Evaluación Comparativa de Herramientas AutoML de Código Abierto en Tareas de Regresión

Adrián Bender[1] y Santiago Nicolet[2]

Facultad de Ingeniería, Universidad del Salvador, Buenos Aires, Argentina
[1]bender.adrian@usal.edu.ar
[2]santiago.nicolet @usal.edu.ar

Resumen. AutoML actúa como puente entre los diferentes niveles de experiencia al momento de generar modelos predictivos y agiliza el proceso de Aprendizaje Automático. Se implementa a través de diversas técnicas, de las cuales existen pocas comparaciones objetivas, la gran mayoría en tareas de clasificación. Presentamos una referencia de las actuales herramientas AutoML de código abierto y un análisis comparativo de su eficacia utilizando conjuntos de datos públicos propicios para este fin. Probamos que los pipelines generados por Auto-sklearn, H2O AutoML y TPOT resultan eficaces para tareas de regresión, y logran mitigar la sobreadaptación que podrían tener sus modelos en búsqueda de la optimización.

Palabras clave: Aprendizaje Automático, Machine Learning, AutoML, Benchmarking, Evaluación Comparativa.

1 Introducción

El Aprendizaje Automático (Machine Learning) ha sido aplicado exitosamente en la resolución de problemas complejos en diversas áreas de negocios y en investigación, y se está abriendo camino en un número creciente de disciplinas [1]. Entre los ejemplos de éxito aplicados a los negocios se pueden mencionar los modelos de riesgo crediticio y los sistemas de recomendación [2]; mientras que entre los desafíos abordados en la investigación se encuentran, por ejemplo: el diagnóstico de enfermedades y la predicción de brotes epidémicos [3].

Sus éxitos dependen fundamentalmente de expertos en el área, quienes preprocesan los datos, y construyen un modelo predictivo eligiendo los algoritmos apropiados y configurando sus hiperparámetros adecuadamente [4]. Sin embargo, el rápido crecimiento de las aplicaciones de Aprendizaje Automático ha dado lugar a la demanda de procesos de aprendizaje que puedan ser generados fácilmente y sin los conocimientos de un experto. Al paradigma resultante se lo conoce como Automatización de Machine Learning: AutoML [5].

A pesar de estar destinado a usuarios sin conocimiento o dominio en Aprendizaje Automático, AutoML beneficia también a los expertos suministrando herramientas que

facilitan y complementan su labor, que en buena parte está compuesta por tareas repetitivas [6-7]. Así, se ha ido generando una creciente comunidad en torno a la creación de herramientas que automatizan dichas actividades [8].

El objetivo del presente trabajo es aportar conocimiento del estado actual de la Automatización del Aprendizaje Automático midiendo la eficacia en la resolución de tareas de regresión de algunas de las herramientas AutoML de código libre (open-source) más relevantes, analizando características que hacen a su usabilidad y complementando la labor previamente realizada en [9].

En las subsecciones siguientes se presenta el flujo de trabajo del Aprendizaje Automático de manera de establecer el objeto de la automatización, y se describen los principales hitos de AutoML. La sección 2 describe las herramientas relevadas en nuestro trabajo y el método empleado para su selección. Las secciones 3 y 4 presentan respectivamente los datos y los métodos utilizados para llevar a cabo la evaluación comparativa (benchmarking). La sección 5 despliega los resultados obtenidos. Finalmente, en la sección 6 se resumen los principales aportes del trabajo y se presentan futuras líneas de investigación.

1.1 Flujo de trabajo del Aprendizaje Automático

El flujo de trabajo tradicional de Aprendizaje Automático se divide en cuatro pasos principales: la adquisición de datos, el preprocesamiento, la creación de un modelo predictivo y la implementación de dicho modelo [10].

La adquisición de datos tiene como objetivo la identificación de conjuntos de datos (datasets) relevantes, y su calidad depende de los datos disponibles. La limpieza de datos (data cleaning), la ingeniería de características (feature engineering) y la selección de características (feature selection) son los pasos de preprocesamiento más comunes [11].

La construcción de un modelo depende de la tarea de aprendizaje a realizar (clasificación, regresión, etc.) e implica la selección de algoritmos de diversas familias de modelos y el ajuste de sus hiperparámetros [12].

El objetivo de todo el flujo es encontrar la secuencia de todas esas tareas (un pipeline) que permita una óptima calidad de generalización. Así, para medir cuan bueno es un pipeline, luego de su generación o entrenamiento (training), se lleva a cabo su evaluación (testing) con datos nunca antes vistos intentando optimizar una métrica dada, como por ejemplo el error cuadrático medio en el caso de tareas de regresión [13].

Encontrar el pipeline adecuado implica generalmente varios ciclos o iteraciones de entrenamiento y evaluación, y la eficiencia de su búsqueda depende en gran medida del conocimiento y experiencia del experto que la ejecuta [4].

1.2 AutoML

Existe hoy en día un gran interés en la Automatización del Aprendizaje Automático. Tal es así que la ICML (International Conference on Machine Learning) llevó a cabo en 2019 la sexta edición del workshop AutoML que, entre sus principales temáticas, incluye la búsqueda de la arquitectura neuronal óptima, el meta-aprendizaje, y la automatización del flujo de trabajo del Aprendizaje Automático [14]. Así también, Kaggle reportó según una encuesta propia que en 2019 el uso de Google Cloud AutoML (una herramienta en la nube) casi se duplicó en comparación con el año anterior [15].

En cuanto a las herramientas AutoML, hay frameworks que trabajan sobre todas las fases del flujo de trabajo, pero también hay otros que apuntan a la automatización de etapas específicas. Estas últimas son utilizadas por empresas que cuentan con expertos en el área permitiéndole a ellos centrarse en procesos más complejos, como por ejemplo la construcción de modelos, y no perder tiempo en tareas que requieren mucho esfuerzo y ensayos de tipo prueba-error, como la ingeniería de características y la optimización de hiperparámetros [16].

Así mismo, basados en la idea que el procesamiento previo a menudo es específico del dominio, mientras que la construcción del modelo de Aprendizaje Automático se aleja de esas particularidades, hay soluciones que sólo incluyen la selección del modelo y el ajuste de sus hiperparámetros, como por ejemplo ATM: Auto-Tuned Models [17].

El problema de la optimización de los hiperparámetros es similar al de selección del modelo, con la diferencia clave que los hiperparámetros son a menudo continuos, y por ende los espacios de búsqueda suelen ser más complejos [18]. Para los usuarios resulta difícil tomar la decisión correcta cuando se enfrentan a estos grados de libertad, así muchos seleccionan algoritmos basados en la reputación o atractivo intuitivo, y dejan los hiperparámetros establecidos en los valores predeterminados [19].

Se establece así uno de los grandes desafíos para AutoML: dado un conjunto de datos, elegir un algoritmo de aprendizaje y configurar sus hiperparámetros para optimizar el rendimiento en forma automática y simultánea. Este desafío es llamado problema de optimización de selección combinada de algoritmo y sus hiperparámetros (CASH: Combined Algorithm Selection and Hyperparameter optimization problem) y ha sido encarado a través de diversas estrategias como la optimización bayesiana, algoritmos genéticos y otros [19-20].

La optimización bayesiana, y en particular aquella basada en modelos secuenciales (SMBO) es un método de optimización estocástico que permite trabajar con hiperparámetros categóricos y continuos, y que puede explotar una estructura jerárquica derivada de parámetros condicionales [21]. En particular, la Configuración Secuencial de Algoritmos Basados en Modelos (SMAC) es una de sus implementaciones con mejores resultados en los frameworks AutoML, entre ellas Auto-sklearn [22], ganador del desafío ChaLearn AutoML I en 2015-2016 y ChaLearn AutoML II en 2017-2018.

2 Selección de Herramientas

Hay diversos frameworks AutoML de código abierto disponibles [23-28], generalmente creados y mantenidos por investigadores de universidades de distintas partes del mundo. Además, existe un número creciente de soluciones comerciales, algunas desarrolladas por gigantes tecnológicos como por ejemplo Google Cloud AutoML [29], y otras por startups [30-32].

Decidimos basar nuestro trabajo en herramientas de código abierto, priorizándolas por su popularidad definida en base a la cantidad de descargas y al número de referencias bibliográficas, y dando preferencia además a aquellas que incluyan la automatización de varias fases del flujo de trabajo [33-34]. Así, elegimos tres de los frameworks existentes que soportan tareas de regresión: Auto-sklearn, H2O AutoML y TPOT.

Es importante considerar que algunos de los frameworks open-source AutoML pueden solo estar disponibles a través de una API para un lenguaje de programación específico, mientras que otros pueden funcionar de forma independiente. Algunos sistemas pueden generar modelos que se pueden usar sin dependencia del paquete AutoML, y en otros casos, el sistema AutoML es necesario para utilizar el modelo. Finalmente, algunos sistemas pueden desarrollarse específicamente para un dominio determinado.

A continuación presentamos los frameworks seleccionados, detallando sus características y haciendo todo lo posible para proporcionar una descripción razonable que destaque algunos aspectos importantes o únicos de cada uno de ellos.

2.1 Auto-sklearn

Auto-sklearn permite optimizar automáticamente procesos de Aprendizaje Automático empleando una búsqueda bayesiana entre pipelines generados a través de métodos de la librería scikit-learn [35] de Python. La optimización contempla las fases de ingeniería de características, selección de características, selección del modelo (tanto de clasificación como de regresión) y optimización de sus hiperparámetros. Entre los métodos de ingeniería de características utilizados están la codificación one-hot, la estandarización de características numéricas, y PCA. Los pipelines generados utilizan modelos ensamblados (ensemble), una técnica que mejora los resultados del Aprendizaje Automático mediante la combinación de varios modelos [36]. El framework de optimización bayesiana utilizado es SMAC3, desarrollado por los creadores de Auto-sklearn. Además, tiene soporte multihilo dando lugar a ejecuciones paralelas y permite restringir el espacio de búsqueda excluyendo fases del flujo y/o algoritmos a ser utilizados en cualquiera de dichas fases [22].

Por defecto, los hiperparámetros se inicializan en valores predeterminados a partir de un pre-entrenamiento realizado sobre 140 conjuntos de datos de OpenML. Así, primero se obtienen estadísticas del dataset a optimizar, se las compara con ese meta-conocimiento, y se inicia la búsqueda con los hiperparámetros óptimos para el registro

más cercano. Esta característica de meta-aprendizaje incorporada permite realizar un arranque del proceso "en caliente" [22].

Permite configurar el tiempo máximo total del proceso, así como también el tiempo máximo dedicado a la evaluación de un pipeline. Por defecto, los valores son de 60 y 6 minutos respectivamente. No obstante el sitio oficial sugiere, de ser posible, estirar a un día el límite de tiempo total y a 30 minutos el límite para una única ejecución [25].

Su licencia es 3-clause BSD y es desarrollado por investigadores de la Universidad de Freiburg (Alemania).

2.2 H2O AutoML

H2O AutoML es una herramienta que permite optimizar flujos de trabajo de Aprendizaje Automático construidos sobre la plataforma de código abierto H2O [37] a través de búsqueda aleatoria. La plataforma H2O está desarrollada en Java, permite manejar grandes volúmenes de datos gracias a la implementación de técnicas de compresión de memoria e incluye interfaces para R, Python, Java, JSON y JavaScript, así como una interfaz web integrada: Flow [26].

Según su página oficial, H2O AutoML es una funcionalidad de H2O que automatiza el proceso de construcción de una gran cantidad de modelos, con el objetivo de encontrar el mejor sin ningún conocimiento previo por parte del Científico de Datos. Para ello, reduce al mínimo los parámetros a especificar en su interfaz. Entre dichos parámetros está el tiempo total para el proceso que, por defecto, es de 60 minutos [26].

H2O AutoML contempla las fases de selección del modelo y optimización de sus hiperparámetros. Para ello, la versión actual construye por defecto los siguientes modelos: Random Forest, Extremely-Randomized Forest, una grilla aleatoria de Gradient Boosting Machines (GBMs), una de Deep Neural Nets, y otra de Generalized Lineal Models (GLMs). Luego realiza dos ensamblados: uno conteniendo a todos los modelos anteriormente generados y otro con solo los mejores de cada familia. Finalmente se queda con el de mejor performance, que en la práctica suele ser alguno de los dos modelos ensamblados. Todos los modelos construidos son evaluados a través de validación cruzada y dichos resultados reportados al usuario a través de un tablero de líderes.

Puede ser implementado en Python y en R, su licencia es Apache-2.0 y es desarrollado y mantenido por H2O.ai.

2.3 TPOT

TPOT (Tree-based Pipeline Optimization Tool) es un optimizador de pipelines de Aprendizaje Automático basado en algoritmos genéticos. Puede ser ejecutado desde la línea de comandos o bien incluido como librería en Python. Extiende la librería scikit-learn con su propio método para tareas de clasificación y de regresión. Las fases que TPOT contempla son ingeniería de características, selección de características, selección del modelo y optimización de sus hiperparámetros [38].

En su sitio oficial se presenta como un asistente de Ciencia de Datos. Su licencia es LGPL-3.0 y es desarrollado por investigadores de la Universidad de Pennsylvania [24].

Busca la configuración óptima utilizando una amplia lista de métodos de preprocesamiento, regresores de scikit-learn y, opcionalmente, XGBoost [24]. Incluso evalúa pipelines con apilamiento (stacking), una táctica bastante común en Aprendizaje Automático. Una vez finalizado el proceso, permite exportar el código de Python para que se pueda reconstruir el pipeline sin dependencias de TPOT [39].

El sitio oficial sugiere ejecutar procesos por períodos prolongados (de horas a días) para permitir que TPOT realice una búsqueda exhaustiva. Así, con la configuración predeterminada TPOT evalúa 10000 configuraciones (100 generaciones con una población de 100 elementos cada una), dándole por defecto un tiempo máximo de ejecución de 5 minutos a cada pipeline. Las restricciones de tiempo se aplican cambiando el tiempo máximo de ejecución, el número de generaciones o el tamaño de la población. El proceso de optimización también admite la pausa y la reanudación [24].

Si bien los pipelines pueden ser de cualquier longitud, apunta a generar resultados con un número pequeño de componentes que a la vez optimicen la métrica especificada.

3 Selección y Preparación de Datos

Los conjuntos de datos fueron obtenidos de OpenML, un entorno colaborativo online para Aprendizaje Automático donde los usuarios pueden cargar y descargar conjuntos de datos, tareas, y ejecutar modelos fuera de la plataforma [40].

Esta vez decidimos centrar nuestro trabajo en tareas de regresión, filtrando nuestra búsqueda a datasets destinados a dicha tarea. Además, la restringimos a conjuntos de datos que no requieran limpieza, que no tengan datos faltantes, y que puedan ser utilizados en un análisis exhaustivo adecuado para experimentos prácticos en máquinas de capacidad básica, así sólo consideramos conjuntos de datos que:

• Posean un número de instancias entre 500 y 200.000, para enfocarnos en conjuntos que no sean demasiado pequeños para un entrenamiento adecuado pero tampoco demasiado grandes para la experimentación práctica.

• No tengan más de 500 atributos, para mantener bajo el tiempo de ejecución.

Una vez realizada la búsqueda con los filtros descriptos, seleccionamos 30 datasets, un número que consideramos suficiente para poder obtener conclusiones y adecuado a los recursos disponibles para el análisis. En la selección final se intentó dejar conjuntos con distintas combinaciones en cuanto a número de instancias, y cantidad de atributos numéricos y categóricos. Las características de estos conjuntos de datos se describen en la Tabla 1.

Una vez descargados, dividimos los datasets en dos partes. Así, el 80% de las instancias fueron utilizadas por las herramientas evaluadas para la construcción de los modelos y para una primera instancia de prueba mediante evaluación cruzada de 10 iteraciones, y el 20% restante se usó en una segunda etapa de evaluación para analizar la capacidad de predicción de los modelos elaborados por las diversas optimizaciones con datos nunca antes vistos.

Tabla 1. Características de los conjuntos de datos utilizados para la evaluación comparativa.

ID	Dataset	Instancias Totales	Instancias Entrenamiento	Instancias Prueba	Atributos Totales	Atributos Categóricos	Atributos Numéricos
183	abalone	4177	3341	836	9	1	8
189	kin8nm	8192	6553	1639	9	0	9
197	cpu_act	8192	6553	1639	22	0	22
215	2dplanes	40768	32614	8154	11	0	11
216	elevators	16599	13279	3320	19	0	19
223	stock	950	760	190	10	0	10
287	wine_quality	6497	5197	1300	12	0	12
296	ailerons	13750	11000	2750	41	0	41
308	puma32h	8192	6553	1639	33	0	33
344	mv	40768	32614	8154	11	3	8
405	mtp	4450	3560	890	203	0	203
422	topo_2_1	8885	7108	1777	267	0	267
503	wind	6574	5259	1315	15	0	15
507	space_ga	3107	2485	622	7	0	7
531	boston	506	404	102	14	2	12
534	cps_85_wages	534	427	107	11	7	4
537	houses	20640	16512	4128	9	0	9
541	socmob	1156	924	232	6	4	2
546	sensory	576	461	115	12	11	1
547	no2	500	400	100	8	0	8
549	strikes	625	500	125	7	0	7
558	bank32nh	8192	6553	1639	33	0	33
666	rmftsa_ladata	508	406	102	11	0	11
688	visualizing_soil	8641	6912	1729	5	1	4
1414	kaggle_bike	10886	8708	2178	11	6	5
4353	concrete	1030	824	206	9	0	9
4544	geographical	1059	847	212	118	0	118
41265	titanic	1307	1045	262	8	0	8
41539	rainfall	16755	13404	3351	4	2	2
41540	black_friday	166821	133456	33365	10	4	6

4 Evaluación comparativa

Basados en que el principal objetivo de las herramientas (según sus sitios oficiales) es ayudar a usuarios no expertos en técnicas de Aprendizaje Automático, decidimos establecer una línea base o de referencia (baseline) para evaluar la eficacia de las herramientas analizadas con los resultados que consideramos que un usuario principiante puede llegar a obtener. A la hora de construir un modelo, el enfoque más sencillo adoptado entre los no expertos es utilizar un regresor simplemente por su popularidad o atractivo intuitivo, con los valores predeterminados de sus parámetros, sin ninguna consideración empírica de las alternativas, y dejando fuera cualquier selección de atributos o ensamblado de modelos.

Utilizamos la librería scikit-learn (versión 0.22.0) para construir la línea base debido a su popularidad y siendo que sus métodos son utilizados por dos de las tres herramientas AutoML analizadas en el presente trabajo. Los regresores incluidos fueron: KNeighborsRegressor, LinearRegression, Ridge, Lasso, GaussianProcessRegressor, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, SVR y MLPRegressor.

Para obtener el regresor correspondiente a la línea base de cada uno de los conjuntos de datos, se eligió el mejor de los modelos mencionados según error cuadrático medio (ECM) en el conjunto de entrenamiento utilizando una validación cruzada de 10 iteraciones y la misma semilla.

Siguiendo los lineamientos de los desarrolladores de las herramientas analizadas, que en sus documentos oficiales sostienen que los algoritmos de AutoML no están diseñados para ejecutarse solo por unos pocos minutos (“Si no ejecuta TPOT durante el tiempo suficiente, es posible que no encuentre el mejor pipeline posible para su conjunto” [27]) se decidió establecer un tiempo de 60 minutos para cada proceso de optimización.

Así mismo, teniendo en cuenta el objetivo de las herramientas antes mencionado, decidimos utilizar para las pruebas, computadoras estándar a la que cualquiera podría tener acceso. Se usaron dos equipos para acelerar los tiempos, ambas con sistema operativo Windows 10, con las siguientes características a nivel hardware: 8GB de RAM, CPU Intel I5-6500 3.20 GHz, 1 TB HDD.

4.1 Configuración Auto-sklearn

Se usó la versión 0.6.0. De todas las herramientas analizadas, resultó ser la de mayor requisitos en cuanto a dependencias. Entre ellas está el modulo resource que no está disponible en Windows, por lo que Auto-sklearn requiere ser ejecutado en Linux. Instalamos Windows 10 Bash Shell, y allí se siguieron los pasos de instalación enumerados en el manual sin presentarse inconveniente alguno.

De los parámetros predeterminados realizamos las siguientes modificaciones para llevar a cabo las pruebas: establecimos la métrica a ser utilizada en ECM, y seleccionamos validación cruzada de 10 iteraciones como la estrategia de remuestreo, habiendo leído que es lo ideal para evitar la sobreadaptación (overfitting) en sets de datos de tamaño pequeño. También eliminamos el arranque en "caliente" para evitar la ventaja que podría tener esta herramienta siendo que dicho módulo de meta-aprendizaje fue generado a partir de sets de datos obtenidos de OpenML. Así mismo, establecimos la semilla para poder reproducir los resultados.

Para realizar las pruebas generamos los scripts que luego ejecutamos desde la línea de comandos. Como Auto-sklearn no maneja atributos de tipo string, incluimos la transformación manual a códigos enteros en los scripts. Además, siendo que viene con su propia variación de codificación one-hot [41] incluida, decidimos incorporar nuestra propia codificación en los scripts antes de la invocación a la librería, para que eso no afecte la comparación con las otras herramientas.

4.2 Configuración H2O AutoML

Se usó la versión 3.28.1.1. Tanto H2O AutoML como sus dependencias no presentaron inconvenientes en su instalación. De los parámetros predeterminados realizamos las siguientes modificaciones para llevar a cabo las pruebas: establecimos en una hora el tiempo máximo del proceso (es también el valor por defecto si es que no se establece un máximo de modelos a evaluar) , seleccionamos validación cruzada de 10 iteraciones como la estrategia de remuestreo y ECM como métrica a utilizar, y establecimos la semilla para poder reproducir los resultados.

Para realizar las pruebas generamos los scripts que luego ejecutamos desde la línea de comandos. Como H2O AutoML no maneja atributos de tipo string, incluimos la transformación manual a códigos enteros en los scripts. También decidimos agregar nuestra propia codificación en los scripts previo a la invocación de la librería para que eso no afecte la comparación con las otras herramientas.

4.3 Configuración TPOT

Se usó la versión 0.11.0. Tanto TPOT como sus dependencias no presentaron inconvenientes en su instalación. No se agregó XGBoost, componente opcional para que los modelos eXtreme Gradient Boosting sean utilizados en la optimización.

De los parámetros predeterminados realizamos las siguientes modificaciones para llevar a cabo las pruebas: bajamos el tamaño de la población a 50, acotamos a una hora el tiempo máximo del proceso, establecimos ECM como métrica a ser utilizada, y establecimos la semilla para poder reproducir los resultados.

Para realizar las pruebas generamos los scripts que luego ejecutamos desde la línea de comandos. Como TPOT no maneja atributos de tipo string, incluimos la transformación manual a códigos enteros en los scripts. También decidimos agregar nuestra

propia codificación en los scripts previo a la invocación de la librería, siendo que TPOT viene con su propia variación de codificación one-hot [41] incluida, para que eso no afecte la comparación con las otras herramientas.

5 Resultados

La Tabla 2 muestra los resultados obtenidos en ambas instancias de evaluación. Las primeras columnas presentan los ECM obtenidos empleando validación cruzada con 10 iteraciones sobre los datos de entrenamiento, tanto por el método de la línea base como por los pipelines generados por las herramientas AutoML. Mientras que las columnas restantes muestran los resultados obtenidos sobre los conjuntos de prueba.

Allí se muestran resaltados los valores obtenidos por las herramientas que resultaron superiores al EMC de la línea base en la correspondiente instancia de evaluación.

Ningún conjunto de valores de ECM presentó una distribución normal según la prueba de Shapiro-Wilk [42]. Por lo tanto procedimos a realizar la prueba no paramétrica de rangos con signo de Wilcoxon [43] para determinar cuales de las herramientas AutoML analizadas logra una diferencia significativa sobre la línea base. Así, rechazaríamos la hipótesis nula que la herramienta AutoML ofrece iguales resultados que la línea base en un nivel de significancia de 0,05.

Auto-sklearn obtuvo resultados superiores a la línea base en 21 de los 30 casos en la instancia de entrenamiento, y en 26 en la instancia de prueba. El test de Wilcoxon reportó que dicha diferencia fue significativa en los conjuntos de entrenamiento ($t=134$, $p=0,043$), y también en los conjuntos de prueba ($t=34$, $p=0,00004$).

Los resultados obtenidos por H2O fueron significativamente superiores a los de la línea base en el conjunto de entrenamiento ($t=49$, $p=0,0002$) en donde logró superar a la línea base en 25 casos, y también en 24 de los conjuntos de prueba ($t=89$, $p=0,003$).

TPOT obtuvo resultados superiores a los de la línea base en 27 de los 30 casos en la instancia de entrenamiento, y 23 en la de prueba. Dichas diferencias fueron significas ($t=51$, $p=0,0002$ en los conjuntos de entrenamiento; y $t=128$, $p=0,032$ en los conjuntos de prueba).

La Tabla 3 muestra los resultados obtenidos en ambas instancias de evaluación por las herramientas AutoML. Allí se muestran resaltados los valores mínimos de ECM logrados para cada dataset en cada instancia de evaluación.

Ejecutamos la prueba no paramétrica de muestras relacionadas de Friedman [44] para determinar si hay diferencias significativas en los resultados obtenidos por las distintas herramientas AutoML analizadas.

Así, rechazaríamos la hipótesis nula que los resultados arrojados por las distintas herramientas AutoML son iguales en un nivel de significancia de 0,05.

El test de Friedman arrojó una diferencia significativa entre los EMC obtenidos por las herramientas en los conjuntos de entrenamiento ($\chi 2=16,487$, $p=0,0003$).

Tabla 2. EMC obtenidos por línea base y herramientas en ambas instancias de evaluación.

EMC con Validación Cruzada				EMC en Conjunto de Pruebas			
Línea base	**Auto-sklearn**	**H2O**	**TPOT**	**Línea base**	**Auto-sklearn**	**H2O**	**TPOT**
4,7562	4,5394	4,2787	4,2715	4,6474	4,1512	4,4219	4,1779
0,0068	0,0136	0,0090	0,0114	0,0066	0,0134	0,0081	0,0109
5,7627	5,8932	4,9562	5,6846	6,0963	5,5633	5,1032	5,8103
1,0071	1,0560	1,0100	1,0008	0,9851	1,0260	0,9860	0,9800
7,73E-06	6,00E-06	4,70E-06	5,66E-06	6,40E-06	4,59E-06	4,13E-06	5,23E-06
0,5978	0,5908	0,5512	0,3979	0,6621	0,5114	0,5608	0,3978
0,3815	0,3841	0,3666	0,3634	0,3528	0,3466	0,3250	0,3260
3,11E-08	3,00E-08	2,58E-08	2,58E-08	2,89E-08	4,20E-08	2,29E-08	2,39E-08
1,00E-04	5,80E-05	5,73E-05	6,02E-05	1,00E-04	5,21E-05	5,77E-05	6,09E-05
0,0051	0,0062	0,0070	0,0101	0,0080	0,0054	0,0074	0,0083
0,0130	0,0149	0,0121	0,0127	0,0138	0,0151	0,0122	0,0132
8,63E-04	8,44E-04	8,48E-04	8,48E-04	7,45E-04	7,40E-04	7,43E-04	7,39E-04
9,7812	9,1525	9,1228	9,2170	9,3264	8,8417	8,7374	9,1665
0,0127	0,0117	0,0105	0,0096	0,0185	0,0129	0,0135	0,0142
10,1814	9,4575	8,7091	7,2093	14,9322	11,7409	16,3264	12,1018
16,4620	17,3232	16,2839	16,3236	25,9664	25,4509	26,1211	27,7868
2,45E+09	2,46E+09	2,00E+09	2,48E+09	2,24E+09	2,17E+09	1,82E+09	2,30E+09
240,9232	185,0104	123,1490	160,6224	151,7773	87,6835	100,8424	182,5834
0,5296	0,4668	0,4956	0,4808	0,4566	0,4240	0,4288	0,4166
0,2106	0,2102	0,2033	0,2063	0,2454	0,2305	0,2309	0,2430
3,24E+05	2,97E+05	2,97E+05	2,89E+05	1,41E+05	6,82E+04	7,90E+04	6,19E+04
0,0070	0,0067	0,0064	0,0066	0,0069	0,0065	0,0060	0,0063
3,1689	2,5529	2,6895	2,3379	4,5982	3,3902	4,1231	3,6689
0,0017	0,0023	0,0102	0,0011	0,0008	0,0007	0,0086	0,0006
5740,4574	5451,0956	4856,8249	5424,4118	5662,7939	5211,1537	4722,9271	5418,7035
25,4024	19,1605	16,5536	17,0092	28,8500	22,2325	23,1955	26,4469
0,2206	0,2160	0,2227	0,2108	0,1736	0,1598	0,1557	0,1760
1370,7938	1324,6211	1280,8367	1220,0326	1211,1664	1164,6887	1502,2649	1357,2170
20892,7055	17883,4232	14587,6570	15566,5133	19569,3102	16545,0194	14410,7953	15336,1513
1,28E+07	1,25E+07	1,20E+07	1,24E+07	1,27E+07	1,23E+07	1,19E+07	1,22E+07

Tabla 3. EMC obtenidos por las herramientas en los conjuntos de entrenamiento y de prueba.

ID	EMC con Validación Cruzada			EMC en Conjunto de Pruebas		
	Auto-sklearn	H2O	TPOT	Auto-sklearn	H2O	TPOT
183	4,5394	4,2787	4,2715	4,1512	4,4219	4,1779
189	0,0136	0,0090	0,0114	0,0134	0,0081	0,0109
197	5,8932	4,9562	5,6846	5,5633	5,1032	5,8103
215	1,0560	1,0100	1,0008	1,0260	0,9860	0,9800
216	6,00E-06	4,70E-06	5,66E-06	4,59E-06	4,13E-06	5,23E-06
223	0,5908	0,5512	0,3979	0,5114	0,5608	0,3978
287	0,3841	0,3666	0,3634	0,3466	0,3250	0,3260
296	3,00E-08	2,58E-08	2,58E-08	4,20E-08	2,29E-08	2,39E-08
308	5,80E-05	5,73E-05	6,02E-05	5,21E-05	5,77E-05	6,09E-05
344	0,0062	0,0070	0,0101	0,0054	0,0074	0,0083
405	0,0149	0,0121	0,0127	0,0151	0,0122	0,0132
422	8,44E-04	8,48E-04	8,48E-04	7,40E-04	7,43E-04	7,39E-04
503	9,1525	9,1228	9,2170	8,8417	8,7374	9,1665
507	0,0117	0,0105	0,0096	0,0129	0,0135	0,0142
531	9,4575	8,7091	7,2093	11,7409	16,3264	12,1018
534	17,3232	16,2839	16,3236	25,4509	26,1211	27,7868
537	2,46E+09	2,00E+09	2,48E+09	2,17E+09	1,82E+09	2,30E+09
541	185,0104	123,1490	160,6224	87,6835	100,8424	182,5834
546	0,4668	0,4956	0,4808	0,4240	0,4288	0,4166
547	0,2102	0,2033	0,2063	0,2305	0,2309	0,2430
549	2,97E+05	2,97E+05	2,89E+05	6,82E+04	7,90E+04	6,19E+04
558	0,0067	0,0064	0,0066	0,0065	0,0060	0,0063
666	2,5529	2,6895	2,3379	3,3902	4,1231	3,6689
688	0,0023	0,0102	0,0011	0,0007	0,0086	0,0006
1414	5451,0956	4856,8249	5424,4118	5211,1537	4722,9271	5418,7035
4353	19,1605	16,5536	17,0092	22,2325	23,1955	26,4469
4544	0,2160	0,2227	0,2108	0,1598	0,1557	0,1760
41265	1324,6211	1280,8367	1220,0326	1164,6887	1502,2649	1357,2170
41539	17883,4232	14587,6570	15566,5133	16545,0194	14410,7953	15336,1513
41540	1,25E+07	1,20E+07	1,24E+07	1,23E+07	1,19E+07	1,22E+07

En dicha instancia H2O AutoML logró superioridad en 15 casos, seguida de TPOT que superó al resto de las herramientas en 12 datasets, mientras que Auto-sklearn registró supremacía en solo 3 conjuntos de datos. Efectuamos un análisis post hoc mediante las pruebas de rangos con signo de Wilcoxon con corrección de Bonferroni [45], lo que dio como resultado un nivel de significación establecido en $p < 0,017$. La superioridad de H2O sobre Auto-sklearn resultó significativa ($t=64$, $p=0,0005$) y también la de TPOT sobre Auto-sklearn ($t=76$, $p=0,001$). Sin embargo, la diferencia de H2O por sobre TPOT no resultó significativa ($t=192$, $p=0,581$).

En los datos de prueba H2O AutoML logró superioridad en 13 casos, esta vez seguida de Auto-sklearn que superó al resto de las herramientas en 11 datasets, mientras que TPOT registró supremacía en los 6 conjuntos de datos restantes. Sin embargo, el test de Friedman arrojó que la diferencia entre los EMC obtenidos por las herramientas en esta instancia de evaluación no es significativa ($\chi2=2,467$, $p=0,291$).

6 Conclusiones

Las tres herramientas logran mejores resultados que los de la línea base en ambas instancias de comparación. Tanto en la evaluación con datos de entrenamiento, utilizando validación cruzada, como en la evaluación con datos de prueba, la superioridad de cada herramienta por sobre la línea base es estadísticamente significativa.

Esto nos permitiría afirmar que, en tareas de regresión, el objetivo de estas herramientas -ayudar a usuarios no expertos en el uso de técnicas de Aprendizaje Automático- está siendo logrado.

El hecho de que la superioridad es significativa en la segunda instancia nos permitiría afirmar que los pipelines generados no sufren sobreadaptación.

En cuanto a la comparación entre ellas, vemos que H2O AutoML y TPOT resultan significativamente superiores a Auto-sklearn en la primer instancia de evaluación, aunque no se ven diferencias significativas en la evaluación con datos de prueba.

Ninguna de las herramientas presentó inconvenientes en su implementación. Todas disponen de mantenimiento activo y presentan una correcta documentación.

Las tres emplean técnicas para generar modelos ensamblados y así poder lograr mejores resultados. Destacamos el módulo de inicio rápido de Auto-sklearn, si bien no fue utilizado en esta evaluación comparativa. También nos resultó interesante la posibilidad de exportar el pipeline generado de TPOT para ser utilizado sin dependencias, y consideramos importante resaltar los resultados de H2O AutoML que, sin incluir selección ni ingeniería de características en sus pipelines, son comparables al de las otras herramientas.

Creemos que realizar una evaluación comparativa de estas herramientas con una línea base más cercana a lo que realiza un usuario con cierto grado de experiencia permitirá profundizar acerca del estado actual del AutoML, y la consideramos una interesante futura línea de investigación.

Referencias

[1] Finley, S., "Artificial Intelligence and Machine Learning for Business: A No-Nonsense Guide to Data Driven Technologies", Relativistic, ISBN 978-1999730307, 2017.
[2] Agrawal, A., Gans, J. and Goldbard, A., "Prediction Machines: The Simple Economics of Artificial Intelligence", Harvard Business Review Press, ISBN 978-1633695672, 2018.
[3] Rouhiainen, M., "Artificial Intelligence: 101 Things You Must Know Today About Our Future", CreateSpace Independent Publishing Platform. ISBN: 978-1982048808, 2018.
[4] Luo, G., "A review of automatic selection methods for machine learning algorithms and hyper-parameter values", Network Modeling Analysis in Health Informatics and Bioinformatics. 5. doi:10.1007/s13721-016-0125-6, 2016.
[5] AutoML.org, https://www.automl.org/automl/.
[6] Perer, A., Reddy, C., Riabov, A., Samulowitz, H., Sow, D., Tesauro, G. and Turaga, D., "Towards cognitive automation of data science", in Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 4268– 4269, 2015.
[7] Kaul, A., Maheshwary, S. and Pudi, V., "AutoLearn — Automated Feature Generation and Selection", in 2017 IEEE International Conference on Data Mining (ICDM), pp. 217-226, doi: 10.1109/ICDM.2017.31, 2017.
[8] Brazdil, P. and Giraud-Carrier, C., "Metalearning and Algorithm Selection: progress, state of the art and introduction to the 2018 Special Issue", Machine Learning 107(1): 1-14, doi:10.1007, 2018.
[9] Bender, A., Macrino, M., Nicolet, S., "Evaluación Comparativa y Análisis de Usabilidad de Herramientas AutoML de Código Abierto", en 7º Congreso Nacional de Ingeniería Informática – Sistemas de Información, 2019.
[10] Sarkar, D., Bali, R., Sharma, T., "Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems", Apress; Edición: 1st ed., ISBN-13: 978-1484232064, 2017.
[11] Cakmak, U. and Das, S., "Hands-On Automated Machine Learning", Packt Publishing, ISBN: 9781788629898, 2018.
[12] Domingos, P., "A few useful things to know about machine learning", Communications of the ACM, vol. 55, no. 10, pp. 78–87, doi: 10.1145/2347736.2347755, 2012.
[13] Mohr, F., Wever, M. and Hüllermeier, E., "ML-Plan: Automated machine learning via hierarchical planning", Mach Learn 107, pp.1495–1515, doi: 10.1007/s10994-018-5735-z, 2018.
[14] ICML: International Conference on Machine Learning, https://icml.cc/.
[15] Kaggle, "State of Machine Learning and Data Science", 2019.
[16] Kanter, J. and Veeramachaneni, K., "Deep feature synthesis: Towards automating data science endeavors", in 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1-10, doi: 10.1109/DSAA.2015.7344858, 2015.
[17] Swearingen, T., Drevo, W., Cyphers, B., Cuesta-Infante, A., Ross, A., and Veeramachaneni, K., "ATM: A distributed, collaborative, scalable system for automated machine learning", in 2017 IEEE International Conference on Big Data (Big Data), pp. 151-162, 2017.
[18] Malkomes, G. Schaff, C., and Garnett, T., "Bayesian optimization for automated model selection", in NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 2900-2908, 2016.
[19] Snoek, J., Larochelle, H., and Adams, R., "Practical bayesian optimization of machine learning algorithms", in Advances in Neural Information Processing Systems, pp. 2951–2959, 2012.
[20] Real, E., Moore, S., Selle, A., Saxena, S.,, Suematsu, Y., Tan, J., Le, Q., and Kurakin, A., "Large-scale evolution of image classifiers", in International Conference on Machine Learning, 2017.
[21] Hutter, F., Hoos, H. and Leyton-Brown, K., "Sequential model-based optimization for general algorithm configuration", in Proceedings of the 5th international conference on Learning

and Intelligent Optimization (LION'05), Carlos Coello Coello (Ed.). Springer-Verlag, pp. 507-523. doi:10.1007/978-3-642-25566-3_40, 2011.
[22] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M. and Hutter, F., "Efficient and Robust Automated Machine Learning", in Advances in Neural Information Processing Systems 28-NIPS2015. Curran Associates, Inc., pp. 2962-2970, 2015.
[23] ML-Plan, https://fmohr.github.io/AILibs/projects/mlplan/.
[24] MLBox, https://mlbox.readthedocs.io/.
[25] auto-sklearn, https://automl.github.io/auto-sklearn/.
[26] H2O AutoML Documentation, http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html.
[27] TPOT, http://epistasislab.github.io/tpot/.
[28] Auto-WEKA, https://www.cs.ubc.ca/labs/beta/Projects/ autoweka/.
[29] Google Cloud AutoML, https://cloud.google.com/automl/.
[30] DataRobot, https://www.datarobot.com/.
[31] Auger AI, https://auger.ai/.
[32] mljar, https://mljar.com/.
[33] "Auto Machine Learning Software/ Tools in 2019: In-depth Guide", AI Multiple Blog, Descargado 30/06/2019, https://blog.aimultiple.com/auto-ml-software/.
[34] "Automatic Machine Learning (AutoML) Landscape Survey", Georgian Impact Blog. Descargado 30/06/2019, https://medium.com/georgian-impact-blog/automatic-machine -learning-aml-landscape-survey-f75c3ae3bbf2.
[35] Pedregosa, F., Varoquaux, G., Gramfort, A., et al., "Scikit-learn: Machine learning in python", Journal of machine learning research, 2825–2830, 2011.
[36] Caruana, R., Niculescu-Mizil, A., Crew, G. and Ksikes, A., "Ensemble selection from libraries of models", in Proc. of ICML'04, page 18, 2004.
[37] H2O Documentation, http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html.
[38] Olson, R., Urbanowicz, R., Andrews, P., Lavender, N., Creis Kidd, L. and Moore, J., "Automating biomedical data science through tree-based pipeline optimization", Applications of Evolutionary Computation, pp. 123-137, 2016.
[39] Olson, R., Bartley, N., Urbanowicz, R. and Moore, J., "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science", in Proceedings of GECCO 2016, pp. 485-492, 2016.
[40] Vanschoren, J., Rijn, J., Bischl, B. and Torgo, L., "Openml: Networked science in machine learning", in SIGKDD Explorations, vol. 15, 2, pp. 49-60, 2013.
[41] Gujarati, D., "Basic econometrics". McGraw Hill. p. 1002. ISBN 0-07-233542-4, 2003.
[42] Shapiro, S., Wilk, M., "An analysis of variance test for normality (complete samples)". Biometrika, 52(3–4), 591–611. https://doi.org/10.1093/biomet/52.3-4.591, 1965.
[43] Wilcoxon, F., "Individual comparisons by ranking methods" Biometrics Bulletin. 1 (6), Dec, p80–83, 1945.
[44] Friedman, M., "A Comparison of Alternative Tests of Significance for the Problem of m Rankings". Ann.Math. Statist.11, no. 1, 86-92. doi:10.1214/aoms/1177731944, 1940.
[45] Bonferroni, C., "Teoria statistica delle classi e calcolo delle probabilità", Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze, 1936.