Análisis del desempeño de las redes bayesianas en el diagnóstico de fallos del proceso Tennessee-Eastman

Paula Daniela Jerez Castillo y Christian Stewart Virviescas Sánchez

Trabajo de Grado para Optar al Título de Ingeniero Químico

Director

Giovanni Morales Medina

Dr. en Ingeniería Química

Universidad Industrial de Santander

Facultad de Ingenierías Fisicoquímicas

Escuela de Ingeniería Química

Programa Académico

Bucaramanga

2024

Dedicatoria

A mis padres Olga Castillo y Ramiro Jerez, por su esfuerzo, su incondicional apoyo y compañía durante todo este camino, a mis hermanas Esperanza, Sandra y Luz, por ser mis guías y ejemplo a seguir, a mis sobrinos, cuyas sonrisas me llenaron de alegría y a todas las personas que hicieron parte de esta etapa, contribuyendo con su presencia y apoyo a la culminación de este logro.

Paula Jerez

A mis padres Luz Dary Sánchez y Mauricio Virviescas, por su paciencia, esfuerzo y cariño que me brindaron durante este proceso, a el profesor Giovanni por su constante guía y apoyo en la realización de este trabajo, a mi hermano Gerson y Diego por ser grandes compañeros, y a todas las personas que, con su presencia, contribuyeron a la realización de este logro. A cada uno de ustedes, gracias de corazón.

Christian Virviescas

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

pág.

Resumen

Título: Análisis Del Desempeño De Las Redes Bayesianas En El Diagnóstico De Fallos Del Proceso Tennessee-Eastman.[*]

Autor: Paula Daniela Jerez Castillo, Christian Stewart Virviescas Sánchez.[**]

Palabras Clave: Procesos industriales, precisión, MATLAB.

Descripción: Este proyecto tuvo como finalidad realizar un análisis del desempeño de las redes neuronales tipo bayesianas para el diagnóstico de fallos en el proceso Tennessee-Eastman (TEP), esta red es un modelo matemático ampliamente utilizado en el aprendizaje causal. En primer lugar, se hizo una revisión bibliográfica de los indicadores operativos y de seguridad para identificar las dos fallas más críticas de la planta, seleccionando la falla 2 (composición de B con proporción de A/C constante) y la falla 12 (temperatura del agua de enfriamiento a la entrada del condensador). El pretratamiento de los datos se hizo mediante z-score para normalizarlos. Se probaron diferentes estructuras de la red variando parámetros como la cantidad de *epoch*, el número de neuronas, la función de activación y *drouput*, además de utilizar un modelo de optimización bayesiana para ajustar estos parámetros, esto con el fin de encontrar la mejor estructura de red bayesiana en el programa de MATLAB, se alcanzó una precisión de 87.707% usando 15, 13 y 11 neuronas para la capa 1, 2 y 3 respectivamente, y combinando las funciones de activación ReLU y leaky ReLU. Además, se evidencio que a partir de la neurona 10 usando una estructura sencilla la precisión se mantenía constante.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Fisicoquímicas. Escuela de Ingeniería Química. Programa académico. Director: Giovanni Morales Medina. Dr. en Ingeniería Química.

Abstract

Title: Analysis of the performance of Bayesian networks in the diagnosis of faults in the Tennessee-Eastman process.[*]

Author(s): Paula Daniela Jerez Castillo, Christian Stewart Virviescas Sánchez.[**]

Key Words: Industrial processes, accuracy, MATLAB.

Description: The purpose of this project was to perform a performance analysis of Bayesian neural networks for fault diagnosis in the Tennessee-Eastman process (TEP). This network is a mathematical model widely used in causal learning. First, a bibliographic review of the operational and safety indicators was carried out to identify the two most critical faults in the plant, selecting fault 2 (B composition with constant A/C ratio) and fault 12 (cooling water temperature at the condenser inlet). Data pretreatment was done using z-score to normalize them. Different network structures were tested by varying parameters such as the number of *epochs*, the number of neurons, the activation function and *drouput*, in addition to using a Bayesian optimization model to adjust these parameters, in order to find the best Bayesian network structure in the MATLAB program, an accuracy of 87.707% was achieved using 15, 13 and 11 neurons for layers 1, 2 and 3 respectively, and combining the RELU and leaky relu activation functions. In addition, it was shown that from neuron 10 using a simple structure the accuracy remained constant.

---

[*] Degree Work

** Facultad de Ingenierías Fisicoquímicas. Escuela de Ingeniería Química. Programa académico. Director: Giovanni Morales Medina. Dr. en Ingeniería Química.

Introducción

Hoy en día la búsqueda de un producto que presente buena calidad es un factor imprescindible en el entorno competitivo y está relacionado con las especificaciones para la obtención de este (Cabezón Gutiérrez, 2014). Por ello, las industrias están obligadas a integrar nuevos sistemas para un control en la fabricación y así evitar fallos, minimizándolos durante el proceso que conlleve a cambios en la calidad del producto deseado (Mejía-Neira et al., 2019).

Las fallas industriales pueden conducir a consecuencias como daños de activos fijos, pérdidas económicas debido a las paradas imprevistas en la planta y fatalidades, por ende, la detección de estas han sido objetos de investigación y desarrollo durante las últimas décadas (Abid A et al., 2017). Un proceso industrial posee gran cantidad de variables controladas y manipuladas, por ello, es necesario monitorear estas variables con el objetivo de adquirir información sobre el comportamiento dinámico que tiene el proceso (Mejía-Neira et al., 2019) para evitar incidentes devastadores. Por ejemplo, la catástrofe de la plataforma *Deepwater Horizon*, en 2010, vertió 5 millones de barriles de crudo en el Golfo de México provocado por una explosión que pudo haber sido evitada (Garcia, 2012), (Mesquida et al., 2010), (Araque et al., 2018), (Chiquito et al., 2016).

En Colombia, la Ley 1523 establece que toda persona natural o jurídica debe adoptar medidas necesarias para administrar el riesgo, actuar con precaución y proporcionar un plan de contingencia si se presenta alguna situación de emergencia (Congreso de Colombia, 2012).

La aplicación de algoritmos de detección y pronóstico de fallas, en conjunto con un sistema de integridad operativa, pueden evitar accidentes con consecuencias de pérdidas de activos, daños ambientales y fatalidades.

Los desempeños de los algoritmos de detección, como los basados en redes neuronales artificiales (RNA) pueden ser aplicados para una detección y un diagnóstico temprano de la aparición de alguna falla en los procesos industriales. Para evaluar el desempeño de las RNA se suelen utilizar casos estándar como la simulación denominada proceso Tennessee-Eastman (TEP).

Las RNA permiten abordar problemas complejos de manera eficiente, ya que sus modelos son dirigidos a partir de datos lo que significa que son capaces de identificar relaciones mediante algoritmos de aprendizaje basados en datos existentes, (Salas, 2004). Estas redes pueden entrenarse con un enfoque basado en probabilidad, denominado entrenamiento bayesiano (Lopez y Caicedo Bravo, 2005). Aunque la predicción de las fallas se puede realizar a través de datos de procesos, estos son difíciles de conseguir debido a secrecía industrial. Una alternativa corresponde al uso de simulaciones para probar algoritmos de predicción de fallas; una de estas simulaciones corresponde al proceso Tennessee-Eastman (TEP) (Márquez et al., 2021).

Por lo anterior, este documento expone los principales resultados de una investigación enfocada a la evaluación del desempeño de RNA Bayesianas, en la detección y el diagnóstico de fallos del proceso Tennessee Eastman. La pregunta de investigación correspondió a, ¿cuáles parámetros de las RNA Bayesianas conducen al mejor desempeño de predicción de dos fallas críticas del TEP?

1. Objetivos

1.1 Objetivo General

Analizar los desempeños de las redes neuronales artificiales tipo Bayesianas en la predicción de dos fallas características del proceso Tennessee-Eastman (TEP).

1.2 Objetivos Específicos

Valorar las fallas en el proceso Tennessee-Eastman (TEP), por medio de indicadores operativos y de seguridad, definiendo las dos fallas de mayor impacto en el proceso.

Desarrollar un código de aplicación de entrenamiento y validación de las RNA Bayesianas, por medio de las funciones y aplicaciones disponibles en MATLAB, definiendo las entradas y los parámetros de variación en la ejecución.

Seleccionar la mejor arquitectura de RNA Bayesiana de predicción de las dos fallas definidas, con base en los desempeños obtenidos con el código desarrollado.

2. Marco Conceptual

***Proceso Tennessee-Eastman*:** Simulación desarrollada por la empresa Eastman Chemical, la cual presenta 21 tipos de fallas que pueden ocurrir en una planta industrial.

Red neuronal artificial: Intenta imitar la forma en la que las neuronas biológicas se transmiten entre sí, combinando informática y estadística.

Red neuronal bayesiana*:* Modelo que permite estimar la probabilidad posterior de las variables no conocidas a partir de las variables que ya se conocen.

Diagnóstico de fallas: Identifica las fallas que afectan el proceso mediante un análisis de señales que ofrecen los sensores del proceso.

3. Marco Teórico

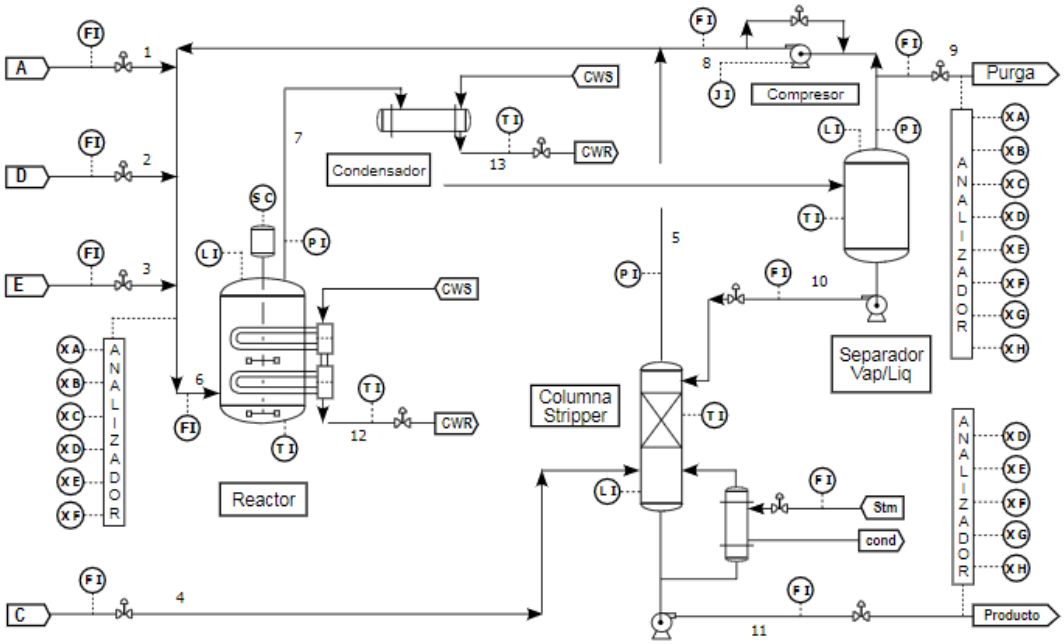
3.1 Proceso Tennessee-Eastman

El proceso Tennessee-Eastman (TEP) es un modelo de simulación de una planta no instalada, desarrollada por la empresa Eastman Chemical Company y publicada por Downs y Vogel (1993) (Iturbe et al., 2016). Este modelo de simulación es utilizado como *benchmarking* en el diagnóstico de fallas, ya que permite el acceso a una gran variedad de datos de sensores y actuadores, así como el análisis de la respuesta del proceso ante la definición de perturbaciones o fallas. La simulación de TEP se compone de cinco unidades de operación (reactor CSTR, condensador, separador liquido-vapor, compresor de reciclo y columna *stripper*). La generación de datos del proceso TEP puede también ser utilizada en el análisis del desempeño de algoritmos de detección y diagnóstico de fallas (Toro y Sotomayor, 2008). La **Figura 1** presenta el diagrama

de flujo del proceso (PFD) de la simulación del TEP. El reactor es un tanque agitado CSTR, en el cual se generan los productos G y H y un subproducto F a partir de los reactivos A, D, E y C. Estos reactivos son acompañados por un inerte B, en su ingreso al proceso. La corriente de salida del reactor pasa a un condensador y a un separador líquido-vapor. Los componentes que aún se encuentran en estado gaseoso son recirculados a través de un compresor hacia la alimentación del reactor, mientras que los productos líquidos pasan al stripper para terminar de separar algunas trazas de reactivos aún presentes. El inerte y el subproducto son purgados del proceso en forma de vapor en el separador líquido-vapor (Downs y Vogel, 1993).

Figura 1.

Diagrama del proceso de la planta TEP



Nota. Adaptado de (Downs y Vogel, 1993b)

En el proceso se consideran 4 reacciones exotérmicas, irreversibles y de primer orden respecto a la concentración de los reactivos, según las siguientes:

$$\boldsymbol{A(g) + C(g) + D(g) \rightarrow G(liq)} \quad \textit{Producto 1} \quad \textit{Ec. 1}$$

$$\boldsymbol{A(g) + C(g) + E(g) \rightarrow H(liq)} \quad \textit{Producto 2} \quad \textit{Ec. 2}$$

$$\boldsymbol{A(g) + E(g) \rightarrow F(liq)} \quad \textit{Subproducto} \quad \textit{Ec. 3}$$

$$\boldsymbol{3D(g) \rightarrow 2F(liq)} \quad \textit{Subproducto} \quad \textit{Ec. 4}$$

El proceso comprende 12 variables manipuladas y 41 variables medidas, como se observa en la **Tabla 1** y en el Apéndice A. Según la información proporcionada en el Apéndice A, las variables de XMEAS (1) a XMEAS (6) y XMEAS (10) son de flujo, XMEAS (26) a XMEAS (41) son de concentración, XMEAS (9,11,18,21,22) son de temperatura y XMEAS (7,13,16) son de presión. Asimismo, XMV son variables manipuladas.

Tabla 1.

Variables manipuladas del TEP

Variable	Nombre de variable	Unidades
XMV (1)	Flujo de alimentación D	kgh^{-1}
XMV (2)	Flujo de alimentación E	kgh^{-1}
XMV (3)	Flujo de alimentación A	kscmh
XMV (4)	Flujo de alimentación A y C	kscmh
XMV (5)	Válvula de recirculación del compresor	%
XMV (6)	Válvula de carga	%
XMV (7)	Flujo de líquido del separador LV	m^3h^{-1}
XMV (8)	Flujo de líquido de la columna de stripper	m^3h^{-1}
XMV (9)	Válvula de vapor de la columna de stripper	%
XMV (10)	Flujo de agua de refrigeración del reactor	m^3h^{-1}
XMV (11)	Flujo de agua en el condensador	m^3h^{-1}
XMV (12)	Velocidad del agitador del reactor	rpm

Nota. Adaptado de (Downs y Vogel., 1993b)

La simulación incluye la codificación de diferentes tipos de fallas, que están distribuidas en todo el proceso, como se presenta en la **Tabla 2**. Según esta tabla, el TEP contiene 21 fallas del proceso, de las cuales 15 son conocidas y 6 desconocidas, clasificadas según el tipo de perturbación. Estas fallas se pueden agrupar en 3 bloques diferentes según (Llanes et al., 2017),

las asociadas al condensador en el cual se encuentran las fallas 5, 12 y 15; las relacionadas al reactor, que son las fallas 4, 11, 13 y 14 y las de alimentación que abarcan las fallas 1, 2, 3, 6, 7, 8, 9 y 10. Estas categorías hacen referencia a la parte principal del proceso donde se ven reflejadas y, por ende, están relacionadas a su vez con variables tales como la temperatura, la presión y el flujo.

Tabla 2.

Fallos del TEP

Fallo	Variables de fallo	Tipo de fallo
IDV (1)	Relación de flujo de alimentaciones A/C, composición de B constante	Escalón
IDV (2)	Composición de B con relación A/C constante	Escalón
IDV (3)	Temperatura de alimentación D	Escalón
IDV (4)	Temperatura de entrada del agua del refrigerante del reactor	Escalón
IDV (5)	Temperatura de entrada del agua refrigerante del condensador	Escalón
IDV (6)	Perdida de alimentación de A	Escalón
IDV (7)	Perdida de presión en la corriente C	Escalón
IDV (8)	Composición de las alimentaciones A, B y C	Variación aleatoria
IDV (9)	Temperatura de alimentación D	Variación aleatoria
IDV (10)	Temperatura de alimentación C	Variación aleatoria
IDV (11)	Temperatura de entrada del agua refrigerante al reactor	Variación aleatoria
IDV (12)	Temperatura de entrada del agua de refrigerante del condensador	Variación aleatoria
IDV (13)	Cinética de las reacciones	Variación lenta
IDV (14)	Válvula del agua de refrigerante del reactor	Bloqueo
IDV (15)	Válvula del agua de refrigerante del condensador	Bloqueo
IDV (16)	Desconocido	No especificado
IDV (17)	Desconocido	No especificado
IDV (18)	Desconocido	No especificado
IDV (19)	Desconocido	No especificado
IDV (20)	Desconocido	No especificado
IDV (21)	Desconocido	Constante

Nota. Adaptado de (Downs y Vogel., 1993b)

Por otra parte, la **Tabla 3** establece los límites del TEP en dos escenarios: operación normal y durante una parada de planta. Estos límites se han definido en cinco variables del proceso, como se indica en esta tabla.

Tabla 3.

Límites de funcionamiento del proceso

	Límites de operación normal		Límites que generan parada de planta	
Variable del proceso	Límite inferior	Límite superior	Límite inferior	Límite superior
Presión del reactor	Ninguno	2895 kpa	Ninguno	3000 kpa
Nivel del reactor	50% (11.8 m^3)	100% (21.3m^3)	2 m^3	24 m^3
Temperatura del reactor	Ninguna	150 °C	Ninguna	175 °C
Nivel separador del producto	30% (3.3 m^3)	100% (9 m^3)	1 m^3	12 m^3
Nivel base de stripper	30% (3.5 m^3)	100% (6.6 m^3)	1 m^3	8 m^3

Nota. Adaptado de (Downs y Vogel., 1993b)

3.2 Redes Neuronales Artificiales (RNA)

En 1943 el neurofisiólogo Warren McCulloch y el matemático Walter Pitts, publicaron el artículo "*A Logical Calculus of Ideas Immanent in Nervous Activity"* que describía cómo las neuronas pueden funcionar y para eso modelaron una red neuronal simple usando circuitos eléctricos, de esta manera introdujeron las bases para el desarrollo de áreas como la inteligencia artificial (McCulloch et al., 2019).

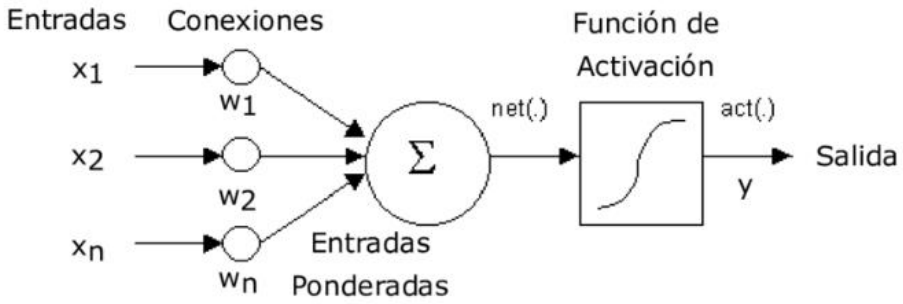
Una RNA es un modelo informático con características particulares, como la capacidad de ajustarse o aprender, de generalizar y organizar la información, gracias a un procesamiento paralelo (Haykin, 1998), es un esquema de computación distribuido, inspirado en la estructura del

sistema nervioso de los humanos, ya que el cerebro humano corresponde a un sistema complejo, no lineal y paralelo (Izaurieta y Saavedra, 2000). Estas se usan como método de resolver problemas de manera individual o combinada con otros métodos ya que permite modelar de manera efectiva problemas grandes y complejos (Salas, 2004).

En la siguiente figura se presenta la estructura básica de una RNA, la cual refleja las interconexiones de la neurona. Los componentes principales de una neurona incluyen: las entradas, que corresponden a la información que reciben del entorno; los pesos (w), que determinan la importancia que tiene cada entrada que llega a la neurona; la función de activación, que transforma la información que llega a cada neurona; y la salida (y), que es la respuesta generada por la RNA.

Figura 2.

Interconexiones en una RNA



Nota. Adaptado de (Medrano Sanz., et al.2019)

En las RNA, las neuronas actúan como unidades básicas de procesamiento. Cada neurona recibe entradas (input), las procesa y genera una salida (output), que se convierte en la entrada de otras neuronas a través de conexiones ponderadas y así sucesivamente. Estas redes suelen constar de múltiples capas de neuronas organizadas secuencialmente (González, M., 2020).

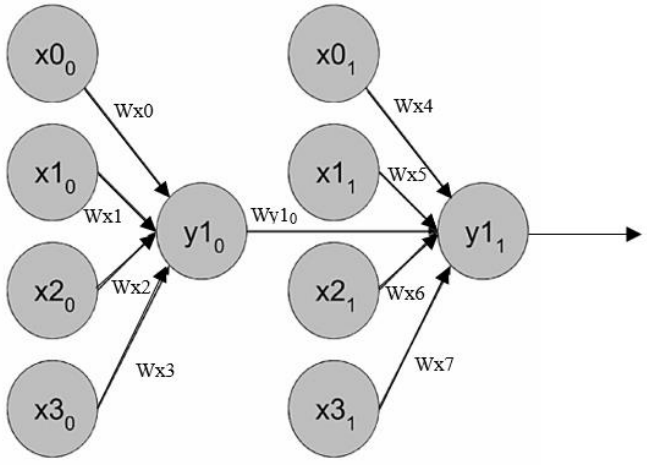
3.3 Redes Bayesianas

Las redes bayesianas describen un fenómeno mediante variables y relaciones de dependencia entre ellas. Se puede estimar la probabilidad de las variables desconocidas a partir de las conocidas (Sucar, 2006). Al entrenar redes neuronales usando un enfoque bayesiano, se adopta una perspectiva probabilística donde los pesos de la red se consideran como distribuciones de probabilidad en lugar de valores fijos (López y Caicedo, 2007). Constituyen uno de los modelos matemáticos que más se emplean en la explicación del aprendizaje causal. Una de sus características principales es que evalúan, de alguna forma, la probabilidad de todas las posibilidades de los sucesos según Rodríguez Mañanes J. (2009).

La **Figura 3** representa la estructura de una red bayesiana, los nodos representan variables aleatorias y las flechas relaciones de dependencia.

Figura 3.

Estructura de una red bayesiana



Nota. Adaptado de (Calvo Valverde., et al. 2019)

Cada nodo en la red tiene una tabla de distribución condicional, la cual muestra las probabilidades de sus posibles valores, según los valores de los nodos que le preceden. En esta

red, la información fluye de manera que cada nodo de salida se convierte en el padre del siguiente, influyendo en su valor (Calvo et al., 2019).

La forma de calcular la probabilidad posterior es usando la regla de Bayes, la cual se ha escrito para el caso de una red neuronal (López y Caicedo Bravo, 2007).

$$p(A|S,H) = \frac{p(S|A,H)p(A|H)}{p(S|H)} \quad \text{(Ec. 5)}$$

Donde:

A= Vector de pesos de la red neuronal.

S= Conjunto de datos utilizados para el entrenamiento.

H= Representa el modelo neuronal que se está entrenando.

$p(A|S,H)$= Densidad de la probabilidad posterior de los pesos.

$p(A|H)$= Densidad de la probabilidad a priori (representa el conocimiento que se tiene de los pesos antes de obtener datos).

$p(S|A,H)$= Densidad de probabilidad condicional de que los datos ocurran dados los pesos.

$p(S|H)$= Factor de normalización que garantiza que la probabilidad total sea 1.

Para asegurar un buen desempeño de las redes, es importante definir la configuración adecuada de los hiperparámetros. Estos hiperparámetros son variables que se ajustan cuando se ejecuta el entrenamiento, lo cual conduce a un desempeño adecuado en la predicción de las salidas de la red; estos hiperparámetros están relacionados con la eficacia del modelo (ssalgadodev, 2024a). Los hiperparámetros más comúnmente utilizados son:

- *Epochs*: Una *epoch* establece el número de iteraciones que va a realizar a través de todo el conjunto de datos de entrenamiento, en donde una época comprende uno o más lotes (Ramos, 2019).

- *Batch size*: El *Batch size* hace referencia al tamaño de lote de datos de entrada y salida en cada iteración de entrenamiento (Salgadodev, 2024b), es un subconjunto del conjunto de entrenamiento que se usa para evaluar el gradiente de la función de pérdida, permitiendo actualizar los pesos.
- *Tasa de aprendizaje inicial*: Permite controlar el tamaño de los pasos que el optimizador realiza para ajustar los pesos del modelo en cada iteración, por ejemplo, si la tasa de aprendizaje es baja, el entrenamiento puede tardar más tiempo (Liu & Nocedal, 1989).
- *Verbose:* Este es un indicador que muestra información sobre el progreso del entrenamiento en tiempo real en la ventana de comandos, permitiendo detectar posibles problemas (Liu & Nocedal, 1989).
- *Sigma1 y sigma 2:* Son parámetros de las probabilidades previas de la distribución de pesos (Blundell et al., 2015).

3.4 Normalización z-score

Es un método de normalización que se basa en la media y en la desviación estándar de los datos (Henderi et al., 2021), la fórmula que se utilizó es la siguiente:

$$X_{new} = \frac{X-\mu}{\sigma} \qquad \text{(Ec. 6)}$$

Donde:

$X_{new} =$ El nuevo valor de los resultados normalizados.

$X =$ Valor antiguo.

$\mu =$ Media poblacional.

$\sigma =$ Valor de desviación estándar.

4. Estado de Arte

(Sun et al., 2020) presentan un innovador método para la detección e identificación de fallas probabilísticas, aplicando un enfoque de aprendizaje profundo con redes neuronales recurrentes bayesianas y abandonos variacionales al proceso Tennessee Eastman. Este enfoque permite no solo la detección simultánea de fallas en procesos químicos, sino también la identificación directa de estas mismas. Al incorporar estimaciones de incertidumbre continuas, el modelo genera intervalos de confianza adaptativos que reflejan la dinámica del sistema en función de la información actual y pasada. La arquitectura de las redes neuronales recurrentes bayesianas que emplearon fue simple y fácil de entrenar, logrando un mejor rendimiento que las estructuras y los tipos de neuronas más complejos, y la distribución predictiva del modelo se aproximó a la distribución gaussiana; el modelo final comprende 80 nodos ocultos en la capa recurrente y lo entrenaron con el parámetro de regularización $\lambda = 10^{-4}$.

(Yang et al., 2022) describen un modelo de aprendizaje automático no supervisado interpretable basado en redes bayesianas como el modelo fundamental que respalda el esquema de monitoreo de procesos. El esquema de detección de fallas opera en dos niveles: en el primer nivel, se calcula y monitorea un índice global para detectar cualquier desviación de las condiciones normales de operación; en el segundo nivel, se emplean dos índices locales para examinar en detalle la estructura de la falla, una vez que ha sido detectada en el primer nivel. Los autores usaron una red bayesiana gaussiana lineal condicionada (CLGBN) que es una red de números híbridos que consta de variables discretas y continuas (donde las continuas no pueden ser las madres de las discretas). Los resultados mostraron que la tasa de detección y la tasa de aislamiento correcto de este esquema de monitoreo basado en redes bayesianas son comparables al rendimiento del

enfoque clásico de Análisis de Componentes Principales (PCA). Además, el método basado en redes bayesianas tiene la ventaja de aislar un menor número de variables durante la fase de diagnóstico.

(Gao et al., 2024) proponen la detección y diagnóstico integrado de fallas en edificios mediante modelado de datos y redes bayesianas. Los autores plantean una nueva forma sistemática de transferir información topológica del sistema de construcción y conocimiento a una red bayesiana capaz de inferir la falla raíz más probable en todo el sistema basándose en síntomas de violación del confort y datos de operaciones; además de diseñar una red bayesiana completa de diagnóstico de fallas a nivel de edificio. En su modelo, los autores utilizaron un nodo de perturbación raíz que representa la temperatura del aire exterior y nueve nodos de falla raíz que corresponden a la falla de un equipo específico o un componente del edificio. Los resultados muestran que, con este método, lograron detectar ciertos problemas, incluso en situaciones donde los datos no eran completos.

(Tarcsay et al., 2024) estudiaron la detección de fallas basada en riesgos mediante redes bayesianas aplicadas en el análisis de modos de falla y efectos. Los autores se centraron en presentar un marco híbrido de detección de fallas y técnicas de evaluación de riesgos que utiliza un número de prioridad de riesgo modificado para resaltar anormalidades de procesos críticas para la seguridad y minimizar la tasa de alarmas superfluas, proponiendo una combinación de análisis de componentes principales dinámicos para la detección de fallas y una red bayesiana basada en análisis de modo de falla y efecto. Con base a los resultados, los autores aplicaron el análisis de modos de falla y efectos (FMEA) como herramienta para la evaluación de riesgos en el reactor de deshidrogenación de LOHC, empleando 7 nodos; por otro lado, en el modelo de referencia de tres tanques, utilizan 6 nodos. Este método logró reducir eficazmente la cantidad de alarmas de proceso

al filtrar fallas de proceso no criticas para la seguridad, reduciendo de esta manera la posibilidad de inundaciones de alarmas entre [11-37] %.

(AlSaleh et al., 2024) presentan un nuevo enfoque de aprendizaje automático para la detección de ataques DDoS en la nube: CNN basada en bayesiana y mejoras en la fusión de datos. Los investigadores recurrieron a técnicas avanzadas de aprendizaje automático y aprendizaje profundo usando redes neuronales para desarrollar modelos de predicción sensibles al contexto para la detección y predicción de ataques DDoS, presentando una metodología detallada para detectar y clasificar ataques de denegación de servicio distribuido, utilizando MATLAB y Python como las herramientas de software. Según los resultados, la mejor arquitectura de red bayesiana está compuesta por 27 capas, usan la función de activación ReLU, obteniendo una precisión promedio aproximada de 99.66%.

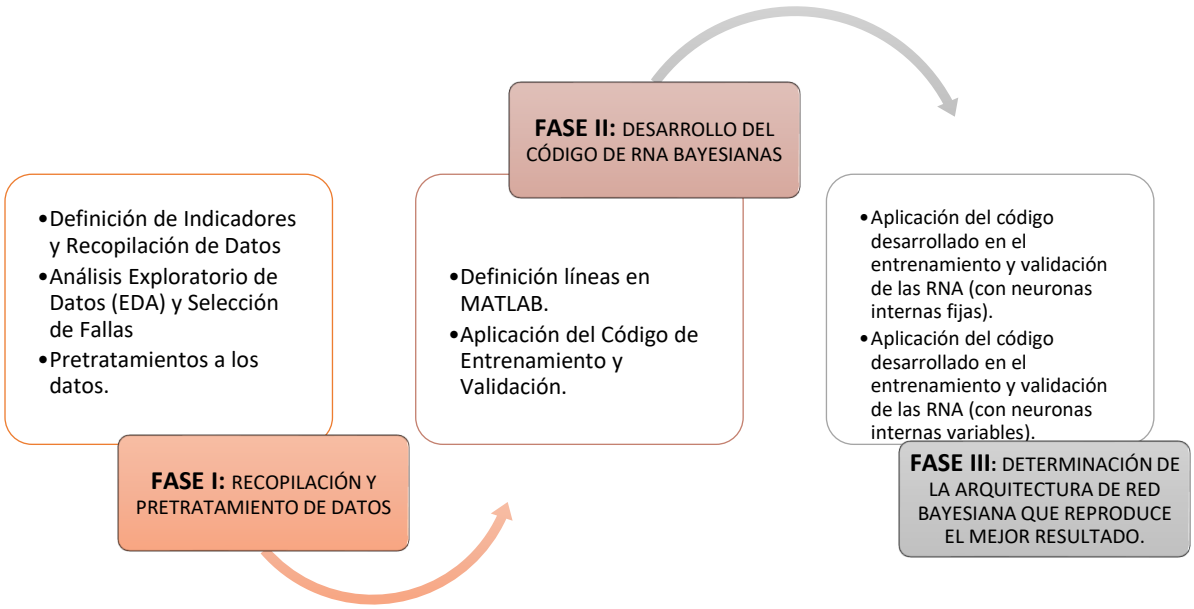
En general, teniendo en cuenta las investigaciones anteriores, se puede concluir que las redes bayesianas pueden obtener modelos con alta precisión ayudando a detectar fallos en ámbitos industriales.

5. Metodología

La metodología empleada en esta investigación está representada en la **Figura 4**. A continuación, se describen las fases en las que se estructuró, las cuales están alineadas con los objetivos específicos.

Figura 4.

Esquema de la metodología



5.1 Fase I: Análisis y Pretratamiento de Datos

5.1.1 Actividad 1: Definición de Indicadores y Recopilación de Datos

La base de datos fue obtenida de la Universidad de Harvard (Rieth et al., 2017). Se compone de 4 archivos divididos en conjuntos de prueba y entrenamiento, con 41 variables medidas y 11 variables manipuladas. Consiste en 500 simulaciones, cada una con 500 o 960 muestras, dependiendo de si es parte del conjunto de prueba o de entrenamiento, respectivamente. Cada muestra fue tomada a intervalo de 3 minutos.

Se recopiló información sobre cada falla y se decidió considerar las más críticas para el proceso teniendo en cuenta algunos indicadores claves como la producción y rendimiento, tiempo de operación, integridad de los equipos y sistema de respuesta. Estos indicadores son importantes para evaluar los niveles de calidad en la etapa de fabricación, ya que garantizan que se cumplan

las expectativas de producción. Además, un sistema de respuestas eficiente permite actuar de manera rápida cuando se presentan fallos, lo que minimiza el tiempo de inactividad y ayuda a reducir tanto las pérdidas de producción como los costos asociados.

5.1.2 Actividad 2: Análisis Exploratorio de Datos (EDA) y Selección de Falla

Para la determinación de la exploración de datos, se realizó una búsqueda mediante diferentes bases de datos como *Scielo*, *Web of Science* y *Sciencedirect* con el fin de determinar las 2 fallas más críticas del sistema teniendo en cuenta los indicadores operativos mencionados previamente. Adicionalmente, se generaron gráficos utilizando un código en Python para analizar el comportamiento de las variables del proceso con y sin fallas, a partir de los datos proporcionados.

5.1.3 Actividad 3: Pretratamientos a los Datos

Para iniciar el pretratamiento de los datos, se organizó la información para guardarla en una variable. En primer lugar, se eliminaron las tres primeras columnas para evitar problemas durante la normalización de los datos. Posteriormente, se aplicó el método de Z-score, que permite estandarizar las variables para que cada una tenga una media de 0 y una desviación estándar de 1. Este procedimiento asegura que todas las variables tengan el mismo peso, lo cual facilita los cálculos y análisis posteriores.

Además, se generaron las características y las etiquetas, creando así una matriz de respuesta donde 0 indica "sin falla", 1 corresponde a "falla 2" y 2 a "falla 12". Luego, un tamaño del 70 % de los datos fue considerado para entrenamiento, mientras que del conjunto de prueba se reservó un 15 % para validación y otro 15 % para nuevos datos de prueba, seleccionados de manera aleatoria.

5.2 FASE II: Desarrollo del Código de RNA Bayesianas

5.2.1 Actividad 1: Definición Líneas en MATLAB

Para definir las líneas en MATLAB para el desarrollo del código de RNA bayesiana primero se cargaron los datos que iban a ser usados para el entrenamiento de la red bayesiana, los cuales habían sido previamente procesados y normalizados como se menciona en el inciso anterior, seguido a esto, se define la estructura de la red neuronal, en donde se usa una capa de entrada *(featureInputLayer*) la cual toma el número de características de entrada (*numSignals*), consecuentemente, se tienen capas completamente conectadas (*bayesFullyConnectedLayer*) que aplican regularización bayesiana a los pesos (*sigma*), seguido a esto, se introducen funciones de activación no lineales (*ReLU*), que permite que la red aprenda relaciones complejas entre los datos y por último, se utiliza una capa (*softmax*) para convertir las salidas (*numClasses*) en probabilidades y una (*classification*) para hacer la respectiva clasificación.

Además, se cuenta con las opciones de entrenamiento que son parámetros del proceso de entrenamiento de la red donde se selecciona el algoritmo Stochastic Gradient Descent with Momentum (*SGDM*) como optimizador del modelo, el cual ajusta los pesos de la red con base en el descenso de gradiente, teniendo un término de *momentum* que acelera la convergencia y evita las oscilaciones. Dentro de las opciones de entrenamiento usando este optimizador se tienen parámetros del proceso de entrenamiento de la red como el número máximo de épocas (*MaxEpoch*), el tamaño de mini-lote (*MiniBatchSize*), la tasa de aprendizaje inicial (InitialLearnRate), también se especifica el conjunto de datos de validación (*ValidationData*), la frecuencia con la que se ejecutará la validación durante el entrenamiento (*ValidationFrequency*) y (*Verbose*).

Para entrenar el modelo se usa (*trainNetwork*), durante el entrenamiento el modelo ajusta los pesos de la red para minimizar la pérdida (*Loss*) y maximizar la precisión (*accuracy*) usando los datos de entrenamiento y validación. La precisión se define como el número de predicciones correctas dividido entre el número total de predicciones, logrando comparar las predicciones (*YPred*) con las etiquetas verdaderas (*YTestSub*) y contando las veces que coinciden, el valor que se obtiene se divide entre el número total de muestras para obtener una proporción. Mientras que la pérdida se define como la pérdida de entropía cruzada entre las predicciones de red y los objetivos codificados binarios, para tareas de clasificación de etiqueta única y etiqueta múltiple.

5.2.2 Actividad 2: Aplicación del Código de Entrenamiento y Validación

La aplicación de la RNA bayesiana para la detección y diagnóstico de fallos consistió en primera instancia en usar redes de una sola capa con neuronas entre 1 y 110 para poder observar el comportamiento, manteniendo constante la función de activación, el optimizador, las *epoch*, el *miniBatchSize* y la tasa de aprendizaje.

5.3 FASE III: Determinación de la Arquitectura de Red Bayesiana que Reproduce el Mejor Resultado

Para el análisis de la estructura optima se realizaron diferentes pruebas, variando las opciones de entrenamiento para definir el comportamiento de la Red bayesiana, entre estas se encuentran las anteriormente descritas, iniciando con una capa oculta.

5.3.1 Actividad 1: Aplicación del Código Desarrollado en el Entrenamiento y Validación de las RNA (con neuronas internas fijas)

En el desarrollo de esta actividad, se comenzó añadiendo una nueva capa a la red previamente creada, utilizando un parámetro fijo de 10 neuronas. Posteriormente, se realizaron

pruebas para determinar la influencia de este ajuste, evaluando su efecto según si la capa se ubicaba en la primera o en la segunda posición.

5.3.2 Actividad 2: Aplicación del Código Desarrollado en el Entrenamiento y Validación de las RNA (con neuronas internas variables)

Este proceso se abordó desde dos enfoques. Inicialmente, se añadió una tercera capa a la red y se implementaron funciones como *dropout* para evitar el sobreajuste, alternando entre diferentes funciones de activación (*ReLU* y *Leaky ReLU*). Una vez configurada esta red, se utilizó una función aleatoria para determinar el número de neuronas, subconjuntos y épocas, con el fin de analizar las diferencias.

El segundo enfoque se llevó a cabo mediante optimización bayesiana, que busca maximizar el porcentaje de precisión variando el número de neuronas, los parámetros *sigma* 1 y *sigma* 2 y la tasa de aprendizaje.

6. Resultados

6.1 Caracterización de las Fallas

En la **Tabla 4** se presentan los indicadores operativos y de seguridad que se tomaron en cuenta para la selección de fallas del TEP, como la producción y rendimiento, el tiempo de operación, la integridad de los equipos y el sistema de respuesta. La falla 2, que afecta el sistema de alimentación (Tabla 2), es crítica porque, si no se controla a tiempo, puede causar parada de planta por seguridad. De igual manera, la falla 12, que afecta el condensador, produce un apagado completo de la planta (*shut down*) (Lyman & Georgakis, 1995).

Como se observa en esta tabla, diversos indicadores operativos y de seguridad fueron evaluados, el primero, producción y rendimiento, que es la capacidad del proceso para generar productos de manera eficiente, y las variaciones en la composición de los reactivos (falla 2) o en las condiciones de refrigeración (falla 12) pueden afectar negativamente; por otro lado, en el tiempo de operación, se aprecia que las dos fallas tienen un impacto importante ya que este indicador mide la continuidad del proceso; otro aspecto clave es la integridad de los equipos, el cual se ve afectado por la falla 12, ya que las variaciones en esta falla pueden generar un desgaste acelerado y finalmente se tiene el sistema de respuesta que se aplica para la falla 12, porque los cambios en la temperatura pueden generar la activación de alarmas y la capacidad para responder eficazmente es clave para prevenir daños mayores.

Tabla 4.

Indicadores operativos y de seguridad

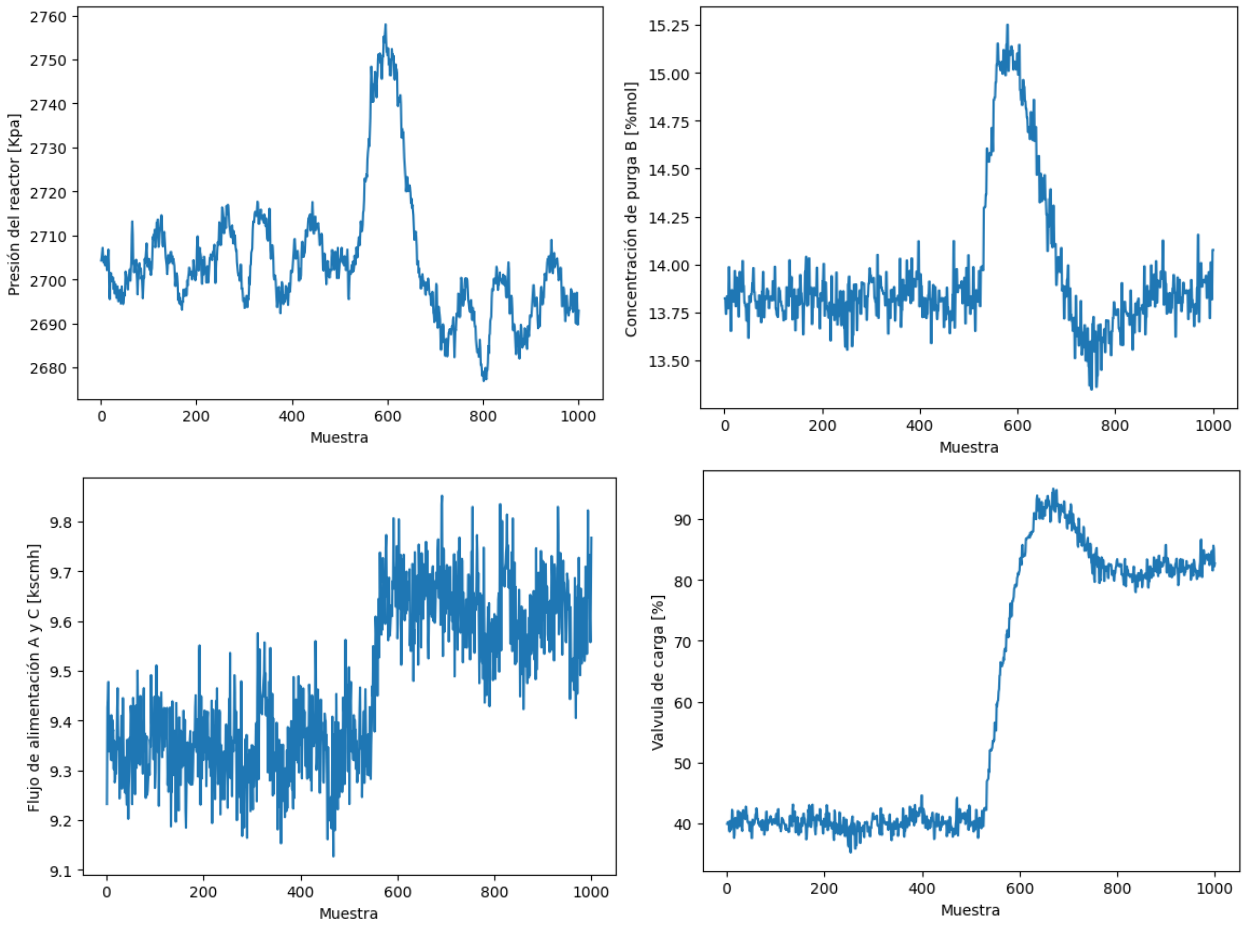
	Falla 2	Falla 12
Producción y rendimiento	X	X
Tiempo de operación	X	X
Integridad de los equipos		X
Sistema de respuesta		X

En la **Figura 5** se presenta el comportamiento de algunas variables medidas del proceso perturbadas correspondientes a la falla 2, como el flujo de alimentación, la presión del reactor y la concentración de purga del inerte, adicional a estas se tiene una variable manipulada que corresponde a la válvula de carga; esto como consecuencia de un aumento en el flujo de B, lo que genera, en primer lugar, un incremento en la presión del reactor que se ve reportada en el primer pico y a medida que esta variable aumenta llega a un punto en el que se debe abrir la válvula de purga. La apertura de esta válvula permite disminuir la presión, para una apertura mayor al 80 %,

perdiendo reactivo valioso. Por lo tanto, el flujo de alimentación de A y C debe aumentar para contrarrestar esto. A pesar de presentarse una perturbación en las variables anteriormente mencionadas, estas se encuentran dentro los límites de operación normales del TEP; el límite superior de presión del reactor es de 2895 kpa y este permanece por debajo de 2760 kpa.

Figura 5.

Comportamiento de la presión del reactor, concentración de purga, flujo de alimentación y válvula de carga

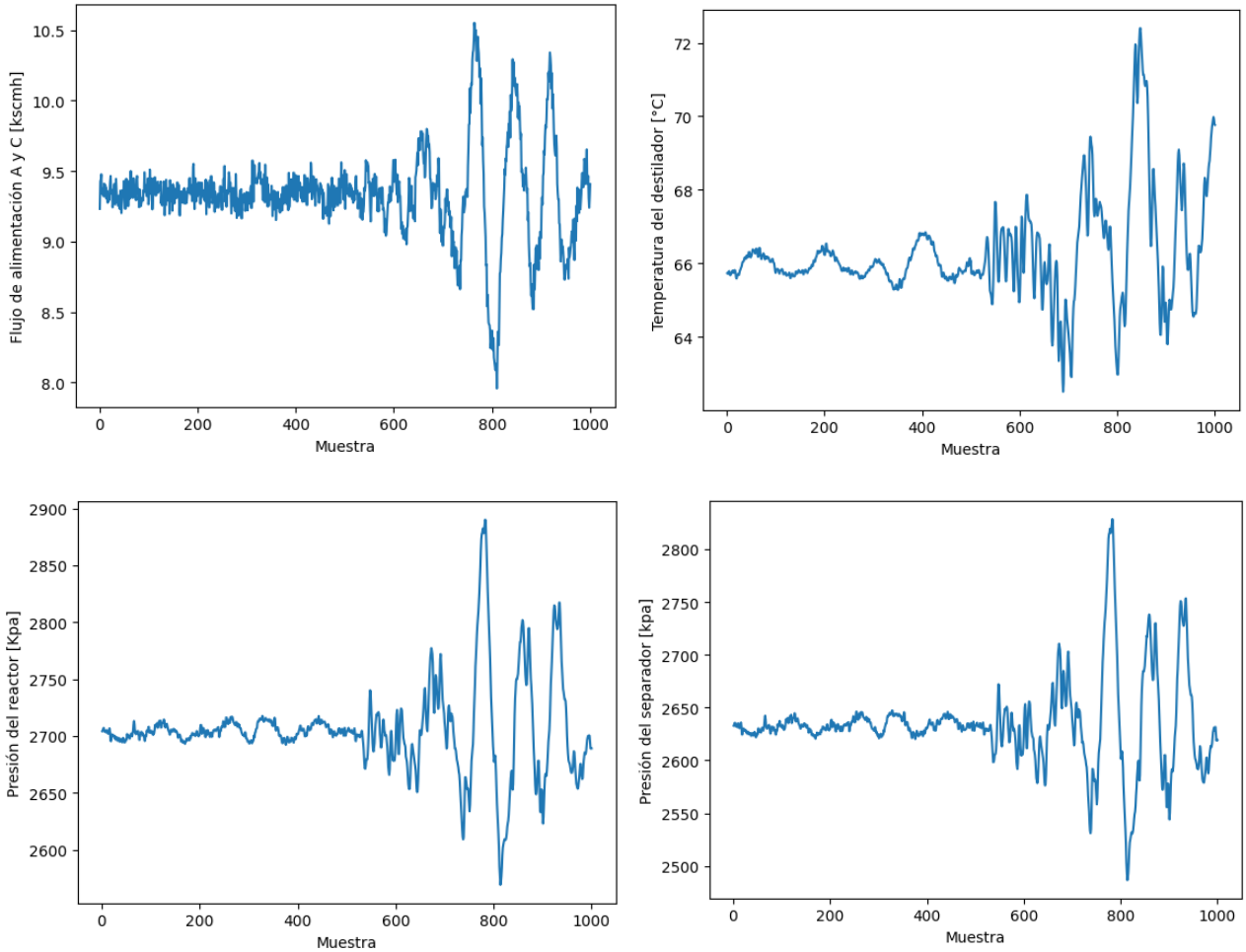


En la **Figura 6** se muestra el comportamiento de las variables del proceso perturbadas con la falla 12. En esta figura se evidencia un aumento en la entrada de vapor al separador L-V, lo que genera un incremento en la presión y de la temperatura del separador, provocando la recirculación

de gran parte del producto deseado junto con los reactivos y aumentando la presión del reactor simultáneamente y de todo el proceso. Conforme a lo mencionado en la literatura, esta falla provoca *shut down*, ya que las variables relacionadas en la **Figura 6** oscilan de manera inestable con la aparición de la falla 12, mostrando valores fuera de los límites de operación normal del TEP.

Figura 6.

Comportamiento del flujo de alimentación, temperatura del destilador, presión del reactor y presión del separador.



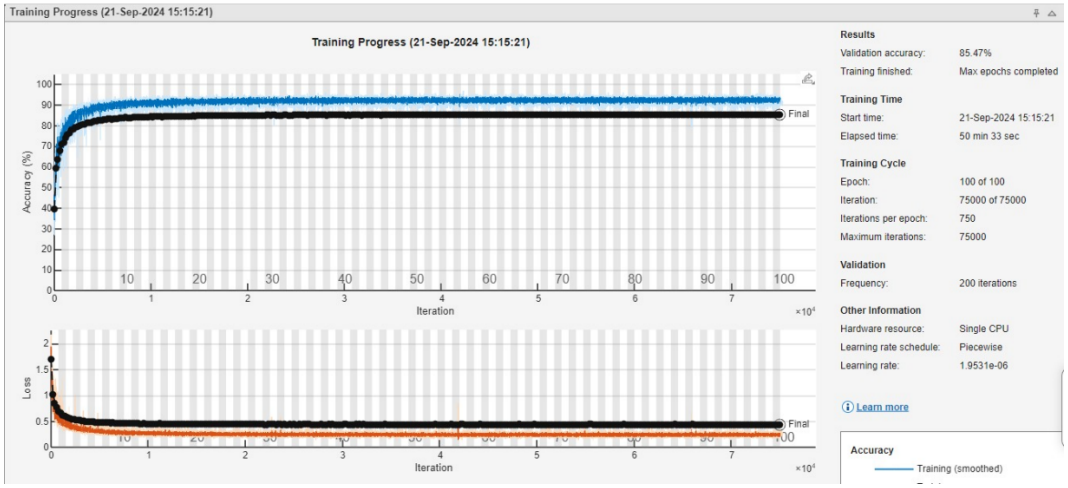
6.2 Red Neuronal de Predicción Base

El primer entrenamiento se realizó con 10 neuronas, 100 *epochs* y *mini-batches* de 1000. Se utilizó una tasa de aprendizaje de 0.001, con valores de sigma 1 y sigma 2 de 1 y 0.5, respectivamente. Además de la función de activación ReLU y el optimizador *SGDM*.

En la **Figura 7** se observa la *Accuracy* en la ordenada y las iteraciones en la abscisa. También se muestran las *epoch*, las cuales se encuentran en el eje horizontal de la **Figura 7**. A partir de la *epoch* 30, la tendencia no muestra un cambio significativo en el comportamiento, por lo cual, en las siguientes pruebas se tomará máximo esta cantidad de *epoch.* Por otro lado, está la gráfica de la pérdida, el valor de esta se encuentra en el eje vertical y en el horizontal se encuentra de igual manera las *epoch.* Durante las primeras épocas se evidencia que la pérdida disminuye rápidamente siendo este el comportamiento esperado, ya que significa que el entrenamiento está progresando correctamente y a medida que esta avanza, se tiende a estabilizar la pérdida, al igual que en la gráfica de *accuracy*, en donde luego de la *epoch* 30 no se observan cambios.

Figura 7.

Comportamiento de training

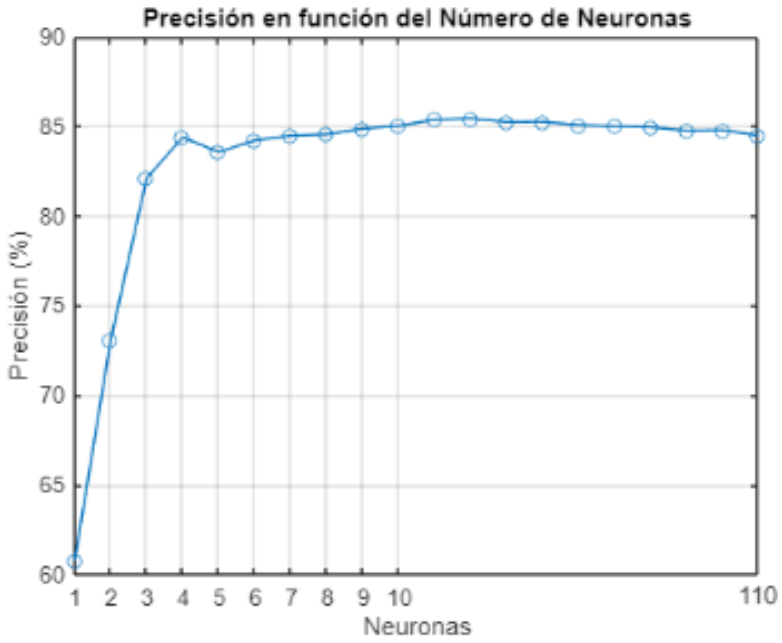


6.2.1 Análisis del Impacto de la Cantidad de Neuronas en la Red

En la **Figura 8** se presenta la variación del porcentaje de *accuracy* respecto al número de neuronas. Según esta figura, la precisión evidencia un aumento desde la neurona 1 hasta la neurona 10; a partir de 10 neuronas, la precisión se mantiene prácticamente constante en 85 %. Por lo tanto, se puede inferir que añadir más neuronas después de la décima no generará una mejora significativa en la precisión del modelo.

Figura 8.

Accuracy (ACC) Vs número de neuronas



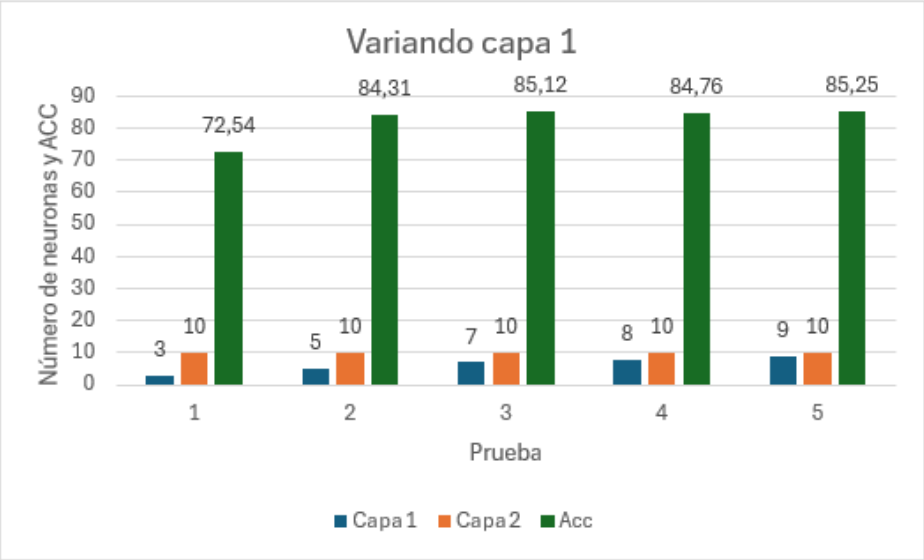
6.2.2 Número de Capas

Para definir la mejor arquitectura, se comenzó aumentando el número de capas, ya que, según la literatura (AlSaleh et al., 2024), las redes neuronales con múltiples capas suelen ofrecer mejores resultados. Al agregar una segunda capa, se decidió mantenerla fija en 10 neuronas, y variar la cantidad de neuronas en la primera capa. Posteriormente, se invirtió el proceso,

manteniendo fija la primera capa y modificando la segunda. De acuerdo con la **Figura 9**, un aumento en las neuronas de la primera capa se relaciona con un incremento en la precisión. Este comportamiento se mantiene hasta alcanzar una estabilidad que corresponde a la prueba 2, donde agregar más neuronas en la capa 1 no aumenta la precisión.

Figura 9.

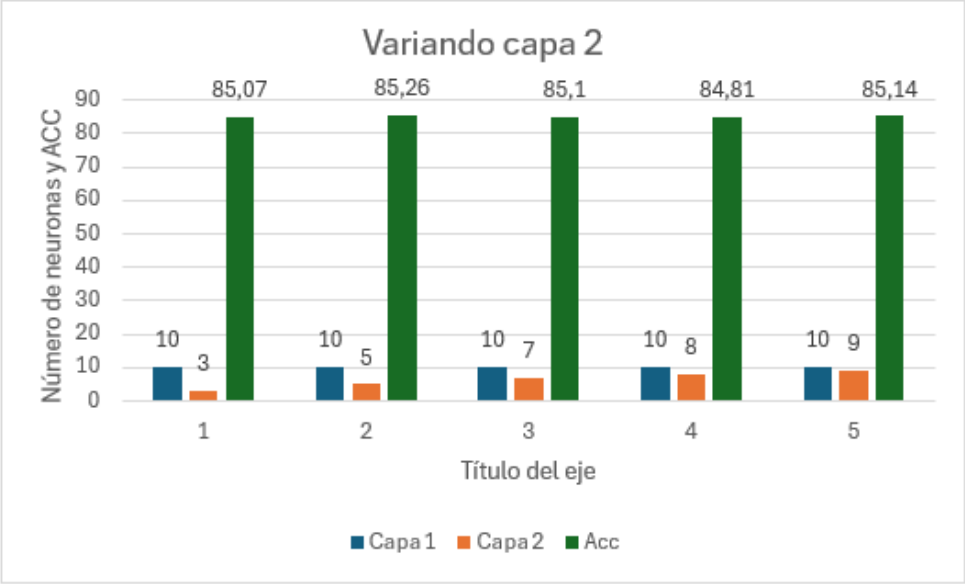
Accuracy (ACC) Vs variación de capa 1



En la **Figura 10** se puede evidenciar que, al mantener constante el número de neuronas de la capa 1, la precisión no se ve significativamente afectada por los cambios en el número de neuronas de la capa 2. Esto indica que el rendimiento de la red no depende en gran medida de las modificaciones de la capa 2.

Figura 10.

Accuracy (ACC) Vs variación capa 2



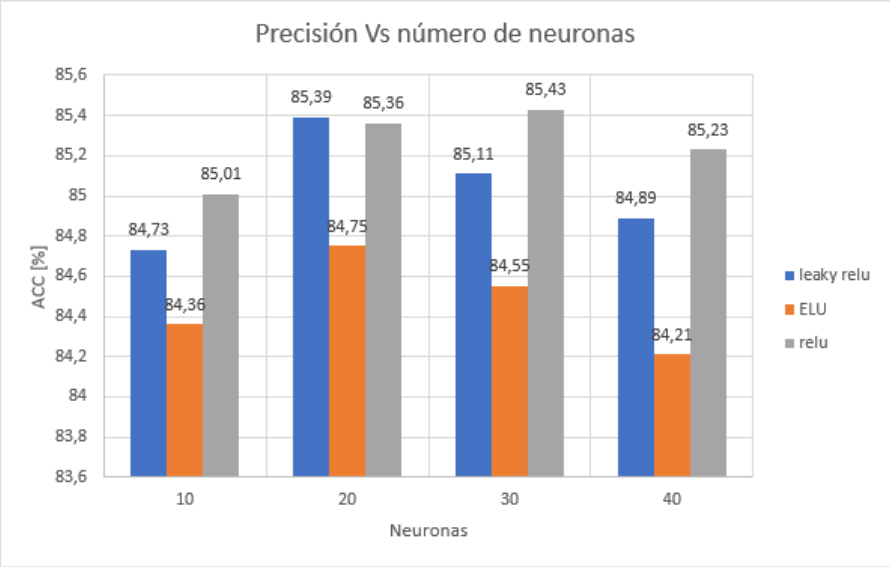
6.2.3 Funciones de Activación

De acuerdo con la **Figura 11**, la función de activación que mejores resultados reportó en el diagnóstico de fallas fue la *ReLU*, alcanzando una precisión de 85.4%. La función *Leaky ReLU* se aproximó a este valor, mientras que *ELU* fue inferior a 84.8%. Por otro lado, se consideraron otros aspectos, como el hecho de que la función de activación *Leaky ReLU* incluye un gradiente pequeño para mitigar el problema de neuronas inactivas (AlSaleh et al., 2024). También se tuvo en cuenta que la función de activación afectaba la velocidad del modelo.

Con lo anterior, se propuso un modelo híbrido que combinó las funciones de activación *ReLU* y *Leaky ReLU* para aprovechar las ventajas de la velocidad de procesamiento, minimizando las neuronas inactivas.

Figura 11.

Accuracy (ACC) Vs cantidad de neuronas



6.2.4 Ajustes en Parámetros y Precisión de la Red Neuronal

Con el propósito de evidenciar cómo se comportaba la red, se agregó otra capa. Se variaron los aspectos mencionados anteriormente, y para evitar sesgos en el modelo, se utilizó una función que ajustaba de manera aleatoria las *epochs*, el tamaño del *minibatch* y el número de neuronas por capa.

Normalmente, los resultados mostraban una precisión cercana al 85 %; sin embargo, al aumentar las condiciones como el tamaño del *minibatch*, se observó una disminución en la precisión debido a que los lotes más grandes tienden a suavizar las actualizaciones de los parámetros durante el entrenamiento. La **Tabla 5** presenta los resultados de las pruebas con los parámetros aleatorios.

Tabla 5.

Parámetros aleatorios

Epochs	48	35	57
MiniBacthsize	1918	1633	700
Numhidden 1	11	33	38
Numhidden 2	22	12	40
Numhidden 3	15	10	23
Accuracy [%]	83.88	83.37	85.65

La **Tabla 6** presenta los resultados de las pruebas de eliminación del sobreajuste, utilizando capas *dropout*. Según esta tabla, al utilizar un 50% de *dropout* en la prueba 3, la RNA reportó *underfitting*. La **Figura 12** ilustra la variación de la precisión de predicción de la RNA para entrenamiento (línea continua) y validación (puntos blancos). Según esta figura, los resultados de validación presentan mayores valores de *accuracy* que los correspondientes a la perdida (*loss*), lo cual sugiere que la estructura entrenada clasifica bien los casos usados en el aprendizaje, pero podría tener una baja eficiencia cuando se tengan nuevos casos (Rojas et al., 2012).

Por otro lado, la mejor respuesta de diagnóstico de falla se obtuvo con un 10% de neuronas inactivas, lo que indica que un nivel bajo de *dropout* optimiza la precisión del modelo.

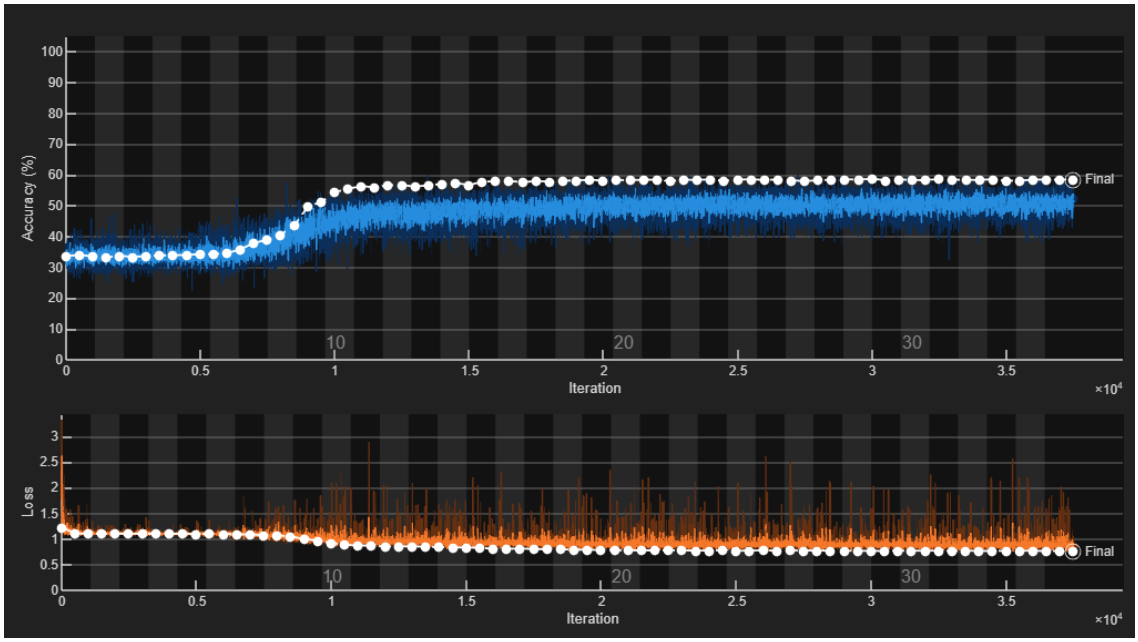
Tabla 6.

Accuracy (ACC) variando dropout

Prueba	1	2	3	4
Dropout	0,2	0,1	0,5	0,1
Epochs	50	40	35	5
MiniBacthsize	250	500	700	50
Capa 1	12	16	14	10
Capa 2	10	12	9	8
Capa 3	8	6	12	6
Accuracy [%]	86,76	86,64	Underfitting	87,33

Figura 12.

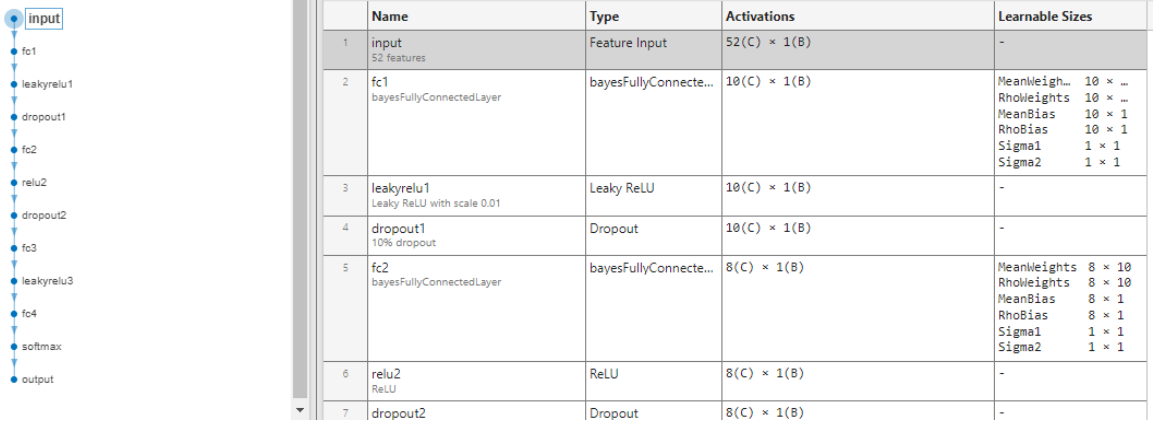
Prueba 3 con underfitting



Por otro lado, en la prueba 4, se utilizaron todos los parámetros mencionados anteriormente, dando como resultado la siguiente estructura:

Figura 13.

Estructura de la red bayesiana



Además, como se mencionó en la metodología, el segundo enfoque que se adoptó fue la optimización bayesiana, la cual permitió obtener mejores parámetros. Esta variaba el número de neuronas en las 3 capas, las sigmas y la tasa de aprendizaje, como se detalla en la **Tabla 7**.

En esta tabla se observa que la precisión se encuentra en orden descendente, donde se evidencia que en la iteración número 8 se obtiene la mayor precisión compuesta por 15, 13 y 11 neuronas para la capa 1, 2 y 3 respectivamente, así como una sigma 1 de 1.9139 y una sigma 2 de 0.12737. Estos parámetros fueron clave para maximizar la precisión del modelo, porque indica que el ajuste del número de neuronas en cada capa y la correcta elección de las sigmas juegan un papel importante en la mejora de la precisión.

Tabla 7.

Parámetros con optimización bayesiana

Iter	Accuracy %	Objective	Capa 1	Capa 2	Capa 3	Sigma1	Sigma2	LearnRate
8	87,707	0,12293	15	13	11	1,9139	0,12737	0,018703
4	87,669	0,12331	29	21	20	1,7953	1,9467	0,017551
5	87,661	0,12339	29	7	12	0,14234	1,8205	0,013501
10	87,638	0,12362	50	8	15	1,9798	0,52632	0,016055
9	87,633	0,12367	28	15	20	0,16919	1,9621	0,014803
6	87,624	0,12376	50	22	9	0,41841	0,91025	0,016511
2	87,386	0,12614	6	28	10	0,67869	0,19218	0,020339
7	87,256	0,12744	35	8	27	0,32799	1,4823	0,049318
3	86,491	0,13509	9	9	13	0,97243	0,98607	0,0011566
1	85,982	0,14018	42	25	16	0,82702	0,57087	0,00097988

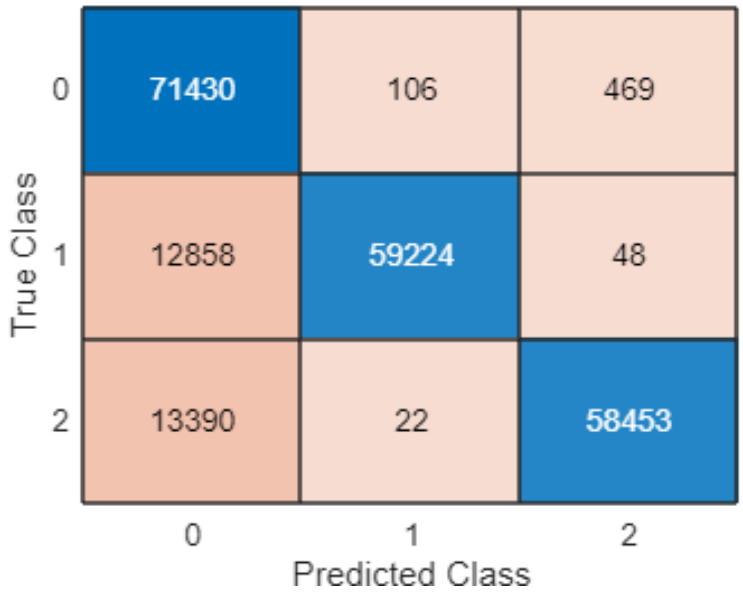
6.3 Selección de la Arquitectura con el Mejor Diagnóstico

La estructura de la Red Neuronal Bayesiana, que reportó el mayor desempeño en la predicción de las fallas 2 y 12, está compuesta por 5 capas, como se aprecia en la **Tabla 7**, esta arquitectura incluye una capa de entrada con 52 variables, seguida de 3 capas completamente conectadas: la primera con 15 neuronas, la segunda con 13 neuronas y la tercera con 11 neuronas. Finalmente, la red termina con una capa de salida que clasifica las entradas en 3 clases diferentes.

La red se entrenó durante 30 *epochs* con un tamaño de *minibacthsize* de 1000. En la **Figura 14** se muestra la matriz de confusión, en la cual se observa que la mayoría de los falsos positivos se encuentran en la identificación de la falla 0. Esto puede deberse a que los datos tomados no varían por la falla.

Figura 14.

Matriz de confusión falla 2 y 12



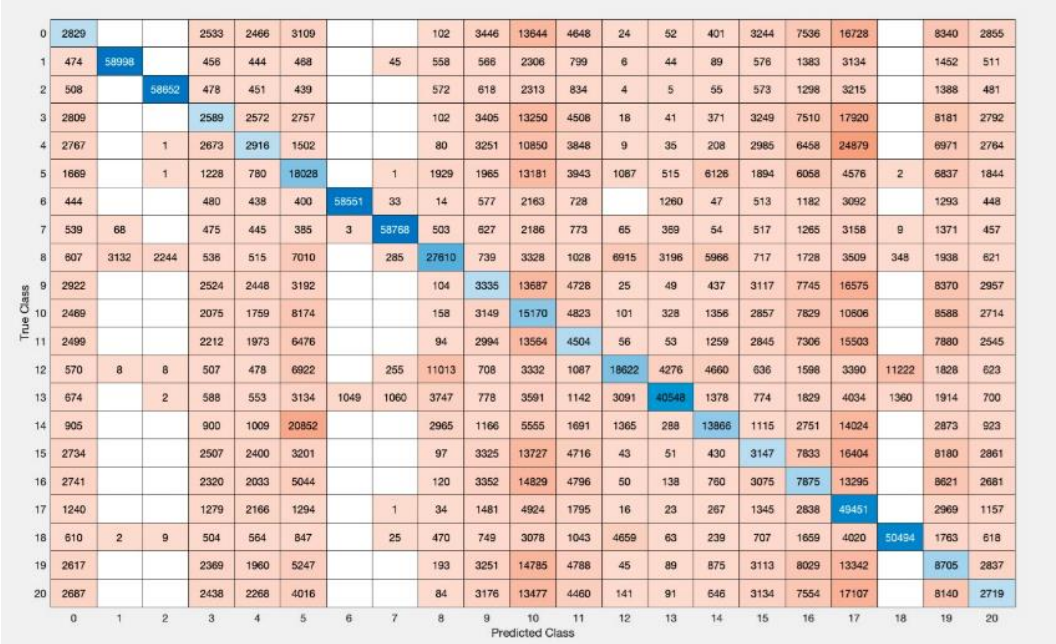
La estructura de RNA Bayesiana con el mejor desempeño fue entrenada para el diagnóstico de todas las fallas del TEP.

La **Figura 15** presenta la matriz de confusión obtenida del entrenamiento de la RNA Bayesiana con todas las fallas del TEP. En esta matriz, las filas representan los datos reales del proceso Tennessee Eastman y las columnas representan los datos que el modelo pudo predecir; la etiqueta 0 corresponde al proceso en operación normal, mientras las demás etiquetas corresponden al proceso operando con la respectiva falla. Las celdas de la diagonal principal de la matriz reportan los aciertos que tuvo la RNA en el diagnóstico correspondiente, mientras que los valores de las

celdas fuera de la diagonal principal corresponden a los falsos positivos (*i.e.* los diagnósticos erróneos reportados por la RNA). La matriz presenta una coloración que significa que, entre más azul oscuro sea, mayor es la precisión, y mientras más oscuro sea el tono rojo, más errores puede presentar la clasificación de la respectiva falla.

Figura 15.

Matriz de confusión todas las fallas



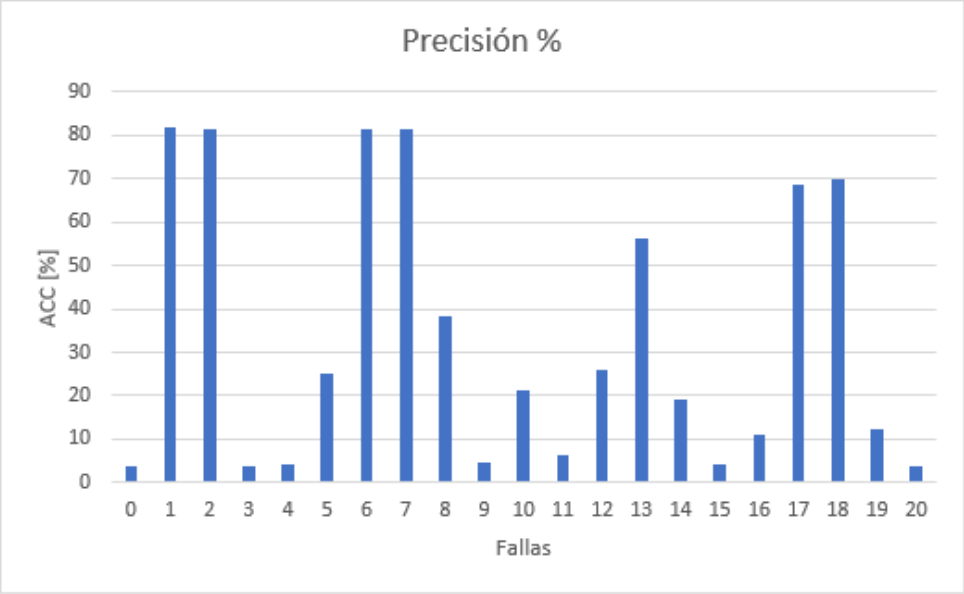
La **Figura 16** ilustra en un gráfico de barras los resultados de la diagonal principal de la matriz de confusión obtenida con la RNA Bayesiana de mejor desempeño; para esto, el porcentaje de acierto por cada diagnóstico se obtuvo con el valor máximo de cada conjunto de datos.

Como se observa en la **Figura 16**, los datos correspondientes a las fallas 1, 2, 6 y 7 presentan un desempeño mayor al 80%, lo que indica una capacidad satisfactoria para el diagnóstico de estas fallas. Por otra parte, la falla 12 reporta un desempeño inferior al 30%, lo cual puede atribuirse a la incorporación de un nuevo conjunto de datos y a los inconvenientes derivados de la incorporación de todas las fallas en el diagnóstico.

Finalmente, dando respuesta a la pregunta de investigación, el mayor porcentaje de *accuracy* obtenido en el diagnóstico de las fallas 2 y 12 del TEP fue de 87.7%. Este nivel de precisión fue obtenido con 3 capas ocultas compuestas cada una por 15, 13 y 11 neuronas, así como una sigma 1 de 1.9139 y una sigma 2 de 0.12737. Estos parámetros fueron clave para maximizar la precisión del modelo, lo cual indica que el ajuste del número de neuronas en cada capa y la correcta elección de los valores de sigma juegan un papel importante en la mejora de la precisión.

Figura 16.

Accuracy (ACC) Vs todas las fallas



7. Conclusiones

Los resultados obtenidos de los indicadores operativos muestran que las fallas identificadas generan problemas significativos en la producción y en el tiempo de operación. En particular la

falla 12, que presenta inconvenientes obligando a detener la planta, lo que resulta en pérdidas económicas considerables.

La mayoría de los modelos de RNA bayesiana que incorporaron más de 10 neuronas en la primera capa alcanzaron un rendimiento cercano al 85% en el diagnóstico de las fallas 2 y 12.

La estructura con mejor desempeño entre las redes neuronales correspondió a la de 3 capas ocultas, usando de manera alternada las funciones *de Leaky ReLU*, *ReLU* y *Leaky ReLU*. Además, se usó un *dropout* del 10 % de las neuronas y se optimizaron parámetros como las sigmas, el número de neuronas y la tasa de aprendizaje.

Según la evaluación del entrenamiento y la validación en todas las fallas, se pueden predecir bien las fallas 1, 2, 6 y 7. Sin embargo, debido al ruido presente en los datos, se tienen inconvenientes para detectar fallas como la 12.

En resumen, las redes neuronales bayesianas obtuvieron una precisión mayor al 85 % en la detección de las fallas 2 y 12 del proceso Tennessee Eastman, lo que indica su potencial en ámbitos industriales.

8. Recomendaciones

Para futuras investigaciones, se sugiere emplear herramientas que permitan detectar y eliminar las variables que se solapan entre sí, optimizando así el procesamiento de todas las fallas en la red bayesiana. Esto contribuirá a disminuir el ruido y a mejorar el ajuste de los pesos de la red.

Además, como trabajo futuro, la exploración de las redes híbridas compuestas por redes bayesianas y redes LSTM es recomendable en el diagnóstico del TEP. Estas últimas redes presentan una gran capacidad para manejar grandes volúmenes de datos.

Referencias bibliográficas

Abid A et al. (2017). *Multidomain Features-Based GA Optimized Artificial Immune System for Bearing Fault Detection*. https://ieeexplore.ieee.org/abstract/document/8031077/authors#authors

AlSaleh, I., Al-Samawi, A., & Nissirat, L. (2024). Novel Machine Learning Approach for DDoS Cloud Detection: Bayesian-Based CNN and Data Fusion Enhancements. *Sensors*, *24*(5), Article 5. https://doi.org/10.3390/s24051418

Araque Arellano, E. A. M., Aviles Sacoto, E. C., Vasconez Cruz, D. M., Alvarez Pulupa, D. N., Cuaran Sarzosa, F. V., Garcia Tumipamba, D. E., & Castro Salvador, P. (2018). *Gestión Ambiental en la Empresa Mediante la Norma Iso 14001-2015*. Editorial Universitaria Abya-Yala. https://dspace.ups.edu.ec/handle/123456789/17067

Blundell et al. (2015). *Entrenar una red neuronal bayesiana—MATLAB y Simulink—MathWorks España*. https://es.mathworks.com/help/deeplearning/ug/train-bayesian-neural-network.html

Cabezón Gutiérrez, S. (2014). *Control de Calidad en la Producción Industrial*. https://uvadoc.uva.es/handle/10324/13153

Calvo-Valverde, L. A., Argüello, S., Guzmán-Alvarez, J.-A., Guzmán-Quesada, M., & González-Zúñiga, M. (2019). Evaluación del uso de Redes Bayesianas Dinámicas para la predicción del avance de la Sigatoka negra y la productividad en cultivos agrícolas. *Tecnología en marcha*, *32*(4), 158-170.

Chiquito Tumbaco, S. L., Loor Alcivar, B. J., & Rodríguez Merchán, S. M. (2016). Sistema de Seguridad y Salud en el Trabajo. Transición de las OHSAS 18001: 2007 a la nueva ISO 45001. *Revista Publicando*, *3*(9), 638-648.

Downs, J. J., & Vogel, E. F. (1993). A plant-wide industrial process control problem. *Computers & Chemical Engineering*, *17*(3), 245-255. https://doi.org/10.1016/0098-1354(93)80018-I

Gao, T., Marié, S., Béguery, P., Thebault, S., & Lecoeuche, S. (2024). Integrated building fault detection and diagnosis using data modeling and Bayesian networks. *Energy and Buildings*, *306*, 113889. https://doi.org/10.1016/j.enbuild.2024.113889

Garcia, R. F. (2012, noviembre 27). *La catástrofe de la plataforma Deepwater Horizon: El coste de una irresponsabilidad*. Diario Responsable. https://diarioresponsable.com/opinion/16097-rse-catastrofe-deepwater-horizonirresponsabilidad

González Velázquez, M. (2020). *Mejora de la calidad de un proceso mediante la detección de anomalías basada en datos*. https://uvadoc.uva.es/handle/10324/44650

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation* (2nd ed.). Prentice Hall PTR.

Henderi, H., Wahyuningsih, T., & Rahwanto, E. (2021). Comparison of Min-Max normalization and Z-Score Normalization in the K-nearest neighbor (kNN) Algorithm to Test the Accuracy of Types of Breast Cancer. *International Journal of Informatics and Information Systems*, *4*(1), Article 1. https://doi.org/10.47738/ijiis.v4i1.73

Iturbe, M., Camacho, J., Garitano, I., Zurutuza, U., & Uribeetxeberria, R. (2016). *Distinguiendo entre perturbaciones de proceso e intrusiones en sistemas de control: Caso de estudio con el proceso Tennessee-Eastman*.

Izaurieta, F., & Saavedra, C. (2000). *Redes Neuronales Artificiales*.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*(1-3), 503-528. https://doi.org/10.1007/BF01589116

Llanes-Santiago, O., Prieto-Moreno, A., de Lázaro, J. M. B., Knupp, D. C., & Neto, A. J. S. (2017). A design proposal for multiblock-based fault diagnosis systems in complex industrial plants. *Chemometrics and Intelligent Laboratory Systems*, *162*, 149-159. https://doi.org/10.1016/j.chemolab.2017.01.015

Lopez, J., & Caicedo Bravo, E. (2005). *Entrenamiento Bayesiano para Redes Neuronales Artificiales.* (p. 41).

Lyman, P. R., & Georgakis, C. (1995). Plant-wide control of the Tennessee Eastman problem. *Computers & Chemical Engineering*, *19*(3), 321-331. https://doi.org/10.1016/0098-1354(94)00057-U

Márquez-Vera, M. A., López-Ortega, O., Ramos-Velasco, L. E., Ortega-Mendoza, R. M., Fernández-Neri, B. J., & Zúñiga-Peña, N. S. (2021). Diagnóstico de fallas mediante una LSTM y una red elástica. *Revista Iberoamericana de Automática e Informática industrial*, *18*(2), 164-175. https://doi.org/10.4995/riai.2020.13611

McCulloch K et al. (2019, enero 11). *Breve Historia de las Redes Neuronales Artificiales (Articulo 1)*. Steemit. https://steemit.com/spanish/@iars.geo/breve-historias-de-las-redes-neuronales-artificiales-articulo-1

Mejía-Neira, Á., Jabba, D., Caballero, G. C., Caicedo-Ortiz, J., Mejía-Neira, Á., Jabba, D., Caballero, G. C., & Caicedo-Ortiz, J. (2019). Influencia de la Ingeniería de Software en los Procesos de Automatización Industrial. *Información tecnológica*, *30*(5), 221-230. https://doi.org/10.4067/S0718-07642019000500221

Mesquida, A. L., Mas, A., Amengual, E., & Cabestrero, I. (2010). *Sistema de Gestión Integrado según las normas ISO 9001, ISO/IEC 20000 e ISO/IEC 2700*. *3*.

Ramos Fernández, J. (2019). *Aprendizaje automático para flujos de datos* [Masters, E.T.S. de Ingenieros Informáticos (UPM)]. https://oa.upm.es/56025/

Rieth, C. A., Amsel, B. D., Tran, R., & Cook, M. B. (2017). *Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation* [Dataset]. Harvard Dataverse. https://doi.org/10.7910/DVN/6C3JR1

Rojas, J. C. S., Pérez, D. U., & Reyes, C. E. H. (2012, junio 21). *Definición de Redes Bayesianas y sus aplicaciones*. Revista Vinculando. https://vinculando.org/articulos/redes-bayesianas.html

Salas, R. (2004). *Redes Neuronales Artificiales*.

Socializacion decreto 1347 del 2021 y Ley 1523 del 2012.pdf. (2012). Google Docs. https://drive.google.com/file/d/1hkEDbkhZhILZPYjbbVFQAywN5mZem-ts/view?usp=drivesdk&usp=embed_facebook

ssalgadodev. (2024a, septiembre 3). *Ajuste de hiperparámetros de un modelo (v2)—Azure Machine Learning*. https://learn.microsoft.com/es-es/azure/machine-learning/how-to-tune-hyperparameters?view=azureml-api-2

ssalgadodev. (2024b, septiembre 3). *Ajuste de hiperparámetros de un modelo (v2)—Azure Machine Learning*. https://learn.microsoft.com/es-es/azure/machine-learning/how-to-tune-hyperparameters?view=azureml-api-2

Sun, W., Paiva, A. R. C., Xu, P., Sundaram, A., & Braatz, R. D. (2020). Fault detection and identification using Bayesian recurrent neural networks. *Computers & Chemical Engineering*, *141*, 106991. https://doi.org/10.1016/j.compchemeng.2020.106991

Sucar, L. E., & Tonantzintla, M. (2006). Redes bayesianas. *Aprendizaje Automático: conceptos básicos y avanzados*, *77*, 100.

Tarcsay, B. L., Bárkányi, Á., Németh, S., Chován, T., Lovas, L., & Egedy, A. (2024). Risk-Based Fault Detection Using Bayesian Networks Based on Failure Mode and Effect Analysis. *Sensors*, *24*(11), Article 11. https://doi.org/10.3390/s24113511

Toro, L. A. A., & Sotomayor, O. (2008). INTEGRACIÓN OPTIMIZACIÓN – CONTROl PREDICTIVO Y APLICACIÓN A LA PLANTA TENNESSEE EASTMAN. *Revista Peruana de Química e Ingeniería Química*, *11*(1), Article 1.

Yang, W.-T., Reis, M. S., Borodin, V., Juge, M., & Roussy, A. (2022). An interpretable unsupervised Bayesian network model for fault detection and diagnosis. *Control Engineering Practice*, *127*, 105304. https://doi.org/10.1016/j.conengprac.2022.105304

Apéndices

Apéndice A.

Variables medidas del TEP

Variable	Nombre de variable	Unidades
XMEAS (1)	Flujo de alimentación A	Kscmh
XMEAS (2)	Flujo de alimentación D	Kscmh
XMEAS (3)	Flujo de alimentación E	Kscmh
XMEAS (4)	Flujo de alimentación A y C	Kscmh
XMEAS (5)	Flujo de recirculación	Kscmh
XMEAS (6)	Flujo de alimentación reactor	kscmh
XMEAS (7)	Presión del reactor	Kpa
XMEAS (8)	Nivel del reactor	%
XMEAS (9)	Temperatura del reactor	°C
XMEAS (10)	Flujo de purga	kscmh
XMEAS (11)	Temperatura del separador	°C
XMEAS (12)	Nivel del separador	%
XMEAS (13)	Presión del separador	kpa
XMEAS (14)	Corriente del separador	m^3h^{-1}
XMEAS (15)	Nivel de destilador (stripper)	%
XMEAS (16)	Presión del destilador (stripper)	kpa
XMEAS (17)	Corriente del destilador (stripper)	m^3h^{-1}
XMEAS (18)	Temperatura del destilador (stripper)	°C
XMEAS (19)	Flujo de vapor del destilador (stripper)	kgh^{-1}
XMEAS (20)	Potencia de compresor	KW
XMEAS (21)	Temperatura de la salida del agua de refrigeración del reactor	°C
XMEAS (22)	Temperatura de la salida del agua de refrigeración del separador	°C
XMEAS (23-28)	Concentración de la alimentación del reactor (A-F)	% mol
XMEAS (29-36)	Concentración de la purga (A-H)	% mol
XMEAS (37-41)	Concentración aguas abajo del destilador (A-H)	% mol

Apéndice B. Comportamiento de variables importantes con falla

Figura 17.

Comportamiento de variables importantes de la falla 2

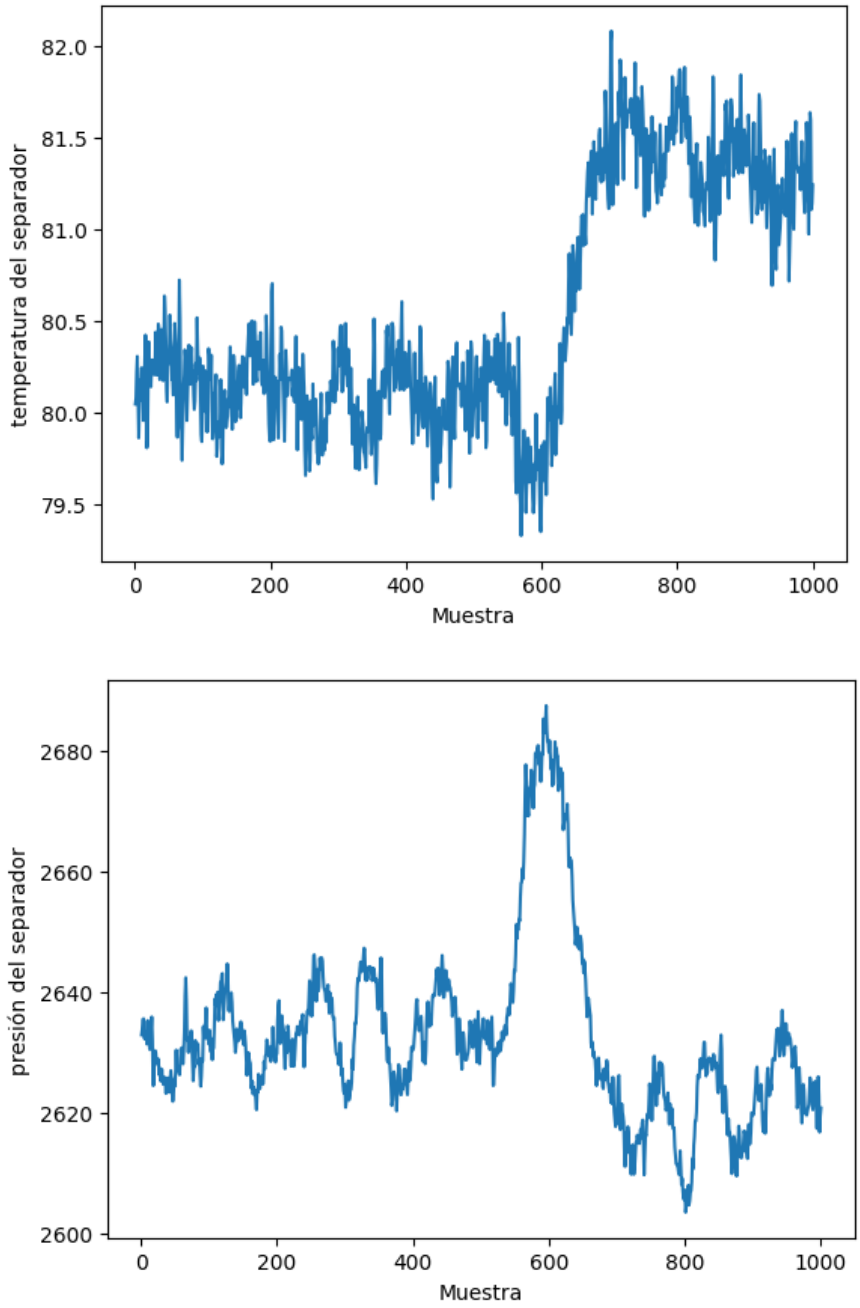
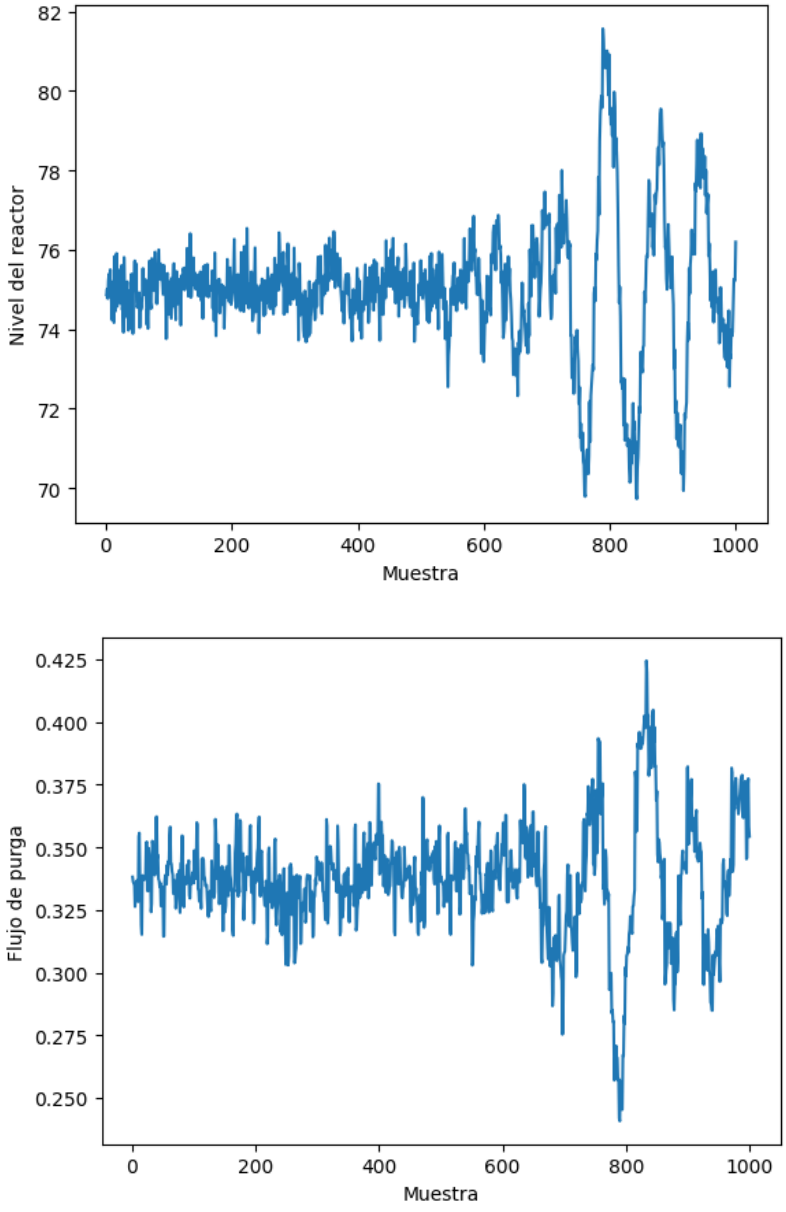


Figura 18.

Comportamiento de variables importantes de la falla 12



Apéndice C. Código en MATLAB para el pretratamiento de los datos

Figura 19.

Código en MATLAB para pretratamiento de datos

```
clc;
clearvars;
close all;

rng(0);

% Cargar datos
load('faultfreetesting.mat');
load('faultfreetraining.mat');
load('faultytesting.mat');
load('faultytraining.mat');

% Filtrar datos
faultytesting = faultytesting(~ismember(faultytesting.faultNumber, [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 1
faultytraining = faultytraining(~ismember(faultytraining.faultNumber, [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18

Training_0 = faultfreetraining;
Testing_0 = faultfreetesting;
```

```

clearvars faultfreetraining faultfreetesting;

% Filtrar datos para faultNumber 2 y 12
Testing_2 = faultytesting(faultytesting.faultNumber == 2, :);
Testing_12 = faultytesting(faultytesting.faultNumber == 12, :);

clearvars faultytesting;

Training_2 = faultytraining(faultytraining.faultNumber == 2, :);
Training_12 = faultytraining(faultytraining.faultNumber == 12, :);

clearvars faultytraining;

% Añadir etiquetas a los datos de entrenamiento y testing
Training_all = vertcat(Training_0, Training_2, Training_12); % Datos de entrenamiento
Testing_all = vertcat(Testing_0, Testing_2, Testing_12); % Datos de testing

% Preprocesamiento: Normalizar características numéricas
Training_all = Training_all(:, 4:55);
```

```
Testing_all = Testing_all(:, 4:55);

numVars = Training_all.Properties.VariableNames;
%z-score
mu = struct();
sigma = struct();
for i = 1:length(numVars)
    mu.(numVars{i}) = mean(Training_all.(numVars{i}));
    sigma.(numVars{i}) = std(Training_all.(numVars{i}));
    Training_all.(numVars{i}) = (Training_all.(numVars{i}) - mu.(numVars{i})) / sigma.(numVars{i});
    Testing_all.(numVars{i}) = (Testing_all.(numVars{i}) - mu.(numVars{i})) / sigma.(numVars{i});
end

% Extraer características y etiquetas
XTrain = table2array(Training_all);
YTrain = categorical([zeros(size(Training_0,1),1); ones(size(Training_2,1),1); 2*ones(size(Training_12,1),1)]);

XTest = table2array(Testing_all);
YTest = categorical([zeros(size(Testing_0,1),1); ones(size(Testing_2,1),1); 2*ones(size(Testing_12,1),1)]);

```

```
% Dividir datos para validación
validationTestIdx = randperm(size(XTest, 1), round(0.15 * size(XTest, 1)));
XValidation = XTest(validationTestIdx, :);
YValidation = YTest(validationTestIdx, :);

% Eliminar los datos de validación de los datos de testing
XTest(validationTestIdx, :) = [];
YTest(validationTestIdx, :) = [];

% Crear otro conjunto con 15% de XTest (tamaño igual a XValidation)
testSubsetIdx = randperm(size(XTest, 1), size(XValidation, 1)); % mismo tamaño que XValidation
XTestSub = XTest(testSubsetIdx, :);
YTestSub = YTest(testSubsetIdx, :);


% Guardar los datos en un archivo MAT
save('/MATLAB Drive/DatosEntrenamiento.mat', 'XTrain', 'YTrain', 'XTestSub', 'YTestSub', 'XValidation', 'YValidation');
```

Apéndice D. Código en MATLAB para en entrenamiento de los datos

Figura 20.

Código en MATLAB para entrenamiento de datos

codigobaye.m

```
clc;
clearvars;
close all;

rng(0);

% Cargar datos
load('faultfreetesting.mat');
load('faultfreetraining.mat');
load('faultytesting.mat');
load('faultytraining.mat');

Training_0 = faultfreetraining;
Testing_0 = faultfreetesting;

clearvars faultfreetraining faultfreetesting;

% Todas las fallas

Testing_1 = faultytesting(faultytesting.faultNumber == 1, :);
Testing_2 = faultytesting(faultytesting.faultNumber == 2, :);
Testing_3 = faultytesting(faultytesting.faultNumber == 3, :);
Testing_4 = faultytesting(faultytesting.faultNumber == 4, :);
Testing_5 = faultytesting(faultytesting.faultNumber == 5, :);
Testing_6 = faultytesting(faultytesting.faultNumber == 6, :);
Testing_7 = faultytesting(faultytesting.faultNumber == 7, :);
Testing_8 = faultytesting(faultytesting.faultNumber == 8, :);
Testing_9 = faultytesting(faultytesting.faultNumber == 9, :);
```

```
Testing_10 = faultytesting(faultytesting.faultNumber == 10, :);
Testing_11 = faultytesting(faultytesting.faultNumber == 11, :);
Testing_12 = faultytesting(faultytesting.faultNumber == 12, :);
Testing_13 = faultytesting(faultytesting.faultNumber == 13, :);
Testing_14 = faultytesting(faultytesting.faultNumber == 14, :);
Testing_15 = faultytesting(faultytesting.faultNumber == 15, :);
Testing_16 = faultytesting(faultytesting.faultNumber == 16, :);
Testing_17 = faultytesting(faultytesting.faultNumber == 17, :);
Testing_18 = faultytesting(faultytesting.faultNumber == 18, :);
Testing_19 = faultytesting(faultytesting.faultNumber == 19, :);
Testing_20 = faultytesting(faultytesting.faultNumber == 20, :);

clearvars faultytesting;

Training_1 = faultytraining(faultytraining.faultNumber == 1, :);
Training_2 = faultytraining(faultytraining.faultNumber == 2, :);
Training_3 = faultytraining(faultytraining.faultNumber == 3, :);
Training_4 = faultytraining(faultytraining.faultNumber == 4, :);
Training_5 = faultytraining(faultytraining.faultNumber == 5, :);
Training_6 = faultytraining(faultytraining.faultNumber == 6, :);
Training_7 = faultytraining(faultytraining.faultNumber == 7, :);
Training_8 = faultytraining(faultytraining.faultNumber == 8, :);
Training_9 = faultytraining(faultytraining.faultNumber == 9, :);
Training_10 = faultytraining(faultytraining.faultNumber == 10, :);
Training_11 = faultytraining(faultytraining.faultNumber == 11, :);
Training_12 = faultytraining(faultytraining.faultNumber == 12, :);
Training_13 = faultytraining(faultytraining.faultNumber == 13, :);
```

```
Training_14 = faultytraining(faultytraining.faultNumber == 14, :);
Training_15 = faultytraining(faultytraining.faultNumber == 15, :);
Training_16 = faultytraining(faultytraining.faultNumber == 16, :);
Training_17 = faultytraining(faultytraining.faultNumber == 17, :);
Training_18 = faultytraining(faultytraining.faultNumber == 18, :);
Training_19 = faultytraining(faultytraining.faultNumber == 19, :);
Training_20 = faultytraining(faultytraining.faultNumber == 20, :);


clearvars faultytraining;

% Añadir etiquetas a los datos de entrenamiento y testing
Training_all = vertcat(Training_0, Training_1, Training_2, Training_3, Training_4, Training_5,


Testing_all = vertcat(Testing_0, Testing_1, Testing_2, Testing_3, Testing_4, Testing_5, Testing

% Preprocesamiento: Normalizar características numéricas
Training_all = Training_all(:, 4:55);
Testing_all = Testing_all(:, 4:55);

numVars = Training_all.Properties.VariableNames;

mu = struct();
sigma = struct();
for i = 1:length(numVars)
    mu.(numVars{i}) = mean(Training_all.(numVars{i}));
```

```
    sigma.(numVars{i}) = std(Training_all.(numVars{i}));
    Training_all.(numVars{i}) = (Training_all.(numVars{i}) - mu.(numVars{i})) / sigma.(numVars{
    Testing_all.(numVars{i}) = (Testing_all.(numVars{i}) - mu.(numVars{i})) / sigma.(numVars{i}
end

% Extraer características y etiquetas
XTrain = table2array(Training_all);
YTrain = categorical([zeros(size(Training_0,1),1);
                      ones(size(Training_1,1),1);
                      2*ones(size(Training_2,1),1);
                      3*ones(size(Training_3,1),1);
                      4*ones(size(Training_4,1),1);
                      5*ones(size(Training_5,1),1);
                      6*ones(size(Training_6,1),1);
                      7*ones(size(Training_7,1),1);
                      8*ones(size(Training_8,1),1);
                      9*ones(size(Training_9,1),1);
                      10*ones(size(Training_10,1),1);
                      11*ones(size(Training_11,1),1);
                      12*ones(size(Training_12,1),1);
                      13*ones(size(Training_13,1),1);
                      14*ones(size(Training_14,1),1);
                      15*ones(size(Training_15,1),1);
                      16*ones(size(Training_16,1),1);
                      17*ones(size(Training_17,1),1);
                      18*ones(size(Training_18,1),1);
                      19*ones(size(Training_19,1),1);
                      20*ones(size(Training_20,1),1);
                      ]);

XTest = table2array(Testing_all);
YTest = categorical([zeros(size(Testing_0,1),1);
                     ones(size(Testing_1,1),1);
                     2*ones(size(Testing_2,1),1);
                     3*ones(size(Testing_3,1),1);
                     4*ones(size(Testing_4,1),1);
                     5*ones(size(Testing_5,1),1);
                     6*ones(size(Testing_6,1),1);
                     7*ones(size(Testing_7,1),1);
                     8*ones(size(Testing_8,1),1);
                     9*ones(size(Testing_9,1),1);
                     10*ones(size(Testing_10,1),1);
                     11*ones(size(Testing_11,1),1);
                     12*ones(size(Testing_12,1),1);
                     13*ones(size(Testing_13,1),1);
                     14*ones(size(Testing_14,1),1);
                     15*ones(size(Testing_15,1),1);
                     16*ones(size(Testing_16,1),1);
                     17*ones(size(Testing_17,1),1);
                     18*ones(size(Testing_18,1),1);
                     19*ones(size(Testing_19,1),1);
                     20*ones(size(Testing_20,1),1);
                                                 ]);
```

```
% Dividir datos para validación
validationTestIdx = randperm(size(XTest, 1), round(0.15 * size(XTest, 1)));
XValidation = XTest(validationTestIdx, :);
YValidation = YTest(validationTestIdx, :);

% Eliminar los datos de validación de los datos de testing
XTest(validationTestIdx, :) = [];
YTest(validationTestIdx, :) = [];

% Crear otro conjunto con 15% de XTest (tamaño igual a XValidation)
testSubsetIdx = randperm(size(XTest, 1), size(XValidation, 1)); % mismo tamaño que XValidation
XTestSub = XTest(testSubsetIdx, :);
YTestSub = YTest(testSubsetIdx, :);


epoch=200;
mb=1000;%MiniBachsize
numHiddenUnits1 = 49;%NeuronasCAPA 1
numHiddenUnits2 = 7;%CAPA 2
numHiddenUnits3 = 8;%Capa 3
numClasses = 21; % Número de clases
numSignals = 52;

% Parámetros sigma ajustados para mejor estabilidad
sigma1 = 0.68321;
sigma2 = 0.46604;

```

```
% Definir las capas de la red
% Definir las capas de la red con regularización y dropout
layers = [
    featureInputLayer(numSignals, 'Name', 'input')
    bayesFullyConnectedLayer(numHiddenUnits1, 'Sigma1', sigma1, 'Sigma2', sigma2, 'Name', 'fc1'
    leakyReluLayer('Name', 'leakyrelu1')
    dropoutLayer(0.1, 'Name', 'dropout1')  % Dropout para prevenir sobreajuste
    bayesFullyConnectedLayer(numHiddenUnits2, 'Sigma1', sigma1, 'Sigma2', sigma2, 'Name', 'fc2'
    reluLayer('Name', 'relu2')
    dropoutLayer(0.1, 'Name', 'dropout2')  % Otro Dropout entre capas
    bayesFullyConnectedLayer(numHiddenUnits3, 'Sigma1', sigma1, 'Sigma2', sigma2, 'Name', 'fc3'
    leakyReluLayer('Name', 'leakyrelu3')
    bayesFullyConnectedLayer(numClasses, 'Sigma1', sigma1, 'Sigma2', sigma2, 'Name', 'fc4')
    softmaxLayer('Name', 'softmax')
    classificationLayer('Name', 'output')
];

% Crear grafo de la red
lgraph = layerGraph(layers);
analyzeNetwork(lgraph);

% Opciones de entrenamiento
options = trainingOptions('sgdm', ...
    'MaxEpochs', epoch, ...
    'MiniBatchSize', mb, ...            % Reducir tamaño de lote para estabilidad
    'InitialLearnRate', 0.0037823, ...        % Tasa de aprendizaje ajustada
    'LearnRateSchedule', 'piecewise', ...
```

```
        'LearnRateDropFactor', 0.5, ...     % Disminuir la tasa de aprendizaje
        'LearnRateDropPeriod', 10, ...
        'ValidationData', {XValidation, YValidation}, ...
        'ValidationFrequency', 200, ...
        'ValidationPatience', 30, ...       % Desactivar parada temprana
        'Verbose', 1, ...
        'Plots', 'training-progress');


    net = trainNetwork(XTrain, YTrain, lgraph, options);
    %% Evaluar el modelo entrenado en los datos de testeo
    YPred = classify(net, XTestSub);  % Predicciones para los datos de prueba
    accuracy = sum(YPred == YTestSub) / numel(YTestSub);  % Calcular la precisión

    % Mostrar precisión
    fprintf('Precisión del modelo en los datos de testeo: %.2f%%\n', accuracy * 100);


    %Matriz de confusión
    confusionchart(YTestSub, YPred);
```

Apéndice E. Código de PYTHON para graficas

```
#Importar las librerias
import pyreadr
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')#Acceso a drive
#Leer los documentos
#dff_test= pyreadr.read_r('/content/drive/MyDrive/Tesis/TEP_FaultFree_Testing.RData')['fault_free_testing']
dff_trai=pyreadr.read_r('/content/drive/MyDrive/Tesis/TEP_FaultFree_Training.RData')['fault_free_training']
#dfy_test=pyreadr.read_r('/content/drive/MyDrive/Tesis/TEP_Faulty_Testing.RData')['faulty_testing']
dfy_trai=pyreadr.read_r('/content/drive/MyDrive/Tesis/TEP_Faulty_Training.RData')['faulty_training']
f2_trai = pd.concat([dff_trai, dfy_trai[dfy_trai["faultNumber"] == 2]])
f12_trai = pd.concat([dff_trai, dfy_trai[dfy_trai["faultNumber"] == 12]])
#Seleccionar la simulacion 1
dff_f2s1 = f2_trai[f2_trai["simulationRun"] == 1]
```

```
#GRAFICAR FALLA 2
dff=dff_f2s1
dff['sample']=list(range(1,1001))
dff.plot(x='sample', y='xmeas_1').set(xlabel='Muestra', ylabel='Flujo de alimentación A y C [kscmh]')
legend = plt.legend()
legend.set_visible(False)
dff.plot(x='sample', y='xmeas_2').set(xlabel='Muestra', ylabel='Presión del reactor [Kpa]')
legend = plt.legend()
legend.set_visible(False)
dff.plot(x='sample', y='xmeas_3').set(xlabel='Muestra', ylabel='Flujo de purga [kscmh]')
legend = plt.legend()
legend.set_visible(False)
dff.plot(x='sample', y='xmeas_4').set(xlabel='Muestra', ylabel='Temperatura del destilador [°C]')
legend = plt.legend()
legend.set_visible(False)
dff['sample']=list(range(1,1001))
dff.plot(x='sample', y='xmeas_5').set(xlabel='Muestra', ylabel='Flujo de alimentación A y C [kscmh]')
legend = plt.legend()
legend.set_visible(False)
```