Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Inteligencia artificial

Integración de aprendizaje automático en software ERP

Trabajo fin de estudio presentado por:	Santiago Noé Bonay Leites
Tipo de trabajo:	Desarrollo software Comparativa de soluciones
Director/a:	Carlos Rubert Escuder
Fecha:	16/05/2024

Resumen

Los software de planificación de recursos empresariales (ERP) se han convertido en herramientas cruciales para la eficiencia y optimización de procesos en el entorno empresarial. Representan una solución integral que unifica y automatiza múltiples procesos de negocio en una única plataforma, englobando funciones críticas como finanzas, gestión de relaciones con clientes (CRM), abastecimiento, gestión de inventario, recursos humanos, producción y logística, entre otros. Al centralizar la información en una base de datos única, los ERP consiguen facilitar la gestión de la información que registra la empresa, consiguiendo datos de alta calidad en tiempo real. Integrar en el ERP soluciones de aprendizaje automático (ML) puede mejorar la productividad y la toma de decisiones, aprovechando la calidad de los datos almacenados. Sin embargo, la complejidad de implementar, adaptar y desplegar modelos de ML sigue suponiendo un desafío considerable. Este trabajo explora el uso de tecnologías de aprendizaje automático automatizado (AutoML) para superar estas barreras, desarrollando un cuadro de mando integrado en el ERP Libra con el objetivo de facilitar la implementación de algoritmos de clasificación automáticos. El objetivo es proporcionar un sistema que democratice el uso de ML en ERPs, haciéndolo accesible a usuarios no especializados. Para analizar la viabilidad y éxito del cuadro de mando desarrollado se han definido dos casos de uso de clasificación. El primero de los casos de uso se trata de una tarea de clasificación supervisada orientada a predecir el riesgo de impuntualidad en el pago de un cliente en el momento de realizar una venta. El segundo caso corresponde a clasificación no supervisada, cuyo objetivo será agrupar los productos en inventario en base a patrones de demanda y características de almacenaje para optimizar la gestión del inventario.

Palabras clave: ERP, aprendizaje automático, clasificación supervisada, clasificación no supervisada, AutoML

Abstract

Enterprise Resource Planning (ERP) software has become a crucial tool for efficiency and process optimization in the business environment. They represent a comprehensive solution that unifies and automates multiple business processes on a single platform, encompassing critical functions such as finance, customer relationship management (CRM), procurement, inventory management, human resources, production, and logistics, among others. By centralizing information in a single database, ERPs facilitate the management of the information recorded by the company, achieving high-quality real-time data. Integrating machine learning (ML) solutions into ERP can improve productivity and decision-making by leveraging the quality of the stored data. However, the complexity of implementing, adapting, and deploying ML models continues to pose a significant challenge. This work explores the use of automated machine learning (AutoML) technologies to overcome these barriers, developing an integrated dashboard in the Libra ERP with the goal of facilitating the implementation of automatic classification algorithms. The aim is to provide a system that democratizes the use of ML in ERPs, making it accessible to non-specialized users. To analyze the viability and success of the developed dashboard, two classification use cases have been defined. The first use case involves a supervised classification task aimed at predicting the risk of payment delinquency by a client at the time of making a sale. The second case corresponds to unsupervised classification, whose goal will be to group inventory products based on demand patterns and storage characteristics to optimize inventory management.

Keywords: ERP, machine learning, supervised classification, unsupervised classification, AutoML

Índice de contenidos

1. Introducción 1
1.1. Motivación 1
1.2. Planteamiento del trabajo 2
1.3. Estructura del trabajo 3
2. Contexto y estado del arte 4
2.1. Evolución del software ERP 4
2.2. Machine Learning en ERP 5
2.3. Clasificación supervisada 8
2.4. Clasificación no supervisada 9
2.5. AutoML 10
2.6. Conclusiones 13
3. Objetivos concretos y metodología de trabajo 14
3.1. Objetivo general 14
3.2. Objetivos específicos 14
3.2.1. Selección de algoritmos de ML 14
3.2.2. Investigación y selección de librerías AutoML 14
3.2.3. Desarrollo de cuadro de mando AutoML integrado en el ERP LIBRA 14
3.2.4. Evaluación del resultado 14
3.3. Metodología del trabajo 15
4. Desarrollo específico de la contribución 18
4.1. Conjuntos de datos 18
4.1.1. Conjunto de datos para clasificación supervisada 18
4.1.2. Conjunto de datos para clasificación no supervisada 20
4.2. Entrenamiento manual de modelos 21

4.2.1. Clasificación supervisada manual 22
4.2.2. Clasificación no supervisada manual 27
4.3. Comparativa soluciones AutoML Open Source 35
4.3.1. AutoGluon 35
4.3.2. PyCaret 38
4.3.3. H2O 42
4.3.4. Comparativa AutoML: clasificación supervisada 43
4.3.5. Comparativa AutoML: clasificación no supervisada 49
4.3.6. Selección de solución AutoML 52
4.4. Integración de PyCaret en ERP Libra 54
4.4.1. Diseño del API AutoML 55
4.4.2. Diseño del dataset en Libra 56
4.4.3. Cuadro de mando AutoML integrado en Libra 56
4.4.4. Consumo del modelo entrenado en Libra 60
5. Conclusiones y trabajo futuro 61
5.1. Conclusiones 61
5.2. Líneas de trabajo futuro 62
Bibliografía 64
Anexo A. Código fuente y datos analizados 67

Índice de figuras

Figura 1: Historia de los ERP 5
Figura 2: Aplicaciones de ML por módulo del ERP 7
Figura 3: Flujo de trabajo AutoML 12
Figura 4: Comparativa de soluciones AutoML 13
Figura 5: Etapas metodología CRISP-DM 16
Figura 6: Flujo de trabajo AGILE 16
Figura 7. Matriz de confusión unificada 27
Figura 8. Gráfico polar Agglomerative Clustering 31
Figura 9. Gráfico polar KMeans 32
Figura 10. Gráfico de tarta para distribución de clústeres 32
Figura 11. Mapa de calor de las features por clúster 33
Figura 12. Gráfico DBSCAN aplicando PCA 34
Figura 13. Gráfico de tarta DBSCAN 34
Figura 14. PyCaret, feature importance plot retail dataset 47
Figura 15. PyCaret, classification report retail dataset 47
Figura 16. PyCaret clustering K-Means. Gráfico 2D PCA 50
Figura 17. PyCaret clustering K-Means. Gráfico distribución de clústeres 51
Figura 18. PyCaret clustering K-Means. Gráfico Silhouette 51
Figura 19. PyCaret Agglomerative clustering. Gráfico 3D TSNE. 52
Figura 20. Diseño de datasets en Libra 56
Figura 21. Cuadro de mando AutoML en el menú de Libra 57
Figura 22. Diseño de cuadro de mano AutoML en Libra 57
Figura 23. Resultados entrenamiento AutoML en Libra 58
Figura 24. Entrenamiento clustering en Libra 59

Figura 25. Historial de entrenamientos Libra 59

Figura 26. Personalización LIBRA para consumo del modelo 60

Figura 27. Prueba de consumo de modelo entrenado. 61

Figura 28. Resultado de la prueba de predicción 61

Índice de tablas

Tabla 1. Clasificación supervisada, variable objetivo 18

Tabla 2. Clasificación supervisada, descripción de variables 20

Tabla 3. Información datasets clasificación supervisada por sector 20

Tabla 4. Clasificación no supervisada, descripción de variables 21

Tabla 5. Información datasets clasificación no supervisada por sector 21

Tabla 6. Resultados entrenamiento manual clasificación supervisada 25

Tabla 7. Rendimiento promedio 26

Tabla 8. Resultados modelos clasificación no supervisada 30

Tabla 9. Promedios por modelo 30

Tabla 10. AutoGluon, parámetros TabularPredictor 36

Tabla 11. AutoGluon, métodos TabularPredictor 37

Tabla 12. AutoGluon, lista de modelos 38

Tabla 13. PyCaret, parámetros de inicialización (setup) 39

Tabla 14. PyCaret, parámetros de los métodos principales 41

Tabla 15. PyCaret, modelos de clasificación disponibles 41

Tabla 16. PyCaret, modelos de clustering disponibles 42

Tabla 17. AutoGluon, evaluación resultados clasificación supervisada 44

Tabla 18. PyCaret, evaluación resultados clasificación supervisada 46

Tabla 19. H2O, resultados clasificación supervisada 48

Tabla 20. H2O, leaderboard para dataset distribución farmacéutica 48

Tabla 21. H2O, importancia de características 49

Tabla 22. PyCaret, resultados clustering 50

1. Introducción

1.1. Motivación

En el entorno empresarial actual, los sistemas de planificación de recursos empresariales (ERP) juegan un papel crucial en la optimización y eficiencia de los procesos operativos (Wieder, 2006). A pesar de su amplia adopción, la integración de soluciones avanzadas de aprendizaje automático (ML) en estos sistemas sigue siendo un desafío significativo, debido a la complejidad técnica y la necesidad de adaptación a contextos empresariales concretos (Godbole, 2023). Este trabajo se centra en abordar el problema de la accesibilidad y la implementación de técnicas de aprendizaje automático embebidas en el ERP mediante técnicas de aprendizaje automático automatizado (AutoML).

Se pretende aprovechar para ello el diseño de los software ERP, que se basan en la utilización de bases de datos relacionales que implementan el modelo ACID (*Atomicity, Consistency, Isolation, Durability).* Esto garantiza en gran medida una alta calidad de los datos registrados en el ERP, evitando duplicidad de datos y reduciendo la cantidad de datos erróneos o mal estructurados (Frank, 2008), lo que proporciona una base sólida para la explotación de los datos utilizando técnicas de aprendizaje automático

Mientras que los ERPs son capaces de gestionar y procesar grandes volúmenes de datos, muchas empresas no logran explotar completamente estas capacidades para mejorar la toma de decisiones basadas en datos y la automatización de tareas. Investigaciones previas han demostrado que la incorporación de ML puede transformar radicalmente la utilidad de los sistemas ERP, proporcionando un mayor conocimiento sobre los datos disponibles (Jawad, 2024). Sin embargo, existe una barrera tanto técnica como de coste para implementar soluciones de este tipo en pequeñas y medianas empresas (PyMEs), pues la puesta en marcha de soluciones de aprendizaje automático requieren una adaptación a las necesidades específicas de cada empresa, lo que supone un gasto en personal especializado o bien en servicios adicionales de consultoría externa (Juli, 2024). Para mitigar este hecho se pretende investigar la viabilidad de dotar al ERP de una serie de algoritmos estándar que puedan ser utilizados por la mayor parte de los clientes del ERP, aprovechando que todos comparten en gran medida la misma estructura de base de datos subyacente al ERP (Wieder, 2006).

En concreto, este trabajo busca ampliar la funcionalidad actual del ERP LIBRA[1], desarrollado por la empresa EDISA, que considera primordial mantener sus productos actualizados con las tendencias recientes en el ámbito del desarrollo de software, donde la inteligencia artificial destaca en los últimos años. EDISA pretende mantener LIBRA como un producto competitivo frente a ERPs desarrollados por empresas de un tamaño muy superior, como Oracle, Microsoft, SAP, Sage o Infor, que ya implementan o están desarrollando soluciones con técnicas de inteligencia artificial[2] (Josyula & Godbole, 2024). En este sentido el dotar a LIBRA de un cuadro de mando de aprendizaje automático puede suponer una ventaja competitiva frente a otros ERP y una base sobre la que incorporar nuevos algoritmos en el futuro.

1.2. Planteamiento del trabajo

El presente trabajo busca desarrollar un cuadro de mando de aprendizaje automático dentro del ERP LIBRA. En dicho cuadro de mando se permitirá a los usuarios avanzados del ERP, seleccionar un origen de datos previamente definido con las herramientas ya existentes en el ERP (*dataset*), una tarea a realizar sobre dichos datos (clasificación supervisada o no supervisada), y ajustar parámetros adicionales en función de la tarea seleccionada.

En el caso de seleccionar una tarea de clasificación supervisada, se deberá indicar cuál es la variable objetivo. Para las tareas de clasificación no supervisada se podrá indicar en cuántos grupos se quiere segmentar el *dataset*. Este enfoque busca una interfaz de usuario sencilla, donde cualquier tipo de usuario experto en el ERP, sea capaz de generar un modelo de ML en pocos pasos.

El proceso de AutoML recibirá el *dataset*, el tipo de tarea a realizar y los parámetros y en base a ello llevará a cabo las operaciones de pre-procesamiento de datos, selección de modelo o modelos ML a utilizar, *tuning* de hiperparámetros de cada modelo y evaluación de los resultados con el conjunto de datos de prueba. Finalmente el modelo con mejor resultado se guardará y podrá ser utilizado para la clasificación de nuevos registros dentro del ERP.

A modo de prueba y validación se llevará a cabo un caso de uso de cada tipo de clasificación, valorando la facilidad de uso, la calidad de los resultados, tiempos de desarrollo y

[1] https://www.edisa.com/
[2] https://medium.com/@top10erp/7-top-erp-systems-with-artificial-intelligence-features-7dcaeb4ac756

entrenamiento del modelo, y los requisitos hardware necesarios para llevar a cabo todo el proceso.

1.3. Estructura del trabajo

Este trabajo se estructura en cinco secciones principales que describen el desarrollo y evaluación de la integración de soluciones AutoML en el ERP LIBRA. La sección 2 “Contexto y estado del arte” proporciona una revisión de los usos actuales de algoritmos de aprendizaje automático en varios sistemas ERP del mercado, así como un análisis del estado del arte de técnicas de clasificación de machine learning y soluciones de AutoML. Este análisis se centra específicamente en identificar y describir los problemas de clasificación, que constituyen el foco inicial de este estudio.

En la sección 3 “Objetivos concretos y metodología de trabajo” se establecerán los objetivos concretos del estudio y el alcance del mismo, así como la metodología de trabajo adoptada para llevarlo a cabo. Dichos objetivos se van a desarrollar y describir en detalle en la sección 4 “Desarrollo específico de la contribución”.

Por último, en el apartado 5 “Conclusiones y trabajo futuro”, se analizan los resultados obtenidos en los escenarios de pruebas y se detallan las conclusiones del estudio y posibles ampliaciones futuras.

2. Contexto y estado del arte

2.1. Evolución del software ERP

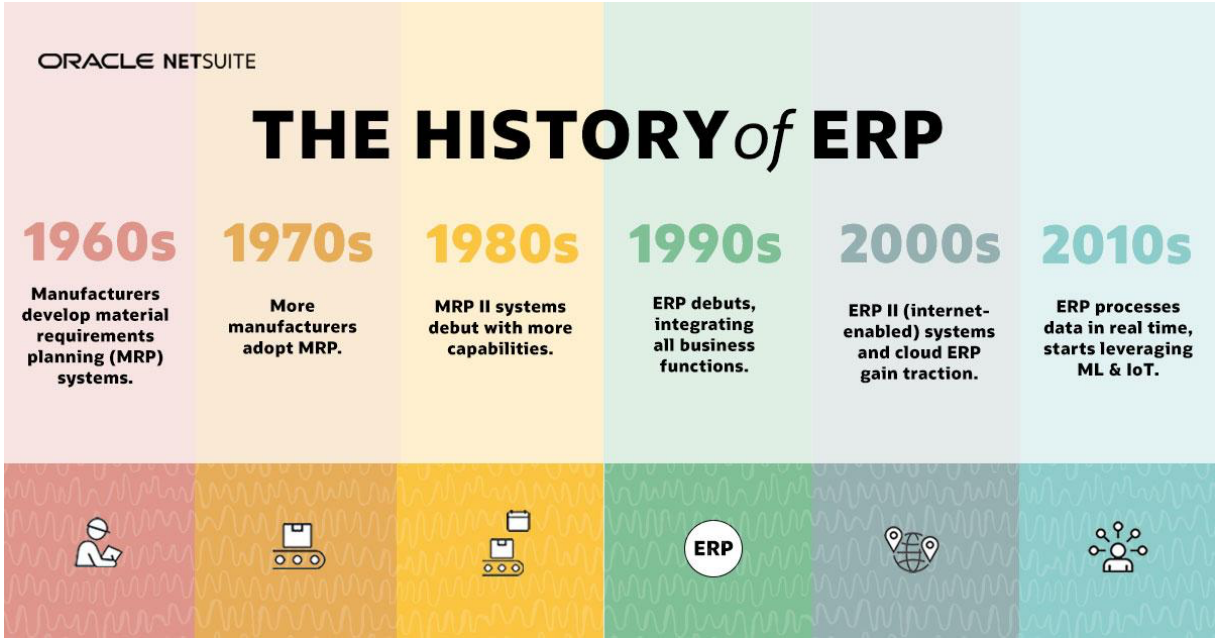
El origen de los ERP procede de los softwares de planificación de materiales de producción (MRP) surgidos en la década de 1960 para tratar la problemática existente en las fábricas respecto a la optimización de su cadena de suministro de materias primas y cálculo de necesidades (Katuu, 2020). Como evolución de estos sistemas surgen los software denominados *Manufacturing Resource Planning* (MRP II) para gestionar la capacidad productiva disponible ajustándola a la planificación de recursos sugerida por el MRP. Estos softwares tuvieron un considerable éxito en empresas productivas de diversas industrias, lo que llevó a la paulatina aparición de diferentes módulos que añadían funcionalidades adicionales a los MRP. En la década de 1990 termina por imponerse la denominación ERP para los sistemas integrados de gestión empresarial, que añadían a la tradicional funcionalidad de la producción, almacenes e ingeniería otras áreas de la empresa como la contabilidad, finanzas, ventas o recursos humanos. A la par que los sistemas ERP surgen una serie de sistemas especializados en diversas áreas de negocio, como pueden ser los CRM (*Customer Relationship Management*) para la gestión de la relación con los clientes, SCM (*Supply Chain Management*) para gestionar la cadena de suministro, BPM (*Business Process Management*) para controlar los flujos de negocio o HCM (*Human Capital Management*) para la gestión de recursos humanos. La tendencia durante las dos últimas décadas ha sido la de integrar todas estas características dentro de los ERP, convirtiéndose en soluciones de software multidimensionales, constituidas por múltiples módulos o funcionalidades, cada una de las cuales atiende una función empresarial diferente (Jawad, 2024).

Los ERP han pasado por tanto a convertirse en una plataforma que unifica gran parte de los procesos empresariales, siendo el sistema que genera y almacena los datos de negocio de forma centralizada e interrelacionada. El siguiente paso en la evolución de los ERP fue expandir su conectividad con otros sistemas, para aprovechar el crecimiento de internet, comercio electrónico e intercambio de información con clientes, proveedores, bancos o entidades gubernamentales. Desde la década de 2010 en adelante, la tendencia ha sido la de convertirse en soluciones software basadas en entorno Cloud y ofrecer accesibilidad desde todo tipo de dispositivos móviles (Ularu, Puican, Suciu, Vulpe, & Todoran, 2013), lo que ha

permitido tanto una reducción de costes de infraestructura como la posibilidad de interactuar con el ERP desde móviles o tabletas, lo cual mejora la interacción en tiempo real con el sistema, permitiendo registrar y consultar los datos de la empresa en tiempo real desde cualquier localización.

En la actualidad la incorporación de ML en los sistemas ERP surge del crecimiento del entorno empresarial, caracterizado por el *big data,* el internet de las cosas (IoT), y la necesidad de tomar decisiones informadas en base a los datos (Wahab & Nor, 2023).

Figura 1: Historia de los ERP



Fuente: www.netsuite.com

2.2. Machine Learning en ERP

El aprendizaje automático es un subconjunto de la inteligencia artificial que permite a los sistemas informáticos aprender sobre un conjunto de datos aportado, sin necesidad de programación explícita. Las técnicas de ML son capaces de procesar grandes volúmenes de datos de forma precisa y rápida, pudiendo conectarse con las fuentes de datos que proporcionan los ERP. También sacan provecho de los avances en potencia de computación, sistemas distribuidos y computación en la nube que permiten construir sistemas complejos y de alta capacidad computacional si así lo requiere el volumen de datos o caso de uso (Jawad, 2024).

Los algoritmos de ML se han convertido en un paradigma que puede transformar cómo se perciben los sistemas ERP, convirtiéndolos en herramientas inteligentes, que apoyen la toma de decisiones en base al conocimiento extraído de los datos y permitan la automatización de procesos o tareas repetitivos, mejorando la productividad de la empresa (M3Systems, 2023).

Las principales empresas desarrolladoras de ERP a nivel mundial como Oracle[3], SAP[4] y Microsoft[5], están integrando de forma progresiva técnicas de ML e inteligencia artificial en sus sistemas para mejorar diversas funcionalidades. Las principales áreas de aplicación para estas tecnologías son el análisis predictivo, análisis de datos, apoyo en la toma de decisiones y automatización de procesos internos del ERP, siendo aplicables en los principales módulos de los ERP (Godbole, 2023).

En la gestión de almacén e inventario son especialmente relevantes los modelos predictivos de demanda, que pueden adaptarse para tener en cuenta múltiples variables como las tendencias de consumo, precios de compra, o tiempos de reabastecimiento. Las técnicas de inteligencia artificial también pueden ayudar a optimizar el espacio del almacén y el envío de mercancía a los clientes (Godbole, 2023) (Qaffas, Ben HajKacem, Nasraoui, & Ben Ncir, 2023).

En cuanto al trato con los clientes de una empresa que utiliza un ERP, podemos encontrar aplicaciones para ML tanto en la gestión de las ventas, relación con los clientes o campañas de marketing. Los modelos de ML se pueden aprovechar tanto de información inherente al ERP como el historial de compras e interacciones de los clientes ya existentes lo que permite la categorización de los mismos, y también pueden obtener información del exterior, por ejemplo en redes sociales, para detectar posibles perfiles de clientes o tendencias de consumo a explotar (Kauppala, 2022).

En el módulo de recursos humanos, se han aplicado técnicas de clasificación automática de aplicaciones de candidatos a ofertas de trabajo, planificación de planes formativos dentro la empresa o ayuda en la redacción y publicación de ofertas laborales (Strang & Sun, 2022).

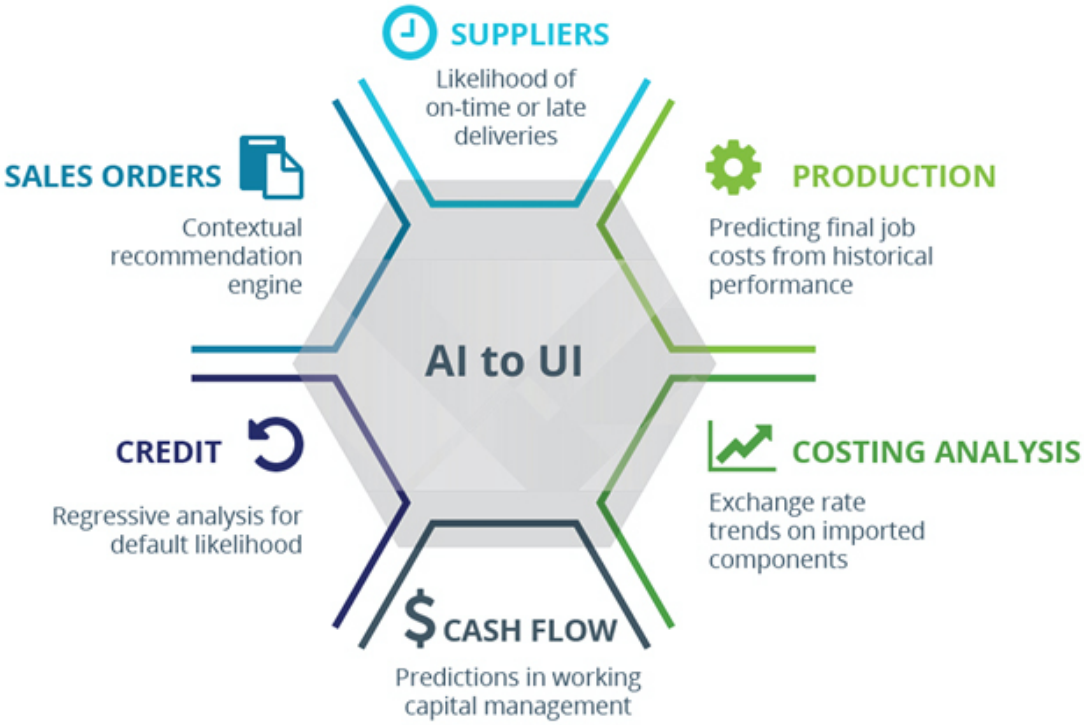
En cuanto a los procesos de planificación de la producción, destaca un conjunto de nuevas tecnologías a aplicar en el ámbito industrial (Industria 4.0), que incluye el uso de sistemas ERP,

[3] https://www.oracle.com/applications/fusion-ai/
[4] https://www.sap.com/spain/products/artificial-intelligence.html
[5] https://www.microsoft.com/es-es/dynamics-365/solutions/ai

IoT, machine learning, automatización y robótica (Stadnicka, y otros, 2022). Con el uso de datos históricos de los procesos productivos de una empresa, se pueden crear modelos para monitorizar los costes de producción presentes, planificación de producción predictiva, detectar ineficiencias en la cadena de producción o prevenir problemas de abastecimiento de materias primas.

El módulo de los sistemas ERP que utilizan todas las empresas independientemente de su área de negocio es el de finanzas y contabilidad. En este ámbito podemos destacar la aplicación de modelos de clasificación de riesgo crediticio, detección de fraude en transacciones económicas o apuntes contables, análisis predictivo aplicado a balances contables o flujo de caja (Josyula & Godbole, 2024).

Figura 2: Aplicaciones de ML por módulo del ERP



Fuente: https://www.syspro.com/product/business-digitalization/artificial-intelligence/

Finalmente, dentro de las técnicas de inteligencia artificial utilizadas en los ERP, se deben destacar las soluciones de visión computacional basadas en inteligencia artificial, por ejemplo para captura y procesamiento de documentos (OCR), recuentos de inventario mediante imágenes o control de calidad mediante detección de anomalías. La tendencia actual pasa por implementar técnicas de inteligencia artificial generativa (GenAI) para aprovechar el auge de

los grandes modelos de lenguaje (LLM), aplicadas en *bots* conversacionales, generación de resúmenes de documentación o apoyo en la generación de textos.

En este estudio nos enfocaremos en algoritmos de machine learning tradicionales debido a su mayor interpretabilidad, eficiencia computacional y facilidad de implementación, características especialmente valiosas para democratizar el uso de ML en el entorno empresarial. Entre todas las técnicas de ML, nos centraremos en las tareas de clasificación que se pueden dividir en supervisada y no supervisada.

2.3. Clasificación supervisada

La clasificación supervisada es una técnica de aprendizaje automático que permite a los sistemas informáticos categorizar un conjunto de datos basándose en ejemplos previos que ya estaban correctamente etiquetados (Soofi & Awan, 2017). En un ERP pueden utilizarse para múltiples propósitos: segmentación de entidades: clientes (según un rating), proveedores (calidad de las compras, entregas y precios), artículos (índice de rotación de inventario) o candidatos a un puesto de trabajo (apto, no apto), clasificación de gastos e ingresos en las cuentas contables apropiadas o incluso para la detección de fraude en transacciones financieras.

Algunos de los algoritmos de clasificación supervisada clásica más importantes son:

- **Árboles de decisión (DT)**. Estructuran las decisiones en formato de árbol, donde cada nodo del árbol se segmenta en base a una condición que busca obtener un subconjunto de datos homogéneo. Los nodos hoja de estos árboles representan las diferentes categorías en las que se segmentarán los datos. Son altamente interpretables, pues permiten conocer en base a qué criterio se ha determinado la categoría de cada registro. Permiten el manejo de variables o *features* tanto numéricas como categóricas (Soofi & Awan, 2017).
- **Random Forest (RF)**. Técnica de *ensemble* que combina múltiples árboles de decisión, cada uno de los cuales ha tenido en cuenta diferentes *features,* para reducir el riesgo de sobreajuste (*overfitting*) y mejorar la precisión de la clasificación. Como desventaja principal frente a los árboles de decisión es que son menos interpretables, al utilizar múltiples árboles de decisión y ponderar los *outputs* de todos ellos (Ding, Huang, Wei, Tang, & Liu, 2021).

- **XGBoost (eXtreme Gradient Boosting)**. Se trata de una implementación optimizada de métodos de *boosting* de gradientes, diseñada para ser altamente eficiente, flexible y escalable. Puede trabajar con datos numéricos, categóricos y tratar datos faltantes. Incorpora técnicas de regularización para prevenir el overfitting. Al igual que Random Forest, sus resultados pueden ser complejos de interpretar al tratarse de un método de *ensemble* (Ren & Wang, 2023).
- **Máquinas de Soporte Vectorial (SVM).** Busca el hiperplano óptimo para la separación de los datos en las diferentes clases posibles, empleando diferentes *kernels* para adaptarse a diferentes conjuntos de datos. Es especialmente útil para clasificación en espacios de alta dimensionalidad o datos no lineales (ElMadany, Alfonse, & Aref, 2021).
- **K Nearest Neighbours (K-NN).** Se busca clasificar los diferentes registros en base a sus *k* vecinos más cercanos, siendo *k* un valor numérico a ajustar para mejorar la precisión del algoritmo (Soofi & Awan, 2017).
- **Naive Bayes**. Técnica estadística basada en el teorema de Bayes. Es efectivo cuando las diferentes variables del conjunto de datos son independientes. Utilizado ampliamente en clasificación de textos (Soofi & Awan, 2017).En este estudio se utilizará un conjunto de datos de ventas y cobros de los clientes de la empresa, para tratar de clasificar las nuevas ventas a los clientes según el riesgo de que el cliente no cumpla con el plazo de pago establecido. Evaluaremos diferentes algoritmos y cómo se integran en soluciones de AutoML, para valorar cuáles son más idóneos para la integración con el ERP LIBRA.

2.4. Clasificación no supervisada.

Los algoritmos de clasificación no supervisada se enfocan en identificar patrones o agrupaciones en un conjunto de datos que no está previamente etiquetado. Al contrario que en la clasificación supervisada, no conocemos a priori las categorías en las que se van a clasificar los registros que componen un determinado *dataset* (Iqbal, Elahi, & Shahzad, 2022). En el ámbito de un ERP pueden utilizarse para las mismas tareas de clasificación o detección de anomalías indicadas en el punto anterior, pero cuando no existe una variable objetivo en el conjunto de datos, es decir, se desconoce el patrón subyacente a los datos y el número de grupos a obtener.

Como algoritmos de clasificación no supervisada de amplio uso tenemos:

- **K-Means**. Tiene como objetivo dividir el dataset en *k* grupos o *clusters*, de manera que cada registro del conjunto de datos pertenezca al grupo con el centroide más cercano. Es importante definir qué valor de *k* es el idóneo, para lo cual existen varios métodos, siendo muy popular el método del codo (*Elbow*). Es efectivo con conjuntos de datos de tamaño moderado, aunque su rendimiento puede disminuir con datasets de alta dimensionalidad (Iqbal, Elahi, & Shahzad, 2022).
- **Clustering jerárquico**. Busca crear una jerarquía de clusters. Existen métodos que parten de los datos individuales y los van combinando creando clusters (aglomerativo o *bottom-up*) y los que parten de un único cluster que se va dividiendo en nuevos clusters más pequeños (divisivo o *top-down*). Este método no requiere establecer de antemano el número de clusters objetivo como K-Means (Sastry, Babu, & Prasada, 2013).
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**. Permite identificar clusters que tienen formas arbitrarias y tiene un buen manejo de outliers. Se deben definir dos parámetros básicos, 'eps' que es la distancia máxima para considerar que dos puntos son vecinos y 'minPts' que indica el mínimo número de puntos que debe tener un cluster. Es efectivo para datos con clusters con una densidad variable y es robusto frente a *outliers* (Iqbal, Elahi, & Shahzad, 2022).

2.5. AutoML

El ML automático o AutoML, se refiere al conjunto de técnicas y procesos que buscan automatizar los pasos que habitualmente siguen los humanos para construir un modelo de ML que se ajuste de forma óptima a un determinado conjunto de datos (Ferreira, Pilastri, Martins, Pires, & Cortez, 2021). En el área de machine learning se suelen seguir una serie de pasos secuenciales que tienen una gran parte de ensayo y error, siendo habitual el realizar varias iteraciones y ajustes en el proceso para lograr el mejor modelo posible para cada caso concreto.

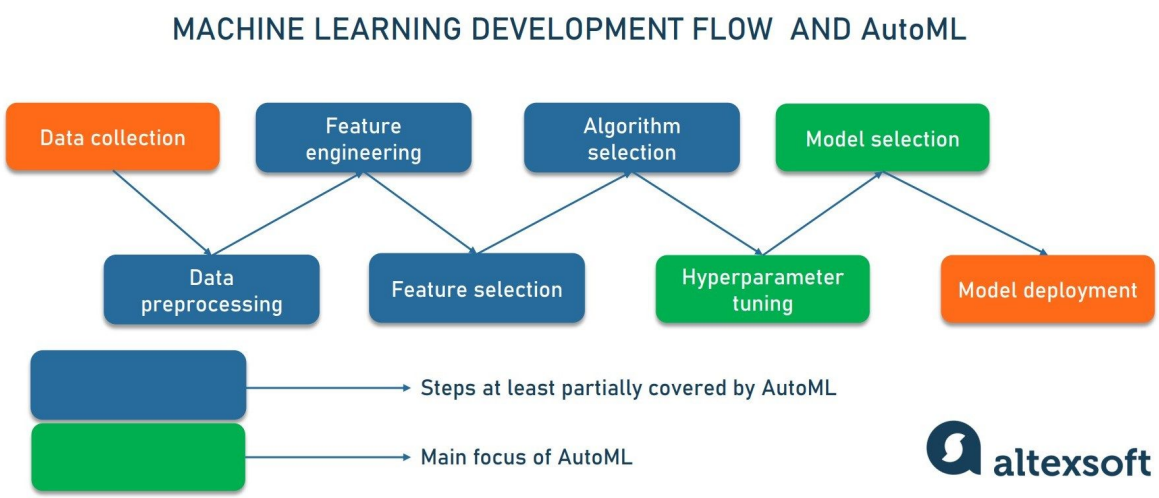
- **Análisis exploratorio de datos (EDA)**. Consiste en hacer un estudio del dataset y aplicar ciertas técnicas, como tratar los valores faltantes, realizar la codificación de variables categóricas, normalizar las variables numéricas o seleccionar las características más

relevantes para el entrenamiento del modelo. Existen librerías que se encargan de realizar un EDA automático sobre el dataset, como DataRobot o H2O.ai (LeDell & Poirier, 2020).

- **Selección de modelos**. Se pueden probar varios modelos diferentes para detectar cuál se ajusta mejor a un conjunto de datos determinado. La experiencia de un profesional del machine learning aporta conocimiento respecto a qué modelos pueden funcionar mejor, acotando el conjunto de modelos a validar. Con técnicas de AutoML, el propio proceso se encargará de probar múltiples modelos hasta encontrar el más adecuado (Waring, Lindvall, & Umeton, 2020).
- **Optimización de hiperparámetros**. Para cada uno de los modelos elegidos, es necesario hacer un ajuste de los hiperparámetros de los que hace uso cada modelo. Este es también un proceso iterativo de ajuste de dichos parámetros hasta encontrar la mejor combinación posible. AutoML ayuda con este proceso empleando técnicas de búsqueda aleatoria (*random search*) o búsqueda en cuadrícula (*grid search*) y otras técnicas específicas de cada librería de AutoML (Real, Liang, So, & Le, 2020).
- **Evaluación de los modelos.** Para decidir qué modelos y qué ajustes son mejores debemos contar con una serie de métricas que nos permitan evaluar el desempeño de cada uno. Para esto se suele dividir el conjunto de datos entre un conjunto de datos de entrenamiento y otro para evaluar el modelo, de forma que podamos usar una parte de los datos originales para medir la precisión del modelo entrenado. Existen múltiples métricas a tener en cuenta según si la tarea a realizar es de regresión o clasificación (LeDell & Poirier, 2020).

Las soluciones de AutoML automatizan este proceso de selección, optimización y evaluación de modelos, realizando múltiples pruebas y seleccionando el modelo con un mejor comportamiento (ver Figura 3).

Figura 3: Flujo de trabajo AutoML



Fuente: https://www.altexsoft.com/blog/automl

Podemos diferenciar entre soluciones empresariales que ofrecen un entorno de desarrollo de machine learning, como Google Cloud AutoML, Oracle AutoML, Azure Machine Learning AutoML, IBM AutoML o AWS SageMaker Autopilot y librerías de código abierto que permiten desarrollar un entorno propio de machine learning automatizado como H2O, AutoGluon, Auto-Sklearn, TPOT o PyCaret (Ferreira, Pilastri, Martins, Pires, & Cortez, 2021).

Figura 4: Comparativa de soluciones AutoML

AutoML PRODUCTS COMPARED

	Classification & regression	Time series forecasting	Neural networks and NAS	NLP	Computer vision
Tech giants					
Google Vertex AI	✓	✓	✓	✓	✓
Amazon SageMaker Autopilot	✓	✓	Limited to one option: MLP	✗	✗
Microsoft Azure AutoML	✓	✓	✓	✓	In preview
IBM Watson AutoAI	✓	✓	✗	✗	✗
End-to-end ML platforms					
H2O Driverless AI	✓	✓	NNs with default values	✓	✓
DataRobot	✓	✓	Automatically pre-tailored NNs	✓	✓
DataBricks	✓	✓	✗	✗	✗
Open source AutoML tools					
AutoGluon (Amazon)	✓	✗	✓	✓	✓
FLAML (Microsoft)	✓	✓	✗	✗	✗
H2O-3 AutoML	✓	✗	✗	✓	✗
Auto-Sklearn	✓	✗	✗	✗	✗
AutoKeras	✓	✓	✓	✓	✓
MLBox	✓	✗	✗	✗	✗

altexsoft

Fuente: https://www.altexsoft.com/blog/automl

2.6. Conclusiones

El futuro de los software ERP pasa por aplicar diferentes técnicas de inteligencia artificial, para ofrecer a las empresas usuarias de estos sistemas una herramienta que les permita ser más productivos y competitivos mediante la explotación de los datos internos nativos del ERP. La integración de técnicas de AutoML en un sistema ERP puede representar un avance sustancial en la democratización del uso de inteligencia artificial en entornos empresariales, especialmente en empresas con recursos limitados, que no cuenten con personal especializado en el tratamiento y explotación de datos mediante machine learning.

3. Objetivos concretos y metodología de trabajo

3.1. Objetivo general

El objetivo principal de este trabajo es desarrollar un cuadro de mando de machine learning automatizado (AutoML) integrado en el ERP LIBRA, que permita la creación de algoritmos de clasificación de manera sencilla y efectiva, de forma que posteriormente se puedan utilizar estos modelos dentro del ERP.

3.2. Objetivos específicos

3.2.1. Selección de algoritmos de ML

Investigar y seleccionar los algoritmos que mejor se comportan con los *datasets* obtenidos de diversas bases de datos del ERP LIBRA en tareas de clasificación supervisada y no supervisada.

Estos *datasets* tendrán la misma estructura de datos en todos los casos, pero variará el contenido de los mismos.

3.2.2. Investigación y selección de librerías AutoML

Evaluar las diferentes opciones de librerías open source existentes de AutoML en combinación con los algoritmos seleccionados previamente y los *datasets* disponibles.

3.2.3. Desarrollo de cuadro de mando AutoML integrado en el ERP LIBRA

Después de seleccionar una de las librerías AutoML e investigar las opciones que ofrece, se desarrollará un cuadro de mando dentro del ERP LIBRA que permita la interacción sencilla con la librería.

El diseño de los *datasets* a utilizar se realizará con herramientas ya existentes dentro del ERP, que cuenta con diseñador de informes que permite la realización de consultas SQL complejas sobre la base de datos del ERP.

3.2.4. Evaluación del resultado

Se realizarán pruebas con dos casos de uso específicos:

- Clasificación de facturas en función del riesgo de impuntualidad en el pago, utilizando algoritmos de clasificación supervisada.

- Clasificación de productos según rotación y espacio que ocupan en el inventario, empleando algoritmos de clasificación no supervisada.

Evaluaremos la facilidad de uso, calidad de los resultados, tiempos de desarrollo y entrenamiento de los modelos y requisitos de hardware necesarios para el despliegue en el entorno del ERP.

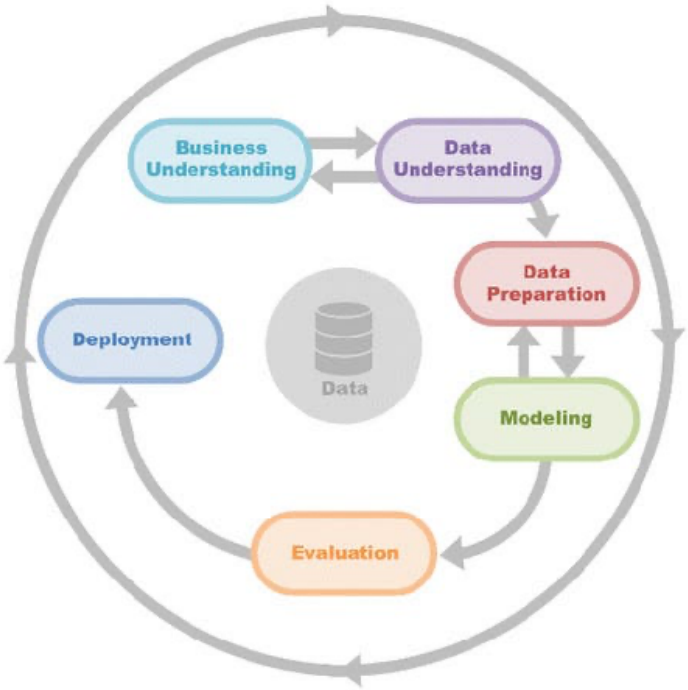
3.3. Metodología del trabajo

Para alcanzar los objetivos propuestos, se adoptará una combinación de la metodología CRISP-DM con principios de desarrollo AGILE para adaptarse a las necesidades específicas de un desarrollo de machine learning que también engloba un apartado de desarrollo de software tradicional (Gerhart, Torres, & Giddens, 2023). CRISP-DM se basa en cuatro pasos (ver Figura 5):

- Comprensión del negocio. Definir los objetivos junto a expertos en el dominio y *stakeholders*.
- Entendimiento de los datos. Obtener, explorar y catalogar los datos disponibles.
- Preparación de los datos. Limpieza y depuración de los datos disponibles para su preparación para su utilización posterior en el entrenamiento de modelos.
- Modelado de datos. Aplicación de uno o varios algoritmos sobre los datos ya preparados.

CRISP-DM se ajusta mejor que otras metodologías a los proyectos de machine learning dada su orientación a los datos.

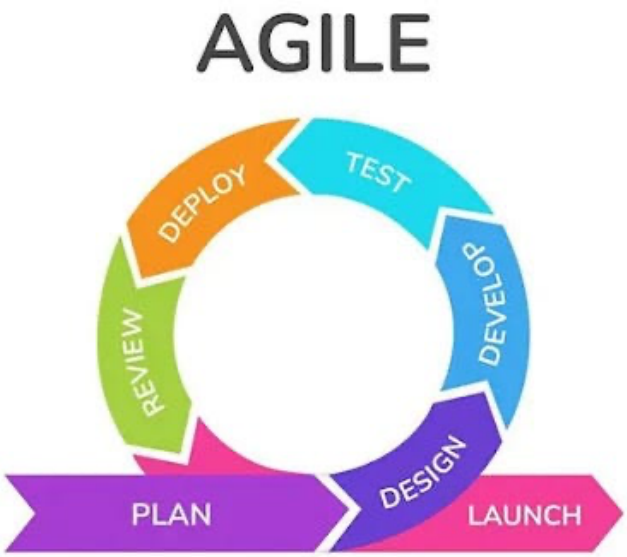
Figura 5: Etapas metodología CRISP-DM



Fuente: Medium.com

Como en este proyecto también se abordan otro tipo de tareas, emplearemos también parte de la metodología AGILE para definir *sprints* en los que abordar las tareas necesarias para el cuadro de mando y su integración en LIBRA de manera iterativa.

Figura 6: Flujo de trabajo AGILE



Fuente: Medium.com

Se estructurará el desarrollo en varias fases iterativas:

1. **Estudio y comprensión de las necesidades propias de un ERP**. Establecer junto a personal de la empresa desarrolladora del ERP LIBRA los objetivos, entregables y criterios de éxito del proyecto.
2. **Preparación y obtención de los *datasets***. Creación de la estructura de los datasets para los dos casos de uso a estudiar. Una vez definidos, se obtendrán N datasets de diferentes bases de datos. Los datos deberán estar total anonimizados para descartar cualquier posibilidad de tratar datos reales de alguna entidad.
3. **Evaluación de diferentes modelos de ML**. En primer lugar se realizará un entrenamiento con los *datasets* obtenidos para seleccionar qué modelos se comportan mejor con las estructuras de datos obtenidas del ERP.
4. **Investigación y pruebas de librerías AutoML.** Investigación de las diferentes librerías open source disponibles y su adecuación a los datasets obtenidos y algoritmos seleccionados. Se realizarán pruebas con una selección de las librerías que se consideren más adecuadas, para evaluar su desempeño.
5. **Desarrollo del cuadro de mando integrado en el ERP LIBRA.** Se creará una interfaz desde la cual poder ejecutar el entrenamiento de un modelo de ML que haga uso de un dataset seleccionado por el usuario e invoque a la librería de AutoML. En este punto se determinará qué opciones de configuración se le ofrecerá a los usuarios, por ejemplo la posibilidad de establecer ciertos hiperparámetros o un tiempo máximo determinado para el cálculo del modelo. Esta sección es la más específica de desarrollo de software tradicional, se abordará mediante *sprints* de duración semanal.
6. **Despliegue y evaluación de resultados.** Se desplegará el cuadro de mando en un entorno controlado y se ejecutará con los *datasets* de dicha base de datos en concreto, para evaluar tanto la facilidad de uso, tiempos de entrenamiento y calidad de los resultados obtenidos.

4. Desarrollo específico de la contribución

4.1.Conjuntos de datos

En colaboración con EDISA y ciertos clientes del ERP LIBRA, se ha procedido al diseño de dos estructuras de datos diferentes, una para el caso de clasificación supervisada de facturas de venta y otro para emplear en tareas de clustering (clasificación no supervisada) para la categorización de productos con la finalidad de optimizar la organización del almacén.

Para cada tarea se han obtenido cinco datasets diferentes, cada uno de un cliente diferente del ERP correspondientes a sectores empresariales distintos: exportación, alimentación, distribución farmacéutica, industria y retail.

Para la extracción de los datos se ha contado con acceso total a las diferentes bases de datos del ERP de estos clientes, que en todos los casos se trata de bases de datos Oracle Enterprise. Como herramienta para la conexión a dichas bases de datos se ha utilizado PLSQL Developer.

4.1.1. Conjunto de datos para clasificación supervisada

Se ha diseñado un proceso almacenado de base de datos para el cálculo y obtención de este conjunto de datos, pues ha sido necesario realizar un preprocesamiento de los datos nativos del ERP para obtener las variables deseadas para la categorización supervisada de los datos ya existentes.

La variable objetivo de este *dataset* es la categoría de calidad de pago de una factura, que puede tomar cinco valores diferentes:

Valor numérico	Etiqueta	Descripción
-2	Riesgo alto	Alta probabilidad de impago total.
-1	Riesgo medio	Cobros fuera de plazo o impagos parciales
0	Riesgo moderado	Cobros fuera de plazo pero recibidos
1	Riesgo bajo	Cobradas pero con probabilidad de retrasos en el pago
2	Riesgo mínimo	Cobradas totalmente dentro de las fechas acordadas

Tabla 1. Clasificación supervisada, variable objetivo

Para el cálculo de esta categoría se han tenido en cuenta varios aspectos:

- El importe cobrado total de la factura.
- La fecha en la que se realizaron los cobros de cada factura respecto a los acuerdos comerciales.
- Si la factura se divide en varios cobros con fechas o métodos de cobro diferentes, se tiene en cuenta lo anterior para cada uno de los cobros.

El conjunto de datos está compuesto de las siguientes variables:

Variable	**Naturaleza**	**Descripción**
categoria_factura	Numérico	Variable objetivo (Tabla 1)
empresa	Categórico	Código de la empresa que realiza la factura
ejercicio	Numérico	Año de la factura
numero_serie	Categórico	Serie de facturación
numero_factura	Numérico	Número de factura
tipo_factura	Categórico	Tipo de factura. Clasificación arbitraria según la configuración del ERP.
organizacion_comercial	Categórico	Tienda que emite la factura
cliente	Categórico	Código del cliente que recibe la factura
fecha_factura	Fecha	Fecha de la factura en formato DD/MM/YYYY
mes_factura	Categórico	Mes de la factura
divisa	Categórico	Código de la divisa en la que se emite la factura, por ejemplo EUR (Euros).
forma_cobro	Categórico	Codificación del tipo de acuerdo para el cobro de la factura. Por ejemplo: 30, 60 y 90 días desde la fecha factura.
centro_contable	Categórico	Centro contable que emite la factura
liquido_factura	Numérico	Importe total de la factura
albaran_factura	Categórico	Indica si la factura tiene su origen en un tipo de venta directo denominado albarán/factura
diario	Categórico	Diario contable de la factura
envio_electronico	Categórico	Indica si la factura ha sido enviada electrónicamente a través de algún servicio para este fin.
fecha_contabilizacion	Fecha	Fecha de contabilización en formato DD/MM/YYYY

numero_asiento	Numérico	Número del asiento contable de la factura en el ERP
cuenta_contable	Numérico	Código de la cuenta contable de la factura
tiene_descuento	Categórico	Indica si la factura tiene algún descuento aplicado o no
count_tipo_transaccion	Numérico	Indica el número de tipos de pago diferentes acordados para esta factura.

Tabla 2. Clasificación supervisada, descripción de variables

Sector	**Registros *dataset***
Alimentación	296822
Retail	121505
Distribución farmacéutica	560932
Exportación	226883
Industrial	79300

Tabla 3. Información datasets clasificación supervisada por sector

4.1.2. Conjunto de datos para clasificación no supervisada

En este caso también ha sido necesario diseñar y realizar un proceso para el cálculo del conjunto de datos. Se han procesado los artículos gestionados por una empresa para obtener datos de cada artículo, sus movimientos de almacén e importe de ventas durante el último año.

El objetivo es realizar una clasificación de los artículos para la optimización de los procesos de inventario, localizando aquellos con un impacto más relevante ya sea sobre las ventas o procesos logísticos.

Variable	**Naturaleza**	**Descripción**
empresa	Categórica	Código de la empresa
articulo	Categórica	Código del artículo
tipo_material	Categórica	Tipo de material del artículo
codigo_familia	Categórica	Familia a la que pertenece el artículo

volumen	Numérica	Volumen de ocupación por cada unidad del artículo
multipresentacion	Numérica	Número de presentaciones (cajas, palets, unidades, etc) en las que se trabaja con el artículo en el almacén
stock_actual	Numérica	Cantidad actual del artículo en el almacén
total_cantidad_movs	Numérica	Total de movimientos de almacén (entradas y salidas) durante el último año.
pmp	Numérica	Precio medio ponderado del artículo
total_importe_ventas	Numérica	Importe facturado del artículo en el último año
total_movs	Numérica	Número total de ventas del artículo en el último año

Tabla 4. Clasificación no supervisada, descripción de variables

Sector	**Registros *dataset***
Alimentación	8795
Retail	6683
Distribución farmacéutica	229
Exportación	1032
Industrial	43852

Tabla 5. Información datasets clasificación no supervisada por sector

4.2. Entrenamiento manual de modelos

Como primera fase del estudio, se ha procedido a realizar un análisis de cada uno de los *datasets* para cada una de las tareas objetivo.

La intención es comprobar la efectividad del entrenamiento manual con diferentes algoritmos de machine learning para comparar posteriormente estos resultados con los obtenidos con librerías de AutoML.

Se ha optado por trabajar en este punto con Jupyter Notebook en un entorno de desarrollo Python 3.10. En todos los casos se ha realizado el entrenamiento en un equipo MacBook Pro M3 Pro con 36 GB de RAM y la configuración Metal activada para aprovechar GPU en los casos donde esto es posible.

4.2.1. Clasificación supervisada manual

Para los cinco *datasets* se ha realizado un entrenamiento de diferentes algoritmos y técnicas de aprendizaje automático, haciendo uso de las librerías *Scikit-Learn* y *XGBoost* para los modelos, *pandas* para la carga y tratamiento de datos y *matplotlib* y *seaborn* para visualización de resultados.

Los pasos seguidos para el entrenamiento manual han sido los siguientes:

1. **Eliminar columnas que no aportan valor.** Se han identificado en todos los datasets que se podían eliminar las variables numero_factura, numero_asiento, tipo_factura, diario, cuenta_contable. Estas variables o bien tenían un valor único por cada registro, o bien tenían un único valor para todo el dataset.
2. **Tratamiento de valores nulos**. Se ha verificado que no existían valores nulos en el dataset, aunque esto venía garantizado por el procesamiento realizado en origen para la obtención del mismo.
3. **Conversión de variables categóricas a numéricas usando label encoding**. Se ha optado por asignar una etiqueta (label) a cada valor de las columnas categóricas para su tratamiento, de forma que posteriormente se puede volver a convertir la etiqueta al valor original.
4. **Sobre muestreo de clases minoritarias.** Se ha empleado SMOTE (Synthetic Minority Over-sampling Technique) para ajustar el dataset y compensar el volumen de datos en las categorías de facturas con riesgo alto y medio, que originalmente tenían un volumen de datos muy inferior al resto de categorías.
5. **Estandarización de características**. Se estandarizaron las características para tener una media de 0 y una desviación estándar de 1, lo que mejora el rendimiento de muchos algoritmos de machine learning.
6. **Separación en datos de entrenamiento y prueba**. Se dividió el dataset en un 80% para datos de entrenamiento y un 20% para datos de prueba, asegurando una correcta validación del modelo.
7. **Creación de un grid de hiperparámetros para cada modelo a entrenar**. Se utilizó la técnica GridSearch para buscar la optimización de los hiperparámetros de cada modelo. Se impuso una limitación de 120 minutos de entrenamiento para cada modelo, ajustando el número de combinaciones para no superar este umbral.

8. **Entrenamiento de los modelos**. Se han entrenado modelos de árboles de decisión, RandomForest, XGBoost, K-Nearest Neighbors y SVM. Adicionalmente, se ha implementado un modelo de Stacking, que combina los modelos anteriores y aplica una regresión logística como estimador final. En todos los casos se ha utilizado la técnica de validación cruzada *StatrifiedKFold*, buscando evitar problemas de sobreajuste. Esta técnica se asegura de que cada fold tenga aproximadamente la misma proporción de muestras de cada clase respecto a la muestra original.
9. **Evaluación de los modelos.** Se han utilizado las métricas de exactitud (accuracy), precisión, recall y f1-score. Adicionalmente se ha obtenido la matriz de confusión en cada caso para evaluar el comportamiento del modelo con cada categoría de la variable objetivo.

4.2.1.1. Configuración de los modelos

Se detallan a continuación los hiperparámetros y sus valores escogidos para la configuración de GridSearch:

- **Árboles de decisión**:
    - criterion. Especifica la función de evaluación utilizada para medir la calidad de una división.
        - Valores: gini, entropy.
    - max_depth. Determina la profundidad máxima del árbol.
        - Valores: None, 20, 50.
    - min_samples_split. Número mínimo de muestras requeridas para dividir un nodo.
        - Valores: 2, 5, 10.
- **RandomForest**:
    - n_estimators. Número de árboles del bosque.
        - Valores: 100, 200.
    - max_features. Número de características a considerar al buscar la mejor división.
        - Valores: sqtr, log2.
    - max_depth. Profundidad máxima de los árboles individuales.
        - Valores: None, 20, 40.

  - criterion. Especifica la función de evaluación utilizada para medir la calidad de una división.
    - Valores: gini, entropy.

- **XGBoost**:
  - n_estimators. Número de árboles de refuerzo a construir.
    - Valores: 50, 100, 200.
  - max_depth. Profundidad máxima de un árbol
    - Valores: 3, 6, 9.
  - learning_rate. Paso de reducción de la contribución de cada árbol.
    - Valores: 0.01, 0.1, 0.2.
  - subsample. Proporción de muestras utilizadas para cada árbol.
    - Valores: 0.6, 0.8, 1.0.

- **K-NN**:
  - n_neighbors. Número de vecinos a utilizar para la clasificación.
    - Valores: 3, 5, 7, 9, 11.
  - weights. Función de paso utilizada en la predicción.
    - Valores: uniform, distance.
  - metric. Métrica de distancia a utilizar.
    - Valores: euclidean, manhattan, minkowski.

- **SVM**. Por motivo del límite en los tiempos de ejecución, se ha optado por no utilizar GridSearch, ajustando los hiperparámetros manualmente. Se han empleado en este caso los valores de:
  - kernel. Tipo de kernel a utilizar.
    - Valor: rbf (Radial Basis Function).
  - C. Parámetro de regularización.
    - Valor: 10.
  - gamma. Coeficiente para el kernel.
    - Valor: 0.1.

4.2.1.2. Resultados

Se muestran en la Tabla 6 los resultados obtenidos para cada uno de los cinco *datasets* disponibles. En la columna *Fits* se indica el número de iteraciones, resultado de multiplicar la

combinación de hiperparámetros para cada modelo por el número de folds empleado para la validación cruzada (5 folds en todos los casos).

Dataset	Modelo	Fits	Tiempo (min)	Exactitud	Precisión	Recall	F1-score
Industrial	Árboles de decisión	90	2	0,88	0,88	0,88	0,88
	RandomForest	120	51	0,92	0,92	0,92	0,92
	XGBoost	405	17	0,93	0,93	0,93	0,93
	K-NN	150	11	0,86	0,86	0,86	0,86
	SVM	1	102	0,84	0,84	0,84	0,84
	Stacking	1	90	0,92	0,92	0,92	0,92
Retail	Árboles de decisión	90	4	0,96	0,96	0,96	0,96
	RandomForest	120	130	0,98	0,98	0,98	0,98
	XGBoost	405	165	0,98	0,98	0,98	0,98
	K-NN	150	40	0,96	0,96	0,96	0,96
	SVM	1	65	0,95	0,95	0,95	0,95
	Stacking	1	230	0,98	0,98	0,98	0,98
Exportación	Árboles de decisión	90	3	0,93	0,93	0,93	0,93
	RandomForest	120	73	0,95	0,95	0,95	0,95
	XGBoost	405	26	0,97	0,97	0,97	0,97
	K-NN	150	35	0,91	0,91	0,91	0,91
	SVM	1	60	0,91	0,91	0,91	0,91
	Stacking	1	227	0,96	0,96	0,96	0,96
Distribución Farmacéutica	Árboles de decisión	90	8	0,96	0,93	0,93	0,93
	RandomForest	120	250	0,97	0,97	0,93	0,94
	XGBoost	405	82	0,98	0,98	0,97	0,98
	K-NN	150	149	0,94	0,91	0,87	0,89
	SVM	1	512	0,93	0,91	0,86	0,88
	Stacking	1	1234	0,98	0,97	0,96	0,97
Alimentación	Árboles de decisión	90	6	0,87	0,87	0,87	0,87
	RandomForest	120	193	0,91	0,91	0,91	0,91
	XGBoost	405	63	0,89	0,89	0,89	0,89
	K-NN	150	121	0,88	0,88	0,88	0,88
	SVM	1	463	0,84	0,84	0,84	0,84
	Stacking	1	1700	0,91	0,91	0,91	0,91

Tabla 6. Resultados entrenamiento manual clasificación supervisada

En cuanto a los resultados, se puede observar un valor elevado en todas las métricas, que en general toman valores muy similares o incluso iguales para el mismo dataset y modelo. Esto último puede deberse a tener un dataset muy balanceado debido al uso de SMOTE.

En base a estos resultados podríamos concluir lo siguiente:

- **Árboles de decisión**. Ofrecen un buen equilibrio entre tiempo de ejecución y rendimiento, manteniendo métricas en torno a la media de los modelos. Son recomendables si se prioriza el tiempo de ejecución.
- **RandomForest y XGBoost**. Alcanzan las mejores métricas en general, siendo recomendables si se busca mayor porcentaje de precisión, a pesar de requerir más tiempo de ejecución.
- **K-NN**. Muestra tiempos de ejecución inferiores a la media, pero puntúa ligeramente por debajo de los árboles de decisión en cuanto al resto de métricas.
- **SVM**. Debido a su alto requerimiento computacional, no se han realizado tantas iteraciones para el ajuste de hiperparámetros. Por este motivo, no destaca en las métricas de precisión.
- **Stacking**. A pesar de requerir mucho tiempo de entrenamiento, no mejora los resultados que podemos obtener con XGBoost o RandomForest.

Modelo	**Exactitud promedia**	**Tiempo promedio (min.)**
Árboles de decisión	0,92	4,6
RandomForest	0,95	130,4
XGBoost	0,95	70,6
K-NN	0,91	71,2
SVM	0,89	240
Stacking	0,95	696,2

Tabla 7. Rendimiento promedio

En la Figura 7 se muestra la matriz de confusión unificada del modelo Stacking para los cinco datasets, dado que en general muestra una distribución similar en todos ellos.

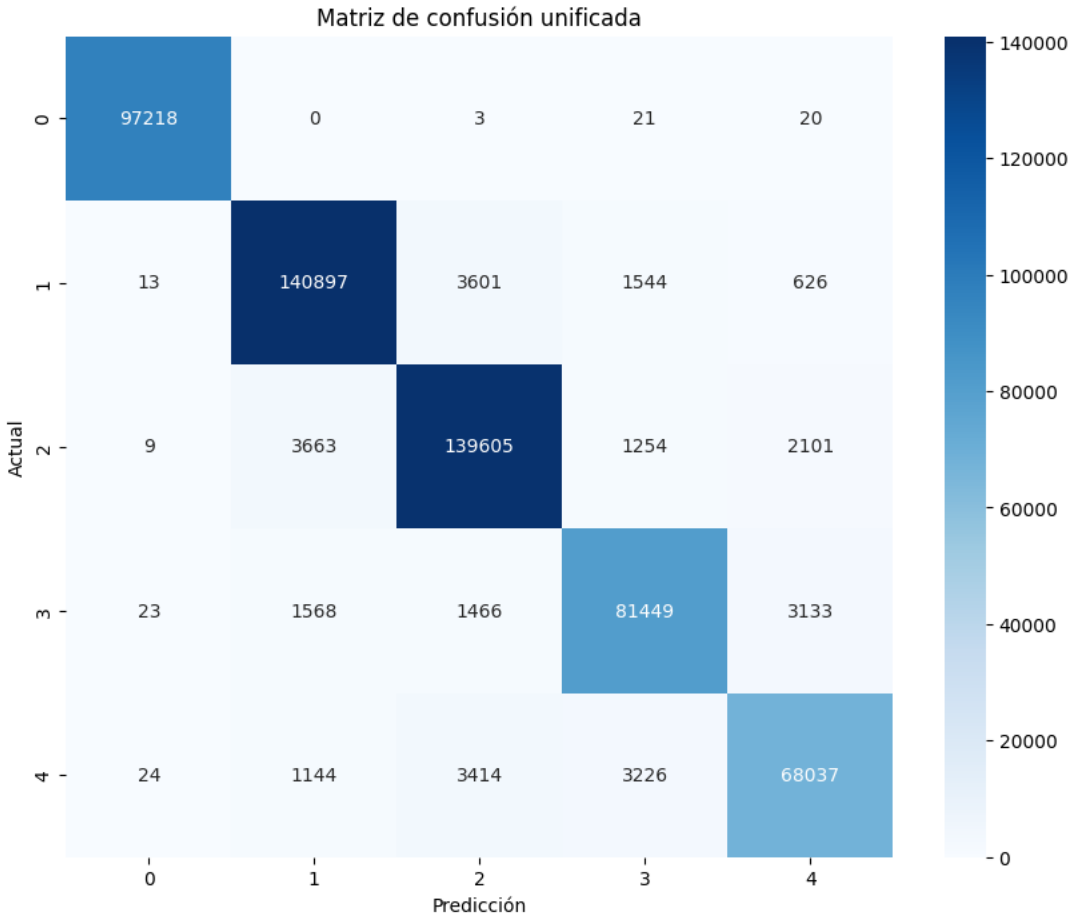


Figura 7. Matriz de confusión unificada

4.2.2. Clasificación no supervisada manual

En el caso de uso de clasificación no supervisada, se busca obtener una división en grupos o clusters de los diferentes artículos. Como en el caso anterior, se hará uso de la librería Scikit-Learn para los modelos de aprendizaje, pandas y numpy para la carga y procesamiento de datos y matplotlib y seaborn para la visualización de resultados.

Los pasos seguidos en este caso han sido:

1. **Eliminar variables con valores únicos**. En cada dataset se encuentran diferentes variables que tienen un único valor, por lo que se eliminan por no aportar información.
2. **Conversión de variables categóricas a numéricas**. Se hace uso de label encoding.
3. **Estandarización de variables**.

4. **IQR para detección y eliminación de outliers**. Se han eliminado los valores atípicos en los datasets aplicando IQR (Interquartile Range), considerando como valor atípico el que esté fuera del rango marcado por los siguientes límites:
    a. Valor IQR: Q3 – Q1
    b. Límite inferior: Q1 – 1.5 x IQR
    c. Límite superior: Q3 + 1.5 x IQR
5. **Estudio de la matriz de correlación**. Se determina si hay alguna variable adicional que se deba eliminar.
6. **Entrenamiento de modelos.** Se han seleccionado los modelos KMeans, DBSCAN y AgglomerativeClustering. En los casos de KMeans y AgglomerativeClustering se ha establecido en 3 el objetivo de clústeres. Para DBSCAN, se ha realizado una búsqueda en grid para obtener la mejor combinación de hiperparámetros para obtener dicho número de clústeres.
7. **Estudio de los resultados**. Se observa el tamaño de cada clúster y el impacto de cada variable sobre cada uno de los clústeres. Se aplica PCA (Análisis de componentes principales) para reducir la dimensionalidad a 2 dimensiones y visualizar los clústeres resultantes.
8. **Evaluación de los modelos.** Se han utilizado las siguiente métricas de evaluación.
    a. **Silhouette score**. Mide la similitud de los puntos dentro de un clúster comparados con los puntos de otros clústeres. Cuanto más cercano a 1
    b. **Davies-Bouldin índex**. Promedio de las razones entre las distancias dentro de los clústeres y las distancias entre los clústeres.
    c. **Inercia**. Únicamente aplica en KMeans. Refleja la suma de las distancias cuadradas de los puntos a los centroides más cercanos. Un valor bajo indica una mejor compactación de los clústeres.

4.2.2.1. Configuración de los modelos

- **KMeans.** Se han empleado los siguientes hiperparámetros:
    - **n_clusters**. Número de clústeres a formar.
        - **Valor**. 3.
    - **init**. Método de inicialización de los centroides.
        - **Valor**. k-means++.

    - **n_init**. Número de inicializaciones diferentes. Se ejecutará este número de veces, seleccionando la mejor solución basada en la inercia (suma de las distancias cuadradas dentro del clúster).
        - **Valor**. 10.
- **Agglomerative Clustering**.
    - **n_clusters**.
        - **Valor**. 3.
    - **linkage**. Método para calcular la distancia entre conjuntos de observaciones.
        - **Valor**. ward.
- **DBSCAN**. Como no se puede establecer directamente el número de clústeres objetivo, se ha realizado una búsqueda en grid ajustando los hiperparámetros para buscar una combinación que se ajustes al objetivo de 3 clústeres. Los valores obtenidos han variado para cada uno de los conjuntos de datos.
    - **eps**. La distancia máxima entre dos muestras para que una sea considerada como vecina de la otra.
    - **min_samples**. Es el número mínimo de muestras en un vecindario para que un punto sea considerado como un punto central.

4.2.2.2. Resultados

En este caso el rendimiento computacional no ha sido objeto de estudio pues el entrenamiento en ningún caso ha superado los 5 minutos, probablemente debido al tamaño más reducido de estos conjuntos de datos (Tabla 5).

La columna Inercia únicamente toma valor para el modelo KMeans.

La columna Hiperparámetros únicamente toma valor para el modelo DBSCAN, para mostrar los valores tomados para lograr separar cada dataset en 3 clústeres.

Dataset	Modelo	Hiperparámetros	Silhouette	Davies-Bouldin	Inercia
Industrial	Kmeans		0,9102	0,8197	160192,6
	Agglomerative Clustering		0,9105	0,8309	
	DBSCAN	eps=0.8, min_samples=35	0,9174	1,5145	
Retail	Kmeans		0,776	0,2242	26631,26
	Agglomerative Clustering		0,7768	0,9771	

	DBSCAN	eps=0.7, min_samples=9	0,7838	1,8813	
Exportación	Kmeans		0,5983	0,9836	3804,3
	Agglomerative Clustering		0,6294	1,1285	
	DBSCAN	eps=0.8, min_samples=3	0,489	1,197	
Distribución Farmacéutica	Kmeans		0,6991	0,9275	697,24
	Agglomerative Clustering		0,7073	1,0041	
	DBSCAN	eps=0.1, min_samples=4	0,3258	1,6067	
Alimentación	Kmeans		0,6038	0,9192	33810,56
	Agglomerative Clustering		0,589	1,1655	
	DBSCAN	eps=0.2, min_samples=8	0,5819	1,361	

Tabla 8. Resultados modelos clasificación no supervisada

Modelo	Silhouette	Davies-Bouldin	Innertia
Kmeans	0,71748	0,77484	45027,192
Agglomerative Clustering	0,7226	1,02122	
DBSCAN	0,61958	1,5121	

Tabla 9. Promedios por modelo

1. **KMeans.**

Se observa en general un buen rendimiento con valores *Silhouette Score* relativamente altos y *Davies-Bouldin Index* bajos. Esto indica que los clústeres formados son compactos y están bien separados. La inercia varía significativamente entre cada dataset, lo que refleja la variabilidad en la densidad y distribución de los datos.

2. **Agglomerative Clustering.**

Tiene el mejor valor para Silhouette aunque muy similar a KMeans. Sin embargo tiene un valor superior para el índice Davies-Bouldin, por lo que podría interpretarse que los clústeres formados no están separados de forma tan clara como en KMeans.

3. **DBSCAN**

Obtiene los peores resultados en ambas métricas. Además, es más complicado de implementar en este caso de uso concreto donde se requiere un número fijo de clústeres. Este algoritmo es más adecuado para datasets con clústeres de formas arbitrarias y presencia de ruido, por lo que puede ser más útil en otros escenarios.

Además de las métricas, para la implementación de estos algoritmos es especialmente relevante el estudio de los clústeres que ha formado. Se debe revisar tanto la distribución de los clústeres y la importancia de cada variable en cada clúster.

Para esto se han visualizado diversos gráficos como los indicados a continuación para el dataset de distribución farmacéutica.

Se muestra una distribución de la importancia de cada variable en cada uno de los clústeres (Figura 8). También una gráfica del tamaño de cada clúster (Figura 10).

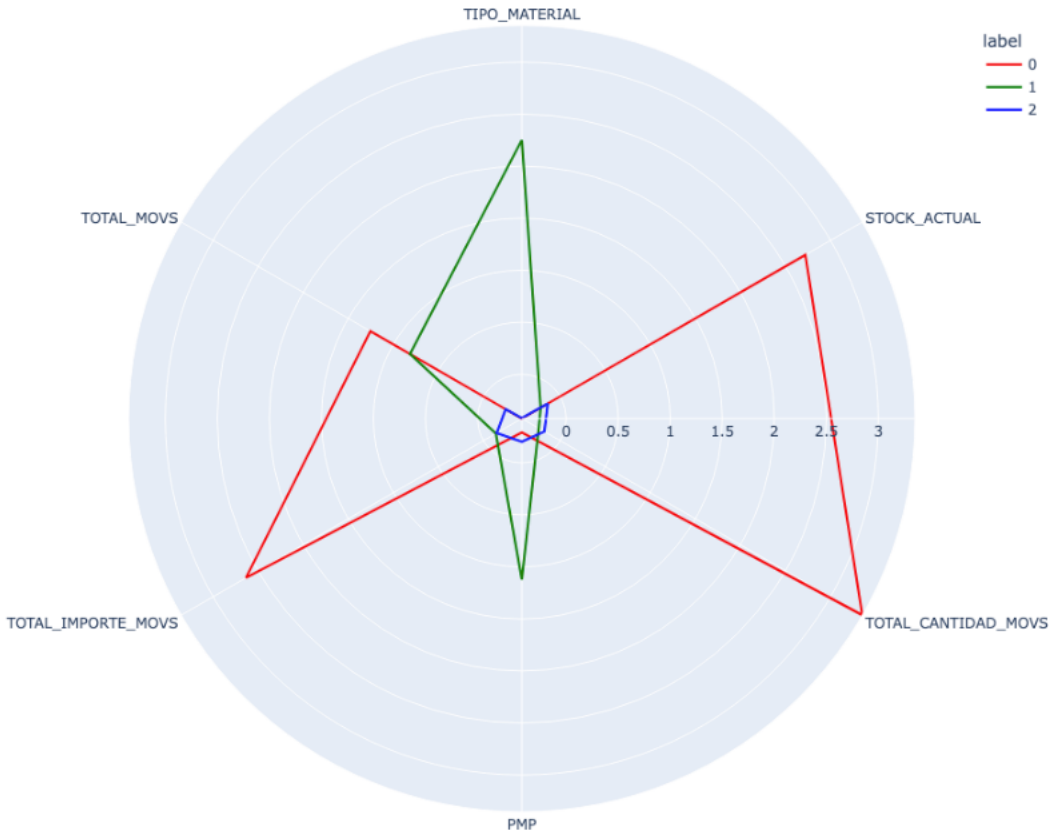


Figura 8. Gráfico polar Agglomerative Clustering

Se puede observar en este caso con Agglomerative Clustering, que el clúster 0 tiene los artículos con más movimientos de almacén y que más impacto tienen en el almacén. El clúster 1 sería el siguiente, con artículos con menos movimientos pero con un valor alto (PMP), mientras que el clúster 2 tiene al resto de artículos con valores bajos en general para todas las características. El gráfico polar con KMeans es prácticamente idéntico (Figura 9).

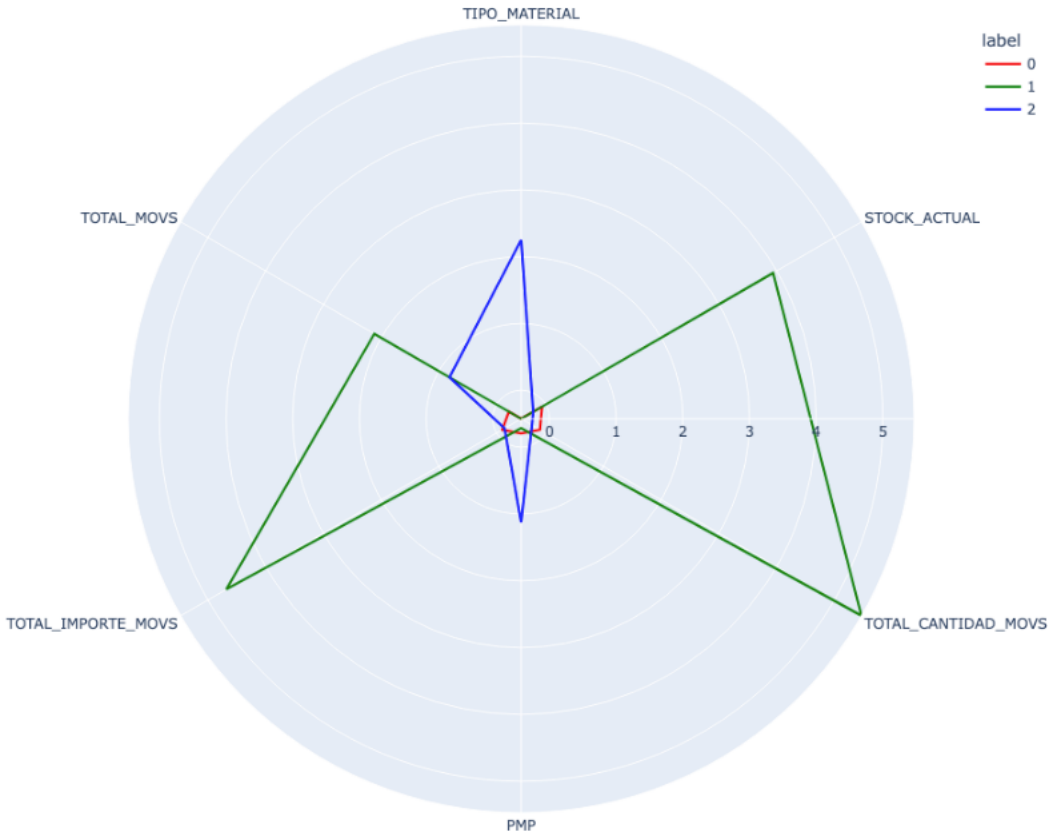


Figura 9. Gráfico polar KMeans

Los gráficos de tarta de KMeans y Agglomerative Clustering también son similares. El clúster de tamaño intermedio es exactamente igual, pero en KMeans toma mayor predominancia el clúster de mayor tamaño respecto al de menor tamaño.

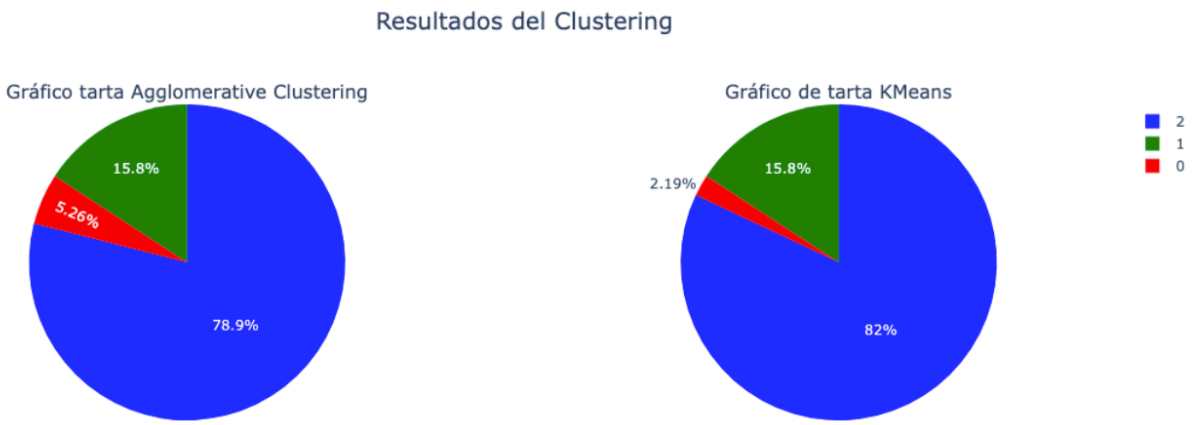


Figura 10. Gráfico de tarta para distribución de clústeres

El tamaño de los clústeres es muy variable, se puede observar que más del 78% de los artículos pertenecen al clúster 2 que corresponde a los artículos con pocos movimientos de almacén. Esto sugiere que esta empresa tiene un gran volumen de artículos con un impacto relativamente bajo en el almacén respecto a los otros clústeres. Podemos observar un comportamiento similar en las mismas gráficas del mismo dataset con Agglomerative Clustering.

Respecto a DBSCAN, mostramos en la Figura 11 la influencia de cada variable en cada uno de los clústeres, donde se aprecia que el PMP es la variable que más afecta para separar los tres clústeres.

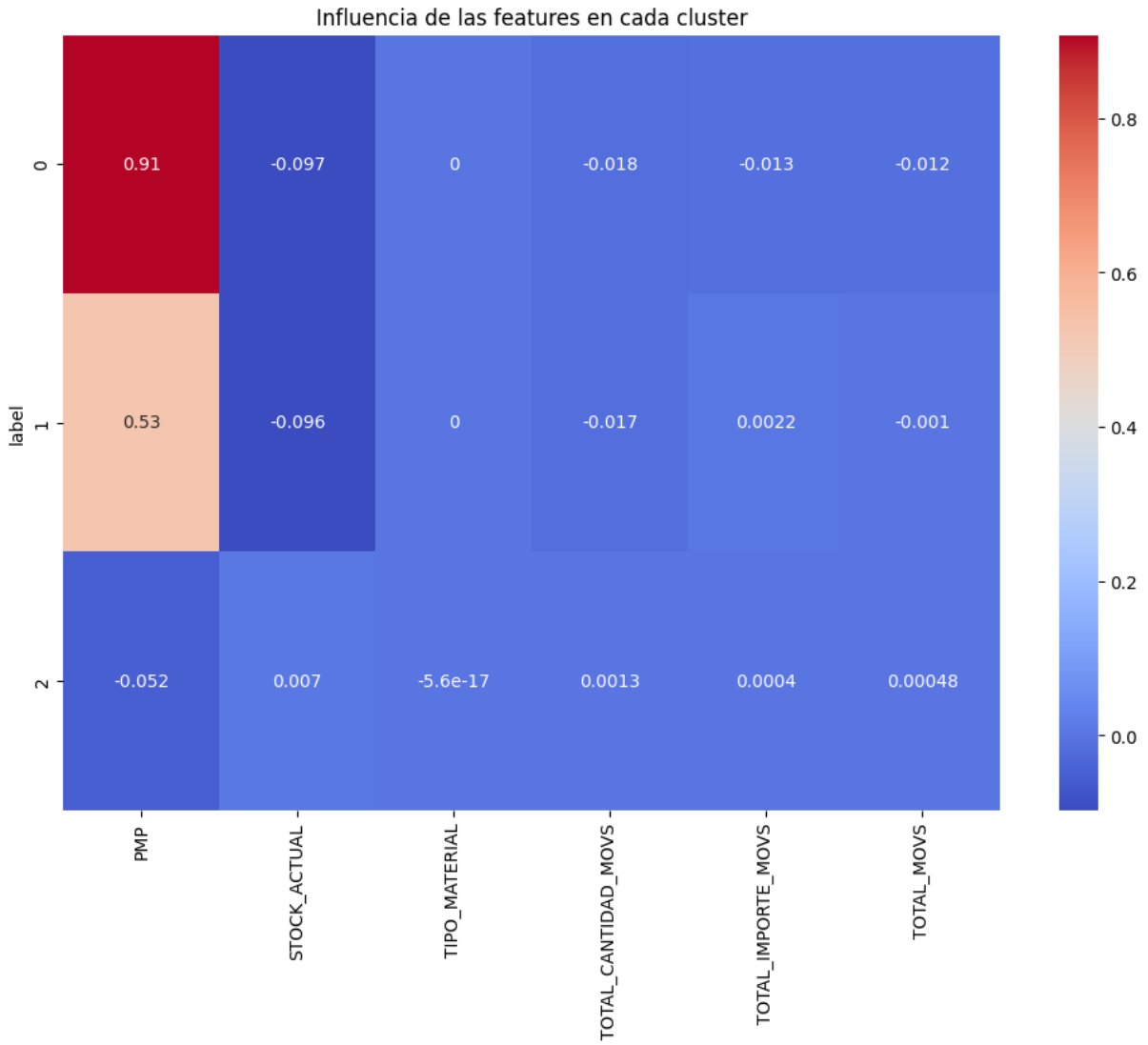


Figura 11. Mapa de calor de las features por clúster

Aplicando la técnica PCA generamos un gráfico para visualizar en un gráfico de ejes los puntos cada clúster en la Figura 12. Se apreciaz un clúster de gran tamaño con valores bajos para el componente principal 1 y que crece en el eje del componente principal 2. Los otros dos clústeres están claramente separados entre sí. En el gráfico de tarta se aprecia la diferencia de tamaño en los clústeres formados (Figura 13).

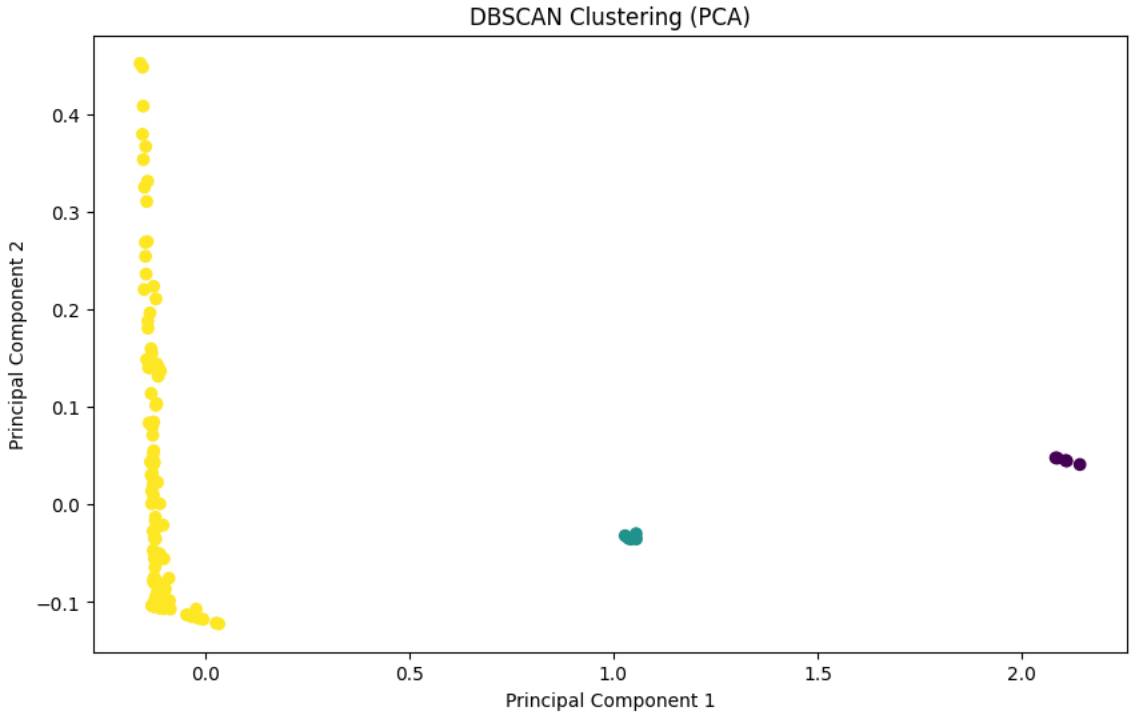


Figura 12. Gráfico DBSCAN aplicando PCA

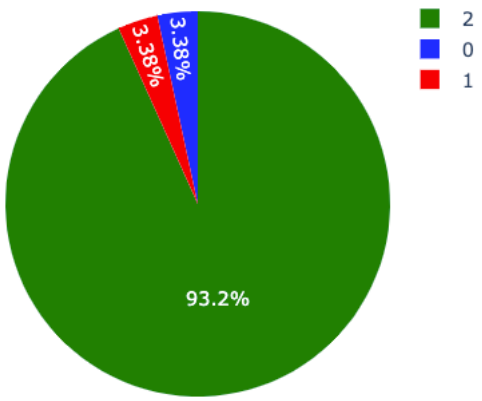


Figura 13. Gráfico de tarta DBSCAN

Concluimos que para el caso estudiado es más sencillo trabajar con KMeans o Agglomerative Clustering, y que entre ambos las métricas obtenidas son mejores en el caso de KMeans.

4.3. Comparativa soluciones AutoML Open Source

Se han seleccionado tres soluciones de AutoML para analizar: AutoGluon, PyCaret y H2O.

En este análisis exploraremos las opciones que ofrecen para llevar a cabo un entrenamiento de algoritmos de aprendizaje automático con los mismos datasets empleados anteriormente para el entrenamiento manual.

Valoraremos tanto las opciones disponibles para el entrenamiento, facilidad de uso, algoritmos disponibles y la información que aporten sobre los modelos generados y su evaluación.

Finalmente se seleccionará una de las opciones para su empleo dentro del ERP LIBRA.

4.3.1. AutoGluon

Desarrollado por un equipo de Amazon AI, permite trabajar con datos tabulares, series temporales y procesamiento de texto e imágenes. Está integrado en el servicio Amazon SageMaker, aunque es de acceso libre como librería Python que se puede instalar en versiones Python desde 3.8 hasta 3.11, en sistemas operativos Windows, Linux y MacOS.

Permite realizar la carga de datos, entrenamiento de modelos, predicción de datos y evaluación de los modelos. Cuenta con tres submódulos:

- **Tabular**. Específico para trabajar con conjuntos de datos tabulares, como archivos csv o Excel, susceptibles de cargarse en un DataFrame de pandas.
- **Multimodal**. Conjunto de modelos que permite el tratamiento de diferentes orígenes de datos: textos, imágenes y datos tabulares.
- **TimeSeries**. Tratamiento de series temporales, donde los datos son una secuencia de mediciones realizadas en intervalos regulares de tiempo.

En este estudio profundizaremos en las opciones del submódulo Tabular, pues se ajusta al formato en el que tenemos nuestros datasets.

Tabular puede producir modelos capaces de predecir el valor de una columna del dataset en función del resto de columnas y se puede utilizar tanto para tareas de clasificación como de

regresión. Permite abstraerse de las tareas de limpieza de datos, ingeniería de características, optimización de hiperparámetros y selección del mejor modelo.

Sin embargo, no dispone directamente de la capacidad de realizar tareas de clasificación no supervisada, por lo que no podrá ser utilizado para este fin.

La carga de datos tabulares se puede realizar usando la clase TabularDataset, que es una clase hija de los DataFrames de pandas.

Para el entrenamiento de modelos, se utiliza la clase TabularPredictor, que admite los siguientes parámetros básicos para su configuración:

Parámetro	**Tipo**	**Descripción**
label	str	Nombre de la variable objetivo
problem_type	str	Tipo de problema: binary, multiclass, regression, quantile. Si no se indica se intenta inferir automáticamente en base a los valores de la columna label.
eval_metric	str o Scorer	Métrica de evaluación del modelo. Si no se indica, se determina automáticamente en base al tipo de problema. - binary, multiclass: accuracy. - regression: rmse. - quantile: pinball_loss.
path	str	Directorio donde almacenar los modelos
log_to_file	bool	Permite guardar los logs generados en un fichero

Tabla 10. AutoGluon, parámetros TabularPredictor

Esta clase ofrece también una serie de métodos para trabajar con el conjunto de datos cargado. Se detallan a continuación los que se consideran relevantes para las tareas de clasificación.

Método	**Parámetros**	**Descripción**
fit	- train_data. Dataset. - time_limit. Tiempo en segundos para establecer el tiempo deseado de entrenamiento. - presets. Configuraciones disponibles ya preconfiguradas.	Lanza el entrenamiento sobre el dataset indicado. Se puede establecer el tiempo de entrenamiento y la calidad deseada mediante el uso de *presets*. Si no se indica valor de *hyperparameters*, se utilizará el

	- hyperparameters. Permite definir qué modelos entrenar y con qué conjunto de hiperparámetros. - excluded_model_types. Permite excluir ciertos modelos del entrenamiento.	conjunto de modelos por defecto para la tarea a realizar.
predict	- data. Conjunto de datos sobre el que realizar predicciones. - model. Modelo a utilizar, si no se indica se usa el mejor evaluado.	Utilizado para producir predicciones en datos sin clasificar.
evaluate	- data. - model. - auxiliary_metrics. Para evaluar métricas adicionales a la asignada por defecto. - detailer_report. Informe detallado.	Evalúa el rendimiento de la capacidad predictiva del modelo.
fit_summary	- show_plot. Muestra un gráfico resumen de los modelos.	Muestra información del rendimiento de cada modelo entrenado al ejecutar *fit*.
feature_importance	- data. - model. - time_limit.	Información sobre la relevancia de cada variable respecto a la variable objetivo y cómo influye en el rendimiento de los modelos.
leaderboard	- data. - extra_info. Mostrar más información. - extra_metrics. Información de métricas adicionales.	Incluye información sobre las puntuaciones de prueba y validación para todos los modelos, los tiempos de entrenamiento de los modelos y los tiempos de inferencia.
save		Guarda el modelo en el path indicado al inicializar el predictor.
load	- path.	Carga un modelo guardado para su uso.

Tabla 11. AutoGluon, métodos TabularPredictor

Por defecto AutoGluon utiliza una lista de modelos prefijados, aunque permite también emplear modelos personalizados. La lista de modelos base se muestra en la Tabla 12, se marcan en azul los que también se han usado en la validación manual.

Modelo
LGBModel
CatBoostModel

XGBoostModel
RFModel
XTModel
KNNModel
LinearModel
TabularNeuralNetTorchModel
NNFastAiTabularModel

Tabla 12. AutoGluon, lista de modelos

4.3.2. PyCaret

Es una librería Python de código abierto disponible desde 2020. Se centra en simplificar el flujo de trabajo de machine learning desde la carga y preprocesamiento de datos, modelado, interpretación de modelos y despliegue con pocas líneas de código. Como base utiliza librerías como scikit-learn, XGBoost, LightGBM, CatBoost, spacy, Hyperopt, Optuna o Ray.

Está disponible para versiones de Python desde 3.8 hasta 3.11 en Ubuntu y Windows. Funciona correctamente en MacOS aunque no se da soporte directamente. Su API tiene dos modos de funcionamiento: orientado a objetos (OOP) o programación funcional. Ambas ofrecen las mismas características.

Permite realizar los siguientes tipos de aprendizaje:

- Clasificación supervisada. Binaria o multiclase.
- Clustering o clasificación no supervisada.
- Regresión.
- Series temporales.
- Detección de anomalías. Otro tipo de clasificación no supervisada, enfocado en detectar los datos considerados anómalos.

PyCaret por lo tanto ofrece la posibilidad de entrenar modelos para los dos casos de estudio de clasificación supervisada y no supervisada.

La carga de datos debe realizarse usando un DataFrame de pandas.

Cualquier flujo de trabajo con PyCaret debe iniciarse con la función *setup,* que admite los siguientes parámetros para configurar el entorno de trabajo.

Parámetro	**Tipo**	**Descripción**

data	pandas.DataFrame	Dataset a utilizar
target	str	Nombre de la columna objetivo a predecir
train_size	float	Proporción del dataset que se usará para entrenamiento.
session_id	int	Semilla de reproducibilidad de los resultados
normalize	bool	Indica si se debe aplicar normalización a las características
normalize_method	str	Método de normalización a aplicar a seleccionar entre: zscore, minmax, maxabs, robust
categorical_features	list	Lista de variables a forzar como categóricas
ignore_features	list	Lista de variables a ignorar para el entrenamiento
numeric_features	list	Lista de variables a forzar como numéricas
date_features	list	Lista de variables a forzar como fechas
remove_outliers	bool	Si se indica True, se eliminan los outliers usando Isolation Forest
imputation_type	str or None	Tipo de imputación de valores faltantes. Puede ser simple, iterative o None si no se quieren imputar
numeric_imputation	int, float or str	Estrategia de imputación de valores faltantes para columnas numéricas si se ha seleccionado el tipo iterativo. A elegir entre: drop, mean, median, mode, knn o un valor fijo.
categorical_imputation	str	Estrategia de imputación de valores faltantes para columnas categóricas. Se ignora si se ha seleccionado el tipo iterativo. A elegir entre: drop, mode o valor fijo
fix_imbalance	bool	Si se indica True se aplica SMOTE para crear datos sintéticos con la finalidad de balancear el dataset

Tabla 13. PyCaret, parámetros de inicialización (setup)

Después de la inicialización del entorno de trabajo con *setup*, se debe crear un experimento usando la clase ClassificationExperiment o ClusteringExperiment según el caso de uso. Ambas cuentan a su vez con un método *setup* si se desea cambiar alguno de los parámetros para un experimento en específico.

Para entrenar y evaluar modelos de clasificación, se utiliza la función ***compare_models***. Al finalizar su ejecución muestra una tabla de clasificación con la lista de modelos entrenados, sus métricas de validación y tiempo de entrenamiento.

En el caso de modelos de clustering, se debe utilizar ***create_model*** indicando el nombre del modelo a utilizar. A continuación mediante el uso de ***assign_model*** se asignan los clústeres al conjunto de datos de entrenamiento.

Para el análisis de resultados se debe utilizar la función ***plot_model***, que permite la visualización de diferentes gráficos sobre los modelos entrenados.

La evaluación de los modelos se realiza mediante la función ***evaluate_model*** y la predicción de nuevos datos sobre un dataset con ***predict_model***.

Se pueden añadir o quitar métricas para la evaluación con los métodos ***add_metric*** y ***remove_metric***.

Se detallan a continuación los parámetros específicos para los métodos mencionados en el caso de clasificación supervisada y clustering.

Método	**Tipo**	**Parámetros**
compare_models	Clasificación	- include. Lista de modelos a incluir en el entrenamiento. - exclude. Lista de modelos a excluir. - budget_time. Tiempo en minutos máximo para la ejecución.
create_model	Clustering	- model. ID del modelo a entrenar. - num_clusters. Número de clusters objetivo.
assign_model	Clustering	- model. Objeto del modelo ya entrenado. - transformation. Aplicar las etiquetas directamente en el dataset.
plot_model	Clasificación	- plot. A elegir entre: auc, confusion_matrix, class_report, rfe, feature, parameter, lift, gain, tree, ks
	Clustering	- plot. A elegir entre: cluster (PCA 2d), tsne, elbow, silhouette, distance, distribution
evaluate_model	Clasificación	- estimator. Modelo entrenado. - fold. Número de folds para validación cruzada.
	Clustering	- model. Modelo entrenado. - feature. Característica a evaluar si el plot es cluster o tsne.
predict_model	Clasificación	- estimator. Model entrenado. - data. Dataframe sobre el que hacer las predicciones. - raw_score. Devuelve la puntuación para todas las etiquetas.
	Clustering	- model. Modelo entrenado. - data. Dataframe a etiquetar.
add_metric	Clasificación	- id. Identificador de la métrica. - name. Nombre para la métrica. - score_func. Función a aplicar para la métrica.
	Clustering	

		- greater_is_better. Indica si la métrica es mejor cuando el valor sea más alto o al contrario. - multiclass. Indica si soporta multiclase.
remove_metric	Clasificación Clustering	- id. Identificador de la métrica a eliminar.

Tabla 14. PyCaret, parámetros de los métodos principales

En las siguientes tablas se visualizan los modelos disponibles para clasificación (Tabla 15) y clustering (Tabla 16). Se marcan en color azul aquellos usados en la validación manual.

ID	Modelo	Librería origen
lr	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression
knn	K Neighbors Classifier	sklearn.neighbors._classification.KNeighborsClassifier
nb	Naive Bayes	sklearn.naive_bayes.GaussianNB
dt	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier
svm	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDClassifier
rbfsvm	SVM - Radial Kernel	sklearn.svm._classes.SVC
gpc	Gaussian Process Classifier	sklearn.gaussian_process._gpc.GaussianProcessClassifier
mlp	MLP Classifier	sklearn.neural_network._multilayer_perceptron.MLPClassifier
ridge	Ridge Classifier	sklearn.linear_model._ridge.RidgeClassifier
rf	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier
qda	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscriminatAnalysis
ada	Ada Boost Classifier	sklearn.ensemble._weight_boosting.AdaBoostClassifier
gbc	Gradient Boosting Classifier	sklearn.ensemble._gb.GradientBoostingClassifier
lda	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscriminantAnalysis
et	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier
xgboost	Extreme Gradient Boosting	xgboost.sklearn.XGBClassifier
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMClassifier
catboost	CatBoost Classifier	catboost.core.CatBoostClassifier
dummy	Dummy Classifier	sklearn.dummy.DummyClassifier

Tabla 15. PyCaret, modelos de clasificación disponibles

ID	Modelo	Librería origen
kmeans	K-Means Clustering	sklearn.cluster._kmeans.KMeans
ap	Affinity Propagation	sklearn.cluster._affinity_propagation.AffinityPropagation
meanshift	Mean Shift Clustering	sklearn.cluster._mean_shift.MeanShift
sc	Spectral Clustering	sklearn.cluster._spectral.SpectralClustering

hclust	Agglomerative Clustering	sklearn.cluster._agglomerative.AgglomerativeClustering
dbscan	Density-Based Spatial Clustering	sklearn.cluster._dbscan.DBSCAN
optics	OPTICS Clustering	sklearn.cluster._optics.OPTICS
birch	Birch Clustering	sklearn.cluster._birch.Birch

Tabla 16. PyCaret, modelos de clustering disponibles

Se puede apreciar que PyCaret tiene todos los algoritmos utilizados durante las pruebas de entrenamiento realizadas manualmente en ambos datasets.

4.3.3. H2O

Se trata de una plataforma de código abierto que permite el entrenamiento de modelos de machine learning de forma distribuida, por lo que ofrece una alta escalabilidad en términos de potencia de computación. Está totalmente desarrollado en Java, aunque se puede utilizar tanto en Java como en Python (3.6 hasta 3.11) o R (3 o superior) y funciona en Windows, Ubuntu, CentOS y MacOS. Para su uso es necesario contar con el JDK de Java instalado, soportando las versiones desde la 8 a la 17.

Para poder utilizar H2O, en primer lugar es necesario ejecutar el comando "*java -jar h2o.jar*" en el terminal, que levanta el servicio por defecto en la url *http://localhost:54321*. En el caso de computación distribuida se puede hacer en combinación con Apache Hadoop.

No soporta directamente AutoML con algoritmos de clasificación no supervisada, pero la librería de H2O sí que cuenta con el algoritmo K-Means para su utilización manual mediante una implementación de uso simple. Su módulo de AutoML ofrece una interfaz sencilla con pocos parámetros a configurar y soporta entrenamiento de modelos de clasificación supervisada y regresión. Revisaremos los principales a utilizar en el caso de utilizar la librería de Python ***h2o***.

Para la carga inicial de datos se puede utilizar un DataFrame de pandas usando ***h2o.H2OFrame*** o cargar un fichero desde una URL o ruta local utilizando ***h2o.import_file***.

A continuación es necesario crear un objeto de clase ***H2OAutoML*** que se puede inicializar con varios parámetros:

- **max_models**. Obligatorio, indica el número máximo de modelos a evaluar.
- **max_runtime_secs**. Obligatorio, corresponde al tiempo máximo en segundos a emplear en el entrenamiento.

- **sort_metric**. Métrica a emplear para la puntuación de los modelos.
- **stopping_metric**. Métrica a utilizar para parar el entrenamiento cuando no se consiga mejorar los resultados (EarlyStopping).
- **balance_clases**. Indica si se deben balancear las clases del dataset.

Una vez creado el objeto H2OAutoML con los parámetros iniciales, este dispone de varios métodos para su utilización. Mediante el método ***train*** se lanza el entrenamiento de los modelos, para ello es necesario únicamente indicar el conjunto de datos (***training_frame***), el conjunto de variables (***x***) y la variable a predecir (***y***). El entrenamiento durará como máximo hasta consumir el tiempo indicado inicialmente.

Al finalizar el entrenamiento, se puede visualizar el resultando invocando al método ***leaderboard***. Si se desea examinar más en detalle el comportamiento del mejor modelo entrenado se puede hacer empleando el método ***leader.model_performance.*** Este objeto es el que devuelve la información necesaria para la evaluación del modelo en función de diferentes métricas según la tarea realizada.

Para la predicción de datos se debe invocar al método ***predict*** indicando como único parámetro el dataset sobre el que realizar las predicciones. Las predicciones se devuelven en otro dataframe, sobre el que es posible aplicar métricas de evaluación manualmente si no bastan las proporcionadas por H2O.

Para el caso de clasificación no supervisada la librería ***h2o*** cuenta con la clase ***H2OKMeansEstimator***, que permite entrenar un modelo K-Means de forma sencilla, aunque no aplica ningún preprocesamiento automático sobre el dataset por lo que no se considera su evaluación dentro del ámbito de este estudio.

4.3.4. Comparativa AutoML: clasificación supervisada

Las tres librerías cuentan con la capacidad de realizar AutoML para tareas de clasificación supervisada, por lo que se ha podido realizar una comparativa de las tres soluciones.

4.3.4.1. AutoGluon

Para el entrenamiento se han fijado los siguientes parámetros:

- time_limit=3600. Fija en 3600 segundos el tiempo a emplear para el entrenamiento.

- presets=best_quality. Busca el mejor resultado posible sin valorar el tiempo de entrenamiento o inferencia.

El resto de parámetros se han dejado en automático, sin limitar el uso de modelos.

En la Tabla 17 se muestran los resultados del entrenamiento con los 5 datasets, mostrando los 5 primeros modelos con mayor puntuación en accuracy sobre los datos de validación. Destacan los modelos WeightedEnsemble, que se construye como combinación de otros modelos de forma similar al Stacking realizado en el entrenamiento manual. RandomForest en su variante gini y entropy también tiene una presencia destacada al entrar en el top 5 en 4 ocasiones.

Dataset	**Evaluación Modelo**		**Feature Importance**	
	Modelo	**Exactitud validación**	**Feature**	**Importancia**
Industrial	NeuralNetFastAI	0,908071	max_fecha_cobro	0,38564
	WeightedEnsemble	0,90599	fecha_factura	0,17504
	LightGBM	0,898676	forma_cobro	0,1404
	KNeighborsDist	0,884111	numero_serie	0,02884
	KNeighborsUnif	0,815637	mes_factura	0,00856
Retail	XGBoost	0,97533	max_fecha_cobro	0,38184
	WeightedEnsemble	0,97533	forma_cobro	0,03232
	NeuralNetFastAI	0,975073	fecha_factura	0,02504
	LightGBM	0,975052	mes_factura	0,00772
	RandomForest(gini)	0,975042	cliente	0,00608
Exportación	WeightedEnsemble	0,974387	max_fecha_cobro	0,42844
	XGBoost	0,973984	count_tipos_transaccion	0,29268
	RandomForest(entropy)	0,973863	forma_cobro	0,24016
	RandomForest(gini)	0,973836	mes_factura	0,1608
	LightGBMXT	0,973775	fecha_factura	0,09104
Distribución Farmacéutica	WeightedEnsemble	0,960784	max_fecha_cobro	0,41096
	RandomForest(entropy)	0,96069	fecha_factura	0,12792
	RandomForest(gini)	0,960508	forma_cobro	0,10668
	ExtraTrees (gini)	0,960109	mes_factura	0,0974
	ExtraTrees (entropy)	0,960104	divisa	0,05844
Alimentación	WeightedEnsemble	0,940406	count_tipos_transaccion	0,25932
	RandomForest(gini)	0,939699	fecha_factura	0,21388
	RandomForest(entropy)	0,939198	num_efectos_factura	0,11736
	NeuralNetFastAI	0,933757	numero_serie	0,08032
	LightGBM	0,932628	max_fecha_cobro	0,04892

Tabla 17. AutoGluon, evaluación resultados clasificación supervisada

4.3.4.2. PyCaret

Se han empleado los siguientes parámetros para la configuración del experimento de entrenamiento:

- folds=10.
- fix_imbalance=True.
- fix_imbalance_method=smote.
- budget_time=3600.
- normalize=True.

Se detecta que a pesar de indicar en el tiempo de entrenamiento un límite de 60 minutos, el máximo tiempo empleado ha sido de 17 minutos para el dataset de distribución farmacéutica, por lo que en todos los casos ha agotado las opciones antes de llegar al límite de tiempo. Esto supone una diferencia respecto a AutoGluon que en todos los casos agotaba el límite establecido antes de finalizar.

Las métricas obtenidas varían significativamente en función del dataset evaluado, consiguiendo el mejor resultado (0,9568 accuracy) para el caso del sector retail y el peor (0,7785 accuracy) para el sector industrial.

Dataset	Tiempo (min)	Modelo	Exactitud	Precisión	Recall	F1-score
Industrial	**4**	Extra Trees Classifier	0,7785	0,7785	0,787	0,7803
		K Neighbors Classifier	0,7591	0,7591	0,7566	0,7559
		Random Forest Classifier	0,7548	0,7548	0,763	0,757
		Light Gradient Boosting Machine	0,7477	0,7477	0,7419	0,7436
		Extreme Gradient Boosting	0,7451	0,7451	0,7392	0,7408
Retail	**10**	Extra Trees Classifier	0,9568	0,9568	0,9588	0,9576
		Random Forest Classifier	0,9455	0,9455	0,9491	0,9465
		CatBoost Classifier	0,9096	0,9096	0,9164	0,9048
		Linear Discriminant Analysis	0,9077	0,9077	0,9068	0,906
		K Neighbors Classifier	0,9041	0,9041	0,8969	0,8981
Exportación	**5**	Extra Trees Classifier	0,8802	0,8802	0,8866	0,8796
		Extreme Gradient Boosting	0,8512	0,8512	0,8535	0,8511
		Random Forest Classifier	0,8502	0,8502	0,8576	0,8491
		CatBoost Classifier	0,8495	0,8495	0,8519	0,8493
		K Neighbors Classifier	0,8453	0,8453	0,8497	0,8443
Distribución Farmacéutica	**17**	Extra Trees Classifier	0,8868	0,8868	0,8825	0,8734
		K Neighbors Classifier	0,8853	0,8853	0,8804	0,8807

		Ridge Classifier	0,8425	0,8425	0,8071	0,7985
		Random Forest Classifier	0,8423	0,8423	0,8249	0,8189
		Extreme Gradient Boosting	0,8422	0,8422	0,8275	0,8291
Alimentación	**13**	Extra Trees Classifier	0,8892	0,8892	0,8825	0,8846
		Random Forest Classifier	0,8587	0,8587	0,8499	0,8524
		Decision Tree Classifier	0,7761	0,7761	0,7691	0,772
		Extreme Gradient Boosting	0,776	0,776	0,7586	0,7643
		CatBoost Classifier	0,7758	0,7758	0,7585	0,7638

Tabla 18. PyCaret, evaluación resultados clasificación supervisada

En vista de los resultados más moderados que AutoGluon en algunos datasets, se ha procedido a aplicar las funcionalidades de ***Blending*** y ***Stacking*** que ofrece PyCaret que permiten combinar los mejores modelos entrenados en uno. Sin embargo el modelo resultante no ha mejorado los resultados del mejor modelo individual en ningún caso, por lo que se ha descartado su aplicación. Destaca el modelo Extra Trees Classifier, siendo el que mejor accuracy tiene para los cinco datasets.

PyCaret ofrece de forma más sencilla una tabla de métricas similar a la realizada en los entrenamientos manuales, pudiendo evaluar directamente la precisión, recall y f1-score.

También ofrece mediante el método plot_model la posibilidad de obtener matrices de confusión, importancia de características o un informe de métricas por cada una de las clases del mejor modelo entrenado. Se muestra a continuación un ejemplo de estos informes para el dataset retail.

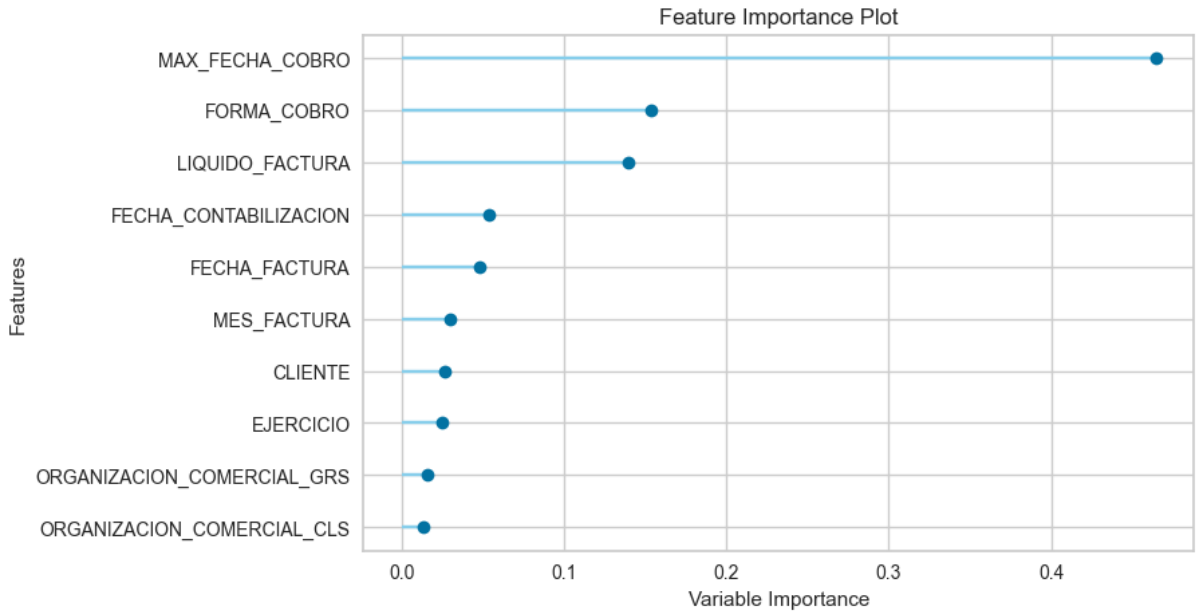


Figura 14. PyCaret, feature importance plot retail dataset

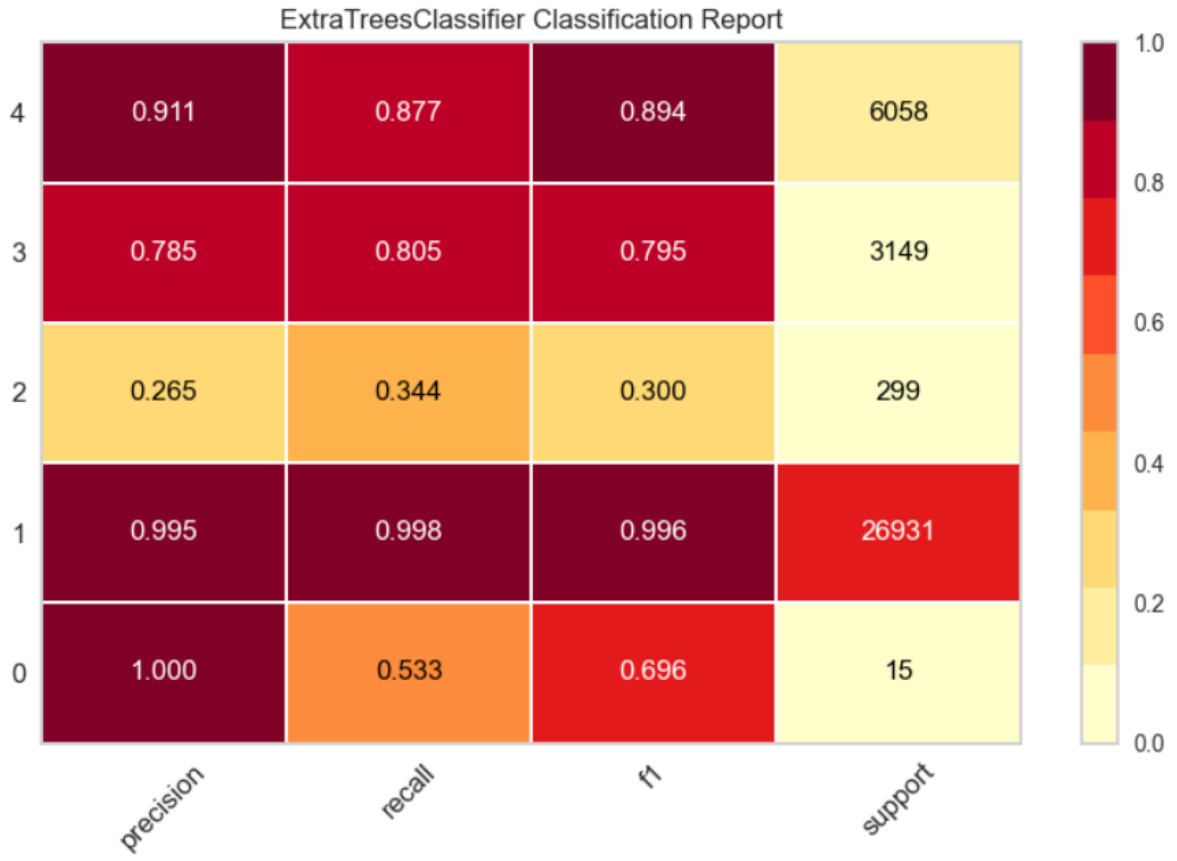


Figura 15. PyCaret, classification report retail dataset

4.3.4.3. H2O AutoML

Para la configuración del entorno de entrenamiento de H2O se han seleccionado los parámetros siguientes, recomendados por la documentación de H2O para entrenamiento multiclase:

- max_models=20.
- max_runtime_secs=1200.
- sort_metric=mean_per_class_error.
- distribution=multinomial.

H2O también ha consumido el máximo tiempo proporcionado de 20 minutos. Se ha tenido que limitar el tiempo de ejecución respecto a AutoGluon y PyCaret, pues en caso contrario H2O incurría en errores de memoria y no llegaba a devolver ningún modelo entrenado. Para los cinco datasets han quedado entre los mejores cinco modelos variantes de GBM (Gradient

Boosting Machine). Se estima que el motivo es el limitado tiempo de ejecución indicado pues no ha llegado a probar todas las opciones de modelos de los que dispone.

Se muestran en la Tabla 19 las métricas del mejor modelo entrenado para cada dataset y en la Tabla 20 un ejemplo de la tabla de mejores modelos (leaderboard) generada en el entrenamiento del dataset de distribución farmacéutica.

Dataset	Modelo	Exactitud	Precisión	Recall	F1-score
Industrial	H2OGradientBoostingMachine	0,8936	0,8914	0,8936	0,8912
Retail	H2OGradientBoostingMachine	0,9683	0,965	0,9683	0,966
Exportación	H2OGradientBoostingMachine	0,9237	0,9246	0,9237	0,9239
Distribución Farmacéutica	H2OGradientBoostingMachine	0,9366	0,9358	0,9366	0,9354
Alimentación	H2OGradientBoostingMachine	0,9405	0,9383	0,9405	0,9381

Tabla 19. H2O, resultados clasificación supervisada

model_id	mean per class error	logloss	rmse	mse	training time ms
GBM_2_AutoML_1_20240626_80652	0,0900303	0,183581	0,227006	0,0515318	27859
GBM_3_AutoML_1_20240626_80652	0,0905416	0,188477	0,227444	0,051731	22108
GBM_5_AutoML_1_20240626_80652	0,0915209	0,184423	0,228492	0,0522086	38452
GBM_4_AutoML_1_20240626_80652	0,0940894	0,198483	0,229904	0,052856	17312
GBM_1_AutoML_1_20240626_80652	0,0959975	0,2145	0,232962	0,0542715	20662

Tabla 20. H2O, leaderboard para dataset distribución farmacéutica

Dataset	Feature	Importancia
Industrial	max_fecha_cobro	0,604
	forma_cobro	0,123
	fecha_factura	0,121
	numero_serie	0,106
	liquido_factura	0,011
Retail	max_fecha_cobro	0,8157
	forma_cobro	0,08
	fecha_factura	0,055
	ejercicio	0,11
	mes_factura	0,011
Exportación	max_fecha_cobro	0,3393
	num_efectos_factura	0,1973
	fecha_factura	0,1155
	cliente	0,073
	forma_cobro	0,06
Distribución Farmacéutica	max_fecha_cobro	0,3864
	divisa	0,1867

	numero_serie	0,1573
	forma_cobro	0,069
	fecha_factura	0,042
Alimentación	fecha_contabilizacion	0,2372
	fecha_factura	0,1865
	count_tipos_transaccion	0,1747
	numero_serie	0,1393
	num_efectos_factura	0,08

Tabla 21. H2O, importancia de características

Aunque H2O AutoML indica que dispone de una herramienta para generación de reportes y plots sobre el mejor modelo entrenado (explain), no funciona con los modelos entrenados por lo que no se ha sido capaz de generar directamente con H2O ningún gráfico.

Los resultados obtenidos en cuanto a métricas superan en todos los casos el 89% de accuracy en validación, sin embargo, H2O ha sido el entorno de trabajo más complicado de desplegar, probar y evaluar debido a su mayor complejidad técnica y consumo de recursos elevado, probablemente debido a su orientación a la computación distribuida.

4.3.5. Comparativa AutoML: clasificación no supervisada

En esta sección únicamente se podrá evaluar el comportamiento de PyCaret, debido a que ni AutoGluon ni H2O AutoML disponen de algoritmos de clustering.

4.3.5.1. PyCaret

Para la evaluación se han seleccionado los tres mismos modelos que en el proceso de entrenamiento manual, K-Means, Agglomerative Clustering y DBSCAN; sin embargo el entrenamiento con DBSCAN no ha devuelto ningún resultado para los cinco datasets, por lo que los resultados obtenidos se limitan a K-Means y Agglomerative Clustering.

Para trabajar con modelos de clustering en PyCaret se debe crear un experimento de tipo ***ClusteringExperiment***, y posteriormente entrenar los modelos deseados uno a uno, pues no ofrece algún automatismo para realizar un único entrenamiento con varios modelos al ser más complicado de evaluar cuál tiene el mejor comportamiento.

El único parámetro indicado explícitamente ha sido ***num_clusters*** con el valor objetivo de 3 clústeres para mantener el mismo criterio que en el entrenamiento manual.

Se muestra una comparativa con las métricas Silhouette y Davies-Bouldin en la Tabla 22.

Dataset	K-Means		Agglomerative Clustering	
	Silhouette	Davies-Bouldin	Silhouette	Davies-Bouldin
Industrial	0,9118	0,2156	0,9873	0,2011
Distribución farmacéutica	0,7453	0,3956	0,7394	0,3238
Retail	0,9228	0,3145	0,905	0,4198
Alimentación	0,7439	0,317	0,7418	0,2417
Exportación	0,8593	0,5246	0,8663	0,4252

Tabla 22. PyCaret, resultados clustering

En la comparación con los resultados obtenidos en el entrenamiento manual de clustering, se puede apreciar una mejora general de ambas métricas, especialmente del índice Davies-Bouldin que es muy inferior en el entrenamiento realizado con PyCaret.

Para evaluar los clústeres formados, PyCaret ofrece varias opciones empleando los métodos ***plot_cluster*** y ***evaluate_model.*** Se muestran varios ejemplos de los gráficos que es posible obtener, para el dataset de distribución farmacéutica. Para el caso de K-Means se permite la obtención de una mayor variedad de gráficas respecto al clustering aglomerativo.

Figura 16. PyCaret clustering K-Means. Gráfico 2D PCA

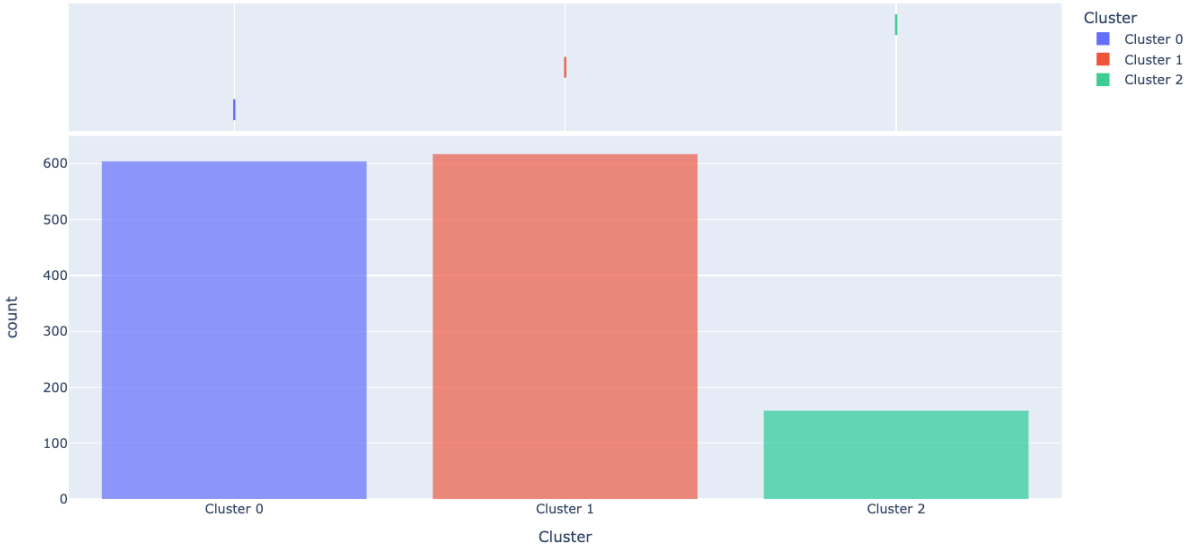


Figura 17. PyCaret clustering K-Means. Gráfico distribución de clústeres

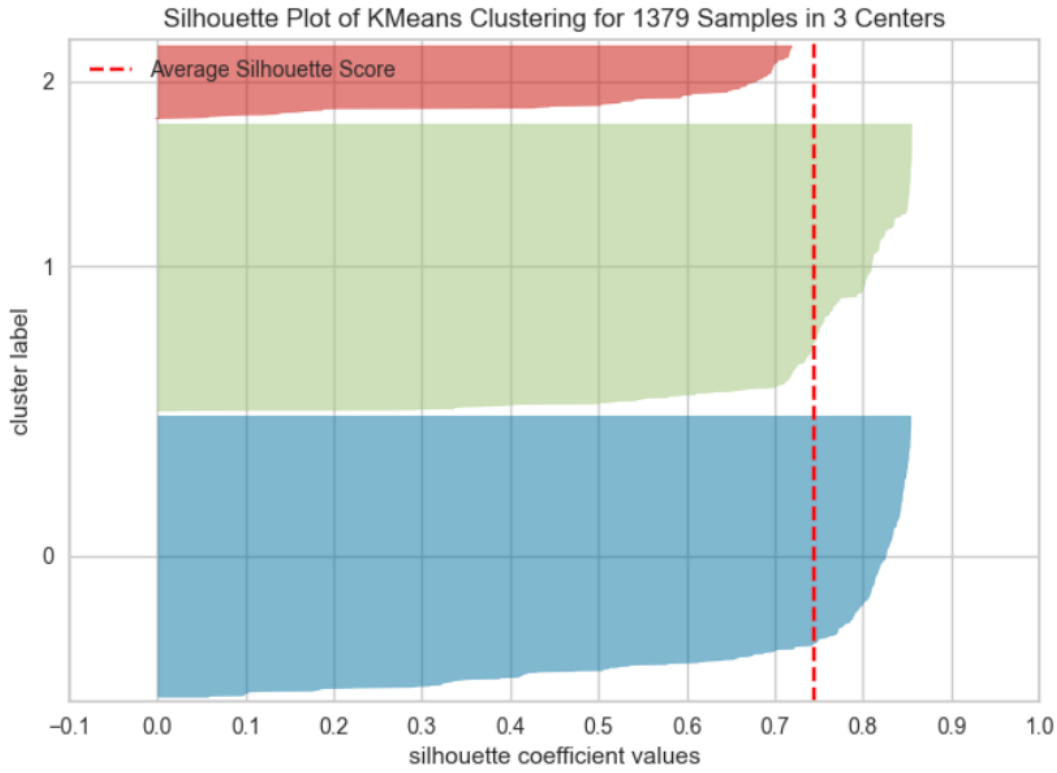


Figura 18. PyCaret clustering K-Means. Gráfico Silhouette

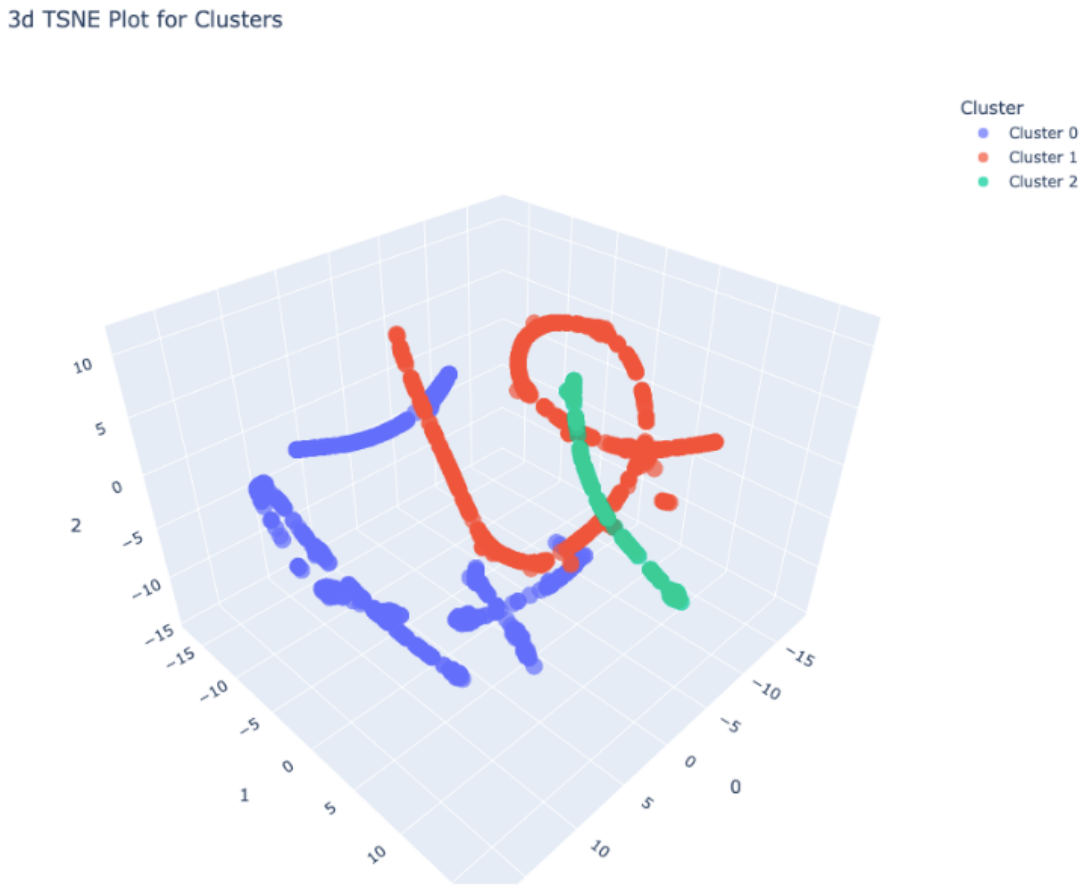


Figura 19. PyCaret Agglomerative clustering. Gráfico 3D TSNE.

4.3.6. Selección de solución AutoML

Una vez evaluadas las opciones AutoML debemos seleccionar una de ellas para su integración con el ERP Libra.

En primer lugar destacamos que únicamente PyCaret ofrece directamente la posibilidad de entrenar mediante su solución algoritmos de clustering, por lo que para esta tarea será la herramienta seleccionada.

En cuanto a tareas de clasificación supervisada, las tres soluciones cuentan con herramientas para llevarla a cabo, por lo que podemos realizar una comparación entre ellas en varios aspectos.

4.3.6.1. Facilidad de uso

En cuanto a la facilidad de instalación, despliegue y configuración tanto AutoGluon como PyCaret han mostrado ser sencillas de implementar, mientras que H2O tiene un punto más de

complejidad al ser una herramienta orientada a la posibilidad de computación distribuida en entornos como Hadoop y hacer uso de un servidor Java.

H2O es la única herramienta en la que se han encontrado problemas de carácter técnico durante el entrenamiento o evaluación de modelos, al causar problemas de memoria que incurrían en un cuelgue del servidor, teniendo que reiniciarlo manualmente.

Respecto a la facilidad para entrenar modelos, evaluarlos y guardarlos, las tres soluciones tienen un comportamiento similar.

4.3.6.2. Rendimiento de los modelos entrenados

En el caso de los datasets de clustering únicamente se puede evaluar PyCaret respecto al entrenamiento manual previamente realizado, concluyendo que PyCaret consigue en general mejores métricas Silhouette y Davies-Bouldin que en el entrenamiento manual.

Para los datasets de clasificación supervisada, las mejores métricas se han obtenido con AutoGluon, seguido de H2O y PyCaret que ha mostrado un rendimiento más variable según el dataset.

Para el uso exclusivo con tareas de clasificación supervisada, se estima que AutoGluon es la solución más fiable, siendo la única que ha consumido el máximo tiempo de entrenamiento fijado y sin ocasionar problemas técnicos como H2O. PyCaret obtiene unos resultados inferiores en cuanto a métricas aunque con unos tiempos de entrenamiento muy inferiores a los que se fijaron como máximos, lo que indica que ha agotado las acciones que realiza por defecto.

4.3.6.3. Interpretación de los modelos entrenados

Cada solución ofrece diferentes opciones en cuanto a métricas y gráficos siendo en este sentido la más potente PyCaret. Con AutoGluon se pueden obtener las métricas de principal interés y la importancia de las características, aunque siempre en formato texto o json.

La herramienta específica para obtención de gráficos de H2O no ha funcionado con los modelos entrenados, pues no es compatible con todos los modelos que entrena automáticamente H2O.

4.3.6.4. Conclusiones

En primer lugar se ha descartado H2O debido a su mayor complejidad de uso, generación de diversos errores en tiempo de ejecución y a que no se necesita hacer uso de su funcionalidad de computación distribuida.

En cuanto al rendimiento general de los modelos entrenados para clasificación supervisada, AutoGluon es la opción más destacada, sin embargo no dispone de algoritmos de clustering o generación de gráficas para evaluar los modelos entrenados. Consideramos este punto como algo relevante dado que el usuario objetivo de esta herramienta será personal no experto en tareas de aprendizaje automático.

Por lo tanto se ha seleccionado PyCaret como la librería de AutoML sobre la que desarrollar la integración con el ERP Libra, destacando su rapidez de entrenamiento, facilidad de uso, disponibilidad de algoritmos de clustering, fiabilidad y amplitud de opciones en cuanto a evaluación de los modelos entrenados.

4.4. Integración de PyCaret en ERP Libra

Para la integración de PyCaret en el ERP, se deben definir los siguientes aspectos:

- **Despliegue de un servidor dedicado**. Se instalará en este servidor un entorno con Linux (Ubuntu 22.04), Python 3.10, PyCaret y el resto de librerías auxiliares necesarias.
- **Creación de un API para AutoML.** Este servidor albergará un API que se consumirá desde el ERP para el entrenamiento de modelos y la publicación de modelos entrenados para su uso con nuevos datos a clasificar.
- **Realización de datasets**. Se utilizará la herramienta ya disponible en Libra denominada “Generador de Informes”, que permite diseñar un informe tabular con todos los orígenes de datos disponibles en el ERP. Para los casos más complejos será necesario realizar algún proceso de base de datos que genere los resultados en una tabla dentro de la base de datos Oracle del ERP.
- **Creación de un cuadro de mando AutoML.** Crearemos dentro del ERP una opción de menú, donde los usuarios con permisos para ello accederán para realizar el entrenamiento de modelos. Será desde este punto donde se consumirá el API de entrenamiento de modelos. También se podrán visualizar los modelos previamente entrenados.

- **Consumo del API para nuevos datos.** Este punto dependerá de cada caso de uso, pues el envío de nuevos datos podrá realizarse desde diferentes puntos del ERP. En este estudio documentaremos el caso de uso de clasificación supervisada para la clasificación del riesgo de impago de una factura de ventas.

4.4.1. Diseño del API AutoML

Dado que estamos trabajando con Python se ha seleccionado FastAPI para la creación del servicio. Se opta por trabajar con Dockers para generar una imagen a usar para el despliegue sencillo y rápido de la aplicación.

El API contará con tres servicios o endpoints:

- **train**. Recibirá un archivo csv que corresponde al dataset y los parámetros para el entrenamiento que inicialmente serán únicamente 4:
    - budget_time. Tiempo máximo para el entrenamiento.
    - task. Tarea a realizar: classification o clustering.
    - target. En el caso de seleccionar tarea de clasificación, será la etiqueta de la variable objetivo.
    - clusters. En el caso de seleccionar una tarea de clustering, será el número de clústeres objetivo.

  Esto lanzará el proceso de entrenamiento con PyCaret, almacenando en una base de datos interna (sqlite) los datos del mejor modelo entrenado y persistiendo en disco el modelo. Se asignará un identificador único al modelo, que será la respuesta de este endpoint, de esta manera posteriormente se podrá consumir el modelo indicando el identificador.

- **model_info**. Recibe como único parámetro el identificador único del modelo. Devuelve los datos del modelo, métricas de evaluación y estructura del dataframe usado para el entrenamiento.
- **predict**. Recibe el identificador único del modelo y un registro de datos que debe tener la misma estructura que el dataframe empleado para el entrenamiento. Devuelve la clase o clúster que predice el modelo para este registro.

4.4.2. Diseño del dataset en Libra

Se empleará una herramienta nativa de Libra para el diseño de los datasets. Se muestra un ejemplo donde se selecciona una tabla DATASET_FACTURAS que contiene datos con la misma estructura que el utilizado para entrenar los modelos de clasificación supervisada de este trabajo.

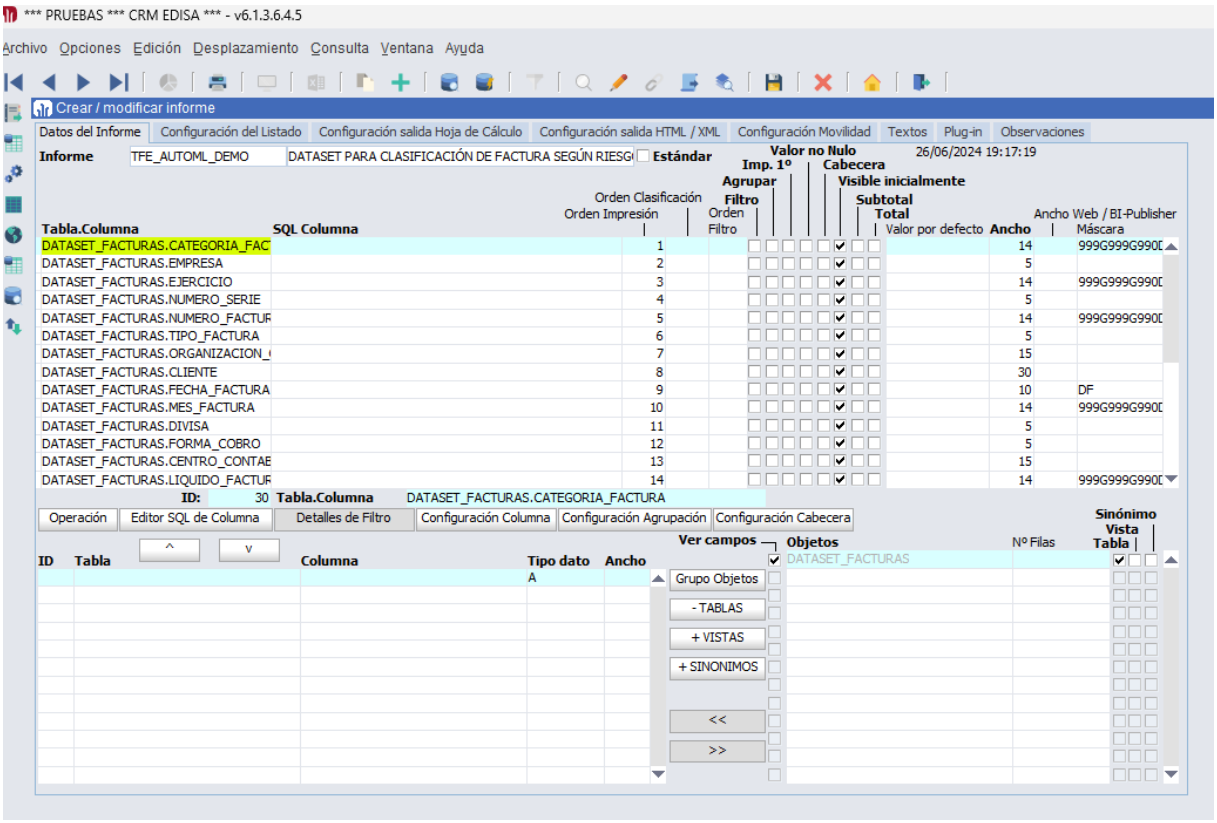


Figura 20. Diseño de datasets en Libra

4.4.3. Cuadro de mando AutoML integrado en Libra

Se ha diseñado una pantalla dentro de Libra, usando Oracle Forms siguiendo la guía de diseño interna de EDISA para el desarrollo del ERP Libra.

Esta pantalla se ha integrado en el menú de Libra con el nombre “Entrenamiento AutoML” dentro del menú “Entorno”, donde se ubican otras herramientas avanzadas de configuración del ERP.

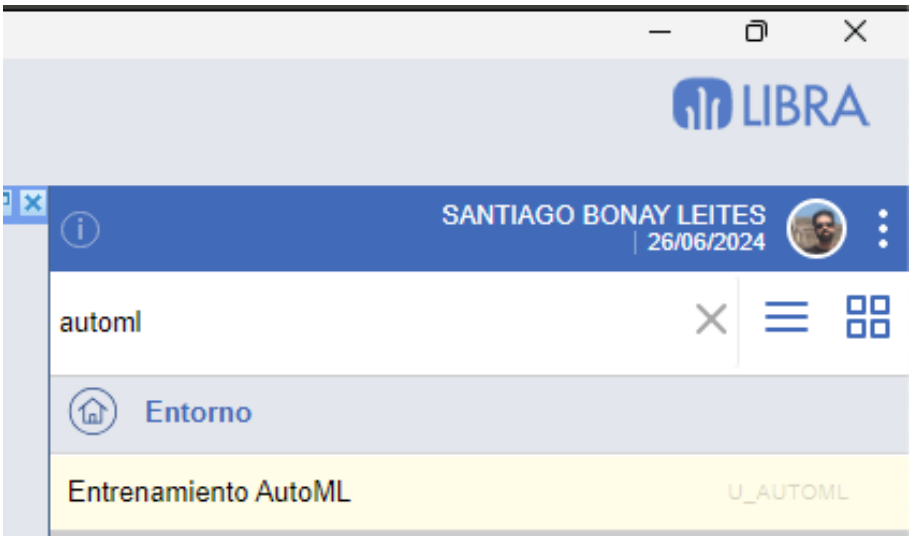


Figura 21. Cuadro de mando AutoML en el menú de Libra

Al seleccionar esta opción se abrirá la pantalla con el cuadro de mando para lanzar el entrenamiento de un modelo de machine learning, invocando al API diseñada en el punto anterior. El usuario deberá seleccionar el origen de datos, que corresponderá con un informe diseñado en Libra previamente y el tiempo límite que desea emplear para el entrenamiento. Según el tipo de tarea especificado indicará el resto de parámetros.

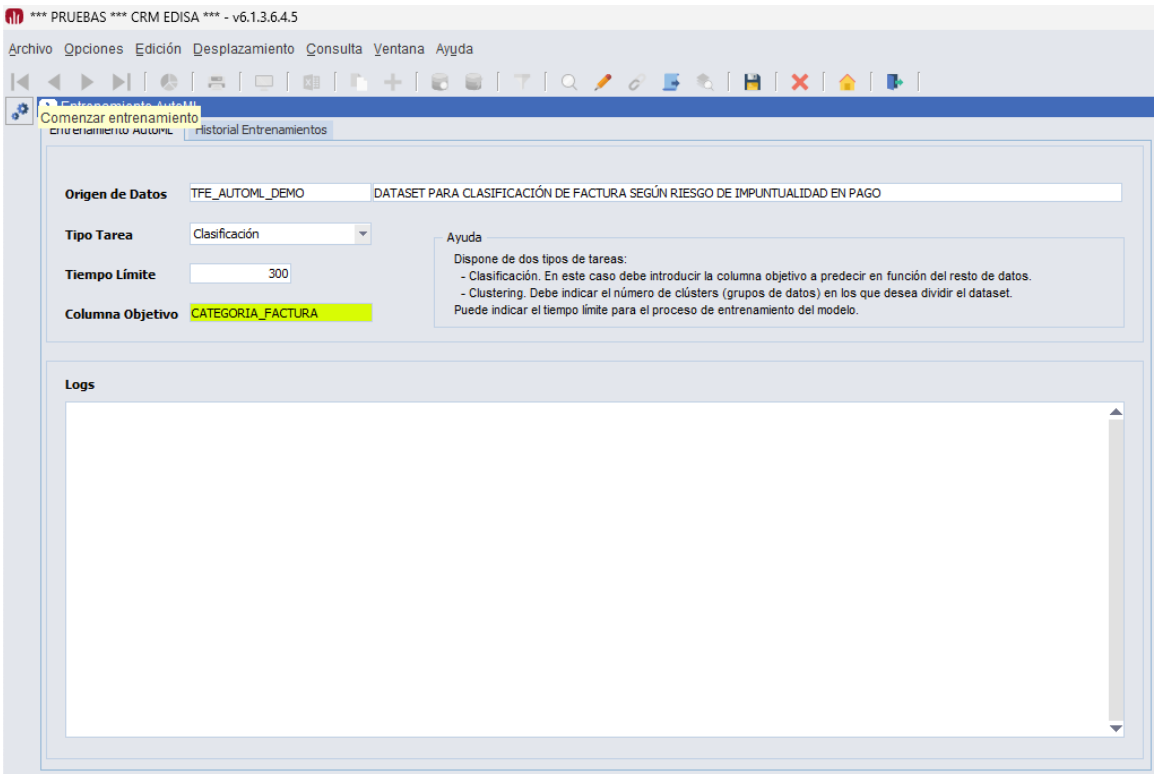


Figura 22. Diseño de cuadro de mano AutoML en Libra

Una vez seleccionados los parámetros podrá pulsar el botón “Comenzar entrenamiento” que invocará el endpoint ***train*** del API. Al finalizar el entrenamiento se mostrará en el campo Logs (Figura 23) el resultado de invocar al endpoint ***model_info***.

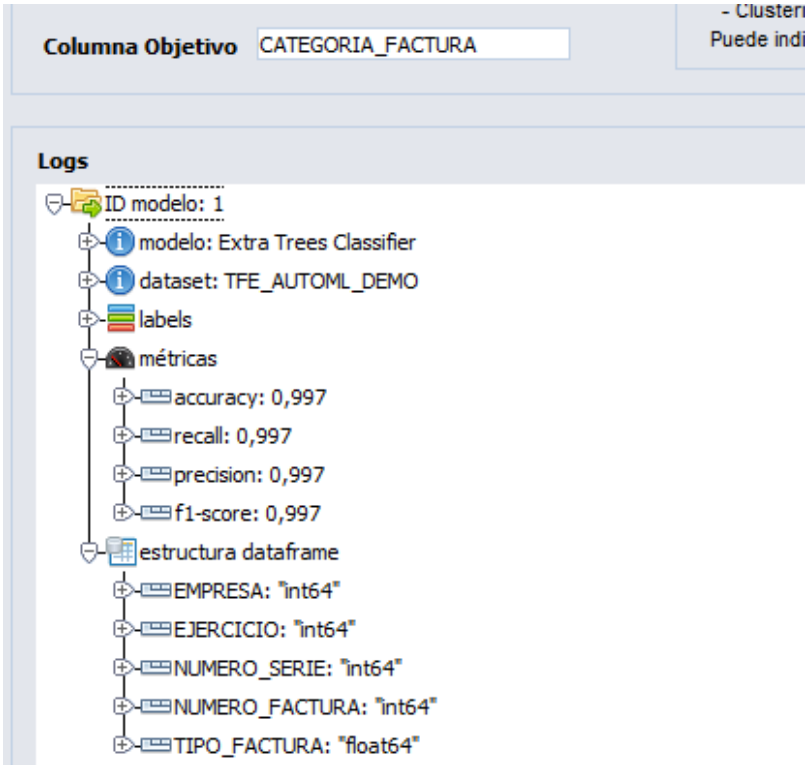


Figura 23. Resultados entrenamiento AutoML en Libra

En el caso de entrenamiento de modelos clustering se mostrará por defecto un gráfico de la métrica Silhouette, pudiendo elegir también un gráfico de distribución (Figura 24).

En la pestaña “Historial Entrenamientos” se podrá ver un logs de los modelos entrenados hasta el momento (Figura 25).

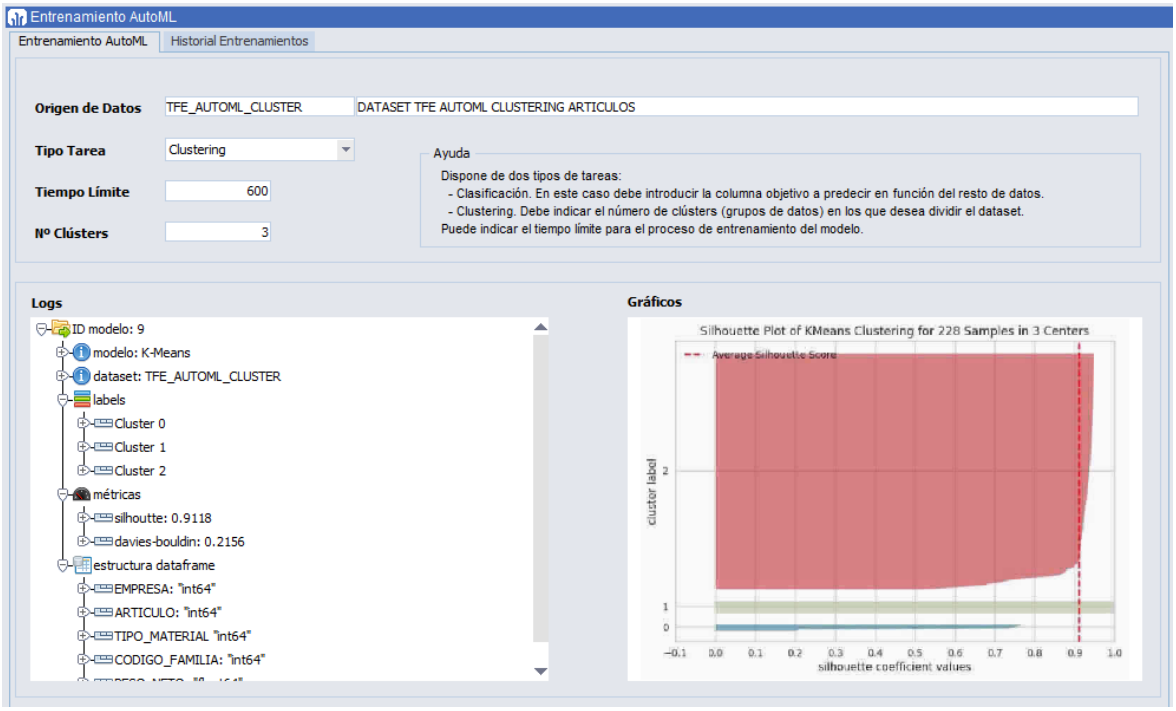


Figura 24. Entrenamiento clustering en Libra

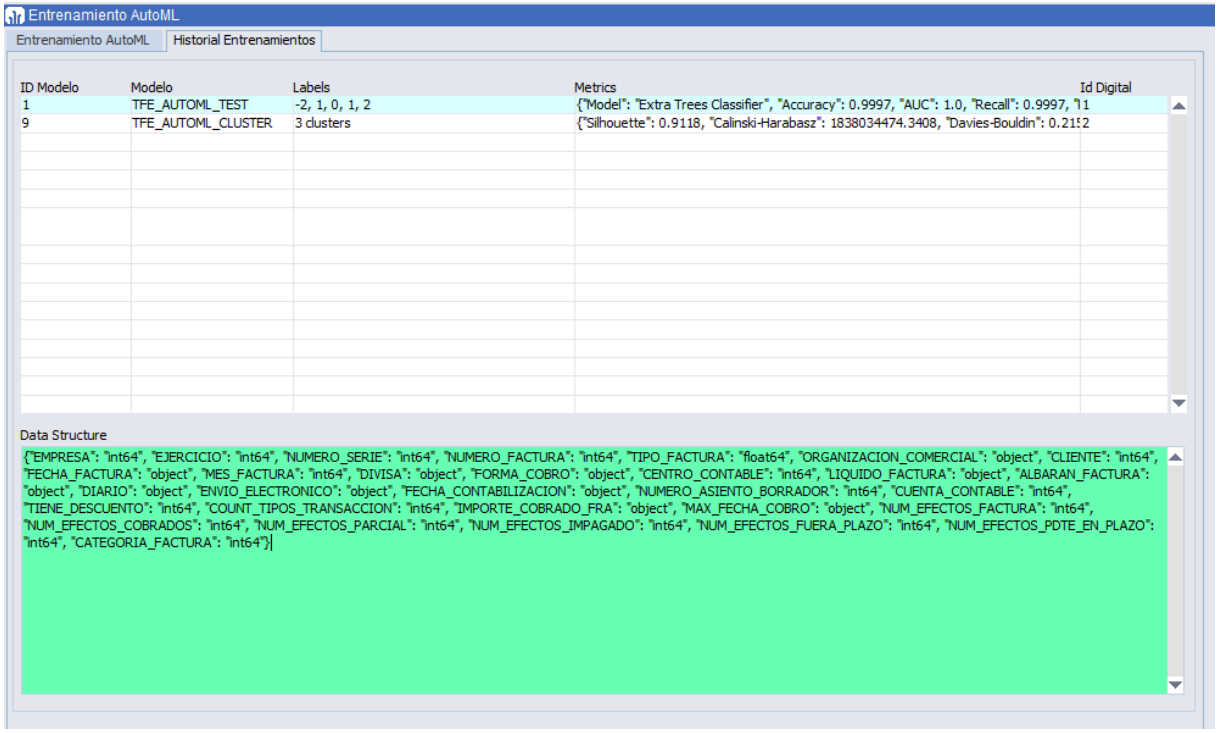


Figura 25. Historial de entrenamientos Libra

4.4.4. Consumo del modelo entrenado en Libra

Para el uso de los modelos entrenados, será necesario conocer el id del modelo. Mostramos a continuación cómo se ha integrado el modelo de clasificación de riesgo de pago de facturas en la consulta de facturas de ventas de LIBRA.

- Realizar una personalización del programa. Libra incluye una herramienta para modificar ciertos comportamientos de sus programas en cada instalación. Se ha realizado una personalización para añadir un botón que invoque al modelo entrenado con id=1. El código necesario es una única línea en la que indicar como parámetro el id del modelo.

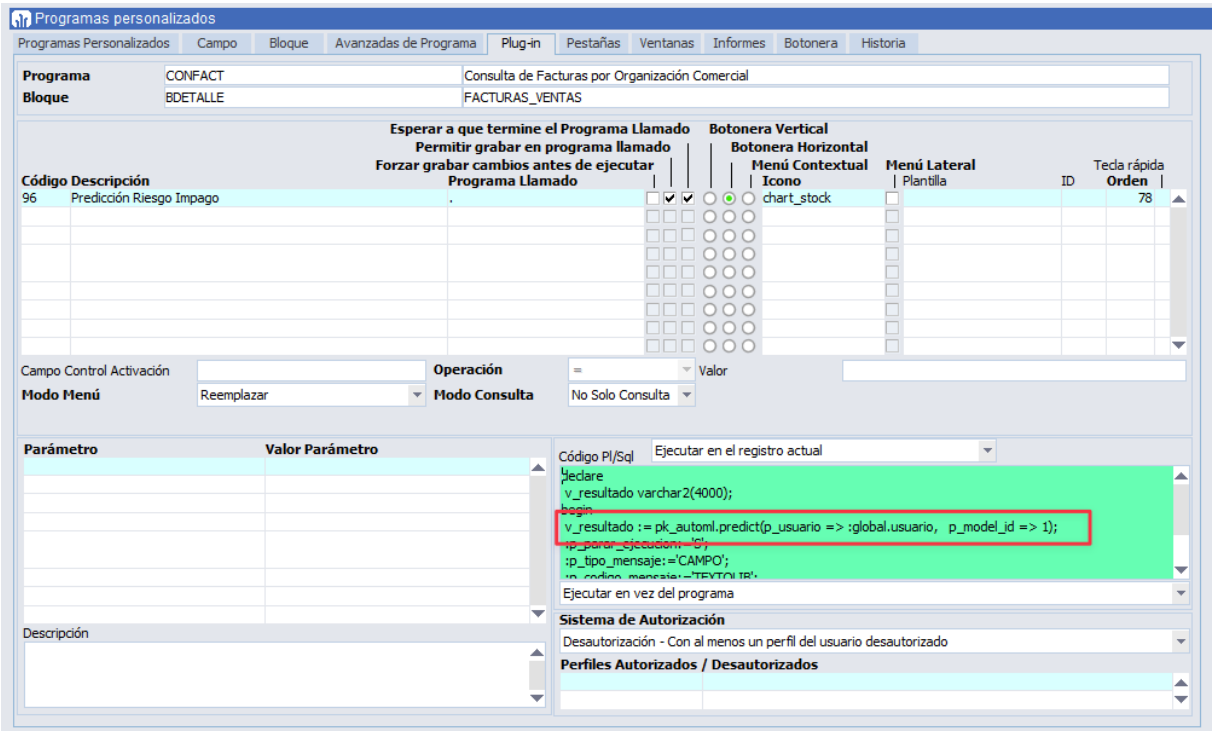


Figura 26. Personalización LIBRA para consumo del modelo

- Para probar la personalización, podemos consultar cualquier factura de Libra y pulsar el botón “Predicción Riesgo Impago” (Figura 27). Obteniendo un mensaje con el resultado de la invocación al API ***predict*** (Figura 28).

Este mismo código podría implementarse en cualquier punto necesario del flujo de funcionamiento del ERP. Por ejemplo podría ejecutarse antes de la contabilización de una factura de venta, para que en caso de superar cierto límite de riesgo de impago, lo tenga que autorizar algún usuario con permisos elevados.

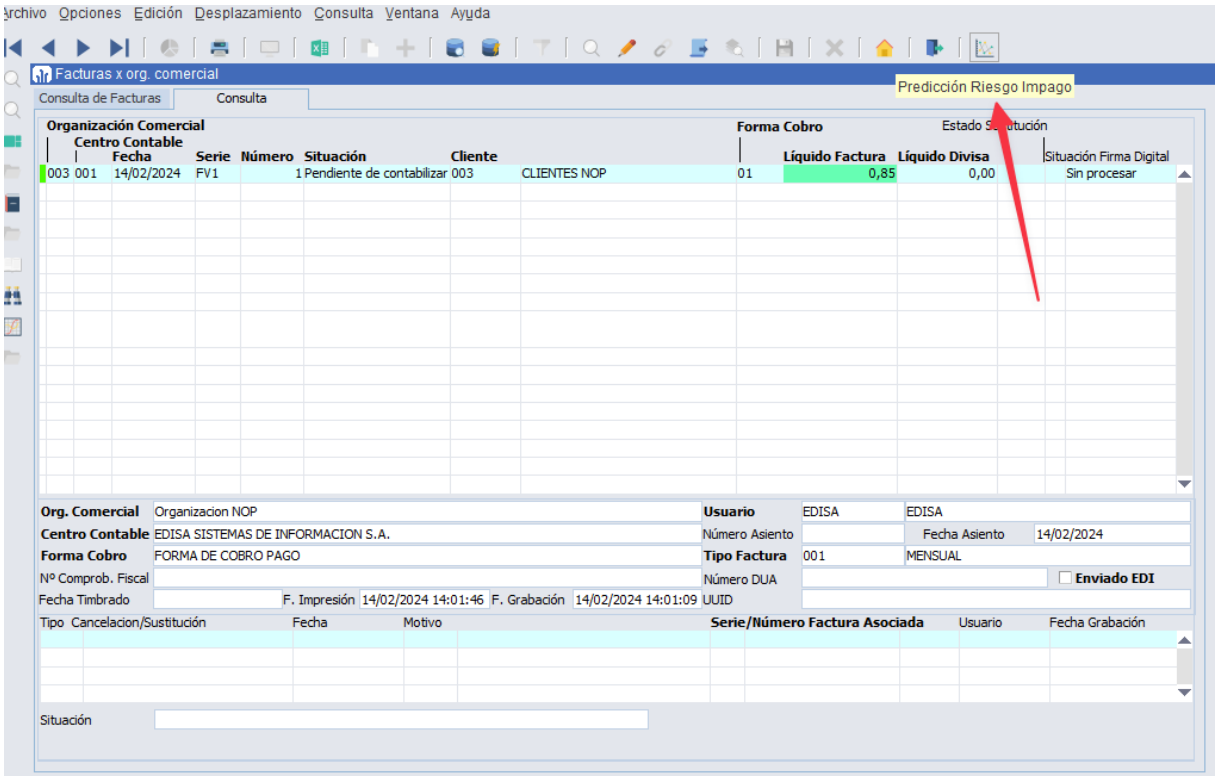


Figura 27. Prueba de consumo de modelo entrenado.

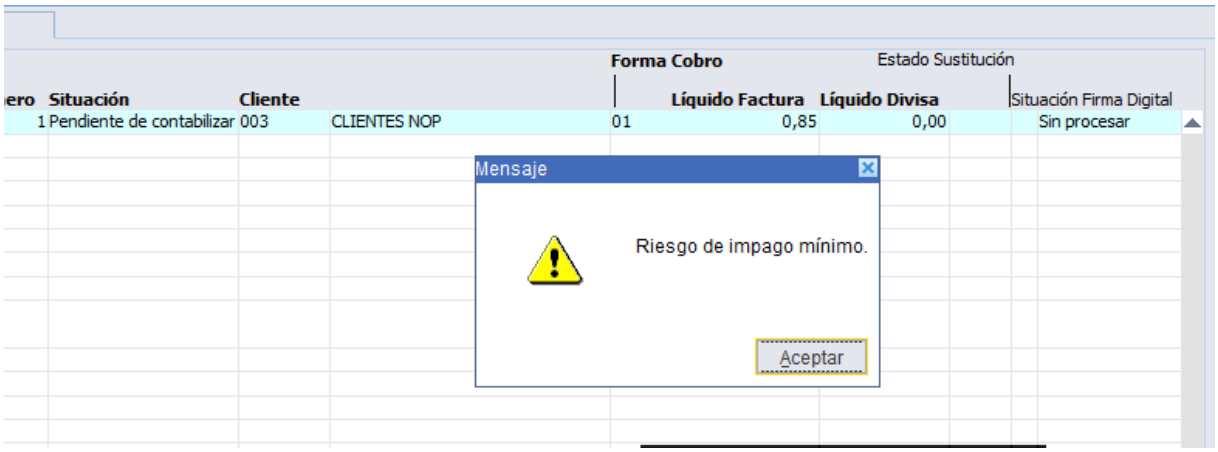


Figura 28. Resultado de la prueba de predicción

5. Conclusiones y trabajo futuro

5.1. Conclusiones

La integración de aprendizaje automático en sistemas ERP representa una evolución significativa en el uso de estos sistemas, permitiendo a las empresas mejorar su eficiencia y toma de decisiones mediante el análisis de datos. Este trabajo ha demostrado que es posible

desarrollar un cuadro de mando de aprendizaje automático integrado en el ERP LIBRA, utilizando tecnologías AutoML para simplificar la creación y uso de modelos de clasificación.

1. **Viabilidad técnica**: La implementación de soluciones AutoML en el ERP LIBRA ha sido técnicamente viable. Las pruebas realizadas con dos casos de uso distintos (clasificación supervisada y no supervisada) han demostrado que los modelos pueden ser entrenados y utilizados con éxito dentro del entorno ERP.
2. **Facilidad de uso**: La interfaz desarrollada permite a los usuarios avanzados del ERP interactuar con los algoritmos de aprendizaje automático de manera intuitiva y sin necesidad de conocimientos profundos en ML. Esto democratiza el uso de técnicas avanzadas de análisis de datos, haciéndolas accesibles a empresas de todos los tamaños.
3. **Calidad de los modelos**: Los modelos entrenados han mostrado un buen desempeño en términos de precisión y eficiencia, en general comparables a los resultados obtenidos mediante el entrenamiento manual. En particular, los algoritmos de clasificación supervisada como RandomForest y XGBoost han proporcionado altos niveles de accuracy, mientras que K-Means ha sido eficaz para la clasificación no supervisada.
4. **Impacto en la toma de decisiones**: La integración de estos modelos en el ERP permitirá mejorar la toma de decisiones en áreas críticas como la gestión de riesgo o gestión de inventario.
5. **Limitaciones**: Se han identificado algunas limitaciones, como la necesidad de procesar grandes volúmenes de datos y la dependencia de la calidad de los datos iniciales, variando la calidad de los modelos entre cada dataset de forma significativa.

   Además, la implementación de técnicas de AutoML requiere una infraestructura tecnológica adecuada y una integración cuidadosa con el ERP.

   Será necesario formar a los usuarios adecuadamente en el uso de la herramienta, la interpretación de los resultados y las limitaciones de los modelos de machine learning.

5.2. Líneas de trabajo futuro

1. **Añadir más tipos de algoritmos.** Integrar también algoritmos de regresión y detección de anomalías permitirá ampliar el tipo de problemas a abordar por la solución de

AutoML, por ejemplo para la predicción de tendencias financieras, planificación de la demanda o detección de fraude.

2. **Ampliación de casos de uso**: Explorar otros casos de uso dentro del ERP para los algoritmos ya disponibles de clasificación y clustering. Esto permitirá expandir las capacidades de análisis y decisión del ERP.
3. **Mejora de la interfaz de usuario**: Continuar mejorando la interfaz de usuario del cuadro de mando para incluir más opciones de personalización y facilitar aún más el uso de técnicas de ML para usuarios sin experiencia técnica.
4. **Optimización de recursos**: Investigar métodos para optimizar el uso de recursos computacionales durante el entrenamiento de modelos.
5. **Monitoreo y mantenimiento de modelos**: Desarrollar mecanismos para el monitoreo continuo y el mantenimiento de los modelos desplegados, asegurando que los modelos se actualicen con nuevos datos y se ajusten según sea necesario para mantener su precisión y relevancia.
6. **Integración con otras tecnologías emergentes**: Explorar la integración con otras tecnologías emergentes como IoT (Internet of Things) para mejorar la recopilación de datos en tiempo real y con GenAI (Inteligencia Artificial Generativa), planteando un asistente virtual que sea capaz de interactuar con el usuario en lenguaje natural para la generación de datasets y entrenamiento de modelos haciendo uso de las herramientas ya desarrolladas.

Bibliografía

Wieder, B. B. (2006). The impact of ERP systems on firm and business process performance. *Journal of Enterprise Information Management, 19(1)*, 13-29.

Godbole, M. (2023). Revolutionizing Enterprise Resource Planning (ERP) Systems through Artificial Intelligence. *International Numeric Journal of Machine Learning and Robots, 7*(7), 1-15.

Frank, L. (2008). Smooth and flexible ERP migration between both homogeneous and heterogeneous ERP systems/ERP modules. *Proceedings of the 2nd Workshop on 3rd Generation Enterprise Resource Planning Systems, Copenhagen business School*.

Katuu, S. (2020). Enterprise Resource Planning: Past, Present, and Future. *New Review of Information Networking, 25*(1), 37-46.

Jawad, Z. B. (2024). Machine learning-driven optimization of enterprise resource planning (ERP) systems: a comprehensive review. *Beni-Suef University Journal of Basic and Applied Sciences, 13*(1), 1.

Juli, M. (2024). AI-Powered ERP: Revolutionizing Usability and Innovation in Enterprise Resource Planning. *EasyChair*, No.12750.

Josyula, H., & Godbole, M. (2024). Navigating The Future: A Comprehensive Analysis of AI, ML, ERP, And Oracle Integration in Financial Digital Transformation Article. *International Journal of Computer Engineering & Technology*, 61-70.

Wahab, N., & Nor, R. (2023). The Role of Enterprise Resource Planning (ERP) Systems in Facilitating Sustainable Business Practices. *AI, IoT and the Fourth Industrial Revolution Review*, 29-39.

Ularu, E., Puican, F., Suciu, G., Vulpe, A., & Todoran, G. (2013). Mobile Computing and Cloud maturity - Introducing Machine Learning for ERP Configuration Automation. *Informatica Economică, 17*, 40-52.

M3Systems. (Mayo de 2023). *m3.systems*. Obtenido de https://m3.systems/understanding-the-role-of-ai-and-machine-learning-in-erp-systems/

Yathiraju, N. (2022). Investigating the use of an Artificial Intelligence Model in an ERP Cloud-Based System. *International Journal of Electrical, Electronics and Computers*, 1-26.

Kauppala, J. (2022). Classifying customer companies in an enterprise resource planning system using machine learning methods. *Lappeenranta-Lahti University of Technology LUT*.

Qaffas, A. A., Ben HajKacem, M. -A., Nasraoui, O., & Ben Ncir, C. -E. (2023). Explainable Artificial Intelligence Approach for Multi-Criteria ABC Item Classification. *J. Theor. Appl. Electron. Commer.*, 848-866.

Strang, K. D., & Sun, Z. (2022). ERP Staff versus AI recruitment with employment real-time big data. *Discov Artif Intell*.

Stadnicka, D., Sęp, J., Amadio, R., Mazzei, D., Tyrovolas, M., Stylios, C., . . . Navarro, J. (2022). Industrial Needs in the Fields of Artificial Intelligence, Internet of Things and Edge Computing. *Sensors, 22*(12).

Soofi, A. A., & Awan, A. (2017). Classification Techniques in Machine Learning: Applications and Issues. *Journal of Bassic & Applied Sciences*, 459-465.

ElMadany, H., Alfonse, M., & Aref, M. (2021). A Proposed Approach for Production in ERP Systems Using Support Vector Machine algorithm. *International Journal of Intelligent Computing and Information Services*, 49-58.

Ding, B. Y., Huang, B., Wei, J., Tang, Y., & Liu, C. (16 de 11 de 2021). Enterprise Risk Assessment Based on Machine Learning. *Computational Intelligence and Neuroscience*.

Ren, Q., & Wang, J. (2023). Research on Enterprise Digital-Level Classification Based on XGBoost Model. *Sustainability, 15*(2699).

Iqbal, T., Elahi, A., & Shahzad, A. (2022). Exploring Unsupervised Machine Learning Classification Methods for Physiological Stress Detection. *Frontiers in Medical Technology*.

Sastry, S. H., Babu, P., & Prasada, M. S. (2013). Analysys & Prediction of Sales Data in SAP-ERP System using Clustering Algorithms. *International Journal of Computational Science and Information Technology*, 1312.2678.

Ferreira, L., Pilastri, A., Martins, C. M., Pires, P. M., & Cortez, P. (2021). A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost. *International Joint Conference on Neural Networks*, 1-8.

LeDell, E., & Poirier, S. (2020). H2O AutoML: Scalable Automatic Machine Learning. *7th ICML Workshop on Automated Machine Learning*.

Waring, J., Lindvall, C., & Umeton, R. (2020). Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial Intelligence in Medicine*.

Real, E., Liang, C., So, D. R., & Le, Q. V. (2020). AutoML-Zero: Evolving Machine Learning Algorithms From Scrath. *37th International Conference on Machine Learning*.

Gerhart, N., Torres, R., & Giddens, L. (2023). Towards and Agile CRISP-DM Process. *Association for Information Systems*.

Anexo A. Código fuente y datos analizados