklearn.classification
import autosklearn.metrics
import time

#LOADING DATA
TrainDataSet= pd.read_csv('../content/milknew.csv')
TrainDataSet.head()



#DATA CLEANING
input_cols = TrainDataSet.columns[:-1]
```

```
input_cols

target_col =  TrainDataSet.columns[-1]

target_col


inputs_df = TrainDataSet[list(input_cols)].copy()

inputs_df

targets = TrainDataSet[(target_col)]

targets


#AUTOML MODEL

X = inputs_df

y = targets

X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X,y,
test_size = 0.2, random_state=1)


automl = autosklearn.classification.AutoSklearnClassifier(
    time_left_for_this_task=100,
    per_run_time_limit=10
)


start = time.time()

history = automl.fit(X_train, y_train, dataset_name="Mobile-Price-Classification")

stop = time.time()
```

```
print(f"Training time: {stop - start}s")


#Evaluar red
predictions = api.predict(X_test)
print("Accuracy score", sklearn.metrics.accuracy_score(y_test, predictions))


automl.show_models()
automl.leaderboard()


#Predecir nuevos resultados
y_pred = automl.predict(X_test)
y_test = np.array(y_test)


error = 0;


for i in range(len(y_test)):
  print("valor real: ", y_test[i])
  print("valor estimado: ", y_pred[i])
  if(y_test[i] != y_pred[i]):
    error = error + 1;
  print("---------------------------------- ")


error_rate = error  / len(y_test)
```

```
print("Error: ",error_rate)


#Curva de precision
poT = automl.performance_over_time_
poT.plot(
    x="Timestamp",
    kind="line",
    legend=True,
    title="Auto-sklearn accuracy over time",
    grid=True,
)
plt.show()
```

Anexo de código 9

```
import numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
import sklearn.metrics
import time
from autoPyTorch.api.tabular_classification import TabularClassificationTask
```

```
#LOADING DATA

TrainDataSet= pd.read_csv('../content/milknew.csv')

TrainDataSet.head()


#Transformacion de columna Grade a numerica

TrainDataSet.loc[TrainDataSet['Grade'] == 'high', 'Grade'] = 1

TrainDataSet.loc[TrainDataSet['Grade'] == 'medium', 'Grade'] = 2

TrainDataSet.loc[TrainDataSet['Grade'] == 'low', 'Grade'] = 3


#DATA CLEANING

input_cols = TrainDataSet.columns[:-1]

input_cols

target_col =  TrainDataSet.columns[-1]

target_col


inputs_df = TrainDataSet[list(input_cols)].copy()

inputs_df

targets = TrainDataSet[(target_col)]

targets


#AUTOML MODEL

X = inputs_df
```

```
y = targets

X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X,y,
test_size = 0.2, random_state=1)


X_train = X_train.astype(np.float32)

X_test = X_test.astype(np.float32)

y_train = y_train.astype(np.float32)

y_test = y_test.astype(np.float32)



api = TabularClassificationTask()


start = time.time()

history = api.search(

    X_train=X_train,

    y_train=y_train,

    X_test=X_test,

    y_test=y_test,

    optimize_metric='accuracy',

    total_walltime_limit=100,

    func_eval_time_limit_secs=10

)

stop = time.time()
```

```
print(f"Training time: {stop - start}s")


#Evaluar red
predictions = api.predict(X_test)
print("Accuracy score", sklearn.metrics.accuracy_score(y_test, predictions))
print(api.sprint_statistics())
score = api.score(y_test, predictions)
print(score)
print(api.show_models())


#Predecir nuevos resultados
y_pred = api.predict(X_test)
y_test = np.array(y_test)


error = 0;


for i in range(len(y_test)):
  print("valor real: ", y_test[i])
  print("valor estimado: ", y_pred[i])
  if(y_test[i] != y_pred[i]):
    error = error + 1;
  print("---------------------------------- ")
```

```
error_rate = error  / len(y_test)

print("Error: ",error_rate)


#Curva de precision

from autoPyTorch.utils.results_visualizer import
PlotSettingParams,ColorLabelSettings

metric_name = 'accuracy'


params = PlotSettingParams(

    xlabel='Segundos',

    ylabel='Accuracy',

    legend=True,

    figname= None,

    show=True,


)

api.plot_perf_over_time(

    metric_name=metric_name,

    plot_setting_params=params,

    color_label_settings= ColorLabelSettings(single_train=('red', None),
single_opt=('white', None), single_test=('green', None), ensemble_train=('white',
None), ensemble_test=('white', None))

)
```