Detección de fallas operacionales con Redes Neuronales Artificiales: aplicación del proceso Tennessee Eastman

Jenifer Dayanna Cárdenas Manrique y Sebastian Reyes Angarita

Trabajo de Grado para Optar el Título de Ingeniero Químico

Director

Giovanni Morales Medina

Doctor en Ingeniería Química

Universidad Industrial de Santander

Facultad de Ingenierías Fisicoquímicas

Escuela de Ingeniería Química

Bucaramanga

2021

Dedicatoria

A Dios a quien le debo todo en mi vida, a mis padres y hermano, Fredy, Tania y Camilo por brindarme su apoyo incondicional aun en los momentos más difíciles de mi carrera, a Christian a quien nunca le terminare de agradecer todo lo que hace por mí, por siempre brindarme su apoyo y ser una persona incondicional en mi vida, a mi familia especialmente a todos aquellos que estuvieron cuando más los necesite en este camino, a mi tía Sandra y mi abuela Lucila que siempre han sido como unas segundas madres para mí, a Gaia por acompañarme siempre, los amo.

Y, por último, pero no menos importante a mis amigos que hicieron de esta etapa de mi vida inolvidable y estuvieron conmigo en los buenos y malos momentos brindándome su cariño y comprensión.

Dayanna Cárdenas

Agradecimientos

A la Universidad Industrial de Santander, especialmente a la escuela de Ingeniería Química que junto a sus Docentes estuvieron brindándonos todos sus conocimientos a lo largo de este camino, al profesor Giovanni por su acompañamiento, paciencia y ayuda en la realización de este proyecto de grado, nada de esto sería posible sin él, y en general a todas las personas que de una u otra manera contribuyeron a que esto fuera posible.

Dedicatoria

A mi madre, Carolina por brindarme su apoyo incondicional en todo momento y por ser la persona más importante en mi vida y a mi familia, mis tíos, tías y mis abuelos por ayudarme en los buenos y malos momentos, los amo.

Sebastian Reyes

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Resumen

Título: Detección de fallas operacionales con Redes Neuronales Artificiales: aplicación del proceso Tennessee Eastman[*]

Autores: Jenifer Dayanna Cárdenas Manrique, Sebastian Reyes Angarita[**]

Palabras Clave: Redes Neuronales Artificiales, proceso Tennessee Eastman, detección de fallas.

Descripción: Este trabajo de grado tiene como finalidad determinar una estructura de Red Neuronal Artificial (RNA) óptima para la detección de fallas operacionales a partir del proceso Tennessee Eastman (TE), Inicialmente se hizo una revisión bibliográfica de los datos históricos del proceso TE, sus variables y fallas, además de las limitaciones que presentan los equipos y sus condiciones de operación normal, con estos datos se realizó el entrenamiento de diferentes estructuras de Red en dos software de simulación. En el software R se tuvo en cuenta el paquete de entrenamiento neuralnet, mientras que en el software Python se usó la interfaz de programación Keras, una vez planteadas las diferentes estructuras se ejecutó el código considerando los datos de fallas y de operación estable del proceso, además se variaron parámetros como la cantidad de neuronas, la función de activación, el número de capas ocultas, entre otras. Esto con el fin de poder establecer la estructura de red que menor error tuviera en la detección de fallas, dando como resultado que la Red que mejor se adapta en el proceso de detección de fallas es la red programada en Python con una estructura de 52:11:1 debido a que presenta el menor error cuadrático medio. Adicional a esto se tuvo como resultado que la RNA presenta una eficiencia en la predicción del 97% para la detección de fallas.

---

[*] Trabajo de Grado

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

Abstract

Title: Detection of Operational Failures with Artificial Neural Network: Application in Tennessee Eastman Process**.**

Authors: Jenifer Dayanna Cárdenas Manrique, Sebastian Reyes Angarita[**]

Key Words: Artificial Neural Network, Tennessee Eastman Process, Failure Detection.

Description: The purpose of this degree paper is to determine an optimal Artificial Neural Network (ANN) structure for the detection of operational failures from the Tennessee Eastman Process (TEP). Initially, a literature review was conducted of the historical data of the TEP, its variables, and failures, in addition to the limitations presented by the equipment and its normal operating conditions. With these data, the training of different network structures was carried out in two simulation software. In the R software, the Neuralnet training package was taken into account, while in the Python software the Keras programming interface was used. Once the different structures had been proposed, the code was executed considering the data of failures and the stable operation of the process. Besides, parameters such as the number of neurons, the activation function, the number of hidden layers, among others, were tested. This, in order to be able to establish the network structure that had the least error in the detection of failures. As result, it was found that the Network that adapts the best to the failure detection process is the network programmed in Python with a structure of 52: 11: 1 because it has the smallest Mean Square Error (MSE). In addition to this, it was discovered that RNA has a prediction efficiency of 97% for the detection of failures.

---

[*] Degree Work

[**] Faculty of Physicochemical Engineering. Chemical Engineering School. Director: Giovanni Morales Medina, Ph.D. Chemical Engineering

Introducción

En la industria, las exigencias ambientales y de confiabilidad de los procesos productivos han impulsado el desarrollo y la aplicación de diferentes estrategias de detección automática de fallas (Ramírez *et al*. , 2018). Las fallas corresponden a una condición subestándar no permitida en el desempeño o integridad de un equipo, llevando a un funcionamiento anómalo e imposibilitando el cumplimiento de su propósito. Las consecuencias de no detectar estas fallas a tiempo pueden envolver eventos de seguridad, accidentes catastróficos, pérdidas económicas, de impacto ambiental y de valor humano. Además, su detección temprana puede evitar alteraciones en la producción, pérdida de repuestos en *stock* y paradas no programadas, que se traducen en ahorros económicos. Todo lo anterior soporta que la detección de fallas en las plantas de procesos se considere un aspecto de importancia en la integridad y la seguridad industrial (Rivas *et al*., 2015).

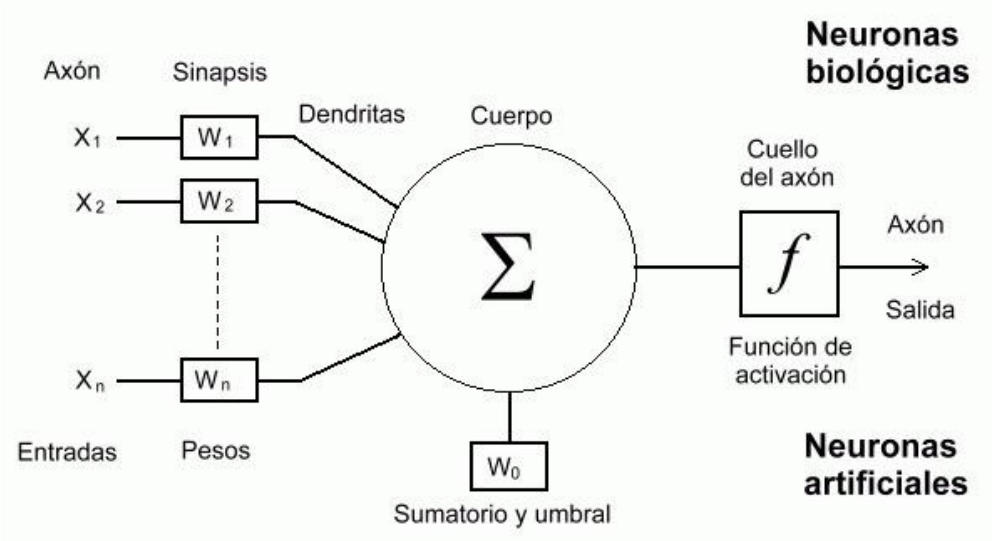
Las fallas pueden ocurrir por variaciones en el modo de operación, falta de calibración en los sensores, operación deficiente de los actuadores, fractura de los materiales o una alteración de la propia planta (Rivas *et al*. , 2015). Cuando un proceso es afectado por una falla, las variables del proceso reaccionan siguiendo trayectorias, llamadas secuencias temporales de datos, que pueden ser utilizadas para identificar el origen de la misma falla (Tarifa and Martinez 2007). Este origen, puede ser diagnosticado con algoritmos de fallas, que analizan el estado del proceso bajo supervisión, con el fin de determinar si la trayectoria temporal se desarrolla de manera normal. Por otra parte, el diagnóstico de fallas puede ser dividido en dos grandes grupos; el primero, denominado modelo explícito, corresponde a los que utilizan información procedente de

predicciones matemáticas del proceso; el segundo grupo se apoya en datos históricos, obtenidos de los sensores instalados para realizar supervisión y control del proceso (Ramírez *et al.* , 2018) (Tarifa and Martinez 2007). Este segundo grupo de diagnóstico presenta la ventaja de trabajar con datos que representan la operación diaria de los procesos, incluyendo los cambios que no son registrados en los documentos de seguimiento operacional. El análisis y la interpretación de los datos históricos pueden ser efectuadas por medio de los procedimientos de inteligencia artificial, como los enmarcados en las redes neuronales artificiales (RNA). Estas RNA son entrenadas para el análisis de la información histórica adquirida por sistemas de monitoreo, con el objetivo de detectar fallas y programar acciones preventivas (Ramírez *et al.* , 2018) (Hurtado-Cortés *et al.* , 2016).

La RNA corresponde a un modelo de procesamiento de señales, basado en la estructura neuronal del cerebro humano; la RNA aprende a través de ejemplos y aborda múltiples problemas con datos incompletos o en presencia de errores (Shanmuganathan 2016). Mediante datos históricos de referencia, las RNA pueden establecer relaciones entre las variables del sistema, con lo cual detectan las trayectorias temporales que indican la aparición de fallas (Esposito *et al.* , 2018) (Caudaudill 1989). Los componentes principales de una RNA son las entradas, los pesos, las funciones de activación y las salidas (Figura 1). En la Figura 1 se ilustra la arquitectura de una RNA de flujo de información unidireccional o *feedfoward,* comparando sus partes con una red neuronal biológica (Guemes 2018).

Figura 1

Esquema RNA frente a neuronas biológicas.



Nota: Tomado de Sancho. F, Redes Neuronales: una visión superficial. 2019.

La estructura funcional de las RNA corresponde a las llamadas neuronas o nodos. Estos nodos se encuentran unidos mediante señales, denominadas pesos, los cuales representan la intensidad entre los enlaces sinápticos de las neuronas, proporcionando la importancia relativa de cada variable de entrada sobre las variables de salida o variables predichas (Chung and Kusiak 1994). Los valores de entrada a la neurona son inicialmente tratados con una suma ponderada, antes de su transformación a través de una función de activación (Chung and Kusiak 1994). Cada neurona posee una función de activación, que le permite modificar el valor de salida, según si la cantidad de información entrante logra superar el umbral asignado. De otro lado, las neuronas que pertenecen a la capa de entrada y de salida son las encargadas de recibir y reportar, respectivamente, la información pertinente del sistema. Por otra parte, las neuronas presentes en las capas ocultas pertenecen al conjunto de niveles formados por neuronas que no tienen un

contacto directo con el entorno exterior; estas neuronas son las encargadas del procesamiento de la información. La combinación de dos o más neuronas forma una capa y una o más capas forman una red; generalmente, los elementos de una misma capa tienen la misma función de activación (Isasi and Galván 2004).

El aprendizaje de las RNA se define como el mecanismo que hace que los pesos tomen los valores necesarios para que desempeñen la tarea deseada. Uno de estos mecanismos es el algoritmo *backpropagation*, que calcula el error asociado al aporte de cada neurona en la última capa, empleando derivadas parciales. La corrección de los pesos en *backpropagation* se efectúa mediante el método de optimización de Levenberg-Marguard. Los pesos corregidos son propagados hacia la primera capa de la red, obteniendo así una relación entre la variable de salida y cada uno de los parámetros de las neuronas. Este algoritmo de aprendizaje es ejecutado hasta alcanzar una tolerancia especificada para las variables predichas (Chelgani *et al*. , 2017).

Una aplicación de las RNA en la detección de fallas operacionales corresponde al proceso Tennessee Eastman (TE). Este proceso TE no es un proceso real, sino que corresponde a una simulación creada por *Eastman Chemical Company*, con el fin de establecer una herramienta con características similares a los resultados de un proceso industrial. Con esta simulación TE es posible la evaluación de la aparición de fallas, así como la evaluación de diferentes estrategias de control en sistemas dinámicos (Chiang *et al*. , 2001). Para esto, el proceso TE contiene un módulo que permite programar la aparición de fallas en diversos tiempos de simulación (Marcos 2001). El proceso TE es no lineal, inestable en lazo abierto y presenta un número considerable de variables medidas y manipuladas (Antelo et al. 2007). De manera sucinta, el proceso TE está compuesto por

un reactor, un condensador, un separador de líquido/vapor, un compresor y una columna de destilación o *stripper*, y comprende ocho componentes, dos reacciones principales (R1 y R2) y dos reacciones secundarias (R3 y R4); las cuatro reacciones cuentan con características exotérmicas e irreversibles (Chiang *et al.* , 2001). Un diagrama representativo de este proceso es mostrado en el Apéndice A.

$$A(gas) + C(gas) + D(gas) \rightarrow G(líquido) \qquad producto \qquad \text{(R1)}$$

$$A(gas) + C(gas) + E(gas) \rightarrow H(líquido) \qquad producto \qquad \text{(R2)}$$

$$A(gas) + E(gas) \rightarrow F(líquido) \qquad subproducto \qquad \text{(R3)}$$

$$3D(gas) \rightarrow 2F(líquido) \qquad subproducto \qquad \text{(R4)}$$

El proceso TE contiene 41 variables medidas, 12 variables manipuladas, así como, 21 fallas, de las cuales 16 son conocidas y 5 desconocidas. Con la programación de las fallas, el proceso TE puede ser utilizado como caso de estudio para el entrenamiento de diferentes arquitecturas de RNA (Downs and Vogel 1993). Los entrenamientos y las validaciones de las RNA pueden ser desarrolladas por medio de los códigos de los programas de uso libre R y Python. Estos programas han sido producidos por Robert Gentleman y Ross Ihaka; y Guido Van Rossum, respectivamente. R es un lenguaje orientado al análisis estadístico, utilizado ampliamente en el ámbito de la ciencia de datos. Por su parte, Python es un lenguaje de alto nivel multipropósito utilizado además en otros campos como desarrollo web, scripting, etc. En términos generales, Python es un lenguaje más rápido en ejecución que R (UNIR 2020) .

En el presente documento se exponen los principales resultados de un análisis de la aplicación de las RNA, entrenadas y validadas con los datos obtenidos del proceso Tennessee

Eastman para la detección de fallas de proceso. Para esto, diversas arquitecturas de Redes Perceptrón Multicapa fueron entrenadas y validadas, mediante los códigos incluidos en los programas de uso libre R y Python, con la finalidad de establecer la arquitectura óptima de detección de fallas.

1. Estado del arte

Demetgul *et al.* (2011) desarrollaron un diagnóstico de fallas en una planta embotelladora usando RNA tipo *back-propagation*, definiendo 6 escenarios de falla, falta de botellas, falla en el cilindro no sujeta la tapa, el cilindro no aprieta la tapa, presión de aire insuficiente, presión de aire baja y falla en la dosificación de agua. Los autores optimizaron el error cuadrático medio de las RNA por medio de un algoritmo genético, considerando la variación en el número de capas y en los nodos ocultos. Asimismo, los autores diseñaron una función de ajuste para minimizar el tiempo de ejecución del modelo de RNA manteniendo el número de capas ocultas y nodos lo más bajo posible. Según los autores el estudio demostró la conveniencia, precisión y rapidez que presenta el uso de los algoritmos genéticos en la definición de la arquitectura de las RNA. También, la estructura de RNA 6:18:1 (seis variables de entrada, una capa interna con 18 neuronas y una variable predicha) condujo a los mejores resultados.

Zhang and Wang (2015) propusieron un modelo de RNA aplicado en sistemas de energía eólica para la detección de fallas en turbinas del rodamiento principal. La RNA consideró grandes cantidades de datos del sistema SCADA, previamente recopilados. El método calculó un valor

teórico de los parámetros de comportamiento anormal por comparación con datos reales del proceso. Según los autores la aplicación de la RNA condujo a la detección de fallas en la turbina, reduciendo costos de mantenimiento y operación. Asimismo, el modelo de RNA reportó alertas tempranas alertando las fallas con hasta 1.5 horas de anticipación.

Maradey (2017) estableció un algoritmo *off-line* para el diagnóstico de fallas en bombas hidráulicas de pistones axiales aplicando RNA; en estas bombas, las válvulas pueden presentar fallas por desgaste, conduciendo a la fuga de componentes. La red fue entrenada con 5 condiciones de fallas relacionadas con la perdida de eficiencia volumétrica de la bomba. Los autores consideraron diferentes arquitecturas de redes neuronales como la *Multiplayer Perceptron* (MLP de una y dos capas), la tipo Adaline y dos redes neuronales no lineales. Las anteriores redes fueron entrenadas y validadas mediante la toma de señales de vibración, presión, flujo volumétrico y procesamiento digital. Según los autores las redes MLP presentaron mejor desempeño en la detección, clasificación y diagnóstico de fallas, mientras que la red Adaline presentó el menor desempeño.

Chadha and Schwung (2017) presentaron una comparación entre dos arquitecturas de RNA, como lo son las redes de apilamiento profundo y los codificadores automáticos apilados dispersos, para la detección de fallas en el proceso Tennessee Eastman. Los autores desarrollaron una comparación entre las dos arquitecturas con diferentes parámetros, reportando que el modelo de codificadores automáticos apilados dispersos presentó una capacidad superior de detección de

fallas. Asimismo, los autores afirmaron que la anterior RNA resultó ser más estable, debido a que reportó una menor variación en la tasa de detección de fallos.

Adeli and Mazinan (2020) propusieron una RNA basada en datos difusos para la detección de fallas operacionales en el proceso Tennessee Eastman. En primer lugar, para la identificación y detección de las fallas consideraron teorías de preprocesamiento de datos, basándose en los datos disponibles y para proporcionar un bloque de decisión fiable se tuvo en cuenta un clasificador de fusión, para el cual los datos brutos, el tiempo y las características de frecuencia se dividen en varias herramientas de clasificación. Asimismo, los autores analizaron el enfoque del control de fallas basados en la regulación del controlador local en caso de que se genere una falla, reportando una efectividad en el enfoque propuesto.

Las anteriores referencias sugieren que la aplicación de las RNA ha mostrado efectividad en la detección y el diagnóstico de fallas. Particularmente, en el proceso TE, las RNA ensayadas han presentado un resultado favorable frente a otros métodos de detección de fallas. En este trabajo se entrenaron y validaron RNA simples de tipo *feedfoward* con los códigos disponibles en los programas R y Python, determinando su aplicación en la detección de fallas reportadas con el proceso TE.

2. Objetivos

2.1 Objetivo General

Determinar la estructura de RNA tipo *feedfoward* con mejor desempeño en la detección de fallas, según los códigos disponibles en los programas R y Python, a partir de datos históricos obtenidos mediante simulación con el proceso Tennessee Eastman.

2.2 Objetivos Específicos

Analizar estadísticamente datos históricos obtenidos mediante la programación de diferentes fallas en la simulación del proceso Tennessee Eastman, estableciendo una base de datos útil en el entrenamiento y validación de RNA.

Definir los desempeños en entrenamiento y en validación de diversas arquitecturas de RNA tipo *feedfoward* en la predicción de la aparición de fallas, utilizando los datos del proceso Tennessee Eastman generados por simulación y los códigos disponibles en los programas R y Python.

Evaluar las eficiencias de las arquitecturas de las RNA entrenadas, mediante un conjunto de datos de prueba del proceso Tennessee Eastman, seleccionando la RNA con menores errores de predicción.

3. Metodología

3.1 Fase I: Análisis de datos históricos del proceso Tennessee Eastman.

3.1.1 Revisión bibliográfica.

Se realizó una búsqueda detallada sobre el proceso Tennessee Eastman, con el fin de obtener información relevante de las variables y fallas que presenta el proceso. Asimismo, se estudiaron algunas estructuras de RNA aplicadas en la detección de fallas de procesos propuestas en diferentes textos.

3.1.2 Recolección de datos.

Los datos de ejecución del proceso TE fueron descargados de la base de datos de la Universidad de Harvard. La base de datos contenía información de las variables cada 3 minutos en un rango de 48 horas.

3.1.3 Reducción de datos.

Para el tratamiento de datos se removieron las columnas que contenía información irrelevante para la RNA, como lo es el número de la simulación y la muestra a la que corresponde, además se extrajeron múltiples datasets, para entrenamiento de la red y para pruebas de validación de los resultados

3.2 Fase II: Evaluación de desempeños de arquitecturas RNA tipo *feedfoward.*

3.2.1 Codificación de una estructura de RNA en el software R.

La librería neuralnet de R fue utilizada en el entrenamiento y la validación de las RNA tipo feedforward. Esta librería cuenta con un algoritmo que evita el sobreajuste de la RNA, denominado rprop+. Las funciones de activación usadas fueron sigmoide (logistic) y tangente hiperbólica (tanh); los datos fueron normaliza entre [0, 1] y [-1, 1]. La función de error definida fue

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(real_i - estimado_i)^2 \qquad Ec.1$$

3.2.2 Codificación de una estructura de RNA en el software Python.

La interfaz de programación Keras, especializada en Deep Learning, fue utilizada en el entrenamiento y la validación de RNA. Además, el optimizador de entrenamiento Adam fue utilizado en la mejora del proceso correspondiente. Los datos fueron normalizados, mientras que el parámetro de evaluación fue el MSE.

3.3 Fase III: Evaluación de las eficiencias de la RNA.

3.3.1 Validación de la RNA.

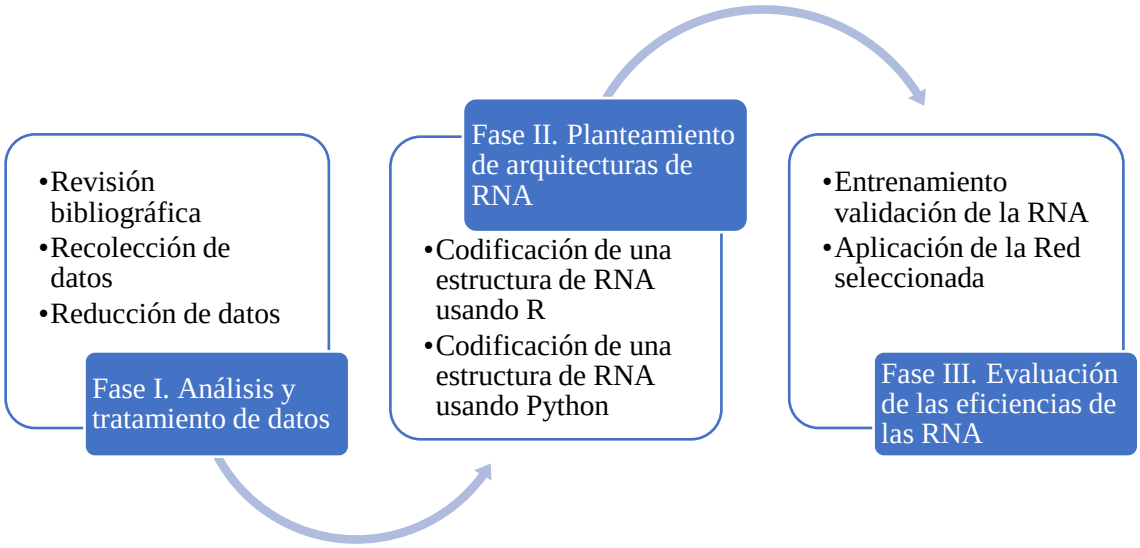
Una vez planteadas las posibles arquitecturas de la RNA, los códigos desarrollados en R y Python fueron ejecutados, considerando diferentes valores en los parámetros de las RNA; los parámetros considerados fueron la cantidad de neuronas, la función de activación, el número de capas ocultas y la tolerancia en la convergencia. La RNA con menor parámetro MSE, calculado con los MSE de validación y entrenamiento, fue seleccionada para la detección de fallas del proceso TE.

3.3.2 Aplicación de la Red seleccionada.

La arquitectura seleccionada de RNA fue puesta a prueba con nuevos datos, determinando su capacidad de predicción ante la aparición de fallas en el proceso.

Figura 2

Metodología aplicada.



4. Análisis de resultados

4.1 Datos del proceso Tennessee Eastman

Los datos utilizados en este trabajo fueron tomados del compendio generado por la simulación de proceso Tennessee Eastman (TE), que se encuentra en la base de datos de la universidad de Harvard (Rieth et al. 2017). El diagrama de proceso TE puede ser consultado en el Apéndice A. La Falla 21 no fue considerada debido a que las estadísticas de identificación de fallas presentaron un desempeño deficiente; esto a causa de que la válvula de flujo del alimento A/B/C en la corriente 4 se encontraba atascada, por lo cual las señales de esta válvula eran constantes. La simulación de Harvard presenta unos límites de operación normal en el reactor, el separador y la columna de destilación. Estos límites son presentados en la Tabla 1. De igual manera, la Tabla 1 presenta los límites que establecen una falla completa de la simulación. Además de los límites del proceso (Tabla 1), la simulación no debe reportar variaciones superiores al 5% molar en la composición de los productos; asimismo, la velocidad de producción no debe cambiar por encima del 5% del valor de referencia (Downs and Vogel 1993).

Tabla 1.

Intervalos de operación permitidos.

Variable	Límite de operación normales		Límites que provocan paro del sistema	
	Límite inferior	Límite superior	Límite inferior	Límite superior
Presión de reactor	Ninguno	2895 kPa	Ninguno	3000 kPa
Nivel de reactor	50% (10,65 m^3)	100% (21,3 m^3)	2,0 m^3	24 m^3
Temperatura de reactor	Ninguno	150 °C	Ninguno	175 °C
Nivel de separador	30% (3,0 m^3)	100% (9 m^3)	1,0 m^3	12,0 m^3
Nivel de columna	30% (1,98 m^3)	100% (6,6 m^3)	1,0 m^3	8,0 m^3

Nota: Adaptado de Downs and Vogel, 1993.

Por otra parte, los datos históricos refieren 12 variables manipulas (Tabla 6, Apéndice A) y 41 variables de salida (Tabla 7, Apéndice A). Las variables manipuladas corresponden al porcentaje de apertura de las válvulas para cada corriente, según se especifica en la Figura 3 . Además, la simulación de Harvard cuenta con 21 perturbaciones o fallas, de las cuales 16 son conocidas y 5 desconocidas (Tabla 2); estas fallas ocurren en el reactor, el condensador y en las zonas de alimentación y se encuentran asociadas a perturbaciones de tipo escalón en las variables de entrada respectivas, así como atascamiento en las válvulas correspondientes.

La base de datos de la simulación descargada contiene un total de 9850000 muestras y 52 variables. Esta base refiere 250000 muestras en operación normal y 480000 muestras con las respuestas de la simulación para la aparición de cada falla (Tabla 2). La Figura 4 ilustra la variación en la temperatura del reactor para la operación normal y para la aparición de algunas fallas contenidas en la base de datos. Según la Figura 4, la temperatura del reactor se ve afectada por las

Fallas 11 y 14, que son la temperatura del agua de enfriamiento a la entrada del reactor y la válvula de agua para enfriamiento del reactor respectivamente. Siendo la Falla 14 la que presenta una perturbación considerablemente mayor, por lo cual se puede concluir que la cantidad de agua de enfriamiento que entra al reactor tiene un efecto mayor en la temperatura de este.

Figura 3

Variables manipuladas en presencia de fallas.

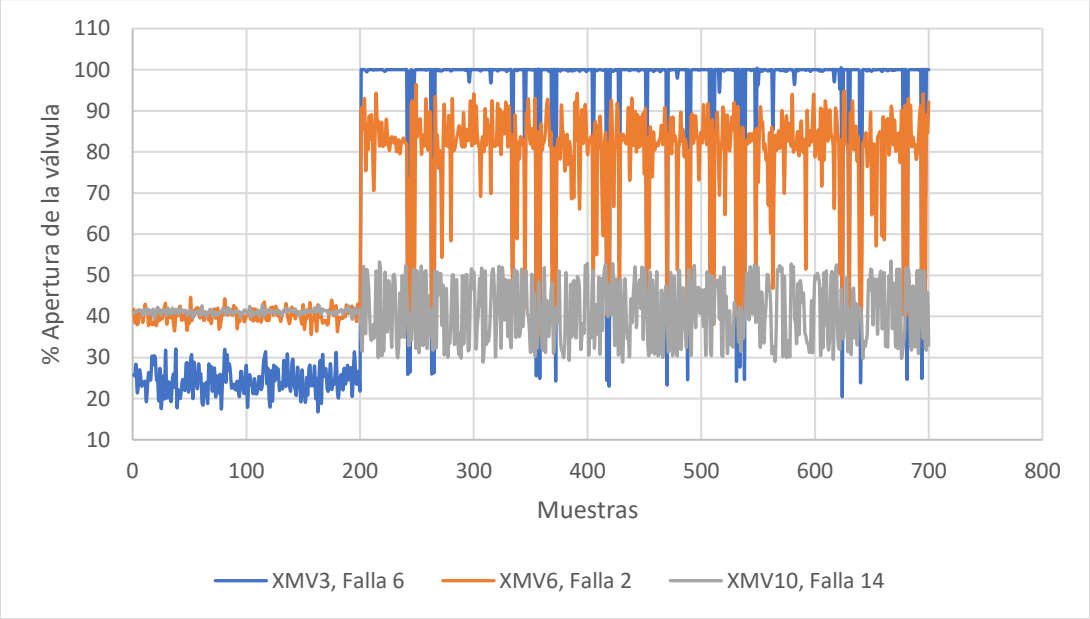


Tabla 2

Fallas simuladas en el proceso TE.

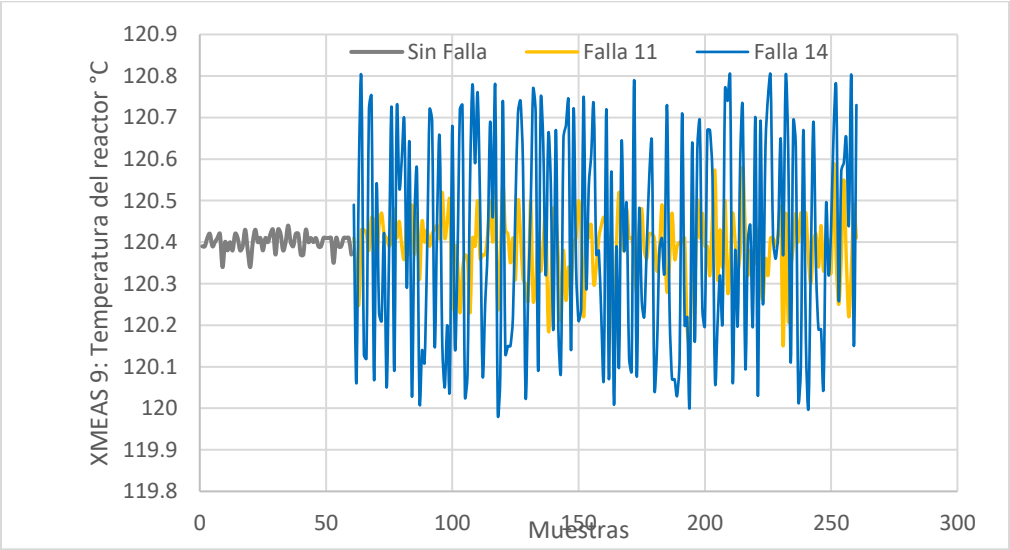
No. Variable (IDV)	Perturbación	Tipo
1	Proporción de alimentación A/C. Composición de B constante (corriente 4)	Escalón
2	Composición de B, proporción de A/C constante (corriente 4)	Escalón
3	Temperatura de alimentación D (corriente 2)	Escalón
4	Temperatura del agua de enfriamiento a la entrada del reactor	Escalón
5	Temperatura del agua de enfriamiento a la entrada del condensador	Escalón
6	Perdida de alimentación A (corriente 1)	Escalón
7	Perdida de presión en alimentación C, reducción de la disponibilidad de esta corriente (corriente 4)	Escalón
8	Composición de alimentación A, B, C (corriente 4)	Variación aleatoria
9	Temperatura de alimentación D (corriente 2)	Variación aleatoria
10	Temperatura de alimentación C (corriente 4)	Variación aleatoria
11	Temperatura del agua de enfriamiento a la entrada del reactor	Variación aleatoria
12	Temperatura del agua de enfriamiento a la entrada del condensador	Variación aleatoria
13	Cinética de reacción	Muy lenta
14	Válvula de agua para enfriamiento del reactor	Variación de apertura
15	Válvula de agua para enfriamiento del condensador	Variación de apertura
16	Desconocido	
17	Desconocido	
18	Desconocido	
19	Desconocido	
20	Desconocido	
21	La válvula para la corriente 4 se fijó en la posición de estado estable	Posición constante

Nota: Adaptado de Chiang, L, E Russell, and R Braatz, 2001.

Por otra parte, respecto al impacto de las fallas, en el momento en que la Falla 1 ocurre, se induce un cambio escalonado en el alimento de A/C de la corriente 4, generando un incremento en el alimento de C y una disminución en el alimento de A. Lo anterior reduce el alimento de A en la corriente de reciclo 5, con lo cual, el lazo de control actua para aumentar el alimento de A en la corriente 1 (Figura 5). Estos dos efectos se contrarrestan, generando una composición constante del alimento de A en la corriente 6. Estas variaciones en las tasas de flujo y en las composiciones de la corriente 6 causan fluctuaciones en el nivel del reactor, a lo cual el lazo de control en cascada actúa, manipulando la corriente 4 (Figura 6), por debajo de su valor de flujo en condiciones normales de operación. Por lo anterior, la Falla 1 ocasiona tanto cambios en composición, como cambios en la presión y el nivel. Con esto, la RNA podría detectar fácilmente la Falla 1, dado que más de la mitad de las variables de salida en la base de datos presentan una desvíación significativa de sus valores normales de operación.

Figura 4

Temperatura del reactor en presencia de fallas.



También, la aparición de la Falla 4 relaciona una disminución en la temperatura del agua de enfriamiento, lo cual produce un aumento repentino en la temperatura del reactor (Figura 7); los respectivos lazos de control regulan la variación en la temperatura. Debido al control, la desviación del valor de la temperatura del reactor con la Falla 4 resulta baja, por lo cual, su detección resulta mas complicada que con la detección de la Falla 1. De igual manera, la Falla 5 induce un cambio escalonado en la temperatura del agua de enfriamiento en el condensador, por lo cual se induce una alteración en el fujo del agua de enfiamiento del condensador (Figura 8). Cuando ocurre esta falla el flujo de la corriente de salida del condensador al separador aumenta, dando como resultado un aumento en la temperatura y el agua de enfriamiento de este (Figura 19, Apéndice B). Los lazos de control regulan los cambios en las temperaturas respectivas después de la aparición de esta falla, de forma similar que en la Falla 4.

Figura 5

Falla 1: Alimentación de A.

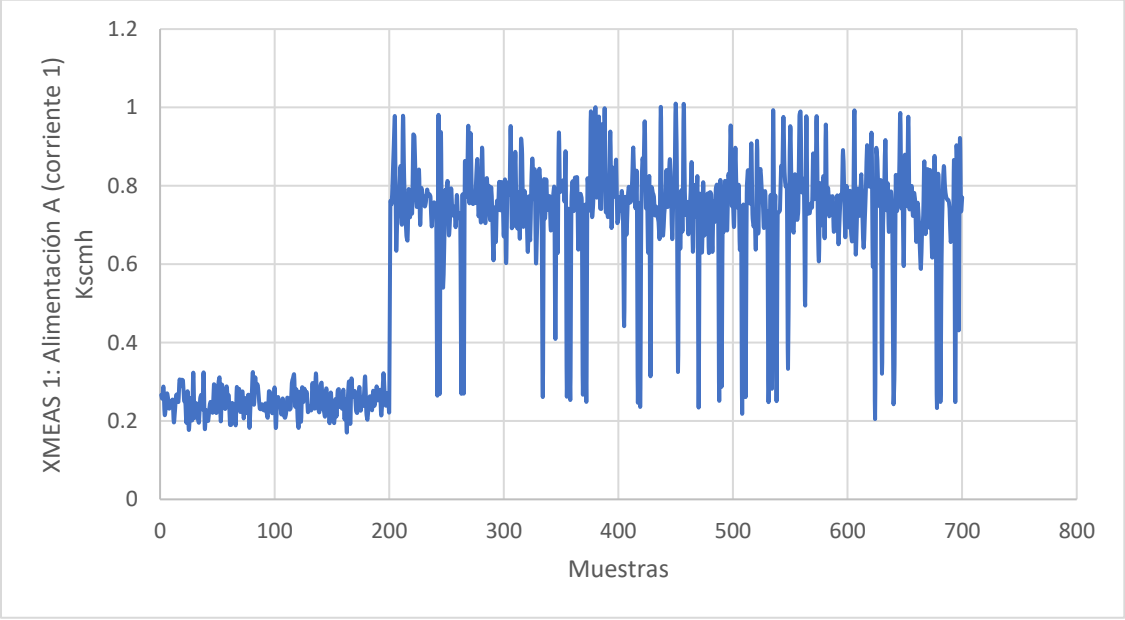


Figura 6

Falla 1: Alimentación A-C.

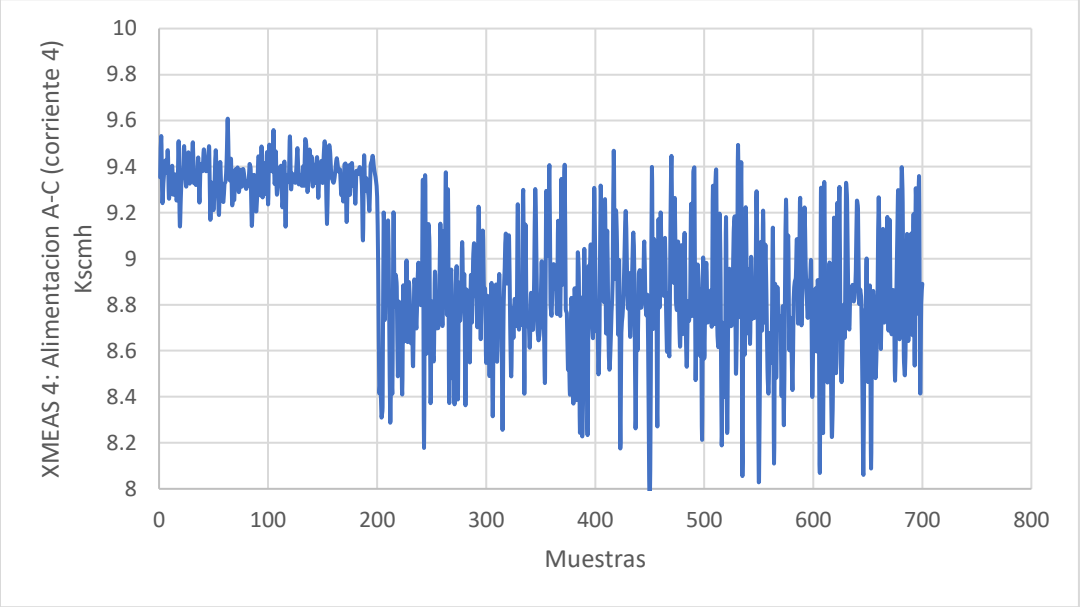


Figura 7

Falla 4: Temperatura del reactor.

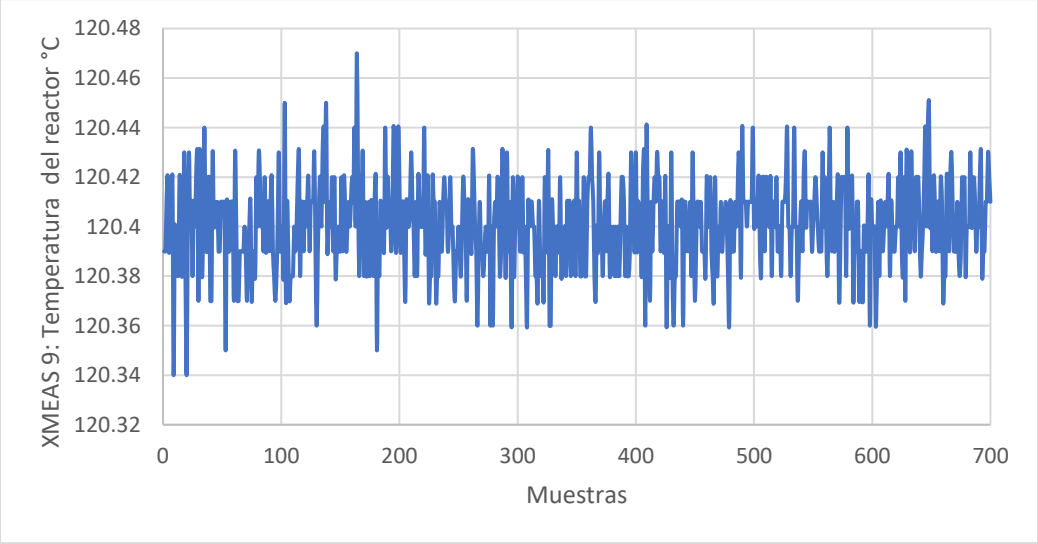
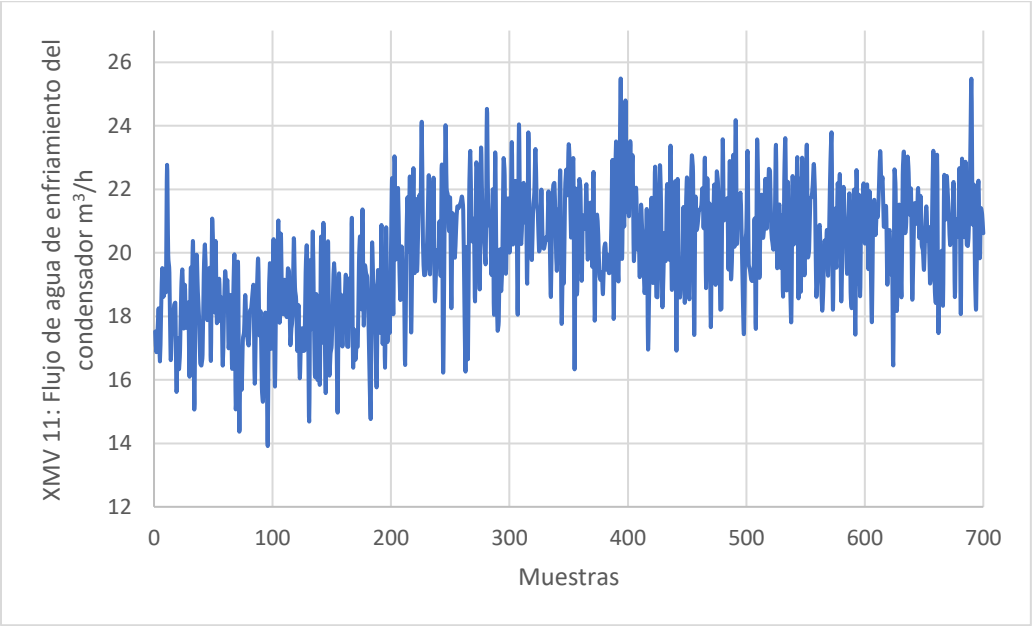


Figura 8

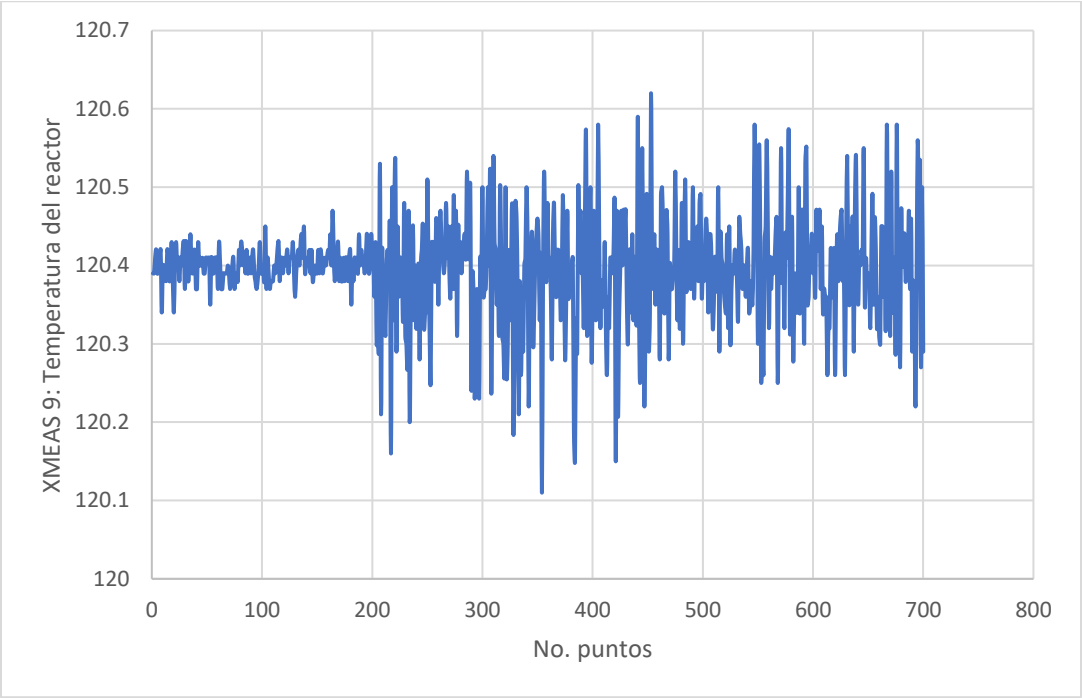
Falla 5: Flujo de agua de enfriamiento del condensador.



De otra parte, la Falla 11 genera grandes oscilaciones en la tasa de flujo del agua de enfriamiento, conduciendo a fluctuaciones en la temperatura del reactor con la entrada de agua de enfriamiento. La Figura 9 presenta la variación en la temperatrura del reactor después de la aparición de la Falla 11. Según la Figura 9, la temperatura del reactor oscila entre 120,1 y 120,6 °C cuando se presenta la falla, mientras en condiciones normales la temperatura fluctúan en 120,4°C; estos cambios no sobrepasan los limites de operación normal del proceso, por lo cual no implican la parada del sistema. Las otras 50 variables permanecen en los puntos de ajuste y se comportan acorde a las condiciones normales de operación. Como la Falla 11 y la Falla 4 afectan la misma variable del proceso, se espera que la Falla 11 influya mayormente en el flujo de agua de enfriamiento del reactor, debido a que la variación de la temperatura del reactor es considerablemente mayor en presencia de esta falla.

Figura 9

Falla 11: Temperatura del reactor.



De los anteriores párrafos, es posible mencionar que cada falla del proceso TE se encuentra asociada a un conjunto de variables de proceso, que pueden ser utilizadas en la identificación de las respectivas fallas, por medio de la aplicación de las RNA. La Tabla 3 exhibe las variables que presentan un vínculo entre la falla y una variable de observación. La Figura 10 representa el comportamiento que tiene la variable manipulada XMV6 (válvula de purga) frente a la Falla 2. Según esta figura, la apertura de la válvula de purga aumenta, debido a una mayor cantidad del inerte B al sistema. El comportamiento de las otras variables de la Tabla 3 se encuentra en el Apéndice B.

Tabla 3

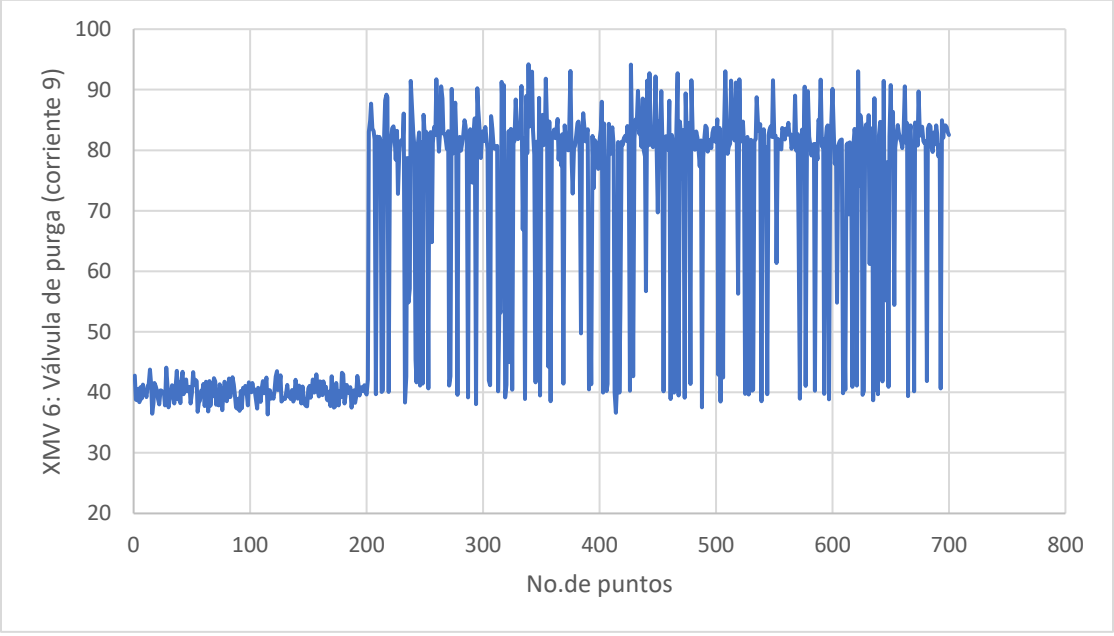
Variables relacionadas con cada perturbación.

Falla (IDV)	Variable del proceso	Descripción de la variable
2	XMV (6)	Válvula de purga (corriente 4)
4	XMV (10)	Flujo de agua de enfriamiento del reactor
5	XMEAS (22)	Temperatura del agua de enfriamiento a la entrada del reactor
6	XMV (3)	Flujo de alimentación A (corriente 1)
11	XMV (10)	Flujo de agua de enfriamiento del reactor
12	XMEAS (22)	Temperatura del agua de enfriamiento a la entrada del reactor
14	XMV (10)	Flujo de agua de enfriamiento del reactor
21	XMV (4)	Flujo de alimentación A-C (corriente 4)

Fuente: Chiang, L, E Russell, and R Braatz. 2001. *Fault Detection and Diagnosis in Industrial Systems*.

Figura 10

Falla 2: Válvula de purga.



4.2 Entrenamiento y validación con la función *neuralnet*

La base de datos de la simulación de la Universidad de Harvard fue utilizada en el entrenamiento y la validación de RNA tipo *feedforward*, por medio de los códigos contenidos en la función *neuralnet* del programa de uso libre R. Las arquitecturas RNA consideraron la simbología 52:*x*:1 logística; es decir, 52 neuronas de entrada, *x* neuronas en la capa interna (desde 1 hasta 104) y la predicción del sistema en operación normal o en falla, con 0 para operación normal y 1 con operación en falla (sin discriminar en el tipo de falla), utilizando la función de activación tipo logística. Debido a los recursos computacionales, la función *neuralnet* fue ejecutada con un dataset de 500 muestras y un threshold de 0,2. El código se encuentra en el Apéndice C.

La influencia del número de neuronas *x* en las predicciones con el conjunto de entrenamiento, el de prueba y el total, son presentados en las Figuras 11 y 12, en términos del coeficiente de determinación y del error cuadrático medio (MSE), respectivamente. Según estas figuras, el aumento de neuronas en la capa oculta conduce a una mejora en el entrenamiento de las RNA; alrededor de 38 neuronas son necesarias para la obtención del menor valor de MSE con el conjunto de entrenamiento en la detección de la operación del proceso en falla (Figura 12). Sin embargo, la validación de la RNA con los datos del conjunto de prueba muestra una disminución en el desempeño (disminución de R2 y aumento de MSE) con el aumento en el número de neuronas. Lo anterior conduce a un desempeño total pobre de las RNA entrenadas con la función *neuralnet*. Este desempeño pobre sugiere que las RNA experimentan sobreajuste en la predicción de la condición de falla; *i.e.* las RNA están especializadas en la ejecución solo con los datos de

entrenamiento. Con esto, no es posible la determinación de un número de neuronas para un óptimo desempeño de la RNA.

Figura 11

Coeficiente de determinación 500datos logistic threshold 0,2.

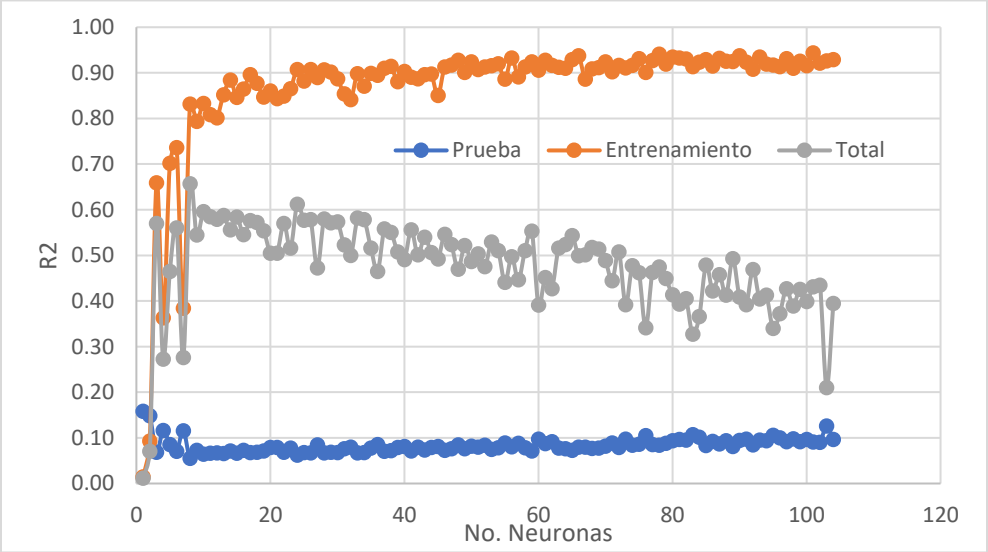
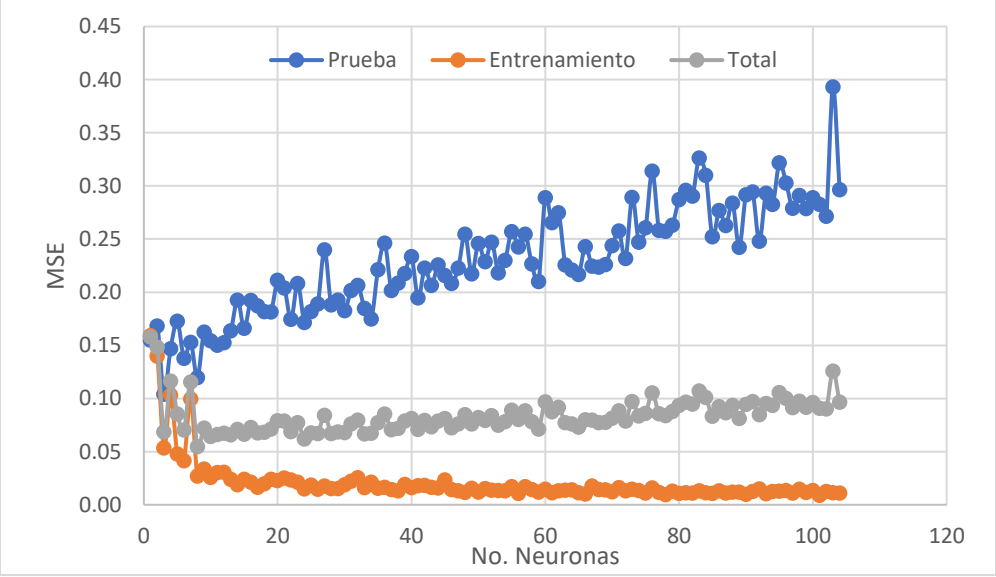


Figura 12

MSE 500datos logistic threshold0,2.



Pruebas adicionales con *neuralnet* fueron efectuadas considerando diferentes opciones de ejecución. El Apéndice D presenta los resultados de la ejecución de *neuralnet* con variaciones en el número de datos de entrada, en el *threshold* y en la función de activación de las neuronas. Ninguno de los resultados obtenidos con la función *neuralnet* condujo a la reducción del sobreajuste en las predicciones con el conjunto de prueba. Debido a lo anterior, la función *neuralnet* fue descartada para su aplicación en la predicción del tipo de falla en particular (Tabla 2).

4.3 Entrenamiento y validación con el paquete *Keras*

El código de Python con el editor de código *Google Colab* y las librerías *Pandas, Keras* y *scikitlearn* fueron utilizadas en el entrenamiento de RNA tipo *feedforward* para la predicción de la condición de falla en los históricos del proceso TE. La capacidad de información que puede gestionar *Google Colab* permitió la ampliación del conjunto de entrenamiento a 83000 muestras, de las cuales 80000 presentaron fallas (4000 por cada una de las 20 fallas); los 3000 restantes presentaban operación normal. Además de los datos de entrenamiento, se tomaron en cuenta 2 *datasets* adicionales, para prueba y validación del entrenamiento de la RNA, con 4800 muestras cada uno (800 en operación normal y 200 por cada falla). La Figura 13 presenta la variación del MSE Total versus el número de neuronas de la capa interna para la predicción de la condición de falla (sin discriminar en el tipo de falla); la función de activación logística fue aplicada en el entrenamiento. Según esta figura, la cantidad que presenta el menor error corresponde a 11 neuronas en la capa oculta, teniendo como estructura 52:11:1. Por su parte, la Figura 14 muestra los resultados del entrenamiento y la validación de la red 52:11:1, según el número de iteraciones

definidas en el código. Según esta figura, las 50 iteraciones definidas en cada cálculo de RNA son suficientes para alcanzar una respuesta consistente.

Figura 13

MSE total con 87800 datos y la función sigmoide 52:x:1.

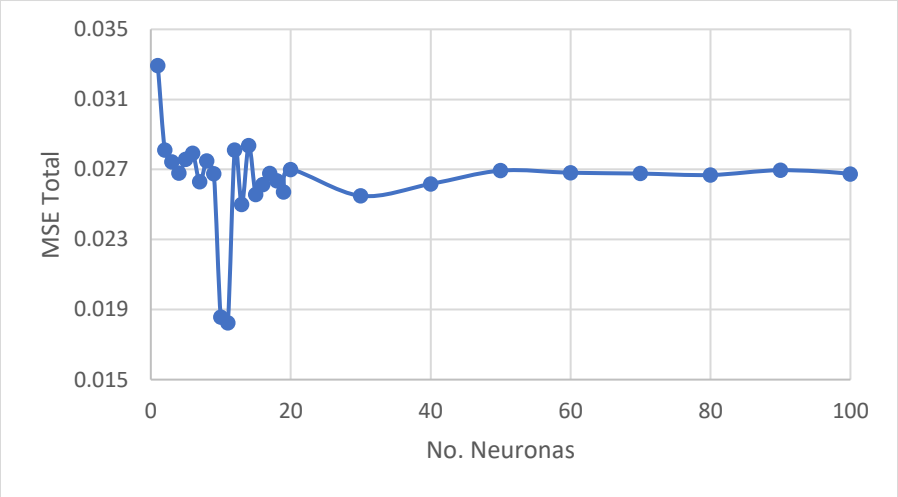
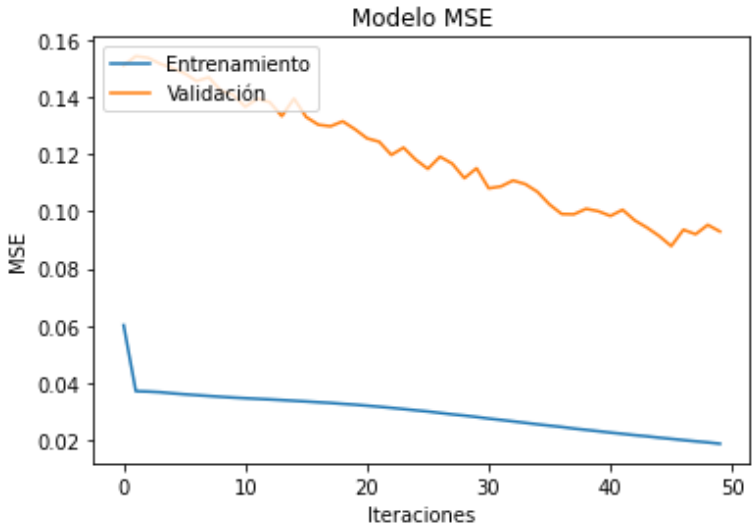


Figura 14

Iteraciones en MSE de entrenamiento y validación con la red 52:11:1.



Por otro lado, un incremento en el número de las capas ocultas de las RNA no favorece la predicción obtenida. La Tabla 4 presenta los resultados de la adición de una segunda capa oculta, considerando 11 neuronas en la primera capa oculta, además se presenta el código en el Apéndice E. Los valores presentados en la Tabla 4 sugieren que la red 52:11:1 logística entrenadas con las mencionadas librerías de Python, conducen a la predicción de la condición de falla en el proceso TE. Del valor de MSE para esta RNA es posible mencionar que la predicción se encuentra en 0±0,27 o 1±0,27 para las condiciones de operación normal o de operación en falla, respectivamente. Los intervalos de confianza se encuentran lo suficientemente lejanos, con lo cual los resultados de la RNA no generan confusión entre operación normal y operación en falla.

Tabla 4.

Verificación del número de capas ocultas usando el MSE total.

No. Capas	No. Neuronas		MSE Total
2	11	10	0,03048013
2	11	20	0,02927603
2	11	30	0,0289162
2	11	40	0,03240583
2	11	50	0,02502678

Con el soporte de los resultados obtenidos en la predicción de la condición en falla, el código desarrollado en Python fue aplicado en el entrenamiento y la validación de diferentes estructuras de RNA, con 21 neuronas en la capa de salida. Estas 21 salidas representan las condiciones de operación normal y las 20 fallas consideradas para el proceso TE; cada neurona de salida tiene asignada la detección de una falla (0 en condición normal y 1 en condición de falla). La Figura 15 presenta el MSE total resultante del entrenamiento de esta RNA variando el número de neuronas de la capa oculta. Según esta figura, el mejor desempeño se obtiene con el uso de 80

neuronas en la capa oculta (RNA 52:80:21). De igual manera, la Tabla 5 presenta los mejores desempeños considerando un mayor número de capas. Los resultados mostrados en esta tabla corresponden a los menores valores de MSE total obtenidos por los cálculos con las RNA. Según la Tabla 5, la adición de nuevas capas a la estructura de RNA contribuye poco a la mejora del MSE total; la mejor red corresponde a la 52:80:50:30:21 logística. La Figura 16 exhibe los valores de MSE totales del conjunto de prueba, reportados por la RNA 52:80:50:30:21. Según esta figura, la predicción obtenida con la RNA 52:80:50:30:21, sobre un conjunto externo al entrenamiento con 400 datos por cada falla, presenta bajo desempeño en el pronóstico de la aparición de las diferentes fallas; los valores de MSE obtenidos con esta RNA conduce a imprecisiones en la aparición de cada una de las fallas en el proceso TE. Con lo anterior, un tipo de RNA más compleja que la *feedforward* es recomendable para su aplicación en la predicción de fallas en el mencionado proceso (Adeli and Mazinan 2020)(Chadha and Schwung 2017). Por otra parte, el código en el Apéndice F presenta las instrucciones codificadas en Python para esta sección. Las estructuras de las 3 redes con mejor rendimiento se encuentran en el Apéndice G.

Figura 15

MSE total con 87800 datos y la función sigmoide 52:X:21.

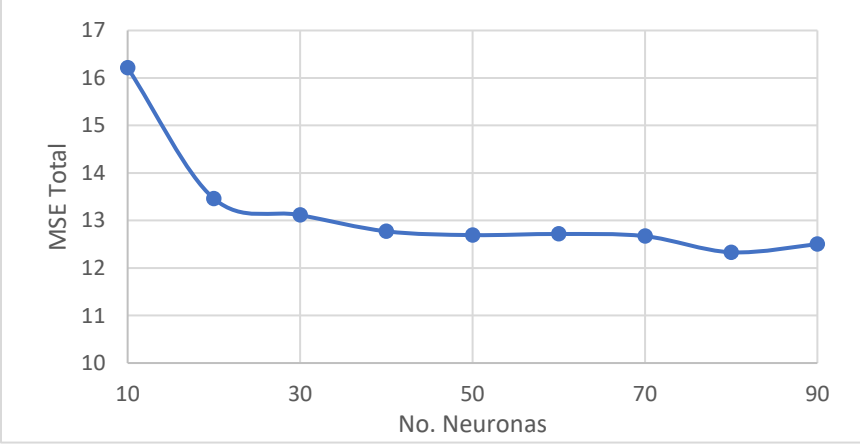


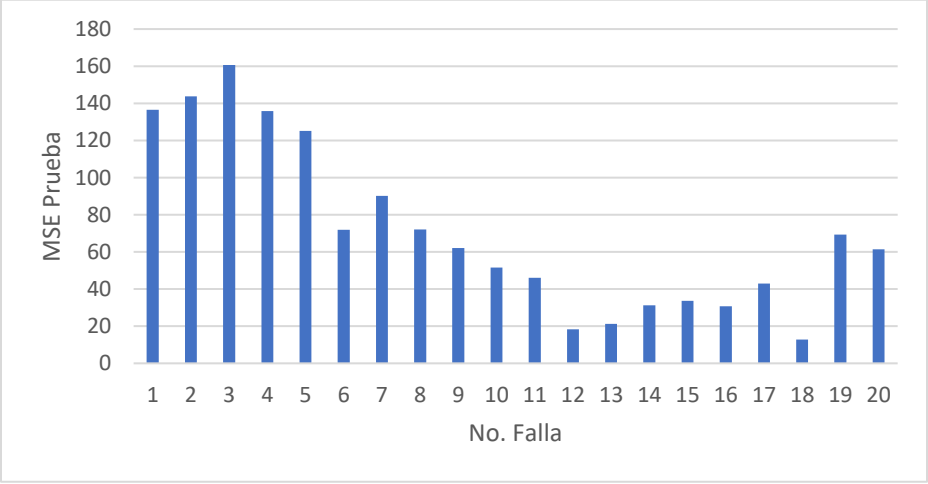
Tabla 5

Verificación del número de capas ocultas usando el MSE total.

No. Capas Ocultas	No. Neuronas			MSE Total
1	80	-	-	12,33051641
2	80	20	-	11,56086976
2	80	30	-	12,29056652
2	80	40	-	11,68425421
2	80	50	-	11,12743931
3	80	50	20	10,07496695
3	80	50	30	9,340096544
3	80	50	40	9,457437581

Figura 16

Resultados del MSE de Prueba ante las fallas RNA 52:80:50:30:21.



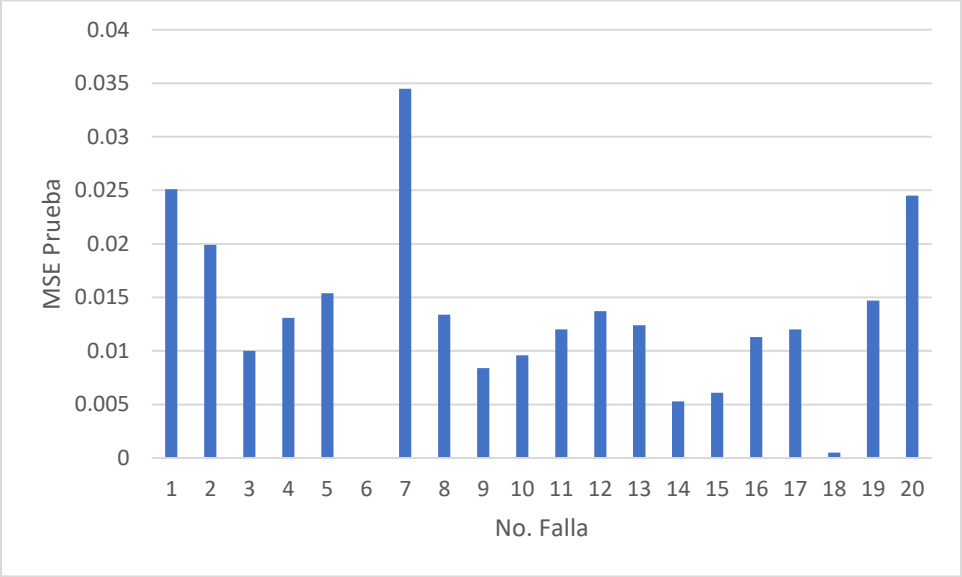
4.4 Evaluación y aplicación de la Red Seleccionada

Con base a los resultados obtenidos del MSE total de cada estructura de red propuesta, se seleccionó la RNA 52:11:1 logística desarrollada en el código Python, en la detección de las fallas ocurridas en el proceso TE. Esta RNA fue probada adicionalmente utilizando 400 muestras por

cada falla con el fin de evaluar la detección ante las mismas. La Figura 17 ilustra los resultados de la aplicación de la RNA 52:11:1 en la predicción de cada falla por separado. Según la Figura 17, el MSE obtenido en la detección de cada falla es menor a 0,035, lo que indica que el entrenamiento de la RNA resultó efectivo, ya que permite la identificación de la aparición de fallas en un nuevo conjunto de datos; las mejores predicciones se presentaron con la Falla 6 y la Falla 18 (MSE de 5,64E-09 y 5,06E-04, respectivamente). Estas fallas corresponden a la perdida de alimentación de A en la corriente 1 y a una falla desconocida respectivamente.

Figura 17

Resultados del MSE de Prueba ante las fallas RNA 52:11:1.



Finalmente se utilizó la RNA 52:11:1 con el dataset externo completo de 4000 muestras en la predicción de la aparición de las diferentes fallas. Según los resultados de predicción con esta RNA, la aparición de la respectiva falla en 3871 muestras fue predicha de manera satisfactoria, con lo cual, la RNA mostró una eficiencia de predicción del 97%; los otros 129 datos se les conoce como “falsos negativos”.

5. Conclusiones

Del análisis de los datos históricos del proceso Tennessee Eastman se concluyó que se presentan 12 variables medidas, 41 variables manipuladas y 21 fallas; la recolección de datos permitió realizar una comparación de las variables más significativas en el proceso ante la presencia de fallas, como lo es la temperatura del reactor la cual presenta cambios en su comportamiento ante la aparición de las Fallas 4, 11 y 14.

La estructura de RNA codificada en el software R con el paquete *neuralnet* resultó tener un bajo desempeño en el entrenamiento de la RNA, debido a que los datos suministrados para entrenamiento no fueron suficientes, por lo que el modelo planteado no logró detectar correctamente las fallas, además el código no arrojó una lectura clara del MSE para establecer una estructura de Red.

El código de entrenamiento desarrollado en Python permitió analizar el desempeño de la detección de fallas, dando como resultado que la estructura de la RNA con menor error es 52:11:1 con un MSE total de 0,01823, esta estructura se puso a prueba ante nuevas fallas obteniendo una eficiencia en la predicción del 97%. Adicionalmente se determinó que el incremento del número de capas ocultas no favorece el error obtenido.

La estructura 52:80:50:30:21 desarrollada en Python presentó un MSE total de 9,34, esta estructura permite detectar el tipo de falla que irrumpe en el sistema, no obstante, en presencia de

nuevas fallas se determinó que la estructura no conduce a una detección satisfactoria del tipo de fallas, puesto que el MSE de Prueba arroja valores muy altos.

La evaluación de la arquitectura 52:11:1 en la predicción de la aparición de fallas, utilizando un nuevo conjunto de muestras, reportó un MSE menor 0,035. Este valor bajo de MSE verifica la eficiencia del entrenamiento de la RNA 52:11:1 en la detección de fallas, en un conjunto de muestras de operación del proceso Tennessee Eastman.

6. Recomendaciones

Realizar un pretratamiento de datos más detallado con los datos de entrenamiento, específicamente el uso de la herramienta *Principal Component Analysis* (PCA) para sistemas de *Machine Learning*, la cual reduce las dimensiones de los datos seleccionando exclusivamente los componentes principales, con el fin de disminuir el ruido que aporta cada variable que no tenga relación directa con cada una de las fallas, viéndose reflejada en la eficiencia y velocidad de aprendizaje de la RNA.

En caso de disponer de un equipo de cómputo de alto rendimiento, evaluar la arquitectura de redes en el software R con mayor cantidad de datos y parámetros más exigentes.

Referencias Bibliográficas

Adeli, M, and A Mazinan. 2020. “High Efficiency Fault - Detection and Fault - Tolerant Control Approach in Tennessee Eastman Process via Fuzzy - Based Neural Network Representation.” *Complex & Intelligent Systems* 6(1): 199–212. https://doi.org/10.1007/s40747-019-0094-3.

Antelo, L et al. 2007. “La Teoría de Redes En Ingeniería de Control Aplicación Al Análisis Dinámico y Al Control de Procesos.” *Revista Iberoameriana de Automática e Informatica industrial* 4: 24–34.

Caudaudill, M. 1989. “Neural Network Primer: Part L.” *Al Expert*.

Chadha, G, and A Schwung. 2017. “Comparison of Deep Neural Network Architectures for Fault Detection in Tennessee Eastman Process.” *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*: 1–8.

Chelgani, S, B Shahbazi, and E Hadavandi. 2017. “Support Vector Regression Modeling of Coal Flotation Based on Variable Importance Measurements by Mutual Information Method.” *Measurement*. http://dx.doi.org/10.1016/j.measurement.2017.09.025.

Chiang, L, E Russell, and R Braatz. 2001. *Fault Detection and Diagnosis in Industrial Systems*.

Chung, Y, and A Kusiak. 1994. “Grouping Parts with a Neural Network.” *Journal of Manufacturing Systems* 13(4): 262–75.

Demetgul, M, M Unal, I Tansel, and O Yaziciolu. 2011. “Fault Diagnosis on Bottle Filling Plant Using Genetic-Based Neural Network.” *Advances in Engineering Software* 42(12): 1051–58.

Downs, J, and E Vogel. 1993. “A PLANT-WIDE INDUSTRIAL PROBLEM PROCESS.” *Pergamon Press Ltd* 17(3): 245–55.

Esposito, A, M Faudez-Zanuy, F Morabito, and E Pasero. 2018. 69 *Multidisciplinary Approaches*

to Neural Computing. http://link.springer.com/10.1007/978-3-319-56904-8.

Guemes, E. 2018. “Diseño de Sensores Software Para El Control de Calidad de Un Proceso.” EII Universidad de Valladolid.

Hurtado-Cortés, L, E Villarreal-López, and L Villarreal-López. 2016. “Fault Detection and Diagnosis through Artificial Intelligence Techniques , a State of Art.” *DYNA* 83: 19–28.

Isasi, P, and I Galván. 2004. *Redes Neuronales Artificiales Un Enfoque Práctico*.

Maradey, J. 2017. “DIAGNÓSTICO DE FALLAS EN BOMBAS HIDRÁULICAS DE PISTONES AXIALES, APLICANDO REDES NEURONALES ARTIFICIALES (ANNS).” universidad industrial de santander.

Marcos, A. 2001. “Diseño de Uns Estructura de Control Para Plantas Químicas.” *ETSII-UPM*.

Ramírez, J, H Sarmiento, and J López-Lezama. 2018. “Diagnóstico de Fallas En Procesos Industriales Mediante Inteligencia Artificial.” *Espacios* 39(24): 12. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85048623147&partnerID=40&md5=58bc326754f3bac152349447ed18ff63.

Rieth, C, B Amsel, R Tran, and M Cook. 2017. “Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation.” *Harvard Dataverse*: V1.

Rivas, R, P Zubieta, and H Garcini. 2015. “Detección y Diagnóstico Automático de Fallos En Procesos Industriales.” *Ingenieria Electrónica, Automática y Cominucaciones* XXIII(3).

Shanmuganathan, Subana. 2016. 628 Studies in Computational Intelligence *Artificial Neural Network Modelling: An Introduction*.

Tarifa, E, and S Martinez. 2007. “Diagnóstico de Fallas Con Redes Neuronales . Parte 1 : Reconocimiento de Trayectorias.” *Revista de ingeniería e investigación* 27(1): 68–76.

UNIR. 2020. “R vs Python: ¿cuál Es Mejor Para El Análisis de Datos?”

https://www.unir.net/ingenieria/revista/r-vs-python/.

Zhang, Z, and K Wang. 2015. “Wind Turbine Fault Detection Based on SCADA Data Analysis Using ANN.” *Advances in Manufacturing* 2(1): 70–78.

Apéndice A

El proceso Tennessee Eastman fue creado por la compañía de químicos Eastman con el fin de suministrar datos claves para la evaluación de diferentes sistemas de control y monitoreo de fallas. Dicho proceso consta de cinco unidades principales: un reactor de dos fases, un separador, un stripper , un compresor y un condensador, y procesa 8 tipos de componentes, en 4 reacciones las cuales son exotérmicas, irreversibles y su cinética se ajusta a primer orden.

$$A(gas) + C(gas) + D(gas) \rightarrow G(liquido) \qquad producto$$

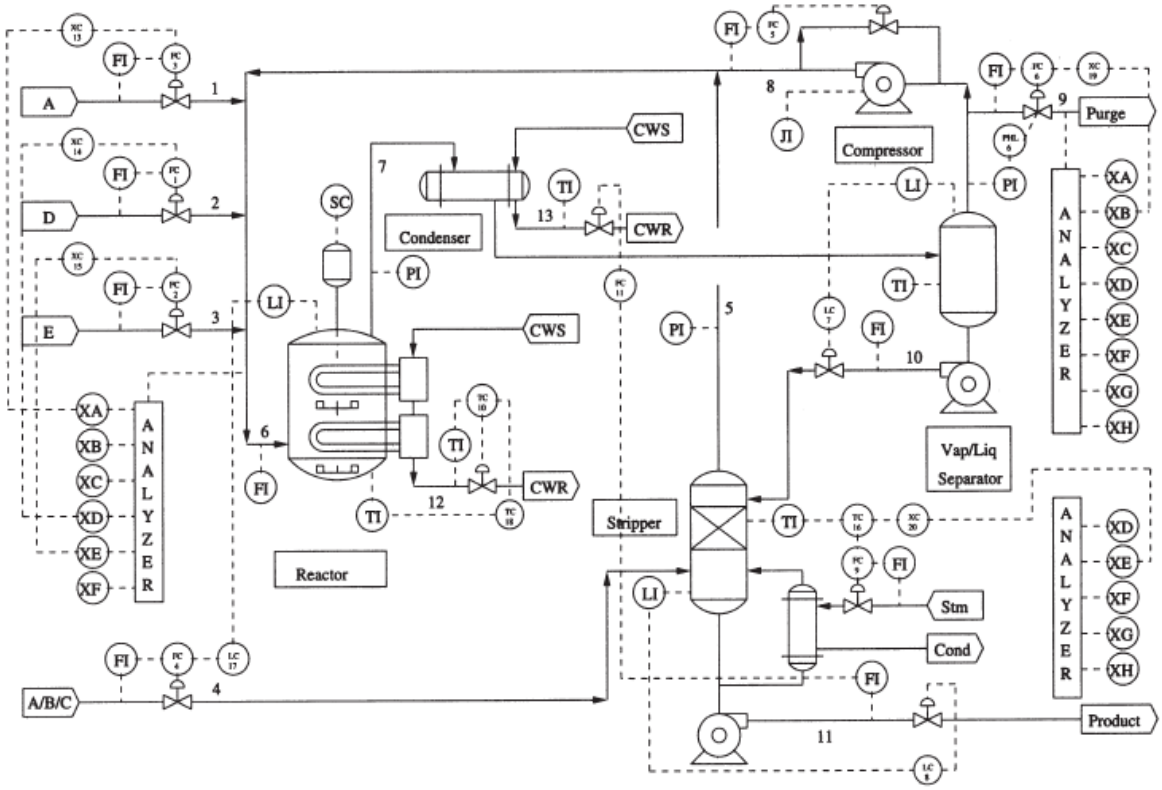
$$A(gas) + C(gas) + E(gas) \rightarrow H(liquido) \qquad producto$$

$$A(gas) + E(gas) \rightarrow F(liquido) \qquad subproducto$$

$$3D(gas \rightarrow 2F(liquido) \qquad subproducto$$

Figura 18

Proceso Tennessee Eastman.



Nota: Marcos, A. 2001. “Diseño de Uns Estructura de Control Para Plantas Químicas.” *ETSII-UPM*.

En la corriente de alimentación existe una pequeña cantidad de inerte B el cual no es condensable, por lo tanto, este componente debe purgarse para evitar acumulaciones en el sistema. En la Figura 18 se muestra un esquema del proceso Tennessee Eastman.

En primer lugar las corrientes de alimentación de los reactivos gaseosos A, D, E y una corriente recirculada del separador de líquido/vapor se alimentan al fondo reactor, el reactor contiene líquido y vapor pero no hay salida de líquido, el vapor producto de las reacciones va hacia el condensador parcial y posteriormente al separador, el líquido del separador se alimenta al plato

superior de la columna de scripting o columna de destilación y el vapor que sale del separador se comprime, una pequeña fracción de este se purga para evitar la acumulación del inerte y de los productos secundarios y el restante se recircula al reactor mediante el compresor. En el stripper existen dos fuentes principales de vapor, en primer lugar, un hervidor y en segundo lugar la alimentación del reactivo C. dentro del reactor se presenta una interacción entre la temperatura, presión y el nivel del líquido, además de presentar un comportamiento no lineal, El producto el cual está compuesto por G y H es obtenido en la base de la columna de destilación (stripper). Además, este proceso presenta 12 variables medidas (tabla 6) y 41 variables manipuladas o variables de salida (tabla 7)

Tabla 6

Variables manipuladas.

No. Variable (XMV)	Variable manipulada
1	Flujo de alimentación de D (corriente 2)
2	Flujo de alimentación de E (corriente 3)
3	Flujo de alimentación A (corriente 1)
4	Flujo de alimentación A-C (corriente 4)
5	Válvula de reciclado del compresor
6	Válvula de purga (corriente 9)
7	Flujo de salida del separador (corriente 10)
8	Flujo de salida de la columna (corriente 11)
9	Válvula de vapor a la columna
10	Flujo de agua de enfriamiento del reactor
11	Flujo de agua de enfriamiento del condensador
12	velocidad del agitador

Nota: Adaptado de Downs and Vogel, 1993.

Tabla 7

Variables de salida.

No. Variable (XMEAS)	Medida del proceso	Unidades
1	Alimentación A (corriente 1)	Kscmh
2	Alimentación D (corriente 2)	Kg/h
3	Alimentación E (corriente 3)	Kg/h
4	Alimentación A-C (corriente 4)	Kscmh
5	Flujo de reciclado (corriente 8)	Kscmh
6	Velocidad de alimentación al reactor (corriente 6)	Kscmh
7	Presión del reactor	Kpa
8	Nivel del reactor	%
9	Temperatura del reactor	°C
10	Velocidad de corriente de la purga (corriente 9)	Kscmh
11	Temperatura del separador	°C
12	Nivel del separador	%
13	Presión del separador	Kpa
14	Flujo inferior del separador (corriente 10)	m^3/h
15	Nivel de la columna	%
16	Presión de la columna	Kpa
17	Flujo inferior de la columna (corriente 11)	m^3/h
18	Temperatura de la columna	°C
19	Flujo de vapor a la columna	Kg/h
20	Trabajo del compresor	kW
21	Temperatura del agua de enfriamiento a la entrada del reactor	°C
22	Temperatura del agua de enfriamiento a la entrada del condensador	°C
	Análisis de la corriente de alimentación al reactor (corriente 6)	
23	Componente A	% mol
24	Componente B	% mol
25	Componente C	% mol
26	Componente D	% mol
27	Componente E	% mol
28	Componente F	% mol
	Análisis de la corriente de purga (corriente 9)	
29	Componente A	% mol
30	Componente B	% mol
31	Componente C	% mol
32	Componente D	% mol
33	Componente E	% mol

34	Componente F	% mol
35	componente G	% mol
36	Componente H	% mol
	Análisis del producto (corriente 11)	
37	Componente D	% mol
38	Componente E	% mol
39	Componente F	% mol
40	Componente G	% mol
41	Componente H	% mol

Nota: Adaptado de Downs and Vogel, 1993.

Apéndice B

Comportamiento de algunas variables frente a la presencia de fallas

Figura 19

Falla 4: Flujo de agua de enfriamiento del reactor.

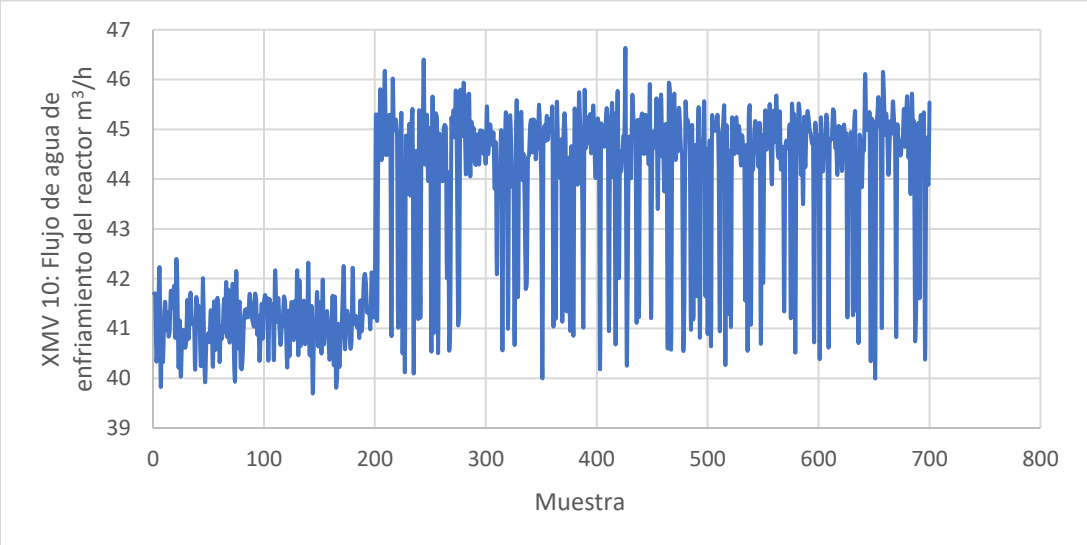


Figura 20

Falla 5: Temp. agua de enfriamiento a la entrada del condensador.

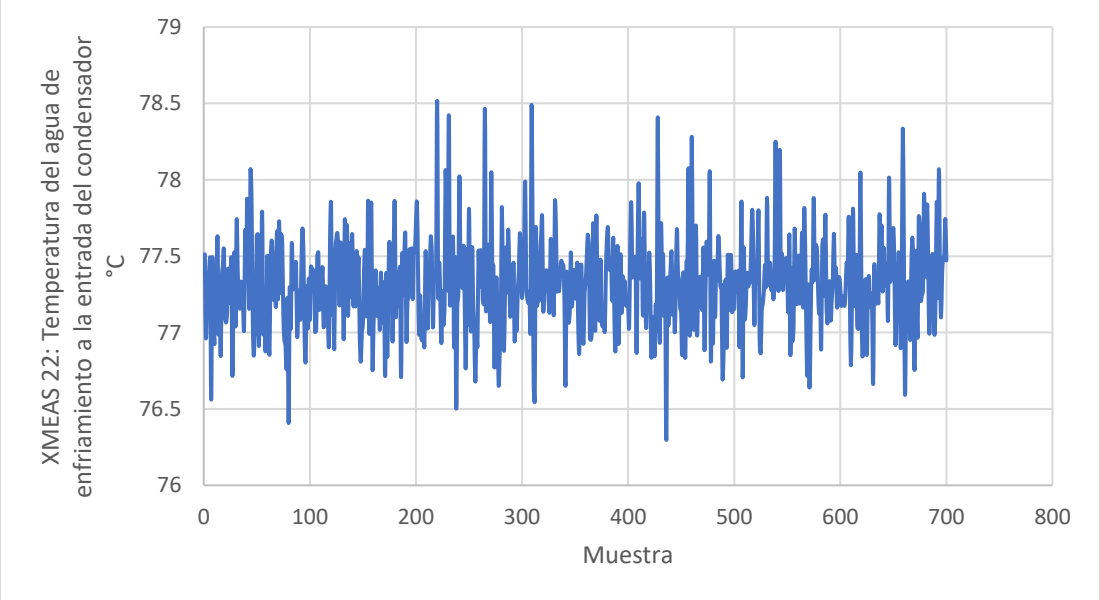


Figura 21

Falla 6: Flujo de alimentación de A.

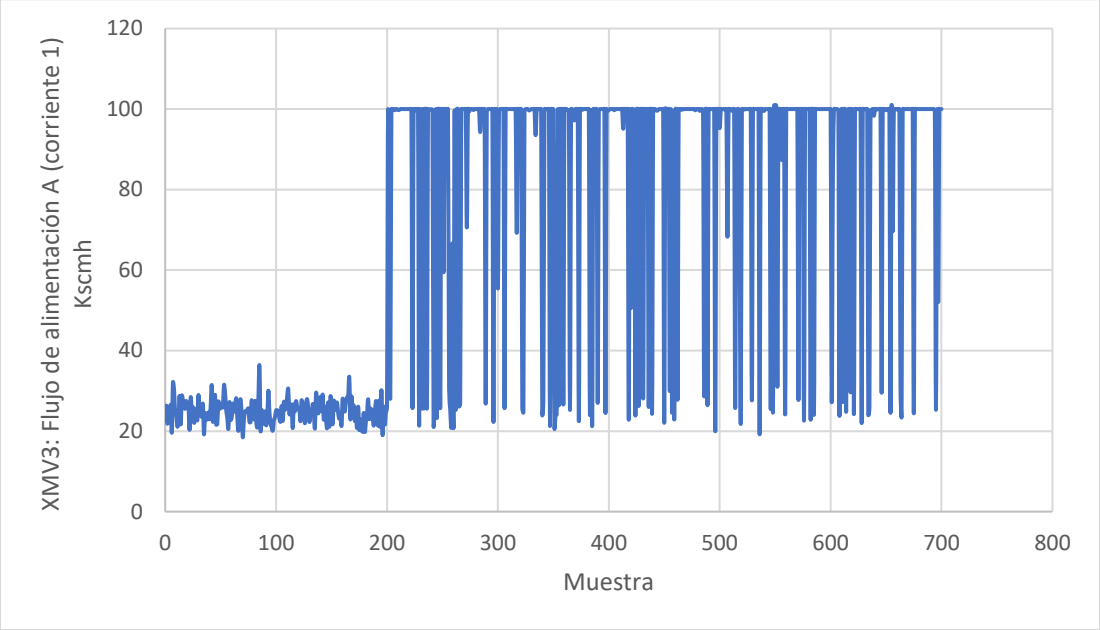


Figura 22

Falla 11: Flujo de agua de enfriamiento del reactor.

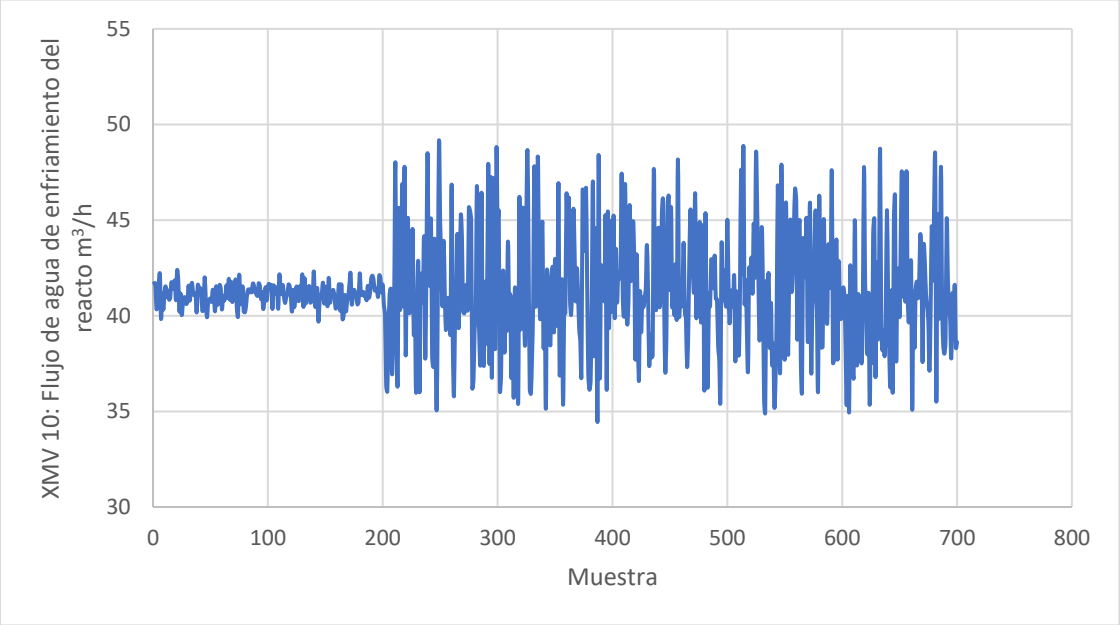


Figura 23

Falla 12: Temp. agua de enfriamiento a la entrada del condensador.

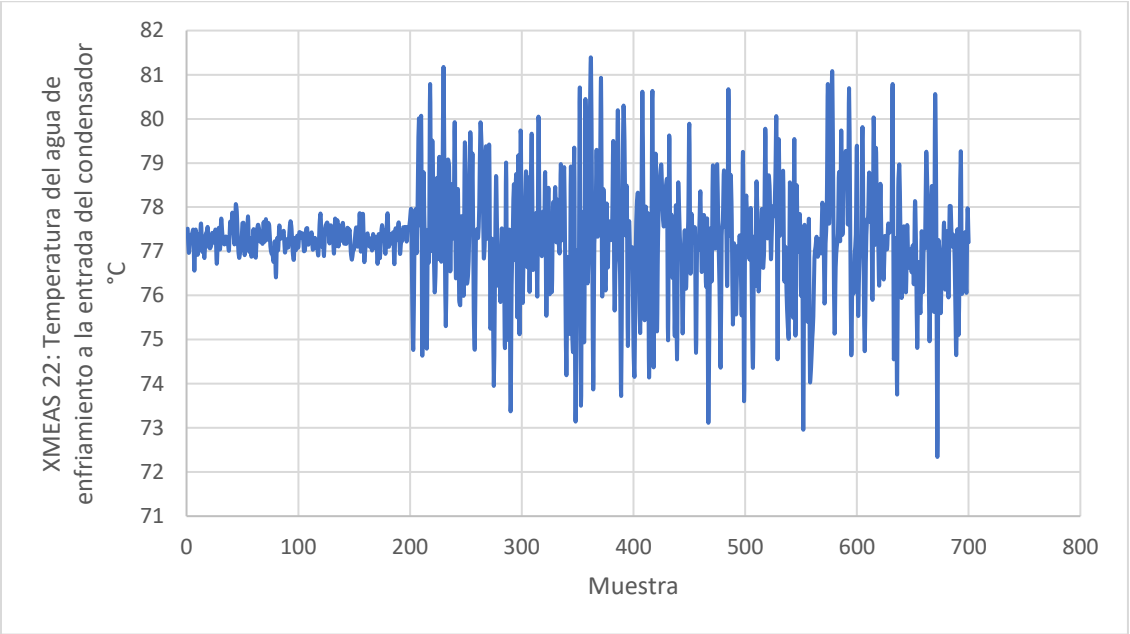
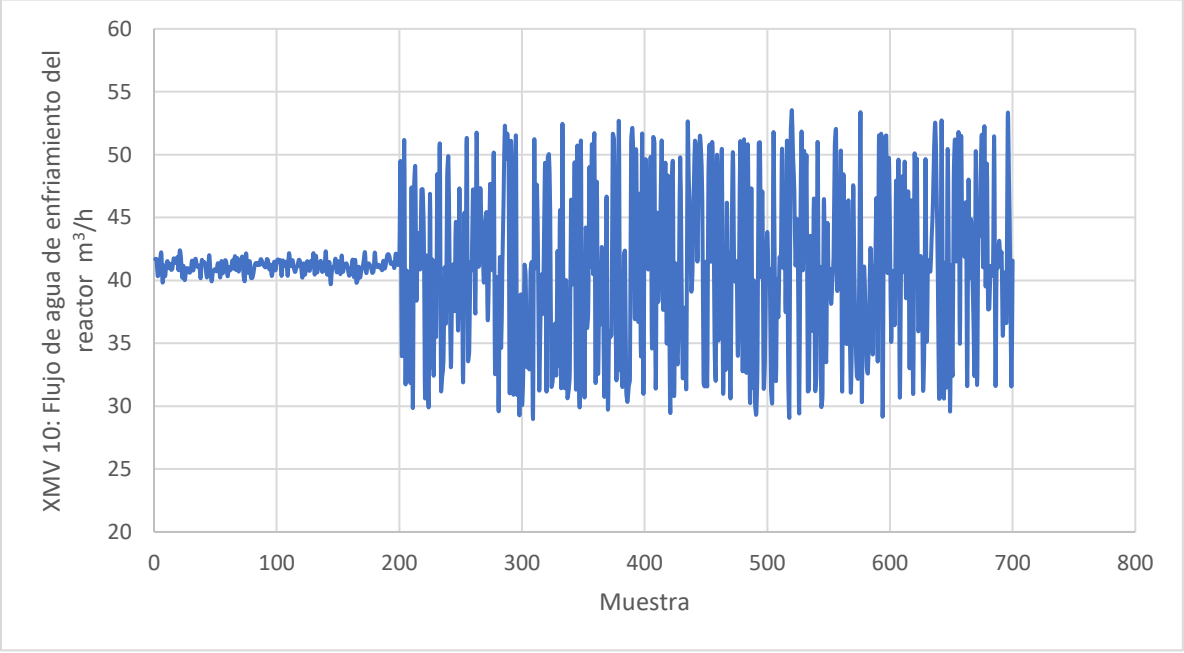


Figura 24

Falla 14: Flujo de agua de enfriamiento del reactor.



Apéndice C

Código de entrenamiento de las RNA usando la función neuralnet y el software R.

```
> library(Rcpp)
> library(RSNNS)
> library(neuralnet)
> library(readxl)
> X500Datos <- read_excel("C:/Users/Sebastian/Downloads/500Datos.xlsx")
> View(X500Datos)
> data<-X500Datos
> n<-names(data)
> maxs <- apply(data[,1:53], 2, max)
> mins <- apply(data[,1:53], 2, min)
> scaled01=normalizeData(data[,1:53],type="0_1")
> samplesize = as.integer(0.70 * nrow(data))
> set.seed(nrow(data))
> i<-0
> MSEnn<-matrix(nrow=104,ncol=5)
> MSEnnt<-matrix(nrow=104,ncol=5)
> MSEnntot<-matrix(nrow=104,ncol=5)
> R2_<-matrix(nrow=104,ncol=5)
> R2_t<-matrix(nrow=104,ncol=5)
> R2_tot<-matrix(nrow=104,ncol=5)
> for (k in 1:104) {
      i=i+1
      for (j in 1:5) {
          index = sample(seq_len(nrow(scaled01)), size = samplesize)
          Dtrain<- scaled01[index,]
          Dtest <- scaled01[-index,]
          colnames(Dtrain) <- n
          colnames(Dtest) <- n
          f <- as.formula(paste("Output ~", paste(n[!n %in% "Output"], collapse = " + "
))))
          model0 <- neuralnet(f,data=Dtrain,hidden=c(k),linear.output=T,err.fct="sse"
,act.fct="logistic",stepmax=100000,threshold=0.2, algorithm="rprop+",rep=3)
          pr.nn <- compute(model0,Dtest[,1:52])
          pr.nn_ <- pr.nn$net.result*(max(data$Output)-min(data$Output))+min
(data$Output)
          test.r <- Dtest[,53]*(max(data$ Output)-min(data$ Output))+min(data$ Output)
          ym_<-mean(Dtest[,53])
          MSEnn[i,j]<-sum((test.r-pr.nn_)^2)/nrow(Dtest)
          R2_[i,j]<-1-sum((test.r-pr.nn_)^2)/sum((test.r-ym_)^2)
          pr.nn <- compute(model0,Dtrain[,1:52])
          pr.nn_t <- pr.nn$net.result*(max(data$ Output)-min(data$ Output))+min(data$
Output)
          test.rt<- Dtrain[,53]*(max(data$ Output)-min(data$ Output))+min(data$ Output)
          ym_t<-mean(Dtrain[,53])
          MSEnnt[i,j]<-sum((test.rt-pr.nn_t)^2)/nrow(Dtrain)
          R2_t[i,j]<-1-sum((test.rt-pr.nn_t)^2)/sum((test.rt-ym_t)^2)
          MSEnntot[i,j]=(sum((test.r-pr.nn_)^2)+sum((test.rt-pr.nn_t)^2))/(nrow(data))
          R2_tot[i,j]<-1-(sum((test.r-pr.nn_)^2)+sum((test.rt-pr.nn_t)^2))/(sum((test.r
-ym_)^2)+sum((test.rt-ym_t)^2))
      }
      print(k)
}
```

Apéndice D

Graficas de los datos obtenidos con 1200 y 500 datos de entrada a la Red, threshold de 0.5 y las funciones de activación logistic y tanh.

Figura 25

Coeficiente de determinación 1200datos logistic threshold 0.5.

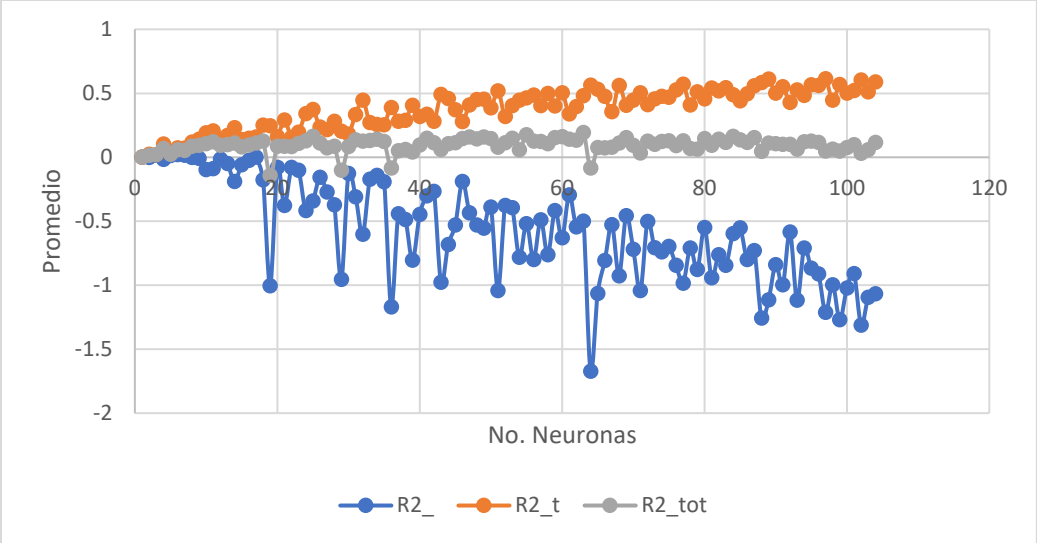


Figura 26

Error cuadrático medio 1200datos logistic threshold 0.5.

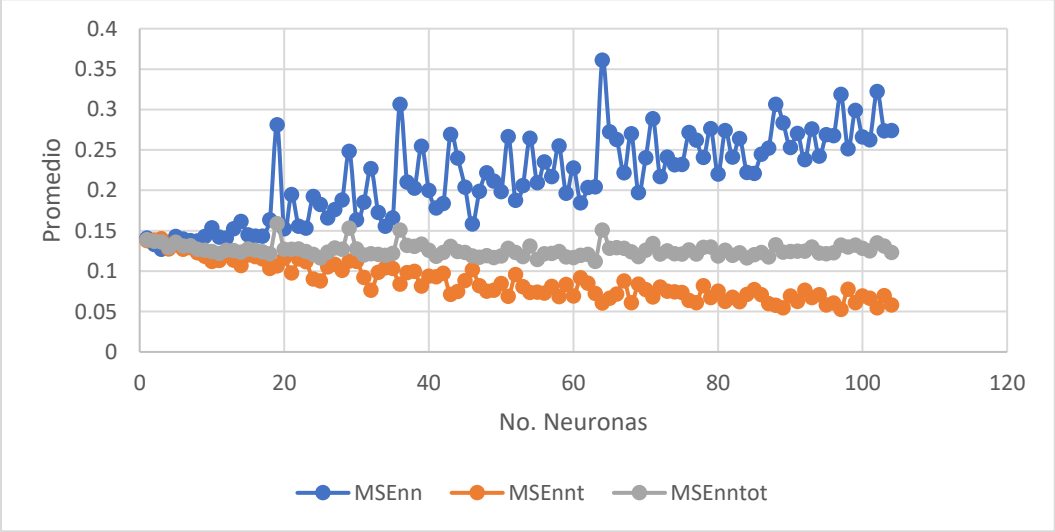


Figura 27

Coeficiente de determinación 1200datos tanh threshold 0.5.

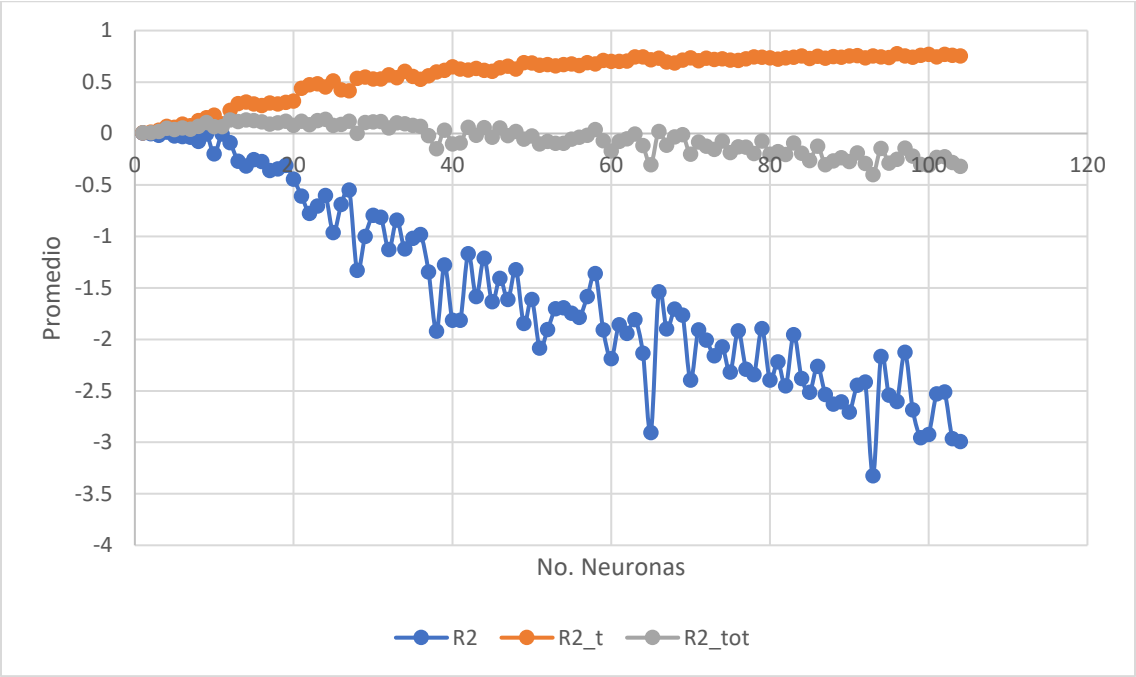


Figura 28

Error cuadrático medio 1200datos tanh threshold 0.5.

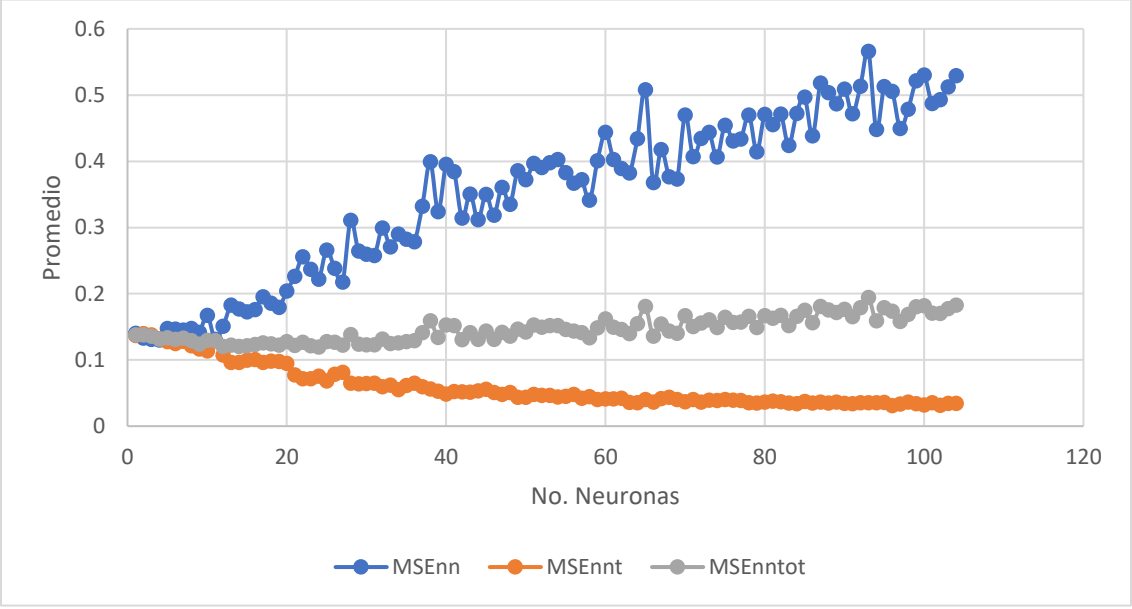


Figura 29

Coeficiente de determinación 500datos logistic threshold 0.5.

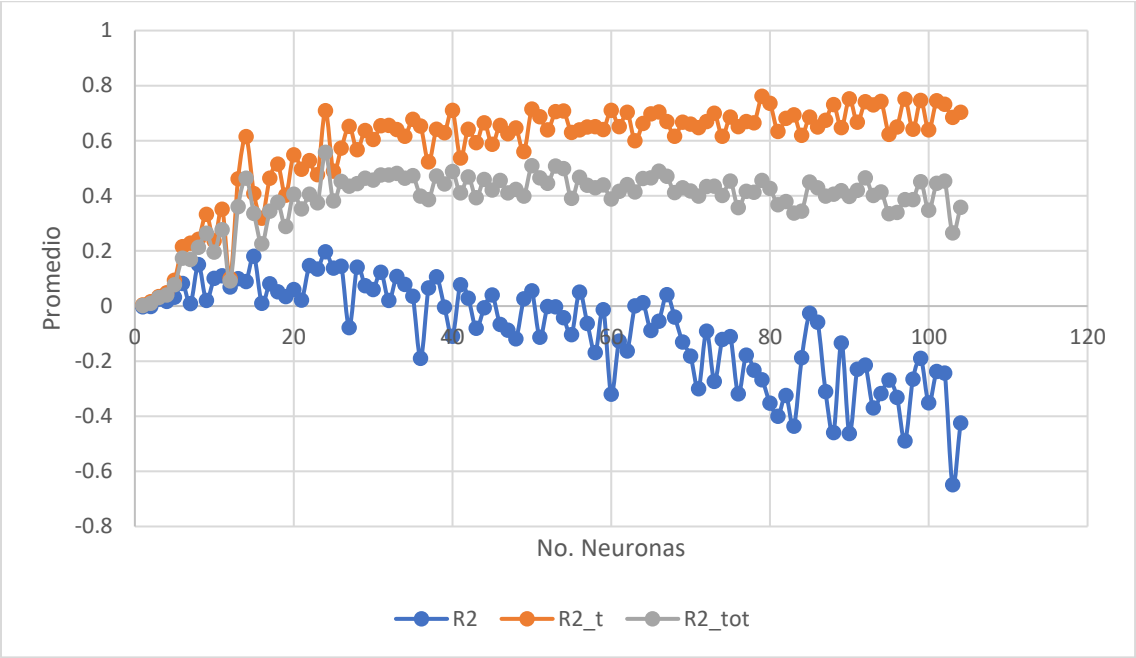


Figura 30

Error cuadrático medio 500datos logistic threshold 0.5.

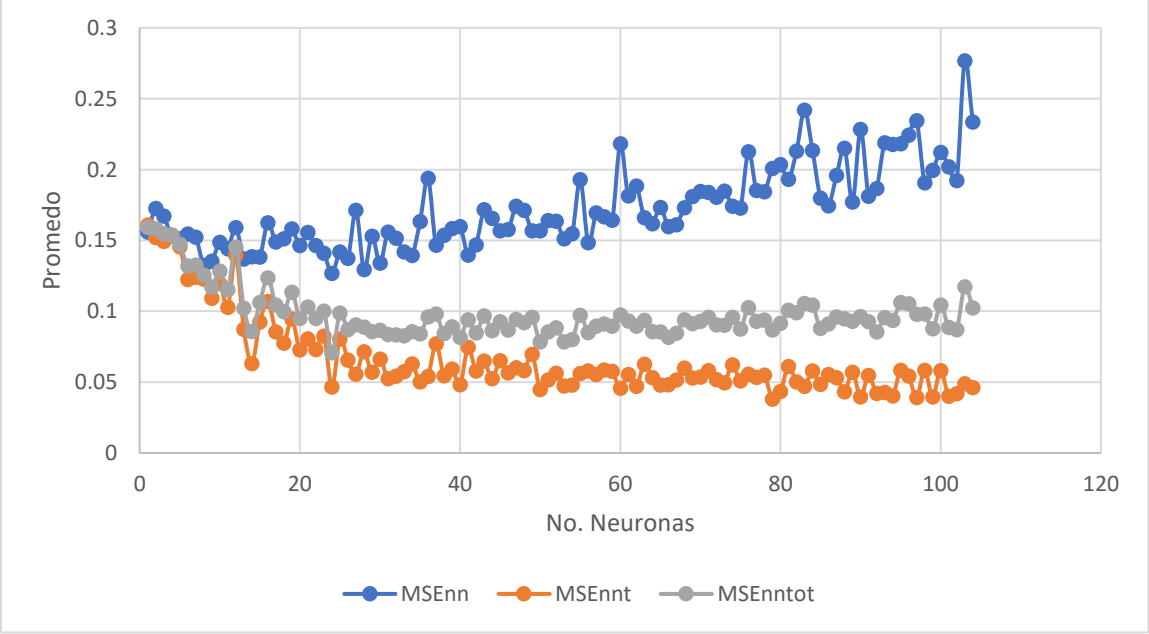


Figura 31

Coeficiente de determinación 500datos tanh threshold 0.5.

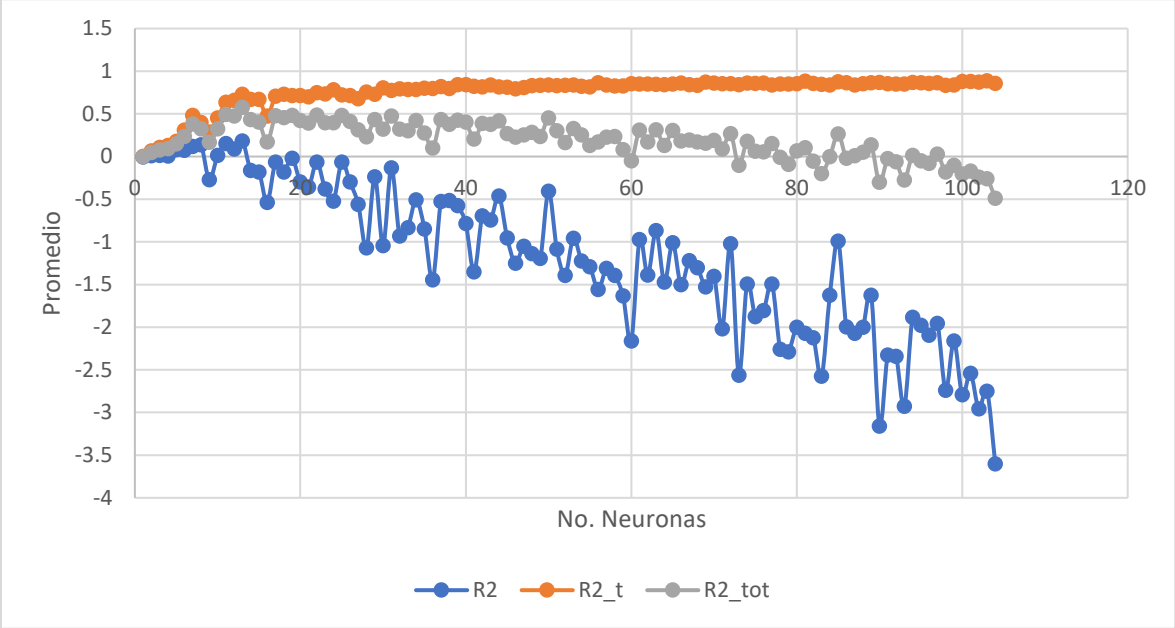
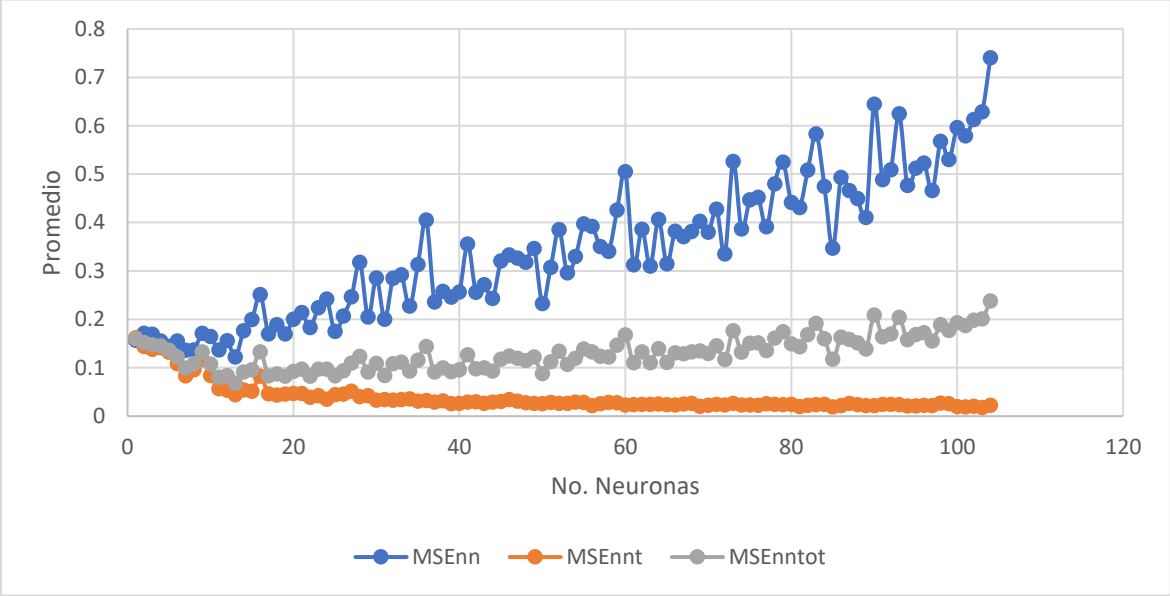


Figura 32

Error cuadrático medio 500datos tanh threshold 0.5.



Apéndice E

Código de entrenamiento y validación de la Red 52:11:1 en Python.

```
[ ] #Llamar las librerias
    from keras.datasets import boston_housing
    import pandas as pd
    from sklearn.preprocessing import StandardScaler
    from keras.models import Sequential
    from keras.layers import Dense,Activation
    import matplotlib.pyplot as plt
    from keras.utils import plot_model

[ ] #Importar los datasets, asignarles una variable, eliminar las columnas innecesarias para la RNA y estandarizar
    #de entrada
    data1=pd.read_csv("DATATRAINCEROS.csv")
    data2=pd.read_csv("DATATESTCEROS1.csv")
    data3=pd.read_csv("DATATESTCEROS3.csv")
    data1=data1.drop(["Unnamed: 0","Unnamed: 0.1","simulationRun","sample","Unnamed: 0.1.1","Unnamed: 0.1.1.1"],axis=1)
    y=data1['faultNumber']
    x=data1.drop(["faultNumber"],axis=1)
    x=StandardScaler().fit_transform(x)

[ ] #Asignar las variables de prueba con la normalización de las variables de entrada
    y_test=data2['faultNumber']
    x_test=data2.drop(["faultNumber","Unnamed: 0","Unnamed: 0.1","simulationRun","sample","Unnamed: 0.1.1"],axis=1)
    x_test=StandardScaler().fit_transform(x_test)

[ ] #Asignar datos para validación con la normalización de las variables de entrada
    y_validation=data3['faultNumber']
    x_validation=data3.drop(["Unnamed: 0","Unnamed: 0.1","simulationRun","sample","faultNumber","Unnamed: 0.1.1"],axis=1)
    x_validation=StandardScaler().fit_transform(x_validation)

[ ] #Modelo de la RNA, definición de numero de capas, numero de neuronas por capa, número de interacción entre
    #los parámetros, función de activación, función de perdida y optimizador
    model=Sequential()
    model.add(Dense(52,input_dim=52,kernel_initializer='normal',activation='sigmoid'))
    model.add(Dense(11,input_dim=11,kernel_initializer='normal',activation='sigmoid'))
    model.add(Dense(1,input_dim=1,kernel_initializer='normal',activation='sigmoid'))
    model.compile(loss='mean_squared_error',optimizer="adam")

[ ] #Entrenamiento de la RNA definiendo el tamaño de muestra, número de iteraciones y datos de validación
    history=model.fit(x,y,batch_size=100,epochs=50,validation_data=(x_validation,y_validation))

[ ] #Evaluación del modelo de entrenamiento con datos de prueba
    resultado=model.evaluate(x_test,y_test)
    print("Evaluación MSE de prueba:", resultado)

[ ] #Presentación gráfica de la función de perdida MSE vs Entrenamiento y Validación
    print(history.history.keys())
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Modelo MSE')
    plt.ylabel('MSE')
    plt.xlabel('Iteraciones')
    plt.legend(['Entrenamiento', 'Validación'], loc='upper left')
    plt.show()
```

Apéndice F

Código de entrenamiento y validación de la Red 52:80:50:30:21 en Python.

```
[ ] #Llamar las librerias
    from keras.datasets import boston_housing
    import pandas as pd
    from sklearn.preprocessing import StandardScaler
    from keras.models import Sequential
    from keras.layers import Dense,Activation
    import matplotlib.pyplot as plt
    from keras.utils import plot_model

[ ] #Importar los datasets, asignarles una variable, eliminar las columnas innecesarias para la RNA y estandarizar
    #de entrada
    data1=pd.read_csv("DATA_TRAINING.csv")
    data2=pd.read_csv("data_test1.csv")
    data3=pd.read_csv("data_test2.csv")
    data1=data1.drop(["Unnamed: 0","simulationRun","sample","Unnamed: 0.1","Unnamed: 0.1.1"],axis=1)
    y=data1['faultNumber']
    x=data1.drop(["faultNumber"],axis=1)
    x=StandardScaler().fit_transform(x)

[ ] #Asignar las variables de prueba con la normalización de las variables de entrada
    y_test=data2['faultNumber']
    x_test=data2.drop(["faultNumber","Unnamed: 0","Unnamed: 0.1","simulationRun","sample"],axis=1)
    x_test=StandardScaler().fit_transform(x_test)

[ ] #Asignar datos para validación con la normalización de las variables de entrada
    y_validation=data3['faultNumber']
    x_validation=data3.drop(["Unnamed: 0","Unnamed: 0.1","simulationRun","sample","faultNumber"],axis=1)
    x_validation=StandardScaler().fit_transform(x_validation)

[ ] #Modelo de la RNA, definición de numero de capas, numero de neuronas por capa, número de interacción entre
    #los parámetros, función de activación, función de perdida y optimizador
    model=Sequential()
    model.add(Dense(52,input_dim=52,kernel_initializer='normal',activation='relu'))
    model.add(Dense(80,input_dim=80,kernel_initializer='normal',activation='relu'))
    model.add(Dense(50,input_dim=50,kernel_initializer='normal',activation='relu'))
    model.add(Dense(30,input_dim=30,kernel_initializer='normal',activation='relu'))
    model.add(Dense(21,input_dim=21,kernel_initializer='normal'))
    model.compile(loss='mean_squared_error',optimizer="adam")

[ ] #Entrenamiento de la RNA definiendo el tamaño de muestra, número de iteraciones y datos de validación
    history=model.fit(x,y,batch_size=100,epochs=50,validation_data=(x_validation,y_validation))

[ ] #Evaluación del modelo de entrenamiento con datos de prueba
    resultado=model.evaluate(x_test,y_test)
    print("Evaluación MSE de prueba:", resultado)

[ ] #Presentación gráfica de la función de perdida MSE vs Entrenamiento y Validación
    print(history.history.keys())
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Modelo MSE')
    plt.ylabel('MSE')
    plt.xlabel('Iteraciones')
    plt.legend(['Entrenamiento', 'Validación'], loc='upper left')
    plt.show()
```

Apéndice G

Arquitectura de las RNA

Figura 33

Arquitectura RNA 52:104:1 en el software R.

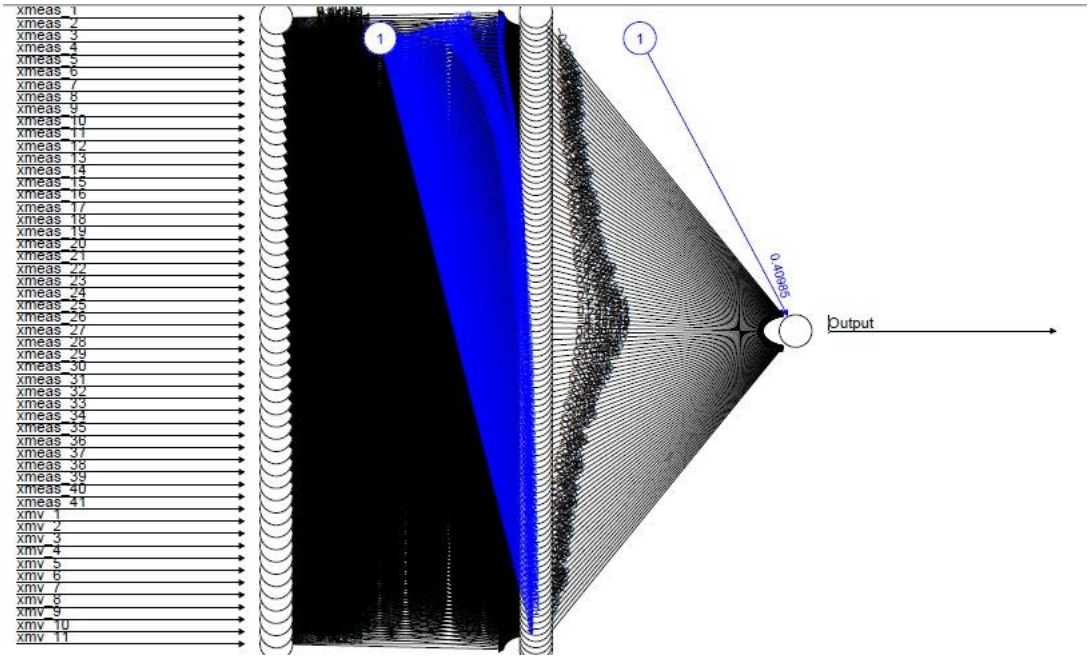


Figura 34

Arquitectura 52:11:1 en código Python.

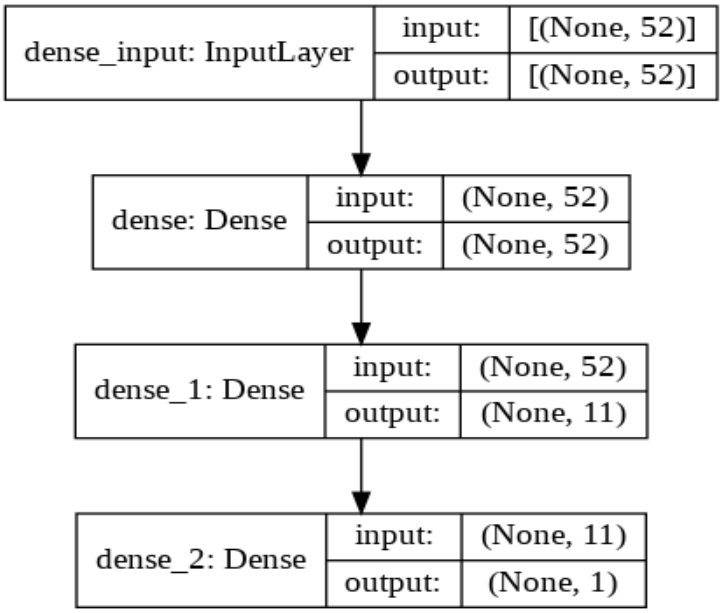
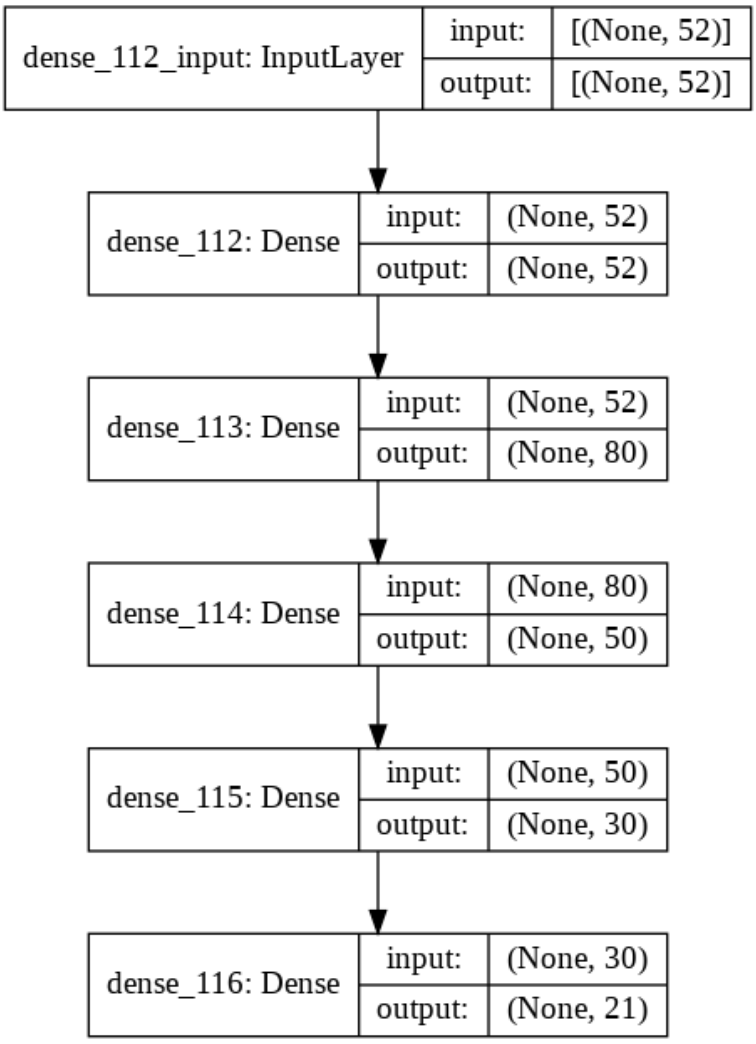


Figura 35

Arquitectura 52:80:50:30:21 en código Python.



Apéndice H

Código de modelo comparativo con predicción de la Red.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow import metrics
y_prediccion=model.predict(x_predict)
e=tf.keras.metrics.FalseNegatives()
e.update_state(y_test,y_prediccion)
print('Falsos Negativo:',e.result().numpy())
```

```
Falsos Negativo: 129.0
```