UNIVERSIDAD DE LA HABANA

Una estrategia de Meta-Learning para flujos genéricos de AutoML.

Concurso Nacional de Computación 2021
Modalidad: Carreras afines a la computación
Carrera: Ciencia de la Computación, Quinto curso

Autor(es):
Loraine Monteagudo García loraine.monteagudo@matcom.uh.cu

Tutor(es):
Dr. Suilan Estévez Velarde, sesteves@matcom.uh.cu
Lic. Daniel Alejandro Valdés Pérez daniel.valdes@matcom.uh.cu

Facultad de Matemática y Computación

Resumen

El campo de aprendizaje de máquinas automático (AutoML) se ha destacado como una de las principales alternativas para encontrar buenas soluciones para problemas complejos de aprendizaje automático. A pesar del reciente éxito de AutoML, todavía quedan muchos desafíos. El aprendizaje de AutoML es un proceso costoso en tiempo y puede llegar a ser ineficiente computacionalmente. Meta-Learning es descrito como el proceso de aprender de experiencias pasadas aplicando varios algoritmos de aprendizaje en diferentes tipos de datos y, por lo tanto, reduce el tiempo necesario para aprender nuevas tareas. Una de las ventajas de las técnicas de meta-learning es que pueden servir como un apoyo eficiente para el proceso de AutoML, aprendiendo de tareas previas los mejores algoritmos para resolver un determinado tipo de problema. De esta manera, es posible acelerar el proceso de AutoML, obteniendo mejores resultados en el mismo período de tiempo. El objetivo de esta tesis es diseñar una estrategia de meta-learning para dominios genéricos en el aprendizaje automático.

Palabras clave:

Aprendizaje Automático, Meta-Learning, AutoML

1. Introducción

En los últimos tiempos ha habido una explosión en la investigación y aplicación del aprendizaje automático, en inglés *machine learning* (ML) [1]. Sin embargo, el rendimiento de muchos métodos de aprendizaje automático es sensible a una gran variedad de decisiones [2, 3], lo que constituye una barrera para nuevos usuarios [4]. Por ejemplo, el científico de datos debe seleccionar entre una amplia gama de posibles algoritmos, incluidas las técnicas de clasificación o regresión (como *support vector machines*, redes neuronales, modelos bayesianos, árboles de decisión, etc.) y ajustar numerosos hiperparámetros del algoritmo seleccionado. Además, el rendimiento del modelo también se puede juzgar por varias métricas (por ejemplo, precisión, sensibilidad, medida F1). Incluso los expertos requieren gran cantidad de recursos y tiempo para crear modelos con buen rendimiento a causa del proceso de prueba y error que es repetido en cada aplicación para desarrollar modelos eficientes de aprendizaje automático.

Por estas razones ha emergido una nueva idea para automatizar el proceso de ML, aprendizaje de máquinas automático, denominada *Automated Machine Learning* o AutoML. AutoML abarca el diseño de técnicas para automatizar y facilitar todo el proceso de implementación, experimentación y despliegue de algoritmos de aprendizaje automático. AutoML está concebido para reducir la carga de trabajo de los científicos de datos y permitir a los expertos construir automáticamente aplicaciones de ML sin mucho conocimiento en el campo. Por lo tanto, AutoML hace accesible enfoques de aprendizaje automático a los usuarios no expertos que están interesados en aplicarlos, pero no tienen los recursos para aprender sobre las tecnologías involucradas en detalle [5].

Sin embargo, una de las limitaciones presentes en los primeros sistemas de AutoML consiste en su inhabilidad de reusar conocimiento previo para solucionar nuevas tareas [2]. Para cerrar esta brecha, las herramientas de AutoML comenzaron a aplicar técnicas de meta-learning, las cuales tienen el objetivo de obtener modelos para nuevas tareas usando experiencias previas. Meta-learning, o *aprender a aprender*, es la ciencia de observar sistemáticamente cómo se desempeñan los diferentes enfoques de aprendizaje automático en una amplia gama de tareas de aprendizaje, y luego aprender de esta experiencia, o meta-datos, para aprender nuevas tareas mucho más rápido de lo que sería posible de otra manera. Esto no solo acelera y mejora drásticamente el diseño de algoritmos de aprendizaje automático, sino que también nos permite reemplazar algoritmos diseñados a mano con enfoques novedosos aprendidos de una manera basada en datos. Este tipo de estrategias ayudan a disminuir el costo de aplicar AutoML, al relacionar un nuevo conjunto de datos con los mejores flujos obtenidos en problemas similares previamente resueltos.

En los recientes años se ha desarrollado un substancial interés en el campo de meta-learning y muchos sistemas de AutoML lo han integrado [6, 7, 8, 9, 10, 11]. Sin embargo, estas herramientas de meta-learning no son suficientemente flexibles para ser utilizadas en problemas prácticos que requieren la combinación de algoritmos y tecnologías de diferente naturaleza. Las técnicas actuales de meta-learning se centran principalmente en un subconjunto específico de algoritmos, a menudo adaptados a una biblioteca o conjunto de herramientas. Resolver problemas complejos, por otro lado, requiere la combinación de diferentes herramientas que podrían no estar disponibles en una misma biblioteca. Para la aplicación de meta-learning es necesario la representación de estos problemas mediante caracterizaciones informativas para los datasets y representaciones descriptivas para las soluciones obtenidas mediante diferentes herramientas. De esta forma, es posible que meta-learning sea capaz de resolver una gran cantidad de tareas.

El objetivo general de este trabajo es el diseño de una estrategia de meta-learning para métodos

genéricos de AutoML, a partir de la combinación de técnicas de aprendizaje automático y optimización. La estrategia implementada tendrá el objetivo de acelerar el proceso de búsqueda de AutoML añadiendo conocimiento previo, de tal manera que se obtengan mejores resultados en el mismo período de tiempo.

Dado un dataset, una tarea de evaluación (por ejemplo, clasificación o regresión), el algoritmo de meta-learning propuesto tiene el objetivo de producir una lista de los modelos candidatos, basada en el rendimiento esperado de estos modelos en el dataset dado. Esta lista es producida solamente con meta-conocimiento ganado del análisis de datasets relacionados y el entrenamiento de combinaciones de algoritmos en dichos datasets, sin ejecutar ninguno de los algoritmos candidatos. Teniendo este meta-conocimiento, es posible estimar el rendimiento de esos flujos y sugerirlos. Esta estimación, aunque no es exacta, mejorará el proceso de búsqueda de sistemas de aprendizaje de máquinas automático.

El enfoque de meta-learning propuesto está compuesto por dos fases principales: la fase offline, de aprendizaje y la fase online, de recomendación. El objetivo de la fase offline es obtener los meta-datos necesarios para la solución del problema de meta-learning propuesto: la obtención de un ranking de modelos de aprendizaje para una determinada tarea. En esta fase se obtiene una caracterización de los datasets y el rendimiento y la estructura de un conjunto soluciones en dichos datasets. Por otro lado en la fase online, dada una tarea con los meta-datos ganados del análisis de las tareas similares y el entrenamiento de un conjunto de algoritmos en dichos datasets, el objetivo es producir una lista de las soluciones prometedoras para resolver la tarea inicial. Esta lista será utilizada para sugerir rápidamente algunas inicializaciones para el proceso de búsqueda de algoritmos de AutoGOAL.

2. Estado del Arte

La principal área de investigación de meta-learning estudiada en este trabajo es la selección de algoritmos, la cual ha recibido una considerable cantidad de investigación. En el caso especial de meta-learning, el aspecto de interés es la relación entre las características de los datos y el rendimiento del algoritmo, con el objetivo final de predecir un algoritmo o un conjunto de algoritmos adecuado para un problema específico. Como motivación está el hecho de que es inviable examinar todas las posibles alternativas de algoritmos en un procedimiento de prueba y error. La aplicación de meta-learning en este campo puede, por lo tanto, ser útil tanto para proveer una recomendación para un usuario final como de paso preliminar para recomendar algoritmos a soluciones más costosas computacionalmente, como los algoritmos de optimización usados en herramientas de AutoML.

El desafío en meta-learning para la selección de modelos es aprender de experiencias pasadas de una forma sistemática e impulsada por los datos. Primero, es necesario extraer los meta-datos que describen las tareas de aprendizaje anteriores y los modelos previamente aprendidos. Estos meta-datos comprenden las configuraciones exactas de los algoritmos empleados para entrenar los modelos, incluyendo:

- Las configuraciones de los hiperparámetros, composiciones de los flujos de algoritmos y/o arquitecturas de redes neuronales.
- Las evaluaciones del modelo resultante, tales como la precisión y el tiempo de entrenamiento.
- Propiedades medibles de la tarea en sí, que son extraídas de los datasets, también conocidas como meta-características.

Luego es necesario aprender de estos meta-datos previos, para extraer y transferir conocimiento de la búsqueda de los modelos óptimos para nuevas tareas. El resto de esta sección presenta una visión general de diferentes enfoques de meta-learning para hacer esto efectivamente. Además, se muestran ejemplos de cómo estos enfoques han sido utilizados como paso preliminar en varias herramientas de AutoML.

En esta sección las técnicas de meta-learning son separadas en grupos de acuerdo al tipo de meta-datos que ellas aprovechan [12]. Primero, se discute cómo caracterizar las tareas para expresar más explícitamente la similitud entre ellas y cómo construir meta-modelos para aprender las relaciones entre las características de los datos y el rendimiento de las distintas evaluaciones de los modelos (Sección 2.1), y luego se describe como se aprende solamente de evaluaciones de los modelos (Sección 2.2).

2.1. Aprendiendo de las propiedades de las tareas

La principal característica de esta técnica es el uso de meta-características para medir la similitud de las tareas. Así, por ejemplo, podemos usar la distancia euclidiana entre $m(t_{new})$ y $m(t_j), \forall t_j \in T$ para transferir información de las tareas más similares a la nueva tarea t_{new}.

Con esta técnica de meta-learning se puede entrenar un meta-modelo (o *meta-learner*) L que predice el rendimiento de las configuraciones recomendadas Θ^*_{new} en una nueva tarea t_{new}. El meta-modelo es entrenado con los meta-datos $P \cup M$, donde P y M son usualmente calculados de antemano, o extraídos de repositorios de meta-datos.

Cómo extraer información adecuada para caracterizar tareas específicas es una de las preguntas fundamentales en meta-learning. Investigadores han intentado contestar esta pregunta observando las características de los datasets que afectan el rendimiento de los algoritmos [13]. Estas caracterizaciones son denominadas meta-características y usualmente se encuentran divididos en cinco grupos. Estos grupos son subconjuntos de medidas de caracterización [14] que comparten similitudes entre ellas:

Simple: representan información básica sobre el dataset. Hasta un determinado punto son concebidas para medir la complejidad del problema subyacente. Algunas de las caracterizaciones incluidas en este grupo son: el número de instancias, el número de atributos, la dimensionalidad del dataset, la proporción de valores faltantes, etc. También son llamadas medidas *generales*.

Estadísticas: son características que capturan las propiedades estadísticas de los datos. Estas métricas capturan los indicadores de distribución de datos, tales como la media, la desviación estándar, la correlación y curtosis.

Teóricas de la información: son características del campo de teoría de la información. Estas medidas están basadas en la entropía, la cual captura la cantidad de información en los datos y su complejidad.

Basados en modelos: son características extraídas de un modelo inducido de los datos de entrenamiento. Las características en este grupo están caracterizadas por la extracción de información de un modelo de aprendizaje de predicción, generalmente, un árbol de decisión.

***Landmarking*:** son características que usan el rendimiento de algoritmos de aprendizaje simples y rápidos para caracterizar los datasets. Los algoritmos deben tener diferentes sesgos y capturar información importante con un costo computacional bajo.

Construyendo un meta-modelo o *meta-learner* L podemos aprender relaciones complejas entre

las meta-características de una tarea y la utilidad de una configuración específica. Dadas las meta-características M de una nueva tarea t_{new} este meta-modelo L tiene el objetivo de recomendar la configuración más útil Θ^*_{new} para esta tarea. Existe un gran grupo de trabajos previos construyendo modelos para la selección de algoritmos y recomendación de hiperparámetros. En esta sección se usan ejemplos de varios meta-modelos de acuerdo al tipo de tarea que resuelven: pueden ser usados para rankear un conjunto determinado de configuraciones o para predecir el rendimiento de una nueva tarea.

Los meta-modelos pueden ser usados para generar un ranking de las K mejores configuraciones dado un conjunto de meta-características M y una nueva tarea t_{new}. Se obtiene así un conjunto prometedor de modelos con sus hiperparámetros para esta nueva tarea.

Uno de los enfoques más populares es construir un meta-modelo de *K-Nearest Neighbor* (kNN) para predecir las tareas que son similares, y luego rankear las mejores configuraciones en estas tareas similares [15, 16]. El ranking ideal corresponde al ordenamiento correcto de los modelos candidatos para una tarea determinada.

Muchos sistemas de AutoML han seguido el enfoque de vecinos cercanos (*nearest neighbor*) para predecir las tareas similares, debido a la simplicidad de esta técnica. Auto-sklearn [6] fue la primera herramienta de AutoML en seguir este enfoque. Es implementado sobre `scikit-learn` [17], una biblioteca popular de aprendizaje automático en Python. Este sistema mejora los métodos existentes de AutoML tomando en cuenta automáticamente la experiencia pasada en datasets similares y construyendo *ensembles* de los modelos evaluados durante la optimización. Auto-sklearn introdujo la idea de meta-learning en la inicialización de la selección de modelos y el ajuste de hiperparámetros [12].

SmartML [7] está equipado con una base de conocimiento constantemente actualizada que guarda información sobre las meta-características de todos los datasets procesados con su rendimiento asociado de los diferentes clasificadores y sus parámetros ajustados. Para cada dataset nuevo SmartML automáticamente extrae sus meta-características y busca en su base de conocimiento el algoritmo que mejor rendimiento tenga para empezar su proceso de optimización. Una vez elegido el algoritmo, usa optimización bayesiana basada en SMAC para la optimización de hiperparámetros. Sigue el mismo procedimiento basado en meta-características de Auto-sklearn para determinar datasets similares, utilizando el enfoque de vecinos cercanos, el cual está seguido por un mecanismo ponderado entre dos factores diferentes para elegir los algoritmos de los datasets similares más prometedores: la distancia euclidiana entre las meta-características del dataset y las meta-características de todos los datasets guardados en la base de conocimiento y el rendimiento de los mejores algoritmos en datasets similares.

ATOMIC (*Automated Imbalanced Classification*) [18] es un enfoque de AutoML para desarrollar soluciones de ML para abordar tareas de Aprendizaje de Dominio Desbalanceado o *Imbalanced Domain Learning* (IDL) basado en meta-learning. Proporcionan un ranking de soluciones más probables de asegurar una aproximación óptima a un nuevo dominio, reduciendo drásticamente la complejidad computacional asociada a esta tarea. Esto lo llevan a cabo anticipando la pérdida de un gran conjunto de soluciones predictivas en una nueva tarea de aprendizaje desbalanceado. A diferencia de los ejemplos anteriores para la predicción de los rankings no usan el enfoque de vecinos cercanos, sino que usan como meta-modelo el algoritmo de aprendizaje XGBoost [19] para generar un ranking de los algoritmos más prometedores.

2.2. Aprendiendo de las evaluaciones de modelos

Otro grupo de técnicas de meta-learning están basadas en aprender de evaluaciones de los modelos. En este contexto, el problema se define como, dado un conjunto de configuraciones

de algoritmos de aprendizaje, tareas anteriores, evaluaciones de las tareas anteriores en dichas configuraciones y un conjunto de evaluaciones conocidas en una tarea nueva, recomendar configuraciones teninedo en cuenta los rendimientos de las configuraciones anteriores y el conjunto de evaluacions de la nueva tarea.

Estas técnicas son usadas generalmente para recomendar configuraciones y espacios de búsqueda útiles, así como transferir conocimiento de tareas empíricamente similares. Algunas de estas técnicas son explicadas a continuación.

Supongamos que no tenemos acceso a ninguna evaluación de la nueva tarea. Aun así se puede aprender una función que dé como resultado un conjunto de configuraciones recomendadas independientes de la nueva tarea. Estas configuraciones pueden ser evaluados en la nueva tarea para seleccionar el mejor, o para inicializar otros enfoques de optimización.

Los sistemas de AutoML han incorporado esta técnica mediante el uso de portafolios, que es comúnmente creado discretizando el conjunto de algoritmos e hiperparámetros en un conjunto de configuraciones candidatas, evaluados en un gran número de tareas anteriores.

Auto-Pytorch [10] es un ejemplo de estos sistemas que usan portafolios para crear un conjunto inicial de flujos de algoritmos para la subsecuente optimización. Auto-Pytorch optimiza la arquitectura de red y los hiperparámetros de entrenamiento para permitir aprendizaje profundo completamente automatizado (*Automated Deep Learning*, AutoDL). Auto-Pytorch simplemente empieza la primera iteración con un conjunto de configuraciones complementarias que cubren bien un conjunto de datasets de meta-entrenamiento, después de esto ejecuta un algoritmo de optimización para la búsqueda de los mejores hiperparámetros. Para esto se construye inicialmente un portafolio inicial. Para construir el portafolio offline, se realiza una ejecución de su algoritmo de optimización en cada uno de los datasets de meta-entrenamiento, dando lugar a un conjunto de candidatos. Las configuraciones de las ejecuciones individuales son entonces evaluadas en todos los datasets, resultando en una meta-matriz de rendimiento. Para cada uno de los candidatos, configuraciones son añadidas iterativamente y de forma *greedy*. Las configuraciones son añadidas de esta manera hasta que un tamaño predefinido del portafolio es alcanzado. Limitando el tamaño del portafolio se balancea entre empezar con configuraciones prometedoras y la sobrecarga inducida por el primer portafolio ejecutado. Este enfoque asume (como todos los enfoques de meta-learning) que tenemos acceso a un conjunto razonable de datasets de meta-entrenamiento que son representativos de los datases de meta-prueba.

Auto-sklearn 2.0 [11] es una extensión de Auto-sklearn [6] que presenta una técnica de meta-learning más simple y perfecciona su manera de manejar algoritmos iterativos. Debido a varios problemas encontrados en el uso de meta-características en el algoritmo de meta-learning de Auto-sklearn, proponen un nuevo enfoque libre de meta-características, haciendo uso de un portafolio, un conjunto de configuraciones complementarias que cubren tantos datasets diversos como sea posible y minimiza el riesgo de fallo cuando se enfrenta en una nueva tarea. Para ello hacen una matriz de rendimiento, guardando el rendimiento de un conjunto de algoritmos de ML en varios datasets. Luego para seleccionar algoritmos del portafolio basado en un nuevo dataset utilizan un algoritmo *greedy*. Mientras el enfoque anterior de meta-learning seguido por Auto-sklearn (*k-nearest neighbors*) tiene el objetivo de usar solo configuraciones que actúan bien, un portafolio es construido de tal manera que al menos hay una configuración que funciona bien, la cual además proporciona una forma diferente de diseño inicial para su algoritmo de optimización.

Las evaluaciones previas también pueden ser usadas para aprender mejores *espacios de configuración*. Incluso siendo independientes de la nueva tarea, esto puede radicalmente acelerar la búsqueda para modelos óptimos, ya que solo las regiones más relevantes de los espacios

de configuración son explorados. Esto es crítico cuando los recursos computacionales están limitados.

Uno de los sistemas de AutoML que ha usado técnicas de meta-learning para aprender mejores espacios de configuración es ATM (*Auto-Tuned Models*) [20]. ATM es un sistema de aprendizaje automático multi-método, multi-parámetro y auto optimizado para la automatización de selección de modelos y el ajuste de hiperparámetros. Su principal contribución fue la presentación de un nuevo método para organizar el espacio de búsqueda jerárquica de los métodos de aprendizaje automático. Emplean técnicas automáticas de meta-learning que iterativamente seleccionan entre estos espacios jerárquicos y ajustan los hiperparámetros.

Existen varias maneras de dar recomendaciones de una tarea específica, pero generalmente se necesita información sobre qué tan similar es la nueva tarea t_{new} es a las tareas anteriores $t_j \in T$. Una manera de hacer esto es evaluando un número de configuraciones recomendadas en la nueva tarea. Si luego se observa que las evaluaciones de esta tarea son similares a las observaciones en las tareas anteriores para un dataset específico, entonces basándose en evidencia empírica t_j y t_{new} pueden ser consideradas similares. Este conocimiento puede ser incluido para entrenar un meta-modelo que predice un conjunto de configuraciones recomendadas para t_{new}. Además, cada configuración seleccionada puede ser evaluada y luego incluida en la base de meta-conocimiento, repitiendo el ciclo y coleccionando más evidencia empírica para aprender cuáles tareas son similares entre sí.

El uso del rendimiento de varios algoritmos en determinados datasets ha servido para medir la similitud de las tareas en varios sistemas de AutoML. Uno de los ejemplos de esto es OBOE [9], que forma una matriz de errores con validación cruzada de un gran número de algoritmos de aprendizaje supervisado (algoritmos juntos con sus hiperparámetros) en un gran número de datasets. Cada fila en la matriz representa un dataset, cada columna representa un algoritmo de ML y cada celda representa el rendimiento de un modelo particular de aprendizaje automático con sus hiperparámetros en un dataset específico. Para encontrar los mejores modelos para un dataset nuevo, OBOE ejecuta un conjunto de algoritmos rápidos, pero informativos en el nuevo dataset y usa sus errores de validación cruzada para inferir el vector de características para el nuevo dataset. OBOE ejecuta un conjunto particular de modelos correspondientes a un subconjunto de columnas en la matriz de error, los cuales son estimados para ejecutar eficientemente en el nuevo dataset. El sistema tiene en cuenta 2 problemas importantes: (1) **Inicializaciones con restricciones de tiempo**: cómo elegir un modelo prometedor inicial bajo restricciones de tiempo y (2) **Active learning**: cómo mejorar la predicción inicial dando más recursos computacionales. La predicción del tiempo de ejecución de un algoritmo depende solo del número de ejemplos y las características en el dataset. El subproblema de *active learning* tiene como objetivo ganar la mayor cantidad de información para guiar el proceso de selección de modelos.

PMF [21] soluciona la tarea de la selección automática de una secuencia de algoritmos de ML de aprendizaje automático usando ideas de filtrado colaborativo y optimización bayesiana. PMF considera que dos dataset son similares si tienen evaluaciones similares en un pequeño conjunto de flujos de algoritmos y, por lo tanto, es más probable que estos datasets tengan similares evaluaciones en el resto de los flujos. En particular, PMF entrena cada flujo de aprendizaje automático en una muestra de cada dataset y entonces evalúa dicho flujo. Esto resulta en una matriz que resume el rendimiento de cada flujo de aprendizaje automático de cada dataset. El problema de predecir el rendimiento de un flujo particular en un nuevo dataset es relacionado con un problema de factorización de matrices.

3. Propuesta

En esta sección se presenta un método para la selección de flujos de algoritmos usando técnicas de meta-learning. Un flujo puede ser visto como un caso especial de algoritmo que aplica cada algoritmo de forma secuencial a la salida del anterior en la sucesión. Por lo tanto, como un flujo está compuesto de varios algoritmos, la búsqueda de estos y de sus hiperparámetros resulta más compleja.

El enfoque desarrollado se propone como un paso preliminar para otras soluciones más costosas computacionalmente, como por ejemplo, para la inicialización de sistemas de AutoML. En esta investigación AutoGOAL también es utilizado como herramienta complementaria en el proceso de búsqueda de flujos. Por lo tanto, se describe como se realiza la incorporación de conocimiento experto a la estrategia de búsqueda usada por AutoGOAL: Evolución Probabilística Gramatical (*Probabilistic Grammatical Evolution*) [22], que no había sido utilizada anteriormente con meta-learning.

El enfoque de meta-learning propuesto está compuesto por dos fases:

- La fase offline, que es de aprendizaje, es donde ocurre la adquisición de meta-conocimiento. El objetivo de esta parte es obtener los meta-datos necesarios para la solución del problema de meta-learning propuesto: la obtención de un ranking de flujos de algoritmos de aprendizaje para una determinada tarea. En esta fase se obtiene una caracterización de los datasets y el rendimiento de un conjunto flujos de algoritmos en dichos datasets.
- La fase online, que es de recomendación, es donde se aplica el meta-conocimiento adquirido en la fase anterior. En esta fase, dada una tarea con los meta-datos ganados del análisis de las tareas similares y el entrenamiento de flujos de algoritmos en dichos datasets, el objetivo es producir una lista de los flujos prometedores para resolver la tarea inicial.

La Figura 1 muestra el flujo de trabajo de la estrategia de meta-learning implementada. En la fase offline (Figura 1 derecha) se obtiene una caracterización de los datasets y la estructura y rendimiento de un conjunto flujos de algoritmos en dichos datasets, esta información es utilizada para entrenar el meta-modelo. En la fase online (Figura 1 izquierda), dado un nuevo dataset, se extraen sus meta-características y de acuerdo a estas y a una métrica de distancia se seleccionan los datasets similares. De estos datasets se obtienen los flujos de algoritmos guardados en la base de conocimiento, y con estos y el meta-modelo se genera un ranking final.

3.1. Adquisición de Meta-Conocimiento

La adquisición de meta-conocimiento se realiza mediante la extracción de caracterizaciones de un conjunto de datasets, es decir, meta-características, y de información referente a un conjunto de algoritmos que deben ser probados en estos datasets. Entre los datos de los algoritmos extraídos se guarda información respecto a los hiperparámetros utilizados y al rendimiento alcanzado en cada una de las tareas para cada uno de los conjuntos de algoritmos usados.

Las meta-características utilizadas para describir los datasets fuerons:

- **Simples**: Supervisado, Tamaño de la muestra, Número de clases, Dimensionalidad de la entrada, Dimensionalidad de la salida, Dimensionalidad del dataset, Cantidad de características categóricas, Características categóricas, Cantidad de características numéricas, Características numéricas, Cantidad de valores faltantes.
- **Estadísticos**: Desviación estándar, Coeficiente de variación, Covarianza media, Coeficiente de correlación lineal, *Skewness* (Oblicuidad), Curtosis, PCA (*Principal Component*

Analysis, Análisis de los Componentes Principales).

- **Teóricos de la Información**: Entropía normalizada de una clase, Entropía normalizada de un atributo, Entropía conjunta, Información mutua de clase y atributo, Número equivalente de atributos, Relación de la señal de ruido.
- **Específicos de AutoGOAL**: Tipo semántico de la entrada y de la salida.

Además de las meta-características, se extrajo información relacionada con las soluciones de las tareas, que fueron utilizadas como características para el meta-modelo. En la fase de adquisición de conocimiento, las soluciones de las tareas deben ser generadas. El enfoque de meta-learning implementado no intenta simplemente determinar un buen algoritmo con sus hiperparámetros como solución, sino un flujo de algoritmos, lo que complica la generación de estos.

Para la generación de los flujos de algoritmos se utilizó AutoGOAL. En cada uno de los datasets se ejecutó AutoGOAL y se guardaron las arquitecturas generadas junto con su rendimiento. El formato en el que se guardaron fue dependiente de la estructura en la que AutoGOAL procesa los flujos. Los algoritmos se ponen primero secuencialmente, luego existe una palabra clave `End` para representar el final de estos. Después, se ponen los hiperparámetros de los algoritmos de la forma '`{algoritmo}_{hiperparámetro}`' para identificarlos.

3.2. Aplicación de Meta-Conocimiento

En un sistema de meta-learning la aplicación de meta-conocimiento puede ser utilizado para ayudar a seleccionar un conjunto de algoritmos de aprendizaje de máquinas que obtengan un buen rendimiento en una tarea determinada. Una vez obtenidos los meta-datos necesarios, el objetivo de esta fase es la obtención de una lista de flujos de algoritmos prometedores para un dataset determinado. Esto se realiza analizando los datasets similares a un nuevo dataset y recomendando los flujos que tuvieron un buen rendimiento en estos conjuntos de datos seleccionados.

La selección de algoritmos fue realizada mediante un enfoque de ranking, en el que para un nuevo dataset se seleccionan los `k` mejores flujos de algoritmos. Para esto se implementaron varias estrategias, que son descritas a continuación.

3.2.1. Estrategia de Vecinos Cercanos

La primera estrategia, de *Nearest Neighbors* o Vecinos Más Cercanos, consiste en extraer los flujos candidatos para un dataset determinado en dependencia de las características de los `n` datasets más similares a él. Se extraen los `m` flujos de algoritmos que hayan tenido mejor rendimiento en cada uno de los `n` datasets, y estos flujos son combinados para formar un nuevo ranking para el nuevo dataset. Este enfoque puede ser dividido en dos pasos: la búsqueda de los vecinos cercanos y la generación de un ranking.

Una de las desventajas de este enfoque es la necesidad de especificar y determinar los mejores valores para la cantidad de datasets similares seleccionados `n` y la cantidad de flujos de algoritmos seleccionados `m`. Por lo cual, se desarrolló otro enfoque.

El otro método para la generación de rankings consiste en un mecanismo ponderado entre dos factores: la distancia entre el nuevo dataset y los datasets del conjunto de entrenamiento, y el valor del resultado de rendimiento de los flujos de algoritmos en los dataset similares. La estrategia seguida para la combinación de estos fue una de las más directas: la división entre el resultado de rendimiento de un flujo de algoritmos en un dataset similar y el valor de distancia

entre los dataset. De esta forma, los datasets que tengan menor distancia entre sí y los flujos que tengan un mejor rendimiento obtendrán un mayor resultado. Los resultados de esta división son los utilizados para generar el ranking de los flujos candidatos para la nueva tarea.

3.2.2. Estrategia utilizando un Meta-Modelo

La segunda estrategia utiliza un meta-modelo para predecir el rendimiento de un flujo de algoritmos en un determinado dataset sin necesidad de ejecutar dicho flujo. Al igual que en la estrategia anterior dado un nuevo dataset se extraen los flujos más similares a él, y se obtiene un conjunto de flujos candidatos. Estos flujos son presentados al meta-modelo junto con el nuevo dataset para obtener un ranking de los algoritmos candidatos. Esta estrategia también funciona en dos pasos.

En el primer paso se computan las meta-características del nuevo dataset, y se sigue un método parecido a la estrategia anterior para obtener los dataset similares.

Los flujos de algoritmos obtenidos necesitan ser pre-procesados para ser presentados al meta-modelo. Con el objetivo de representar los flujos de una forma compacta, se eligió representar la topología de un flujo como una secuencia de números. Cada algoritmo del flujo está representado como un número único que lo identifica. Por lo tanto, se genera una secuencia de números para representar cada flujo, donde el orden de los números determina el orden de aplicación de los algoritmos que componen el flujo.

En la segunda fase de la estrategia, los flujos de algoritmos candidatos pre-procesados son concatenados con el vector de características del nuevo dataset, formando los meta-datos, y son presentados al meta-modelo. Como resultado, el meta-modelo retorna un ranking basado en el rendimiento esperado de cada uno de los flujos en el dataset. Se retorna una lista con los `k` mejores flujos y, como se mencionó anteriormente, estos flujos son pos-procesados para obtener los hiperparámetros con los que fueron utilizados en su dataset original.

Como meta-modelo, se utilizó el modelo XGBRanker de la biblioteca XGBoost [19] de Python. Este meta-modelo fue elegido porque el problema seleccionado es en esencia un problema de ranking, y trabajo previo ha demostrado que XGBoost es altamente recomendado para producir listas rankeadas [23]. Además, XGBoost ha sido usado con anterioridad en problemas de meta-learning para la selección de algoritmos [23, 18].

3.3. Implementación en AutoGOAL

La estrategia llevada a cabo hasta ahora puede funcionar como un sistema de recomendación de flujos de algoritmos independiente, o puede servir como un paso preliminar para otras soluciones más complejas computacionalmente. Sin embargo, a pesar de que mediante meta-learning puede sugerir rápidamente algunas inicializaciones de los algoritmos de ML que probablemente tengan buenos resultados, no es posible obtener información detallada sobre qué tan buen rendimiento tendrá en un dataset nuevo. En contraste, los algoritmos de optimización utilizados en herramientas de AutoML son lentos al buscar en grandes espacios de hiperparámetros, pero son capaces de obtener información más detallada sobre el rendimiento de los algoritmos de ML en un nuevo dataset [6]. Es por esto que el enfoque de meta-learning propuesto es complementario al proceso de optimización, usando el ranking de configuraciones elegidas para inicializar un proceso de optimización.

Como herramienta de AutoML complementaria a esta solución se eligió el sistema de AutoGOAL [24], que ha sido diseñado por el grupo de investigación de Inteligencia Artificial de la

Facultad de Matemáticas y Computación. La estrategia seguida para la inicialización del proceso de optimización de AutoGOAL es la modificación del modelo probabilístico que usa AutoGOAL para generar y evaluar los flujos de algoritmos válidos. Después de obtener la lista de flujos de algoritmos recomendados en la fase de meta-learning, se extraen los modelos probabilísticos asociados a estos flujos. Estos modelos son guardados en la fase inicial de meta-learning, de obtención de meta-conocimiento, cuando se generan los flujos con AutoGOAL. Luego, se crea un modelo inicial σ^*, que es el resultado de la mezcla de los modelos probabilísticos de los flujos recomendados. En las primeras iteraciones se usa el modelo σ con valores neutrales para cada distribución, generando flujos aleatorios, para lograr una mayor exploración y luego este modelo es mezclado con el modelo σ^*, que contiene las inicializaciones que son resultado de la fase de meta-learning.

4. Resultados Experimentales

Para los experimentos realizados se extrajeron datasets de clasificación de OpenML [25]. OpenML tiene alrededor de 19000 datasets disponibles para descargar y ofrece una API Web[1] a través de la cual pueden ser enviados nuevos recursos y resultados. OpenML-Python [26] es una integración al ecosistema popular de Python ML[2], que elimina la complejidad del acceso a la API Web proporcionando un fácil acceso en Python a todos los datos de OpenML y automatizando el intercambio de nuevos experimentos. Los datasets seleccionados fueron extraídos utilizando esta API.

Para obtener un conjunto representativo de datasets se consideraron todos los que tenían más de 300 y menos de 500 000 instancias con más de 2 atributos y menos de 300 atributos, terminando en un total de 305 datasets.

Para la evaluación de la propuesta realizada se separaron los 305 datasets en dos conjuntos: D_{train} y D_{test}, donde el primero representa el $75\,\%$ del total de datasets y el segundo el $25\,\%$. D_{train} fue utilizado en el entrenamiento de las estrategias seleccionadas y D_{test} en la prueba de las mismas. En los resultados mostrados en el resto de este capítulo, se usa todo el conjunto de datasets D_{train} para el entrenamiento de los modelos obtenidos con las distintas estrategias implementadas y se exponen los resultados obtenidos en D_{test}.

Todos los flujos de algoritmos usados durante el entrenamiento y evaluación de la propuesta implementada fueron generadas usando AutoGOAL. Los flujos generados con AutoGOAL consisten en algoritmos presentes en varias bibliotecas de Python, entre las que se encuentran: Sklearn [17], Pytorch [27], Keras [28], NLTK [29] y Gensim [30].

AutoGOAL se ejecutó en cada uno de los 305 datasets seleccionados y se almacenaron todas las arquitecturas generadas junto con los resultados obtenidos. Para cada uno de los datasets, AutoGOAL se configuró para que realizara la búsqueda de flujos durante 1 hora, teniendo un total de 5 minutos para la evaluación de un flujo de algoritmos. De esta forma, se excluyeron flujos muy complejos y se garantizó la evaluación de al menos 12 flujos por dataset. Esta generación de flujos duró un total de 305 horas y se generaron en promedio 640.28 flujos de algoritmos por dataset, y en total se obtuvieron 248 430 flujos.

4.1. Comparación de las diferentes estrategias

Los datasets y los flujos de algoritmos extraídos fueron utilizados con las diferentes estrategias de meta-learning implementadas:

[1] `http://www.openml.org`
[2] `https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/`

Estrategia Vecinos Cercanos Simple: El método de vecinos cercanos fue probado utilizando la distancia L2 estándar. El ranking final generado para un nuevo dataset es de 15 flujos. En esta versión se utilizó la estrategia simple, en la que se seleccionan `n` datasets más cercanos y de ellos los `m` flujos de algoritmos que hayan obtenido un mejor rendimiento, luego el ranking es formado seleccionando a los 15 mejores flujos de algoritmos. Para la evaluación de esta estrategia se utilizó `m = 15` y `n = 15`.

Estrategia Vecinos Cercanos Ponderado: El método de vecinos cercanos se vuelve a evaluar, pero utiliza la otra estrategia explicada en la Sección 3.2.1. Al igual que en la estrategia anterior, se utiliza la distancia L2 estándar y se genera un ranking final de 15 flujos para una tarea nueva.

Estrategia usando XGBRanker: En esta estrategia, explicada en la sección 3.2.2, se utilizó como meta-modelo XGBRanker de la biblioteca XGBoost para generar los rankings de una nueva tarea. Se usaron las siguientes configuraciones de hiperparámetros: `objective = 'rank:pairwise'`, `n_estimators = 150`, `tree_method = 'hist'`, `max_depth = 10`, `learning_rate = 0.1`, `subsample = 0.95`.

4.2. Resultados

Para la evaluación de las estrategias de meta-learning implementadas es importante conocer los resultados obtenidos al incorporarse a un sistema de AutoML. Cómo el ranking de configuraciones elegidas son usadas para inicializar el proceso de optimización, y qué resultados este proceso puede obtener al usar el conocimiento previo adquirido mediante meta-learning. En esta sección se discuten las experimentaciones realizadas para evaluar estos aspectos.

Para la realización de los experimentos se ejecutó la búsqueda de algoritmos de AutoGOAL con y sin meta-learning, probando los métodos de meta-learning mencionados anteriormente: vecinos cercanos con la estrategia simple, vecinos cercanos utilizando mecanismos ponderados y utilizando un modelo de XGBoost, XGBRanker. Para estudiar su rendimiento bajo una estricta restricción de tiempo, y además, debido a limitaciones de los recursos computacionales utilizados, se limitó la búsqueda para cada ejecución a 30 minutos. Igualmente, el tiempo de ejecución de un solo modelo se limitó a la sexta parte de este tiempo (5 minutos). Las evaluaciones realizadas en esta sección muestran los resultados obtenidos en los datasets de prueba, utilizando los datasets de entrenamiento para entrenar los métodos implementados.

Uno de los aspectos interesantes para evaluar es el comportamiento de las soluciones obtenidas a través de las distintas iteraciones de AutoGOAL. La Figura 2 muestra la media y el intervalo de confianza del 95 % de los resultados de rendimiento obtenidos en las primeras 200 iteraciones en las diferentes estrategias evaluadas. Se puede observar como, a partir de determinado punto (cuando se terminan las iteraciones iniciales que generan resultados aleatorios) las soluciones que utilizan meta-learning alcanzaron mejoras drásticas a partir de la primera configuración que se seleccionó. La mejora fue más pronunciada al principio y, con el tiempo, AutoGOAL sin meta-learning también encontró buenas soluciones, lo que le permitió alcanzar los mismos resultados obtenidos en las versiones que usan meta-learning en algunos datasets (mejorando así su resultado final).

En la Figura 3 se muestra el mejor resultado final obtenido en la búsqueda de los flujos de algoritmos en cada uno de los datasets de prueba para las diferentes estrategias. Como se puede apreciar, se obtienen mejores resultados finales en cada uno de las estrategias seguidas con respecto a AutoGOAL sin meta-learning. La diferencia entre la estrategia de vecinos cercanos simple no es tan pronunciada con respecto a la versión que no usa meta-learning, y esto se cree que se debe a que, a pesar de que mediante esta estrategia se obtienen los valores óptimos

más rápidos, AutoGOAL es capaz de alcanzar a esta versión, y encontrar buenas soluciones. El método de vecinos cercanos ponderado y el que usa XGBRanker, un algoritmo de ranking de XGBoost, sí obtiene resultados un poco más significativos, siendo este último el que mejor resultado final obtiene. Por lo tanto, se puede concluir que la agregación de conocimiento de experiencias pasadas mediante el método de meta-learning implementado mejoran la búsqueda final de flujos de algoritmos, obteniendo flujos que tienen mejores resultados en la mayoría de los datasets en el mismo intervalo de tiempo.

La Figura 4 muestran los resultados obtenidos estandarizados. Los valores atípicos fueron omitidos para un mejor análisis de la gráfica. Para su normalización, fue usada la Puntuación Z o *Z-Score*, que se utiliza en estadística para comparar datos procedentes de diferentes muestras o poblaciones. Los resultados muestran una pequeña mejora en la estrategia de Vecinos Cercanos Simple y XGBRanker con respecto a AutoGOAL, mientras que no se puede apreciar mucha diferencia con respecto a los resultados obtenidos utilizando el método de Vecinos Cercanos Ponderados. Sin embargo, es necesario tener en cuenta que esta métrica no fue la usada en el entrenamiento de los modelos de meta-learning. Las estrategias que usan directamente el resultado de rendimiento, y no este valor normalizado, como la de Vecinos Cercanos Ponderado y XGBRanker, pueden estar afectadas en la comparación con esta métrica. A pesar de esto, se puede concluir que los resultados alcanzados por AutoGOAL usando las estrategias de meta-learning superan aquellos obtenidos por AutoGOAL sin meta-learning.

5. Conclusiones

El método propuesto tiene también algunas limitaciones, las cuales se pueden mejorar en trabajos futuros. Con respecto a la métrica usada para determinar el rendimiento de cada flujo, se usa la métrica por defecto de AutoGOAL, *accuracy*. Sin embargo, en escenarios prácticos, puede ser necesario equilibrar diferentes métricas de rendimiento, incluido también el uso del tiempo y la memoria, y cualidades más subjetivas como la interpretabilidad de los modelos o su capacidad para lidiar con datos sesgados.

La experimentación llevada a cabo se realizó solo con datasets de clasificación, por lo que solo se estudiaron los resultados obtenidos en esa tarea. Para demostrar la utilidad de la estrategia de meta-learning implementada en una gran gama de tareas es necesario la experimentación en diferentes tipos de tareas de aprendizaje, por ejemplo, de regresión. Sin embargo, el modelo propuesto es adaptable a cualquier tipo de problema de aprendizaje automático siempre y cuando se tenga una medida para determinar la eficiencia de un flujo de algoritmos en un dataset.

Meta-learning es muy dependiente de la calidad de los meta-ejemplos que usa para su entrenamiento [31]. Usualmente, es difícil obtener buenos resultados, ya que las meta-características son frecuentemente bastante ruidosas, porque el cálculo de estas está propenso a errores. Además, el número de problemas disponibles para la generación de meta-ejemplos generalmente es limitada. En este trabajo, se obtienen buenos resultados debido al uso de meta-características que han sido ampliamente utilizadas en la literatura y a la gran variedad de datasets extraídos. Sin embargo, se considera que un análisis más detallado de la distribución de los datasets para excluir aquellos que tienen características atípicas, y la inclusión de datasets de mayor tamaño puede proporcionar mejores resultados. De igual manera, un estudio de las meta-características para la eliminación de aquellas que proporcionen información poco útil debe proporcionar mejoras en el método propuesto. Además, el uso de la medida normalizada (*Z-Score*) en el entrenamiento de las estrategias implementadas, puede proporcionar mejoras para la selección justa de flujos de algoritmos en distintos datasets.

Referencias

[1] T. Hey, K. Butler, S. Jackson, and J. Thiyagalingam, "Machine learning and big scientific data," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, p. 20190054, 03 2020.

[2] S. Dyrmishi, R. El Shawi, and S. Sakr, "A decision support framework for automl systems: A meta-learning approach," pp. 97–106, 11 2019.

[3] R. E. Shawi, M. Maher, and S. Sakr, "Automated machine learning: State-of-the-art and open challenges," *CoRR*, vol. abs/1906.02287, 2019.

[4] A. Crisan and B. Fiore-Gartland, *Fits and Starts: Enterprise Use of AutoML and the Role of Humans in the Loop*. New York, NY, USA: Association for Computing Machinery, 2021.

[5] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*. Springer Publishing Company, Incorporated, 1st ed., 2019.

[6] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[7] M. Maher and S. Sakr, "Smartml: A meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms," in *EDBT: 22nd International Conference on Extending Database Technology*, 2019.

[8] I. Drori, Y. Krishnamurthy, R. Rampin, R. Lourenço, J. One, K. Cho, C. Silva, and J. Freire, "Alphad3m: Machine learning pipeline synthesis," in *AutoML Workshop at ICML*, 2018.

[9] C. Yang, Y. Akimoto, D. W. Kim, and M. Udell, "OBOE: collaborative filtering for automl initialization," *CoRR*, vol. abs/1808.03233, 2018.

[10] L. Zimmer, M. Lindauer, and F. Hutter, "Auto-pytorch: Multi-fidelity metalearning for robust autodl," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[11] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: The next generation," *ArXiv*, vol. abs/2007.04074, 2020.

[12] J. Vanschoren, "Meta-learning: A survey," 2018.

[13] A. Rivolli, L. P. F. Garcia, C. Soares, J. Vanschoren, and A. Carvalho, "Towards reproducible empirical research in meta-learning," *ArXiv*, vol. abs/1808.10406, 2018.

[14] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning - Applications to Data Mining*. 01 2009.

[15] P. Santos, T. Ludermir, and R. Prudêncio, "Selection of time series forecasting models based on performance information," pp. 366–371, 01 2004.

[16] P. Brazdil, C. Soares, and J. Costa, "Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results," *Machine Learning*, vol. 50, pp. 251–277, 03 2003.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] N. Moniz and V. Cerqueira, "Automated imbalanced classification via meta-learning," *Ex-*

pert Systems with Applications, vol. 178, p. 115011, 04 2021.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016.

[20] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, "Towards automatically-tuned neural networks," in *Workshop on Automatic Machine Learning*, pp. 58–65, PMLR, 2016.

[21] N. Fusi, R. Sheth, and M. Elibol, "Probabilistic matrix factorization for automated machine learning," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[22] H.-T. Kim and C. W. Ahn, "A new grammatical evolution based on probabilistic context-free grammar," in *Proceedings in Adaptation, Learning and Optimization*, pp. 1–12, Springer International Publishing, 2015.

[23] D. Laadan, R. Vainshtein, Y. Curiel, G. Katz, and L. Rokach, "Rankml: a meta learning-based approach for pre-ranking machine learning pipelines," *CoRR*, vol. abs/1911.00108, 2019.

[24] S. Estevez-Velarde, Y. Gutiérrez, A. Montoyo, and Y. Almeida Cruz, "Automatic discovery of heterogeneous machine learning pipelines: An application to natural language processing," in *Proceedings of the 28th International Conference on Computational Linguistics*, (Barcelona, Spain (Online)), pp. 3558–3568, International Committee on Computational Linguistics, Dec. 2020.

[25] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: Networked science in machine learning," *SIGKDD Explor. Newsl.*, vol. 15, p. 49–60, June 2014.

[26] M. Feurer, J. N. van Rijn, A. Kadra, P. Gijsbers, N. Mallik, S. Ravi, A. Müller, J. Vanschoren, and F. Hutter, "Openml-python: an extensible python API for openml," *CoRR*, vol. abs/1911.02490, 2019.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019.

[28] F. Chollet *et al.*, "Keras," 2015.

[29] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[30] K. Khosrovian, D. Pfahl, and V. Garousi, "Gensim 2.0: A customizable process simulation model for software process evaluation," vol. 5007, pp. 294–306, 05 2008.

[31] T. Gomes, P. Miranda, R. Prudêncio, C. Soares, and A. Carvalho, "Combining meta-learning and optimization algorithms for parameter selection," in *5 th PLANNING TO LEARN WORKSHOP WS28 AT ECAI 2012*, p. 6, Citeseer, 2012.

A. Anexos

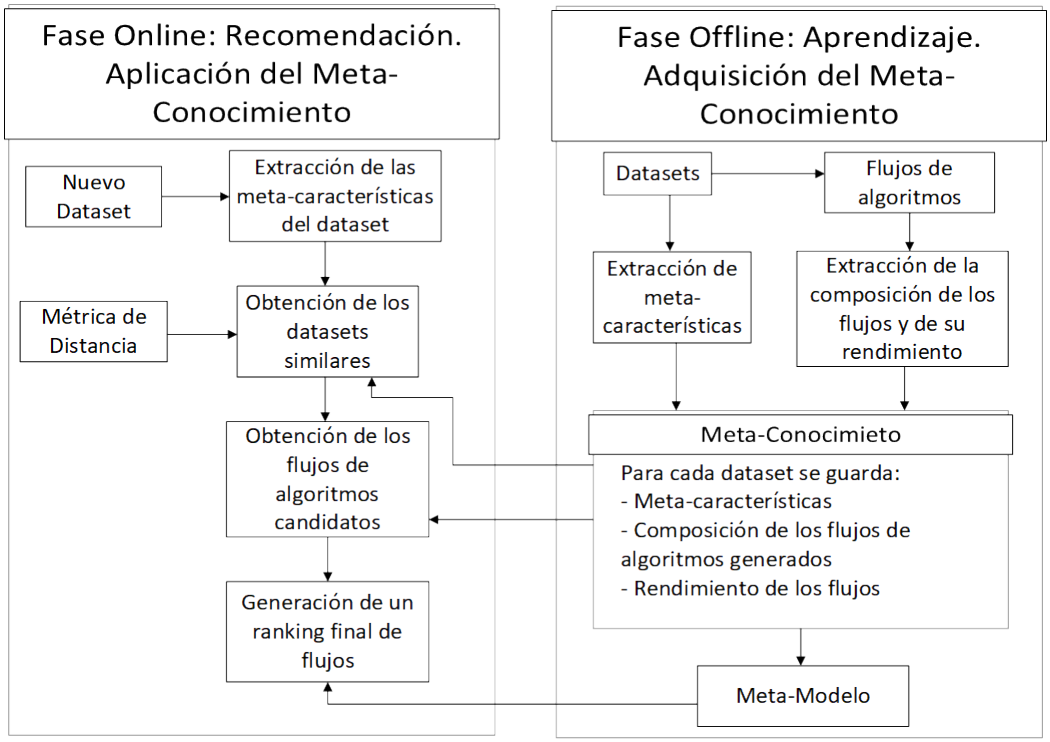

Figura 1: Flujo de trabajo del enfoque de meta-learning propuesto.

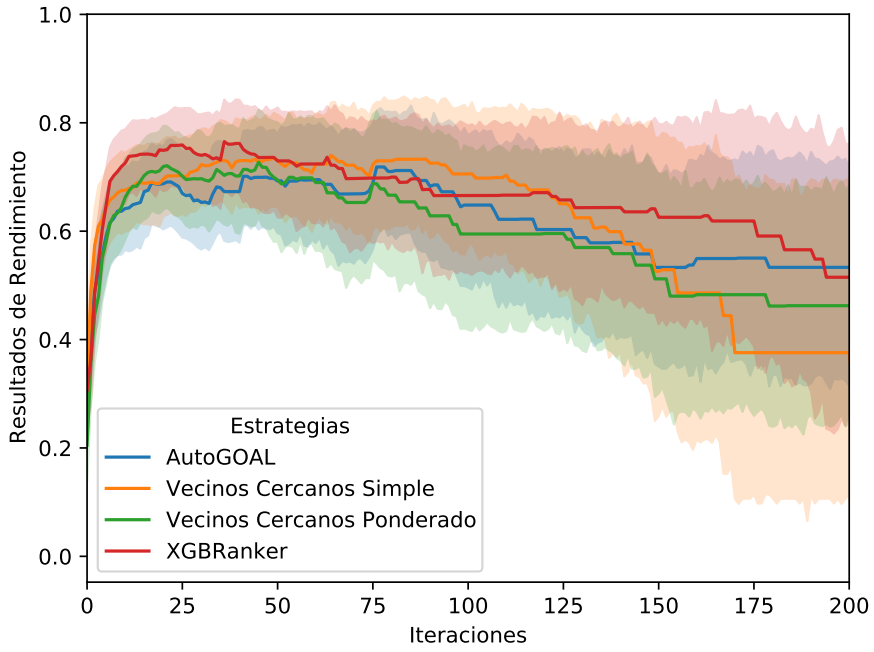

Figura 2: Resultados de rendimiento obtenidos en los datasets de prueba en las primeras 200 iteraciones, se muestra la media y el intervalo de confianza del 95 %.

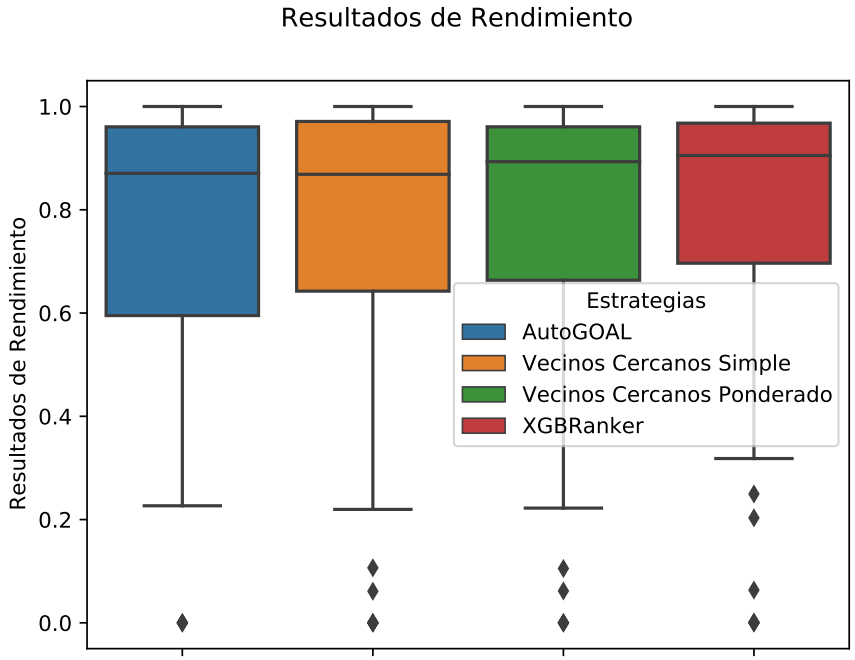

Figura 3: Mejor resultado de rendimiento obtenido en la búsqueda de los flujos de algoritmos en cada uno de los datasets de prueba para las diferentes estrategias.

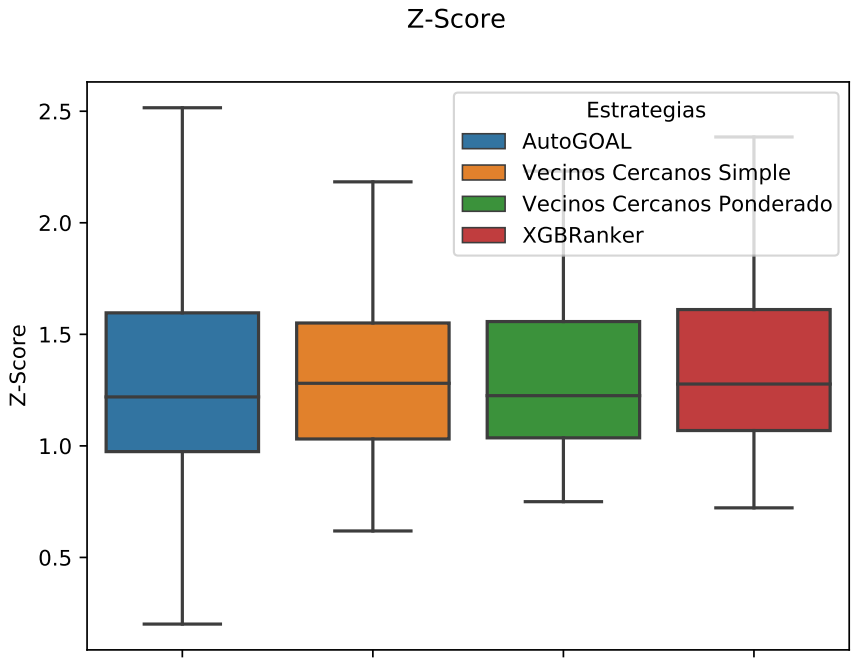

Figura 4: Puntuación Z obtenida en la búsqueda de los flujos de algoritmos en cada uno de los datasets de prueba para las diferentes estrategias.