



Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

**SAMUEL CAMBA FERNÁNDEZ**

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

**Selección y evaluación de tecnologías de  
aprendizaje máquina automatizado para la  
resolución de problemas de inteligencia  
artificial en el ámbito industrial**

Febrero de 2021

Tutores: Antonio Miguel López Rodríguez (Uniovi)  
Ángel Conde Manjón (Ikerlan)

## AGRADECIMIENTOS

Quiero dedicar este pequeño espacio a mi padre, agradecerte todo el esfuerzo que dedicaste para sacar adelante una familia y tres agradecidos hijos que te tendrán siempre en el recuerdo. Agradecerte que, si no fuera por ti, hoy no tendría esta afición por los ordenadores y la informática, que al final se convirtió en mi estudio y profesión. Seguro estarías orgulloso.

Con amor  
Samuel

# ÍNDICE

1.- Resumen.....	5
2.- Motivación.....	6
3.- Introducción.....	9
3.1.- Antecedentes.....	9
3.2.- Estudio de alternativas.....	20
3.3.- Selección de alternativas.....	31
3.3.1.- Google Cloud Platform.....	32
3.3.2.- AutoGluon.....	33
3.3.3.- Amazon Web Services.....	34
3.3.4.- H2O.....	35
4.- Descripción de los casos de uso.....	36
4.1.- Caso 1: Tratamiento de datos de carácter desbalanceado.....	36
4.2.- Caso 2: Detección de anomalías en series temporales.....	40
4.3.- Caso 3: Clasificación de imágenes y localización de objetos para la detección de anomalías.....	43
5.- Metodología.....	47
5.1.- Tradicional (Ad-Hoc).....	47
5.1.1.- Caso 1.....	47
5.1.2.- Caso 2.....	51
5.1.3.- Caso 3.....	57
5.2.- Aprendizaje Máquina Automatizado (AutoML).....	62
5.2.1.- Caso 1.....	62
5.2.1.1.- Google Cloud Platform.....	62
5.2.1.2.- AutoGluon.....	66
5.2.1.3.- H2O.....	68

5.2.2.-	Caso 2.....	70
5.2.2.1.-	Amazon Web Services.....	70
5.2.2.2.-	AutoGluon.....	74
5.2.3.-	Caso 3.....	74
5.2.3.1.-	Google Cloud Platform.....	74
5.2.3.2.-	Amazon Web Services.....	77
5.2.3.3.-	AutoGluon.....	79
6.-	Resultados.....	81
6.1.-	Tradicional (Ad-Hoc).....	81
6.1.1.-	Caso 1.....	81
6.1.2.-	Caso 2.....	82
6.1.3.-	Caso 3.....	85
6.2.-	Aprendizaje Máquina Automatizado (AutoML).....	87
6.2.1.-	Caso 1.....	87
6.2.1.1.-	Google Cloud Platform.....	87
6.2.1.2.-	AutoGluon.....	89
6.2.1.3.-	H2O.....	90
6.2.2.-	Caso 2.....	92
6.2.2.1.-	AutoGluon.....	92
6.2.2.2.-	Amazon Web Service.....	93
6.2.3.-	Caso 3.....	94
6.2.3.1.-	Google Cloud Platform.....	94
6.2.3.2.-	Amazon Web Services.....	97
6.2.3.3.-	AutoGluon.....	98
7.-	Discusión.....	100
8.-	Conclusiones y trabajo futuro.....	104
9.-	Bibliografía.....	106

10.- Planificación temporal.....	110
11.- Glosario.....	111

# 1.- Resumen.

Debido a la mejora tecnológica y el incremento en la digitalización en la industria, cada vez más compañías proponen soluciones a sus problemas basadas en modelado e inteligencia artificial. Como consecuencia de eso, muchas empresas se ven superadas o desfasadas en cuanto a infraestructura y expertise. Además, los ingenieros de ML tienen una gran demanda, y mejorar la eficiencia de estos es un desafío fundamental. El aprendizaje máquina automatizado (AutoML) ha surgido como una forma de democratizar el aprendizaje automático, y ahorrar tiempo y esfuerzo en tareas repetitivas que componen el desarrollo de soluciones basadas en ML, como el preprocesamiento de datos, la ingeniería de características, la selección de modelos, la optimización de hiperparámetros y el análisis de resultados de predicción.

## 2.- Motivación.

La renovación tecnológica y la constante evolución por no quedarse atrás, en lo que ya se denota con el término de cuarta revolución industrial (industria 4.0), está al orden del día. Las producciones industriales están gobernadas por una gran competencia y cambios constantes en la demanda. Las fábricas de la industria moderna deben satisfacer unos requisitos de adaptabilidad a las necesidades y a los procesos de producción, además de una asignación eficiente de recursos. Esta orientación está guiando una evolución tecnológica en dos vías, “Sistemas ciberfísicos” e “Internet de las cosas” (Figura 2-1). Dotando a las manufacturas de más automatización, mayor conectividad y generando más información digital, es posible hoy obtener un mayor conocimiento de los procesos productivos, mejorar la eficiencia y la calidad de éstos, y alinear los objetivos empresariales con las necesidades de los clientes. Como se cita en [1], la Industria 4.0 puede resultar en una reducción en:

- los costos de producción en un 10-30%,
- los costes de logística en un 10-30%,
- los costos en gestión de calidad en un 10-20%.

Por otro lado, también son destacables beneficios en los aspectos de: (1) un menor tiempo-al-mercado de nuevos productos, (2) una mayor capacidad de respuesta por parte de los clientes (3) mayor flexibilidad en personalizar la producción sin aumentar costes generales, (4) usos más eficientes de los recursos naturales y la energía.

## Industry 4.0

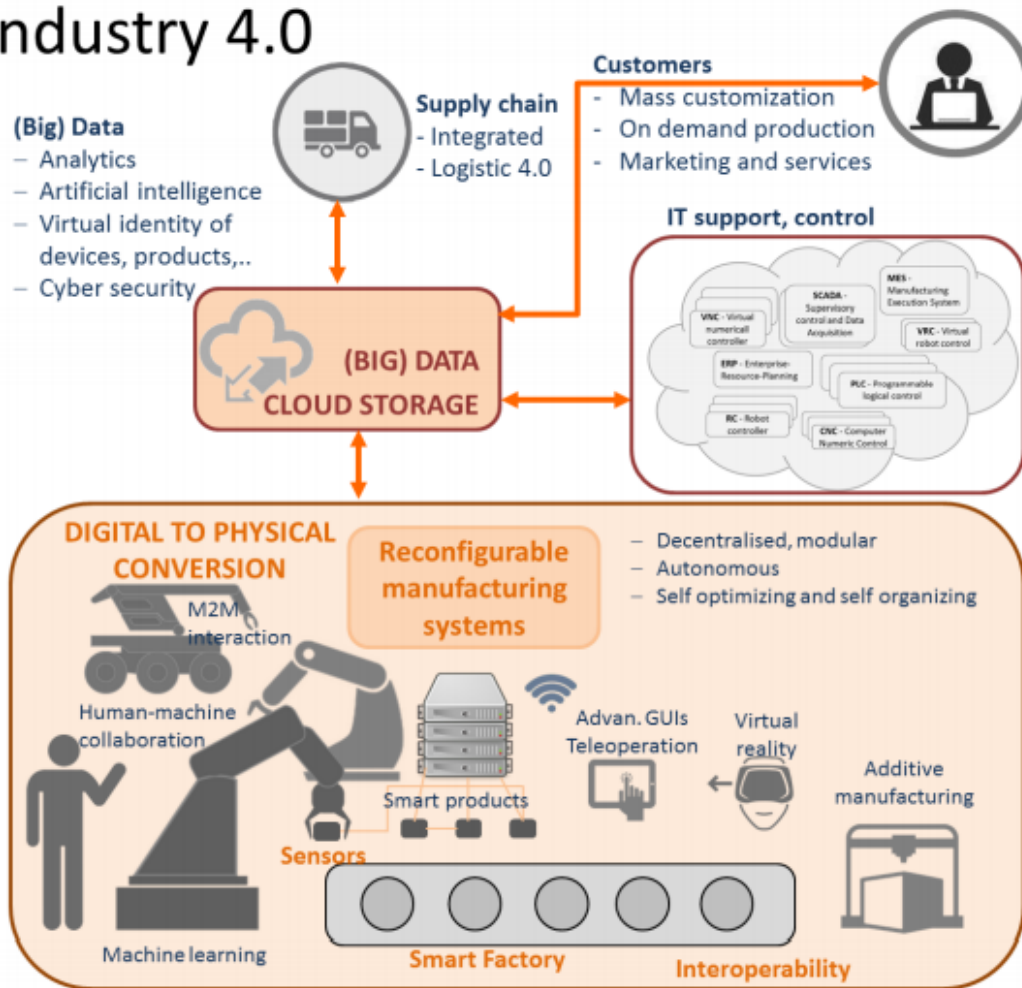


Figura 2-1.- Esquema gráfico de la Industria 4.0.

Un apartado de gran importancia en la industria 4.0 es el tratamiento y análisis de la cantidad ingente de datos, datos que desde hace años se han recopilado y almacenado y hoy en día se ve en ellos una gran fuente de conocimiento y valor. El uso del análisis de datos para mejorar los sistemas de manufacturación está teniendo una gran repercusión en la literatura. [2] Generalmente el análisis de datos puede ayudar a las compañías en tres aspectos: (1) Analítica Descriptiva, utilizando los datos recopilados para responder a la pregunta de “¿Qué ha pasado?” para intentar comprender el porqué de lo sucedido. (2) Analítica Predictiva, donde se utilizan algoritmos y técnicas para construir modelos que predicen el futuro, trata de contestar la pregunta de “¿Qué va a suceder?”. (3) Analítica Prescriptiva, proporcionando valor comercial a través de mejores decisiones estratégicas y operacionales, se basa en la analítica predictiva, donde se trata de prever el resultado o la influencia de cada acción, responde a la



pregunta de “¿Qué debo hacer?”. Actualmente con los avances en el campo de la inteligencia artificial y el desarrollo tecnológico, han surgido y se estudian soluciones basadas en datos que tratan los desafíos que plantea la industria 4.0.

El centro tecnológico de Ikerlan, en Mondragón, se enfrenta día a día con proyectos en el ámbito del análisis de datos e inteligencia artificial. Junto con las universidades y empresas colaboradoras siguen una línea de investigación, innovación y desarrollo de soluciones para la industria basada en modelos de inteligencia artificial. Debido al incremento en la demanda, cada vez es más la necesidad de buscar soluciones eficientes, más rápidas y fáciles de usar, además de la necesidad de expertos en el campo de la inteligencia artificial. En los últimos años ha surgido un nuevo paradigma denominado “Aprendizaje Máquina Automatizado” (en inglés Automated Machine Learning, AutoML) que trata de automatizar el desarrollo de modelos de inteligencia artificial, haciendo este proceso más accesible a cualquier interesado en esta ciencia.

El objetivo de este proyecto es realizar un estudio del arte acerca del aprendizaje máquina automatizado, analizando la variedad y madurez de las herramientas disponibles que cumplen este propósito, y si estas son aplicables en problemas presentes en el ámbito industrial.

Se describirán tres casos de uso que recogen algunos de los problemas más habituales en el desarrollo de soluciones de ML en la industria. Se presentarán diversas metodologías del estado del arte de cómo abordar estos problemas en la actualidad. Por otro lado, se propondrá una metodología utilizando los sistemas de aprendizaje máquina automatizado. Por último, se presentarán los resultados cuantitativos y una discusión comparando ambos enfoques.

Cabe destacar que el objetivo de este proyecto no es superar u optimizar los resultados presentes en la literatura. Los resultados cuantitativos no son relevantes, aunque si servirán para evaluar ambas metodologías, y argumentar los puntos de importancia para la compañía referentes a la resolución de los casos de uso.

## 3.- Introducción.

### 3.1.- Antecedentes

En el año 1959, A. L. Samuel [3] publicaba un artículo donde proponía diversos algoritmos de programación que hacían a una computadora capaz de aprender a jugar mejor a las damas, el juego de tablero. Como se concluye en el reporte realizado por Frank Gabel en el año 2000 [4], y con el que estarían de acuerdo la mayoría de las personas que trabajan en aprendizaje automático, A. L. Samuel preparó el camino para el trabajo posterior en el aprendizaje de la máquina, o en inglés *Machine Learning* (ML).

Actualmente el ML se ha convertido en un tema de gran actividad entre la comunidad académica e incluso en las empresas e industria. La investigación en este campo ha crecido enormemente en la última década, el incremento de la potencia de cálculo de los computadores, el abaratamiento de los costes del hardware, la concienciación acerca de los posibles beneficios por la utilización del aprendizaje automático, son factores que favorecen al avance acelerado de estas técnicas. Hoy en día existen diversas herramientas que la comunidad puede utilizar para desarrollar algoritmos y modelos predictivos como WEKA [5], plataforma desarrollada por la Universidad de Waikato en 1993 para el aprendizaje de datos y minería de datos, o la famosa librería de acceso abierto "Scikit-learn" [6] (2013), para el lenguaje de programación Python, entre otras, utilizadas por millones de interesados por el ML que ven en este campo una ventaja competitiva.

Para entender más profundamente como es el flujo de trabajo (pipeline) con un modelo de ML, en [7] se describen los pasos que generalmente componen el desarrollo y despliegue de un modelo de predicción (Figura 3-1):

1. Análisis de datos: Recopilación de estadísticas, como las entradas que faltan, cuantiles, asimetría, correlación con el objetivo y muchas otras, es útil para comprender los datos y descubrir anomalías, entradas defectuosas, etc.
2. Definición del problema: Determinar de qué tipo de problema se trata, un problema de regresión, clasificación, supervisado o no supervisado.

3. Detección del tipo de dato: Para cada columna del conjunto de datos en bruto identificar el tipo de su contenido: numérico, categórico, lenguaje natural, marca de tiempo, geolocalización, etc.
4. Extracción de los rasgos característicos del conjunto de datos: Extraer los descriptores de los conjuntos de datos, como el número de filas y columnas, la dispersión, la distribución de las columnas entre los tipos de características, las características que identifican el rendimiento de algún modelo de ML en la submuestra, etc.
5. Preprocesado de datos: Como es necesario preparar los datos para que el modelo tenga la mayor precisión posible.
6. Selección del algoritmo: Qué algoritmo es el más apropiado para la resolución del problema.
7. Entrenamiento del modelo y ajuste de hiperparámetros: Selección del conjunto óptimo de hiperparámetros para el correcto rendimiento del modelo y entrenamiento posterior de este.
8. Validación del modelo: Prueba y validación del modelo para su posterior funcionamiento en un caso de uso real (datos desconocidos para el modelo).



Figura 3-1.- Pipeline de desarrollo de un modelo de machine learning (Ref. [5]).

Para la ejecución de las distintas etapas del flujo de desarrollo de un modelo de aprendizaje automático se necesita cierto conocimiento en estadística, matemáticas, e informática, es por tanto que, para las personas no expertas en estas áreas, o empresas que no pueden costearse un departamento en ciencia de datos e inteligencia artificial, el acceso a soluciones basadas en machine learning puede resultar en fracaso o directamente ser inviable [8]. Incluso disponiendo de la experiencia y la capacitación para usar métodos de inteligencia artificial, la utilidad y el rendimiento de muchos métodos de ML esta

grandemente condicionado a las decisiones de diseño. El ajuste de los modelos tiende a ser repetitivo, temporalmente costoso y dependiente de cada conjunto de datos, donde en la gran mayoría de ocasiones se recurre a ajustes de prueba y error. El campo del aprendizaje máquina automatizado (AutoML), tiene como objetivo automatizar todas estas decisiones utilizando los datos de entrada y de forma objetiva, converger en una solución particular a cada conjunto de datos. El uso de sistema de esta tecnología puede ocasionar un gran ahorro económico de recursos y temporal. Además, el AutoML puede ser visto, para las personas que carecen de los conocimientos necesarios o que no disponen de los recursos, como una democratización de la inteligencia artificial.

En el libro "Automated Machine Learning" [9] se hace una diferenciación entre (i) métodos de AutoML y (ii) sistemas de AutoML. Los métodos de AutoML se centran en automatizar una o varias etapas del pipeline de forma aislada, en cambio, los sistemas de AutoML automatizan todo el pipeline o gran parte, ofreciendo una solución integral con el que desarrollar un modelo funcional desde la entrada de los datos hasta la obtención de este (Figura 3-2).

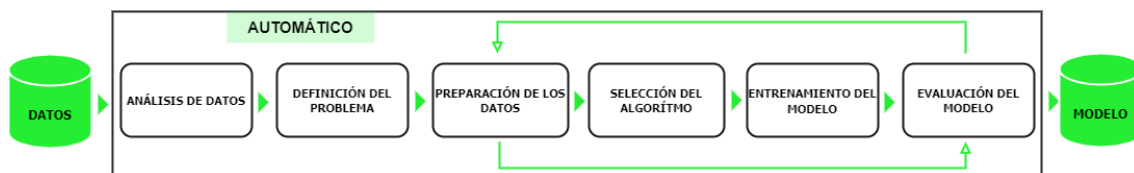


Figura 3-2.- Pipeline 'ideal' de la implementación de un modelo de ML de forma automática (AutoML)

Aunque también se utilizarán algunos métodos de AutoML, este proyecto se centrará especialmente en los sistemas de AutoML. Para comprender más detalladamente como funciona o de qué se compone un framework<sup>1</sup> de AutoML, es necesario explicar algunas de las etapas del flujo de desarrollo y que métodos se emplean en cada una de éstas.

Una de las etapas más importantes del pipeline es la del preprocesado de datos. Debido a varios factores en la obtención de los datos, pe.: los sensores presentan fallas en el instante de medición, un operador ejecutó errores en alguna etapa del

<sup>1</sup> En el desarrollo de software, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

proceso, o el programa de adquisición de datos presenta un comportamiento anómalo. Ante esta problemática puede ser necesario procesar los datos crudos para obtener una visión completa o más representativa de la realidad que están midiendo dichos datos. [10][11] Algunos de los pasos más comunes (Figura 3-3) a aplicar son:

- Limpieza de datos: Los datos pueden tener muchas partes irrelevantes o datos perdidos. Existen muchas soluciones, imputación de datos, estimación según datos vecinos (interpolación, series temporales, etc.)
- Transformación de los datos: Puede ser necesario transformar los datos originales para adecuarlos al método que utilizará los datos: normalización, discretización, selección de atributos, etc.
- Reducción de los datos: La cantidad o la dimensión de los datos puede ser muy grande, por tanto, en ocasiones puede ser necesario obtener una muestra representativa y de menor dimensión/tamaño que el conjunto original.

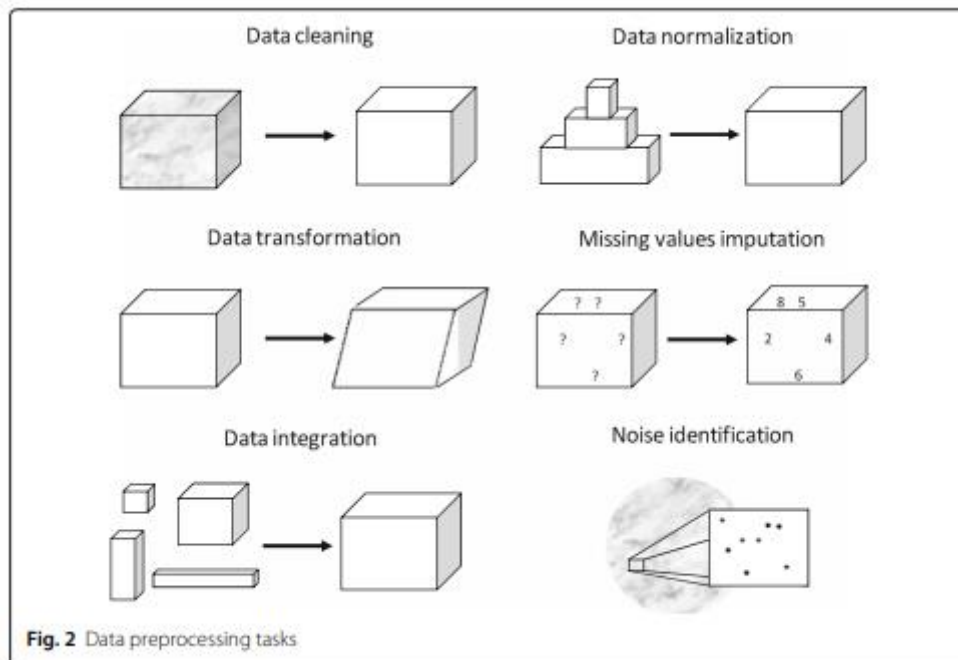


Figura 3-3.- Representación gráfica de los diferentes tratamientos de los datos en la etapa de preprocesamiento de datos.

En ocasiones este tratamiento de datos puede resultar muy sistemático, repetitivo y/o tedioso, es por ello que surgen métodos que tratan de simplificar y automatizar esta etapa del pipeline, ahorrando tiempo y esfuerzo al analista de datos.

Otra de las etapas importantes del pipeline es la de selección del algoritmo. [12][13][14] Para resolver un problema de aprendizaje automático, el humano debe seleccionar de forma manual un algoritmo de ML y establecer generalmente más de un parámetro denominados hiperparámetros<sup>2</sup>(Tabla 3.1). Para el correcto funcionamiento de estos algoritmos el usuario debe conocer ante qué tipo de problema se enfrenta, ya sea regresión o clasificación, supervisado o no supervisado, y tener una noción de los datos con los que funcionará el algoritmo. Las decisiones de diseño pueden tener un gran impacto en el rendimiento del modelo resultante, pero su selección requiere una experiencia especial, así como muchas iteraciones manuales que requieren mucha mano de obra. Al conjunto de las etapas, "Selección del algoritmo", "Entrenamiento del modelo" y "Evaluación del modelo" se le conoce como "Selección de algoritmos combinados y optimización de hiperparámetros" (en inglés Combined Algorithm Selection and Hyperparameter Optimization, CASH problem). Existe un gran interés en la literatura acerca de automatizar y hacer más eficientes los procesos que componen las etapas del problema CASH.

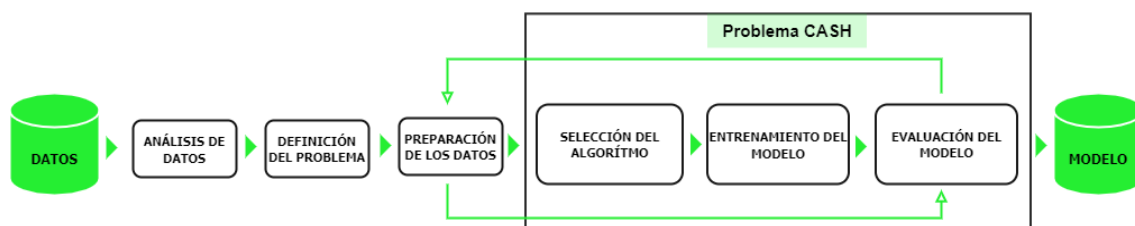


Figura 3-4 Pipeline con CASH problem.

Por otro lado, también es importante la parte de optimización de hiperparámetros. Según el pipeline indicado anteriormente (Figura 3-4), dentro de la etapa "Entrenamiento del modelo" se incluye la optimización de hiperparámetros (en inglés Hyperparameter Optimization, HPO). El rendimiento de los modelos de inteligencia artificial se ve grandemente condicionado al ajuste de los diferentes

<sup>2</sup> Los hiperparámetros de un modelo son los valores de las configuraciones utilizadas durante el proceso de entrenamiento. Son valores que generalmente se no se obtienen de los datos, por lo que suelen ser indicados por el científico de datos. El valor óptimo de un hiperparámetro no se puede conocer a priori para un problema dado.

parámetros que lo componen. En ocasiones el espacio de búsqueda de los valores posibles para cada hiperparámetro es infinito, por tanto, es necesario recurrir a técnicas inteligentes para la búsqueda de los valores óptimos. Un caso particular, que también tiene mucho interés en la literatura, es el de “Búsqueda de Arquitectura de Redes Neuronales” (en inglés Neural Network Search, NAS), que tiene como objetivo la optimización de los parámetros característicos de las redes neuronales (arquitectura). El problema CASH y métodos más concretos como pueden ser el HPO y NAS, son estudiados para su automatización, ya que son parte fundamental del pipeline del modelo.

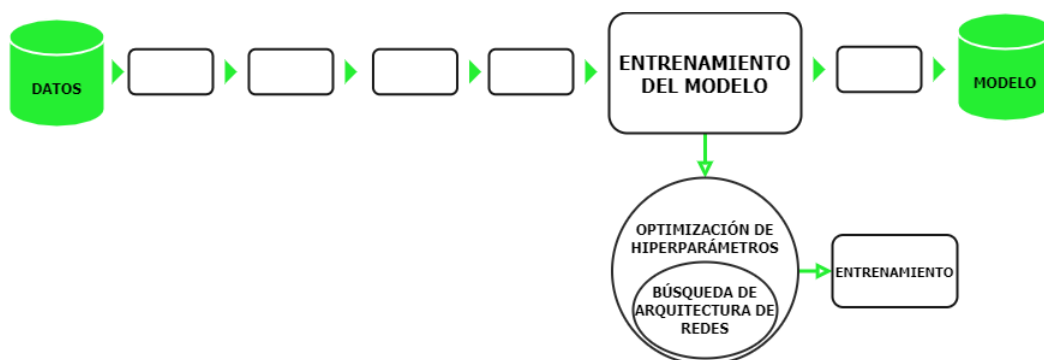


Figura 3-5 Pipeline con HPO y NAS.

Por último, un método de gran interés en el desarrollo del pipeline es el “Meta-Learning”<sup>3</sup> [15]. En cada nueva tarea de ML se crea un pipeline individualizado desde cero, pero, al igual que un experto adquiere experiencia con cada tarea, es posible aprender de anteriores soluciones para no partir de cero, y hacer este proceso de forma más eficiente y con una mayor probabilidad de éxito. El campo del Meta-Learning consiste en capacitar a los algoritmos a modificar su comportamiento en base a pasadas tareas. Este aprendizaje puede suponer una mejora en:

- Espacio de búsqueda: Reducir/aumentar o refinar el espacio de búsqueda de hiperparámetros.

<sup>3</sup> El metaaprendizaje es un subcampo del aprendizaje automático en el que se aplican algoritmos de aprendizaje automático a los metadatos sobre experimentos de aprendizaje automático. El objetivo es entender como resolver problemas de aprendizaje con mayor flexibilidad, mejorando el desempeño de algoritmos existentes o induciendo al algoritmo de aprendizaje en si.

- Algoritmos candidatos: Reducir/aumentar la lista de posibles algoritmos en base a resultados anteriores, evitando ejecuciones no válidas o insatisfactorias.
- Comienzo en caliente: Se puede empezar con modelos previamente entrenados o una secuencia de hiperparámetros ya utilizada en tareas similares a la que se quiere resolver.
- Estructura del pipeline: Se puede adaptar la estructura del pipeline bajo demanda en base a la experiencia con otras tareas, no realizar el paso de limpieza de datos, o utilizar una codificación de variables categóricas concreta, etc.



Algoritmo	Ejemplo parámetro ordinario	Ejemplo hiperparámetro
Decision tree	The input variable used at each internal node, the threshold value chosen at each internal node	The minimal number of data instances at a leaf node, the pruning strategy for the tree after training
Random forest	The input variable used at each internal node of a decision tree, the threshold value chosen at each internal node of a decision tree	The number of decision trees, the number of input variables to consider at each internal node of a decision tree
Support vector machine	The support vectors, the Lagrange multiplier for each support vector	The kernel to use, the degree of a polynomial kernel, the regularization constant C, the $\epsilon$ for round-off error, the tolerance parameter
Neural network	The weight on each edge	The number of hidden layers, the number of nodes on each hidden layer, the number of epochs to train through, the learning rate for the backpropagation algorithm
k-nearest neighbor		The number of nearest neighbors used (k), the mechanism of weighting the nearest neighbors, the distance function to use
Naïve Bayes	The probability of an input variable taking on a particular value given a specific class $p(x_i   c)$ , the prior probability of each class $p(c)$	Whether kernel density estimator or normal distribution is used for numeric attributes, the window width of a kernel density estimator
Multiboost with decision stumps	The input variable used by each decision stump, the threshold value chosen by each decision stump, the weight of each decision stump	The number of iterations, the number of sub-committees, whether resampling is used for boosting

Tabla 3.1.- Ejemplo de algoritmos de ML, sus parámetros ordinarios e hiperparámetros (Ref. [14])

En el segundo de los enfoques (sistemas AutoML) se busca desarrollar sistemas que incluyan y faciliten la utilización de los métodos de AutoML. A diferencia de los métodos de AutoML, este enfoque no se centra únicamente en una o algunas etapas del pipeline, sino en automatizar el proceso completo. Los sistemas de AutoML dan solución al desarrollo completo de un modelo de inteligencia artificial de forma integral. Es interesante remarcar que los sistemas de AutoML implementan los métodos antes mencionados, pero de una forma más transparente para el usuario, no siendo necesario conocerlos ni saber cómo funcionan. Las herramientas de AutoML ofrecen, desde librerías de programación para los usuarios más expertos, hasta interfaces de usuario (Figura 3-6, Figura 3-7) amigables para que puedan ser usadas por una mayor variedad de usuarios.



Figura 3-6.- Ejemplo de GUI de la herramienta H2O.ai Driverless.

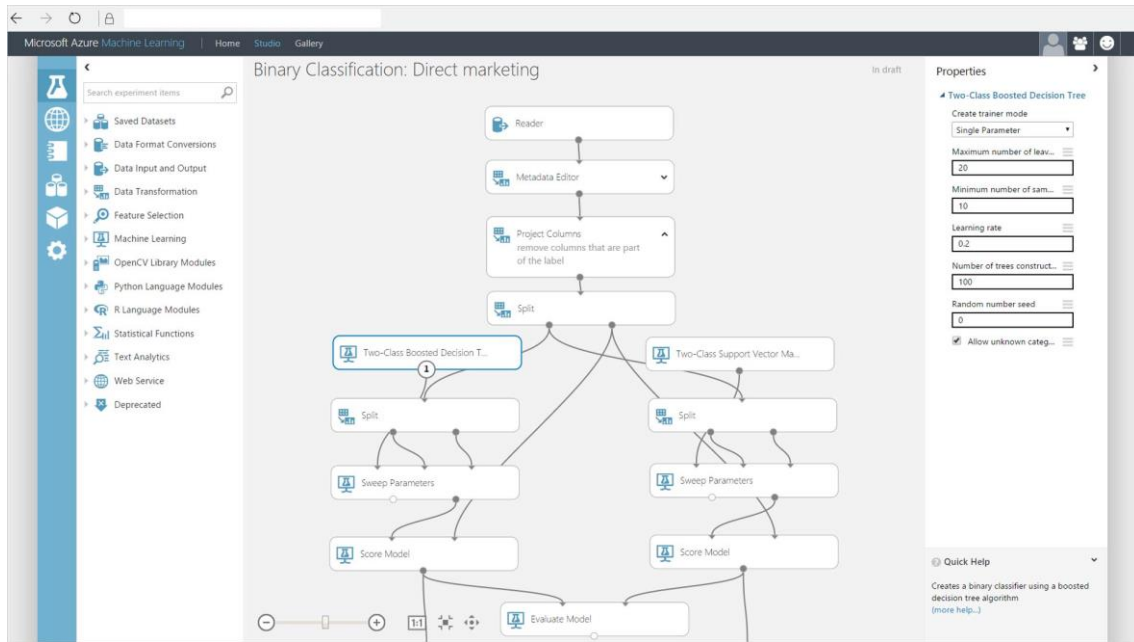


Figura 3-7.- Ejemplo de GUI de la herramienta Azure Machine Learning.

Aunque solo es objetivo de este trabajo las etapas de desarrollo y entrenamiento del modelo, los sistemas de AutoML dan soporte también en otras fases del ciclo de vida (Figura 3-8) de un modelo de machine learning. Una vez desarrollado y determinado que el modelo cumple su función y ofrece resultados aceptables, éste puede ser sacado a producción, lo que se conoce como la fase de despliegue del modelo. El modelo puede ser desplegado sobre una plataforma móvil, un servidor, etc. Otro aspecto importante es la monitorización y control de versiones del modelo. Estas etapas son de gran interés para las compañías, automatizar estas tareas puede resultar en grandes beneficios de tiempo, coste y eficiencia. En la actualidad, grandes y pequeñas compañías están dedicando mucho esfuerzo en desarrollar herramientas competitivas que aporten valor en estos aspectos (Figura 3-9, Figura 3-10).



Figura 3-8.- Ciclo de vida de un modelo de ML.

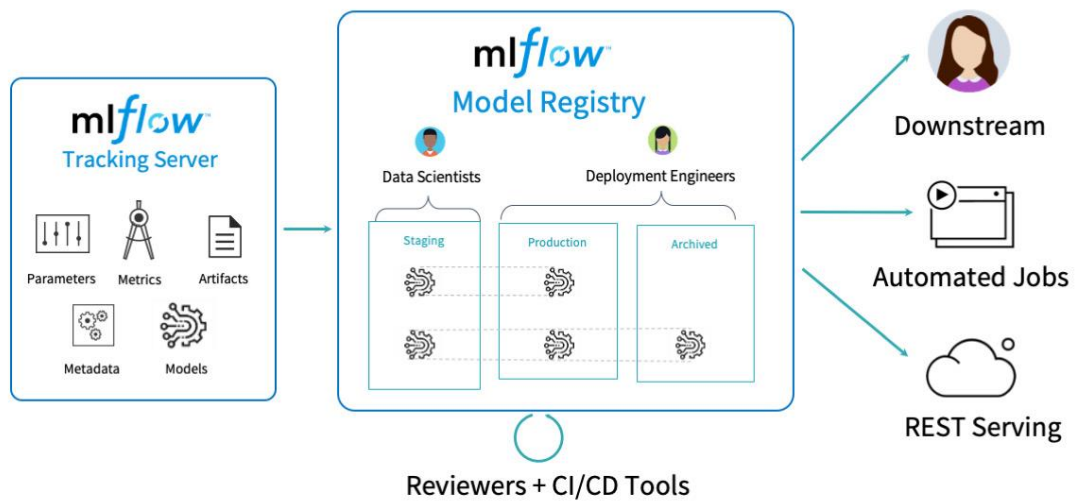


Figura 3-9.- Esquema de funcionamiento de la aplicación MLFLOW.

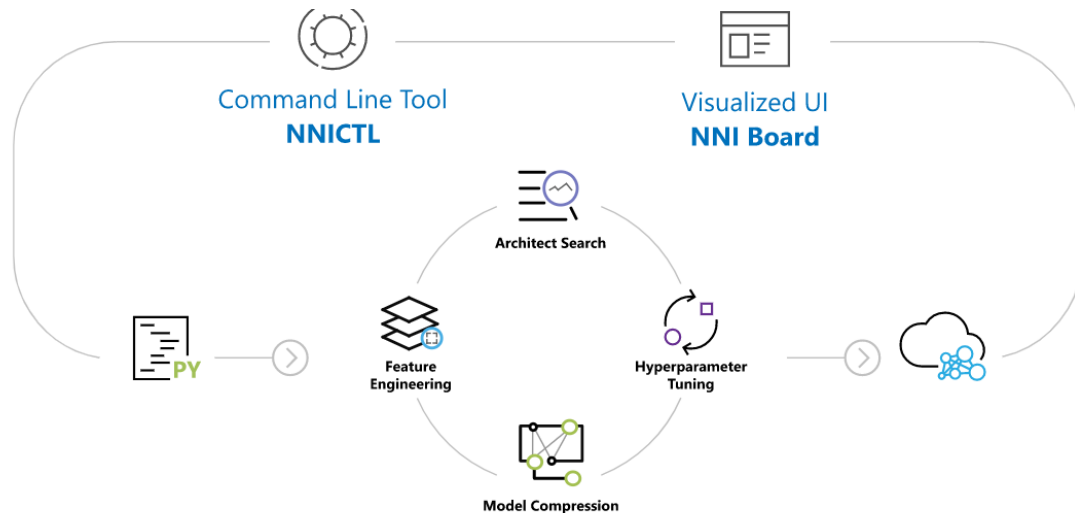


Figura 3-10.- Esquema de funcionamiento de la aplicación NNI.

### 3.2.- Estudio de alternativas.

Se han desarrollado una gran variedad de soluciones de código abierto y comerciales en los últimos años. Cada una de estas herramientas tiene como objetivo extraer valor de los datos de forma rápida y más fácil posible. Estas plataformas ofrecen diversas funcionalidades que satisfacen distintas tareas asociadas a la construcción e implementación del pipeline de un modelo de ML [16].

En la literatura de los últimos años (2018-2020) hasta hoy, se han recogido, estudiado, y comparado las diversas alternativas. El colegio de ciencia de datos y estadística de la universidad de Siegen (abril 2020) propone una lista de 14 herramientas [17]. AIMultiple [18], el blog sobre transformación digital (enero 2021) recoge 18 herramientas atendiendo a si éstas son de acceso libre o requieren de pago o suscripción. En esta publicación se hace una división en base a la compañía dueña de las herramientas: "Open Source", "Startup" o "Gigante Tecnológico" (Tabla 11.1)

[19] En 2018 se publicó el primer artículo realizando una evaluación comparativa (benchmark<sup>4</sup>) de 4 herramientas (*Auto\_ml*, *Auto-sklearn*, *TPOt* y *H2O*). Aunque en

<sup>4</sup> Es una técnica utilizada para medir el rendimiento de un sistema o uno de sus componentes. Más formalmente puede entenderse que una prueba de rendimiento es el resultado de la ejecución de un programa informático o

este benchmark se compare únicamente el desempeño en términos de precisión de los algoritmos, las herramientas pueden realizar más tareas como, ingeniería de características, preprocesado de datos, codificación de características categóricas y escalado numérico.

Para probar estas plataformas se utilizaron conjuntos de datos de la librería de acceso abierto "OpenML" [20], un total de 57 conjuntos de datos para clasificación y 30 para regresión (Figura 3-12).

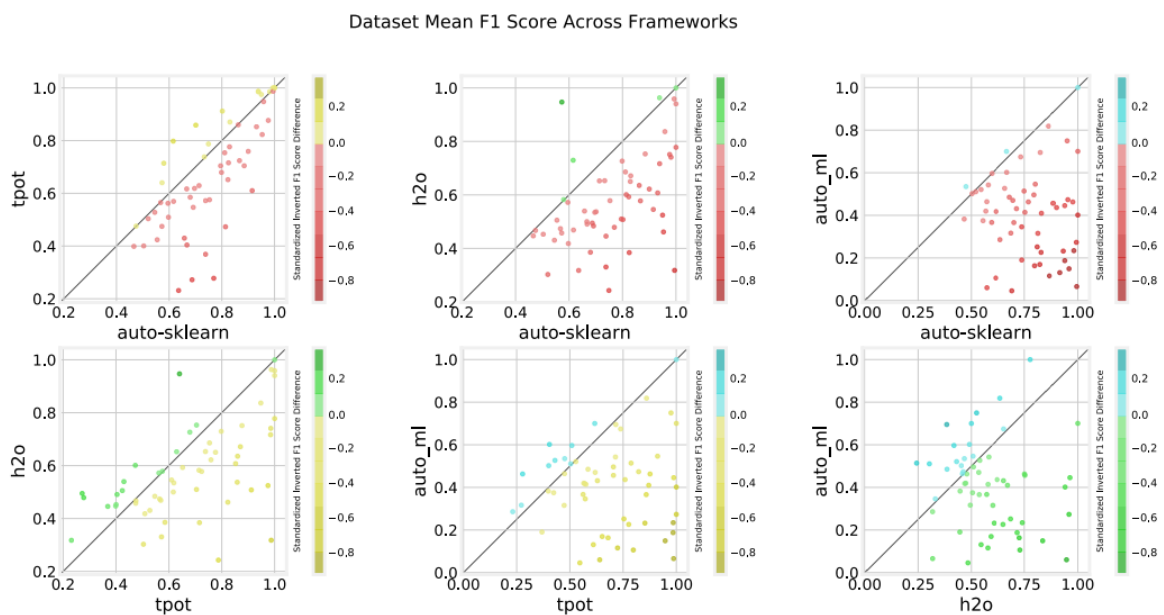


Figura 3-11.- Comparativa cara-a-cara del F1-score entre distintas herramientas aplicadas a un problema de clasificación. (Ref. [19])

---

un conjunto de programas en una máquina, con el objetivo de estimar el rendimiento de un elemento concreto, y poder comparar los resultados con máquinas similares.

Dataset Mean MSE Across Frameworks

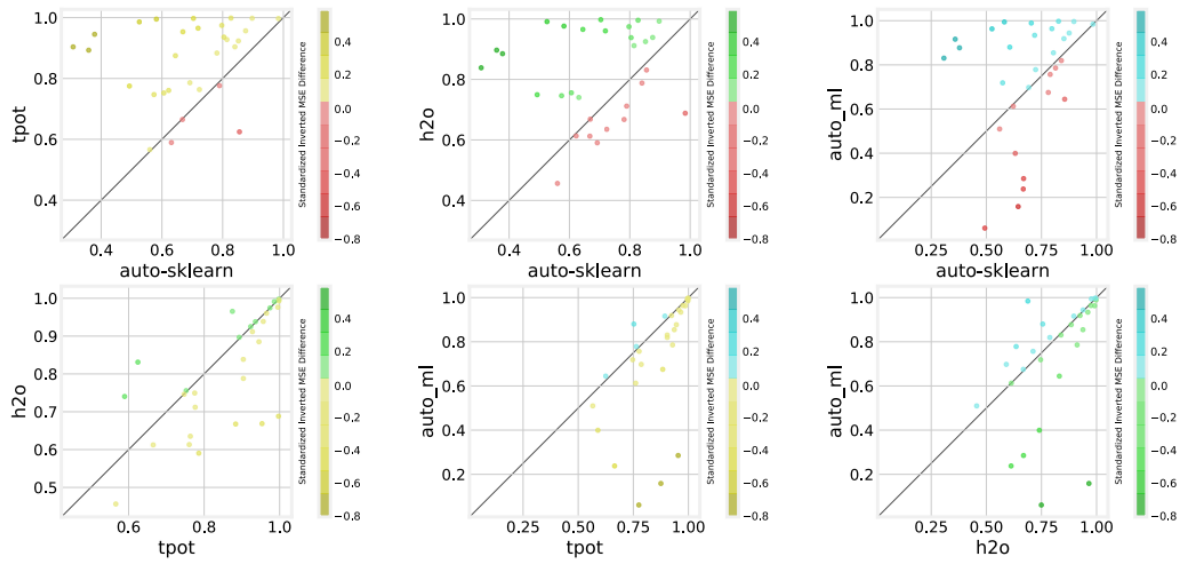


Figura 3-12.- Comparativa cara-a-cara del RMSE-score entre distintas herramientas aplicadas a un problema de regresión. (Ref. [19])

(Figura 3-13) En el artículo se determina que “auto-sklearn” rinde mejor en clasificación que el resto de frameworks, y “TPOT” lo hace mejor en los problemas de regresión, aunque existe una gran varianza en los resultados y se propone como trabajo futuro un estudio más granular acerca de la ingeniería de características, selección de modelos y la optimización de parámetros, proponiendo también el estudio de un mayor abanico de plataformas.

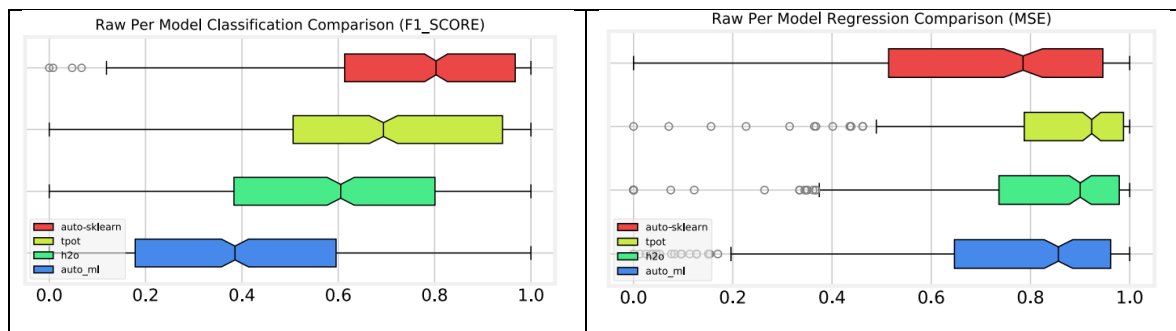


Figura 3-13.- Desempeño de las herramientas para todos los conjuntos de datos, a la izquierda para los problemas de clasificación y la derecha para los de regresión.

[15] En 2020 se estudia un conjunto de soluciones de código abierto para el automatizado del ML. Se realiza una comparativa de un conjunto de 5 herramientas de AutoML (*TPOT*, *Hyperopt-sklearn*, *Auto-sklearn*, *ATM* y *H2O*), en esta publicación, al igual que la anterior presentada, se centra exclusivamente en la resolución de las etapas de selección, entrenamiento y evaluación de algoritmos

(problema CASH). Estas herramientas son comparadas frente a 10 algoritmos aplicados por expertos (*Dummy, Random Forest, Grid Search, Random Search, RoBO, BTB, Hyperopt, SMAC, BOHB, Optunity*). Además de los resultados, en esta publicación se recoge de forma resumida como es la estructura y las características del pipeline que construyen las herramientas (Tabla 3.2):

Framework	CASH Solver	Structure	Ensem.	Cat.	Parallel	Time.
DUMMY	–	Fixed	no	no	no	no
RANDOM FOREST	–	Fixed	no	no	no	no
TPOT	Genetic Prog.	Variable	no	no	Local	yes
HPSKLEARN	HYPEROPT	Fixed	no	yes	no	yes
AUTO-SKLEARN	SMAC	Fixed	yes	Enc.	Cluster	yes
RANDOM SEARCH	Random Search	Fixed	no	Enc.	Cluster	yes
ATM	BTB	Fixed	no	yes	Cluster	no
H2O AUTOML	Grid Search	Fixed	yes	yes	Cluster	yes

Tabla 3.2.- Características del pipeline de las herramientas de AutoML (Ref. [15]).

- Estructura fija/variable: El pipeline es una secuencia lineal de múltiples pasos de limpieza de datos, selección de características, preprocesamiento y modelado. La complejidad frente a pipeline de estructura variable es muy inferior, pero puede resultar en un menor rendimiento.
- Ensamblado: Un método de ensamblado combina múltiples modelos de ML para realizar las predicciones. El coste computacional de la evaluación es generalmente mayor al de un modelo individual, en cambio, la precisión puede verse incrementada significativamente.
- Variables Categóricas: Es posible o no la utilización de variables categóricas en los datos de entrada.
- Paralelización: Se construyen/evalúan múltiples pipelines de forma paralela, haciendo más eficiente la búsqueda del óptimo y ahorrando coste temporal.
- Tiempo: Es posible o no establecer un límite de tiempo en la búsqueda de modelos.



Respecto a los resultados cuantitativos, en este artículo se concluye que todas las herramientas de AutoML tienen un rendimiento similar, con una diferencia de solo el 2.2% del peor/mejor caso. La precisión de las herramientas difiere en un 6.7% en promedio en los diferentes conjuntos de datos. Por otro lado, se ha observado que los algoritmos CASH tienen un rendimiento mayor que las herramientas de AutoML en el 48% de los conjuntos de datos.

	TPOT	HPSKLEARN	AUTO-SKLEARN	Random	ATM	H2O
TPOT	—	0.7571	0.6086	0.8529	0.6000	0.5000
HPSKLEARN	0.2285	—	0.2816	0.5571	0.4117	0.2898
AUTO-SKLEARN	0.3623	0.7042	—	0.8000	0.4848	0.5294
Random	0.1323	0.4428	0.2000	—	0.3846	0.3283
ATM	0.3692	0.5735	0.4848	0.6153	—	0.4687
H2O	0.4705	0.7101	0.4558	0.6716	0.5156	—
Avg. Rank	2.6027	4.0410	2.9863	4.4109	3.4931	3.1643

Tabla 3.3.- Fracción de conjuntos de datos en los que el framework en cada fila rindió mejor que el framework en cada columna. (Ref. [15])

Comparando las plataformas de AutoML frente a expertos, todos los algoritmos son superados por el humano. En su lugar un humano experto requiere 8.57 horas de media para obtener resultados notables, frente a 1 hora que requieren las plataformas.

	<i>Otto</i>			<i>Santander</i>		
	Validation	Test	Ranking	Validation	Test	Ranking
Human	—	0.38055	—	—	0.84532	—
TPOT	0.81066	1.05085	0.7908	0.83279	0.83100	0.6827
HPSKLEARN	0.81177	0.58701	0.6216	0.66170	0.64493	0.8789
AUTO-SKLEARN	0.55469	0.55081	0.5155	0.83547	0.83346	0.6543
Random	0.88702	0.89943	0.7777	0.82806	0.82427	0.7235
ATM	0.74912	2.43115	0.8459	0.68721	0.69043	0.8653
H2O	0.45523	0.49628	0.3774	0.83406	0.83829	0.5329

Tabla 3.4.- Comparación de expertos frente a herramientas de AutoML para dos conjuntos de datos (Otto: valores pequeños es mejor | Santander: valores mayores es mejor) (Ref. [15])

En adicción se remarca que la resolución del problema CASH es solo una porción del pipeline que compone el desarrollo e implementación de un modelo de ML (Figura 3-4), un científico de datos utiliza entre el 60-80% del tiempo en la limpieza de los datos y selección de características, y solo el 4% en el ajuste de los modelos.

En [21] (2019) se comparan 4 plataformas gratuitas de AutoML (*Auto-WEKA*, *auto-sklearn*, *TPOt* y *H2O AutoML*). Las herramientas se seleccionaron en base a su popularidad. En este artículo únicamente se resuelven problemas de clasificación, con un total de 39 conjuntos de datos de OpenML. Se realizaron 2 pruebas, con un tiempo máximo de 1 hora y de 4 horas. No se determinaron diferencias notables en las ejecuciones atendiendo al tiempo, salvo una pequeña mejora en la clasificación del framework TPOt. Por otro lado, *Auto-WEKA* mostró signos de sobreajuste en las ejecuciones de más duración, sobre todo en los problemas de clasificación multi-clase. Como conclusión no hay ningún sistema de AutoML que destaque diferenciadamente sobre el resto, y se observa una gran varianza en los resultados, como ya se ha mencionado en anteriores benchmarks.

En ese mismo año (2019), en [22] se realiza otra evaluación comparativa más exhaustiva y considerando más aspectos de las plataformas de AutoML. En el artículo se recoge lo siguiente: (i) que funcionalidades de ML proporcionan las herramientas; (ii) como se desempeñan las herramientas frente a problemas reales; (iii) encontrar el equilibrio entre la optimización del tiempo y la precisión; (iv) la reproducibilidad de los resultados (robustez).

El preprocesado de datos es habitualmente el primer paso en el pipeline del ML. En el artículo se remarca que esta funcionalidad no cumple su propósito de forma robusta, y requiere aún mucha intervención humana. Además, no todas las herramientas comparadas disponen de ello, como es el caso de TPOt y *Auto-keras*. Por otro lado, *H2O-AutoML*, *H2O DriverlessAI*, *DataRobot*, *MLjar* y *Darwin* en su lugar, ofrecen la posibilidad de detectar tipos de datos básicos o esquemas (numéricos, categóricos y series temporales), en contrapartida *Auto-ml*, *Auto-sklearn*, *AzureML* y *Ludwig*, que requieren de la especificación del tipo de dato por el usuario de forma manual. Con respecto a los algoritmos, se incluyen en estas herramientas: regresión logística, algoritmos basados en árboles, SVM y redes neuronales, entre otros. Además, herramientas como *DataRobot*, *H2O-DriverlessAI* y *Darwin* pueden trabajar con métodos de ajuste no supervisado, como "clustering" y detección de

“outliers”. En la tabla Tabla 3.6 se reúnen las características más destacables del AutoML para cada herramienta.

Por último, destacar que la mayoría de las plataformas comerciales como H2O-DriverlessAI, DataRobot y Darwin ofrecen la funcionalidad de interpretado del modelo y análisis de resultados, posibilitando la utilidad de representación de resultados mediante “dashboards” (Figura 3-6, Figura 3-7, Figura 3-14), importancia de características y distintos métodos de visualización, funcionalidades que las herramientas no comerciales en su mayoría no ofertan, centrándose exclusivamente en la resolución de las etapas del problema CASH (construcción, entrenamiento y evaluación del modelo).

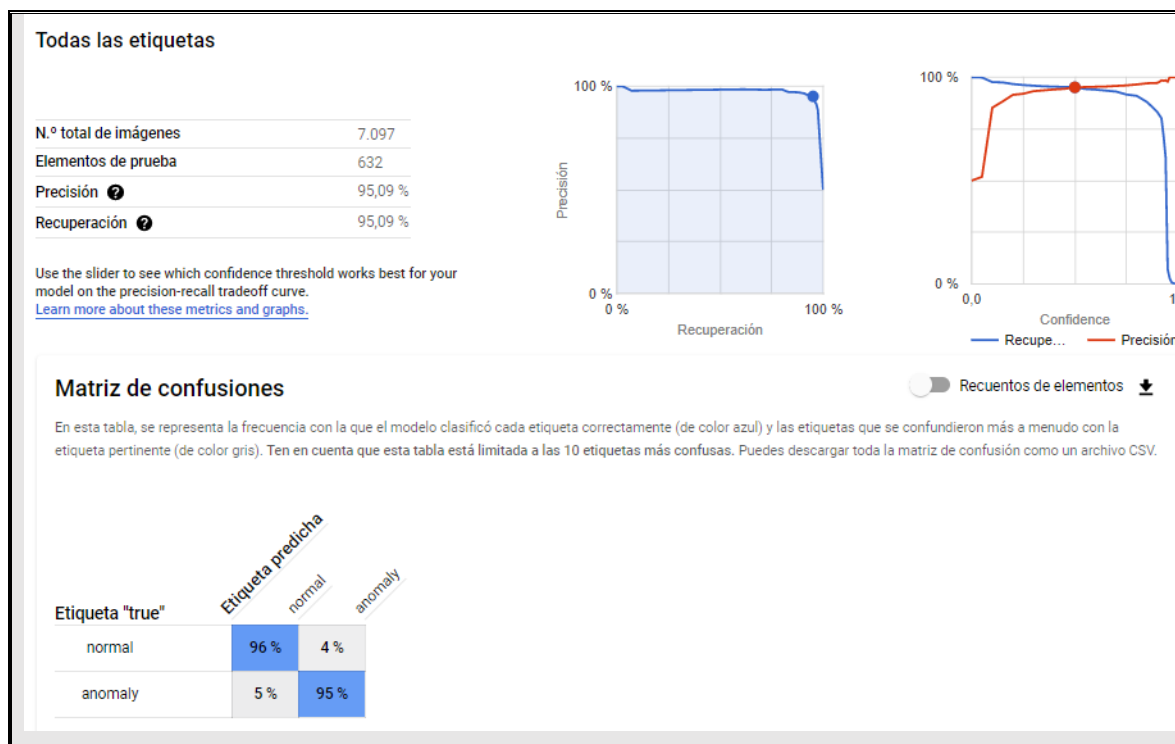


Figura 3-14.- Fragmento de dashboard de la herramienta Google Cloud Vision.

Con referente a los resultados cuantitativos obtenidos en el artículo, se han utilizado un total de 300 conjuntos de datos de OpenML. Las herramientas H2O-Automl y Ludwig tienen un rendimiento estable para los distintos problemas (regresión, clasificación binaria y multi-clase). En cambio, Darwin, Auto-keras y Auto-ml presentan un rendimiento menos estable.

Como aspecto común, ninguna herramienta resalta notablemente respecto del resto, como resumen general H2O-Automl, Auto-keras y Auto-sklearn rinden mejor que Ludwig, Darwin, TPOT y Auto-ml. H2O-Automl destaca ligeramente en

clasificación binaria y regresión, convergiendo más rápido. Auto-keras es estable en todas las tareas y destaca ligeramente en problemas de clasificación multi-clase y empatando con H2O-Automl en regresión.

En [23] se hace una comparativa basándose en la robustez de 3 herramientas (TPOT, H2O AutoML y AutoKeras) aplicando clasificación de imágenes. En esta publicación se menciona la poca madurez o temprana etapa en la que se encuentra el paradigma del aprendizaje máquina automatizado, y que en los próximos años se verá un incremento de publicaciones centradas en el estudio de la robustez de los frameworks. Para la selección de las herramientas se han basado en los siguientes criterios:

- Todos proporcionan una simple API de Python y el uso básico requiere apenas unas pocas líneas de código.
- El enfoque del desarrollo del pipeline de cada herramienta difiere lo suficiente para realizar una comparativa.
- No requieren conocimientos previos de los datos y no es necesario indicar ningún espacio de búsqueda para los modelos e hiperparámetros, haciendo estas herramientas fáciles para usuarios ocasionales.

Para la comparativa se usan dos conjuntos de imágenes, las cuales se alteran de forma aleatoria, rotándolas y añadiendo un ruido gaussiano (Figura 3-15). AutoKeras destaca notablemente en la mayoría de las pruebas, tanto en precisión como en tiempo de ejecución. A diferencia de H2O y TPOT, AutoKeras ofrece soporte completo para el uso de procesador gráfico (GPU), lo que reduce los tiempos de entrenamiento de los modelos. Además, AutoKeras cuenta con la funcionalidad de "data augmentation"<sup>5</sup>, consiguiendo en los modelos resultantes, una mayor robustez ante ruido o alteraciones en la rotación de las imágenes que se usan en la clasificación. Con respecto al uso de recursos, AutoKeras y H2O utilizan aproximadamente el 100% de los recursos disponibles por otro lado, TPOT presenta problemas en el rendimiento al ser ejecutado sobre múltiples núcleos, usando tan solo un 30% de la CPU.

---

<sup>5</sup> El aumento de datos en el análisis de datos son técnicas que se utilizan para aumentar la cantidad de datos agregando copias ligeramente modificadas de datos ya existentes o datos sintéticos recién creados a partir de datos existentes.

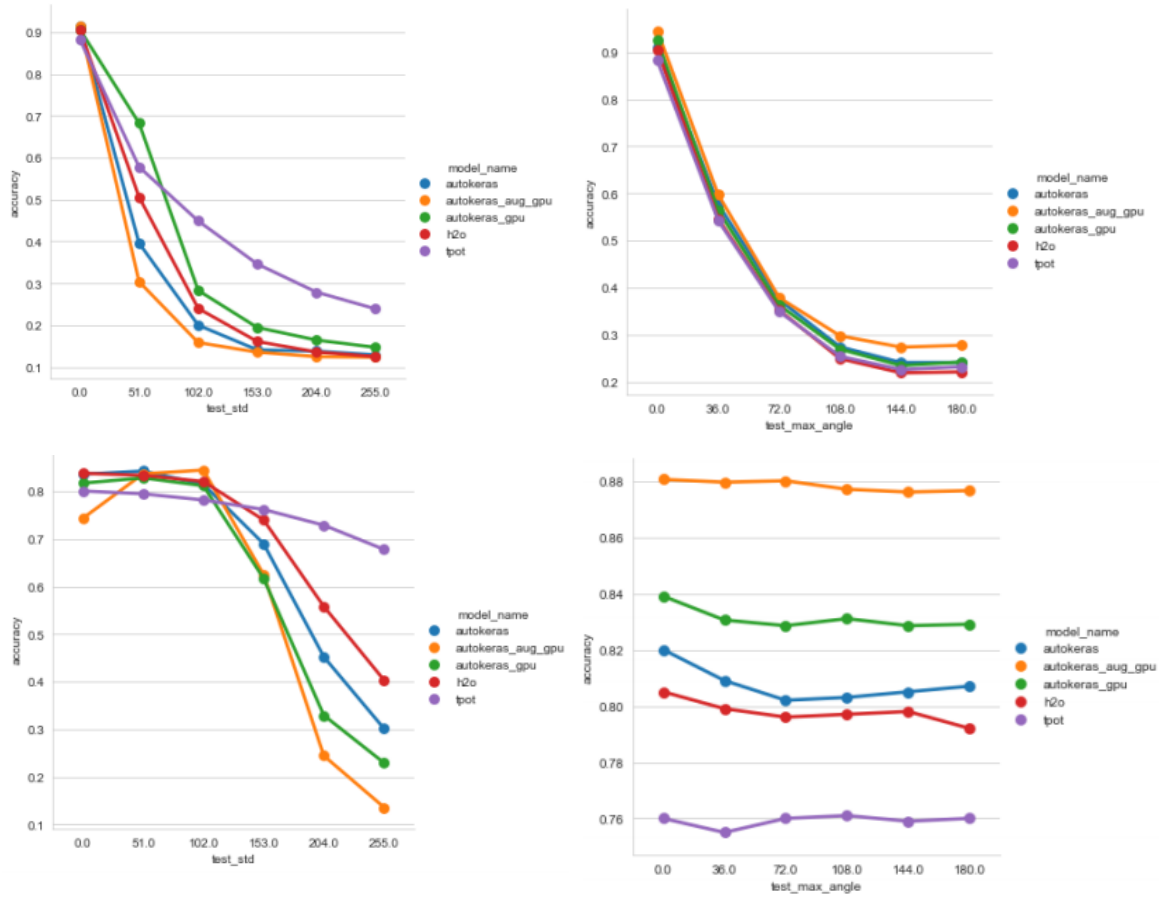


Figura 3-15.- Precisión de cada herramienta de AutoML comparando la robustez. (Arriba | Izquierda) Con ruido en los datos de test y datos de train limpios. (Arriba | Derecha) Con rotación en los datos de test y datos de train limpios. (Abajo | Izquierda) Con ruido en los datos de test y ruido en datos de train. (Abajo | Derecha) Con rotación en los datos de test y rotación en los datos de train.

Con respecto al procesamiento de texto, en [24] se evalúan 4 herramientas (TPOT, H2O, auto-sklearn y AutoGluon), además de comparar los resultados obtenidos frente al desempeño humano. En esta publicación se utilizan 13 conjuntos de datos populares incluyendo competiciones de Kaggle. En esta comparativa se utilizan todas las herramientas como cajas negras, procurando mantener el nivel máximo de abstracción posible para el uso de cada herramienta. AutoGluon es la única de estas que permite la entrada de texto en crudo, sin preprocesamiento necesario. En 4 de 13 ocasiones (Tabla 3.5) al menos una de las herramientas de AutoML logran una mejor precisión que el humano. AutoGluon tiene un mejor rendimiento que el resto de las herramientas, con un margen del 8.7% de media y 7 victorias. Le sigue auto-sklearn con 5, H2O con 1 y TPOT, esta última con 0 victorias. Frente al desempeño humano, en los 4 casos donde el AutoML obtuvo mejores resultados el margen es de 1.4% de media.

Table 2: Best performing AutoML tools with and without AutoGluon Text

Number	Best score by AutoML (acc)	Number of AutoML tools better or equal	Best known human score (acc)
1	0.989	4	0.966 <sup>4</sup>
2	0.708	0	0.776 <sup>5</sup>
3	0.715	0	0.829 <sup>6</sup>
4	0.946	0	0.962 <sup>7</sup>
5	0.837	0	0.87 <sup>8</sup>
6	0.657	0	0.836 <sup>9</sup>
7	0.862	0	0.926 <sup>10</sup>
8	0.961	3	0.944 <sup>11</sup>
9	0.171	2	0.169 <sup>12</sup>
10	0.52	1	0.519 <sup>13</sup>
11	0.653 (F1)	0	0.713 <sup>14</sup> (F1)
12	0.768 (F1)	0	1 <sup>15</sup> (F1)
13	0.718	0	0.832 <sup>16</sup>

Tabla 3.5.- Precisión de las herramientas de AutoML en el procesamiento de texto para los distintos conjuntos de datos. (Ref. [24])

Según las conclusiones de la publicación, el AutoML actualmente no es capaz de superar en rendimiento al humano en clasificación de texto, estos frameworks pueden facilitar el uso del ML para principiantes y establecer la línea base para usuarios más avanzados.

Tool	Platform	Input data sources		Data pre-processing	Data types detected					Feature engineering				ML Tasks		Model selection and Hyperparameter optimization					Quick start / early stop			Model evaluation / Result analysis/ Visualization		
		Spreadsheet datasets	Image, text		Numerical	Categorical	Datetime	Time-series	Other (hierarchical types) (7*)	Datetime, categorical processing	Imbalance, missing values	Feature selection, reduction	Advanced feature extraction (8*)	Supervised learning (9*)	Unsupervised learning (10*)	Ensemble	Genetic algorithm	Random search	Bayesian search	Neural architecture search	Quick finding of starting model	Allow maximum limit search time	Restrict time consuming combination of components	Model dashboard	Feature importance	Model explainability and interpretation, and reason code (11*)
TransmogrifAI	Apache Spark	Y	N	Y(1*)	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Y	Y	N	N			Y	Y		
H2O-AutoML	AWS, GCP, Azure	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	N	Y	N	Y	N	Y	N	N	N	Y	Y	Y	Y	Y
Darwin (+)	GCP	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	N	Y	Y	Y
DataRobot (+)	AWS, GCP, Azure	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y(12*)	Y		Y	Y	Y
Google AutoML (+)	Google Cloud	N	Y	Y						N	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Auto-sklearn		Y	N	N	N	N	N	N	N	Y(2*)	Y	Y	Y	Y	N	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y
MLjar (+)	MLJAR Cloud	Y(3*)	N	Y	Y	Y	N	N	N	Y	Y(4*)	N	N	Y(5*)	N	Y	N	Y	N	N	N	N	N	Y	Y	N
Auto_ml		Y	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	N	Y	N	Y	Y	N	N	N	N	Y	Y	Y
TPOT		Y	N	N	N	N	N	N	N	N	Y	N	Y	Y	N	Y	Y	N	N	N	N	Y	N	Y	Y	N
Auto-keras		Y	Y	N	N	N	N	N	N	N	Y	Y	N	Y	N	N	N	Y	Y	Y	Y	Y	N	Y	N	Y
Ludwig		Y	Y	Y(1*)	Y	Y	N	Y	Y	N	Y	Y	Y	Y	N	Y	N	Y	Y	Y	Y	N	N	Y	Y	N
Auto-Weka		Y	N	N	Y	Y	N	N	N	N	Y	Y	N	Y	N	Y	N	Y	Y	N	N	Y	Y	Y	N	N
Azure ML (+)	Azure	Y	Y	Y(6*)	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y	N	Y	Y	N		Y	Y	Y	Y	
H2O-Driverless AI (+)	AWS, GCP, Azure	Y(3*)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y

Tabla 3.6.- Comparativa de las funcionalidades y características de las distintas herramientas de AutoML. (Ref. [22])

### 3.3.- Selección de alternativas.

Como objetivo se han seleccionado y probado diversas de las anteriores herramientas. Se han tenido en cuenta una lista de criterios para la selección de las herramientas. Estos criterios han sido definidos en función de las necesidades de la compañía asociada a este proyecto y a la adecuación de estas a los problemas que se tratarán más adelante en este documento. Las herramientas han de cumplir todos o la mayoría de los siguientes criterios:

- Madurez: En qué etapa del desarrollo se encuentra la herramienta (prototipo, estable, etc...) y si cuenta con todas las funcionalidades que oferta.
- Rendimiento/Benchmarking: Resultados/comparativas del rendimiento de la herramienta frente a sus competidores.
- Curva de aprendizaje: El tiempo necesario de un desarrollador en obtener un modelo aceptable o comparable frente al desarrollado con herramientas clásicas/manuales.
- Funcionalidades: Cuantas etapas del pipeline del desarrollo del modelo cubre la herramienta y que funcionalidades extra ofrece.
- Documentación disponible/actividad de la comunidad: Soporte/documentación disponible acerca de la herramienta y resolución de problemas.
- Flexibilidad/escalabilidad: El grado de abstracción posible en el uso de la herramienta, personalización, interoperabilidad con otras herramientas y escalado.
- Adecuación al problema: Si la herramienta es capaz de resolver los problemas relevantes para la compañía.



Se han probado un total de 13 herramientas<sup>6</sup>, de las cuales se han seleccionado 4 de ellas: AutoGluon, H2O, Google Cloud Platform y Amazon Web Services. Se analizarán más en detalle, en base a los problemas planteados, las funcionalidades que aportan dichas herramientas, posibles ventajas y desventajas y conclusiones, del uso de éstas frente a como se abordarían los problemas de una forma más clásica o manual.

Cabe destacar que los aspectos relacionados con rendimiento y requisitos de recursos no han sido medidos rigurosamente. La discusión referente a estos puntos será basada en la literatura (comparativas y benchmarking) e informalmente.

### **3.3.1.- Google Cloud Platform**

Google ofrece varias herramientas de aprendizaje máquina automatizado dentro de su plataforma "Google Cloud Platform (GCP)". Esta plataforma dispone de una gran variedad de funcionalidades dentro del ámbito del análisis de datos y la inteligencia artificial. En GCP es posible gestionar un almacén/base de datos en la nube, gestionar una red de máquina bajo una infraestructura virtualizada en la red, y desarrollar modelos de inteligencia artificial, así como gestionar su ciclo de vida. Los módulos que se utilizarán de esta plataforma son "Google Cloud Tables", para el modelado con datos tabulares, y "Google Cloud Vision", para problemas de visión artificial.

La herramienta de Cloud Tables permite el ajuste de datos tabulares para problemas de clasificación, binaria y multi-clase, y regresión. Cloud Vision, permite el entrenamiento de algoritmos para clasificación de imágenes pertenecientes a una y varias clases. Además es posible abordar problemas de detección de objetos en imágenes.

Todas las herramientas ofrecen una interfaz web. Todas las opciones ofrecen mensajes explicativos y preguntas y respuestas frecuentes resueltas en el portal de ayuda de la plataforma. Las opciones y parámetros para el desarrollo de los

---

<sup>6</sup> MLJar, TPOT, Auto-sklearn, Auto-keras, H2O-automl, Google Cloud Tables, Google Cloud Vision, Azure AutoML, Auto-weka, AutoGluon, McFly y Amazon Lookout Vision y Amazon Forecast.

modelos son limitadas, facilitando el uso para usuarios nóveles o sin conocimientos de inteligencia artificial, pero limitando la flexibilidad de uso de las herramientas.

Con estas herramientas es posible exportar los modelos entrenados en un contenedor Docker con TensorFlow. Además, la plataforma de Google permite el despliegue y monitorización de los modelos en la nube. La infraestructura cloud permite realizar predicciones online a través de su plataforma.

Las limitaciones técnicas respecto a los datos de entrenamiento de Google Cloud Tables son los siguientes: un máximo de 100GB de datos, entre 1000-200M registros (filas) y 2-1000 características (columnas). También existen limitación respecto al número de predicciones para los datos tabulares, y el número de imágenes clasificadas en el módulo de visión no podrá superar las 1800 imágenes como al minuto, y un tamaño máximo de 20MB por imagen.

Por último, esta herramienta está sujeta a un sistema de facturación por suscripción. Además se facturará por el entrenamiento, el espacio requerido por el modelo, el despliegue y por cada predicción, se facturará mientras el modelo esté alojado en su portal por el que se pueden realizar predicciones online.

### **3.3.2.- AutoGluon**

AutoGluon es un sistema de código libre gratuito [25]. Este sistema está basado en la interfaz Gluon de código abierto de las compañías Amazon Web Services y Microsoft. El rendimiento de este sistema ha sido sobresaliente en la literatura y benchmarking, además de contar con un gran apoyo de la comunidad y de la compañía desarrolladora.

Para utilizar AutoGluon es necesario disponer de un sistema con Python. AutoGluon es importado como una librería convencional de Python. Actualmente están disponibles tres módulos orientados a distintas tareas: el módulo "Tabular Prediction", para datos con forma tabular (csv, parquet, etc.), "Image Classification" para la clasificación de imágenes y "Text Prediction" para el entrenamiento supervisado con texto.

Todos los módulos de AutoGluon disponen de una función común (`'fit'`) que, con una sola llamada a esta, se realizan todas las etapas del pipeline: preprocesado de datos, ajuste de parámetros, entrenamiento del modelo y evaluación del

modelo. Los argumentos mínimos para esta función son: conjunto de datos de entrada (del tipo `'pandas'` o un tipo definido en la librería denominado `'TabularDataset'`), y el nombre dentro de la tabla de datos de la columna objetivo. AutoGluon también dispone de una gran variedad de parámetros para el ajuste de los métodos internos que forman cada etapa. Es posible modificar el tiempo límite de ejecución, la métrica de rendimiento, habilitar el ensamblado de modelos, el tipo de uso de memoria, entre otros.

AutoGluon permite flexibilidad respecto a entrenamiento de modelos y la evaluación y despliegue de estos, ya que utiliza librerías populares como "scikit-learn" y "mxnet". No es posible el entrenamiento con datos de más de 2 dimensiones y las predicciones de más de un valor (una dimensión).

### **3.3.3.- Amazon Web Services**

El sistema de AutoML de Amazon está disponible bajo la infraestructura de "Amazon Web Services (AWS)". AWS es una colección de servicios de computación en la nube como: gestión de almacenamiento cloud, gestión de bases de datos, virtualización distribuida, servicios de inteligencia de negocio e inteligencia artificial entre otros. Los servicios tratados en este proyecto son los referentes al AutoML, concretamente dos módulos, "Amazon Lookout Vision", para problemas de visión artificial, y "Amazon Forecast", para la predicción de valores con series temporales.

La herramienta de Lookout Vision permite la clasificación binaria de imágenes. Esta herramienta está orientada a la inspección de calidad mediante visión artificial, clasificando imágenes que contienen o no anomalías. Amazon Forecast, usa el aprendizaje automático para combinar datos de serie temporal con variables adicionales para crear previsiones. Amazon Forecast no requiere experiencia previa en aprendizaje automático para empezar a usarlo, solo es necesario proporcionar datos históricos, además de cualquier información adicional que se considere que pueda influir en sus previsiones.

El sistema de AWS ofrece un portal web con una interfaz de usuario de fácil uso. Al igual que el sistema de Google, todas las herramientas y funcionalidades ofrecen

menús de ayuda y soporte. También es posible la utilización del sistema a través de comandos de CLI de AWS.

Respecto a las limitaciones técnicas, el sistema de Lookout Vision solo soporta imágenes en formato PNG y JPG de dimensión máxima 4096x4096 píxeles y tamaño de 8MB. El conjunto de imágenes puede contener únicamente 16000 imágenes y el entrenamiento no puede superar las 24 horas. El sistema de Forecast admite datos de hasta 1 billon de registros (filas) y 25 características (columnas). El fichero no puede superar los 30GB.

Solo se factura en el entrenamiento y predicción del modelo, no es necesaria suscripción previa ni tasas fijas. Las cuotas están sujetas al país asociado.

### **3.3.4.- H2O**

H2O [26] es un sistema de código abierto que ofrece una serie de métodos para el desarrollo de modelos. El sistema de H2O es una API de Python bajo una infraestructura Java. Para utilizar H2O es necesario un sistema con Python y Java instalado. H2O es importado como una librería de Python convencional.

El sistema de H2O puede abordar problemas de regresión y clasificación de una y varias clases. H2O dispone de una función `'H2OAutoML'` que automatiza la selección de algoritmos y optimización de hiperparámetros. Esta función tiene una gran variedad de parámetros para modificar aspectos como, el número y tipo de algoritmos a utilizar, la estrategia de búsqueda y optimización, la utilización de recursos, etc. No es posible el entrenamiento con datos de más de 2 dimensiones y las predicciones de más de un valor (una dimensión).

El sistema de H2O ofrece una gran granularidad respecto a la implementación y modificación de los parámetros de entrenamiento, pero, a diferencia de la funcionalidad que ofrece más automatización (H2OAutoML), son requeridos conocimientos de modelado y programación.

## 4.- Descripción de los casos de uso.

### 4.1.- Caso 1: Tratamiento de datos de carácter desbalanceado

[27] Los centros de datos modernos albergan cientos de miles de servidores que coordinan las tareas para ofrecer servicios de computación en nube de alta disponibilidad. Estos servidores consisten en múltiples discos duros, módulos de memoria, tarjetas de red, procesadores, etc., cada uno de los cuales, aunque cuidadosamente diseñados, son capaces de fallar. Si bien la probabilidad de que se produzca un fallo de este tipo durante la vida útil (normalmente de 3 a 5 años en la industria) de un servidor puede ser algo pequeña, estas cifras se amplían en todos los dispositivos alojados en un centro de datos (Figura 4-1).

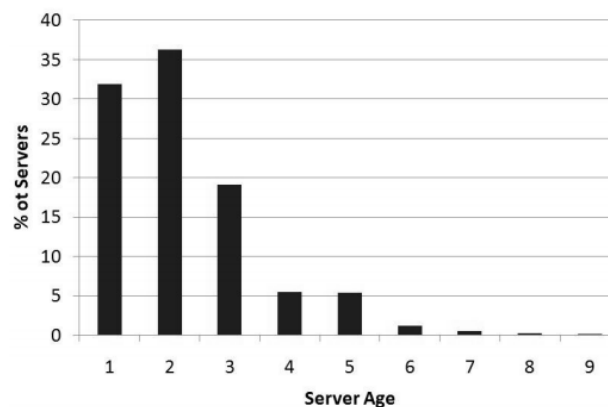


Figura 4-1.- Distribución de la vida útil (en años) de los discos duros de un servidor.

Un buen conocimiento de las cifras, así como de las causas de estos fallos, ayuda a mejorar la experiencia operativa, ya que no sólo permite una mejor equipación para tolerar los fallos, sino también reducir el coste del hardware mediante la ingeniería, lo que supone un ahorro directo para la empresa.

[28] Desde 1995 se ha implementado una tecnología estándar de autocontrol, análisis e informe (SMART) en gran parte de los discos duros modernos, los discos que incluyen la tecnología SMART monitorean su estado de vida y rendimiento (Figura 4-2). En muchos casos, el propio disco es capaz de advertir de un comportamiento anómalo, ayudado a evitar pérdida de información o una inutilización total inesperada.

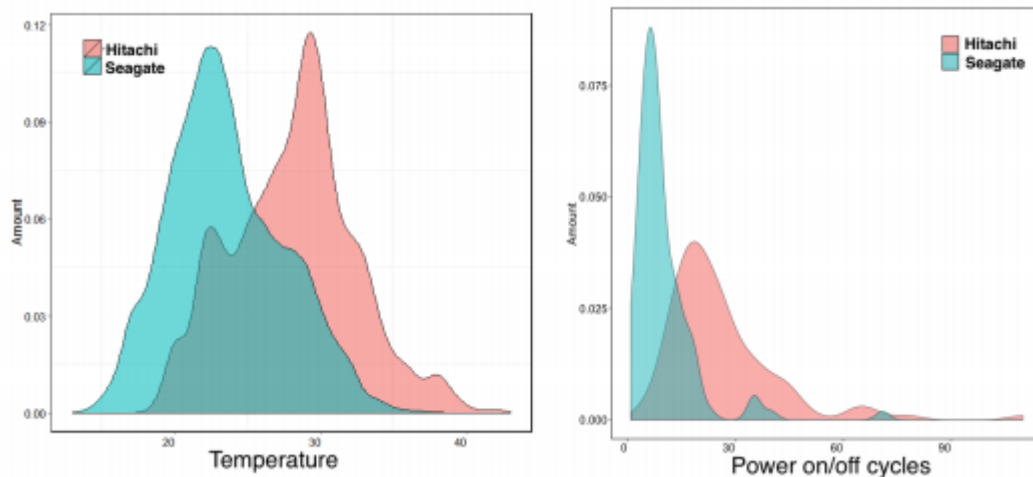


Figura 4-2.- Distribución de la temperatura y de los ciclos de encendido y apagado de los discos reemplazados de dos modelos distintos.

Para mejorar la precisión en la predicción de un fallo se han propuesto y desarrollado diversos métodos estadísticos y de inteligencia artificial, utilizando los atributos SMART como conjunto de datos [29], [30], [31], [32], [33]. En la literatura se proponen soluciones para la predicción del comportamiento anómalo del disco incluso un mes antes de que ocurra el fallo. Se han propuesto métodos que aporten un indicador del grado de deterioro, ajustándose así a las necesidades de las compañías.

Una de las principales características de los datos recopilados por la tecnología SMART es su naturaleza desbalanceada (Figura 4-3). El termino de desbalanceado se refiere a la cantidad de información asociada a cada clase (en este caso "sano"/"fallo") no está proporcionada. Un escenario balanceado tiene una proporción de aproximadamente 50% de registros para cada clase, en un sistema de larga escala como el de Microsoft Azure solo 3 discos de 10.000 fallan cada día.

El entrenamiento de los modelos se ve gravemente afectado con el uso de datos desbalanceados, por lo general afecta a los algoritmos en su proceso de generalización de la información, y perjudicando a la clase minoritaria.

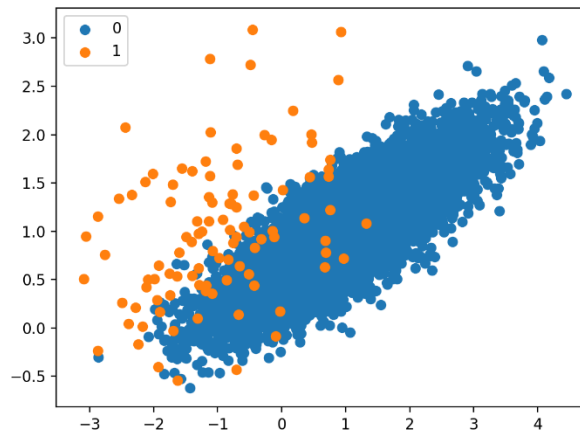


Figura 4-3.- Ejemplo gráfico de dos clases con proporción desbalanceada.

[31] Otro aspecto para tener en cuenta es el carácter temporal de los datos. Algunas características, especialmente las señales, son sensibles al tiempo (sus valores siguen cambiando drásticamente a lo largo del tiempo) o al medio ambiente (la distribución de los datos cambia significativamente debido a cambios o perturbaciones en el entorno).

Es importante que, a la hora de realizar el entrenamiento y predicción con cualquier modelo, no sería adecuado la utilización de datos de diversas fuentes (no procedentes del mismo entorno físico), o la partición de los datos atendiendo a la variable del tiempo.

Desde 2013 la empresa BackBlaze [34] ha publicado los datos recogidos en su centro de datos, además de estadísticas y conocimiento basado en los discos duros. Dado el interés sobre el desarrollo de técnicas novedosas sobre la predicción de fallos en discos duros, y la calidad de los datos publicados por esta compañía, existe una gran variedad de publicaciones donde son utilizados.

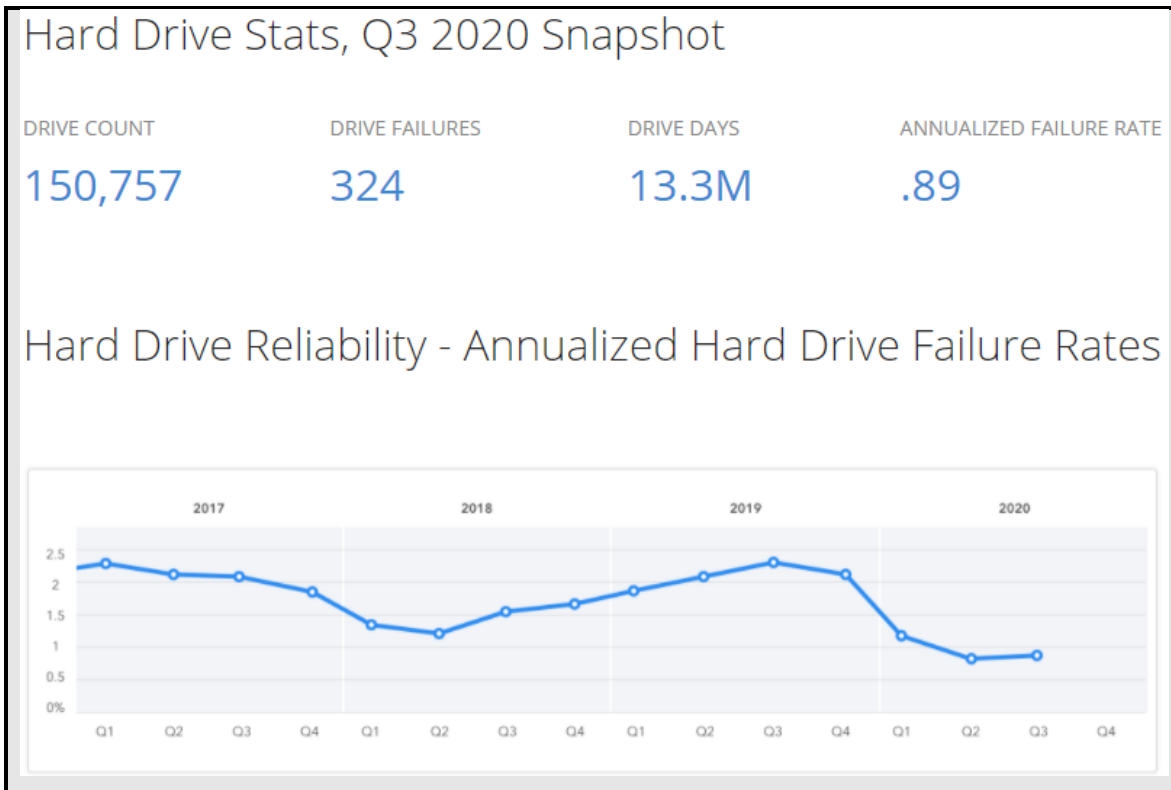


Figura 4-4.- Resumen de estadísticas de los discos monitorizados por el centro de BackBlaze para el tercer cuatrimestre del año 2020 (arriba), y la gráfica de los 4 últimos años (abajo). (Ref.: [34])

Se utilizarán los datos correspondientes a los años 2013, 2014 y 2015 del modelo "ST4000DM000" de Segate, ya que es la muestra más utilizada en la literatura. El conjunto de datos consta de aproximadamente 10,5 millones de registros, correspondientes a 29965 discos duros. El número de fallos registrados, es decir, el número de discos que han sido reemplazados dentro de la ventana temporal es de 872, esto supone un ratio de fallo de 1:12126 frente a los registros de funcionamiento normal. En total se monitorizan 49 atributos.

Se realizará un estudio sobre la metodología actual para resolver la detección de anomalías en discos duros utilizando los valores SMART. Por otro lado, se propondrá una metodología alternativa utilizando herramientas de AutoML. Para realizar una comparativa de resultados objetiva, se utilizarán exclusivamente fuentes donde se hayan empleado los datos procedentes de BackBlaze.



## 4.2.- Caso 2: Detección de anomalías en series temporales

[35] La cantidad de agua que se pierde en el mundo en los sistemas de distribución (WDS) varía entre el 15% y el 50% del agua. La pérdida de agua causa costos ambientales y socioeconómicos y podría tener graves repercusiones en las infraestructuras urbanas. La detección y localización de fugas constituye una prioridad en la gestión de sistemas de agua potable (Figura 4-5). Actualmente para la detección de fugas se utilizan técnicas basadas en acústica, monitorización, inyección de gas, termografía, radares, etc.

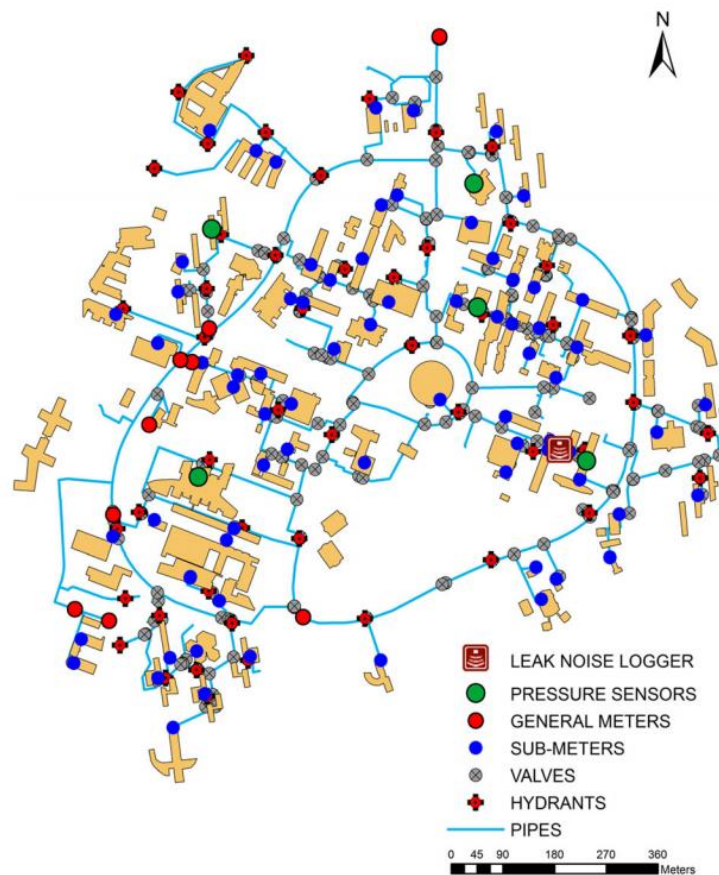


Figura 4-5.- Mapa GIS del sistema de agua del Campus Científico de la Universidad de Lille. (Ref.: [35])

[36] Los recientes avances tecnológicos han traído consigo importantes mejoras en la recopilación de datos, que permiten reunir una gran cantidad de datos a lo largo del tiempo y, por lo tanto, generar series temporales. La extracción de estos datos se ha convertido en una tarea importante para los investigadores y profesionales en los últimos años, incluida la detección de valores atípicos o anomalías que pueden representar errores o acontecimientos de interés. La

detección de anomalías se refiere al problema de encontrar patrones en datos que no se ajustan al comportamiento esperado (Figura 4-6).

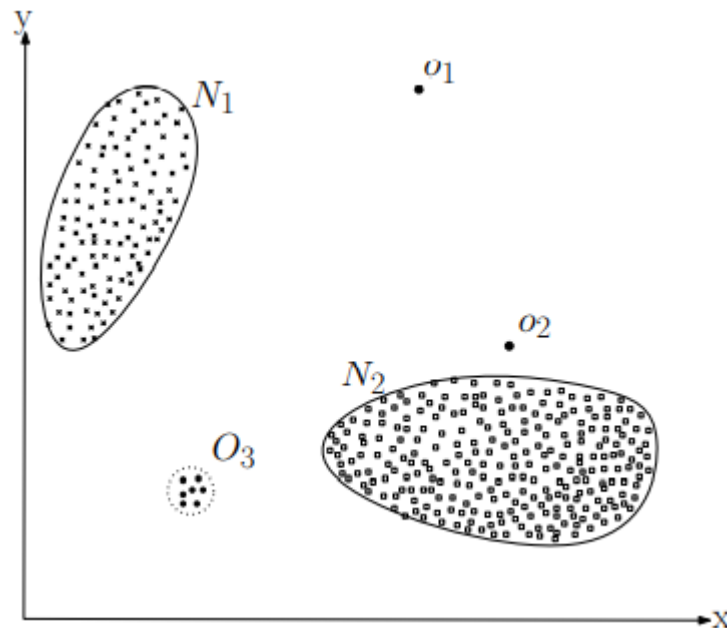


Figura 4-6.- Ejemplo gráfico en dos dimensiones de anomalías (puntos  $o_1$  y  $o_2$ ) (Ref.: [36]).

Con la monitorización de los sistemas de agua, es posible desarrollar algoritmos basados en datos que automaticen la tarea de detección de un comportamiento anómalo. En este proyecto se plantea como caso de uso la detección de anomalías en el sistema de aguas de un pueblo en País Vasco, Azkoitia (Figura 4-7). Para ello se dispone de los datos recopilados por sus sistemas en los años 2017, 2018 y 2019. El sistema de Azkoitia se compone de 5 zonas generales (zona 270, 339, 340, 341 y 343) y 13 subzonas, estas últimas dependen de las anteriores y por tanto no serán estudiadas.

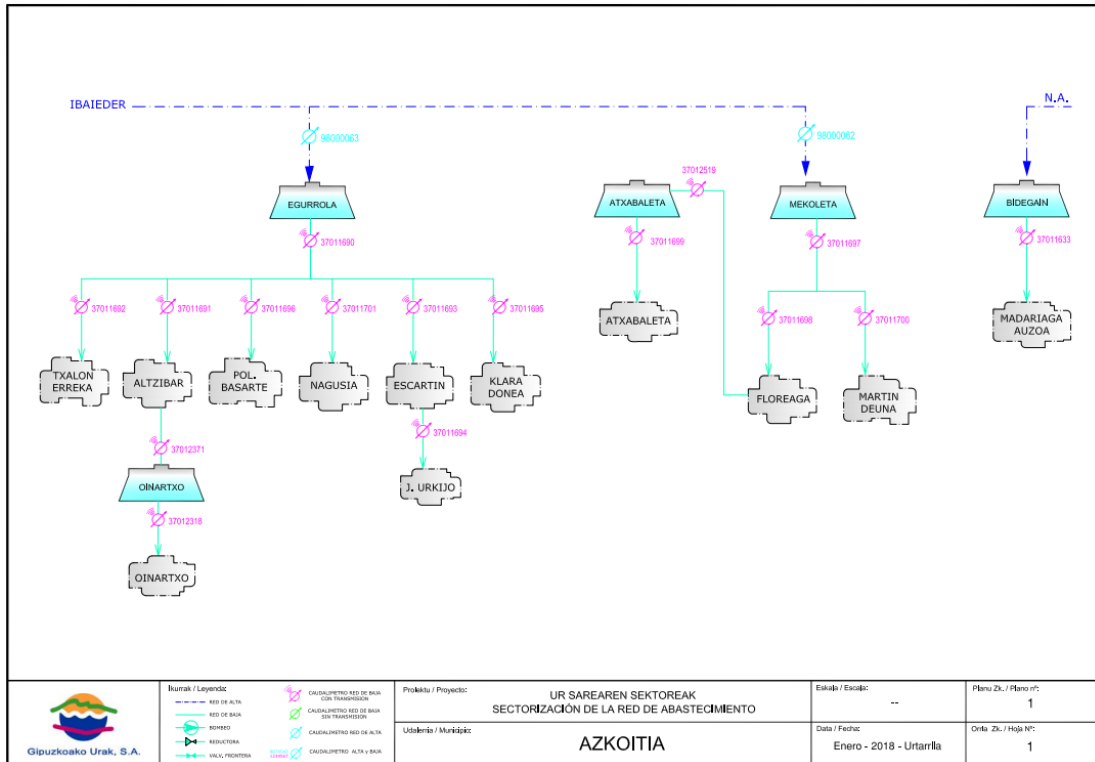


Figura 4-7.- Esquema de la red de aguas de Azkoitia proporcionado por la empresa Gipuzkoako Urak, S.A.

Los datos disponibles constan de: (i) el caudal medido por el sistema cada 5 minutos durante las 24 horas del día (Figura 4-8), (ii) la zona a la que corresponden las medidas, (iii) y una estampa del instante de tiempo en el que se realizó cada registro.

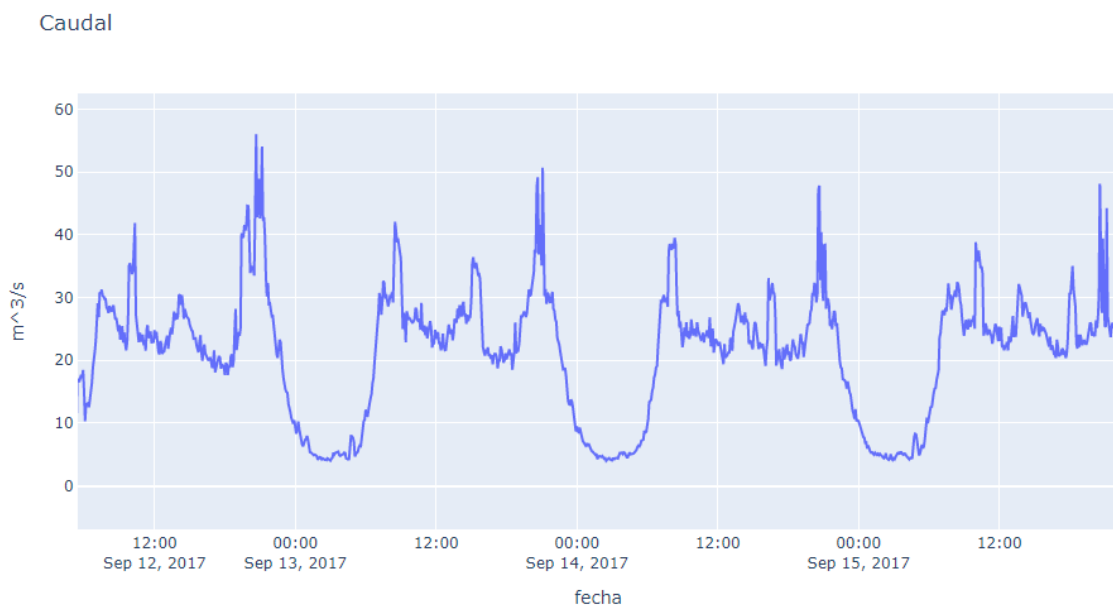


Figura 4-8.- Ventana temporal de lectura de caudal por el sistema.

Actualmente para realizar la detección de fugas se utiliza un umbral, calculado con el mínimo valor de caudal registrado en la noche anterior. Además, se realizan revisiones periódicas del sistema de tuberías. No se dispone de un registro de las fugas resueltas/detectadas con anterioridad.

Como objetivo de este caso de uso, se presentarán diversas metodologías del estado del arte sobre la búsqueda y detección de anomalías en series temporales. Por otro lado, se propondrá una posible metodología utilizando sistemas de AutoML.

### **4.3.- Caso 3: Clasificación de imágenes y localización de objetos para la detección de anomalías**

La gestión de la calidad es un aspecto imprescindible en el proceso de manufactura. Para satisfacer la demanda las manufacturas deben incrementar su tasa de producción manteniendo unos estándares de calidad. [37] Actualmente se atribuye a los sistemas de gestión de la calidad como uno de los avances tecnológicos más importantes para el rendimiento empresarial.

La visión por computador proporciona una alternativa para una técnica automatizada, no destructiva y rentable para cumplir estos requisitos, [38], [39], [40]. Este enfoque de inspección basado en el análisis y el procesamiento de imágenes (Figura 4-9) ha encontrado una variedad de aplicaciones diferentes en la industria. Estas técnicas prometen una gran rentabilidad económica y un buen nivel de precisión sin la ayuda del humano. Es obvio pensar que en los próximos años la mayoría de las tareas de inspección de calidad serán realizadas por sistemas de visión por computador, o al menos consistirá en sistemas híbridos de operadores y sistemas robóticos.

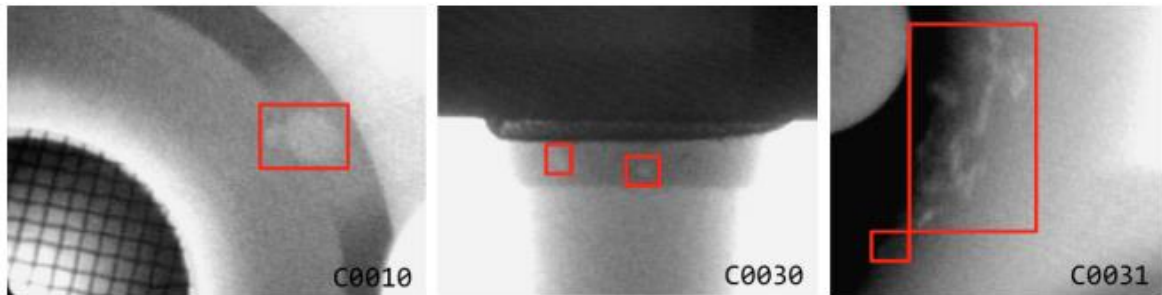


Figura 4-9.- Distintos defectos en ruedas de aluminio. (Ref.: [39])

Aunque los sistemas de inspección automatizados están sustituyendo con éxito a los manuales en muchos sectores, sigue siendo necesario mejorar la precisión del proceso para reducir la tasa de falsos positivos (productos clasificados como buenos cuando son defectuosos) y falsos negativos (productos clasificados como defectuosos cuando son buenos), así como para reducir el tiempo de procesamiento. En la actualidad existen diversas soluciones para este cometido, pero debido al auge de la inteligencia artificial y la viabilidad/éxito de su aplicación en diversos campos, cada vez son más populares técnicas basadas en modelado. Con la construcción de un modelo de inteligencia artificial este será capaz de identificar en una imagen si el producto contiene algún defecto. Para ello, es necesario disponer de un conjunto de imágenes (Figura 4-10) representativas de los defectos que se intentan clasificar.

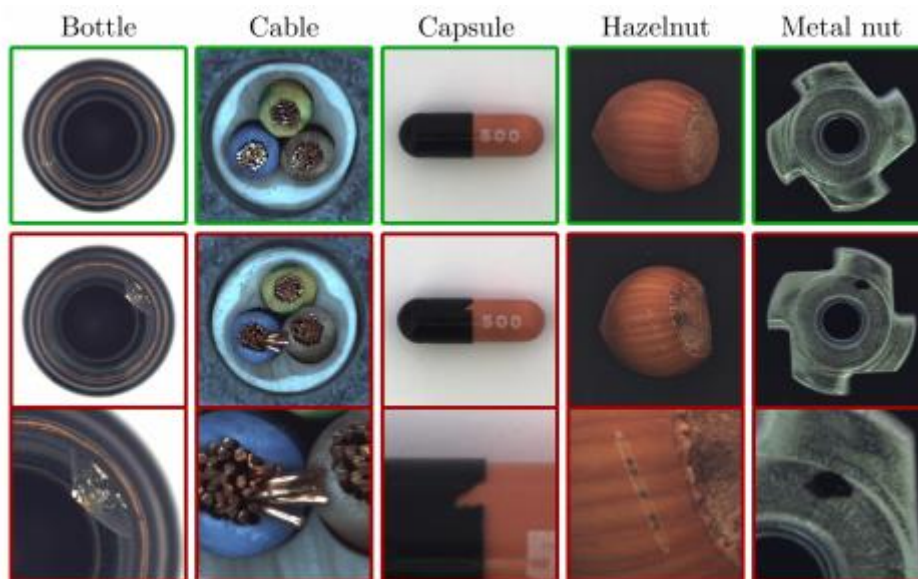


Figura 4-10.- Conjunto de imágenes de productos sin defectos (encuadrado verde) y con defectos (encuadrado rojo).

Para este caso de uso se plantean tres escenarios: (1) la clasificación de imágenes que contienen o no defectos (clasificación binaria), (2) la clasificación de imágenes que contienen varios defectos, para este problema existen varios tipos de defectos y una imagen puede tener uno, varios o ningún defecto, (3) y por último la localización del defecto dentro de la imagen, para este problema se indicará dentro de la imagen donde se encuentra el defecto, si contiene alguno.

Se utilizarán los datos publicados por distintas fuentes: [40], [41], [42].

- Clasificación binaria: Tres conjuntos de imágenes para el problema de clasificación binaria, "metal\_nut", "transistor" y "cable" (Fuente: [42]). Las imágenes están etiquetadas, es decir, se indica si estas contienen defectos o no (Figura 4-11 y Tabla 4.1).

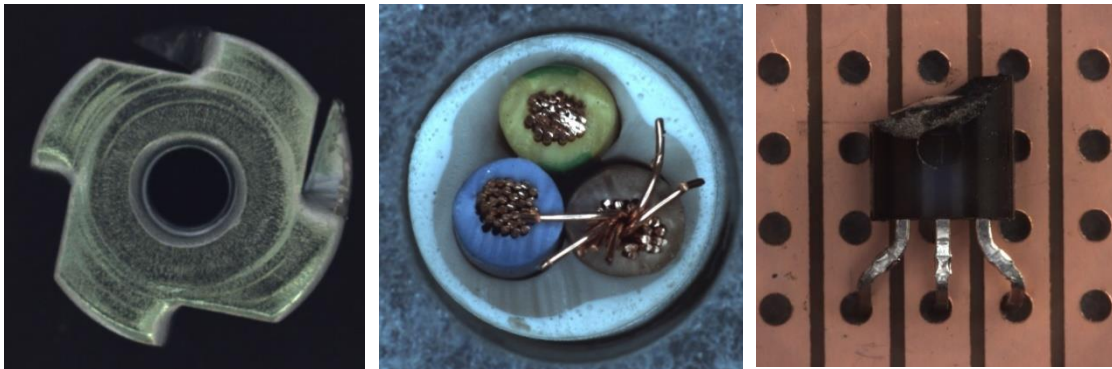


Figura 4-11.- Muestra de los conjuntos de imágenes de clasificación binaria (normal/anomalía) (Ref.: [42])

Datasef	Nº Img. Normal	Nº Img. Anomalia	Total
metal_nut	242	93	335
transistor	273	40	313
cable	282	92	374

Tabla 4.1.- Tamaño de los conjuntos de imágenes para clasificación binaria.

- Clasificación multi-etiqueta: Para este problema se utilizará un dataset, "concrete", que contiene imágenes de paredes de hormigón con 5 defectos distintos (Fuente: [40]): "CorrosionStain", "Crack", "Efflorescence", "ExposedBars" y "Spallation", además de imágenes que no contienen ningún defecto, "Background" (Figura 4-12). Hay 10789 etiquetas distribuidas en 7677 imágenes, es decir, una imagen puede pertenecer a varias clases (Figura 4-13).



Figura 4-12.- Muestras del conjunto de imágenes de clasificación multi-etiqueta. (De izquierda a derecha: (1) Grietas, (2) corrosión y (3) eflorescencia. (Ref.: [40])

Clase	CorrosionStain	Crack	Efflorescece	ExposedBars	Spallation	Background
Nº etiquetas	1559	2507	833	1507	1898	2485

Figura 4-13.- Número de etiquetas de cada clase presentes en las 7677 imágenes etiquetadas.

- Localización de defectos: Se utilizará el dataset "Castings" del repositorio GDxray (Fuente: [41]). Este conjunto de imágenes contiene 2727 imágenes y 851 defectos. Cada imagen puede tener, ningún defecto, uno, o varios defectos. En el dataset se incluyen las coordenadas XY de las esquinas de la región de interés, es decir, el área rectangular que encierra el defecto.

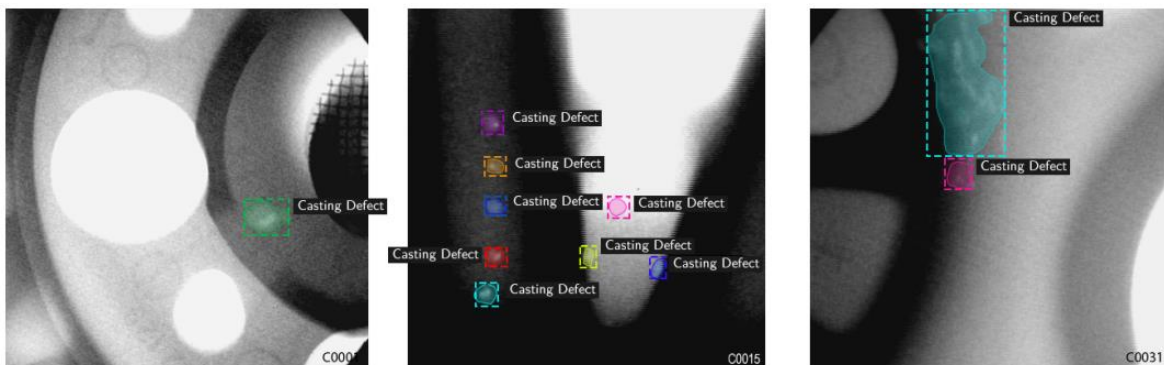


Figura 4-14.- Detección y segmentación de defectos en imágenes. Una imagen puede contener más de un defecto. (Ref.: [39])

Se presentará para cada conjunto de datos una metodología y resultados obtenidos según las publicaciones citadas. También se presentará para cada caso una posible metodología utilizando sistemas de AutoML.

# 5.- Metodología.

## 5.1.- Tradicional (Ad-Hoc)

### 5.1.1.- Caso 1

El objetivo para este problema es clasificar de entre los valores SMART registrados, cuando el disco duro está sano o se encuentra en un estado de fallo. Como se mencionó en la descripción del caso de uso, los datos SMART se caracterizan por dos aspectos: (i) proporción desbalanceada de las clases, (ii) series temporales en los datos. Si se entrenara un modelo directamente, debido a estos dos puntos, este minimizaría el error en la predicción clasificando todos, o un alto porcentaje de los discos, como sanos. Es por ello necesario realizar varios pasos de preprocesado e ingeniería en los datos previo al entrenamiento del modelo.

Si se presenta este escenario (datos desbalanceados) y no se dispone de conocimiento sobre el dominio con el que se va a trabajar, existen librerías y técnicas de fácil uso para el tratamiento de datos desbalanceados. Las técnicas más populares y utilizadas son:

- **Ponderación de clases con pesos:** se indica por medio de un valor número el peso que se le asigna a cada clase. Utilizando un valor mayor para la clase minoritaria, se le estaría dando una mayor importancia a esta, mejorando así el rendimiento del modelo.
- **Submuestreo:** algoritmos basados en la vecindad para obtener una submuestra representativa, disminuyendo el número de muestras de la clase mayoritaria.
- **Sobremuestreo:** se replican muestras de la clase minoritaria para igualar estas al número de muestras de la clase mayoritaria.
- **“Synthetic Minority Oversampling Technique” (SMOTE):** similar al sobremuestreo, se utilizan técnicas de vecindad para añadir muestras sintéticas a la clase minoritaria.



No obstante, en ocasiones puede no ser suficiente el uso de estas técnicas. Para determinados problemas es requerida una ingeniería de datos más dependiente a este, en [33] realizan una serie de cuatro pasos basándose en su conocimiento sobre el dominio. Con esta etapa previa se busca seleccionar los atributos más representativos y balancear la proporción entre discos duros sanos y discos duros fallidos. Los pasos son los siguientes: (1) selección de los atributos SMART relevantes, (2) compactación de las series temporales, (3) balanceo de las clases sano y fallo mediante submuestreo, y (4) selección del modelo para la predicción.

- (1) **Detección de los puntos de cambio:** cuando un atributo SMART es informativo del replazo de un disco duro se espera un cambio significativo en sus valores en algún punto en el tiempo antes de que este sea remplazado (Figura 5-1). El método para detectar los puntos de cambio para cada característica consiste en: evaluar cada punto comparándolo con su adyacente antecesor, si el máximo entre los dos valores excede un threshold previamente calculado se estima como un punto de cambio.

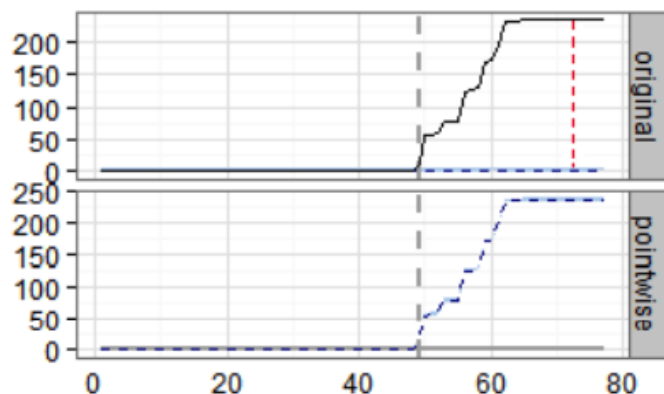


Figura 5-1.- Gráfica de un punto de cambio de un atributo SMART.

- (2) **Selección de características:** se omitirán los atributos con más del 70% de registros con valores vacíos o nulos, se obtienen 35 de 49 características. Además, se cuentan las frecuencias de correlación (porcentaje de unidades para las que se observa una correlación con el disco) para las características SMART con puntos de cambio. Se seleccionan las características con una correlación mayor e igual a 10%. Se toman 24 atributos de los 35 totales (Figura 5-2).

Correlation frequencies			
	percent		
smart_1_normalized	30.62%	smart_188_raw	0.85%
smart_1_raw	24.57%	smart_189_normalized	2.36%
smart_3_normalized	14.65%	smart_189_raw	2.46%
smart_4_raw	33.08%	smart_190_normalized	54.06%
smart_5_normalized	6.14%	smart_190_raw	53.97%
smart_5_raw	20.51%	smart_192_raw	5.01%
smart_7_normalized	41.78%	smart_193_normalized	30.06%
smart_7_raw	61.91%	smart_193_raw	58.32%
smart_9_normalized	54.25%	smart_194_normalized	53.97%
smart_9_raw	58.79%	smart_194_raw	53.97%
smart_12_raw	33.08%	smart_197_normalized	9.74%
smart_183_normalized	12.76%	smart_197_raw	44.71%
smart_183_raw	12.76%	smart_198_normalized	9.74%
smart_184_normalized	2.08%	smart_198_raw	44.71%
smart_184_raw	2.08%	smart_199_raw	0.38%
smart_187_normalized	33.93%	smart_240_raw	58.32%
smart_187_raw	33.93%	smart_241_raw	48.02%
		smart_242_raw	65.12%

Figura 5-2.- Correlación de cada atributo SMART con la columna objetivo (fallo).

(3) **Compactar las series temporales:** Se utiliza un método de suavizado exponencial. Para el tamaño de ventana (parámetro del método de suavizado) se utiliza la mediana de los valores desde que el punto de cambio es detectado hasta que se reemplaza el disco (Figura 5-3). Se aplica 4 veces iterativamente. Compactando todas las series temporales de todos los discos disponibles en el dataset, se obtiene un total de 29965 registros para los datos de los años 2013, 2014 y 2015.

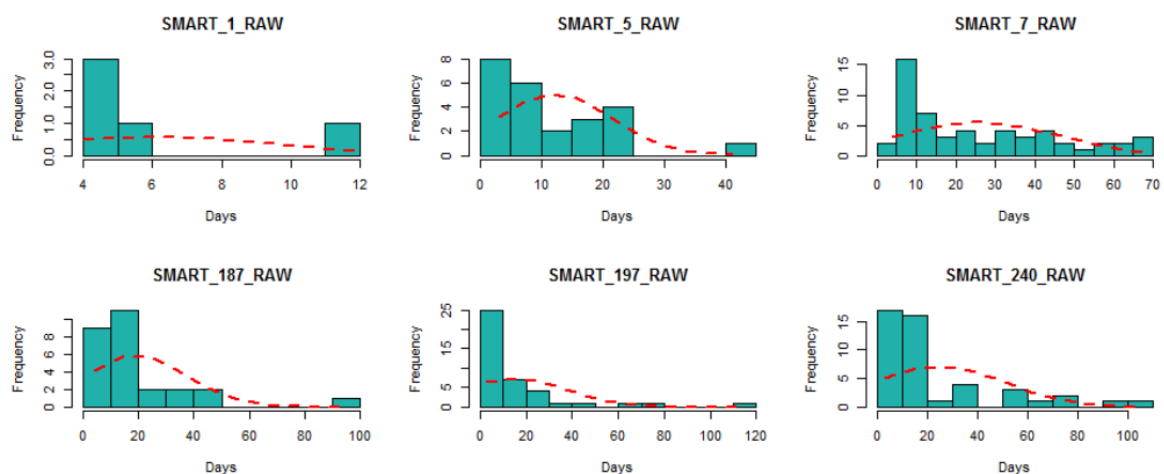


Figura 5-3.- Días antes del fallo después de la detección del primer punto de cambio.

- (4) **Submuestreo de la clase mayoritaria:** Se utiliza un método de clustering (K-means) para obtener un conjunto representativo de la clase mayoritaria (muestras sanas). Se aplica el método para los valores de k (número de vecinos) de 100 y 50. Para cada cluster se escoge los 10 puntos más próximos al centroide de este. Se obtiene una muestra de 1000 discos sanos.
- (5) **Entrenamiento del modelo de clasificación:** Se utiliza un método de validación cruzada (k-fold) con un valor de K de 10. Se utilizan un total de 6 métodos de clasificación: Regularized Greedy Forest (RGF), Gradient Boosted Decision Trees (GBDT), Random Forests (RF), Support Vector Machines (SVM), Logistic Regression (LR) y decision trees (DT).

Para la elección de los hiperparámetros de cada modelo se ha realizado una búsqueda "Grid" con 5 particiones (parámetro cv). Se ha utilizado 'F1' como métrica de evaluación. Se ha conservado el espacio de búsqueda definido en [43]. Finalmente, los hiperparámetros con los que se han obtenido mejores resultados se muestran en la siguiente tabla (Tabla 5.1). Para los parámetros no especificados se han conservarán los valores por defecto según la librería que implementa cada algoritmo.

ALGORITMO	HIPERPARÁMETROS
<b>RFC</b>	<pre>n_estimators: [13] max_features: ['auto'] criterion: ['entropy'] max_depth: [18] min_samples_split: [2] min_samples_leaf: [1] min_weight_fraction_leaf: [0] max_leaf_nodes: [None] bootstrap: [False] random_state: [0] class_weight: ['balanced']</pre>
<b>SVM</b>	<pre>C: [1.05] kernel: ['linear'] gamma: [0.05] class_weight: [{0:0.005,1:0.995}] max_iter: [-1] random_state: [0]</pre>
<b>GBDT</b>	<pre>n_estimators: [169] min_samples_split: [2]</pre>

	<pre>min_samples_leaf: [2] max_depth: [8] max_features: ['sqrt'] subsample: [0.93] random_state: [0]</pre>
<b>RGFC</b>	<pre>max_leaf: [10000] l2: [1.0] s12: [0.005] test_interval: [100] verbose: [True]</pre>
<b>DT</b>	-
<b>LR</b>	<pre>C: np.linspace(0.1, 0.5, 10) class_weight: ['balanced'] solver: ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']</pre>

Tabla 5.1.- Tabla de hiperparámetros de los algoritmos utilizados en la clasificación de anomalías en discos duros.

Por último, se han generado 100 particionado aleatorias de los datos con la proporción 80% de los datos para el entrenamiento, y 20% para test.

Ya que en la publicación [33] no se especifica el umbral de clasificación utilizado, se ha realizado un análisis del número de discos clasificados correctamente modificando el umbral de confianza. Se han establecido tres valores de umbral, 0.5, 0.8 y 0.9 y se ha calculado la precisión y recuperación en un horizonte de 30 días. Para realizar la evaluación del modelo una vez entrenado, se han utilizado los datos obtenidos en el paso previo a la compactación de las series temporales.

### 5.1.2.- Caso 2

Actualmente en el sistema de Azkoitia implementa un sistema para la detección de las fugas basado en el mínimo nocturno. Este método está detallado en [35] para otro escenario. Este consiste en utilizar un umbral, el mínimo de los valores medidos en la noche anterior. El umbral se determina en la franja horaria nocturna (concretamente en Azkoitia de 2:00 a 4:00 am) cuando la demanda es menor y, por tanto, menos fluctuaciones en los valores de caudal. Cuando el caudal supera el umbral determinado se genera una alarma de posible fuga, posteriormente el caso será revisado por un operario.

Aunque el método de umbralizado sea sencillo de implementar es sensible a grandes fluctuaciones y ruido en los datos de entrada. Estos problemas ocasionan una gran frecuencia de falsos positivos (alarmas que finalmente no son fugas). Como solución, en la literatura se proponen métodos más robustos basados en modelado.

En [44] utilizan máquinas de soporte vectorial (SVM) para la detección de anomalías en series temporales. Para el entrenamiento del modelo transforman los datos de tal forma que la variable objetivo se base en los valores temporalmente actuales y pasados (en inglés "lagged data"). El número de saltos temporales se denomina tamaño de ventana (m). Para determinar el valor de m recomiendan un tamaño del 10% al 30% de la periodicidad de interés en la serie temporal. Otro método muy común es calcular el mínimo de la función de autocorrelación, la autocorrelación es una medida de la dependencia lineal entre la variable objetivo y su correspondiente versión desplazada en el tiempo (en inglés time-shifted).

	t	t-1	t-2	t-3
1	0.753	-	-	-
2	0.586	0.753	-	-
3	0.574	0.586	0.753	-
4	0.123	0.574	0.586	0.753
5	0.699	0.123	0.574	0.586
6	0.564	0.699	0.123	0.574

Tabla 5.2.- Ejemplo de tabular de datos retardados (lagged data) con tamaño de ventana 3 (m).

Realizando el análisis de autocorrelación sobre los datos de Azkoitia se obtiene la curva mostrada en la figura Figura 5-8, el tamaño estimado de ventana para este caso es de 25.

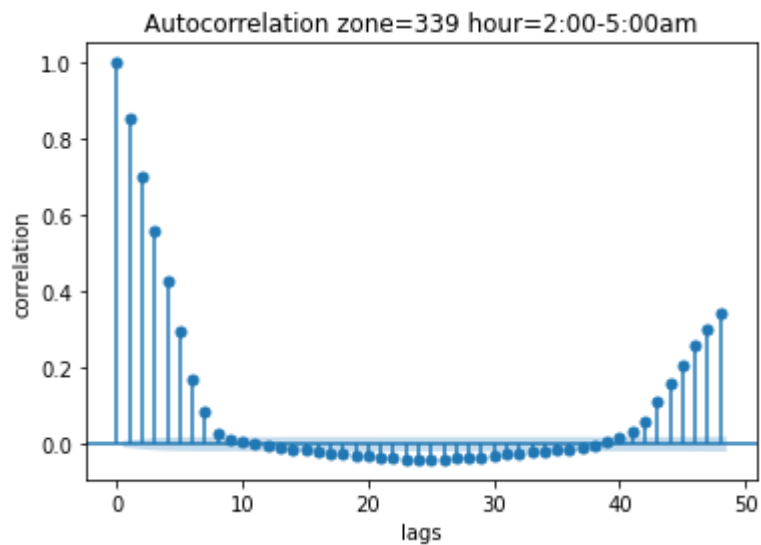


Tabla 5.3.- Autocorrelación de la variable dependiente (caudal).

Como modelo se ha utilizado un algoritmo de regresión, "GradientBoostingRegressor" (GBR). Se ha realizado una búsqueda "Grid" con 10 particiones para la elección de los hiperparámetros.

ALGORITMO	HIPERPARÁMETROS
GBR	<pre>n_estimators: [500] loss: ['huber'] learning_rate: [0.001] subsamples: [0.5] max_depth: [12] min_samples_leaf: [5]</pre>

La proporción usada en la partición de los datos de entrenamiento y test es de 75% y 25% respectivamente. Una vez entrenado el modelo y obtenidos los resultados, se ha calculado el threshold de la siguiente forma (Figura 5-4): (i) se calcula el error absoluto medio para cada día de los datos observados y los predichos en los datos de entrenamiento, (ii) el threshold se corresponde al percentil 95 de los valores de error obtenidos, (iii) si el error absoluto medio del día, en los datos de test, supera al valor del umbral, este se estimará como anomalía.

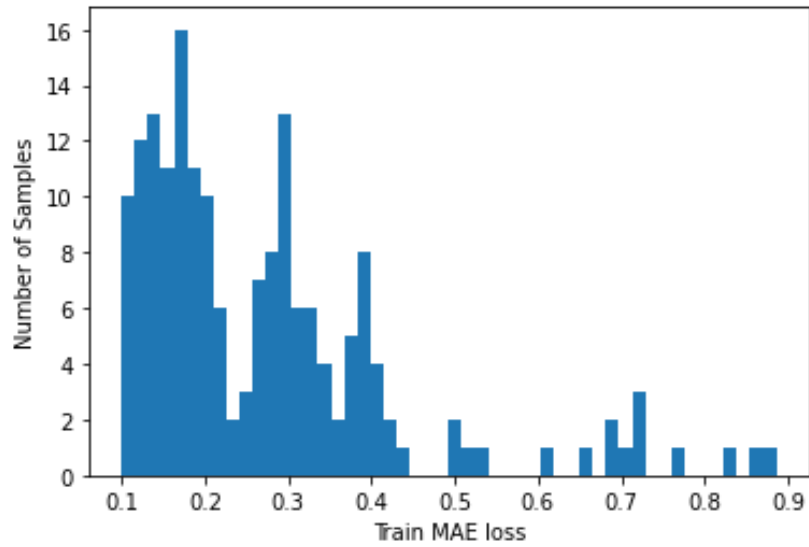


Figura 5-4.- Histograma de error entre los valores observados y la predicción. Para esta gráfica el error medio es de 0.2267 y el threshold (percentil 95) es de 0.42.

En la Figura 5-5 se puede visualizar el error umbralizado y la detección de una posible fuga:



Figura 5-5 En la figura superior el error obtenido tras la predicción y el threshold calculado (línea roja horizontal). En la figura inferior se ha destacado los puntos (puntos verdes) correspondientes a un día completo cuyo error ha superado el threshold.

La desventaja de utilizar modelos lineales como SVM, o modelos convencionales de regresión, es que el horizonte de predicción es de un valor o un instante de tiempo. Existen algoritmos capaces de predecir más de un valor, por ejemplo, predecir los valores de caudal correspondientes a un día entero. En [45] proponen una solución para el tratamiento de series temporales basado en redes neuronales LSTM (del inglés, Long Short-Term Memory). Debido a la no linealidad de los datos,



LSTM está tomando gran popularidad por su rendimiento en este tipo de escenarios.

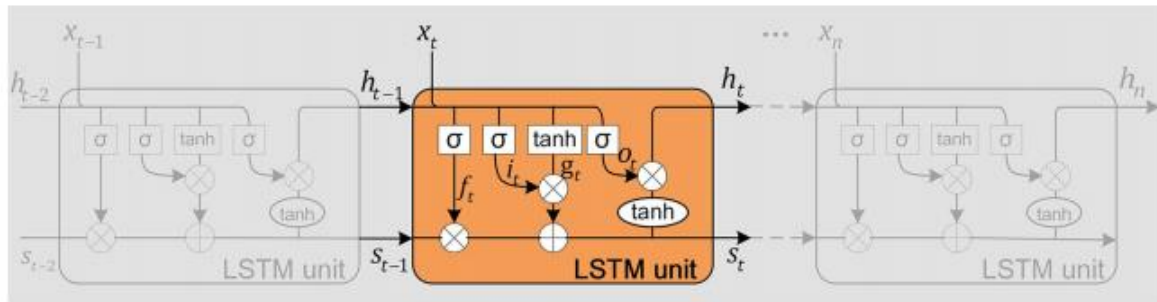


Figura 5-6.- Esquema gráfico de la composición de una unidad LSTM.

La metodología es similar a la presentada con los modelos convencionales: se entrena una red neuronal, se predicen los valores y se calcula el residuo entre los valores observados y los predichos, si el error supera un umbral se determina como anomalía.

La arquitectura propuesta se muestra en la Figura 5-7. Como función de activación se usará ReLU en las dos primeras capas, y una sigmoide en la salida. El dropout y learning\_rate se establece en 0.2 y 0.001 respectivamente. Como métrica de error en el entrenamiento se utiliza un error cuadrático medio.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 25, 1)]	0
lstm (LSTM)	(None, 128)	66560
batch_normalization (BatchNo	(None, 128)	512
activation (Activation)	(None, 128)	0
reshape (Reshape)	(None, 128, 1)	0
lstm_1 (LSTM)	(None, 64)	16896
batch_normalization_1 (Batch	(None, 64)	256
activation_1 (Activation)	(None, 64)	0
dense (Dense)	(None, 60)	3900
activation_2 (Activation)	(None, 60)	0
Total params: 88,124		
Trainable params: 87,740		
Non-trainable params: 384		

Figura 5-7.- Arquitectura propuesta para la solución utilizando LSTM.

### 5.1.3.- Caso 3

Como se ha detallado anteriormente el objetivo de este caso de uso es el de clasificar y/o detectar anomalías en imágenes. Se presentarán varias metodologías para tres escenarios, clasificación binaria, clasificación multi-clase y detección de objetos.

- Clasificación binaria:

En [42] proponen diversos algoritmos basados en redes neuronales: (i) Generative Adversal Networks (GAN), Deep Convolutional Autoencoders (CAEs), y CNN pre-entrenadas. La metodología para cada caso se detalla a continuación:

(i) La red generativa adversaria es entrenada únicamente con imágenes libres de defectos. El generador tiene la capacidad de producir imágenes de aspecto realista. Por otro lado, un discriminador se encarga de determinar si la imagen es real o generada (Figura 5-8).

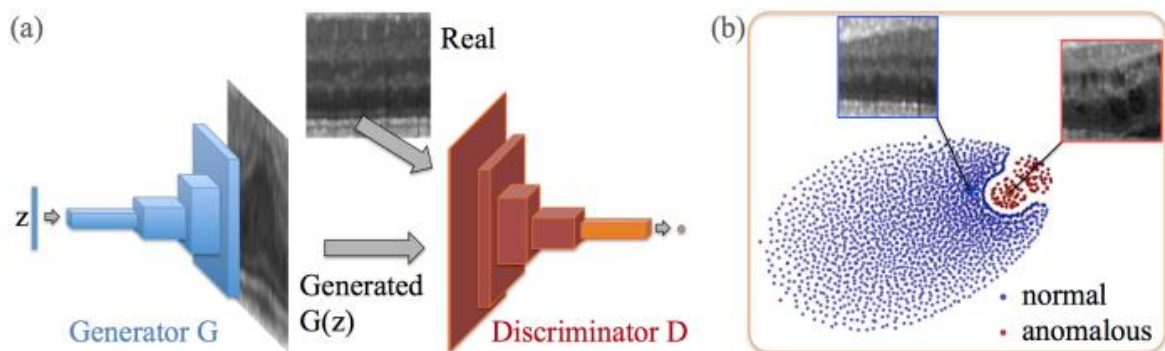


Figura 5-8.- Representación gráfica de una red neuronal generativa. (Ref.: [46])

La dimensión del espacio latente se establece en 64, generando imágenes de 128x128 píxeles. Se realizan 50 épocas (epochs) con un ratio de aprendizaje (learning rate) de 0.0002. Para la prueba se realizan 300 iteraciones de búsqueda en el espacio latente con un learning rate de 0.02. El mapa de anomalías es obtenido con una comparación L2 por píxel de la imagen de entrada y la salida generada por la red.

Para la evaluación se utiliza data augmentation, generando imágenes nuevas aplicando rotaciones y translaciones aleatorias. Adicionalmente se utiliza una

proyección en espejo para las imágenes que lo permiten. Se aumenta en 10000 cada conjunto de entrenamiento.

(ii) Se entrena una red que trata de reconstruir imágenes libres de defectos a través de un espacio latente. La red solo podrá reproducir imágenes vistas en el conjunto usado para el entrenamiento. Si una imagen difiere de su equivalente reconstrucción obtenida con la red esta se clasificará como anomalía (Figura 5-9).

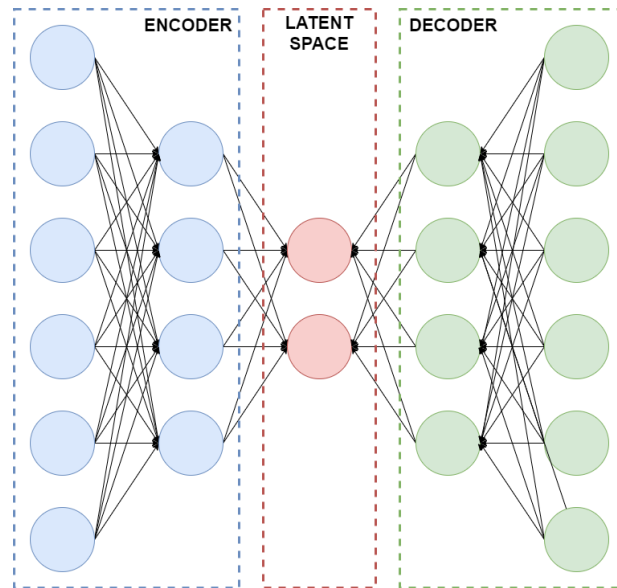


Figura 5-9.- Esquema gráfico simplificado de una arquitectura de un auto-encoder.

La arquitectura de la red se detalla en la siguiente tabla (Tabla 5.4):

Layer	Output Size	Kernel	Parameters	Stride	Padding
Input	128x128x1				
Conv1	64x64x32	4x4		2	1
Conv2	32x32x32	4x4		2	1
Conv3	32x32x32	3x3		1	1
Conv4	16x16x64	4x4		2	1
Conv5	16x16x64	3x3		1	1
Conv6	8x8x128	4x4		2	1
Conv7	8x8x64	3x3		1	1
Conv8	8x8x32	3x3		1	1
Conv9	1x1xd	8x8		1	0

Tabla 5.4.- Arquitectura de la red autoencoder. Se incluye una unidad lineal rectificadora (ReLU) con pendiente 0.2 después de cada capa de activación. (Ref.: [47])

(iii) Por último se propone una red neuronal convolucional de arquitectura ResNet-18. Adicionalmente la red neuronal ha sido previamente entrenada en ImageNet. La arquitectura de la red de ImageNet está detallada en [48] (Figura 5-10).

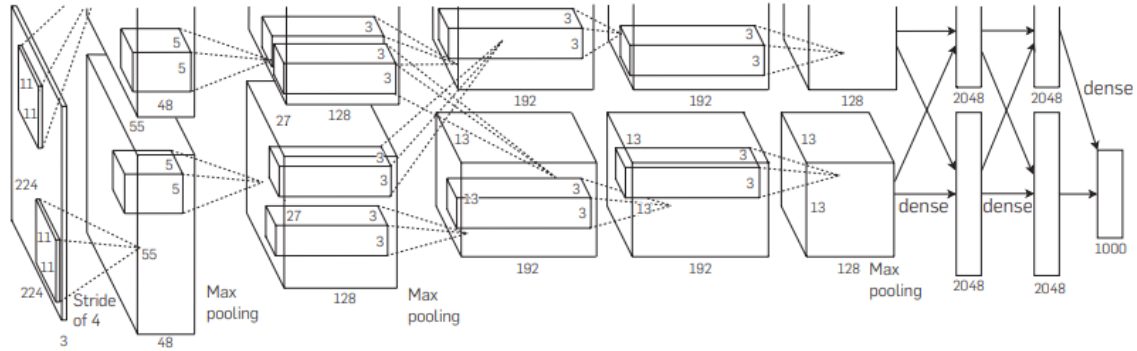


Figura 5-10.- Esquema gráfico de la arquitectura de una CNN. (Ref.: [48])

- Clasificación multi-clase:

En [40] proponen diversos algoritmos para la resolución de un problema de clasificación multi-clase. En este artículo se utiliza el meta-learning para el entrenamiento de redes neuronales convolucionales (CNN). Además se utilizan otras redes del estado del arte: Alexnet, T-CNN, VGG-A, WRN, y DenseNet.

Respecto a los parámetros del meta-learning se utiliza una recompensa (reward) MetaQNN y ENAS:

- **MetaQNN:** Se utiliza un planificador w-greedy. Respecto a la arquitectura, después de las capas de convolución se utiliza un 'pooling' adaptativo con una separación piramidal (SPP) de escala 3, 4 y 5. El tamaño de las capas es de 32, 64 y 128. Todas las capas son seguidas de una capa de normalización por lotes y una capa de activación ReLU.
- **ENAS:** A diferencia de MetaQNN, la arquitectura en ENAS es flexible respecto al número de capas. El número de capas está predeterminado por el número de nodos del grafo de dirección acíclico (DAG). Las opciones disponibles son, convolución de tamaño de filtro 3 y 5, convoluciones separables en profundidad y 'max pooling' y 'average-pooling' con un kernel de 3x3. Cada capa está seguida de una normalización en lotes y una activación ReLU.

El número de parámetros y de capas para cada red está definido en la siguiente tabla (Tabla 5.5):

Architecture	Multi-target accuracy [%]		Params [M]	Layers
	best val	bv-test		
Alexnet	63.05	66.98	57.02	8
T-CNN	64.30	67.93	58.60	8
VGG-A	64.93	70.45	128.79	11
VGG-D	64.00	70.61	134.28	16
WRN-28-4	52.51	57.19	5.84	28
Densenet-121	65.56	70.77	11.50	121
ENAS-1	65.47	70.78	3.41	8
ENAS-2	64.53	68.91	2.71	8
ENAS-3	64.38	68.75	1.70	8
MetaQNN-1	66.02	68.56	4.53	6
MetaQNN-2	65.20	67.45	1.22	8
MetaQNN-3	64.93	72.19	2.88	7

Tabla 5.5.- Arquitectura de cada red de convolución, meta-learning y del estado del arte (Ref.: [40]).

- Detección de regiones:

Por último, en [39] se propone un sistema basado en una arquitectura R-CNN para la detección y segmentación de defectos (Figura 5-11). Para la clasificación de imágenes se utilizan dos métodos, una red CNN problema de clasificación, y otra para la segmentación del defecto. Mediante la segmentación es posible utilizar las máscaras generadas para inferir el defecto según los valores de los píxeles.

El sistema está completo (incluyendo la segmentación) por cuatro módulos:

- (i) Un extractor de características que genera una representación de alto nivel de la imagen de entrada.
- (ii) Una CNN que propone regiones de interés.
- (iii) Una CNN que intenta clasificar los objetos en cada región de interés.
- (iv) Segmentar la imagen para generar una máscara binaria para cada región.

También se propone utilizar redes pre-entrenadas en ImageNet. Mediante técnicas de “Transfer Learning”<sup>7</sup> es posible adecuar una red previamente entrenada al dominio tratado en el problema (imágenes de rayos x).

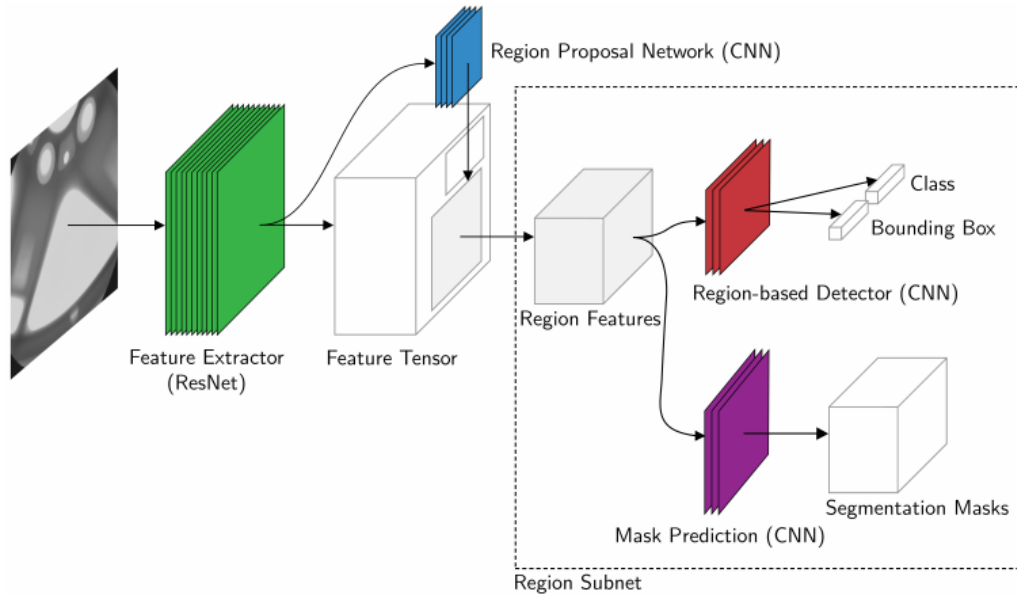


Figura 5-11.- Esquema gráfico de la arquitectura de una R-CNN. (Ref.: [39])

La arquitectura propuesta está basada en una red ResNet-101, se muestra en la Tabla 5.6:

Layer Name	Filter Size	Output Size
conv1	$7 \times 7, 64, \text{stride } 2$	$112 \times 112 \times 64$
conv2_x	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$112 \times 112 \times 64$
conv3_x	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$112 \times 112 \times 64$
conv4_x	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$112 \times 112 \times 64$

Tabla 5.6.- Arquitectura de la R-CNN. (Ref.: [39])

<sup>7</sup> El aprendizaje de transferencia es un problema de investigación en el aprendizaje automático que se centra en almacenar el conocimiento adquirido mientras se resuelve un problema y se aplica a un problema diferente pero relacionado

## **5.2.- Aprendizaje Máquina Automatizado (AutoML)**

### **5.2.1.- Caso 1**

Se han utilizado 3 sistemas de AutoML para la resolución de este caso de uso: AutoGluon, H2O, Google Cloud Tables. Todos los sistemas seleccionados permiten la resolución de problemas de clasificación. Únicamente el sistema de H2O ofrece parámetros para el tratamiento de clases desbalanceadas.

Como metodología utilizando los sistemas de AutoML se proponen dos ensayos: (1) inicialmente utilizar el conjunto de datos en crudo, es decir, sin preprocesado previo (ingeniería y limpieza de datos). Con esta práctica se pretende comprobar como responden los sistemas de AutoML ante conjuntos de gran tamaño y datos desbalanceados. (2) Después se utilizará el conjunto de datos reducido obtenido en la metodología tradicional (apartado 5.1.1.-), con esto observar el desempeño de los sistemas de AutoML sobre un problema de clasificación, y también obtener unos resultados cuantitativos y objetivos para comparar con los obtenidos en dicha metodología.

#### **5.2.1.1.- Google Cloud Platform**

Se utilizará el módulo Tables de GCP, para el entrenamiento de modelos con datos tabulares. El módulo Tables permite importar el conjunto de datos (Figura 5-12) desde "BigQuery", una herramienta de GCP para la realización de consultas en bases de datos en la nube, importarlos desde un archivo en formato "csv" almacenado en "Cloud Storage", o desde el dispositivo desde el que se esté utilizando GCP.



Figura 5-12.- Opciones de importado de GCP (Consola de GCP).

Una vez importado los datos es necesario indicar cual es la columna objetivo (Figura 5-13), es decir, que característica se quiere predecir, y automáticamente la herramienta determinará de que problema se trata, regresión o clasificación. En este caso el objetivo es predecir la columna "failure" que contiene el valor binario 0/1 (0: disco sano, 1: disco fallido).

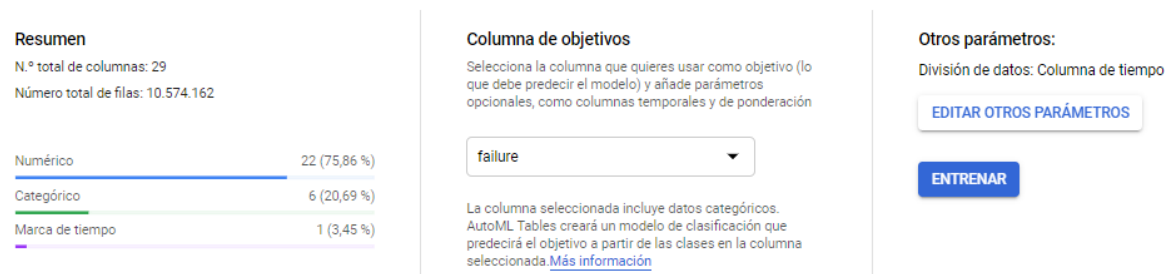


Figura 5-13.- Información del conjunto de datos importado y columna objetivo (Consola de GCP).

La herramienta mostrará los valores únicos para cada característica, el recuento de valores nulos/no válidos y la correlación con el objetivo (Figura 5-14). Es posible escoger si se desean valores nulos o no. Si no se admiten, las filas que contengan valores nulos no serán tenidas en cuenta en el entrenamiento del modelo, en otro caso, no se especifica que procedimiento se realiza. Para el dataset completo el sistema ha detectado 670245 filas con valores nulos.



Nombre de la columna ? ↑	Tipo de datos ?	Nulabilidad ?	Porcentaje restante (recuento) ?	Valores no válidos ?	Valores distintos ?
date	Marca de tiempo	<input type="checkbox"/> Admite valores nulos	0 % (0)	0 % (0)	964
<input checked="" type="checkbox"/> failure Objetivo	Categorico	<input type="checkbox"/> Admite valores nulos	0 % (0)	0 % (0)	2
serial_number	Categorico	<input type="checkbox"/> Admite valores nulos	0 % (0)	0 % (0)	29.965
smart_1_normalized	Numérico	<input checked="" type="checkbox"/> Admite	6,339 % (670.247)	0 % (0)	34

Figura 5-14.- Estadísticas de las características. (Consola de GCP)

Por último, es posible seleccionar como se dividirán los datos en el entrenamiento (Figura 5-15), conjunto de entrenamiento, validación y test: (i) de forma aleatoria con una proporción de 80%, 10%, 10% respectivamente, (ii) utilizando una columna adicional donde se indique a que conjunto pertenece cada registro, (iii) utilizando una característica de tipo temporal.

### Otros parámetros

#### División de datos ?

Automática (predeterminada)

El 80 % de las filas de datos se utiliza para la preparación; el 10 %, para la validación, y el otro 10 %, para las pruebas

Manual

No hay columnas categóricas de entrenamiento ni de pruebas. Añade una de ellas para dividir datos de forma personalizada.

Columna de tiempo ?

Usa esta opción si las filas de datos están ordenadas por marca de tiempo y tu modelo predice valores que se darán en el futuro

[RESTABLECER](#)

date

Figura 5-15.- Opciones de división del conjunto de datos. (Consola de GCP)

Para el primer ensayo (conjunto de datos completo) se utilizará la división de los datos utilizando la columna "date". Esta columna contiene la fecha de cada instante de tiempo en el que se registraron los valores SMART. Para el segundo ensayo (conjunto de datos reducido), ya que se han compactado las series temporales, y por tanto se elimina la temporalidad en los valores, se utilizará una división automática en tres conjuntos (entrenamiento, validación y test).



Como parámetros del modelo solo es posible seleccionar la métrica de evaluación de entre 5 opciones (Figura 5-16): (i) el área por debajo de la curva ("area under

curve", AUC), (ii) pérdida logarítmica, (iii) área por debajo de la curva de precisión-recuperación ("área under curve precisión-recall", AUCPR), (iv) según un valor determinado de precisión y (v) según un valor determinado de recuperación. Para este caso de uso la métrica más apropiada es la de AUCPR ya que maximiza la curva de precisión-recuperación para la clase minoritaria.

#### Opciones avanzadas

##### Objetivo de optimización

Según los resultados que intentes obtener, puedes entrenar tu modelo para que optimice un objetivo u otro. [Más información](#)

- Área por debajo de la curva de la característica operativa del receptor  
Se distingue entre clases
- Pérdida logarítmica  
Se mantienen las probabilidades de predicción lo más precisas posible.
- Área por debajo de la curva de precisión-recuperación  
Se maximiza la curva de precisión-recuperación de la clase menos común
- Precisión  
- Recuperación  

##### Detención temprana

Detiene la preparación del modelo cuando Tables detecta que ya no hay margen de mejora (el importe restante del presupuesto de la preparación se devuelve). Si la detención temprana está desactivada, la preparación continuará hasta que se agote el presupuesto. [Más información](#)

Figura 5-16.- Especificación de la métrica de evaluación para el entrenamiento del modelo en Google Tables. (Consola de GCP)

Para el entrenamiento del modelo es necesario indicar el presupuesto del nodo. El presupuesto es el tiempo máximo de entrenamiento (entre 1 a 72 horas) para el modelo definido. Si el modelo deja de mejorar, el sistema detendrá el entrenamiento, solo facturando las horas útiles de entrenamiento. Según la estimación de la propia herramienta (basándose en el número de filas), se han utilizado 3 horas de nodo.

### 5.2.1.2.- AutoGluon

Para este caso de uso es necesario usar el módulo "Tabular Prediction". Se ha utilizado la función de la librería disponible para un entrenamiento automatizado, `'fit'`. Para la llamada a esta función se ha indicado el conjunto de datos y la columna objetivo `'failure'`. AutoGluon infiere, en base a los valores de cada característica, de que tipo son. Ya que la variable objetivo es una variable binaria de 2 valores AutoGluon determina que se trata de un problema de clasificación binaria.

Es posible seleccionar de entre un conjunto de configuraciones del entorno de ejecución mediante el uso del parámetro `'presets'` (Figura 5-17). Por defecto se establece la opción de `'medium_quality_faster_train'`, recomendada para un rápido prototipado ya que es ~20% más rápida que la versión de calidad mayor. Esta opción selecciona de forma determinista un conjunto de hiperparámetros para cada modelo. El espacio de búsqueda y el tipo/cantidad de algoritmos viene predeterminada para esta opción. Los algoritmos que se entrenan son los siguientes: KNeighborsRegressorDis, KNeighborsRegressorUnif, NeuralNetRegressor, CatboostRegressor, RandomForestRegressorMSE, LightGBMRegressor, ExtraTreesRegressorMSE, y un ensamblado entre los dos mejores modelos para el entrenamiento concreto.

Una vez se disponga de más conocimiento sobre el dominio, o se requiera un menor error del modelo, se puede utilizar la opción de `'best_quality'`. Esta opción afecta considerablemente al uso de recursos, ya que utiliza el ensamblado de modelos y métodos de ajuste y optimización más avanzados. La opción de máxima calidad realizará una búsqueda de parámetros maximizando la métrica de evaluación que se haya seleccionado, esta opción realizará por defecto 3 niveles de ensamblado. Es posible también establecer tiempo límite para el entrenamiento y modificar el número de niveles de ensamblado. En este caso se mantendrán los valores por defecto.

**presets** : *list or str or dict, default = 'medium\_quality\_faster\_train'*

List of preset configurations for various arguments in *fit()*. Can significantly impact predictive accuracy, memory-footprint, and inference latency of trained models, and various other properties of the returned *predictor*. It is recommended to specify presets and avoid specifying most other *fit()* arguments or model hyperparameters prior to becoming familiar with AutoGluon. As an example, to get the most accurate overall predictor (regardless of its efficiency), set *presets='best\_quality'*. To get good quality with minimal disk usage, set *presets=[good\_quality\_faster\_inference\_only\_refit, optimize\_for\_deployment]* Any user-specified arguments in *fit()* will override the values used by presets. If specifying a list of presets, later presets will override earlier presets if they alter the same argument. For precise definitions of the provided presets, see file: *autogluon/tasks/tabular\_prediction/presets\_configs.py*. Users can specify custom presets by passing in a dictionary of argument values as an element to the list.

Available Presets: ['best\_quality', 'best\_quality\_with\_high\_quality\_refit', 'high\_quality\_fast\_inference\_only\_refit', 'good\_quality\_faster\_inference\_only\_refit', 'medium\_quality\_faster\_train', 'optimize\_for\_deployment', 'ignore\_text'] It is recommended to only use one *quality* based preset in a given call to *fit()* as they alter many of the same arguments and are not compatible with each-other.

Figura 5-17.- Atributo de 'presets' de AutoGluon (Ref.: [49]).

Con respecto al preprocesado de datos desbalanceados, AutoGluon no ofrece ninguna herramienta o parámetro. Si dispone de un conjunto de métricas de rendimiento que pueden favorecer al entrenamiento con una proporción de clases no balanceada. Se dispone un total de 18 métricas (Figura 5-18) para clasificación. Entre las 18, 'balanced\_accuracy', 'f1\_weighted'/'precision\_weighted'/'recall\_weighted' y 'roc\_auc' pueden favorecer a la obtención de mejores resultados para este caso. Por otro lado, el sistema ha interpolado los valores nulos utilizando una interpolación cuadrática. Se ha inferido un tipo numérico real sobre los valores SMART y categórico sobre la columna objetivo.

**eval\_metric** : *function or str, default = None*

Metric by which predictions will be ultimately evaluated on test data. AutoGluon tunes factors such as hyperparameters, early-stopping, ensemble-weights, etc. in order to improve this metric on validation data. If *eval\_metric = None*, it is automatically chosen based on *problem\_type*. Defaults to 'accuracy' for binary and multiclass classification and 'root\_mean\_squared\_error' for regression. Otherwise, options for classification:

*['accuracy', 'balanced\_accuracy', 'f1', 'f1\_macro', 'f1\_micro', 'f1\_weighted', 'roc\_auc', 'average\_precision', 'precision', 'precision\_macro', 'precision\_micro', 'precision\_weighted', 'recall', 'recall\_macro', 'recall\_micro', 'recall\_weighted', 'log\_loss', 'pac\_score']*

Figura 5-18.- Métricas de evaluación disponibles en AutoGluon TabularPrediction (Ref.: [49]).

Se han utilizado también otras opciones útiles para este caso de uso, pero que requieren una mayor experiencia de programación y en modelado. AutoGluon permite definir el espacio de búsqueda de hiperparámetros para cada modelo y

la estrategia de búsqueda (Figura 5-19). Con esta opción es posible indicar por medio de un diccionario clave-valor, el espacio de valores para cada hiperparámetro del algoritmo. Según la información proporcionada por AutoGluon, solo está disponible la optimización de hiperparámetros en 3 de los 7 algoritmos 'NN' (neural network), 'GBM' (lightGBM boosted trees), 'CAT' (CatBoost boosted trees). Se han definido varios espacios de búsqueda para los algoritmos 'NN', 'GBM' y 'CAT'. Para la definición del espacio se ha utilizado la documentación de las librerías que implementan los algoritmos 'Scikit-learn' [50], 'CatBoost' [51] y 'LightGBM' [52].

### Search Space

<b>Real</b>	Search space for numeric hyperparameter that takes continuous values.
<b>Int</b>	Search space for numeric hyperparameter that takes integer values.
<b>Bool</b>	Search space for hyperparameter that is either True or False.
<b>Categorical</b>	Nested search space for hyperparameters which are categorical.
<b>List</b>	Nested search space corresponding to an ordered list of hyperparameters.
<b>Dict</b>	Nested search space for dictionary containing multiple hyperparameters.
<b>AutoGluonObject</b>	Searchable objects, created by decorating a custom Python class or function using the <code>autogluon.obj()</code> or <code>autogluon.func()</code> decorators.

Figura 5-19.- Tipos de datos permitidos para la definición de espacios de búsqueda. (Ref.: [49]).

Por último, se han particionado los datos en dos subconjuntos, entrenamiento y prueba, con la proporción 80% y 20%. AutoGluon particionará el conjunto de entrenamiento y validación utilizando diferentes métodos como K-Fold y CrossValPredict, disponibles en la librería 'scikit-learn'. Es posible modificar los parámetros de estos métodos o utilizar otro subconjunto para la validación.

#### 5.2.1.3.- H2O

Inicialmente la API de H2O se importa por medio de una librería convencional de Python. El sistema de H2O se alberga en un servidor Java iniciado de forma local o remota (Figura 5-20). Es posible conectarse a un servidor a través del puerto 54323. Si no es detectado ningún servidor se inicializará uno nuevo. Además de la

programación en Python, la herramienta de H2O ofrece un entorno web a través del puerto utilizado (Figura 5-21).

```

1 import h2o
2 from h2o.automl import H2OAutoML
3 h2o.init()

```

Checking whether there is an H2O instance running at <http://localhost:54321> ..... not found.  
Attempting to start a local H2O server...  
; Java HotSpot(TM) Client VM (build 25.271-b09, mixed mode, sharing)  
Starting server from C:\Users\YO\anaconda3\Lib\site-packages\h2o\backend\bin\h2o.jar  
Ice root: C:\Users\YO\AppData\Local\Temp\tmpktytfa3pc  
JVM stdout: C:\Users\YO\AppData\Local\Temp\tmpktytfa3pc\h2o\_YO\_started\_from\_python.out  
JVM stderr: C:\Users\YO\AppData\Local\Temp\tmpktytfa3pc\h2o\_YO\_started\_from\_python.err  
Server is running at <http://127.0.0.1:54323>  
Connecting to H2O server at <http://127.0.0.1:54323> ... successful.

H2O_cluster_uptime:	01 secs
H2O_cluster_timezone:	Europe/Paris
H2O_data_parsing_timezone:	UTC
H2O_cluster_version:	3.32.0.3
H2O_cluster_version_age:	29 days
H2O_cluster_name:	H2O_from_python_YO_fir1s0
H2O_cluster_total_nodes:	1
H2O_cluster_free_memory:	247.5 Mb
H2O_cluster_total_cores:	0
H2O_cluster_allowed_cores:	0
H2O_cluster_status:	accepting new members, healthy
H2O_connection_uri:	<a href="http://127.0.0.1:54323">http://127.0.0.1:54323</a>
H2O_connection_proxy:	{"http": null, "https": null}
H2O_internal_security:	False
H2O_API_Extensions:	Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
Python_version:	3.8.5 final

Figura 5-20.- Resumen del servidor desplegado en java para el sistema H2O.

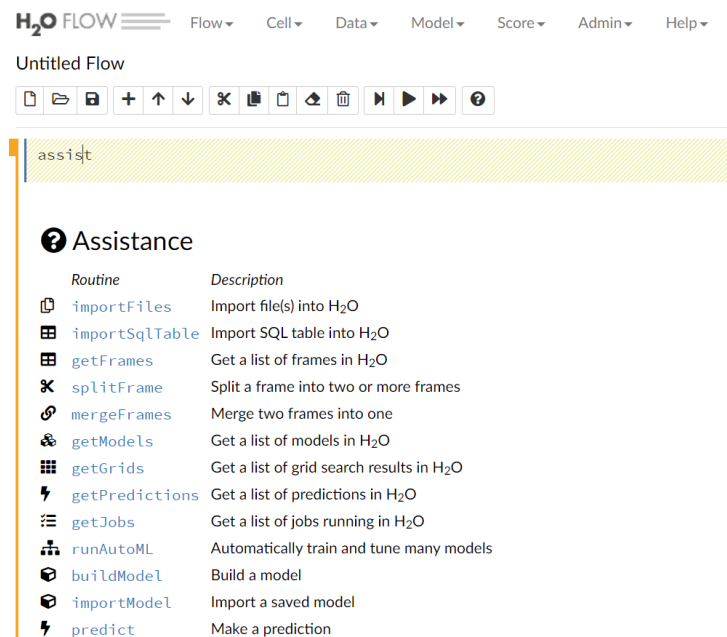


Figura 5-21.- Interfaz web del sistema de AutoML H2O.

Para el tratamiento de los datos de entrada H2O utiliza su propia clase (H2OFrame) similar a la librería 'Pandas'. Como parámetro es necesario indicar la ruta local o web donde se alberguen los datos, en formato CSV. En el proceso de importación de los datos H2O realizará la conversión de tipos según los valores de las columnas.

Para el entrenamiento del modelo se utilizará la función 'H2OAutoML'. Esta función como se explicó en el apartado de selección de características, entrena unos modelos predefinidos además del ensamblado de modelos. Es posible también seleccionar y entrenar modelos de forma independiente. Los parámetros mínimos para esta función son los nombres dentro del H2OFrame de los atributos y el nombre de la variable objetivo, junto con el correspondiente H2OFrame.

Algunos modelos soportan la opción de balanceo de clases a través del parámetro 'balance\_classes'. Cuando este parámetro está activado se realizará un sobremuestreo y submuestreo de los datos de entrada. Otro parámetro asociado es 'max\_after\_balance\_size' donde se indica la proporción del conjunto de datos después del balanceo, por defecto 0.5. Se comprobará el rendimiento del sistema con el balanceo de clases activado y desactivado.

Respecto al particionado de datos, se utilizará un 80% de los datos para el entrenamiento y 20% para test. En el entrenamiento se utiliza validación cruzada con 5-kfold por defecto.

### **5.2.2.- Caso 2**

En este apartado se propone una metodología utilizando sistemas de AutoML para la resolución de detección de anomalías en series temporales. Se han utilizado un total de 2 sistemas: AutoGluon y Amazon Web Services. Todos los sistemas propuestos ofrecen herramientas para ajustar modelos de regresión, pero únicamente el sistema de Amazon ofrece herramientas para la resolución de problemas con series temporales, con la que además es posible la predicción de varios valores (forecasting).

#### **5.2.2.1.- Amazon Web Services**

El módulo Forecast de Amazon Web Services permite entrenar un modelo para la predicción de valores. Este módulo está orientado a la predicción de demanda,

evolución de mercado, estimación de capacidad, etc., pero también ofrece una opción para otro tipo de problemas. Como se muestra en la siguiente figura el sistema ofrece 7 tipos de dominio (Figura 5-22). Para este caso se utilizará la predicción de demanda.

**Forecasting domain Info**

A forecasting domain defines a forecasting use case. You can choose a predefined domain, or you can create your own domain.

Choose a forecasting domain ▲	
<b>Retail</b>	This is a predefined domain for forecasting demand for a retailer.
<b>Custom</b>	Choose this domain if none of the other domains are applicable to your forecasting needs.
<b>Inventory planning</b>	Forecast demand for raw materials and determine how much inventory of a particular item to stock.
<b>Ec2 capacity</b>	Forecast your Amazon Elastic Compute Cloud (Amazon EC2) capacity.
<b>Work force</b>	Use this domain to plan and identify the amount of work force you require.
<b>Web traffic</b>	Forecast web traffic to a web property or a set of web properties.
<b>Metrics</b>	This domain is for forecasting metrics such as revenue, sales, and cashflow.

Figura 5-22.- Opciones de problemas disponibles en Amazon Forecast. (Consola de AWS)

En segundo lugar, es necesario indicar el esquema de los datos de entrada (Figura 5-23). En esta etapa se especifica el intervalo de muestreo de la variable de tiempo, para este caso concreto cada 5 minutos, y la distribución y nombre del resto de atributos. Después se indicará el contenedor de S3 donde están albergados los datos en formato CSV. Es posible utilizar una zona horaria para mejorar la precisión del modelo.



**Dataset name**  
The name can help you distinguish this dataset from other datasets on your Datasets dashboard.

caudal\_5T\_339\_17

The dataset name must have 1 to 63 characters. Valid characters: a-z, A-Z, 0-9, and \_

**Frequency of your data**  
This is the frequency at which entries are registered into your data file.

Your data entries have a time interval of

**Data schema Info**  
Use the data schema section to specify the attribute types for each column in your dataset. You can specify the schema in two ways:

**Schema builder**  
Specify your Attribute Name, Attribute Type, and attribute order in the text boxes provided.

**JSON schema**  
Specify AttributeName and AttributeType in the JSON format.

**Schema Builder Info**  
The attributes below are required for your chosen domain. You may add additional attributes. All attributes displayed must exist in your CSV file and must be ordered in the same order that they appear in your CSV file. To reorder the attributes, simply drag and drop each attribute to the correct position.

Column

1	Attribute Name item_id	Attribute Type string	
2	Attribute Name timestamp	Attribute Type timestamp	Timestamp Format Info yyyy-MM-dd HH:mm:ss
3	Attribute Name demand	Attribute Type float	

Figura 5-23.- Especificación del esquema de datos en Amazon Forecast. (Consola de AWS)

Como parámetros de entrenamiento es necesario indicar el horizonte en la predicción y la frecuencia entre los valores (Figura 5-24), pe.: un horizonte de 288 con una frecuencia de 5 minutos se corresponde a un día, 24 horas. La herramienta permite únicamente predicciones de 500 valores para cualquier frecuencia. Como opciones de frecuencia están disponibles: minutos, horas, días, semanas, meses y años. Para este caso se utilizarán predicciones de minutos y horas.

Respecto a la selección del algoritmo es posible seleccionar dos opciones, totalmente automática, donde el sistema escogerá el algoritmo más apropiado, y una lista de 6 modelos (Figura 5-25): CNN-QR, Prophet, ARIMA, DeepAR+, NPTS y ETS. En este caso se utilizará la opción de AutoML.

**Predictor settings**

**Predictor name**  
The name can help you distinguish this predictor from your other predictors.  
forecast\_5T\_339\_2017\_24h  
The predictor name must have 1 to 63 characters. Valid characters: a-z, A-Z, 0-9, and \_

**Forecast horizon Info**  
This number tells Amazon Forecast how far into the future to predict your data at the specified forecast frequency.  
1

**Forecast frequency**  
This is the frequency at which your forecasts are generated.  
Your forecast frequency is 1 day

Figura 5-24.- Frecuencia y horizonte en la predicción del modelo con Amazon Forecast (Consola de AWS)

**Algorithm selection Info**  
An algorithm is used to train your predictor.

Automatic (AutoML)  
Let Amazon Forecast choose the right algorithm for your dataset.

Manual  
Explore the algorithms and choose one.

**Algorithm**  
The algorithm that you want Amazon Forecast to use to train your predictor.

CNN-QR  
An Amazon proprietary convolutional neural network algorithm. Useful for 100+ time series and flexible with all forms of related time series and item metadata.

DeepAR+  
An Amazon proprietary recurrent neural network algorithm. Useful for 100+ time series with forward-looking related time series and item metadata.

Prophet  
A Bayesian structural time series model. Useful for time series with strong seasonal effects and several seasons of historical data.

NPTS  
Non-Parametric Time Series. An Amazon proprietary algorithm useful for sparse and intermittent time series.

ARIMA  
Autoregressive Integrated Moving Average. A statistical algorithm useful for datasets with fewer time series.

ETS  
Exponential Smoothing. A statistical algorithm useful for datasets with fewer time series and seasonality.

Figura 5-25.- Modelos disponibles en Amazon Forecast. (Consola AWS)

Por último, es posible seleccionar la opción de optimización de hiperparámetros (HPO), además, utilizar la estacionalidad, el histórico referente a datos meteorológicos y las fechas vacacionales, todo ello ajustado a la zona horaria seleccionada y el país. Para este caso de uso se utilizará únicamente la opción que tiene en cuenta los festivos.

### 5.2.2.2.- AutoGluon

El sistema de AutoGluon no ofrece soporte para el tratamiento de series temporales. Tampoco es posible la predicción de varios valores. Se probará el sistema para la resolución de problemas de regresión.

Para el entrenamiento de los modelos se utiliza la función `'fit'`. Se conservarán los valores por defecto exceptuando para el parámetro `'presets'`, que se establecerá con la configuración máxima calidad `'best_quality'`.

Respecto a los datos de entrada, se generarán retardos (`lagged_data`) con un tamaño de ventana 25. El conjunto de datos resultante se particionará con una proporción 80% y 20% para entrenamiento y test respectivamente.

Para el cálculo de las anomalías se seguirá el mismo procedimiento que en la metodología clásica, utilizando un umbral.

### 5.2.3.- Caso 3

En este apartado se utilizarán 3 sistemas AutoML para la resolución del caso de uso, clasificación de imágenes y detección de anomalías. Los sistemas seleccionados son: Google Cloud Platform, Amazon Web Services y AutoGluon.

Los sistemas de Google Vision y AutoGluon ofrecen funcionalidades para todos los escenarios planeados (clasificación binaria, multiclase o detección de objetos), aunque la detección de objetos solo será implementada con el sistema de Google por limitaciones temporales. Por otro lado, el sistema de Amazon Lookout Vision solo permite la clasificación binaria. Se realizarán las pruebas correspondientes para cada caso.

#### 5.2.3.1.- Google Cloud Platform

La plataforma de Google Cloud dispone de un módulo de visión artificial "Vision". Con esta herramienta es posible abordar tres problemas (Figura 5-26): (i) Clasificación de una sola etiqueta; cada imagen puede pertenecer solo a un tipo, para este caso de uso normal/anómalo. (ii) Clasificación multi-etiqueta; cada imagen puede pertenecer a varios tipos. (iii) Localización de objetos; se predice la ubicación de los objetos de interés dentro de las imágenes.

Selecciona el objetivo del modelo



Figura 5-26.- Funcionalidades del módulo Vision de GCP.

Una vez seleccionado el tipo de problema es necesario importar las imágenes. Las imágenes pueden estar albergadas en el equipo local o en Cloud Storage (Figura 5-27). No es necesario que las imágenes estén etiquetadas, se puede realizar posteriormente de forma manual. Los conjuntos de imágenes que se usarán en este caso de uso ya están etiquetados. Es necesaria la implementación de un archivo de manifiesto (archivo CSV). Este manifiesto contendrá la ruta relativa donde se localicen las imágenes, dentro del equipo local o Cloud Storage, y de que tipo (etiqueta) es cada una de ellas. Si se trata de un problema de localización de objetos es necesario indicar las coordenadas de la región de interés (Tabla 5.7).

ruta	etiqueta	X1	Y1	X2	Y2
gs://rayosx/Castings/C0001/C0001_0001.png	anomaly	0.257812	0.861888	0.222656	0.823427
gs://rayosx/Castings/C0001/C0001_0002.png	anomaly	0.328125	0.895105	0.294271	0.851399
gs://rayosx/Castings/C0001/C0001_0003.png	anomaly	0.386719	0.907343	0.352865	0.863636

Tabla 5.7.- Ejemplo de manifiesto para importar imágenes en Vision.

### Seleccionar archivos para importarlos

AutoML Vision utiliza tus imágenes para entrenar un modelo personalizado de aprendizaje automático. Antes de continuar con la importación, asegúrate de que el conjunto de imágenes está etiquetado correctamente y contiene imágenes suficientes para el resultado que quieres conseguir.

- Se debe asignar una etiqueta a cada cuadro delimitador
  - Puedes incluir cuadros delimitadores y etiquetas en tu CSV o cuando termine la importación
  - Te recomendamos que proporciones 100 cuadros delimitadores por etiqueta
- Subir imágenes desde el ordenador  
 Seleccionar un archivo CSV en Cloud Storage

Figura 5-27.- Menú de importación de imágenes en Vision. (consola GCP)

Como se adelantó anteriormente, es posible etiquetar las imágenes de forma manual, tanto las etiquetas simples como las regiones de interés (Figura 5-28). Además, GCP ofrece un módulo de etiquetado manual. Este módulo solicita a un equipo de personas que etiqueten de forma manual las imágenes basándose en una muestra representativa.

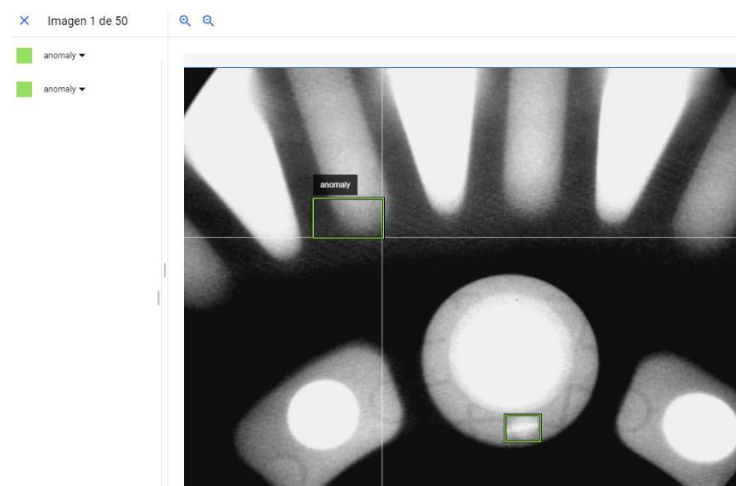


Figura 5-28.- Etiquetado manual en Vision. (consola de GCP)

No es posible seleccionar ningún parámetro para el entrenamiento. Existen dos tipos de configuraciones, "mayor precisión" y "predicciones rápidas" (Figura 5-13). Esta opción repercutirá sobre el tamaño del modelo y por tanto la latencia en las predicciones. La herramienta estima una latencia de 300ms-500ms en la predicción para la configuración rápida y 800ms-1500ms para el modelo más preciso. Por último, es necesario incluir el presupuesto del entrenamiento indicando las horas máximas que durará este.

✓ Define tu modelo

2 ¿Para qué quieres optimizar el modelo?

Objetivo	Precisión	Latency for Cloud
<input checked="" type="radio"/> Higher accuracy	Higher	800ms - 1,500ms
<input type="radio"/> Faster predictions	Lower	300ms - 500ms

Please note that prediction latency estimates are for guidance only. Actual latency will depend on your network connectivity.


CONTINUAR

3 Set a node hour budget

Figura 5-29.- Tipos de configuraciones en Vision. (Consola GCP)


Al igual que el módulo Tables, los modelos obtenidos con el módulo de visión artificial son exportables a TensorFlow y en contenedor Docker (Figura 5-30). Está disponible además una opción "Lite", orientada a modelos que serán desplegados en dispositivos móviles o dispositivos perimetrales.

### Use your model




**TF Lite**

Export your model as a TF Lite package to run your model on edge or mobile devices.



**TensorFlow.js**

Export your model as a TensorFlow.js package to run your model in the browser and in Node.js.



**Container**

Export your model as a TF Saved Model to run on a Docker container.

Figura 5-30.- Tipos de descarga del modelo entrenado en Vision. (consola de GCP)

### 5.2.3.2.- Amazon Web Services

Amazon Web Services (AWS) dispone de un módulo de visión artificial para la clasificación de imágenes "Amazon Lookout Vision" (ALV) (Figura 5-31). Este módulo solo permite la clasificación binaria normal/anomalía.

**Cómo preparar el conjunto de datos**

- Crear conjunto de datos**  
Agregue imágenes a su conjunto de datos. Las imágenes se utilizan para entrenar y probar el modelo. Para obtener mejores resultados, incluya imágenes con contenido normal y anómalo.
- Agregar etiquetas**  
Agregue etiquetas para clasificar las imágenes del conjunto de datos como normales o anómalas.

**Cómo entrenar su modelo**

- Entrenar modelo**  
Entrene el modelo con su conjunto de datos. Después del entrenamiento, el modelo puede detectar anomalías en nuevas imágenes. Es posible que el modelo requiera entrenamiento adicional para poder utilizarlo.

**Cómo evaluar su modelo**

- Comprobar las métricas de rendimiento**  
Utilice las métricas de rendimiento para evaluar la preparación del modelo. Puede seguir mejorando el modelo hasta que decida que está listo. Cuando el modelo esté listo, puede implementarlo.
- Mejorar el modelo**  
Para mejorar el modelo, ejecute una tarea de detección de prueba y verifique los resultados. A continuación, vuelva a entrenar el modelo para mejorar los resultados.

Buttons: Creado, Agregar etiquetas, Entrenar modelo, Comprobar métricas, Ejecutar detección de prueba

Figura 5-31.- Interfaz web del módulo de visión artificial de AWS, Amazon Lookout Vision.

Para la importación del conjunto de imágenes existen tres opciones (Figura 5-32): (i) mediante S3, el servicio de almacenamiento en la nube ofrecido por Amazon, (ii) desde el equipo local, las imágenes deben estar localizadas en dos carpetas denominadas “Normal”, para las imágenes que no tengan anomalías, y “Anomaly”, para las imágenes que sí, (iii) y desde “SageMaker Ground Truth”, el módulo de etiquetado manual de imágenes, que se mencionará más adelante en este apartado.

**Importar imágenes Información**  
Importe imágenes de alguna de los siguientes orígenes.

- Importar imágenes del bucket de S3**  
Utilice imágenes de un bucket de S3 existente introduciendo el URI del bucket de S3. Puede agregar etiquetas de manera automática en función de los nombres de las carpetas del bucket de S3.
- Cargar imágenes desde el equipo**  
Agregue imágenes cargando archivos de su equipo local. Puede cargar hasta 30 imágenes en cada oportunidad.
- Importar imágenes etiquetadas por SageMaker Ground Truth**  
Proporcione la ubicación del archivo .manifest. Si ha etiquetado los conjuntos de datos con otro formato, conviértalos a un formato .manifest.

Figura 5-32.- Opciones de importación de imágenes en Lokoout Vision. (Consola AWS)

Como segunda opción en el importado de imágenes, ALV permite dos opciones de estructuración de los datos, un solo conjunto de datos, o división en datos de entrenamiento y datos de prueba.

Con respecto al etiquetado de imágenes, es posible etiquetar estas a posteriori de su importación, o modificar las ya importadas (Figura 5-33). Al tratarse de un problema de clasificación binaria existen dos etiquetas, Normal/Anomaly.



Figura 5-33.- Ejemplo de etiquetado manual en ALV. (Consola de AWS)

Una vez importadas las imágenes es posible entrenar el modelo. ALV no ofrece ninguna opción/parámetro para el entrenamiento.

Finalizado el entrenamiento están disponibles ciertas métricas de rendimiento que se detallarán en el apartado de resultados.

Por último, ALV dispone de opción de descarga y despliegue del modelo a través de comandos de CLI de AWS. También es posible realizar predicciones a través de la interfaz web añadiendo nuevas imágenes.

### 5.2.3.3.- AutoGluon

Se utilizará el módulo de clasificación de imágenes y detección de objetos del sistema AutoGluon. Los módulos están disponibles a través de la API de Python. Como aspecto común a todos los módulos, AutoGluon ofrece el automatizado completo del pipeline de modelado a través de una función denominada `fit`. Solo es necesario indicar el conjunto de datos que contiene las imágenes para su uso.

Para crear el conjunto es necesario definir el directorio que contiene las imágenes. Las imágenes estarán distribuidas en carpetas, cuyo nombre se corresponde al de



la clase a la que pertenece la imagen. Como parámetros adicionales en la creación del conjunto de datos, es posible modificar el 'input\_size' y 'crop\_ratio', se conservarán los valores por defecto 224 y 0.875 respectivamente.

La ejecución utilizará la GPU por defecto, si se dispone de una. Es posible la ejecución distribuida utilizando el parámetro 'ngpus\_per\_trial', a través del cual se indica el número de GPUs.

El módulo de clasificación utilizará la arquitectura resnet50. Es posible utilizar cualquier arquitectura implementada en mxnet, librería de Gluon, utilizando el parámetro 'net'.

AutoGluon utilizará redes neuronales pre-entrenadas si las clases utilizadas en el conjunto de datos coinciden con las clases utilizadas para el entrenamiento de diversas redes disponibles.

Por defecto AutoGluon realizará un particionado con una proporción de 80% y 20% para los datos de entrenamiento y validación respectivamente.

Otra funcionalidad avanzada, pero que no se utilizará es la búsqueda de arquitectura de la red utilizando aprendizaje por refuerzo. Esta tecnología, denominada ProxylesNas (Figura 5-34), utiliza el entrenamiento de varias redes CNN de forma distribuida. Como novedad el entrenamiento se ve acelerado ya que utiliza aprendizaje por refuerzo y el premio "reward" se ve compartido entre las diferentes redes, evitando evaluar todos los datos de validación en cada red.

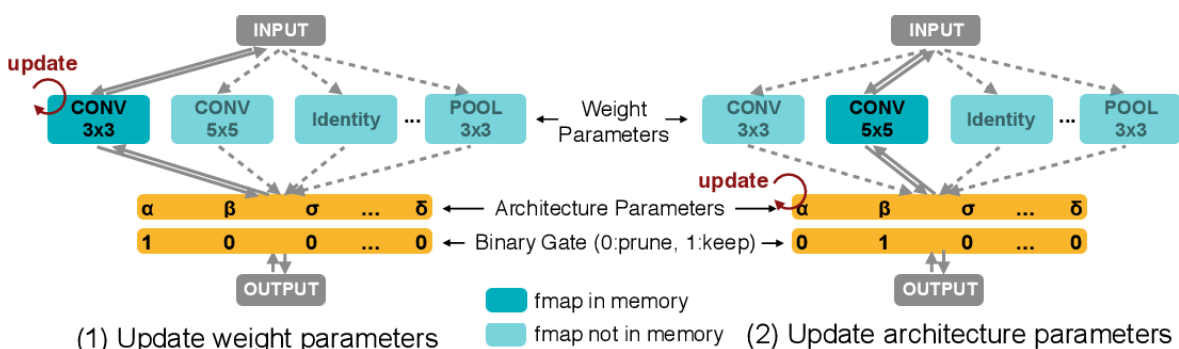


Figura 5-34.- Esquema gráfico de la tecnología ProxylesNas.

Para el entrenamiento de los modelos se han conservado los valores por defecto de todos los parámetros.

## 6.- Resultados.

### 6.1.- Tradicional (Ad-Hoc)

#### 6.1.1.- Caso 1

En este apartado se exponen los resultados utilizando los algoritmos implementados en la metodología detallada en [33]. En la Tabla 6.1 se muestran los resultados obtenidos con los 6 algoritmos propuestos. Para cada algoritmo se calculó la métrica de precisión (P) y recuperación (R). Únicamente se muestran los resultados para la clase minoritaria (fallo).

Métrica	RGF	GBDT	RF	SVM	LR	DT
<b>P (1)</b>	0.98	0.97	0.93	0.93	0.73	0.89
<b>R (1)</b>	0.98	0.96	0.94	0.5	0.81	0.87

Tabla 6.1.- Resultados en la clasificación de discos duros defectuosos con la metodología del estado del arte. (Ref.: [33])

En la tabla se observa que el desempeño de la mayoría de los modelos es sobresaliente. Se obtiene una media de 0.905 para la métrica de precisión y una recuperación de 0.84 de media. Dado que el modelo de vectores de soporte (SVM) es un clasificador lineal, y el dominio planteado es más complejo, este modelo es el que arroja peores resultados respecto a la recuperación, con un valor de 0.5. Por otro lado, se pueden apreciar mejores resultados con los modelos basados en árboles de decisión, RGF, GDBT y RF.

Respecto a los resultados obtenidos variando el umbral de confianza en la predicción, se observa que (Figura 6-1), con una confianza del 50%, la precisión en la predicción 30 días antes del fallo es de 0.56% aproximadamente. Variando el umbral de confianza esta precisión disminuye a 35% y 22% para los umbrales de 0.8 y 0.9 respectivamente. Con una confianza del 90% sería posible predecir los fallos, hasta una semana antes, de la mitad de los discos.

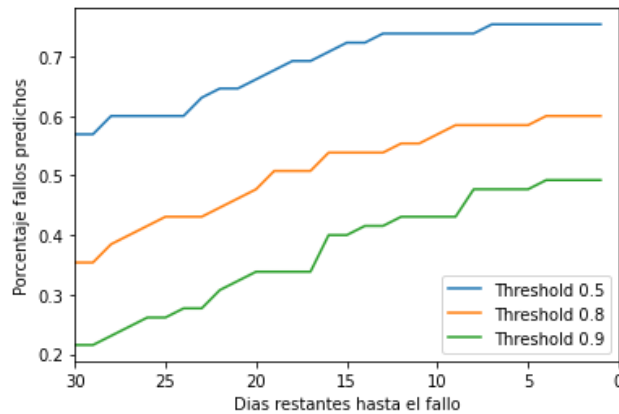


Figura 6-1.- Evolución de la precisión variando el umbral de confianza en la ventana temporal de 30 días previos al fallo del disco.

### 6.1.2.- Caso 2

Utilizando el método del mínimo nocturno sobre los datos históricos (años 2017, 2018 y 2019) se obtienen un elevado número de alarmas (posibles fugas en el sistema). Como se observa la Tabla 6.2, se genera una media de 442 positivos para todas las zonas en el periodo utilizado de 3 años, esto supone un total de 2.5 alarmas al día. Debido a las grandes fluctuaciones y ruido en las mediciones de caudal, el número de positivos es muy elevado, los cuales en su gran mayoría son falsos positivos. El método del mínimo nocturno es poco robusto ante grandes variaciones en el caudal.

ZONA	ALARMAS
340	517
339	486
341	406
270	249
343	552

Tabla 6.2.- Número de alarmas generadas utilizando el método del mínimo nocturno.

Por otro lado, el número de positivos utilizando la red neuronal LSTM ha sido muy inferior al método anterior. En la Tabla 6.3 se presentan las alarmas que se obtienen en un conjunto de datos de prueba. En la tabla se puede apreciar como el número de alarmas es menor utilizando las redes LSTM, hasta una diferencia de 31 positivos para la zona 339. Respecto a los valores de predicción de caudal, el error de estos frente a los valores reales es de 0.01433, utilizando una métrica de error absoluto

medio. El bajo error obtenido en la predicción puede servir de indicativo optimista como estimador de la precisión en las alarmas generadas, es decir, cuantos positivos son finalmente fugas y cuantos falsos positivos.

ZONA	Positivos LSTM	Positivos Mínimo Noc.	Dif.	MAE
270	3	21	+18	0.01824
<b>339</b>	<b>4</b>	<b>35</b>	<b>+31</b>	<b>0.02397</b>
340	5	28	+23	0.04228
341	1	7	+6	0.02476
343	1	7	+6	0.09742

Tabla 6.3.- Días estimados como anomalía utilizando LSTM frente al método del mínimo nocturno. El mejor incremento de mejora ha sido resaltado.

En la Tabla 6.4 se presentan las curvas de aprendizaje correspondientes a cada zona del sistema de aguas. Observando las curvas se puede determinar que no hay sobreajuste, y, por tanto, que el modelo a generalizado lo suficiente para nuevos datos de prueba.

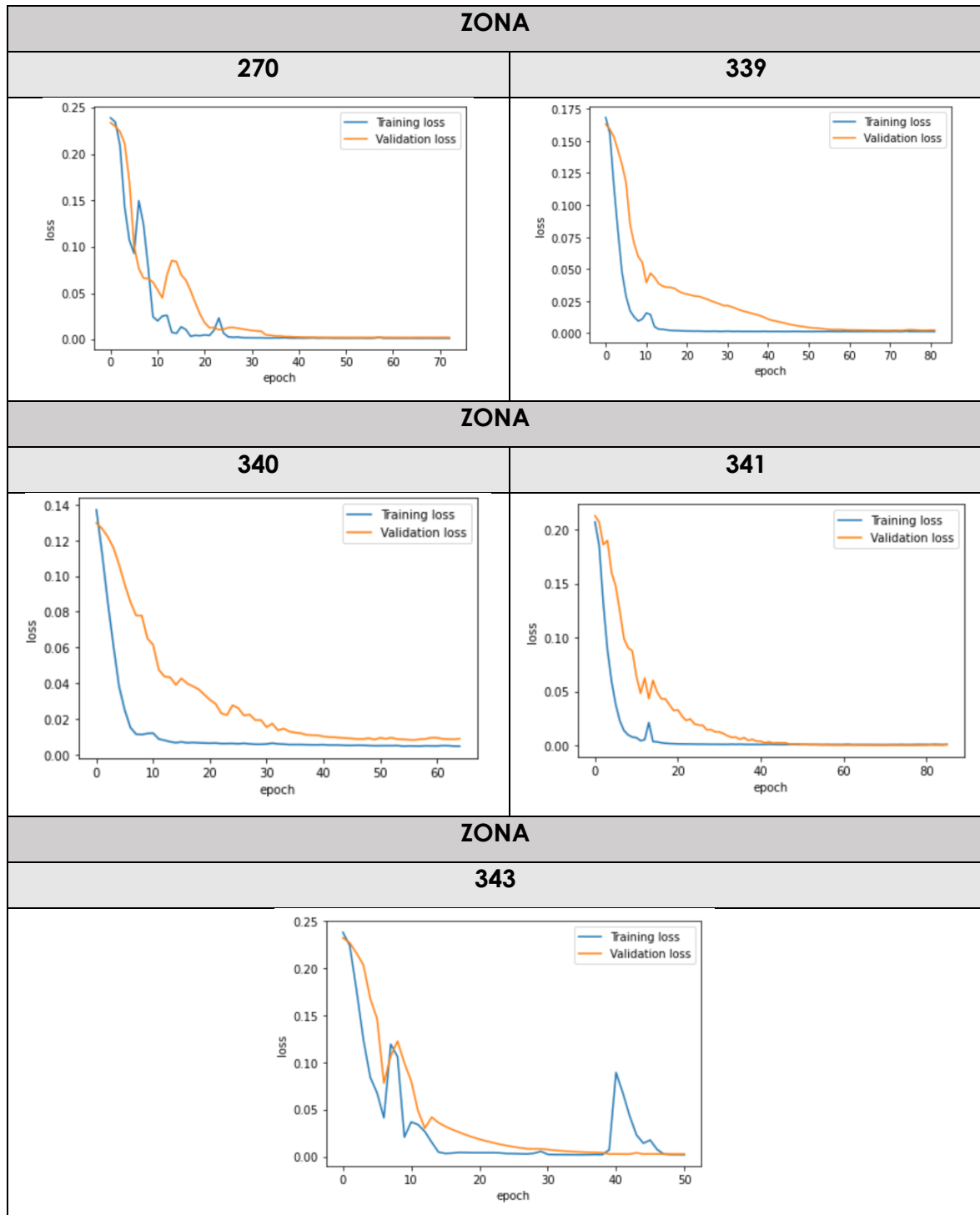


Tabla 6.4.- Curvas de aprendizaje de la red LSTM para cada zona.

Por último, en la Figura 6-2 se puede apreciar un ejemplo de gráfica para el periodo de 3 años en la que se indican las posibles fugas (en verde).

anomalies

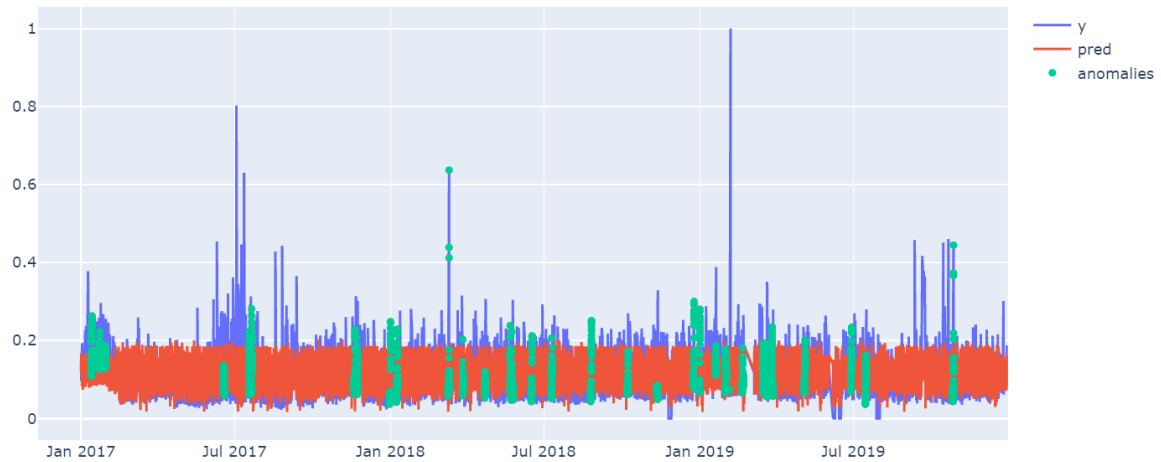


Figura 6-2.- Gráfico de los valores de caudal reales (azul), la predicción (rojo) y las anomalías destacadas (verde) para los 3 años históricos en la zona 339.

### 6.1.3.- Caso 3

A continuación se presentan los resultados en la clasificación de imágenes para la detección de anomalías. En la Tabla 6.5 se muestran los resultados en la clasificación binaria (la imagen contiene o no anomalías). En el mejor de los casos, para el conjunto de imágenes “cable”, se han logrado clasificar correctamente un 74% de las imágenes sin anomalías y un 48% con anomalías, con el algoritmo AutoEncoder con SSIM. Por otro lado, con L2 en los dataset de “metal\_nut” y “transistor”, con un 68% y 97% en las imágenes sin anomalías respectivamente, y 77% y 45% con anomalías. Como resultados generales, el número de falsos positivos para ambas clases es muy elevado, 20% de media para las imágenes sin anomalías y un 44% para las imágenes con anomalías.

Dataset	AE (SSIM)	AE (L2)	AnoGAN	CNN
metal_nut	1.00	<b>0.68</b>	0.86	0.55
	0.08	<b>0.77</b>	0.13	0.74
cable	<b>0.74</b>	0.93	0.98	0.97
	<b>0.48</b>	0.18	0.07	0.24
transistor	1.00	<b>0.97</b>	0.98	1.00
	0.03	<b>0.45</b>	0.35	0.15

Tabla 6.5.- Resultados de la evaluación de los métodos en clasificación binaria. Arriba el ratio de las imágenes sin defectos clasificadas correctamente, abajo el ratio para las imágenes con anomalías. Los valores del método con mejor resultado han sido resaltados.

Respecto a la clasificación multi-clase, en la Tabla 6.6 se comparan los resultados obtenidos por los algoritmos más populares del estado del arte y las tres arquitecturas propuestas por el artículo [40]. Ningún algoritmo destaca sobre el resto, se obtienen resultados similares con todas las soluciones propuestas, una precisión del 62% de media. Los valores de precisión no son elevados, como posible causa se puede destacar que solo se dispone de 851 imágenes con anomalías y 5 tipos de defecto, un conjunto de datos pequeño, ya que cada imagen puede contener más de un tipo de defecto. En un escenario favorable se recomienda disponer de 100 imágenes donde aparezca exclusivamente ese defecto.

Arquitectura	Precisión (%)
Alexnet	63.1
T-CNN	64.3
VGG-A	64.9
WRN-28-4	64.0
Densenet-121	52.5
ENAS-1	65.6
ENAS-2	65.5
ENAS-3	64.5
MetaQNN-1	66.0
MetaQNN-2	65.2
MetaQNN-3	64.9

Tabla 6.6.- Resultados para las diferentes redes neuronales aplicadas a un problema de clasificación multi-clase.

Por último, los resultados para el problema detección de objetos. Los resultados con las soluciones propuestas en el artículo [39] se muestran en la Tabla 6.7. Observando la tabla se puede apreciar como la segmentación en las imágenes favorece a una mejor clasificación, 95.7% de precisión en la detección de anomalías. Por otro lado, es destacable también el resultado obtenido con la red pre-entrenada de ImageNet, con la que se obtiene la misma precisión (95.7%) que con la red realizada por los expertos (93.1%, 95.7% y 92.1%).

Método	Accuracy
Detección de objetos	0.931
Detección de objetos + segmentación	0.957
RCNN	0.921
SDD ResNet101	0.762
Pre-entrenada ImageNet	0.957

Tabla 6.7.- Tasa de anomalías clasificadas correctamente en el problema de localización de anomalías en imágenes.

## 6.2.- Aprendizaje Máquina Automatizado (AutoML)

### 6.2.1.- Caso 1

#### 6.2.1.1.- Google Cloud Platform

En el primero de los ensayos con el sistema de AutoML de Google Tables, utilizando el conjunto de datos completo, no se obtienen resultados satisfactorios (Figura 6-3). Respecto a la precisión del modelo, esta métrica ha sido optimizada, con un valor del 100%, en cambio solo se ha detectado un disco fallido del conjunto de prueba de 71 discos, lo que supone una recuperación del 1.4%. Esta discrepancia entre el valor de precisión y el número de discos clasificados es debido al carácter desbalanceado de los datos.

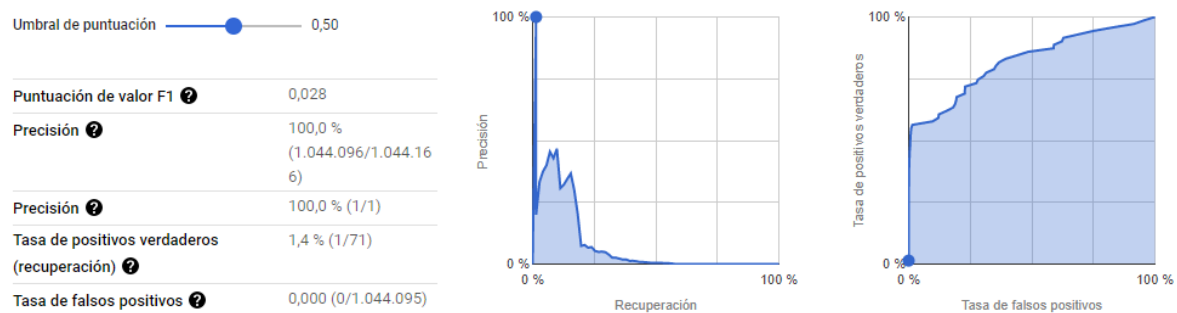


Figura 6-3.- Resultados de PRF para la clase minoritaria.

En el segundo de los ensayos se ha utilizado el conjunto de datos sub-muestreado (series temporales compactadas). Se obtiene una precisión del 98.9% y una recuperación del 78.4%, ambos con un umbral de confianza del 50%. Se logran clasificar correctamente 40 discos de 51 discos fallidos (Figura 6-4). La diferencia



respecto al primer ensayo es notoria, reduciendo los registros de la clase mayoritaria (un ratio de 33:1) se clasifica un 98.9% de 1489 discos de prueba en su clase correspondiente.

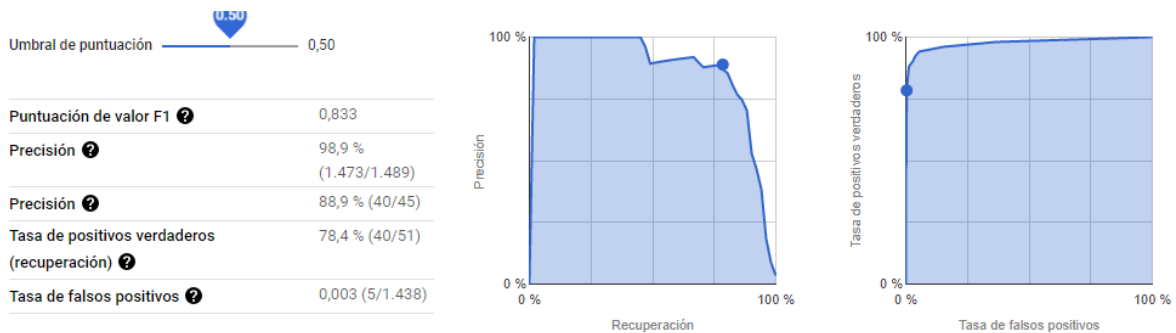


Figura 6-4.- Métricas de PRF para un umbral de clasificación del 50%.

En la siguiente gráfica (Figura 6-5) se muestra la importancia de características inferida con el último modelo entrenado. Los atributos de "smart\_240\_raw", "smart\_241\_raw" tienen un 29% y 27% de importancia respectivamente, seguido de "smart\_242\_raw" y "smart\_7\_normalized" con un 11% y 6% respectivamente.

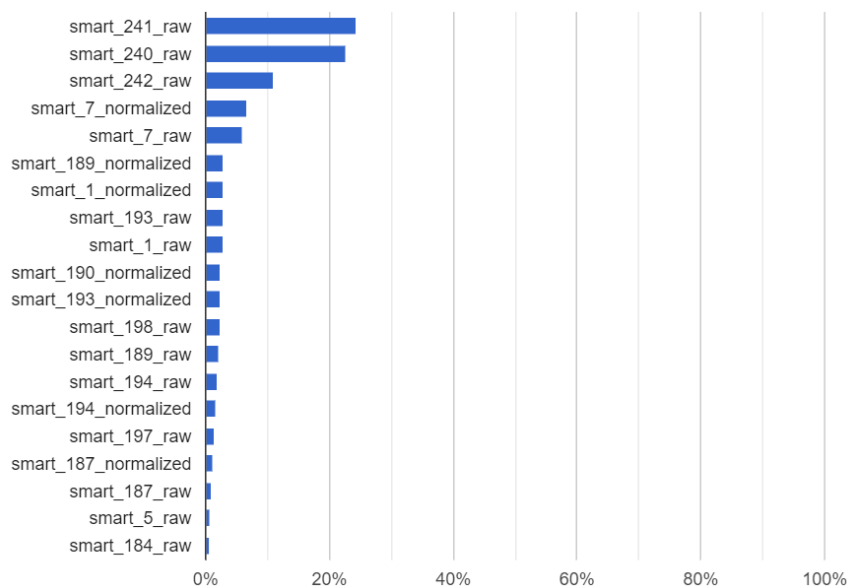


Figura 6-5.- Importancia de características obtenida con el sistema de Google Cloud Tables. (Consola de GCP)

**6.2.1.2.- AutoGluon**

Respecto al entrenamiento con el conjunto de datos completo utilizando AutoGluon, no se han obtenido resultados favorables. No se ha clasificado ningún disco fallido correctamente. Además, varias de las ejecuciones realizadas con la configuración de calidad máxima ('presets' con 'best\_quality') no han finalizado correctamente debido al alto uso de memoria.

Respecto a los resultados obtenidos con el conjunto de datos sub-muestreado, ratio de 33:1, en la Tabla 6.8 se muestran las matrices de confusión para ambas clases (0: disco sano y 1: disco fallido) correspondientes a las diversas pruebas, además del valor de F1. Utilizando el parámetro 'presets' por defecto los valores no varían, ya que los modelos utilizados y sus hiperparámetros para el entrenamiento son fijos. Por otro lado, con la configuración 'best\_quality' el valor de F1 promedio para la clase mayoritaria es del 94%, y el de la clase minoritaria del 81%. La elección de la métrica no altera destacablemente los resultados. Si se puede observar una ligera mejoría en la recuperación (135/170 discos) de la clase anómala con la métrica de 'balanced\_accuracy', en contraposición se han producido más falsos positivos, 36 discos frente a 17 y 16 con las otras métricas. Los resultados generales han mejorado con la configuración 'best\_quality', hasta una puntuación F1 del 82% con la métrica de 'Accuracy', frente al valor de 76% con la configuración por defecto.

<b>Métrica \ Configuración</b>	<b>defecto</b>				<b>best_quality</b>			
<b>Accuracy</b>		0	1	F1		0	1	F1
	0	5812	11	0.994	0	5806	17	<b>0.995</b>
	1	59	111	0.76	1	40	130	<b>0.82</b>
<b>Balanced_accuracy</b>		0	1	F1		0	1	F1
	0	5812	11	0.994	0	5787	36	0.993
	1	59	111	0.76	1	35	135	0.791
<b>F1_weighted</b>		0	1	F1		0	1	F1
	0	5812	11	0.994	0	5807	16	0.995
	1	59	111	0.76	1	41	129	0.819

Tabla 6.8.- Matrices de confusión para las distintas métricas de evaluación y los distintos valores del parámetro 'presets' en AutoGluon. El mejor resultado ha sido resaltado.

Por último, el sistema de AutoGluon propone la siguiente importancia de características (Figura 6-6). Los atributos más relevantes son "smart\_197\_raw", "smart\_194\_raw", "smart\_190\_normalized", "smart\_240\_raw" y "smart\_198\_raw", con valores de 0.121, 0.035, 0.023, 0.021 y 0.017 respectivamente (en tanto por 1).

```

smart_197_raw      0.121018
smart_194_raw      0.035643
smart_190_normalized 0.023996
smart_240_raw      0.021610
smart_198_raw      0.017822
smart_7_raw        0.009823
smart_187_raw      0.007493
smart_189_raw      0.006231
smart_1_raw        0.004659
smart_189_normalized 0.003564
smart_193_raw      0.002891
smart_190_raw      0.002807
smart_194_normalized 0.001965
smart_5_raw        0.000982
smart_187_normalized 0.000730
smart_198_normalized 0.000112
smart_197_normalized 0.000000
smart_184_raw      0.000000
smart_184_normalized 0.000000
smart_5_normalized -0.000028
smart_188_raw      -0.000028
smart_1_normalized -0.004631
smart_193_normalized -0.004659
smart_241_raw      -0.008728
smart_242_raw      -0.009683
smart_7_normalized -0.021470
dtype: float64

```

Figura 6-6.- Importancia de características calculadas con el sistema de AutoGluon.

### 6.2.1.3.- H2O

A continuación, se exponen los resultados obtenidos con el sistema de AutoML H2O. En primer lugar, la ejecución que utiliza todo el conjunto de datos no ha sido posible. Debido a que esta herramienta se ejecuta sobre un sistema Java, la carga de memoria y recursos es excesivamente elevada, junto con el gran tamaño del conjunto de datos. En la Tabla 6.9 se muestran los resultados con el conjunto de datos reducido (ratio 33:1) y los parámetros de H2O por defecto. El valor de recuperación para la clase positiva (1: discos fallidos) es del 60% y una precisión del 76%, lo que supone una métrica de F1 del 67.5% para un total de 165 discos fallidos en el conjunto de pruebas.

	0	1	F1 (%)
0	5700	31	99.1
1	65	100	67.5

Tabla 6.9.- Matriz de confusión para la clasificación con el sistema H2O sin utilizar balanceo de clases.

Para la segunda de las pruebas, utilizando la funcionalidad de balanceo de clases, se han conseguido clasificar 130 discos correctamente de 165 con un F1 del 83.3% (Tabla 6.10). Respecto a la clase mayoritaria (0: discos sanos), el valor de F1 es del 99.5%. Se puede determinar que el parámetro de balanceo de clases ha influido positivamente en la precisión de los modelos entrenados. Cabe destacar que el consumo de recursos es mayor con este parámetro, ya que se generan muestras sintéticas para balancear la proporción de registros para cada clase.

	<b>0</b>	<b>1</b>	<b>F1 (%)</b>
<b>0</b>	5714	17	99.5
<b>1</b>	35	130	83.3

Tabla 6.10.- Matriz de confusión para la clasificación con el sistema H2O utilizando balanceo de clases.

Por otro último, las características más relevantes (Figura 6-7) según el sistema de H2O son 'smart\_198\_raw' y 'smart\_197\_raw' con un 19% y 16% respectivamente, le sigue "smart\_187\_normalized" con un 5% de importancia.

	variable	relative_importance	scaled_importance	percentage
0	smart_198_raw	1.000000	1.000000	0.196827
1	smart_197_raw	0.825013	0.825013	0.162385
2	smart_187_normalized	0.254857	0.254857	0.050163
3	smart_187_raw	0.212781	0.212781	0.041881
4	smart_241_raw	0.187195	0.187195	0.036845
5	smart_188_raw	0.177898	0.177898	0.035015
6	smart_242_raw	0.166780	0.166780	0.032827
7	smart_7_normalized	0.165543	0.165543	0.032583
8	smart_193_raw	0.150229	0.150229	0.029569
9	smart_193_normalized	0.145825	0.145825	0.028702
10	smart_5_normalized	0.143420	0.143420	0.028229
11	smart_189_normalized	0.140230	0.140230	0.027601
12	smart_5_raw	0.136394	0.136394	0.026846
13	smart_189_raw	0.130532	0.130532	0.025692
14	smart_197_normalized	0.124822	0.124822	0.024568
15	smart_7_raw	0.123999	0.123999	0.024406
16	smart_1_raw	0.115839	0.115839	0.022800
17	smart_198_normalized	0.115317	0.115317	0.022698
18	smart_1_normalized	0.111809	0.111809	0.022007
19	smart_184_raw	0.107042	0.107042	0.021069

Figura 6-7.- Importancia de características calculado con el sistema de H2O.

## 6.2.2.- Caso 2

### 6.2.2.1.- AutoGluon

Se presentan los resultados obtenidos con el sistema de AutoGluon. Se han estimado las alarmas generadas por el modelo para los datos de test, y las alarmas que se obtienen con el método de mínimo nocturno (el utilizado actualmente) (Tabla 6.11). Únicamente en 2 de 5 zonas se ha logrado reducir el número de positivos, zona 339 y 340, con una diferencia de 2 y 12 alarmas respectivamente. Por otro lado, en el resto de las zonas el número de alarmas es muy elevado, hasta 77 para la zona 343, lo que supone 70 positivos más que los obtenidos con el mínimo nocturno. Cabe destacar que, aunque el error absoluto medio sea razonable, 0.01786 de media para todas las zonas, AutoGluon utiliza algoritmos que no contemplan el carácter temporal en los datos, y estos pueden no ser los más apropiados sobre este dominio.

ZONA	Positivos AutoGluon	Positivos Mínimo	Dif (Ud)	MAE
270	32	21	-11	0.00447
339	33	35	+2	0.00371
<b>340</b>	<b>16</b>	<b>28</b>	<b>+12</b>	<b>0.07542</b>
341	29	7	-22	0.00237
<b>343</b>	<b>77</b>	<b>7</b>	<b>-70</b>	<b>0.00334</b>

Tabla 6.11.- Resultados en la regresión con el sistema de AutoGluon. Se muestran los positivos obtenidos con el modelo de AutoGluon y los positivos utilizando el método de mínimo nocturno. La mejor y la peor tasa de mejora ha sido resaltada.

### 6.2.2.2.- Amazon Web Service

El sistema de Amazon Forecast permite la predicción de únicamente 500 puntos en el tiempo, por tanto, no ha sido posible calcular la tabla de positivos que se ha realizado para el resto de las herramientas. Por otro lado, Amazon Forecast permite observar el error por cuantiles de las predicciones. Es posible visualizar también el error obtenido con cada modelo individualmente. Se ha obtenido un error cuadrático medio de 0.6653 para el modelo con tamaño de ventana de 500 puntos con una frecuencia de 5 minutos, y para el modelo de 500 puntos con una frecuencia de 1 hora un MSE de 13.4978.

Para la realización de predicciones con Amazon Forecast es necesario indicar, a través de la interfaz web, la ventana temporal que se quiere predecir. Se mostrará de forma gráfica los valores estimados (Figura 6-8).

Como se explicó en el apartado de metodología 5.2.2.1.-, el cálculo de las alarmas (posibles fugas), no ha sido posible, ya que el sistema solo permite predecir 500 puntos.

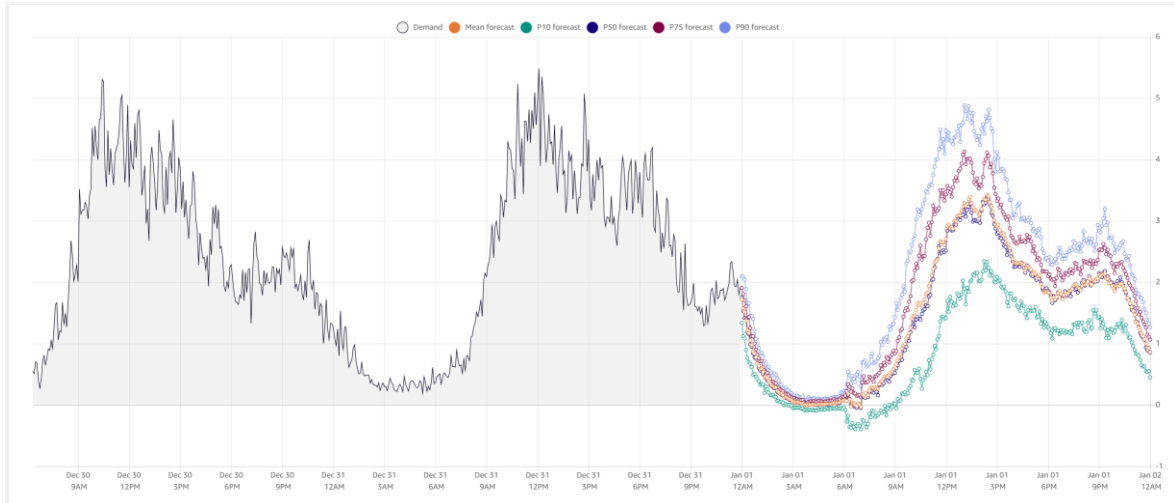


Figura 6-8.- Predicción de 500 puntos (con frecuencia de 5 minutos) con el sistema de Amazon Forecast.

### 6.2.3.- Caso 3

#### 6.2.3.1.- Google Cloud Platform

Referente a la clasificación binaria con el sistema de Google Vision, los resultados de precisión y recuperación para 4 niveles de umbral se muestran en la Tabla 6.12. Los resultados son favorables para ambos conjuntos de imágenes. En "metal\_nut" se logra una precisión del 100% para todos los niveles de umbral, y una recuperación del 57% con el nivel de confianza de 0.9. Para el conjunto "transistor" la recuperación se ve afectada por el aumento del umbral, descendiendo a un 0% de imágenes con anomalías correctamente clasificadas. En este último caso, aunque con un umbral del 0.5 la precisión tenga un valor elevado del 84%, observando la matriz de confusión que se muestra en la Tabla 6.13, se puede apreciar cómo no se ha logrado clasificar correctamente ninguna de las imágenes con anomalías (4 imágenes). En este caso la proporción desbalanceada de imágenes sin y con anomalías ha afectado negativamente al entrenamiento del modelo, ocasionando resultados no favorables.

Dataset	Umbral	Precisión (%)	Recuperación (%)
<b>metal_nut</b>	0.5	100	100
	0.75	100	91.43
	0.85	100	82.86
	0.9	100	57.14
<b>transistor</b>	0.5	84.38	83.38
	0.75	100	71.88
	0.85	100	46.88
	0.9	100	0

Tabla 6.12.- Precisión y Recuperación en clasificación binaria para varios valores de confianza (umbral) en la clasificación. Los valores de PR están calculados para ambas clases.

	Normal	Anomaly
<b>Normal</b>	25	-
	27	1
<b>Anomaly</b>	-	10
	4	-

Tabla 6.13.- La matriz de confusión para un umbral de 0.5. Arriba los resultados para 'metal\_nut', abajo para 'transistor'.

Los resultados para la clasificación multi-clase se muestran en la Tabla 6.14. Los valores de PR son buenos para un umbral de del 0.5. La precisión se mantiene constante a medida que se aumenta el umbral de clasificación, alcanzando un 94.9%. Por otro lado, la recuperación descende hasta valores del 50% con un umbral de confianza de 0.9.

Dataset	Umbral	Precisión (%)	Recuperación (%)	F1 (%)
<b>concrete</b>	0.5	85.3	83.	84.2
	0.75	91.6	71.6	80.3
	0.85	93.6	62.2	74.7
	0.9	94.9	50.1	65.6

Tabla 6.14.- Precisión y Recuperación en clasificación multi-clase para varios valores de confianza (umbral) en la clasificación.

Respecto a las clases clasificadas (Figura 6-9), "ExposedBars" es la clase que más falsos positivos y falsos negativos con 55% de las anomalías incorrectamente



clasificadas, le sigue "Spallation" y "CorrosionStain" con un 41% y 39% respectivamente.

Etiqueta "true"	Etiqueta predicha						
	Efflorescence	Crack	ExposedBars	CorrosionStain	Background	Spallation	
Efflorescence	66 %	8 %	1 %	3 %	15 %	8 %	
Crack	1 %	82 %	3 %	4 %	6 %	4 %	
ExposedBars	1 %	1 %	45 %	14 %	2 %	37 %	
CorrosionStain	3 %	-	-	61 %	27 %	9 %	
Background	2 %	-	1 %	1 %	96 %	-	
Spallation	-	24 %	-	-	18 %	59 %	

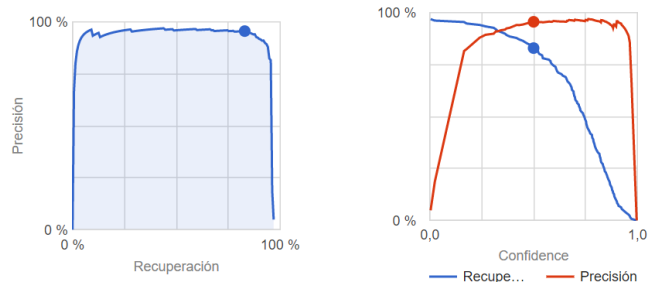
Etiqueta "true"	Etiqueta predicha						
	Efflorescence	Crack	ExposedBars	CorrosionStain	Background	Spallation	
Efflorescence	98	12	1	4	22	12	
Crack	1	115	4	6	9	5	
ExposedBars	1	2	62	20	3	51	
CorrosionStain	1	-	-	20	9	3	
Background	3	-	1	2	144	-	
Spallation	-	4	-	-	3	10	

Figura 6-9.- Matriz de confusión para un umbral de 0.5. A la izquierda en tanto por ciento, a la derecha en unidades.

Para el problema de detección de objetos es posible modificar el umbral de clasificación y el umbral de intersección sobre unión. Para un umbral de clasificación y de intersección de 0.5 se obtiene una precisión de 95.58% y una recuperación de 82.93%, resultados sobresalientes.

N.º total de imágenes	765
Elementos de prueba	86
Número total de objetos	287
Índice de objetos por imagen	3,34
Precisión	95,58 %
Recuperación	82,93 %

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.  
[Learn more about these metrics and graphs.](#)



A través de la interfaz web es posible visualizar las imágenes clasificadas errónea y correctamente (Figura 6-10, Figura 6-11). También se puede visualizar las regiones de interés propuestas por el algoritmo, tanto para las anomalías detectadas correctamente como para los falsos positivos.

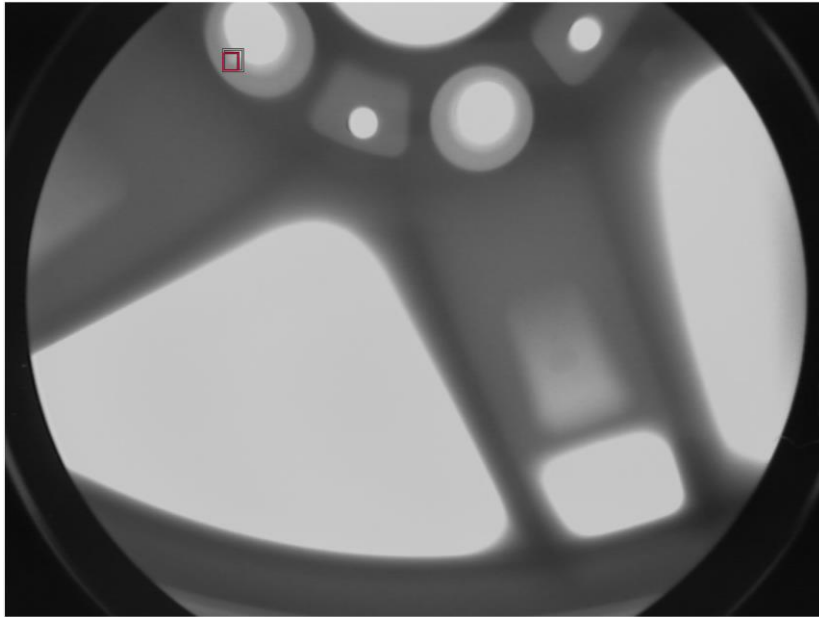


Figura 6-10.- Falso positivo. En rojo la región de interés propuesta como anomalía por el algoritmo.

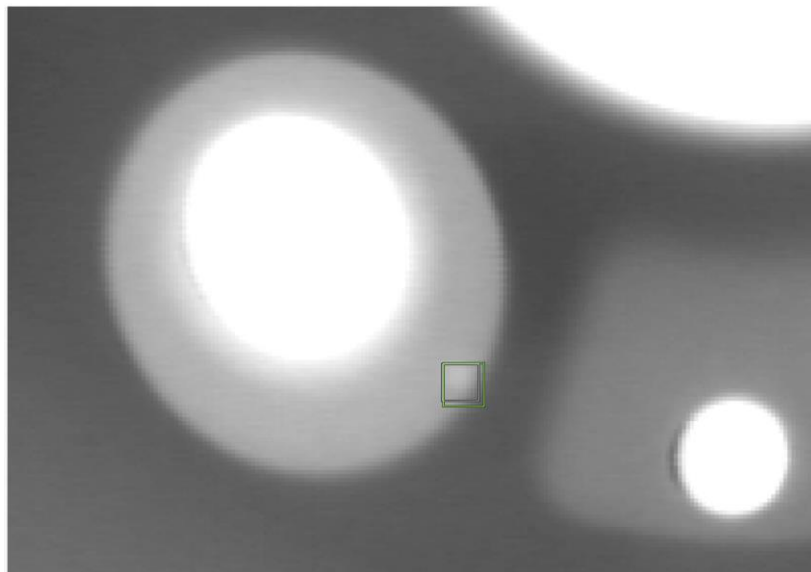


Figura 6-11.- Positivo verdadero. En verde la región de interés propuesta como anomalía por el algoritmo.

### 6.2.3.2.- Amazon Web Services

Los resultados para los 3 conjuntos de imágenes probados en el problema de clasificación binaria utilizando Amazon Lookout Vision son sobresalientes. Como se muestra en la Tabla 6.15 se han clasificado todas las imágenes correctamente, exceptuando un falso positivo en el dataset "cable".

Dataset	Precisión (%)	Recuperación (%)	F1 (%)	Matriz de confusión		
					Normal	Anomaly
metal_nut	100	100	100		Normal	Anomaly
				Normal	58	0
				Anomaly	0	30
transistor	100	100	100		Normal	Anomaly
				Normal	61	0
				Anomaly	0	15
cable	100	95.2	97.6		Normal	Anomaly
				Normal	65	1
				Anomaly	0	20

Tabla 6.15.- Métricas de rendimiento para la clasificación binaria con Amazon Lookout Vision. Los valores de PRF están dados para la clase anomalía.

Es importante destacar que en determinados casos el sistema utiliza un umbral de confianza excesivamente bajo para realizar la clasificación (Figura 6-12). Como se muestra en la siguiente figura se han clasificado anomalías con una confianza de hasta el 2.2%. Como se remarcó anteriormente no es posible modificar dicho umbral.



Figura 6-12.- Ejemplo de imágenes correctamente clasificadas con un umbral de confianza bajo.

### 6.2.3.3.- AutoGluon

En la Tabla 6.16 se exponen los resultados obtenidos con el sistema de AutoML de AutoGluon en el problema de clasificación binaria de imágenes. Se han obtenido buenos valores de precisión para todos los casos, exceptuando el dataset "metal\_nut" con un input\_size de 128, con una precisión del 68.7%. Por otro lado, respecto a la recuperación no se han obtenido resultados favorables, obteniendo

valores cercanos al 50% para los dos mejores casos de precisión. El peor de los casos ha sido con el conjunto de imágenes de “transistor”, donde se han clasificado únicamente 1/7 imágenes con anomalías.

Dataset	Precisión (%)	Recuperación (%)	F1 (%)	Matriz de confusión		
<b>metal_nut</b> <b>input_size=288</b>	90.9	52.6	66.6		Normal	Anomaly
				Normal	50	1
				Anomaly	9	10
<b>metal_nut</b> <b>input_size=128</b>	68.7	57.8	62.8		Normal	Anomaly
				Normal	46	5
				Anomaly	8	11
<b>transistor</b>	100	12.5	<b>22.2</b>		Normal	Anomaly
				Normal	55	0
				Anomaly	<b>7</b>	<b>1</b>
<b>cable</b>	90.9	52.6	66.6		Normal	Anomaly
				Normal	56	1
				Anomaly	9	10

Tabla 6.16.- Métricas de rendimiento para la clasificación binaria con AutoGluon. Los valores de PRF están dados para la clase anomalía. El peor caso ha sido resaltado.

Observando la curva de aprendizaje mostrada en la Figura 6-13, se puede determinar que el entrenamiento es correcto y que con un mayor número de imágenes se obtendrían mejores resultados.

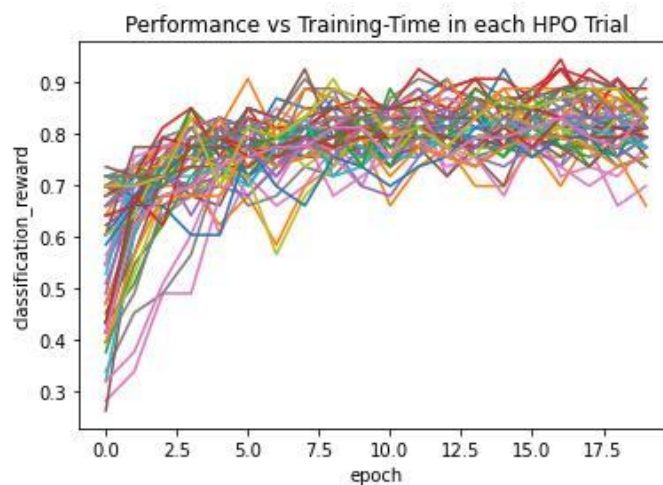


Figura 6-13.- Curva de aprendizaje para cada red entrenada.

## 7.- **Discusión.**

El avance tecnológico y la popularidad en los temas relacionados con la inteligencia artificial propician en la investigación y desarrollo de cada vez algoritmos más útiles y eficientes. Además, al tratarse de un tema de intereses, existe un gran apoyo de la comunidad permitiendo a los usuarios/compañías más nóveles introducirse en él. Por otro lado, existe también una gran variedad de herramientas de AutoML. Multinacionales como Google, Amazon y Azure están volcando un gran esfuerzo en desarrollar y mejorar estos productos, además, están disponibles también herramientas competitivas de AutoML gratuitas y de código abierto. A continuación, se presentan una serie de puntos que se han considerado interesantes respecto al uso de tecnologías de AutoML, y una visión dual de estas frente al uso de metodologías más tradicionales utilizadas en el estado del arte.

En primer lugar, existe una gran variedad de herramientas de AutoML. Los problemas más habituales dentro de la resolución de problemas utilizando modelado, regresión, clasificación, imágenes, etc., son resolubles utilizando estas herramientas. También es posible encontrar herramientas para todos los niveles de experiencia en inteligencia artificial, así como diversos niveles de informática y programación. Por un lado, sistemas como las ofrecidas por Google, Amazon, Azure, etc., aportan una infraestructura hardware que capacita la implementación de modelos complejos y sofisticados, además de ofrecer un portal web que habilita el uso de estas herramientas a personas sin conocimientos de informática e inteligencia artificial. Respecto a las herramientas como AutoGluon, H2O, TPOT, Auto-Sklearn, etc., son necesarios conocimientos de programación, ya que todas ellas están basadas en lenguajes como Python, Java, o R. Es importante destacar que estos sistemas permiten, de una forma muy automatizada, la implementación de un pipeline relativamente complejo, con apenas unas líneas de código.

Respecto a las funcionalidades de los sistemas de AutoML, las soluciones comerciales soportan la interoperabilidad con toda la infraestructura de servicios que ofrecen, Google Cloud, Amazon Web Services, Microsoft Azure, etc. Estos servicios aportan funcionalidades extra a parte de la implementación del modelo, como el despliegue de este, la monitorización, y el control de versiones, entre otros.

Por otro lado, las herramientas de código abierto se ven limitadas en uso exclusivamente a las etapas que se corresponden con el desarrollo. Estos últimos en cambio, disponen de una gran granularidad respecto a implementación y personalización de los parámetros de entrenamiento, aspecto del que carecen las herramientas comerciales.

Comparando los sistemas de AutoML con la implementación de algoritmos Ad-Hoc, estos últimos requieren un mayor conocimiento en programación y modelado. Además, la implementación de un modelo, junto al resto de etapas del pipeline, requiere más tiempo que los sistemas automatizados. Es importante destacar que, al igual que en los sistemas de AutoML, una misma implementación del pipeline para un problema concreto, puede ser utilizada en diversos problemas requiriendo ligeras modificaciones, ahorrando tiempo y recursos.

Otro aspecto importante es el coste económico. Las herramientas comerciales de los grandes fabricantes tienen precios elevados, tanto por suscripción periódica, como tasas fijas por implementación, entrenamiento y despliegue. Por ejemplo, con el sistema de Google el entrenamiento de un modelo tabular con un presupuesto de 3 horas tiene un coste de aproximadamente 50€, en contrapartida de los sistemas opensource como AutoGluon, o la implementación Ad-Hoc, que no requieren de coste adicional salvo la infraestructura hardware, que corre por cuenta del desarrollador.

Referente a los resultados cuantitativos, se han obtenido resultados aceptables para ambas metodologías, y para los tres problemas de modelado resueltos, clasificación, regresión y clasificación de imágenes. En los problemas de clasificación, los 3 sistemas utilizados, Google, H2O y AutoGluon ofrecen resultados aceptables. Para este caso se pueden destacar las herramientas gratuitas frente a la de Google ya que permiten una mayor personalización de los parámetros de entrenamiento y la rapidez en la implementación es muy superior. La gran mayoría de sistemas de AutoML no ofrecen soporte para el tratamiento de datos desbalanceados, únicamente el sistema de H2O ofrece estas funcionalidades, pero se ve limitado enormemente por los requisitos de hardware, siendo la implementación clásica la única solución para este escenario.

Por otro lado, solo una minoría de sistemas ofrece funcionalidades para el tratamiento de series temporales y problemas de forecasting, y, al igual que en los problemas de clasificación desbalanceada, el uso de AutoML para este problema no es una opción viable. Se puede destacar el sistema de Amazon frente al resto de herramientas, por ser la única en dar soporte a los problemas de series temporales y forecasting, además de ofrecer diversos parámetros de entrenamiento y selección de varios algoritmos que se adecuen más al tipo de problema.

En la resolución de problemas de clasificación de imagen y detección de objetos, los sistemas de AutoML son comparables a la implementación Ad-Hoc, obteniendo resultados similares. Los sistemas comerciales como el de Google y Amazon destacan notablemente, tanto en su facilidad de uso e implementación, como en los resultados cuantitativos. Las herramientas gratuitas arrojan también resultados aceptables, pero son más lentos que las comerciales y además consumen una gran cantidad de recursos.

Otro aspecto de interés es el tiempo y flexibilidad en la implementación de modelos en una etapa temprana de prototipado y exploración. Cuando no se tiene un gran conocimiento en el dominio, o se presenta un problema nuevo, es habitual el entrenamiento de un elevado número de modelos variando la configuración y parámetros, como desventaja, las herramientas de AutoML comerciales tienen un alto coste económico, y realizar prueba y error puede no ser viable debido este aspecto.

Como tónica general la resolución de problemas de inteligencia artificial y modelado requiere aún muchos conocimientos técnicos en estadística, matemáticas, ingeniería, etc., requerimiento que en los sistemas de AutoML se atenúan, en cambio, la ingeniería de datos en los problemas de modelado es un paso indispensable para ambas metodologías, el conocimiento en el dominio y el expertise sigue siendo un factor crítico.

Otro aspecto interesante es el apoyo de la comunidad y el soporte de ayuda. Referente a este punto todas las herramientas seleccionadas cuentan con una gran cantidad de documentación y manuales, tanto de la comunidad opensource, como del equipo de soporte de las herramientas comerciales.

Además, todas estas herramientas se están aún mejorando, disponiendo de versiones más actualizadas, más robustas, y con más funcionalidades cada día.

Atendiendo al uso de recursos y rendimiento, los sistemas de AutoML requieren unas especificaciones de hardware elevadas. La demanda de recursos en los sistemas de AutoML es notablemente mayor a los algoritmos ad-hoc, pero las herramientas de los grandes fabricantes abstraen los requerimientos de hardware del desarrollo, permitiendo un uso más flexible y escalable.

Por último, debido a las limitaciones temporales y de recursos no se han expuesto en este trabajo aspectos relacionados con el tratamiento de datos multidimensionales y de bigdata. Se ha realizado una aproximación a este escenario con el problema abordado en [53]. Como conclusión, la gran mayoría de sistemas de AutoML no soportan datos de más de dos dimensiones ("mcfly" [54]). Por otro lado, estos sistemas tampoco permiten, o se dificulta, el uso de datos de gran tamaño. Las herramientas comerciales limitan el uso de recursos y no es posible plantear este escenario, por otro lado, las herramientas gratuitas tienen un gran consumo de recursos y es necesario disponer de infraestructura adecuada para ello.



## 8.- Conclusiones y trabajo futuro.

Como puntos que se han abordado en este documento se puede destacar que:

- Se han presentado diversas metodologías del estado del arte para la resolución de problemas de clasificación, regresión, y clasificación de imágenes y localización de objetos, para la detección de anomalías.
- Se han presentado diversas soluciones del estado del arte para el tratamiento de datos de naturaleza desbalanceada.
- Se han presentado diversas soluciones del estado del arte para el tratamiento de datos de carácter o series temporales.
- Se ha realizado un estudio de alternativas de soluciones de AutoML de código abierto y comerciales.
- Se han utilizado diversas herramientas de AutoML para la resolución de tres casos de uso.

Como conclusiones se destacan los siguientes puntos:

- Existe una gran variedad de herramientas de AutoML. Estas herramientas cuentan con madurez necesaria para la resolución problemas de inteligencia artificial.
- Las herramientas de AutoML pueden resolver los problemas más habituales enumerados en el primer punto.
- Las herramientas de AutoML son útiles para prototipado debido a su rapidez de implementación y tiempo en el entrenamiento.
- Los resultados en la clasificación de imágenes y detección de regiones son sobresalientes con las herramientas de AutoML.
- Existen herramientas de AutoML que ofrecen soluciones sin necesidad de conocimientos en IA ni en programación.
- Las distintas soluciones de AutoML ofrecen varios grados de granularidad que se ajustan a los distintos perfiles de usuario, nóveles y expertos en el área del modelado y la informática.

- Los resultados en el entrenamiento de los modelos son dependientes de la calidad de los datos, tanto para los sistemas de AutoML como en la metodología ad-hoc.
- La ingeniería de datos y el conocimiento en el dominio es indispensable en la resolución de problemas de modelado e inteligencia artificial.
- La mayoría de los sistemas de AutoML no ofrece funcionalidades para la resolución de problemas con datos de carácter desbalanceado, series temporales, forecasting, multidimensionalidad y grandes conjuntos de datos (bigdata).
- El uso de recursos en los sistemas gratuitos de AutoML es elevado y necesaria una infraestructura hardware adecuada.
- Las herramientas comerciales de AutoML ofrecen una infraestructura hardware y de servicios que dan soporte a todas las etapas del ciclo de vida de un modelo de inteligencia artificial.
- Las herramientas comerciales de AutoML pueden no ser accesibles para todos los usuarios debido a su coste económico.

Como trabajo futuro se probarán otros aspectos relevantes dentro del ciclo de vida del ML, como el despliegue y la monitorización de los modelos. Por otro lado, se probarán los sistemas de AutoML en entornos de producción y requisitos de rendimiento exigentes. Se estudiarán otros sistemas de AutoML y demás funcionalidades compatibles con estos.

## 9.- Bibliografía.

- [1] A. Rojko, "Industry 4 . 0 Concept : Background and Overview," vol. 11, no. 5, pp. 77–90, 2017.
- [2] A. Ustundag and E. Cevikcan, *Managing The Digital Transformation*. 2018.
- [3] A. L. Samuel, "Some Studies in Machine Learning," *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, 1959.
- [4] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Dev.*, vol. 44, no. 1–2, pp. 207–219, 2000.
- [5] J. A. Wass, "Weka machine learning workbench," *Scientific Computing*, vol. 24, no. 3. 2007.
- [6] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller, "Scikit-learn," *GetMobile Mob. Comput. Commun.*, vol. 19, no. 1, pp. 29–33, 2015.
- [7] P. Das *et al.*, "Amazon SageMaker Autopilot: A white box AutoML solution at scale," *Proc. 4th Work. Data Manag. End-To-End Mach. Learn. DEEM 2020 - conjunction with 2020 ACM SIGMOD/PODS Conf.*, 2020.
- [8] T. K. Finley, "The Democratization of Artificial Intelligence: One Library's Approach," *Inf. Technol. Libr.*, vol. 38, no. 1, pp. 8–13, 2019.
- [9] F. Hutter, *Meta-learning*, vol. 498. 2014.
- [10] E. Irissou, J. Legoux, B. Arsenault, and C. Moreau, "NRC Publications Archive (NPARC) Archives des publications du CNRC (NPARC)," *J. Therm. Spray Technol.*, vol. 16, p. 5, 2007.
- [11] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: methods and prospects," *Big Data Anal.*, vol. 1, no. 1, pp. 1–22, 2016.
- [12] F. Neumann, "Automated Algorithm Selection :," vol. 27, no. October, pp. 3–45, 2018.
- [13] H. Guo, "A bayesian approach for automatic algorithm selection," *IJCAI 2003 - Proc. Int. Jt. Conf. Artif. Intell. Work. AI Auton. Comput.*, no. August, pp. 1–5, 2003.
- [14] G. Luo, "A review of automatic selection methods for machine learning algorithms and hyper-parameter values," *Netw. Model. Anal. Heal. Informatics Bioinforma.*, vol. 5, no. 1, pp. 1–16, 2016.
- [15] M.-A. Zöllner and M. F. Huber, "Benchmark and Survey of Automated Machine Learning Frameworks," vol. 1, no. 1993, pp. 1–15, 2019.
- [16] M. Milutinovic, B. Schoenfeld, D. Martinez-Garcia, S. Ray, S. Shah, and D. Yan, "On Evaluation of AutoML Systems," *ICML Work. Autom. Mach. Learn.*, 2020.
- [17] ISG, "List of AutoML Tools and Software Libraries," <https://isg.beel.org/blog/2020/04/09/list-of-automl-tools-and-software->

libraries/..

- [18] AIMultiple, "AutoML Software / Tools in 2021: In-depth Guide," <https://research.aimultiple.com/auto-ml-software/>..
- [19] A. Balaji and A. Allen, "Benchmarking Automatic Machine Learning Frameworks," 2018.
- [20] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," vol. 15, no. 2, pp. 49–60, 2014.
- [21] P. Gijssbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, "An Open Source AutoML Benchmark," pp. 1–8, 2019.
- [22] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools," *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, vol. 2019-Novem, no. 2017, pp. 1471–1479, 2019.
- [23] T. Halvari, J. K. Nurminen, and T. Mikkonen, "Testing the robustness of automl systems," *Electron. Proc. Theor. Comput. Sci. EPTCS*, vol. 319, pp. 103–116, 2020.
- [24] M. Blohm, "Leveraging Automated Machine Learning for Text Classification : Evaluation of AutoML Tools and Comparison with Human Performance."
- [25] N. Erickson *et al.*, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data," 2020.
- [26] M. Feuerer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," *Adv. Neural Inf. Process. Syst.*, vol. 2015-Janua, pp. 2962–2970, 2015.
- [27] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," *Proc. 1st ACM Symp. Cloud Comput. SoCC '10*, pp. 193–203, 2010.
- [28] I. Kecerdasan and P. Ikep, "Monitoring Hard Disks with SMART," p. 6.
- [29] C. A. C. Rincón, J. F. Paris, R. Vilalta, A. M. K. Cheng, and D. D. E. Long, "Disk failure prediction in heterogeneous environments," *Simul. Ser.*, vol. 49, no. 10, pp. 113–119, 2017.
- [30] J. Li *et al.*, "Hard drive failure prediction using classification and regression trees," *Proc. Int. Conf. Dependable Syst. Networks*, pp. 383–394, 2014.
- [31] Y. Xu *et al.*, "Improving service availability of cloud systems by predicting disk error," *Proc. 2018 USENIX Annu. Tech. Conf. USENIX ATC 2018*, pp. 481–493, 2020.
- [32] Y. Z. Lin, "Machine Learning for Storage System Reliability Prediction."
- [33] M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann, "Predicting disk replacement towards reliable data centers," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-Augu, no. 1, pp. 39–48, 2016.
- [34] "BackBlaze," <https://www.backblaze.com/b2/hard-drive-test-data.html>.

- [35] E. Farah and I. Shahrour, "Leakage Detection Using Smart Water System: Combination of Water Balance and Automated Minimum Night Flow," *Water Resour. Manag.*, vol. 31, no. 15, pp. 4821–4833, 2017.
- [36] N. R. Prasad, S. Almanza-Garcia, and T. T. Lu, "Anomaly detection," *Comput. Mater. Contin.*, vol. 14, no. 1, pp. 1–22, 2009.
- [37] B. Wang, "The Future of Manufacturing: A New Perspective," *Engineering*, vol. 4, no. 5, pp. 722–728, 2018.
- [38] F. Adamo, F. Attivissimo, G. Cavone, N. Giaquinto, and A. M. L. Lanzolla, "Artificial vision inspection applied to leather quality control," *18th IMEKO World Congr. 2006 Metrol. a Sustain. Dev.*, vol. 3, pp. 1970–1972, 2006.
- [39] M. Ferguson, R. Ak, Y. T. T. Lee, and K. H. Law, "Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning," *Smart Sustain. Manuf. Syst.*, vol. 2, no. 1, pp. 137–164, 2018.
- [40] M. Mundt, S. Majumder, S. Murali, P. Panetsos, and V. Ramesh, "Meta-learning convolutional neural architectures for multi-target concrete defect classification with the concrete defect bridge image dataset," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 11188–11197, 2019.
- [41] D. Mery *et al.*, "GDxray: The Database of X-ray Images for Nondestructive Testing," *J. Nondestruct. Eval.*, vol. 34, no. 4, pp. 1–12, 2015.
- [42] P. Bergmann, "MVTec AD — A Comprehensive Real-World Dataset for Unsupervised.pdf," pp. 9592–9600.
- [43] "Machine Learning for Storage System Reliability Prediction," p. 2018, 2018.
- [44] S. R. Mounce, R. B. Mounce, and J. B. Boxall, "Novelty detection for time series data analysis in water distribution systems using support vector machines," *J. Hydroinformatics*, vol. 13, no. 4, pp. 672–686, 2011.
- [45] X. Wang, G. Guo, S. Liu, Y. Wu, X. Xu, and K. Smith, "Burst Detection in District Metering Areas Using Deep Learning Method," *J. Water Resour. Plan. Manag.*, vol. 146, no. 6, p. 04020031, 2020.
- [46] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10265 LNCS, pp. 146–147, 2017.
- [47] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, "Improving unsupervised defect segmentation by applying structural similarity to autoencoders," *VISIGRAPP 2019 - Proc. 14th Int. Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl.*, vol. 5, pp. 372–380, 2019.
- [48] B. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2012.
- [49] "AutoGluon," <https://auto.gluon.ai/api/autogluon.task.html>.

- [50] "Scikit-learn," <https://scikit-learn.org/stable/>. .
- [51] "CatBoost," <https://catboost.ai/docs/concepts/parameter-tuning.html>. .
- [52] "LightGBM," <https://lightgbm.readthedocs.io/en/latest/Parameters.html>. .
- [53] M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, 2019.
- [54] D. van Kuppevelt, C. Meijer, F. Huber, A. van der Ploeg, S. Georgievska, and V. T. van Hees, "Mcfly: Automated deep learning on time series," *SoftwareX*, vol. 12, p. 100548, 2020.
- [55] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 1946–1956, 2019.
- [56] M. Feurer, K. Eggenberger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-Sklearn 2.0: The Next Generation," pp. 1–18, 2020.
- [57] R. S. Olson and J. H. Moore, "TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning," pp. 151–160, 2019.
- [58] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," *GECCO 2016 - Proc. 2016 Genet. Evol. Comput. Conf.*, pp. 485–492, 2016.

# 10.- Planificación temporal.

La distribución temporal de las tareas se ve detallada en el siguiente diagrama GANT (Figura 10-1).

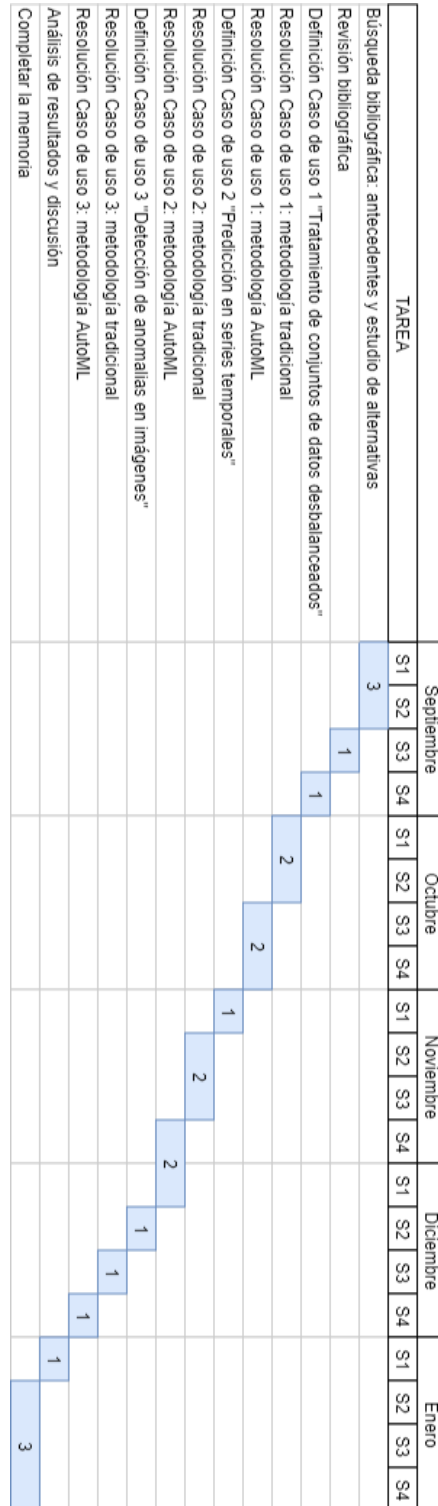


Figura 10-1.- Diagrama GANT de la distribución temporal de las tareas del proyecto.

# 11.- Glosario.

Aplicación	Categoría	Año de lanzamiento
Auger.ai	Startup	2017
Auto-sklearn	Open source (Python)	-
Auto-WEKA	Open source	2013
BigML OptiML	Open source	-
Business Insight's TIMi Suite	Startup	-
Caret	Open source (R)	-
Compellon	Startup	2011
DataRobot	Startup	2012
DMWay Analytics	Startup	2014
Enhancer	Startup	2017
Google Cloud AutoML		2018
H2O.ai	Startup	2012
Kortical	Startup	2016
MLJAR	Startup	2016
PurePredictive	Startup	2011
Tazi.ai	Startup	2015
TPOt	Open source (Python)	-
Xpanse Analytics	Startup	2014

Tabla 11.1.- Herramientas de AutoML recogidas en el blog AIMultiple (Ref. [18])



HERRAMIENTA	Auto-Keras	H2O	Auto-sklearn	AutoGluon	TPOT
<b>PLATAFORMA</b>	API Python	API Python y Java	API Python	API Python	API Python
<b>BACKEND</b>	TensorFlow, Keras y Torch (Python)	Backend propio sobre Java	Scikit-learn (Python)	Scikit-learn, XGBoost (Python)	Scikit-learn, XGBoost (Python)
<b>TIPO DE PROBLEMA</b>	Supervisado: Regresión, Clasificación binaria, Clasificación multi-clase	Supervisado: Regresión, Clasificación binaria, Clasificación multi-clase	Supervisado: Clasificación	Supervisado: Regresión, Clasificación binaria, Clasificación multiclase	Supervisado: Clasificación binaria y multiclase
<b>PIPELINE</b>	Redes neuronales profundas	Generalized Linear Models (GLM), Distributed Random Forests (DRF), XGBoost, Gradient Boosting Machines (GBM), and Deep Learning (NN).	15 modelos: General linear models (2), SVM (2), discriminant análisis (2), NN (1), nave Bayes (3), decisión tres (1), and ensembles (4)	Neural Networks, LightGBM boosted tres, CatBoost boosted tres, Random Forests, Extremely Randomized Trees, and k-Nearest Neighbors.	Basado en árboles: Naïve Bayes, Random Forest, Gradient Boosting, Linear SVC, Logistic Regression

Tabla 11.2.- Características generales de algunas herramientas de AutoML (A).

HERRAMIENTA	Auto-Keras	H2O	Auto-sklearn	AutoGluon	TPOT
<b>MÉTODO DE OPTIMIZACIÓN</b>	Optimización Bayesiana	Grid-search	Optimización Bayesiana	Optimización Bayesiana y ensamblado.	Programación genética
<b>PREPROCESADO DE DATOS</b>	Si	Si	Si	Si	Si
<b>SOPORTA GPU</b>	Si	Si (solo con modelos de XGBoost)		Si (solo para clasificación de imágenes y texto)	No
<b>GRATUITO/OPEN</b>	Si	Si (solo la versión API)	Si	Si	Si
<b>OTROS</b>	Data augmentation	Ensamblado de modelos. Escalado con clustering.	Ensamblado de modelos. Comienzo en caliente.	Ensamblado de modelos.	
<b>REF. BIBL.</b>	[55]	[26]	v.1 [26] v.2 [56]	[25]	[57], [58]

Tabla 11.3.- Características generales de algunas herramientas de AutoML (B).

