UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO - SALAMANCA
DIVISIÓN DE INGENIERÍAS

"SISTEMA DE MONITOREO PARA LA DETECCIÓN DE FALLAS EN MOTORES ELÉCTRICOS"

TESIS PROFESIONAL

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA

PRESENTA:

ING. AGUILAR PUENTE ANGEL DE JESÚS

ASESORES:

DR. LEDESMA OROZCO SERGIO EDUARDO

DR. GONZALEZ PARADA ADRIAN

SALAMANCA, GUANAJUATO SEPTIEMBRE 2021

Dedicado primeramente a Dios
Y a mi familia por haber sido
Parte fundamental en esta tesis.

Agradecimientos Institucionales

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por haberme apoyado con la asignación de la beca nacional para el estudio de programas de posgrado, lo cual me permitió llevar a cabo mis estudios en una institución académica de prestigio internacional, teniendo como número de registro de CVU: 1007465.

A la Universidad de Guanajuato por permitirme formarme profesionalmente en la División de Ingenierías Campus Irapuato – Salamanca (DICIS).

A la DICIS, al Posgrado de Ingeniería Eléctrica por las instalaciones, la atención y el apoyo que me proporcionaron durante la realización de mis estudios de maestría y de esta tesis, además, del apoyo que me brindaron los profesores de la institución.

Agradecimientos Personales

Le doy gracias a Dios por permitirme cumplir una meta más en mi vida, por mantenerme con buena salud a lo largo de mi vida.

Le doy gracias a mi familia por siempre estar conmigo apoyándome en cada uno de mis logros y además de apoyarme cuando tengo momentos de debilidad siempre están para aumentar mi autoestima.

Quiero agradecer profundamente a mis asesores el Dr. Adrián Gonzalez Parada y al Dr. Sergio Eduardo Ledesma Orozco, por confiar en mi y compartir de sus bastos conocimientos. Por apoyarme siempre y estar al pendiente cuando tenia alguna duda con la elaboración de este trabajo de tesis, además por la paciencia que me brindaron.

En especial quisiera agradecer al Dr. Juan Gabriel Aviña Cervantes por preocuparse que adquiriéramos los conocimientos necesarios para estar al nivel que demanda la maestría, además de ser un excelente investigador y profesor de la Universidad de Guanajuato.

Finalmente quiero agradecer a todos mis amigos y compañeros de la maestría: Viridiana, Luis Miguel, Juan Manuel, Andrés, Julio, Giselle, Cesar; por brindarme su amistad y hacer de esta aventura una muy agradable experiencia. Además de una buena amistad que formamos, yo los considero una parte más de mi familia. Mis amigos fuera de la maestría que siempre me apoyaron y estuvieron al pendiente de mi como lo fue mi mejor amigo Juan José y Sandra.

Resumen

Este trabajo de tesis describe la importancia de la detección temprana de fallas en motores eléctricos de inducción, además de las técnicas de análisis tradicionales y una propuesta utilizando redes neuronales artificiales.

Se investigaron las fallas eléctricas más comunes que pueden presentar los motores eléctricos de inducción. De tal manera se diseño una red neuronal en el software Neural Lab que fuera capas de detectar las fallas que este presentando el motor eléctrico de inducción.

Adicionalmente en este trabajo se implementó la técnica tradicional por sus siglas en ingles (Motor Current Signature Analysis MCSA) para la detección de fallas eléctricas de motores de inducción. Se diseñó una red neuronal artificial capas de analizar los datos en el dominio del tiempo y en el dominio de la frecuencia, además, se diseñó el clasificador Support Vector Machine (SVM) en el dominio del tiempo para comparar los resultados obtenidos por la red neuronal y el SVM.

Se puede concluir que con la elaboración de este trabajo de tesis se tendrá un sistema de monitoreo óptimo implementando redes neuronales en la detección de fallas de motores eléctricos de inducción. Por lo que será de gran ayuda para las industrias que utilicen estos tipos de maquinas eléctricas ya que podrán monitorear su comportamiento antes de que presenten una falla.

Abstract

This thesis work describes the importance of early fault detection in induction electric motors. In addition, traditional analysis techniques and a proposal using artificial neural networks (ANN).

The most common electrical faults that can occur in induction electric motors were investigated. In such a way, a ANN was designed in the Neural Lab software, that was able to detect the faults that the electric induction motor is presenting.

In addition, in this work the traditional Motor Current Signature Analysis technique was implemented for the detection of electrical failure of induction motors. A neural network capable of analyzing data in the time domain and in the frequency domain was designed. Likewise, the Support Vector Machine (SVM) classifier was designed in the time domain to compare the results obtained by the ANN and the SVM.

It can be concluded that, with the elaboration of this thesis work, an optimal monitoring system will be implemented by implementing ANN in the detection of induction electric motor failures. Therefore, it will be of great help to the industries that use these types of electrical machines, as they will be able to control its behavior before it presents a failure.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Un sistema de análisis en tiempo real puede ser una herramienta de trabajo muy útil para cualquier empresa o investigación que se esté desarrollando. Si este sistema además cuenta con innovaciones de análisis; como pueden ser las redes neuronales (RNA), sería un sistema eficiente e innovador. Ya que las RNA se pueden aplicar a cualquier tipo de problema que se pueda presentar en la vida cotidiana, como pueden ser en: problemas de ingeniería, nuevos métodos de investigación, ciencias de la salud y una infinidad de aplicaciones que se le pueden atribuir.

Hace algunas décadas, se pudo pensar que las RNA no tendrían un impacto tan influyente en cualquiera de las áreas de su aplicación actual. Sin embargo, hoy en día varias áreas de investigación y de ingeniería buscan poder optimizar sus procesos de análisis. Consecuentemente, las RNA son de gran ayuda, ya que pueden ser adaptables a un amplio rango de problemas.

En la última década, se a estudiado el análisis de las características de la corriente de un motor de inducción por sus siglas en ingles (Motor Current Signature Analysis MCSA) es una técnica de monitoreo de máquinas eléctricas. Que son las técnicas de monitoreo más comunes, en la detección de fallas. Además, son los análisis que mejores resultados han generado en la detección de fallas eléctricas en motores de inducción.

Debido al gran impacto que han tenido las redes neuronales en el campo de la ingeniería eléctrica; principalmente en la detección de fallas eléctricas. Se han implementado las técnicas de MCSA con la ayuda de redes neuronales;

generando un gran impacto de análisis de fallas. Consecuentemente, estos análisis han sido de mayor confiabilidad en sus resultados. En la figura 1.1 se puede observar una red neuronal para detectar el estado del rotor de un motor de inducción.

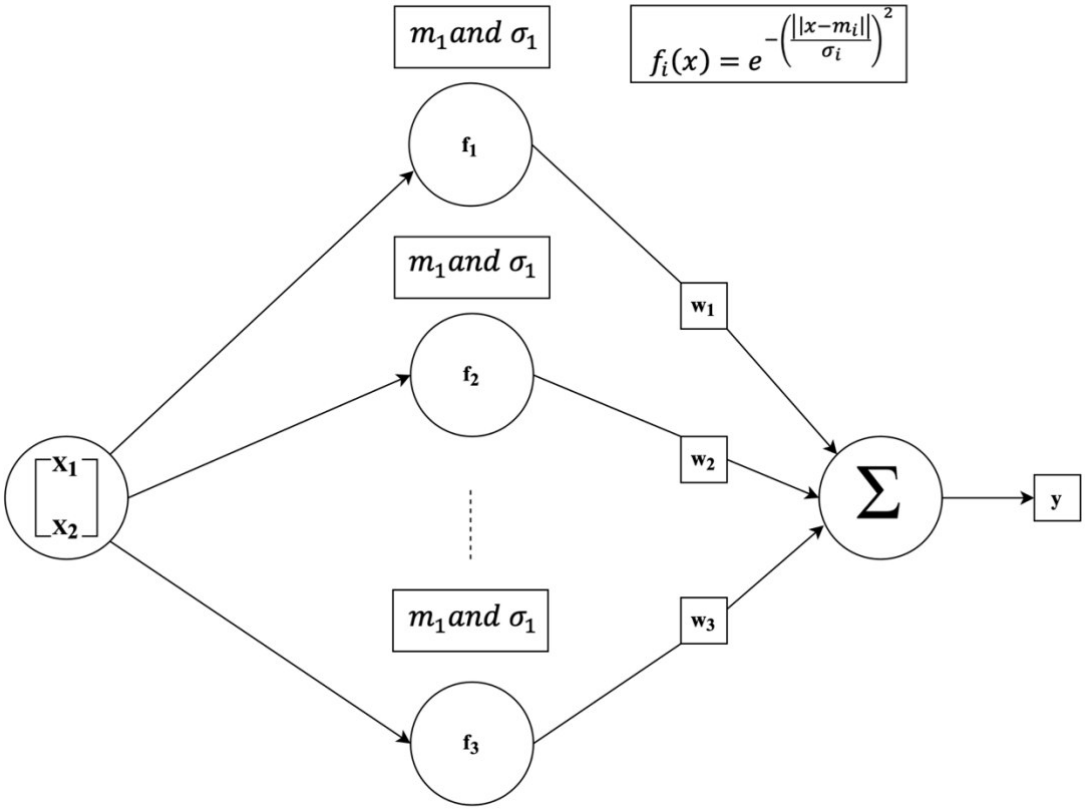

Figura 1.1. Red neuronal con 12 neuronas en la capa oculta para detectar el estado del rotor.

T. Yektaniroumand et al. / IJE TRANSACTIONS B: Applications Vol. 31, No. 11, (November 2018) 1876-1882.

Las RNA son eficientes en sus análisis porque se pueden programar a la necesidad de quien las diseña. Estas redes son confiables siempre y cuando se tenga presente que las redes se entrenan y se validan para que los resultados sean lo más confiable posible.

Todos los análisis que se hagan con RNA deben de tener mejoras, ya que estas redes están diseñadas para que aprendan a resolver los problemas para los cuales se le entrenan. Por lo que estás, deben ser capaces de interpretar los diferentes valores que pueda presentar el sistema en las diferentes condiciones en las que se puedan encontrar los motores eléctricos, por lo tanto, la RNA debe de tomar las mejores decisiones de análisis y arrojar datos confiables.

Capítulo 1

1.1 Antecedentes

El motor de inducción es la máquina eléctrica mas utilizada en la actualidad. Este es utilizado en el 70% de las aplicaciones industriales y este tipo de motores consumen más del 50% de la energía generada en las naciones industrializadas **[1].**

Adicionalmente, una falla imprevista de un motor de inducción puede ocasionar el paro de producción de una empresa. Esta falla puede tener una consecuencia enorme de pérdidas de producción estas pueden llegar a ser de miles de dólares por minuto **[2].**

Una detección temprana de una falla prolonga el periodo de la vida útil de un motor. Además de que esto puede reducir el costo de partes de repuesto o refacciones que el motor pueda necesitar cuando este presente una falla. Como bien se sabe, las fallas más comunes de un motor de inducción se pueden presentar en el rotor, el devanado del estator o en los rodamientos **[3].**

El análisis de MCSA, es una técnica de monitoreo de condición que ahora se utiliza ampliamente para diagnosticar problemas como:

- barras rotas en el rotor.
- niveles anormales de excentricidad del entrehierro.
- cortocircuitos en los devanados del estator.
- problemas mecánicos y eléctricos.

La idea es que la corriente del estator contenga componentes directamente relacionados con el flujo de rotación, causado por fallas por barras rotas del rotor, excentricidad, entre otras. A pesar de que la falla del motor puede detectarse mediante el análisis del espectro de vibración mecánica, la vibración es un defecto de segundo orden en comparación con los componentes actuales y, en muchos casos, la gravedad de la falla (es decir el número de barras rotas del rotor) debe ser alta para ser detectado por análisis de vibraciones **[4].**

Existen diferentes métodos de diagnóstico de fallas para motores de inducción, donde se pueden categorizar ampliamente como:

- Técnica basada en modelos.
- Técnica de procesamiento de señales.
- Técnicas de computación suave.
- Técnicas de análisis con redes neuronales.

En donde estos métodos tienen que analizar las señales provenientes de los sensores. De esta forma es posible determinar cual técnica es la más adecuada y óptima posible para determinar las posibles fallas que pueda presentar cualquier equipo eléctrico.

Con el objetivo de realizar un análisis de distintas perspectivas desde las cuales se ha abordado la detección y diagnóstico de fallas mediante técnicas de inteligencia artificial **[5].**

En la figura 1.2 se puede observar una clasificación de métodos de identificación y diagnóstico de fallas con inteligencia artificial.

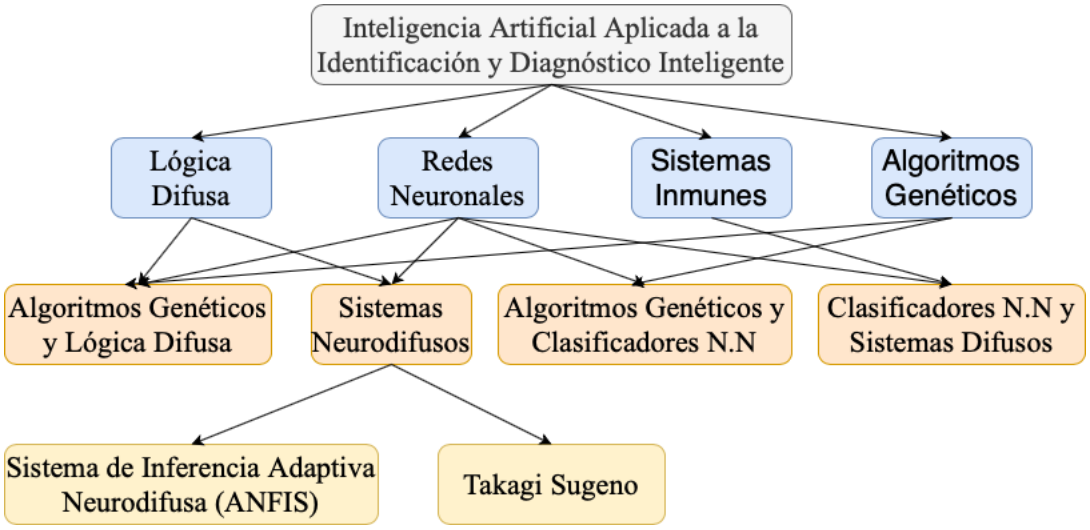

Figura 1.2: Clasificación de métodos de identificación y diagnóstico de fallas con inteligencia artificial.

1.2 Planteamiento del problema

Los motores de inducción son los equipos eléctricos más utilizados en la industria. De esta forma, es necesario mantener a los motores de inducción en constante monitoreo. Debido a la gran importancia de los motores en la industria, estor pueden representar perdidas considerables si algún motor llegara a fallar

en una línea de producción. Además, de producir perdidas monetarias a las empresas este tipo de fallas pueden ocasionar perdidas humanas que se encuentren trabajando en las líneas de producción.

Actualmente los motores eléctricos de inducción representan un gran trabajo de investigación para detectar fallas eléctricas o mecánicas. Estos métodos de investigación se concentran en detectar estas fallas con tiempo y no cuando la falla ya este presente; por lo que es muy importante identificar una falla incipiente.

Por lo tanto, en este trabajo se resolverá el problema de la detección incipiente de fallas eléctricas en un motor de inducción. Este trabajo se ayuda de un análisis por medio de RNA para obtener un sistema eficiente de detección temprana de dichas fallas. Adicionalmente, el método propuesto en este trabajo se diseño para garantizar el adecuado funcionamiento de los motores de inducción.

1.3 Objetivos

Crear un sistema de análisis en frecuencia e identificación de patrones para la detección de fallas en motores eléctricos. Este sistema se basará en sensores no intrusivos y se implementará por medio de inteligencia artificial. Consecuentemente, este sistema podrá ser aplicado a equipos eléctricos nuevos o en servicio de alguna empresa o industria. Los objetivos particulares son:

1.3.1 Implementar un sistema de identificación de patrones para la detección incipiente de fallas.

1.3.2 Proponer un algoritmo de análisis en frecuencia e identificación de patrones para detección incipiente de fallas utilizando inteligencia artificial.

1.3.3 Interpretación de los datos obtenidos, para determinar una posible falla.

1.3.4 Implementación del sistema de alarma de posible falla.

1.4 Justificación

Actualmente el crecimiento de las empresas en México es cada día mayor. De igual modo, la producción de estas empresas también ha crecido en los últimos años. Por estas razones, se busca ayudar a las industrias a que su línea de producción no se vea afectada por alguna falla que se pueda presentar en sus equipos eléctricos. Ya que estos equipos son elementos fundamentales dentro de la línea de producción en la industria, lo más importante es estar atentos en su funcionamiento a fin de predecir en el momento que pueda ocurrir una falla. Esto es, es importante detectar la falla antes y no hasta cuando la falla ya está presente. Así, el sistema propuesto en este trabajo de tesis basado en el análisis en frecuencia e indentificación de patrones ayudará a que la producción en las industrias no se vea afectada. Adicionalemente, el trabajo de esta tesis puede usarse para corregir cualquier desperfecto que se presente al inicio de una falla. En resumén, el método propuesto en este trabajo se diseñó para reducir las pérdidas monetarias, y principalmente las pérdidas humanas, en las empresas Mexicanas.

Los motores de inducción son uno de los principales equipos eléctricos que se encuentran en la industria, un laboratorio de pruebas eléctricas o en alguna empresa. De esta forma es muy importante tener un sistema de análisis para monitorear estos equipos eléctricos. Se desarrollará, un algoritmo que sea capaz de identificar el tipo de falla que puede presentar el motor (barras rotas, fallas en el entrehierro, corto circuitos en los devanados del estator entre otros). Con la implementación del algoritmo, y las adecuaciones necesarias se podrá monitorear cualquier equipo eléctrico o cable de alimentación eléctrica.

Con este sistema de análisis se planea ayudar a la industria, laboratorios de pruebas y cualquier empresa con equipos eléctricos. Garantizando un funcionamiento adecuado e ininterrumpido de las instalaciones y de los equipos.

1.5 Descripción de los capítulos

Esta tesis está dividida en 4 capítulos. El capítulo 2 describe la información teórica a utilizar para el desarrollo de este trabajo. Así el capítulo 2 incluye los conceptos básicos de las redes neuronales y las diferentes técnicas de análisis

que se pueden utilizar en la detección de fallas. El capítulo 2 termina explicando las principales fallas que se pueden presentar en los motores de inducción y las principales características del análisis de la Transformada Rápida de Fourier a los equipos eléctricos.

El capítulo 3 describe la metodología que se llevó a cabo para el desarrollo de este trabajo. Este capítulo incluye el desarrollo de la base datos de las pruebas realizadas al motor. Adicionalmente, el capítulo explica el algoritmo de análisis para la detección de fallas incipientes a equipos eléctricos, con técnicas tradicionales y la metodología propuesta. El capítulo finaliza con una explicación para determinar el número de neuronas y capas implementadas para el análisis con el fin de poder determinar la vialidad de la metodología propuesta.

En el capítulo 4 se presentará la comunicación que se utilizó para la adquisición de datos. El capítulo 4 incluye también los resultados de las simulaciones por computadora usando el software Neural Lab. Este capítulo describe las distintas pruebas realizadas al motor de inducción, y al mismo tiempo, describe a detalle los resultados obtenidos. Finalmente, incluye las conclusiones generales obtenidas durante el desarrollo de este trabajo. Además, se menciona el trabajo a futuro y las posibles mejoras que pueden realizarse. En el apartado bibliográfico se encuentran los datos de la literatura que sirvió de apoyo para la escritura de esta tesis.

1.6 Resumen del primer capítulo

En resumen, este primer capítulo consistió en dar a conocer los antecedentes más relevantes acerca de la evolución tecnológica que se han ido desarrollando en la utilización de las redes neuronales. La infinidad de los diferentes campos de investigación que existen con las redes neuronales. Adicionalmente, dando a conocer las investigaciones más recientes que se han estado realizando para la detección de fallas de equipos eléctricos. Además, se mencionó una descripción general acerca del planteamiento del problema, los objetivos y la justificación para este trabajo de tesis.

Capítulo 2

Marco Teórico

En este capítulo se mencionarán los conceptos teóricos de las técnicas que se implementarán para el desarrollo de este trabajo.

2.1 Historia de las Redes Neuronales

1957 - Frank Rosenblatt. Comenzó el desarrollo del Perceptrón. Esta es la RNA más antigua; utilizándose hoy en día para aplicación como reconocedor de patrones. Este modelo era capaz de generalizar, es decir, después de haber aprendido una serie de patrones podía reconocer otros similares, aunque no se le hubiesen presentado anteriormente.

1960 - Bernard Widrow/Marcial Hoff. Desarrollaron el modelo Adaline (ADAptative LINear Elements). Esta fue la primera red neuronal aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas).

1974 - Paul Werbos. Desarrolló la idea básica del algoritmo de aprendizaje de propagación hacia atrás (backpropagation); cuyo significado quedó definitivamente aclarado en 1985.

1986 - David Rumelhart/G. Hinton. Redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation). A partir de 1986, el panorama fue alentador con respecto a las investigaciones y el desarrollo de las redes neuronales.

Capítulo 2

2.2 Definición de una red neuronal Artificial

Una RNA es un sistema de procesamiento de la información que tiene muchas características similares a las redes neuronales que se encuentran en el cerebro. Las RNA se desarrollaron con la finalidad de generalizar modelos matemáticos de cognición humana o de biología neuronal basadas en suposiciones las cuales se mencionan a continuación:

- El procesamiento de la información se lleva acabo en muchos elementos simples, los cuales reciben el nombre de neuronas.
- Cada una de las señales se pasan entre neuronas las cuales están conectadas entre si, unas con otras o también conocidos como enlaces de conexión.
- Cada uno de estos enlaces de conexión están asociados a un peso, el cuál, en una red neuronal su función es multiplicar la señal que se esta transmitiendo.
- Cada una de las redes neuronales aplica funciones de activación (lineal y no lineal) para poder determinar la señal que se obtendrá a la salida.

2.2.1 Preprocesamiento

Es uno de los temas más importantes que se tienen que tomar en cuenta al momento de que se va a desarrollar una RNA, ya que si llegaran a existir datos con errores o con características que no son favorables para la red podrían afectar gravemente los datos obtenidos por la red, por lo que se tiene que hacer un Pre-procesamiento, el cual se puede realizar por diferentes procedimientos, ya que la prioridad del Pre-procesamiento es acondicionar los datos de entrada a la RNA para que esta funcione adecuadamente centrándose solo en las características que le interesan.

Algunas de las técnicas empleadas para el procesamiento pueden ser las siguientes:

- Tablas de resumen de atributos.
- Resúmenes de estadísticas.
- Diagramas.

2.2.2 Pesos de las neuronas

En una RNA, cada neurona está conectada a otra neurona usando un peso w que es un valor numérico. Cada peso tiene dos subíndices que indican el índice de la neurona objetivo, mientras que el segundo subíndice representa el índice de la neurona fuente. Debido a que en cada capa las neuronas se numeran utilizando los mismos índices, es muy importante indicar a qué capa pertenece la neurona.

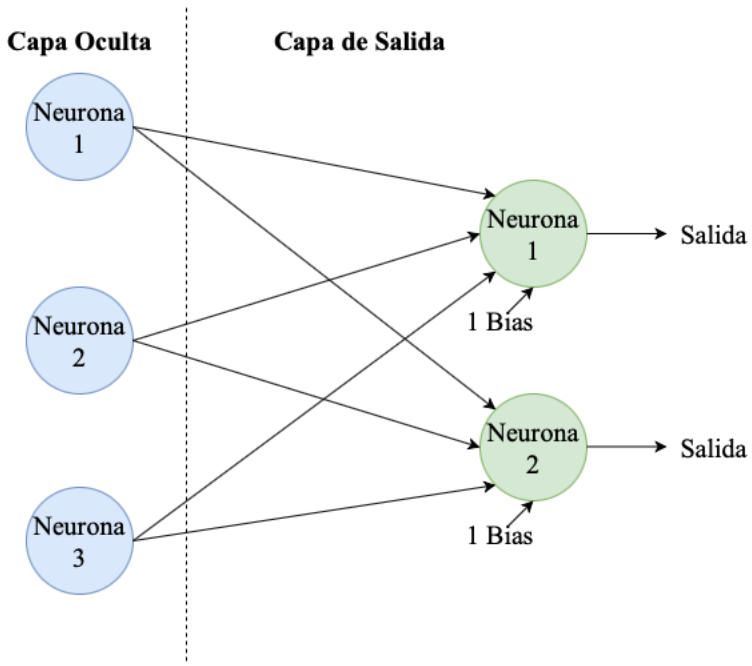

Figura 2.1: Pesos de una red neuronal.

2.2.3 Funciones de Activación

En las RNA como las biologicas, una neurona no solamente transmite la entrada que recibe, existe un proceso adicional, el cual es una función de

activación, la función de activación utiliza la misma suma del producto de la entrada anterior, y la transforma una vez más como salida.

$$z = b + \sum_i w_i x_i \tag{2.1}$$

Se han propuesto muchas funciones de activacion, en esta ocasicón solo se mencionaran dos en detalle: la sigmoide y unidad lineal rectificada (ReLU, por sus siglas en ingles).

La función sigmoide es la funcion de activacion más popular, y se define como:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

e es la constante exponencial, que es aproximadamente igual a 2.71828. Una neurona utiliza la sigmoide como función de activación se llama *neurona sigmoide.* Primero se establece que la variable z equivale a la suma ponderada de entrada y después pasa a traves de la funcion sigmoide.

$$z = b + \sum_i w_i x_i \tag{2.3}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

Aunque la ecuación parece complicada y arbitraria, en realidad tiene una forma bastante simple, se puede ver si se traza el valor de $\sigma(z)$ como función de la entrada z.

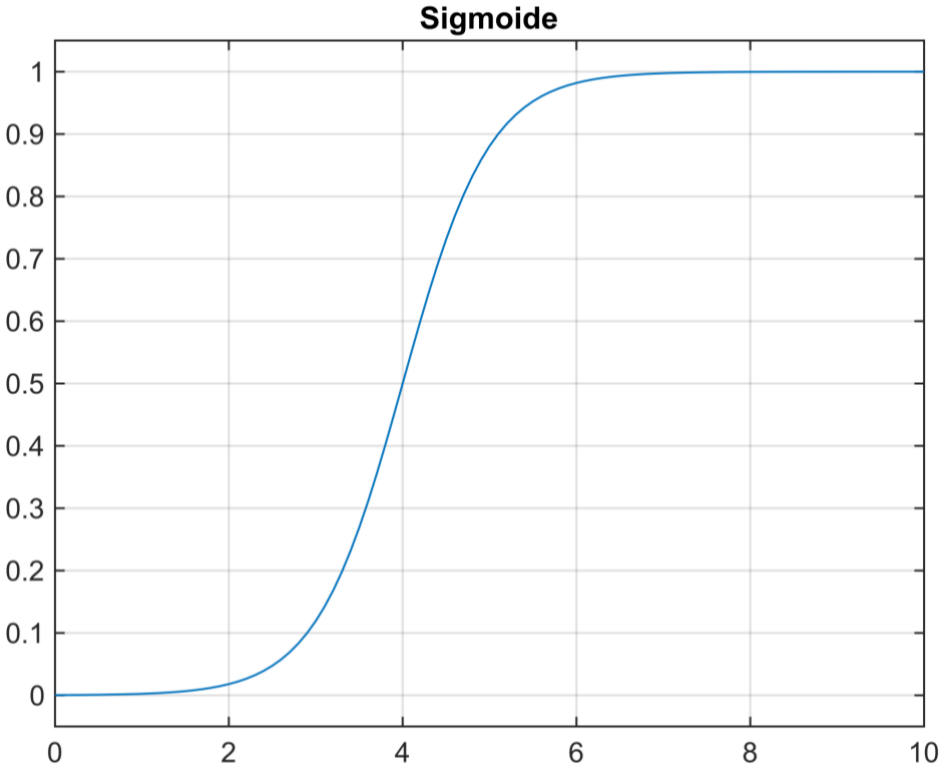

Figura 2.2: Función de activación sigmoide en Neural Lab.

Se puede observar que $\sigma(z)$ actua como una especie de función que comprime la salida a un rango de 0 a 1. En el centro, donde

$$z = 0 \tag{2.5}$$

$$\sigma(0) = \frac{1}{1 + e^{0}} = 0.5 \tag{2.6}$$

Para valores negativos grandes de z, el término e^{-z} en el denominador crece exponencialmente, y $\sigma(z)$ se aproxima a 0. Al contrario, valores positivos grandes de z, reducen e^{-z} hacia 0, y $\sigma(z)$ se aproxima a 1.

La mayoria de las RNA actuales usan otro tipo de funcion de activacion llamada ReLU, se define como:

$$R(z) = \max(0, z) \tag{2.7}$$

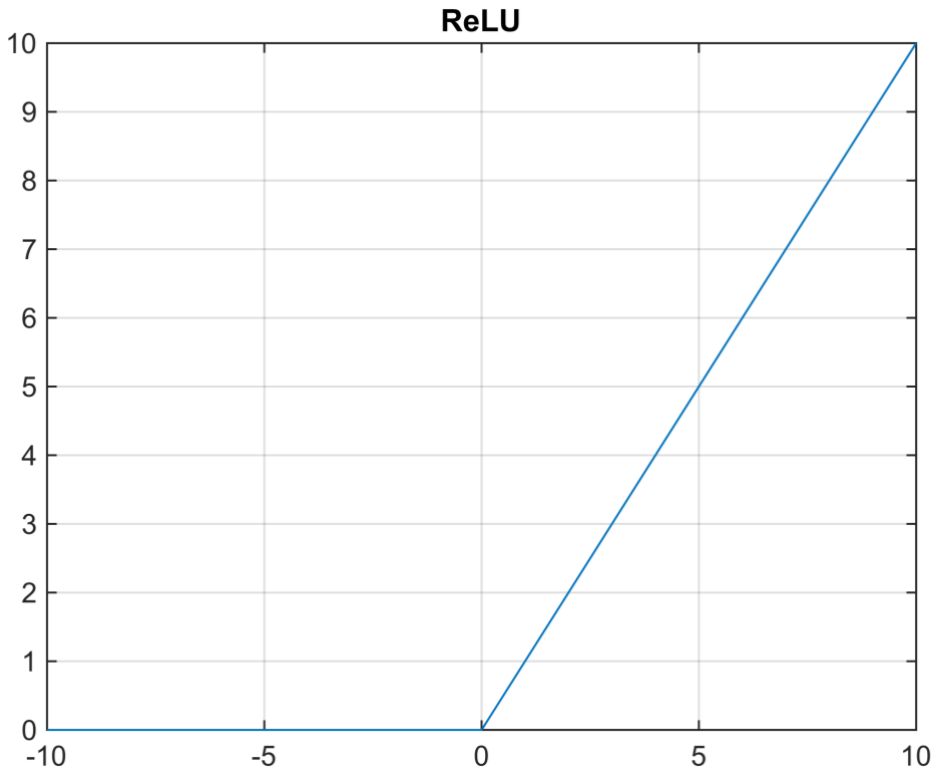

Figura 2.3: Función de activación ReLU en Neural Lab.

Las ReLUs permiten el paso de todos los valores positivos sin cambiarlos, pero todos los valores negativos los pasa a cero. Aunque existen muchas funciones de activación más recientes, la mayoría de las RNA de hoy utilizan ReLU o una de sus variantes.

Independientemente de la función de activación que se utilice, se puede visualizar una neurona individual como se muestra en la figura 2.4, que es una visualización representativa e intuitiva del comportamiento de una neurona.

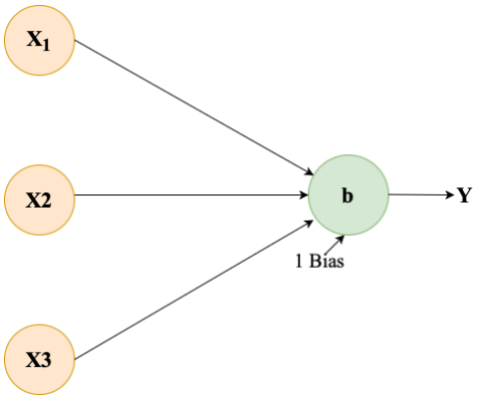

Figura 2.4: Neurona artificial.

Este diagrama muestra una neurona con tres entradas, que genera un único valor y como salida. Primero se calcula la suma ponderada de las entradas como se muestra en la ecuación (2.8), y despues se evalua mediante la función de activación.

$$z = b + w_1 x_1 + w_2 x_2 + w_3 x_3 \tag{2.8}$$

$$y = \sigma(z) \tag{2.9}$$

En la figura 2.5 se puede observar la estructura de una RNA implementando lo que es una función de activación.

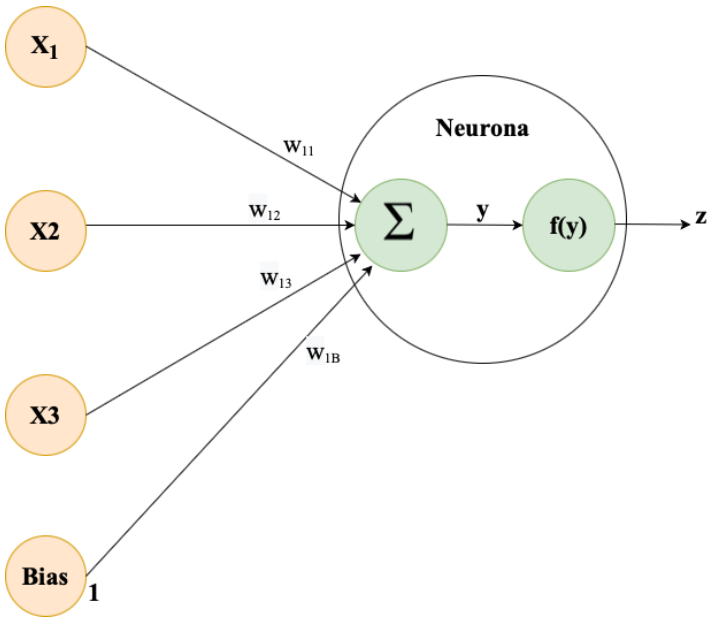

Figura 2.5. Estructura de una RNA.

2.2.4 Entrenamiento

Antes de que alguna RNA vaya a ser implementada en una actividad especifica, esta red debe de ser entrenada. El entrenamiento es una parte fundamental de las RNA ya que es cuando la red aprende la actividad deseada.

El aprendizaje de una red neuronal se realiza empleando conjuntos de datos. Estos conjuntos de datos deben estar organizados en casos de

entrenamiento. Adicionalmente, estos casos deben de representar la actividad que van a desempeñar por tal motivo se deben de tener una gran cantidad de casos.

El conjunto de entranamiento tiene dos componentes: el conjunto de datos de entrada y el conjunto de datos deseados. El conjunto de datos de entrada contiene las entradas que se le van a aplicar a la red neuronal. Por otro lado, el conjunto de datos deseados son los valores que se esperarian tuviera la red neuronal a la salida.

2.2.4.1 Métodos de Entrenamiento

Al utilizar los métodos de entrenamiento es un algoritmo computacional implementado en el entrenamiento de las RNA. Su principal objetivo es buscar que las RNA puedan establecer los pesos de las neuronas más adecuados y que ofrezcán un mejor resultado, además de que se busca minimizar el error que se puedan presentar en el entrenamiento.

Existen métodos de entrenamiento que se aplican a las RNA. Los más populares o conocidos en el software Neural Lab son los siguientes:

- Simulated annealing (Templado Simulado).
- Genetic algorithms (Algoritmos Genéticos).
- Conjugate Gradient (Gradiente Conjugado).
- Variable metric (Métrica Variable).
- Levenberg Marquardt.

A continuación, solo se explicarán los métodos empleados en la elaboración de este trabajo de tesis.

2.2.4.1.1 Templado Simulado

La analogía es la siguiente:

- Las soluciones corresponden a estados del sistema físico.
- El costo de la solución corresponde a la energía del estado.

- Se introduce un parámetro de control que corresponde a la temperatura.

El algoritmo de templado simulado se puede ver como una iteración de algoritmos.

Si se baja la temperatura de una manera se alcanza el equilibrio térmico en cada temperatura. Esto se hace mediante la generación de transiciones en cada temperatura.

Se puede demostrar que bajando suficientemente lento el parámetro asociado a la temperatura y generando suficientes transiciones en cada temperatura se puede alcanzar la configuración óptima.

En cada iteración, se genera una solución xj perteneciente al espacio de soluciones vecinas de la solución actual xi y se acepta como nueva solución actual a través de la aplicación de la siguiente probabilidad de aceptación donde $f(x)$ es el nivel de energía analizado:

$$P_T(aceptar\ x_j) = \begin{cases} 1 & Si\ f(x_i) \leq f(x_j) \\ e^{\frac{f(i)-f(j)}{T}} & Si\ f(x_i) > f(x_j) \end{cases} \quad (2.10)$$

En la figura 2.6 se muestra el diagrama de flujo del algoritmo del templado simulado para una mejor percepción del método.

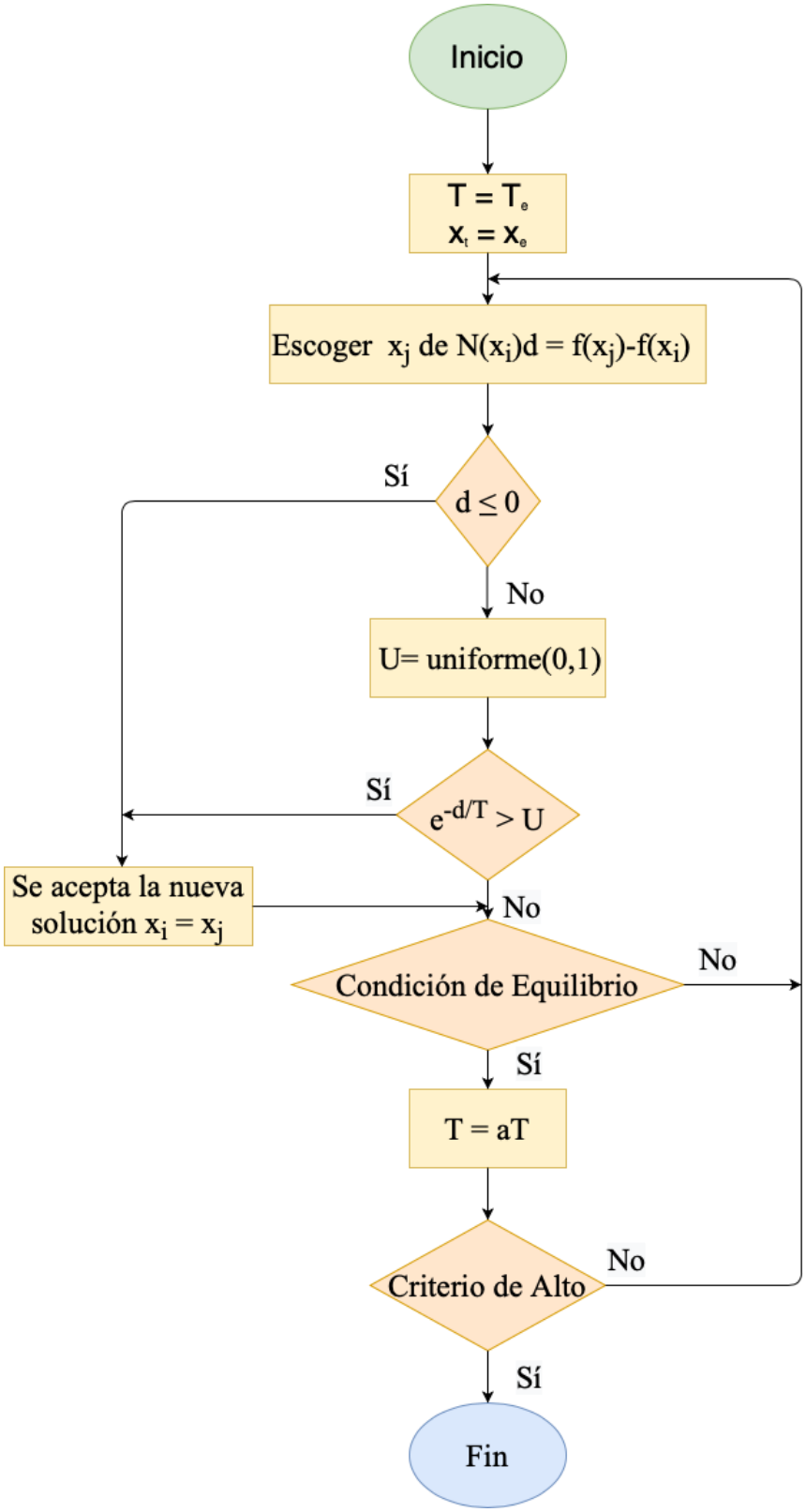

Figura 2.6. Diagrama de flujo del algoritmo templado simulado.

2.2.4.1.2 Gradiente Conjugado

El método del *Gradiente Conjugado* puede considerarse como un método intermedio entre el Descenso del Gradiente y el método de Newton. Viene motivado por el deseo de acelerar la convergencia, habitualmente lenta, obtenida con el Descenso del Gradiente, y a la vez evitar los requisitos de computación asociados a la evaluación, almacenamiento e inversión del Hessiano, como requiere el método de Newton.

En este algoritmo de entrenamiento la búsqueda se realiza a lo largo de direcciones conjugadas, lo que normalmente produce una convergencia más rápida que las direcciones obtenidas con el Descenso del Gradiente. Dos direcciones, u y v se dicen conjugadas respecto de una matriz A si $U^T\,Av\,=\,0$. En el caso que nos ocupa estas direcciones se conjugan con respecto a la matriz Hessiana.

El método del Gradiente Conjugado construye una sucesión de direcciones de entrenamiento dada por:

$$d_{i+1}\;=\;g_{i+1}+d_i\gamma_i \tag{2.11}$$

donde γ_i se denomina *parámetro conjugado*, y hay diferentes maneras de calcularlo. Dos de las más utilizadas se deben a Fletcher y Reeves, y a Polak y Ribiere. Pero sea cual sea la forma, la dirección de entrenamiento se restablece periódicamente a la negativa del gradiente para evitar la acumulación de errores en las aproximaciones.

Los valores de los parámetros de la función se calculan entonces con la forma habitual:

$$w_{i+1}=w_i+d_iv. \tag{2.12}$$

Este método ha demostrado ser más eficaz que el de Descenso del Gradiente, y como no requiere el cálculo del Hessiano también se recomienda cuando tenemos redes neuronales muy grandes.

La figura 2.7 muestra una comparativa del comportamiento del descenso del gradiente usual (verde) frente al comportamiento del Gradiente Conjugado (rojo).

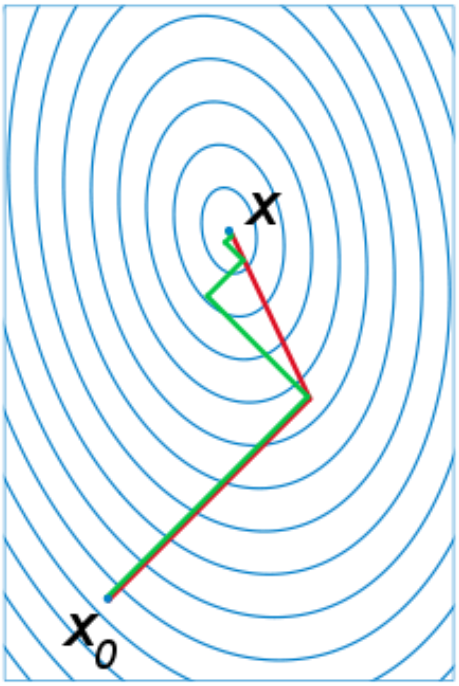

Figura 2.7. Comparativa del comportamiento del gradiente conjugado.

_____

Fernando Sancho Caparrini, http://www.cs.us.es/~fsancho/?e=165

2.2.5 Validación

Una vez realizado el entrenamiento de la red neuronal se tiene que validar el rendimiento de la misma. Normalmente la validacion se realiza utilizando otro conjunto de datos llamado conjunto de datos de validación. El conjunto de validacion es parecido al conjunto de entrenamiento. Al utilizar todo el conjunto de casos de los datos se garantizará que la red neuronal se comporte de manera similar con el conjuto de datos de entrenamiento como con el conjunto de datos de validación.

2.2.6 Secuencia de pasos para crear una red neuronal

En la figura 2.8 se muestra una serie de pasos que se pueden seguir para la creacion de una red neuronal en el software Neural Lab.

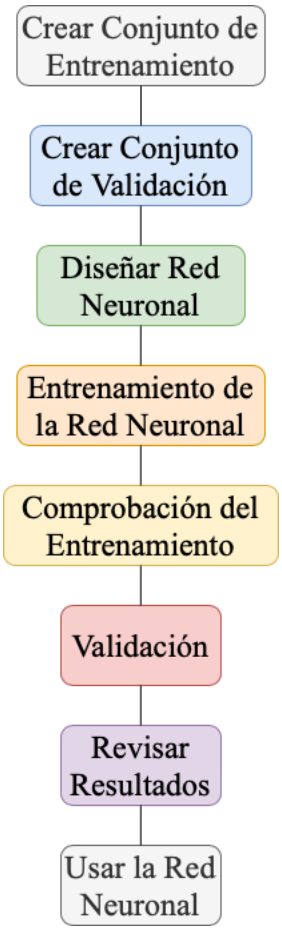

Figura 2.8. Pasos para crear una red neuronal.

2.3 Clasificadores

Clasificar un objeto consiste en asignarlo a una de las clases disponibles. Los objetos se pueden definir por una serie de características, como pueden ser el color de sus píxeles, su textura o su tamaño.

Para poder clasificar objetos es necesario definir las fronteras entre las diferentes clases. Normalmente estas fronteras se calculan mediante un proceso de entrenamiento en el que se usan las características de una serie de prototipos de ejemplo de las clases. Clasificar un objeto desconocido consiste en asignarlo

a la clase en la cual las características usadas durante el entrenamiento tienen más correspondencia con las características del objeto.

Los clasificadores se utilizan principalmente en:

- Reconocimiento de objetos.
- Control de calidad.
- Detección de cambios o defectos en objetos.
- Reconocimiento óptico de caracteres.

En la figura 2.9 se muestran los pasos para crear un clasificador.

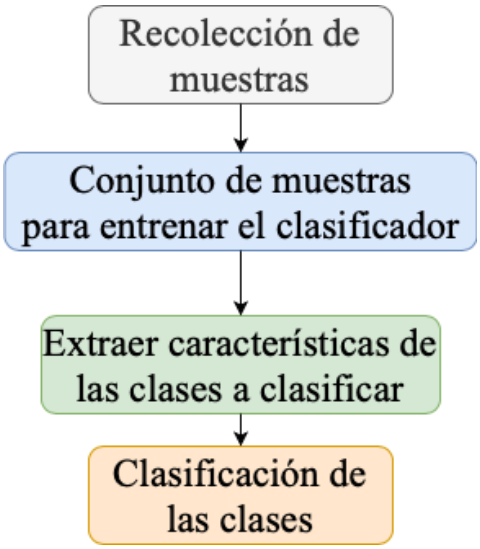

Figura 2.9. Pasos para crear un clasificador.

Existen muchos tipos de clasificadores con diferentes aplicaciones. A continuacion se mencionan algunos de los más comunes:

- Gaussian Mixture Model (GMM).
- Support Vector Machine (SVM).
- Hidden Markov Model (HMM).
- Artificial Neural Network (ANN).

2.3.1 Métricas de Evaluación de Modelos de Clasificación.

Estas métricas son fundamentales al utilizar modelos de aprendizaje automatico, ya que permiten optener una mejor persepción del desempeño que

esta teniendo el algoritmo utilizado. Existen diferentes métricas y métodos de evaluación los cuales se mencionan a continuación.

2.3.1.1 Matriz de confusión.

Es una representacion matricial de los resultados de las predicciones de cualquier prueba binaria y de multiples clases que se utilizá para describir el rendimiento del modelo sobre un conjunto de datos de prueba. En la tabla 2.1 se puede apreciar una matriz de confusión de dos clases.

	Clase 1	**Clase 2**	**Rechazo**
Clase 1	905	0	0
Clase 2	0	932	0
Rechazo	0	0	1844

Tabla 2.1. Matriz de confusión de dos clases.

En la primera línea y la primera columna de la tabla 2.1 se puede observar el número “905” el cual representa el número de datos analizados de la “clase 1” y de los cuales todos los datos fueron clasificados correctamente. Además, este dato tambien pude ser conocido como Verdadero Positivo por sus siglas en ingles (True Positive “TP”). En la línea dos y en la columna 2 se puede observar el número “932” el cual representa el número de datos analizados de la “clase 2” de los cuales todos se clasificaron de manera correcta. Adicionalmente, este número tambien puede ser conocido como Verdadero Negativo por sus siglas en ingles (True Negative “TN”). Los ceros representan el número de errores que se tengan de la clasificación. En el renglon 2 columna 1 se tiene un valor de cero este valor tambien es conocido como Falso Negativo por sus siglas en ingles (False Negative “FN”). En el renglo 1 columna 2 se tiene otro valor de cero el cual tambien puede ser conocido como Falso Positivo por sus siglas en ingles (Flase Positive “FP”). En la tercera línea columna 3 se tiene el número “1844” que representan el número de datos adicionales que utilizo el clasificador para determinar si el clasificador es confiable al darle nuevos datos que desconoce y ver el rendimiento que tiene el clasificador con datos desconocidos.

E2.3.1.2 Exactitud

Indica el número de elementos clasificados correctamente en comparación con el número total de datos. En la ecuación (2.13) se puede observar como se calcula la exactitud.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.13}$$

2.3.1.3 Precisión

Esta métrica representa el número de datos clasificados correctamente comparandolos con los datos correctos predichos por el clasificador. En la ecuación (2.14) se puede observar como se calcula la precisión.

$$Precisión = \frac{TP}{TP + FP} \tag{2.14}$$

2.3.1.4 Sensibilidad

Está métrica muestra la cantidad de datos clasificados correctamente por el modelo en funcion del número de datos correctos. En la ecuación (2.15) se puede observar como se calcula la sensibilidad.

$$Sensibilidad = \frac{TP}{TP + FN} \tag{2.15}$$

2.3.1.5 Especificidad

Esta métrica se trata de los casos negativos que el algotirmo ha clasificado correctamente, expresa si el modelo puede detectar bien una clase. En la ecuación (2.16) se puede observar como se calcula la especificidad.

$$Especificidad = \frac{TN}{TN + FP} \tag{2.16}$$

2.3.1.6 Tasa de falsos positivos

Esta métrica es la probabilidad de que se produzca un falso resultado: que se obtenga un resultado positivo cuando el valor verdadero es negativo. En la ecuacion (2.17) se puede observar como se calcula la tasa de falsos positivos.

$$Tasa\ de\ falsos\ positivos\ = \frac{FP}{FP\ +\ TN} \tag{2.17}$$

2.3.2 Support Vector Machine (SVM)

Es un algoritmo de clasificación de aprendizaje automático supervisado que se ha vuelto extremadamente popular hoy en día debido a sus resultados extremadamente eficientes.

El objetivo principal de una máquina de vector de soportes es clasificar los datos dados de forma óptima, como se muestra en la figura 2.10.

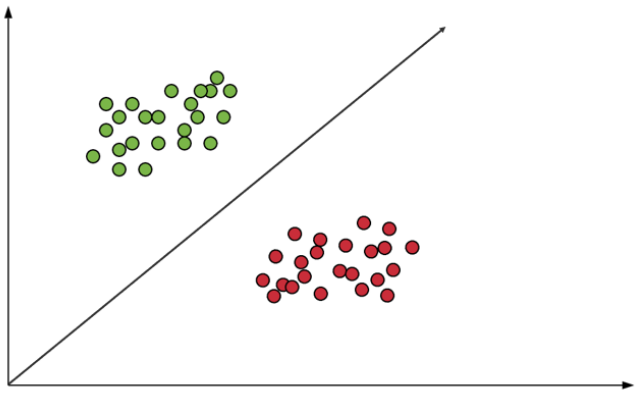

Figura 2.10. Clasificador SVM.

https://www.edureka.co/blog/support-vector-machine-in-python

Un kernel SVM básicamente agrega más dimensiones a un espacio de baja dimensión para facilitar la clasificación de los datos. Una máquina vectorial de soporte es implementada en la práctica por un kernel. La ventaja del kernel ayuda a tener un clasificador menos complejo.

- **Kernel lineal**: un kernel lineal se puede utilizar como un producto de punto normal entre dos observaciones dadas cualesquiera. El producto entre los

dos vectores es la suma de la multiplicación de cada par de valores de entrada.

- **Kernel polinómico:** es una forma bastante generalizada del kernel lineal. Puede distinguir el espacio de entrada curvo o no lineal.
- **Kernel de función de base radial:** el kernel de función de base radial se utiliza comúnmente en la clasificación SVM, puede mapear el espacio en dimensiones infinitas.

2.4 Transformada de Fourier

La transformada de Fourier de una señal unidimensional o función continua $f(x)$ es una transformación de dicha señal que nos permite calcular la contribucion de cada valor de frecuencia a la formación de la señal, en la ecuación 2.16 se muestra la representación matemática.

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx \tag{2.16}$$

Donde $i = \sqrt{-1}, e^{(-i2\pi ux)} = cos(2\pi ux) - isen(2\pi ux)$ y la variable “u” que aparece en la función F(u) representa a las frecuencias. Puede demostrarse ademas que esta transformación tiene inversa, es decir, dada la funcion F(u) podemos calcular la función f(x), la ecuación 2.12 muestra dicha transformación.

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{(i2\pi ux)} du \tag{2.17}$$

Estas dos funciones f(x) y F(u) se denominan par de transformadas de Fourier. En general las funciones deben de cumplir las condiciones de las expresiones anteriores.

2.5 Motores Eléctricos de Inducción

El motor de inducción es la maquina eléctrica más utilizada en la actualidad, es utilizada en el 70% de las aplicaciones industriales y estos consumen más del 50% de la energía generada en las naciones industrializadas. Alguna falla imprevista de un motor de inducción puede ocasionar el paro de producción de una empresa, lo que tendría una consecuencia enorme de pérdidas de producción, éstas pueden llegar a ser de miles de dólares por minuto.

En la figura 2.11 se muestran las partes que componen un motor de inducción.

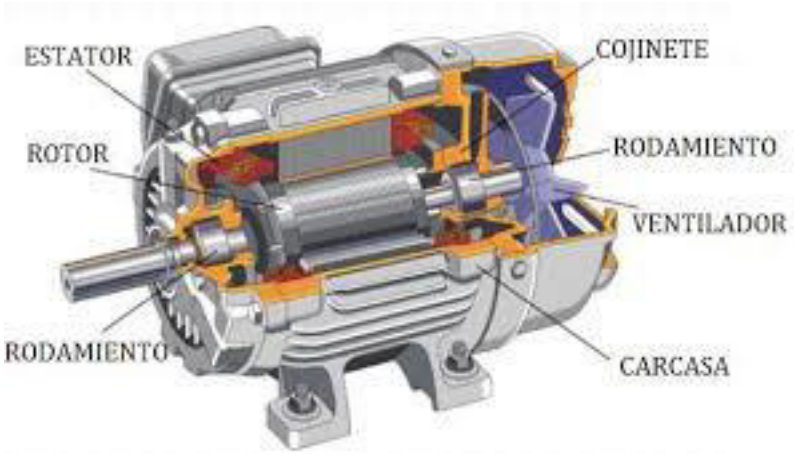

Figura 2.11. Partes de un motor eléctrico de inducción.

_____

http://www.ticgalicia.com/2016/06/el-motor-de-jaula-de-ardilla-la-base-de.html

2.5.1 Fallas en motores Eléctricos

Como sabemos los motores eléctricos son la principal máquina eléctrica utilizada en las industrias. Por lo que se tienen que mantener en un monitoreo constante para prevenir cualquier tipo de falla que puedan presentar. Como se conoce existen fallas que van desde el área eléctrica hasta el área mecánica como las principales. Dichas fallas se presentan en equipos trifásicos y monofásicos de inducción las cuales se mencionan a continuación.

- Chumaceras o cojines desgastados.

- Interrupción de fases.
- Sobrecargas.
- Corto Circuito.
- Conexiones internas erróneas.
- Contactos a tierra de los devanados.
- Cojines excesivamente apretados.
- Tapas mal montadas.
- Eje torcido.
- Barras del rotor flojas o rotas.
- Condensador defectuoso (en motores monofásicos con condensador).
- Interrupción en el devanado de arranque (motores monofásicos).

A continuación se definen algunas de las fallas más comunes de los motores de inducción.

2.5.1.1 Sobrecargas Eléctricas

La sobrecarga eléctrica es causada por un flujo de corriente excesivo dentro de los devanados del motor, que excede la corriente de diseño que el motor puede transportar de manera eficiente y segura. Esto puede ser causado por un bajo voltaje de suministro, lo que resulta en que el motor consuma más corriente. También puede ser el resultado de conductores en cortocircuito o un suministro de voltaje excesivo.

La sobrecarga eléctrica se puede prevenir instalando una protección contra sobrecorriente que detecte sobrecorriente e interrumpa el suministro.

2.5.1.2 Baja Resistencia de Aislamiento

Es la falla más común de un motor eléctrico, la baja resistencia es causada por la degradación del aislamiento de los devanados debido a las condiciones asociadas a sobrecalentamiento, corrosión o daños físicos. Por lo que nos llevaría a un aislamiento insuficiente entre los conductores o los devanados del motor, lo que causaría fugas, cortocircuitos y una prominente falla del motor.

Una solución a esta falla seria realizar mantenimientos preventivos o predictivos, para detectar signos de desgaste en el aislamiento, y reemplazar el aislamiento antes de que se presente la baja resistencia.

2.5.1.3 Sobrecalentamiento

La mayoría de los fallos en el aislamiento de los motores eléctricos ocurre debido al sobrecalentamiento, el cual puede ser causado por una mala calidad de la energía o un entorno operativo de altas temperaturas. Es fundamental que el motor se mantenga lo más frío posible.

2.5.1.4 Contaminación

La contaminación por polvo, suciedad y productos químicos es una de las principales causas de fallo del motor eléctrico. Los cuerpos extraños que se abren paso dentro del motor y pueden afectar a las pistas de rodadura y los cojinetes, lo que genera altos niveles de vibración y desgaste. También puede bloquear el ventilador de enfriamiento, lo que limita la capacidad del motor para regular su temperatura y aumenta la probabilidad de sobrecalentamiento.

2.5.1.5 Vibraciones en motores eléctricos

La vibración puede conducir a muchos problemas con el motor, y eventualmente puede causar que el motor falle. La vibración a menudo es causada por la posición del motor sobre una superficie irregular o inestable. Sin embargo, la vibración también puede ser el resultado de un problema con el motor, como cojinetes flojos, desalineación o corrosión. A continuación, se mensionan algunas recomendaciones para evitar fallas en motores eléctricos:

- Limpieza General.
- Condiciones eléctricas.
- Ventilación apropiada.
- Alineamiento con la carga.

- Lubricación apropiada, desgaste de las chumaceras del motor y de la carga.
- Deterioro del aislamiento en los devanados.
- Condición del motor.
- Desgaste en los switches o interruptores.
- Deterioro de los capacitores (si los tiene).

2.6 Motor Current Signature Analysis (MCSA)

La técnica de análisis MCSA permite detectar las fallas más comunes de los motores de inducción, debido a que es un método no invasivo. Además, esta técnica permite detectar fallos antes de que puedan ser detectados mediante análisis de ruido acústico y vibraciones.

Esta técnica se centra en la monitorización de la corriente del estator. Los fallos que presenta un motor modifican el campo magnético, dependiendo de la falla, éste cambio será de mayor o menor percepción. Esta técnica consiste en analizar las corrientes del estator del motor en el domino de la frecuencia.

La detección temprana de fallas eléctricas en motores de induccion permite realizar un mantenimiento preventivo antes de que el motor sufra una falla grave.

2.7 Resumen del segundo capítulo

En este capítulo se mencionó la teoria de las redes neuronales desde sus inicios. Ademas, se mensiona como se diseñan las redes neuronales. De igual manera se mensionaron los tipos de entrenamientos que se le pueden dar a una red neuronal.

Se mencionó que son los clasificadores y los clasificadores más populares que se encuentran hasta el momento. Ademas, se habla del clasificador SVM y como diseñarlo.

Finalmente se habla de los motores de inducción y de las fallas más comunes que pueden presentar. Además, se habla de la técnica de análisis MCSA para la detección de fallas eléctricas en motores de induccion.

Capítulo 3

Metodología

En este capítulo se muestra el desarrollo y simulación de las fallas eléctricas en motores de inducción. Consecuentemente se utilizarán dos técnicas del área de inteligencia artificial para la clasificación de las fallas del motor, incluyendo las RNA y el clasificador SVM.

En la figura 3.1 se puede observar a grandes rasgos de lo que se hablará en este capítulo.

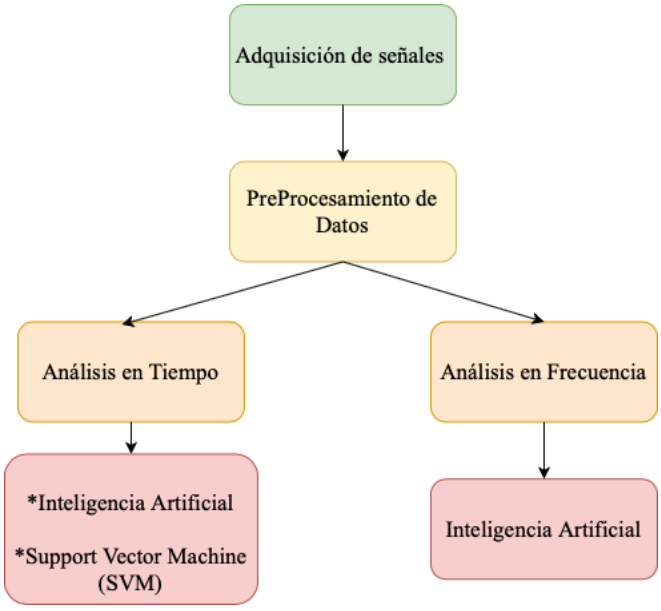

Figura 3.1. Metodología.

Capítulo 3

3.1 Captura de datos

En esta sección se mostrará el proceso para la recolección de muestras utilizadas en la clasificación de fallas de motores eléctricos de inducción.

3.1.1 Banco de pruebas

La recolección de muestras para la clasificación de fallas se elaboró por medio de un banco de pruebas. Este banco de pruebas permite analizar la falla de corto circuito de manera evolutiva, esto es, por porcentajes de falla de corto circuito.

Las muestras de análisis para la falla de barras rotas se obtuvieron con el mismo banco de pruebas, pero en este caso, analizando el motor con dos barras rotas. En la figura 3.2 se observa el banco de pruebas para la recolección de muestras.

Figura 3.2. Banco de Pruebas.

3.1.2 Osciloscopio

Para la adquisición de datos se utilizó un osciloscopio Fluke 190 Scope meter. Este osciloscopio permite obtener datos de cada una de las fases del motor de manera individual guardando todos los datos.

En la figura 3.3 se puede observar el osciloscopio utilizado para la adquisición de datos.

Figura 3.3. Osciloscopio Fluke 190 Scope Meter.

Imagen obtenida de fluke.com

El osciloscopio cuenta con un software (flukeviewforms) que permite visualizar cada una de las señales obtenidas por el analizador. Además de que este software permite exportar las señales a formatos de Microsoft Excel.

En la figura 3.4 se muestra la señal sana del motor en el software flukeviewforms.

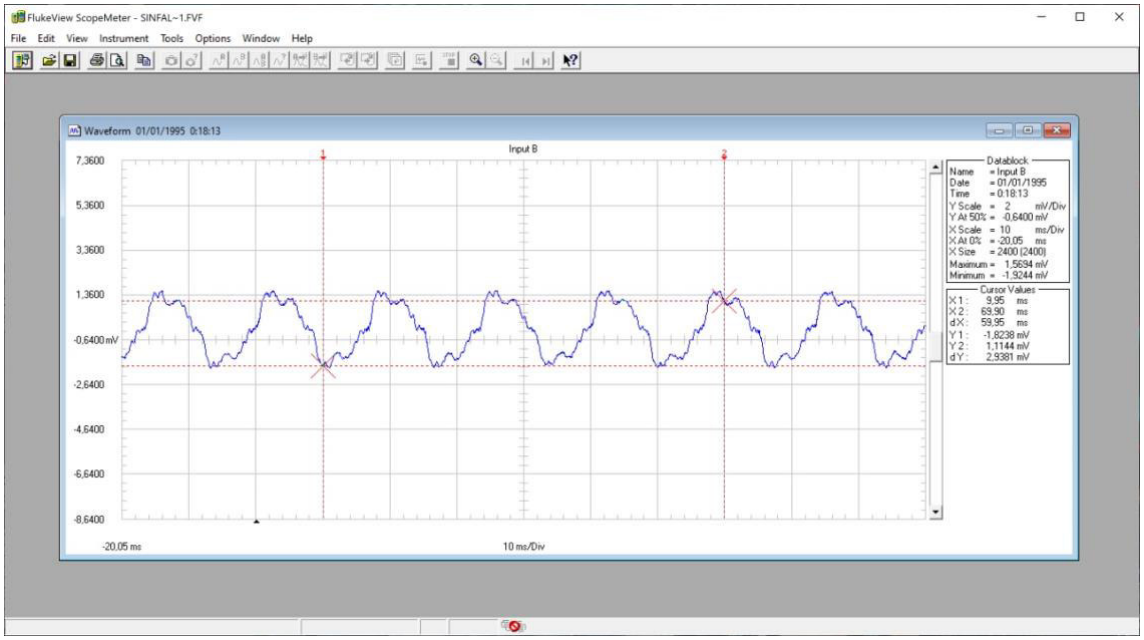

Figura 3.4. Señal sana del motor.

En la figura 3.5 se muestra la señal de falla de corto circuito al 40% en el software flukeviewforms.

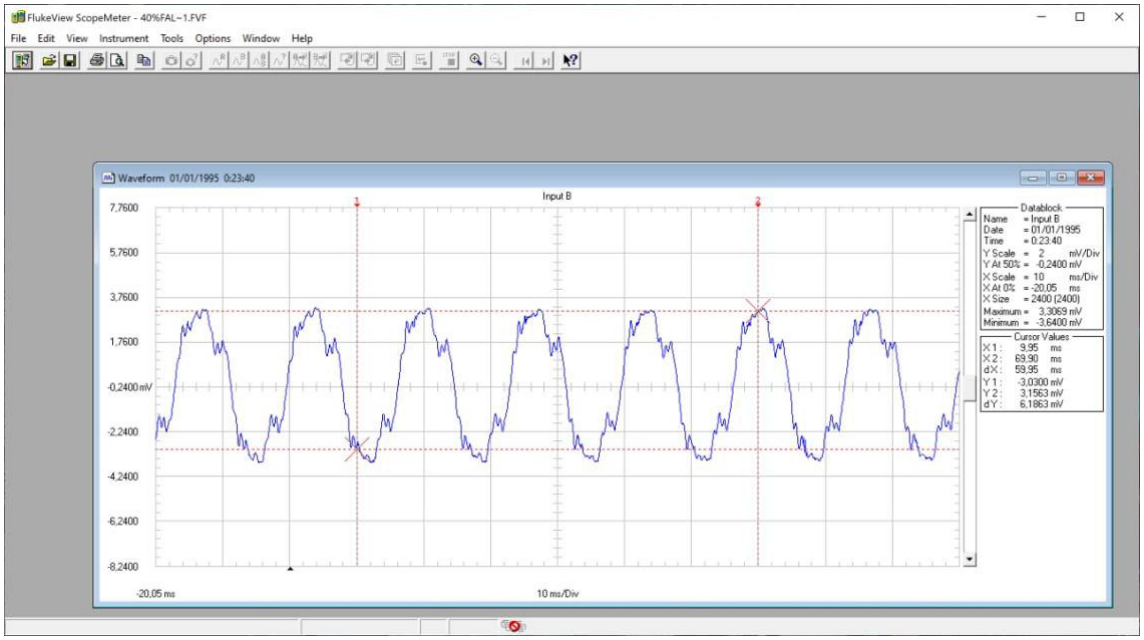

Figura 3.5. Señal de falla de corto circuito al 40%.

En la figura 3.6 se muestra la señal de falla de barras rotas en el software flukeviewforms.

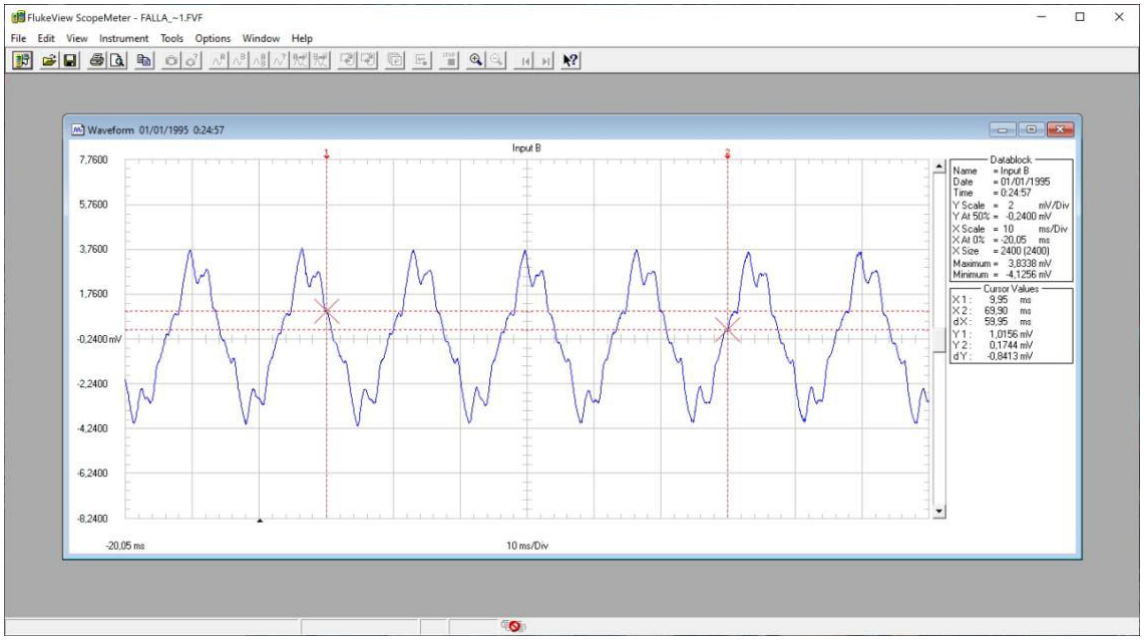

Figura 3.6. Señal de falla de barras rotas.

3.2. Preprocesamiento de los datos

En esta sección se explica como se desarrolló el preprocesamiento de los datos obtenidos por el osciloscopio.

Primero, cada una de las señales se convirtió a valores separados por coma (comma separated values, CSV). Este tipo de formato es muy simple y alta mente compatible ya que permite obtener cada uno de los separados por comas en un archivo de texto. Por lo que este representa un formato conveniente para la lectura y el análisis por cualquier otro software.

3.2.1. Creación del conjunto de datos

Para crear cada uno de los conjuntos de datos se utilizó el software de Matlab. Específicamente, el software flukeviewforms generó un archivo de datos con solamente una columna. Sin embargo, el software de simulación para redes neuronales requiere varias columnas. Así, el conjunto de datos se creó incrementando el número de columnas debido a que en el archivo original solamente había una columna.

Se llevó a cabo, en cada uno de los casos, un incremento de columnas en valores que fueran de interés analizar. Que en este caso los valores fueron 32, 64 y 128, en el caso del análisis en frecuencia. Además, se obtuvó un número de muestras de 4672 en el caso del análisis de 128 entradas. Sin embargo, para el análisis en tiempo se utilizaron 100 columnas y 4698 muestras. En la figura 3.7 se puede observar a grandes rasgos como se realizó el preprocesamiento.

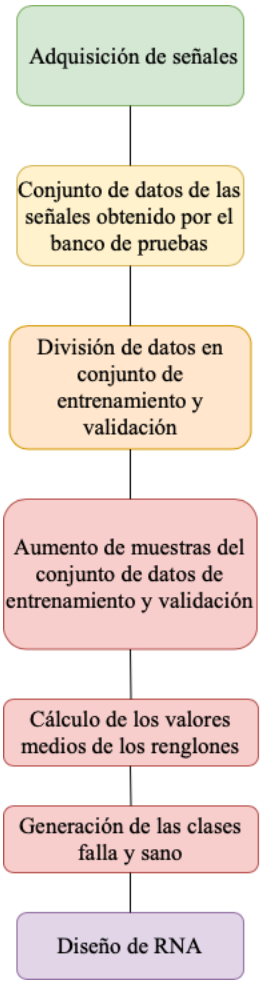

Figura 3.7. Preprocesamiento del conjunto de datos.

En el siguiente código de Matlab se puede observar como se llevó a cabo la segmentación de muestras.

```
numRenglones = size(muestras,1) - numCols - 1;
for inicio=1:numRenglones
   col = 1;
    for i = inicio:inicio+numCols - 1
       matMuestras(inicio,col) = muestras(i);
```

```
            col=col+1;
        end
    end
```

Código 3.1. Incrementó de columnas.

En este código se utilizaron dos ciclos “for”. El primer ciclo se encarga de mover el número de renglones. Mientras tanto el segundo ciclo se encarga de moverse entre columnas. La variable “matMuestras” guarda los resultados de los ciclos.

Cuando se obtuvieron las divisiones del conjunto de datos de entrenamiento y validación se procedió al cálculo de los valores medios de cada uno de los renglones para generar el conjunto de datos de entrada, “dataSetInput”. Los cuales representan los datos de entrada para llevar a cabo la clasificación de los datos sanos y de falla. Como se muestra en el código 3.2 de Matlab.

```
%% Media de todas las muestras
mediaRenglon = mean(matMuestras, 2);
```

Código 3.2. Cálculo de los valores medios de los renglones.

En este código la variable “mediaRenglon” calcula el valor medio del conjunto de datos creado. Consecuentemente la variable “mediaRenglon” es el valor sano que se va a utilizar para la generación de las clases falla y sano.

En el código 3.3 de Matlab. Se puede observar como se hizo la comparación para la generación de los ceros y unos para el conjunto de datos deseados a la salida del clasificador, “dataSetTarget”.

```
%% Calcular las clases falla y sana
claseFalla = mediaRenglon > umbral;
claseSana = claseFalla <= 0;
%% Calcular y guardar el trainSetTarget
target = [claseFalla, claseSana];
trainSetTarget = double(target);
```

Código 3.3. Generación de las clases falla y sano.

En este código, la variable “claseFalla” se calcula utilizando la variable “media_Renglon” la cual incluye los valores medios de la señal. Así, cada valor de la variable “claseFalla” vale uno cuando el valor medio es mayor a un valor de umbral (-0.00015175). De igual modo la variable “claseFalla” vale cero cuando el valor es menor o igual al umbral. En la siguiente línea, la variable “claseSana” se calcula invirtiendo los valores de la variable “claseFalla”. Esto es, los valores de cero se convierten en uno. Finalmente, el conjunto de datos deseados a la salida del clasificador (trainSetTarget) se construye concatenando las columnas de “claseFalla” y “claseSana”.

Con los datos ya preprocesados, se puede continuar con el diseño de la red neuronal, además, de hacer la debida separación de los datos de entrenamiento y de validación; para clasificar las fallas en el dominio del tiempo y la frecuencia. Y finalmente diseñar el clasificador SVM.

3.2.2. Datos de entrenamiento y validación

Con los datos previamente preprocesados y divididos en entrenamiento y validación, se garantiza que los resultados de la clasificación de la red sean confiables. Y con ello evitar la realización de la validación cruzada en el caso de la red neuronal. Al emplear el 80% de los datos para el entrenamiento y el 20% de los datos para la validación se puede comprobar si la clasificación de la RNA es confiable. Adicionalmente, como se lleva a cabo una rectificación de entrenamiento y validación con los valores de rechazo se puede comprobar que la clasificación es correcta. De tal forma se puede no hacer la validación cruzada en este caso particular al tener un alto número de muestras.

En el código 3.4 de Neural Lab, se muestra como se carga el archivo con el conjunto de datos.

```
Matrix dataSetInput;
dataSetInput.Load();
```

```
Matrix dataSetTarget;
dataSetTarget.Load();
```

Código 3.4. Carga de datos.

En el código 3.5 de Neural Lab se muestra como se tomarán los datos para el conjunto de entrenamiento y validación.

```
int totalCases = dataSetInput.GetRowCount();
int trainCount = toint (0.8*totalCases); //80% de los casos
```

Código 3.5. Generación de casos.

En este código se puede observar como se utilizarán el 80% de los casos de entrenamiento.

En el código 3.6 de Neural Lab se muestra como se generan los índices.

```
//_____________________________Generación de los indices aleatorios
Vector trainCases;
trainCases.CreateRandomSet(trainCount, totalCases-1);
```

Código 3.6. Generación de indices aleatorios.

En este código se puede observar como la variable “trainCases” genera aleatoriamente los índices de casos para el entrenamiento.

En el código 3.7 de Neural Lab se muestra como se genera el conjunto de entrenamiento.

```
//_____________________________ Conjunto del Entrenamiento
Matrix trainSetInput = dataSetInput.GetRows(trainCases);
Matrix trainSetTarget = dataSetTarget.GetRows(trainCases);
//_____________________Guardar conjunto del entrenamiento
trainSetInput.Save();
trainSetTarget.Save();
```

Código 3.7. Generación del conjunto de entrenamiento.

En este código se puede observar como se generan las variables “trainSetInput” y “trainSetTarget”, que representan los datos de entrada y de la salida deseada de la red neuronal respectivamente. Con el 80% de los datos totales para el entrenamiento.

En el código 3.8 de Neural Lab se muestra como se genera el conjunto de datos para la validación.

```
//_____________________________Creación de la Validación
Matrix validSetInput = dataSetInput;
validSetInput.DeleteRows(trainCases);
Matrix validSetTarget = dataSetTarget;
validSetTarget.DeleteRows(trainCases);
//_____________________________Guardar la Validación
validSetInput.Save();
validSetTarget.Save();
```

Código 3.8. Generación del conjunto de validación.

En este código se puede observar como se generan las variables “validSetInput” y “validSetTarget”, que representan los datos de entrada y de la salida deseada de la red neuronal respectivamente. Con el 20% de los datos totales para la validación.

3.3 Arquitectura de la red neuronal

En la figura 3.8 se puede observar la estructura que tendrá la red neuronal para llevar acabo la clasificación de las fallas del motor.

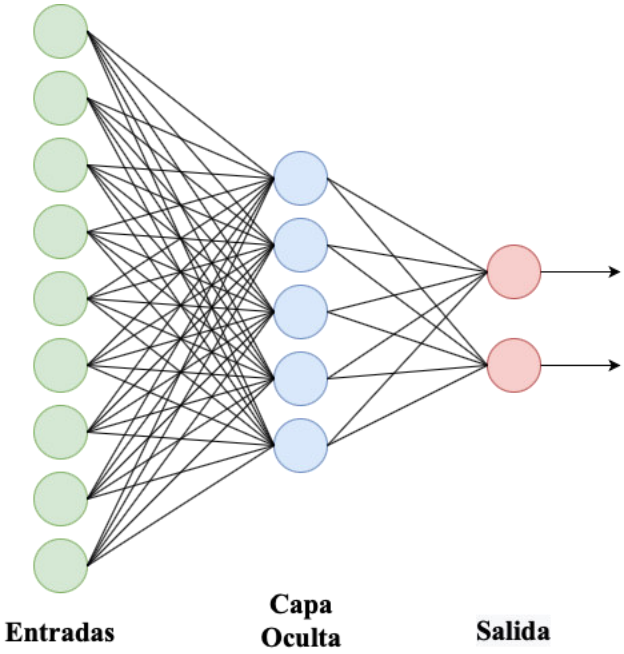

Figura 3.8. Estructura de la red neuronal.

Para la clasificación de cada una de las fallas, se tiene que tomar en cuenta la estructura de la red. Dicha estructura se diseñará en el software Neural Lab.

3.3.1 Número de entradas

Debido al uso de la transformada rápida de Fourier, el número de entradas en la red neuronal fue siempre potencias de dos. Al utilizar este número de entradas se facilita el análisis en frecuencia. El uso de la transformada de Fourier permite que la red neuronal clasifique de una manera sencilla y eficiente.

3.3.2 Número de neuronas en la capa oculta

El número de neuronas en la capa oculta se determinará dependiendo de los resultados que se obtengan de las matrices de confusión y validación. Estas matrices se calcularán para las clasificaciones de las fallas de corto circuito y de barras rotas. Con los valores de las matrices se podrá determinar si el número de neuronas es el indicado en la capa oculta o si se llega a un sobre entrenamiento de la RNA que pueda influir en el desempeño de la misma.

3.3.3 Análisis en el dominio del tiempo

Con este análisis se pondrá a prueba la red neuronal para observar los valores que se obtienen del clasificador con los datos en el dominio del tiempo. Además de que se podrán obtener las matrices de confusión con un cierto número de neuronas que se utilizarán en la clasificación.

3.3.3.1 Falla de Corto Circuito al 10%

En este porcentaje de la falla se utilizarán 100 entradas, 8 neuronas en la capa oculta y 2 salidas, para la creación de la red neuronal, como se muestra en el código 3.9 de Neural Lab.

```
//_____________________________ 1. Network setup
LayerNet net;
net.Create(100, 8, 0, 2); // entradas,neuronas en la capa oculta
        1,neuronas en la capa oculta 2 , salidas
```

Código 3.9. Creación de la red neuronal.

En este código se crea la red neuronal. En la línea número dos se fija el número de entradas, neuronas y salidas que tendrá la red neuronal.

En el código 3.10 de Neural Lab se pueden observar los valores de los escaladores que se utilizaron en la entrada y la salida para la red neuronal.

```
Matrix trainSetTarget;
trainSetTarget.Load();
for(i=0; i<100; i++)
{
        net.SetInScaler(i, -0.00204, 0.00174);
}
//_____________________________ 3. Output scaling
net.SetOutScaler(0, 0.0, 1.0);
net.SetOutScaler(1, 0.0, 1.0);
```

Código 3.10. Valores de los escaladores de la entrada y la salida.

La Variable “trainSetTarget” contiene los datos de entrenamiento que se utilizarán para la clasificación. En el ciclo “for” se observan los rangos máximos y mínimos de los valores de entrada del clasificador (-0.00204, 0.00174). La variable “net.SetOutScaler” muestra los valores de cero y uno deseados para la salida del clasificador.

En el entrenamiento de la red neuronal se utilizará el templado simulado (simulated annealing), en el código 3.11 de Neural Lab, se pueden observar los parámetros utilizados para el entrenamiento.

```
//_____________ 5. Train using simulated annealing
net.TrainSimAnneal(
        20, //Number of Temperatures
        20, //Number Iterations
        150, //Initial Temperature
        0.001, //Final Temperature
        true, //Is Cooling Schedule Linear?
        4, //Number of Cycles
        1.0e-12 //Goal (desired mse)
);
```

Código 3.11. Entrenamiento utilizando Templado Simulado.

En este código se pueden observar los parámetros de entrenamiento usando templado simulado: número de temperaturas, el número de iteraciones, la temperatura inicial, el valor de la temperatura final, el número de ciclos y el valor “mse” deseado.

También se entrenará la red utilizando el gradiente conjugado para garantizar que la clasificación sea la mas óptima. En el código 3.12 de Neural Lab se puede observar los valores de entrenamiento.

```
//_____________ 6. Train using conjugate gradient
```

```
net.TrainConjGrad(2000, 1.0e-20);
```

Código 3.12. Entrenamiento utilizando Gradiente Conjugado.

En el código 3.13 de Neural Lab se puede observar como se llevo a cabo la comprobación del entrenamiento.

```
CheckTrain.lab
//_____________________________1. Load the training set
Matrix trainSetInput;
trainSetInput.Load();
Matrix trainSetTarget;
trainSetTarget.Load();
//_____________________________2. Load the network
LayerNet net;
net.Load();
//_____________________________________________3. Run
Matrix output = net.Run(trainSetInput);
//_________________4. Compute the confusion matrix
Matrix trainConf = ConfusionMatrix(output, trainSetTarget, 0.5);
trainConf.Save();
//_________________5. Compute the number of errors
int numErrors = toint(trainConf.GetSum()) -
toint(trainConf.GetDiagonalSum());
```

Código 3.13 Comprobación del entrenamiento.

En este código se puede observar como en la parte incial del código se cargan los conjuntos de datos de entrenamiento. En la parte número dos se carga la red neuronal previamente entrenada en los códigos 3.11 y 3.12. En la parte número tres se aprecia como se realiza la evaluación de la red neuronal con los datos de entrada (trainSetInput). En la parte número cuatro se crea la matriz de confusión con los datos de salida deseados (trainSetTarget). Finalmente se calculan el número de errores del entrenamiento.

En el código 3.14 de Neural Lab se puede observar como se llevo a cabo la validación del entrenamiento.

```
Validation.lab
//_____________________1. Load the validation set
Matrix validSetInput;
validSetInput.Load();
Matrix validSetTarget;
validSetTarget.Load();
//_____________________________2. Load the network
LayerNet net;
net.Load();
//_____________________________ 3. Run
Matrix output = net.Run(validSetInput);
//_________________4. Compute the confusion matrix
Matrix validConf = ConfusionMatrix(output, validSetTarget, 0.5);
validConf.Save();
//_________________5. Compute the number of errors
int numErrors = toint(validConf.GetSum()) -
toint(validConf.GetDiagonalSum());
```

Código 3.14 Validación de la red neuronal.

Este código es idéntico al código 3.13 con la única diferencia que utiliza los conjuntos de datos de validación.

3.3.3.1.2 Falla de Corto Circuito al 20%, 30%, 40% y Falla de Barras rotas

Para cada progresión de falla de corto circuito y la falla de barras rotas, se respeta la arquitectura y la metodología descritas en las secciones 3.3 y 3.3.3.1.1. En estas pruebas los únicos cambios que se pueden observar son el número de neuronas utilizadas y los valores de los escaladores de entrada.

3.3.4 Análisis en el dominio de la frecuencia

Se analizó el comportamiento de la red neuronal empleando la transformada rápida de Fourier a cada uno de los datos de las fallas. Además de que se analizaron con que número de entradas (32, 64, 128) se tienen mejores resultados en el clasificador.

3.3.4.1 Creación del Conjunto de Datos

La creación del conjunto de datos sigue la misma metodología como se explicó en la sección 3.2.1.

3.3.4.2 Datos de Entrenamiento y Validación

Con los datos previamente Preprocesados, se realizó la transformada rápida de Fourier a todos los datos de falla.

En el código 3.15 de Neural Lab se muestra como se efectuó la transformada rápida de Fourier.

```
Matrix dataSetInput;
dataSetInput = realfft(dataSetInput);
dataSetInput.Save();
```

Código 3.15. Transformada rápida de Fourier de los datos de falla.

En el código 3.15 de Neural Lab se puede observar como en la variable “dataSetInput” se almacena el resultado del cálculo de la transformada rápida de Fourier de esta misma variable. Esta variable contiene inicialmente los valores en el dominio del tiempo, pero después del cálculo de la transformada rápida de Fourier, esta misma variable almacena las señales en el dominio de la frecuencia.

Con los datos previamente transformados al dominio de la frecuencia se dividirán en datos de entrenamiento y validación como se mostró en la sección 3.2.2. El entrenamiento de la red neuronal se llevó de la misma manera como se ilustró en la sección 3.2.2. De igual forma se realizó la verificación del entrenamiento y la validación en los códigos “CheckTrain” y “Validation” para cada una de las fallas mostradas en la sección 3.3.3.

3.3.4.3 Falla de Corto Circuito y Barras Rotas

Estas pruebas se realizarán con diferentes números de entradas (32, 64, 128). Cada uno de los valores de las entradas determinará el número de neuronas adecuado para la clasificación.

Todas las simulaciones de la red en frecuencia respetarán la arquitectura y metodología de las simulaciones en el dominio del tiempo de la sección 3.3.3. Con la única diferencia del cambio de número de entradas que se menciona en el código 3.5. Dicho número de entradas son los mencionados anteriormente.

3.4 Support Vector Machine (SVM)

Este clasificador es un algoritmo de aprendizaje supervisado. Además, es un clasificador muy popular por los resultados eficientes que se obtienen. En el diseño de la red neuronal se utilizó el software Neural Lab. Pero las simulaciones del SVM se implementaron en el lenguaje de Python.

3.4.1 Creación del Conjunto de Datos

La creación del conjunto de datos en SVM sigue la misma metodología que se explicó en la sección 3.2.1. Sin embargo, este tipo de clasificador no requiere de una división de datos de entrenamiento y validación. El clasificador utiliza todo el conjunto de datos para el entrenamiento.

3.4.2 Creación de las clases de falla y sana

En el código 3.16 de Python se puede observar como se crearon las clases de las señales.

```
falla = falla.to_numpy()
sizefalla = len(falla)
falla1 = np.zeros(sizefalla)
for i in range(sizefalla):
    if falla[i][0] == 1 or falla[i][1] == 1: falla1 [i] = 1
```

Código 3.16. Etiquetas de falla y sana.

La variable “falla” contiene los datos de la señal en una matriz. La variable “sizefalla” contiene la longitud de la variable “falla” para determinar el número de caracteres de una cadena. En la variable “falla1” se almacena un arreglo donde se tiene un vector de ceros con la longitud de la variable “sizefalla”. Dentro del ciclo “*for”* se usa el comando “if” para determinar la clase correcta. Asi, la clase vale uno cuando se tiene un valor de falla, el cual se puede originar por dos diferentes valores como se ve en el código.

3.4.3 División de datos de entrenamiento y pruebas

En el código 3.17 de Python se puede observar como se dividieron los datos para la clasificación.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(sana,falla1,test_size=
0.2,random_state=109)
```

Código 3.17. División de datos de entrenamiento y pruebas.

Los datos se dividieron usando la función “*train_test_split()”.* Se necesitan tres parámetros para obtener la clasificación de las señales. Estos

parámetros son necesarios para entrenar el modelo (sana), el conjunto de datos a analizar (falla1) y el tamaño del conjunto de pruebas (random_state=109).

3.4.4 Modelo del SVM

En el código 3.18 de Python se puede observar como se creó el modelo del clasificador SVM.

```
#Modelo del SVM
from sklearn import svm
#clasificador
cls = svm.SVC(kernel = "poly") #linear, poly, rbf
#Entrenamiento del modelo
cls.fit(X_train, y_train)
#Predicción de la respuesta
pred = cls.predict(X_test)
```

Código 3.18. Modelo del SVM.

Para la generación del modelo primeramente se tuvo que utilizar el módulo SVM de "*sklearn"* para la creación del clasificador SVM "SVC()" pasando el argumento del *kernel* como el núcleo (lineal, polinomial o radial). Posteriormente se entrena el conjunto de datos usando "*set()".* Finalmente se hacen las predicciones del modelo utilizando la función "*predict()"*.

3.4.5 Evaluación del desempeño del modelo

En el código 3.19 de Python se puede observar el cálculo de las métricas del clasificador o modelo.

```
from sklearn import metrics
#Accuracy
print("accuracy:", metrics.accuracy_score(y_test, y_pred=pred))
#Precision score
```

```
print("precision:", metrics.precision_score(y_test,y_pred=pred))
#recall score
print("recall", metrics.recall_score(y_test,y_pred=pred))
print(metrics.classification_report(y_test, y_pred=pred))
```

Código 3.19. Cálculo de métricas del clasificador.

En el código 3.19 de Python se puede predecir la precisión que tiene el clasificador. Así que se cálculo la exactitud de predicciones positivas que fueron correctas, la precisión del porcentaje de casos positivos y la sensibilidad de casos positivos detectados correctamente para evaluar el modelo.

3.4.6 Validación cruzada

La validación cruzada es una técnica de análisis que permite determinar si algún modelo o clasificador es confiable.

En el código 3.20 de Python se puede observar la validación cruzada del modelo implementado para la detección de fallas.

```
#Cross validation
X,y=make_classification(n_samples=1000,n_features=20,
n_informative=15, n_redundant=5, random_state=1)
cv = KFold(n_splits=10, random_state=1, shuffle=True)
# creación del modelo
model = LogisticRegression()
# evaluación del modelo
scores=cross_val_score(model,X,y,scoring='accuracy',cv=cv,n_jobs=
-1)
# reporte
Print ('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

Código 3.20. Validación Cruzada.

La función "*make_classifcation()*" se utilizó para crear un conjunto de datos de clasificación binaria sintética. En este caso, se evaluo el modelo

utilizando "*LogisticRegression"* y se utilizó la clase "*KFold"* para realizar la validación cruzada. Así, la clase "*KFold"* fue configurada para combinar el conjunto de datos y establecer k=10, el cual es la cantidad de divisiones o iteraciones independientes que se harán e indicaran el tamaño del conjunto de validación.

La función "*cross_val_score()"* se utilizó para realizar la evaluación, tomando el conjunto de datos de prueba y la configuración de la validación cruzada. Finalmente, se obtuvieron los resultados del desempeño del clasificador.

3.5 Resumen del tercer capítulo

En este capítulo se consideraron las diferentes técnicas de análisis que se pueden presentar en la clasificación de fallas eléctricas de un motor.

En el análisis en el dominio del tiempo se implementaron las redes neuronales para la clasificación de fallas eléctricas de motores de inducción. Adicionalmente, se realizó un análisis con un clasificador SVM para complementar el análisis en el dominio tiempo de la clasificación de fallas eléctricas.

En el análisis en el domino de la frecuencia se implementaron las redes neuronales para la clasificación de fallas eléctricas en motores de inducción. Esta implementación es de suma importancia para la comparación de los resultados y poder establecer conclusiones claras de este trabajo de tesis.

Capítulo 4

Resultados

En este capítulo se presentarán, mediante simulaciones, los resultados obtenidos por una técnica de análisis tradicional de detección de fallas en motores eléctricos (MCSA). Además, se presentarán los resultados obtenidos en el dominio del tiempo por la red neuronal y el clasificador SVM. Finalmente, se presentarán los resultados en el dominio de la frecuencia obtenidos por la red neuronal.

4.1 Introducción a las pruebas

En la prueba tradicional, MCSA, se podrá observar la progresión de la falla de corto circuito. Adicionalmente, dentro de este tipo de pruebas se analizará la falla de barras rotas. Debido a la naturaleza de la prueba MCSA, estos dos tipos de fallas se realizarán en el dominio de la frecuencia. Una de las ventajas de MCSA es que se puede observar de una manera visual el comportamiento de las fallas del motor.

En las pruebas en el dominio del tiempo, se podrán observar las diferencias que se obtienen en los resultados al realizar diferentes métodos de análisis. Por otro lado, las pruebas en el dominio de la frecuencia serán de suma importancia ya que se podrán comparar los resultados con los del domino del tiempo y de la técnica tradicional. Y finalmente, determinar en cuales métodos se obtienen los mejores resultados.

4.2 Prueba MCSA

La técnica de análisis de MCSA permite detectar las fallas eléctricas de los motores de inducción principalmente. Esta técnica muestra como es el comportamiento de las fallas que se pueden presentar en los motores. Por lo que esta técnica solo brinda información de cuando la falla ya ocurrió y no de cuando esta falla se este presentando. Por lo tanto, el objetivo principal de esta tesis es usar las redes neuronales artificiales para el análisis y la detección de fallas incipientes en motores de inducción. Aunque la técnica MCSA se puede adaptar a las condiciones de detección de fallas, en este trabajo de tesis no se implementará.

En la figura 4.1 se puede observar el comportamiento de la falla de corto circuito al 10% implementando la técnica de análisis MCSA.

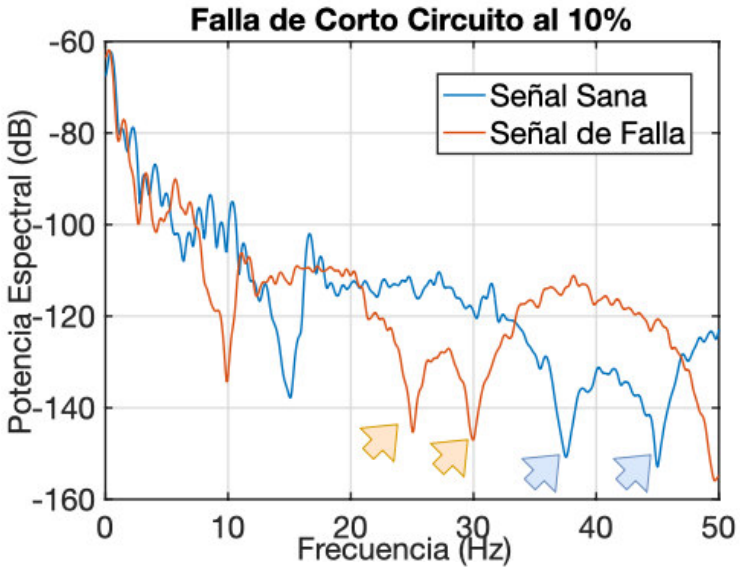

Figura 4.1. Falla de corto circuito al 10% utilizando MCSA.

En la gráfica de la figura 4.1 se muestra el comportamiento de la falla de corto circuito al 10% con la señal sana del motor. En estas señales se puede observar una distorsión. Además, las señales se encuentran en un desfase de frecuencia como se observa en las frecuencias de treinta y ocho y cuarenta y cinco Hertz de la señal sana y las frecuencias de veinticinco y treinta Hertz de la señal de falla.

En la figura 4.2 se puede observar el comportamiento de la falla de corto circuito al 20% implementando la técnica de análisis MCSA.

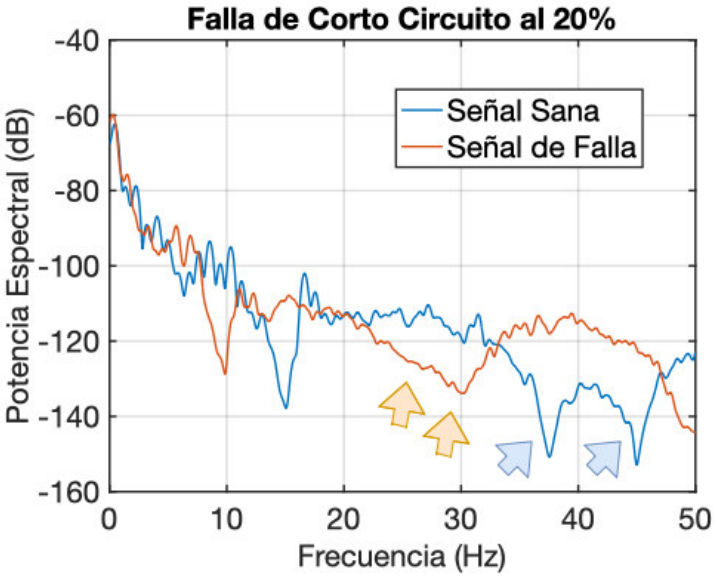

Figura 4.2. Falla de corto circuito al 20% utilizando MCSA.

En la gráfica de la figura 4.2 se muestra el comportamiento de la falla de corto circuito al 20% en comparación con la señal sana del motor. En estas señales se observa un cambio muy significativo en las frecuencias de veinticinco y treinta Hertz de la señal de falla. En la señal de falla mostrada en color naranja se puede observar como el pico de la frecuencia veinticinco desapareció y el pico de la frecuencia de treinta aún es visible. Por lo que en estas frecuencias se muestra una distorsión completa de la señal. Así, se puede apreciar la evolución de la falla con respecto a la mostrada en la figura 4.1.

En la figura 4.3 se puede observar el comportamiento de la falla de corto circuito al 30% implementando la técnica de análisis MCSA.

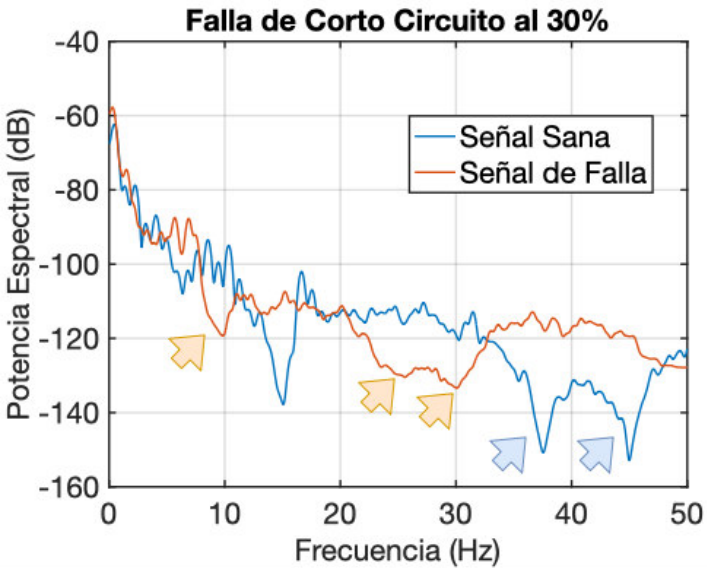

Figura 4.3. Falla de corto circuito al 30% utilizando MCSA.

En la gráfica de la figura 4.3, la falla ya es evidente y muy clara cuando se compara con la señal sana del motor mostrada en color azul. Por ejemplo, en la señal de falla en las frecuencias de veinticinco y treinta Hertz es muy evidente la distorsión de la falla comparándola con la señal sana del motor en las frecuencias de treinta y ocho y cuarenta y cinco Hertz. Además, en la frecuencia de 10 Hertz se aprecia como la señal sufrió un cambio muy significativo en la amplitud del pico de la señal. Por lo tanto, en esta progresión de la falla ya es muy evidente que el motor no se encuentra en buen estado.

En la figura 4.4 se puede observar el comportamiento de la falla de corto circuito al 40% implementando la técnica de análisis MCSA.

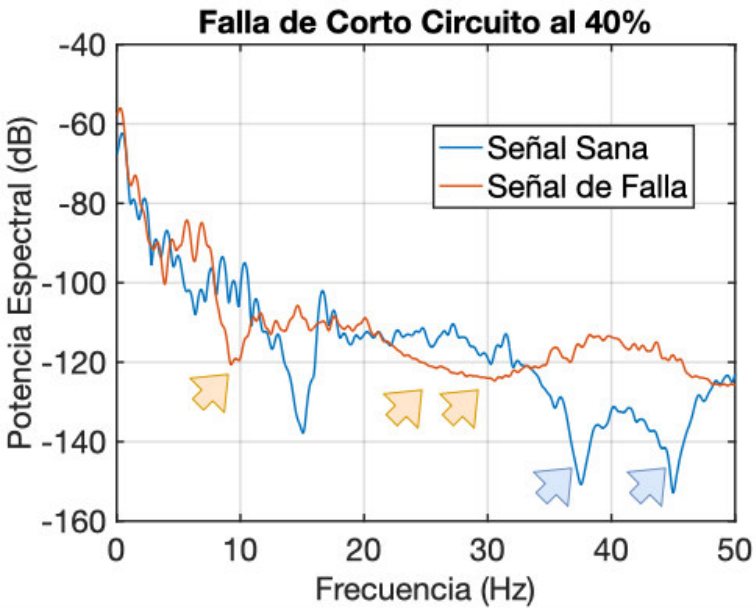

Figura 4.4. Falla de corto circuito al 40% utilizando MCSA.

En la gráfica de la figura 4.4 se muestra un cambio muy significativo en la señal de falla del motor en comparación con la señal sana del motor. En tres valores de la frecuencia se puede observar una distorsión completa de la señal, estos valores de frecuencia son 20, 25 y 10 Hz. Por otro lado, en la frecuencia de diez Hertz se aprecia como se forman dos pequeños picos casi imperceptibles. Estos picos se pueden interpretar como un funcionamiento anómalo del motor.

En la figura 4.5 se puede observar el comportamiento de la falla de barras rotas implementando la técnica de análisis MCSA.

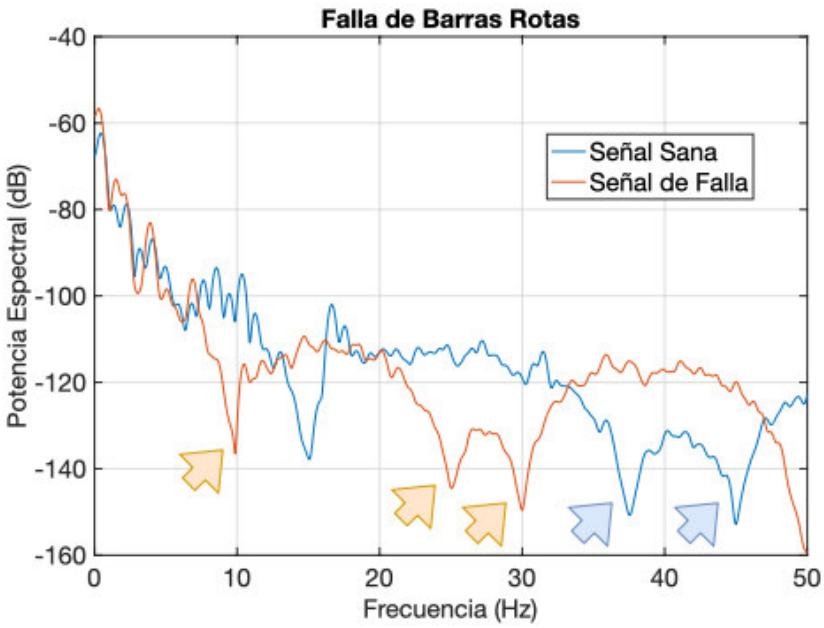

Figura 4.5. Falla de barras rotas utilizando MCSA.

En la gráfica de la figura 4.5 se muestra el comportamiento de la falla de barras rotas donde se pueden observar las variaciones de las distorsiones que presenta la señal en la frecuencia de diez Hertz. En la parte central se pueden observar unas distorsiones en las frecuencias de veinticinco y treinta Hertz donde se aprecia un funcionamiento anómalo del motor.

4.3 Pruebas en el dominio del tiempo

A continuación, se muestran los resultados obtenidos por la red neuronal en el dominio del tiempo.

4.3.1 Red Neuronal

Con esta prueba se podrá observar el comportamiento de la red neuronal para la clasificación de fallas. Adicionalmente, en estas pruebas se determinará si es confiable el diseño de la red neuronal, por medio de las matrices de confusión.

4.3.1.1 Falla de Corto Circuito al 10%

En esta prueba se pueden observar los resultados de la red neuronal utilizando 100 entradas y 8 neuronas para la clasificación de la falla. En las tablas 4.1 y 4.2 se pueden apreciar los resultados obtenidos.

	Sana	Falla	Rechazo
Sana	927	0	0
Falla	0	930	0
Rechazo	0	0	1824

Tabla 4.1. Matriz de confusión de entrenamiento de falla de corto circuito al 10%.

	Sana	Falla	Rechazo
Sana	207	0	1*
Falla	0	235	1*
Rechazo	3*	1*	473

Tabla 4.2. Matriz de confusión de validación de falla de corto circuito al 10%.

En la tabla 4.1 se puede observar que no se obtuvieron errores en el entrenamiento de la red. En cambio, en la tabla 4.2 se puede observar que no se pudieron clasificar 6 datos.

Adicionalmente, se calcularán las métricas que se obtuvieron de las matrices de confusión para obtener una mejor percepción del rendimiento del clasificador. En las ecuaciones (2.13), (2.14) y (2.15) se puede observar las métricas a evaluar.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} = \frac{235 + 207}{235 + 207 + 0 + 0} = 1.0 \qquad (2.13)$$

$$Precisión = \frac{TP}{TP + FP} = \frac{235}{235 + 0} = 1.0. \qquad (2.14)$$

$$Sensibilidad = \frac{TP}{TP + FN} = \frac{235}{235 + 0} = 1.0 \qquad (2.15)$$

Con los valores obtenidos de las métricas se puede decir que el clasificador obtuvo un desempeño óptimo en este porcentaje de la falla ya que los valores de la exactitud, precisión y sensibilidad obtuvieron valores de “1.0".

4.3.1.2 Falla de Corto Circuito al 20%

En esta prueba se pueden observar los resultados de la red neuronal utilizando 100 entradas y 6 neuronas para la clasificación de la falla. En las tablas 4.3 y 4.4 se pueden ver los resultados obtenidos.

	Sana	Falla	Rechazo
Sana	905	0	0
Falla	0	932	0
Rechazo	0	0	1844

Tabla 4.3. Matriz de confusión de entrenamiento de falla de corto circuito al 20%.

	Sana	Falla	Rechazo
Sana	231	1*	0
Falla	1	231	0
Rechazo	0	0	457

Tabla 4.4. Matriz de confusión de validación. de falla de corto circuito al 20%.

En la tabla 4.3 se puede observar que no se tuvieron errores en el entrenamiento de la red. En cambio, en la tabla 4.4 de la validación del entrenamiento se puede observar que no se clasifico del todo bien teniendo un error de la clase sana como clase falla y viceversa.

Con las ecuaciones (2.13), (2.14) y (2.15) se calcularán las métricas del clasificador para una mejor percepción de su confiabilidad.

$$Exactitud = \frac{231 + 231}{231 + 231 + 1 + 1} = 0.99$$

$$Precisión = \frac{231}{231 + 1} = 0.99$$

$$Sensibilidad = \frac{231}{231 + 1} = 0.99$$

Con los valores obtenidos de las métricas tan cercanos a uno, se puede decir que el clasificador obtuvo un desempeño óptimo en este porcentaje de la falla. Finalmente concluimos que debido a que los valores de las métricas son cercanos a uno, el clasificador es confiable.

4.3.1.3 Falla de Corto Circuito al 30%

En esta prueba se puede observar los resultados de la red neuronal utilizando 100 entradas y 4 neuronas para la clasificación de la falla. En las tablas 4.5 y 4.6 se pueden ver los resultados obtenidos.

	Sana	Falla	Rechazo
Sana	938	0	0
Falla	0	904	0
Rechazo	0	0	1839

Tabla 4.5. Matriz de confusión de entrenamiento de falla de corto circuito al 30%.

	Sana	Falla	Rechazo
Sana	230	0	1*
Falla	0	228	0
Rechazo	0	0	462

Tabla 4.6. Matriz de confusión de validación de falla de corto circuito al 30%.

En la tabla 4.5 se puede observar que no se obtuvieron errores durante el entrenamiento de la red neuronal. Por otro lado, en la tabla 4.6 se obtuvo un único error de rechazo que no se pudo clasificar.

Con las ecuaciones (2.13), (2.14) y (2.15) se calcularán las métricas del clasificador para una mejor percepción de su confiabilidad.

$$Exactitud = \frac{230 + 228}{230 + 228 + 0 + 0} = 1.0$$

$$Precisión = \frac{230}{230 + 0} = 1.0$$

$$Sensibilidad = \frac{230}{230 + 0} = 1.0$$

Con los valores obtenidos de las métricas se puede decir que el clasificador obtuvo un desempeño óptimo al obtener una clasificación perfecta.

4.3.1.4 Falla de Corto Circuito al 40%

En esta prueba se puede observar los resultados de la red neuronal utilizando 100 entradas y 4 neuronas para la clasificación de la falla. En las tablas 4.7 y 4.8 se pueden ver los resultados obtenidos.

	Sana	**Falla**	**Rechazo**
Sana	932	0	0
Falla	0	900	0
Rechazo	0	0	1849

Tabla 4.7. Matriz de confusión de entrenamiento de falla de corto circuito al 40%.

	Sana	Falla	Rechazo
Sana	235	0	1*
Falla	0	233	0
Rechazo	0	0	452

Tabla 4.8. Matriz de confusión de validación de falla de corto circuito al 40%.

En la tabla 4.7 se puede observar que no se obtuvieron errores de entrenamiento. En cambio, en la tabla 4.8 se puede observar que no se pudo clasificar un dato de rechazo.

Con las ecuaciones (2.13), (2.14) y (2.15) se calcularán las métricas del clasificador para una mejor percepción del desempeño del clasificador.

$$Exactitud = \frac{235 + 233}{235 + 233 + 0 + 0} = 1.0$$

$$Precisión = \frac{235}{235 + 0} = 1.0$$

$$Sensibilidad = \frac{235}{235 + 0} = 1.0$$

Con los valores obtenidos de las métricas se puede decir que el clasificador obtuvo un desempeño óptimo al obtener una clasificación perfecta.

4.3.1.2 Falla de Barras Rotas

En esta prueba se puede observar los resultados de la red neuronal utilizando 100 entradas y 12 neuronas para la clasificación de la falla. En las tablas 4.9 y 4.10 se pueden ver los resultados obtenidos.

	Sana	Falla	Rechazo
Sana	920	0	0
Falla	0	916	0
Rechazo	0	0	1845

Tabla 4.9. Matriz de confusión de entrenamiento de falla de barras rotas.

	Sana	Falla	Rechazo
Sana	215	0	0
Falla	1*	249	0
Rechazo	0	0	456

Tabla 4.10 Matriz de confusión de validación de falla de barras rotas.

En la tabla 4.9 se puede observar como no se tienen errores de entrenamiento. Por otro lado, en la tabla 4.10 se puede observar que se clasifico mal un dato de la clase de falla clasificándolo como clase sana.

Con las ecuaciones (2.13), (2.14) y (2.15) se calcularán las métricas del clasificador para una mejor percepción del desempeño del clasificador.

$$Exactitud = \frac{215 + 249}{215 + 249 + 0 + 1} = 0.99$$

$$Precisión = \frac{215}{215 + 0} = 1.0$$

$$Sensibilidad = \frac{215}{215 + 1} = 0.99$$

Con los resultados de las métricas que se obtuvieron del clasificador al ser cercanos a uno se tiene un clasificador confiable y preciso con una sensibilidad y exactitud muy precisas.

4.3.2 Pruebas con SVM

Con la elaboración de esta prueba se podrán comparar los resultados obtenidos en la sección 4.3.1. Ya que esta prueba permite observar los resultados en tres métricas diferentes. Por lo que se podrá determinar cual es el resultado óptimo de la clasificación.

En la tabla 4.11 se muestran los resultados obtenidos por el clasificador SVM.

Tipo de Falla	# Errores de Clasificación	Exactitud	Precisión	Sensibilidad
Clasificador SVM Kernel Lineal				
Corto Cir. 10%	480	0.4788	0.4788	1.0
Corto Cir. 20%	480	0.4788	0.4788	1.0
Corto Cir. 30%	480	0.4788	0.4788	1.0
Corto Cir. 40%	480	0.4788	0.4788	1.0
Barras Rotas	480	0.4788	0.4788	1.0
Clasificador SVM Kernel Polinomial				
Corto Cir. 10%	462	0.4983	0.4769	0.4920
Corto Cir. 20%	369	0.5993	0.5949	0.5623
Corto Cir. 30%	245	0.7339	0.9094	0.5918
Corto Cir. 40%	206	0.7763	0.9094	0.5918
Barras Rotas	300	0.6742	0.6535	0.6802
Clasificador SVM Kernel Radial				
Corto Cir. 10%	25	0.9728	0.9463	1.0
Corto Cir. 20%	0	1.0	1.0	1.0
Corto Cir. 30%	0	1.0	1.0	1.0
Corto Cir. 40%	0	1.0	1.0	1.0
Barras Rotas	0	1.0	1.0	1.0

Tabla 4.11. Resultados del clasificador SVM.

En la tabla 4.11 se puede observar el comportamiento que obtuvo el clasificador en los diferentes kernels de análisis. En el kernel lineal se puede observar que en la exactitud y precisión se obtuvieron valores por debajo de “0.5”, y una sensibilidad de “1.0”; con estos valores el clasificador en este kernel no es confiable. Por otro lado, en el kernel polinomial se puede apreciar que las métricas varían un poco mas, pero de igual manera siguen siendo valores bajos en la exactitud y la sensibilidad con valores por debajo de “0.80” y “0.70” respectivamente, aunque la precisión ya fue de “0.90” no es suficiente para confiar en el clasificador. Finalmente se puede observar un mejor desempeño del clasificador en el kernel radial, ya que en su exactitud y precisión en la falla de corto circuito al 10% sus valores fueron muy cercanos a “1.0” y con una sensibilidad de “1.0” y en las fallas restantes todos sus valores fueron de “1.0”, por lo que con este kernel se obtuvo un excelente desempeño del clasificador.

En la tabla 4.12 se muestran los valores verdaderos positivos y negativos, además de los falsos positivos y negativos de la clasificación del SVM.

Tipo de Falla	TP	TN	FP	FN
Clasificador SVM Kernel Lineal				
Corto Cir. 10%	0	441	480	0
Corto Cir. 20%	0	441	480	0
Corto Cir. 30%	0	441	480	0
Corto Cir. 40%	0	441	480	0
Barras Rotas	0	441	480	0
Clasificador SVM Kernel Polinomial				
Corto Cir. 10%	242	217	238	224
Corto Cir. 20%	304	248	176	193
Corto Cir. 30%	419	257	61	184
Corto Cir. 40%	454	261	26	180
Barras Rotas	321	300	159	141
Clasificador SVM Kernel Radial				
Corto Cir. 10%	455	441	25	0
Corto Cir. 20%	480	441	0	0
Corto Cir. 30%	480	441	0	0
Corto Cir. 40%	480	441	0	0
Barras Rotas	480	441	0	0

Tabla 4.12. Matriz de confusión del SVM.

Además de calcular cada una de las métricas de las clasificaciones obtenidas por el clasificador SVM. Se cálculo la confiabilidad del clasificador. Por tal motivo se realizo la validación cruzada del clasificador obteniendo un valor de “0.868”.

4.4 Pruebas en el dominio de la frecuencia

A continuación, se muestran los resultados obtenidos por la red neuronal en el dominio de la frecuencia.

4.4.1 Red Neuronal

Con esta prueba se podrá observar el comportamiento de la red neuronal en el dominio de la frecuencia. Donde se determinará cual técnica de análisis es confiable en la clasificación de fallas.

Tipo de Falla	# Errores de Entrenamiento			Exactitud	Precisión	Sensibilidad	Diseño de la Red	
	Entrenamiento	Validación	Rechazo				Entradas	Neuronas
10%	0	1	1	0.99	1.0	0.99	128	19
20%	0	0	0	1.0	1.0	1.0	128	9
30%	0	0	0	1.0	1.0	1.0	128	7
40%	0	2	1	0.99	0.99	0.99	128	5
Barras Rotas	0	1	0	0.99	1.0	0.99	128	5

Tabla 4.13. Resultados en el domino de la frecuencia por la red neuronal.

En la tabla 4.13 se muestra el desempeño de la red neuronal en la clasificación de fallas. Donde se puede apreciar que en la falla de corto circuito al 10% y al 40% se obtuvieron algunos errores de validación y de rechazo. Estos valores son muy pocos ya que no afecto en el desempeño del clasificador, ya que este desempeño se puede apreciar en las métricas obtenidas del mismo, estos valores fueron muy cercanos a “1.0” por lo que es confiable el clasificador en la detección de fallas eléctricas de motores de inducción. Por otro lado, en la falla de barras rotas se obtuvo un único error de validación, lo cual no afecto de alguna manera en la clasificación de la falla, los valores de las métricas del clasificador de igual manera fueron muy cercanas a “1.0” por lo que en esa falla el clasificador sigue siendo confiable y óptimo.

4.5 Conclusiones

Las pruebas por computadora se realizaron utilizando métodos clásicos en el estado del arte para propósitos de comparación. Sin embargo, en este trabajo de tesis se propone el uso de otras técnicas de el área de inteligencia artificial. Específicamente, las pruebas incluyeron la aplicación de las redes neuronales artificiales a la detección temprana de fallas en un motor de inducción. Adicionalmente las simulaciones incorporaron el uso de Support Vector Machine como una metodología nueva en la predicción de una falla en este tipo de motores. Por otro lado, la adquisición de muestras fue una de las partes fundamentales de esta tesis, ya que el banco de pruebas tiene la manera de poder observar la progresión de la falla de corto circuito y analizarla más detalladamente. El Preprocesamiento de las muestras es una parte sumamente importante debido a que esto garantiza, que los resultados obtenidos sean eficientes. Además, el diseño de la red neuronal depende de que las muestras se capturen de una manera precisa y de que el Preprocesamiento de estas muestras sea excelente para garantizar el buen desempeño de la red y de el clasificador SVM.

En primer lugar, se pudo obtener una correcta comprobación de detección de fallas implementando técnicas de análisis tradicionales como lo es el MCSA. Donde se aprecio claramente la evolución de las fallas de corto circuito y de barras rotas. Adicionalmente, se obtuvieron las gráficas del comportamiento de la falla de corto circuito, debido a que se simularon cada uno de los progresos de la falla permitiendo conocer y analizar la falla a detalle. Por otro lado, se pudo obtener la visualización de la falla de barras rotas del motor implementando la misma técnica de análisis, con lo cual se pudo apreciar el comportamiento que tiene la señal de falla con la señal sana del motor.

En la implementación de las redes neuronales se pudo comprobar que la inteligencia artificial se puede adaptar a casi cualquier tipo de problemas que se pueden presentar en la industria y la investigación siendo una herramienta muy útil para la detección de fallas eléctricas.

Con el diseño de la red neuronal en el dominio del tiempo se obtuvieron excelentes resultados en cada una de las fallas analizadas. Una parte fundamental que se tiene que resaltar es que la red implemento muy pocas neuronas en sus análisis. Que en este caso se logro el objetivo de poder clasificar las fallas incipientes de corto circuito que presenten los motores eléctricos de inducción. Además, en el porcentaje del 10% se obtuvieron algunos errores que pueden ser insignificantes ya que son muy pocos, lo que permite obtener un clasificador eficiente y confiable en la detección de fallas incipientes de motores eléctricos de inducción. En cambio, en el porcentaje del 20% se obtuvieron menos errores de clasificación por lo que se obtuvo un clasificador confiable y eficiente en fallas incipientes de motores en el dominio del tiempo. Finalmente, se puede concluir que las redes neuronales en el dominio del tiempo obtuvieron mejores resultados con muy pocas neuronas, en comparación a los resultados mencionados en el estado del arte de esta tesis.

Por otro lado, en el diseño del clasificador SVM se obtuvieron resultados excelentes en el dominio del tiempo siendo el único dominio donde se implemento. Al ser un clasificador donde se pueden analizar por diferentes kernels (lineal, polinomial y radial), lo cual resulta muy útil al momento de la clasificación de fallas. Ya que permite obtener diferentes resultados de la misma falla y poder obtener una mejor percepción de la clasificación. Al utilizar cada uno de los kernels se pudo observar que el kernel radial mantuvo una clasificación perfecta en cada una de las fallas analizadas. Con esto se comprobó que este clasificador resulta muy útil en la detección de fallas incipientes de motores eléctricos, gracias a que este clasificador mantuvo un desempeño perfecto en la detección de las fallas analizadas en el dominio del tiempo.

Al diseñar la red neuronal en el dominio de la frecuencia se pudo observar que resulta muy eficiente al tener resultados casi perfectos en su clasificación, utilizando aun menos neuronas que la red diseñada en el dominio del tiempo de esta tesis. Por lo que la implementación de esta red neuronal permitió detectar las fallas incipientes en motores de inducción de una manera más precisa que en el dominio del tiempo. Debido a que, este tipo de redes neuronales aprende

lo que se le enseña, por lo tanto, le es mas fácil detectar las fallas en cuanto identifica alguna variación de lo que ya aprendió. Por lo que este tipo de redes son muy adaptables al problema que se desee analizar. Y en particular para este trabajo de tesis resultaron ser muy eficientes en la detección incipiente de fallas eléctricas de motores de inducción.

4.6 Trabajo a futuro

A continuación, se mencionan las posibles mejoras que pueden ser implementadas como trabajo a futuro:

- Implementar una red neuronal de aprendizaje profundo, donde posiblemente se puedan analizar más fallas que pueda presentar el motor y ver el comportamiento de este tipo de redes neuronales.
- Implementar el clasificador SVM en el dominio de la frecuencia para observar como se comporta. Además, de que se puede implementar utilizando mas fallas y ver si sigue siendo una buena opción de clasificación de fallas eléctricas en motores de inducción.
- Desarrollar el sistema físico que permita al usuario implementar estas técnicas de análisis en sus áreas de trabajo, ya que seria de suma importancia para las industrias y las personas poder evitar perdidas monetarias y sobre todo perdidas humanas.

4.7 Resumen del cuarto capítulo

En la primera sección se muestran las señales obtenidas con la técnica de análisis MCSA. Con esta técnica se muestra el comportamiento de las fallas del motor en el dominio de la frecuencia, comparando las señales sanas y de falla del motor.

En la segunda sección, se muestran los resultados en el dominio del tiempo por la red neuronal y el SVM. En donde se observan las diferencias entre los dos análisis. Además, se muestran las métricas obtenidas por cada uno de los análisis. En el caso del SVM se calcularon las métricas del clasificador y los

diferentes resultados de la clasificación de las fallas con la variación de los kernels.

En la tercera sección se muestran los resultados obtenidos por la red neuronal en el dominio de la frecuencia. Además, se muestran las matrices de confusión y las métricas obtenidas por el clasificador.

Finalmente se muestran las conclusiones y los trabajos a futuro que se pueden emplear en esta tesis.

Bibliografías

[1] Arkan, M., D. Kostic-Perovic y P. J. Unsworth, *Modelling and simulation of induction motors with inter-turn faults for diagnostics*, Electric Power Systems Research, 75, 57–66 (2005).

[2] Tallam, R.M., Lee, S.B., Stone, G., Kliman, G.B., Yoo, J., Habetler, T.G., Harley, R.G., *A survey of methods for detection of stator related faults in induction machines*, 4th IEEE International Symposium on Diagnostic for Electric machines, Power Electronics and Drives, SDEMPED 2003, 35-46 (2003).

[3] S. Nandi, H. A. Toliyat, and X. Li, "Condition monitoring and fault diagnosis of electrical motors-A review," IEEE Trans, on Energy Conversion, vol. 20, no 4, pp. 719-729, Sept. (2005).

[4] Thomson, W.T., Fenger, M., *Current signature analysis to detect induction motor faults*. IEEE Industry Applications Magazine, 26-34, (2001).

[5] Torbar, T. W., *Online current monitoring and application of a finite element method to predict the level of static airgap eccentricity in three-phase induction Motors*, IEEE Trans. On energy conversion, vol. 14, no.4, pp 347-357, (1998).

[6] Villada, F.,Cadavid, D. R., *Diagnóstico de fallas en motores de inducción mediante la aplicación de redes neuronales artificiales*, Información Tecnológica,18(2), 105-112, Chile (2007).

[7] M. Eltabach and A. Chahata, "Comparative investigation of electrical diagnostic procedures in induction motors," SPEEDAM 2006, Taormina, Italy, 7p, 23-26 May (2006).

[8] M El Hachemi Benbouzid, *"A review of induction motors signature analysis as a medium for faults detection"*, IEEE *Transactions on industrial electronics*, Vol 47, No. 5, pag 984-993, Oct, (2000).

[9] M.E.H. Benbouzid, M. Vieira, C. Theys, *"Induction motors faults detection and localization using stator current advanced signal processing techniques"*, IEEE *Transactions on Power Electronics*, Vol. 14, No. 1, pp 14-22, Jan (1999).

[10] Greety Jose, P.G Scholar “Induction Motor Fault Diagnosis Methods: A Comparative Study” International Conference on Electrical Engineering (ICEE - 2013) July 6 – 7,2013, Hyderabad, India.

[11] Bazan, G.H., Scalassara, P.R., Endo, W., Goedtel, A., Godoy, W.F., and Palácios, R.H.C., “Stator fault analysis of three-phase induction motors using information measures and artificial neural networks”, Electric Power SystemsResearch, Vol. 143, No. 143, (2017), 347–356.

[12] Shi, P., Chen, Z., Vagapov, Y., and Zouaoui, Z., “A new diagnosis of broken rotor bar fault extent in three phase squirrel cage induction motor”, Mechanical Systems and Signal Processing, Vol. 42, No. 1–2, (2014), 388–403.

[13] Fault detection and diagnosis through artificial intelligence techniques, a state of art Luini Leonardo Hurtado-Cortés [a], Edwin Villarreal-López [b] & Luís Villarreal-López [c]February 4rd, 2016.

[14] Analizador de vibraciones en estado estable y transitorio para la detección de fallas en motores de inducción C. Rodríguez-Doñate, student member, IEEE, J. J. Rangel-Magdaleno, student member, IEEE L. M. Contreras-Medina, student member, IEEE R. J. Romero-Troncoso, senior member IEEE, A. García-Pérez. CIINDET 2008 6º Congreso Internacional sobre Innovación y Desarrollo Tecnológico, 8 al 10 de octubre de 2008, Cuernavaca, Morelos, México.

[15] Konar, P., and Chattopadhyay, P., “Multi-class fault diagnosis of induction motor using Hilbert and Wavelet Transform”, Applied Soft Computing, Vol. 30, (2015), 341–352.

[16] Osman, S., and Wang, W., “A Morphological Hilbert-Huang Transform Technique for Bearing Fault Detection”, IEEE Transactions on Instrumentation and Measurement, Vol. 65, No. 11, (2016), 2646–2656.

[17] Eberhart, R., and Kennedy, J., “A new optimizer using particle swarm theory”, In MHS’95, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, IEEE, (1995), 39–43.

[18] Tarafdar Hagh, M., and Ghadimi, N., “Radial Basis Neural Network Based Islanding Detection in Distributed Generation”, International Journal of Engineering - Transactions A: Basics, Vol. 27, No. 7, (2014), 1061–1070.

[19] Strumiłło, P., and Kamiński, W., “Radial Basis Function Neural Networks: Theory and Applications”, Physica-Verlag HD, Heidelberg, (2003), 107–119.
[20] Mousavi, Y., and Alfi, A., “A memetic algorithm applied to trajectory control by tuning of Fractional Order Proportional- Integral-Derivative controllers”, Applied Soft Computing, Vol. 36, No. 36, (2015), 599–617.