INSTITUTO TECNOLOGICO DE DURANGO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

“Predicción de la producción sustentable de formaldehido utilizando aprendizaje automático”

TESIS

Como parte de los requisitos para obtener el grado de

MAESTRO EN SISTEMAS AMBIENTALES

Presenta:

Ing. Miguel Ángel Carreón Félix

Director(a) de tesis:

Dr. Sergio Valle Cervantes

Codirector(a)

Dr. Rubén Guerrero Rivera

Victoria de Durango, Dgo., México Octubre, 2024

Victoria de Durango, Dgo., a 19 / Noviembre / 2024.

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
DEPI / C / 519 / 2024.
ASUNTO: Autorización de Tema de Tesis de Maestría.

C. MIGUEL ÁNGEL CARREÓN FÉLIX
No. DE CONTROL G05040564
P R E S E N T E .

Con base en el Reglamento en vigor y teniendo en cuenta el dictamen emitido por el Jurado que le fue asignado, se le autoriza a desarrollar el tema de tesis para obtener el **Grado de Maestra en Sistemas Ambientales** cuyo título es:

"Predicción de la Producción Sustentable de Formaldehido Utilizando Aprendizaje Automático"

CONTENIDO:

Sin otro asunto en particular, quedo de Usted.

A T E N T A M E N T E .

Excelencia en Educación Tecnológica®
"La Técnica al Servicio de la Patria"

C. FRANCISCO JAVIER GODÍNEZ GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN SECRETARÍA DE EDUCACIÓN PÚBLICA | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE DURANGO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

FJGG'ammc.

Victoria de Durango, Dgo., a 19 / Noviembre / 2024.

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
DEPI / C / 520 / 2024.
ASUNTO: Autorización de Impresión de Tesis de Maestría.

C. MIGUEL ÁNGEL CARREÓN FÉLIX
No. DE CONTROL G05040564
P R E S E N T E .

De acuerdo al reglamento en vigor y tomando en cuenta el dictamen emitido por el jurado que le fue asignado para la revisión de su trabajo de tesis para obtener el **Grado de Maestra en Sistemas Ambientales,** esta División de Estudios de Posgrado e Investigación le autoriza la impresión del mismo, cuyo título es:

"Predicción de la Producción Sustentable de Formaldehido Utilizando Aprendizaje Automático"

Sin otro particular de momento, quedo de Usted.

A T E N T A M E N T E .

Excelencia en Educación Tecnológica®
"La Técnica al Servicio de la Patria"

C. FRANCISCO JAVIER GODÍNEZ GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE DURANGO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

FJGG'ammc.

Durango, Dgo. a 15 de octubre de 2024

Por este conducto se hace constar que la tesis cuyo título es **Predicción de la producción sustentable de formaldehido utilizando aprendizaje automático**, del alumno **Miguel Ángel Carreón Félix**, con número de control **05040564**, estudiante del **programa Maestría en Sistemas Ambientales**, revisada por segunda vez, se sometió al software Turnitin Similarity para ser examinado, en donde el reporte del software mencionado mostró un porcentaje de 13**% de similitud general.**

En base al reporte de Turnitin, al criterio ético y profesional del director de tesis y al área del programa, se considera que **cumple** con los criterios de originalidad.

En base a lo anterior se emite el siguiente dictamen.

Protocolo. *Se autoriza el desarrollo del protocolo examinado*

Tesis. *Se autoriza dar inicio a los trámites de obtención de grado.*

Atentamente,

Dr. Sergio Valle Cervantes

Director de la Tesis

Vo.Bo.

Dr. Rubén Guerrero Rivera

Asesora

M. I. María Dolores Josefina Rodríguez Rosales

Asesor

Dr. Jaime Cristóbal Rojas Montes

Asesora

Ccp

Alumno

Coordinación del programa

LICENCIA DE USO OTORGADA POR MIGUEL ANGEL CARREON FELIX, de nacionalidad mexicana, mayor de edad, con domicilio ubicado en Calle Privada Los Trigales 107, Fraccionamiento Colinas del Saltito, Durango, Dgo., en mi calidad de titular de los derechos patrimoniales y morales y autor de la tesis denominada **"PREDICCIÓN DE LA PRODUCCIÓN SUSTENTABLE DE FORMALDEHIDO UTILIZANDO APRENDIZAJE AUTOMÁTICO"** en adelante **"LA OBRA**" quien para todos los fines del presente documento se denominará **"EL AUTOR Y/O EL TITULAR",** a favor del Instituto Tecnológico de Durango del Tecnológico Nacional de México, y con fundamento en lo dispuesto en los artículos 5, 18, 24, 25, 27, 30, 32 y 148 de la Ley Federal de Derechos de Autor , se establecen las cláusulas siguientes:

PRIMERA –OBJETO: "EL AUTOR Y/O TITULAR", mediante el presente documento otorga al Instituto Tecnológico de Durango del Tecnológico Nacional de México, licencia de uso gratuita e indefinida respecto de "LA OBRA", para almacenar, preservar, publicar, reproducir y/o divulgar la misma, con fines académicos, por cualquier medio en forma física y a través del repositorio institucional y del repositorio nacional, éste último consultable en la página: (https://www.repositorionacionalcti.mx/).

SEGUNDA - TERRITORIO: La presente licencia se otorga, de manera no exclusiva, sin limitación geográfica o territorial alguna, de manera gratuita e indefinida.

TERCERA -ALCANCE: La presente licencia contempla la autorización para formato uso de "LA OBRA" en cualquier formato o soporte material y se extiende a la utilización, de manera enunciativa más no limitativa a los siguientes medios: óptico, magnético, electrónico, virtual (red), mensaje de datos o similar conocido por conocerse.

CUARTA – EXCLUSIVIDAD: La presente licencia de uso aquí establecida no implica exclusividad en favor del Instituto Tecnológico de Durango; por lo tanto, "EL AUTOR Y/O TITULAR" conserva los derechos patrimoniales y morales de "LA OBRA", objeto del presente documento.

QUINTA – CRÉDITOS: El Instituto Tecnológico de Durango y/o el Tecnológico Nacional de México reconoce que el "AUTOR Y/O TITULAR" es el único, primigenio y perpetuo titular de los derechos morales sobre "LA OBRA"; por lo tanto, siempre deberá otorgarle los créditos correspondientes por la autoría de la misma.

SEXTA – AUTORÍA: "EL AUTOR Y/O TITULAR" manifiesta ser el único titular de los derechos de autor que derivan de "LA OBRA" y declara que el material objeto del presente fue realizado por él, sin violentar o usurpar derechos de propiedad intelectual de terceros; por lo tanto, en caso de controversia sobre los mismos, se obliga a ser el único responsable.

Dado en la Ciudad de Durango, Dgo., a los seis días del mes de diciembre de 2024.

"EL AUTOR Y/O TITULAR"

C. MIGUEL ANGEL CARREON FELIX

Resumen

En esta investigación se desarrolló un modelo de aprendizaje automático, empleando redes neuronales profundas e inteligencia artificial con el objetivo de predecir la producción de formaldehido en una planta química. El modelo se generó mediante el uso del lenguaje de programación Python y la Liberia Scikit-Learn, empleando datos del sistema de monitoreo de la planta, el cual cuenta con 220 sensores. Se determinó una estructura para la red neuronal con 4 capas ocultas, y se realizaron variaciones en el número de neuronas para cada capa, aumentando el número en cada corrida hasta obtener valores de predicción más precisos, desde 10 neuronas hasta 1,024. La arquitectura elegida fue de 1 capa de entrada con 135 neuronas, 4 capas ocultas con 128, 256, 512 y 1,024 neuronas respectivamente, y una capa de salida. Se obtuvieron predicciones con precisiones en el rango de 98.5% a 99.8% y coeficiente de determinación R^2 de 0.998 para los registros con frecuencia de 1 segundo, mostrando esto que este modelo es eficaz.

Abstract

In this research, a machine learning model was developed using deep neural networks and artificial intelligence, with the goal of predicting formaldehyde production in a chemical plant. The model was generated using the Python programming language and the Scikit-Learn library, utilizing data from the plant's monitoring system, which has 220 sensors. A structure for the neural network was determined with 4 hidden layers, and variations in the number of neurons of each layer were made, increasing the number in each run until more accurate prediction values were obtained, ranging from 10 neurons to 1,024. The chosen architecture consisted of 1 input layer with 135 neurons, 4 hidden layers with 128, 256, 512 and 1,024 neurons respectively, and one output layer. Predictions were obtained with accuracies ranging from 98.5 to 99.8% and a coefficient of determination R^2 of 0.998, for the records with a frequency of 1 second, demonstrating that this model is effective.

Índice General

Lista de Figuras

Lista de símbolos y/o Nomenclatura

Dataset	Conjunto de datos
R^2	Coeficiente de determinación
MSE	Error cuadrático medio
MAE	Error medio absoluto
RMSE	Raíz del error medio cuadrático
σ	Desviación estándar
μ	Media aritmética
x_i	Valores dentro del conjunto de datos
Σ	Sumatoria de valores
n	Numero de valores totales
y_i	Registro real del dataset

$\hat{y}_i$	Valor obtenido de la predicción
Machine Learning	Aprendizaje automático
***f*(x)**	Función matemática de un valor x
ReLU	Función de unidad lineal rectificada
ANN	Red neuronal artificial

Lista de Tablas

Dedicatoria

A mi padre†

En memoria de quien me enseñó el valor del trabajo arduo y la importancia de nunca rendirse. Gracias por seguir guiando mi camino.

A mi madre

Te agradezco por siempre creer en mí, incluso cuando yo mismo no lo hacía. Este logro es tanto tuyo como mío, y espero que sepas cuánto te amo y admiro por todo lo que has hecho por mí.

Capitulo 1 Introducción

1.1 Introducción

El objetivo principal de cualquier industria en la actualidad, independientemente del giro de sus actividades, es ofrecer productos o servicios de alta calidad al menor costo posible, por lo cual, las empresas se esfuerzan por ser parte de un proceso continuo de adaptación de tecnologías cada vez más recientes en sus procesos de producción. El desgaste o perdida de eficiencia de los componentes dentro de los procesos en una industria es un fenómeno que se presenta de manera común, y que, si no se contempla con tiempo, puede ocasionar costos excesivos de mantenimiento correctivo, bajas de rendimiento en la producción y eventualmente en pérdidas económicas. Tener en cuenta esta información es elemental para la toma de decisiones adecuadas, que permitan mantener una producción constante, reduciendo perdidas tanto de tiempo como de dinero. El uso de las tecnologías de la información en la industria ha permitido mejorar los procesos de producción, aumentar la eficiencia y eficacia de los sistemas empleados y mejorar la administración de la misma. El uso de diferentes herramientas digitales, como software de diseño, gestión y control de los procesos de una forma remota, etc., ha llevado a contar un sistema de fábricas y empresas cada vez más autónomas, conectadas entre sí mediante sistemas digitales y formando una industria inteligente. Como resultado de utilizar esos sistemas digitales podemos encontrarnos con la generación de grandes volúmenes de información, que, por su gran tamaño, complejidad y su gran velocidad de crecimiento, hacen más complicado su captura e interpretación mediante los métodos tradicionales actuales. Es este punto donde el concepto de machine learning o aprendizaje automático entra en función, como una alternativa para analizar el gran volumen de información generado. El uso de las nuevas tecnologías computacionales de análisis e interpretación de datos se han convertido en un elemento primordial para obtener mejores resultados dentro de la industria, ya sea reduciendo tiempos de producción, mejorando la calidad de los

productos o prediciendo posibles cambios en el funcionamiento de los sistemas empleados que ayudan en la toma de decisiones.

1.2 Antecedentes

El aprendizaje automático (machine learning) es una herramienta que, mediante el uso de algoritmos, modelos matemáticos y estadísticos, permite analizar e interpretar datos en tiempo real, detectando patrones y prediciendo tendencias en el comportamiento de procesos, de forma automatizada y cada vez con menor intervención de capital humano.

ARAUCO es una de las más grandes empresas forestales en América latina, líder en la producción de celulosa de Kraft de mercado, paneles y madera en diferentes presentaciones. Cuenta con 2 plantas en México ubicadas en Zitácuaro, Mich. (19°29'03.1"N 100°23'20.5"W), y Durango, Dgo. (24°04'54.1"N 104°40'02.6"W), donde producen una variedad de resinas ureica, fenólica, melamínica, melamina-urea-formaldehido, así como una variedad de tableros de aglomerado y MDF. Uno de los procesos primordiales dentro de la empresa, es la producción de formaldehido mediante el uso de catalizadores de hierro-molibdeno. Es este proceso en donde se presenta un problema con la eficiencia de la producción, al no contar con programas de mantenimiento debidamente establecidos, debido a la falta de conocimiento con precisión de los tiempos de envejecimiento y ciclos de vida en la producción.

Por lo cual la empresa busca, mediante un proyecto de investigación, identificar los periodos de tiempo ideales aplicables a sus sistemas, para lograr una producción sustentable de formaldehido. La obtención de dicha información ayudará a que se reduzcan los tiempos de inactividad en la planta, programando con tiempo suficiente los mantenimientos necesarios logrando una operación sustentable de la planta, confiable y con mayor costo-eficiencia.

1.3 Justificación

El proceso de producción de formaldehido es una de las etapas más importantes dentro de la planta química de la empresa ARAUCO, debido a que su producción

requiere llevar un seguimiento cuidadoso del estado de envejecimiento del catalizador. Al producirse una disminución de la eficiencia, es necesario emplear una mayor cantidad de calor para mantener el proceso trabajando a un mismo ritmo, lo cual implica que se genere una mayor afectación al medio ambiente, debido al incremento en la producción de CO_2 y GEI en la atmosfera, así como de residuos de manejo especial.

Para llevar este control, ARAUCO recurre a una empresa que cuenta con software especializado en la predicción del desgaste del catalizador, que es utilizado como método de prevención y de programación de suministro de más catalizador para la planta, con la desventaja de que deben de estar pagando por el uso de esa licencia, al no existir otra opción en el mercado.

1.3.1.1 Impacto Ambiental

Reducir los GEI emitidos por la planta ARAUCO S.A. de C.V. Estos se producen debido al incrementarse el flujo de combustible y algunos otros insumos, para evitar que disminuya la eficiencia del catalizador. En el aspecto de generación de conocimiento científico, se podrá contar con un modelo de aprendizaje automático que podrá ser utilizado en diferentes procesos de distintas plantas químicas que lo requieran. Considerando que la predicción del comportamiento de la planta ayuda directamente a la elaboración del programa de control preventivo, así como en la reducción de la generación de contaminantes de GEI a la atmosfera, se considera que es congruente con la línea de investigación establecida de *análisis y control de sistemas ambientales.*

1.4 Objetivo General

Predecir la producción de formaldehido utilizando aprendizaje automático (machine learning) a través de la actividad del catalizador de fierro-molibdeno utilizado en el proceso de obtención de formaldehido (ARAUCO S.A. de C.V.), utilizando los datos capturados por el sistema de monitoreo, para definir si se continua con la misma carga de catalizador o se adiciona una carga nueva.

1.5 Objetivos Específicos

• Examinar, categorizar y limpiar la información obtenida del sistema de monitoreo de producción de formaldehido.

• Establecer mediante Python y aprendizaje automático (machine learning), un modelo que permita predecir el comportamiento de la producción de formaldehido de la planta de la empresa ARAUCO S.A. de C.V.

• Comparar los resultados obtenidos en el modelo de aprendizaje automático con los datos reales de la planta de producción de formaldehido para verificar el funcionamiento del modelo.

• Estimar la producción futura sustentable de formaldehido de la planta y proponer se utilice una carga nueva o repetir la carga del catalizador existente.

1.6 Descripción del contenido de los capítulos

En el primer capítulo se hablará sobre los antecedentes existentes de trabajos similares y/o relacionados al tema de esta investigación, indicando el origen, la metodología empleada y los resultados obtenidos. Antecedentes en la utilización de los modelos de aprendizaje automático (Machine Learning) para realizar predicciones de diferentes procesos, empleando redes neuronales profundas (DNN), la justificación de este proyecto y los objetivos trazados para lograrlo.

En el capítulo dos se hablará sobre el formaldehido, sus características principales y su importancia en la industria química y sus principales aplicaciones, además se mencionarán conceptos importantes como el Aprendizaje Automático (Machine Learning) e Inteligencia Artificial (IA)

En el capítulo tres se desarrolla la metodología empleada para la obtención del modelo de para realizar predicciones en procesos dentro de una planta química empleando Redes Neuronales Profundas (DNN), además de los algoritmos y librerías empleadas para su obtención.

En el capítulo cuatro se presentarán los resultados obtenidos al procesar datos reales de la planta química en la producción de formaldehido, las predicciones obtenidas y como este tipo de modelos ayudan en la toma de decisiones y en la reducción de emisiones de contaminantes al medio ambiente.

En el capítulo cinco, se muestran las conclusiones y recomendaciones sobre el uso de Inteligencia Artificial y Aprendizaje Automático para predecir el comportamiento de los distintos procesos en la planta química, así como posibles áreas de oportunidad y de mejora que se pueden conseguir si se da seguimiento a estas técnicas.

Capitulo 2 Marco Teórico

2.1 Estado del Arte

La aplicación del machine learning enfocado a procesos productivos reales o mejoras en la industria son un área relativamente nueva, por lo que, en procesos químicos dentro de la industria existen pocos antecedentes.

Dodhia et al. (2021) desarrollaron un modelo basado en machine learning de control predictivo de procesos de reacción-difusión, el cual fue diseñado para sistemas de ecuaciones diferenciales parciales (PDE) no lineares (parabólicas) con datos de mediciones de una serie de tiempo de un proceso, obteniendo resultados con errores menores a 2%, con lo cual corroboraron que el modelo funciona.

Bogojeski et al. (2020) evaluaron una amplia variedad de modelos enfocados en datos, utilizando modelos tradicionales como son regresión lineal, regresión ridge del kernel y redes neuronales prealimentadas (feed-forward, en inglés) mediante el uso del software Python, comparándolo con modelos más complejos de redes neuronales recurrentes.
El método empleado para este caso fue determinar cuanta información histórica era necesaria para poder entrenar cada uno de los modelos con una base de datos de la cual se conocía su dinámica de funcionamiento, para posteriormente ser probados con datos reales de una planta química de grande escala. Los resultados obtenidos mostraron que los modelos recurrentes producen predicciones cercanas a la perfección cuando se entrenan con datasets grandes y mantienen un desempeño bueno cuando se entrenan con datasets pequeños, mientras que los modelos más simples solo obtienen resultados comparables a los obtenidos de datasets pequeños.

Schweidtmann et al. (2021) realizaron una evaluación del estado actual del uso de machine learning en procesos relacionados con la ingeniería química, logrando identificar los 6 retos principales para resolver problemas mediante machine learning

en procesos químicos. Los aspectos identificados fueron: la toma óptima de decisiones, la aplicación de física en machine learning, la representación de la información y el conocimiento existente, la heterogeneidad de los datos, la seguridad y confianza en los procesos de machine learning y la creatividad. Haciendo énfasis en que, de no adaptar los métodos computarizados de inteligencia artificial, así como el uso de energías renovables, convertirán a la industria química de producción en el principal consumidor de petróleo para el año 2030.

Stock et al. (2022) desarrollaron un modelo con base en regresiones lineales y redes neuronales profundas para determinar de manera temprana la calidad de las celdas de baterías de ion de litio, así como su clasificación. Para alimentar su modelo emplearon información de 24 características obtenidas de pruebas de espectroscopia de impedancia electroquímica y datos ciclados del proceso de producción de las baterías. La estructura de la red neuronal artificial (ANN) que elaboraron, se conformó por 1 capa de entrada, 5 capas ocultas de 40, 32, 24 16 y 8 neuronas respectivamente, y finalmente 1 capa de salida con 2 neuronas para lograr la clasificación deseada. Logrando obtener un modelo que logró un error en el proceso de prueba de 10.1% con tiempos de observación de menos de 2 días, así como también en el proceso de clasificación en 2 distintos grupos de ciclo de vida obtuvieron una precisión máxima de 97%. Estos resultados son de gran importancia ya que ayudan a mejorar los procesos para incrementar el volumen de baterías elaboradas, así como en general mejorar la calidad de las celdas.

Ravindiran et al. (2023) formularon un modelo de aprendizaje automático para predecir la calidad del aire a través del índice de calidad del aire (AQI, por sus siglas en inglés) en la ciudad costera de Visakhapatnam en la India, enfocándose en 12 contaminantes y observando 10 parámetros meteorológicos con datos desde julio de 2017 hasta septiembre de 2022. El enfoque empleado en este proyecto se basó en modelos como LightGBM, Bosque Aleatorio, Catboost, Adaboost, y XGBoost. Logrando obtener los mejores resultados a través del modelo Catboost, los cuales

obtuvieron valores del coeficiente de determinación R^2 de 0.9998, un error medio absoluto (MAE) de 0.60, y un error medio cuadrado (MSE) de 0.58.

Davies et al. (2023) empleando aprendizaje automático realizaron predicciones de la producción de reacciones químicas de Buchwald-Harwig con datos obtenidos de sensores *in situ,* los cuales medían temperaturas, presión y color. Empleando la información obtenida de 12 sensores, lograron obtener resultados en las predicciones con un error absoluto medio (MAE) de 1.2%, además, lograron predicción con tiempos de anticipación de 30, 60 y 120 minutos, con errores de 3.4, 4.1 y 4.6% respectivamente. Los modelos que emplearon fueron a base de regresión lineal, regresión cubica polinomial, bosque aleatorio, regresión de potenciación del gradiente y redes neuronales de memoria corto plazo. Todo esto desarrollado mediante el software de uso libre Python.

Solaimany-Aminabad et al. (2013) desarrollaron un modelo confiable de pronóstico para plantas de tratamiento de aguas residuales de la ciudad de Sanandaj, Irán, para predecir la calidad de agua del influente y poder usar esta información como base en la operación del proceso. Empleando redes neuronales artificiales (ANN) con Feed-Forward, Back-Propagation, redes de auto regresión no lineal, con datos recolectados en un periodo de 2 años para la elaboración del modelo. Obtuvieron resultados de las predicciones con coeficiente de correlación (R^2) de 0.93. Los valores medidos fueron la alcalinidad, pH, calcio, dióxido de carbono, temperatura, dureza total, turbidez, sólidos disueltos totales y conductividad eléctrica. Empleando el coeficiente de determinación R^2 como verificación obtuvieron valores en el rango de 0.86 para la alcalinidad y hasta 0.54 para la conductividad eléctrica.

Lee (2003) desarrolló un sistema inteligente llamado I-PreConS, para obtener resultados in situ sobre la resistencia del concreto, con el objetivo de facilitar el retiro de cimbras y moldes y apoyar en la programación en la construcción. Se empleó un sistema a base de redes neuronales artificiales (ANN) alimentado con datos de

resultados de pruebas de resistencia de cilindros de concreto, así como de patrones de entrenamiento. Para elaborar el modelo, encadenaron 5 redes neuronales sencillas, ejecutando el modelo durante un periodo de 24 horas, y logrando predecir la resistencia a la compresión del concreto al séptimo, y vigésimo octavo día del colado. Como datos de entrada para la red, se ingresaron 73 registros del concreto, incluyendo temperatura, humedad, proporciones de los agregados, hora de colado, día de colado, tipo de cemento, entre otros. El modelo obtuvo resultados en de coeficiente de determinación (R^2) de 0.90 y hasta 0.98, con lo cual demostraron su eficacia.

Wu & Lo (2008) elaboraron un modelo para determinar la dosificación óptima de coagulante para el tratamiento de aguas residuales, mediante el uso de redes neuronales artificiales (ANN) y sistemas de inferencia neuro-borroso adaptativo, logrando modelar la dosificación de cloruro de polialuminio para el agua superficial del norte de Taiwán. Las redes se entrenaron con información de 819 procesos, entre los cuales se controla la temperatura, turbidez, color y pH. Se utilizaron datos de agua en diferentes etapas, tanto de agua cruda, como de agua floculada, agua sedimentada, y agua previamente tratada en un depósito. La estructura considerada para la red fue de 1 capa de entrada con 2 neuronas, 3 capas ocultas con 6, 9 y 9 neuronas respectivamente, y una capa de salida con una neurona. Los resultados obtenidos para ANN fueron superiores a los del sistema de inferencia neuro-borroso adaptativo, con valores de coeficiente de determinación R^2 superiores a 0.85, después de 7 corridas. El modelo óptimo para predecir la dosificación de coagulante fue el que consideró la turbidez del agua cruda y la dosificación de días anteriores de coagulante, presentando un valor de R^2 de 0.8685 y con error cuadrático medio (RMSE) de 0.00424.

Martín et al. (2006) diseñaron una red neuronal artificial (ANN) para predecir si la calidad de las uniones soldadas por resistencia por puntos cumple o no con determinado nivel de calidad. Emplearon como datos de entrada el tiempo de

soldadura, el tipo de electrodo y la intensidad de corriente utilizada. La gran cantidad de puntos de soldadura realizados en la industria automotriz fue la razón por la cual se decidió realizar esta investigación, con un rango de entre 3,000 y 4,000 puntos de soldadura por cada vehículo. Se entrenó la red con datos de 225 puntos de soldadura, cada uno con su respectivo par de entrada y salida *i/t,* se propusieron 7 diferentes estructuras de red neuronal todas con 3 neuronas de entrada, 2 capas ocultas con neuronas en incrementos de 1, desde 1 hasta 7 neuronas, y con una neurona de salida. Se verificó la eficiencia del modelo mediante el uso del error cuadrático medio (MSE), con resultados desde 0.0955 para la primera red y de 8.70E-17 para la séptima iteración.

Benne et al. (2000) analizaron mediante redes neuronales el comportamiento de un molino de caña de azúcar, para predecir los múltiples efectos de la evaporación en la industria de la caña. Emplearon datos obtenidos de medidas realizadas en varias campañas en el molino de azúcar de Bois Rouge. La evaporación es el primer paso en el proceso de concentración de sacarosa. Los datos fueron obtenidos del sistema de control MODUMAT 8000®, el cual se tenía implementado en la planta. Se revisaron 3,000 muestras de la base de datos del año 1998 que representan aproximadamente 20 horas de datos de operación de la planta, con resultados de error relativo entre la predicción y el registro real menor al 5%, que se encuentra dentro del parámetro aceptable para las condicione de la planta.

Bombela Jiménez et al. (2019) mediante el uso de redes neuronales con entrenamiento supervisado y de tipo back propagation, obtuvieron predicciones una hora despues sobre el estado de la calidad del aire, empleando datos obtenidos de la estacion CICEG de la ciudad de Leon, Guanajuato. Se midieron cinco contaminanates como parametros de referencia: NO_2, PM_{10}, SO_2, O_3 y CO, considerando tambien el mes, el dia y la hora de la medicion de cada contaminante. Los datos obtenidos de la base datos del Sistema Estatal de Información de Calidadl del Aire (SEICA), cuenta con 35,040 registros de un periodo de 3 años, de enero de

2017 a diciembre de 2019. La estructura de la red neuronal utilizada fue de 15 neuronas de entrada, 1 capa oculta de 10 neuronas y 5 neuronas como capa de salida. Se obtuvieron resultados de la raiz del error cuadratico medio (RSME) de 0.1488, 0.1144, 0.1406 y 0.1654 para los años 2014, 2015, 2016 y 2017 respectivamente.

Fauzi et al. (2019) desarrollaron un modelo mediante algoritmos de redes neuronales artificiales (ANN) en conjunto con el software Alteryx, para obtener un análisis predictivo del flujo de trabajo sobre la vida útil restante en la industria del petróleo y el gas, gracias a su capacidad de trabajar con grandes cantidades de información y su alta precisión. Emplearon un dataset de 8,760 columnas y 3 filas, que comprende un año de datos de una empresa multinacional de petróleo y gas, logrando obtener predicciones con valores de RSME de 0.003.

Kumru & Kumru, (2014) realizaron un estudio en una planta de producción de autopartes de repuesto, en la cual existía la problemática de no poder cumplir a tiempo las requisiciones de los clientes, debido a la ausencia de datos de tiempos de operación para las partes ordenadas con diferentes especificaciones. Se comparó el resultado obtenido de las predicciones contra datos de modelos de regresión lineales y no lineales, determinando que los datos obtenidos de las ANN fueron los más precisos. Como datos de entrada consideraron como la dureza o suavidad del material, el tipo de material ya sea cobre, hierro o acero, y el área total por trabajar de cada pieza. Se probaron distintas combinaciones de estructura de la red neuronal, desde 0 hasta 6 capas ocultas, para tratar de obtener el mejor modelo posible, logrando obtener con la mejor configuración valores de MSE de 0.153404, siendo este método más efectivo que los de los modelos lineales o no lineales.

Valente et al. (2014) obtuvieron predicciones precisas mediante el uso de ANN, con el modelo que realizaron para predecir la demanda química de oxígeno después de un tratamiento de electrocoagulación en efluentes de la industria láctea.

Consideraron como parámetros de entrenamiento la densidad de corriente eléctrica, pH inicial del efluente, contenido de sólidos, turbidez y demanda química de oxígeno, logrando obtener coeficientes de correlación de 0.96 entre los valores predichos y los valores obtenidos de experimentos. Los datos se obtuvieron de una muestra de desagüe de los 15,000 litros diarios de lácteos por día que se procesan en la planta, usando 275 muestras recolectadas en intervalos de 1 hora durante un tiempo de 8 a 17 hora continuas. Usando el método de prueba y error determinaron que el número de capas ocultas fuese variando entre 1, 2, 5, 10 y hasta 20, observando que, a mayor número de neuronas en estas capas, disminuía el valor del MSE, pero, con 20 o más neuronas, el valor del error R disminuía. Definiendo 10 como el número seleccionado para la estructura, ya que otorgaba los mejores valores de predicción en comparación con los resultados reales de los experimentos.

Azadeh et al. (2008) utilizaron el enfoque de redes neuronales para analizar el alto consumo de energía en distintos sectores industriales como la química, la metalurgia y minerales no metálicos. Mediante el uso de un perceptrón multicapa supervisado (MLP) para estimar el consumo anual con el menor error posible. Los datos empleados se obtuvieron de varias empresas de Irán, de los años 1979 a 2003, consideraron como datos de entrada el precio de la electricidad, el número consumidores de cada sector, el precio medio del combustible fósil, la intensidad de electricidad para industrias de alto consumo y el consumo anual en cada sector. Las arquitecturas consideras para la red neuronal fueron de 0, 1 y 2 capas ocultas, obteniendo como resultado predicciones con valores de porcentaje medio absoluto de error (MAPE) de 0.0099 en la red más grande y hasta 0.035 para la red más pequeña.

Eren et al. (2012) desarrollaron un modelo para predecir la tasa de rechazo de la sal (NaCl) por nanofiltración, basado en datos experimentales, analizando el impacto que se tenía bajo condiciones de operación normal, al variar la presión de alimentación, el pH y la velocidad de flujo cruzado, entre otros. La estructura de la

red se desarrolló a base prueba y error hasta obtener la arquitectura óptima, formada por una capa de entrada con 5 neuronas, una capa oculta con 25 neuronas, y una capa de salida con una neurona empleando un algoritmo de Back Propagation. Se revisó la precisión de las predicciones mediante el coeficiente de determinación (R^2), el error cuadrático medio (MSE) y raíz del error cuadrático medio (RSME) obteniendo valores en la etapa de prueba de 0.98, 0.024 y 0.035 respectivamente. Concluyendo que este tipo de modelos pueden proveer información de gran importancia para la optimización de procesos, así como para reducir el número de experimentos realizados, lo cual representa a su vez un menor costo.

Dentro del sector de la minería, se ha podido aplicar de igual forma la inteligencia artificial y el aprendizaje automático, como se muestra en la investigación realizada por Siami-Irdemoosa & Dindarloo (2015), quienes mediante redes neuronales realizaron predicciones sobre el consumo de combustible en camiones de volteo empleados en este sector, cuyo impacto es de alrededor del 30% del total de la energía usada en las minas superficiales, de igual forma son la principal fuente generadora de gases de efecto invernadero (GEI). En esta investigación consideraron como variables la carga total, el tiempo de carga, el tiempo en espera al cargar, tiempo total cargado, tiempo total vacíos y tiempo en espera vacíos. La salida de la red, o la predicción fue la cantidad de combustible consumido en un ciclo completo. Obteniendo porcentaje de error medio absoluto de 10% y determinando que la variable con mayor influencia en el consumo de combustible es el tiempo de espera al cargar. Este resultado permitió identificar que parte del ciclo requería mayor atención y mejor programación para evitar tiempos de espera innecesarios.

Camarena Martínez et al. (2023) aplicaron un método hibrido de optimización por enjambre de partículas (PSO) y de redes neuronales artificiales (ANN) para predecir fugas en biodigestores, con una estructura de 3 neuronas en la capa de entrada, 3 más en la capa oculta y 1 neurona como capa de salida, se logró obtener resultados

de coeficiente de determinación (R^2) de 0.967, superiores al uso únicamente de la ANN cuyo valor de dicho coeficiente fue de 0.94.

Kermet-Said et al. (2024) desarrollaron un modelo para predecir la eficiencia de remoción de la Demanda Química de Oxígeno (DQO) y Sólidos Suspendidos (SS) de agua de desecho de una planta de tratamiento de una empresa farmacéutica, usando redes neuronales multicapa. Para esta investigación utilizaron datos recolectados en un periodo de 4 años. Se elaboró una red de dos capas ocultas con 8 neuronas, y obteniendo resultados de R^2 de 0.9783 y error cuadrático medio (MSE) de 0.001695, por lo cual se considera como un buen método de predicción.

Guo et al. (2023) elaboraron un modelo para obtener predicciones diarias de partículas $PM_{2.5}$, en la ciudad de Shanghái en China, debido a los grandes decrementos en esta variable durante el periodo de 2014 a 2020 de alrededor de 39%. Por lo cual requerían identificar cuales factores influían más en este incremento, y utilizaron 14 elementos meteorológicos para el diseño como la temperatura atmosférica mínima, presión atmosférica máxima, temperatura atmosférica máxima. El dataset utilizado fue de enero del 2014 a diciembre de 2020, obtenido de 20 estaciones de monitoreo en Shanghái. Con una arquitectura definida para la red considerada como la óptima, a base de prueba y error, de 17 neuronas de entrada, 15 en capas ocultas y 1 neurona de salida. Logrando valores de (R_2) como método de verificación de la eficiencia de 0.9852.

Kiiza et al. (2020) mediante el uso de ANN, implementaron un modelo para obtener predicciones de remoción de contaminantes en humedales construidos para el tratamiento de aguas pluviales. Usaron como base un dataset con registros de dos años (2014-2016) de 8 diseños piloto de humedales verticales ubicados en el techo del edificio sur de la Escuela de Ingeniería de la Universidad de Cardiff. De estas 8 unidades, solo se consideraron viables los datos recolectados de 6, debido a la intermitencia en las lecturas obtenidas de los 2 restantes. Se enfocó el diseño en

obtener principalmente las reducciones de nutrientes de fósforo y de nitrógeno, con resultados de R^2 en rango desde 0.25 hasta 0.71 para las distintas configuraciones de red para el nitrógeno y valores de R^2 en rango de 0.73 hasta 0.83 para el fósforo, considerando estos resultados como satisfactorios y comprobando que las redes neuronales son una herramienta útil para este tipo de experimentos.

También dentro del área de contaminación al medio ambiente se considera la investigación realizada por Adams et al. (2020) en la que mediante redes neuronales profundas (DNN) obtuvieron un modelo para predecir emisiones de SOx y NOx asociadas con la conversión de carbón para la producción de energía. El modelo se entrenó con datos de una planta comercial obteniendo coeficientes de eficiencia de 0.8925 y 0.9904 para SOx and NOx respectivamente. Los datos se registran por el sistema centralizado captura de la planta de energía, donde se controlan parámetros como flujos, presión, temperatura, composición, propiedades de los materiales, entre otros, se seleccionaron registros de septiembre de 2017 a mayo de 2019, con rango de lecturas cada 1 minuto, para formar un dataset inicial de 25,064 registros de 380 variables. Se estableció como parámetro para verificar la eficacia del modelo mediante el error medio absoluto (MAE) y la raíz del error cuadrático medio (RSME). Se empleo una red profunda con estructura de 64 neuronas en la capa de entrada, 32, 16 y 4 para las 3 capas ocultas, y 1 neurona para la capa de salida, realizando diferentes variantes de los datos que se emplearon como entrada, usando en cada variación distintas propiedades de los materiales empleados en la planta, logrando obtener valores de RSME de 4.551ppm y de MAE 2.942ppm.

Otra de las áreas donde se comienza a utilizar la inteligencia artificial como herramienta de apoyo para la toma de decisiones es la industria médica (Rodríguez Zúñiga & Pérez Esparza, 2024), de manera muy clara como a base de redes neuronales artificiales (ANN) se puede obtener una detección más temprana y oportuna del cáncer de mama, identificando patrones obtenidos de datos públicos del

Gobierno de México sobre estudios realizados a pacientes diagnosticados en los años 2021 y 2022.

Bautista-Loaiza & Ávila-Camacho (2023) propusieron un modelo de inteligencia artificial que permite predecir si un paciente infectado con Covid-19, tomando en cuenta sus síntomas, requerirá de hospitalización, empleando registros de más de 13 millones de pacientes de México, así como las principales 12 características que afectan en la evolución de dicha enfermedad, como son la edad, sexo, si tiene diabetes o no, si presenta hipertensión o no, entre otros. Con una red neuronal artificial estructurada con 3 capas, con 12 neuronas en la de entrada, 24 neuronas en la capa oculta y una neurona de salida, y mediante un Análisis ROC, para evaluar la precisión de las predicciones obtenidas lograron obtener resultados de precisión de 80%. Considerando este modelo como una herramienta útil para ser utilizada en pacientes futuros.

De igual forma en la agricultura se emplea el aprendizaje automático y la inteligencia artificial, para la detección de enfermedades en cultivos de maíz, analizando imágenes y redes neuronales convolucionales (CNN) (Ruiz Tamayo et al., 2024). Se realizó un estudio en la zona del “Bajío” de México, identificando las enfermades más comunes que se presentan en los cultivos de maíz como son la roya de maíz, manchas foliares o tizón, plagas y virus del rayado del maíz, recolectando imágenes tomadas con equipo especializado, en exteriores y con luz natural solar, para formar un dataset de 400 imágenes para entrenar el modelo y 40 fotos nuevas para probar la eficacia en la detección de enfermedades. Se estructuro la red convolucional con 144 capas y 170 conexiones, teniendo además 11 unidades de procesamiento de entrada. Los resultados obtenidos después de 10 épocas de entrenamiento fueron de un 85.32% de precisión en la detección de enfermedades en hojas de maíz.

2.2 Formaldehido

El formaldehído es un gas incoloro que tiene la característica de ser inflamable a temperatura ambiente, y tener un olor muy característico, en niveles altos puede producir una sensación de ardor en los ojos, los pulmones y la nariz. Es también conocido como metanal, óxido de etileno, oximetileno, aldehído metílico y oxometano (ATSDR, 1999).

El formaldehído es uno de los compuestos básicos más importantes de la industria química, se emplea para elaborar las resinas de formaldehido que sirven como adhesivos en los tableros y paneles de madera. Es una sustancia orgánica natural que está presente en la mayoría de los organismos vivos, incluido el cuerpo humano, generándose en pequeñas cantidades como parte de los procesos metabólicos normales (Fig. 2.1). Se encuentra de forma natural en el aire que respiramos; se metaboliza rápidamente por lo que no se acumula en el cuerpo.

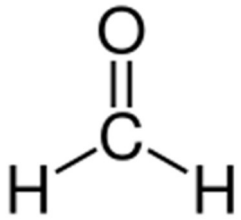

Fig. 2.1 Formula química del formaldehido

Se puede producir formaldehido mediante oxidación parcial de metanol y aire, empleando catalizadores y algunos óxidos metálicos como el molibdeno o hierro. Dos de los factores primordiales dentro de este proceso son la presión y la temperatura del catalizador que se emplee, así mismo es importante que las cantidades de formaldehido, aire y metanol estén rigurosamente controladas para permanecer sin rebasar los límites de explosión en sus mezclas. El calor generado en el reactor se logra mantener mediante fluidos térmicos sintéticos (Dowtherm), los cuales tienen propiedades que les permiten utilizarse para transferencias de calor en altas temperaturas, y son suministrados dentro del condensador de transferencia de calor. Dentro del reactor también es necesario suministrar aire fresco, mediante

sopladores que lo mezclan con el aire de recirculación utilizando un turbo cargador. Al interior del vaporizador se introduce el metanol por la parte superior, para ser vaporizado, y en la parte inferior se calienta el gas de proceso. Una vez que el metanol oxidado alcanza su máxima temperatura es cuando se convierte en formaldehido (Fig. 2.2)

Al reaccionar la mayor parte del metanol, se presenta un descenso en la temperatura y por los tubos del reactor emerge el gas, los cuales pasan al vaporizador, para ser enfriados a una temperatura de 140°C aproximadamente y pasar a la torre de absorción. Posteriormente se realiza una circulación del formol, el cual se remueve mediante un controlador de nivel en la parte inferior del reactor para obtener el producto final, la solución de formaldehido (De Casas Reyes, 2022).

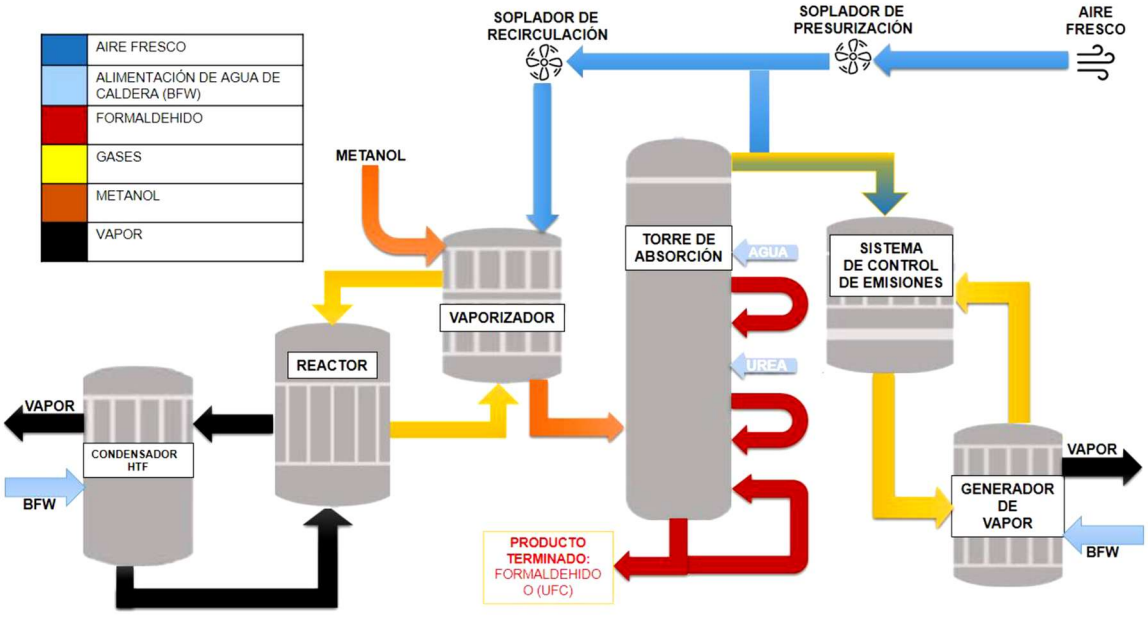

Fig. 2.2 Proceso de producción de formaldehido en Planta Arauco

2.3 Machine Learning

Machine learning es la ciencia que hace que los ordenadores “aprendan” a partir de los datos y la información. En vez de programar, paso a paso, cada solución específica para cada necesidad planteada, tal y como se realiza en el enfoque de la programación convencional, el área de machine learning está enfocada en el desarrollo de algoritmos genéricos que pueden extraer patrones de diferentes tipos de datos; (Bogojeski et al., 2020; Bovadilla, 2020). En machine learning, los datos son la base fundamental de todo; si no se cuenta con suficientes datos, éstos no son representativos o presentan información sesgada, no se podrá llegar a generar un aprendizaje automático. Cuando la cantidad de datos es insuficiente, los algoritmos de machine learning no pueden generalizar los resultados, simplemente aprenden los patrones de las muestras existentes. Incluso si disponemos de suficiente cantidad de datos, éstos podrían no ser aceptables para algunos propósitos específicos de machine learning si no son representativos o están sesgados (Bovadilla, 2020). Existen distintas formas de emplear el aprendizaje automático, según su clasificación, como se muestra en la Fig. 2.3.

Las clasificaciones más comunes dentro del machine learning son:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje semi-supervisado
- Aprendizaje por refuerzo Stock et al. (2022)

Fig. 2.3 Tipos de aprendizaje automático

El aprendizaje supervisado en machine learning se aplica cuando cada dato, o conjunto de datos de entrada (muestra) está ligado a una etiqueta. Partiendo de este conjunto de datos se pueden usar diferentes algoritmos de clasificación de machine learning con el objetivo de entrenar un modelo y poder, al acabar el entrenamiento, predecir la etiqueta correspondiente a una nueva muestra. El modelo puede ser tan simple como la solución lineal que mejor ajuste las muestras de origen a los valores objetivo, o mucho más complejo, como la búsqueda de factores ocultos que representen la información más importante que está contenida en los datos (Bovadilla, 2020).
Dentro de este tipo de aprendizaje podemos distinguir dos tipos principales: Regresión y clasificación, como se observa en la Fig. 2.4.

Mediante la regresión se puede predecir un valor continuo, utilizando variables de entrada, es decir, la relación entre variables dependientes y variables independientes. Dependiendo del número de variables independientes empleadas, será el tipo de regresión encontrada, siendo esta del tipo simple lineal cuando se tiene solo una variable, y regresión lineal múltiple o regresión no lineal (polinómica) cuando se tienen dos o más variables. En el caso del tipo clasificación, se busca que el valor que se predecirá sea del tipo discreto, asociado a una categoría o instancia en específico, previamente establecida.

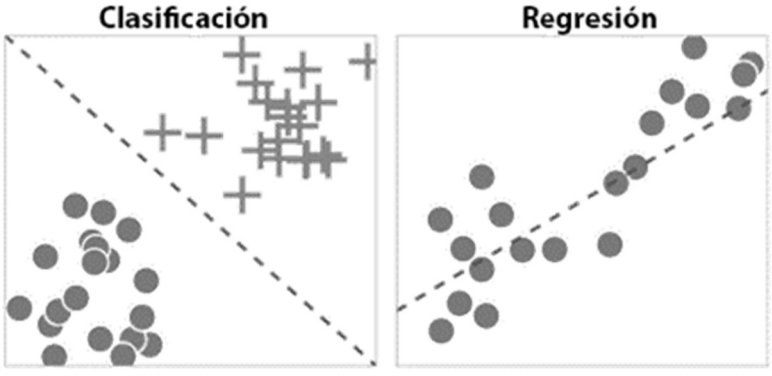

Fig. 2.4 Ejemplos de aprendizaje supervisado

El aprendizaje no supervisado utiliza información no etiquetada, por lo cual busca encontrar la relación que hay entre los datos considerando las características en tengan en común. Los dos tipos principales dentro de este tipo de aprendizaje son el agrupamiento (Clustering, en inglés) y la reducción de la dimensionalidad. El objetivo de la técnica de clustering es agrupar muestras como se aprecia en la Fig. 2.5, mediante el uso del algoritmo k medias (k-means, en inglés), el cual crea clústeres o grupos de datos con base en criterios de distancia o similitud. Par el caso de la reducción de dimensionalidad, los algoritmos empleados tienen como objetivo principal reducir el número de variables, y proyectarlas en espacio donde los datos estén más condensados y permitan obtener resultados con una mayor precisión. Este método tiene la ventaja de que, al reducir la cantidad de información procesada, disminuye la carga computacional que se requeriría si se trabajara con el total de las características.

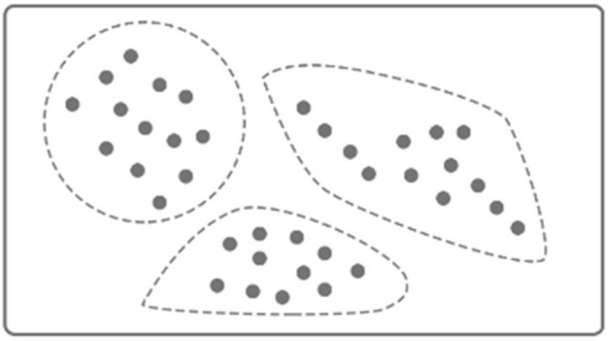

Fig. 2.5 Ejemplo de clustering en aprendizaje no supervisado

El aprendizaje semi-supervisado trata con conjuntos de datos en los que una porción de los datos está etiquetada y el resto no. Normalmente, la cantidad de muestras etiquetadas es mucho más pequeña que las no etiquetadas. La mayoría de los algoritmos de aprendizaje semi-supervisado son una mezcla de métodos supervisados y no supervisados (Bovadilla, 2020).

Considerando lo mostrado en la Fig. 2.6, en el aprendizaje por refuerzo, el resultado se obtiene a base de ensayo y error, y es un área innovadora y con un gran futuro, ya que está inspirada en mecanismos naturales. En este caso, el algoritmo de

aprendizaje recibe información de un entorno real o simulado, y cuando el sistema realiza una acción es recompensado o penalizado, tal y como pasa con los seres vivos. Este tipo de aprendizaje utiliza la retroalimentación (feedback, en inglés) como su principal elemento. Tales algoritmos de aprendizaje se denominan agentes y pueden aprender siguiendo los principios de la evolución natural. Los agentes aprenden estrategias, denominadas “políticas”, que maximizan las recompensas y minimizan las penalizaciones (Bovadilla, 2020).

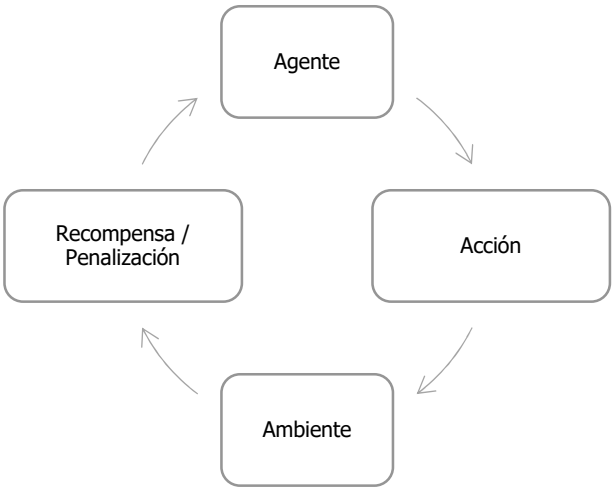

Fig. 2.6 Ciclo de aprendizaje automático por refuerzo

2.4 Redes Neuronales

Los elementos más empleados para el desarrollo de modelos aprendizaje automático son las redes neuronales artificiales, las cuales buscan imitar el comportamiento de las neuronas de cerebro humano, por lo cual es importante conocer el funcionamiento de una neurona biológica y su relación con una neurona artificial.

2.4.1 Neuronas biológicas

Las neuronas biológicas existen dentro del cerebro humano en cantidades aproximadas de 80 mil millones, y aproximadamente 100 millones de millones de

conexiones entre sí. Dentro de las partes que las conforman, el núcleo es el elemento primordial, ya que es el encargado de procesar la información que recibe, la cual llega mediante las dendritas, que se conectan a otras neuronas o nervios del cuerpo humano. Esta información que se procesa es transmitida a lo largo del axón y es dirigida hacia otras neuronas o partes del cuerpo humano por medio del elemento terminal llamado sinapsis (Fig. 2.7). Todo este procesamiento de información, desde su recepción hasta su salida hacia otra neurona, tiene una velocidad aproximada de 1 a 2 milisegundos (Bovadilla, 2020)

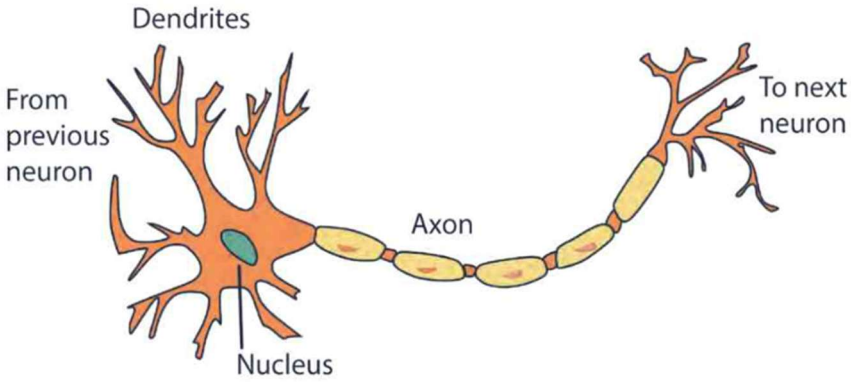

Fig. 2.7 Estructura de una neurona biológica

2.4.2 Neuronas artificiales

Aprovechando el rápido crecimiento en las capacidades de procesamiento de los equipos de cómputo actuales, surge la idea de crear redes de neuronas artificiales, conectadas entre sí y capaces de procesar información a grandes velocidades y con alta precisión.

En la Fig. 2.8 se muestra como la estructura considerada para una neurona artificial es similar a la de su contraparte biológica, formada por una entrada de información similar a las dendritas, un núcleo de procesamiento y una salida o resultado equivalente al axón en la parte biológica, además de un valor de inhibición o excitación entre las neuronas que equivale al efecto que existe entre la sinapsis que conecta las neuronas biológicas (Bovadilla, 2020).

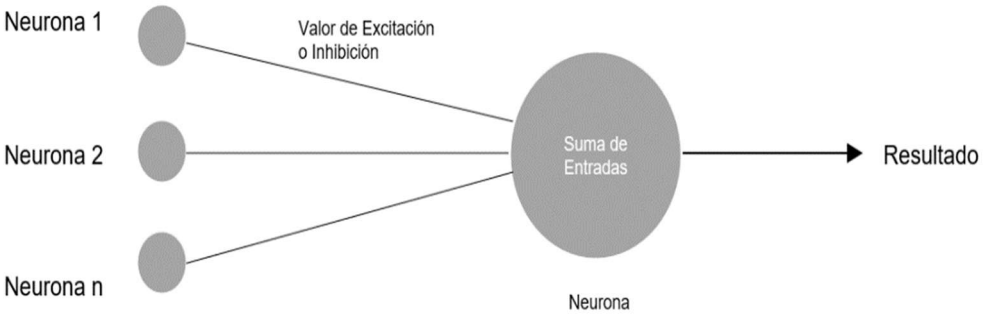

Fig. 2.8 Estructura básica de neurona artificial

Capitulo 3 Metodología

3.1 Datos

Los datos que se encuentran almacenados de manera digital en algún formato en específico (CSV; XLS, JSON), se conocen como el conjunto de datos (dataset, en inglés), los cuales pueden ser clasificados en estructurados, cuando cuentan con una estructura predefinida, y no estructurados cuando no cuentan con un formato en específico. Un ejemplo de estos tipos de datasets puede ser una tabla en una base de datos (estructurado) y el cuerpo de una carta en un documento de texto (no estructurado) (Pineda Pertuz, 2021).

Los conjuntos de datos más comúnmente utilizados para el aprendizaje automático son en formato CSV (comma separated values), en los cuales la información se encuentra en forma de tabla, separadas como su nombre lo indica por medio de comas (,) con columnas que indican el nombre de características o de variables, y filas que representan las muestras o instancias. En la Tabla 4.1, se muestra como los datos contenidos dentro de un archivo CSV, ya sean variables o atributos, están clasificados según el tipo de valor que contienen.

Tabla 4.1: Estructura de un archivo tipo CSV

Tipo de valor	Descripción	Ejemplo
Numérico	Cantidades numéricas dentro del conjunto de números reales	Peso: 85 Edad: 35
Categórico	Valores dentro de un rango limitado (indican cualidades o categorías)	Colores: Rojo, Amarillo, Verde, Blanco
Ordinales	Cadenas que respetan un orden entre si	Tallas CH > M > G

Las variables de los conjuntos de datos se pueden clasificar como descriptivas y de etiqueta o de destino. Las primeras sirven para describir las instancias que están almacenadas en el conjunto de datos, mientras que las de destino, se usan para etiquetar las instancias, lo cual será necesario para poder realizar el entrenamiento

del modelo. Generalmente estas variables de destino son las que representan los atributos que se quieren predecir.

Los datos se obtuvieron del sistema de control y monitoreo con el que cuenta la planta Arauco, con información registrada diariamente y con frecuencia de 1 segundo de cada uno de los sensores instalados para el proceso de producción de formaldehido, los cuales controlan temperatura, presión, flujo, niveles, y misceláneos. Estos datos se encuentran en una hoja de cálculo de Excel proporcionada por la empresa.

3.2 Flujo de trabajo

Para poder trabajar la información obtenida del sistema de monitoreo de la planta química de la empresa ARAUCO, se empleó el flujo de trabajo mostrado en la Fig. 3.1.

Fig. 3.1 Procedimiento para obtención del modelo de aprendizaje automático

Preprocesamiento de los datos

En primer lugar, se realizó un pre-procesamiento de datos, que es la etapa donde se ejecutó la limpieza de los datos, así como la transformación de estos, de tal forma que pudieran ser utilizados por el algoritmo de aprendizaje automático que se elaboró. Esta etapa es primordial para el buen funcionamiento del modelo, ya que para garantizar su eficiencia, la información deberá estar en el formato correcto,

todos bajo una misma escala y únicamente manejando datos numéricos. Una vez que el conjunto de datos se encontraba debidamente preprocesado, se separó en los subconjuntos de prueba, entrenamiento y validación, que es la forma más común de procesar datos en el aprendizaje automático.

Mediante el uso del software Python ®, se importaron los datos de los archivos que contienen la información del sistema de control y monitoreo de la planta, utilizando la librería Pandas para hacer manejable la información.
Una vez que se tuvieron cargados los datos en el software, se procedió a obtener la desviación estándar de cada una de las variables del Dataset (3.2), con el objetivo de medir la variabilidad o dispersión de la información. Tomando en cuenta que una desviación estándar baja expresa que los valores son cercanos a la media, y una desviación estándar alta indica una gran dispersión de los valores, se descartaron aquellas variables cuyo valor obtenido fue 0, ya que representaba que todos los valores eran constantes y por consiguiente tenían una variabilidad nula (Rustom J., 2012) **.**

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N} - \mu^2} \qquad (\ 3.2\ )$$

Posterior a la revisión de las desviaciones estándar de las variables, se procedió a analizar la información para localizar valores erróneos o faltantes dentro del dataset, ocasionados por fallas de medición en los sensores, descartando aquellas variables que contaban con datos faltantes en más del 25% del total, al no considerarse como una variable confiable.

3.2.1 Separación en conjunto de entrenamiento y prueba

El conjunto de entrenamiento tiene como fin entrenar y estimar los parámetros del modelo, mientras que el de prueba sirve para probar el modelo, analizar si la

respuesta es la esperada, y hacer las predicciones correspondientes. Dependiendo del tamaño de los conjuntos de datos, las proporciones empleadas para elaborar los subconjuntos de prueba y entrenamiento más comunes son: 30% - 70% y 20% - 80% respectivamente, para el caso de este estudio, se empleó una distribución 70% - 20% - 10%, separando la información en datos de entrenamiento, datos de prueba y datos de validación.

3.2.2 Escalamiento y centrado de los datos

Los datos obtenidos del sistema de control y monitoreo de la planta Arauco, se encuentran indicados en unidades distintas para algunas variables del proceso, por lo cual fue necesario realizar un proceso de normalización de la información, con el objetivo de ajustar valores medidos en escalas distintas, a una escala común para mejorar su manejo e interpretación. En el aprendizaje automático es importante realizar este paso, ya que evita que el algoritmo de mayor importancia a los datos con escalas más grandes. Para evitar esto se empleó la normalización estándar o z score (3.3), lo cual centra los valores con una media de 0 y desviación estándar de 1. Este proceso resta el valor de la media de cada variable del dataset, a cada dato la variable, y dividiendo el resultado entre la desviación estándar.

$$Xestándar = \frac{x_i - \mu}{\sigma} \qquad (3.3)$$

3.2.3 Configuración del algoritmo

La tercera etapa dentro del procedimiento se enfocó en la creación del algoritmo que se emplearía, definiendo los hiperparametros con los valores que el analista de datos debió ajustar de manera prudente. Al contar con el algoritmo debidamente establecido, se procedió a diseñar la red neuronal, utilizando distintas configuraciones de capas y neuronas, para proceder a realizar el entrenamiento del modelo, alimentándolo con el subconjunto de datos de entrenamiento, comenzando

el proceso de aprendizaje, para de esta manera estimar los parámetros del modelo. Una vez que el modelo se encuentra debidamente entrenado, la siguiente etapa es probar que tan bueno es su nivel de predicción, esto se logró alimentándolo con la información del subconjunto de prueba, observando el resultado de la predicción que se genera sobre una muestra, obteniendo un valor continuo o un valor discreto, el cual deberá ser lo más cercano posible al valor esperado.

3.2.4 Entrenamiento y evaluación del modelo

La validación del modelo es uno de los puntos clave en el proceso, debido a que es donde se verificará que el modelo empleado sea útil para los diferentes conjuntos de datos, ya que puede presentarse el caso de que ajuste bien para para el dataset de entrenamiento, pero no así para el de prueba. La evaluación es el último paso en el procedimiento establecido, la cual consiste en determinar en forma numérica la efectividad del modelo de aprendizaje automático elaborado. Se puede considerar de manera general, que entre menor sea la diferencia entre la respuesta esperada y la respuesta obtenida en la predicción, mejor será la evaluación (Pajares Martinsanz & Cruz García, 2011).

Se realizó la validación del modelo obtenido empleando el 10% de información reservada para este paso. Para evaluar el modelo y la precisión de sus predicciones, se utilizó la función de perdida de regresión del error cuadrático medio (MSE, por sus siglas en inglés) el cual representa el promedio de la diferencia que resulta entre el valor de la predicción obtenida y el valor real, todo esto elevado al cuadrado (3.5). Los valores obtenidos en este punto deben estar lo más cercano posible a cero, para poder ser considerados como fiables, ya que esto indica un error pequeño en la predicción.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (\ 3.5\ )$$

Como método de verificación del rendimiento de modelo, se empleó el coeficiente de determinación R^2, o prueba de bondad de ajuste (3.7), para determinar que tan bien se ajusta el modelo a los datos reales obtenidos de la planta Arauco. Los valores obtenidos de aplicar este método se sitúan en un rango entre 0 y 1, en donde un valor cercano a 1 representa un modelo fiable para realizar predicciones, mientras que un valor cercano a 0 indica que el modelo no es fiable en sus predicciones obtenidas.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2} \qquad (3.7)$$

Capitulo 4 Resultados y discusión

4.1 Dataset Original

La planta química de la empresa ARAUCO, cuenta con un sistema de control y monitoreo a base de sensores que miden temperaturas, flujos, presiones y variables misceláneas. Para el control y visualización de esta información emplean una interfaz gráfica (Fig. 4.1), donde se encuentran representados todos los elementos que forman parte del proceso de producción del formaldehido. Cada uno de los sensores registran la información con frecuencia de un valor cada segundo.

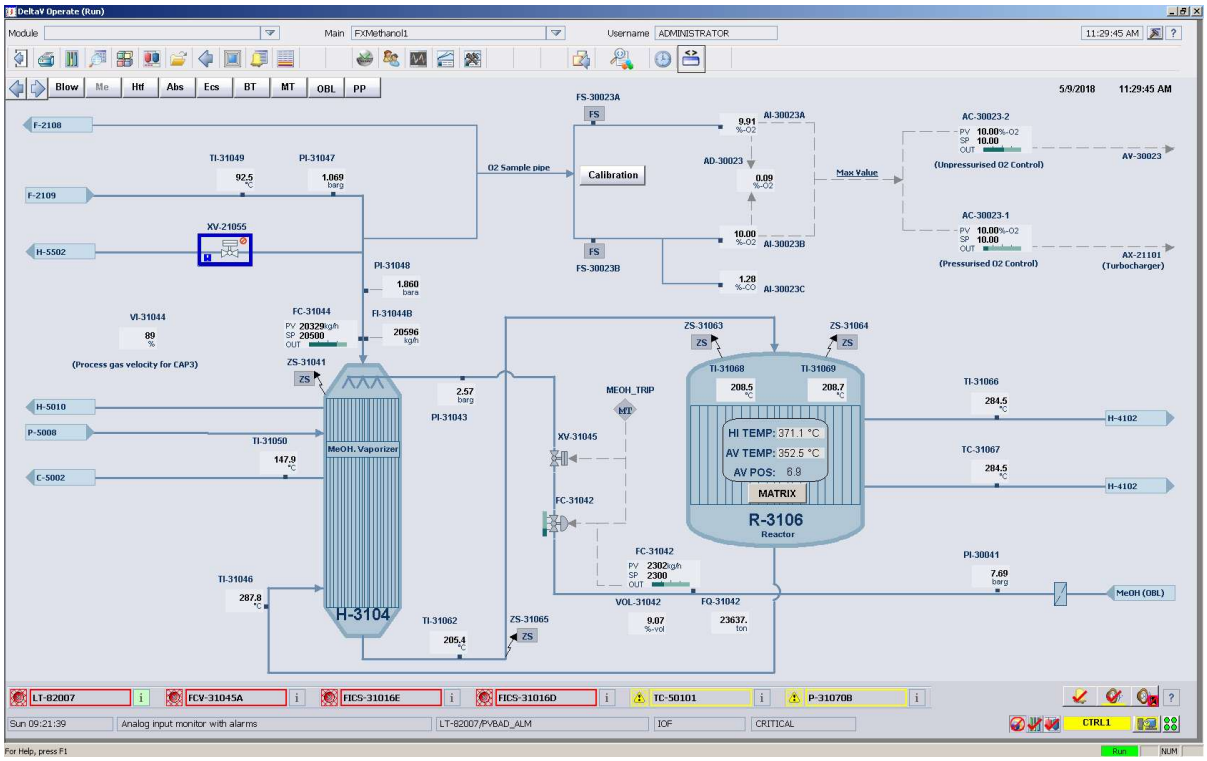

Fig. 4.1 Captura de pantalla de sistema de monitoreo y control de planta química ARAUCO (2019)

De este sistema de control y monitoreo, se cuenta con registros de los sensores, almacenados con 2 frecuencias: registros cada 1 segundo y registros cada 4 horas. Estos registros se consideran como 2 procesos diferentes ya que se realizaron

cambios en los procesos de la planta para la obtención del formaldehido. Por lo cual se analizaron por separado, y se identificaron como “proceso 2019” a los registros obtenidos en los meses de enero y febrero de 2019 y “proceso 2024” para los obtenidos en los años 2016, 2017, 2018, 2019, 2023 y 2024.

En la Tabla 5.1 se muestran los datos de los días 15, 17, 23, 29 y 30 de enero, 2, 11, 14 y 15 de febrero de 2019 00:01am a 11:59pm, con registros de cada sensor en frecuencia de 1 segundo (86,400 registros por día), para contar con un dataset inicial de 147,600 registros y 220 variables que representan cada sensor existente en la planta.

Tabla 5.1: Registros para elaboración del modelo con frecuencia de 1 segundo (2019)

Dia de registro	Hora registro inicial	Hora registro final	Registros totales	Variables
15 enero 2019	8:00:00 am	11:59:00 am	10,800	220
17 enero 2019	8:00:00 am	11:59:00 am	10,800	220
23 enero 2019	8:00:00 am	11:59:00 am	10,800	220
29 enero 2019	8:00:00 am	11:59:00 am	10,800	220
30 enero 2019	00:00:01 am	11:59:59 pm	86,400	220
2 febrero 2019	00:00:01 am	11:59:59 pm	86,400	220
11 febrero 2019	00:00:01 am	11:59:59 pm	86,400	220
14 febrero 2019	00:00:01 am	11:59:59 pm	86,400	220
15 febrero 2019	00:00:01 am	11:59:59 pm	86,400	220

De igual manera como se observa en la Tabla 5.2, se cuenta también con datos registrados con frecuencias de 4 horas, los cuales cubren un periodo desde

septiembre de 2016 hasta el 20 de enero de 2024, formando un dataset inicial de 4,929 registros y 220 variables que representan cada sensor existente en la planta.

Tabla 5.2 Registros para elaboración del modelo con frecuencia de 4 horas (2024)

Dia de registro	Hora registro inicial	Hora registro final	Registros totales	Variables
Registros 2016				
Sept, Oct, Nov	8:00:00 am	11:59:00 am	547	220
Registros 2017				
Ene – Dic	8:00:00 am	11:59:00 am	2,005	220
Registros 2018				
Ene – Dic	00:00:00 am	11:59:59 pm	1,825	220
Registros 2019				
Enero	00:00:00 am	11:59:59 pm	186	220
Registros 2023				
Nov - Dic	00:00:00 am	11:59:59 pm	366	220
Registros 2024 (Validación Nuevo Proceso)				
1-20 Enero	00:00:00 am	11:59:59 pm	118	220

De los registros del proceso identificado como “proceso 2024”, se dejaron los correspondientes al mes de enero para su posterior uso como método de verificación de la eficacia del modelo, de manera que pueda trabajar con datos que no han sufrido ninguna modificación y son desconocidos para el modelo.

Las variables que representan cada uno de los sensores localizados dentro del proceso de la planta para la producción de formaldehido, se encuentran numeradas de manera consecutiva, así como con identificadas con una clave que sirve de

referencia para identificar a que grupo pertenece y en que parte del sistema se encuentra localizado el sensor, como se puede ver en la Fig. 4.2.

El archivo con los registros seleccionados para la elaboración del modelo fue entregado en archivo de hoja de cálculo (*.xlsx), el cual cuenta con una fila de encabezado indicando el número de sensor de cada variable, así como una columna inicial con la fecha y hora del registro. (Fig. 4.3)

Se genera un archivo de hoja de cálculo (*.xlsx) para cada día de registros del sistema de monitoreo, por lo que, como paso previo, se procedió a unir todos los archivos en una sola hoja de cálculo, para facilitar su manejo.

Se realizó el mismo preprocesamiento para los dos Datasets que se tienen, tanto el de 1 segundo como el de 4 horas, para analizar los resultados bajo diferentes lapsos de tiempo.

No.	Variable	Descripción	Group	System	Trip
1	AC-30023-1/PID1/PV.CV	Valvula del control de presion del oxígeno	01 ANALYSE	Reactor / Sopladores	Plant trip
2	AC-30023-2/PID1/PV.CV	Valvula del control de presion del oxígeno de entrada de aire fresco	01 ANALYSE	Reactor / Sopladores	Plant trip
3	AD-30023/AI1/PV.CV	Analizador de diferencial de oxígeno	01 ANALYSE	Reactor	Plant trip
4	AI-30023A/AI1/PV.CV	Contenido de oxígeno Bajo / Alto	01 ANALYSE	Reactor	MeOH trip
5	AI-30023B/AI1/PV.CV	Contenido de oxígeno Bajo / Alto	01 ANALYSE	Reactor	MeOH trip
6	AI-30023C/AI1/PV.CV	Contenido de monóxido de carbono Bajo / Alto	02 FLOW	Reactor	Plant trip
7	FC-31042/PID1/PV.CV	Flujo de MeOH	02 FLOW	Reactor	MeOH trip
8	FC-31044/PID1/PV.CV	Flujo de gas de proceso	02 FLOW	Reactor / Sopladores	MeOH trip
9	FI-31044B/AI1/PV.CV	Flujo de gas de proceso	02 FLOW	Reactor	MeOH trip
10	FI-40081/AI1/PV.CV	Flujo HTF	02 FLOW	Condensador	Plant trip
11	FI-92044/AI1/PV.CV	Flujo BFW - condensador HTF	03 FLOWTOTAL	Condensador	Plant trip
12	LC-41023/PID1/PV.CV	Nivel de BFW en el condensador HTF	04 LEVEL	Condensador	MeOH trip
13	LI-40042/AI1/PV.CV	Indicador de nivel de tanque de HTF	04 LEVEL	Condensador	Plant trip
14	LI-41029/AI1/PV.CV	Nivel del condensador HTF	05 PRESSURE	Condensador	MeOH trip
15	PC-21364/PID1/PV.CV	Valvula de la presion de lubricacion de TC	05 PRESSURE	Sopladores	Plant trip
16	PC-41026/PID1/PV.CV	Presión condensador HTF	05 PRESSURE	Condensador	Blower trip
17	PC-55022/PID1/PV.CV	Sistema presurizado Bajo / Alto	05 PRESSURE	Sopladores	Blower trip
18	PI-21024/AI1/PV.CV	Indicador de presion de entrada de aire freso al TC	05 PRESSURE	Sopladores	Plant trip
19	PI-21042/AI1/PV.CV	Indicador de presion de aire de salida del TC	05 PRESSURE	Sopladores	Plant trip
20	PI-30041/AI1/PV.CV	Presion de metanol despues del filtro antes del vaporizador (after strain)	05 PRESSURE	Reactor	Plant trip
21	PI-31043/AI1/PV.CV	Presion de metanol antes del vaporizador	05 PRESSURE	Reactor	Plant trip
22	PI-31047/AI1/PV.CV	Presion del gas de proceso a la entrada del vaporizador	05 PRESSURE	Reactor / Sopladores	Plant trip
23	PI-31048/AI1/PV.CV	Presion del gas de proceso a la entrada del vaporizador despues de las recirculaciones	05 PRESSURE	Reactor / Sopladores	Plant trip
24	PI-55024/AI1/PV.CV	Presión de la columna de absorción	05 PRESSURE	Sopladores	Blower trip
25	PI-55066/AI1/PV.CV	Presion del gas de proceso antes del TC	05 PRESSURE	Sopladores	Plant trip
26	PI-55067/AI1/PV.CV	Presion del gas de proceso despues del TC	05 PRESSURE	Sopladores	Plant trip
27	PI-80004/AI1/PV.CV	Presion ambiental dentro del cuarto de soplantes	05 PRESSURE	Sopladores	Plant trip
28	PI-92048/AI1/PV.CV	Indicador de presion de entrada al condensador BFW	06 TEMP1	Condensador	Plant trip
29	TC-21321/ALM1/PV.CV	Temperatura del aceite de lubricante del TC	06 TEMP1	Sopladores	Plant trip
30	TC-31067/ALM1/PV.CV	Temperatura del vapor HTF	06 TEMP1	Reactor / Condensador	MeOH trip
31	TC-40083/ALM1/PV.CV	Temperatura de HTF a la salida de los calentadores	06 TEMP1	Reactor / Condensador	Plant trip
32	TI-21043/AI1/PV.CV	Temperatura de aire a la salida del turbocargador	06 TEMP1	Sopladores	Plant trip
33	TI-21363/AI1/PV.CV	Temperatura del aceite lubricante del TC	06 TEMP1	Sopladores	Plant trip
34	TI-31046/AI1/PV.CV	Temperatura PG después del reactor FA	06 TEMP1	Reactor	Blower trip
35	TI-31049/AI1/PV.CV	Temperatura del gas de proceso despues de soplantes	07 TEMP2	Reactor	Plant trip
36	TI-31050/AI1/PV.CV	Tempreatura del PG antes del Absorbedor	07 TEMP2	Reactor	Blower trip
37	TI-31062/AI1/PV.CV	Temperatura del PG antes del reactor FA	07 TEMP2	Reactor	Blower trip
38	TI-31066/AI1/PV.CV	Temperatura del vapor HTF	07 TEMP2	Reactor / Condensador	MeOH trip
39	TI-31068/AI1/PV.CV	Temperatura del Reactor alto	07 TEMP2	Reactor	Blower trip
40	TI-31069/AI1/PV.CV	Temperatura del Reactor alto	07 TEMP2	Reactor	Blower trip
41	TI-40041/AI1/PV.CV	Temperatura del tanque de almacenamiento HTF	07 TEMP2	Condensador	Plant trip
42	TI-40082A/AI1/PV.CV	Temperatura de calentadores de HTF	07 TEMP2	Condensador	Plant trip
43	TI-40082B/AI1/PV.CV	Temperatura de calentadores de HTF	07 TEMP2	Condensador	Plant trip
44	TI-40082C/AI1/PV.CV	Temperatura de calentadores de HTF	07 TEMP2	Condensador	Plant trip
45	TI-41021/AI1/PV.CV	Temperatura HTF dentro condensador baja	07 TEMP2	Condensador	Plant trip
46	TI-41025/AI1/PV.CV	High temp HTF condensor top	07 TEMP2	Condensador	MeOH trip
47	TI-50091/AI1/PV.CV	Temperatura de aire de recirculacion del adsorbedor a los sopladores	07 TEMP2	Reactor	Plant trip
48	TI-55064/AI1/PV.CV	Temperatura después de TC	07 TEMP2	Reactor	Blower trip
49	TI-55065/AI1/PV.CV	Temperatura después de la cama catlítica	07 TEMP2	Reactor	Blower trip
50	TI-80001/AI1/PV.CV	Temperatura del cuarto de soplantes	08 MISC	Sopladores	Plant trip

Fig. 4.2 Muestra de nomenclatura y descripción de variables de proceso.

Date	AC-30023-1/PID1/OUT.CV	AC-30023-1/PID1/PV.CV	AC-30023-1/PID1/SP.CV	AC-30023-2/PID1/OUT.CV	AC-30023-2/PID1/PV.CV	AC-30023-2/PID1/SP.CV	AC-50064/PID1/OUT.CV	AC-50064/PID1/PV.CV	AC-50064/PID1/SP.CV	AD-30023/AI1/PV.CV	AI-30023/AI1/PV.CV	AI-30023/BI1/PV.CV	AI-30023/CI1/PV.CV	DI-50064/AI1/PV.CV	FC-31042/PID1/OUT.CV	FC-31042/PID1/PV.CV	FC-31042/PID1/SP.CV	FC-31044/AO1/PV.CV	FC-31044/AO2/PV.CV	FC-31044/PID1/OUT.CV	FC-31044/PID1/PV.CV	FC-31044/PID1/SP.CV	FC-50077/PID1/OUT.CV	FC-50077/PID1/PV.CV
29/01/2019 08:00:00	0	11.11277	10	52	11.11277	10	0	54.07832	53.6	0.274689	11.11275	10.83784	1.015204	1137.858	9.518686	1342.57	1350	64.76784	64.76784	64.76788	15805.54	15788	0	0
29/01/2019 08:00:01	0	11.11275	10	52	11.11275	10	0	54.07829	53.6	0.2747506	11.11274	10.83781	1.015206	1137.858	9.518598	1342.569	1350	64.76788	64.76788	64.76793	15805.92	15788	0	0
29/01/2019 08:00:02	0	11.11274	10	52	11.11274	10	0	54.07827	53.6	0.2748123	11.11272	10.83779	1.015207	1137.858	9.51851	1342.568	1350	64.76793	64.76793	64.76798	15806.3	15788	0	0
29/01/2019 08:00:03	0	11.11272	10	52	11.11272	10	0	54.07825	53.6	0.274874	11.11271	10.83777	1.015209	1137.858	9.518422	1342.567	1350	64.76798	64.76798	64.76804	15806.68	15788	0	0
29/01/2019 08:00:04	0	11.11271	10	52	11.11271	10	0	54.07822	53.6	0.2749356	11.11269	10.83774	1.01521	1137.858	9.518333	1342.565	1350	64.76804	64.76804	64.76808	15807.07	15788	0	0
29/01/2019 08:00:05	0	11.11269	10	52	11.11269	10	0	54.07819	53.6	0.2749973	11.11268	10.83772	1.015212	1137.858	9.518246	1342.564	1350	64.76808	64.76808	64.76813	15807.45	15788	0	0
29/01/2019 08:00:06	0	11.11268	10	52	11.11268	10	0	54.07817	53.6	0.2750589	11.11266	10.8377	1.015213	1137.859	9.518158	1342.563	1350	64.76813	64.76813	64.76818	15807.83	15788	0	0
29/01/2019 08:00:07	0	11.11266	10	52	11.11266	10	0	54.07815	53.6	0.2751206	11.11265	10.83768	1.015214	1137.859	9.51807	1342.563	1350	64.76818	64.76818	64.76823	15808.21	15788	0	0
29/01/2019 08:00:08	0	11.11265	10	52	11.11265	10	0	54.07813	53.6	0.2751822	11.11263	10.83765	1.015216	1137.859	9.517982	1342.561	1350	64.76823	64.76823	64.76828	15808.6	15788	0	0
29/01/2019 08:00:09	0	11.11263	10	52	11.11263	10	0	54.0781	53.6	0.2752439	11.11261	10.83763	1.015217	1137.859	9.517894	1342.56	1350	64.76828	64.76828	64.76833	15808.98	15788	0	0
29/01/2019 08:00:10	0	11.11261	10	52	11.11261	10	0	54.07807	53.6	0.2753055	11.1126	10.83761	1.015219	1137.859	9.517806	1342.559	1350	64.76833	64.76833	64.76838	15809.36	15788	0	0
29/01/2019 08:00:11	0	11.1126	10	52	11.1126	10	0	54.07805	53.6	0.2753672	11.11258	10.83758	1.01522	1137.859	9.517717	1342.558	1350	64.76838	64.76838	64.76843	15809.75	15788	0	0
29/01/2019 08:00:12	0	11.11258	10	52	11.11258	10	0	54.07803	53.6	0.2754288	11.11257	10.83756	1.015221	1137.859	9.51763	1342.557	1350	64.76843	64.76843	64.76848	15810.13	15788	0	0
29/01/2019 08:00:13	0	11.11257	10	52	11.11257	10	0	54.078	53.6	0.2754905	11.11255	10.83754	1.015223	1137.859	9.517542	1342.556	1350	64.76848	64.76848	64.76852	15810.51	15788	0	0
29/01/2019 08:00:14	0	11.11255	10	52	11.11255	10	0	54.07797	53.6	0.2755522	11.11254	10.83751	1.015224	1137.859	9.517454	1342.555	1350	64.76852	64.76852	64.76858	15810.89	15788	0	0
29/01/2019 08:00:15	0	11.11254	10	52	11.11254	10	0	54.07795	53.6	0.2756138	11.11252	10.83749	1.015226	1137.859	9.517366	1342.554	1350	64.76858	64.76858	64.76863	15811.28	15788	0	0
29/01/2019 08:00:16	0	11.11252	10	52	11.11252	10	0	54.07793	53.6	0.2756754	11.1125	10.83747	1.015227	1137.859	9.517278	1342.553	1350	64.76863	64.76863	64.76868	15811.66	15788	0	0
29/01/2019 08:00:17	0	11.1125	10	52	11.1125	10	0	54.0779	53.6	0.2757371	11.11249	10.83745	1.015228	1137.859	9.51719	1342.552	1350	64.76868	64.76868	64.76872	15812.04	15788	0	0
29/01/2019 08:00:18	0	11.11249	10	52	11.11249	10	0	54.07787	53.6	0.2757987	11.11247	10.83742	1.01523	1137.859	9.517101	1342.551	1350	64.76872	64.76872	64.76877	15812.42	15788	0	0
29/01/2019 08:00:19	0	11.11247	10	52	11.11247	10	0	54.07785	53.6	0.2758604	11.11246	10.8374	1.015231	1137.86	9.517014	1342.55	1350	64.76877	64.76877	64.76882	15812.8	15788	0	0
29/01/2019 08:00:20	0	11.11246	10	52	11.11246	10	0	54.07783	53.6	0.2759221	11.11244	10.83738	1.015233	1137.86	9.516926	1342.549	1350	64.76882	64.76882	64.76888	15813.19	15788	0	0
29/01/2019 08:00:21	0	11.11244	10	52	11.11244	10	0	54.0778	53.6	0.2759837	11.11243	10.83735	1.015234	1137.86	9.516838	1342.548	1350	64.76888	64.76888	64.76892	15813.57	15788	0	0
29/01/2019 08:00:22	0	11.11243	10	52	11.11243	10	0	54.07778	53.6	0.2760454	11.11241	10.83733	1.015235	1137.86	9.51675	1342.547	1350	64.76892	64.76892	64.76897	15813.95	15788	0	0
29/01/2019 08:00:23	0	11.11241	10	52	11.11241	10	0	54.07775	53.6	0.276107	11.1124	10.83731	1.015237	1137.86	9.516662	1342.546	1350	64.76897	64.76897	64.76902	15814.33	15788	0	0
29/01/2019 08:00:24	0	11.1124	10	52	11.1124	10	0	54.07773	53.6	0.2761686	11.11238	10.83729	1.015238	1137.86	9.516574	1342.545	1350	64.76902	64.76902	64.76907	15814.72	15788	0	0
29/01/2019 08:00:25	0	11.11238	10	52	11.11238	10	0	54.07771	53.6	0.2762303	11.11237	10.83726	1.01524	1137.86	9.516485	1342.544	1350	64.76907	64.76907	64.76912	15815.1	15788	0	0
29/01/2019 08:00:26	0	11.11237	10	52	11.11237	10	0	54.07768	53.6	0.276292	11.11235	10.83724	1.015241	1137.86	9.516397	1342.543	1350	64.76912	64.76912	64.76917	15815.48	15788	0	0
29/01/2019 08:00:27	0	11.11235	10	52	11.11235	10	0	54.07765	53.6	0.2763536	11.11234	10.83722	1.015242	1137.86	9.51631	1342.542	1350	64.76917	64.76917	64.76922	15815.87	15788	0	0
29/01/2019 08:00:28	0	11.11234	10	52	11.11234	10	0	54.07763	53.6	0.2764153	11.11232	10.83719	1.015244	1137.86	9.516222	1342.541	1350	64.76922	64.76922	64.76927	15816.25	15788	0	0
29/01/2019 08:00:29	0	11.11232	10	52	11.11232	10	0	54.07761	53.6	0.2764769	11.1123	10.83717	1.015245	1137.86	9.516134	1342.54	1350	64.76927	64.76927	64.76932	15816.63	15788	0	0
29/01/2019 08:00:30	0	11.1123	10	52	11.1123	10	0	54.07758	53.6	0.2765386	11.11229	10.83715	1.015247	1137.86	9.516046	1342.539	1350	64.76932	64.76932	64.76936	15817.01	15788	0	0

Fig. 4.3 Hoja de cálculo con registros de información de la planta ARAUCO.

4.2 Preprocesamiento del Dataset

Como primer paso dentro del preprocesamiento, se identificaron los valores nulos o no válidos dentro del dataset, los cuales se identifican como vacíos en los registros o valores no numéricos dentro de una variable numérica (Fig. 4.4).

TI-50028/AI1/PV.CV	TI-50029/AI1/PV.CV	TI-50061/AI1/PV.CV	TI-50091/AI1/PV.CV	TI-55021/AI1/PV.CV	TI-55043/AI1/PV.CV
41.10678	32.04612	46.52566	19.97828	48.62207	115.3914
31.67849	28.93981	46.26264	22.35748	46.96323	103.6379
25.80664	25.34405	42.34486	21.50101	42.51519	96.28956
20.88625	21.51507	37.02627	19.07625	37.82144	89.91763
15.74801	17.07274	29.05927	15.50568	31.42752	79.48084
-9.741845	-9.180991	-4.279894	-3.624846	-49.0319	-76.45934
-6.969788	-6.709443	-1.406059	-0.4930282	-44.45176	-74.40624
19.66814	19.49909	34.97829	20.12213	35.41417	63.06142
19.10648	19.07811	30.1731	19.47709	33.40751	57.70278
16.31739	17.5594	25.57607	16.72519	28.77856	52.17826
13.22125	15.31196	22.7828	14.23708	25.1071	46.97317
12.06304	13.82518	22.58861	12.98912	22.92923	42.21275
7.508113	8.428791	16.70187	10.58375	2.643701	2.684279
???	???	???	???	???	???
???	???	???	???	???	???
???	???	???	???	???	???
???	???	???	???	???	???

Fig. 4.4 Ejemplo de registros no válidos en del dataset.

Posteriormente se procedió a identificar las variables con valores no numéricos, y en las cuales se determinó que, al no variar su contenido a lo largo del proceso, se podrían descartar (Fig. 4.5), ya que su contenido no representa información relevante para el proceso.

-2109_IL/FEEDBACK.CV	F-2110_IL/FEEDBACK.CV	HTF_IL/FEEDBACK.CV	MEOH_TRIP/FEEDBACK.C	PROD_MODE/PROD_MOD	PROD_MODE/UFC_MODE	RUNTIME1/DAYS.CV
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88
1.00	1.00	0.00	1.00	FA	Excess	464.88

Fig. 4.5 Registros no numéricos constantes en dataset de 1 segundo

En el caso de los registros con frecuencia de 4 horas se identificaron registros no numéricos que presentaban variaciones a lo largo del proceso (Fig. 4.6), por lo cual se procedió a sustituir cada uno de ellos por valores numéricos enteros, empleando la librería SciKit-Learn y la librería de Label Encoding.

‹110_IL/FEEDBACK.CV	HTF_IL/FEEDBACK.CV	MEOH_TRIP/FEEDBACK.(	PROD_MODE/PROD_MOL	PROD_MODE/UFC_MODE	RUNTIMEI/DAYS.CV
1.00	0.00	1.00	FA	Excess	173.00
1.00	0.00	1.00	FA	Excess	173.00
1.00	0.00	1.00	FA	Excess	173.00
1.00	0.00	1.00	FA	Excess	173.45
1.00	0.00	1.00	FA	Excess	173.95
1.00	0.00	1.00	FA	Excess	174.00
1.00	0.00	1.00	UFC	Excess	174.00
1.00	0.00	1.00	UFC	Excess	174.00
1.00	0.00	1.00	UFC	Excess	174.00
1.00	0.00	1.00	UFC	Excess	174.45
1.00	0.00	1.00	UFC	Excess	174.95
1.00	0.00	1.00	UFC	Excess	175.00
1.00	0.00	1.00	UFC	Excess	175.00
1.00	0.00	1.00	UFC	Excess	175.00
1.00	0.00	1.00	UFC	Excess	175.00
1.00	0.00	1.00	UFC	Excess	175.45
1.00	0.00	1.00	UFC	Excess	175.95
1.00	0.00	1.00	UFC	Excess	176.00

Fig. 4.6 Registros no numéricos variables en dataset de 4 horas

Una vez que se cuenta con el archivo de registros cargado en el software Python, se obtuvo la información estadística de cada variable dentro del dataset, incluyendo la media, desviación estándar, máximo, mínimo, y cuartiles. (Fig. 4.7)

	AC-30023-1/PID1/OUT.CV	AC-30023-1/PID1/PV.CV	AC-30023-1/PID1/SP.CV	AC-30023-2/PID1/OUT.CV	AC-30023-2/PID1/PV.CV	AC-30023-2/PID1/SP.CV	AC-50064/PID1/OUT.C
count	97200	97200	97200	97200	97200	97200	97200
mean	0.477	10.603	10.000	50.243	10.603	10.000	0.000
std	1.278	0.241	0.000	0.836	0.241	0.000	0.000
min	0.000	6.796	10.000	45.000	6.796	10.000	0.000
25%	0.000	10.545	10.000	50.000	10.545	10.000	0.000
50%	0.000	10.576	10.000	50.000	10.576	10.000	0.000
75%	0.000	10.621	10.000	50.000	10.621	10.000	0.000
max	20.000	12.951	10.000	57.000	12.951	10.000	0.000

Fig. 4.7 Información estadística de cada variable del dataset.

Tomando como referencia aquellas variables cuya desviación estándar resultó con valor de 0, indican que los valores se mantienen constantes y no presentan variación, se procedió a descartar dichas variables del dataset.

4.3 Separación de datos de entrenamiento, prueba y validación

Con el dataset preprocesado, y con las dimensiones finales bien definidas, se separó la información en datos de entrenamiento, prueba y validación como se muestra en la Tabla 5.3 y Tabla 5.4, tomando la proporción de 70% - 20% - 10%, como la deseable para este modelo, considerando el tamaño del dataset.

Tabla 5.3 Separación de Dataset con frecuencia de 1 segundo

Dataset	Registros totales	Variables
Completo	134,496 x 136	100%
Entrenamiento	100,437 x 136	70%
Prueba	25,109 x 136	20%
Validación	13,950 x 136	10%

Tabla 5.4 Separación del Dataset con frecuencia de 4 horas

Dataset	Registros totales	Variables
Completo	3,135 x 132	100%
Entrenamiento	2,195 x 132	70%
Prueba	627 x 132	20%
Validación	313 x 132	10%

Una vez que se tienen los datos separados en entrenamiento, validación y prueba, se selecciona la variable que sobre la cual se realizaran las predicciones, que para este caso será aquella identificada dentro del Dataset con el nombre de “Yield from

flow” que representa el rendimiento del flujo de formaldehido. Los registros de esta variable serán los que se predecirán empleando el modelo generado.

4.4 Escalamiento de Datos

Debido a que los registros de cada sensor de la planta miden distintas variables como temperatura, presión, flujo, etc., se realizó un proceso de normalización de la información, con el objetivo de ajustar valores medidos en escalas distintas, a una escala común para mejorar su manejo e interpretación.

Se generaron histogramas de cada una de las variables, tanto de los datos originales (Fig. 4.8), como de los datos normalizados para verificar que esta etapa mantiene la información con las mismas características iniciales (Fig. 4.9), observando que la información mantiene la misma estructura en ambos casos.

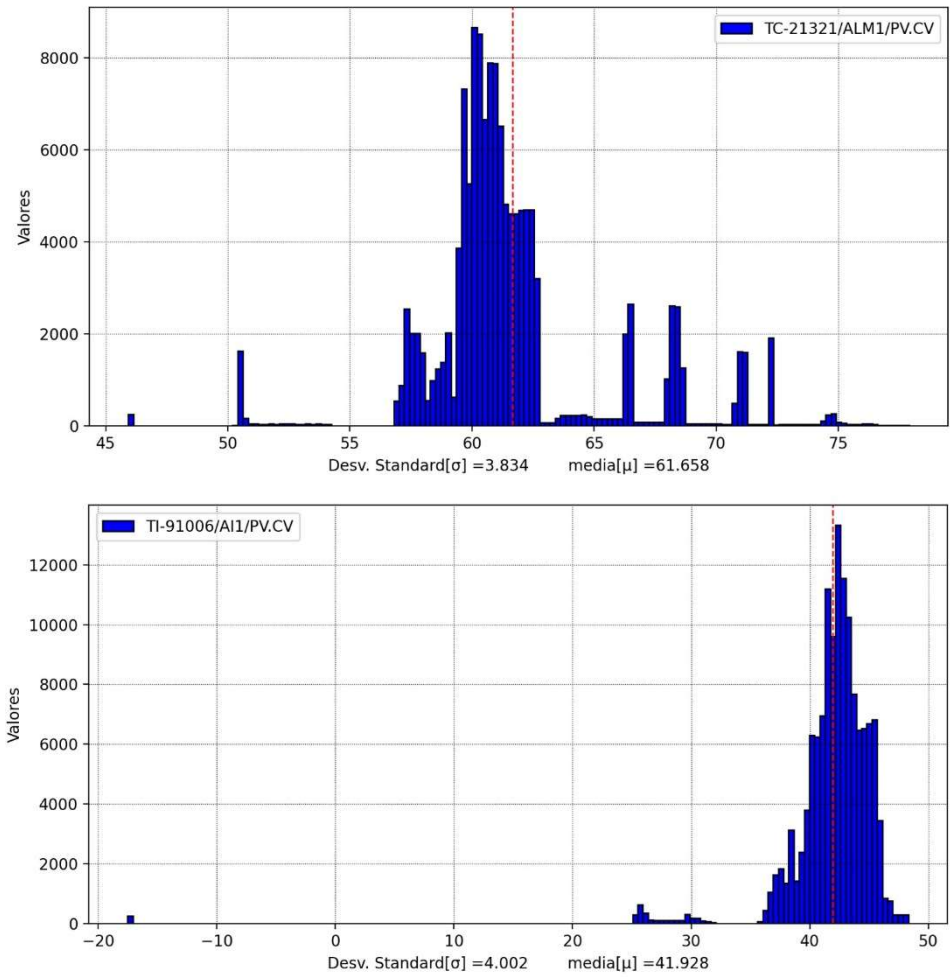

Fig. 4.8 Histograma de variables del proceso con unidades originales

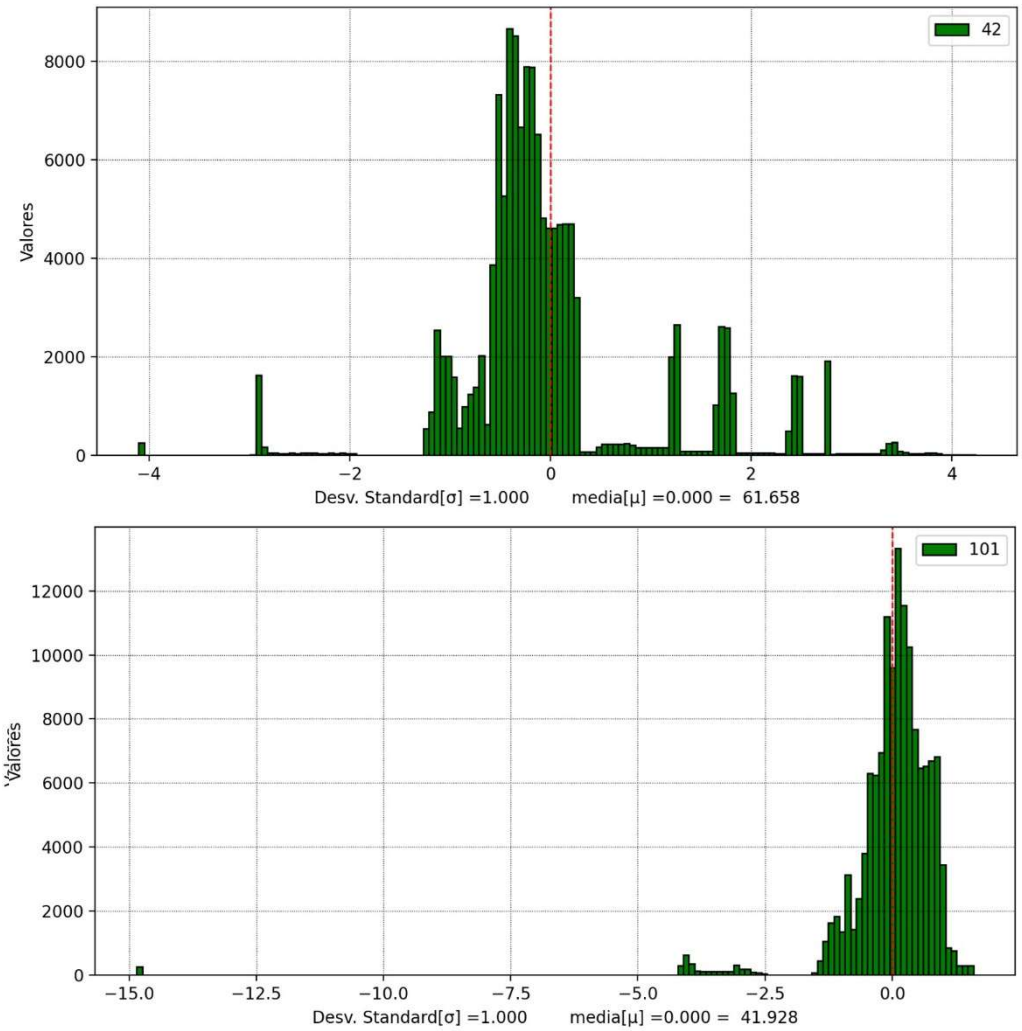

Fig. 4.9 Histograma de variables del proceso normalizado

4.5 Redes Neuronales

Una vez que se tienen definidos los Datasets preprocesados, se definieron los parámetros para la elaboración de la red neuronal a emplear (Tabla 5.5). Para los datos con frecuencia de 1 segundo se realizó una primera red, tomando las 135 variables resultantes del preprocesamiento como capa de entrada, así como el uso de 4 capas ocultas, formadas por 128, 256, 512 y 1024 neuronas respectivamente, y una capa de salida de 1 neurona (Fig. 4.10).

Tabla 5.5 Estructura de red neuronal para datos con frecuencia de 1 segundo

Capa	Neuronas
Input Layer	135
Hidden Layer 1	128
Hidden Layer 2	256
Hidden Layer 3	512
Hidden Layer 4	1024
Output Layer	1

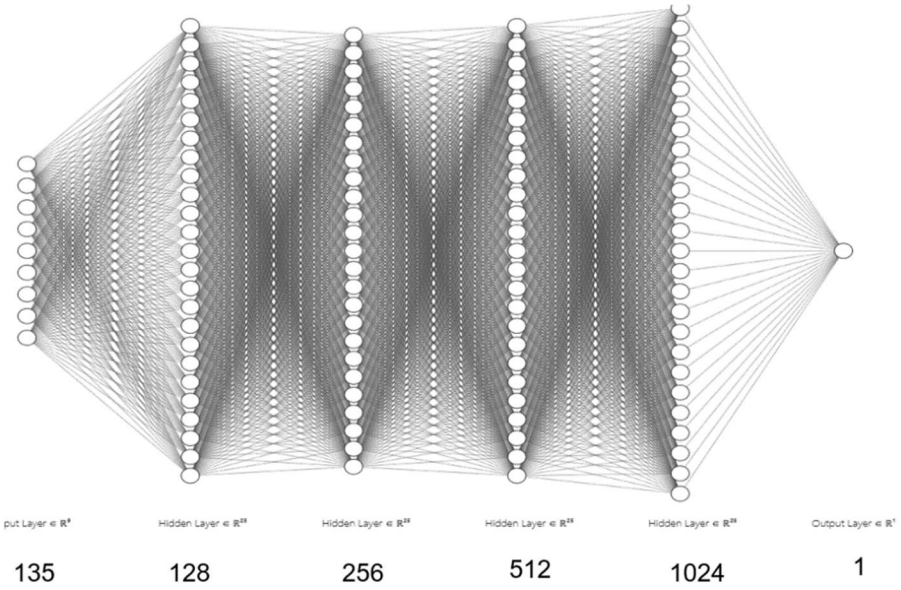

Fig. 4.10 Estructura de la red neuronal propuesta para 1 segundo

4.6 Función de Activación

Se consideró dentro de la elaboración de algoritmo, el uso de la función de activación de Unidad Lineal Rectificada (ReLU por sus siglas en inglés) (4.1), tomando en cuenta que es la de uso más común en la actualidad para introducir la no linealidad en el modelo (Fig. 4.11). Esta función considera que, si el valor de entrada es positivo devuelve el mismo valor, y, si el resultado es negativo, el valor devuelto es 0, de esta forma se eliminan todos los valores negativos.

$$f(x) = max(0, x) \qquad (4.1)$$

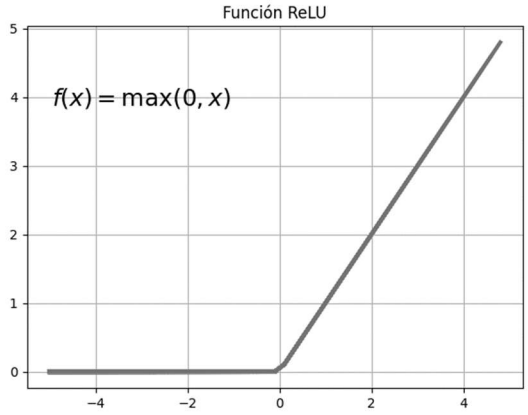

Fig. 4.11 Representación del funcionamiento de la función de activación ReLU

4.7 Predicciones

Para la realización de las predicciones empleando el dataset de 1 segundo se establecieron como parámetros en el modelo, el uso de 50 épocas de entrenamiento y el número de neuronas y capas mencionado en la Tabla 5.5.

Al ejecutar el código para la obtención del modelo, se graficaron los errores cuadrados medios (MSE) (3.5) obtenidos en la fase de entrenamiento como en la fase de validación, los cuales sirven para analizar el comportamiento del modelo, observando en la Fig. 4.12, la reducción en el error obtenido conforme se avanza en cada una de las épocas establecidas.

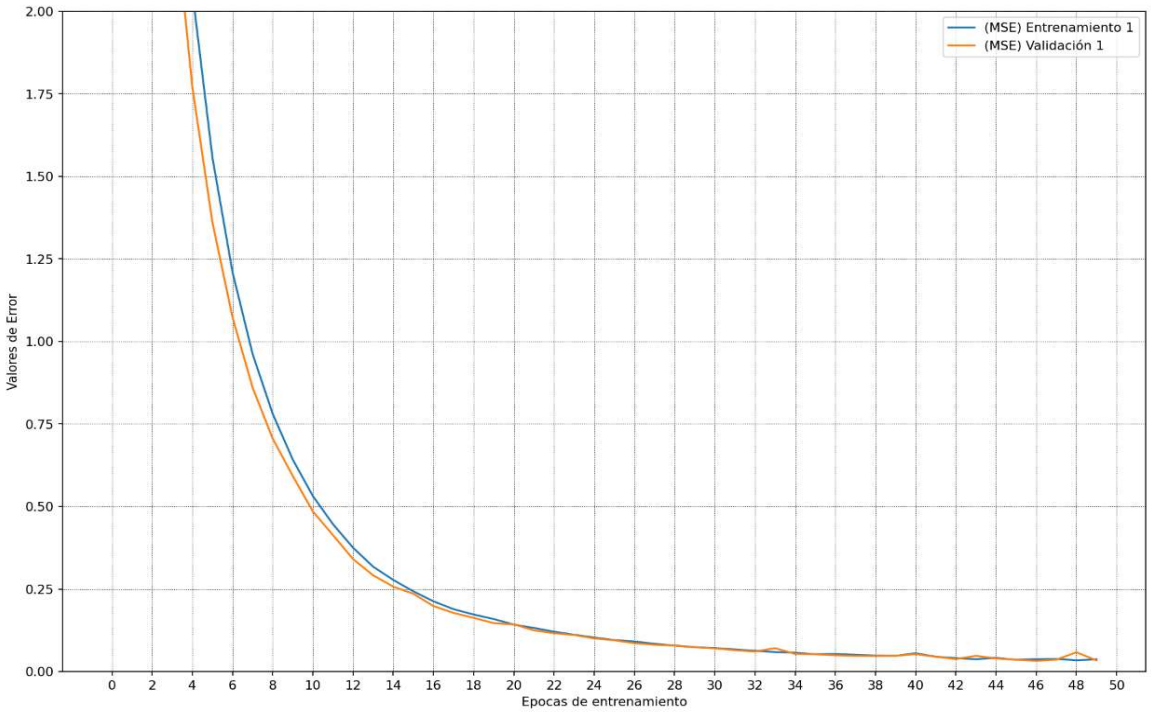

Fig. 4.12 Gráfica de MSE por épocas de dataset de 1 segundo

Cuando el modelo finaliza de ejecutarse se obtienen las predicciones de los 13,950 valores de la variable “Yield from flow”, mismos que se comparan contra los registros reales almacenados en la fase de preprocesamiento, y que se mantienen sin ninguna alteración. En la Tabla 5.6 se muestran los resultados para las primeras 11 predicciones, así como su variación y precisión.

Tabla 5.6 Comparativa entre predicciones obtenidas y registros reales

Consecutivo	Predicción	Registro real	Variación	Precisión
0	73.61932	72.43802	1.18130	98.40%
1	162.69939	163.24840	-0.54901	100.34%
2	93.31695	93.19500	0.12195	99.87%
3	92.51402	92.41209	0.10192	99.89%
4	92.71635	92.41209	0.30426	99.67%
5	163.55571	162.95970	0.59601	99.64%
6	90.55207	90.55775	-0.00568	100.01%
7	167.74394	166.71710	1.02684	99.39%
8	90.73024	90.72399	0.00625	99.99%
9	91.11675	90.72399	0.39276	99.57%
10	93.10378	92.76141	0.34237	99.63%

4.8 Eficiencia de la predicción

Con los datos de las predicciones se elaboró un histograma que permitió analizar el comportamiento de las diferencias obtenidas entre los valores reales y los predichos (Fig. 4.13), donde se observó que los resultados tienen una distribución normal, con variaciones entre -1.0 y hasta 0.75, por lo cual se puede concluir que el modelo funciona y no tiene sesgo positivo o negativo.

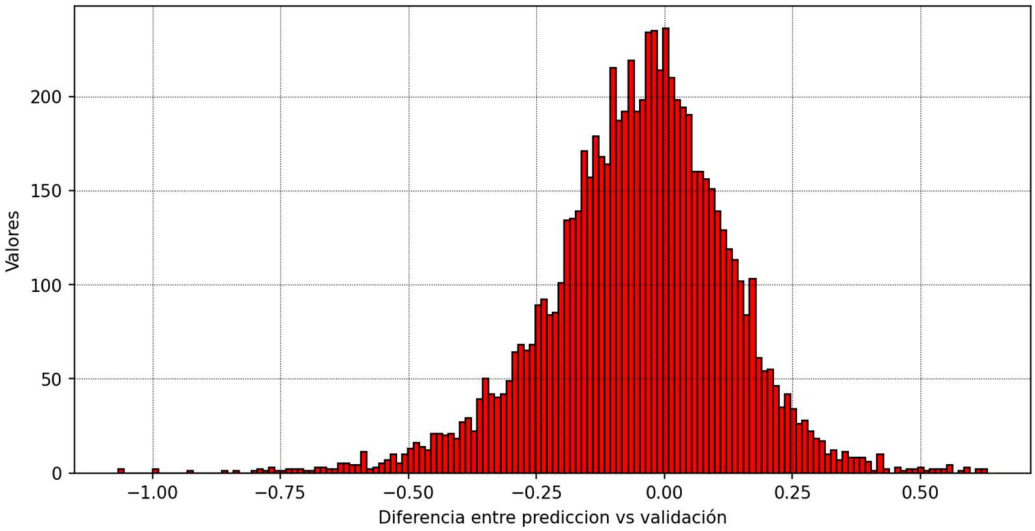

Fig. 4.13 Histograma de variaciones entre predicción y registro real.

De la misma forma en la Fig. 4.14 se observa el rango de la precisión de las predicciones obtenidas, el cual se encuentran los resultados, de donde podemos notar que la precisión se encuentra entre 99.0 y 99.8%

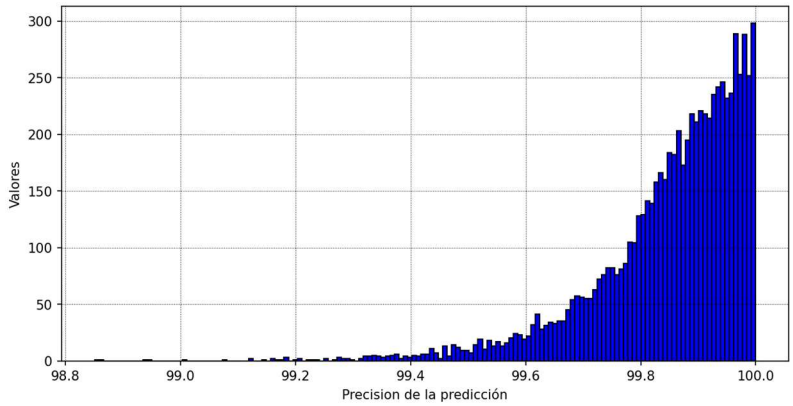

Fig. 4.14 Histograma de precisión de predicción vs registro real.

Como se estableció desde el inicio, y para analizar el comportamiento de las predicciones, se empleó el coeficiente de determinación R^2 para analizar y determinar la calidad del modelo generado. Obteniendo un valor de R^2 de 0.9982 para el dataset con registros de 1 segundo y con la estructura de red neuronal formadas por 128, 256, 512 y 1,024 neuronas, y una capa de salida de 1 neurona, como se muestra en la Fig. 4.15. En esta grafica se observa que se cuenta con un valor R^2 cercano a 1, lo cual indica que el modelo obtenido es de buena calidad, ya que los valores de las predicciones son muy cercanos a los registros reales.

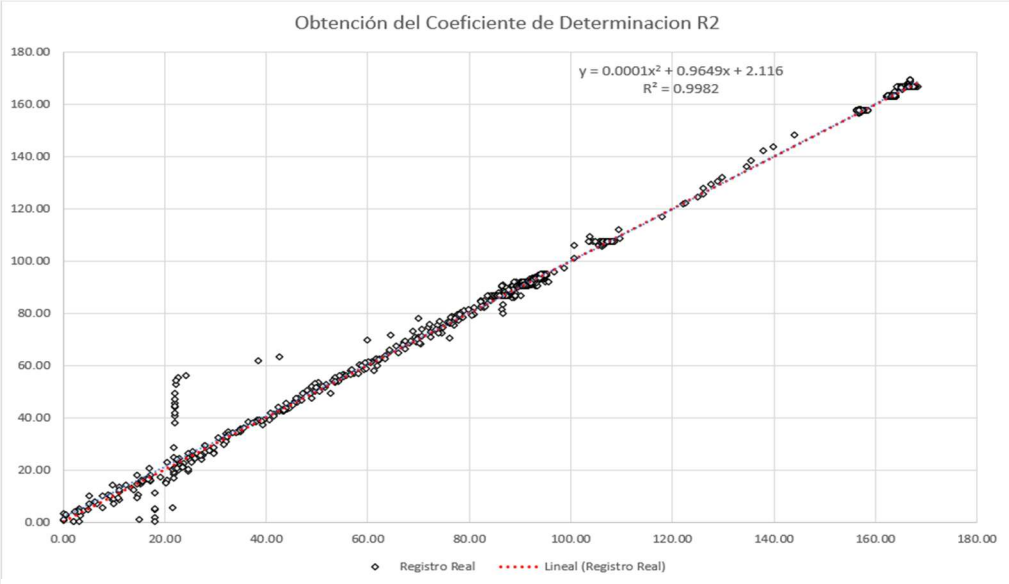

Fig. 4.15 Determinación del coeficiente de determinación.

4.9 Alternativas de estructura

Se probaron varias alternativas en la estructura de la red neuronal, buscando analizar el comportamiento y los resultados obtenidos al modificar el número de neuronas de las capas ocultas en cada iteración, como se aprecia en la Tabla 5.7, con la finalidad de tratar de obtener una eficiencia similar con un consumo menor de recursos computacionales y tiempo de obtención de la predicción.

Tabla 5.7 Alternativas de estructuras de redes neuronales

	Red Original	**Alternativa 1**	**Alternativa 2**
Capa	Neuronas	Neuronas	Neuronas
Input Layer	135	135	135
Hidden Layer 1	10	30	100
Hidden Layer 2	20	40	200
Hidden Layer 3	30	50	200
Hidden Layer 4	40	60	100
Output Layer	1	1	1

Estas variaciones de la estructura de las redes neuronales se probaron con el Dataset de 4 horas de frecuencia obteniendo los resultados mostrados en las Fig. 4.16, Fig. 4.17 y Fig. 4.18 para la estructura de la red original.

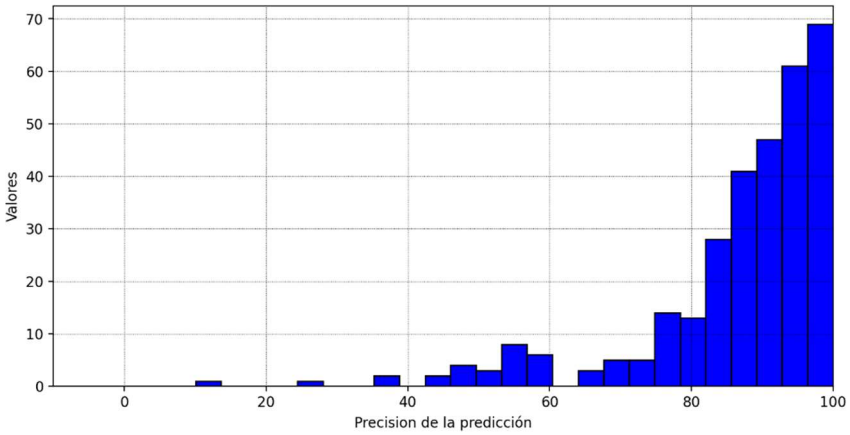

Fig. 4.16 Histograma de precisión de predicción dataset 4 horas Red Original

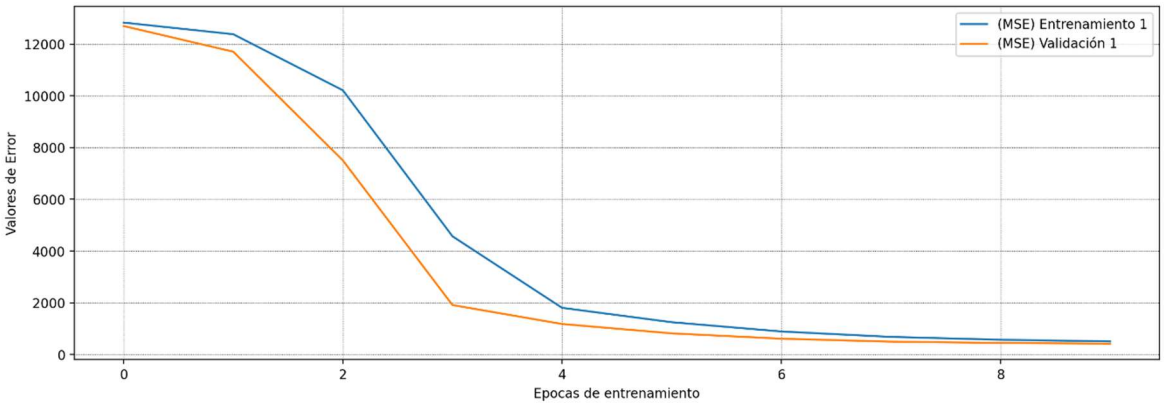

Fig. 4.17 Gráfica de MSE por épocas de dataset de 4 horas Red Original

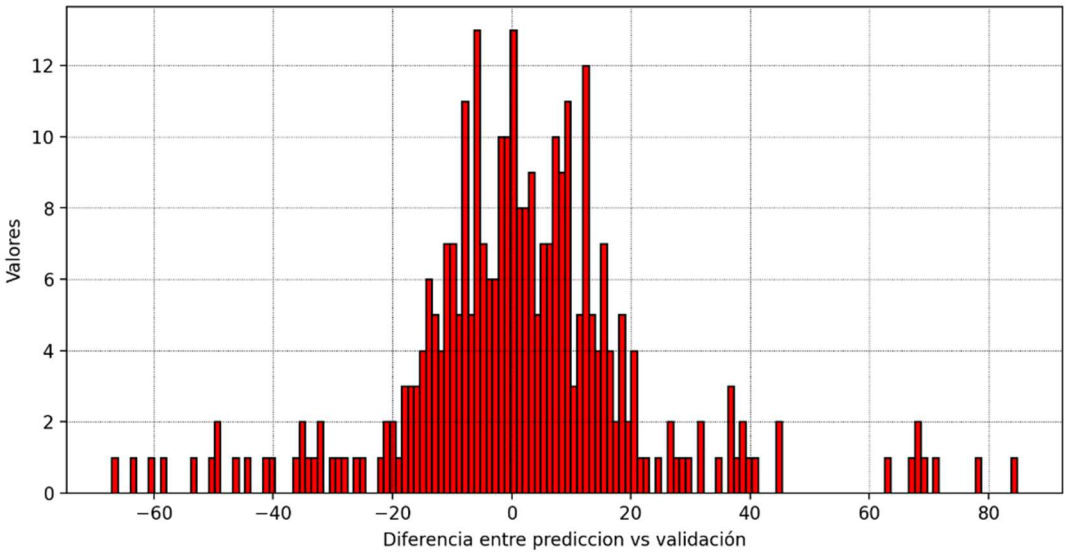

Fig. 4.18 Histograma de variaciones entre predicción y registro real dataset 4 horas Red original

En la Fig. 4.19 podemos observar las predicciones obtenidas por el modelo, así como los registros reales del sistema de monitoreo, donde se aprecia que los valores obtenidos se encuentran por debajo de los reales, pero siguiendo una tendencia similar.

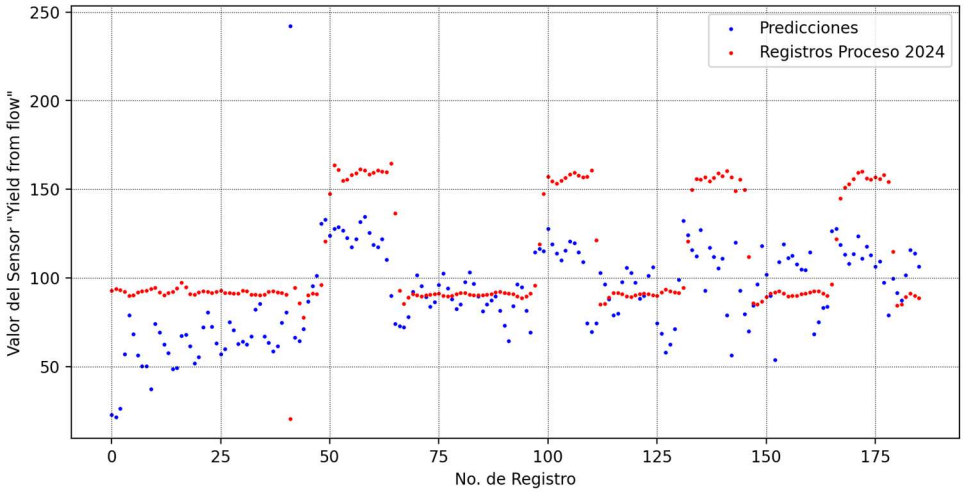

Fig. 4.19 Grafica de precisión de predicciones vs registros reales.

Se ejecutó el modelo nuevamente empleando para la estructura de la red neuronal la alternativa número 1 (ver Tabla 5.7), con el dataset de 4 horas de frecuencia. Se puede ver en la Fig. 4.20 como el valor del MSE se reduce de manera rápida en las primeras épocas de entrenamiento, para posteriormente mostrar una variación muy pequeña a lo largo de las épocas.

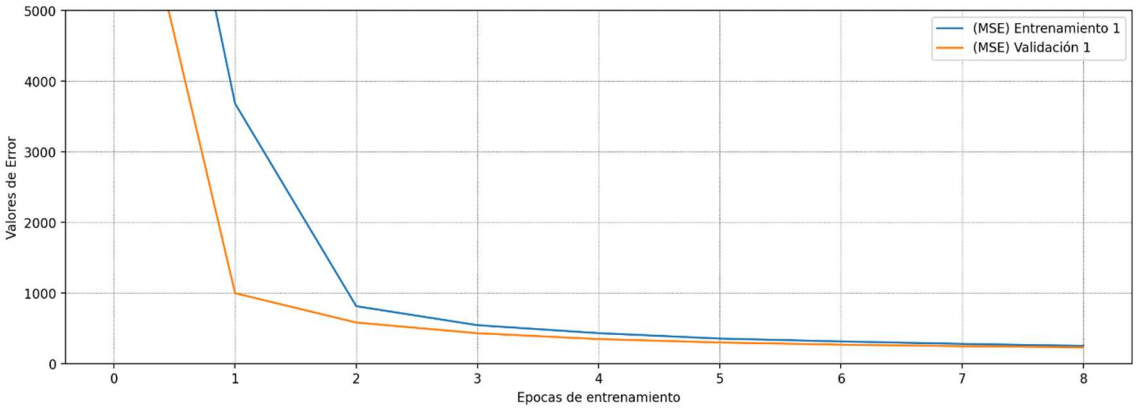

Fig. 4.20 Gráfica de MSE por épocas de red alternativa 1

De manera similar a la estructura anterior, podemos ver en la que los valores obtenidos presentan una distribución normal y no sesgada (Fig. 4.21), además las predicciones se encuentran en su mayoría en un rango entre 80% y 99%. (Fig. 4.22)

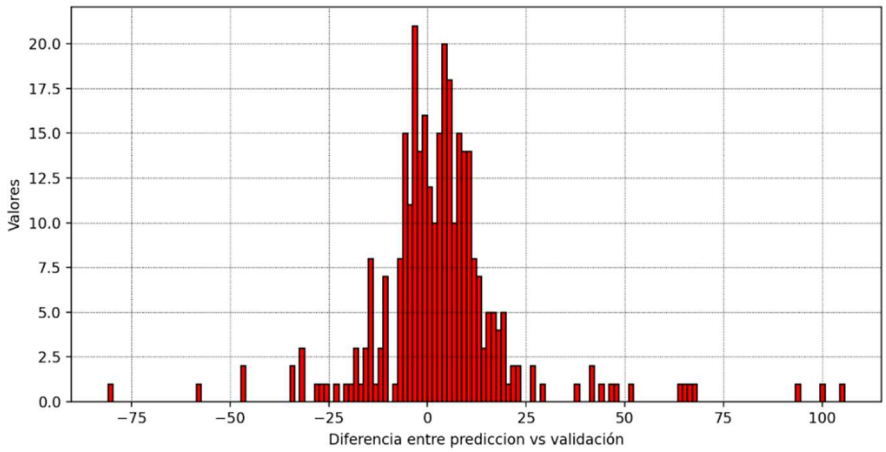

Fig. 4.21 Histograma de variaciones entre predicciones y registros reales alternativa 1

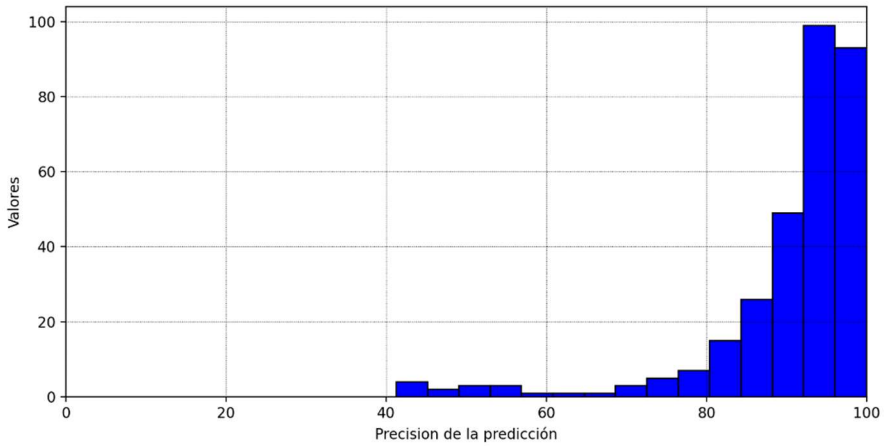

Fig. 4.22 Histograma de precisión de prediccion dataset 4 horas Red alternativa 1

En esta red se observó que los valores de las predicciones se van acercando más a los registros reales, al aumentar el número de neuronas que se consideran en cada capa oculta, manteniendo de igual forma la tendencia de los valores como se muestra en la Fig. 4.23.

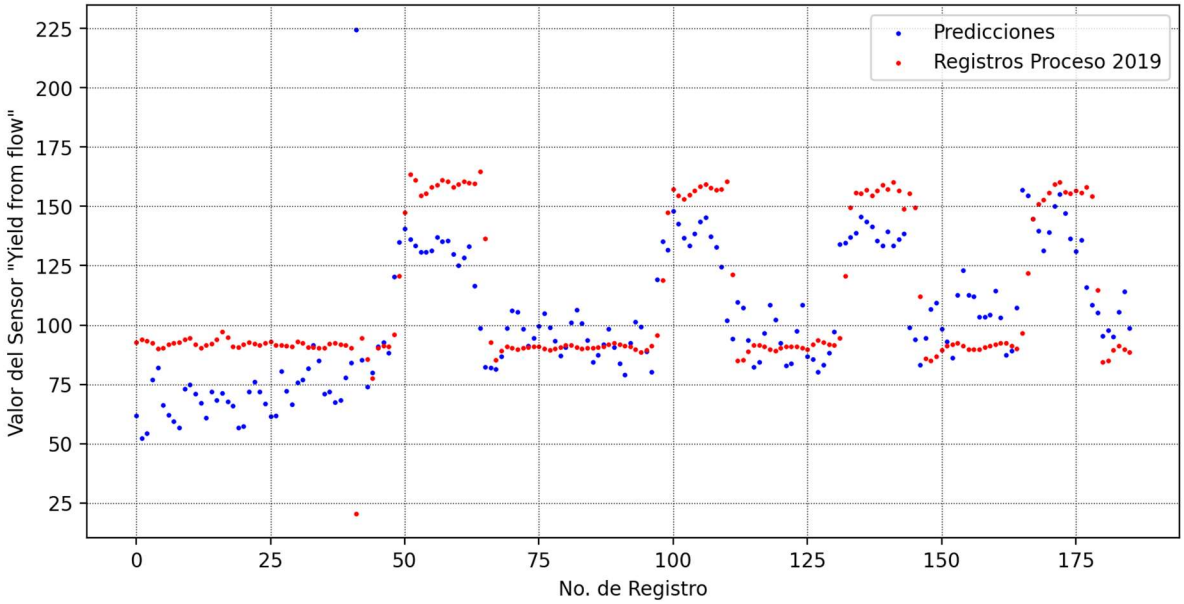

Fig. 4.23 Gráfica de precisión de predicciones vs registros reales alternativa 1

Por último, se realizó la tercer corrida con la estructura de la red neuronal con la alternativa 2 indicada en la Tabla 5.7, y representando los resultados obtenidos de MSE, distribución de las predicciones y rango de precisión de las predicciones como se muestra en las Fig. 4.24, Fig. 4.25 y Fig. 4.26, y respectivamente.

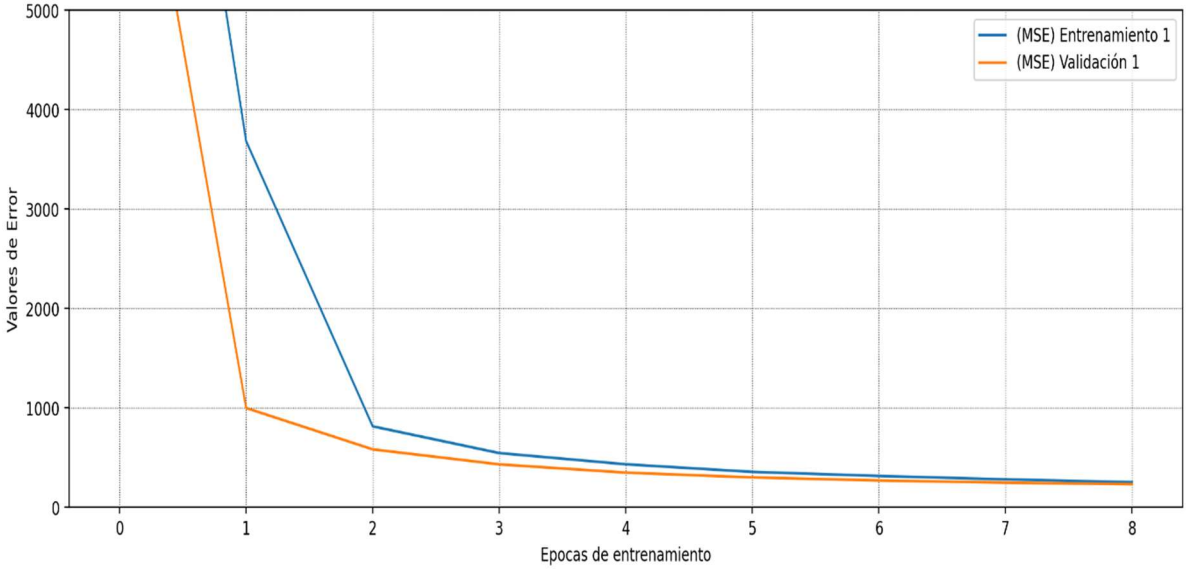

Fig. 4.24 Gráfica de MSE por épocas de red alternativa 2

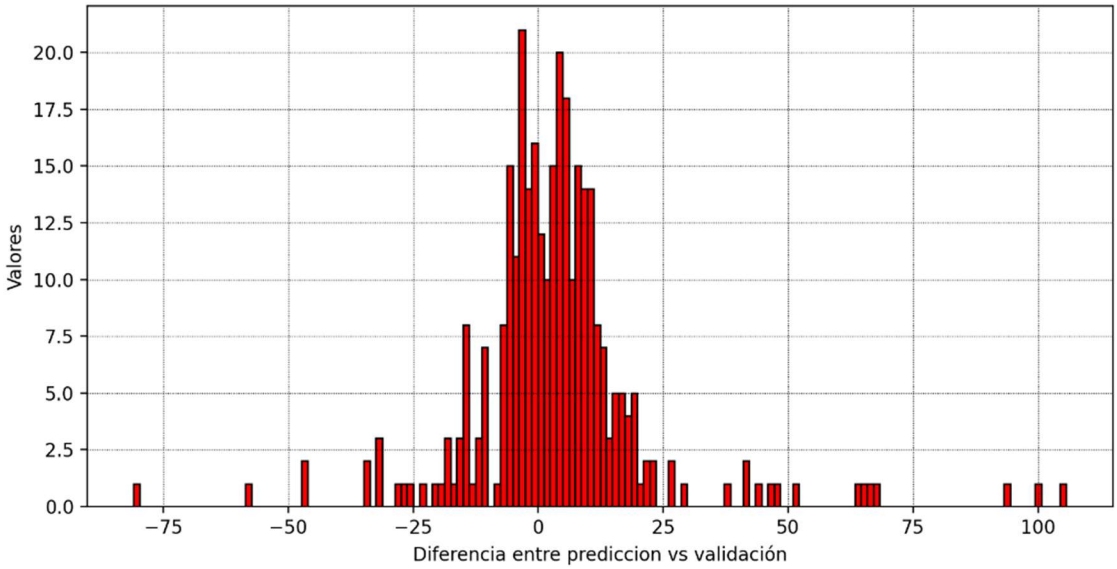

Fig. 4.25 Histograma de variaciones entre predicciones y registros reales alternativa 2

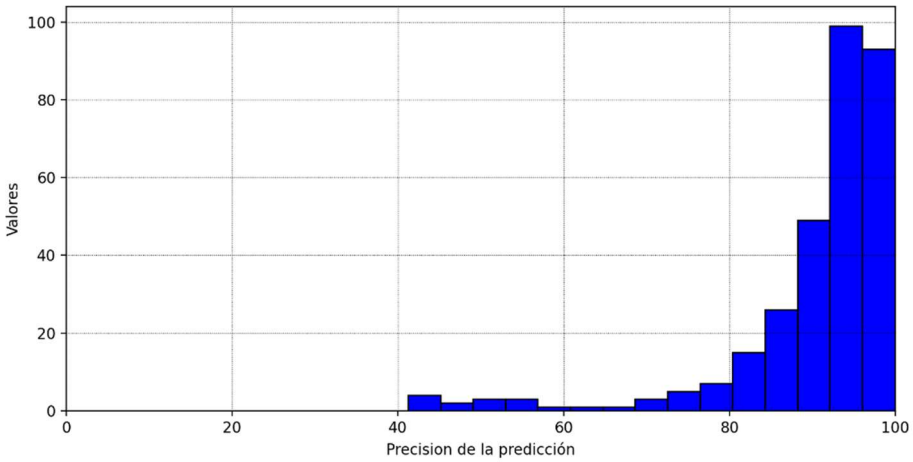

Fig. 4.26 Histograma de precisión de prediccion dataset 4 horas alternativa 2

De igual forma que en las redes propuestas original y alternativa 1, se observa que los resultados obtenidos de la red 2 presentan una distribución normal cuando se analiza la diferencia entre el valor predicho y el registro real, y se encuentra centrada en el 0, por lo cual no cuenta con sesgo a valores positivos o negativos. En la Fig. 4.27, los valores de las predicciones obtenidas por el modelo para el proceso 2019 se ajustan de mejor manera que las de las primera dos alternativas, por lo cual se considera que esta estructura presenta los mejores resultados entre las tres.

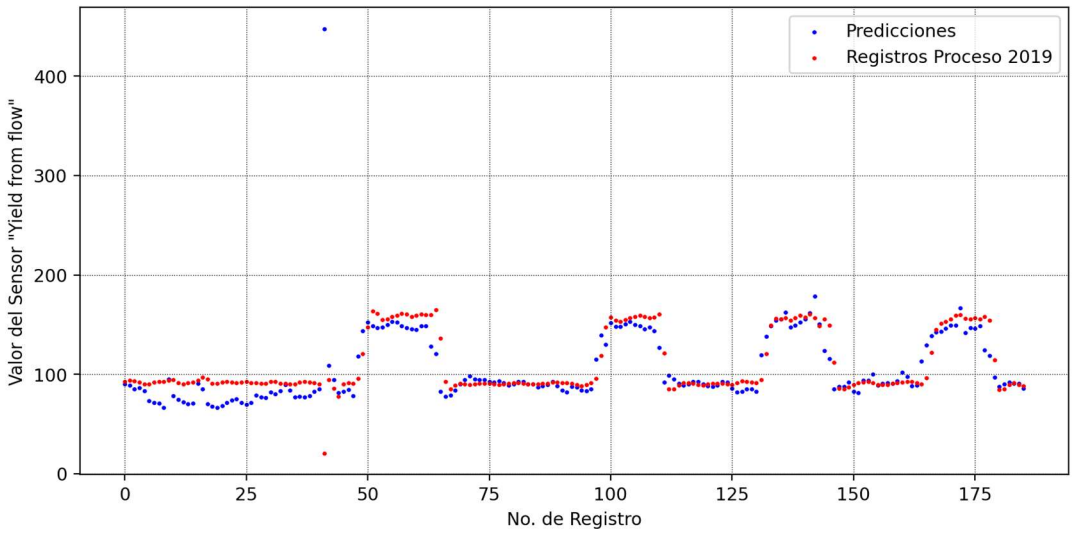

Fig. 4.27 Gráfica de precisión de predicciones vs registros alternativa 2

Con los datos obtenidos de las predicciones y los registros reales, se obtuvieron los coeficientes de determinación para cada una de las distintas variantes de estructuras

de red neuronal, donde podemos apreciar que el valor R^2 se aproxima mas a 1, y por consiguiente demuestra que el modelo cumple mejor su función, entre mayor sea el número de neuronas empleadas para entrenamiento (Ver Fig. 4.28, Fig. 4.29 y Fig. 4.30).

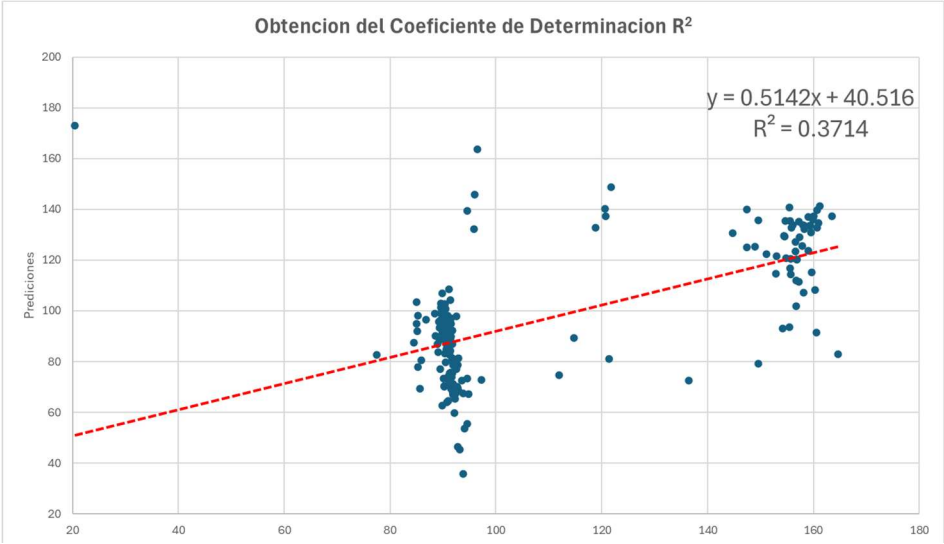

Fig. 4.28 Gráfica de coeficiente de determinación dataset 4 horas alternativa 2

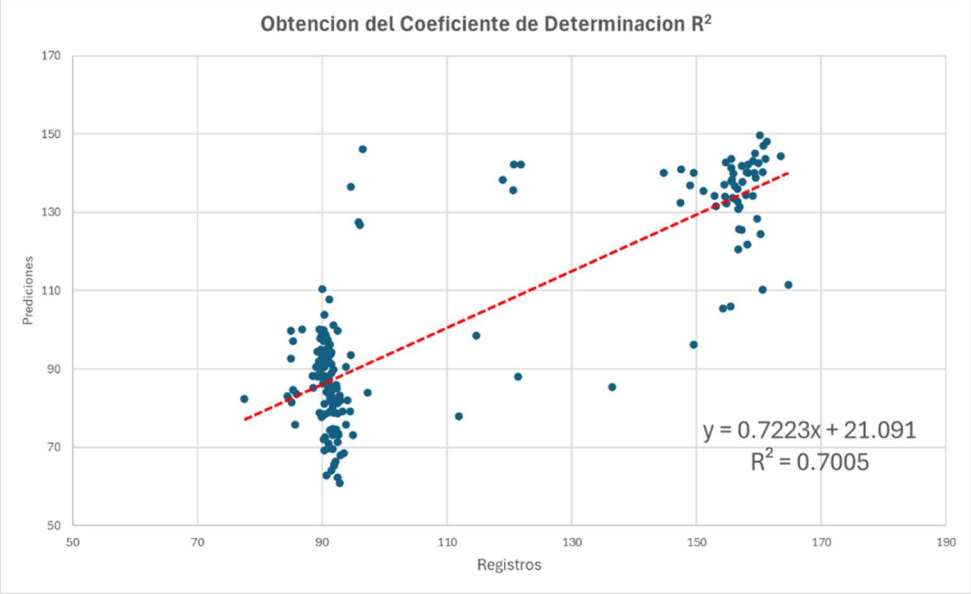

Fig. 4.29 Grafica del coeficiente de determinación dataset 4 horas alternativa 1

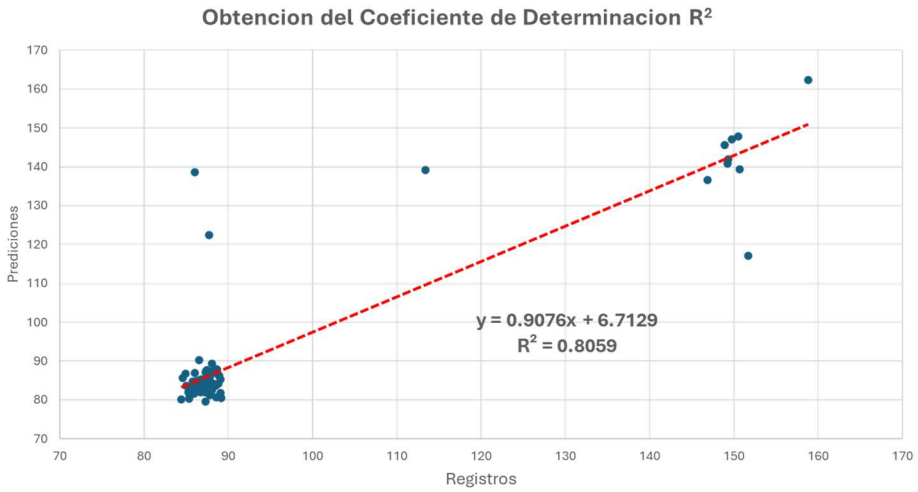

Fig. 4.30 Gráfica de coeficiente de determinación dataset 4 horas alternativa 2

Una vez que se tiene definida la estructura más eficiente de las 3 propuestas, se procede a utilizar los datos que se guardaron desde el inicio correspondientes al nuevo proceso que emplea la planta ("proceso 2024"), el cual corresponde a los valores del mes de enero de 2024 y se analizaran con el modelo seleccionado. Esto con el objetivo de determinar si el modelo es eficiente para el proceso de la planta con las modificaciones realizadas en los últimos 6 meses.

La nueva comparativa de resultados muestra que el ajuste de las predicciones contra los registros reales es más preciso, reduciendo el sesgo un poco, como se puede ver en la Fig. 4.31.

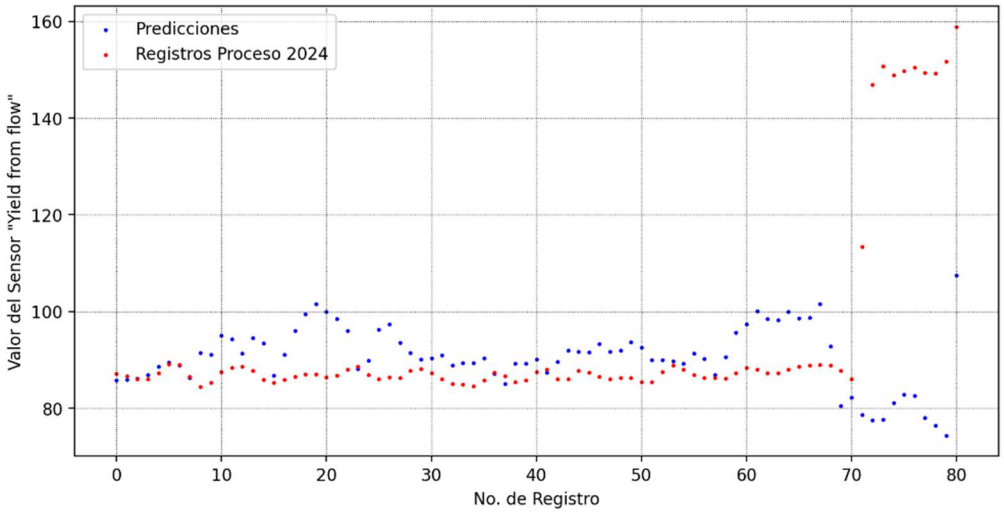

Fig. 4.31 Gráfica de precisión de predicciones vs registros reales dataset 4 horas alternativa 1

Examinando la Fig. 4.32 vemos que los valores obtenidos de predicciones mantienen una tendencia similar a los registros reales del sistema de monitoreo, pero mostrando un sesgo hacia valores superiores en los registros comprendidos entre 80 y 100 e inferiores en los registros entre 140 y 160, por lo cual se ejecuta de nuevo el modelo ahora con la estructura de la alternativa 1.

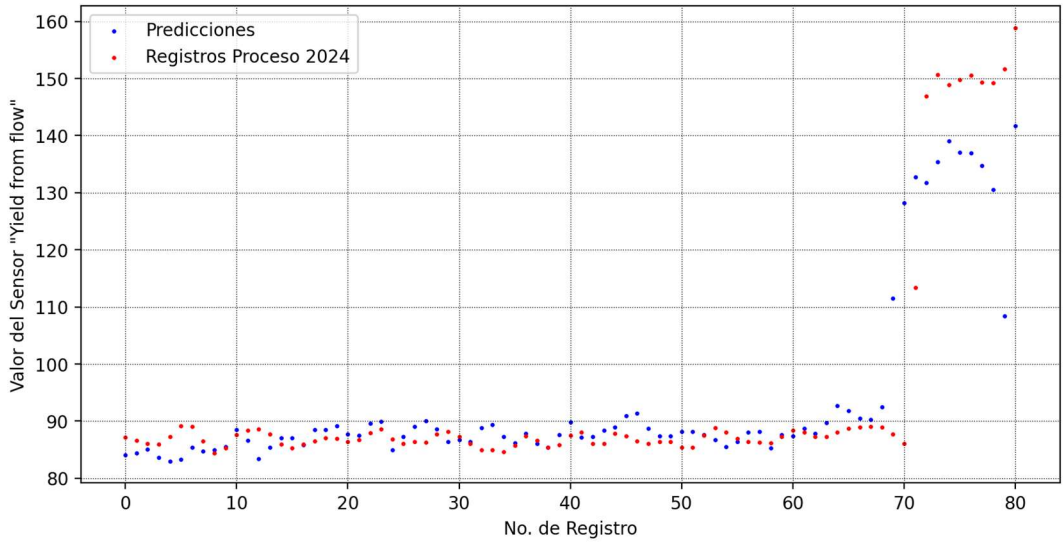

Fig. 4.32 Gráfica de precisión de predicciones vs registros reales dataset 4 horas alternativa 2

Se ejecutó la red neuronal de la alternativa 3, y obtenemos resultados que se ajustan de mejor forma a los registros reales, con lo que podemos definir que esta es la red que mejores resultados presenta (Fig. 4.33).

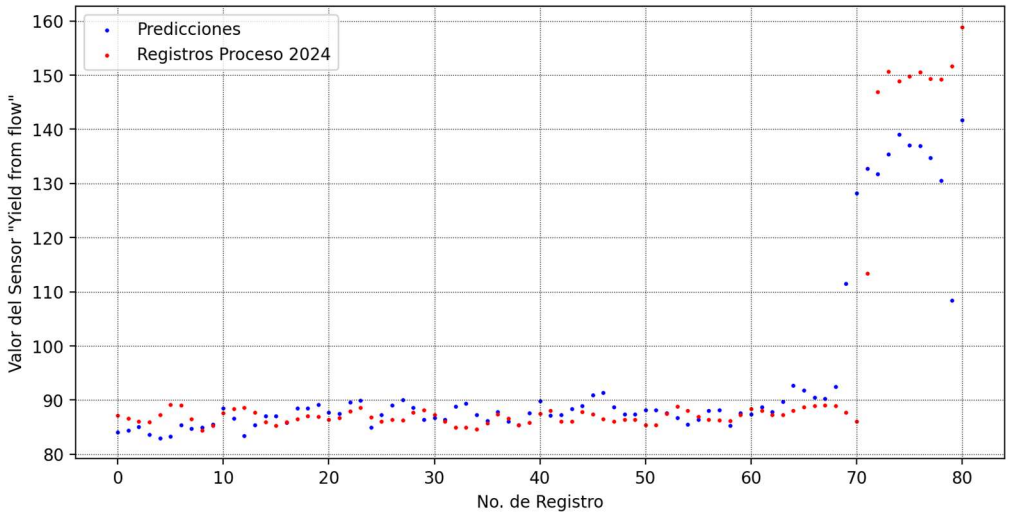

Fig. 4.33 Gráfica de precisión de predicciones vs registros reales alternativa 3

Finalmente, se obtuvieron las gráficas de coeficiente de determinación para los datos del proceso 2024, de las 3 redes propuestas, donde nuevamente se identificó un incremento en el valor obtenido de R^2, con tendencia a acercarse a 1, conforme se incrementa el número de neuronas de la red, siendo el más bajo el de la red original (Fig. 4.35) el intermedio la alternativa 1 (Fig. 4.34) y el mayor valor en la red de la alternativa 3 (Fig. 4.36).

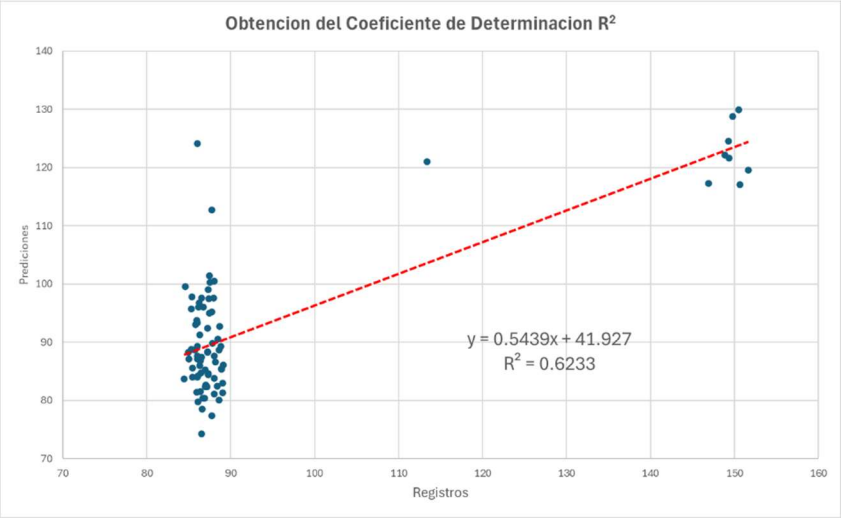

Fig. 4.35 Gráfica de coeficiente de determinación dataset 4 horas con red original (proceso 2024)

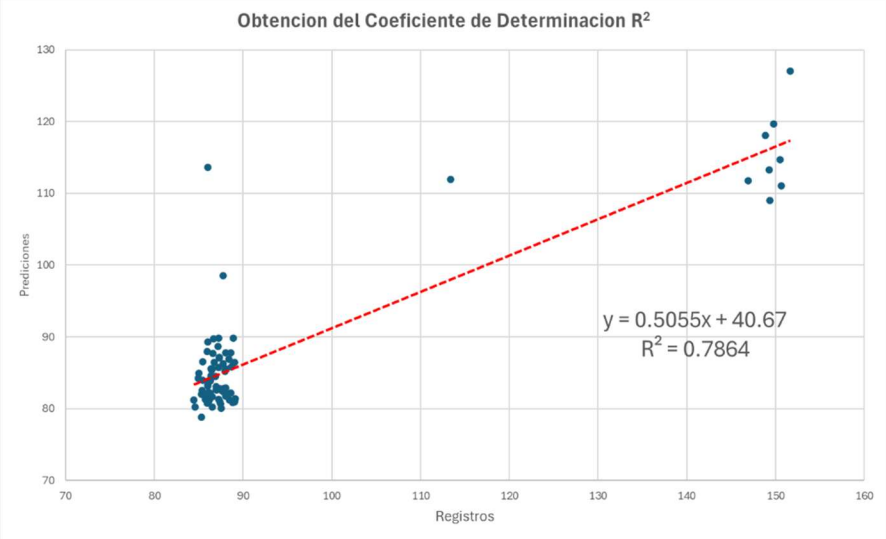

Fig. 4.34 Gráfica de coeficiente de determinación dataset 4 horas alternativa 2 (proceso 2024)

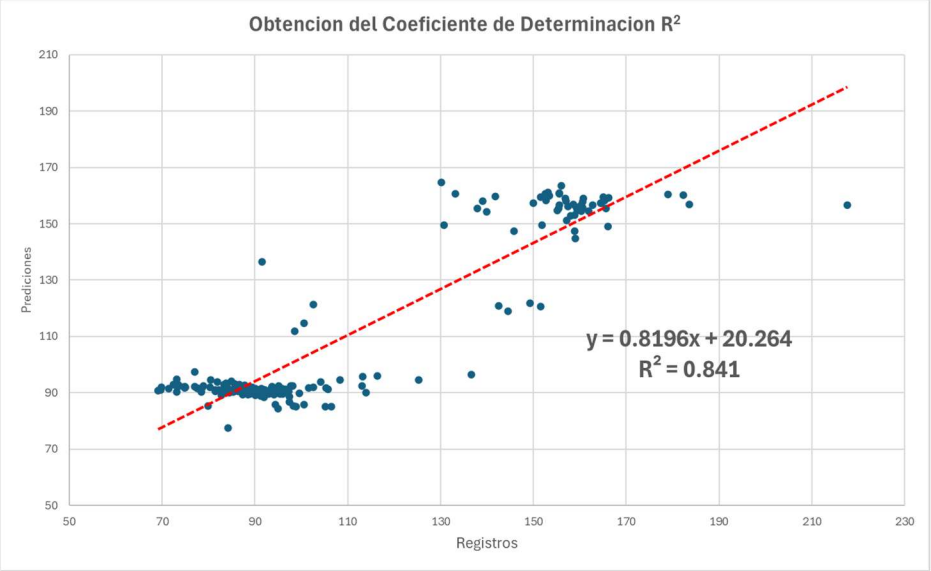

Fig. 4.36 Gráfica de coeficiente de determinación dataset 4 horas alternativa 1 (proceso 2024)

4.10 Prevención de emisión de contaminantes a la atmosfera

La predicción precisa de la producción de formaldehido usando Inteligencia Artificial, ayuda en la programación de mantenimientos preventivos del proceso, evitando de esta forma el uso de grandes cantidades de combustible y su correspondiente emisión de gases efecto invernadero al medio ambiente, ocasionados por paros inesperados en la planta debido al decaimiento del rendimiento del catalizador, de igual manera se reduce la cantidad de residuos generados al lograr disminuir la frecuencia con que se realizan paros en la planta.

Capitulo 5 Conclusiones y recomendaciones

5.1 Conclusiones

Se logró obtener predicciones precisas sobre la producción de formaldehido en la planta ARAUCO, empleando la información obtenida del sistema de monitoreo existente, analizando los registros de los sensores, y comparando estos registros contra lo obtenido del modelo, verificando las variaciones en los resultados mediante el valor del Error cuadrático medio y el Coeficiente de determinación.

Se puede emplear el modelo de redes neuronales profundas para la detección de bajas en la producción de formaldehido de la planta, aun en caso de que algunos sensores presenten fallas o lleguen estados críticos, ya que el modelo considera estos fallos en su entrenamiento, sin embargo, es deseable que los datos alimentados al modelo sean preferentemente valores válidos de sensores debidamente calibrados que reflejen el estado real de la planta.

Estas detecciones de bajas en la producción, a su vez, permitirán a los encargados de la planta, la elaboración de un programa de mantenimiento preventivo y predictivo más preciso, que servirá como apoyo a los protocolos y tiempos que ya tienen establecidos para las programaciones, tomando en cuenta su experiencia en el manejo de la planta.

El uso de modelos de inteligencia artificial, aprendizaje automático y redes neuronales profundas, son una técnica muy eficiente para analizar el comportamiento de una planta química o de cualquier industria, que cuente con la suficiente información del proceso, lo cual, debido a la modernización constante de los procesos industriales, se convertirá eventualmente en una herramienta básica e indispensable para prevenir y minimizar el impacto ocasionado al medio ambiente.

5.2 Recomendaciones

Es importante mencionar que para que el modelo genere predicciones precisas, se deberá alimentar la mayor cantidad de información posible, que a su vez cuente con datos válidos y que reflejen el comportamiento real de la planta, ya que de esto depende en gran parte la precisión de las predicciones. Para la obtención de un modelo más preciso, y más rápido es recomendable que los datos que se utilicen para alimentar al modelo sean de un periodo mayor de tiempo, o de una frecuencia menor de los registros, ya que el modelo aprenderá de los datos de mejor forma entre más información reciba. También es importante mencionar que la velocidad de obtención del modelo está directamente relacionada con la potencia computacional con que se cuente, y que, entre mayores sean lo recursos del equipo empleado para el entrenamiento del modelo, mayor será la velocidad con que el modelo quedara listo para su uso.

Capitulo 6 Bibliografía

Adams, D., Oh, D.-H., Kim, D.-W., Lee, C.-H., & Oh, M. (2020). Prediction of SOx–NOx emission from a coal-fired CFB power plant with machine learning: Plant data learned by deep neural network and least square support vector machine. *Journal of Cleaner Production*, *270*, 122310. https://doi.org/10.1016/j.jclepro.2020.122310

ATSDR, A. (1999). *Resumen de Salud Pública*. https://www.atsdr.cdc.gov: https://www.atsdr.cdc.gov/es/phs/es_phs111.pdf

Azadeh, A., Ghaderi, S. F., & Sohrabkhani, S. (2008). Annual electricity consumption forecasting by neural network in high energy consuming industrial sectors. *Energy Conversion and Management*, *49*(8), 2272–2278. https://doi.org/10.1016/j.enconman.2008.01.035

Bautista-Loaiza, A., & Ávila-Camacho, F.-J. (2023). *Predicción de la condición de hospitalización para pacientes Covid-19 utilizando modelos de clasificación Prediction of hospitalization condition for Covid-19 patients using classification models*.

Benne, M., Grondin-Perez, B., Chabriat, J.-P., & Herbe, P. (2000). *ArtiÆcial neural networks for modelling and predictive control of an industrial evaporation process*.

Bogojeski, M., Sauer, S., Horn, F., & Müller, K.-R. (2020). *Forecasting Industrial Aging Processes with Machine Learning Methods* (No. arXiv:2002.01768). arXiv. http://arxiv.org/abs/2002.01768

Bombela Jiménez, S. P., Lino Ramírez, C., Gutierrez Hernández, D. A., Zamudio Rodríguez, V. M., & Casillas Araiza, M. Á. (2019). Red Neuronal Artificial para la Clasificación y Predicción de la Calidad del Aire. *Programación Matemática y Software*, *11*(2). https://doi.org/10.30973/progmat/2019.11.2/7

Bovadilla, J. (2020). *Machine learning y deep learning: Usando python, scikit y keras*. Ediciones de la U.

Camarena Martínez, R. C., Camarena Martínez, S. C., & Barrios Sánchez, J. M. B. (2023). *Aplicación de un modelo híbrido de optimización por enjambre de partículas y red neuronal artificial para la predicción de fugas en biodigestores*.

Davies, J. C., Pattison, D., & Hirst, J. D. (2023). Machine learning for yield prediction for chemical reactions using in situ sensors. *Journal of Molecular Graphics and Modelling*, *118*, 108356. https://doi.org/10.1016/j.jmgm.2022.108356

De Casas Reyes, D. (2022). *Disminucion de la Emision de Contaminantes a la Atmosfera en la Produccion de Formaldehido utilizando PCA*. Tecnologico Nacional de Mexico / Instituto Tecnologico de Durango.

Dodhia, A., Wu, Z., & Christofides, P. D. (2021). Machine learning-based model predictive control of diffusion-reaction processes. *Chemical Engineering Research and Design*, *173*, 129–139. https://doi.org/10.1016/j.cherd.2021.07.005

Eren, B., Ileri, R., Dogan, E., Caglar, N., & Koyuncu, I. (2012). Development of artificial neural network for prediction of salt recovery by nanofiltration from textile industry wastewaters. *Desalination and Water Treatment*, *50*(1–3), 317–328. https://doi.org/10.1080/19443994.2012.719743

Fauzi, M. F. A. M., Aziz, I. A., & Amiruddin, A. (2019). The Prediction of Remaining Useful Life (RUL) in Oil and Gas Industry using Artificial Neural Network (ANN) Algorithm. *2019 IEEE Conference on Big Data and Analytics (ICBDA)*, 7–11. https://doi.org/10.1109/ICBDA47563.2019.8987015

Guo, Q., He, Z., & Wang, Z. (2023). Predicting of Daily PM2.5 Concentration Employing Wavelet Artificial Neural Networks Based on Meteorological Elements in Shanghai, China. *Toxics*, *11*(1), 51. https://doi.org/10.3390/toxics11010051

Kermet-Said, H., Ladeg, S., & Moulai-Mostefa, N. (2024). Prediction of the removal of solid suspensions and chemical oxygen demand from a pharmaceutical wastewater plant using a neural network approach. *Desalination and Water Treatment*, *317*, 100059. https://doi.org/10.1016/j.dwt.2024.100059

Kiiza, C., Pan, S., Bockelmann-Evans, B., & Babatunde, A. (2020). Predicting pollutant removal in constructed wetlands using artificial neural networks (ANNs). *Water Science and Engineering*, *13*(1), 14–23. https://doi.org/10.1016/j.wse.2020.03.005

Kumru, M., & Kumru, P. Y. (2014). Using artificial neural networks to forecast operation times in metal industry. *International Journal of Computer Integrated Manufacturing*, *27*(1), 48–59. https://doi.org/10.1080/0951192X.2013.800231

Lee, S.-C. (2003). Prediction of concrete strength using artificial neural networks. *Engineering Structures*, *25*(7), 849–857. https://doi.org/10.1016/S0141-0296(03)00004-X

Martín, O., López, M., & Martín, F. (2006). Artificial neural networks for prediction of quality in resistance spot welding. *Revista de Metalurgia*, *42*(5), 345–353. https://doi.org/10.3989/revmetalm.2006.v42.i5.32

Pajares Martinsanz, G., & Cruz Garcia, J. M. (2011). *Aprendizaje automatico.* Ediciones de la U. http://tecnacionalmx.microsite-ebooks724.com/?il=8160

Pineda Pertuz, C. (2021). *Aprendizaje automatico y profundo en python: Una mirada hacia la inteligencia artificial.* Ediciones de la U. http://tecnacionalmx.microsite-ebooks724.com/?il=18081

Ravindiran, G., Hayder, G., Kanagarathinam, K., Alagumalai, A., & Sonne, C. (2023). Air quality prediction by machine learning models: A predictive study on the indian coastal city of Visakhapatnam. *Chemosphere*, *338*, 139518. https://doi.org/10.1016/j.chemosphere.2023.139518

Rodríguez Zúñiga, M. A., & Pérez Esparza, E. (2024). Integración de la Inteligencia Artificial en el Diagnóstico y Pronóstico del Cáncer de Mama en México. *Ciencia Latina Revista Científica Multidisciplinar*, *8*(1), 3358–3377. https://doi.org/10.37811/cl_rcm.v8i1.9683

Ruiz Tamayo, J., Trasviña Osorio, J., & Rojas Mancer, E. (2024). *Detección de enfermedades en cultivos de maíz mediante imágenes con visión artificial: Un caso práctico Disease detection in corn crops through images with artificial vision: A practical case*. *24*(41).

Rustom J., A. (2012). *Estadística descriptiva, probabilidad e inferencia, Una visión conceptual y aplicada*.

Schweidtmann, A. M., Esche, E., Fischer, A., Kloft, M., Repke, J., Sager, S., & Mitsos, A. (2021). Machine Learning in Chemical Engineering: A Perspective. *Chemie Ingenieur Technik*, *93*(12), 2029–2039. https://doi.org/10.1002/cite.202100083

Siami-Irdemoosa, E., & Dindarloo, S. R. (2015). Prediction of fuel consumption of mining dump trucks: A neural networks approach. *Applied Energy*, *151*, 77–84. https://doi.org/10.1016/j.apenergy.2015.04.064

Solaimany-Aminabad, M., Maleki, A., & Hadi, M. (2013). Application of artificial neural network (ANN) for the prediction of water treatment plant influent characteristics. *Journal of Advances in Environmental Health Research*, *1*(2). https://doi.org/10.22102/jaehr.2013.40130

Stock, S., Pohlmann, S., Günter, F. J., Hille, L., Hagemeister, J., & Reinhart, G. (2022). Early Quality Classification and Prediction of Battery Cycle Life in Production Using Machine Learning. *Journal of Energy Storage*, *50*, 104144. https://doi.org/10.1016/j.est.2022.104144

Valente, G. F. S., Mendonça, R. C. S., Pereira, J. A. M., & Felix, L. B. (2014). Artificial neural network prediction of chemical oxygen demand in dairy industry effluent treated by electrocoagulation. *Separation and Purification Technology*, *132*, 627–633. https://doi.org/10.1016/j.seppur.2014.05.053

Wu, G.-D., & Lo, S.-L. (2008). Predicting real-time coagulant dosage in water treatment by artificial neural networks and adaptive network-based fuzzy inference system. *Engineering Applications of Artificial Intelligence*, *21*(8), 1189–1195. https://doi.org/10.1016/j.engappai.2008.03.015

ANEXO 1

Código en lenguaje Python para la predicción de la producción sustentable de formaldehido en tiempo real a través de la actividad del catalizador utilizando aprendizaje automático

Archivo: 1_seg.py

```
#-----------------------------------------------------------------------------------------------------
#PREPROCESAMIENTO

#importamos las librerías
import tkinter as tk
from tkinter import filedialog
import os
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

#Definimos la función
def select_folder():
    global data_file_folder
    data_file_folder = filedialog.askdirectory()
    root.destroy()

# Crear ventana
root = tk.Tk()
root.title("Seleccionar Carpeta")
root.geometry("400x200")
root.configure(bg="gray")

# Etiqueta
```

```
label = tk.Label(root, text="Haz clic para seleccionar una carpeta con archivo de datos", bg="gray")
label.pack(pady=20)

# Botón
button = tk.Button(root, text="Seleccionar Carpeta", command=select_folder)
button.pack(pady=10)

# Variable para almacenar la carpeta seleccionada
data_file_folder = None

# Loop de la aplicación
root.mainloop()

# Una vez que la ventana se cierra, el programa continúa aquí
print("Carpeta seleccionada:", data_file_folder)

#Generamos una lista que contenga todos los dataframes de los archivos dentro de la carpeta y se les da formato según lo requerido
lista = []
for file in os.listdir(data_file_folder):
    if file.endswith('.xlsx'):
        print('Cargando archivos {0}...'.format(file))
        lista.append(pd.read_excel(os.path.join(data_file_folder, file), sheet_name=1, header=3))

for i in lista:
    # Set the first column as the index
    i.set_index('Date', inplace=True)
```

```
#Unimos los archivos ya formateados
df_inicial = pd.concat(lista, axis=0)

# Guardamos los datos en un archivo nuevo
# Guardar como csv
df_inicial.to_csv('Datos_completos_1seg.csv')

# #Guardar como excel
# df_inicial.to_excel('Datos_completos_1seg.xlsx')

#_____________________________________________________________________________
# Leer la base de datos
df = pd.read_csv('Datos_completos_1seg.csv',  na_values= ['???'], index_col=0)
df = df.fillna(0)
# df.iloc[:, 212:] = df.iloc[:, 212:].astype(float)

df.drop(df[df['Yield from flow'] >= 1000].index, inplace = True)
df.drop(df[df['Yield from flow'] <= 0].index, inplace = True)

df_types = df.dtypes
df_info = df.describe()
# Identificar cuantos valores perdidos tenemos en cada columna
total_Nan = df.isnull().sum()

#Remover columnas con valores de sensores OUT y SetPoint
df2 = df.drop(df.filter(regex='OUT.CV').columns, axis=1)
df2 = df2.drop(df2.filter(regex='SP.CV').columns, axis=1)
# df2 = df2.drop(df2.filter(regex='Date').columns, axis=1)

df2 = df2.dropna()
```

```
df2 =df2[df2['PROD_MODE/PROD_MODE.CV'] !=0]
df2 =df2[df2['PROD_MODE/UFC_MODE.CV'] !=0]

#convertir columnas de texto en números

# from sklearn import preprocessing
# le = preprocessing.LabelEncoder()
# le.fit(df2['PROD_MODE/PROD_MODE.CV'])
#
df2['PROD_MODE/PROD_MODE.CV']=le.transform(df2['PROD_MODE/PROD_MOD
E.CV'])
# le.fit(df2['PROD_MODE/UFC_MODE.CV'])
#
df2['PROD_MODE/UFC_MODE.CV']=le.transform(df2['PROD_MODE/UFC_MODE.C
V'])

#describir el dataset y ver sus propiedades, media, std, mediana, min y max.
df2_info = df2.describe()

#obtener tipo de datos dentro de cada columna para poder eliminar las no numéricas
a = df2.dtypes
df3 = df2.select_dtypes(exclude=['object'])

# df3=df2
# Obtener desviaciones estándar de las columnas
desvstd = df3.std()
df3_info = df3.describe()

#Eliminar columnas con desviación estándar igual a 0
df4 = df3.drop(df3.std()[(df3.std() <= 0.1)].index, axis = 1)
```

```
df4_info = df4.describe()

#Eliminar columnas con valores todos negativos
threshold = 0.0
# loop a traves de cada columna del dataframe
for col in df4:
    # obtener el valor máximo de cada columna
    # verificar si es menor o igual que el rango establecido
    if df4[col].max() < threshold:
        # si es menor, se elimina la columna
        df4 = df4.drop([col], axis=1)

#Eliminar columnas con 25% o más de valores en ceros
ceros = df4[df4 == 0].count(axis=0)/len(df.index)
df5 = df4.loc[:, (df==0).mean() < .25]
# df5.to_excel('df5_1seg.xlsx')

# #Guardar archivo de Excel de cada dataframe después de eliminar columnas
# df2.to_excel('df2_4horas.xlsx')
# Identificar cuantos valores perdidos tenemos en cada columna
total_Nan = df4.isnull().sum()

 #Importar librería para la normalización estándar
# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# df5_sc = scaler.fit_transform(df5)

# df5_sc = pd.DataFrame(df5_sc)
# # df5_sc.to_excel('df5_escalado_1seg.xlsx')
# df5_sc_info = df5_sc.describe()
```

```
# #imprimir todos los histogramas juntos cuando no es array
# df5.hist(bins=100, figsize=(30, 30))
# df5_sc.hist(bins=100, figsize=(30, 30))

# #imprimir histogramas separados automáticamente
# count = 0
# for col in df5.columns:
#     plt.figure(figsize=(10,5))
#     df5[col].plot.hist(bins = 150, color= 'blue', edgecolor = 'black')
#     dev = df5[col].std()
#     mean = df5[col].mean()
#      plt.xlabel('Desv. Standard[\u03C3] ='+'{:.3f}'.format(dev) + '          media[\u03BC]
='+'{:.3f}'.format(mean))
#     plt.ylabel('Valores')
#     plt.legend()
#     plt.axvline(df5[col].mean(), color='red', linestyle='dashed', linewidth=1)
#     plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
#                                         plt.savefig("C:/Users/migue/Documentos/Maestria/3er
Semestre/Figuras/Original/fig_"+str(count)+".png")
#     count+=1
#     plt.show()

# #imprimir histogramas separados automáticamente
# count = 0
# for col in df5_sc.columns:
#     plt.figure(figsize=(10,5))
#     df5_sc[col].plot.hist(bins = 150, color='green', edgecolor = 'black')
#     dev2 = df5_sc[col].std()
#     mean_og = df5.iloc[:, count].mean()
```

```
#     mean2 = df5_sc[col].mean()
#      plt.xlabel('Desv. Standard[\u03C3] ='+'{:.3f}'.format(dev2) + '         media[\u03BC]
='+'{:.3f}'.format(mean2) + ' = '+' {:.3f}'.format(mean_og))
#     plt.ylabel('Valores')
#     plt.legend()
#     plt.axvline(df5_sc[col].mean(), color='red', linestyle='dashed', linewidth=1)
#     plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
#                                         plt.savefig("C:/Users/migue/Documentos/Maestria/3er
Semestre/Figuras/Standarizado/fig_sc_"+str(count)+".png")
#     count+=1
#     plt.show()

#Eliminar columnas con desviación estándar mayores de 3 y menores de -3
#para quedarnos con solo el 99.7% de la información y alfa = 0.01
#-----------------------------------------------------------------------------------------------------
# df10 = df5
# df10_sc = df5_sc
# df5_info = df5.describe()

# col_borrar = 0

# for col in df10_sc.columns:
#     #Crear lista de valores que estan dentro del rango +-3 desv std
#     k = df10_sc[df10_sc.iloc[:,col_borrar]<-3].index
#     j = df10_sc[df10_sc.iloc[:,col_borrar]>3].index
#     l = j.union(k)

# #Crea nuevo dataframe escalado después de remover filas de la lista anterior
#     df11 = df10_sc.drop(df10_sc.index[l])
```

```
# #Crea nuevo dataframe sin escalar despues de remover filas de la lista anterior
#     df10 = df10.drop(df10.index[l])

# #Estandarizar el nuevo dataframe para obtener las nuevas desv. std
#     df11_sc = scaler.fit_transform(df10)
#     df10_sc = pd.DataFrame(df11_sc)

# #Repetir proceso con siguiente columna
#     col_borrar+=1

# #imprimir todos los histogramas juntos cuando no es array
# df11.hist(bins=100, figsize=(30, 30))
# df10.hist(bins=100, figsize=(30, 30))

# #imprimir histogramas separados automáticamente
# count = 0
# for col in df11.columns:
#     plt.figure(figsize=(10,5))
#     df11[col].plot.hist(bins = 150, color='purple', edgecolor = 'black')
#     dev3 = df11[col].std()
#     mean_og = df5.iloc[:, count].mean()
#     mean3 = df11[col].mean()
#      plt.xlabel('Desv. Standard[\u03C3] ='+'{:.3f}'.format(dev3) + '         media[\u03BC]
='+'{:.3f}'.format(mean3) + ' = '+ {:.3f}'.format(mean_og))
#     plt.ylabel('Frecuencia')
#     plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
#     plt.legend()
#     plt.axvline(df11[col].mean(), color='red', linestyle='dashed', linewidth=1)
#                                         plt.savefig("C:/Users/migue/Documentos/Maestria/3er
Semestre/Figuras/Standarizado/fig_final_"+str(count)+".png")
```

```
#     count+=1
#     plt.show()

# df11.to_excel('df11_1seg.xlsx')
# df10.to_excel('df10_2seg.xlsx')
# df10_info = df10.describe()

#Eliminar columnas con desviación estándar igual a 0
df12 = df5.drop(df5.std()[(df5.std()== 0)].index, axis = 1)
df12_info = df12.describe()
# df12.to_excel('preprocesado.xlsx')

#cambiar nombre de variable
datos_iniciales=df12
# datos_iniciales.to_excel('Datos_limpios.xlsx')

#eliminar columna date
datos_iniciales = datos_iniciales.reset_index()
datos_iniciales = datos_iniciales.drop('Date', axis =1 )

datos_iniciales2 = datos_iniciales[datos_iniciales['Yield from flow'] !=0]
datos_iniciales2_info = datos_iniciales2.describe()

y = datos_iniciales2['Yield from flow']
x = datos_iniciales2.drop('Yield from flow', axis = 1)

x = np.asarray(x)
y = np.asarray(y)

# x = x.reshape (-1,136)
```

```
y = y.reshape (-1,1)

neuronas = len(datos_iniciales2.axes[1])-1

#--------------------------------------------------------------------------
# RED NEURONAL 1

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import InputLayer, Dense
from sklearn.model_selection import train_test_split

#se leen los valores de x, y del disco para separar en 90%-10%
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.1)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

arauco = Sequential()
arauco.add(InputLayer((neuronas)))
arauco.add(Dense(10, activation = 'relu'))
arauco.add(Dense(20, activation = 'relu'))
arauco.add(Dense(30, activation = 'relu'))
arauco.add(Dense(40, activation = 'relu'))
# arauco.add(Dense(90, activation = 'relu'))
arauco.add(Dense(1, activation = 'relu'))

#Se usa mean squared error para calcular el error vs el dato real
```

```
arauco.compile(optimizer = 'adam',
               loss = 'mse',
               metrics = ['mae'])

#Crear un callback que guarde el mejor modelo
#Se indica la ruta en caso de que se quiera indicar en la siguiente línea de código
#model_path = os.path.join(SUBDIR, 'tmp/checkpoint')
#Se guarda el modelo con el mejor valor monitoreando la precisión de la accuracy
save_model_callback                                                              =
tf.keras.callbacks.ModelCheckpoint(filepath=r'\Users\migue\Documentos\Maestria\4o
Semestre\1 Segundo',
                                                  monitor='val_accuracy',
                                                  mode='auto',
                                                  save_best_only=True)

#Se guarda la instrucción para evitar el sobre-entrenamiento
avoid_overtraining = tf.keras.callbacks.EarlyStopping(patience=3, monitor='val_loss')
my_callbacks = [save_model_callback, avoid_overtraining]

#Se pueden generar alertas relacionadas al guardar el estado del optimizador
#Estas advertencias estan para desalentar el uso caducado, y pueden ser ignoradas

history = arauco.fit(X_train, y_train, batch_size = 150, epochs =10, validation_split =
0.2, callbacks = my_callbacks)

prediccion_red1 = arauco.predict(X_test)
error_red1 = y_test - prediccion_red1

plt.figure(figsize=(10,5))
plt.hist(error_red1,bins = 200, color='red', edgecolor = 'black')
```

```
plt.xlabel('Diferencia entre prediccion vs validación', loc= 'center')
plt.ylabel('Valores')
plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
plt.xlim((-5,5))
plt.show()

porcentaje_error_red1 = (error_red1 / y_test)*100
eficiencia_red1 = 100-abs(porcentaje_error_red1)

error_red1_1df = pd.DataFrame(error_red1)
eficiencia_red1_df = pd.DataFrame(eficiencia_red1)
prediccion_red1_df = pd.DataFrame(prediccion_red1)
val_real_red1_df = pd.DataFrame(y_test)

error_red1_1df.to_csv('error_1segundo.csv')
eficiencia_red1_df.to_csv('eficiencia_1segundo.csv')

# eficiencia_red1_df = eficiencia_red1_df.drop(eficiencia_red1_df.index[287])
# eficiencia_red1_df.drop(eficiencia_red1_df.index[163], inplace=True)

prediccion_red1_df.to_csv('prediccion_1segundo.csv')
val_real_red1_df.to_csv('valor_real_1segundo.csv')

#Histograma de precisión
plt.figure(figsize=(10,5))
plt.hist(eficiencia_red1_df, bins = 50, color='blue', edgecolor = 'black', label =
'Histograma')
plt.xlabel('Precision de la predicción', loc= 'center')
plt.ylabel('Valores')
plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
```

```
plt.xlim((80,100))
plt.show()

#Verificar si hay overfitting
plt.figure(figsize=(15,5))
plt.plot(history.history['loss'], label = '(MSE) Entrenamiento 1')
plt.plot(history.history['val_loss'], label = '(MSE) Validación 1')
plt.xlabel('Epocas de entrenamiento')
plt.ylabel('Valores de Error')
plt.legend()
plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
plt.ylim((0,25.0))
#
plt.xticks([0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50]
)

val_loss1 = history.history['val_loss']
loss1= history.history['loss']

val_loss_pd = pd.DataFrame(val_loss1)
val_loss_pd.to_csv('val_loss_1seg.csv')
loss_pd = pd.DataFrame(loss1)
loss_pd.to_csv('loss_1seg.csv')

arauco.save('arauco_1seg.h5')
arauco.summary()
arauco.get_weights()
arauco.optimizer
```

```
# Grabar a disco el motor de escalamiento
import joblib
scaler_filename = "scaler_1seg.save"
joblib.dump(scaler, scaler_filename)

# # Grabar a disco el motor de labels
# import joblib
# label_filename = "label_4hrs.save"
# joblib.dump(le, label_filename)

col_predic = pd.DataFrame(datos_iniciales2.columns.tolist())

col_predic.to_csv('Variables para prediccion_1seg.csv')



#-------------------------------------------------------------------------------------------

# # Leer tambien el motor de escalamiento
# import joblib
# scaler2 = joblib.load('scaler_4hrs.save')
from sklearn.utils.validation import check_is_fitted
check_is_fitted(scaler)

import tensorflow as tf

#Cargar el modelo guardado
model = tf.keras.models.load_model('arauco_1seg.h5')

model.summary()
model.get_weights()
```

```
model.optimizer

import pandas as pd
# Leer la base de datos
variables = pd.read_csv('Variables para prediccion_1seg.csv', sep = ',', index_col=0)
# Leer la base de datos
variables = col_predic
#Indicamos la carpeta donde estan los datos que queremos predecir

#importamos las librerias
import tkinter as tk
import os
from tkinter import filedialog
import numpy as np

#Definimos la función
def select_folder():
    global data_file_folder
    data_file_folder = filedialog.askdirectory()
    root.destroy()

# Crear ventana
root = tk.Tk()
root.title("Seleccionar Carpeta")
root.geometry("400x200")
root.configure(bg="gray")

# Etiqueta
label = tk.Label(root, text="Haz clic para seleccionar una carpeta con archivo de datos", bg="gray")
```

```
label.pack(pady=20)

# Botón
button = tk.Button(root, text="Seleccionar Carpeta", command=select_folder)
button.pack(pady=10)

# Variable para almacenar la carpeta seleccionada
data_file_folder = None

# Loop de la aplicación
root.mainloop()

# Una vez que la ventana se cierra, el programa continúa aquí
print("Carpeta seleccionada:", data_file_folder)

#Generamos una lista que contenga todos los Dframes de los archivos dentro de la carpeta
#y se les da formato segun lo requerido
lista = []
for file in os.listdir(data_file_folder):
    if file.endswith('.xlsx'):
        print('Cargando archivos {0}...'.format(file))
        lista.append(pd.read_excel(os.path.join(data_file_folder, file), sheet_name=1, header=3))

for x in lista:
    # Set the first column as the index
    x.set_index('Date', inplace=True)

#Unimos los archivos ya formateados
```

```
df_inicial = pd.concat(lista, axis=0)

#Guardamos los datos en un archivo nuevo
#Guardar como excel
# df_inicial.to_excel('Datos_completos.xlsx')

#Guardar como CSV
df_inicial.to_csv('Datos_formateados_1seg.csv')

#cargar y dar formato al archivo csv
df_inicial = pd.read_csv('Datos_formateados_1seg.csv', sep = ',', na_values= ['???'],
index_col=0)
df_inicial =  df_inicial.fillna(0)
df_inicial = df_inicial.loc[(df_inicial!=0).any(axis=1)]

#convertir columnas de texto en numeros

# from sklearn import preprocessing
# le = preprocessing.LabelEncoder()

# # Leer también el motor de escalamiento
# import joblib
# le = joblib.load('label_1seg.save')

# le.fit(df_inicial['PROD_MODE/PROD_MODE.CV'])
#
df_inicial['PROD_MODE/PROD_MODE.CV']=le.transform(df_inicial['PROD_MODE/P
ROD_MODE.CV'])

# le.fit(df_inicial['PROD_MODE/UFC_MODE.CV'])
```

```
#
df_inicial['PROD_MODE/UFC_MODE.CV']=le.transform(df_inicial['PROD_MODE/UFC_MODE.CV'])

#obtener tipo de datos dentro de cada columna para poder eliminar las no numericas
r = df_inicial.dtypes
df_inicial = df_inicial.select_dtypes(exclude=['object'])
df_inicial.drop(df_inicial[df_inicial['Yield from flow'] >= 1000].index, inplace = True)
df_inicial.drop(df_inicial[df_inicial['Yield from flow'] <= 0].index, inplace = True)
variables = col_predic[0].tolist()
df_ajustado = df_inicial[df_inicial.columns.intersection(variables)]
#eliminar columna date
df_ajustado = df_ajustado.reset_index()
df_ajustado = df_ajustado.drop('Date', axis =1 )
df_ajustado = df_ajustado[df_ajustado['Yield from flow'] !=0]
df_ajustado_info = df_ajustado.describe()

#definir x, y
y2 = df_ajustado['Yield from flow']
x2 = df_ajustado.drop('Yield from flow', axis = 1)

#convertir en array
x2 = np.asarray(x2)
y2 = np.asarray(y2)

#dar formato al array
y2 = y2.reshape (-1,1)

#Predicción
x_pred = scaler.transform(x2)
```

```
#realizar la predicción
predicciones = arauco.predict(x_pred)

y3 = pd.DataFrame(y2)
x3 = pd.DataFrame(predicciones)

#imprimir comparativa entre predicciones y registros reales
plt.figure(figsize=(10,5))
plt.legend()
plt.scatter(x3.index, x3, color='blue', edgecolor = 'blue', label = 'Predicciones', s= 2)
plt.scatter(y3.index, y3, color='red', edgecolor = 'red', label = 'Registros Proceso 2019',s= 2)
plt.xlabel('No. de Registro', loc= 'center')
plt.ylabel('Valor del Sensor "Yield from flow"')
plt.legend()
plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
plt.savefig(r'\Users\migue\Documentos\Maestria\4o Semestre\1 Segundo\Figuras\Comparativa.png')
plt.show()
```

Código en lenguaje Python para la predicción de la producción sustentable de formaldehido en tiempo real a través de la actividad del catalizador utilizando aprendizaje automático

Archivo: 4_horas.py

```
#PREPROCESAMIENTO

#importamos las librerías
import tkinter as tk
from tkinter import filedialog
```

```
import os
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

#Definimos la funcion
def select_folder():
    global data_file_folder
    data_file_folder = filedialog.askdirectory()
    root.destroy()

# Crear ventana
root = tk.Tk()
root.title("Seleccionar Carpeta")
root.geometry("400x200")
root.configure(bg="gray")

# Etiqueta
label = tk.Label(root, text="Haz clic para seleccionar una carpeta con archivo de datos", bg="gray")
label.pack(pady=20)

# Botón
button = tk.Button(root, text="Seleccionar Carpeta", command=select_folder)
button.pack(pady=10)

# Variable para almacenar la carpeta seleccionada
data_file_folder = None
```

```
# Loop de la aplicación
root.mainloop()

# Una vez que la ventana se cierra, el programa continúa aquí
print("Carpeta seleccionada:", data_file_folder)

#Generamos una lista que contenga todos los Dframes de los archivos dentro de la carpeta
#y se les da formato segun lo requerido
lista = []
for file in os.listdir(data_file_folder):
    if file.endswith('.xlsx'):
        print('Cargando archivos {0}...'.format(file))
        lista.append(pd.read_excel(os.path.join(data_file_folder, file), sheet_name=1, header=1))

for i in lista:
    # Set the first column as the index
    i.set_index('Date', inplace=True)



#Unimos los archivos ya formateados
df_inicial = pd.concat(lista, axis=0)

#Guardamos los datos en un archivo nuevo
#Guardar como csv
# df_inicial.to_csv('Datos_completos_4Horas.csv')
```

```
#Guardar como excel
df_inicial.to_excel('Datos_completos_4Horas.xlsx')

#_____
_____
# Leer la base de datos
df = pd.read_excel('Datos_completos_4Horas.xlsx',  na_values= ['???'], index_col=0)
df = df.fillna(0)
# df.iloc[:, 212:] = df.iloc[:, 212:].astype(float)

df.drop(df[df['Yield from flow'] >= 1000].index, inplace = True)
df.drop(df[df['Yield from flow'] <= 0].index, inplace = True)

df_types = df.dtypes
df_info = df.describe()
# Identificar cuantos valores perdidos tenemos en cada columna
total_Nan = df.isnull().sum()

#Remover columnas con valores de sensores OUT y SetPoint
df2 = df.drop(df.filter(regex='OUT.CV').columns, axis=1)
df2 = df2.drop(df2.filter(regex='SP.CV').columns, axis=1)
# df2 = df2.drop(df2.filter(regex='Date').columns, axis=1)

df2 = df2.dropna()
df2 =df2[df2['PROD_MODE/PROD_MODE.CV'] !=0]
df2 =df2[df2['PROD_MODE/UFC_MODE.CV'] !=0]

#convertir columnas de texto en numeros

# from sklearn import preprocessing
```

```
# le = preprocessing.LabelEncoder()
# le.fit(df2['PROD_MODE/PROD_MODE.CV'])
#
df2['PROD_MODE/PROD_MODE.CV']=le.transform(df2['PROD_MODE/PROD_MODE.CV'])
# le.fit(df2['PROD_MODE/UFC_MODE.CV'])
#
df2['PROD_MODE/UFC_MODE.CV']=le.transform(df2['PROD_MODE/UFC_MODE.CV'])


#describir el dataset y ver sus propiedades, media, std, mediana, min y max
df2_info = df2.describe()

#obtener tipo de datos dentro de cada columna para poder eliminar las no numéricas
a = df2.dtypes
df3 = df2.select_dtypes(exclude=['object'])

# df3=df2
# Obtener desviaciones estándar de las columnas
desvstd = df3.std()
df3_info = df3.describe()

#Eliminar columnas con desviación estándar igual a 0
df4 = df3.drop(df3.std()[(df3.std() <= 0.1)].index, axis = 1)
df4_info = df4.describe()

#Eliminar columnas con valores todos negativos
threshold = 0.0
# loop a través de cada columna del dataframe
```

```
for col in df4:
    # obtener el valor máximo de cada columna
    # verificar si es menor o igual que el rango establecido
    if df4[col].max() < threshold:
        # si es menor, se elimina la columna
        df4 = df4.drop([col], axis=1)

#Eliminar columnas con 25% o más de valores en ceros
ceros = df4[df4 == 0].count(axis=0)/len(df.index)
df5 = df4.loc[:, (df==0).mean() < .25]
# df5.to_excel('df5_1seg.xlsx')

# #Guardar archivo de Excel de cada dataframe después de eliminar columnas
# df2.to_excel('df2_4horas.xlsx')
# Identificar cuantos valores perdidos tenemos en cada columna
total_Nan = df4.isnull().sum()

# #Importar librería para la normalización estándar
# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# df5_sc = scaler.fit_transform(df5)

# df5_sc = pd.DataFrame(df5_sc)
# # df5_sc.to_excel('df5_escalado_1seg.xlsx')
# df5_sc_info = df5_sc.describe()

# #imprimir todos los histogramas juntos cuando no es array
# df5.hist(bins=100, figsize=(30, 30))
# df5_sc.hist(bins=100, figsize=(30, 30))
```

```
# #imprimir histogramas separados automáticamente
# count = 0
# for col in df5.columns:
#     plt.figure(figsize=(10,5))
#     df5[col].plot.hist(bins = 150, color= 'blue', edgecolor = 'black')
#     dev = df5[col].std()
#     mean = df5[col].mean()
#      plt.xlabel('Desv. Standard[\u03C3] ='+'{:.3f}'.format(dev) + '          media[\u03BC]
='+'{:.3f}'.format(mean))
#     plt.ylabel('Valores')
#     plt.legend()
#     plt.axvline(df5[col].mean(), color='red', linestyle='dashed', linewidth=1)
#     plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
#                                         plt.savefig("C:/Users/migue/Documentos/Maestria/3er
Semestre/Figuras/Original/fig_"+str(count)+".png")
#     count+=1
#     plt.show()

# #imprimir histogramas separados automáticamente
# count = 0
# for col in df5_sc.columns:
#     plt.figure(figsize=(10,5))
#     df5_sc[col].plot.hist(bins = 150, color='green', edgecolor = 'black')
#     dev2 = df5_sc[col].std()
#     mean_og = df5.iloc[:, count].mean()
#     mean2 = df5_sc[col].mean()
#      plt.xlabel('Desv. Standard[\u03C3] ='+'{:.3f}'.format(dev2) + '          media[\u03BC]
='+'{:.3f}'.format(mean2) + ' = '+ {:.3f}'.format(mean_og))
#     plt.ylabel('Valores')
#     plt.legend()
```

```
#     plt.axvline(df5_sc[col].mean(), color='red', linestyle='dashed', linewidth=1)
#     plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
#                                         plt.savefig("C:/Users/migue/Documentos/Maestria/3er
Semestre/Figuras/Standarizado/fig_sc_"+str(count)+".png")
#     count+=1
#     plt.show()

#Eliminar columnas con desviación estándar mayores de 3 y menores de -3
#para quedarnos con solo el 99.7% de la información y alfa = 0.01
#-------------------------------------------------------------------------------
# df10 = df5
# df10_sc = df5_sc
# df5_info = df5.describe()

# col_borrar = 0

# for col in df10_sc.columns:
#     #Crear lista de valores que estan dentro del rango +-3 desv std
#     k = df10_sc[df10_sc.iloc[:,col_borrar]<-3].index
#     j = df10_sc[df10_sc.iloc[:,col_borrar]>3].index
#     l = j.union(k)

# #Crea nuevo dataframe escalado despues de remover filas de la lista anterior
#     df11 = df10_sc.drop(df10_sc.index[l])

# #Crea nuevo dataframe sin escalar despues de remover filas de la lista anterior
#     df10 = df10.drop(df10.index[l])

# #Estandarizar el nuevo dataframe para obtener las nuevas desv. std
#     df11_sc = scaler.fit_transform(df10)
```

```
#     df10_sc = pd.DataFrame(df11_sc)

# #Repetir proceso con siguiente columna
#     col_borrar+=1

# #imprimir todos los histogramas juntos cuando no es array
# df11.hist(bins=100, figsize=(30, 30))
# df10.hist(bins=100, figsize=(30, 30))

# #imprimir histogramas separados automáticamente
# count = 0
# for col in df11.columns:
#     plt.figure(figsize=(10,5))
#     df11[col].plot.hist(bins = 150, color='purple', edgecolor = 'black')
#     dev3 = df11[col].std()
#     mean_og = df5.iloc[:, count].mean()
#     mean3 = df11[col].mean()
#      plt.xlabel('Desv. Standard[\u03C3] ='+'{:.3f}'.format(dev3) + '         media[\u03BC]
='+'{:.3f}'.format(mean3) + ' = '+' {:.3f}'.format(mean_og))
#     plt.ylabel('Frecuencia')
#     plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
#     plt.legend()
#     plt.axvline(df11[col].mean(), color='red', linestyle='dashed', linewidth=1)
#                                         plt.savefig("C:/Users/migue/Documentos/Maestria/3er
Semestre/Figuras/Standarizado/fig_final_"+str(count)+".png")
#     count+=1
#     plt.show()

# df11.to_excel('df11_1seg.xlsx')
# df10.to_excel('df10_2seg.xlsx')
```

```
# df10_info = df10.describe()

#Eliminar columnas con desviación estándar igual a 0
df12 = df5.drop(df5.std()[(df5.std()== 0)].index, axis = 1)
df12_info = df12.describe()
# df12.to_excel('preprocesado.xlsx')

#cambiar nombre de variable
datos_iniciales=df12
# datos_iniciales.to_excel('Datos_limpios.xlsx')

#eliminar columna date
datos_iniciales = datos_iniciales.reset_index()
datos_iniciales = datos_iniciales.drop('Date', axis =1 )

datos_iniciales2 = datos_iniciales[datos_iniciales['Yield from flow'] !=0]
datos_iniciales2_info = datos_iniciales2.describe()



y = datos_iniciales2['Yield from flow']
x = datos_iniciales2.drop('Yield from flow', axis = 1)

x = np.asarray(x)
y = np.asarray(y)

# x = x.reshape (-1,136)
y = y.reshape (-1,1)

neuronas = len(datos_iniciales2.axes[1])-1
```

```
#-------------------------------------------------------------------------
# RED NEURONAL 1

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import InputLayer, Dense
from sklearn.model_selection import train_test_split

#se leen los valores de x, y del disco para separar en 90%-10%
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.1)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()



X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

arauco = Sequential()
arauco.add(InputLayer((neuronas)))
arauco.add(Dense(30, activation = 'relu'))
arauco.add(Dense(40, activation = 'relu'))
arauco.add(Dense(50, activation = 'relu'))
arauco.add(Dense(60, activation = 'relu'))
# arauco.add(Dense(90, activation = 'relu'))
arauco.add(Dense(1, activation = 'relu'))

#Se usa mean squared error para calcular el error vs el dato real
arauco.compile(optimizer = 'adam',
               loss = 'mse',
```

```
            metrics = ['mae'])

#Crear un callback que guarde el mejor modelo
#Se indica la ruta en caso de que se quiera indicar en la siguiente línea de código
#model_path = os.path.join(SUBDIR, 'tmp/checkpoint')
#Se guarda el modelo con el mejor valor monitoreando la precisión de la accuracy
save_model_callback =
tf.keras.callbacks.ModelCheckpoint(filepath=r'\Users\migue\Documentos\Maestria\4o
Semestre\4 Horas',
                                   monitor='val_accuracy',
                                   mode='auto',
                                   save_best_only=True)

#Se guarda la instrucción para evitar el sobre-entrenamiento
avoid_overtraining = tf.keras.callbacks.EarlyStopping(patience=3, monitor='val_loss')
my_callbacks = [save_model_callback, avoid_overtraining]

#Se pueden generar alertas relacionadas al guardar el estado del optimizador
#Estas advertencias están para desalentar el uso caducado, y pueden ser ignoradas


history = arauco.fit(X_train, y_train, batch_size = 150, epochs =20, validation_split =
0.2, callbacks = my_callbacks)


prediccion_red1 = arauco.predict(X_test)
error_red1 = y_test - prediccion_red1


plt.figure(figsize=(10,5))
```

```
plt.hist(error_red1,bins = 150, color='red', edgecolor = 'black')
plt.xlabel('Diferencia entre prediccion vs validación', loc= 'center')
plt.ylabel('Valores')
plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
plt.show()


porcentaje_error_red1 = (error_red1 / y_test)*100
eficiencia_red1 = 100-abs(porcentaje_error_red1)

error_red1_1df = pd.DataFrame(error_red1)
eficiencia_red1_df = pd.DataFrame(eficiencia_red1)
prediccion_red1_df = pd.DataFrame(prediccion_red1)
val_real_red1_df = pd.DataFrame(y_test)

error_red1_1df.to_csv('error.csv')
eficiencia_red1_df.to_csv('eficiencia.csv')

eficiencia_red1_df = eficiencia_red1_df.drop(eficiencia_red1_df.index[287])

# eficiencia_red1_df.drop(eficiencia_red1_df.index[163], inplace=True)

prediccion_red1_df.to_csv('prediccion.csv')
val_real_red1_df.to_csv('valor_real.csv')


plt.figure(figsize=(10,5))
plt.hist(eficiencia_red1_df, bins = 100, color='blue', edgecolor = 'black', label =
'Histograma')
plt.xlabel('Precision de la predicción', loc= 'center')
```

```
plt.ylabel('Valores')
plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
# plt.xlim((0,110))
plt.show()

#Verificar si hay overfitting
plt.figure(figsize=(15,5))
plt.plot(history.history['loss'], label = '(MSE) Entrenamiento 1')
plt.plot(history.history['val_loss'], label = '(MSE) Validación 1')
plt.xlabel('Epocas de entrenamiento')
plt.ylabel('Valores de Error')
plt.legend()
plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
# plt.ylim((0,1500.0))
#
plt.xticks([0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50]
)

val_loss1 = history.history['val_loss']
loss1= history.history['loss']

val_loss_pd = pd.DataFrame(val_loss1)
val_loss_pd.to_csv('val_loss.csv')
loss_pd = pd.DataFrame(loss1)
loss_pd.to_csv('loss.csv')

arauco.save('arauco.h5')
arauco.summary()
arauco.get_weights()
arauco.optimizer
```

```
# Grabar a disco el motor de escalamiento
import joblib
scaler_filename = "scaler_4hrs.save"
joblib.dump(scaler, scaler_filename)

# # Grabar a disco el motor de labels
# import joblib
# label_filename = "label_4hrs.save"
# joblib.dump(le, label_filename)

col_predic = pd.DataFrame(datos_iniciales2.columns.tolist())

col_predic.to_csv('Variables para prediccion_4hrs.csv')

#-----------------------------------------------------------------------------------------------------

# # Leer tambien el motor de escalamiento
# import joblib
# scaler2 = joblib.load('scaler_4hrs.save')


from sklearn.utils.validation import check_is_fitted
check_is_fitted(scaler)

import tensorflow as tf

#Cargar el modelo guardado
model = tf.keras.models.load_model('arauco.h5')
```

```
model.summary()
model.get_weights()
model.optimizer


import pandas as pd

# Leer la base de datos
variables = pd.read_csv('Variables para prediccion_4hrs.csv', sep = ',', index_col=0)
# Leer la base de datos
variables = col_predic
#Indicamos la carpeta donde están los datos que queremos predecir

#importamos las librerias
import tkinter as tk
import os
from tkinter import filedialog
import numpy as np

#Definimos la función
def select_folder():
    global data_file_folder
    data_file_folder = filedialog.askdirectory()
    root.destroy()

# Crear ventana
root = tk.Tk()
root.title("Seleccionar Carpeta")
root.geometry("400x200")
```

```
root.configure(bg="gray")

# Etiqueta
label = tk.Label(root, text="Haz clic para seleccionar una carpeta con archivo de datos", bg="gray")
label.pack(pady=20)

# Botón
button = tk.Button(root, text="Seleccionar Carpeta", command=select_folder)
button.pack(pady=10)

# Variable para almacenar la carpeta seleccionada
data_file_folder = None

# Loop de la aplicación
root.mainloop()

# Una vez que la ventana se cierra, el programa continúa aquí
print("Carpeta seleccionada:", data_file_folder)

#Generamos una lista que contenga todos los Dframes de los archivos dentro de la carpeta
#y se les da formato segun lo requerido
lista = []
for file in os.listdir(data_file_folder):
    if file.endswith('.xlsx'):
        print('Cargando archivos {0}...'.format(file))
        lista.append(pd.read_excel(os.path.join(data_file_folder, file), sheet_name=1, header=1))
```

```
for x in lista:
    # Set the first column as the index
    x.set_index('Date', inplace=True)

#Unimos los archivos ya formateados
df_inicial = pd.concat(lista, axis=0)

#Guardamos los datos en un archivo nuevo
#Guardar como excel
# df_inicial.to_excel('Datos_completos.xlsx')

#Guardar como CSV
df_inicial.to_csv('Datos_formateados_4hrs.csv')

#cargar y dar formato al archivo csv
df_inicial = pd.read_csv('Datos_formateados_4hrs.csv', sep = ',', na_values= ['???'],
index_col=0)
df_inicial =  df_inicial.fillna(0)
df_inicial = df_inicial.loc[(df_inicial!=0).any(axis=1)]

#convertir columnas de texto en numeros

# from sklearn import preprocessing
# le = preprocessing.LabelEncoder()

# # Leer también el motor de escalamiento
# import joblib
# le = joblib.load('label_1seg.save')

# le.fit(df_inicial['PROD_MODE/PROD_MODE.CV'])
```

```
#
df_inicial['PROD_MODE/PROD_MODE.CV']=le.transform(df_inicial['PROD_MODE/PROD_MODE.CV'])

# le.fit(df_inicial['PROD_MODE/UFC_MODE.CV'])
#
df_inicial['PROD_MODE/UFC_MODE.CV']=le.transform(df_inicial['PROD_MODE/UFC_MODE.CV'])

#obtener tipo de datos dentro de cada columna para poder eliminar las no numericas
r = df_inicial.dtypes
df_inicial = df_inicial.select_dtypes(exclude=['object'])


df_inicial.drop(df_inicial[df_inicial['Yield from flow'] >= 1000].index, inplace = True)

df_inicial.drop(df_inicial[df_inicial['Yield from flow'] <= 0].index, inplace = True)

variables = col_predic[0].tolist()


df_ajustado = df_inicial[df_inicial.columns.intersection(variables)]

#eliminar columna date
df_ajustado = df_ajustado.reset_index()
df_ajustado = df_ajustado.drop('Date', axis =1 )

df_ajustado = df_ajustado[df_ajustado['Yield from flow'] !=0]
df_ajustado_info = df_ajustado.describe()
```

```
#definir x, y
y2 = df_ajustado['Yield from flow']
x2 = df_ajustado.drop('Yield from flow', axis = 1)

#convertir en array
x2 = np.asarray(x2)
y2 = np.asarray(y2)

#dar formato al array
y2 = y2.reshape (-1,1)

#Predicción
x_pred = scaler.transform(x2)

#realizar la predicción
predicciones = arauco.predict(x_pred)

y3 = pd.DataFrame(y2)
x3 = pd.DataFrame(predicciones)

#imprimir comparativa entre predicciones y registros reales
plt.figure(figsize=(10,5))
plt.legend()
plt.scatter(x3.index, x3, color='blue', edgecolor = 'blue', label = 'Predicciones', s= 2)
plt.scatter(y3.index, y3, color='red', edgecolor = 'red', label = 'Registros Proceso 2019',s= 2)
plt.xlabel('No. de Registro', loc= 'center')
plt.ylabel('Valor del Sensor "Yield from flow"')
plt.legend()
plt.grid(color='black', linestyle = 'dotted', linewidth=0.5)
```

```
plt.savefig(r'\Users\migue\Documentos\Maestria\4o Semestre\4 Horas\Figuras\Comparativa.png')
plt.show()
```