UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

ÁREA INGENIERÍA INDUSTRIAL

INGENIERÍA EN SISTEMAS INDUSTRIALES

TRABAJO FIN DE GRADO
MACHINE LEARNING APLICADO AL MANTENIMIENTO PREDICTIVO DE UN SISTEMA HIDRÁULICO

Alumno: GERARDO MARTÍNEZ ALONSO
Director: IVÁN IGLESIAS SÁNCHEZ

JUNIO 2023

AUTOR:

Gerardo Martínez Alonso

DIRECTOR DEL PROYECTO: Iván Iglesias Sánchez

FECHA: junio 2023

Listado de Documentos

I- MEMORIA
II- PLANOS DEL SISTEMA
III- PLIEGO DE CONDICIONES
IV- PRESUPUESTO

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

ÁREA INGENIERÍA INDUSTRIAL

INGENIERÍA EN SISTEMAS INDUSTRIALES

TRABAJO FIN DE GRADO
MACHINE LEARNING APLICADO AL MANTENIMIENTO PREDICTIVO DE UN SISTEMA HIDRÁULICO

MEMORIA

Alumno: GERARDO MARTÍNEZ ALONSO
Director: IVÁN IGLESIAS SÁNCHEZ

JUNIO 2023

RESUMEN

En el presente proyecto se ha abordado la elaboración de un estudio destinado al mantenimiento predictivo mediante un sistema de inteligencia artificial.

Para ello se elabora un código para el estudio y evaluación de resultados de un algoritmo de inteligencia artificial que ha sido confeccionado gracias a los datos aportados por la empresa alemana ZeMA gGmbH y el grupo de investigación comandado por Nikolai Helwig.

Se ha procedido a analizar la naturaleza del sistema hidráulico y de los datos de los que se dispone. Esto adquiere una gran importancia en la fase de preprocesado de los datos, donde conceptos como la escalabilidad de los datos, equilibrio de estos, cantidad de estados o clases que se pretende predecir o valores faltantes, entre otros, son de gran relevancia.

Cuando ya se ha dispuesto de los conjuntos de datos necesarios preprocesados, se ha procedido a la fase de aplicación de los modelos, que requiere un conocimiento sólido de programación y conocimiento de librerías de aprendizaje automático. En esta fase se ha necesitado barajar qué modelos se van a utilizar y cómo afrontar su aplicación en función del conjunto de datos preprocesado.

Por último, se ha comprobado y validado los resultados para verificar que los algoritmos diseñados son recomendables para un uso real con nuevos datos del sistema no pertenecientes al conjunto original, es decir, se ha evaluado la generalización de los modelos con el fin de seleccionar los que presenten mejor adaptabilidad a nuevos datos y predicciones más precisas.

ABSTRACT

In this project, a study has been conducted for predictive maintenance using an artificial intelligence system.
To this end, a code has been developed for the study and evaluation of the results of an artificial intelligence algorithm that has been created thanks to the data provided by the German company ZeMA gGmbH and the research group led by Nikolai Helwig.
The nature of the hydraulic system and the data available were analysed. This is significant in the data pre-processing phase, where concepts such as data scalability, data balance, number of states or classes to be predicted or missing values, among others, are of great importance.
Once the necessary pre-processed datasets are available, we proceed to the model application phase, which requires a solid knowledge of programming and knowledge of machine learning libraries. In this phase, it was necessary to consider which models to use and how to deal with their application depending on the pre-processed dataset.
Finally, the results have been checked and validated to verify that the algorithms designed are recommendable for real use with new system data not belonging to the original set, i.e., the generalisation of the models has been evaluated in order to select those with the best adaptability to new data and more accurate predictions.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Listado de abreviaturas

IA	Inteligencia artificial
gGmbh	Sociedad limitada alemana
PCA	Análisis de componentes principales
RF	Random forest
SVM	Máquinas de soporte vectorial
RBF	Kernel gaussiano
KNN	K-Nearest Neighbors
NB	Naive Bayes
MLP	Multilayer Perceptron
LBFGS	Limited-memory-Broyden-Fletcher-Goldfarb-Shanno
SGD	Gradiente de Descenso Estocástico
API	Application Programming Interface

Capítulo 1. Introducción

1.1 Justificación

Actualmente, la inteligencia artificial (en adelante, "IA") es un campo que se encuentra en un punto de inflexión y su proyección y desarrollo de cara al futuro es de carácter totalmente exponencial.
La ingeniería industrial es un área estrechamente relacionada con la IA, sobre todo en el ámbito del control de calidad y mantenimiento predictivo.

La aplicación de IA es vital para reforzar la competitividad de las empresas industriales ya que permite ahorrar fundamentalmente en mano de obra en procesos repetitivos generando unos mayores beneficios y eficiencia.

En el mantenimiento predictivo de maquinaria este ahorro de costes cobra gran importancia, concretamente en las máquinas hidráulicas. El principal problema de los sistemas hidráulicos son las fugas, por lo que debe realizarse un mantenimiento exhaustivo.
Habida cuenta de la problemática en esta materia, se ha llegado a la conclusión de que la IA se trata de una solución eficaz para dicha cuestión.

Las dos ramas principales de la IA son, el machine learning, basado en algoritmos de aprendizaje automático, y el deep learning con redes neuronales.

Para este proyecto se ha optado por aplicar machine learning, ya que se han tratado datos provenientes de sensores y atributos para elaborar un modelo de aprendizaje que consiga evaluar de manera predictiva el estado de los componentes hidráulicos con mayor incidencia de fallo, es decir, se ha realizado una categorización basada en bases de datos definidas y de carácter generalmente numérico.

A pesar de que el deep learning emplea redes neuronales que simulan la capacidad de abstracción del cerebro humano y son aplicadas a sistemas con datos de entrada más complejos como imágenes o textos, también se ha estudiado su comportamiento.

Los conjuntos de datos seleccionados corresponden a los atributos de entrada y a los estados de los componentes que se pretende predecir durante la recogida de datos. Más concretamente, 17 dataframes (bases de datos) corresponden a los atributos de entrada y el otro dataframe corresponde a los estados categorizados durante la prueba hidráulica de la toma de datos.

Los datos corresponden a un sistema hidráulico de la empresa alemana ZeMA gGmbH y han sido estudiados y evaluados desde el año 2015 por un equipo de investigación liderado por Nikolai Helwig.

Por último, cabe destacar el especial interés mostrado hacia el sector de la inteligencia artificial debido a su extenso campo de aplicabilidad y gran proyección, que permite ser integrado con los conocimientos adquiridos en el sector de la ingeniería industrial.

1.2 Objetivos

El objetivo principal del trabajo ha sido conseguir aplicar algoritmos de inteligencia artificial de machine learning de aprendizaje automático supervisado a un sistema hidráulico, para conseguir predecir las condiciones de los distintos componentes con mayor susceptibilidad de fallar.

Se ha pretendido conseguir el mayor rendimiento posible de los modelos aplicados, así como la mayor exactitud posible en las predicciones, para que el estudio pueda ser aplicado a un software de mantenimiento predictivo real que generalice bien con datos nuevos.

Para ello, se ha dispuesto de una serie de bases de datos como entrada que han sido recogidos por la empresa alemana ZeMA gGmbH y el equipo de investigación liderado por Nikolai Helwig.

Todo el trabajo se ha orientado a la aplicación y estudio de los modelos de aprendizaje automático, así como el preprocesado de los datos. Para ello se ha recurrido al compilador Spyder, que se trata de un entorno dedicado a la programación en el lenguaje Python orientado a objetos y la implementación de librerías gracias al distribuidor de código abierto Anaconda.

1.3 Estructura del Proyecto

En primera instancia, se ha descrito el sistema hidráulico y los fundamentos teóricos del mantenimiento predictivo, así como una descripción detallada de la morfología de las bases de datos empleadas y su naturaleza.

Se prosigue con una descripción teórica de los conceptos matemáticos que atañan las técnicas de preprocesado y aplicación de los modelos escogidos para llevar a cabo el estudio.

Una vez que se ha definido dicha contextualización teórica de los fundamentos de machine learning para un proyecto de clasificación, se ha instado a la integración de las fases de preprocesado y aplicación de los modelos que han sido elaborados en el pertinente código de Python. Ello incluye la fase de preprocesado, optimización de hiperparámetros y definición de los modelos junto con su entrenamiento y elaboración de predicciones de los datos de prueba.

A continuación, se han llevado a cabo varios métodos de evaluación de los modelos para comprobar el rendimiento de estos y descartar posibles faltas de eficiencia o ajuste desmedido de los datos. De esta manera, se han descartado posibles síntomas de falta de generalización de los modelos.

Por último, se han escogido los resultados óptimos mediante una discusión de resultados en los que conseguido plasmar los modelos más eficientes para cada uno de los componentes del sistema hidráulico.

Capítulo 2. Marco Teórico

2.1 Mantenimiento predictivo

El mantenimiento predictivo en el sector industrial se trata de una rama del mantenimiento que contempla la recopilación de datos e información que son recogidos mediante sensores con el fin de realizar predicciones que permitan una monitorización de un sistema.

La principal función de esta técnica es realizar predicciones de posibles fallos en un equipo o sistema industrial generando así un aumento de eficiencia en el mantenimiento de dichos equipos, generando un alto grado de anticipación ante averías o cualquier tipo de anomalía previamente contemplado.

El mantenimiento predictivo puede llegar a reducir enormemente los costes de intervención y mano de obra en el mantenimiento, así como el aumento de control y eficiencia de cualquier proceso industrial. Esto, implica un aumento en la vida útil de los sistemas industriales debido a la detección temprana de fallos o averías.

Sin embargo, se trata de un proceso complejo que requiere una alta capacitación técnica del personal encargado de la monitorización. Además, implica un alto coste de equipos y en ocasiones un aumento de los nodos computacionales debido a la alta densidad de datos que en ocasiones es requerida para un buen resultado de las predicciones.

Existen diversos tipos de mantenimiento predictivo, como la termografía mediante cámaras infrarrojas, análisis de ultrasonidos, monitoreo de vibraciones o de cualquier tipo de variable relevante para las predicciones.

En el presente proyecto se han empleado datos procedentes de sensores con distintas magnitudes y factores, se ha estudiado su posible correlación y se ha conseguido predecir el estado de los componentes principales de un sistema hidráulico.

En el sector industrial, generalmente este tipo de software de predicción suele ser subcontratado por las empresas industriales, ya que se tratan de algoritmos de inteligencia artificial cuyo desarrollo es complejo y requiere una alta capacitación y criterio.

Es de vital importancia que se identifiquen los componentes críticos del sistema que se quieren monitorizar, así como realizar un profundo estudio de las posibles variables y datos que se van a seleccionar para que el algoritmo genere las suficientes correlaciones como para generar una precisión correcta en las predicciones.

2.2 Contextualización del sistema hidráulico

El sistema hidráulico objeto de estudio, consta de un módulo primario y otro secundario, destinados a la investigación de un mantenimiento predictivo aplicado a un banco de pruebas hidráulico.

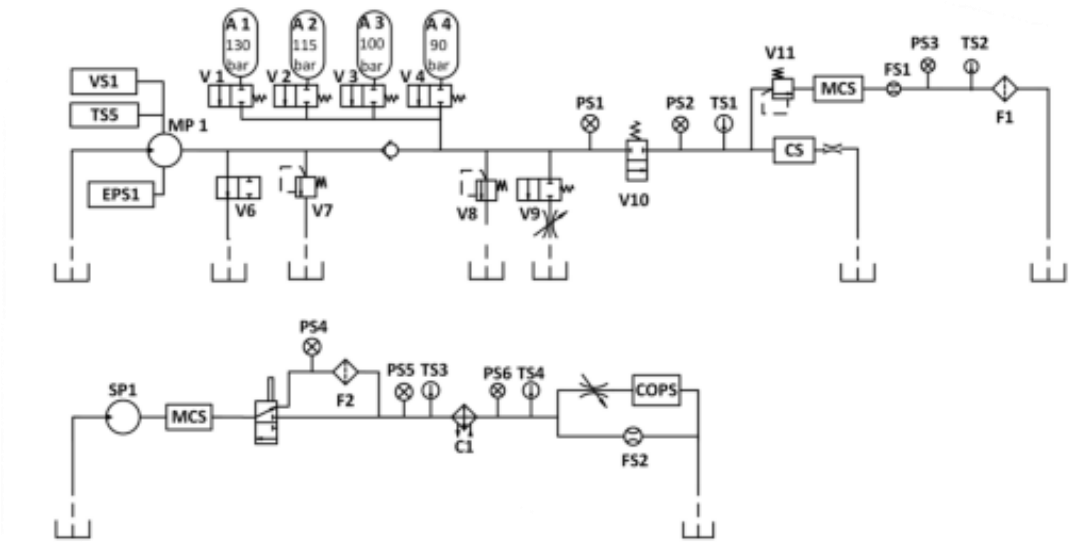


Ilustración 1: Esquema hidráulico del sistema Fuente[20]

El pionero de los estudios previos se hace llamar Nikolai Helwig, quien en colaboración con la empresa alemana ZeMA gGmbh impulsaron el estudio del sistema desde el año 2015 hasta el año 2018.

Tras la finalización de los estudios del 2018 tuvo lugar la publicación de las bases de datos, que fueron donadas por los científicos colaboradores de Helwig, M. Bastuck y T. Schneider a un repositorio de datos llamado UCI Machine Learning. Este repositorio, de uso libre y abierto, es muy recomendable para realizar modelos de inteligencia artificial de cualquier ámbito siempre y cuando se atienda a las peticiones de citación y validando la procedencia de los datos para un correcto estudio.

Los componentes críticos o susceptibles al fallo con sus respectivos posibles estados a predecir son los siguientes:

1. Estado del enfriador C1:	Cercano a fallo total Eficiencia reducida Eficiencia máxima
2. Estado de la válvula V10:	Funcionamiento óptimo Pequeño retraso Retraso considerable Cerca del fallo total
3. Fuga interna de la bomba MP1:	No hay fuga Pequeña fuga Fuga preocupante
4. Presión del acumulador A1-A4:	Presión óptima: 130 bar Presión ligeramente reducida: 115 bar Presión notablemente reducida: 100 bar Cerca del fallo total: 90 bar
5. Estabilidad del sistema:	Condiciones estables Condiciones estables posiblemente no alcanzadas

Tabla 1: Componentes del sistema hidráulico y posibles estados Fuente[2]

2.2.1 Adquisición de los datos y estructura del dataframe

Para poder elaborar un software de mantenimiento predictivo es necesario aplicar técnicas de inteligencia artificial. Por tanto, es fundamental disponer de una cantidad de datos consistente que pueda permitir estudiar las correlaciones entre ellos para conseguir predecir los estados de los componentes deseados.

En cualquier toma de datos susceptible de estudio con sistemas de inteligencia artificial es necesario definir correctamente cuales son los atributos de entrada y de salida.

Los datos de entrada han sido tomados mediante sensores y se han añadido variables virtuales o ficticias que representan una tasa o factor del sistema.

En total son 17 variables que se han considerado de entrada del algoritmo. Por tanto, los datos se estructuran en 17 bases de datos distintas, que en este caso han sido plasmados en archivos en formato txt. Para la toma de datos de cada variable se ha realizado un muestreo de cada sensor y variable virtual durante 2204 minutos, que se han considerado como ventanas de tiempo secuenciales de orden creciente correspondientes a las filas de cada dataframe. En cada ventana de tiempo se ha realizado la toma de muestras correspondiente de cada sensor y variable virtual, por tanto, es necesario el conocimiento de la frecuencia de muestreo del sensor para tener en cuenta el número de muestras, que en este caso se han interpretado como las columnas de los conjuntos de datos.

Es decir, se han considerado 17 bases de datos con 2204 filas y un número de columnas variable en función de la frecuencia del sistema o sensor de muestreo de datos.

Las características de los datos de entrada son las siguientes:

Variable	Descripción	Magnitud	Frecuencia
PS1	Presión	bar	100 Hz
PS2	Presión	bar	100 Hz
PS3	Presión	bar	100 Hz
PS4	Presión	bar	100 Hz
PS5	Presión	bar	100 Hz
PS6	Presión	bar	100 Hz
EPS1	Potencia del motor primario	W	100 Hz
FS1	Caudal primario	l/min	10 Hz
FS2	Caudal secundario	l/min	10 Hz

TS1	Temperatura	°C	1 Hz
TS2	Temperatura	°C	1 Hz
TS3	Temperatura	°C	1 Hz
TS4	Temperatura	°C	1 Hz
VS1	Vibración	mm/s	1 Hz
CE	Eficiencia enfriadora (virtual)	%	1 Hz
CP	Potencia de enfriamiento (virtual)	kW	1 Hz
SE	Factor de eficiencia	%	1 Hz

Tabla 2: Listado de sensores y características de entrada de los datos Fuente[2]

En la tabla se pueden apreciar las distintas frecuencias muestreo de cada variable. Dado que las ventanas de tiempo son de 60 segundos, es importante tener en cuenta la frecuencia de las variables ya que en este caso es de 60 tomas para las variables de 1 Hz de frecuencia, 600 tomas para las variables de 10 Hz de frecuencia y 6000 tomas para las variables de 100 Hz de frecuencia.

Por tanto, las bases de datos de entrada de las 17 variables serán de 2204 filas y de 60, 600 o 6000 columnas, en función de la frecuencia de muestreo de la variable correspondiente a cada una de las bases de datos.

Los datos de salida corresponden al estado de los 5 componentes del sistema hidráulico que se pretende predecir y su adquisición es de carácter manual.

La base de datos de las características a predecir consta de 2204 ventanas de tiempo y 5 columnas (una para cada variable o atributo que se quiere clasificar).

En resumen, la adquisición de los datos necesaria para realizar las predicciones ha sido definida durante 2204 minutos. Se han tomado datos de las entradas provenientes de los sistemas de muestreo (sensores) y se han comprobado los respectivos estados de los 5 componentes hidráulicos principales en cada ventana de tiempo correspondiente a cada minuto de la fase de adquisición.

2.3 Machine learning

Machine learning, también conocido como aprendizaje automático, se trata de una rama de la inteligencia artificial destinada a al desarrollo de algoritmos que den paso a un aprendizaje autónomo de una máquina mediante la aplicación de modelos y técnicas algorítmicas.

Esto se consigue gracias a la presencia de datos que consiguen describir una situación concreta o experiencia previa susceptible de predecir en el futuro.

Es decir, en lugar de elaborar una solución a un problema específico conociendo la raíz de este, se aplican modelos matemáticos que establecen correlaciones entre datos para conseguir una solución de manera automática sin necesidad de ahondar de manera manual en la raíz del problema.

El lenguaje de programación más eficiente para aplicar técnicas y modelos matemáticos de aprendizaje automático es Python, debido a su amplia gama de librerías y su orientación a objetos que la hace de una herramienta muy útil para trabajar con dataframes (objeto de Python que describe un conjunto de datos).

Para proceder a aplicar un modelo de machine learning se requiere definir un vector o matriz de características correspondiente a las entradas y un vector de etiquetas correspondiente a las salidas que se quieren predecir.

En este caso, la matriz de características consta de 2204 instancias que se corresponden a ventanas de tiempo y 43680 atributos o características que son fruto a la concatenación de las 17 bases de datos con sus respectivos muestreos. Es decir, cada atributo corresponde a una muestra temporal de un sensor en concreto.

También es necesario dividir previamente los datos en conjuntos de entrenamiento y de prueba. Esto se realiza normalmente adquiriendo un 80% de la matriz de características y del vector de etiqueta como conjunto de entrenamiento y el 20% restante como conjunto de prueba una vez entrenado el modelo.

Sin embargo, es importante destacar que dicho ajuste dependerá de los tipos de datos y del objeto de estudio.

Existen dos categorías principales de machine learning: supervisado y no supervisado.

2.3.1 Machine learning supervisado y no supervisado

El machine learning supervisado contempla todas las técnicas que requieren un conjunto de datos que conste de una salida o respuesta previamente conocida, es decir, una etiqueta que define los estados o características que son objeto de predicción.

Por tanto, los datos de entrada se encuentran etiquetados con sus salidas o estados que se quiere predecir. El objetivo es conseguir establecer una predicción de nuevos datos de entrada de

origen similar para conseguir determinar su etiqueta o estado con el mayor grado de precisión. Son técnicas de aprendizaje automático supervisado algoritmos de clasificación como Random Forest o algoritmos de regresión, entre otros.

Por otra parte, los algoritmos de aprendizaje automático no supervisado tienen como objeto realizar una predicción de datos no etiquetados previamente. Es decir, pretenden generar o encontrar nuevas etiquetas a partir de conjuntos de datos.

Su principal función se basa en encontrar patrones aparentemente ocultos en los datos que carecen de etiquetas necesarias para poder ser considerados como datos de valor.

Son técnicas de aprendizaje automático supervisado algoritmos de clustering, de reducción de dimensionalidad o de análisis de componentes principales (PCA).

Este tipo de algoritmos resultan de gran valor en ocasiones en las que la cantidad de datos es masiva, por lo que es conveniente seleccionar una serie de datos o características consideradas de mayor relevancia para proceder a generar nuevas etiquetas mediante técnicas de clustering.

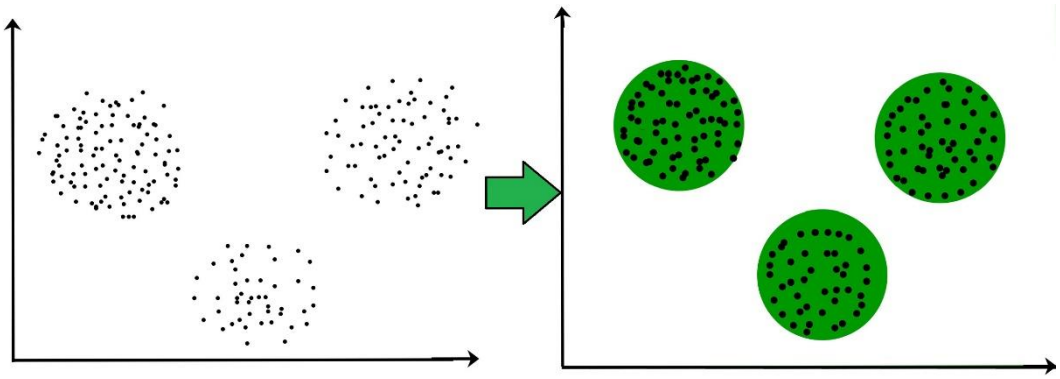


Ilustración 2: Recreación de proceso de Clustering Fuente[3]

En el caso de estudio de este proyecto se han usado técnicas de aprendizaje automático supervisado, debido a que cada ventana de tiempo del conjunto de datos ha sido previamente etiquetada de manera manual durante la recogida de datos.

Tampoco han sido necesarias técnicas de reducción de dimensionalidad o selección de características, dado que los datos han sido provistos por sensores destinados a la medición de variables de influencia directa en los estados de las etiquetas.

Es decir, por la naturaleza de los datos se ha previsto que en su mayoría tienen un impacto directo en las correlaciones de los modelos y haber considerado lo contario, podría haber sido el origen de una posible reducción o dificultad de éxito en la precisión de los modelos.

2.4 Técnicas de preprocesado empleadas

Antes de proceder a la fase de aplicación y estudio de modelos, es necesario que los datos se traten realizando las transformaciones necesarias que dependerán del tipo de técnicas empleadas y de la propia naturaleza de los datos.

El acondicionamiento de los datos en un proyecto de machine learning puede ayudar a aumentar significativamente el rendimiento y la precisión de los modelos. Los datos sin preprocesar, es decir los datos brutos provenientes de la adquisición pueden contener valores atípicos o faltantes y desbalanceados que pueden repercutir negativamente en la fase de modelado.

Los datos pueden encontrarse en escalas distintas o desbalanceados. Para solventar esto se emplean técnicas de escalado o de balanceado para poder mejorar la comprensión de los modelos aplicados ante el conjunto de datos.

Además, varias técnicas como la selección de características o reducción de dimensionalidad, pertenecientes a la rama de aprendizaje automático no supervisado, deben ser contempladas como parte del preprocesado. Sin embargo, no se ha considerado conveniente aplicar preprocesado en exceso ya que se podría haber perdido el significado real de los datos, generando falta de eficacia por parte de los modelos a la hora de generar las correlaciones necesarias.

Para el caso de estudiado se ha decidido usar técnicas de escalado y de balance de datos.

Se ha prescindido de técnicas de reducción de dimensionalidad y de selección de características, ya que se ha obtenido una gran precisión con únicamente técnicas de escalado y de balance. Cabe destacar que las bases de datos originales carecían de datos faltantes, por lo que no ha sido necesaria la reconstrucción de estos.

2.4.1 Escalado de los datos

Las 17 bases de datos correspondientes las muestras temporales de cada atributo miden variables con magnitudes distintas. La mayoría de los modelos de aprendizaje automático requieren de datos normalizados en una escala proporcional para que todas las correlaciones que se generan sean proporcionales y todos los datos se interpreten con el mismo peso. En caso de que los datos no presenten una normalización, los datos con valores muy elevados o pequeños generarían un rendimiento poco eficiente a la hora de aplicar los modelos, debido al carácter dispar de los pesos atribuidos a cada dato por parte de los modelos.

A modo de ejemplo, se propone una situación en la que existe un conjunto de datos que representa dos atributos: la edad y los ingresos anuales de una persona. Cada instancia o fila representa a una persona y las dos columnas las características o atributos. La escala de los ingresos tiene un rango entre 0 y 1 millón de euros, mientras que la escala de edad comprende valores de 0 a 100 años.

Si se procede a aplicar un modelo de aprendizaje automático a datos de esta naturaleza sin ser escalados, el peso de los datos de los ingresos sería mucho mayor. Esto es inconsistente ya que en aprendizaje automático es fundamental que todos los atributos o características posean un peso similar acorde a una escala previamente establecida.

Se suele aplicar una técnica de escalado y en función de los resultados de los modelos se tendrá en cuenta la adición de otras técnicas para mejorar o alcanzar la precisión conveniente para la aplicación deseada.

2.4.2 Balance de los datos

En muchas ocasiones balancear los datos de salida que se plasman en un vector de etiquetas es fundamental para poder asegurar un buen rendimiento del modelo.

Los datos del vector o matriz de etiquetas pueden contener clases desbalanceadas. En caso de que una clase se repita mucho más que otra, puede hacer que el modelo de aprendizaje automático sea susceptible a tener una tendencia de predicción más alta para la clase más común, haciendo que las predicciones de la clase minoritaria sean menos eficientes y generando así un desbalance en las predicciones.

Para el caso de este proyecto, esta técnica ha sido efectiva para aumentar la precisión de predicción de la estabilidad del sistema, ya que se trata de una clase binaria (0 o 1). Dicho desbalance es lógico porque en cualquier sistema industrial la clase que predomina es la de la estabilidad en la mayoría de los casos.

Se ha recurrido a esta técnica una vez entrenados los modelos para mejorar la precisión lo máximo posible. Es decir, se ha comenzado el entrenamiento de los modelos únicamente escalando los datos y en función de los resultados se ha decidido aumentar la cantidad de técnicas de preprocesado de los datos, que en este caso ha sido un reequilibrio o balanceo de las clases del conjunto de datos original.

2.4.3 Conceptos de sobreajuste y sesgo

El preprocesado es una de las fases más importantes en un proyecto de aprendizaje automático debido a que uno de los principales objetivos es balancear el sesgo y sobreajuste que son dos conceptos que juegan un papel crucial.

El sesgo se encarga de medir el grado de ajuste de un modelo a los datos haciendo referencia directa a la precisión de las predicciones. Cuando un modelo de aprendizaje automático tiene un alto sesgo, tiende a suponer de manera demasiado simplificada las predicciones de salida en base a los datos de entrada lo que hace que el modelo tenga una eficiencia demasiado reducida.

El impacto del sesgo puede influir en una sobrestimación o subestimación de la complejidad del conjunto de datos disminuyendo la precisión del modelo, lo que se conoce como subajuste.

Una de las maneras más comunes de equilibrar el sesgo de un modelo es asegurarse que la complejidad de los datos es la adecuada y en caso de no serlo se deberán implementar técnicas adicionales de preprocesado que hagan que el modelo opere con datos lo suficientemente complejos. El sesgo también se puede mejorar con técnicas de aumento de dimensionalidad de los datos, que añadan información adicional, aportando así una mayor complejidad para posibilitar un suficiente grado de correlación generado por el modelo, para que este aprenda correctamente a etiquetar o predecir las nuevas clases.

Por otro lado, el sobreajuste es una métrica que hace referencia a la capacidad del modelo para ajustarse demasiado a los datos. Esto puede desembocar en que el modelo memorice los patrones existentes en los datos en lugar de aprender sus correlaciones, afectando notablemente a la variabilidad y generalización del modelo ante datos de entrada nuevos que no pertenezcan al conjunto de prueba de la matriz de características.

Cuando se establece un modelo sobreajustado, este se ajusta demasiado al ruido o a las características irrelevantes y no consigue abstraer la información fundamental del conjunto de datos.

En resumen, el sesgo influye en una reducción notable de las métricas de precisión del sistema, mientras que una gran precisión en el conjunto de prueba puede ser un indicativo de que el modelo se encuentra sobreajustado y no generaliza correctamente, no siendo apto para ser aplicado a nuevos datos.

Por tanto, el principal objetivo es conseguir un modelo que minimice el sesgo, permitiendo un desempeño de los modelos con suficiente complejidad para establecer unas predicciones lo más precisas posibles y un sobreajuste o varianza mínimo que aporte una buena generalización ante nuevos datos, haciendo que el modelo genere una variación mínima ante nuevos datos.

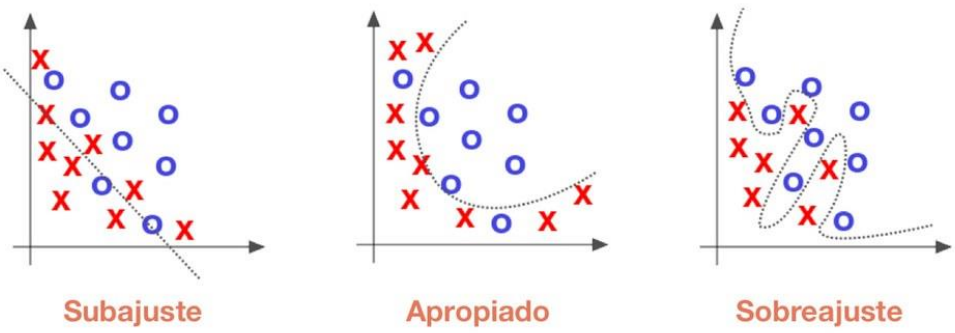


Ilustración 3: Gráficos de ejemplo de conceptos de subajuste y sobreajuste Fuente[6]

En este caso se han empleado técnicas para evaluar el sobreajuste, como la validación cruzada con bootstrapping de variables categóricas junto con técnicas de representación y visualización de resultados como las curvas de aprendizaje.

2.5 Algoritmos empleados

A continuación, se detallarán los distintos algoritmos y técnicas correspondientes a la fase del modelado.

2.5.1 Hiperparámetros

En primer lugar, es necesario definir el concepto de hiperparámetro.

Los hiperparámetros son parámetros que deben ser establecidos antes de la aplicación y entrenamiento del modelo. Cada modelo de machine learning posee sus propios hiperparámetros dado que son variables matemáticas que regulan el modo de aplicación del entrenamiento por parte del modelo.

Es importante conocer que, una vez establecidos los hiperparámetros, estos no cambian durante el proceso de entrenamiento del modelo, a diferencia de los parámetros de ajuste del modelo que van variando y cambiando mientras se entrena el modelo.

Juegan un papel fundamental en la precisión del modelo y en este proyecto se ha considerado un factor de alta incidencia. Se ha intentado optimizar los hiperparámetros de manera exhaustiva mediante técnicas como la búsqueda de cuadrícula.

2.5.2 Optimización de Hiperparámetros

La optimización de los parámetros es una fase fundamental en un proyecto de machine learning si se quieren conseguir precisiones altas y un rendimiento óptimo de los modelos.

Para llevarlo a cabo, se ha recurrido a la búsqueda de cuadrícula que se ha implementado mediante la función GridSearch de la librería de Scikit-learn.

GridSearch permite encontrar la combinación adecuada de hiperparámetros para el modelo que se pretende aplicar.

En primera instancia, es necesario definir los hiperparámetros del modelo que se vaya a aplicar en un diccionario, que es un objeto de Python que permite definir varias características. Cada modelo cuenta con varios tipos de hiperparámetros y cada uno de esto se plasma en el diccionario creado que deberá contener una lista que recoja los posibles valores que puede tomar el hiperparámetro correspondiente.

GridSearch usa un sistema de validación cruzada con k-fold, que consiste en realizar varios "pliegues" o divisiones a modo de cuadrículas que deberán ser establecidos previamente de la ejecución del algoritmo. En este proyecto se ha decidido emplear un número consistente equivalente a 5 pliegues del conjunto de datos para conseguir la mejor combinación de hiperparámetros.

De esta manera, una vez que el algoritmo ha completado los pliegues, tiene lugar la validación cruzada de estos, es decir, en cada pliegue se prueba cada una de las posibles combinaciones de hiperparámetros definidas en el diccionario que toma como entrada. Las combinaciones de hiperparámetros de cada pliegue van siendo validadas con las combinaciones de los otros pliegues, para así realizar un buen contraste de los resultados por votación y así determinar la lista de hiperparámetros óptima, que es devuelta por la función GridSearch como un vector o lista de Python.

2.5.3 Clasificación binaria y multiclase

Los modelos que han sido aplicados en este proyecto son de clasificación multiclase y binaria para la última clase (estabilidad del sistema).

La clasificación binaria se usa para predecir una salida que puede adoptar dos estados o clases mientras que la clasificación multiclase se aplica a predicciones en las que el vector de etiquetas contiene 3 posibles estados o clases.

2.5.4 Random Forest (RF)

El modelo Random Forest o de bosques aleatorios es un modelo de aprendizaje automático muy útil en problemas de clasificación que se basa en la combinación de numerosos árboles de decisión.

Se conoce como árbol de decisión a una estructura matemática de datos en la que a partir de una entrada se establecen una serie de preguntas abstractas con las que se llega a una conclusión o predicción. Consta de varios nodos internos que sirven para representar una pregunta acerca de los datos de entrada y las hojas del árbol vendrían a representar una salida. El árbol se construye atendiendo a las mejores características que subdividan al conjunto de datos de entrada en otros subconjuntos. El modelo establece que cada subconjunto sea más pequeño que el conjunto original y que la varianza de los datos sea lo más mínima posible.

Para conseguir llegar a las mejores predicciones, un árbol de decisión comienza a establecer preguntas en orden descendente desde la raíz del árbol hasta dar con una hoja que sea la correcta, es decir, la salida con un grado de predicción más alta.

A modo de ejemplo, se tiene un conjunto de datos con dos posibles clases a predecir.

El conjunto de datos de la imagen siguiente tiene dos características o atributos. Una vez estructurado el conjunto de datos, el árbol comienza a ramificarse realizando subdivisiones más pequeñas siguiendo un criterio de clasificación que en este caso es una pregunta para verificar si la salida es rojo o verde.

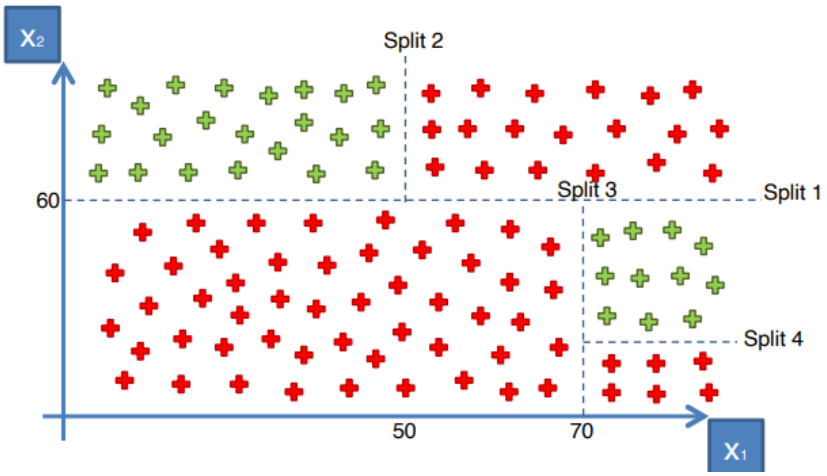


Ilustración 4: Subdivisiones en espacio bidimensional de un árbol de decisión Fuente[7]

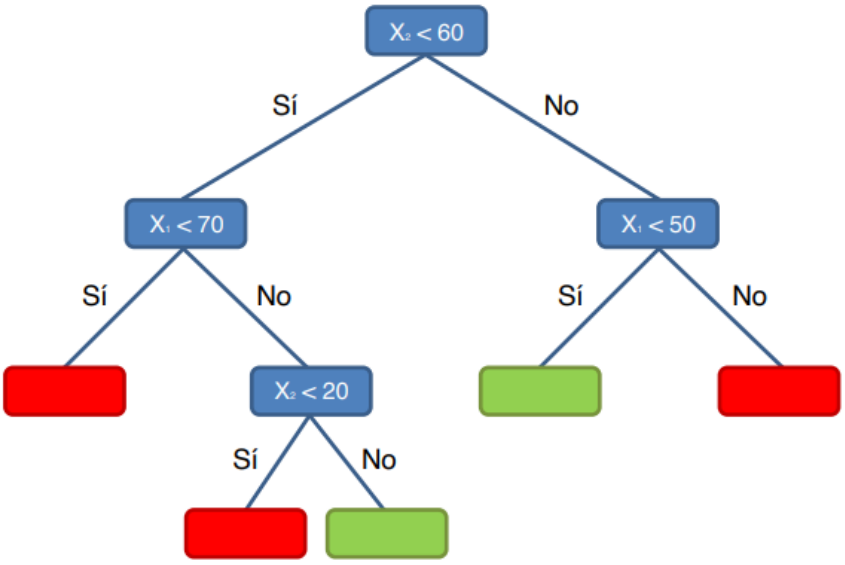


Ilustración 5: Árbol de decisión Fuente[7]

Se puede apreciar como a medida que se realizan divisiones el árbol comienza a ramificarse estableciendo preguntas de clasificación binaria. En caso de que la clasificación fuera multiclase el proceso sería similar. Cabe destacar que la dimensión de la proyección del conjunto de datos depende del número de atributos, es decir, en caso de tener 3 atributos la representación sería tridimensional.

Una vez definido el concepto de árbol de decisión, un bosque aleatorio no es más que una combinación de modelos de árboles de decisión que ocurren de manera concurrente o paralela. Se trata de un modelo de aprendizaje en conjunto que selecciona por votación aleatoria el conjunto de árboles de decisión con mayor rendimiento.

Las fases para un modelo de Random Forest son las siguientes:

1. Primeramente, se selecciona un número aleatorio de K puntos proveniente de los datos del conjunto de entrenamiento.
2. Se construye un árbol de decisión para esos K puntos del conjunto de entrenamiento.
3. Se elige un número de árboles como hiperparámetro y se establecen los árboles de decisión acorde al hiperparámetro seleccionado.
4. Una vez alcanzado el número de árboles seleccionados de manera aleatoria, se establece el bosque aleatorio donde uno de esos árboles realiza su predicción y tras una votación de los resultados más sólidos se establece una conclusión con la clasificación más votada para los datos de entrada.

Los hiperparámetros más relevantes para un modelo de Random Forest son:

- Max_depth: regula la profundidad de cada árbol del bosque aleatorio. Esto, puede incidir en la prevención del sobreajuste del conjunto de entrenamiento. A medida que se aumenta la profundidad, los árboles tienden a ajustarse en mayor medida a los datos lo cual puede repercutir negativamente en la generalización del modelo ante nuevos datos. Por tanto, es importante establecer un valor de profundidad que encuentre un equilibrio alcanzando la mayor precisión posible para un bajo sobreajuste.

- Max_features: limita el máximo número de características que tiene en cuenta el modelo a la hora de establecer una división en cada nodo de cada uno de los árboles generados. Un valor bajo conlleva una simplificación mayor de los árboles pudiendo derivar en un menor grado de precisión y buena generalización, mientras que un valor alto implica lo contrario.

- Min_samples_leaf: establece cual es el valor mínimo de muestras necesarias para que un árbol genere una hoja. Durante el entrenamiento del modelo, si un nodo de un árbol contiene un número menor o igual al establecido como hiperparámetro, el nodo dará lugar a una hoja. Este valor incide en el sobreajuste. Un número menor de muestras relevantes puede implicar una mejor generalización del modelo.

- Min_samples_split: la dinámica es similar al número mínimo de muestras para generar una hoja, pero en este caso, para dar lugar a un nodo en el árbol. No se generarán más nodos en un árbol si número prestablecido como hiperparámetro es superior al número de muestras del nodo en el proceso de entrenamiento. El hecho de establecer un número mínimo de muestras para generar un nodo implica que el modelo tiene en cuenta un número insuficiente de muestras permitiendo controlar el sobreajuste con mayor exactitud.

- N_estimators: hace referencia al número de árboles que se ensamblan en el bosque aleatorio, afectando tanto a su complejidad, como a la capacidad de generalización y precisión del modelo. Tiene un gran impacto en el tiempo de entrenamiento del modelo.

2.5.5 Support Vector Machine (SVM)

Existen varios tipos de SVM dependiendo del núcleo o kernel que se configure como hiperparámetro antes de entrenar el modelo. Los más comunes son el gaussiano (rbf) o lineal. En este caso se ha establecido el SVM lineal ya que es el hiperparámetro que se ha determinado en el proceso de optimización de hiperparámetros para el conjunto de datos de estudio.

SVM lineal trata de buscar un hiperplano que genere una separación de las clases do etiquetas correspondientes a los datos de entrada. Matemáticamente, el hiperplano es una superficie lineal que tiene como objetivo establecer un espacio de separación máximo entre las regiones de separación de las clases. A esa distancia se le conoce como margen máximo.

Para ello, el modelo traza dos vectores de soporte que surgen desde los dos puntos del espacio creado más cercanos al margen del hiperplano creado. De esta manera, se haya un hiperplano óptimo que permite clasificar los nuevos datos de entrada o de prueba en función de la zona del hiperplano en la que se ubiquen.

En el siguiente ejemplo, se aprecian el modelo gráfico de una máquina de soporte vectorial lineal con datos de entrada o atributos bidimensionales para una clasificación binaria.

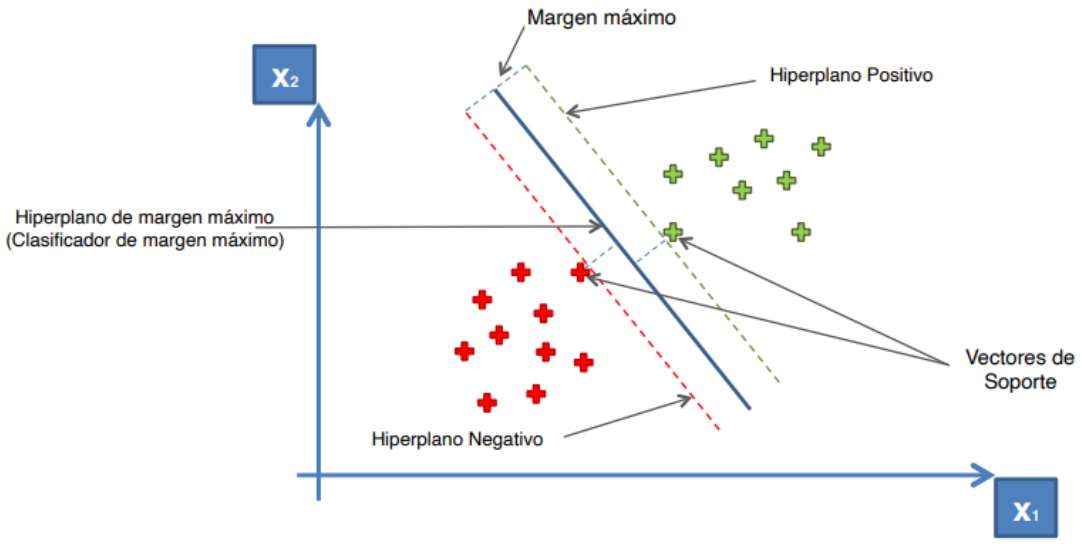


Ilustración 6: Componentes de una máquina de soporte vectorial lineal Fuente [7]

Los hiperparámetros más destacables de un modelo SVM son:

- C: es un coeficiente que regula la relación que existe entre el margen máximo y el número mínimo de errores en la clasificación. Es decir, es la tolerancia que se establece para el modelo e incide en el número de puntos o características del modelo que no han sido correctamente clasificados. Cuando el coeficiente C alcanza valores bajos el margen máximo es más amplio asumiendo un mayor error en las clasificaciones, mientras que a medida que aumenta C se reduce la tolerancia minimizando el error en las clasificaciones. Sin embargo, esto aumenta el grado de complejidad del modelo lo cual puede generar una peor generalización derivada de un sobreajuste de los datos.

- Gamma: indica la magnitud de influencia de una sola instancia de entrenamiento a la hora de formar el límite de decisión del modelo. Un valor alto de este hiperparámetro implica una mayor influencia de los puntos más distantes en el margen de decisión, mientras que uno más bajo implica una menor influencia de los puntos más lejanos. Por tanto, gamma regula la flexibilidad del modelo.

- Kernel: es el tipo de función que se aplica al modelo en un espacio de mayor dimensión y es responsable del mapeo de los datos de entrada en un espacio previamente definido. Dicho espacio puede ser aplicado de manera lineal o no lineal. Los más comunes son el kernel lineal o el kernel gaussiano (rbf o radial basis function). Para este proyecto, mediante la optimización de hiperparámetros por búsqueda de cuadrícula, se ha determinado que el kernel más eficiente es el lineal. Un kernel lineal, es un espacio que se recrea mediante el producto escalar de los vectores de características de la matriz de características, siendo para cada par de vectores de características:

$$K(X, y) = X^T \cdot y$$

  Siendo X e y los vectores de características de un dato concreto que corresponde a una coordenada del espacio creado por el modelo SVM lineal.
  En este caso, el kernel lineal ha sido establecido como el más eficiente debido a que los vectores de características del conjunto de datos son linealmente separables y se pueden representar como funciones lineales para recrear un espacio con un hiperplano lineal.

  Además, una de las mayores ventajas de usar un kernel lineal es su elevada eficiencia en términos computacionales, requiriendo así de un menor tiempo de entrenamiento del modelo comparado con kernel no lineales como el gaussiano que funcionan mejor con datos no linealmente separables.

2.5.6 K-Nearest Neighbors (KNN)

El algoritmo KNN es un modelo de aprendizaje automático supervisado que se puede emplear para la clasificación. El modelo pretende encontrar un número de K vecinos que se encuentren más cerca de los datos de entrada de una muestra seleccionada. KNN trata de identificar muestras de datos del conjunto de entrenamiento que se encuentren más cercanas a las muestras de los datos de entrada. Dicha identificación suele seguir una métrica de distancia (hiperparámetro) previamente establecida que suele ser la distancia euclidiana. Posteriormente, se ajusta por votación la clase mayoritaria de la muestra de entrada.

Los hiperparámetros más determinantes para que el modelo tenga un buen rendimiento, son: el número de K vecinos y el tipo de métrica, que establece los vecinos más cercanos hasta llegar a K vecinos.

En el siguiente caso se representa el algoritmo gráficamente de un conjunto de datos de entrada de dos atributos de un espacio bidimensional que pretende realizar una clasificación KNN binaria. Los hiperparámetros establecidos son K vecinos = 5 y métrica de distancia euclidiana.

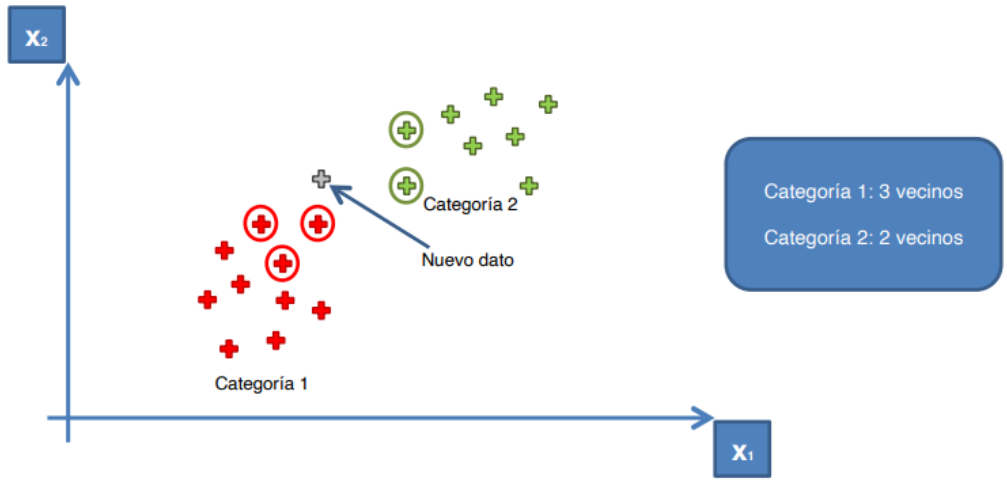


Ilustración 7: Proceso de estimación de vecinos del algoritmo KNN Fuente[7]

El nuevo dato tiene 3 K vecinos cercanos de la clase roja y es una cantidad de K vecinos mayoritaria del total de K vecinos = 5, por tanto, el nuevo dato pertenecerá a la clase roja.

Los hiperparámetros más relevantes para el caso de estudio son los siguientes:

- Metric: especifica el tipo de métrica empleada en el espacio recreado para determinar los vecinos más cercanos. Las más usadas son la euclídea o manhattan, siendo la euclídea la raíz cuadrada de la suma de las diferencias de las coordenadas en determinada dimensión elevada al cuadrado y manhattan la distancia absoluta entre dos puntos en determinada dimensión.

Ilustración 8: Distancia euclídea y de Manhattan Fuente[12]

- N_neighbors: es el valor de K vecinos que establece la decisión de clasificación. Cuando se define un valor elevado de vecinos, puede hacer que la sensibilidad de clasificación sea más alta, sin embargo, el algoritmo también será más sensible al ruido, lo cual puede llegar a ser perjudicial debido a la influencia de variables que pueden llegar a ser no relevantes o no tener el mismo peso que otras.

- Weights: establece la relevancia relativa de los vecinos más cercanos previamente definidos a la hora de establecer las métricas. Hay dos posibilidades que son las más frecuentes y útiles para este hiperparámetro: “uniform” o “distance”. En “uniform” todos los vecinos desempeñan un mismo peso o importancia eliminando el término de importancia relativa de los vecinos, mientras que en “distance” los vecinos ponderan una mayor o menor importancia en función de la distancia de estos al nuevo dato de entrada en el espacio generado por el algoritmo.

2.5.7 Naive Bayes

El modelo de Naive Bayes se basa en el Teorema de Bayes, que se define como la actualización de una hipótesis a medida que nueva información es incorporada a dicha hipótesis, es decir, permite calcular la probabilidad de una hipótesis o evento que viene condicionada por otro evento. La expresión que define el teorema de Bayes es la siguiente:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Donde, *P(A|B)* es la probabilidad de *A* condicionada del suceso *B*, siendo *P(B|A)* el caso análogo y *P(A)* y *P(B)* las probabilidades de los eventos *A* y *B* sin ser condicionadas.

El modelo aplica la probabilidad condicional para determinar si una muestra de los datos de entrada pertenece a una muestra específica perteneciente a una de las clases existentes.

Se calcula la probabilidad de cada clase de salida para cada muestra de entrenamiento y una vez realizado el barrido de probabilidades de cada muestra, se determina la mayor de las probabilidades de la clase asociada a los datos de entrada.

Es decir, se calcula la probabilidad de cada clase existente en el vector de etiquetas y se calculan las probabilidades de cada atributo de la matriz de características para cada clase del conjunto de entrenamiento.

Después, se aplica el teorema de Bayes para clasificar los datos nuevos de prueba calculando la probabilidad de cada posible clase condicionada a los nuevos datos de entrada. Una vez calculadas las probabilidades condicionadas de cada clase asociada a su instancia, se determina la clase con mayor probabilidad condicionada para determinar la predicción. Cabe destacar que este modelo no dispone de hiperparámetros por lo que no es necesario aplicar el algoritmo de búsqueda de cuadrícula o GridSearch.

2.5.8 Multilayer Perceptron (MLP)

El algoritmo MLP es un tipo de algoritmo perteneciente a una rama del machine learning llamada deep learning y supone el algoritmo más representativo de dicho campo.

El deep learning o aprendizaje profundo es una rama del aprendizaje automático que se basa en la superposición de varias capas o neuronas artificiales que se encuentran entre la entrada del conjunto de datos y la salida. Dicho sistema matemático se denomina red neuronal.

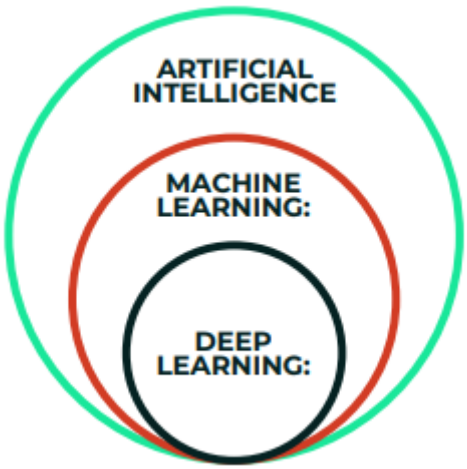


Ilustración 9: Mapa conceptual de las ramas de la inteligencia artificial Fuente[7]

Una red neuronal consta de dos procesos o fases diferenciales. En primer lugar, el entrenamiento, seguido de la inferencia. En la fase entrenamiento la red neuronal se ajusta sus operaciones a los datos de entrenamiento previamente etiquetados.

Es decir, realiza operaciones entre cada capa hasta lograr una similitud con los datos de entrenamiento etiquetados. Una vez alcanzada dicha similitud, tiene lugar el proceso de inferencia en el que se realiza la clasificación o la predicción de los datos de nueva instancia.

Las fases de funcionamiento son las siguientes:

1. Capa de entrada: Es la capa inicial de la red neuronal y almacena los datos de entrada. Cada dato proporcionado a la capa de entrada se transforma en un vector de valores que consiga plasmar la información proporcionada por el dato inicial.

2. Capas ocultas: se denomina capas ocultas a las capas intermedias entre la de entrada y de salida. Cada capa oculta se usa como recipiente de información proveniente de las capas anteriores y como transformador de información generando una salida que es enviada a la siguiente capa. Se ejecuta una función de activación que consigue abstraer la información de entrada como una combinación lineal que consigue generar una salida no lineal. Una capa oculta está constituida por numerosas neuronas artificiales.

3. Sesgos y pesos: las neuronas de una red neuronal constan de una serie de pesos y un sesgo. Un peso hace referencia a la intensidad de conexión existente entre una neurona y las distintas entradas de la capa previa.
   Por otro lado, el sesgo es un valor concreto que se añade en una capa oculta a las entradas previamente al proceso de ejecución de las combinaciones lineales de la función de activación.

4. Función de pérdida: métrica de una red neuronal que cuantifica el grado de error cometido entre los datos de salida y las etiquetas que se pretendía predecir en el conjunto de prueba.

5. Capa de salida: es la respuesta del modelo y su representación es vectorial.

6. Retropropagación: es un método de autoajuste de la red neuronal que tiene lugar de manera secuencial y continua durante el proceso de entrenamiento del modelo. Trata de optimizar a lo largo del proceso tanto los pesos como los sesgos de todas las neuronas para poder optimizar la red neuronal y convertir la función de pérdida lo más desfavorable posible, para así conseguir un mejor rendimiento y menor error en las predicciones de la red neuronal.

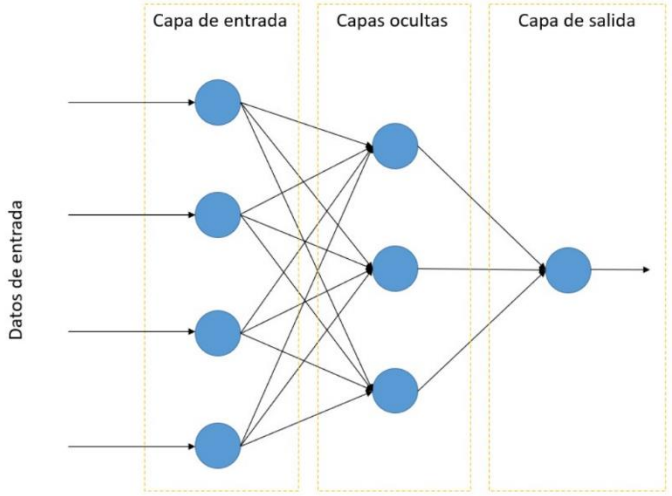


Ilustración 10: Sintaxis y morfología de una red neuronal Fuente[4]

Las redes neuronales se suelen aplicar en proyectos con conjuntos de datos masivos y de naturaleza compleja como imágenes, texto o voz. En este proyecto se ha contemplado su uso en la fase de estudio debido a que la densidad de datos es muy alta (aproximadamente 96 millones de datos).

Los hiperparámetros de este modelo han sido previamente definidos con los ajustes por defecto de la librería Scikit-learn, ya que un algoritmo de optimización de hiperparámetros para redes neuronales requiere un alto coste computacional teniendo en cuenta los con los medios disponibles para la realización del proyecto.

Los hiperparámetros que se han definido en el algoritmo, son los siguientes:

- Hidden_layer_sizes: determina el número de capas ocultas que conforman la arquitectura de la red neuronal y el número de neuronas correspondiente a cada capa.

- Activation: es la función de activación de las neuronas de cada capa oculta y determina la salida no lineal de la neurona en función de la combinación lineal de los datos de entrada. Existen varios tipos de funciones de activación: la función sigmoide o "logistic", la función de activación de la tangente hiperbólica o "tanh" o la función lineal rectificada o "relu".

  En este caso se ha usado la función de activación "relu", ya que es la función por defecto de la librería Scikit-learn. Se trata de una función no lineal definida por la siguiente expresión:

  $$relu(x) = \max(0, x)$$

  Donde *x* son los datos de entrada de la neurona y *max (0, x)* devuelve el valor de entrada positivo sin alterar y los valores de entrada negativos con el valor de *0*. Esto implica una mayor sensibilidad del modelo ante valores no lineales. Las funciones de activación son de carácter no lineal ya que, en muchos casos, los datos pueden no ser lineales. Para ello se emplean funciones de activación que permiten plasmar las correlaciones de los datos de entrada, ya sean lineales o no lineales.

  Es decir, la relu permite plasmar las correlaciones de los datos de entrada de manera no lineal, para que no tenga que ser contemplada la linealidad o no linealidad de los datos de entrada y asegurar una correcta representación de las relaciones entre los datos de entrada.

  Por tanto, la principal razón por la que devuelve los datos positivos sin alterar es que asegura que sigan presentes las correlaciones de entrada. Al mismo tiempo, incluye la no linealidad mediante la inclusión de valores equivalentes a 0 en lugar de negativos.

- Solver: es un hiperparámetro que define el tipo de algoritmo de optimización de que se encarga de realizar el ajuste de la red neuronal durante la fase de entrenamiento. Existen varios tipos: LBFGS, SGD o gradiente de descenso estocástico o el optimizador Adam. Este último es el que ha sido empleado para el caso de estudio y es de los más comunes a la hora de aplicar redes neuronales.

  El optimizador Adam basa su funcionamiento en la combinación de momentos de primer y segundo orden con el fin de conseguir una adaptación del grado de aprendizaje del modelo mientras se entrena el modelo de la red neuronal.

  Durante el proceso de optimización tiene lugar un proceso iterativo donde cada ciclo equivale a una época y consta de las siguientes fases:

En primer lugar, Adam genera un vector de momento de primer orden y otro vector de momento de segundo orden, con la misma dimensión que los parámetros de la red neuronal de cada neurona. Es notable que la inicialización de los vectores se define de tal manera que las componentes de estos sean ceros.

A continuación, durante el proceso de entrenamiento tiene lugar la fase de retropropagación, en la que se establecen los gradientes de los pesos de los parámetros de la red neuronal en relación con la función de pérdida. Se denomina retropropagación debido se comienza calculando el gradiente de la capa de salida de la red, retrocediendo hasta llegar a la capa inicial aplicando derivadas encadenadas.

Una vez obtenidos los gradientes, se actualiza el vector de momento de primer orden mediante la media exponencial del gradiente actual calculado. Al mismo tiempo, el vector de momentos de segundo orden se actualiza de manera similar implementando la media exponencial de los cuadrados de los gradientes calculados. Las siguientes expresiones corresponden a los vectores de momentos de primer y segundo orden respectivamente:

$$m(n+1) = \beta 1 \cdot m(n) + (1+\beta 1) \cdot \Delta W$$

$$v(n+1) = \beta 2 \cdot v(n) + (1+\beta 2) \cdot \Delta W^2$$

Donde:

m y *v* son los vectores de momentos de primer y segundo orden, *n* la iteración correspondiente, $\beta 1$ y $\beta 2$ son dos coeficientes, 0.9 y 0.99 en la mayoría de los casos respectivamente y ΔW es el gradiente de los pesos correspondientes a los parámetros de las neuronas.

Por último, se determina una actualización de los pesos de los parámetros internos de las neuronas mediante la siguiente expresión:

$$W(n+1) = W(n) - \frac{m(n+1) \cdot \mu}{\sqrt{v(n+1)} + \delta}$$

Donde:

W(n+1), es la actualización de los pesos de los parámetros de las neuronas en la época o iteración n+1, y W(n), corresponde a los pesos de los parámetros de la época anterior.
El coeficiente μ es la tasa de aprendizaje o "learning rate" que regula la magnitud de cada época y δ es una constante de valor muy pequeño cuya principal función es evitar que el denominador del cociente sea nulo.
m(n+1) y *v(n+1)*, equivalen a los vectores de momentos de primer y segundo orden actualizados, correspondientes a la época actual del ciclo de entrenamiento de optimización.

- Alpha: se conoce como el coeficiente de regulación destinado al control de las penalizaciones aplicadas por el algoritmo en los pesos de las neuronas.
  Su función es ayudar a prevenir posibles sobreajustes del algoritmo. Los valores elevados de Alpha generan una mayor generalización en el algoritmo, mientras que un valor bajo implica que los pesos de los parámetros de las neuronas disponen de una mayor libertad, haciendo que el modelo pueda llegar a desempeñar un mayor sobreajuste.

- Learning_rate: es la tasa de aprendizaje empleada en el algoritmo Adam de optimización. Establece el tamaño y extensión de cada época o iteración, de tal manera que valores altos de dicha tasa permiten un entrenamiento más veloz, pero puede generar una falta de estabilidad del algoritmo. En caso de pretender una alta precisión, son deseables valores más bajos de la tasa de aprendizaje, lo que conllevaría una mayor lentitud de procesamiento por parte del modelo.

- Max_iter: limita el número de épocas o iteraciones del proceso de optimización a lo largo del proceso de entrenamiento de la red neuronal.

2.6 Evaluación del rendimiento de modelos de clasificación

La fase de evaluación se ha enfocado en el uso de técnicas que aseguren una buena precisión del modelo y que consigan descartar un posible sobreajuste, para así poder verificar que la generalización del modelo es consistente de cara al uso de nuevos datos no pertenecientes a los conjuntos de entrenamiento y prueba originales.

2.6.1 Matriz de confusión

La matriz de confusión es una de las principales representaciones gráficas del rendimiento de un modelo de clasificación y sirve para representar las métricas positivas y negativas de los resultados predichos. Es una representación matricial de los resultados de clasificación de las predicciones de cada clase contrastados con los valores reales, es decir, los que deberían ser.

El concepto de positivo y negativo es abstracto y depende del enfoque y naturaleza del sistema al que pertenecen los datos. En el caso de estudio se habla de positivos cuando existe un fallo y negativos cuando el sistema no falla o la incidencia no es crítica.

Por tanto, un verdadero positivo indica que el sistema falla cuando va a fallar y un verdadero negativo que el sistema funciona de forma correcta cuando realmente es así.

Por otra parte, se considera un falso positivo cuando el sistema no falla y se predice un funcionamiento incorrecto y falso negativo cuando el sistema funciona incorrectamente y se predice un funcionamiento correcto.

En este caso se deberá contemplar la existencia de falsos negativos, que será el error que más se debe tener en cuenta en el caso de detección de fallos para un sistema hidráulico, ya que implicaría no detectar un fallo cuando este está presente.

Ilustración 11: Componentes de una matriz de confusión Fuente[8]

2.6.2 Exactitud, precisión, función F1, recall y desviación estándar.

La exactitud es una métrica que determina la cantidad de instancias que se han predicho correctamente respecto del conjunto total de instancias.

La precisión es una métrica de evaluación del rendimiento de un modelo que hace referencia al número de casos o instancias positivas que han sido predichas correctamente por el modelo. Mide la capacidad para evitar clasificar como positivas ventanas o instancias que eran negativas en el conjunto real. Se define como la tasa de verdaderos positivos entre la suma de los falsos positivos y verdaderos positivos.

La función recall indica la tasa de verdaderos positivos que han sido predichos correctamente del total de casos positivos reales, mientras que la función F1 es la media armónica de las tasas de precisión y de recall.

En otras palabras, la función F1 mide el equilibrio existente entre la precisión y la función recall. Se dice que el modelo es consistente cuando la precisión y el recall tienen valores elevados y parecidos.

La gran diferencia entre la precisión y el recall es que la precisión se encarga de medir si el modelo clasifica correctamente las instancias positivas y el recall mide el grado de certeza que tiene el modelo para identificar las instancias positivas.

Por último, la desviación estándar indica la variabilidad del modelo en caso de este ser ejecutado múltiples veces. Para el caso de estudio siempre será muy baja, debido a la solidez, complejidad y cantidad de los datos, por lo que no se ha decidido evaluar dicha métrica. En su lugar se usan técnicas que aportan más valor, como la validación cruzada o curvas de aprendizaje, donde se puede apreciar la desviación estándar de manera visual.

Únicamente se ha evaluado la exactitud, la función recall (herramienta útil para minimizar los falsos negativos, que son el resultado más perjudicial en el problema a tratar) y la función F1 que relaciona el recall y la precisión. La evaluación de la desviación estándar se ha apreciado gracias al área que envuelve las líneas que componen la curva de aprendizaje.

2.6.3 Validación cruzada con k-folds y Booststrapping

La validación cruzada con k-folds y bootstrapping es una técnica que se ha usado para comprobar si el modelo tiene una buena tasa de generalización, es decir, que no está sobreajustado y que tiene una buena capacidad para predecir datos nuevos no procedentes del conjunto de entrenamiento y de prueba.

La validación cruzada con k-folds consiste en un proceso de evaluación que segmenta los datos originales en subconjuntos, que son los llamados “folds”. Cada subconjunto tendrá un tamaño similar. Una vez realizada la división, el modelo se entrena k veces, siendo k un parámetro ajustable. En cada iteración de entrenamiento se utiliza un subconjunto diferente de datos como conjunto de prueba para pasar a determinar las métricas correspondientes de cada iteración. Al mismo tiempo, los subconjuntos que no se seleccionan como prueba son empleados como un solo conjunto de entrenamiento en cada iteración. Dicha técnica se emplea tanto para entrenar el modelo como para evaluarlo, ya que el contraste de estos resultados con el entrenamiento y métricas obtenidas en el modelo sin validación cruzada permite una evaluación global de la solidez del modelo y de su comportamiento ante el sobreajuste.

En este caso, para aumentar la solidez de la evaluación se ha implementado una variante adicional a la validación cruzada convencional que se denomina validación cruzada con k-folds con bootstrapping. El bootstrapping permite realizar una estimación de la incertidumbre del modelo aplicado, así como una generación de intervalos de confianza. Esto se traduce en que los subconjuntos de entrenamiento que genera la validación cruzada ya no son fracciones del conjunto de datos original, sino que el bootstrapping genera *n* subconjuntos llamados bootstraps por medio de técnicas de muestreo aleatorio por reemplazo. Es decir, se seleccionan de forma aleatoria observaciones procedentes del conjunto de datos original para después ser introducidos a los subconjuntos de entrenamiento. Por último, esta observación es devuelta al conjunto de datos original. Esto ocurre hasta el momento en que la cantidad de reemplazos realizados equivale a la cantidad de muestras o instancias del conjunto de datos original.

De esta manera se evalúa el modelo con validación cruzada con k-folds mediante múltiples combinaciones de datos de entrenamiento pertenecientes a subconjuntos de datos generados por la técnica de bootstrapping, consiguiendo una considerable cantidad de métricas para distintas posibles situaciones que simulan nuevos datos a los que podría enfrentarse el modelo.

Después se ha ejecutado la media de las métricas de cada iteración con sus predicciones asociadas a cada subconjunto de prueba y se devuelven los resultados. En este caso, se han definido 100 bootstraps (100 subconjuntos de datos de entrenamiento con datos aleatorios del dataframe original generados por reemplazo) y se han validado las métricas de exactitud, función recall, F1 y desviación estándar.

Cabe destacar que, para establecer una lista final de resultados de las etiquetas predichas en cada iteración, se emplea una función de votación para variables categóricas que hace que cada etiqueta de la lista de predicciones final sea la mayoritaria en cada una de las iteraciones en dicha instancia.

Los resultados obtenidos en la validación cruzada deben ser contrastados con los resultados obtenidos en las métricas de las predicciones de los conjuntos de entrenamiento y prueba originales. Si los resultados de la validación cruzada y los de la aplicación del modelo al conjunto original son similares, se podrá determinar que lo más posible es que no exista sobreajuste apreciable, por tanto, la generalización y variabilidad del modelo ante nuevos datos de entrada puede que sea eficaz.

Sin embargo, en un proyecto de machine learning se deben usar varias técnicas de validación para asegurar que no existe sobreajuste, por lo que se ha recurrido adicionalmente a la evaluación de los resultados mediante curvas de aprendizaje.

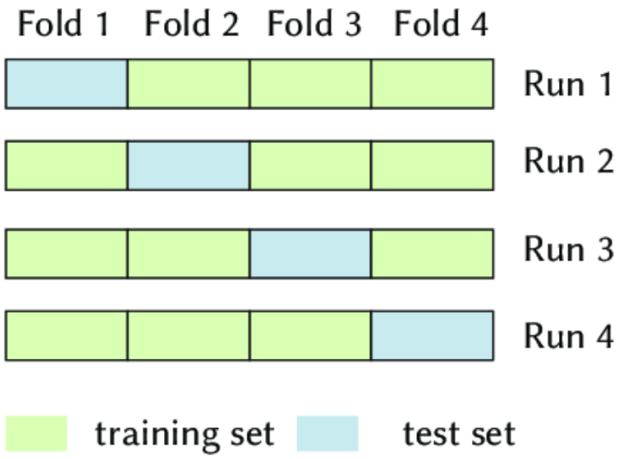


Ilustración 12: Esquema conceptual de las iteraciones de la validación cruzada con k-folds Fuente[10]

2.6.4 Curva de Aprendizaje

El último método de evaluación que se ha considerado determinante a la hora de validar la solidez de los resultados es la curva de aprendizaje. Las curvas de aprendizaje establecen la forma en la que un modelo es capaz de generalizar y operar de manera correcta ante nuevos datos no pertenecientes al conjunto de datos original.

Su métrica se basa en mostrar como la exactitud de un modelo evoluciona a medida que se aumentan la cantidad de instancias o muestras durante el proceso de entrenamiento del modelo.

Por tanto, representan la exactitud del modelo en función del número de instancias y se caracteriza por la presencia de dos líneas o componentes principales:

- Línea de entrenamiento (azul): representa el grado de exactitud en función del número de instancias durante el proceso de entrenamiento sin aplicar validación cruzada. Es decir, el modelo se aplica al conjunto de entrenamiento original sin métodos de generación de subconjuntos de entrenamiento nuevos, evaluándose únicamente con los datos originales.
- Línea de validación cruzada (verde): representa el grado de exactitud en función del número de instancias durante el proceso de entrenamiento aplicando validación cruzada. Esta línea indica como el modelo es capaz de generalizar ante nuevos datos mediante subconjuntos de entrenamiento iterativos que simulan situaciones distintas al entrenamiento, únicamente con los datos originales.

El principal fin de este método de evaluación es comprobar el grado de sobreajuste del modelo, así como determinar qué cantidad de instancias de entrenamiento generan un mayor rendimiento o exactitud del modelo.

En la siguiente ilustración, se pueden apreciar dos gráficas. La de la izquierda, se trata de un modelo que no consigue generalizar correctamente ante nuevos datos y por tanto tiene sobreajuste. Esto se debe a que la línea de validación cruzada (verde) no consigue converger con la línea de entrenamiento, por lo que el modelo fracasa a la hora de generalizar los nuevos datos resultados de la validación cruzada.

En cambio, en la imagen de la derecha se puede observar como las líneas de entrenamiento y validación consiguen converger, verificando así que el modelo generaliza con éxito ante nuevos datos descartando el sobreajuste.

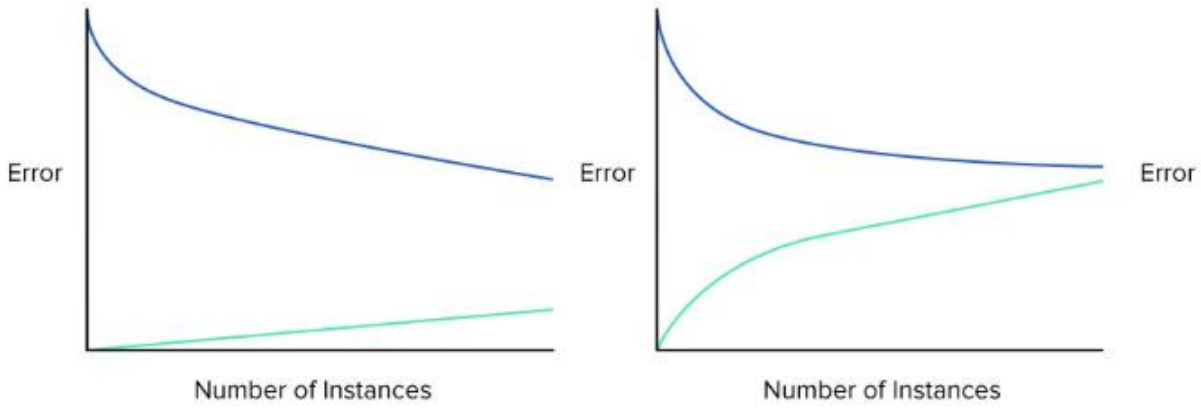


Ilustración 13: Ejemplos de curva de aprendizaje Fuente[9]

Capítulo 3. Desarrollo del estudio

3.1 Entorno y librerías utilizadas

El lenguaje de programación más usado para llevar a cabo proyectos de inteligencia artificial y ciencia de datos es Python.

Python es un lenguaje orientado a objetos y es altamente eficiente debido a su gran capacidad para manejo de datos, existiendo una amplia gama de librerías que lo hacen el medio ideal a la hora de aplicar modelos de machine learning y deep learning.

Además, Python tiene una sintaxis sencilla e intuitiva, lo que posibilita una comprensión y escritura de código ideal si se tienen conocimientos en programación previos como C/C++. También permite la inclusión de librerías de otros lenguajes, para hacer aún más eficiente la capacidad de cómputo a la hora de ejecutar tareas intensivas con una alta densidad de datos.

El entorno de programación de Python que se ha seleccionado para la realización del código es Spyder que ha sido provisto por el distribuidor de Python Anaconda, que se encara al desarrollo de aplicaciones y software de inteligencia artificial y análisis de datos.

Anaconda permite una personalización y actualización de una amplia gama de paquetes y entornos como Spyder o Jupyter, así como un extenso abanico de librerías y herramientas que permiten la instalación directa de estas.

Existe una extensa lista de librerías: NumPy, Pandas, Polars, Scikit-Learn, TensorFlow, PyTorch o Matplotlib, entre otros. Todas estas librerías están diseñadas para aplicaciones de aprendizaje automático, manipulación de datos, visualización de datos, redes neuronales o procesamiento de imágenes o lenguaje natural.

En el caso de estudio se han usado las siguientes librerías:

- Scikit-Learn: es una de las librerías más usadas y conocidas para llevar a cabo proyectos de machine learning. Se ha recurrido a dicha librería para poder aplicar los distintos modelos de clasificación, validación cruzada, preprocesamiento de datos, visualización de métricas y curvas de aprendizaje.

- Pandas: librería ampliamente usada en el sector de la ciencia de datos que se ha usado con el fin de estructurar las distintas bases de datos en dataframes, que son el tipo de objetos bidimensionales que permiten representar grandes conjuntos de datos y permite etiquetar con índices en cada fila y cada columna del objeto. También es indispensable para la lectura de bases de datos en distintos formatos, que, en este caso, son archivos en formato txt. Por último, hace posible la concatenación y manipulación de características de un dataframe y se integra muy bien con Scikit-Learn y demás librerías principales destinadas a la ciencia de datos.

- NumPy: es una librería fundamental a la hora de realizar operaciones y cálculos numéricos con dataframes multidimensionales y desempeña una alta eficiencia a la hora de ser integrada con la mayoría de las librerías, destacando Pandas y Scikit-Learn.

- Polars: se trata de una librería que ha sido desarrollada muy recientemente, que realiza prácticamente las mismas funciones que pandas, pero destaca en su velocidad de procesamiento en la lectura de datos. Polars, ha sido fundamental en este proyecto debido a que la densidad de datos es muy alta (casi 100 millones de datos), por lo que la velocidad de computación en las lecturas de datos, sobre todo con los recursos disponibles, ha sido notablemente reducida gracias a Polars.

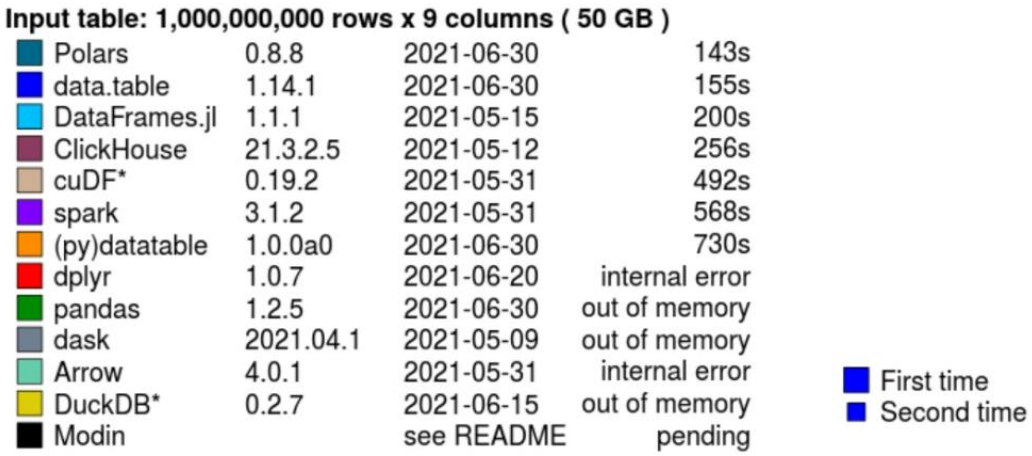


Ilustración 14: Comparación de los tiempos de ejecución de librerías de manipulación y lectura de conjuntos de datos Fuente[11]

- Matplotlib: es una herramienta muy potente y personalizable de visualización de datos en Python. Ofrece la posibilidad de crear gráficos en dos y tres dimensiones de carácter estático o interactivo. Es indispensable si se pretende visualizar los resultados gráficamente mediante la creación de matrices de confusión y las curvas de aprendizaje.

- Seaborn: es una librería que se ha utilizado como complemento de Matplotlib para representar las matrices de confusión, ya que permite crear mapas de calor o cuadrículas con valores asociados a los resultados de los modelos de Scikit-Learn.

- Collections: permite la creación del objeto “Counter” cuyo uso principal es establecer recuentos de otros objetos asociados a unas etiquetas determinadas. Es realmente útil si se quiere establecer las votaciones de las etiquetas mayoritarias en cada instancia de las iteraciones de la validación cruzada o visualizar el recuento de clases del vector de etiquetas de cada variable.

3.2 Fase de preprocesado

En la fase de preprocesado se ha incluido: la lectura de las 18 bases de datos en formato texto, su manipulación, generación de archivos csv, escalado y balanceado de datos.

Se ha dispuesto de 17 bases de datos que corresponden a las características y representan las muestras temporales de un tipo de sensor, ya sea físico o virtual. Además, se dispone de una base de datos con 5 columnas que corresponde a las 5 variables con sus respectivas etiquetas que se pretende predecir.

Para comenzar el preprocesado, ha sido necesario tener en cuenta cómo se iban a aplicar los modelos. En este caso, se ha aplicado un modelo independiente a cada una de las 5 variables, es decir, cada variable ha sido tratada con su propio algoritmo para una obtención de resultados más precisa, concluyente y funcional.

Por tanto, se ha contado con 5 dataframes en los que las primeras 43.680 columnas han constituido las muestras temporales de los sensores y la última columna ha correspondido a cada una de las variables respectivamente en cada dataframe.

Para ello, se han leído los 17 dataframes provenientes de los archivos de texto y se han sido concatenados en un dataframe que ha sido determinado como la matriz de características.

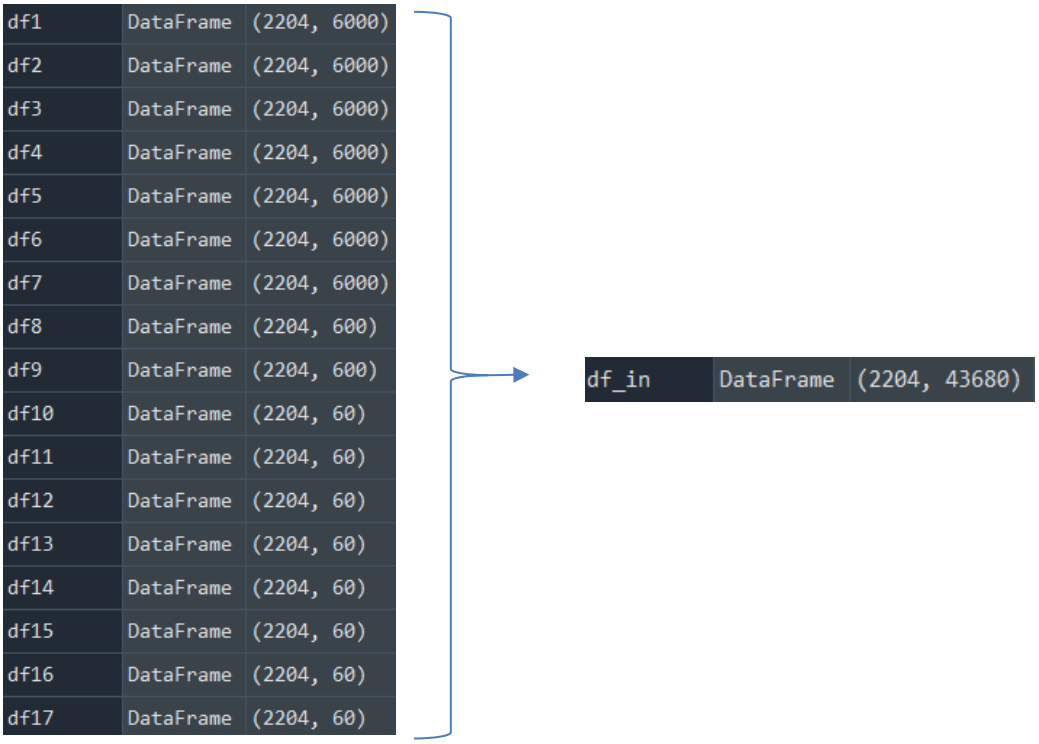


Ilustración 15: Concatenación de bases de datos para la formación de la matriz de características

Una vez que se ha formado la matriz de características es necesario realizar un escalado de los datos en dicha matriz para un correcto rendimiento e interpretación de los datos de los modelos. Para ello se ha implementado la función StandardScaler de la librería Scikit-Learn que crea un objeto "scaler" que sirve para aplicar una transformación al objeto dataframe de la matriz de características. Esta transformación genera un dataframe cuyos valores tienen una media de 0 y una desviación estándar de 1, haciendo que no haya datos dispares y la escala de los datos sea proporcional.

A continuación, se ha procedido a concatenar la matriz de características normalizada con cada uno de los vectores de etiquetas de la última base de datos que ha sido leída y segmentada en 5 columnas o dataframes de 1 columna y 2204 instancias cada uno.

De esta manera, se han obtenido 5 dataframes. Cada uno de los dataframes está formado por una matriz de características de 2204 instancias y 43680 columnas, concatenados a un vector de etiquetas de 1 columna y 2204 instancias correspondientes a las clases o estados de cada uno de los componentes que se pretende monitorizar.

df1	dataframe.frame.DataFrame	(2204, 43681)	DataFrame object of polars.dataframe.frame
df2	dataframe.frame.DataFrame	(2204, 43681)	DataFrame object of polars.dataframe.frame
df3	dataframe.frame.DataFrame	(2204, 43681)	DataFrame object of polars.dataframe.frame
df4	dataframe.frame.DataFrame	(2204, 43681)	DataFrame object of polars.dataframe.frame
df5	dataframe.frame.DataFrame	(2204, 43681)	DataFrame object of polars.dataframe.frame

Ilustración 16: Dataframes para cada una de las variables

También se ha realizado el conteo de las clases de los vectores de etiquetas de cada variable para detectar posibles desequilibrios de clases, pero no estaban notablemente desbalanceadas, ya que al ser una clasificación multiclase muchas clases se han visto compensadas por otras.

Sin embargo, en la última variable (estabilidad del sistema), al tratarse de una clasificación binaria, el desbalanceo presente en una de las dos clases originaba una falta de precisión a la hora de aplicar los modelos, por lo que al aplicar un reequilibrio de los datos en el vector de etiquetas se lograron unos resultados más precisos.

El reequilibrio se ha logrado con técnicas de sobremuestreo en las que se seleccionan aleatoriamente instancias del dataframe original de la clase minoritaria para crear otro dataframe en el que se han añadido dichas instancias aleatorias hasta igualar la cantidad de etiquetas de ambas clases.

Con el fin de no tener que realizar de nuevo el preprocesado, a la hora de aplicar cada uno de los modelos a todas las variables se ha decidido generar 5 archivos csv correspondientes a cada dataframe preprocesado.

Una vez que se leído el correspondiente dataframe de estudio, se ha necesitado dividir el dataframe en la matriz de características (X) y el vector de etiquetas (y), para hacer una división de conjuntos de entrenamiento y prueba mediante la biblioteca Scikit-Learn con la función "train_test_split ()".

Esta función, recibe como entrada 2 objetos que son la matriz de características y el vector de etiquetas y genera 4 objetos:

- X_train: conjunto de la matriz de características de entrenamiento
- y_train: conjunto del vector de etiquetas de entrenamiento
- X_test: conjunto de prueba de la matriz de características que se tratan de los datos que recibe el modelo para establecer las predicciones.
- y_test: porción del vector de etiquetas de prueba con el que se han contrastado las predicciones para obtener las métricas de exactitud, recall, F1 y matriz de confusión.

La proporción en la que se ha realizado la división es: 80% del dataframe original como entrenamiento y el 20% restante ha sido destinado al conjunto de prueba.

3.3 Fase de aplicación de los modelos

Una vez preprocesado todo el conjunto de datos y estructurado la matriz de características y vector de etiquetas, se ha procedido a la fase de entrenamiento de los modelos para realizar una visión preliminar de su rendimiento y poder tomar las consideraciones de la fase posterior de evaluación con un mayor nivel de solidez y eficiencia.

Para ello se han aplicado los modelos: RandomForest, SVM Lineal, KNN, Naive Bayes y Redes Neuronales (MLP). La estrategia de la primera fase de estudio ha sido conformada por las fases de optimización de hiperparámetros, definición y entrenamiento de los modelos y establecimiento de las predicciones de los conjuntos de prueba del vector de etiquetas de cada variable.

Para ambos procesos se ha recurrido a la biblioteca Scikit-learn.

Primero, se ha inicializado el modelo como un objeto llamando a la librería del modelo correspondiente de Scikit-learn con los hiperparámetros optimizados en la búsqueda de cuadrícula como argumento de formación del objeto.

A continuación, se ha realizado una transformación del objeto con los conjuntos de entrenamiento. Esta es la fase del código en la que se realiza la ejecución del proceso de entrenamiento.

Una vez terminado el proceso de entrenamiento, se ha empleado la función “predica” que recibe como argumento el conjunto de prueba y devuelve un vector de predicciones que se trata del vector que ha sido contrastado con el conjunto de prueba del vector de etiquetas para posteriormente calcular las métricas de exactitud, recall y F1.

3.3.1 Rango de hiperparámetros del GridSearch

En primera instancia, antes del proceso de definición y entrenamiento de los modelos para establecer las predicciones, se ha procedido a la programación del algoritmo de optimización de parámetros mediante la técnica de búsqueda de cuadrícula.

Para conseguir dicho propósito se ha definido un objeto para cada modelo de tipo diccionario que contiene todos los hiperparámetros de cada respectivo modelo.

A cada parámetro del diccionario se le ha introducido un vector que contiene el rango de hiperparámetros con los que se han realizado las iteraciones y combinaciones encargadas de determinar qué parámetro de la lista de cada hiperparámetro debe ser usado, con el fin de alcanzar una mayor eficiencia y rendimiento del modelo.

Cabe destacar que la optimización de hiperparámetros se ha aplicado a los modelos de RF, SVM y KNN. Esto se debe a que el algoritmo de Naive Bayes carece de hiperparámetros y las Redes Neuronales requieren un alto tiempo de ejecución del algoritmo y sería necesario un hardware mucho más potente que el disponible para la realización del proyecto.

Así mismo, la optimización de hiperparámetros en redes neuronales es más común en procesamiento de imágenes o lenguaje natural, debido a que la naturaleza de los datos es mucho más abstracta que en el caso de estudio.

A lo sumo, los resultados obtenidos con los hiperparámetros definidos por defecto en la librería Scikit-learn han resultado notablemente exitosos, ya que han sido establecidos por los desarrolladores de la librería de manera equilibrada. En caso de que los resultados hubieran sido muy desfavorables, hubiera sido conveniente recurrir a servicios computacionales en la nube como Amazon Web Services, pero en este caso no se ha requerido dicho proceso.

Modelo	Hiperparámetros	Rango
RF	max_depth	[5, 10, 15]
	max_features	['auto', 'sqrt']
	min_samples_leaf	[1, 2, 4]
	min_samples_split	[2, 5, 10]
	n_estimators	[50, 100, 150]
SVM	C	[0.1, 1, 10]
	gamma	['scale', 'auto']
	kernel	['linear', 'rbf']
KNN	metric	['euclidean', 'manhattan']
	n_neighbors	[1, 2, 3, 4, 5, 6, 7, 9]
	weights	['uniform', 'distance']
Naive bayes	No procede	No procede
Redes neuronales	hidden_layer_sizes	100
	activation	relu
	solver	adam
	alpha	0.0001
	learning_rate	constant
	max_iter	200

Tabla 3: Hiperparámetros empleados en el código y su rango de optimización

El rango de hiperparámetros para la búsqueda de cuadrícula requiere tiempos de ejecución muy altos incluso con no demasiados valores para cada hiperparámetro, sin embargo, el rango que se ha establecido ha sido suficiente y se ha diseñado entorno a los valores por defecto que establece Scikit-learn.

Una vez terminado el proceso de ejecución del algoritmo, se ha impreso un vector en la ventana de comandos con la combinación de hiperparámetros que mejor se ajusta a los datos de entrenamiento para poder proseguir con la definición y entrenamiento de los modelos.

3.3.2 Estudio del estado de la enfriadora

La primera variable es el estado de la enfriadora cuyas clases o posibles estados son los siguientes:

	Descripción
Clase 0	Cercano a fallo total
Clase 1	Eficiencia máxima
Clase 2	Eficiencia reducida

Tabla 4: Clases de la Enfriadora

Una vez ejecutado el proceso de optimización de hiperparámetros y el entrenamiento de los modelos y predicciones, se han obtenido los siguientes resultados resumidos en la siguiente tabla:

Modelo	Hiperparámetros	Métricas
Random Forest	max_depth=15, max_features='auto', min_samples_leaf=2, min_samples_split=10, n_estimators=50	Recall: 1.0 Puntuación F1: 1.0 Exactitud: 1.0000
SVM Lineal	C = 0.1, gamma = 'scale', kernel = 'linear'	Recall: 0.9977 Puntuación F1: 0.9977 Exactitud: 0.9977
KNN	metric= 'manhattan', n_neighbors= 3, weights='distance'	Recall: 0.9954 Puntuación F1: 0.9954 Exactitud: 0.9955

Naive Bayes	NO	Recall: 0.9523 Puntuación F1: 0.9523 Exactitud: 0.9524
Redes neuronales	hidden_layer_sizes=100 activation='relu' solver='adam' alpha=0.0001 learning_rate='constant' max_iter=200	Recall: 1.0 Puntuación F1: 1.0 Exactitud: 1.0000

Tabla 5: Resultados del GridSearch y métricas del conjunto de prueba

3.3.3 Estudio del estado de la válvula

	Descripción
Clase 0	Funcionamiento óptimo
Clase 1	Cerca del fallo total
Clase 2	Retraso considerable
Clase 3	Pequeño retraso

Tabla 6: Clases del Estado de la válvula

Modelo	Hiperparámetros	Métricas
Random Forest	max_depth=5, max_features='auto', n_estimators=50	Recall: 1.0 Puntuación F1: 1.0 Exactitud: 1.0000
SVM Lineal	C = 0.1, gamma = 'scale', kernel = 'linear'	Recall: 1.0 Puntuación F1: 1.0

		Exactitud: 1.0000
KNN	metric='euclidean', n_neighbors=3, weights='distance'	Recall: 0.9070 Puntuación F1: 0.9063 Exactitud: 0.9070
Naive Bayes	NO	Recall: 0.8321 Puntuación F1: 0.8266 Exactitud: 0.8322
Redes neuronales	hidden_layer_sizes=100 activation='relu' solver='adam' alpha=0.0001 learning_rate='constant' max_iter=200	Recall: 0.9727 Puntuación F1: 0.9727 Exactitud: 0.9728

Tabla 7: Resultados del GridSearch y métricas del conjunto de prueba del Estado de la válvula

3.3.4 Estudio de la fuga interna de la bomba

	Descripción
Clase 0	No hay fuga
Clase 1	Pequeña fuga
Clase 2	Fuga preocupante

Tabla 8: Clases de la Fuga interna de la bomba

Modelo	Hiperparámetros	Métricas
Random Forest	max_depth=15, max_features='auto', n_estimators=150	Recall: 0.9909 Puntuación F1: 0.9909 Exactitud: 0.9909
SVM Lineal	C=0.5, kernel='linear', gamma= 'scale'	Recall: 0.9954 Puntuación F1: 0.9954 Exactitud: 0.9955
KNN	metric='manhattan', n_neighbors=9	Recall: 0.9886 Puntuación F1: 0.9886 Exactitud: 0.9887
Naive Bayes	NO	Recall: 0.5873 Puntuación F1: 0.5277 Exactitud: 0.5873
Redes neuronales	hidden_layer_sizes=100 activation='relu' solver='adam' alpha=0.0001 learning_rate='constant' max_iter=200	Recall: 0.9229 Puntuación F1: 0.9238 Exactitud: 0.9229

Tabla 9: Resultados del GridSearch y métricas del conjunto de prueba de la Fuga interna de la bomba

3.3.5 Estudio de la presión del acumulador hidráulico

	Descripción
Clase 0	Presión notablemente reducida 100 bar
Clase 1	Presión ligeramente reducida 115 bar
Clase 2	Presión óptima 130 bar
Clase 3	Cerca del fallo total 90 bar

Tabla 10: Clases de la Presión del acumulador hidráulico

Modelo	Hiperparámetros	Métricas
Random Forest	max_depth=10, max_features='auto', min_samples_split=5, n_estimators=50	Recall: 0.9863 Puntuación F1: 0.9863 Exactitud: 0.9864
SVM Lineal	C = 0.1, gamma = 'scale', kernel = 'linear'	Recall: 0.9727 Puntuación F1: 0.9726 Exactitud: 0.9728
KNN	metric='manhattan', n_neighbors=3, weights='distance'	Recall: 0.9591 Puntuación F1: 0.9591 Exactitud: 0.9592
Naive Bayes	NO	Recall: 0.4376 Puntuación F1: 0.3948 Exactitud: 0.4376
Redes neuronales	hidden_layer_sizes=100 activation=’relu’ solver=’adam’ alpha=0.0001	Recall: 0.9070 Puntuación F1: 0.9084

	learning_rate=’constant’ max_iter=200	Exactitud: 0.9070

Tabla 11: Resultados del GridSearch y métricas del conjunto de prueba de la Presión del acumulador

3.3.6 Estudio de la estabilidad del sistema

	Descripción
Clase 0	Condiciones estables
Clase 1	Condiciones estables no alcanzadas

Tabla 12: Clases de la Estabilidad del sistema

Modelo	Hiperparámetros	Métricas
Random Forest	'max_depth': 40, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100	Recall: 0.9862 Puntuación F1: 0.9862 Exactitud: 0.9862
SVM Lineal	C = 0.1, gamma = 'scale', kernel = 'linear'	Recall: 0.9810 Puntuación F1: 0.9810 Exactitud: 0.9810
KNN	metric='manhattan', n_neighbors=4, weights='distance'	Recall: 0.9904 Puntuación F1: 0.9904 Exactitud: 0.9904

Naive Bayes	NO	Recall: 0.8637 Puntuación F1: 0.8636 Exactitud: 0.8638
Redes neuronales	hidden_layer_sizes=100 activation='relu' solver='adam' alpha=0.0001 learning_rate='constant' max_iter=200	Recall: 0.9517 Puntuación F1: 0.9517 Exactitud: 0.9517

Tabla 13: Resultados del GridSearch y métricas del conjunto de prueba

Capítulo 4. Evaluación de los resultados

Una vez que se han entrenado los modelos y se han obtenido las métricas preliminares correspondientes a las predicciones de los conjuntos de prueba de cada variable, es fundamental validar los resultados.

Las validaciones y visualizaciones de los resultados de evaluación del rendimiento de los modelos se han considerado de manera analítica para determinar qué modelo era el más eficiente para cada variable.

Por tanto, la última fase ha consistido en realizar de nuevo el entrenamiento de todos los modelos mediante la técnica de validación cruzada con k-folds y bootstrapping para contrastar los resultados con las curvas de aprendizaje.

También, ha sido conveniente estudiar las matrices de confusión de cada modelo en cada variable para comprobar las clases más propensas a ser predichas con un error más elevado. Al mismo tiempo, ha permitido evaluar el comportamiento de los falsos negativos y falsos positivos.

Es decir, tienen más incidencia fallos en las predicciones cuando la clase que no se ha predicho correctamente es desfavorable para el sistema hidráulico y constituye un fallo detectado como un comportamiento normal.

La incidencia de estos casos puede llegar a ser crítica, ya que si hay fallos que se están clasificando como funcionamiento correcto del sistema se puede llegar a generar pérdidas económicas en la empresa y falta de control en la supervisión.

Debido a esto se ha intentado incluir en el criterio de selección y evaluación de los modelos la repartición de los resultados en las matrices de confusión, ya que constituye una parte fundamental en un proyecto de mantenimiento predictivo con aprendizaje automático.

Es importante destacar que el coste computacional del proceso de validación cruzada es considerablemente alto. Por ello es importante tener en cuenta sistemas de computación en la nube en caso de que los tiempos de ejecución sean demasiado elevados.

En este proyecto, se ha conseguido una buena optimización de los tiempos de ejecución incluyendo comandos que han permitido optimizar los recursos del hardware disponible, permitiendo un aprovechamiento completo de los núcleos del procesaros y memoria RAM, por lo que no es necesario recurrir a procesos de computación en la nube como Amazon Web Services (AWS) o Microsoft Azure.

La tarjeta gráfica no se ha empleado como recurso debido a que el rendimiento de una GPU es requerido en caso de estudios de machine learning con imágenes y procesamiento de lenguaje natural.

A continuación, se incluyen los resultados de la validación cruzada del conjunto de entrenamiento contrastados con las métricas obtenidas para el conjunto de prueba sin validación cruzada para cada una de las variables, junto con sus respectivas matrices de confusión y curvas de aprendizaje de cada uno de los modelos.

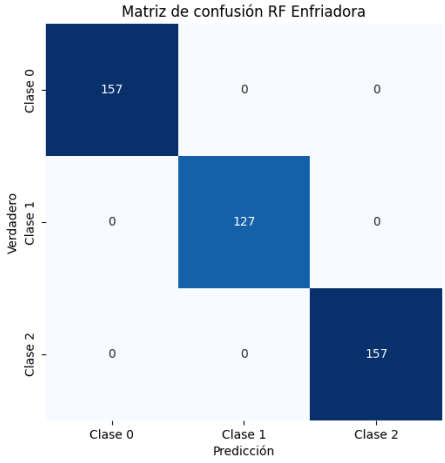
4.1 Evaluación de resultados del enfriador

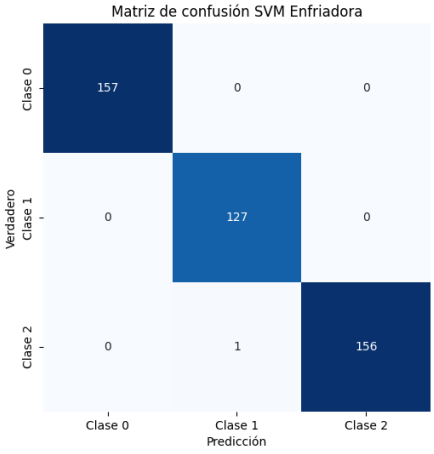
Modelo	Métricas validación cruzada	Métricas conjunto de prueba
Random Forest	Recall medio = 1.0 Puntuación F1 media = 1.0 Exactitud media = 1.0	Recall: 1.0 Puntuación F1: 1.0 Exactitud: 1.0000
SVM Lineal	Recall medio = 0.9982 Puntuación F1 media = 0.9982 Exactitud media = 0.9982	Recall: 0.9977 Puntuación F1: 0.9977 Exactitud: 0.9977
KNN	Recall medio = 0.9965 Puntuación F1 media = 0.9965 Exactitud media = 0.9965	Recall: 0.9954 Puntuación F1: 0.9954 Exactitud: 0.9955
Naive Bayes	Recall medio = 0.9137 Puntuación F1 media = 0.9140 Exactitud media = 0.9137	Recall: 0.9523 Puntuación F1: 0.9523 Exactitud: 0.9524
Redes neuronales	Recall medio = 0.9977 Puntuación F1 media = 0.9977 Exactitud media = 0.9977	Recall: 1.0 Puntuación F1: 1.0 Exactitud: 1.0000

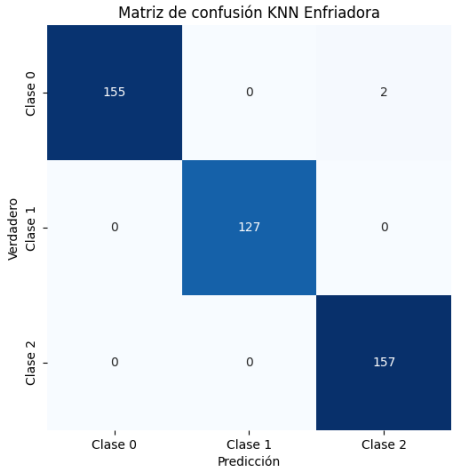
Tabla 14: Comparación de resultados de las métricas de validación y prueba de la Enfriadora

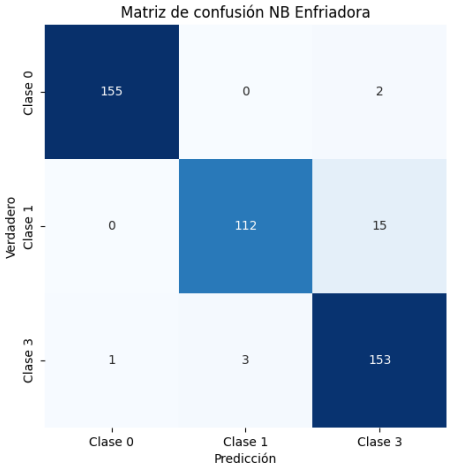
Se puede apreciar como los resultados de la validación cruzada han sido parecidos en la mayoría de los casos a las métricas del conjunto de prueba por lo que esto puede considerarse un indicativo importante, pero no concluyente, de que no hay un sobreajuste considerable.

En segunda instancia se han evaluado las matrices de confusión, dejando a la vista que la clasificación de clases del conjunto de prueba es muy favorable para esta variable en prácticamente todos los modelos, siendo RF y las redes neuronales modelos que consiguen clasificar correctamente todas las instancias del conjunto de prueba.









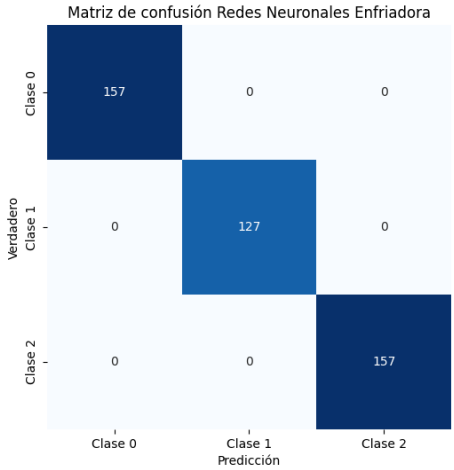


Ilustración 17: Matrices de confusión de los modelos aplicados a la Enfriadora

Por último, para poder concluir qué modelo es el óptimo para clasificar los estados del enfriador, se han visualizado las curvas de aprendizaje de visualizan las curvas de aprendizaje para verificar una correcta convergencia de las líneas de entrenamiento y validación cruzada.

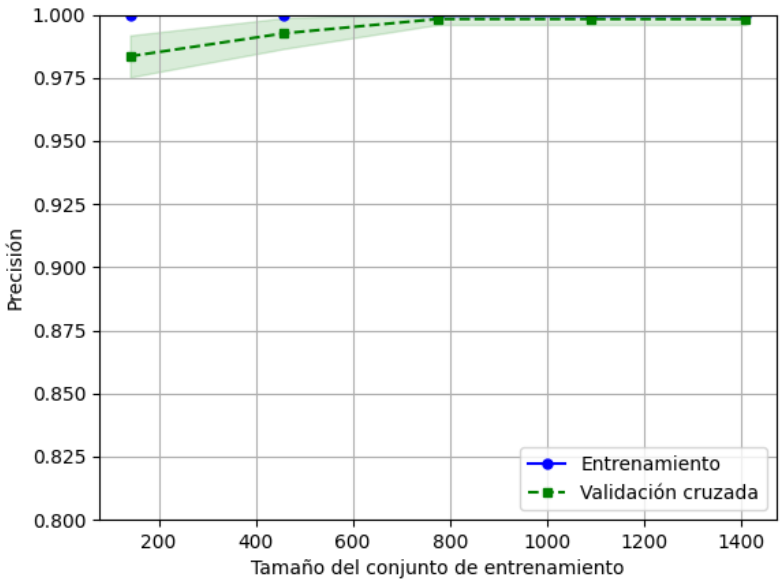


Ilustración 18: Curva de aprendizaje RF Enfriadora

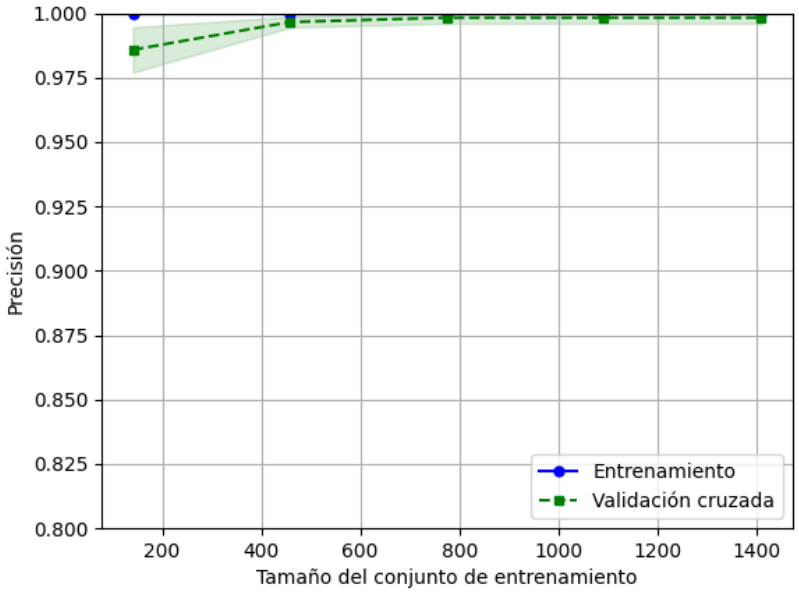


Ilustración 19: Curva de aprendizaje SVM Enfriadora

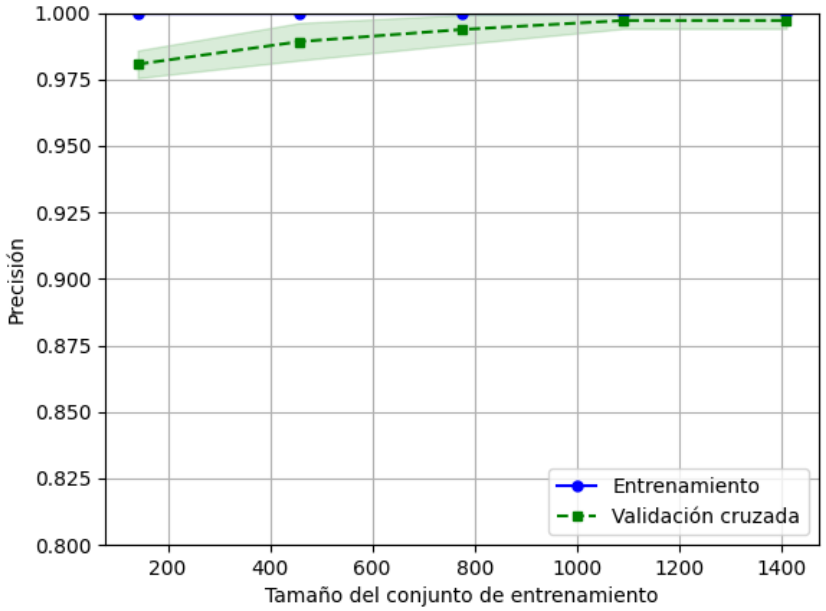


Ilustración 20: Curva de aprendizaje KNN Enfriadora

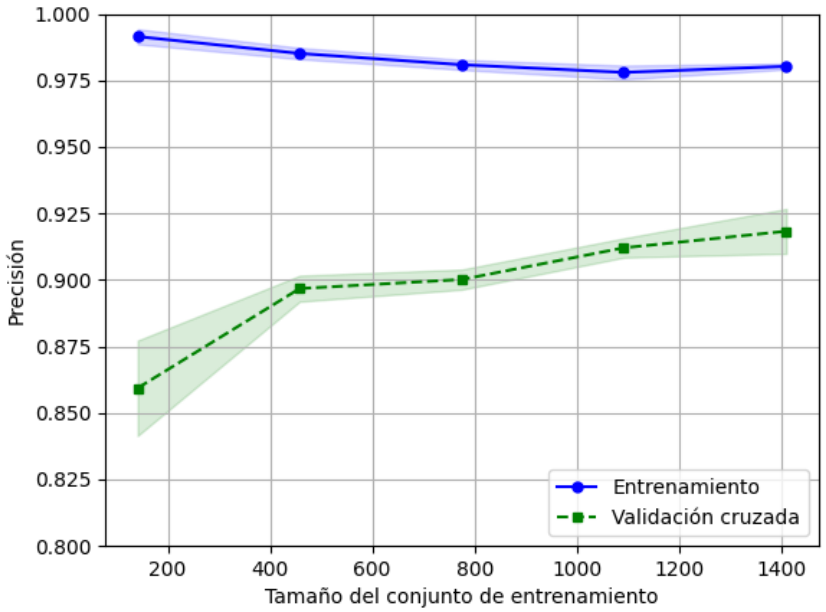


Ilustración 21: Curva de aprendizaje NB Enfriadora

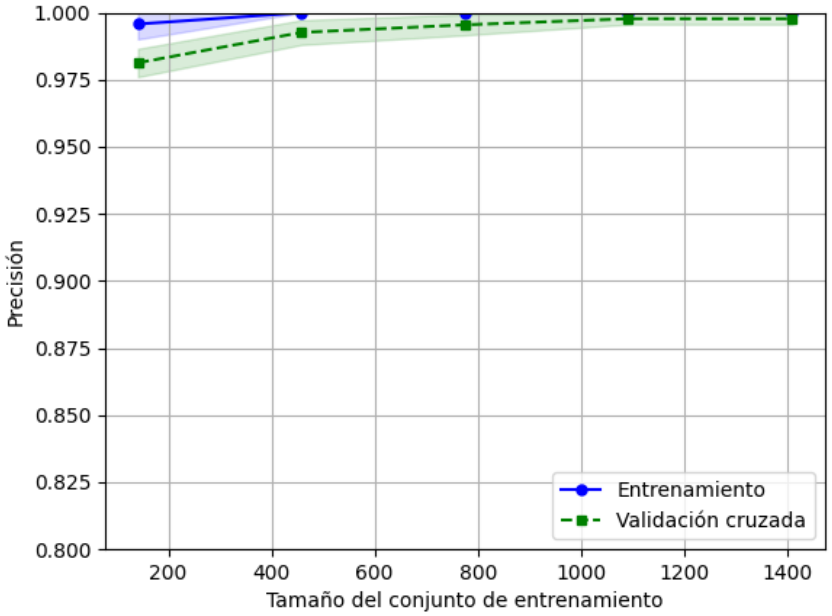


Ilustración 22: Curva de aprendizaje Redes Neuronales Enfriadora

Según los resultados, el modelo con un mayor rendimiento y sobreajuste no apreciable es el modelo de Random Forest, con una exactitud del 100% tanto en el conjunto de prueba como de validación, además de conseguir una convergencia total de la línea de entrenamiento y validación de la curva de aprendizaje lo que implica una buena generalización con un sobreajuste muy bajo.

Además, el área que envuelve a las curvas de aprendizaje representa la desviación estándar de las predicciones y se puede apreciar que en Random Forest es mínima, por lo que el modelo no varía prácticamente al ser ejecutado varias veces con un mismo conjunto de datos.

4.2 Evaluación de los modelos del estado de la válvula

Modelo	Métricas Validación cruzada	Métricas conjunto de prueba
Random Forest	Recall medio = 1.0 Puntuación F1 media = 1.0 Exactitud media = 1.0	Recall: 1.0 Puntuación F1: 1.0 Exactitud: 1.0000
SVM Lineal	Recall medio = 0.9988 Puntuación F1 media = 0.9988 Exactitud media = 0.9988	Recall: 1.0 Puntuación F1: 1.0 Exactitud: 1.0000
KNN	Recall medio = 0.9216 Puntuación F1 media = 0.9281 Exactitud media = 0.9216	Recall: 0.9070 Puntuación F1: 0.9063 Exactitud: 0.9070
Naive Bayes	Recall medio = 0.8275 Puntuación F1 media = 0.8167 Exactitud media = 0.8275	Recall: 0.8321 Puntuación F1: 0.8266 Exactitud: 0.8322
Redes neuronales	Recall medio = 0.9909 Puntuación F1 media = 0.9909 Exactitud media = 0.9909	Recall: 0.9727 Puntuación F1: 0.9727 Exactitud: 0.9728

Tabla 15: Comparación de resultados de las métricas de validación y prueba del Estado de la válvula

Los resultados obtenidos para el conjunto de prueba han sido notablemente satisfactorios y dada la alta similitud de los resultados de las métricas de validación cruzada y del conjunto de prueba es probable que no haya sobreajuste, hecho que debe ser contrastado con las curvas de aprendizaje. Las matrices de confusión para el conjunto de prueba son las siguientes:

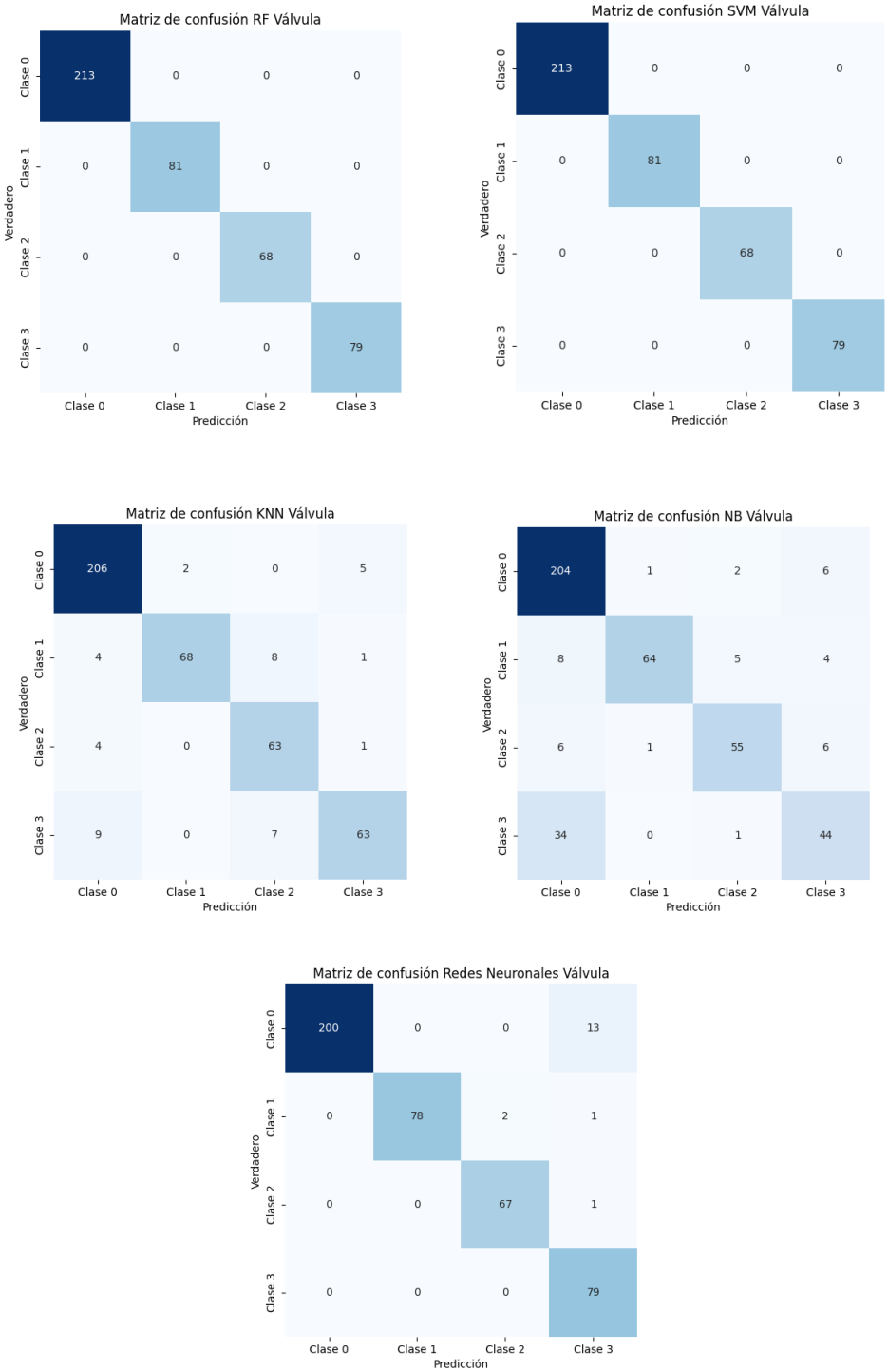


Ilustración 23: Matrices de confusión del Estado de la válvula

Se puede observar cómo los modelos con mayor éxito en las predicciones son RF y SVM, siendo Naive Bayes el modelo más impreciso. El sobreajuste es considerable en los modelos de KNN, Naive Bayes y redes neuronales debido a una falta de convergencia en la curva de los dos primeros y un descenso e irregularidad a medida que se aumenta el tamaño de las muestras, sobre todo en las redes neuronales.

Se puede concluir a la vista de los resultados de la curva de aprendizaje que RF es el modelo con mayor rendimiento, ya que la convergencia de la línea de validación y entrenamiento es muy buena, al mismo tiempo que el área que envuelve a la línea de validación es muy reducida.

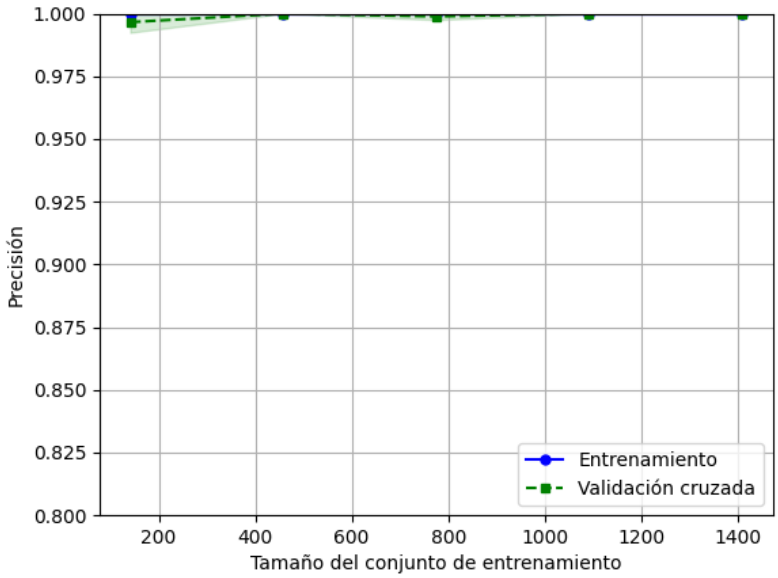


Ilustración 24: Curva de aprendizaje RF Válvula

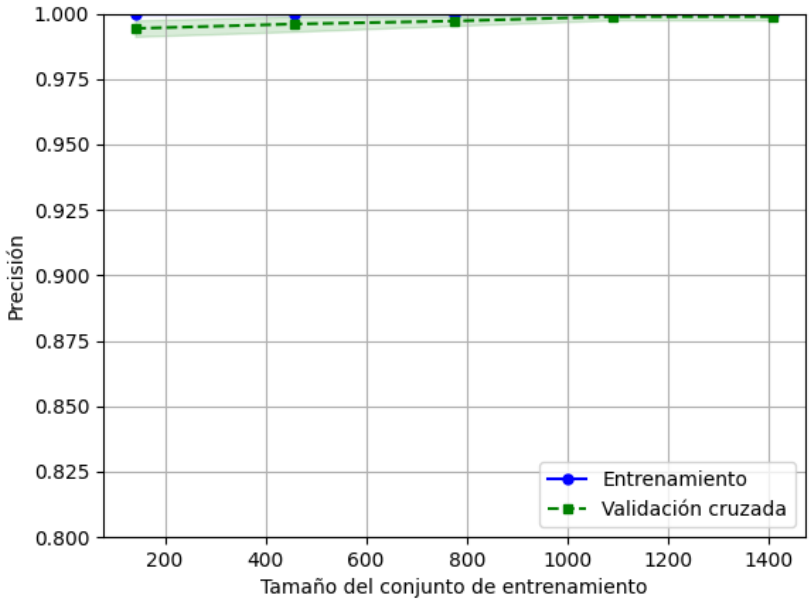


Ilustración 25: Curva de aprendizaje SVM Válvula

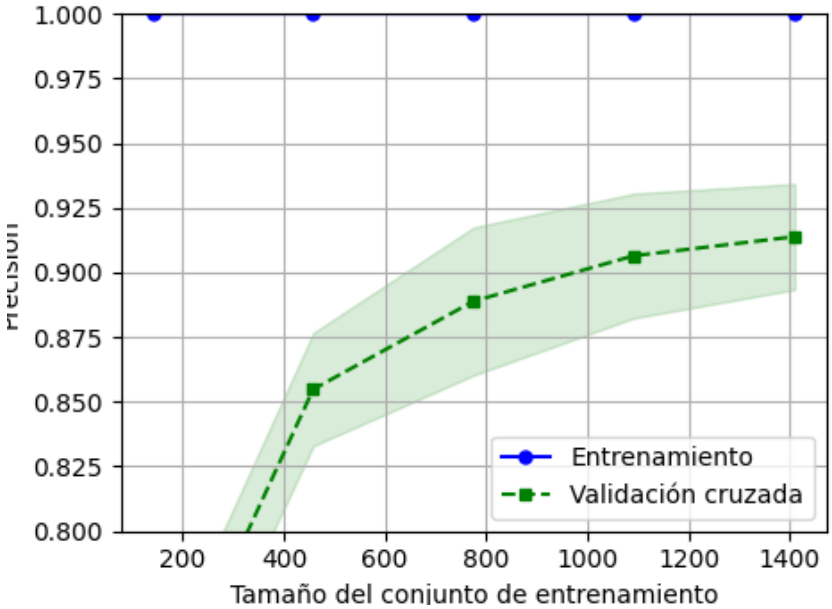


Ilustración 26: Curva de aprendizaje KNN Válvula

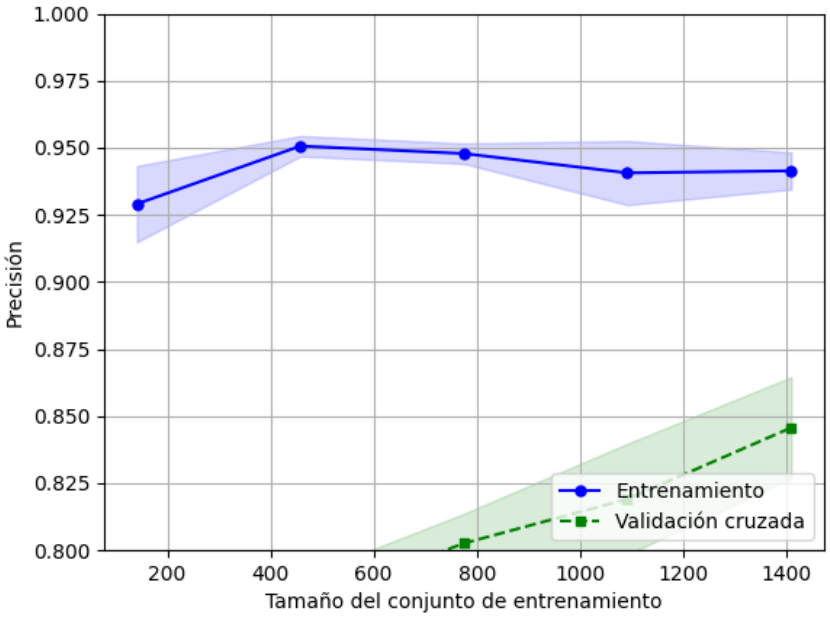


Ilustración 27: Curva de aprendizaje NB Válvula

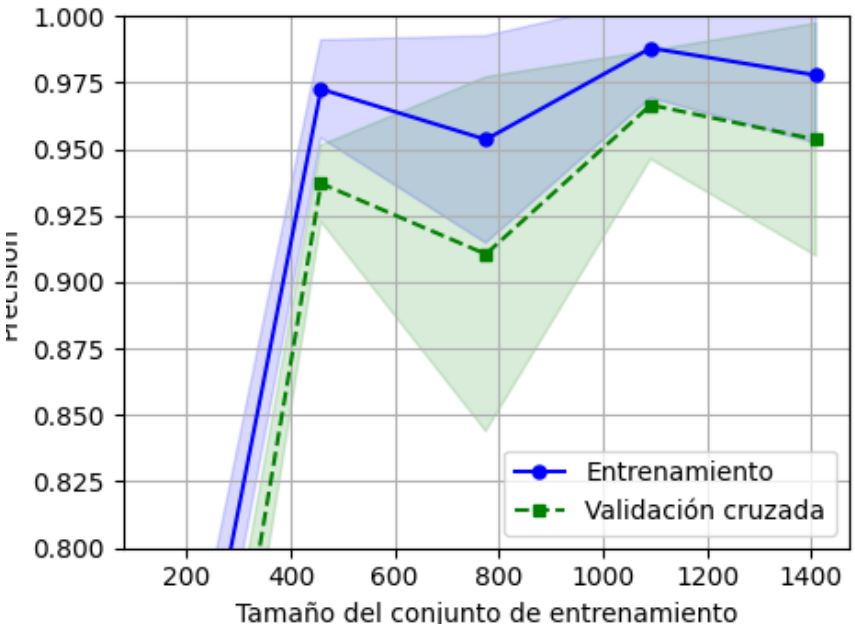


Ilustración 28: Curva de aprendizaje Redes Neuronales Válvula

4.3 Evaluación de los modelos de la fuga interna de la bomba

Modelo	Métricas Validación cruzada	Métricas conjunto de prueba
Random Forest	Recall medio = 0.9971 Puntuación F1 media = 0.9971 Exactitud media = 0.9971	Recall: 0.9909 Puntuación F1: 0.9909 Exactitud: 0.9909
SVM Lineal	Recall medio = 0.9988 Puntuación F1 media = 0.9988 Exactitud media = 0.9988	Recall: 0.9954 Puntuación F1: 0.9954 Exactitud: 0.9955
KNN	Recall medio = 0.9937 Puntuación F1 media = 0.9937 Exactitud media = 0.9937	Recall: 0.9886 Puntuación F1: 0.9886 Exactitud: 0.9887

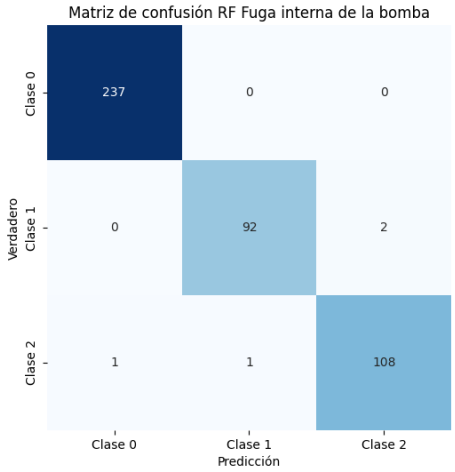
Naive Bayes	Recall medio = 0.5927 Puntuación F1 media = 0.5242 Exactitud media = 0.5927	Recall: 0.5873 Puntuación F1: 0.5277 Exactitud: 0.5873
Redes neuronales	Recall medio = 0.9102 Puntuación F1 media = 0.9113 Exactitud media = 0.9109	Recall: 0.9229 Puntuación F1: 0.9238 Exactitud: 0.9229

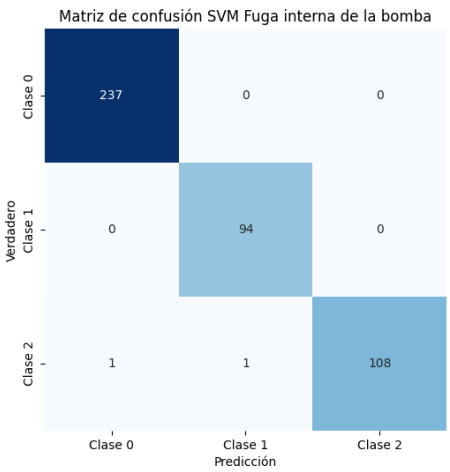
Tabla 16: Comparación de resultados de las métricas de validación y prueba de la Fuga interna de la bomba

Los resultados obtenidos son bastante precisos en la mayoría de los modelos superando un 90% de exactitud a excepción del caso de Naive Bayes, donde el modelo no parece ser eficiente en cuanto a exactitud en las predicciones.

Parece ser que los dos modelos que en un principio están en torno al 99% de exactitud, cuyas validaciones cruzadas verifican las métricas del conjunto de prueba son: Random Forest y SVM. Esto es indicativo de que no hay sobreajuste, pero esto debe ser verificado con las curvas de aprendizaje y contrastado con la repartición de errores de predicción plasmados en sus respectivas matrices de confusión.

Puesto que las matrices de confusión son exactamente iguales para ambos modelos la valoración concluyente reside en el análisis de la curva de aprendizaje.





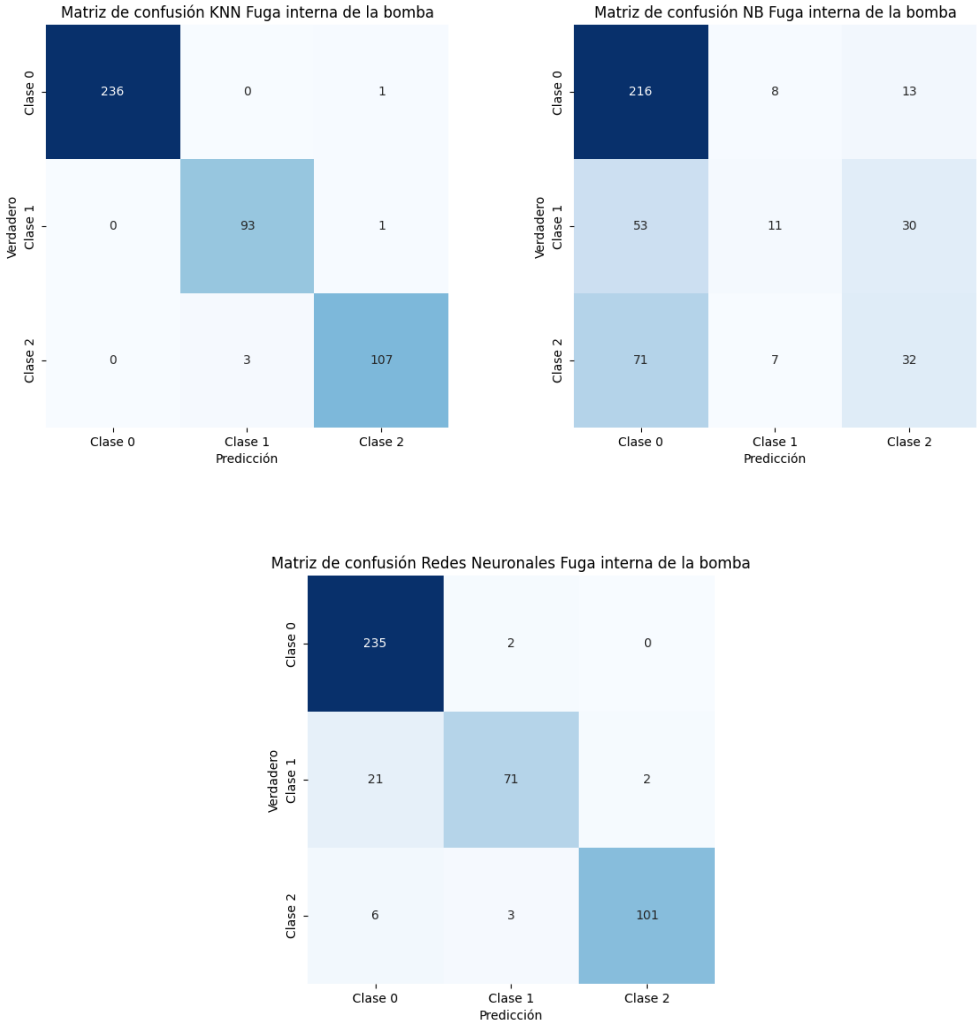


Ilustración 29: Matrices de confusión de la Fuga interna de la bomba

Las curvas de aprendizaje para SVM y Random Forest son muy similares, sin embargo, SVM posee una pequeña mayor convergencia por lo que se concluye que el modelo con mayor exactitud y menor sobreajuste y por tanto el más eficiente para predecir las fugas de la bomba es el modelo SVM.

Por otro lado, en los demás modelos la convergencia de las curvas no es del todo desfavorable, pero menor al modelo seleccionado, y con las exactitudes sucede lo mismo.

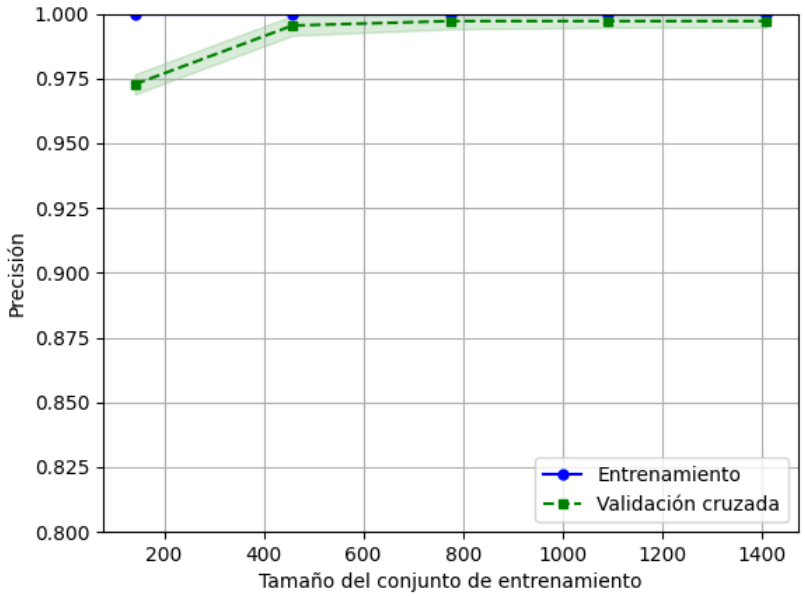


Ilustración 30: Curva de aprendizaje RF Bomba

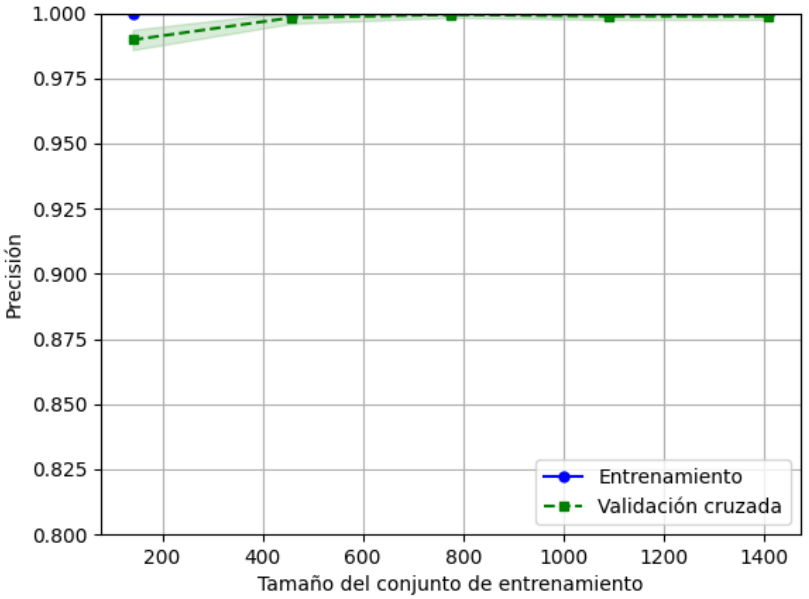


Ilustración 31: Curva de aprendizaje SVM Bomba

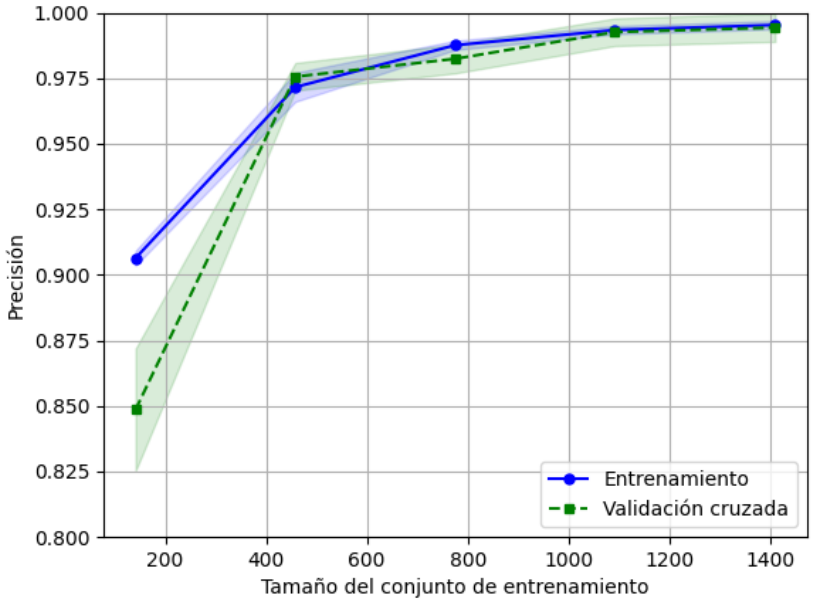


Ilustración 32: Curva de aprendizaje KNN Bomba

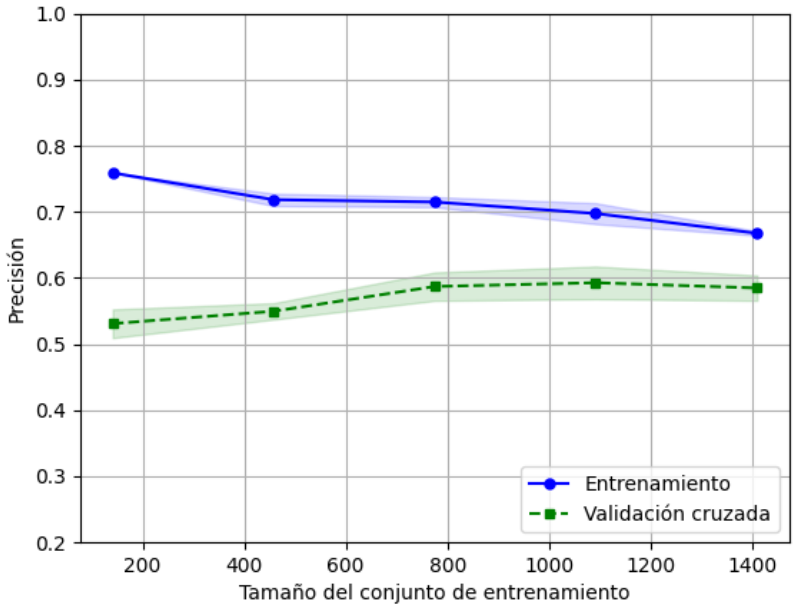


Ilustración 33: Curva de aprendizaje NB Bomba

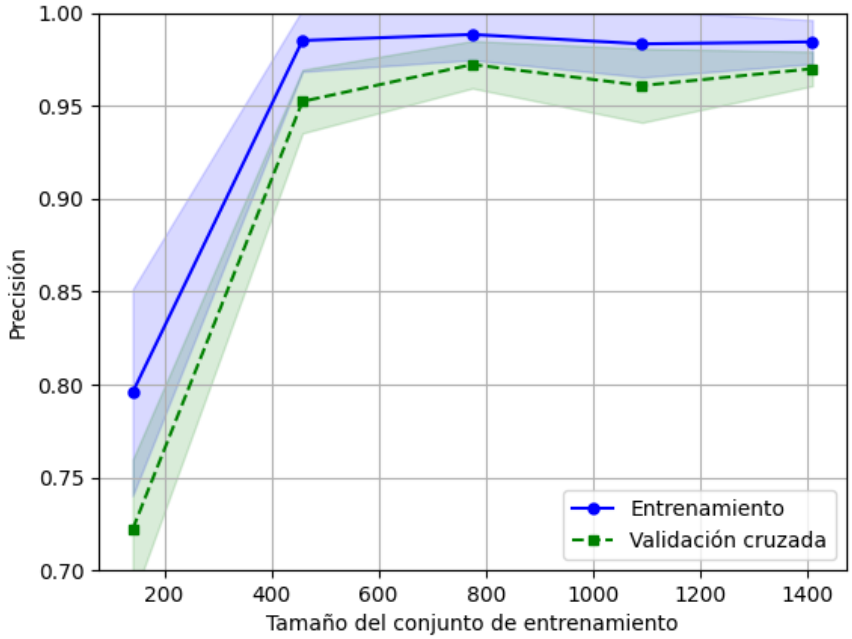


Ilustración 34: Curva de aprendizaje Redes Neuronales)

4.4 Evaluación de los modelos de la presión del acumulador

Modelo	Métricas Validación cruzada	Métricas conjunto de prueba
Random Forest	Recall medio = 0.9807 Puntuación F1 media = 0.9805 Exactitud media = 0.9807	Recall: 0.9863 Puntuación F1: 0.9863 Exactitud: 0.9864
SVM Lineal	Recall medio = 0.9710 Puntuación F1 media = 0.9709 Exactitud media = 0.9710	Recall: 0.9727 Puntuación F1: 0.9726 Exactitud: 0.9728
KNN	Recall medio = 0.9529 Puntuación F1 media = 0.9532	Recall: 0.9591 Puntuación F1: 0.9591 Exactitud: 0.9592

	Exactitud media = 0.9529	
Naive Bayes	Recall medio: 0.4010 Puntuación F1 media: 0.4010 Exactitud: 0.4007	Recall: 0.4376 Puntuación F1: 0.3948 Exactitud: 0.4376
Redes neuronales	Recall medio = 0.9381 Puntuación F1 media = 0.9385 Exactitud media = 0.9381	Recall: 0.9070 Puntuación F1: 0.9084 Exactitud: 0.9070

Tabla 17: Comparación de resultados de las métricas de validación y prueba de la Presión del acumulador

A la vista de los resultados, se cumple el mismo patrón que en el caso de la fuga de la bomba, ya que todos los modelos son bastante precisos a excepción de Naive Bayes.

Random Forest y SVM son de nuevo los modelos con mayor exactitud por lo que es necesario contrastar qué clases son las que fallan para comprobar su repercusión en el funcionamiento real del sistema.

En los dos modelos se predice correctamente cuando el fallo en el acumulador es inminente (Clase3). Random Forest falla más en las predicciones cuando la presión se reduce severamente (Clase 0) y SVM cuando la presión se reduce ligeramente (Clase 1). También, cabe destacar que SVM falla más a la hora de predecir cuando la presión es óptima en comparación con Random Forest. Sin embargo, este último fallo de SVM no tiene un impacto crítico, ya que se consideran falsos positivos en el sistema puesto que predice fallos en el sistema cuando este realmente no va a fallar.

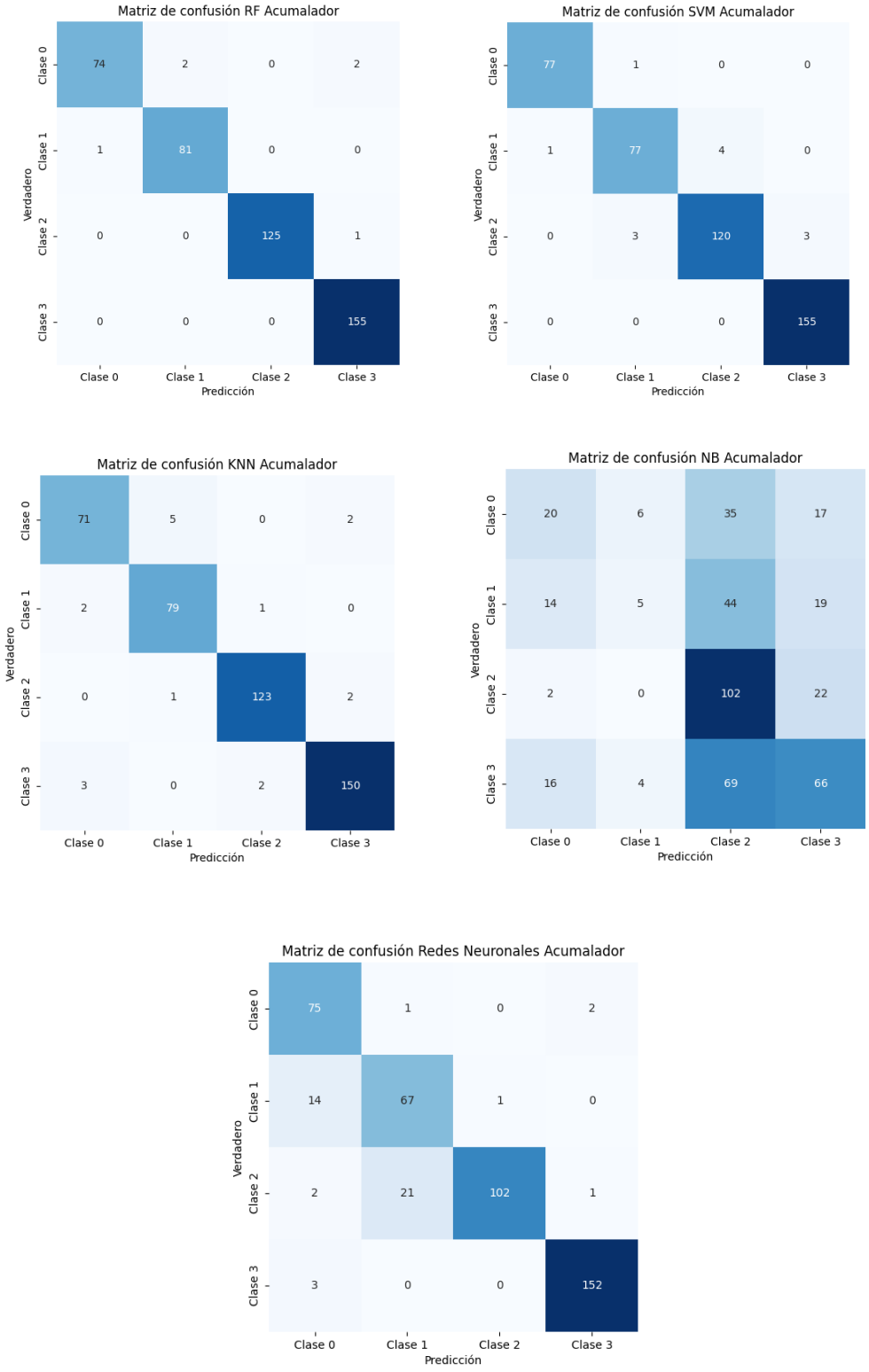


Ilustración 35: Matrices de confusión de la Presión del acumulador

Para concluir qué modelo escoger entre Random Forest o SVM se analiza la curva de aprendizaje de ambos modelos para descartar un posible sobreajuste. En este caso, la curva de aprendizaje no llega del todo a converger con la línea de entrenamiento, aunque el sobreajuste es mínimo. El 1% mayor de precisión del conjunto de prueba en Random Forest respecto de SVM puede deberse a esta situación y a una mayor desviación estándar (área de la línea de validación), por lo que no es concluyente dicha diferencia en las métricas del conjunto de prueba.

De esta manera, pasan ser determinantes los resultados de las matrices de confusión, curvas de aprendizaje y métricas de validación. Los resultados de la matriz de confusión de SVM son menos críticos y la curva se estabiliza mejor. Es importante destacar que la curva de aprendizaje de validación de RF desciende ligeramente a medida que aumenta el taño del conjunto de entrenamiento, mientras que en SVM no se aprecia ningún punto de inflexión.

Los demás modelos no generalizan de manera tan satisfactoria y las precisiones son bastante más reducidas. En las redes neuronales, la curva de aprendizaje comienza a descender cuando la cantidad de instancias o muestras es muy grande lo que denota una generalización no deseable y una desviación estándar demasiado alta, llegando hasta un 20%, por lo que no es para nada un modelo consistente.

Por ende, teniendo en cuenta todos estos factores, el modelo que mejor equilibrio posee entre su exactitud y generalización ante nuevos datos sumado a errores no críticos en la matriz de confusión es el modelo SVM, hecho que lo convierte en el modelo más recomendable para el conjunto de datos y finalidad del mantenimiento predictivo.

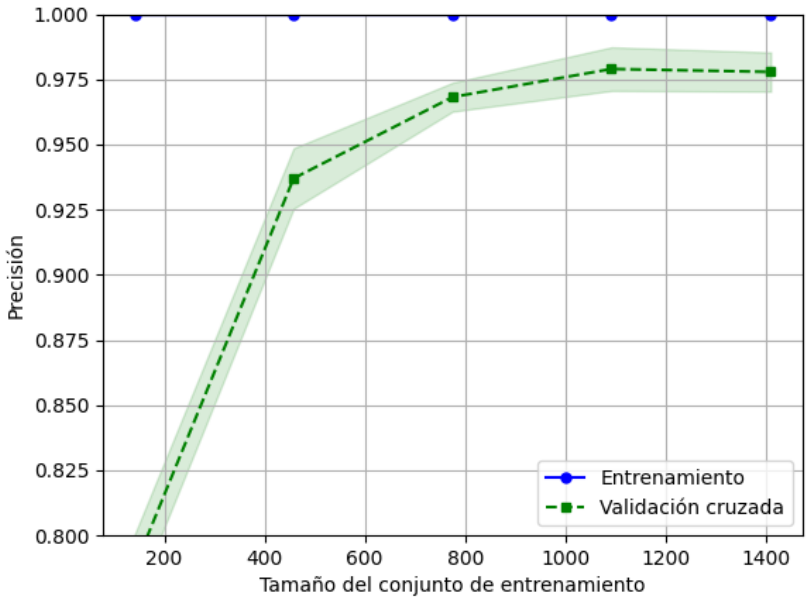


Ilustración 36: Curva de aprendizaje RF Acumulador

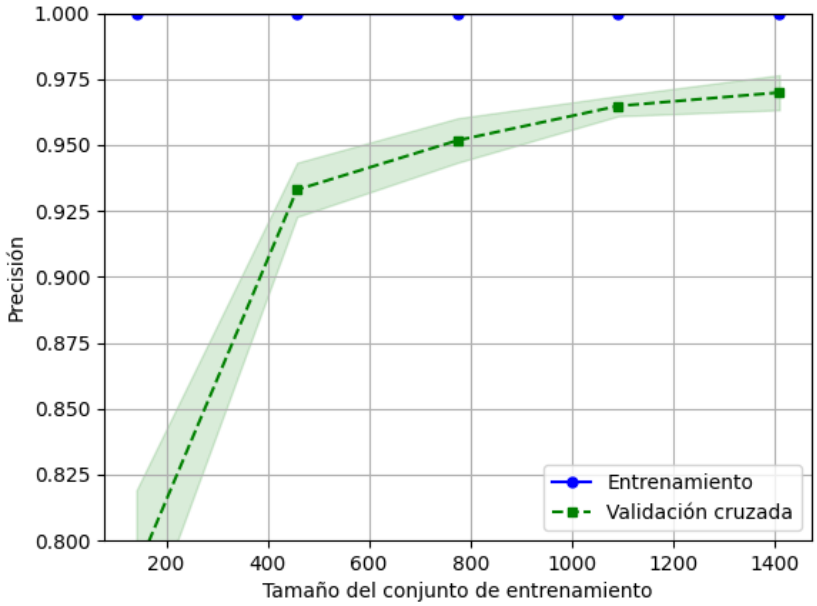


Ilustración 37: Curva de aprendizaje SVM Acumulador

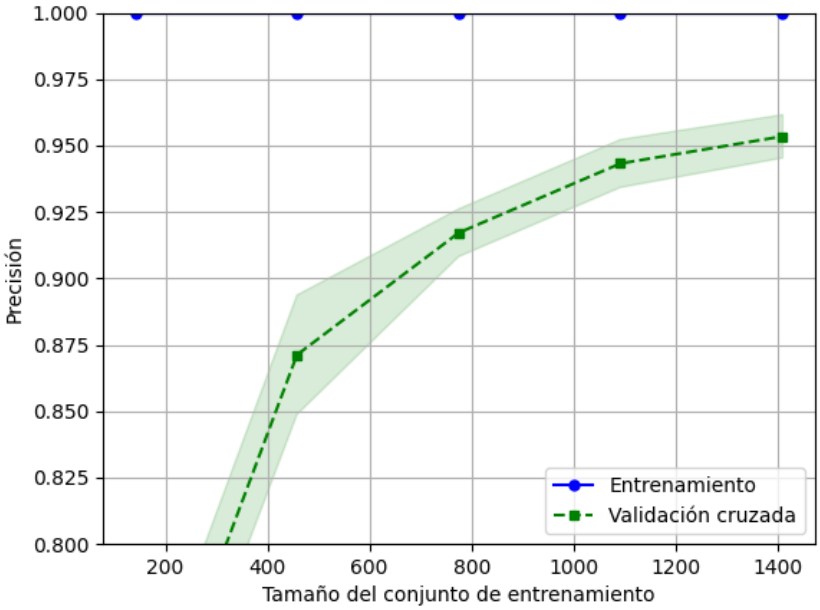


Ilustración 38: Curva de aprendizaje KNN Acumulador

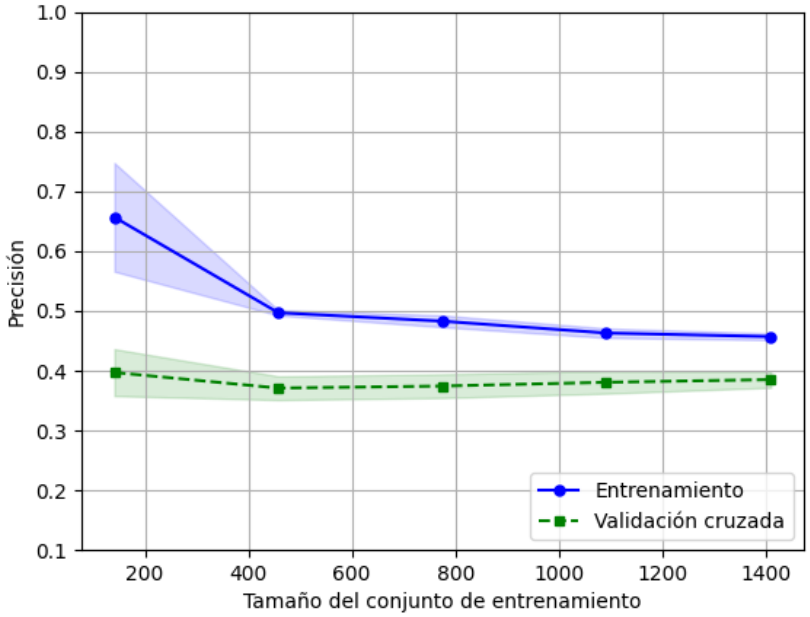


Ilustración 39: Curva de aprendizaje NB Acumulador

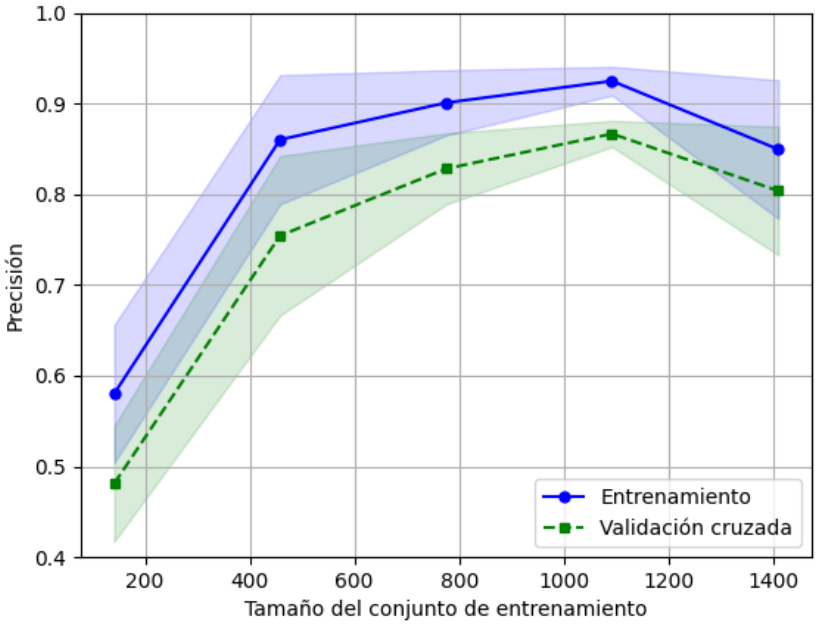


Ilustración 40: Curva de aprendizaje Redes neuronales Acumulador

4.5 Evaluación de los modelos de la estabilidad del sistema

Modelo	Métricas Validación cruzada	Métricas conjunto de prueba
Random Forest	Recall medio = 0.9848 Puntuación F1 media = 0.9845 Exactitud media = 0.9848	Recall: 0.9862 Puntuación F1: 0.9862 Exactitud: 0.9862
SVM Lineal	Recall medio = 0.9523 Puntuación F1 media = 0.9524 Exactitud media = 0.9523	Recall: 0.9810 Puntuación F1: 0.9810 Exactitud: 0.9810
KNN	Recall medio = 0.9882 Puntuación F1 media = 0.9882 Exactitud media = 0.9882	Recall: 0.9904 Puntuación F1: 0.9904 Exactitud: 0.9904
Naive Bayes	Recall medio = 0.8455 Puntuación F1 media = 0.8455 Exactitud media = 0.8455	Recall: 0.8637 Puntuación F1: 0.8636 Exactitud: 0.8638
Redes neuronales	Recall medio = 0.9402 Puntuación F1 media = 0.9403 Exactitud media = 0.9433	Recall: 0.9517 Puntuación F1: 0.9517 Exactitud: 0.9517

Tabla 18: Comparación de resultados de las métricas de validación y prueba de la Estabilidad del sistema

Las métricas de la validación cruzada y del conjunto de prueba son muy similares y las métricas de conjunto de prueba son satisfactorias para la mayoría de los modelos siendo KNN el modelo con mejores métricas.

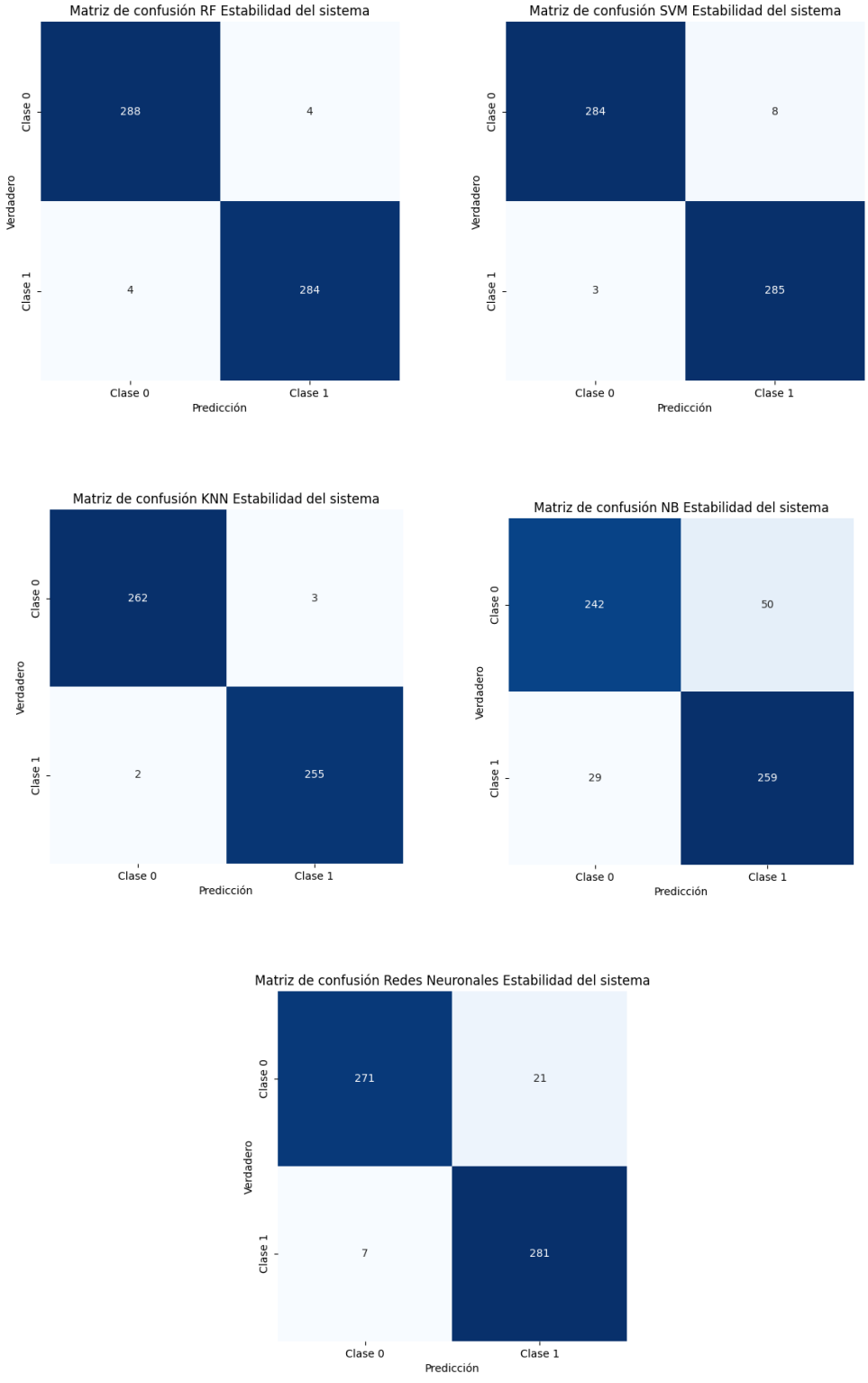


Ilustración 41: Matrices de confusión de la Estabilidad del sistema

Para el algoritmo de KNN de la estabilidad del sistema se ha usado una proporción del 82% del conjunto de datos original en la muestra de entrenamiento debido a que se obtienen unos mejores resultados para los datos de prueba.

La línea de aprendizaje de validación es la que mejor converge con la línea de entrenamiento en comparación con los demás modelos. No posee ningún punto de inflexión que haga decaer la línea de validación al aumentar el número de muestras de entrenamiento del modelo.

La desviación estándar en Naive Bayes y redes neuronales es bastante alta debido a que el área que envuelve las líneas de ambas curvas de aprendizaje es muy amplia, lo que hace que sean modelos inestables sumado a una precisión no tan deseable como en KNN. Random Forest y SVM son modelos con menor exactitud y cierto sobreajuste debido a una falta de convergencia que se acentúa de manera destacable en SVM.

Debido a esto, se concluye que el modelo con mayor rendimiento, que mejor generaliza y que consigue una menor desviación estándar para predecir la estabilidad del sistema es KNN.

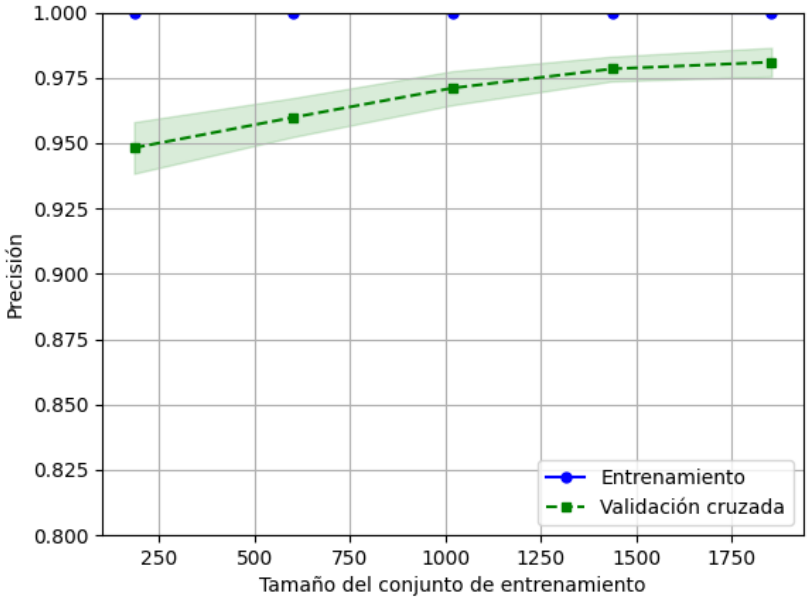


Ilustración 42: Curva de aprendizaje RF Estabilidad

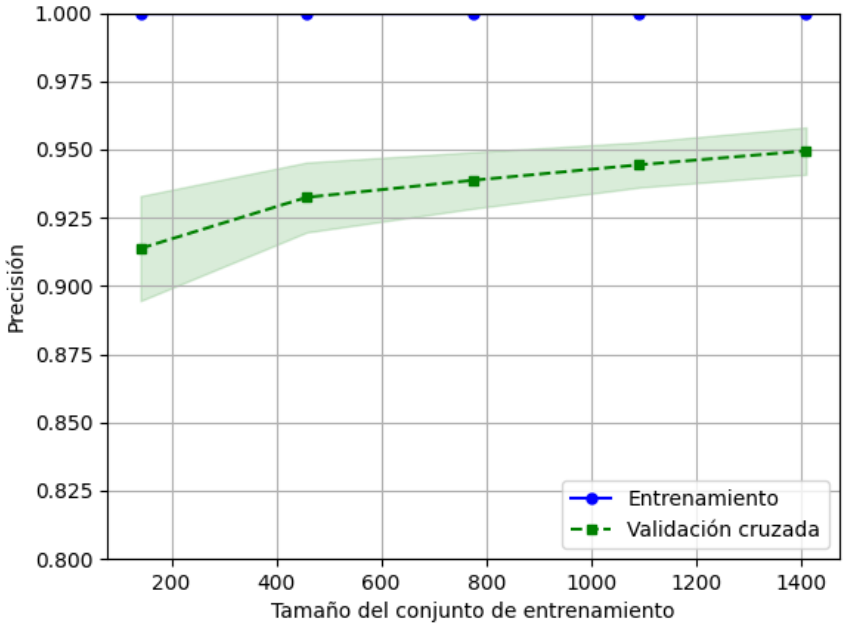


Ilustración 43: Curva de aprendizaje SVM Estabilidad

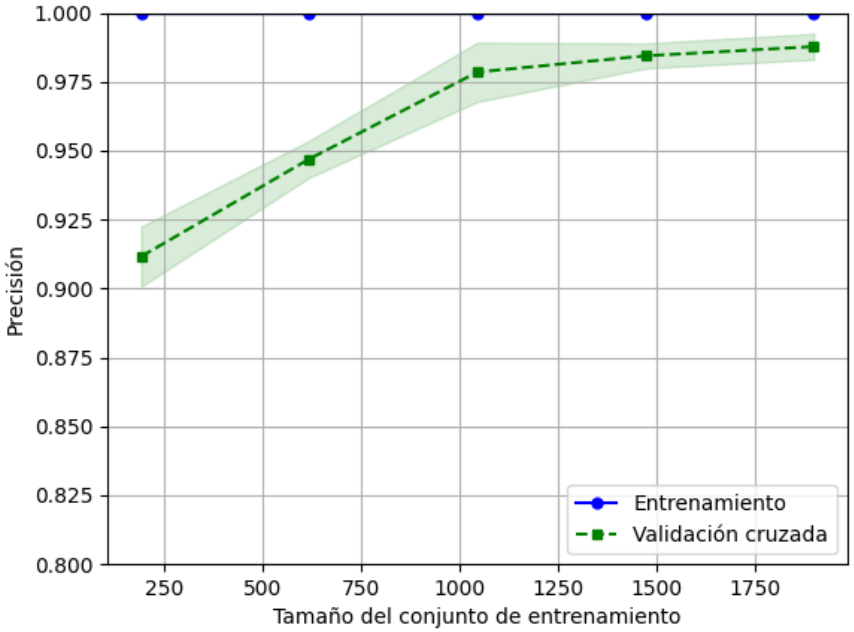


Ilustración 44: Curva de aprendizaje KNN Estabilidad

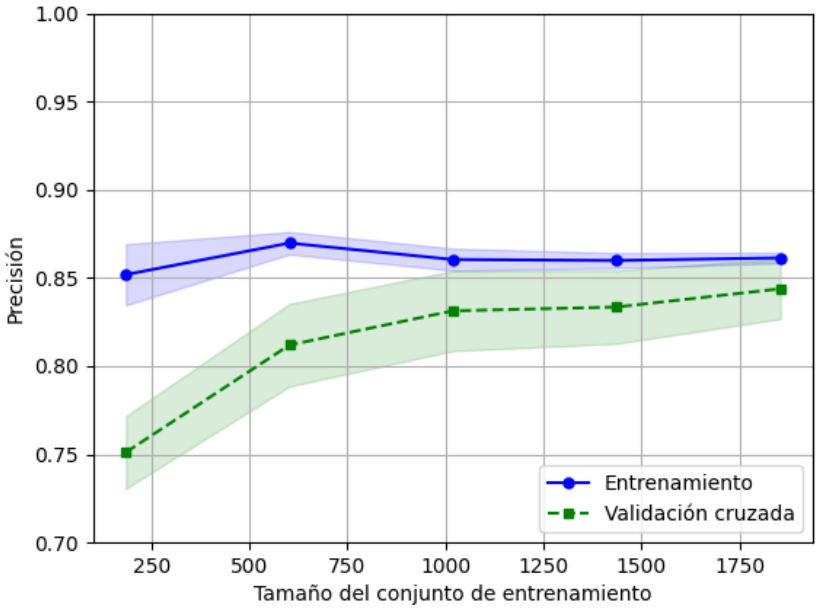


Ilustración 45: Curva de aprendizaje NB Estabilidad

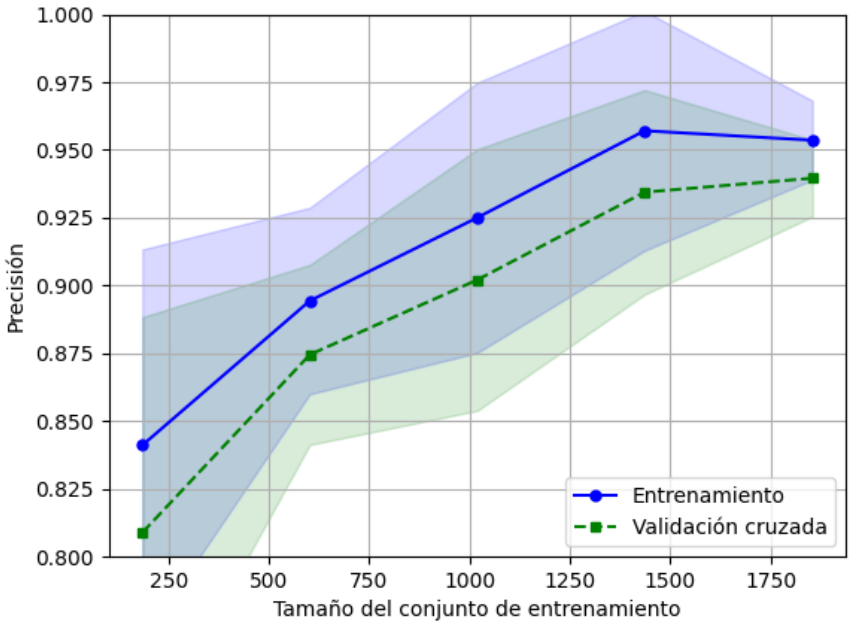


Ilustración 46: Curva de aprendizaje Redes Neuronales Estabilidad

4.6 Resultado general más eficaz

Una vez realizado el proceso de evaluación y validación de todos los modelos de cada variable se han concluido los resultados con mayor exactitud en las predicciones y que mejor generalizan en la siguiente tabla:

Variable	Modelo	Exactitud media validada
Enfriador	Random Forest	100%
Estado de la válvula	Random Forest	100%
Fuga de la bomba	SVM	99,88%
Presión del acumulador	SVM	97,10%
Estabilidad del sistema	K-NN	98,82%

Tabla 19: Resultados más eficientes para cada variable

Se ha tomado como resultado final la exactitud media de validación que es similar en todos los casos a los resultados obtenidos en las predicciones del conjunto de prueba.

Todas las curvas de estos modelos son altamente satisfactorias y las matrices de confusión no son resultados altamente críticos para el correcto desempeño de un software de mantenimiento predictivo que los incluya.

Además, la aplicación de los presentes modelos diseñados se deberá desempeñar en tiempo real siendo el tiempo de muestreo de la monitorización de 1 minuto. Esto hace que los resultados sean altamente satisfactorios, ya que en caso de un falso negativo desfavorable (que se prediga un funcionamiento correcto cuando hay fallo) en un determinado minuto, se verá compensado con una validación por parte de un supervisor del software que verifique si la secuencia en el tiempo del estado de determinado componente se mantiene en el tiempo o es puntual. De esta manera, cualquier fallo en las predicciones, al existir una gran solidez y precisión de los modelos, no tendrá una gran incidencia debido a su contraste con la trama de predicciones en los próximos minutos.

Para ejemplificar de manera temporal el funcionamiento se muestra la siguiente equivalencia de las exactitudes con los minutos de funcionamiento y los minutos medios en los que el sistema funciona correctamente en cada uno de los componentes durante 20 horas.

Componente	Minutos de funcionamiento	Minutos con estado correcto
Enfriador	1200	1200
Válvula	1200	1200
Bomba	1200	1198
Acumulador	1200	1165
Estabilidad	1200	1185

Tabla 20: Número de minutos en los que las predicciones son correctas durante 20 horas

Capítulo 5. Conclusiones

5.1 Discusión de los resultados

A pesar del alto rendimiento de los modelos, no se puede asegurar que ante nuevos datos el sistema rinda con el mismo grado de exactitud, ya que la generalización de los modelos nunca es absoluta.

Sin embargo, los métodos de evaluación empleados aseguran que el rango de exactitud de los modelos ante conjuntos de datos no pertenecientes al original será sólido y con un rendimiento alto. Esto, sumado a una desviación estándar en los modelos seleccionados muy baja en todos los casos seleccionados, los convierte en modelos deseables. La desviación se ha valorado de manera visual por medio de las curvas de aprendizaje, ya que es el área que envuelve a las líneas de entrenamiento y validación. Además, también se han calculado las desviaciones estándar de los modelos en los procesos de validación y en prácticamente todos los casos el valor era insignificante y no valorativo.

Todos los modelos que se han concluido como los de mayor rendimiento presentan una alta solidez en la convergencia de curvas de aprendizaje, descartando así un sobreajuste apreciable lo que permite una buena capacidad de generalización ante nuevos datos.

Esto, sumado a las métricas con un alto valor de exactitud y unos matrices de confusión con resultados no críticos hace que el proyecto y el código elaborado pueda ser aplicable en un contexto real del sistema hidráulico descrito.

No obstante, es importante tener en cuenta que, aunque se haya profundizado de manera consistente en los métodos de evaluación, los resultados nunca son absolutos y en proyectos de machine learning, nunca se pueden realizar afirmaciones absolutas, por lo que se debe asumir la verificación y validación de los modelos con escepticismo debido a las consecuencias éticas que puede conllevar un proyecto de inteligencia artificial.

Este escepticismo proviene de una responsabilidad por parte de los desarrolladores que es fruto de la falta de control en muchos casos en los procesos de entrenamiento de los modelos debido a su alto nivel de complejidad matemática. En muchas ocasiones, los fallos en inteligencia artificial son considerados como “cajas negras” siendo de alta importancia que la sociedad oriente su perspectiva hacia este tipo de tecnologías con responsabilidad y escepticismo.

5.2 Dificultades del estudio

Este proyecto requiere un conocimiento avanzado de programación en Python, algoritmos de aprendizaje automático y capacidad de análisis de datos a gran escala debido a la alta densidad de datos (96 millones de datos).

Para la realización de este trabajo se ha tenido que ahondar en el sector con el objetivo de conocer el trasfondo matemático y ser consciente de todos los procesos para la elaboración del código. El código elaborado es extenso y complejo siendo en total aproximadamente 3500 líneas para todo el estudio.

La mayor complejidad reside en la elevada cantidad de datos, hecho altamente incidente en los tiempos de procesamiento y complejidad computacional.

Se han valorado múltiples alternativas, el proyecto se ha iniciado recurriendo a servidores de computación en la nube y recurriendo al framework PySpark de Apache que está orientado a este tipo de procesos y permite la implementación de librerías de machine learning (Spark ML). La idea inicial que se propuso consistió en procesar todos los datos en la nube mediante servidores y nodos computacionales de procesamiento paralelo (cluster) de Amazon Web Services, elaborando el código con las librerías de PySpark para reducir los tiempos de ejecución.

Sin embargo, dichos servidores e instancias computacionales son de pago y no era rentable su uso a largo plazo. Por tanto, finalmente se ha optado por usar las librerías de aprendizaje automático tradicionales como Scikit-learn en el hardware local, lo cual resultó eficiente y los tiempos de espera eran moderados, asumibles y han permitido una programación más sencilla sin tener en cuenta procesos de conectividad con servidores.

Los tiempos de entrenamiento son moderados con el hardware disponible, pero el proceso se puede optimizar con un procesador mucho más potente, con un mayor número de núcleos, memoria RAM más amplia y con una velocidad de lectura y escritura con más hercios.

5.3 Líneas futuras

En este proyecto se ha realizado el estudio y evaluación de los modelos en base a un conjunto de datos del que se disponía. Sin embargo, esto no constituye realmente un producto que pueda ser competente el mercado. Para ello, la siguiente fase se denomina deployment de los modelos.

El deployment consiste en la generación de un fichero con los modelos más eficientes para cada variable para poder ser subido a la nube. Este proceso consiste en la construcción de la API para poder materializar en un software la funcionalidad de la aplicación del proyecto.

Una API permite el acceso al servicio de los modelos sin dar acceso al código fuente a los usuarios. La API constituye realmente el producto completo del estudio realizado y permite una introducción al mercado gracias a la posibilidad de uso por parte de usuarios externos.

Dicha API permitiría la introducción de datos en tiempo real del sistema hidráulico y las predicciones de los estados de las componentes se almacenarían en la nube, mostrándose con una interfaz.

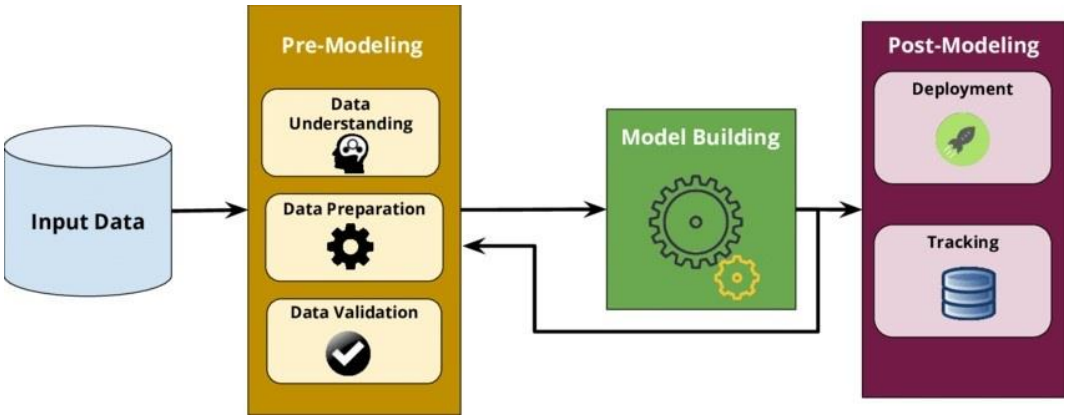


Ilustración 47: Fases de un proyecto de machine learning Fuente[19]

Existen varias formas de realizar este proceso:

- Plataformas de computación en la nube: los servicios de deployment son ofertados por numerosas plataformas en la nube como por ejemplo Microsoft Azure, Google Cloud o Amazon Web Services. Permiten el uso de herramientas de gestión y alojamiento de modelos de inteligencia artificial y la creación de APIs que dan la posibilidad de interactuar a los usuarios con las funcionalidades del modelo.

En concreto, Amazon Web Services tiene una herramienta llamada AWS Lambda que permite ejecutar código con escalabilidad de datos, sin tener que prestar atención a los servidores subyacentes, es decir, no es necesario crear una instancia de AWS EC2 y emplear una máquina virtual, sino que únicamente permite al programador la gestión del código y de la interfaz de la API. Esta falta de interacción con el servidor se gestiona mediante solicitudes HTTP, que hacen que el código se ejecute de manera independiente por fragmentos teniendo en cuenta la demanda y la necesidad de respuestas a sucesos concretos por parte del usuario.

- Frameworks: existen frameworks destinados a Python que permiten diseñar una aplicación web que permita la conformación de la API para poder ser depositada en cualquier servicio de computación en la nube. Los más destacables son Django y Flask.

- Plataformas específicas de deployment: son plataformas especializadas en deployment de modelos de inteligencia artificial pertenecientes a frameworks que están generalmente orientados a modelos de aprendizaje profundo con imágenes y lenguaje natural. Actualmente, los más conocidos son PyTorch y Tensor Flow que incluyen funciones de deployment como TensorFlow Serving y PyTorch Lightning, respectivamente.

- Contenedores: el más común es Docker, que permite materializar los modelos en paquetes que pueden ser gestionados con orquestadores como Kubernete que habilitan la capacidad de controlar y escalar los modelos en instancias de núcleos de computación en la nube dando acceso a la API a los usuarios.

Bibliografía

[1] Wikipedia contributors. (s/f). *Mantenimiento predictivo*. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Mantenimiento_predictivo&oldid=146888201

[2] *UCI Machine Learning Repository: Condition monitoring of hydraulic systems Data Set*. (s/f). Uci.edu. Recuperado el 26 de mayo de 2023, de https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems

[3] Priy, S. P. (2018, enero 15). *Clustering in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/clustering-in-machine-learning/

[4] *Esquema de la red neuronal para clasificación*. (s/f). Interactivechaos.com. Recuperado el 26 de mayo de 2023, de https://interactivechaos.com/es/manual/tutorial-de-deep-learning/esquema-de-la-red-neuronal-para-clasificacion

[5] *GPT-4*. (s/f). Openai.com. Recuperado el 26 de mayo de 2023, de https://openai.com/product/gpt-4

[6] Vila, M. A. S. (s/f). *[Azure Machine Learning] Sobreajuste y subajuste (Overfitting and Underfitting)*. Epicalsoft — Superheroic Software Development Blog. Recuperado el 26 de mayo de 2023, de http://epicalsoft.blogspot.com/2019/02/azure-machine-learning-sobreajuste-y.html

[7] (S/f). Udemy.com. Recuperado el 26 de mayo de 2023, de https://www.udemy.com/course/machinelearning-es/learn/lecture/14133335#overview

[8] Arce, J. I. B. (2019, julio 26). *La matriz de confusión y sus métricas*. Juan Barrios. https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

[9] Miller, V. (2018, enero 29). *Explorando algoritmos de aprendizaje automático supervisado*. Toptal Engineering Blog; Toptal. https://www.toptal.com/machine-learning/explorando-algoritmos-de-aprendizaje-automatico-supervisado

[10] (S/f). Researchgate.net. Recuperado el 26 de mayo de 2023, de https://www.researchgate.net/figure/The-technique-of-KFold-cross-validation-illustrated-here-for-the-case-K-4-involves_fig10_278826818

[11] Nairal. (2023, enero 6). *Pandas de Python vs Polars: ¿Quién gana esta pelea de librerías para data science?* Nairal.

[12] Grima, C. (2017, septiembre 4). *Las distancias en Manhattan*. Revista Mètode.

[13] *Cómo Entrenar una CNN Usando Adam en Keras - DataSmarts Español*. (2020, enero 18). DataSmarts Español; DataSmarts. https://datasmarts.net/es/como-entrenar-una-cnn-usando-adam-en-keras/

[14] *Polars*. (s/f). Pola.Rs. Recuperado el 26 de mayo de 2023, de https://www.pola.rs/

[15] Varoquaux, G., Buitinck, L., Louppe, G., Grisel, O., Pedregosa, F., & Mueller, A. (2015). Scikit-learn: Machine learning without learning the machinery. *GetMobile Mobile Computing and Communications*, *19*(1), 29–33. https://doi.org/10.1145/2786984.2786995

[16] *Matplotlib — visualization with python*. (s/f). Matplotlib.org. Recuperado el 26 de mayo de 2023, de https://matplotlib.org/

[17] *pandas*. (s/f). Pydata.org. Recuperado el 26 de mayo de 2023, de https://pandas.pydata.org/

[18] Bisong, E. (2019). NumPy. En *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 91–113). Apress.

[19] *Herramientas de automatización de Machine Learning (Auto ML) :* (2021, diciembre 2). baobab soluciones. https://baobabsoluciones.es/blog/2021/12/02/herramientas-de-automatizacion-de-machine-learning-auto-ml/

[20] Nikolai Helwig, Eliseo Pignanelli, Andreas SchÃ¼tze, â€˜Condition Monitoring of a Complex Hydraulic System Using Multivariate Statisticsâ€™, in Proc. I2MTC-2015 - 2015 IEEE International Instrumentation and Measurement Technology Conference, paper PPS1-39, Pisa, Italy, May 11-14, 2015, doi: 10.1109/I2MTC.2015.7151267.

[21] Descripción, D. (s/f). Bosch.com. Recuperado el 4 de junio de 2023, de https://dc-br.resource.bosch.com/media/br/training/treinamentos_2020/arquivos_2/Simbologia_Grafica_ISO1219_ES.pdf

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

ÁREA INGENIERÍA INDUSTRIAL

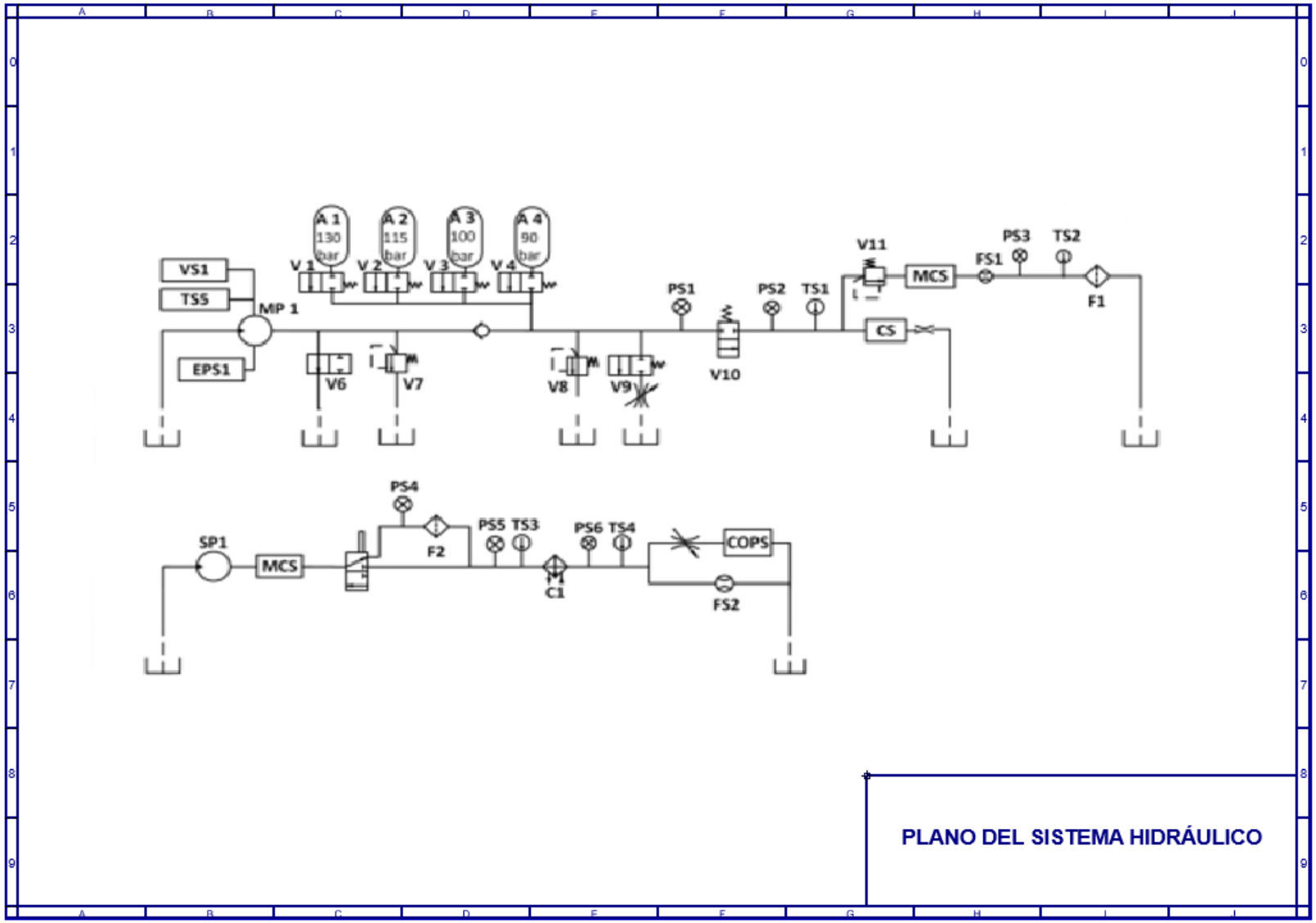
INGENIERÍA EN SISTEMAS INDUSTRIALES

TRABAJO FIN DE GRADO
MACHINE LEARNING APLICADO AL MANTENIMIENTO PREDICTIVO DE UN SISTEMA HIDRÁULICO

PLANOS DEL SISTEMA

Alumno: GERARDO MARTÍNEZ ALONSO
Director: IVÁN IGLESIAS SÁNCHEZ

JUNIO 2023


A 1
130 bar
A 2
115 bar
A 3
100 bar
A 4
90 bar
V 1
V 2
V 3
V 4
VS1
TS5
MP 1
EPS1
V6
V7
V8
V9
V10
PS1
PS2
TS1
V11
MCS
PS3
FS1
TS2
F1
CS
PS4
SP1
MCS
F2
PS5
TS3
PS6
TS4
C1
COPS
FS2
PLANO DEL SISTEMA HIDRÁULICO

Componente	Descripción
VS1	Medidor de vibraciones del motor hidráulico primario
TSS	Indicador de variaciones en el par del motor hidráulico primario
EPS1	Medidor de la potencia del motor hidráulico primario
MP 1	Motor hidráulico primario de volumen fijo que transfiere energía mecánica a la bomba
A1. A2, A3, A4	Acumuladores hidráulicos
V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11	Válvulas hidráulicas
PS1, PS2, PS3, PS4, PS5, PS6	Sensores de presión
TS1, TS2, TS3, TS4	Sensores de temperatura
FS1, FS2	Caudalímetros
SP1	Motor hidráulico secundario de volumen fijo que transfiere energía mecánica a la bomba
C1	Refrigerador portador de fluido refrigerante
F1, F2	Filtros hidráulicos
MCS	Sistema de control maestro-esclavo
CS	Control de la válvula de apertura-cierre de paso
COPS	Sistema de control del regulador de caudal secundario

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

ÁREA INGENIERÍA INDUSTRIAL

INGENIERÍA EN SISTEMAS INDUSTRIALES

TRABAJO FIN DE GRADO
MACHINE LEARNING APLICADO AL MANTENIMIENTO PREDICTIVO DE UN SISTEMA HIDRÁULICO

PLIEGO DE CONDICIONES

Alumno: GERARDO MARTÍNEZ ALONSO
Director: IVÁN IGLESIAS SÁNCHEZ

JUNIO 2023

1- Condiciones generales

El presente proyecto consiste en la elaboración de algoritmos de aprendizaje automático orientados a la implementación de un software de mantenimiento predictivo de un sistema hidráulico. Para ello se dispone de varios conjuntos de datos que deben ser convenientemente citados debido a que la adquisición de estos ha sido realizada por la empresa alemana ZeMA. Los estudios previos han sido liderados por el científico Nikolai Helwig, quien se ha encargado de la divulgación de los datos. Los datos se han publicado en el repositorio UCI Machine learning, que es un repositorio de conjuntos de datos destinado al uso libre de estos para el desarrollo de algoritmos de machine learning.

Por tanto, el requisito legal y ético de los datos recae en la obligatoria citación de los estudios previos y de la empresa responsable de la adquisición de los datos. Los proyectos de desarrollo de algoritmos de machine learning deben contemplar de manera minuciosa la propiedad intelectual de los datos y la posibilidad del uso libre de estos con las citaciones correspondientes. La propiedad intelectual y las leyes de protección de datos desempeñan un papel fundamental en la actualidad, por lo que siempre que se usen o manipulen datos se debe comprobar las fuentes de estos, así como la naturaleza de los datos, sobre todo cuando estos son de carácter personal o privado.

En definitiva, los datos que han sido tratados en este proyecto son de carácter industrial y provienen de sensores. Han sido depositados en un repositorio de uso libre, por lo que el único aspecto legal que se ha tenido en cuenta a la hora de llevar a cabo el proyecto es la correcta citación de la fuente de adquisición de datos y estudios previos.

2- Especificaciones de Material y Equipos

El único requerimiento para poder llevar este proyecto de machine learning con un conjunto de datos de aproximadamente 100 millones es un hardware de gama media. Los requisitos mínimos del hardware son: 16 Gigabytes de RAM y un procesador con al menos 8 núcleos procesamiento principal. La tarjeta gráfica no tiene impacto ya que no se trata de un proyecto de aprendizaje profundo con imágenes. Por tanto, se ha escogido un portátil HP Victus de 16 Gigabytes de memoria RAM con 8 núcleos de procesamiento principal y 500 Gigabytes de memoria sólida.

Sin embargo, a pesar de que el hardware empleado es de gama media, es recomendable una memoria RAM de 32 Gigabytes y 16 núcleos de procesamiento principal. Esto se debe a que se han llevado a cabo algoritmos de evaluación como la validación cruzada, de optimización de hiperparámetros de los modelos. Además, se han entrenado redes neuronales que requieren un

coste computacional más alto en comparación con los demás modelos. De hecho, la optimización de hiperparámetros se ha llevado a cabo en todos los modelos excepto en las redes neuronales ya que el tiempo de ejecución de este algoritmo puede llegar a tomar bastantes horas.

Por ello, un equipo con mayor capacidad de procesamiento habría permitido optimizar aún más los modelos. Cabe destacar que se han empleado funciones de la librería Scikit-learn que permiten la ejecución de los algoritmos aprovechando todos los núcleos y memoria RAM del equipo disponible.

3- Especificaciones de Ejecución

Con el equipo definido, los tiempos de ejecución de los algoritmos han sido notablemente altos, sobre todo en las validaciones cruzadas llegando a tiempos de ejecución de 10 horas en el caso de las redes neuronales. En los demás modelos, la optimización de parámetros y validación cruzada ha llegado a tomar tiempos de ejecución que oscilaban entre los 30 minutos y las 2 horas de ejecución. Debido a esto, es recomendable el uso de un equipo más sofisticado para la realización de este proyecto con una mayor eficiencia. Sin embargo, a pesar de las dificultades con el hardware, se han conseguido unos resultados que satisfacen los objetivos marcados permitiendo una aplicación real del estudio para el mantenimiento predictivo del sistema hidráulico descrito.

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

ÁREA INGENIERÍA INDUSTRIAL

INGENIERÍA EN SISTEMAS INDUSTRIALES

TRABAJO FIN DE GRADO
MACHINE LEARNING APLICADO AL MANTENIMIENTO PREDICTIVO DE UN SISTEMA HIDRÁULICO

PRESUPUESTO

Alumno: GERARDO MARTÍNEZ ALONSO
Director: IVÁN IGLESIAS SÁNCHEZ

JUNIO 2023

Materiales

Descripción	Referencia comercial	Cantidad	Precio Unitario	Total
Ordenador Portátil Victus HP Intel i7-RTX 3050 Nvidia-16 Gb RAM-8 núcleos	HP Victus	1	1.149 euros	1.149 euros

Mano de Obra

Descripción	Cantidad	Precio Unitario	Total
Horas de trabajo	330	25 euros/hora	8.250 euros

Presupuesto total (costes directos)

Descripción	Precio total
Precio total materiales	1.149 euros
Precio total mano de obra	8.250 euros
Precio total del proyecto	9.399 euros