INTEGRACIÓN DEL APRENDIZAJE AUTOMÁTICO EN UNA PYME: CASO PRÁCTICO EN DARWINEX

MACHINE LEARNING INTEGRATION IN AN SME: CASE STUDY IN DARWINEX

TRABAJO FIN DE MÁSTER
CURSO 2020-2021

AUTOR
SERGIO GARCÍA ALONSO

DIRECTORES
JUAN LUIS PAVÓN MESTRAS
FRANCISCO JAVIER GARIJO MAZARÍO

MÁSTER EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

INTEGRACIÓN DEL APRENDIZAJE AUTOMÁTICO EN UNA PYME: CASO PRÁCTICO EN DARWINEX

MACHINE LEARNING INTEGRATION IN AN SME: CASE STUDY IN DARWINEX

TRABAJO DE FIN DE MÁSTER EN INGENIERÍA INFORMÁTICA
DEPARTAMENTO DE INGENIERÍA DE SOFTWARE E INTELIGENCIA ARTIFICIAL

AUTOR
SERGIO GARCÍA ALONSO

DIRECTOR
JUAN LUIS PAVÓN MESTRAS
FRANCISCO JAVIER GARIJO MAZARÍO

CONVOCATORIA: JUNIO 2021
CALIFICACIÓN: 10 - SOBRESALIENTE

MÁSTER EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

7 DE JULIO DE 2021

RESUMEN

INTEGRAR EL APRENDIZAJE AUTOMÁTICO EN UNA PYME: CASO PRÁCTICO EN DARWINEX

La implantación de una tecnología nueva, como puede ser el Aprendizaje Automático, puede resultar un reto para una PYME, ya sea por la falta de recursos de personal, tiempo o dinero. Por ello, al plantearlo es necesario tener una guía que permita evaluar y facilitar el proceso de implantación, así como contar con las herramientas que mejor se adapten a las peculiaridades del Aprendizaje Automático. En el presente proyecto se ha realizado una investigación previa para elegir la metodología más acorde a los requisitos del Aprendizaje Automático, la librería necesaria para implementarla que permita agilizar el desarrollo, y la infraestructura que nos permita desarrollar y desplegar el proyecto adecuándose tanto a los requisitos del proyecto, la tecnología, así como a las capacidades de la empresa. Posteriormente, con las herramientas elegidas, se ha realizado el procedimiento de implantación en una PYME realizando el desarrollo de un proyecto según las necesidades marcadas por la empresa. En este desarrollo se ha puesto a prueba la metodología, la librería y la infraestructura utilizando un caso real, lo cual permite evaluar de una manera real las herramientas. Estas herramientas, junto a la metodología, han permitido desarrollar un sistema con unos modelos cuyas métricas han superado las expectativas. En conjunto se tiene una guía que describe como podría ser el proceso de creación del primer proyecto de Aprendizaje Automático en una PYME.

Palabras clave

Aprendizaje Automático, CRISP-ML(Q), PYME, AutoML, PyCaret, Python, SageMaker

ABSTRACT

MACHINE LEARNING INTEGRATION IN AN SMA: PRACTICAL CASE IN DARWINEX

The implementation of a new technology, such as Machine Learning, can be a challenge for an SME due to the lack of resources in terms of employees, time, or money. For this reason, it is necessary to have a guide to evaluate and help with the implementation process, as well as to have the tools that fit to the peculiarities of Machine Learning. In this project, a previous research has been carried out to choose the most suitable methodology for the Machine Learning requirements, the necessary library to implement it in order to speed up the development, and the infrastructure that allows us to develop and deploy the project according to the requirements of the project and the technology, as well as to the company's capabilities. Afterwards, with the chosen tools, the implementation procedure has been carried out in an SME, developing a project according to the needs set by the company. In this development, the methodology, the library, and the infrastructure have been tested using a real case, which allows to evaluate the tools in a real way. These tools, together with the methodology, have allowed the development of a system with models, whose metrics have exceeded expectations. Overall, we have a guide that describes how the process of creating the first Machine Learning project in an SME could be.

Keywords

Machine Learning, CRISP-ML(Q), PYME, AutoML, PyCaret, Python, SageMaker

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ABREVIATURAS

AA: Aprendizaje Automático

AUC: *Area under curve*, Área bajo la curva

BI: *Bussiness Inteligence*, Inteligencia de Negocio

EAM, MAE: Error Absoluto Medio, *Mean Absolute Error*

ECM, MSE: Error Cuadrático Medio, *Mean Squared Error*

ELCM, RMSLE: Error logarítmico cuadrático medio, *Root Mean Square Logarithmic Error*

EPAM, MAPE: Error Porcentual Absoluto Medio, *Mean Absolute Percentage Error*

FN: *False Negatives*, Falsos Negativos

FP: *False Positives, Falsos Positivos*

KNN: *K-Nearest-Neighbor,* K-Vecinos Cercanos

MD: Minería de Datos

MLaaS: *Machine Learning as a Service, Aprendizaje Automático como Servicio*

RN, ANN: Redes Neuronales, *Artifical Neural Network*

RECM, RMSE: Raíz del Error Cuadrático Medio, *Root Mean Square Error*

TN: *True Negatives*, Verdaderos Negativos

TP: *True Positives*, Verdaderos Positivos

Capítulo 1 - Introducción

1.1 Contexto

La adopción del Aprendizaje Automático (AA) y la Inteligencia Artificial (IA) por las empresas españolas es aún reducida, aunque en línea con la media europea. Un 9% de las empresas españolas usan algún tipo de tecnología de IA, frente a un 7% de la media de la UE27. Si nos centramos en los datos de España, y diferenciamos por tamaño de empresa, observamos que un 27% de las grandes empresas (aquellas con más de 249 empleados) incorporan la IA, frente a un 8% en el caso de las PYMES (entre 10 y 249 empleados). Para el caso que aplica de AA, un 4% de las PYMES la incorporan, frente a un 18% de las grandes empresas. La diferencia de capacidad para afrontar nuevos retos o la falta de recursos de las PYMES (equipos y personal cualificado, tiempo necesario o capacidad económica) frente a las grandes empresas está detrás de estas grandes diferencias. [1]

En este proyecto nos centramos en las PYMES puesto que es el entorno empresarial en el que se va a poner a prueba.

Aumentar la conversión de clientes, mejorar los procesos de soporte, personalizar las campañas de marketing, la optimización de procesos es entre otras, alguna de las ventajas que las técnicas de AA pueden aportar a las empresas en general y a las PYMES en particular. Esto se consigue gracias a las características del AA como son la capacidad de detección de patrones en los datos, el tratamiento de grandes volúmenes de datos, la capacidad de automatización o la de mejora continua, pero su integración puede conllevar una serie de retos.

Impacto coste/ beneficio en la empresa

Pensando en su impacto en la empresa, es necesario analizar correctamente como va a afectar a la empresa, que se pretende obtener con este sistema y cuando se considera que su implantación ha sido satisfactoria.

Uso de librerías necesarias

Estas son indispensables para desarrolla de una manera óptima y sencilla cualquier proyecto de AA. Las diferentes librerías de AA facilitan la aplicación de los algoritmos y las diferentes fases que conlleva el proyecto. Algunas de ellas, conocidas como AutoML, son capaces de automatizar el tratamiento y transformación de los datos e incluso la creación y comparación de diferentes modelos. Posteriormente se analizarán algunas de las principales librerías que existen, y se profundizará principalmente en las AutoML por aportar un avance más rápido y sencillo en los primeros pasos en la incorporación del AA en las empresas.

Usar la infraestructura adecuada

Encontrar una infraestructura adecuada para desplegar los modelos, y realizar consultas sobre ellos desde el resto de los procesos de la empresa, es indispensable para que el trabajo realizado no se quede en una mera investigación. La incorporación de estos modelos en los flujos de trabajo de la empresa permitirá aprovechar el potencial del AA al máximo.

Estabilidad de los modelos obtenidos con AA

Los resultados que ofrecen los modelos de AA no son estables, puesto que los datos tienden a cambiar con el tiempo, siendo necesario nuevos entrenamientos, por ello resulta muy importante detallar como se va a mantener o monitorizar el sistema.

Al plantear el desarrollo de un sistema de AA hay que seguir los mismos principios que cualquier proyecto de ingeniera de software, no por ser AA es diferente. Por todo esto se plantea que desarrollar un sistema de AA tiene que ser siguiendo una metodología, utilizando las librerías o *frameworks* adecuados siguiendo el principio de no reinventemos la rueda, y elegir la infraestructura adecuada para su despliegue e integración con el resto del ecosistema tecnológico de la empresa, porque al final si no hay integración e interacción con otros sistemas no podremos obtener ninguno de los beneficios que nos aporta el AA.

Este proyecto describe el proceso de incorporación del AA en una empresa financiera, concretamente Darwinex. La empresa quiere mejorar diferentes procesos

del día a día de diferentes departamentos. La empresa ha considerado positivamente las ventajas que puede aportar el AA y han dado su apoyo para llevar a cabo este proyecto con objeto de poner a prueba la integración de la tecnología AA en la empresa.

El proyecto se ha desarrollado en colaboración con la empresa Darwinex, responsable de la web https://www.darwinex.com. El alumno autor de este proyecto trabaja como *Data Team Leader* en la empresa y conoce a fondo el negocio y los datos de la empresa.

Darwinex es una empresa española fundada en 2012 y que cuenta en 2021 con más de 50 empleados. Perteneciente al mundo Fintech, se considera un proveedor de tecnología que ayuda a los traders a desarrollar sus habilidades y construir un *track record* certificado, o lo que es lo mismo, actúan como certificadores de la calidad de los traders. Esto les permite ofrecer, a través de un marketplace propio, la oportunidad a los traders de captar capital de inversores y cobrar un 15% de comisión por éxito bajo la cobertura legal que ofrecen al estar regulados como bróker y gestora de capitales con el regulador británico, FCA. Por simplificar su funcionamiento, para facilitar el entendimiento del contexto empresarial donde se ha desarrollado este proyecto, Darwinex ofrece dos servicios diferenciados a dos clases de clientes, traders e inversores:

- Un servicio orientado al concepto de trader tradicional, Darwinex es un broker online que permite operar con diversos activos de Forex, Acciones Americanas, Materias Primas, índices y Futuros.
- El segundo servicio orientado a los clientes inversores, a los cuales Darwinex, como gestora de activos financieros, les ofrece un nuevo producto financiero llamado Darwins. Un Darwin no es más que una marca o activo creado en sobre la estrategia de un trader, es decir, un trader que tiene una estrategia puede decidir crear un Darwin sobre ella y permitir a los inversores invertir dinero en él. Es un concepto evolucionado del trading social donde unos usuarios replican la operativa de un usuario que opera. En Darwinex este concepto se ha llevado más allá y haciendo uso de tecnología propietaria se ha envuelto todo en un producto financiero con una

cotización propia y un sistema de control de riesgo para asegurar la inversión de los inversores y su capital.

1.2 Objetivos

Los objetivos del proyecto se han planteado considerando los requisitos académicos y empresariales, recogiendo ambos tipos a continuación.

Objetivo General

- Integrar técnicas de Aprendizaje Automático en una PYME a través del desarrollo de un sistema software y conectándolo al resto del sistema

Objetivos Específicos

Para llevar a cabo la integración del AA en los sistemas de información de la empresa será necesario:

- Junto a la empresa, definir los siguientes casos de negocio con los que evaluar los posibles beneficios para la empresa:
    - Crear un modelo para clasificar a los traders que replique la clasificación actual de la empresa y compararlos (modelo de aprendizaje supervisado de clasificación multiclase)
    - Crear un modelo que estime los ingresos mensuales futuros de la empresa basado en datos pasados (modelo de aprendizaje supervisado de regresión)
    - Crear un modelo que sea capaz de anticipar la conversión de un usuario trader antes de que se produzca (modelo de aprendizaje supervisado de clasificación binaria)
    - Crear un modelo que sea capaz de anticipar la conversión de un usuario inversor antes de que se produzca (modelo de aprendizaje supervisado de clasificación binaria)

    - Crear un modelo de recomendación de Darwins en base a la detección de patrones de compra (modelo de aprendizaje no supervisado basado reglas de asociación)
- Elegir una metodología adecuada para el desarrollo del sistema teniendo en cuenta las peculiaridades del AA.
- Evaluar las librerías existentes para identificar las más adecuadas que faciliten las fases del desarrollo del sistema.
- Encontrar la infraestructura más adecuada para ejecutar el sistema de forma eficiente.

1.3 Plan de trabajo

El plan de trabajo ha sido planificado para realizarse entre diciembre 2020 y mayo 2021, ambos incluidos. Los meses previos se han dedicado a investigación y aprendizaje de las herramientas y los conceptos esenciales de AA y metodologías para desarrollar el proyecto.

Durante todo el proyecto se ha utilizado la metodología CRISP-ML(Q), la cual se documenta y desarrolla en el apartado 2.1 de este documento, por lo que la planificación del trabajo se ha realizado acorde a las fases que esta describe.

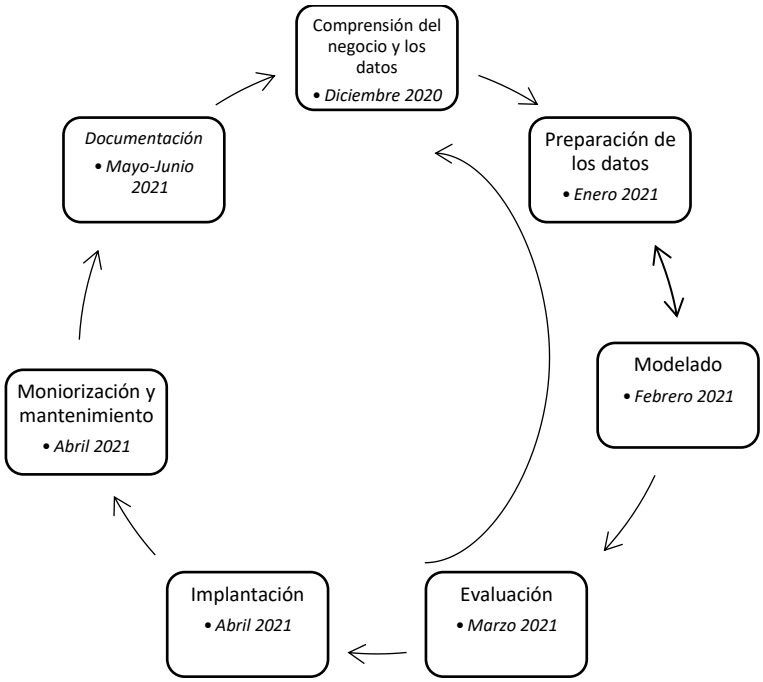


Ilustración 1 - Planificación de fases

Esta planificación es una aproximación contemplando que en cada fase, a partir de la primera, se ha dedicado un tiempo a revisar el trabajo de las fases previas para ver si es necesario realizar algún ajuste en la recopilación o tratamiento de datos, de la creación de los modelos, etc. Por ello, no se contemplan varias iteraciones sobre todo el ciclo puesto que en cualquier momento podremos volver a una fase anterior para aplicar cambios. La documentación del proyecto y el desarrollo del presente documento, aunque se ha efectuado durante todo el proceso, se contempla una última etapa de la planificación para revisión y finalización.

1.4 Estructura del documento

En el capítulo 1 se expresa la motivación del proyecto, el contexto empresarial, los objetivos propuestos, tanto a nivel técnico como empresarial. Se describe la metodología utilizada para el desarrollo del proyecto, analizando algunas de las metodologías para proyectos de AA y justificando la elección que se ha realizado.

En el capítulo 2, se describen los conceptos de AA aplicables al desarrollo del proyecto, se detallan las métricas para evaluar el sistema, las librerías que pueden utilizarse para facilitar el desarrollo, así como las diferentes infraestructuras, más concretamente en la nube, para desplegar y poner en marcha el proyecto.

En el capítulo 3 se detallan las diferentes fases de la metodología para llevar a cabo el desarrollo del proyecto, se describen las tareas y las decisiones tomadas, finalizando con la evaluación de los modelos y las tareas planteadas para mantener y monitorizar el sistema.

Las conclusiones del proyecto y el trabajo futuro se exponen en el capítulo 4. Se analizan el alcance de los resultados obtenidos tanto por las elecciones realizadas de la metodología, las librerías y la infraestructura, se exponen las ventajas y las conclusiones extraídas. Se mencionan finalmente, las líneas de trabajo futuras para mejorar o ampliar este proyecto.

1.5 Código y datos del proyecto

El código desarrollado en este proyecto, y los datos para entrenar los modelos, se encuentran alojados en GitHub, accesible a través de la url *https://github.com/sgavmp/tfm_ml_code* y bajo licencia MIT.

Capítulo 2 - Conceptos y tecnologías aplicables al proyecto

2.1 Elección de la metodología del proyecto

La incorporación de la tecnología AA en entornos empresariales ha dado lugar a un conjunto de metodologías, que han sido analizadas para estudiar la que mejor se adaptan a los requisitos de Darwinex.

2.1.1 CRISP-DM

CRISP- DM es una de las metodologías más utilizada para proyectos de Minería de Datos (MD), y muy utilizada en proyectos de AA [2]. Define un marco de referencia para realizar proyectos de MD y define una serie de fases para llevar a buen término el proyecto. Es una metodología en cascada, contemplando retrocesos de fase, e iterativa, contemplando repetir las fases una vez terminada para conseguir una mejor versión de los resultados. La metodología contempla 6 fases, desde la comprensión del negocio hasta el despliegue, y define en cada una de estas fases una serie de tareas básicas a generar para obtener los resultados correctos, aunque no define como llevar a cabo estas tareas. [3]

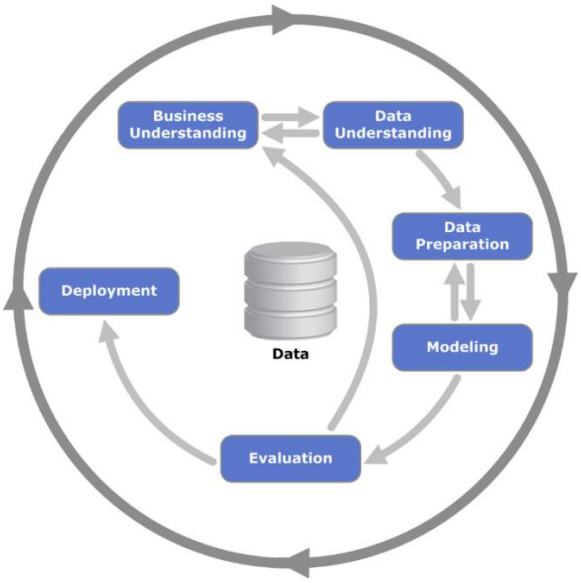


Ilustración 2 - Fases CRISP-DM [4]

2.1.2 Microsoft Software Engineer for Machine Learning

Microsoft analizó [5] la forma de trabajar de los diferentes equipos que han desarrollado aplicando ingeniería de software en el desarrollo de proyecto de AA llegando a crear un marco común de referencia para su implementación. Ha desarrollado una guía [5] con 9 pasos que recogen desde la recopilación de los requisitos del modelo hasta la monitorización del sistema. Comparte pasos y fases con CRISP-DM, añadiendo un punto de monitorización, muy importante en sistemas de AA dada la degradación que se produce en los modelos con el paso del tiempo. Aunque, por otra parte, se ignora la fase inicial de entendimiento del negocio, algo muy importante para entender y comprender correctamente los datos que se van a utilizar en el sistema.

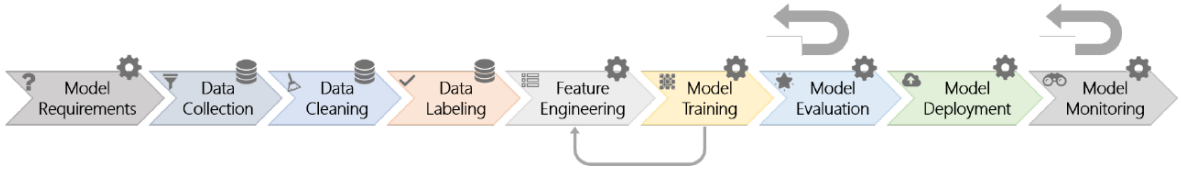


Ilustración 3 - Fases Microsoft SE for ML [5]

2.1.3 Google ML Test Score

Google, buscando formas de mejorar los resultados de los proyectos de AA, ha desarrollado una guía [6] de pruebas para detectar los puntos débiles de los proyectos con el fin de mejorarlos.

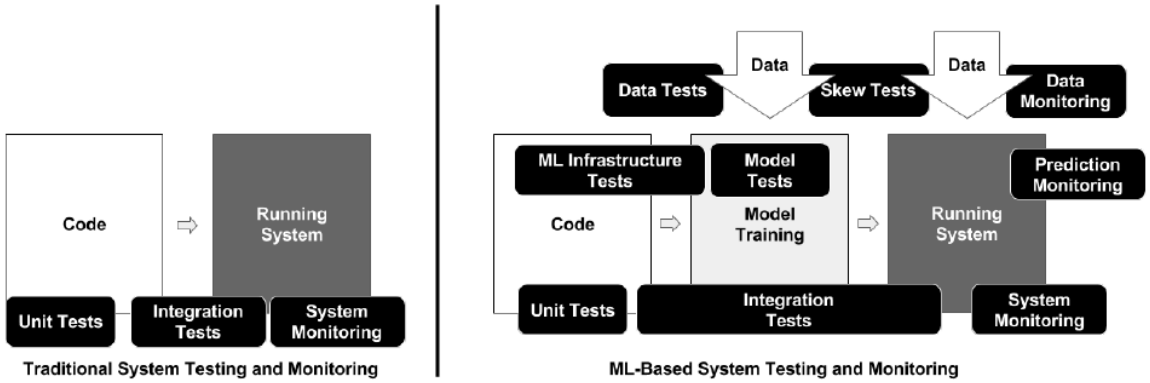


Ilustración 4 - Google ML Test Score VS Test tradicionales [6]

Como ocurre con la guía de Microsoft, aunque contemplan pruebas para todas las fases del proyecto, incluyendo la monitorización del modelo tras desplegarlo, no contempla ninguna prueba para asegurar el entendimiento del negocio.

2.1.4 Metodología elegida: CRISP-ML(Q)

Para el desarrollo del proyecto se ha seguido la metodología CRISP-ML(Q) [7] la cual es una evolución de CRISP-DM, adaptada para proyectos que hagan uso de técnicas de IA y AA. Esta revisión sobre CRISP-DM incluye tareas de control de calidad en cada fase, así como una fase final de monitorización y mantenimiento más pensada para proyectos de desarrollo de aplicaciones como el que aquí se aborda para Darwinex. Comparándolo con las metodologías o guías anteriormente planteadas, CRISP-ML(Q) cubre todas las fases, desde la comprensión del negocio y los datos, hasta la monitorización del modelo desplegado, cubriendo las carencias de las predecesoras.

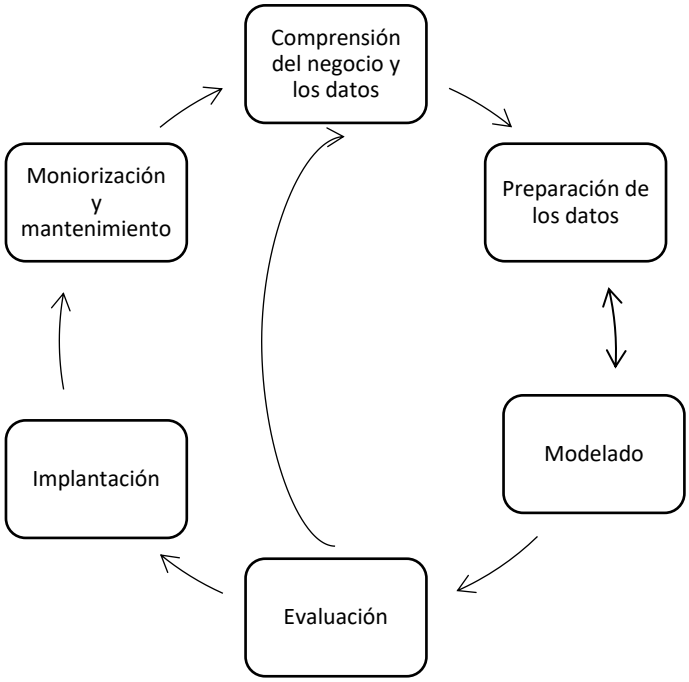


Ilustración 5 - Diagrama de fases de CRIPS-ML(Q). Adaptado de [7]

A continuación, resumimos las fases involucradas:

1. Comprensión del negocio y los datos

En la fase inicial se recogen las actividades orientadas a entender el negocio, los datos y trasladarlos a la definición del sistema de AA. Incluye las actividades para la recolección de los datos iniciales, descubrir los primeros datos o relaciones entre los datos, así como asegurar y verificar la calidad de los datos.

2. Preparación de los datos

A partir de las conclusiones obtenidas de la fase anterior se construye el conjunto o conjuntos de datos finales para ser utilizado en las siguientes fases. Incluye tareas de selección, transformación y limpieza de los datos. No es una fase estática y lo normal es que se itere varias veces sobre ella tras las fases posteriores de modelado o despliegue si se encuentran posibles mejoras.

3. Modelado

La selección de las técnicas de creación del modelo depende del lenguaje, las librerías que queramos utilizar, así como de los objetivos del proyecto y del negocio. En función de las restricciones o requisitos planteados se preseleccionan una o varias técnicas de creación de modelos partiendo de la literatura y se aplican de forma iterativa calibrando los parámetros de entrenamiento a sus valores óptimos. Como ya se ha mencionado, en función de las técnicas elegidas puede ser necesario volver a la fase anterior para adaptar los datos a los requisitos.

4. Evaluación

En esta fase se incluyen las tareas de evaluación, robustez y explicación del modelo. El objetivo es validar la calidad del modelo obtenido y si cumple con los requisitos de completitud y validez del negocio. A la hora de evaluar los modelos hay que tener en cuenta que alcanzar el 100% de precisión es prácticamente imposible por lo que no hay que plantear este como objetivo, aunque sí alcanzar un valor aceptable, teniendo en cuenta el compromiso con otros parámetros, que se describen más adelante en la sección 2.2.3.

5. Implantación

Si la fase anterior ha concluido de forma satisfactoria el proyecto está listo para ser desplegado. Como cualquier aplicación será necesario definir el entorno de despliegue, el control y evaluación del rendimiento en producción, los posibles errores que se produzcan, etc. Puede ser que el proyecto no tenga que ser desplegado porque el objetivo no es desarrollar una aplicación, sino solo analizar los datos, en estos casos esta fase contempla la generación de un informe final que recoja de forma completa todos los resultados obtenidos.

6. Monitorización y mantenimiento

Como cualquier aplicación, tras el despliegue, esta tiene que ser mantenida. Es importante monitorizar el rendimiento de los modelos y evaluar la necesidad de reentrenar o contemplar nuevos conjuntos de datos para mantener la fiabilidad de los modelos desplegados en producción. En esta fase se incluye el mantenimiento requerido por cambios de hardware o software. La infraestructura existente de AA está evolucionando muy rápidamente y puede ser necesario adaptar o crear un modelo desde cero por incompatibilidades.

2.2 Aprendizaje Automático

El AA es la capacidad de un sistema de mejorar sus prestaciones y/o adaptar su funcionalidad durante su funcionamiento y/o utilización. Las técnicas de AA permiten, entre otras cosas, a partir de un conjunto de datos, encontrar patrones o realizar predicciones sin que se requiera una programación específica. [8]

2.2.1 Tipos de AA

Para sentar las bases del AA con las que trabajaremos en el documento, vamos a mencionar de forma resumida la clasificación más típica que se suele aplicar en función del tipo de salida que produce y de cómo se tratan los datos.

2.2.1.1 Aprendizaje Supervisado

Este tipo de sistema aprende a partir de un conjunto de entrenamiento con datos etiquetados, el objetivo del sistema es encontrar el patrón que relaciona los datos del conjunto con la etiqueta. Esta etiqueta, o variable de salida, puede ser de tipo categórico, siendo binario o multiclase, o numérico.

Cuando la variable de salida es de tipo categórico, decimos que es un sistema de aprendizaje supervisado de clasificación, puesto que el sistema lo que intentará es hallar el patrón que relaciona a cada entrada de datos con la clase correspondiente. Como ya hemos indicado antes, la clase a averiguar puede ser binaria, por ejemplo, Si/No, o multiclase, en función del número de clases podremos aplicar unos algoritmos u otros.

En el caso de que la variable objetivo sea de tipo numérico, decimos que se trata de un sistema supervisado de regresión.

Algunos de los algoritmos que podemos utilizar para este tipo de problemas son: *decision trees*, *random forest* y *Support Vector Machines*. [9]

2.2.1.2 Aprendizaje No-Supervisado

Al contrario que en los sistemas supervisados, en los no-supervisados no hay una variable objetivo, es decir, a diferencia del caso anterior, los datos no están etiquetados. En estos sistemas el objetivo es identificar patrones a partir de las

deducciones y las relaciones entre las variables del conjunto de entrada. Podemos utilizarlo para encontrar reglas que definan grupos de datos o para reducir la dimensionalidad de estructura de datos, pero manteniendo sus características fundamentales [10] por ejemplo. Uno de los algoritmos más utilizado es el *K-means* para descubrir agrupaciones en un conjunto de datos.

En la siguiente imagen podemos ver de manera visual un ejemplo de agrupamiento de datos. En la parte izquierda observamos un conjunto de datos, en la derecha observamos cómo se han agrupado en varios grupos por cercanía.

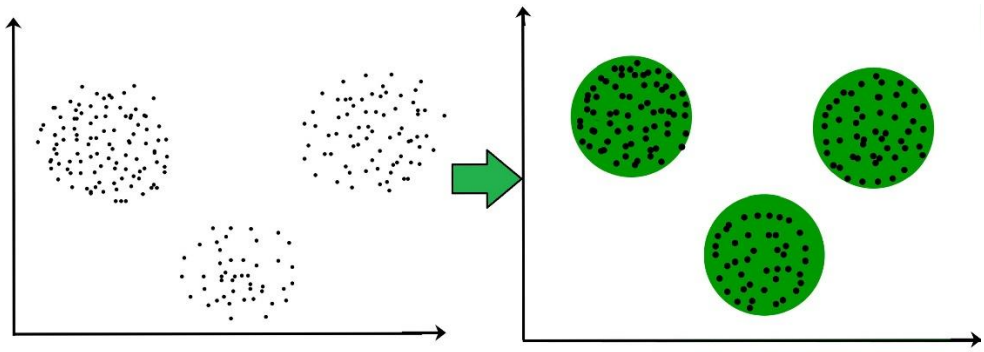


Ilustración 6 - Ejemplo de aprendizaje no supervisado [11]

2.2.1.3 Aprendizaje Semi-Supervisado

Es el resultado de la combinación de los dos tipos anteriores. Obtener un conjunto de datos bueno, limpio y etiquetado puede llevar bastante tiempo. Los sistemas semi-supervisados permiten aumentar la cantidad de datos etiquetados con nuevos datos o mezclar datos etiquetados con no etiquetados. [12]

2.2.1.4 Aprendizaje por refuerzo

Funciona en base a un sistema de recompensas con el fin de mejorar la salida. Similar al funcionamiento de la enseñanza basada en refuerzo positivo que se aplica con los niños, el sistema proporciona una salida y se comprueba si esta es positiva utilizando para ello un sistema auxiliar que es capaz de evaluarla. Este sistema auxiliar proporciona otra entrada al sistema para validar o invalidar la salida previa.

2.2.1.5 Transducción

Muy similar al aprendizaje supervisado, pero en este caso el sistema no crea un modelo para replicar los resultados (método inductivo-deductivo), sino que utiliza los datos proporcionados en la entrada para obtener los resultados para nuevos datos. [10]

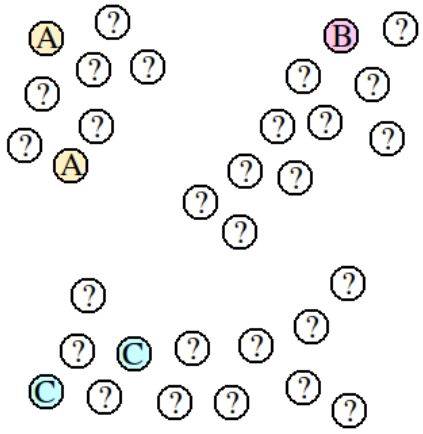


Ilustración 7 - Ejmplo de transducción [13]

En la imagen podemos ver un ejemplo claro de aplicación de transducción. Como se observa en la imagen hay varios grupos bien definidos, pero solo conocemos la clase de unos pocos. El número de ejemplos es muy reducido como para aplicar aprendizaje supervisado, pero podemos utilizar un algoritmo de K-Vecinos Cercanos (KNN) para obtener la clase resultante de los datos desconocidos sin necesidad de crear un modelo de inferencia.

2.2.1.6 Aprendizaje multi-tarea

Engloba todos aquellos métodos de aprendizaje que usan conocimiento previamente aprendido por el sistema, u otros sistemas ya creados, de cara a enfrentarse a problemas parecidos a los ya vistos [10]. Estos sistemas relacionados pueden actuar como una variable más de los datos de entrada. Es muy utilizada en redes neuronales (RN) encadenando varias RN donde cada una resuelve una casuística concreta.

2.2.2 ¿Cuándo usar AA? Ventajas y Desventajas

Aplicar AA es una decisión que deben tomar las empresas en base al beneficio/coste que proporcione su implementación. El AA no es la solución a todos los problemas y en algunos casos hay soluciones más robustas que se pueden aplicar sin recurrir al AA, por ejemplo, no tiene sentido reemplazar un sistema de reglas funcional y estable por un modelo de AA si no se va a conseguir alguna mejora reseñable. [14]

Una de las ventajas que ofrecen los sistemas de AA frente a otras soluciones es su capacidad de detección de patrones en base a la relación que existe entre las variables, ya sean entre sí o con una variable objetivo. Aunque existen técnicas de minería de datos para que una persona pueda realizar este trabajo, la eficiencia y rapidez en aplicarlas no son comparables.

Otra ventaja relacionada con la anterior es la capacidad de manejo de grandes volúmenes de datos. La cantidad de datos que pueden manejar los sistemas está relacionada con la potencia de cómputo y no con el algoritmo de entrenamiento. Cuanto más grande sea el volumen de datos más difícil será aplicar técnicas alternativas al AA dada la dificultad que añade el tamaño del conjunto de los datos para ser revisados o analizados.

Por último, hay que destacar la facilidad de mejora continua de los modelos a partir de la generación de nuevos datos con el paso del tiempo. La disponibilidad de nuevos datos permite entrenar el modelo para mejorar los resultados, y, como veremos más adelante, una vez tengamos desarrollado el proceso de entrenamiento, ampliar el volumen del conjunto de datos original no afecta al resto del proceso. [15]

Como desventaja destacable del AA puede citarse la dificultad de la adquisición de datos, tanto en cantidad, como en calidad. Dependiendo del problema que vayamos a resolver la cantidad de datos de ejemplo que necesitaremos para el entrenamiento puede ser muy alta, dándose el problema quizás de que no tengamos la cantidad de datos suficientes y tengamos que aplicar técnicas de generación de datos. También podría ocurrir que de las variables que tengamos, pocas sean importantes o relevantes para el problema. Al final, crear el conjunto de

datos para entrenar un modelo de AA es un trabajo laborioso, que requiere de muchas pruebas para dar con el correcto.

El tiempo y los recursos son otras de las desventajas en algunos casos, puesto que entrenar un modelo, dependiendo del algoritmo y el tamaño de los datos, puede requerir de mucho tiempo, lo cual se traduce en gastos de recursos. Por suerte hay soluciones en la nube que ofrecen grandes capacidades de cálculo pagando por uso, como por ejemplo AWS Sagemaker que veremos más adelante.

Por último, hay que destacar una de las principales desventajas de estos sistemas, y es la dificultad de explicar e interpretar los resultados. Muchos de los sistemas de AA funcionan como una caja negra, sabemos que dentro se realizan unos cálculos, pero no sabemos realmente cómo han llegado a la salida que nos proporcionan. Depende mucho del algoritmo que elijamos. Algunos algoritmos como el *Decision Tree* y sus variantes permiten generar una representación gráfica del árbol de decisión, pero no suelen ser simples de visualizar ni entender. Es importante tener esto en cuenta cuando nos decidimos por usar el AA en la empresa, puesto que en muchos casos la decisión y los resultados tenemos que ser capaces de explicarlos a responsables que no tienen por qué saber sobre AA y por tanto el elegir sistemas más complejos y que dificultan su explicabilidad puede ser un hándicap. [15]

2.2.3 Métricas de rendimiento

El rendimiento de los sistemas de AA se mide utilizando una serie de métricas que permiten, por un lado, saber el grado de corrección de los resultados, y por otro, comparar diferentes modelos entrenados entre sí para elegir el que consideramos que funciona mejor según la métrica.

Las métricas dependen del tipo de sistema, diferenciando los sistemas supervisados de clasificación, los supervisados de regresión y los no supervisados

2.2.3.1 Métricas de rendimiento para modelos supervisados de clasificación

La mayoría de las métricas están basadas en cálculos sobre la Matriz de Confusión. La matriz de confusión no es más que calcular los Verdaderos Positivos (TP), Verdaderos Negativos (TN), Falsos Positivos (FP) y Falsos Negativos (FN). [16]

		Actual	
		Positivos	Negativo
Predicho	Positivo	TP	FN
	Negativo	FN	TN

Tabla 1- Matriz de confusión. Adaptado de [16]

Esta matriz, aunque no es una métrica, proporciona mucha información sobre el rendimiento del modelo. La matriz representada es para un sistema binario, pero se puede adaptar a cualquier número de categorías.

Sobre la matriz de confusión se calculan las principales métricas para este tipo de modelos:

Exactitud/Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Representa el porcentaje de elementos correctamente clasificados. Es un valor de 0 a 1 y cuanto más alto mejor. Es un valor muy simple pero que aporta mucha información, aunque hay que tener cuidado cuando las clases no están balanceadas, es decir, una de las clases tiene mucha más representación en el volumen de datos de entrenamiento. Esto puede provocar que tengamos una Exactitud/*Accuracy* alta por la clase predominante, pero que para la clase menos representada no acertemos nunca.

Es una opción válida para evaluar los modelos de clasificación con las variables del conjunto de datos y las clases balanceadas entre sí. [17]

Precisión/Precision

$$Precision = \frac{TP}{TP + FP}$$

Es el número total de elementos positivos correctamente asignados entre el número total de positivos calculados, o visto de otra manera, la probabilidad de acertar la clase positiva cuando realmente es positiva. Valor de 0 a 1 y cuanto mayor mejor.

Esta métrica es conveniente cuando queremos asegurarnos de nuestra predicción, es decir, encontrar el mayor número de verdaderos positivos posible. Al centrarnos en esta métrica se deja de lado la exhaustividad. [17]

Exhaustividad/Recall

$$Recall = \frac{TP}{TP + FN}$$

Es el número total de elementos positivos correctamente asignados entre el número total de positivos de la muestra. Al igual que los anteriores, su valor es entre 0 y 1 y cuanto mayor mejor.

Es ideal para asegurarnos encontrar el mayor número de positivos posible. Un ejemplo muy claro es un sistema para predecir si una persona tiene cáncer o no, nos interesa encontrar todos los que darán positivo, aunque demos por positivos casos que no lo tienen. Como ocurre con la Precisión, si nos centramos en esta métrica vamos en detrimento de la anterior. [17]

F1

$$F1 = 2 * \frac{\mathrm{Pr} * Rc}{\mathrm{Pr} + Rc}$$

Esta métrica es una combinación de Precisión y Exhaustividad usando para ello una media aritmética.

Nos permite mantener un balance entre la Precisión y la Exhaustividad para elegir el mejor modelo resultante. Por defecto la formula da la misma importancia a ambas métricas, pero en función de los requisitos del modelo se puede dar más importancia a uno frente al otro. [17]

$$F_\beta = (1 + \beta^2) * \frac{\mathrm{Pr} * Rc}{(\beta^2 * \mathrm{Pr}) + Rc}$$

El valor de β marca cuanto más importante es la Exhaustividad sobre la Precisión.

2.2.3.2 Métricas de rendimiento para modelos supervisados de regresión

Error absoluto medio (EAM/MAE)

Es la media de las diferencias absolutas entre los valores objetivos y las predicciones. Es una media lineal, por lo que no penaliza más a los grandes errores. Tampoco sirve para comparar los resultados entre series con diferentes escalas puesto que el resultado depende directamente de la escala de la serie. [18]

$$MAE = \frac{\sum|x - y|}{n}$$

Error cuadrático medio (ECM/MSE)

Se calcula en base a la media del cuadrado de la diferencia entre los valores objetivos y las predicciones. A diferencia del MAE, este si penaliza los grandes errores. [18]

$$MSE = \frac{\sum(x - y)^2}{n}$$

Raíz del error cuadrático medio (RECM/RMSE)

Se calcula aplicando la raíz al error cuadrático medio, por lo que comparte las mismas características, pero es capaz de reflejar mejor los grandes errores. [18]

$$RMSE = \sqrt{\frac{\sum(x - y)^2}{n}}$$

Error R2

Esta métrica indica como de bueno es el modelo dando un resultado no dependiente de la escala de la serie si no valores entre -∞ y 1. Cuanto más cerca de 1 mejor es el modelo. La forma de calcularlo sería dividir el resultado del MSE del modelo a medir entre un modelo base calculado a partir de las diferencias con la media de la serie. [18]

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)}$$

Error logarítmico cuadrático medio (ELCM/RMSLE)

Similar al RMSE, pero en base logarítmica, por lo que no penaliza demasiado los grandes errores. [18]

$$RMSE = \sqrt{\frac{\sum(\log(x+1) - \log(y+1))^2}{n}}$$

Error porcentual absoluto medio (EPAM/MAPE)

Mide el tamaño del error absoluto en términos porcentuales. Su formulación es similar al MAE, pero ofreciendo un resultado porcentual, por lo que facilita su interpretación. [18]

$$MAPE = \frac{\sum \left|\frac{x-y}{x}\right|}{n}$$

2.3 Librerías de Aprendizaje Automático

La cantidad de librerías y frameworks existentes para AA es abrumadora. Podemos encontrar varias de propósito general como Weka, Scikit-Learn, Shogun, u otras más especializadas en RN y Aprendizaje Profundo como TensorFlow o PyTorch, y aquellas que funcionan como complemento o ayuda como Keras. [19]

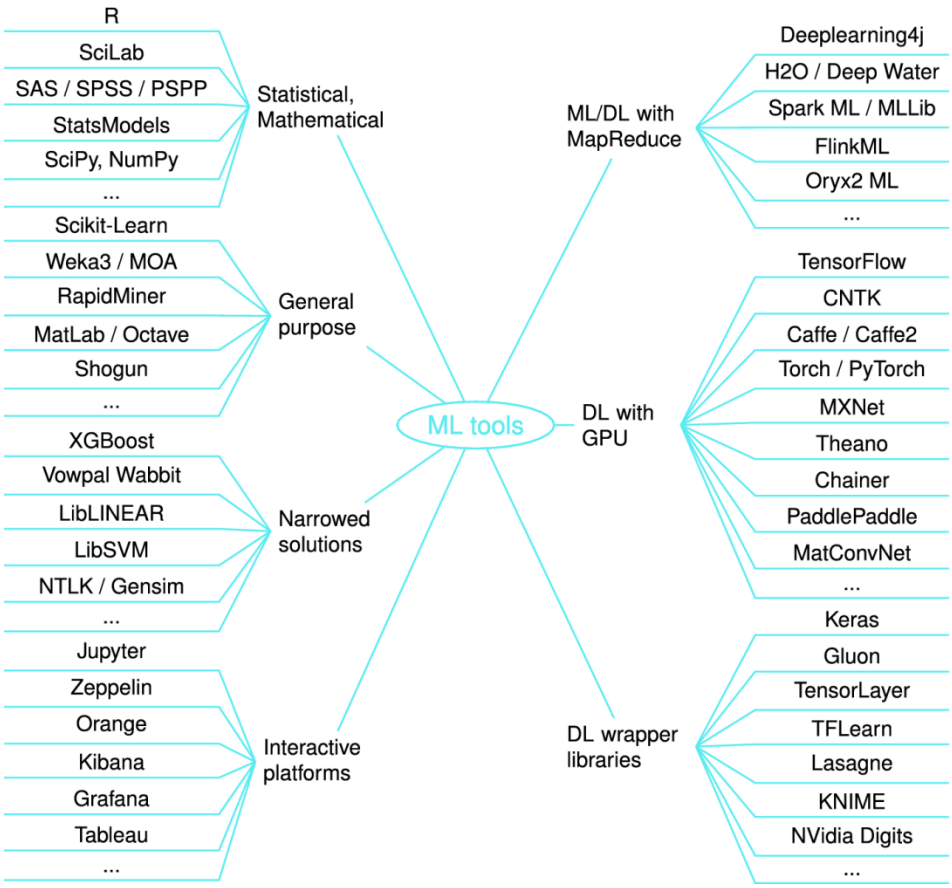


Ilustración 8 - Ejemplos de librerías de AA [20]

La elección de la librería más adecuada depende de los requisitos del proyecto, del conocimiento del equipo que desarrolla el proyecto y de la infraestructura disponible para ejecutarlo.

En el caso del presente proyecto lo más decisivo para elegir una librería ha sido que tenga una curva de aprendizaje corta y que simplifique en la medida de lo posible todo el proceso.

Con esto en mente nos hemos decidido por centrarnos en las librerías de tipo AutoML ya que estas reducen la curva de aprendizaje frente a librerías de AA más

tradicionales, y agilizan y aceleran las diferentes etapas del desarrollo de aplicaciones de AA.

2.3.1 ¿Qué es AutoML?

El objetivo de *AutoML* es facilitar y agilizar el proceso de creación de modelos de AA automatizando algunos de los pasos que se realizan en su creación. Se entiende mejor visualizando algunos de los pasos que se realizan en la creación de los modelos como vemos en la siguiente figura.

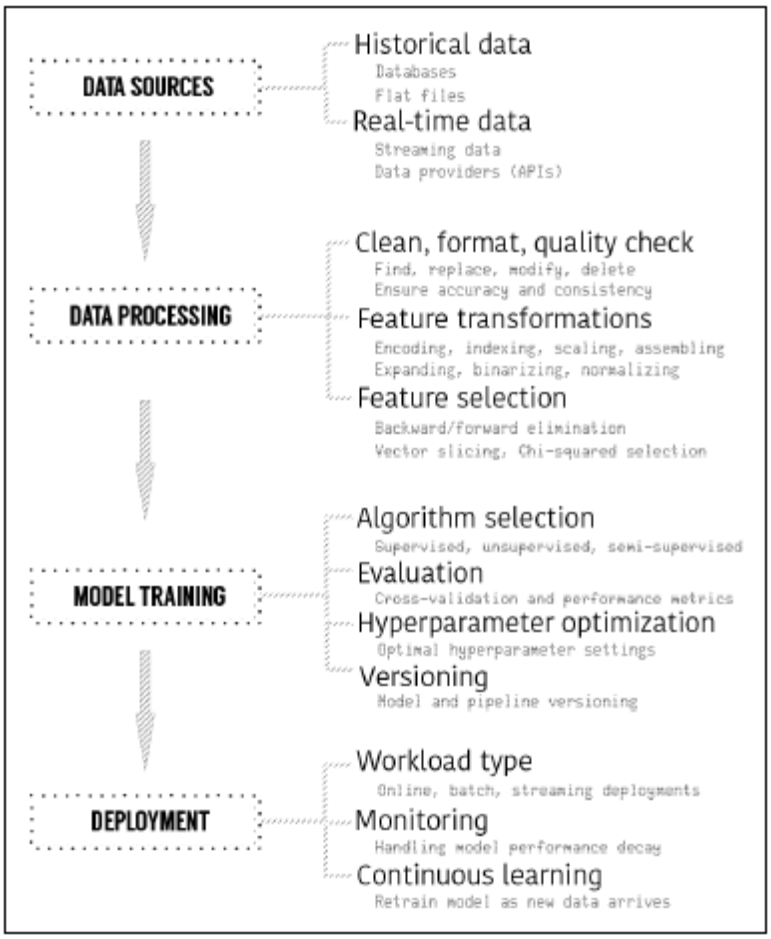


Ilustración 9 – Fases de desarrollo de AA [21]

Las librerías AutoML suelen automatizar las acciones desde la fase de Procesado de Datos. Mediante ciertos parámetros de configuración el sistema es capaz desde limpiar y elegir los datos más adecuados, comparar los diferentes algoritmos existentes para el tipo de problema a resolver, hasta el despliegue final del modelo en alguna plataforma en la nube.

Existen muchas librerías *AutoML* como *Pycaret*, *Auto-Sklearn*, *H2O AutoML*, *TPOT* o *AutoML Tables* de Google.

Dada las diferentes opciones hemos preferido optar por revisar comparaciones que se hayan realizado para basar la decisión final. Concretamente nos hemos basado en el estudio [22] donde se comparan 34 librerías de AutoML, eligiendo la mejor opción, PyCaret.

La selección de PyCaret como mejor opción se basa en que se trata de un proyecto que está siendo mantenido de forma continuada, que ofrece un gran abanico de funcionalidades en cuanto a tipo de problemas a resolver, así como integración con herramientas de Inteligencia de Negocio (BI por sus siglas en inglés) como Power BI.

2.3.2 PyCaret

PyCaret es una librería de AutoML para Python que publicó su primera versión estable, la versión 1.0.0, en abril de 2020, y que, en apenas un año, ha evolucionado a un ritmo muy acelerado, incluyendo nuevas funcionalidades y tipos de problemas para resolver. A fecha de febrero de 2021 fue publicada la versión 2.3 de la librería.

La mejor forma de saber que es PyCaret es acudir a la definición que su web expone:

> *PyCaret is an open source **low-code** machine learning library in Python that aims to reduce the hypothesis to insights cycle time in a ML experiment. It enables data scientists to perform end-to-end experiments quickly and efficiently. In comparison with the other open source machine learning libraries, PyCaret is an alternate **low-code** library that can be used to perform complex machine learning tasks with only few lines of code. PyCaret is **simple and easy to use**. [23]*

Se destaca el texto en negrita, ***low-code***, ***simple*** *y* ***easy to use***, puesto que son estas sus mejores cualidades Su diseño permite realizar todas las fases de creación de un modelo de AA con menos de 10 líneas de código.

Hay que destacar que PyCaret no aplica sus propias implementaciones de los algoritmos, sino que se basa en librerías muy utilizadas que ya han implementado los

algoritmos de AA más comunes, como *scikit-learn*, *XGBoost* u otras, y las envuelve añadiendo las capas de automatización.

2.3.2.1 Funciones

PyCaret sirve como solución integral para 7 tipos de problemas de AA diferentes: Clasificación, Regresión, Agrupación, Detección de Anomalía, Procesamiento del Lenguaje Natural y Reglas de Asociación.

En el presente proyecto nos centraremos en Clasificación, Regresión y Reglas de Asociación.

Dentro de las funciones compartidas para todos los tipos de problemas, PyCaret facilita el trabajo de la división de los datos para entrenar y para validar, proporciona una serie de herramientas, configurable mediante variables, para la preparación de los datos para el entrenamiento de los modelos, normalización y transformación de los datos, tiene opciones para crear y combinar columnas del modelo de datos y aplicar técnicas de selección de características (*Feature Selection*) que permiten, de una forma automatizada, elegir las columnas del conjunto de datos que mejor se comporten en base a algoritmos de selección. Todas estas opciones se configuran vía variables en la inicialización de la librería. Esto permite, con muy pocos y sencillos cambios, probar diferentes opciones hasta dar con la mejor configuración para cada aplicación.

Clasificación

Dentro de la Clasificación, PyCaret permite comparar, para el mismo conjunto de datos, 18 algoritmos diferentes de AA. La librería permite comparar los modelos resultantes del entrenamiento en base a diferentes métricas: Exactitud, AUC, Precisión, Exhaustividad, F1, Kapa. Podemos elegir a partir de qué métrica ordenar los resultados.

Regresión

Para Regresión tenemos la opción de trabajar con 25 algoritmos diferentes y compararlos entre sí para obtener el mejor resultado para cada problema. Las métricas con las que podemos comparar los resultados son: MAE, MSE, RMSE, R2, RMSLE

y MAPE. Podremos elegir aquella métrica más adecuada en función de qué queramos medir.

Reglas de Asociación

En este caso, la librería solo ofrece un algoritmo para extraer las reglas de asociación. En este caso las métricas ofrecidas muestran el ajuste de las reglas sobre el conjunto de datos de entrenamiento.

2.4 Infraestructura de Aprendizaje Automático en la Nube

Esta infraestructura está formada por todas las herramientas que los proveedores en la nube ofrecen para trabajar con AA, ya sea a modo de notebooks ejecutados sobre maquinas especializadas, herramientas de AutoML, algoritmos ya predefinidos para entrenar modelos o servidores específicos donde desplegar los modelos para ejecutar consultas. Obviamente los proveedores más grandes son los que ofrecen más opciones, Azure de Microsoft, Google y AWS de Amazon. Estos servicios son conocidos como Aprendizaje Automático como Servicio (MLaaS, por sus siglas en inglés) puesto que todas ellas ofrecen todas las opciones enmascarando la infraestructura necesaria.

A la hora de elegir el servicio más adecuado habría que ver los requisitos que buscamos, qué necesitamos y, como en muchas empresas, qué servicio estamos utilizando en el momento. En nuestro caso, en Darwinex, nuestro proveedor en la nube es AWS, por lo que esto decanta claramente la decisión, pero aun así hemos querido buscar información sobre las opciones mencionadas anteriormente para elegir aquella opción más adecuada en función de cada caso.

Es muy fácil encontrar en las web comparaciones entre estos servicios como [24], [25] y [26]. De esta última obtenemos la siguiente tabla resumen que compara los diferentes servicios.

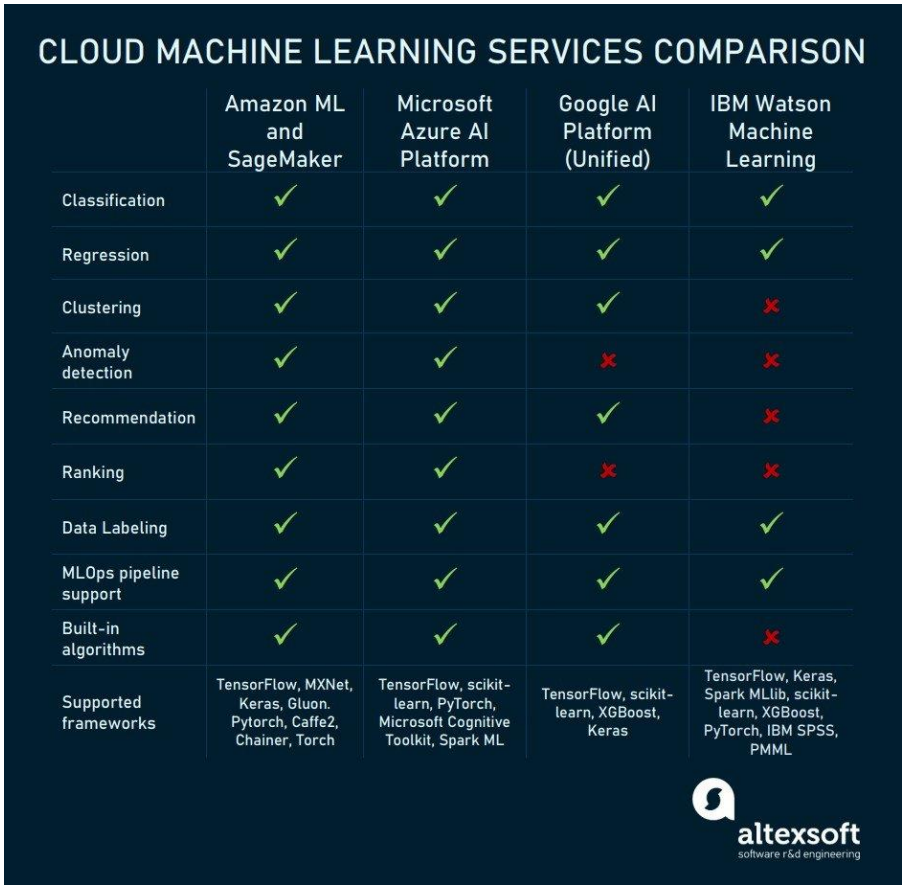


Ilustración 10 - Tabla comparativa de MLaaS [26]

En resumen, todas ofrecen servicios y funcionalidades muy similares, diferenciándose principalmente en los algoritmos que implementan, las librerías que integran y el precio.

En nuestro caso, necesitamos un servicio que permita utilizar PyCaret como librería y nos ofrezca herramientas para automatizar el proceso de entrenamiento y el despliegue de los modelos para integrarlos con el ecosistema de la empresa. Como ya hemos comentado, al usar AWS como proveedor en la nube, la decisión era clara, pero aun así tras revisar las comparativas hemos visto que AWS es la que nos facilita más el proceso de integración de una librería externa como es nuestro caso.

2.5 Herramientas de desarrollo

Para el desarrollo del proyecto se han utilizado las siguientes herramientas:

- **Pycharm**: Entorno de desarrollo o IDE específico para Python de la empresa JetBrains.

- **Jupyter Notebook**: Entorno de desarrollo interactivo para desarrollar en Python, entre otros lenguajes, permitiendo integrar en un mismo documento fragmentos de código ejecutable, texto e imágenes.
- **Sagemakaer Notebook**: Versión propia de Jupyter Notebook que funciona en los servidores de AWS y que nos permite elegir las características técnicas del servidor sobre el que se ejecuta.

Capítulo 3 - Desarrollo del proyecto

El desarrollo se realiza siguiendo las fases de la metodología CRISP-ML(Q). Se detalla a continuación cada fase con las actividades correspondientes.

3.1 Comprensión del negocio y de los datos

En la fase inicial se recogen las actividades orientadas a entender el negocio, los datos y trasladarlos a la definición del sistema de AA. Incluye las actividades para la recolección de los datos iniciales, descubrir los primeros datos o relaciones entre los datos, así como asegurar y verificar la calidad de los datos.

3.1.1 Alcance

A la hora de plantear el proyecto se decidió primero consultar a los departamentos de la empresa por diversas necesidades que tuviesen para ver cuáles de ellas era más adecuada de resolver mediante técnicas de AA. Los departamentos consultados fueron CST, Operaciones y Finanzas. De estas consultas concluimos resolver las siguientes necesidades, lo que viene siendo los requisitos funcionales del sistema en ingeniería de software:

Replicar la clasificación de clientes del sistema actual basado en reglas mediante técnicas de AA

Se ha decidido incluir una funcionalidad para clasificar a usuarios replicando un sistema de clasificación existente basado en reglas. Aunque ya hemos comentado que el AA no es la solución más adecuada cuando existen soluciones más robustas, como un sistema de reglas definidas, hemos considerado oportuno incluirlo en el proyecto para ejemplificar diferentes tipos de problemas, siendo en este caso un sistema de multiclase. Haremos referencia a este requisito, y al modelo resultante, tanto en este documento como en el código como *ModClasClientes.*

Estimación de ingresos futuros

En finanzas es muy recurrente realizar estimaciones de ingresos futuros para planes de inversión, incremento de gastos, etc. Actualmente se hace una estimación

de los ingresos de la empresa basada en modelos matemáticos simples siguiendo la tendencia de los meses previos, pero queremos mejorar esta estimación creando un modelo que se ajuste más a la curva de ingresos Utilizaremos los datos de ingresos de la empresa para crear un modelo que permita completar los datos de una serie temporal, por lo que el periodo temporal a predecir vendrá definido por la entrada de datos no siendo un periodo fijo predeterminado. Usaremos *ModEstIngresos* como nombre del requisito y del modelo resultante.

Potencialidad de transformación de clientes que no han operado

Para el departamento de Soporte a Cliente, poder saber cómo de probable es si un usuario llegará a operar o no puede servirle para optimizar los esfuerzos humanos de atención a clientes y definir de mejor manera las campañas de marketing. En esta funcionalidad diferenciaremos los usuarios que se registran como traders, de los que se registran como inversores. Los datos que utilizaremos serán diferentes para cada tipo de usuario por lo que en la práctica crearemos dos modelos diferentes. Cada uno servirá para predecir la potencialidad de transformación de cada tipo de clientes. Para referenciar a este requisito funcional, así como al modelo resultante, usaremos *ModConversion*, diferenciando el tipo de usuario.

Recomendar patrones de inversión basado Darwins a otros usuarios

Se quiere realizar un sistema de recomendación de inversión en Darwins basado en los Darwins que otros usuarios compran o venden en sus portfolios. La idea es que sea un sistema simple y escalable. Haremos referencia al modelo y a la funcionalidad asociada como ModRecInvDarwin.

Todos estos planteamientos o cuestiones son funcionalidades nuevas que ahora mismo no están resueltas y que, dada las características que hemos visto anteriormente del AA, podemos aplicarlo a resolver estos problemas.

Las funcionalidades aquí planteadas se corresponden con los objetivos empresariales definidos en el proyecto.

3.1.2 Criterios de éxito

En todo proyecto es importante definir unos criterios de éxito o completitud del proyecto. En este caso hace falta definir dos tipos de criterio para cada funcionalidad, el criterio de éxito para el negocio, el cual definirá cuando se considera que el uso de estos modelos es satisfactorio para la empresa, y el criterio de éxito del sistema de AA, aquel que marca el visto bueno desde el aspecto puramente técnico de que el modelo resultante se ajusta correctamente a los datos de entrenamiento y validación.

3.1.2.1 Criterios de éxito para negocio

Junto a negocio se han definido los siguientes criterios para cada funcionalidad definida en el alcance:

- ***ModClasClientes***: El objetivo desde la empresa es evaluar el resultado del modelo frente al sistema actual, comprobando si el sistema es capaz de calificar a los usuarios en los mismos grupos y comparar los criterios de clasificación de ambos. Con estos resultados podemos facilitar el entendimiento de los sistemas de AA haciendo una comparativa directa entre el modelo resultante del AA con el sistema de clasificación existente en la empresa.
- ***ModEstIngresos***: Para la empresa los datos que reporta el modelo serán correctos cuando el error relativo entre el valor obtenido por el modelo de AA y el valor real que se mida de manera posterior para el periodo calculado sea inferior al 10%. Uno de los objetivos de tener una estimación de ingresos es poder ajustar el presupuesto de los diferentes departamentos, por ello necesitamos que entre el valor estimado y el que se produce en realidad no haya demasiada desviación.
- ***ModConversion***: Incrementar un 10% los usuarios que llegan a operar para ambos tipos de usuarios. En este caso el objetivo de la funcionalidad es aumentar el porcentaje de usuarios que tras registrarse llegan a operar por lo que el criterio de éxito será que se produzca un aumento de estos.
- ***ModRecInvDarwin***: En esta funcionalidad el éxito sería medible en base a si los usuarios compran los Darwins que el sistema les recomienda o no. De manera secundaria estas recomendaciones podrían servir para mejorar la

distribución de la inversión en Darwins modificando las recomendaciones para descartar aquellos que tienen más inversión. Puesto que la empresa no realiza ningún tipo de recomendación, ni automática ni manual por parte de empleados, no tenemos otro sistema con el que comparar los resultados.

3.1.2.2 Criterios de éxito para el sistema de AA

De cara al desarrollo del sistema los criterios de éxito difieren con respecto a lo planteado por negocio puesto que se basan en las métricas existentes en cada problema. Para cada funcionalidad se definirá un umbral sobre las métricas elegidas para cada modelo.

- *ModClasClientes*: Se utilizará Exactitud/*Accuracy* con un valor mínimo de 0.9 para considerar resultados favorables. En este caso, lo que queremos es que el sistema sea capaz de clasificar el mayor número de usuarios de manera correcta, y puesto que los datos utilizados están correctamente balanceados, o lo que es lo mismo, tenemos el mismo número de ejemplos de cada clase, los resultados de esa métrica mostrarán claramente el buen funcionamiento del sistema.
- *ModEstIngresos*: Se utilizará MAPE con un valor máximo de 0.25 para obtener resultados favorables. EL utilizar esta métrica es porque al ser representada en % simplifica mucho su entendimiento a simple vista.
- *ModConversion*: Se utilizará Exhaustividad/*Recall* para ambos tipos de usuarios con valores mínimos de 0,9 para considerar los resultados como favorables. El motivo es que queremos que el sistema sea capaz de detectar el mayor número posibles de usuarios que si llegarán a operar, y eso lo logramos centrándonos en esta métrica.
- *ModRecInvDarwin*: Este modelo es diferente a los anteriores, aun siendo AA las métricas son específicas de este algoritmo. De las métricas que ofrece el algoritmo utilizaremos la de Confianza/*Confidence*, con un valor mínimo de 0.6 para cada regla encontrada, y además definiremos que la métrica Soporte/*Support* no sea inferior a 0,1 para poder considerar una regla. La métrica principal es representada como *confidence*(X => Y). Esta métrica viene a representar la probabilidad de *P(Y|X), en nuestro contexto de que cuando* en una cartera de un inversor existan X darwins, siendo X uno o más

darwins, se compre el Darwin Y. La métrica secundaria, soporte/support, representa el número de veces que aparece un conjunto X de darwins en las carteras de los inversores entre el total de carteras de inversores. [27]

3.1.3 Comprobación de posibilidad de desarrollo

Una de las actividades de la metodología seguida es comprobar la viabilidad técnica de los modelos definidos que se quieren crear. Para ello se puede desarrollar una *Prueba de Concepto* que nos permita evaluar si podemos llegar a crear estos modelos de forma satisfactoria, o localizar sistemas o modelos similares que nos permita tener cierta seguridad de que el trabajo realizado podrá resultar en unos modelos funcionales. Los modelos que se van a desarrollar para este proyecto son modelos básicos basados en múltiples ejemplos que podemos encontrar en la web como los que pone de ejemplo la propia librería en [28] o uno más similar al que vamos a desarrollar para el *ModEstIngresos* [29].

3.1.4 Obtención de los datos y verificación de los datos

La obtención de los datos se realizará a partir del almacén de datos central de la empresa y de las bases de datos de Darwinex. Todos los datos necesarios son datos estructurados. A continuación, describimos brevemente los datos que necesitaremos para cada modelo. En el apartado de 3.2 preparación de los datos se darán más detalles del contenido, cantidad y su distribución de cara al entrenamiento.

3.1.4.1 Nota sobre los datos utilizados en el proyecto

Los datos obtenidos en todos los conjuntos de datos serán transformados para poder mostrarse de forma pública sin comprometer la seguridad o privacidad de los usuarios o de la empresa.

3.1.4.2 Datos para obtener el modelo ModClasClientes

Puesto que este modelo pretende replicar la funcionalidad del sistema de clasificación que existe actualmente en Darwinex, se utilizarán los mismos datos para ello. Actualmente los datos resultantes de la clasificación actual se almacenan junto al resto de variables de la clasificación, y disponemos un importante histórico de estos

datos, lo cual nos puede permitir crear un conjunto de datos amplio y bien balanceado entre las diferentes clases.

3.1.4.3 Datos para obtener el modelo ModEstIngresos

Como parte de la estrategia de datos de Darwinex, actualmente tenemos disponibles un histórico diario de todos los conceptos de ingresos que tiene, lo cual nos permite con facilidad crear un conjunto de datos de la evolución de los ingresos. Con los ingresos diarios crearemos un Rolling de 28 días naturales. Se pretende predecir el valor de este Rolling para un día con los datos de los 28 días previos. Los datos utilizados serán los obtenidos desde 2019 hasta marzo de 2021, para evitar cambios de tendencias en los datos por modificaciones internas del producto.

3.1.4.4 Datos para obtener el modelo ModConversion

Partiendo de los datos que se almacenan de los diferentes pasos o acciones que realizan los usuarios, construiremos un conjunto de datos con aquellos que impactan directamente en la conversión de los usuarios. Sabemos que cuanto menos tiempo transcurra entre que un usuario se registra y comienza a realizar las diferentes acciones más posibilidades hay de que se conviertan, por lo que el conjunto de datos contendrá el tiempo, en días, desde la fecha del registro, hasta la fecha de realización de las acciones elegidas.

En función del tipo de usuario, el cual se define en el momento del registro, utilizaremos datos diferentes:

- Traders: Fecha de registro, días hasta empezar el proceso MIFID, días hasta finalizar el proceso MIFID, días hasta el primer depósito, días hasta conectar la primera cuenta desde otro bróker, días hasta crear la primera cuenta de trading en demo, días hasta que opera con la cuenta de trading en demo, días hasta que sube los datos de una cuenta para analizar, días hasta que convirtió y, por último, un indicador de si el usuario ha convertido o no.
- Inversores: Fecha de registro, días hasta empezar el proceso MIFID, días hasta finalizar el proceso MIFID, días hasta el primer depósito, días hasta que creó la primera cuenta de inversor en demo, días hasta que empezó a operar

con la cuenta de inversor en demo, días hasta que convirtió y, por último, un indicador de si el usuario ha convertido o no.

3.1.4.5 Datos para obtener el modelo ModRecInvDarwin

Los datos necesarios para este modelo partirán del histórico de datos de compra/venta de Darwins por parte de los usuarios en inversiones reales. Se utilizarán datos desde 2018 hasta la actualidad, obviando los datos anteriores por diferentes cambios en la lógica interna que pueden afectar a las reglas obtenidas por los algoritmos. De los datos extraídos se obviarán los darwins con mayor inversión para evitar el efecto llamada cuando un Darwin es comprado por muchos inversores. Queremos obtener recomendaciones basadas en la relación entre los darwins, por ello estos darwins que se compran por el simple hecho de tener mucha inversión (si tienen mucha inversión será por algo) pueden afectar a los resultados.

3.2 Preparación de los datos

Dentro de todo sistema de AA hay un proceso de extracción, limpieza, agregación y almacenamiento de los datos para ser procesados de forma posterior por las librerías de entrenamiento. En este proyecto todos estos pasos ya están realizados a nivel de empresa y los datos necesarios para el entrenamiento de los modelos están alojados en el *Data Lake* de la empresa.

Parte del trabajo de esta fase lo realizará la propia librería PyCaret de forma automática: transformación, normalización, selección y creación de nuevas propiedades, balanceo de clases, imputación de datos, etc.

El formato elegido para almacenar todos los conjuntos de datos será en ficheros CSV, y se almacenarán en AWS S3, lo cual facilita el acceso a estos desde la plataforma AWS Sagemaker, que utilizaremos como sistema de procesado.

Para todos los datos se ha generado un perfilado de datos utilizando la librería PandasProfiling [30]. Con este perfilado obtenemos información sobre los datos como la frecuencia de datos nulos, de ceros, la relación entre las distintas columnas, posibles problemas como datos no normalizados o imbalanceados. Con esta información podemos ajustar los datos para corregir posibles problemas antes de usarlos y ajustar los parámetros de la librería.

3.2.1.1 Análisis de datos del modelo ModClasClientes

Como hemos comentado, ya tenemos un sistema basado en reglas que clasifica a los usuarios, por ello utilizaremos los datos históricos que tenemos de este proceso para entrenar el modelo. El archivo que se utiliza contiene 18000 registros, contando con 3000 ejemplos de cada clase, siendo 6 las posibles clases a obtener. Es un archivo con formato CSV y contiene los siguientes datos:

	userid	has_darwin	pfees	darwinia	dscore	commissions	investment	old_darwin	label
0	4acac908-516a-40f3-8d9e-a1a54b59d937	True	329.131762	513.338761	811.833174	6.100919	56.571613	True	2
1	53896cf7-e0ff-4c99-b71b-2640f2bdeb5c	True	329.131762	513.338761	626.675995	6.100919	0.000000	True	3
2	e079ea73-c4c6-4492-9fcf-39e877bd6cc8	True	0.000000	0.000000	0.000000	0.284045	0.000000	True	3
3	7c344536-ee4b-42dc-9731-d344078d0f54	True	0.022516	0.000000	0.000000	2.369245	0.000000	True	3
4	2be80745-12a6-4865-b27a-86f65c03e51a	False	0.000000	0.000000	0.000000	0.113741	0.000000	False	5

Ilustración 11 - Contenido archivo para ModClasClientes

La imagen visualiza las primeras filas del archivo para crear el ModClasClientes. Está compuesto por 9 columnas, siendo la primera, *userid*, un identificador del usuario asociado a los datos, y *label*, la columna objetivo. El resto de las columnas las explicamos a continuación:

Columna	Descripción
has_darwin	Si el usuario tiene o no un Darwin creado en la plataforma. De tipo *Boolean*
pfees	Cantidad pagada a los usuarios con Darwins por parte de los inversores por los beneficios. Tipo *Double*
darwinia	Cantidad pagada a los usuarios con Darwins por parte de los inversores por los beneficios. Tipo *Double*
dscore	Valor del DScore del Darwin del usuario. Este DScore es una nota que se calcula para cada Darwin, nos indica la calidad de este. Tipo *Double*
commissions	Comisiones generadas por el usuario. Tipo *Double*
investment	Inversión en los Darwins del usuario. Tipo *Double*
old_darwin	Este campo, de tipo *Boolean*, indica si el Darwin del usuario tiene más de 6 meses

Tabla 2 - Columnas de los datos para el ModClasClientes

De este archivo utilizaremos un 70% de los datos para el entrenamiento y el 30% restante ara validación. Este porcentaje de datos es el que usa Pycaret por defecto y tras ejecutar varias pruebas modificando estos porcentajes no se han apreciados cambios en los resultados.

3.2.1.2 Análisis de datos del modelo ModEstIngresos

El conjunto de datos utilizado para este modelo es una histórico parcial de ingresos de la empresa

El conjunto de datos está compuesto por los datos de la serie temporal que abarcan desde enero de 2019 hasta enero 2021, siendo un total de 787 días al no incluir los sábados puesto que no hay actividad ese día. Es un dato diario que muestra la evolución de los ingresos con un Rolling de 28 días naturales para cada muestra, este será el dato que queremos predecir. Para predecir el dato utilizaremos los datos de los 28 días previos.

	date	incomes
0	2019-01-16	100000.000000
1	2019-01-17	103618.241851
2	2019-01-18	107010.720018
3	2019-01-20	110127.401465
4	2019-01-21	113058.144967
5	2019-01-22	115916.384413
6	2019-01-23	118670.868615
7	2019-01-24	121518.786440
8	2019-01-25	124185.171231
9	2019-01-27	126628.648569

Ilustración 12 - Contenido archivo ModEstIngresos

Para este modelo utilizaremos los mismos porcentajes para el entrenamiento y la validación que en el modelo *ModClasClientes, 70% y 30% respectivamente.*

3.2.1.3 Análisis de datos de los modelos ModConversion

Para estos modelos utilizaremos dos conjuntos de datos, uno para cada tipo de usuario, TRADER e INVERSOR, con una serie de campos que sabemos influyen en el paso de los usuarios del registro a la conversión. La mayoría de estos campos están

obtenidos del flujo regulatorio de la plataforma, ya que por ley los usuarios tienen que rellenar una serie de formularios y enviar su documentación personal para permitirles operar. Este proceso es uno de los que más fricción causa, junto al primer depósito en la plataforma, por lo que los datos relevantes a cuanto de avanzado están en este proceso y el tiempo que tardan desde que se registran hasta que lo comienzan o finaliza, influyen directamente en la conversión final del usuario. Este proceso regulatorio es el MIFID.

De forma paralela, hay una serie de funcionalidades que los usuarios pueden realizar en la plataforma sin haber terminado el proceso regulatorio, como son la opción de operar con cuentas demo, es decir, con dinero ficticio, tanto como trader o como inversor, conectar y analizar cuentas de otros brokers o subir un archivo con la operativa de una cuenta y analizarlo en nuestra plataforma. Todas estas acciones actúan como impulsores del proceso de conversión y por tanto su uso o no también influye en la conversión.

A continuación, se detallan los campos utilizados.

Columna	Descripción	Tipo
user_currency	Moneda de la cuenta del usuario	AMBOS
user_country	País del usuario (codificado numéricamente)	AMBOS
start_mifid_days	Días transcurridos desde el registro hasta empezar el MIFID	AMBOS
has_finished_mifid	Si ha finalizado todo el proceso de MIFID	AMBOS
finish_mifid_days	Días desde el registro hasta finalizar el MIFID	AMBOS
has_deposit	Si ha depositado dinero en la plataforma	AMBOS
first_deposit_days	Días transcurridos desde que se registra hasta que deposita	AMBOS
first_deposit_amount	Cantidad del primer depósito (normalizada)	AMBOS
first_deposit_platform	Plataforma utilizada en el primer deposito (codificada numéricamente)	AMBOS
mifid_actual_savings	Pregunta del MIFID sobre ahorros (categoría numérica)	AMBOS

mifid_next_year_savings	Pregunta del MIFID sobre ahorros (categoría numérica)	AMBOS
mifid_qualifications	Pregunta del MIFID sobre conocimientos de trading (categoría numérica)	AMBOS
mifid_experience	Pregunta del MIFID sobre conocimientos de trading (categoría numérica)	AMBOS
mifid_money_other_brokers	Pregunta del MIFID sobre si el usuario tiene dinero en otros brokers (1/0)	AMBOS
mifid_invested_other_brokers	Pregunta del MIFID sobre la cantidad que el usuario tiene en otros brokers (categoría numérica)	AMBOS
user_flow_name	Flujo de la web utilizado para registrarse (categoría numérica)	AMBOS
first_trade_investor_account_demo_days	Días transcurridos desde el registro hasta realizar su primera operación como inversor en demo	INVESTOR
days_until_conversion_or_today	Días transcurridos hasta que el usuario operó o hasta el día en curso de los datos	AMBOS
Is_converted	Si el usuario ha operado o no (1/0)	INVESTOR
has_linked_account	Si el usuario ha conectado una cuenta de otro bróker para analizar (1/0)	TRADER
linked_account_days	Días transcurridos desde que se registró hasta que conecto la cuenta	TRADER
has_demo_account	Si tiene cuenta de trader demo (1/0)	TRADER
demo_account_days	Días transcurridos desde que se registró hasta que creo la cuenta demo de trader	TRADER
has_demo_trade	Si ha operado en la cuenta demo (1/0)	TRADER
demo_trade_days	Días transcurridos desde que se registró hasta que realizo el primer trade en la cuenta demo	TRADER
has_mock_account	Si ha subido los datos de una cuenta para analizar (1/0)	TRADER

mock_account_days	Días transcurridos desde que se registró hasta que subió los datos	TRADER
is_converted	Si el usuario ha operado o no (1/0)	TRADER

Tabla 3 - Columnas de los datos para el modelo ModConversion

En total disponemos de 18004 usuarios para el modelo de los usuarios de tipo TRADER, y de 6720 para el modelo de usuarios de tipo INVESTOR. Se usará la misma distribución de porcentajes para el entrenamiento y la validación, 70% y 30% respectivamente.

	user_currency	user_country	start_mifid_days	has_finished_mifid	finish_mifid_days	has_deposit	first_deposit_days	first_deposit_amount
0	EUR	46	0.0	1	1.0	1	1.0	2.0
1	EUR	46	0.0	1	0.0	1	2.0	20.0
2	USD	43	4.0	1	7.0	0	NaN	0.0
3	USD	31	0.0	0	NaN	0	NaN	0.0
4	USD	22	0.0	0	NaN	0	NaN	0.0
5	EUR	36	0.0	1	1.0	0	NaN	0.0
6	EUR	50	0.0	1	1.0	0	NaN	0.0
7	USD	41	0.0	0	NaN	0	NaN	0.0
8	USD	22	0.0	1	0.0	0	NaN	0.0
9	EUR	153	0.0	1	1.0	0	NaN	0.0

Ilustración 13 - Ejemplo de columnas y filas de los datos para el modelo ModConversion de TRADER

3.2.1.4 Análisis de datos del modelo ModRecInvDarwin

El conjunto de datos utilizado es una "foto" realizada con cada cambio del contenido de las carteras de inversión de los inversores, es decir, con cada compra o venta de un Darwin que modifica el portfolio de los usuarios se recopilan los datos del portfolio. Se han obtenido un total de 7049 operaciones de compra/venta de darwins entre el periodo 2018-2021, resultando un conjunto de datos con más de 500000 registros. Este conjunto de datos nos permitirá modelar la relación que existe entre los diferentes Darwins entre sí para ser comprados. Teniendo en cuenta que el contexto puede ser complicado de entender ponemos un ejemplo más cercano al público general para entender el contenido del conjunto de datos.

Supongamos unos tickets de compra que contengan los siguientes artículos.

Ticket 1	Ticket 2	Ticket 3	Ticket 4
Hamburguesas	Pasta	Tomate	Queso
Queso	Tomate	Pasta	Hamburguesas
Pan	Queso	Jamón	Pan
Agua	Jabón	Queso	Patatas
	Lechuga		

Tabla 3 - Ejemplo de tickets de compra

En base a estos datos podemos sacar unas reglas muy básicas de qué artículos recomendar a alguien que este comprando:

- Hamburguesas & Queso = Pan
- Pasta = Queso

Algo similar queremos hacer con los portfolios de los inversores. Cuando los usuarios compran un Darwin en concreto se basa en la relación y en cómo complementa este al resto de su portfolio, por lo que hay una motivación directa para ello, como ocurre con las hamburguesas, el queso y el pan. Suponemos que cada cambio del portfolio (entrada y salida de un Darwin) representa un ticket de compra y en base a esto obtener las reglas de recomendación.

Trasladando el ejemplo anterior al contexto de Darwinex, los datos de entrada quedarían de la siguiente manera:

Portfolio 1	Portfolio 2	Portfolio 3	Portfolio 4
HHZT	FJVM	MVUT	LLFH
LLFH	LLFH	XPWN	HHZT
XPWN	XPWN	FJVM	MVUT
MVUT	MVUT	LLOP	PRTY

Tabla 4 - Ejemplo de portfolios para reglas

Siguiendo el ejemplo anterior podemos sacar, entre otras, las siguientes reglas:

- HHZT y MVUT -> LLFH
- XPWN y MVUT -> FJVM
- MVUT -> XPWN

El conjunto de datos contiene un id de la situación del portfolio, así como los Darwins que tenía en el portfolio tras la compra/venta.

	orderid	darwin
0	8657972536	BBHI
1	655119019	RRQO
2	655119019	WLLK
3	655119019	BJFB
4	655119019	ARTN

Ilustración 14 - Contenido archivo ModRecInvDarwin

En este caso todo el conjunto de datos es utilizado para obtener las reglas, no siendo necesario dividir el conjunto de datos.

3.3 Modelado

La librería PyCaret permite crear modelos a partir de los conjuntos de datos y compararlos entre sí para elegir aquel que de mejores resultados, esto facilita la elección del mejor modelo.

El notebook utilizado para este proceso en cada modelo se encuentra en el repositorio de código enlazado [31].

3.3.1 Configuración

Para cada tipo de problema, la forma de inicializar Pycaret es diferente, aunque luego las funciones a utilizar son comunes o similares. Lo primero que se hace es importar la librería de Pycaret necesaria para cada tipo de problema:

- *from pycaret.classification import **
    - Para aquellos problemas de clasificación, independientemente del número de clases a clasificar. Este lo usaremos para los modelos *ModClasClientes* y *ModConversion*.
- *from pycaret.regression import **
    - Para aquellos problemas de regresión para el cálculo de una variable numérica, como es el caso del modelo *ModEstIngresos*.
- *from pycaret.arules import **
    - Para los problemas de búsqueda de reglas, como el modelo *ModRecInvDarwin*.

Una vez importado el módulo correspondiente, toca inicializarlo, para ello se utiliza la función *setup* [32]. Dependiendo del módulo cargado, es decir, del tipo de problema, tendrá diferentes opciones de inicialización para ajustar el sistema a los datos y a los requisitos del modelo que queremos obtener.

```
clf1 = setup(dataset_raw, normalize=False, target = 'label', ignore_features=['userid'], silent=True)
```

Ilustración 15 - Ejemplo de setup para modelo 1

3.3.2 Comparación

Excepto para el modelo *ModRecInvDarwin*, que solo ofrece un algoritmo de entrenamiento, para el resto de los problemas, Pycaret implementa múltiples algoritmos, y permite comparar los resultados de todos ellos con una sola función, *compare_models* [33].

Con esta función Pycaret realizará el entrenamiento, utilizando los datos suministrados en el *setup,* y los usará con cada algoritmo, mostrando una tabla con los resultados obtenidos y ordenándolos según la métrica indicada, por defecto Exactitud/*Accuracy*.

```
top = compare_models(n_select=1)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dt	Decision Tree Classifier	0.9999	1.0000	0.9999	0.9999	0.9999	0.9999	0.9999	0.0140
rf	Random Forest Classifier	0.9998	1.0000	0.9998	0.9998	0.9998	0.9998	0.9998	0.4180
gbc	Gradient Boosting Classifier	0.9998	1.0000	0.9998	0.9998	0.9998	0.9998	0.9998	2.6690
lightgbm	Light Gradient Boosting Machine	0.9988	0.9999	0.9988	0.9988	0.9988	0.9986	0.9986	0.3810
et	Extra Trees Classifier	0.9983	1.0000	0.9983	0.9983	0.9983	0.9980	0.9980	0.3690
knn	K Neighbors Classifier	0.9600	0.9946	0.9601	0.9608	0.9600	0.9520	0.9521	0.0980
lr	Logistic Regression	0.8541	0.9780	0.8534	0.8578	0.8516	0.8249	0.8269	1.6960
nb	Naive Bayes	0.7647	0.9624	0.7633	0.7619	0.7527	0.7176	0.7219	0.0120
svm	SVM - Linear Kernel	0.7031	0.0000	0.7018	0.7595	0.6967	0.6438	0.6612	0.1750
lda	Linear Discriminant Analysis	0.6668	0.9166	0.6643	0.7266	0.6019	0.5998	0.6215	0.0150
ridge	Ridge Classifier	0.6351	0.0000	0.6325	0.6857	0.5529	0.5618	0.5882	0.0090
ada	Ada Boost Classifier	0.5508	0.8427	0.5533	0.5572	0.4947	0.4609	0.5106	0.1840
qda	Quadratic Discriminant Analysis	0.1666	0.0000	0.1667	0.0278	0.0476	0.0000	0.0000	0.0100

Ilustración 16 - Ejemplo para el modelo ModClasClientes

Además, esta función devolverá el modelo resultante de aquel algoritmo que ofrezca mejor resultado según la métrica utilizada.

3.3.3 Crear modelo único

Tanto para el modelo *ModRecInvDarwin* de reglas, como para el resto de los problemas, Pycaret ofrece la función *create_model* [34] que nos permite crear un modelo usando un algoritmo específico para cada módulo. En el caso del modelo *ModRecInvDarwin*, al no haber múltiples algoritmos, con la función se crearía el único modelo posible.

```
model = create_model(min_support=0.1)
```

Ilustración 17 - Ejemplo de llamada a create_model del modelo ModRecInvDarwin

3.3.4 Ajuste de hiperparámetros

Una vez obtenido el modelo base, podemos hacer uso de la función *tune_model* [35] para ajustar los hiperparámetros del modelo. Los hiperparámetros son los parámetros de configuración de cada algoritmo al ejecutar el proceso de entrenamiento. Si estuviésemos utilizando los algoritmos fuera de Pycaret, tendríamos que ejecutar un proceso iterativo de mejora de estos hiperparámetros para cada algoritmo, pero Pycaret ya ofrece esta funcionalidad de manera automática.

```
top_tuned = tune_model(top)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.9992	0.9995	0.9992	0.9992	0.9992	0.9990	0.9990
2	0.9976	1.0000	0.9976	0.9977	0.9976	0.9971	0.9971
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
5	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
6	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
7	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
8	0.9984	1.0000	0.9984	0.9984	0.9984	0.9981	0.9981
9	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
Mean	0.9991	1.0000	0.9991	0.9991	0.9991	0.9990	0.9990
SD	0.0007	0.0001	0.0007	0.0007	0.0007	0.0008	0.0008

Ilustración 18 - Ejemplo tune_model para ModClasClientes

Al igual que ocurre con la función de comparar, se devuelve una tabla con los resultados obtenidos en las diferentes iteraciones que se ejecutan para optimizar la métrica elegida, por defecto Exactitud/Accuracy.

3.3.5 Calibrado

En los problemas de clasificación, podemos hacer que el modelo devuelva una métrica de probabilidad de la clase elegida. Esto nos permitirá después saber la exactitud con la que tratar el resultado de cada clasificación. Para ello hay que ejecutar la función *calibrate_model* [36].

```
calibrated_dt = calibrate_model(top_tuned)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.9992	0.9995	0.9992	0.9992	0.9992	0.9990	0.9990
2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
5	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
6	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
7	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
8	0.9984	1.0000	0.9984	0.9984	0.9984	0.9981	0.9981
9	0.9992	1.0000	0.9992	0.9992	0.9992	0.9990	0.9990
Mean	0.9994	1.0000	0.9994	0.9994	0.9994	0.9992	0.9992
SD	0.0005	0.0001	0.0005	0.0005	0.0005	0.0006	0.0006

Ilustración 19 - Ejemplo de calibrate_model para el modelo ModClasClientes

3.3.6 Evaluación y visualización de resultados

Para los módulos de clasificación y regresión, Pycaret ofrece una función que genera un HTML interactivo para visualizar las diferentes graficas de los resultados, *interpret_model* [37]. Aun así, tanto para estos módulos como para el resto también podemos usar la función *plot_model* [38], que nos permite visualizar el rendimiento de los modelos con diferentes graficas como el are bajo la curva (*auc*), matriz de confusión (*confusión_matrix*), error de las clases predichas (*error*) o la importancia de cada columna de datos (*feature_importance*).

```
plot_model(calibrated_dt, plot='learning')
```

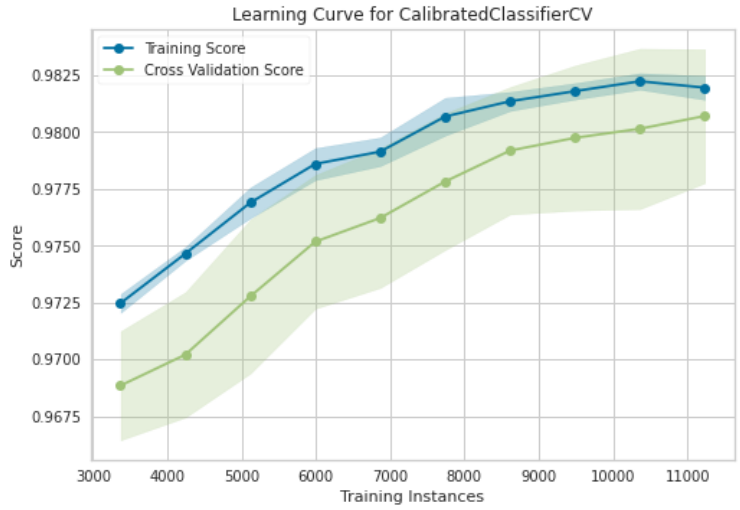


Ilustración 20 - Ejemplo de plot_model de la curva de aprendizaje para el modelo ModConversion de TRADER

```
plot_model(top_tuned, plot='feature')
```

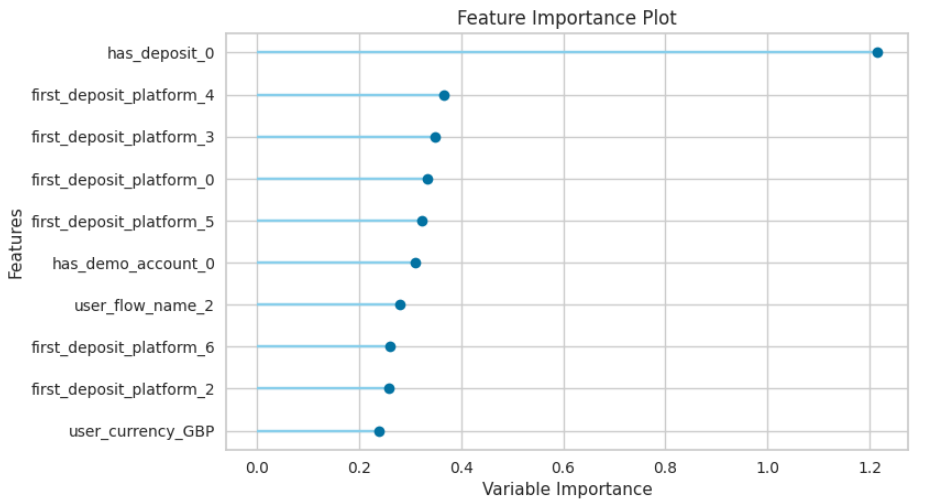


Ilustración 21 - Ejemplo de la importancia de cada columna en el modelo ModConversion de TRADER

Ilustración 22 - Ejemplo de la matriz de confusión del modelo ModClasClientes

3.3.7 Finalizar modelo

Para los módulos de clasificación y regresión, los datos facilitados en el *setup* se dividen en dos conjuntos de datos, un 70% de los datos se utilizan para entrenar el modelo, y el resto se usan para verificar la calidad del modelo. Este 70% de datos para entrenamiento es que establece por defecto Pycaret, y tras varias pruebas variando este porcentaje no se han obtenido cambios en los resultados, por lo que se decidió dejarlo en 70%. Por ello, tras validar el modelo tenemos la opción de utilizar la función *finalize_model* [39] para que se use el 100% de los datos en el entrenamiento antes de guardar el modelo.

3.3.8 Implementaciones

3.3.8.1 Implementación del modelo ModClasClientes

Para crear este modelo vamos a utilizar el módulo de Pycaret de Clasificación [40]. Tras ejecutar las funciones mencionadas anteriormente ajustadas para este modelo obtenemos que el algoritmo que mejor resultado ofrece es el Árbol de Decisión (*Decision Trees*). Se ha utilizado la métrica de Exactitud/*Accuraccy* puesto que los datos de entrenamiento están bien balanceados y por tanto esta métrica no

ofrecerá resultados engañosos. Las métricas obtenidas y las gráficas de rendimiento se detallarán en el siguiente punto.

3.3.8.2 Implementación del modelo ModEstIngresos

En este modelo utilizaremos el módulo de regresión de Pycaret. Usaremos la métrica RMSLE para la comparación de los diferentes algoritmos en la fase de creación del modelo, y la métrica R2, la cuales, por defecto, en la optimización de los hiperparámetros del modelo. El algoritmo que mejor resultado da es Regresión Lineal (*Linear Regression*). El resto de las métricas y las gráficas del rendimiento del modelo se verán en el siguiente punto.

3.3.8.3 Implementación de los modelos ModConversion

En este caso generamos dos modelos diferentes, uno para cada tipo de usuario de Darwinex, TRADER o INVESTOR. Para generar cada modelo tenemos dos conjuntos de datos diferentes y por tanto hay dos procesos de entrenamiento y análisis de resultados. Para ambos casos utilizamos le métrica Exhaustividad/*Recall* para optimizar que el modelo se capaz de clasificar el mayor número posible de usuarios que se convierten, es decir, que operan. Para la optimización de los hiperparámetros usaremos la misma métrica. Para ambos modelos, el algoritmo que mejor resultado ha dado es *Ridge*, esto se debe a que los conjuntos de datos utilizados comparten la mayoría de las columnas y tipo de datos. Como en el resto de los modelos, veremos en más profundidad las métricas y los resultados en el siguiente punto.

3.3.8.4 Implementación del modelo ModRecInvDarwin

Como ya hemos comentado en varias ocasiones, para este modelo utilizaremos el módulo de reglas de Pycaret, el cual ofrece solo un algoritmo, Apriori [41]. Es un algoritmo muy conocido para extracción de reglas asociativas. A la hora de crear el modelo se ha modificado el umbral inferior que limita las reglas a considerar a aquellas que aparezcan en el 10% de las carteras, estando por defecto en el 5%, el resto de los parámetros no se han modificado. Se revisarán los resultados y las reglas encontradas en el siguiente punto.

3.4 Evaluación

En esta fase se evalúa y comprueba la robustez y calidad de los resultados obtenidos por lo modelos. Además, comprobamos el cumplimiento de los criterios técnicos y de negocios establecidos para decidir si se despliega o no los modelos.

Junto a las tablas con las métricas finales de cada modelo, se han incluido algunas visualizaciones para reforzar los resultados obtenidos.

3.4.1 Evaluación del modelo ModClasClientes

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Ilustración 23 - Resultados del modelo ModClasClientes

Los resultados de estos modelos son prácticamente perfectos. Las métricas obtenidas tras el entrenamiento y los ajustes son de 1 en todas. Estos resultados de podrían hacer pensar en un sobreajuste con los datos del entrenamiento, pero en realidad se debe a la similitud de los datos de entrada. Los datos utilizados para entrenar este modelo son datos que consideramos estables, es decir, no tienen valores extremos y se distribuyen muy bien en una campana de Gauss, esto hace que, aunque aumentemos la cantidad de datos de entrenamiento de los 18000 actuales a 180000 o 400000, el resultado será el mismo.

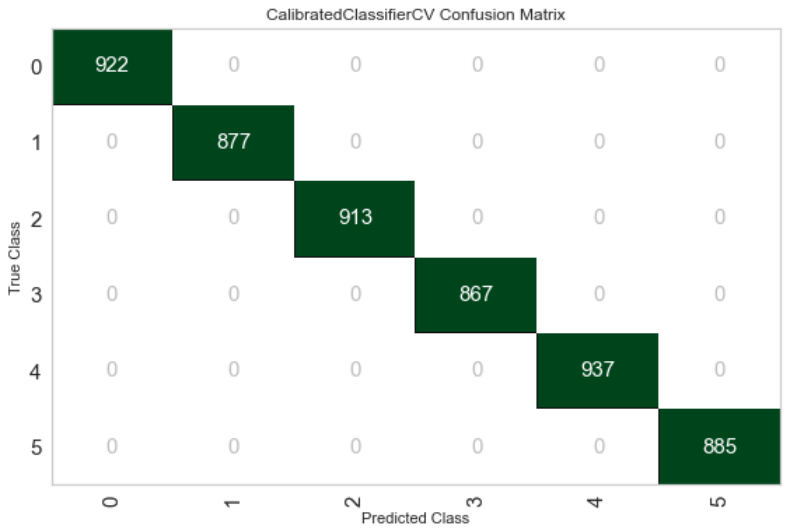


Ilustración 24 - Matriz de confusión del modelo ModClasClientes

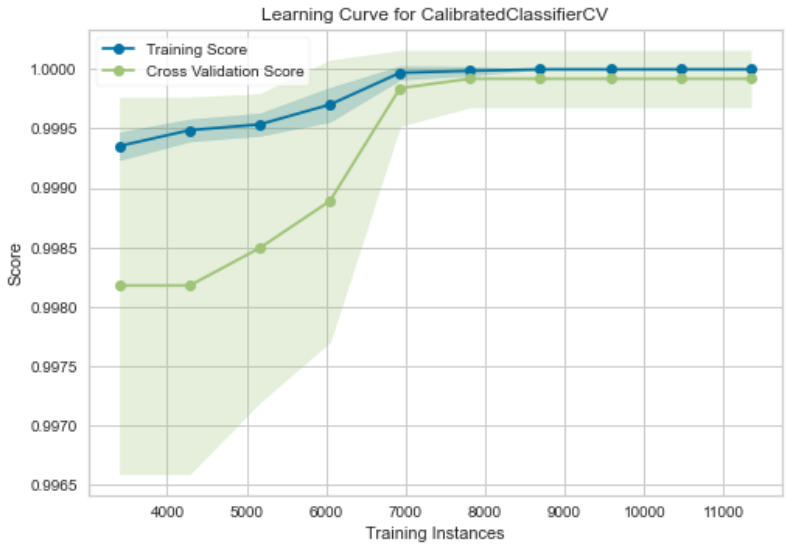


Ilustración 25 - Curva de aprendizaje del modelo ModClasClientes

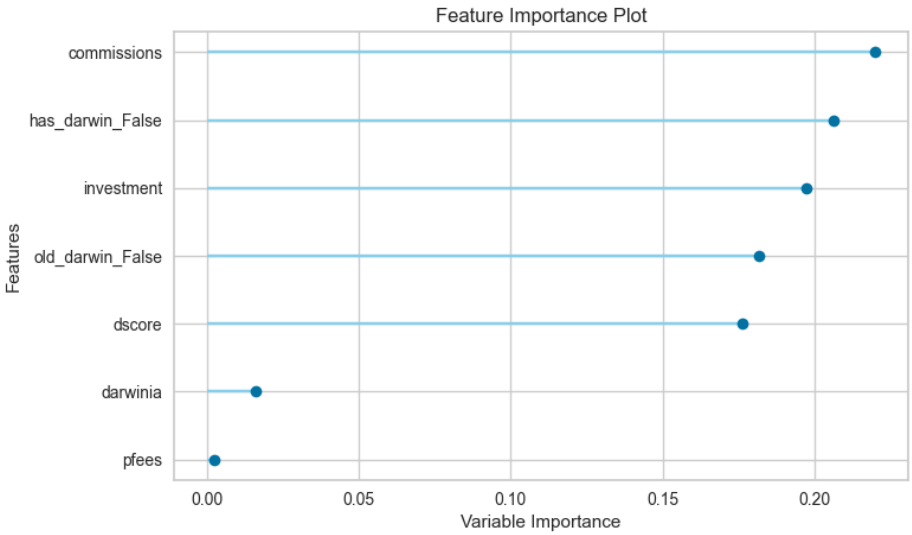


Ilustración 26 - Importancia de las columnas del modelo ModClasClientes

Con la representación del árbol de decisión resultante podemos comparar los criterios de cada clase con los que se aplican actualmente en Darwinex.

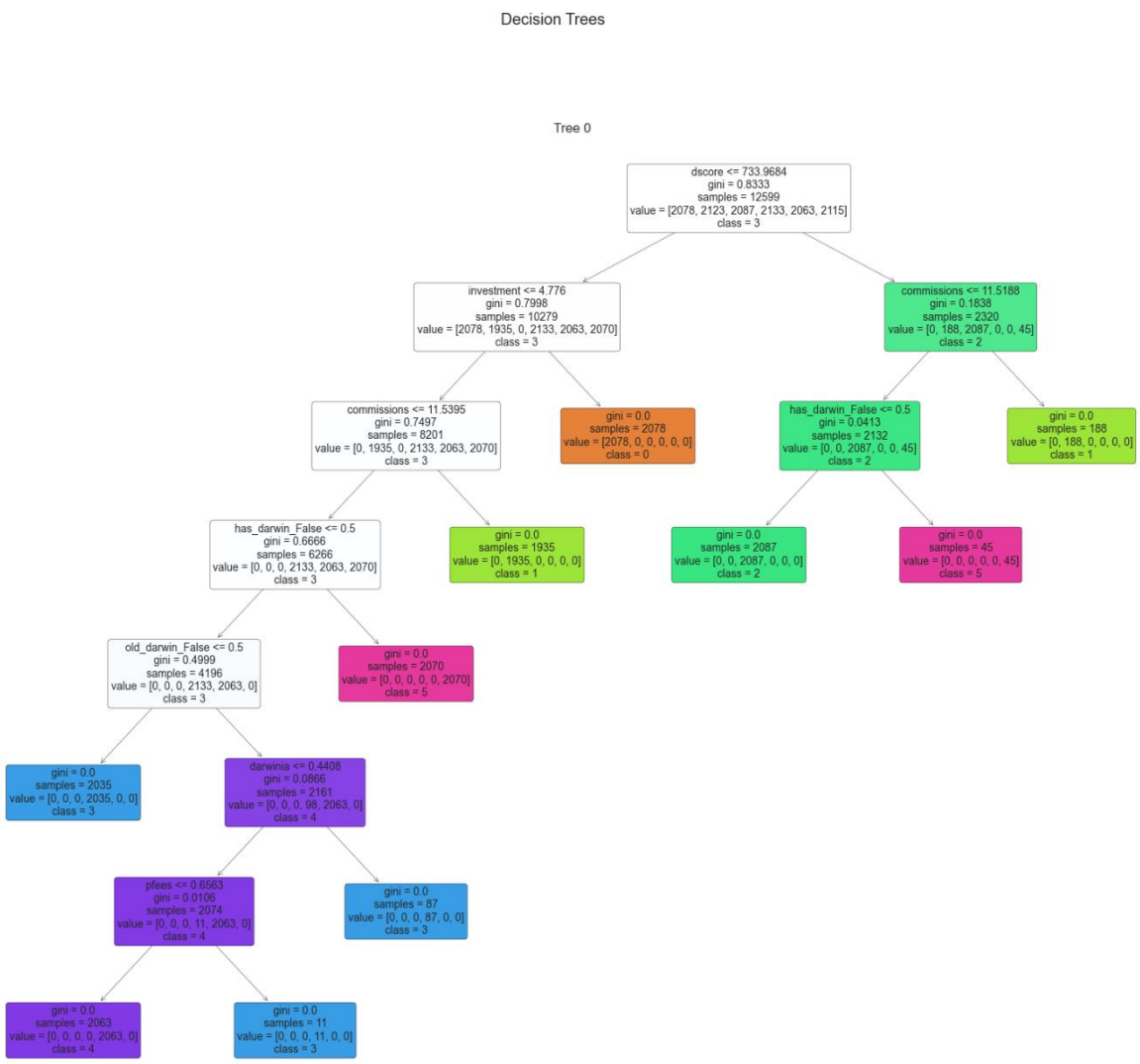


Ilustración 27 - Árbol de decisión del modelo ModClasClientes (Verdadero a la izquierda)

SI comparamos este árbol con el criterio que usamos en la clasificación actual observamos que, exceptuando la clase 5, en el resto de las clases los criterios son prácticamente idénticos, con pequeñas diferencias en los umbrales. En el caso de la clase 5, aunque el criterio obtenido por el modelo es a partir de múltiples variables, obtenemos el mismo resultado.

Clase	Darwinex	Modelo
0	Investment >=4.81096370750229 dscore <= 734.202471737399	Investment > 4.776 dscore <= 733.9684
1	commissions >= 11.543172134840798 dscore > 734.202471737399	commissions > 11.5188 dscore > 733.9684

	OR dscore <= 734.202471737399 investment < 4.81096370750229 commissions >= 11.543172134840798	OR dscore <= 733.9684 investment <= 4.776 commissions > 11.5395
2	commissions < 11.543172134840798 dscore > 734.202471737399 has_darwin = 1	commissions <= 11.5188 dscore > 733.9684 has_darwin = 1
3	dscore <= 734.202471737399 investment < 4.8109637075022 commissions < 11.543172134840798 has_darwin = 1 (darwinia > 0 OR pfees >= 0.6202833268151894 OR old_darwin = 1)	dscore <= 733.9684 investment <= 4.776 commissions <= 11.5395 has_darwin = 1 (darwinia > 0.4408 OR pfees > 0.6563 OR old_darwin = 1)
4	dscore <= 734.202471737399 investment < 4.8109637075022 commissions < 11.543172134840798 old_darwin = 0 darwinia = 0 pfees < 0.6202833268151894 has_darwin = 1	dscore <= 733.9684 investment <= 4.776 commissions <= 11.5395 old_darwin = 0 darwinia <= 0.4408 pfess <= 0.6563 has_darwin = 1
5	has_darwin = 0	commissions <= 11.5188 dscore > 733.9684 has_darwin = 0 OR commissions <= 11.5395 dscore >= 733.9684 investment <= 4.776 has_darwin = 0

Tabla 5 - Criterios de clasificación sistema actual vs modelo AA

Podemos concluir que este modelo cumple con los criterios de éxito definidos para este modelo y por tanto podríamos desplegarlo para usarlo en producción, aunque dado que el fin de crear este modelo es comparar los resultados con el

sistema de clasificación que hay actualmente en la empresa, no se contempla desplegarlo.

3.4.2 Evaluación del modelo ModEstIngresos

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Linear Regression	186.1927	157196.7969	396.4805	0.9995	0.0015	0.0007

Ilustración 28 - Resultados del modelo ModEstIngresos

Con el modelo *ModEstIngresos* obtenemos un resultado muy similar al primer caso con ajustes casi del 100%. Visualizando la gráfica del error comparando los valores calculados con los reales observamos como los puntos tienen un ajuste casi completo.

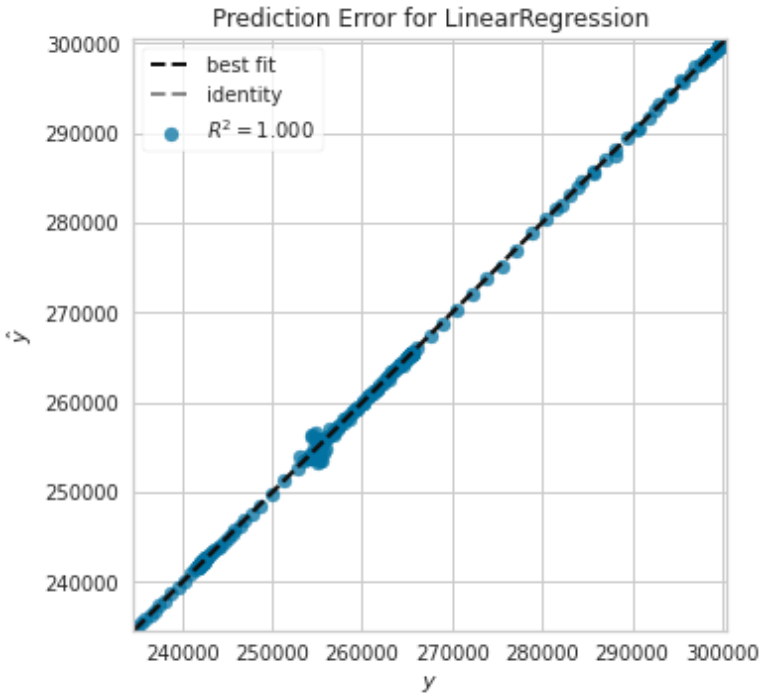


Tabla 6 - Visualización del error del modelo ModEstIngresos

Si nos centramos en el peso que tiene cada columna en la predicción de los nuevos observamos como las columnas que más peso tienen son la de los días justo anteriores, concretamente los 5 días previos al que se quiere calcular. Puesto que los datos de ingresos son bastantes estables, el uso de los datos previos para calcular los

nuevos, sobre todo el de los días previos más cercanos, nos permite obtener este ajuste casi perfecto, al menos para el cálculo de nuevos valores a partir de datos reales.

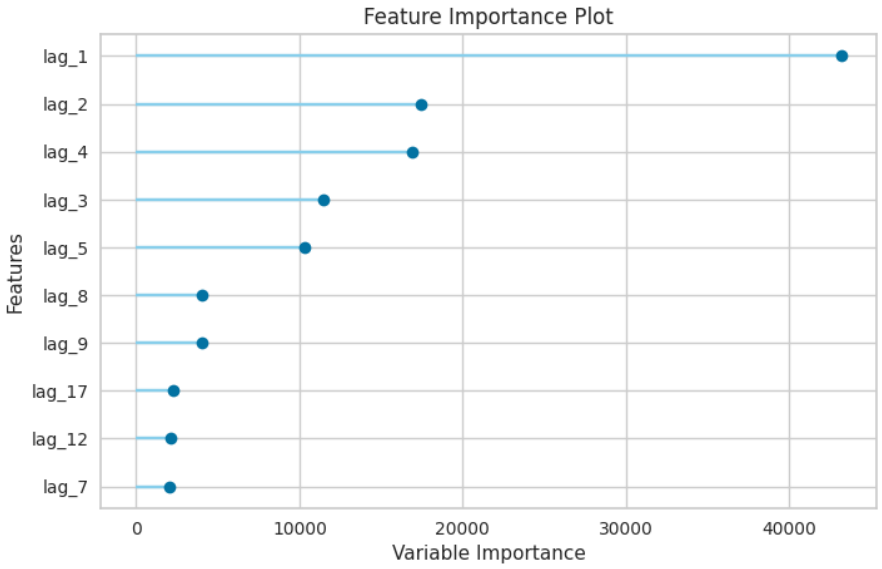


Ilustración 29 - Importancia de las columnas del modelo ModEstIngresos

Con los resultados obtenidos podemos concluir que se cumplen los criterios definidos para el sistema de AA. Pero obtenemos datos extraños al calcular datos a futuro, inicialmente obtiene unos datos que encajan con los datos reales, pero tiende a generar una línea recta. Esta tendencia podemos comprobarla en la siguiente gráfica. Se deja pendiente evaluar el modelo con datos reales nuevos durante los siguientes meses.

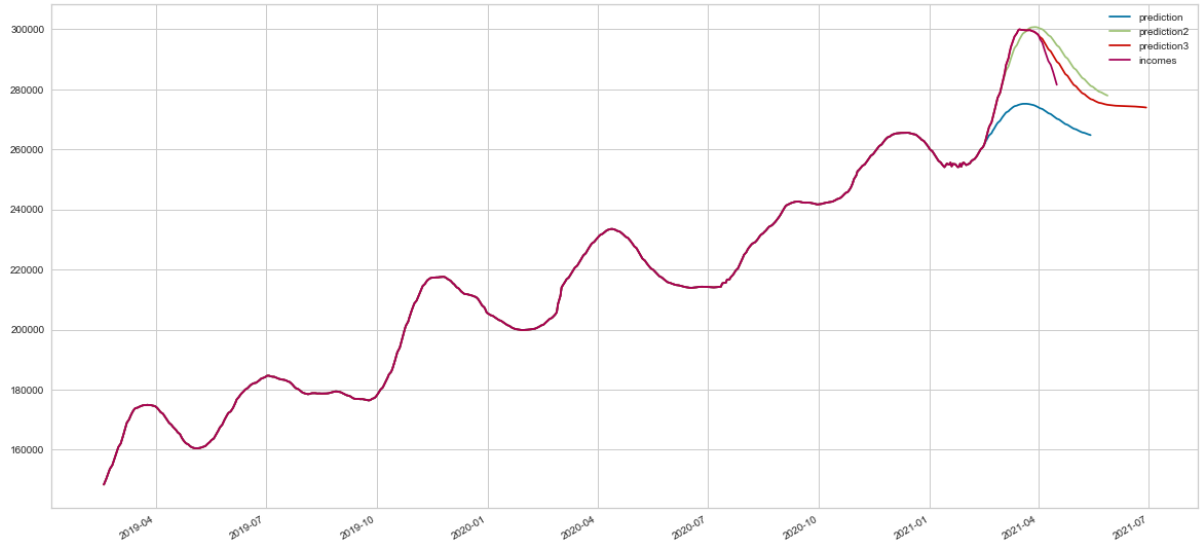


Ilustración 30 - Predicción nuevos datos desde abril para el modelo ModEstIngresos

3.4.3 Evaluación de los modelos ModConversion

Para los modelos *ModConversion* los resultados son muy similares para ambos casos, puesto que la mayoría de los datos son comunes era algo predecible esta coincidencia en los resultados. En ambos casos ya hemos mencionado que lo que se quería era optimizar la identificación del mayor número posible de usuarios que se convierten.

3.4.3.1 Investor

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Ridge Classifier	0.9503	0.9815	0.9919	0.8359	0.9072	0.8736	0.8796

Ilustración 31 – Resultados del modelo ModConversion para INVESTOR

Si visualizamos el resultado de la matriz de confusión vemos como se materializa el uso de la Exhaustividad/*Recall* como criterio de elección del algoritmo, prediciendo casi el 100% de los usuarios que se convierten, a costa de obtener falsos positivos.

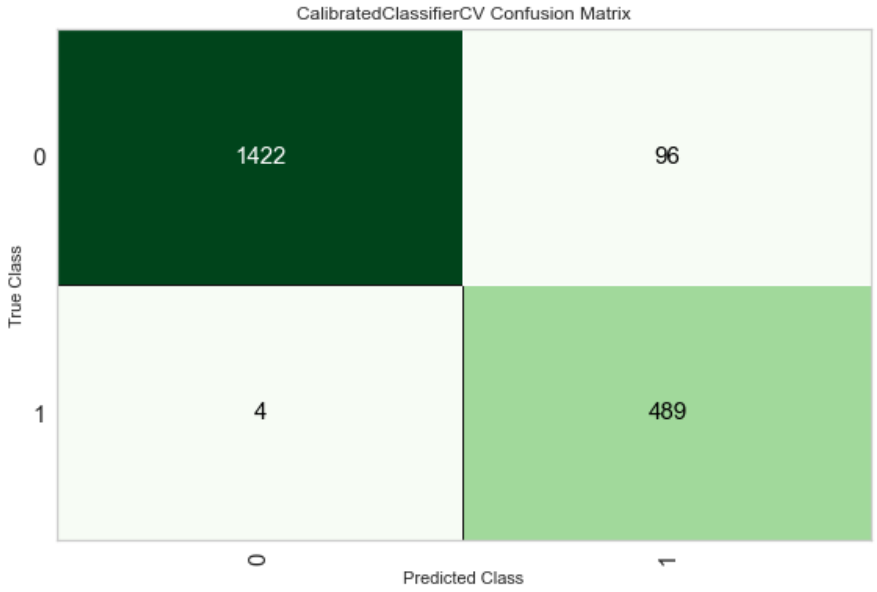


Ilustración 32 - Matriz de confusión del modelo ModConversion para INVESTOR

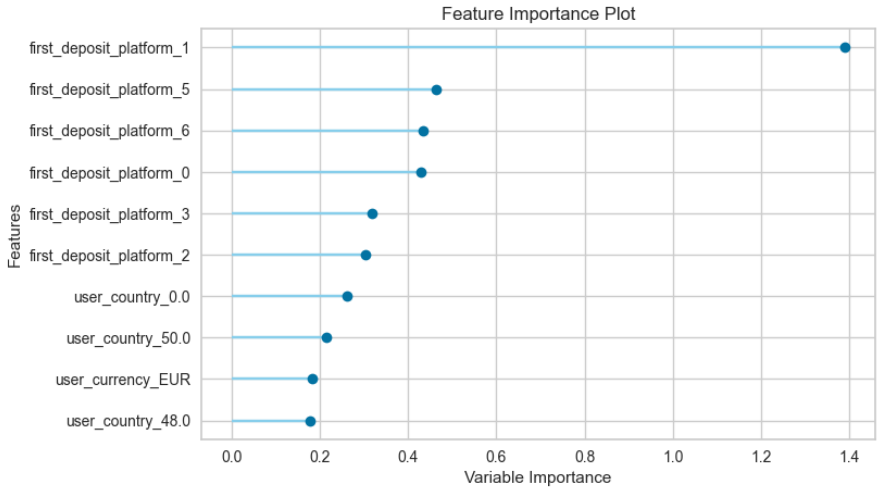


Ilustración 33 - Importancia de las columnas del modelo ModConversion para INVESTOR

3.4.3.2 Trader

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Ridge Classifier	0.9793	0.9992	1.0000	0.9207	0.9587	0.9449	0.9463

Ilustración 34 - Resultados del modelo ModConversion para TRADER

Como ocurre con el modelo para investor, hemos utilizado la Exhaustividad/*Recall,* obteniendo en este caso el 100% en los resultados de verdaderos positivos, es decir, encontrado el 100% de los usuarios que se convierten.

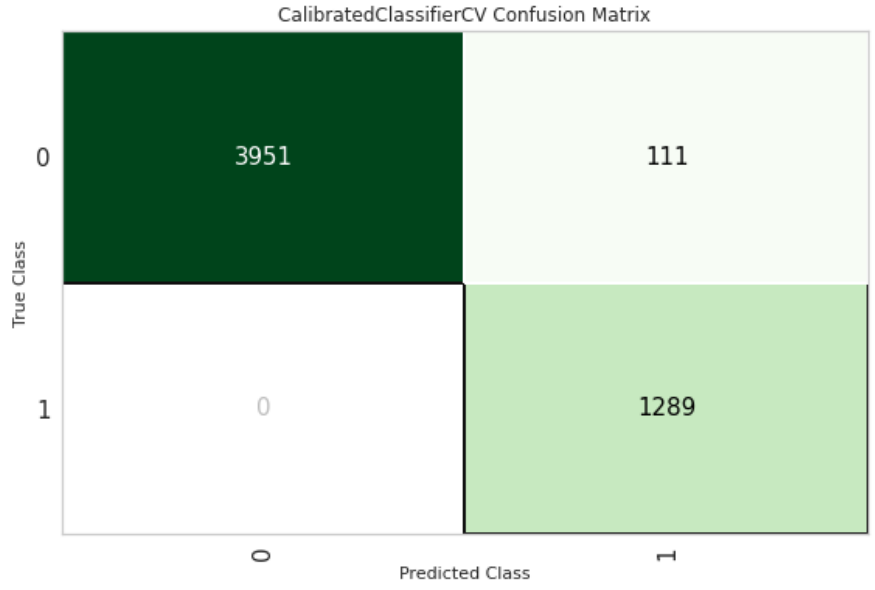


Ilustración 35 - Matriz de confusión del modelo ModConversion para TRADER

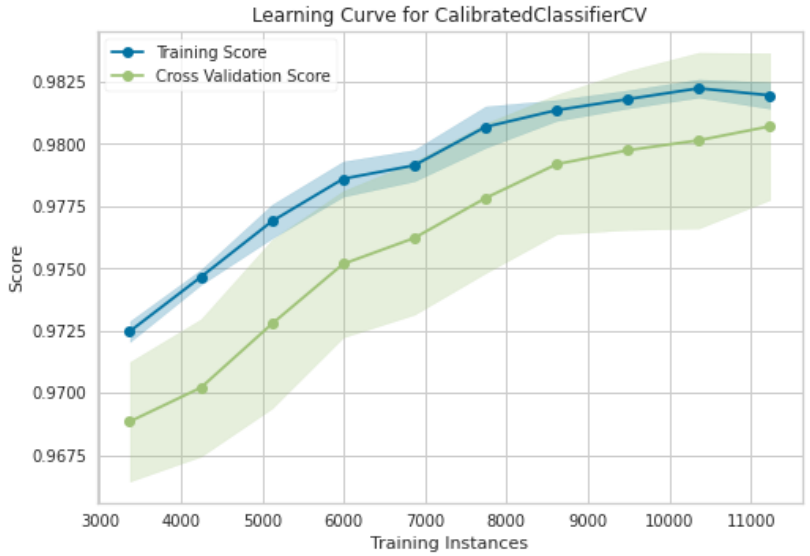


Ilustración 36 - Curva de aprendizaje del modelo ModConversion para TRADER

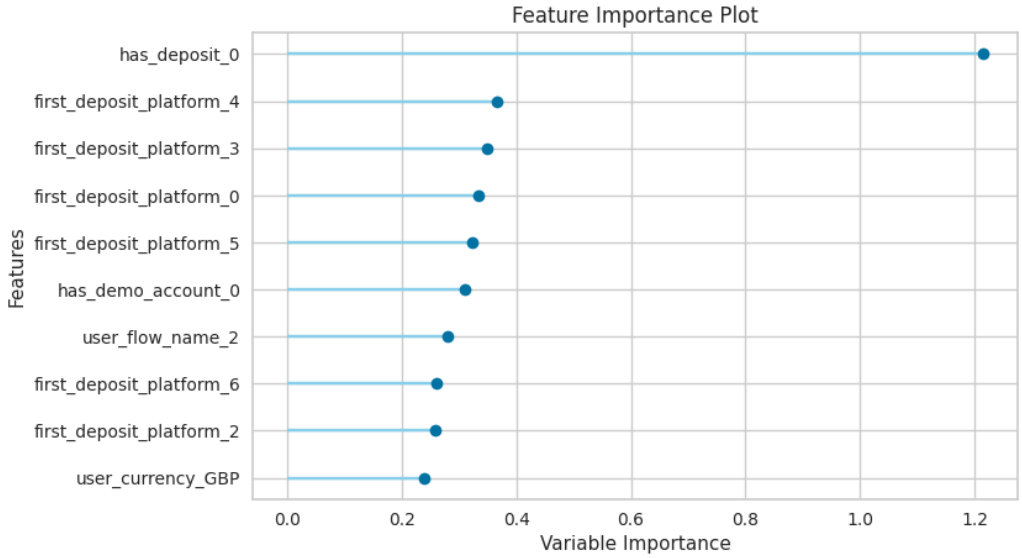


Ilustración 37 - Importancia de las columnas del modelo ModConversion para TRADER

En ambos modelos podemos concluir que se cumple con los criterios definidos para el sistema de AA. Según la observación realizada sobre nuevos datos tras probarlo con el sistema comprobamos que obtenemos resultados favorables, acertando en un 90% de los casos. Hay que destacar que el modelo obtiene mejores resultados cuanto más avanzado está el usuario en el *funnel* de la web. El *funnel*, también llamado flujo de conversión o embudo de conversión incluye todos los pasos

que tiene que dar un usuario en una plataforma para cumplir un objetivo determinado, en el caso de Darwinex, el *funnel* contempla todos los pasos desde el registro hasta que empieza a operar. Por tanto, cuanto más avanzado en el *funnel*, mejores resultados ofrece el modelo para los usuarios. El criterio de éxito definido por negocio es aumentar el número de clientes y ambos tipos, puesto que un impacto así en la plataforma requiere de varios meses de datos para evaluarlo no es posible concretar ahora mismo si se cumple con este criterio. Puesto que en este caso para cumplir el criterio definido por negocio es necesario desplegar el modelo, y que cumple con los criterios definidos para el sistema de AA se ha considerado oportuno y necesario desplegarlo.

3.4.4 Evaluación del modelo ModRecInvDarwin

En este modelo la evaluación es diferente, los resultados cumplen el criterio de éxito marcado para dar por buena las reglas, al menos en lo referente al sistema de AA.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(YFZH)	(OPVB)	0.1023	0.1102	0.1020	0.9972	9.0469	0.0907	320.7627
1	(OPVB)	(YFZH)	0.1102	0.1023	0.1020	0.9254	9.0469	0.0907	12.0263
2	(UCHT, KOGV)	(CQTT)	0.1304	0.3496	0.1187	0.9108	2.6055	0.0732	7.2898
3	(UCHT, JZPL, RFEV)	(CQTT)	0.1297	0.3496	0.1175	0.9059	2.5916	0.0721	6.9129
4	(UCHT, CJFJ, RFEV)	(CQTT)	0.1180	0.3496	0.1057	0.8954	2.5616	0.0644	6.2204
...	...	...	...	...	...	...	...	...	...
122	(CJFJ)	(JZPL, RFEV)	0.2376	0.1623	0.1196	0.5033	3.1011	0.0810	1.6865
123	(UCHT, CQTT)	(CJFJ, RFEV)	0.2102	0.1538	0.1057	0.5027	3.2689	0.0734	1.7016
124	(RFEV)	(CQTT, CJFJ)	0.2433	0.1612	0.1220	0.5015	3.1116	0.0828	1.6826
125	(JZPL, CQTT)	(UCHT, CJFJ)	0.2012	0.1500	0.1009	0.5014	3.3438	0.0707	1.7049
126	(UCHT)	(CQTT, JZPL)	0.3029	0.2012	0.1518	0.5012	2.4914	0.0909	1.6014

127 rows × 9 columns

Ilustración 38 - Ejemplos de las reglas resultantes del modelo ModRecInvDarwin

En cuanto a los criterios definidos por negocio, como ocurre con el modelo *ModConversion*, será necesario evaluarlo transcurrido varios meses tras su uso para poder tener los datos necesarios para extraer las conclusiones, y por tanto de manera análoga, y cumpliendo los criterios definidos para el sistema de AA, se desplegará el modelo.

3.5 Despliegue

Una vez obtenidos los modelos ya ajustados y finalizados, se pasa a la fase de despliegue para su integración con otros sistemas. En el siguiente diagrama podemos ver los principales componentes de AWS que se han utilizado en el sistema, así como los componentes de la empresa con los que se integra, Jenkins, Qlik Sense y la web de Darwinex. Por una cuestión de privacidad de la empresa, todos los procesos y servicios que forman la web se han agrupado en un único componente para no revelar datos internos de la infraestructura.

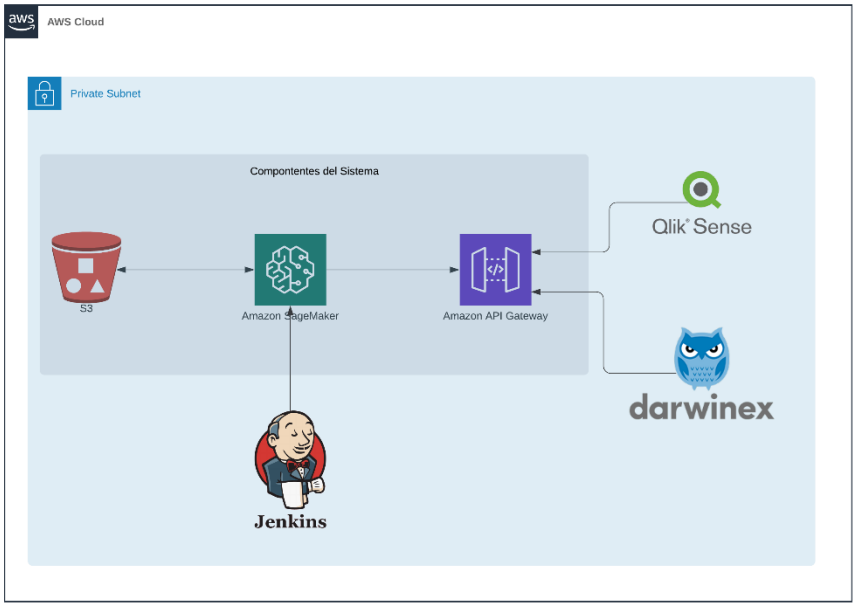


Ilustración 39 - Diagrama de relaciones de componentes del sistema y exteriores al sistema

Se usa AWS Sagemaker para desplegar los modelos. Se accederá a los resultados obtenidos por los modelos vía petición web a través de los *endpoints* de Sagemaker. Además, para los modelos *ModEstIngresos*, *ModConversion* (ambos tipos de usuarios) y *ModRecInvDarwin*, se programarán tareas de entrenamiento de los modelos con nuevos datos para mejorar el ajuste de los modelos con más datos. Estas tareas se ejecutan de manera automática y realizan el despliegue del modelo de forma transparente. Este proceso automático se gestiona a través de Jenkins, que ejecuta los scripts en Python que contiene la lógica para entrenar y crear los *endpoints* en Sagemaker. Jenkins es un sistema para automatización de los procesos de integración y despliegue continuo. Se configuran varias tareas, una para cada modelo, donde se ejecutan el script de Python correspondiente a cada modelo.

Para el modelo *ModClasClientes*, el entrenamiento se realizará de forma manual y no se plantea un proceso automático para actualizarlos con nuevos datos, aunque se ha desarrollado un script que realiza todo el proceso de entrenamiento y despliegue del modelo como en el resto de los modelos.

El uso de los *endpoints* como sistema de comunicación y consulta a los modelos permite facilitar la integración con algunos de los sistemas de la empresa. Para los modelos *ModClasClientes, ModEstIngresos* y *ModConversion* la integración se realiza con Qlik Sense, el sistema de *Business Intelligence* (BI) de la empresa. Esta aplicación nos permite crear tablas y visualizaciones donde mostrar los datos de los diferentes modelos para su uso en los procedimientos de los diferentes equipos de la empresa. El modelo *ModRecInvDarwin* se integra con la web de Darwinex, donde se mostrará las recomendaciones que devuelve el modelo para los Darwins que tiene cada inversor en su portfolio. Esta integración, al requerir modificaciones de un producto externo al proyecto no se puede realizar durante el desarrollo de este.

En el siguiente diagrama se muestran los componentes del sistema, así como las partes software que se han desarrollado, se han obviado los sistemas ajenos al proyecto que utilizarán los modelos.

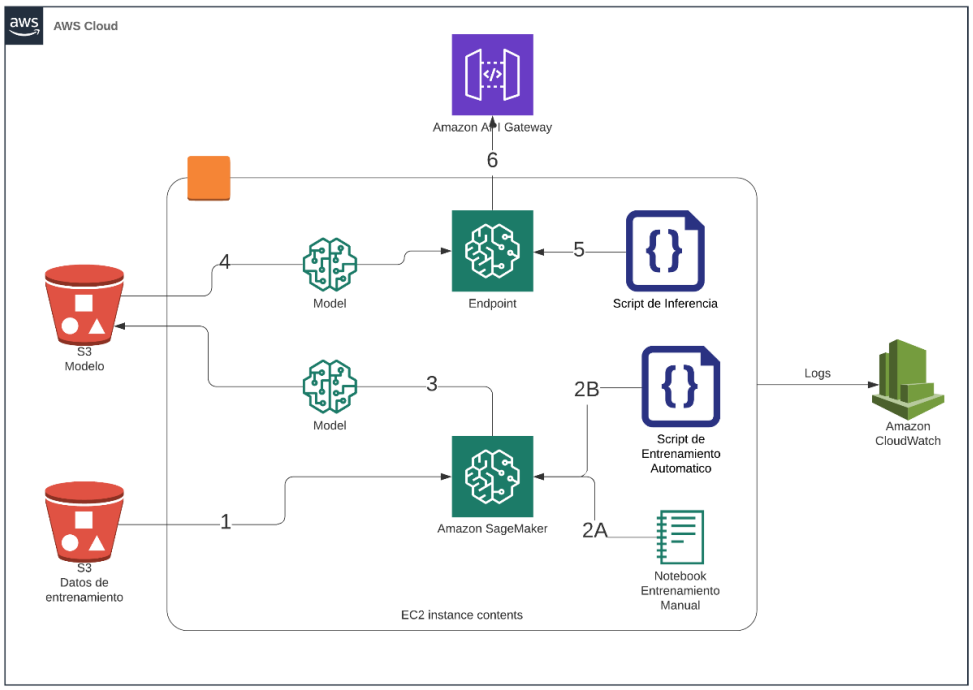


Ilustración 40 - Diagrama de componentes en AWS

3.5.1 Componentes de AWS

- **S3**: Este servicio de AWS almacena los datos para realizar las tareas de entrenamiento, así como los modelos resultantes de esta tarea.
- **Amazon Sagemaker**: Representa la ejecución de la tarea de entrenamiento, para ello Sagemaker utiliza Docker para envolver la lógica del proceso y ejecutarlo de manera transparente.
- **Endpoint**: Un *endpoint* es un punto de entrada de datos para realizar el proceso de inferencia de datos utilizando el modelo entrenado. También se utiliza Docker para envolver la lógica y ejecutarlo de manera transparente.
- **CloudWatch**: Es un servicio para monitorizar procesos. Mas concretamente en este proyecto sirve para alojar en los logs y salidas de todos los procesos ejecutados en Sagemaker. También nos sirve para almacenar las métricas resultantes de los modelos y configurar alertas para detectar anomalías.
- **API Gateway**: Este servicio nos permite realizar llamadas vía URL a los *endpoints* creados para cada modelo.

Todos los componentes de AWS utilizados son *Serverless*, es decir, no hay que gestionar ni configurar servidores para su utilización, AWS los gestiona de manera transparente.

Siguiendo el diagrama anterior podemos ver el flujo de trabajo seguido para cada modelo.

1. Obtenemos los datos para el entrenamiento de cada modelo desde S3.
2. En función de si estamos usando un notebook para las pruebas iniciales (2A) o el script automático de entrenamiento (2B).
3. Almacenamos el modelo resultante en S3.
4. Cargamos la última versión del modelo desde S3.
5. Creamos el *Endpoint* en AWS Sagemaker con el script de cada modelo.

6. Con API Gateway podemos convertimos el *Endpoint* de AWS Sagemaker en una URL accesible de manera pública y con un dominio propio que facilite su uso.

El despliegue en AWS Sagemaker se realiza a través de Docker. AWS Sagmekaer utiliza Docker para ejecutar tareas de entrenamiento, de ajustes de hiperparámetros y para consultar e inferir datos con los modelos. Se ha creado un Docker a partir de los que ofrece AWS, más concretamente a partir de la imagen de Docker de Scikit-learn. La modificación ha sido menor ya que solo ha sido necesario instalar las dependencias de Pycaret para poder usarlo.

Para cada requisito se ha creado un notebook para crear una primera versión de los modelos de forma manual.

Todo el sistema de logs resultante de las ejecuciones se almacenará en CloudWatch, pero en esta parte profundizaremos más en la siguiente fase.

Como sistema de almacenamiento se utiliza S3 para los datos de entrenamiento y los modelos finalizados, de esta manera mantenemos todo el sistema dentro de AWS.

Si se quisiese generalizar este proyecto a otras PYMEs, habría que centrarse en las partes de software: los notebooks, los scripts para entrenar los modelos e implementar la inferencia de datos, y en los scripts para automatizar los entrenamientos y creación de endpoints que se ejecutan en este caso desde Jenkins. El resto de los componentes no requieren parte de desarrollo ni de configuración previa.

3.5.2 Integración con AWS Sagemaker

Integrar el código con Sagemaker es bastante sencillo, para ejecutar las tareas de entrenamiento, e implementar la funcionalidad de consultas vía web.

Para realizar esta integración hemos implementado las clases y proceso que indica el Docker de Scikit-Learn para Sagemaker [42] que estamos utilizando de base.

Se ha desarrollado un script en Python para cada modelo que incluye la funcionalidad para ejecutar ambas funciones. Para entrenar los modelos los scripts incluyen una función *main* que se encarga de ejecutar todo el procedimiento de obtener los datos, realizar el proceso de entrenamiento y almacenamiento del modelo

resultante en S3. Para la parte de consulta vía web hay que implementar una serie de métodos que utiliza para ejecutar las diferentes fases en la consulta de datos para su inferencia.

3.5.2.1 Entrenamiento

La fase de entrenamiento en Sagemaker se realiza ejecutando el *main* del *script* en Python creado para cada modelo. El proceso es muy simple y replica lo ya creado en cada *notebook*. Además, de cara a la monitorización de los modelos, se envían las métricas resultantes de cada modelo a CloudWatch para su monitorización. De manera análoga, todos los logs generados por estos procesos se envían a CloudWatch, este proceso lo realiza Sagemaker de manera automática.

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()

    # Sagemaker specific arguments. Defaults are set in the environment variables.
    parser.add_argument('--train', type=str, default=os.environ['SM_CHANNEL_TRAIN'])

    args = parser.parse_args()

    _df = pd.read_csv(os.path.join(args.train, 'dataset_model1.csv'))

    clf1 = setup(_df, normalize=True, target='label', ignore_features=['userid'], silent=True, html=False,
                 verbose=False)

    top = compare_models(n_select=1, verbose=False)
    top_tuned = tune_model(top, verbose=False)
    calibrated_dt = calibrate_model(top_tuned, verbose=False)
    final_model = finalize_model(calibrated_dt)
    last_metrics = pull()

    save_model(final_model, model_name='model1', verbose=False)
    S3.upload_file('model1.pkl', 'tfm-2021-darwinex', 'models/model1/final_model.pkl')
    S3.upload_file('model1.pkl', 'tfm-2021-darwinex', 'models/model1/history/{}_model'.format(datetime.datetime.now()))

    send_metric('model1', 'Accuracy', last_metrics['Accuracy'].iloc[0])
    send_metric('model1', 'AUC', last_metrics['AUC'].iloc[0])
    send_metric('model1', 'Recall', last_metrics['Recall'].iloc[0])
    send_metric('model1', 'Prec.', last_metrics['Prec.'].iloc[0])
    send_metric('model1', 'F1', last_metrics['F1'].iloc[0])
    send_metric('model1', 'Kappa', last_metrics['Kappa'].iloc[0])
    send_metric('model1', 'MCC', last_metrics['MCC'].iloc[0])

    print("saved model!")
```

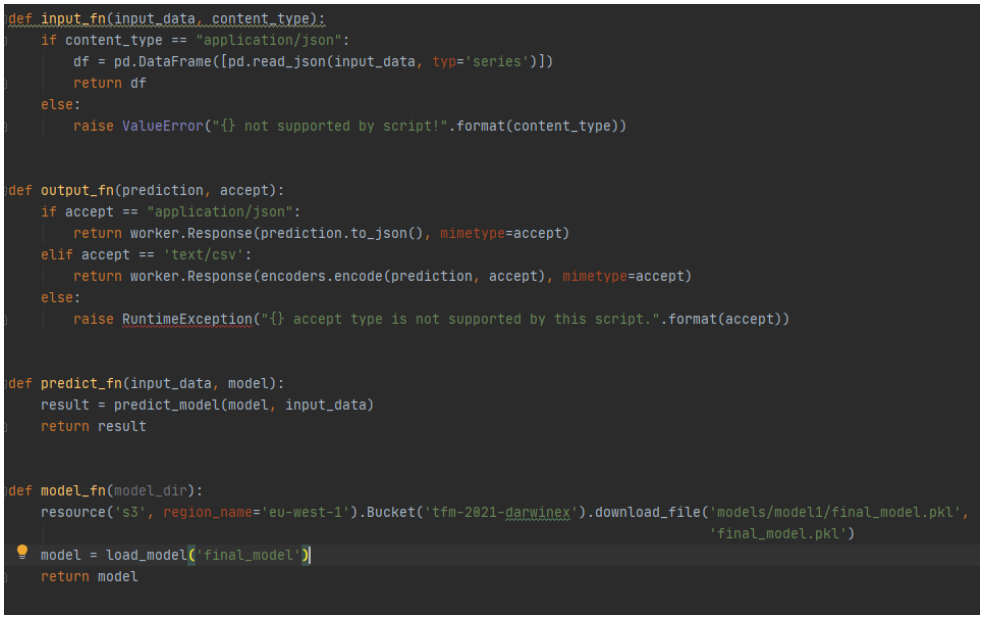
Ilustración 41 - Detalle código de creación del modelo

ModClasClientes

3.5.2.2 Inferencia

Como ya hemos comentado, utilizamos de base el Docker predefinido creado para Scikit-Learn para Sagemaker, el cual define unos métodos predefinidos para implementar la lógica de inferencia para los modelos. Aunque estos métodos tienen

una implementación por defecto, dado que usamos Pycaret y no Scikit-Learn, tenemos que realizar nuestra propia implementación.



```
def input_fn(input_data, content_type):
    if content_type == "application/json":
        df = pd.DataFrame([pd.read_json(input_data, typ='series')])
        return df
    else:
        raise ValueError("{} not supported by script!".format(content_type))


def output_fn(prediction, accept):
    if accept == "application/json":
        return worker.Response(prediction.to_json(), mimetype=accept)
    elif accept == 'text/csv':
        return worker.Response(encoders.encode(prediction, accept), mimetype=accept)
    else:
        raise RuntimeException("{} accept type is not supported by this script.".format(accept))


def predict_fn(input_data, model):
    result = predict_model(model, input_data)
    return result


def model_fn(model_dir):
    resource('s3', region_name='eu-west-1').Bucket('tfm-2021-darwinex').download_file('models/model1/final_model.pkl',
                                                                                      'final_model.pkl')
    model = load_model('final_model')
    return model
```

Ilustración 42 - Detalle código de la inferencia del modelo ModClasClientes

Los métodos para implementar son:

- **input_fn**: Procesa los datos de entrada, en nuestro caso utilizamos JSON como método de entrada.
- **model_fn**: Carga el modelo para realizar la inferencia de datos, en nuestro caso se lee el modelo desde S3.
- **predict_fn**: Este método ejecuta la inferencia de los datos utilizando el modelo cargado previamente y los datos de entrada.
- **output_fn**: Procesa los datos de salida, al igual que la entrada es en formato JSON, aunque también permite devolver los datos en CSV.

3.5.3 Actualización automáticamente los modelos

Para la actualización automática de los modelos se han creado *scripts* en Python que utilizan la SDK de Sagemaker para la creación y despliegues de los nuevos modelos.

```
sklearn_preprocessor = SKLearn(
    entry_point=script_path,
    role=role,
    image_uri=image,
    sagemaker_session=sagemaker_session,
    base_job_name=job_name,
    job_name=job_name,
    metric_definitions=[
        {'Name': 'Accuracy', 'Regex': 'Accuracy=(\d\.\d+)'},
        {'Name': 'AUC', 'Regex': 'AUC=(\d\.\d+)'},
        {'Name': 'Recall', 'Regex': 'Recall=(\d\.\d+)'},
        {'Name': 'Prec.', 'Regex': 'Prec.=(\d\.\d+)'},
        {'Name': 'F1', 'Regex': 'F1=(\d\.\d+)'},
        {'Name': 'Kappa', 'Regex': 'Kappa=(\d\.\d+)'},
        {'Name': 'MCC', 'Regex': 'MCC=(\d\.\d+)'}
    ],
    instance_type=instance_type)

sklearn_preprocessor.fit({'train': data_folder})
```

Ilustración 43 - Detalle código de entrenamiento automático del modelo ModClasClientes

Estos scripts utilizan el código mencionado en 3.5.1 para realizar las tareas. Para la parte de inferencia, se utiliza el SDK para eliminar el *endpoint* que estuviese desplegado hasta ese momento, y después crea el nuevo *endpoint* a partir del nuevo modelo.

```
try:
    SM_CLIENT.describe_endpoint(EndpointName=endpoint_name)
    SM_CLIENT.delete_endpoint(EndpointName=endpoint_name)
    SM_CLIENT.delete_endpoint_config(EndpointConfigName=endpoint_name)
    desc = _wait_until(lambda: _deploy_done(SM_CLIENT, endpoint_name), 30)
except ClientError as ex:
    pass

predictor = model.deploy(endpoint_name=endpoint_name, initial_instance_count=1,
                         serializer=JSONSerializer(), deserializer=JSONDeserializer(),
                         instance_type=instance_type)
```

Ilustración 44 - Detalle código de endpoint automático del modelo ModClasClientes

3.6 Monitorización y mantenimiento

En esta fase no solo se plantean las tareas relacionadas con la monitorización y mantenimiento del código, sino también de los datos. En AA el resultado del modelo puede degradarse por cambios de patrones en los datos con los que se creó el modelo, lo cual obliga no solo a monitorizar el funcionamiento del sistema, sino a monitorizar cambios en los datos como la distribución o estacionalidad.

3.6.1 Monitorización del sistema

Todo el sistema está conectado con AWS CloudWatch, para centralizar todos los logs generados. Con esta herramienta podemos monitorizar y crear alertas para detectar y actuar frente a cualquier problema que surja en el sistema.

3.6.1.1 Logs

Los logs de cada ejecución, tanto la parte de entrenamiento, que se guarda en */aws/sagemaker/TrainingJobs*, como los logs del punto de enlace para las peticiones webs, que se guardan en */aws/sagemaker/Endpoints/**, se almacenan en CloudWatch para facilitar su consulta en todo momento.

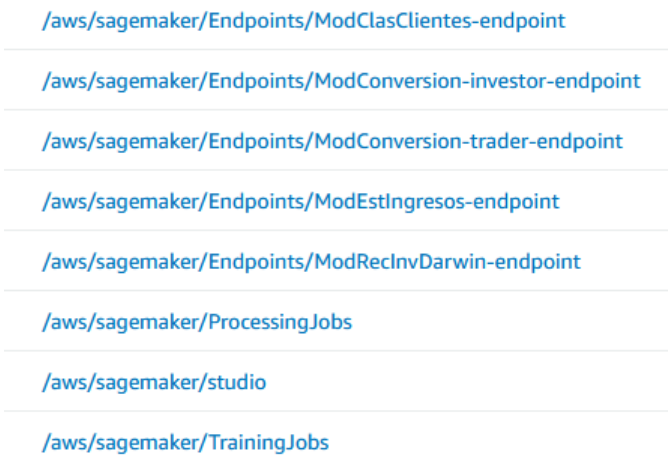


Ilustración 45 - Entradas en cloudwatch de cada modelo

3.6.1.2 Métricas

Para controlar las métricas de los modelos, y detectar las posibles pérdidas de rendimiento de estos, se envían las métricas principales a CloudWatch con cada ejecución.

ModClasClientes	Kappa
ModClasClientes	MCC
ModClasClientes	AUC
ModClasClientes	Recall
ModClasClientes	Prec.
ModClasClientes	F1
ModClasClientes	Accuracy
ModConversion-investor	Prec.
ModConversion-investor	F1
ModConversion-investor	Accuracy

Ilustración 46 - Ejemplos de las métricas enviadas a CloudWatch

Podemos ver la evolución de estas métricas con cada ejecución y configurar alertas para detectar problemas en la generación de los modelos, de esta forma controlamos la calidad de las inferencias realizadas.

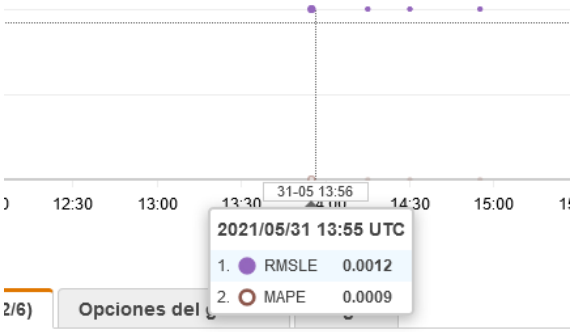


Ilustración 47 - Ejemplo del grafico de métricas del modelo ModEstIngresos

3.6.2 Monitorización de los datos

Sabemos, por experiencia con los datos de la empresa, que los cambios en estos son derivados por modificaciones en las aplicaciones más que por cambios de comportamientos de los usuarios, por lo que no se plantea la automatización de esta monitorización, sino que se planificarán tareas manuales asociados a los cambios en el producto que puedan ser susceptibles de alterar los datos. De estas tareas manuales se derivan las acciones de corrección de los modelos que sean necesarias para que el sistema siga ejecutándose de forma correcta.

3.6.3 Mantenimiento de los modelos

De las tareas manuales anteriores que resulten en cambios importantes que puedan afectar a la inferencia de nuevos datos se ejecutarán nuevas acciones para volver a entrenar los modelos con un nuevo conjunto de datos que contemple los nuevos tipos de datos. Estas tareas serán manuales, realizándose siempre un proceso de investigación, lo cual en la práctica significa iterar sobre las diferentes fases de la metodología.

También, como hemos comentado anteriormente, los modelos *ModEstIngresos*, *ModConversion* y *ModRecInvDarwin* se actualizarán de forma automática a partir de nuevos datos que se vayan generando, esto nos permitirá, en la medida de lo posible, mantener el modelo lo más actualizado posible.

Capítulo 4 - Conclusiones y trabajo futuro

El presente trabajo analiza el uso de técnicas de AA en una PYME y se proponen soluciones para la integración de las técnicas de AA en el sistema de información de la empresa. El proyecto pretende servir de guía para la implementación de la tecnología AA en una PYME a través de un ejemplo concreto para Darwinex. Como resultados relevantes destacamos los siguientes:

Se ha utilizado la metodología CRISP-ML(Q), la cual ha guiado cada fase y proporcionado las tareas a realizar en cada fase para completar el proyecto. La metodología resulta ágil, planteando ya desde el principio avances y retrocesos entre fases por los nuevos conocimientos que se van obteniendo de los datos y de los modelos a medida que se avanza en el proceso de desarrollo. Hay que destacar y valorar muy positivamente la fase final de monitorización y mantenimiento, la cual no se suele contemplar en muchos proyectos, no solo de AA si no de ingeniera de software en general. Dada la naturaleza de los modelos que se crean de los datos, y que suelen ser cambiantes y evolucionan con el tiempo, es necesario considerar estos aspectos en los proyectos de AA para obtener resultaros erróneos cuando los datos varíen y no se hayan actualizado los modelos. El detalle de las fases y de las tareas a realizar en cada fase, proporcionadas por la metodología CRISP-ML(Q), permite que una PYME, sin experiencia previa en este tipo de proyectos pueda llevarlo a cabo sin mayor dificultad como ha sido el caso de Darwinex. Pero como toda metodología, es importante adaptarla a los procedimientos y posibilidades de la PYME.

El uso de la librería Pycaret ha permitido realizar y comparar múltiples algoritmos y modelos con muy poco esfuerzo de desarrollo, el cual era uno de los objetivos a la hora de elegir la librería. La integración de múltiples algoritmos con la facilidad de comparar al momento todos los disponibles para el tipo de entrenamiento facilita la elección del mejor algoritmo. El diseño de su API permite, mediante parámetros automatizar todo el trabajo de tratamiento de datos y facilita los primeros pasos al crear el modelo, siendo muy sencillo probar diferentes opciones hasta dar con el resultado óptimo. En resumen, la librería Pycaret es una herramienta muy potente para los primeros proyectos de AA que se quieran llevar a cabo en una PYME. Por otra parte, tanta facilidad y automatización puede resultar limitante para empresas que ya

tienen experiencia y quieren implementar sus propios procesos de tratamiento, o proyectos que persigan utilizar técnicas basadas en RN las cuales no están contempladas en esta librería.

La infraestructura utilizada, Sagemaker, ofrece las herramientas necesarias para ejecutar y automatizar las tareas de creación de los modelos e inferencia de nuevos datos a partir de estos. Lo más destacable de la infraestructura es la capacidad de adecuar los recursos utilizados, tanto durante las primeras fases de creación de los modelos, pudiendo ejecutar los notebooks con la potencia y memoria requerida por cada modelo, como durante la fase final de automatización de las tareas de entrenamiento e inferencia, pudiendo elegir los tipos de instancias para cada tarea en cada modelo. Esta adecuación de los recursos resulta muy útil en las PYMES pudiendo hacer uso de la infraestructura según los recursos económicos de los que se puedan disponer. De serie, Sagemaker ofrece numerosos algoritmos y recursos que pueden ser utilizados directamente o adaptados, como ha sido el caso de este proyecto, donde se han adaptado las imágenes Docker utilizadas para las tareas de entrenamiento e inferencia para poder usar la librería Pycaret. También proporciona una documentación muy completa y ejemplos para poder usarla o adaptarla a las necesidades de cada proyecto. Y como todos los productos de AWS, ofrece múltiples integraciones con otros productos de la misma nube como es el caso de AWS CloudWatch, para el almacenamiento de logs y monitorización de las métricas de los modelos, algo muy importante para poder evaluar el rendimiento de los resultados.

En cuanto a los resultados obtenidos por la implantación del AA en Darwinex, las métricas de todos los modelos han superado los umbrales marcados en los criterios de éxito.

Modelo	Métrica	Valor definido	Valor obtenido
ModClasClientes	Exactitud/*Accuracy*	*0.9*	1
ModEstIngresos	MAPE	0.25	0.07
ModConversion - INVESTOR	Exhaustividad/*Recall*	0.9	0.9919
ModConversion - TRADER	Exhaustividad/*Recall*	0.9	1
ModRecInvDarwin	-	-	-

Tabla 7 - Resumen de métricas por modelo

Como se muestra en la tabla, en todos los casos se han obtenido valores satisfactorios. Para el caso de los modelos de *ModConversion* y *ModClasClientes* los resultados son del 100% no siendo posible obtener mejores resultados, y para el modelo ModEstIngresos hemos obtenido un valor de métrica un 72% mejor al valor definido como criterio de éxito. En el caso de *ModRecInvDarwin* no había métricas para medir el modelo. Estos resultados han sido satisfactorios y han incentivado el avance en la integración del sistema en la empresa.

La elección de la metodología de desarrollo CRISP-ML(Q), de la librería Pycaret, y la infraestructura AWS Sagemaker, ha permitido plantear y desarrollar de forma satisfactoria un sistema de AA en Darwinex sin tener experiencia previa en este tipo de proyectos. Se han obtenido un conjunto de modelos funcionales que cumplen sobradamente con los criterios técnicos marcados para los modelos.

La utilización de la metodología y de las herramientas puede facilitar y agilizar la implantación del AA en las PYMES, mejorar significativamente su competitividad, sus resultados y los servicios que ofrecen a sus clientes.

4.1 Líneas futuras

Los resultados obtenidos en el proyecto han puesto en evidencia un conjunto de limitaciones que abren la vía a posibles mejoras y a la continuación del trabajo en diferentes direcciones:

- **Fiabilidad de los datos**: Los modelos *ModEstIngresos, ModConversion* y *ModRecInvDarwin* necesitan ser evaluados durante varios meses de uso para poder evaluar su eficacia para el uso por parte de la empresa.
- **Mejora de los modelos**: Con el fin de mejorar los modelos se pueden realizar incrementar las iteraciones sobre la metodología, tras obtener los resultados de su uso en la empresa.
- **Utilización de herramientas en la nube**: La utilización de diferentes sistemas y herramientas en la nube, como Google, abre la posibilidad de prescindir de la librería de PyCaret, en este caso para utilizar solamente las herramientas que ofrece la nube. Sería necesario realizar un desarrollo paralelo utilizando herramientas y sistemas en la nube para evaluar, si se obtienen mejores resultados, o si facilitase las tareas de procesamiento y selección de los conjuntos de datos.
- **Uso de algoritmos basados en RN**: Hacer uso de otros tipos de algoritmos, más concretamente basados en RN, dado el potencial que están mostrando en tareas similares a las que se han implementado, para ello se podrían utilizar librerías como H2O, Auto-Keras o TPOT, las cuales permiten crear modelos de RN utilizando AutoML como se ha hecho con Pycaret.
- **Guía de implantación**: Las herramientas utilizadas en el proyecto pueden facilitar la implantación de sistemas de AA en otras PYME, para ello sería necesario evaluar su uso en otras empresas y obtener una guía de desarrollo de sistemas de AA en PYME que incorporara las experiencias obtenidas.

Chapter - Introduction

1. Context

The adoption of Machine Learning (ML) and Artificial Intelligence (AI) by Spanish companies is still low, although in the average with the European Union (EU27). The 9% of Spanish companies use some type of AI technology, compared to 7% of the EU27 average. If we focus on the data for Spain, and differentiate by company size, we observe that 27% of large companies (those with more than 249 employees) incorporate AI, compared to 8% in the case of SMEs (between 10 and 249 employees). In the case of ML, the 4% of SMEs incorporate it, compared with the 18% of large companies. The difference in capacity to face new challenges or the lack of resources of SMEs (equipment and qualified personnel, time required or economic capacity) compared to large companies, is behind these large differences. [1]

In this project we focus on SMEs since this is the business environment in which it will be tested.

Increasing customer conversion, improving support processes, personalization of marketing campaigns and processes optimization are just some of the advantages that ML techniques can bring to companies in general and SME. This is achieved thanks to the characteristics of ML such as the ability to detect patterns in data, the processing of large volumes of data, the capacity for automation or continuous improvement, but its integration can bring with it a series of challenges.

Cost/benefit impact on the business

Thinking about its impact on the company, it is necessary to analyze correctly how it will affect the company, what is intended to be obtained with this system and when it is considered that its implementation has been satisfactory.

Use of necessary libraries

These are essential to develop any ML project in an optimal and simple way. The different ML libraries facilitate the application of the algorithms and the different phases involved in the project. Some of them, known as AutoML, can automate the processing and transformation of data and even the creation and comparison of different models. Subsequently, some of the main existing libraries will be analyzed, and we will focus mainly on AutoML because it provides a faster and simpler way forward in the first steps in the incorporation of ML in companies.

Using the right infrastructure

It is essential to find an adequate infrastructure to deploy the models and to perform queries on them from the rest of the company's processes so that the work done does not remain a mere investigation. The incorporation of these models into the company's workflows will make it possible to take full advantage of the potential of ML.

Stability of the models obtained with ML

The results offered by the ML models are not stable since the data tend to change over time, being necessary new trainings, therefore, it is especially important to detail how the system will be maintained or monitored.

When considering the development of an ML system, it is necessary to follow the same principles as any other software engineering project, it is no different because it is ML. For all these reasons, it is suggested that the development of an ML system must follow a methodology, using the appropriate libraries or frameworks following the principle of not reinventing the wheel, and choosing the appropriate infrastructure for its deployment and integration with the rest of the technological ecosystem of the company, because in the end if there is no integration and interaction with other systems, we cannot obtain any of the benefits that ML brings us.

This project describes the process of incorporating ML in a financial company, specifically Darwinex. The company wants to improve different day-to-day processes in different departments. The company has positively considered the advantages that ML

can bring and has given its support to carry out this project to test the integration of ML technology in the company.

The project has been developed in collaboration with the company Darwinex, responsible for the web https://www.darwinex.com. The student author of this project works as Data Team Leader in the company and knows the company's business and data.

Darwinex is a Spanish company founded in 2012 and it has more than 50 employees in 2021. Belonging to the Fintech world, it is considered a technology provider that helps traders to develop their skills and build a certified track record, or in other words, they act as certifiers of traders' quality. This allows them to offer, through their own marketplace, the opportunity for traders to raise capital from investors and charge a 15% success fee under the legal cover of being regulated as a broker and asset manager with the British regulator, FCA. To simplify its operation, to facilitate the understanding of the business context where this project has been developed, Darwinex offers two different services to two types of clients, traders, and investors: A service oriented to the concept of traditional trader, Darwinex is an online broker that allows trading with various assets of Forex, Stocks, Commodities, indices, or Futures.

- A service oriented to the concept of traditional trader, Darwinex is an online broker that allows trading with various assets of Forex, American Stocks, Commodities, indices, and Futures.
- The second service is oriented to investor clients, to whom Darwinex, as a financial asset manager, offers a new financial product called Darwins. A Darwin is nothing more than a brand or asset created on the strategy of a trader, that is, a trader who has a strategy can decide to create a Darwin on it and allow investors to invest money in it. It is an evolved concept of social trading where some users replicate the operation of a user who operates. In Darwinex this concept has been taken further and using proprietary technology everything has been wrapped in a financial product with its own quotation and a risk control system to ensure the investment of investors and their capital.

2. Objectives

The objectives of the project have been set considering both academic and business requirements as follows.

General Objective

- To integrate Machine Learning techniques into a business through the development of a software system and connecting it to the rest of the system.

Specific Objectives

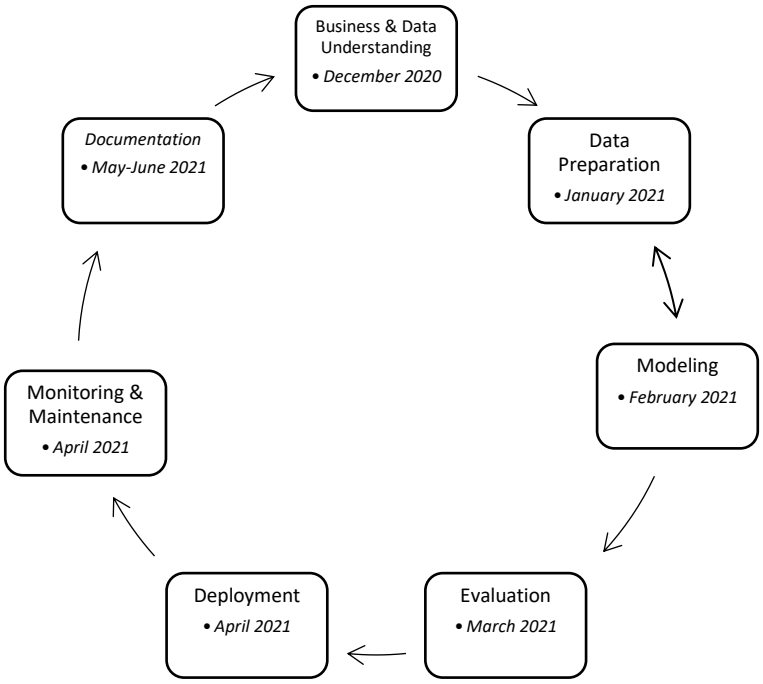
To carry out the integration of ML into the company's information systems it will be necessary:

- Together with the company, define the following business cases with which to evaluate the possible benefits for the company:
    - Create a model to classify traders that replicates the company's current classification and compare them (supervised learning model of multi-class classification).
    - Create a model that estimates the future monthly revenue of the firm based on past data (supervised learning regression model).
    - Create a model that can anticipate a user trader's conversion before it occurs (binary classification supervised learning model).
    - Create a model that can anticipate the conversion of an investor user before it occurs (binary classification supervised learning model).
    - Create a Darwins recommendation model based on the detection of buying patterns (unsupervised learning model based on association rules).
- Choose an appropriate methodology for the development of the system considering the peculiarities of the ML.
- Evaluate the existing libraries to identify the most suitable ones to facilitate the system development phases.
- Find the most suitable infrastructure to run the system efficiently.

3. Work Plan

The work plan has been planned to be carried out between December 2020 and May 2021, both included. The previous months have been dedicated to research and learning the essential ML tools and concepts and methodologies to develop the project.

The CRISP-ML(Q) methodology, which is documented and developed in section 2.1 of this document, has been used throughout the project, so the work planning has been carried out according to the phases described therein.



Ilustration 48 - Planning phases

This planning is an approximation contemplating that in each phase, starting from the first one, time has been dedicated to review the work of the previous phases to see if it is necessary to make any adjustments in the data collection or processing, model creation, etc. Therefore, several iterations over the whole cycle are not contemplated since at any time we can go back to a previous phase to apply changes. The documentation of the project and the development of this document, although it has been carried out during the whole process, a last stage of the planning is contemplated for review and finalization.

4. Structure of the document

Chapter 1 describes the motivation of the project, the business context, the proposed objectives, both at the technical and business level. The methodology used for the development of the project is described, analyzing some of the methodologies for ML projects and justifying the choice made.

Chapter 2 describes the ML concepts applicable to the development of the project, details the metrics to evaluate the system, the libraries that can be used to facilitate the development, as well as the different infrastructures, more specifically in the cloud, to deploy and implement the project.

Chapter 3 details the different phases of the methodology to carry out the development of the project, describes the tasks and decisions taken, ending with the evaluation of the models and the tasks proposed to maintain and monitor the system.

The conclusions of the project and future work are presented in chapter 4. The scope of the results obtained by the choices made in terms of methodology, libraries and infrastructure are analyzed, and the advantages and conclusions drawn are presented. Finally, the lines of future work to improve or extend this project are mentioned.

5. Project code and data

The code developed in this project, and the data to train the models, are hosted on GitHub, accessible through the URL https://github.com/sgavmp/tfm_ml_code and under MIT license.

Chapter - Conclusions and future work

This paper analyzes the use of ML techniques in an SME and proposes solutions for the integration of ML techniques in the company's information system. It is intended to serve as a guide for the implementation of ML in an SME through a concrete example for Darwinex. As relevant results we highlight the following:

The CRISP-ML(Q) methodology has been used, which has guided each phase and task to be carried out to complete the project. The methodology has been very agile, and from the very beginning it has proposed advances and setbacks between phases due to the new knowledge obtained from the data and the models as the process progresses. The final phase of monitoring and maintenance, which is not usually contemplated in many projects, not only in ML but also in software engineering in general, must be highlighted and valued very positively. Given the nature of the models, which are created from the data, and that these are usually changing and evolve over time, it is necessary to consider these aspects in the ML projects to obtain erroneous results when the data change and the models have not been updated. The detail of the phases and the tasks to be carried out in each one of them allows an SME, without previous experience in this type of project, to carry it out without major difficulty, as has been the case of Darwinex. But like any methodology, it is important to adapt it to the procedures and possibilities of the SME.

The use of the Pycaret library has allowed the realization and comparison of multiple algorithms and models with very little development effort, which was one of the objectives when choosing the library. The integration of multiple algorithms with the facility to instantly compare all available algorithms for the type of training facilitates the choice of the best algorithm. The design of its API that allows, by means of parameters, to automate all the work of data processing facilitates the first steps in creating the model, being extremely easy to try different options until finding the optimal result. In summary, the Pycaret library is a powerful tool for the first ML projects to be carried out in an SME. On the other hand, such ease and automation can be limiting for companies that already have experience and want to implement their own treatment processes, or projects that seek to use NR-based techniques that are not covered by this library.

The infrastructure used, Sagemaker, provides the necessary tools to execute and automate the tasks of creating models and inferring new data from them. The most outstanding feature of the infrastructure is the capacity to adapt the resources used, both during the first phases of model creation, being able to run the notebooks with the power and memory required by each model, and during the final phase of automation of the training and inference tasks, being able to choose the types of instances for each task in each model. This resource adequacy is very useful for small and medium sized companies, being able to make use of the infrastructure according to the economic resources available. As standard, Sagemaker offers many algorithms and resources that can be used directly or adapted as has been the case in this project, where we have adapted the Docker images used for training and inference tasks to use the Pycaret library. It offers a very complete documentation and many examples to use it or adapt it to the needs of each project. And like all AWS products, it offers multiple integrations with other products in the same cloud, such as AWS CloudWatch for storing logs and monitoring model metrics, which is very important for evaluating the performance of the results.

As for the results obtained by the implementation of ML in Darwinex, the metrics of all the models have exceeded the thresholds set in the success criteria.

Model	Metrics	Define Value	Measure Value
ModClasClientes	Accuracy	*0.9*	1
ModEstIngresos	MAPE	0.25	0.07
ModConversion - INVESTOR	Recall	0.9	0.9919
ModConversion - TRADER	Recall	0.9	1
ModRecInvDarwin	-	-	-

Table 8 - Model metrics summary

As can be seen in the table, in all cases we have obtained very good values. For the case of the ModConversion and ModClasClients models the results are 100% and it is not possible to obtain better results, and for the ModEstIngresos model we have obtained a metric value 72% better than the value defined as success criterion. In the case of ModRecInvDarwin there were no metrics to measure the model. These results

have been very good and have allowed further progress in the integration of the system in the company.

The choice of the CRISP-ML(Q) development methodology, the Pycaret library, and the AWS Sagemaker infrastructure, has made it possible to propose and develop a satisfactory ML system in Darwinex without previous experience in this type of project, and a set of functional models has been obtained that more than meets the technical criteria established for the models. The use of these tools could facilitate and speed up the implementation of ML in SMEs to improve their competitiveness, their results, or the services they offer to their customers.

1. Future directions

The results obtained in the project have revealed a set of limitations that open the way for possible improvements and the continuation of the work in different directions:

- **Data reliability**: The ModEstIncome, ModConversion and ModRecInvDarwin models need to be evaluated over several months of use to assess their effectiveness for business use.
- **Improvement of the models**: To improve the models, several more iterations on the methodology can be performed after obtaining the results of their use in the company.
- **Use of tools in the cloud**: The use of different systems and tools in the cloud, such as Google, opens the possibility of dispensing with the PyCaret library in this case to use only the tools offered by the cloud. A parallel development using tools and systems in the cloud would be necessary to evaluate whether the cloud offers a better result, or whether it facilitates the tasks of processing and selection of the data sets.
- **Use of Artificial Neural Networks algorithms**: Make use of other types of algorithms, more specifically Artificial Neural Networks (ANN), given the potential they are showing in tasks like those that have been implemented, for which libraries such as H2O, Auto-Keras or TPOT could be used, which allow the creation of ANN models using AutoML as has been done with Pycaret.

- **Implementation guide**: The tools used in the project could be used to facilitate the implementation of ML systems in other SMEs, for this it would be necessary to evaluate their use in other companies could be evaluated and a more general SME ML systems development guide could be obtained.

BIBLIOGRAFÍA

[1] Observatorio Nacional de Tecnología y la Sociedad, «Indicadores de uso de Inteligencia Artificial en,» 2021. [En línea]. Available: https://www.ontsi.red.es/es/dossier-de-indicadores-pdf/indicadores-uso-inteligencia-artificial-empresas-espanolas.

[2] «Kdniggets,» 10 2014. [En línea]. Available: https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html. [Último acceso: 03 2021].

[3] «IBM,» [En línea]. Available: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=dm-crisp-help-overview. [Último acceso: 3 2021].

[4] «Wikimedia,» [En línea]. Available: https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png.

[5] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi y T. Zimmermann, *Software Engineering for Machine Learning: A Case Study. International Conference on Software Engineering,* ICSE, 2019.

[6] E. B. S. C. E. N. M. S. D. Sculley, *The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction,* 2017.

[7] T. B. B. C. D. A. H. L. W. S. P. K.-R. M. Stefan Studer, *Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology,* 2020.

[8] T. Mitchel, Machine Learning, McGraw-Hill Science/Engineering/Math, 1997.

[9] U. M. C. Sibanjan Das, Hands-On Automated Machine Learning - a beginner's guide to building automated machine learning systems using AutoML and Python, Packt, 2018.

[10] F. S. Caparrini, «Fernando Sancho Caparrini - Aprendizaje Supervisado y No Supervisado,» 14 12 2020. [En línea]. Available: http://www.cs.us.es/~fsancho/?e=77. [Último acceso: 06 2021].

[11] «GeeksForGeeks,» 23 2 2020. [En línea]. Available: https://www.geeksforgeeks.org/clustering-in-machine-learning/. [Último acceso: 6 2021].

[12] P. Z. Rob Thomas, The AI Ladder. Accelerate Your Journey to AI, O'Reilly, 2020.

[13] «Wikimedia,» 2 11 2009. [En línea]. Available: https://en.wikipedia.org/wiki/File:Labels.png. [Último acceso: 06 2021].

[14] «AWS,» [En línea]. Available: https://docs.aws.amazon.com/machine-learning/latest/dg/when-to-use-machine-learning.html.

[15] M. Kubat, An Introduction to Machine Learning, USA: Springer, 2017.

[16] «Sitio Big Data,» 19 1 2019. [En línea]. Available: https://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/#. [Último acceso: 3 2021].

[17] «Towards Data Science,» 17 9 2019. [En línea]. Available: https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226. [Último acceso: 3 2021].

[18] «Sitio Big Data,» 27 5 2019. [En línea]. Available: https://sitiobigdata.com/2019/05/27/aprendizaje-automatico-seleccionando-metricas-regresion/. [Último acceso: 3 2021].

[19] «Altexsoft,» 18 8 2017. [En línea]. Available: https://www.altexsoft.com/blog/datascience/choosing-an-open-source-machine-learning-framework-tensorflow-theano-torch-scikit-learn-caffe/. [Último acceso: 3 2021].

[20] S. D. B. T. L. G. H. M. H. Giang Nguyen, «Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey,» *Artificial Intelligence Review,* nº 52, pp. 77-124, 2019.

[21] S. Das y U. M. Cakmak, Hands-On Automated Machine Learning - a beginner's guide to building automated machine learning systems using AutoML and Python,

United Kingdom: Packt Publishing Ltd., 2018.

[22] V. H. Ulla Gain, *Low-code AutoML-augmented Data Pipeline – A Review and Experiments,* 2021.

[23] «Pycaret,» [En línea]. Available: https://pycaret.org/about/. [Último acceso: 6 2021].

[24] «Whizlabs,» 14 2 2020. [En línea]. Available: https://www.whizlabs.com/blog/comparing-machine-learning-as-a-service/. [Último acceso: 03 2021].

[25] «Towards Data Science,» 22 9 2020. [En línea]. Available: https://towardsdatascience.com/data-science-in-the-cloud-239b795a5792. [Último acceso: 3 2021].

[26] «Altexsoft,» 25 4 2021. [En línea]. Available: https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/.

[27] J. A. Rodrigo, «Cienciadedatos,» 6 2018. [En línea]. Available: https://www.cienciadedatos.net/documentos/43_reglas_de_asociacion. [Último acceso: 3 2021].

[28] «PyCaret,» [En línea]. Available: https://pycaret.readthedocs.io/en/latest/tutorials.html#tutorials.

[29] «Towards Data Science,» 6 4 2019. [En línea]. Available: https://towardsdatascience.com/time-series-machine-learning-regression-framework-9ea33929009a. [Último acceso: 3 2021].

[30] «PandasProfiling,» [En línea]. Available: https://github.com/pandas-profiling/pandas-profiling. [Último acceso: 06 2021].

[31] S. G. Alonso, 31 05 2021. [En línea]. Available: https://github.com/sgavmp/tfm_ml_code. [Último acceso: 05 2021].

[32] «Pycaret,» [En línea]. Available: https://pycaret.org/setup/. [Último acceso: 5 2021].

[33] «Pycaret,» [En línea]. Available: https://pycaret.org/compare-models/. [Último

acceso: 5 2021].

[34] «Pycaret,» [En línea]. Available: https://pycaret.org/create-model/. [Último acceso: 5 2021].

[35] «Pycaret,» [En línea]. Available: https://pycaret.org/tune-model/. [Último acceso: 5 2021].

[36] «Pycaret,» [En línea]. Available: https://pycaret.org/calibrate-model/.

[37] «Pycaret,» [En línea]. Available: https://pycaret.org/interpret-model/. [Último acceso: 5 2021].

[38] «Pycaret,» [En línea]. Available: https://pycaret.org/plot-model/. [Último acceso: 5 2021].

[39] «Pycaret,» [En línea]. Available: https://pycaret.org/finalize-model/. [Último acceso: 5 2021].

[40] «Pycaret,» [En línea]. Available: https://pycaret.readthedocs.io/en/latest/api/classification.html. [Último acceso: 05 2021].

[41] R. R. S. Agrawal, «Fast algorithms for mining association rules,» 1994.

[42] «Sagemaker Scikit-Learn,» [En línea]. Available: https://sagemaker.readthedocs.io/en/stable/frameworks/sklearn/using_sklearn.html. [Último acceso: 6 2021].

[43] AWS, «Amazon SageMaker,» [En línea]. Available: https://aws.amazon.com/es/sagemaker/.

[44] P. D. [@pmddomingos], 3 9 2019. [En línea]. Available: https://twitter.com/pmddomingos/status/1168713477009137665. [Último acceso: 06 2021].

[45] R. Allen, «Machine Learning in Practice - Medium,» 21 1 2020. [En línea]. Available: https://medium.com/machine-learning-in-practice/how-machine-learning-differs-from-traditional-software-80d0a235ff3b. [Último acceso: 6 2021].

[46] A. Pinhasi, «Assaf Pinhasi - Medium,» 18 12 2019. [En línea]. Available: https://assaf-pinhasi.medium.com/towards-a-development-methodology-for-machine-learning-part-i-f1050a0bc607. [Último acceso: 6 2021].