

S.E.P.

S.E.S.

T.N.M

INSTITUTO TECNOLÓGICO DE AGUASCALIENTES



“Comparación de estrategias de machine learning para la clasificación de fallas en un sistema de manufactura”

TESIS

**Para Obtener el Grado de:
Maestría en Ciencias de la Ingeniería**

PRESENTA:

Ing. Abner Birzabith Montejano Leija

Director de tesis:

Dra. Elvia Ruiz Beltrán

Co Director de tesis:

Dr. Jorge Luis Orozco Mora

Sinodal:

M.C. Jorge Octavio Valdés Valadez

Aguascalientes, Ags., Noviembre del 2024.

RESUMEN

Uno de los principales desafíos en la industria son las fallas inesperadas en los equipos, las cuales generan pérdidas significativas para las empresas debido a los tiempos de inactividad. Para minimizar el tiempo de inactividad causado por estas fallas, es fundamental anticiparse a ellas y actuar rápidamente. Esto se puede lograr mediante el análisis de datos recolectados del monitoreo del equipo y la implementación de un plan de mantenimiento predictivo efectivo. En este trabajo se compara el rendimiento de varios algoritmos de Machine Learning utilizando los datos obtenidos al inducir ciertas fallas en un motor, una electroválvula y un cilindro neumáticos, componentes muy presentes en la industria. Para que con base en ciertas métricas de evaluación, se determine el algoritmo que mejor resultado obtenga y se considere esta opción como parte del mantenimiento predictivo dentro de la industria.

Palabras clave: Machine Learning, Diagnóstico de Fallas, Mantenimiento Predictivo, Industria 4.0.

ABSTRACT

One of the main challenges in the industry is unexpected equipment failures, which result in significant losses for companies due to downtime. To minimize the downtime caused by these failures, it is essential to anticipate them and act quickly. This can be achieved through the analysis of data collected from equipment monitoring and the implementation of an effective predictive maintenance plan. This work compares the performance of various Machine Learning algorithms using data obtained by inducing certain failures in a motor, a pneumatic solenoid valve and a pneumatic cylinder, components commonly found in the industry. To, based on specific evaluation metrics, determine which algorithm yields the best results and consider this option as part of predictive maintenance in the industry.

Keywords: *Machine Learning, Failures Diagnosis, Predictive Maintenance, Industry 4.0.*



Instituto Tecnológico de Aguascalientes
División de Estudios de Posgrado e Investigación

Aguascalientes, Ags, 28/noviembre/2024
Oficio No. DEPI 486/2024

**C. ABNER BIRZABITH MONTEJANO LEIJA
PRESENTE**

Los abajo firmantes, Miembros del Jurado para Examen de Grado de Maestría, hacemos CONSTAR que, habiendo revisado el Trabajo de Tesis desarrollado por Usted, bajo el título **"Comparación de estrategias de machine learning para la clasificación de fallas en un sistema de manufactura"**, hemos dictaminado que éste es de aceptarse, por lo que se autoriza su impresión.

ATENTAMENTE

*Excelencia en Educación Tecnológica-
Ingenio, Cultura y Saber, que conducen a la Excelencia*

PRESIDENTE

Elvia Ruiz Beltrán
DRA. ELVIA RUIZ BELTRÁN

SECRETARIO

Jorge Luis Orozco Mora
DR. JORGE LUIS OROZCO MORA

VOCAL

Jorge Octavio Valdés Valadez
M.C. JORGE OCTAVIO VALDÉS VALADEZ



Av. Adolfo López Mateos #1801 Fracc. Bona Gens. C.P. 20256, Aguascalientes, Ags. Tel. 449 910 5002 ext. 127
e-mail: jefatura.depi@aguascalientes.tecnm.mx www.tecnm.mx | www.aguascalientes.tecnm.mx





Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Aguascalientes
División de Estudios de Posgrado e Investigación

Aguascalientes, Ags., 12/diciembre/2024
Asunto: Constancia de Originalidad

**CONSEJO DE POSGRADO
MAESTRÍA EN CIENCIAS DE LA INGENIERÍA
PRESENTE**

El director de tesis de el/la **C. Abner Birzabith Montejano Leija**, estudiante de **Maestría en Ciencias de la Ingeniería** con número de control **G22153217**, hace constar que, después de haber realizado la revisión del informe de originalidad generado con la herramienta Turnitin, el trabajo **"Comparación de estrategias de machine learning para la clasificación de fallas en un sistema de manufactura"** es original conforme a los criterios establecidos. El porcentaje de similaridad es de **05%** y se adjunta resumen del reporte.

ATENTAMENTE

*Excelencia en Educación Tecnológica®
Ingenio, Cultura y Saber, que conducen a la Excelencia*

Vo.Bo.

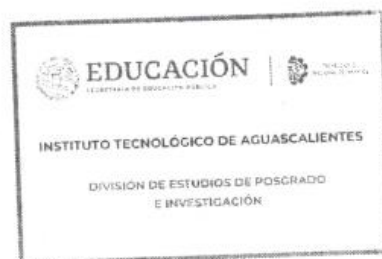
Elvia Ruiz Beltrán.

**DRA. ELVIA RUIZ BELTRÁN
PROFESORA DE TIEMPO COMPLETO
DPTO. DE SISTEMAS Y COMPUTACIÓN**

ccp. Archivo

mar081

**DR. MARIO ALBERTO RODRÍGUEZ DÍAZ
COORDINADOR
MAESTRÍA EN CIENCIAS DE LA INGENIERÍA**



Av. Adolfo López Mateos #1801 Fracc. Bona Gens C.P. 20256 Aguascalientes, Ags. Tel. 449 9105002 Ext. 127
e-mail: jefatura.depi@aguascalientes.tecnm.mx tecnm.mx | www.aguascalientes.tecnm.mx



AUTORIZACIÓN DE USO DE DERECHOS DE AUTOR OTORGADO POR

Abner Birzabith Montejano Leija, mayor de edad, con domicilio ubicado en Revolución #105, Loma Bonita, Ags. Ags., en mi calidad de titular y autor de la tesis denominada Comparación de estrategias de machine learning para la clasificación de fallas en un sistema de manufactura quien para todos los fines del presente documento se denominará **EL AUTOR Y/O TITULAR**, suscribo el presente documento de autorización de uso de derechos patrimoniales de autor a favor del Tecnológico Nacional de México/Instituto Tecnológico de Aguascalientes el cual se registrá por las cláusulas siguientes:

PRIMERA – AUTORIZACIÓN: **EL AUTOR Y/O TITULAR**, mediante el presente documento autoriza la utilización de los derechos patrimoniales de autor al Tecnológico Nacional de México/Instituto Tecnológico de Aguascalientes, de la tesis denominada Comparación de estrategias de machine learning para la clasificación de fallas en un sistema de manufactura, a través del Repositorio Institucional del Tecnológico Nacional de México (en lo sucesivo TecNM) y en el Repositorio Nacional, que puede ser consultado en la liga electrónica: (www.repositorionacionalcti.mx/).

SEGUNDA – OBJETO: Por medio del presente escrito, **EL AUTOR Y/O TITULAR** autoriza al Tecnológico Nacional de México/Instituto Tecnológico de Aguascalientes, a través del Repositorio Institucional del TecNM y en el Repositorio Nacional para que de conformidad con la Ley Federal del Derecho de Autor y la Ley de la Propiedad Industrial, use los derechos del documento antes referido, con fines exclusivamente académicos.

TERCERA – TERRITORIO: Los derechos aquí autorizados se dan sin limitación geográfica o territorial alguna.

CUARTA – ALCANCE: La presente autorización se da tanto para formato o soporte material, y se extiende a la utilización en medio óptico, magnético, electrónico, en red, mensajes de datos o similar conocido o por conocer, del ejemplar o número respectivo de la publicación.

QUINTA – EXCLUSIVIDAD: La autorización de uso aquí establecida no implica exclusividad en favor del Tecnológico Nacional de México/Instituto Tecnológico de Aguascalientes. Por lo tanto, **EL AUTOR Y/O TITULAR** en su carácter de autor de la obra objeto del presente documento se reserva el derecho de publicar directamente, u otorgar a cualquier tercero, autorizaciones de uso similares o en los mismos términos aquí acordados.

SEXTA – DERECHOS MORALES (Créditos y mención): La Autorización de los derechos antes mencionados no implica la cesión de los derechos morales sobre los mismos por cuanto en conformidad con lo establecido en los artículos 18, 19, 20, 21, 22 y 23 de la Ley Federal de Derechos de Autor, dada la cuenta que estos derechos son inalienables, imprescriptibles, irrenunciables e inembargables. Por lo tanto, los mencionados derechos seguirán radicados en cabeza de **EL AUTOR Y/O TITULAR**, y siempre deberá mencionarse su nombre cuando se utilice la obra.

SÉPTIMA – AUTORÍA: **EL AUTOR Y/O TITULAR**, declara y ratifica que el material objeto de la presente fue realizada por él o por ella sin violar o usurpar derechos de Propiedad Intelectual de terceros.

Aguascalientes, Aguascalientes, a 10 días del mes de Diciembre de 2024.

Autor de la Tesis

Director de Tesis

Abner B. Montejano Leija

Elvia Ruiz Beltrán

Abner Birzabith Montejano Leija

Elvia Ruiz Beltrán

Nombre y firma

Nombre y firma

AGRADECIMIENTOS

En primer lugar, agradezco profundamente a la Dra. Elvia Ruiz Beltrán por su valioso tiempo y orientación a lo largo de todo el proyecto. Su experiencia y guía han sido fundamentales para el desarrollo de esta investigación.

Agradezco también al Dr. Jorge Luis Orozco Mora y al M.C. Jorge Octavio Valdés, quienes me brindaron orientación, ya que sus aportes y sugerencias que fueron clave para el desarrollo del proyecto.

Agradezco a mi familia, por su constante apoyo y comprensión. Sin ellos, este proyecto no habría sido posible. En especial a mis padres, por su paciencia y motivación durante esta etapa.

Finalmente, quiero agradecer a las instituciones que hicieron posible la realización este trabajo. Al Consejo Nacional de Humanidades, Ciencias y Tecnologías CONAHCYT, con número de becario No. CVU 1242162, por el apoyo económico otorgado para la realización de mis estudios de Maestría. De igual manera, al Tecnológico Nacional de México / Instituto Tecnológico de Aguascalientes, por haberme brindado la oportunidad de formar parte de esta institución, así como por permitirme hacer uso de sus instalaciones y recursos, los cuales fueron esenciales para el desarrollo de esta tesis.

ÍNDICE GENERAL

CAPÍTULO 1. INTRODUCCIÓN.....	13
1.1 PLANTEAMIENTO DEL PROBLEMA	15
1.2 OBJETIVOS	16
1.2.1 OBJETIVO GENERAL	16
1.2.2. OBJETIVOS ESPECÍFICOS	16
1.3 ANTECEDENTES.....	16
1.4 JUSTIFICACIÓN.....	19
CAPÍTULO 2. MARCO TEÓRICO	21
2.1 MANTENIMIENTO	21
2.1.1 OBJETIVOS DE MANTENIMIENTO.....	22
2.1.2 INDICADORES DE MANTENIMIENTO	23
2.1.3 TIPOS DE MANTENIMIENTO	26
2.1.4 ANÁLISIS DE CRITICIDAD	29
2.1.5 TÉCNICAS DE MANTENIMIENTO PREDICTIVO	32
2.2 INTELIGENCIA ARTIFICIAL: MACHINE LEARNING	51
2.2.1 ¿QUÉ ES MACHINE LEARNING?	52
2.2.2 PROCESO DE DESARROLLO DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO	56
2.2.3 MÉTRICAS DE EVALUACIÓN DE MODELOS DE APRENDIZAJE AUTOMÁTICO	59
2.2.4 ÁRBOLES DE DECISIÓN	61
2.2.5 MÉTODOS DE ENSAMBLE.....	64
2.2.6 SVM.....	68
2.2.7 KNN.....	78
2.2.8 K-MEANS.....	79
2.2.9 CLUSTERING JERÁRQUICO.....	82
2.2.10 ANÁLISIS DE COMPONENTES PRINCIPALES	85
2.3 INTELIGENCIA ARTIFICIAL: DEEP LEARNING	88
2.3.1 HISTORIA DE LAS REDES NEURONALES.....	90
2.3.2 ANALOGÍA ENTRE NEURONAS BIOLÓGICAS Y ARTIFICIALES	91
2.3.3 PERCEPTRÓN	93

2.3.4 FUNCIÓN DE ACTIVACIÓN.....	95
2.3.5 PERCEPTRÓN MULTICAPA.....	98
2.3.6 ENTRENAMIENTO DE REDES NEURONALES	100
2.3.7 REDES NEURONALES CONVOLUCIONALES.....	117
2.3.8 REDES RECURRENTEES	129
CAPÍTULO 3. CASO DE ESTUDIO	144
CAPITULO 4. MATERIALES Y MÉTODOS.....	149
CAPÍTULO 5. RESULTADOS.....	172
CAPÍTULO 6. CONCLUSIONES.....	201
PUBLICACIONES REALIZADAS	203
CAPÍTULO 7. REFERENCIAS BIBLIOGRÁFICAS	204
ANEXOS	210

ÍNDICE DE FIGURAS

Figura 1. Neurona biológica	92
Figura 2. Neurona Artificial.....	93
Figura 3 Diferencia entre el SGD y el SGD con momentum	113
Figura 4 Estructura de una CNN.....	119
Figura 5 Ejemplo de operación de convolución.....	121
Figura 6 Diferencia entre Half padding y Full-padding	124
Figura 7 Convolución aplicando distintos valores de paso	125
Figura 8 RNN.....	130
Figura 9 Representacion en capas de tiempo de una RNN simple.....	131
Figura 10 Ejemplo de Red Neuronal Recurrente Bidireccional	134
Figura 11 LSTM.....	139
Figura 12 GRU	142
Figura 13 Banda transportadora del sistema de enlatado.....	149
Figura 14 Tensor de la banda transportadora	150
Figura 15 Módulo de enlatado.	151
Figura 16 Estación de colocación de tapa.	152
Figura 17 a) Muestra el empaque que se extrae de la válvula, b) Salidas de la válvula obstruidas por suciedad.....	153
Figura 18 Representación gráfica de las distintas fallas en el dominio del tiempo	158
Figura 19 Representación gráfica de las distintas fallas en el dominio de la frecuencia	159
Figura 20 División del Dataset, creación de conjuntos de entrenamiento y prueba	161
Figura 21 Normalización de los datos	161
Figura 22 Selección de hiperpárametros del modelo SVM	162
Figura 23 Selección de hiperpárametros del modelo Árbol de decisión.....	164
Figura 24 Selección de hiperpárametros del modelo KNN	165
Figura 25 Selección de hiperpárametros del modelo MLP	168
Figura 26 Desarrollo del modelo SVM.....	169
Figura 27 Desarrollo del modelo Árbol de decisión	170
Figura 28 Desarrollo del modelo AdaBoost	170
Figura 29 Desarrollo del modelo Random Forest	170
Figura 30 Desarrollo del modelo Random Forest	170
Figura 31 Desarrollo del modelo Stacking	170
Figura 32 Desarrollo del modelo MLP.....	170
Figura 33 Ejemplo de matrices de confusión obtenidas	171

ÍNDICE DE TABLAS

Tabla 1 Matriz de criticidad.....	31
Tabla 2 Severidad de la vibración según norma ISO 10816-3	35
Tabla 3 Matriz de confusión SVM.....	146
Tabla 4 Matriz de confusión Árbol de decisión.....	146
Tabla 5 Matriz de confusión Boosting.....	146
Tabla 6 Matriz de confusión Bagging.....	146
Tabla 7 Matriz de confusión Stacking.....	147
Tabla 8 Matriz de confusión KNN.....	147
Tabla 9 Evaluación de rendimiento de modelos de Machine Learning.	147
Tabla 10 Hiperparámetros utilizados en SVM.....	163
Tabla 11 Hiperparámetros árbol de decisión	165
Tabla 12 Hiperparámetros KNN	166
Tabla 13 Hiperparámetros Red Neuronal	168
Tabla 14 Resultados de clasificación de fallas en banda transportadora aplicando PCA.....	172
Tabla 15 Resultados de clasificación de fallas en banda transportadora aplicando eliminación hacia atrás	173
Tabla 16 Resultados de clasificación de fallas en banda transportadora sin reducción de variables	174
Tabla 17 Resultados de clasificación de fallas en válvula neumática aplicando PCA	175
Tabla 18 Resultados de clasificación de fallas en válvula neumática aplicando eliminación hacia atrás	176
Tabla 19 Resultados de clasificación de fallas en válvula neumática sin reducción de variables	177
Tabla 20 Resultados de clasificación de fallas en cilindro neumático aplicando PCA.....	179
Tabla 21 Resultados de clasificación de fallas en cilindro neumático aplicando eliminación hacia atrás	180
Tabla 22 Resultados de clasificación de fallas en cilindro neumático sin reducción de variables	181
Tabla 23 Resultado promediado de clasificación de fallas en banda transportadora.....	182
Tabla 24 Resultado promediado de clasificación de fallas en válvula neumática	183
Tabla 25 Resultado promediado de clasificación de fallas en cilindro neumático	183
Tabla 26 Valores obtenido al restar la desviación estándar al resultado promediado de clasificación de fallas en banda transportadora	184
Tabla 27 Valores obtenido al restar la desviación estándar al resultado promediado de clasificación de fallas en válvula neumática.....	185
Tabla 28 Valores obtenido al restar la desviación estándar al resultado promediado de clasificación de fallas en cilindro neumático	186
Tabla 29 Generalización de los resultados obtenidos en la clasificación de fallas.....	187

Tabla 30 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de vibración y presión	188
Tabla 31 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de vibración y sonido.....	189
Tabla 32 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de vibración y temperatura	189
Tabla 33 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de presión y sonido	190
Tabla 34 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de presión y temperatura.....	191
Tabla 35 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de sonido y temperatura	192
Tabla 36 Comparación entre resultados obtenidos sin reducción de variables y los obtenidos mediante eliminación hacia atrás.....	193
Tabla 37 Comparación entre resultados obtenidos sin reducción de variables y los obtenidos mediante reducción por PCA	193
Tabla 38 Resultados de clasificación de fallas de la banda transportadora al dividir el conjunto de datos 50% para entrenamiento y 50% de prueba.....	194
Tabla 39 Resultados de clasificación de fallas de la banda transportadora al dividir el conjunto de datos 20% para entrenamiento y 80% de prueba.....	194
Tabla 40 Resultados de clasificación de fallas de la banda transportadora utilizando el 10% de las muestras de fallas.	195
Tabla 41 Resultados de clasificación de fallas de la banda transportadora utilizando el 5% de las muestras de fallas.	196
Tabla 42 Resultados de clasificación de fallas de la banda transportadora utilizando el 10% de las muestras de fallas y el 20% de los datos como entrenamiento.	197
Tabla 43 Resultados de clasificación de fallas de la banda transportadora utilizando el 5% de las muestras de fallas y el 20% de los datos como entrenamiento.	198
Tabla 44 Resultado promediado de clasificación de fallas en banda transportadora en condiciones similares al entorno	199

CAPÍTULO 1. INTRODUCCIÓN

El mantenimiento industrial se define como un conjunto de actividades que tienen como objetivo garantizar el correcto funcionamiento de las máquinas e instalaciones que conforman un proceso de producción (Olarte, 2010). Uno de los tipos de mantenimiento que más se aplican dentro de la industria es el mantenimiento correctivo, el cual tiene lugar cuando una falla funcional ocurre de manera inesperada y, por lo general, detiene o afecta a la producción, restaurando al equipo, mediante la reparación o sustitución, a su condición estándar. ¿Por qué no es recomendable aplicar solo este tipo de mantenimiento? La desventaja del mantenimiento correctivo es que puede generar pérdidas de producción, accidentes laborales, incumplimientos a los clientes, problemas de calidad, daños ambientales, entre otras (Montilla, 2016).

Un estudio realizado en el periodo 2021-2022 por Siemens (Siemens, 2023) revela que el tiempo de paro les cuesta a las empresas listadas en Fortune Global 500 el 11% de su facturación actual, que es casi \$1.5 billones de dólares; un aumento considerando que hace 2 años este costo fue de \$864 millones de dólares (8% de la facturación). El costo de una hora perdida actualmente oscila entre \$39000 dólares y más de \$2000000 dólares, dependiendo de la industria (el costo más elevado lo tiene el ramo automotriz). Además, se reporta que una planta de manufactura promedio sufre aproximadamente 20 incidentes que provocan tiempo de paro al mes. Las pérdidas no son solo económicas, según (Vanson Bourne, 2023), el tiempo de paro de producción promedio por fallas inesperadas en los últimos tres años es de 4 horas para las empresas encuestadas, también el 45% de estas mencionaron que, en alguna ocasión, no han podido brindar el servicio al cliente debido a paros ocasionados por fallas inesperadas.

Lamentablemente, cuando una falla se presenta, el mantenimiento correctivo es la única opción. Y, aunque hay estrategias y procedimientos para reducir el tiempo de reparación y reaccionar de mejor manera a estas fallas, esperar a que el equipo falle para actuar no es la mejor opción para reducir costos y cumplir con los objetivos de producción. El principal problema de las fallas inesperadas es que se pueden presentar en el momento menos oportuno, por lo que, cuando estas se presentan, es posible no contar con las refacciones o las herramientas necesarias para realizar el servicio del equipo, prolongado de manera

indefinida el tiempo de producción perdido. Partiendo de esto, la mejor opción para reducir el tiempo de paro es evitar estas fallas inesperadas, por lo que es necesario tratar de predecir cuándo ocurrirán y actuar antes de que sucedan.

El mantenimiento predictivo es aquel que tiene como objetivo predecir fallas en los equipos a partir de ciertos parámetros y factores. Esto implica la constante supervisión del rendimiento y condición del equipo durante la operación de producción. En el momento en el que el parámetro medido alcance determinado nivel, deberá programarse una intervención al equipo y poder restituir su nivel de rendimiento. Por lo tanto, el mantenimiento predictivo debe tener fundamentos, valiéndose solamente de datos reales y concretos recopilados para este objetivo (Montilla, 2016).

El mantenimiento predictivo se apoya en tecnologías y técnicas para recopilar datos sobre la condición del equipo, por ejemplo, análisis de vibraciones para maquinaria rotativa, análisis por ultrasonidos para valorar el funcionamiento de válvulas o detección de fugas, termografías para detección de fallas en instalaciones eléctricas, entre otras (Olarde, 2010).

La necesidad de implementar nuevas tecnologías a la industria ha hecho evolucionar a la industria hacia una nueva era, la llamada “Industria 4.0” la cual se basa en tecnologías de comunicación y de recopilación de información tales como IoT (Internet of things), CPS (Cyber Physical Systems o sistemas ciber físicos), Big Data, etc. (Angelopoulos, 2019). Esta capacidad de comunicación y recopilación de datos ha llevado al mantenimiento predictivo a ser una parte constituyente de la industria 4.0, ya que estas tecnologías lo vuelven más preciso, eficiente y rentable, por lo que el mantenimiento predictivo es un concepto cada vez más presente dentro de la industria.

Como se mencionó anteriormente, la industria 4.0 nos provee de una gran cantidad de datos, la cual es necesario analizar y procesar de tal manera que estos datos se vuelvan información concreta del estado de los equipos. Debido a la gran cantidad de datos, se plantea que la mejor estrategia para la extracción de características o patrones que nos permitan llevar a cabo la predicción de una falla es mediante la aplicación de inteligencia artificial, específicamente Machine Learning (ML). El ML o aprendizaje automático permite que las operaciones sean más autónomas y menos dependientes de la intervención humana, por lo que es una opción adecuada para un análisis preciso de la condición de la maquinaria.

1.1 PLANTEAMIENTO DEL PROBLEMA

La industria manufacturera se enfrenta a un entorno cada vez más competitivo, en el cual la eficiencia y la reducción de costos son claves para garantizar su sostenibilidad y crecimiento. Partiendo de esto, el mantenimiento de los equipos se ha vuelto un factor crítico bastante importante para asegurar la continua operación de la maquinaria. Usualmente, la mayoría de las empresas adoptan enfoques de mantenimiento correctivo y preventivo (hasta el 2022, solamente el 38% de las empresas han adoptado el mantenimiento predictivo (Siemens, 2023)), que aunque pueden parecer eficaces, resultan bastante limitadas. La limitación más significativa que presentan estos enfoques de mantenimiento es la incapacidad para anticiparse a los fallos, lo cual genera paros inesperados, afectando la producción y generando costos adicionales que perjudican a la rentabilidad de la empresa.

La implementación de un programa de mantenimiento predictivo conlleva una serie de problemas que pueden llevar a la empresa a cancelar o posponer este tipo de proyectos. Uno de estos es el elevado costo inicial requerido, el cual abarca la adquisición de herramientas y equipos necesarios para el mantenimiento predictivo, además de la capacitación del personal para el correcto uso de este equipo. Este segundo punto es crítico para un correcto programa de mantenimiento predictivo, ya que la eficiencia de este depende de la experiencia de la persona que interpreta los datos obtenidos de los equipos. Lamentablemente, algunos cambios en la lectura de los datos, que pueden indicar la presencia de alguna falla, pueden pasar inadvertidos para el humano, comprometiendo la eficiencia del mantenimiento predictivo.

La inteligencia artificial (IA) se presenta como una solución viable para solucionar este problema. Mediante el uso de algoritmos y análisis de datos, la inteligencia artificial mejora la capacidad predictiva en el mantenimiento industrial, permitiendo identificar fallas antes de que estas se vuelvan críticas y detengan la operación de la empresa. Sin embargo, a pesar de las ventajas que ofrece la inteligencia artificial, gran parte de las empresas no han adoptado estas tecnologías. Una de las dificultades que se presentan es la falta de claridad sobre que algoritmos se ajustan mejor a las necesidades del mantenimiento industrial. Esto se debe a que existen varios algoritmos, cada uno con sus ventajas y desventajas, que complican la

elección; esto sin mencionar la diversidad de componentes y equipos que existen en la industria. Ante este panorama, surge la necesidad de llevar a cabo una comparación exhaustiva entre distintos algoritmos de inteligencia artificial para identificar cuál es el más efectivo en el ámbito del mantenimiento industrial.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

Comparar las técnicas de Machine Learning más comunes en la literatura para la predicción de fallas intermitentes en sistemas automáticos de manufactura.

1.2.2. OBJETIVOS ESPECÍFICOS

- ❖ Seleccionar los enfoques de Machine Learning que tuvieron mejor desempeño en trabajos previos para la predicción de fallas en sistemas automáticos de manufactura.
- ❖ Seleccionar el tipo de componente del sistema de manufactura y las fallas a predecir.
- ❖ Diseñar y construir un sistema de medición y registro de parámetros de control para los componentes seleccionados.
- ❖ Aplicar los enfoques de inteligencia artificial que tuvieron mejor desempeño para el diagnóstico de fallas en sistemas automáticos de manufactura.
- ❖ Comparar el desempeño de los algoritmos de aprendizaje para el diagnóstico de fallas.

1.3 ANTECEDENTES

Consultando la literatura respecto a la implementación de Machine Learning al mantenimiento predictivo, se han encontrado diversos modelos de diagnóstico y predicción de fallos en distintos componentes usando algoritmos de Machine Learning.

Gran parte de estos modelos tratan de diagnosticar fallas en motores y componentes rotativos. Tal es el caso de (Tayyab et al., 2020), en este artículo se propone un modelo de diagnóstico de fallas para rodamientos y engranajes. Estas fallas son: daño en esferas, anillo externo y

anillo interno para el caso de los rodamientos, mientras que para los engranajes las fallas a clasificar fueron: desgaste en tres dientes consecutivos y un diente del engrane roto. A partir de la medición de la vibración y la extracción de características estadísticas derivadas de la señal en el dominio del tiempo y la curtosis espectral, se obtienen un total de 16 características. Se proponen dos modelos basados en K-Nearest Neighbor (KNN), uno aplicando todas las características extraídas y otro realizando una selección por medio de un algoritmo genético. En el primer modelo se obtuvo una precisión mayor al 90%, mientras que al implementar el algoritmo genético la precisión aumentó al 100%, además de una reducción en el tiempo de cómputo del 73% para los rodamientos y un 48% en el caso de los engranajes. En (Villanueva Llorente, 2022) se realiza una comparación entre varios modelos para el diagnóstico de la condición de rodamientos en motores eléctricos. Para la obtención de muestras que serán procesadas por los distintos modelos se realiza un análisis espectral de la corriente del estator del motor. Se realizaron mediciones variando la frecuencia de la alimentación del motor y simulando cargas por medio de un freno de polvo magnético. Los grados de desgaste del rodamiento fueron clasificados en: rodamiento sano, fallo leve, fallo moderado, fallo grave y rodamiento destruido, se provocaron contaminando la grasa de éste con carburo de silicio. Se utilizaron diversos modelos supervisados y no supervisados como redes neuronales, KNN, SVM (Support Vector Machines), k-means, k-medoids y clustering jerárquico, entre otros. Los resultados obtenidos por medio de los algoritmos de aprendizaje supervisado fueron mejores que los obtenidos por algoritmos de aprendizaje no supervisados, además, la clasificación de fallas es mejor entre menor sea el número de categorías, según este estudio.

También se han realizado modelos de detección y diagnóstico de fallas para válvulas, elementos muy comunes dentro de la industria manufacturera. Un ejemplo de esto es (Amaya, 2022) en la que se realiza la detección de fallas en válvulas antirretorno, este modelo es capaz de determinar si una válvula se encuentra en buen estado o no. Para esto, se midieron distintos parámetros que son: la vibración, la emisión acústica y ultrasonidos y se extraen distintas características en el dominio del tiempo. En este trabajo se comparan diversas estructuras de redes neuronales, tales como la Patternet, Googlenet, Squeezenet y una red convolucional propuesta por el investigador. En el caso de la red Patternet, primeramente se realiza una reducción dimensional mediante Análisis de Componentes Principales (PCA, por

sus siglas en ingles), t-SNE (T-distributed Stochastic Neighbor Embedding) y UMAP (Uniform Manifold Approximation and Projection for Dimension Reduction), posteriormente se realiza la selección de características con SFS (Sequential Feature Selector) y una clasificación con el algoritmo K-Means; todo esto con el fin de realizar un entrenamiento no supervisado y compararlo con los métodos supervisados. Como resultado se obtuvo que las mejores características para diagnosticar el estado de válvulas antirretornos son la vibración en el eje x y la emisión acústica. Además, en el caso de los métodos no supervisados se obtuvieron mejores resultados al realizar la reducción con t-SNE, y, en el caso de los supervisados, las estructuras Googlenet y Squeezenet mostraron un mejor desempeño al clasificar los datos. Estos modelos no se limitan solo a la clasificación de fallas, sino que pueden llegar incluso a predecir la vida útil restante de la válvula. En (Utah & Jung, 2020) se realiza una clasificación de fallas en una válvula activada por solenoide, las cuales son: salida bloqueada, sin flujo o sin carga, degradación del embolo en niveles 1,2,3 y 4, fallo al abrir y fallo al cerrar. Para esto se realiza un análisis de la corriente eléctrica tanto en el dominio del tiempo como en el de la frecuencia. Se compararon diversos algoritmos de clasificación como el SVC (Support Vector Classification), KNN, Árbol de decisión y Random Forest, siendo SVC el algoritmo con el se obtuvieron mejores resultados con un 94% en todas las métricas de evaluación. Para la predicción de la vida útil restante de la válvula se utilizó una red neuronal profunda propuesta por el autor y el modelo de regresión XGBoost, ambos modelos fueron entrenados con el nivel de degradación del émbolo, obteniendo una predicción bastante precisa con el modelo XGBoost reggresor.

No solo es posible realizar la clasificación de fallas a componentes, sino que esta clasificación se puede ampliar a todo un proceso si es necesario. En (Acosta Ruiz, 2023), se realiza un diagnóstico de fallas en un sistema de clasificación de cajas simulado en el software Factory I/O. Se realizó la detección de fallas en las cuatro bandas transportadoras del sistema y del sensor de peso que clasifica las fallas. Por medio de 16 variables las cuales son RPM (revoluciones por minuto) de los motores de las distintas bandas, salida del sensor de peso, sensores de presencia y medición del tiempo es posible realizar una clasificación de fallas de los distintos conveyors y en el sensor que pesa las cajas. Los modelos de machine learning que se utilizaron fueron: Red neuronal multicapa, árbol de decisión y random forest. Para mejorar la clasificación se seleccionaron variables mediante la eliminación hacia atrás.

Obteniendo el 100% de precisión con Random Forest y árbol de decisiones y un 95.3% mediante la red neuronal. Otro ejemplo de esto es (Márquez Vera et al., 2021). En este artículo se realiza la clasificación de un total de 20 fallas en el proceso “Tennessee Eastman”, el cual es un modelo de un proceso industrial real. Empleando 41 variables obtenidas por medio de los distintos sensores del proceso donde se proponen dos modelos para la clasificación de fallas. En el primero se somete la señal a una wavelet tipo sombrero mexicano y se realiza la reducción de dimensiones mediante Análisis de componentes independientes (ICA), posteriormente, todos estos datos entrenan una red neuronal multicapa para la clasificación de las distintas fallas. En el segundo modelo, la selección de variables se realiza mediante una red neuronal elástica y la clasificación se realiza mediante una red neuronal recurrente LSTM. Con ambos modelos se obtiene una detección de fallas bastante precisa, pero en el caso del segundo modelo, tanto la detección como la clasificación de estas es de casi el 100%, superando al primer modelo.

1.4 JUSTIFICACIÓN

La competencia y la necesidad de maximizar la eficiencia de los procesos en la industria han llevado a las empresas a buscar soluciones que les permitan ser más competitivas. La inteligencia artificial es una herramienta clave para este propósito, ya que ofrece la posibilidad de optimizar el proceso de mantenimiento gracias a su capacidad para analizar grandes cantidades de datos y realizar predicciones.

La importancia de este estudio se debe a la incertidumbre que existe sobre que algoritmos de inteligencia artificial son los más adecuados para aplicarlos al mantenimiento industrial. Al realizar una comparación entre varios algoritmos, se busca proporcionar información a las empresas que les permita tomar decisiones para una correcta implementación de estas tecnologías avanzadas al mantenimiento. Con la identificación del algoritmo más efectivo, se obtendrán algunos beneficios, uno de estos será el impacto económico, ya que se reducirá la pérdida de dinero provocado por paros de producción debido a mantenimiento, al saber en qué momento será necesario cambiar una pieza se reducen inventarios de refacciones,

reduciendo el costo de almacén. Además, la producción se incrementa, por lo que es más fácil lograr la meta o la orden del cliente, mejorando la relación con éste.

Además, esta investigación contribuirá al desarrollo de un marco de referencia que facilite la integración de la inteligencia artificial a la industria manufacturera, ofreciendo orientación sobre que algoritmos utilizar según las características de los datos. Esto no solo beneficiará a las empresas de manera aislada, sino que también tendrá un impacto positivo en la industria en general, promoviendo la innovación y el desarrollo.

CAPÍTULO 2. MARCO TEÓRICO

En la industrial actual, una gestión eficiente del mantenimiento es indispensable para garantizar el buen funcionamiento y la competitividad de las empresas. El mantenimiento predictivo asistido por inteligencia artificial ha surgido como una estrategia innovadora que permite a las empresas anticipar y prevenir fallas en los equipos antes de que se conviertan en problemas más graves y costosos, tal y como se menciona en (Angelopoulos, 2019). A diferencia de los enfoques tradicionales, que se basan en el mantenimiento correctivo o preventivo, el mantenimiento predictivo utiliza datos en tiempo real para evaluar el estado de las máquinas y tomar decisiones informadas sobre intervenciones necesarias.

La inteligencia artificial se ha ganado un papel importante en el mantenimiento predictivo al permitir el análisis de grandes volúmenes de datos generados por sensores y sistemas de monitoreo y recopilación de datos. Mediante técnicas de aprendizaje automático y minería de datos, las empresas pueden identificar patrones y tendencias que indican un desgaste o mal funcionamiento en los equipos. Esto ayuda a mejorar la disponibilidad de los activos, reducir el riesgo de fallos inesperados y minimizar el tiempo de paro.

Este marco teórico examinará los principios del mantenimiento y la inteligencia artificial, con esto se pretende resaltar el impacto significativo que la inteligencia artificial puede tener en la optimización de las estrategias de mantenimiento y en la sostenibilidad de las operaciones industriales.

2.1 MANTENIMIENTO

La ISO 14224 define al mantenimiento como la “combinación de acciones técnicas y administrativas, incluyendo supervisión, cuyo fin es mantener o reparar el aparato para que opere en un estado que le permita realizar las funciones requeridas” (International Organization for Standardization, 1999). Partiendo de esta definición se podría pensar el mantenimiento tiene como objetivos, únicamente, el mantener los equipos en funcionamiento y restablecer el funcionamiento del equipo en condiciones estándar, pero el mantenimiento no solo se limita a estos objetivos, sino que toma un papel activo dentro del cumplimiento de

los objetivos de la empresa. Como es sabido, el fin de una empresa es fabricar productos, proveer servicios y generar utilidades para los accionistas, con base en este planteamiento, el mantenimiento debe apoyar a cumplir estas metas mediante el cumplimiento de los siguientes objetivos (Montilla, 2016).

2.1.1 OBJETIVOS DE MANTENIMIENTO

- ❖ **Minimizar las fallas funcionales (paradas imprevistas) de los equipos, y los costos asociados a ellas:** Las fallas funcionales son aquellas que impiden que un equipo o máquina continúe operando. Estas fallas acarrearán una serie de consecuencias, que pueden ser: la detención de la producción, pérdidas de materia prima y producto en proceso, defectos de calidad, riesgo de accidente laboral, riesgo de daño al medio ambiente, incremento de gastos por refacciones y mano de obra de mantenimiento correctivo, incumplimiento a los clientes, horas extras del personal, entre otras. Debido al gran efecto que tienen estas fallas dentro de una empresa es imprescindible el tratar de reducir el número y la recurrencia de éstas.
- ❖ **Asegurar unos costos operativos razonables de los equipos, y procurar mejoras de ellos:** El costo de operación se define como el costo variable para operar y mantener productiva la máquina. Este costo se puede dividir en tres rubros: consumo de combustible, consumo de lubricantes y costos de mantenimiento; y se calcula de la siguiente manera:

$$CO = C_{co} + C_{lu} + CM + C_{rm} + C_{pd} \quad (1)$$

Donde, CO es el costo de operación de la máquina, C_{co} es el costo de los combustibles necesarios para que la máquina opere (electricidad, gas, gasolina, etc.), C_{lu} es el costo de lubricación (aceites, grasas y filtros), CM es el costo de labores rutinarias de mantenimiento (incluye mano de obra, refacciones, herramienta y equipo necesarios), C_{rm} es el costo de reparaciones mayores no rutinarios como reparaciones anuales o

modificaciones del equipo y C_{pd} es el costo de las partes de desgaste (aquellas que entran en contacto con la materia prima, como troqueles, cuchillas de corte, etc.).

- ❖ **Maximizar la inversión en planta y equipos, asegurando el cumplimiento de mínimo su vida útil esperada:** Cuando se adquiere una máquina se realiza una proyección de costos tomando como referencia una cierta cantidad de años de vida útil de la misma, el costo de adquisición y reposición, la tasa de producción y los costos de operación. Se espera que al finalizar la vida útil del equipo, se retorne la inversión realizada en éste, además del capital suficiente para la adquisición del equipo que lo reemplazará. El costo de posesión se refiere a la cantidad de dinero que el propietario ha invertido para adquirir la maquinaria o el equipo, y constituye un gasto continuo, ya sea que se utilice o no.

$$CP = VRN + DP + SI \quad (2)$$

Donde, CP es el costo de posesión de la máquina, VRN es el valor de reposición o nuevo (es el costo del equipo al momento de la adquisición o el costo de reponer el equipo una vez que llegó al final de su vida útil), DP es la depreciación que sufre una maquina debido al deterioro que sufre durante su operación y SI es el valor de las pólizas de seguros e impuestos que se deben pagar por la posesión del equipo. Sumando el costo de posesión y el costo de operación se extrae el Costo de Operación y Posesión (COP)

- ❖ **Asegurar que los equipos operen de manera segura para los usuarios y para el medio ambiente:** Los equipos deben de operar de manera segura para los usuarios y evitando arrojar desecho que sobrepasen los límites permitidos, de no ser así, se pueden generar multas y sanciones que llevan a la empresa al cierre. Debido a esto mantenimiento debe tener en cuenta las normas ambientales (ISO 14000) y de Seguridad Industrial y Salud Ocupacional (OHSAS 18000) durante sus labores.

2.1.2 INDICADORES DE MANTENIMIENTO

Para poder evaluar el cumplimiento de estos objetivos es necesario la implementación de Índices o Indicadores. Un indicador se define como “la relación entre las variables

cuantitativas o cualitativas, que permite observar la situación y las tendencias de cambio generadas en el objeto o fenómeno observado, respecto de objetivos y metas previos e influencias esperadas” (Jaramillo, 1998). Los indicadores permiten conocer si se están cumpliendo la misión, objetivos y metas, además de tomar decisiones de manera objetiva. Existen diversos índices los cuales se pueden clasificar en índices de gestión de equipamientos, gestión de costos y gestión de mano de obra. De todos los indicadores que existen, hay seis que se denominan “indicadores de clase mundial”, los cuales son utilizados en todo el mundo, de los cuales, cuatro pertenecen a indicadores de gestión de equipamientos y dos a gestión de costos (Tavares, 1999).

- ❖ **MTBF (Mean Time Between Failures):** El tiempo medio entre fallas permite conocer el tiempo de actividad total de las máquinas o, dicho de otra manera, nos permite conocer la frecuencia con que suceden las averías. También, se le conoce como confiabilidad.

$$MTBF = \frac{\text{Número de horas totales del periodo de tiempo analizado}}{\text{Número de averías}} \quad (3)$$

- ❖ **MTTR (Mean Time to Repair):** El tiempo medio de reparación mide la capacidad para corregir la falla, o, el grado de importancia de ésta considerando el tiempo requerido para restituir el equipo.

$$MTTR = \frac{\text{Número de horas de paro por avería}}{\text{Número de averías}} \quad (4)$$

- ❖ **Disponibilidad:** La disponibilidad indica la capacidad de una máquina de llevar a cabo con éxito la función requerida, muestra cuanto tiempo puede operar según lo programado.

$$\begin{aligned} \text{Disponibilidad} &= \frac{\text{Total de horas} - \text{Horas de paro por mantenimiento}}{\text{Total de horas}} * 100 \\ &= \frac{MTBF}{MTBF + MTTR} * 100 \end{aligned} \quad (5)$$

- ❖ **MTTF (Mean Time to Failure):** El tiempo medio hasta el fallo mide el tiempo medio entre fallos, pero, solo se aplica a activos que no son reparables, o sea, a equipos que no se pueden reparar o es más eficiente y barato sustituirlos, como rodamientos de bolas o bombillas. Dicho de otro modo, se refiere a la vida útil promedio de estos activos.

$$MTTF = \frac{\text{Número de horas totales}}{\text{Número de activos}} \quad (6)$$

- ❖ **Costo de Mantenimiento por Facturación:** Es la relación entre el costo total de mantenimiento y la facturación o ganancias de la empresa en el periodo considerado. Este índice es útil ya que permite evaluar la eficiencia de la gestión del mantenimiento, además que, un costo elevado de mantenimiento puede llegar a repercutir en el precio final del producto.

$$CMF = \frac{\text{Costo total de mantenimiento}}{\text{Facturación en bruto}} * 100 \quad (7)$$

- ❖ **Costo de mantenimiento por el Valor de Reposición:** Es la relación entre el costo total de mantenimiento de un equipo y el valor de compra de ese mismo equipo nuevo. Este índice analiza el costo de mantenimiento para determinar si es conveniente seguir manteniendo el equipo o adquirir uno nuevo y debe ser calcula para los activos más importantes o críticos dentro de la empresa.

$$CPMV = \frac{\text{Costo total de mantenimiento}}{\text{Valor de reemplazo del activo}} * 100 \quad (8)$$

El valor máximo sugerido para este índice es de 6%, aunque no se debe considerar solo esta métrica para decidir sustituir el equipo, sino que hay que considerar distintos factores como el plazo de entrega de un nuevo equipo o el costo que tendrá poner en marcha el equipo nuevo (Fractal, s.f.).

Como se puede observar, las fallas funcionales afectan demasiado a los indicadores, por lo que el atender estas fallas no debe ser la prioridad de mantenimiento, sino llevar una planificación correcta y anticiparse a las fallas. En el caso de que una falla aparezca, se debe analizar cómo afecta a los índices y, con base en esto, aplicar mejoras y llevar una mejor gestión del mantenimiento.

2.1.3 TIPOS DE MANTENIMIENTO

Los objetivos de mantenimiento son muy variados, por lo tanto, existen distintos tipos de mantenimiento, cada uno con funciones y objetivos distintos. Dentro de la industria no se utiliza un solo tipo de mantenimiento, sino que se debe realizar una planeación adecuada que integre los diferentes tipos, con el fin de optimizar los recursos y la disponibilidad de los equipos (González et al., 2007). Los tipos de mantenimiento más comunes dentro de la industria son los siguientes:

Mantenimiento correctivo

El mantenimiento correctivo, también conocido como mantenimiento reactivo, es aquel cuyo objetivo es poner en marcha el funcionamiento de la máquina cuando esta deja de operar debido a una falla, afectando la productividad el menor tiempo posible. Este tipo de mantenimiento, a nivel industrial, es utilizado en un alto porcentaje en países de Latinoamérica y muchos otros países subdesarrollados (Pérez, 2021).

El mantenimiento correctivo no se limita solamente a reparar la avería del equipo, sino que también busca diagnosticar y corregir la causa que provocó dicha avería; previniendo así, la rápida reaparición de la falla (González et al., 2007). Existen dos tipos o clases de mantenimiento correctivo (Montilla, 2016):

- ❖ **Mantenimiento correctivo de emergencia:** El mantenimiento correctivo de emergencia, también conocido como mantenimiento correctivo no programado, tiene lugar cuando ocurre una falla funcional de manera inesperada dentro de la jornada de producción, la cual detiene o trastorna la producción.
- ❖ **Mantenimiento correctivo programado:** Tiene lugar cuando se detecta una falla potencial, esto se refiere al tipo de falla que no inhabilita a la máquina pero en un

determinado momento propicia las condiciones para que ocurra un paro en la jornada de producción, pero su efecto no es dramático y permite culminar la jornada o el ciclo de producción para realizar la reparación.

Aplicar este tipo de mantenimiento ofrece algunas ventajas, como prolongar la vida útil de los equipos ya que se realiza el reemplazo hasta que la vida útil del componente se agota, en el corto plazo resulta un mantenimiento económico, no requiere planificación ni programación, además que, en algunos sistemas, el mantenimiento correctivo es la única opción ya que el mantenimiento preventivo no tiene efecto, como los dispositivos electrónicos. Por otra parte, las desventajas que presenta el mantenimiento correctivo son numerosas, por ejemplo, la falla se puede presentar en un momento inoportuno, además que estas fallas pueden ocasionar daños más complejos o irreparables en los equipos, grandes inventarios de refacciones, la producción se vuelve impredecible y poco fiable, costos elevados a mediano y largo plazo, y, las fallas pueden no solo afectar a la producción, sino que ponen en riesgo al personal y al medio ambiente (Pérez, 2021).

Mantenimiento preventivo

El mantenimiento predictivo se basa en la ejecución de tareas básicas previamente planificadas, como inspecciones, evaluaciones, cambios o reemplazos, calibración, lubricación, reparación, etc., a frecuencias predeterminadas. Esto con el propósito de prevenir la ocurrencia de fallas en las máquinas y garantizar que esta cumpla con las funciones requeridas en la jornada de producción. Además de las tareas básicas, se realizan algunas tareas programadas adicionales que pueden ser modificaciones en los equipos, mantenimiento correctivo programados, etc. (Montilla, 2016).

El mantenimiento preventivo más común es el mantenimiento preventivo planificado (Planned Preventive Maintenance), el cual consiste en establecer una rutina de sustitución de piezas en intervalos periódicos de tiempo. Esta sustitución se realiza sistemáticamente, en la mayoría de los casos, basándose en el tiempo de trabajo de la máquina, o el número de ciclos realizados, y la información histórica del tiempo medio entre fallos (MTBF) del componente, sin considerar la condición de la pieza a sustituir. Es importante elegir correctamente los intervalos de sustitución de piezas, ya que de esto depende el éxito del mantenimiento preventivo planificado en la reducción de fallos inesperados. También existe otro tipo de

mantenimiento preventivo llamado mantenimiento preventivo de oportunidad, que consiste en realizar acciones de mantenimiento preventivo aprovechando algún paro, ya sea por avería o por cuestiones de producción, etc. (González et al., 2007).

Las ventajas del mantenimiento preventivo son: Aumento de la disponibilidad de los equipos, ya que se disminuyen las fallas y los tiempos muertos, aumento en la vida útil de las máquinas, componentes e instalaciones, se reducen los inventarios de refacciones y sus costos, ahorro económico a mediano y largo plazo, ya que el costo de reparación es menor, se elabora un plan de mantenimiento y se definen indicadores de mantenimiento, se aumenta la seguridad para el personal y se mejora el enfoque de contaminación ambiental, entre otras. Mientras que las desventajas de este tipo de mantenimiento son: generación de sobrecostos debido al reemplazo de piezas sin suficiente desgaste, para iniciar y realizar un programa de mantenimiento se requiere de tiempo extra del personal, búsqueda de información (en manuales, historiales, inventarios, etc.), además de una inversión inicial y personal mejor capacitado (Pérez, 2021).

Mantenimiento predictivo

El mantenimiento predictivo, también conocido como mantenimiento basado en la condición, es una fase avanzada del mantenimiento preventivo, ya que este corrige las desventajas del mantenimiento preventivo planificado, reemplazando las sustituciones periódicas por inspecciones periódicas para evaluar el estado de la máquina. El mantenimiento predictivo asocia la relación de parámetros físicos con el desgaste o estado del equipo, cuando los parámetros medidos detectan la inminencia de un fallo, o la avería de algún componente, se actúa con un plan de mantenimiento, reparando o sustituyendo las piezas dañadas o desgastadas antes de que fallen. De esta manera, se reducen los tiempos de paro y se aprovecha al máximo la vida útil de los componentes (Pérez, 2021) (González et al., 2007). Para la medición de los parámetros generalmente se utilizan técnicas no invasivas y sin necesidad de parar la máquina ni interrumpir la producción, en algunos casos, esta medición se realiza de forma continua, dando lugar al mantenimiento predictivo continuo, en otros casos, la medición se realiza de manera periódica. En este último caso, se debe fijar el intervalo de inspección de manera que permita detectar variaciones en la condición del equipo, y, en caso de que las haya, poder programar una intervención para corregir o sustituir

los elementos. Los parámetros más utilizados como indicador del estado del equipo dentro de la industria son: el nivel de ruido, nivel de vibración (el cual es el más utilizado ya que permite detectar con mayor fiabilidad un gran número de fallas potenciales), nivel de partículas metálicas en el lubricante, la temperatura, caudal, presión, intensidad de corriente, voltaje, entre otras (González et al., 2007).

Las principales ventajas del mantenimiento predictivo son: la vida útil de las piezas se aprovecha al máximo (disminuyendo el número de intervenciones), se reduce la necesidad de almacenamiento de piezas (ya que permite adquirir las piezas a sustituir con antelación), permite no solo detectar los síntomas de la falla sino también detectar la causa de ésta, la intervención al equipo se vuelve más rápida ya que se detecta previamente el punto a corregir, las pruebas y mediciones se realizan con la máquina en operación (por lo tanto, no afecta al proceso de producción). Como desventajas del mantenimiento predictivo se tiene que se debe realizar una gran inversión para equipos de medición, registro de parámetros y elaboración de base de datos, y se requiere disponer de un personal calificado para la utilización de los equipos, así como para la interpretación de los valores de los parámetros. Otro factor que se puede tomar como una desventaja es que el mantenimiento predictivo, debido a su alto costo, está indicado solo en aquellas máquinas en las que un fallo produce un riesgo elevado para la seguridad o un costo elevado, ya sea por la posibilidad de un fallo catastrófico de la máquina (máquinas caras o únicas) o por provocar paros en la producción (máquinas críticas para el proceso de producción) (González et al., 2007).

2.1.4 ANÁLISIS DE CRITICIDAD

En una empresa no todos los equipos tienen la misma importancia, además, los recursos de la empresa son limitados, por lo que se debe destinar la mayor parte de estos recursos a los equipos de mayor importancia dentro de la empresa. Para poder realizar esta diferenciación es necesario realizar un análisis de criticidad basado en el impacto e importancia del equipo dentro de la organización, el nivel de criticidad se clasifica mediante el método ABC (Garrido, 2010):

- A. **Equipos críticos:** Son aquellos cuyo paro o mal funcionamiento tiene un impacto considerable en los resultados de la empresa.

- B. **Equipos importantes:** Son aquellos cuyo paro, avería o mal funcionamiento afecta a la empresa, pero las consecuencias son asumibles.
- C. **Equipos prescindibles:** Son aquellos con una influencia mínima en los resultados. En el peor de los casos, supondrían una ligera incomodidad, un cambio poco relevante o un costo adicional menor.

Una vez que se obtienen los niveles de criticidad, es necesario determinar los criterios que se utilizarán para clasificar cada uno de los equipos en los distintos niveles de criticidad. Para poder clasificar a los equipos es necesario considerar la influencia que las fallas tienen en los aspectos de producción, calidad, mantenimiento y seguridad.

- ❖ **Producción:** Se valora como un fallo en la máquina afecta a la producción, ya que este puede detener todo el proceso o una zona de producción preferente para equipos productivos o que no influya en la producción.
- ❖ **Calidad:** El equipo puede tener un impacto determinante en la calidad del producto o servicio final, un impacto moderado que generalmente no ocasiona problemas, o ninguno en absoluto.
- ❖ **Mantenimiento:** El equipo puede presentar averías frecuentes o muy caras, o con un costo mediano, o que no falle frecuentemente y el costo de reparación sea bajo.
- ❖ **Seguridad y medio ambiente:** Un fallo del equipo puede suponer un accidente muy grave para las personas o el entorno; también es posible que un fallo cause un accidente, pero con baja probabilidad; o puede tratarse de un equipo que no tenga ningún impacto en la seguridad.

Ya que se explicaron los criterios bajo los que se clasificará la criticidad de un equipo, en (Garrido, 2010) se propone la siguiente tabla (ver Tabla 1), también conocida como matriz de criticidad:

Tabla 1 Matriz de criticidad.

Tipo de equipo	Seguridad y medio ambiente	Producción	Calidad	Mantenimiento
A CRÍTICO	Puede originar accidente muy grave.	Su parada afecta al plan de Producción.	Es clave para la calidad del producto.	Alto coste de reparación en caso de avería.
	Necesita revisiones periódicas frecuentes(mensuales).		Es el causante de un alto porcentaje de rechazos.	Averías muy frecuentes.
	Ha producido accidentes en el pasado.			Consume una importante de los recursos de mantenimiento (mano de obra y/o materiales).
B IMPORTANTE	Necesita revisiones periódicas (anuales).	Afecta a la producción, pero es recuperable (no llega a afectar al cliente o al Plan de Producción).	Afecta a la calidad, pero habitualmente no es problemático.	Coste Medio en Mantenimiento.
	Puede ocasionar un accidente grave, pero las posibilidades son remotas.			
C PRESCINDIBLE	Poca influencia en seguridad.	Poca influencia en producción.	No afecta a la calidad.	Bajo coste de mantenimiento.

La categoría asignada a la máquina que se está evaluando será el nivel más alto obtenido en cualquiera de los cuatro aspectos.

A cada uno de los niveles de criticidad corresponde un sistema o plan de mantenimiento distinto. Para los equipos de nivel A, se plantea un plan de mantenimiento basado en los tres tipos de mantenimiento (correctivo, preventivo y predictivo), ya que el equipo deberá contar con una alta disponibilidad (por encima del 90%) y una frecuencia de fallas bastante baja

para maximizar la producción o reducir el riesgo de que ocurra un accidente o catástrofe. Para los equipos de nivel B, el plan de mantenimiento deberá estar basado en los tipos de mantenimiento correctivo y predictivo; y, por último, para los equipos de nivel C, la aplicación de mantenimiento correctivo será suficiente. De esta manera, se aprovechan los recursos al máximo, ya que se evitan costos en equipos que resultan prescindibles para el proceso de producción y se asegura el cumplimiento del plan de producción y la seguridad de los empleados.

2.1.5 TÉCNICAS DE MANTENIMIENTO PREDICTIVO

Como se mencionó anteriormente, el mantenimiento predictivo está basado en el análisis del estado de la máquina. Este análisis generalmente se realiza mediante la medición de parámetros representativos de su condición. En la mayoría de los casos, estos parámetros se miden sin necesidad de detener la máquina ni interrumpir la producción. En algunos casos el valor de estos parámetros se mide de forma continua mientras que en otros la medida se realiza con una periodicidad definida. El intervalo de inspección debe fijarse en un tiempo que permita detectar variaciones en el estado de la máquina, en caso de que las haya habido, y corregir o reemplazar los componentes necesarios antes de que se produzca el fallo. En muchos casos el intervalo puede cambiar en función del estado de la máquina, reduciéndose cuando los parámetros indiquen un fallo cercano, es decir, cuando se supere un cierto nivel de alerta. Así, cuando se estima que el fallo se producirá en poco tiempo, la vigilancia de los parámetros se realiza con mayor frecuencia. A su vez, cuando otro nivel, considerado de alarma, se supera, se procede a la reparación.

Análisis de vibraciones

Todas las máquinas vibran durante su funcionamiento y cada una presenta una vibración característica que la diferencia de las demás, llamada firma de vibración. Esta vibración depende de varios factores como su diseño, fabricación, condiciones de operación y estado del equipo. Cuando un equipo se encuentra funcionando en condiciones óptimas, éste genera una vibración con una intensidad considerada como normal, pero cuando el equipo se deteriora se incrementa la intensidad de la vibración. Por este motivo la vibración es el

parámetro más utilizado en el mantenimiento predictivo, ya que bajos niveles de vibración indican que el equipo se encuentra en buen estado, mientras que si estos niveles se incrementan es un indicio claro de que algo comienza a ir mal. Mediante el análisis de vibraciones es posible detectar fácilmente problemas como: desequilibrio, desalineación, holguras, aflojamientos, problemas de lubricación en cojinetes, daños en rodamientos, en engranajes, en motores eléctricos, problemas hidráulicos, etc. (González et al., 2007)(Power MI, 2018).

Según (González et al., 2007) para establecer el límite aceptable de vibración en una máquina, es importante considerar los siguientes aspectos:

Según (González et al., 2007) para establecer el límite aceptable de vibración en una máquina, es importante considerar los siguientes aspectos:

- ❖ Es crucial evitar que la vibración no interfiera con el correcto desempeño de la máquina, como por ejemplo, asegurando que la vibración de una rectificadora no afecte al acabado superficial que se desea obtener.
- ❖ Se busca asegurar que la vibración aceptable no cause un desgaste prematuro significativo en los componentes de la máquina.
- ❖ Es importante que la vibración de la máquina no afecte el funcionamiento adecuado de otras máquinas cercanas ni comprometa la integridad de otros sistemas o instalaciones.
- ❖ Se debe prevenir cualquier efecto perjudicial en las personas causado por la vibración de la máquina. Las vibraciones pueden afectar a las personas de distintas maneras: afecciones osteoarticulares, afecciones neurológicas, afecciones vasculares, alteraciones musculares, afecciones de la columna vertebral, etc. (Díaz, 2002).

Establecer límites estándares aceptables para la vibración en una variedad de máquinas es un proceso complicado debido a la gran diversidad de equipos que hay dentro de la industria. Cada máquina tiene sus propias características y condiciones operativas únicas que pueden influir en sus niveles de vibración. Sin embargo, para proporcionar una guía inicial, existen normas como la ISO 10816, que ofrece una clasificación y directrices para la evaluación de la vibración en diferentes tipos de máquinas; y así, posteriormente establecer los límites de vibración admisibles específicos de cada equipo.

En la norma ISO 10816, en la sección 3, se clasifican los equipos en función de su tamaño y características específicas. A continuación se presenta la clasificación típica que se encuentra en esta sección (Power MI, 2018):

- ❖ **Grupo 1:** Máquinas rotatorias grandes con potencia superior 300 kW. Máquinas eléctricas con altura de eje $H \geq 315$ mm.
- ❖ **Grupo 2:** Máquinas rotatorias medianas con potencia entre 15 y 300 kW. Máquinas eléctricas con altura de eje $160 \leq H \leq 315$ mm.
- ❖ **Grupo 3:** Bombas con impulsor de múltiples álabes y con motor separado (flujo centrífugo, axial o mixto) con potencia superior a 15 kW.
- ❖ **Grupo 4:** Bombas con impulsor de múltiples álabes y con motor integrado (flujo centrífugo, axial o mixto) con potencia superior a 15 kW.

Además de clasificar los equipos según su tamaño y potencia, también se considera la flexibilidad del soporte sobre el cual están montados estos equipos. La flexibilidad del soporte se divide en dos categorías principales:

- ❖ **Máquinas montadas sobre soportes rígidos:** Estas son máquinas que están montadas sobre fundaciones o cimientos que son relativamente rígidos. Esto significa que ofrecen poca flexibilidad o amortiguación a las vibraciones generadas por la máquina. Las vibraciones en estas máquinas tienden a propagarse con menos atenuación a través de la estructura del edificio o la planta.
- ❖ **Máquinas montadas sobre soportes flexibles:** Por otro lado, estas son máquinas montadas sobre fundaciones o cimientos que son relativamente flexibles. Esto significa que ofrecen cierta capacidad de absorción de vibraciones y pueden reducir la transmisión de vibraciones a la estructura circundante.

La norma establece que si la primera frecuencia natural del sistema máquina-soporte en la dirección de la medición es al menos un 25% mayor que su frecuencia principal de excitación (es decir, la frecuencia de rotación), entonces el sistema de soporte se puede considerar rígido en esa dirección.

En la tabla 2 se muestran los niveles de vibración permitidos en la norma ISO 10816 para cada una de las clases:

Tabla 2 Severidad de la vibración según norma ISO 10816-3.

ISO 10816-3 Estándar de Vibración		Grupo 4		Grupo 3		Grupo 2		Grupo 1	
		Motor integrado		Motor separado		Motores 160 mm ≤ H < 315 mm		Motores 315 mm ≤ H	
Velocidad		Bombas > 15kW Flujo radial, axial o mixto				Tamaño medio 15 kW < P ≤ 300 kW		Grandes máquinas	
mm/s rms	in/s rms								
11 7.1 4.5 3.5 2.8 2.3 1.4 0.71	0.43 0.28 0.18 0.14 0.11 0.09 0.06 0.03								
					D				
					C				
					B				
Base		Rígido	Flexible	Rígido	Flexible	Rígido	Flexible	Rígido	Flexible

Donde; A corresponde a valores de vibración de máquinas nuevas o reacondicionadas; B a niveles de vibración donde los equipos pueden funcionar indefinidamente; C indica que la condición de la máquina no es apta para una operación continua, sino solo para un tiempo limitado; mientras que D señala que los niveles de vibración son peligrosos y la máquina podría sufrir daños.

Para medir la intensidad de la vibración se usan transductores. El transductor es un componente esencial en la medición de vibraciones, ya que convierte la energía mecánica generada por la vibración en una señal eléctrica que puede ser procesada. La selección del tipo de transductor elegido dependerá del parámetro que se desee medir. Existen diferentes tipos de transductores diseñados para medir diferentes parámetros de vibración, como el desplazamiento, la velocidad y la aceleración (González et al., 2007).

- ❖ **Transductores de desplazamiento:** Estos transductores son utilizados para medir el movimiento relativo entre el eje y la carcasa de la máquina. Funcionan mediante el principio de las corrientes de Eddy, que detectan cambios en el campo magnético generado por el movimiento relativo entre dos superficies conductoras.

- ❖ **Transductores de velocidad:** Estos transductores miden la velocidad de la carcasa de la máquina y pueden ser útiles para evaluar el movimiento oscilatorio o rotatorio de la máquina. Sin embargo, su uso puede ser menos común en comparación con los acelerómetros debido a sus limitaciones en la medición de vibraciones de alta frecuencia y su sensibilidad a las variaciones en la velocidad de la máquina.
- ❖ **Acelerómetros:** Los acelerómetros son ampliamente utilizados en la medición de vibraciones debido a su versatilidad y capacidad para medir la aceleración experimentada por la carcasa de la máquina. Los acelerómetros piezoeléctricos son los más comunes, estos generan una señal eléctrica en respuesta a la deformación mecánica producida por la vibración.

Las medidas de desplazamiento son ideales para vibraciones de baja frecuencia y son un indicador del esfuerzo o la deformación que sufre la máquina. Por otra parte, las mediciones de velocidad son muy útiles, ya que la velocidad indica la severidad de la vibración y está directamente relacionada al nivel de fatiga que está soportando el sistema mecánico. Las mediciones de aceleración son más efectivas para estudiar fenómenos a altas frecuencias y ofrecen una mejor visión de las fuerzas internas relacionadas a una fuente de vibración particular. Convertir una señal de desplazamiento a velocidad o de velocidad a aceleración es necesario tomar la derivada de la señal. Por el contrario, pasar de aceleración a velocidad o de velocidad a desplazamiento es como integrar la señal. El problema con la derivación es que ésta puede distorsionar la señal o añadir ruido, mientras que la integración se puede llevar a cabo con precisión utilizando circuitos electrónicos económicos o mediante software (Power MI, 2018).

Además de la variable de interés, la elección del transductor depende de otros factores:

- ❖ **La rigidez relativa entre eje y carcasa y el amortiguamiento de la vibración:** Si el nivel de amortiguamiento de la vibración desde el eje hasta la carcasa es alto, se recomienda realizar la medición sobre el eje con un transductor de desplazamiento, mientras que si el amortiguamiento es bajo se puede medir la vibración desde la carcasa.
- ❖ **La frecuencia de interés:** Si la frecuencia de muestreo necesaria es alta, entonces es mejor optar por un acelerómetro, de no ser así, se puede utilizar un transductor de desplazamiento o velocidad; o bien, utilizar un acelerómetro e integrar la señal.

- ❖ **El peso del transductor:** Si el peso del transductor puede alterar la frecuencia natural del sistema, es preferible utilizar un acelerómetro, debido a su reducido tamaño y peso. Sin embargo, este problema no suele ser relevante en el mantenimiento predictivo industrial, ya que las máquinas suelen ser mucho más grandes que el transductor.

Las señales de vibración obtenidas directamente de la máquina se encuentran en el dominio del tiempo, ya que son una representación de su funcionamiento en tiempo real. Estas señales contienen una gran cantidad de datos sobre el comportamiento de cada parte de la máquina. Sin embargo, al analizarlas surge un problema: estas señales están cargadas de información de cada componente en forma muy compleja, lo que dificulta la interpretación de los comportamientos característicos de cada uno de estos. Una vez que se han recolectado las diferentes señales en los puntos de medición en la máquina, se almacenan en un sistema de análisis, donde se someten a estudio para extraer información relevante y diagnosticar el estado del equipo. El número de análisis posible es muy variado, a continuación se describen algunas de las técnicas de análisis más habituales (González et al., 2007):

Análisis de la onda temporal: El análisis de la onda temporal es un proceso en el que se registran y analizan las señales de vibración en función del tiempo, por lo que es necesario observar cómo cambia la señal del transductor a lo largo del tiempo de adquisición. Durante este análisis se examina la forma de la onda vibratoria para detectar alguna irregularidad, como picos, crestas, valles, etc., y, en función de estas anomalías, poder diagnosticar determinados problemas. Por ejemplo, el desequilibrio se refleja en forma de una onda aproximadamente senoidal, mientras que roces entre el eje de la máquina con alguna parte pueden generar achatamientos en la forma de la onda de desplazamiento. Es posible detectar desgaste y problemas en los rodamientos, ya que estos se manifiestan en forma de picos elevados en la señal, sobre todo en la aceleración de la vibración, acompañados de actividad a alta frecuencia. El análisis de la onda temporal también permite detectar vibraciones acopladas entre equipos cercanos con un régimen de giro similar, lo cual da lugar a un fenómeno conocido como batimiento, que ocasiona que la amplitud de la vibración llegue a niveles en los que pueda ocasionar daños en la máquina.

Análisis de espectro: El análisis de espectro de frecuencia es una herramienta clave en el diagnóstico de causas de vibración en la maquinaria. Para esto, es necesario analizar las

señales en el dominio de la frecuencia, y funciona descomponiendo las señales de vibración en sus componentes de frecuencia para revelar su contenido espectral. El espectro de frecuencias muestra la amplitud de la vibración en función de la frecuencia, revelando las frecuencias predominantes dentro de la vibración y sus amplitudes. Este proceso consiste en una transformación matemática, realizada mediante la Transformada de Fourier o alguna de sus variantes, comúnmente se utiliza la Transformada Rápida de Fourier (FFT, por sus siglas en inglés).

Antes de explicar lo que es la Transformada de Fourier, es necesario comprender lo que son las Series de Fourier. Las series de Fourier nos permiten descomponer una señal periódica como la suma de componentes senoidales y cosenoidales. Entonces, mientras la función sea continua y convergente para todo su periodo se podrá representar como la combinación de frecuencias. Las series de Fourier se representan por medio de la siguiente serie trigonométrica (Oppenheim et al., 1998) (Kreyszig, 2000):

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)] \quad (9)$$

Donde:

- ❖ $f(t)$ es una función periódica en el tiempo
- ❖ ω_0 es la frecuencia fundamental
- ❖ a_0 es el valor promedio de la señal
- ❖ a_n es el coeficiente de los componentes coseno
- ❖ b_n es el coeficiente de los componentes seno

Los coeficientes de las Series de Fourier, a_0 , a_n y b_n , se calculan mediante integrales, y dependen de la forma particular de la señal periódica.

$$a_0 = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) dt \quad (10)$$

$$a_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cos(n\omega_0) dt \quad (11)$$

$$b_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \sin(n\omega_0) dt \quad (12)$$

Donde T es el periodo de la señal y n es un entero positivo que representa el número de armónicos.

Las series de Fourier nos permite descomponer señales periódicas y la Transformada de Fourier nos permite aplicar estos conceptos a señales no periódicas. La transformada de Fourier convierte una señal en el dominio del tiempo en una representación en el dominio de la frecuencia, expresando la señal como una suma de componentes senoidales y cosenoidales con diferentes frecuencias y amplitudes. Estos componentes se expresan como exponenciales complejas de la forma $e^{j\omega t}$. La Transformada de Fourier se define matemáticamente como:

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt \quad (13)$$

En donde $X(j\omega)$ es conocida como transformada de Fourier o integral de Fourier de $x(t)$, $x(t)$ es la señal en el dominio del tiempo y j es la unidad imaginaria.

La transformada de Fourier se aplica solo a señales de tiempo continuo, lo cual significa que existe un valor para cada instante en el tiempo. Como se mencionó anteriormente, el mantenimiento predictivo monitorea los equipos mediante sensores a una frecuencia de muestreo establecida, por lo que los valores de la señal están definidos solo en momentos discretos y separados por intervalos regulares, a este tipo de señales se les conoce como señales de tiempo discreto. Una señal de tiempo discreto se puede representar como $x[n]$, donde n es un índice entero que denota el momento del tiempo discreto en el que se toma la muestra. Cada valor de la señal $x[n]$ corresponde a un tiempo discreto específico n . Para señales discretas, se utiliza la Transformada de Fourier Discreta (DFT, por sus siglas en inglés), que se define como:

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-\frac{j2\pi kn}{N}} \quad (14)$$

Donde $X(k)$ es la Transformada de Fourier Discreta de la señal $x[n]$ en el dominio de la frecuencia discreta k y N es la longitud (Proakis & Manolakis, 2007).

La principal desventaja de esta técnica es que la DFT puede ser considerablemente lenta, computacionalmente hablando, especialmente para señales largas. La Transformada Rápida de Fourier (FFT) es un algoritmo eficiente que reduce significativamente el tiempo de cálculo requerido para realizar la transformada discreta de Fourier. Debido a esto, la FFT es ampliamente utilizada para la implementación en tiempo real y el análisis de señales en sistemas digitales. Existen varias formas de implementar esta FFT, hay versiones como el algoritmo Radix-2 o el Radix-4, entre otros. Estos, básicamente, consisten en dividir la secuencia de datos en subconjuntos más pequeños y calcular la transformada de Fourier para estos subconjuntos, esto se le conoce como el principio dividir y conquistar (Proakis & Manolakis, 2007).

Una vez que la señal se ha transformado, se presenta un espectro de frecuencias, la ventaja del análisis de la frecuencia es que la mayor parte de los defectos tienen un periodo característico, por lo que tendrán una frecuencia predominante. Esto nos permite encontrar picos en el espectro que señalan problemas específicos en la máquina, como desequilibrios o fallos en rodamientos. La altura y la aparición de bandas alrededor de estos picos nos indican la gravedad de los problemas. La elección entre el uso del espectro de desplazamiento, velocidad o aceleración depende de la región de frecuencia que se quiere estudiar. El desplazamiento es más útil para detectar actividad a baja frecuencia, mientras que la aceleración es preferible para frecuencias elevadas.

Los pasos típicos en el análisis de vibración son:

- 1) Identificación de los picos de vibración en el espectro: El primer paso consiste en localizar el pico de primer orden (1X), que corresponde a la velocidad de rotación del eje de la máquina. En equipos con múltiples ejes, cada uno tendrá su propia frecuencia característica de 1X. Los picos de 1X suelen ir acompañados de armónicos o múltiplos de 1X. Por ejemplo, en una bomba de seis álabes, es común encontrar un pico destacado en 6X.

- 2) Diagnóstico de la máquina: Se evalúa la gravedad de los problemas de la máquina analizando las amplitudes y la relación entre los picos de vibración en el espectro.
- 3) Recomendaciones para reparaciones adecuadas: Se basan en la gravedad de los problemas identificados en el análisis de vibración, lo que guía las acciones necesarias para corregir las anomalías detectadas en la máquina.

Además de diagnosticar problemas, el análisis de espectro también es útil para evaluar la efectividad de las medidas de mantenimiento; comparando los espectros de vibración antes y después de realizar intervenciones de mantenimiento, se puede determinar si se han corregido los problemas identificados.

Análisis de fase: El análisis de fase es útil para entender cómo varía la sincronización entre la señal de vibración y el movimiento rotativo de la máquina. Ésta es una técnica que nos permite examinar la relación temporal entre los picos de una señal medida y los pulsos generados por un dispositivo de referencia, como un tacómetro o un estroboscopio, que opera a la misma velocidad que la rotación de la máquina. Entonces, se le denomina fase al desfase que existe entre los picos fundamentales de la señal tomada en el punto de la medición y los pulsos enviados por el tacómetro o dispositivo de referencia, esta fase se expresa en grados o radianes.

Al analizar la fase de la señal en función de la frecuencia, podemos identificar patrones en los desfases que pueden estar asociados a problemas específicos en la máquina. Una situación común en la que se utiliza el análisis de fase es para detectar problemas como el desalineamiento. Por ejemplo, si hay un desalineamiento entre dos ejes, es probable que las señales de vibración medidas en esos ejes estén fuera de fase. Esto se debe a que el movimiento vibratorio generado en un eje desalineado no estará sincronizado con el movimiento en el otro eje. Además del desalineamiento, el análisis de fase puede ayudar a detectar otros problemas como holguras, excentricidad y deformaciones (Power MI, 2018).

Análisis de Spike Energy: Este análisis se emplea para la detección de fallos en rodamientos. Cuando un rodamiento se desgasta, se generan golpes o impactos en los cuerpos rodantes. La duración de estos impactos es corta, por lo que se presentan con frecuencias altas y desplazamientos pequeños. El análisis de Spike Energy, también llamado espectro gSE, se realiza de la siguiente manera (Scheffer & Girdhar, 2004):

- 1) Mediante un filtro pasa altas se elimina el contenido de baja frecuencia de la señal.

- 2) Luego, la señal filtrada pasa a través de un rectificador RC, convirtiendo todas las partes negativas de la señal en positivas.

La señal filtrada pasa a través de un detector de pico a pico, que aplica una constante de tiempo de caída cuidadosamente seleccionada. La constante de tiempo de caída está directamente relacionada con la frecuencia máxima del espectro (F-max). La señal de salida de un detector pico a pico de Spike Energy es una señal en forma de diente de sierra.

- 3) Luego, esta señal se suaviza electrónicamente y se pasa a través del convertidor analógico a digital.
- 4) Una vez suavizada la señal, se detectan los pulsos significativos que indican la presencia de defectos en la maquinaria. Estos pulsos sobresalientes son aquellos que están por encima de cierto umbral de amplitud, si no hubiera defectos, no habrá pulsos sobresalientes, tal vez solo ruido de señal o instrumento de baja amplitud.
- 5) Para eliminar el contenido de alta frecuencia, la información digitalizada se envía a través de un filtro pasa bajo. La forma de onda resultante es la capa o envolvente de la modulación.
- 6) Finalmente, se realiza un análisis del espectro para identificar las frecuencias fundamentales y sus armónicos asociados a partir de la forma de onda resultante del proceso de Spike Energy. Esto proporciona información adicional sobre la naturaleza y la gravedad de los problemas de vibración en la maquinaria.

Análisis de Curtosis: El análisis de curtosis se centra en la forma de la distribución de datos y proporciona información sobre la concentración de los datos alrededor de la media, la presencia de picos extremos y el peso de las colas. Para el análisis de vibraciones, la curtosis se aplica a la señal en el dominio del tiempo y se utiliza para evaluar la forma de la señal registrada por los sensores. En el caso de una distribución normal, la curtosis tiene un valor de 3; los valores de curtosis mayores que 3 indican un aumento en los picos de la vibración, lo que podría ser indicativo de problemas en la maquinaria, como rodamientos defectuosos. Matemáticamente, la curtosis se define como el cuarto momento central de una distribución de datos, normalizado por la desviación estándar elevada a la cuarta potencia. La fórmula para calcular la curtosis de una muestra de datos es (Newbold et al., 2008):

$$Curtosis = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^4}{ns^4} \quad (15)$$

donde $\bar{x}$ y s son la media y la desviación típica muestrales, respectivamente.

Análisis de firmas eléctricas

El análisis de firmas eléctricas, o ESA (Electrical Signature Analysis), es una técnica de diagnóstico utilizada en el mantenimiento predictivo de equipos eléctricos y electromecánicos. Se basa en el hecho de que los pequeños cambios en el funcionamiento de una máquina afectan al campo magnético del motor conectado, y esto a su vez, afecta a la tensión de alimentación y a la corriente de funcionamiento. Consiste en monitorear las señales eléctricas generadas por el equipo durante su funcionamiento para identificar anomalías que puedan indicar problemas de funcionamiento o deterioro (Samotics, 2023).

En 1985, el Laboratorio Nacional de Oak Ridge (ORNL) comenzó un proyecto para evaluar la antigüedad de las válvulas motorizadas de centrales nucleares, con el objetivo de extraer información útil de la corriente de funcionamiento de un motor de manera no intrusiva y, sobre todo, a distancia. Este trabajo dio lugar al desarrollo de varias técnicas de análisis de firma de corriente del motor (MCSA) para detectar daños y fallos inminentes. Más tarde, el ORNL agregó monitoreo de la tensión y la potencia, dando lugar al término "análisis de firmas eléctricas" para englobar estas técnicas.

La principal ventaja del análisis de firmas eléctricas es que no requiere que los sensores estén físicamente cerca del equipo que se está monitoreando. A diferencia de los sistemas de monitoreo basados en la vibración o temperatura, donde se utilizan acelerómetros y cámaras de infrarrojos, respectivamente, colocados cerca de los equipos para medir vibraciones o capturar imágenes de calor, los sistemas de análisis de firma eléctrica utilizan sensores eléctricos, como transformadores de corriente y tomas de tensión, instalados en el gabinete de control del motor.

Para realizar el análisis de firma eléctrica, se utilizan diversos sensores que capturan diferentes aspectos de las señales eléctricas producidas por los equipos en funcionamiento:

- ❖ **Sensores de corriente:** Los sensores de corriente son dispositivos diseñados para medir la cantidad de corriente eléctrica que fluye a través de los conductores de un

equipo eléctrico. Existen diferentes tipos de sensores de corriente, siendo los más comunes los transformadores de corriente (CT) y los sensores de efecto Hall. Los transformadores de corriente funcionan según el principio de inducción electromagnética y constan de dos bobinas principales: una bobina primaria y una bobina secundaria. La bobina primaria se conecta en serie con el conductor por el que circula la corriente a medir, mientras que la bobina secundaria está diseñada para generar una corriente secundaria proporcional a la corriente primaria que pasa por la bobina primaria. La relación entre la corriente primaria y la secundaria depende del número de vueltas de alambre en ambas bobinas. Los sensores de efecto Hall, por otro lado, utilizan el efecto Hall para detectar la presencia de corriente eléctrica y generar una señal de salida proporcional a la corriente medida. Para esto, se coloca un imán o se genera un campo magnético cerca del sensor, este campo magnético debe ser perpendicular a la dirección de la corriente eléctrica que se desea medir, se hace pasar una corriente eléctrica a través del conductor donde está ubicado el sensor y cuando la corriente eléctrica atraviesa el conductor en presencia del campo magnético, se produce una acumulación de cargas en los bordes del conductor debido al efecto Hall. Este efecto genera una diferencia de potencial eléctrico en dirección perpendicular tanto al campo magnético como a la corriente eléctrica (Smith, 2024).

- ❖ **Sensores de voltaje:** El sensor de voltaje se conecta a los dos puntos del circuito eléctrico entre los cuales se desea medir la diferencia de potencial. Cuando se aplica una diferencia de potencial entre los dos puntos de conexión del sensor, se genera una corriente eléctrica a través de éste. Esta corriente es proporcional a la diferencia de potencial medida. La corriente generada por el sensor se convierte en una señal eléctrica proporcional a la diferencia de potencial medida. En algunos casos, la señal de voltaje medida puede ser amplificada y procesada electrónicamente para mejorar la precisión y la resolución de la medición.

La mayoría de las fallas mecánicas y eléctricas que pueden surgir en un motor de inducción son detectables mediante esta técnica de análisis de corriente denominada como MCSA, o a veces simplemente como CSA (Análisis de firma de corriente). El análisis de firma de corriente consta principalmente de tres pasos (Karmakar et al., 2016):

- 1) **Adquisición de datos:** Se captura la corriente del estator del motor utilizando un transductor de corriente, como un transformador de corriente o un sensor Hall. Normalmente, se monitorea una fase, pero en ocasiones se monitorean las tres fases para mayor precisión. Los datos capturados se digitalizan mediante el muestreo y, en algunos casos, se filtran para eliminar componentes no deseados de frecuencias.
- 2) **Extracción de características:** En este paso, se procesan los datos para extraer las características. Estos datos se procesan en el dominio del tiempo, en el dominio de la frecuencia o en ambos dominios tiempo-frecuencia. Para el procesamiento, se utilizan diferentes herramientas como la transformada rápida de Fourier (FFT), la transformada discreta de Fourier (DFT), la transformada de Fourier de tiempo corto (STFT), la Transformada Wavelet (WT), la transformada de Hilbert (HT), el enfoque de vector de Park, entre otras, así como técnicas de inteligencia artificial como redes neuronales y lógica difusa. Estas herramientas separan las frecuencias constituyentes y detectan componentes armónicos generados por fallas en el motor.
- 3) **Valoración de fallos:** En este paso se identifica cualquier fallo presente en el motor y se evalúa su gravedad. Normalmente, esto se hace mediante la comparación del espectro de frecuencia del motor estudiado con el de un motor sano. Se necesita un conocimiento detallado del motor específico que está siendo evaluado para llevar a cabo esta comparación de manera precisa.

La mayoría de los análisis utilizados para el diagnóstico de fallas se han realizado utilizando herramientas basadas en la transformada rápida de Fourier en la firma de corriente. Sin embargo, la Transformada rápida de Fourier tiene algunas limitaciones, como el enmascaramiento de frecuencias características o ser inadecuada para señales transitorias. Por lo que, para abordar estas limitantes, actualmente se emplean herramientas que ofrecen soluciones más efectivas (Karmakar et al., 2016).

Transformada de Fourier de tiempo corto: El STFT es una técnica de análisis de señales basada en Fourier que permite estudiar cómo varían las frecuencias de una señal en función del tiempo. El STFT divide la señal en segmentos temporales más pequeños y aplica la transformada de Fourier a cada segmento por separado, siendo capaz de analizar señales transitorias que varían con el tiempo. Si en una secuencia el índice de tiempo está fijo, entonces el STFT se convierte en una transformada de Fourier normal de la secuencia. El

STFT opera en ventanas de tiempo finitas y estas ventanas tienen una duración limitada y se desplazan a lo largo de la señal de entrada, lo que permite capturar cambios en la frecuencia de la señal a medida que avanza en el tiempo. Se pueden aplicar diferentes tipos de funciones de ventana para mejorar el análisis, como las funciones de ventana de Hamming y Hanning. Sin embargo, una limitación del STFT es el uso de una ventana de tamaño constante, lo que puede dificultar la coincidencia con el contenido de frecuencia de la señal. Para superar esto, se propone el uso de ventanas de tamaño variable, como en la Transformada Wavelet.

Transformada Wavelet: La Transformada Wavelet (WT) es una herramienta poderosa de procesamiento de señales que proporciona información detallada tanto en el dominio del tiempo como en el de la frecuencia al descomponer una señal en diferentes escalas y niveles de resolución utilizando una función prototipo llamada wavelet madre. Esta técnica supera las limitaciones de la Transformada de Fourier al adaptarse dinámicamente a las diferentes frecuencias de la señal. La ecuación gobernante de la WT es:

$$C(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \Psi\left(\frac{t-b}{a}\right) dt \quad (16)$$

En donde a es el parámetro de dilatación y no puede tener un valor de 0 y b es el parámetro de traslación, que también se denominan escala de la wavelet, que controla el estiramiento o compresión de la wavelet madre en el dominio del tiempo, y posición, respectivamente. $x(t)$ es la señal y $\Psi(t)$ es la función wavelet madre. Hay dos tipos principales de WT: la Transformada Wavelet Continua (CWT) y la Transformada Wavelet Discreta (DWT). La transformada wavelet continua se define de la siguiente manera:

$$CWT(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \Psi_{a,b}^*\left(\frac{t-b}{a}\right) dt \quad (17)$$

Aquí, $\Psi^*(t)$ denota el conjugado complejo de la función madre wavelet. La CWT opera sobre todas las posibles escalas y posiciones, es decir, aquí ocurre una dilatación y traslación continua.

Si la función madre wavelet se dilata y se traslada de forma discreta seleccionando $a = a_0^j$ y $b = kb_0a_0^j$ donde a y b son valores fijos de tal manera que $a > 1$ y $b > 0$ y, además, j y k son enteros positivos, entonces la función madre wavelet se vuelve discreta. Utilizando esta función madre wavelet discreta se le denomina DWT. En cada nivel de escalado y para diversas posiciones, la correlación entre la señal y la wavelet se llama coeficientes wavelet. La ecuación para la DWT es:

$$DWT(j, k) = \frac{1}{\sqrt{a_0^j}} \int_{-\infty}^{\infty} x(t) \Psi_{j,k}^* \left(\frac{t - kb_0a_0^j}{a_0^j} \right) dt \quad (18)$$

La DWT es más eficiente computacionalmente y se utiliza comúnmente para la compresión de datos, mientras que la CWT se utiliza para el análisis detallado de señales. La DWT descompone la señal sucesivamente mediante filtros pasa bajo y pasa alto, generando coeficientes de detalle (filtro pasa alto) y aproximación (filtro pasa bajo) que representan diferentes niveles de resolución de la señal. Estos coeficientes pueden utilizarse para reconstruir la señal original con precisión. Los investigadores han aplicado la WT para detectar fallas en motores mediante el análisis de transitorios como la corriente de arranque y la vibración, pero la selección arbitraria de la wavelet madre y la distorsión en los bordes pueden ser desafíos. Para superar estas limitaciones, se han propuesto enfoques alternativos como el análisis de envolvente de señales mediante la transformada de Hilbert.

Transformada de Hilbert: La transformada de Hilbert se define como:

$$H[s(t)] = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{s(\tau)}{(t - \tau)} d\tau \quad (19)$$

Usando el teorema del valor medio, se evalúa de la siguiente manera:

$$H[s(t)] = \frac{1}{\pi t} \otimes s(t) \quad (20)$$

El teorema del valor medio permite evaluar la transformada de Hilbert, $H[s(t)]$, de una señal $s(t)$ mediante la convolución de la función $\frac{1}{\pi t}$ con la señal original. La transformada de Hilbert desfasa teóricamente la señal original en $\frac{\pi}{2}$ y duplica las amplitudes de las frecuencias positivas, eliminando las frecuencias negativas. La transformada de Hilbert se utiliza para obtener la señal analítica a partir de una señal real para calcular la envolvente de una señal. La forma compleja de una señal se conoce como señal analítica. La representación compleja única de una señal real y se da por:

$$z(t) = s(t) + jH[s(t)] \quad (21)$$

La envolvente de una señal compleja, $z(t)$, se define como:

$$E(t) = |s(t) + jH[s(t)]| \quad (22)$$

La envolvente de una señal compleja proporciona una mejor detección que el análisis espectral original, ya que elimina la frecuencia de potencia y es útil para identificar características importantes de la señal, como los picos y los valles.

Volviendo al análisis de la firma eléctrica, al igual que en el MCSA, los datos se digitalizan y preprocesan, pero en el ESA se analizan conjuntamente la corriente y la tensión para obtener información sobre el estado eléctrico del motor. El análisis de la firma eléctrica puede identificar fallos mecánicos, como problemas de una bomba que cavita, un acoplamiento desalineado o daños en rodamientos. Estos problemas generan vibraciones que afectan el espacio entre el estator y el rotor del motor, perturbando el campo magnético. Cada perturbación produce cambios específicos en las firmas de frecuencia de corriente y tensión. El análisis de la firma eléctrica sobresale especialmente en la detección de fallos eléctricos. Los sistemas de ESA pueden identificar estos fallos antes que muchas otras técnicas de monitoreo de estado. Esto abarca desde barras de rotor rotas hasta corrientes de rodamiento, fallos en el estator y problemas de calidad de energía. La razón radica en que los cambios eléctricos afectan directamente al campo magnético del motor. Al medir la corriente y la

tensión, el análisis de la firma eléctrica detecta de manera temprana la señal inicial de que un daño está empezando a ocurrir (Samotics, 2023).

Análisis termográfico

El análisis termográfico es una técnica utilizada en el mantenimiento predictivo de equipos e instalaciones industriales. La termografía utiliza instrumentos especializados para medir la radiación infrarroja emitida de los objetos, o bien, su temperatura superficial. Al detectar áreas con temperaturas anómalas, ya sea más altas o bajas de lo esperado, se puede identificar una variedad de problemas en la planta antes de que se manifiesten (Mobley, 2002).

La tecnología infrarroja se basa en el principio de que todos los objetos con una temperatura superior al cero absoluto emiten radiación. La radiación infrarroja es una forma de energía emitida, y tiene longitudes de onda más cortas que la luz roja, siendo invisible sin instrumentación especial. La intensidad de esta radiación depende de la temperatura superficial del objeto. Sin embargo, medir la temperatura con métodos infrarrojos es complicado debido a que se pueden detectar tres fuentes de energía térmica de cualquier objeto: la energía emitida por el objeto, la energía reflejada por él y la energía transmitida a través de él. Para el mantenimiento predictivo, solo la energía emitida es relevante; las energías reflejadas y transmitidas pueden distorsionar los datos infrarrojos y deben filtrarse antes de realizar un análisis preciso. Las variaciones en la condición de la superficie, como la pintura o recubrimientos protectores, y otras variables pueden afectar el factor de emisividad real del equipo. La atmosfera entre el objeto y el instrumento de medición se deben de considerar también, ya que, por ejemplo, el vapor de agua y otros gases absorben la radiación infrarroja, y el polvo en suspensión, algunas luces y otras variables en la atmósfera circundante pueden distorsionar la radiación infrarroja medida. Tres tipos generales de instrumentos pueden ser utilizados para el mantenimiento predictivo (Mobley, 2002):

- ❖ **Termómetros Infrarrojos:** Están diseñados para proporcionar la temperatura superficial real en un solo punto en una máquina o superficie. Son útiles para monitorear puntos críticos en la maquinaria, pero tienen limitaciones debido a que solo proporcionan la temperatura en un punto. Se utiliza típicamente para monitorear temperaturas de tapas de rodamientos, temperaturas de devanados de motores,

verificaciones puntuales de temperaturas de tuberías de proceso y aplicaciones similares.

- ❖ **Escáneres de Línea:** Este tipo de instrumento infrarrojo proporciona un escaneo unidimensional o línea de radiación comparativa. Aunque este tipo de instrumento proporciona un campo de visión algo más, está limitado en aplicaciones de mantenimiento predictivo.
- ❖ **Imagen Infrarroja:** La imagen térmica o infrarroja proporciona los medios para escanear las emisiones infrarrojas de máquinas completas, procesos o equipos en muy poco tiempo. Funcionan como cámaras de video, permitiendo al usuario ver el perfil de emisión térmica de un área amplia. Hay una variedad de instrumentos de imágenes térmicas, que van desde escáneres en blanco y negro relativamente económicos hasta sistemas a color basados en microprocesadores.

La teoría básica de infrarrojos explica que la energía infrarroja es luz que está fuera del rango de percepción del ojo humano, pero que puede ser captada y medida mediante cámaras de infrarrojos. Estas cámaras convierten la energía infrarroja en imágenes digitales llamadas termogramas, donde cada píxel representa una temperatura. Este espectro está formado por ondas electromagnéticas cuyo rango está comprendido entre 0,75 μm (400 THz) y 1000 μm (300 GHz). A medida que aumenta la frecuencia de la onda, disminuye la longitud de onda y aumenta la energía contenida en la onda. La emisividad se refiere al porcentaje de energía que un objeto emite. En termografía infrarroja, se utiliza el término "cuerpo negro" para describir un emisor térmico perfecto con una emisividad del 100%. La magnitud de la radiación térmica está relacionada directamente con la temperatura del objeto y se puede calcular por medio de la ecuación (Olarte et al., 2011):

$$E = \varepsilon \sigma T_e^4 \quad (23)$$

Donde ε es la emisividad, σ es la constante de Stefan Boltzman, T es la temperatura del objeto medida en $^{\circ}\text{K}$. Los objetos que escaneará tendrán cada uno un valor de emisividad diferente. Sin embargo, en la mayoría de las inspecciones infrarrojas, que son termografías cuantitativas, el valor exacto de emisividad no es tan crítico. Las medidas cualitativas son imágenes que poseen diversos colores por medio de los cuales se pueden identificar los

puntos más calientes, mientras que las medidas cuantitativas son valores exactos de temperatura de los puntos calientes del equipo.

La inspección por infrarrojos busca identificar y registrar problemas en sistemas eléctricos o mecánicos de manera clara y comprensible. Muchos problemas suelen ocurrir en terminaciones y conexiones, especialmente en aquellas de cobre a aluminio. Es importante que las conexiones no se vean más calientes que los conductores si están dimensionadas correctamente. Cualquier conexión problemática debe ser revisada, limpiada, reensamblada o reemplazada según sea necesario.

2.2 INTELIGENCIA ARTIFICIAL: MACHINE LEARNING

Como se mencionó, el mantenimiento predictivo tradicional se basa en técnicas como el análisis de vibraciones y el análisis termográfico para detectar fallas en los equipos. Estos métodos, pese a ser efectivos, dependen de la experiencia de la persona que interpreta los datos, por lo que están sujetos a ciertas limitaciones. Una de estas limitaciones es la experiencia de la persona que realiza el análisis, ya que la interpretación puede estar sujeta a errores humanos, lo que podría resultar en diagnósticos equivocados. Por otra parte, el análisis requiere de mucho tiempo y esfuerzo, debido a la gran cantidad de equipos que hay dentro de una industria, por lo que existe la necesidad de reducir el número de equipos que se pueden monitorear. Además, los cambios más sutiles en la lectura de los parámetros pueden ser indicio de la presencia de alguna falla, pero estos cambios pueden ser imperceptibles para el humano, lo que limita la detección temprana de las fallas.

Los avances tecnológicos y la implementación de la inteligencia artificial han transformado el mantenimiento predictivo. Por un lado, la integración de IoT a los equipos industriales permite la recopilación de grandes cantidades de datos en tiempo real sobre la condición de los equipos, lo que facilita la generación de diferentes bases de datos. Mientras que la inteligencia artificial ofrece una gran capacidad de análisis de datos que complementan a las técnicas tradicionales de mantenimiento predictivo. Debido a esta gran capacidad de análisis, la inteligencia artificial permite una detección temprana de las fallas que apenas se están manifestando; además, los algoritmos basados en inteligencia artificial pueden aprender de

datos históricos, por lo que puede mejorar y obtener diagnósticos cada vez más precisos (Angelopoulos et al., 2019).

La inteligencia artificial se centra en desarrollar sistemas o programas que, mediante aprendizaje, puedan realizar tareas que, normalmente, requieren de la inteligencia humana. Los investigadores en inteligencia artificial están de acuerdo en que la inteligencia implica varias capacidades clave, como razonamiento, uso de estrategias, resolución de problemas bajo incertidumbre, representación del conocimiento, planificación, aprendizaje, comunicación en lenguaje natural e integración de habilidades hacia objetivos comunes. Estas capacidades son fundamentales en la definición y desarrollo de sistemas de inteligencia artificial (Teigens et al., 2020). Dentro del ámbito de la inteligencia artificial, la subdisciplina que se centra en desarrollar algoritmos y modelos que permiten a las máquinas aprender de los datos y realizar tareas específicas sin una programación explícita, es conocida como Machine Learning o Aprendizaje automático.

2.2.1 ¿QUÉ ES MACHINE LEARNING?

Machine Learning es una rama de la inteligencia artificial que utiliza algoritmos para descubrir relaciones en los datos. Estos algoritmos pueden aprender patrones y relaciones complejas a partir de grandes conjuntos de datos, lo que les permite realizar predicciones, tomar decisiones y resolver problemas de manera efectiva. Machine Learning permite a las computadoras aprender de la experiencia, esto implica que los sistemas inteligentes pueden adaptarse y mejorar con el tiempo mediante la retroalimentación, a diferencia de la programación explícita en la que se especifica paso a paso como se deben realizar las tareas (Norman, 2019). La inteligencia artificial implica programar computadoras para mejorar su rendimiento utilizando datos de ejemplo o experiencias pasadas. Este proceso implica definir un modelo con parámetros específicos y luego optimizar esos parámetros mediante el aprendizaje con datos de entrenamiento o experiencias previas. Los modelos pueden ser predictivos para hacer predicciones futuras, descriptivos para obtener información de los datos, o ambos. La inteligencia artificial se basa en la teoría estadística para hacer inferencias a partir de muestras de datos. La informática desempeña un papel crucial en proporcionar algoritmos eficientes para el entrenamiento y la inferencia, así como en gestionar grandes

cantidades de datos. El aprendizaje automático abarca varias funciones y aplicaciones que revolucionan la forma en que se procesan datos y se extrae conocimiento (Awad & Khanna, 2015) (Alpaydin, 2014):

- ❖ **Clasificación:** Los algoritmos de clasificación son eficaces para problemas con límites claros, entradas específicas y salidas categóricas. El proceso de clasificación implica crear un archivo de experiencias evaluando nuevas entradas con patrones previamente observados. Si se encuentra un patrón coincidente, la entrada se asocia con un comportamiento predictivo predefinido. De lo contrario, se examina más para determinar si es un patrón válido no descubierto o inusual. Los algoritmos de clasificación supervisada aprenden a través de ejemplos de toma de decisiones precisas, utilizando entradas preparadas.
- ❖ **Regresión:** Al igual que la clasificación, la regresión es un método de aprendizaje donde se intenta predecir una salida basada en ciertas entradas. En la regresión, la salida es un número, mientras que en la clasificación, la salida es una clase. Estos algoritmos de aprendizaje automático ajustan un modelo a los datos de entrenamiento para aprender la relación entre las entradas y la salida, optimizando los parámetros del modelo para minimizar el error de aproximación.
- ❖ **Agrupamiento:** El agrupamiento o clustering es un proceso de descubrimiento de conocimiento que organiza elementos en grupos basados en características similares. Implica un algoritmo iterativo que opera en la suposición de similitud y se detiene cuando se satisface un criterio de terminación. El objetivo es encontrar una función que mida la similitud entre elementos y los parámetros del clustering, dependen de las aplicaciones y del conjunto de datos.
- ❖ **Reducción de dimensionalidad:** La reducción de dimensionalidad implica disminuir la cantidad de variables aleatorias mediante la selección y extracción de características. Esto conlleva tiempos de entrenamiento más cortos, mejor generalización y prevención del sobreajuste. La selección de características consiste en elegir un subconjunto de variables relevantes para la construcción del modelo, mientras que la extracción de características implica transformar el espacio de alta dimensionalidad en uno de dimensiones más reducidas mediante la combinación de atributos.

- ❖ **Filtrado Colaborativo:** Filtrado Colaborativo es un proceso de filtrado de información que utiliza métodos colaborativos entre múltiples fuentes de datos para hacer recomendaciones basadas en las preferencias de usuarios similares. Estos algoritmos deben hacer recomendaciones efectivas en poco tiempo, a pesar de datos dispersos, problemas de privacidad, entre otros inconvenientes.

Los algoritmos, basándose en identificar y comprender las relaciones entre los datos de entrada y las salidas esperadas durante la fase de entrenamiento del modelo, se pueden clasificar en las siguientes categorías (Awad & Khanna, 2015):

- ❖ **Aprendizaje supervisado:** El aprendizaje supervisado es un método donde se establece una relación entre los datos observados (también llamados datos de entrada) y una variable objetivo que se quiere predecir (variable dependiente o etiqueta). Se utiliza un conjunto de datos etiquetados para entrenar un modelo que pueda generalizar esta relación y hacer predicciones precisas sobre nuevos datos. El objetivo de los algoritmos de aprendizaje es minimizar el error para un conjunto dado de entradas. Un modelo bien entrenado mediante aprendizaje supervisado puede hacer predicciones precisas sobre etiquetas de clase incluso para datos nuevos u ocultos. Sin embargo, si el conjunto de datos de entrenamiento es de baja calidad, el modelo puede sobre ajustarse, lo que lleva a predicciones incorrectas.
- ❖ **Aprendizaje no supervisado:** Los algoritmos de aprendizaje no supervisado buscan descubrir patrones en conjuntos de datos sin etiquetar, donde no se conoce la salida deseada. Se entrenan mediante modelos probabilísticos y su objetivo es crear representaciones de los datos para la toma de decisiones, pronósticos y agrupamiento de información eficientes. Estos algoritmos son útiles en diversas áreas como la compresión de datos, detección de anomalías, clasificación, entre otros. Ejemplos comunes son el agrupamiento y la reducción de dimensionalidad.
- ❖ **Aprendizaje semisupervisado:** El aprendizaje semisupervisado combina un pequeño número de conjuntos de datos etiquetados con una gran cantidad de conjuntos de datos no etiquetados para generar un modelo o clasificador. Esto es útil porque etiquetar datos es costoso y laborioso, mientras que los datos no etiquetados son más económicos y fáciles de obtener. El aprendizaje semisupervisado se sitúa entre el aprendizaje no supervisado y supervisado y puede mejorar significativamente

la precisión del aprendizaje. Recientemente ha ganado prominencia debido a la disponibilidad de grandes cantidades de datos no etiquetados en diversas áreas como web, mensajería, acciones, biología, imágenes, etc.

- ❖ **Aprendizaje por refuerzo:** El aprendizaje por refuerzo (Reinforcement Learning, RL) implica que un agente inteligente (RL-agent) explore y ajuste sus acciones en un entorno para maximizar la recompensa acumulativa. El agente aprende mediante un proceso de prueba y error, ajustando su política basado en experiencias pasadas y recompensas recibidas. Se busca un equilibrio entre explorar nuevas acciones y evitar el fracaso, ya que se necesitan múltiples intentos para determinar la eficacia de las acciones no probadas. Los componentes críticos del aprendizaje por refuerzo son: la política del agente, que mapea las acciones de control al estado percibido del entorno; el crítico, que evalúa la efectividad de las acciones; la función de recompensa que guía el comportamiento; y los modelos, que ayudan a predecir futuras acciones.
- ❖ **Aprendizaje transductivo:** El aprendizaje transductivo intenta predecir funciones de modelo exclusivas en casos de prueba específicos mediante el uso de observaciones adicionales en el conjunto de datos de entrenamiento. Se establece un modelo local al ajustar nuevas observaciones individuales (datos de entrenamiento) en un único punto en el espacio, a diferencia del modelo global que abarca todo el espacio de problemas. Es útil cuando se quiere adaptar el modelo para hacer predicciones precisas en un conjunto específico de prueba.
- ❖ **Inferencia inductiva:** La inferencia inductiva estima la función del modelo considerando todo el espacio de hipótesis y luego utiliza este modelo para predecir valores de salida para datos fuera del conjunto de entrenamiento. Básicamente, se refiere a la capacidad de un algoritmo para hacer predicciones precisas sobre datos no vistos durante el entrenamiento. Algunos algoritmos populares que se utilizan en este enfoque son SVM, modelos de redes neuronales y algoritmos neurodifusos. Además, en ciertos casos, se puede aplicar un modelo de aprendizaje perezoso, que continuamente desarrolla un espacio de hipótesis más rico basado en nuevos datos aplicados al modelo existente.

2.2.2 PROCESO DE DESARROLLO DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

El proceso de desarrollo de algoritmos de aprendizaje automático implica los siguientes pasos (Awad & Khanna, 2015):

- 1) **Recopilación de datos:** Seleccionar los atributos de datos relevantes para resolver el problema, evitando la redundancia. Los datos pueden ser obtenidos de diversas fuentes, como APIs de transmisión de datos, y pueden requerir preprocesamiento estadístico.
- 2) **Preprocesamiento de datos:** Los datos se deben presentar en un formato utilizable, limpiando los datos corruptos o faltantes, y normalizándolos o discretizándolos según sea necesario. El muestreo puede ser utilizado para reducir la redundancia sin perder información.
- 3) **Transformación de datos:** Adaptar los datos al algoritmo y al problema en cuestión, escalando características, descomponiendo o agregando atributos para extraer información útil.
- 4) **Entrenamiento del algoritmo:** Seleccionar conjuntos de datos de entrenamiento y prueba, y alimentar el algoritmo con los datos transformados para extraer conocimiento. Este conocimiento se almacena como un modelo para su validación cruzada y uso futuro.
- 5) **Prueba del algoritmo:** Evaluar la eficacia y rendimiento del algoritmo frente a datos de prueba. Si es necesario, ajustar el modelo mediante cambios en los datos, transformaciones, entre otros.
- 6) **Aplicación del aprendizaje por refuerzo:** En casos donde el aprendizaje supervisado no es práctico, emplear el aprendizaje por refuerzo para aprender de las interacciones con el entorno y mejorar el rendimiento del modelo.
- 7) **Ejecución:** Aplicar el modelo validado para realizar predicciones en situaciones reales. Si se encuentran nuevos datos, volver a entrenar el modelo según sea necesario.

Uno de los pasos más importantes es el de evaluar el rendimiento del algoritmo, el desafío central en el aprendizaje automático es generalizar bien para nuevas entradas no vistas

anteriormente, lo que se llama generalización. Por lo general, al entrenar un modelo de machine learning, disponemos de un conjunto de datos de entrenamiento, en el cual podemos evaluar alguna métrica de error, conocida como error de entrenamiento, y trabajamos para minimizar este error. Los factores que determinan cuán bien funcionará un algoritmo de aprendizaje automático son su capacidad para (Goodfellow et al., 2016):

- ❖ Hacer que el error de entrenamiento sea pequeño.
- ❖ Hacer que la brecha entre el error de entrenamiento y el error de prueba sea pequeña.

Estos dos aspectos se relacionan con los principales desafíos en el aprendizaje automático: subajuste (underfitting) y sobreajuste (overfitting). El subajuste sucede cuando el modelo no logra alcanzar un nivel suficientemente bajo de error en el conjunto de entrenamiento. Por otro lado, el sobreajuste ocurre cuando existe una diferencia considerable entre el error registrado durante el entrenamiento y el error de prueba observado en el modelo.

Una de las formas de controlar si un modelo es más propenso a sobreajustar o a subajustar es alterando su capacidad. Esta capacidad se refiere a la habilidad del modelo para adaptarse a una amplia gama de funciones. Modelos con baja capacidad pueden no ajustarse bien al conjunto de entrenamiento, mientras que aquellos con alta capacidad pueden sobreajustarse al memorizar detalles irrelevantes del conjunto de entrenamiento. Una manera de ajustar la capacidad de un modelo es variando el número de características de entrada y al mismo tiempo introduciendo nuevos parámetros que estén relacionados con esas características adicionales. Aumentar la cantidad de características de entrada y la cantidad de parámetros asociados con esas características permite al modelo capturar relaciones más complejas entre los datos de entrada y las salidas deseadas. Por otro lado, reducir la cantidad de características o parámetros puede simplificar el modelo y hacerlo más generalizable, pero también puede limitar su capacidad para capturar la complejidad de los datos subyacentes. Además, la capacidad de un modelo está determinada por la familia de funciones que puede seleccionar mientras ajusta los parámetros para minimizar el error de entrenamiento. En la práctica, encontrar la mejor función dentro de esta familia puede ser un desafío de optimización complejo, y los algoritmos de aprendizaje a menudo no logran encontrar la función óptima, sino una que reduzca el error de entrenamiento de manera significativa. Estas limitaciones, junto con la imperfección de los algoritmos de optimización, pueden hacer que la capacidad

efectiva del modelo sea menor que su capacidad representacional teórica (Goodfellow et al., 2016).

Otra forma de controlar el sobreajuste consiste en recopilar más ejemplos de entrenamiento, esto puede ayudar a reducir la brecha entre el error de entrenamiento y el error de generalización, mejorando así el rendimiento del modelo en nuevas entradas. El ensamblaje también es otra forma de reducir la posibilidad de sobreajuste. Esta técnica básicamente combina predicciones de diferentes modelos de Machine Learning, al combinar múltiples modelos en un ensamblaje. Los diferentes modelos pueden capturar diferentes aspectos de la relación entre las características y la variable objetivo, y al promediar o combinar sus predicciones, se puede reducir la varianza del modelo (Marina, 2022).

Por otra parte, la selección adecuada de hiperparámetros es crucial para el rendimiento y la generalización del modelo. Un hiperparámetro incorrectamente ajustado puede llevar a un sobreajuste o a un subajuste. Los hiperparámetros son configuraciones que determinan el comportamiento y la capacidad de los algoritmos de aprendizaje. A diferencia de los parámetros del modelo, que se aprenden durante el entrenamiento, los hiperparámetros deben establecerse antes del proceso de aprendizaje y no se ajustan durante el entrenamiento del modelo en sí. Ejemplos comunes de hiperparámetros incluyen la tasa de aprendizaje, la profundidad máxima de un árbol de decisión o el número de vecinos en el algoritmo de vecinos más cercanos. Para elegir los mejores valores de los hiperparámetros, se utiliza un proceso de validación cruzada. Esto implica dividir el conjunto de datos en un conjunto de entrenamiento y un conjunto de validación. Típicamente, se usa alrededor del 80% de los datos de entrenamiento para entrenamiento y el 20% para validación. El conjunto de entrenamiento se utiliza para ajustar los parámetros del modelo, mientras que el conjunto de validación se utiliza para evaluar el rendimiento del modelo con diferentes configuraciones de hiperparámetros. Se repite este proceso varias veces con diferentes divisiones de datos para encontrar la combinación óptima de hiperparámetros que maximiza el rendimiento del modelo en datos no vistos (Goodfellow et al., 2016).

Dividir un conjunto de datos en un conjunto de entrenamiento y otro de prueba fijo puede ser problemático si el conjunto de prueba resulta ser pequeño. Cuando los conjuntos de datos son pequeños, se utilizan procedimientos alternativos como la validación cruzada k-fold. En la validación cruzada k-fold, el conjunto de datos se divide en K partes iguales. En cada

iteración, una parte se utiliza como conjunto de validación y las demás como conjunto de entrenamiento. Esto se repite K veces, con una parte diferente excluida en cada iteración. El valor de K se elige según el tamaño del conjunto de datos, aunque K suele ser 10 ó 30. Cuando K aumenta, el porcentaje de instancias de entrenamiento aumenta y obtenemos estimadores más robustos, pero el conjunto de validación se vuelve más pequeño; además, está el costo de entrenar el clasificador K veces, lo cual también se incrementa. Un caso extremo de k -fold es el "dejar uno fuera", donde de las N instancias, solo se deja una instancia fuera como conjunto de validación y el entrenamiento utiliza las $N - 1$ instancias (Alpaydin, 2014).

2.2.3 MÉTRICAS DE EVALUACIÓN DE MODELOS DE APRENDIZAJE AUTOMÁTICO

La evaluación de modelos implica seleccionar el modelo más adecuado considerando su rendimiento predictivo. Se utilizan diversas medidas de rendimiento y se enfoca en las propiedades de generalización del modelo. Se pueden utilizar diversas medidas de rendimiento dependiendo de la aplicación prevista del modelo considerado. La matriz de confusión es una herramienta utilizada para evaluar el rendimiento de los modelos clasificadores, ofreciendo información crucial sobre su capacidad para predecir clases específicas y sus habilidades de generalización. Sin embargo, es importante tener en cuenta que la matriz de confusión puede no ser suficiente para comparar y clasificar eficazmente varios modelos en términos de sus características o desempeño, dificultando la elección del mejor entre múltiples modelos candidatos. La matriz de confusión, también conocida como matriz de error, es una representación que muestra cómo se desempeña un algoritmo de clasificación utilizando los datos disponibles. Compara las clasificaciones predichas con las clasificaciones reales, cada predicción puede ser uno de cuatro resultados, basado en cómo coincide con el valor real (Subasi, 2020) (Awad & Khanna, 2015):

- ❖ **Verdadero Positivo (TP):** La salida es identificada correctamente como positivo.
- ❖ **Verdadero Negativo (TN):** La salida es identificada correctamente como negativo.
- ❖ **Falso Positivo (FP):** La salida es identificada incorrectamente como positivo.
- ❖ **Falso Negativo (FN):** La salida es identificada incorrectamente como negativo.

El ideal es que el modelo tenga cero falsos positivos y cero falsos negativos. La matriz de confusión en sí misma no es una métrica de desempeño, pero la mayoría de las métricas de desempeño se derivan de la matriz de confusión y los valores que contiene (Subasi, 2020). Una de estas es la exactitud (Accuracy), que es la proporción de clasificaciones correctas respecto a todas las instancias que se utilizan, puede representarse mediante la fórmula:

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \quad (24)$$

La precisión (precision) del modelo representa la proporción de instancias correctamente clasificadas como positivas en comparación con todas las instancias clasificadas como positivas:

$$Precision = \frac{TP}{TP + FP} \quad (25)$$

El recall, también conocido como tasa de verdaderos positivos (TPR), se calcula utilizando la proporción de instancias clasificadas correctamente como positivas respecto a todas las instancias positivas:

$$Recall = \frac{TP}{TP + FN} \quad (26)$$

Una forma común de evaluar la precisión de una prueba es calcular el F-score, que es la media armónica de la precisión y el recall. El F-score es útil para evaluar la precisión de las pruebas, sin embargo, no tiene en cuenta los verdaderos negativos.

$$F1 = 2 \frac{(recall * precision)}{(recall + precision)} \quad (27)$$

Ya que ha quedado claro que son, como se clasifican y como se evalúan; a continuación se describen algunos de los algoritmos utilizados en Machine Learning:

2.2.4 ÁRBOLES DE DECISIÓN

Árboles de decisión

Los árboles de decisión son una herramienta poderosa en el ámbito del aprendizaje supervisado, utilizada tanto para problemas de clasificación como de regresión. Se caracterizan por su estructura jerárquica, donde cada nodo de decisión divide el espacio de entrada en regiones locales, de manera recursiva. Está compuesto por nodos de decisión internos y hojas terminales. Cada nodo de decisión realiza una prueba con resultados discretos para etiquetar las ramas. El proceso comienza en el nodo raíz y se repite hasta llegar a un nodo hoja, donde el valor determinado constituye la salida (Subasi, 2020) (Alpaydin, 2014). El árbol tiene tres tipos de nodos (Tan et al., 2014):

- ❖ **Nodo raíz:** Este tipo de nodos no tiene aristas entrantes y cero o más aristas salientes. Representa a toda la población o muestra y en él se produce la primera división en función de la variable más importante.
- ❖ **Nodos internos:** Después de la primera división se encuentran estos nodos, que vuelven a dividir el conjunto de datos. Cada uno de estos nodos tiene exactamente una arista entrante y dos o más aristas salientes.
- ❖ **Nodos hoja o terminales:** Son los últimos nodos en un árbol de decisión y cada uno de los cuales tiene exactamente una arista entrante y ninguna arista saliente. Representan la clasificación final.

Los árboles de decisión no son modelos paramétricos, lo que significa que no asume una forma específica para las densidades de clase, sino que su estructura se adapta durante el aprendizaje según la complejidad de los datos. Una de las ventajas clave de los árboles de decisión es su interpretabilidad. Dado que el árbol se puede representar como un conjunto de reglas SI-ENTONCES, es fácil para los usuarios comprender y explicar cómo se toman las decisiones del modelo (Alpaydin, 2014).

En el caso de un árbol de decisión para clasificación la calidad de una división se cuantifica mediante una medida de impureza. La construcción de un árbol de decisión implica encontrar la mejor división en cada etapa, lo que implica evaluar diferentes medidas de impureza o ganancia de información para determinar qué atributo y valor de corte maximiza la pureza de las hojas resultantes. Un nodo se considera puro cuando todas las muestras que contiene

pertenecen a la misma clase. Una posible función para medir la impureza es la entropía y la fórmula de la entropía se da como (Tan et al., 2014):

$$Entropía = - \sum_{i=1}^c p_i \log_2(p_i) \quad (28)$$

En donde, C es el número de clases y p_i es la proporción de muestras del nodo que pertenecen a la clase. La entropía alcanza su valor máximo, 1, cuando la distribución entre clases es uniforme; y su valor mínimo cuando el nodo es puro. Pero la entropía no es la única medición posible, el índice Gini es otra manera de medir la impureza:

$$Gini = 1 - \sum_{i=1}^c (p_i)^2 \quad (29)$$

Al igual que la entropía, el índice gini alcanza su valor máximo, 0.5, cuando la distribución entre clases es uniforme; y su valor mínimo, 0, cuando el nodo es puro. Otra forma de medir la impureza del nodo es mediante el error de clasificación incorrecta:

$$Error = 1 - \max(p_i) \quad (30)$$

Las medidas de impureza como la entropía y el índice de Gini prefieren atributos con muchos valores distintos. Una condición de prueba que resulta en un gran número de resultados puede no ser deseable porque el número de registros asociados con cada partición es demasiado pequeño para permitirnos hacer predicciones confiables. Hay dos estrategias para superar este problema: la primera es restringir las condiciones de prueba solo a divisiones binarias; la cual es utilizada en árboles de decisión como los CART; y la segunda, es ajustar el criterio de división para considerar el número de resultados de cada atributo, como el algoritmo C4.5, que utiliza el ratio de ganancia como criterio de división:

$$Ratio\ de\ Ganancia = \frac{\Delta_{info}}{Split\ Info} \quad (31)$$

Δ_{info} se refiere a la Ganancia de Información, y se calcula como la diferencia entre la entropía antes de dividir los datos y la entropía después de la división; mientras que *Split Info* o Información Intrínseca, la información intrínseca es como la entropía, pero se enfoca en un solo atributo, por lo que si un atributo tiene muchos valores diferentes, su información intrínseca será alta.

Existen distintos algoritmos de aprendizaje automático utilizados para construir árboles de decisión, entre los más populares se encuentran (IBM, s.f.):

- ❖ **ID3:** Desarrollado por Ross Quinlan, utiliza la entropía y la ganancia de información para evaluar las divisiones de los datos y construir árboles de decisión. Es conocido por su simplicidad y facilidad de comprensión.
- ❖ **C4.5:** También creado por Ross Quinlan, es una mejora del ID3. Puede utilizar la ganancia de información o las proporciones de ganancia para evaluar las divisiones. Es más versátil que el ID3 y puede manejar valores faltantes y atributos con múltiples valores.
- ❖ **CART:** Introducido por Leo Breiman, se centra en árboles de clasificación y regresión. Utiliza la impureza de Gini para identificar el mejor atributo para dividir los datos. CART es versátil y puede utilizarse para problemas de clasificación y regresión.

El CART divide el conjunto de entrenamiento en dos subconjuntos utilizando una característica k y un umbral t_k , eligiendo el par (k, t_k) que produce subconjuntos más puros. Utiliza una función de costo para minimizar la impureza en cada división (Géron, 2019):

$$J(k, t_k) = \frac{m_{izq}}{m} G_{izq} + \frac{m_{der}}{m} G_{der} \quad (32)$$

Donde G_{der} y G_{izq} mide la impureza de los subconjuntos derecho e izquierdo, y m_{der} y m_{izq} son el numero de instancias en los subconjuntos derecho e izquierdo. Después de la primera

división, CART continúa subdividiendo recursivamente los subconjuntos hasta alcanzar la profundidad máxima especificada o hasta que no se pueda reducir más la impureza.

Los árboles de decisión tienen la tendencia de sobreajustar, para evitar el sobreajuste de los datos de entrenamiento, es necesario restringir la libertad del árbol de decisión durante el entrenamiento. Existen algunos hiperparámetros que nos permiten realizar esta regularización: la profundidad máxima del árbol de decisión, el número mínimo de muestras que un nodo debe tener antes de poder dividirse, el número mínimo de muestras que un nodo hoja debe tener, el número máximo de nodos hoja, el número máximo de características que se evalúan para dividir en cada nodo, entre otros (Géron, 2019). Otros algoritmos funcionan primero entrenando el Árbol de Decisión sin restricciones y luego podando (eliminando) nodos innecesarios. En la pre-poda del árbol de decisión, se detiene la construcción del árbol si el número de instancias de entrenamiento en un nodo es menor que un cierto porcentaje del conjunto de entrenamiento, lo que ayuda a evitar el sobreajuste. En cambio, en la post-poda del árbol, se construye el árbol completo y luego se identifican y podan los subárboles que causan sobreajuste. Se utiliza un conjunto de poda separado del conjunto de entrenamiento original para evaluar la efectividad de la poda. Aunque la pre-poda es más rápida, la post-poda generalmente produce árboles más precisos (Alpaydin, 2014). Por último, y como se mencionó con anterioridad, otra forma de reducir esta tendencia al sobreajuste, además de mejorar la capacidad de generalización, es mediante el ensamblaje.

2.2.5 MÉTODOS DE ENSAMBLE

Los modelos combinados, también conocidos como conjuntos de modelos, son enfoques altamente efectivos en el aprendizaje automático, superando generalmente el rendimiento de los modelos individuales. Se basan en el concepto de sabiduría de la multitud, que se ilustra al considerar que la agregación de respuestas de muchas personas al azar a menudo supera a la de un experto. Sin embargo, estos modelos de conjunto suelen conllevar un mayor costo algorítmico y complejidad. Una teoría estadística importante sugiere que promediar mediciones puede generar estimaciones más estables y confiables al reducir el efecto de las fluctuaciones aleatorias. Entonces, al formar un modelo de conjunto usando los mismos datos de entrenamiento, es posible reducir el efecto de las variaciones aleatorias en los modelos

individuales. Para crear diversidad entre los modelos del conjunto, se suelen entrenar modelos en subconjuntos aleatorios de los datos o con características aleatorias. Los modelos de conjunto suelen construir múltiples modelos predictivos a partir de adaptaciones del conjunto de entrenamiento mediante remuestreo o ponderación, y luego combinan las predicciones de estos modelos, generalmente mediante promedios o votación. Este enfoque tiende a mejorar la precisión general del modelo al aprovechar las fortalezas de diversos modelos individuales. Algunos de los métodos de conjunto más populares son el bagging, boosting y stacking, así como los Bosques Aleatorios (Random Forest).

Bagging: El bagging (Bootstrap Aggregating) es una técnica de aprendizaje automático que se utiliza para mejorar el rendimiento de los modelos predictivos mediante la creación de un conjunto de modelos. Funciona entrenando múltiples modelos en subconjuntos aleatorios de datos de entrenamiento y luego combinando sus predicciones para producir una predicción final. Esto se logra mediante bootstrap, que genera L muestras diferentes partiendo de un conjunto de entrenamiento X con N elementos, extraemos N instancias de forma aleatoria y con reemplazo, lo que nos permite construir muestras diversas para su posterior análisis. Como resultado, algunas instancias pueden ser seleccionadas más de una vez y otras pueden no ser seleccionadas en absoluto. Cuando el muestreo se realiza sin reemplazo, se le denomina pasting. Entre todas las L muestras hay similitudes ya que se derivan de la misma muestra original, pero también, debido al azar, presentan pequeñas variaciones. Un algoritmo de aprendizaje se considera inestable si pequeños cambios en los datos de entrenamiento provocan grandes diferencias en el modelo final, lo que sugiere una alta variabilidad en el algoritmo. El Bagging se aplica tanto en clasificación como en regresión. En la clasificación se toma el promedio, mientras que en la regresión, se puede usar la mediana en lugar del promedio para mejorar la robustez al combinar las predicciones. Si bien el bagging es una técnica poderosa para mejorar el rendimiento de los modelos, no es adecuado para todos los tipos de algoritmos de aprendizaje automático. Por ejemplo, puede no funcionar bien con clasificadores estables como los vecinos más cercanos, mientras que funciona perfectamente para algoritmos de aprendizaje inestables como redes neuronales y árboles de decisión. El bagging es particularmente efectivo cuando se combina con modelos de árboles de decisión, ya que estos modelos tienden a ser sensibles a las variaciones en los datos de entrenamiento. Cuando se utiliza con árboles de decisión, el bagging puede mejorar

aún más el rendimiento al agregar aún más diversidad al conjunto de modelos. Esta combinación se conoce como random forest, donde cada árbol se construye a partir de un subconjunto aleatorio de características (Alpaydin, 2014)(Subasi, 2020).

Random Forest: El bosque aleatorio es un método de aprendizaje en conjunto para la clasificación que utiliza múltiples árboles de decisión decorrelacionados. En lugar de depender de un solo árbol profundo, combina la salida de varios árboles poco profundos para reducir el sesgo y la varianza. El proceso implica la selección aleatoria de muestras del conjunto de datos original y la construcción de múltiples árboles que seleccionan aleatoriamente un subconjunto de características en cada división. Para clasificar nuevos datos, se promedia la predicción de todos los árboles para obtener una clasificación final. Este enfoque reduce la varianza de la estimación final y ofrece buena precisión en conjuntos de datos grandes, incluso cuando hay datos faltantes. Además, el bosque aleatorio permite estimar la importancia relativa de cada característica en el modelo, lo que facilita la selección de características (Awad & Khanna, 2015).

Boosting: El Boosting es un enfoque dentro del aprendizaje de conjunto que se asemeja al bagging, pero adopta un método más elaborado para generar conjuntos de entrenamiento diversos. Su objetivo es mejorar el rendimiento de los aprendices débiles mediante la optimización iterativa en los datos de entrenamiento. A diferencia del bagging, donde los aprendices base se generan aleatoriamente, en el boosting se intenta generar aprendices base complementarios entrenando al siguiente aprendiz sobre los errores de los aprendices anteriores. Dos de los métodos de refuerzo más populares son AdaBoost y Gradient Boosting.

AdaBoost utiliza el mismo conjunto de entrenamiento repetidamente, evitando así la necesidad de un conjunto de entrenamiento muy grande. Además, puede combinar un número variable de aprendices base, lo que lo hace más flexible. El funcionamiento de AdaBoost implica modificar las probabilidades de selección de instancias para el entrenamiento de cada nuevo aprendiz en función de los errores cometidos por los aprendices anteriores. Este enfoque se centra en las instancias difíciles de clasificar, lo que ayuda a mejorar el rendimiento del modelo final. Estos son los pasos a seguir para entrenar un clasificador AdaBoost son:

- 1) Comienza asignando pesos iguales a todas las instancias de entrenamiento.

- 2) Entrena un primer predictor y calcula su tasa de error ponderada.
- 3) Calcula el peso del predictor en función de su precisión.
- 4) Actualiza los pesos de las instancias, dando más peso a las instancias mal clasificadas.
- 5) Normaliza los pesos de las instancias.
- 6) Entrena un nuevo predictor utilizando los pesos actualizados y repite el proceso.
- 7) Se detiene cuando se alcanza el número deseado de predictores o cuando se encuentra un predictor perfecto.
- 8) Para hacer predicciones, AdaBoost pondera las predicciones de todos los predictores utilizando sus pesos y elige la clase con la mayoría de votos ponderados.

Por otro lado, Gradient Boosting es otra técnica popular dentro del refuerzo. A diferencia de AdaBoost, que ajusta los pesos de las instancias en cada iteración, Gradient Boosting intenta ajustar el nuevo predictor a los errores residuales cometidos por el predictor anterior. Este método agrega secuencialmente predictores a un conjunto, cada uno corrigiendo a su predecesor.

- 1) Se entrena un modelo base, como un árbol de decisión, con los datos de entrenamiento.
- 2) Se calcula la diferencia entre las predicciones del modelo inicial y los valores reales, lo que da como resultado el error residual.
- 3) Se entrena un nuevo modelo para predecir estos errores residuales en lugar de los valores reales.
- 4) El nuevo modelo se suma al modelo anterior, corrigiendo los errores cometidos por el primer modelo.
- 5) Este proceso se repite varias veces, con cada nuevo modelo tratando de corregir los errores residuales de los modelos anteriores.
- 6) Finalmente, se combinan todos los modelos para hacer una predicción final, generalmente ponderando las predicciones de cada modelo en función de su rendimiento.

Este método es especialmente útil para problemas de regresión, donde se pueden ajustar los residuos de las predicciones anteriores para mejorar gradualmente la precisión del modelo (Alpaydin, 2014) (Subasi, 2020).

Stacking: Stacking es un método de ensamblaje que consiste en entrenar un modelo para combinar las predicciones de otros modelos en un conjunto. En lugar de utilizar métodos simples como el voto mayoritario, se entrena un modelo adicional llamado "mezclador" o "meta-aprendiz" para realizar esta combinación. El proceso básico de stacking implica los siguientes pasos:

1. Se entrenan varios modelos base utilizando diferentes algoritmos de aprendizaje y/o configuraciones de parámetros.
2. Cada modelo base hace predicciones sobre el conjunto de datos de entrenamiento.
3. Las predicciones hechas por los modelos base se utilizan como características de entrada, y el valor real objetivo se conserva.
4. Se entrena un modelo adicional (el mezclador) utilizando el conjunto de datos creado en el paso anterior. Este modelo aprende a combinar las predicciones de los modelos base para obtener una predicción final.
5. Una vez entrenado el mezclador, se utiliza para hacer predicciones sobre nuevos datos, utilizando las predicciones de los modelos base como entradas.

Para implementar el stacking, se busca que los modelos base sean lo más diferentes posible entre sí y que estén basados en diferentes algoritmos de aprendizaje. Este enfoque permite que los modelos base se complementen mutuamente (Alpaydin, 2014) (Géron, 2019).

2.2.6 SVM

Las Máquinas de Vectores de Soporte (SVM, por sus siglas en inglés) son reconocidas por su alta precisión en la clasificación, superando a otros métodos como la Regresión Logística y los Árboles de Decisión. Su capacidad para manejar eficazmente espacios de entrada no lineales mediante el uso de trucos de kernel es ampliamente conocida.

Las SVM construyen un hiperplano en un espacio de múltiples dimensiones para separar distintas clases de datos. Este hiperplano óptimo es generado de manera iterativa para minimizar el error. La idea principal de las SVM es encontrar un hiperplano con el mayor margen posible que divida de forma óptima el conjunto de datos en diferentes clases. Un hiperplano, según la geometría, es un subespacio que tiene una dimensión menor que el espacio en el que está contenido. Por ejemplo, en un espacio n-dimensional, un hiperplano

es un plano con una dimensión de $n - 1$. Su función principal es dividir el espacio en dos partes.

Se tiene un conjunto de ejemplos representados por $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$, donde cada uno tiene un vector de características x_n y una categoría y_n . Comenzando con dos clases, etiquetándolas como -1 y +1, si es posible dividir estas muestras en dos clases usando un hiperplano $D(x)$ de manera que cada ejemplo esté en el lado correcto, se dice el conjunto es separable (Alpaydin, 2014) (Carmona, 2016).

$$D(x) = (w_1x_1 + \dots + w_dx_d) + b = \langle w, x \rangle + b \quad (33)$$

Donde $\langle w, x \rangle$ representa el producto escalar de estos vectores. El objetivo es encontrar un hiperplano que separe estas clases de manera óptima. Esto implica encontrar un vector de peso (w) y un término de sesgo (b) de manera que todas las muestras estén del lado correcto del hiperplano con un margen de separación adecuado. Este hiperplano debe cumplir algunas restricciones para todo x_i :

$$\begin{aligned} \langle w, x_i \rangle + b &\geq 0 & \text{si } y_i &= +1 \\ \langle w, x_i \rangle + b &\leq 0 & \text{si } y_i &= -1, \quad i = 1, \dots, n \end{aligned} \quad (34)$$

Que se puede expresar de la siguiente manera:

$$y_i(\langle w, x_i \rangle + b) \geq 0 \quad (35)$$

O de manera más compacta:

$$y_i D(x_i) \geq 0 \quad (36)$$

Existen múltiples hiperplanos separables que pueden separar los ejemplos de manera correcta, y surge la pregunta sobre cómo definir uno óptimo. Para esto, se introduce el concepto de margen, τ , que se refiere a la mínima distancia entre el hiperplano y el ejemplo más cercano de cualquier clase, siendo un hiperplano óptimo aquel cuyo margen es máximo.

Por lo tanto, el hiperplano óptimo equidista del ejemplo más cercano de cada clase. La distancia entre un hiperplano de separación $D(x)$ y un ejemplo x' viene dada por:

$$\frac{|D(x')|}{\|w\|} \quad (37)$$

Donde, $\|w\|$ es la norma (o longitud) del vector de pesos w del hiperplano de separación y, típicamente se calcula mediante la norma euclidiana:

$$\|w\| = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2} = \sqrt{\langle w, w \rangle} \quad (38)$$

Entonces, para todos los ejemplos de entrenamiento se debe cumplir que la distancia entre cada uno de los ejemplos y el hiperplano de separación óptimo sea igual o mayor que el margen τ .

$$\frac{y_i D(x_i)}{\|w\|} \geq \tau \quad (39)$$

Los ejemplos situados en la frontera que delimita el margen a cada lado del hiperplano, es decir, que la distancia entre estos ejemplos y el hiperplano de separación es igual al margen τ , reciben el nombre de vectores soporte. Estos vectores de soporte son los ejemplos más cercanos al hiperplano de separación, son los más difíciles de clasificar y son los únicos que se consideran al construir el hiperplano. El objetivo es maximizar τ , pero debido a que el vector w puede ser escalado de manera infinita, existen infinitas soluciones que cumplen este objetivo. Para garantizar una solución única, la escala del producto de τ y la norma de w se fija a la unidad:

$$\tau \|w\| = 1 \quad (40)$$

Esto implica que para maximizar el margen, es necesario encontrar los valores de w y b que minimicen la norma de w , quedando el problema de la siguiente manera:

$$\begin{aligned}
\min \frac{1}{2} \|w\|^2 &= \frac{1}{2} \langle w, w \rangle \\
\text{sujeto a } y_i(\langle w, x_i \rangle + b) &\geq +1
\end{aligned} \tag{41}$$

Este dilema de optimización con restricciones se identifica como un problema de programación cuadrática y se puede abordar a través de la teoría de la optimización. Esta teoría establece que un problema de optimización, conocido como primal, posee una forma dual si la función que se desea optimizar y las restricciones son funciones estrictamente convexas. En tales circunstancias, resolver el problema dual proporciona la solución al problema primal. La ecuación presentada para el problema de optimización cumple con el criterio de convexidad y, por lo tanto, tiene un dual. Siguiendo este criterio, se pueden llevar a cabo la transformación del problema primal en su dual. Para lograr esto, se utiliza la técnica de multiplicadores de Lagrange α_i :

$$\begin{aligned}
L(w, b, \alpha) &= \frac{1}{2} \langle w, w \rangle - \sum_{i=1}^n \alpha_i [y_i(\langle w, x_i \rangle + b) - 1] \\
&= \frac{1}{2} \langle w, w \rangle - \sum_{i=1}^n \alpha_i y_i (\langle w, x_i \rangle + b) + \sum_{i=1}^n \alpha_i
\end{aligned} \tag{42}$$

El problema debe minimizarse con respecto a w y b , y maximizarse con respecto a α^t , que debe ser mayor o igual a cero. La solución se encuentra en el punto de silla, que, en términos generales, es un punto crítico donde la función tiene una forma que parece una silla de montar, es decir, tiene un máximo en una dirección y un mínimo en otra. Dado que el problema es convexo, se puede resolver de manera equivalente al problema dual haciendo uso de las condiciones de Karush-Kuhn-Tucker:

$$\frac{\partial L(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^n \alpha_i y_i x_i = 0 \tag{43}$$

$$\frac{\partial L(w, b, \alpha)}{\partial b} = \sum_{i=1}^n \alpha_i y_i = 0 \quad (44)$$

$$\alpha_i [1 - y_i (\langle w, x_i \rangle + b)] = 0 \quad (45)$$

Implementando esto a la función lagrangiana queda de la siguiente manera:

$$\begin{aligned} L(\alpha) &= \frac{1}{2} \left(\sum_{i=1}^n \alpha_i y_i x_i \right) \left(\sum_{j=1}^n \alpha_j y_j x_j \right) - \left(\sum_{i=1}^n \alpha_i y_i x_i \right) \left(\sum_{j=1}^n \alpha_j y_j x_j \right) + \sum_{l=1}^N \alpha_l \\ L(\alpha) &= -\frac{1}{2} \left(\sum_{i=1}^n \alpha_i y_i x_i \right) \left(\sum_{j=1}^n \alpha_j y_j x_j \right) + \sum_{l=1}^N \alpha_l \\ L(\alpha) &= \sum_{l=1}^N \alpha_l - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \end{aligned} \quad (46)$$

De esta se forma, se obtiene el problema dual, el cual hay que maximizar con respecto a α :

$$\begin{aligned} \max L(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{sujeto a } &\sum_{i=1}^n \alpha_i y_i = 0, \quad \alpha_i \geq 0 \end{aligned} \quad (47)$$

La solución del problema dual permite encontrar la solución del problema primal. Esto se logra substituyendo la solución del problema dual en la función de decisión en el problema primal:

$$D(x) = \sum_{i=1}^n \alpha_i^* y_i \langle x, x_i \rangle + b^* \quad (48)$$

Donde α_i^* se refiere a la solución óptima del problema dual y b^* se refiere al término de sesgo óptimo.

Cuando se espera que el hiperplano de separación divida perfectamente las clases, sin permitir que ningún punto de datos se encuentre en el lado incorrecto del hiperplano o dentro del margen, se le denomina margen duro. En la práctica, los datos del mundo real no son linealmente separables, lo que significa que no se puede encontrar un hiperplano que separe perfectamente las clases sin cometer errores de clasificación. Para manejar esta situación, se introduce el concepto de margen blando. Para esto se buscan hiperplanos que minimicen el error, introduciendo variables de holgura $\xi_i \geq 0$ para manejar dos tipos de desviación: cuando una instancia está mal clasificada y cuando está dentro del margen pero en el lado correcto del hiperplano. Relajando la restricción de la ecuación 41, queda de la siguiente manera:

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i \quad (49)$$

Si $\xi_i = 0$, no hay ningún problema con el punto x_i . Si $0 < \xi_i < 1$, x_i está clasificado correctamente pero dentro del margen. Si $\xi_i > 1$, x_i está mal clasificado. El número de clasificaciones erróneas es $\#\{\xi_i > 1\}$, y el número de puntos no separables es $\#\{\xi_i > 0\}$. Se introduce el concepto de 'error suave' como:

$$\sum_{i=1}^n \xi_i \quad (50)$$

y se incluye como un término de penalización en la función de pérdida:

$$f(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad (51)$$

El parámetro de regularización, C , en SVM, determina cómo se equilibra el ancho del margen y la cantidad de clasificaciones incorrectas. A medida que C aumenta, el margen se vuelve

más estrecho y se prioriza la minimización de errores. Así, un valor de C muy grande permitirá valores de ξ_i muy pequeños. Cuando C disminuye, se permite más flexibilidad, priorizando un margen más amplio, permitiendo valores de ξ_i más grandes. La forma primal del problema de optimización es el siguiente:

$$\begin{aligned} \min \quad & \frac{1}{2} \langle w, w \rangle + C \sum_{i=1}^n \xi_i \\ \text{sujeto a} \quad & y_i(\langle w, x_i \rangle + b) \geq +1 - \xi_i, \quad y \quad \xi_i \geq 0 \end{aligned} \quad (52)$$

En la forma dual, la formulación del SVM de margen suave es:

$$\begin{aligned} \max \quad & L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{sujeto a} \quad & \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \end{aligned} \quad (53)$$

Estos hiperplanos de separación son efectivos para clasificar cuando los ejemplos son separables o casi separables. Si la dimensionalidad del problema es baja, se puede resolver directamente el problema de optimización primal. Sin embargo, si la dimensionalidad es alta, es más conveniente transformar el problema primal en su forma dual. Pero, cuando el problema es no lineal, en lugar de ajustar un modelo no lineal directamente, se puede transformar el problema a un nuevo espacio utilizando funciones de base no lineales, a esto se le conoce como el truco kernel. En este nuevo espacio, se utiliza un modelo lineal que, en realidad, representa un modelo no lineal en el espacio original. A cada uno de estos espacios se les conoce como espacio de características F . Se utiliza una función de transformación $\phi(x)$ que asigna cada vector de entrada x a un punto en un espacio de características F :

$$\phi(x) = [\phi_1(x), \dots, \phi_m(x)] \quad (54)$$

En este espacio, se busca crear un hiperplano de separación lineal. La frontera de decisión lineal en el espacio de características se convierte en una frontera de decisión no lineal en el espacio original de entradas. La función de decisión en el espacio de características está dada por:

$$D(x) = (w_1\phi_1(x) + \dots + w_m\phi_m(x)) = \langle w, \phi(x) \rangle \quad (55)$$

Y la forma dual de la función de decisión es:

$$D(x) = \sum_{i=1}^n \alpha_i^* y_i \langle \phi(x), \phi(x_i) \rangle \quad (56)$$

El producto escalar $\langle \phi(x), \phi(x_i) \rangle$ se calcula a partir de lo que se denomina función kernel. Una función kernel K asigna a cada par de elementos del espacio de entrada x un valor real, que representa el producto escalar de las imágenes de estos elementos en un espacio de características F :

$$K(x, x') = \langle \phi(x), \phi(x') \rangle = (\phi_1(x)\phi_1(x'), \dots, (\phi_m(x)\phi_m(x'))) \quad (57)$$

Sustituyendo la función kernel en la función de decisión:

$$D(x) = \sum_{i=1}^n \alpha_i^* y_i K(x, x_i) \quad (58)$$

Por lo tanto, el problema es encontrar el valor de α que maximice el siguiente problema dual:

$$\max \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (59)$$

$$\text{sujeto a } \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C$$

Estos son algunos ejemplos de funciones de kernel:

❖ **Kernel lineal**

$$K_L(x, x') = \langle x, x' \rangle \quad (60)$$

❖ **Kernel polinómico de grado p**

$$K_P(x, x') = [\gamma \langle x, x' \rangle + \tau]^p, \quad \gamma > 0 \quad (61)$$

❖ **Kernel gaussiano**

$$K_G(x, x') = \exp(-\gamma \|x - x'\|^2) \equiv \exp(-\gamma \langle (x - x'), (x - x') \rangle), \quad \gamma > 0 \quad (62)$$

❖ **Kernel sigmoidal**

$$K_G(x, x') = \tanh(\gamma \langle x, x' \rangle + \tau) \quad (63)$$

El parámetro γ es un factor de ajuste que determina la suavidad del límite de decisión y controla la varianza del modelo. Cuando γ es grande, los límites de decisión son irregulares y oscilantes, lo que puede causar una alta varianza y sobreajuste. Por otro lado, si γ es pequeño, el límite de decisión es más suave y estable, con una varianza reducida.

Cuando el número de clases es mayor a dos, se utilizan varias estrategias para abordar este problema, entre los más importantes se encuentran el uno-contra-todos, la clasificación por pares y formulación de clasificación multiclase (Awad & Khanna, 2015):

Uno-contra-todos: La técnica Uno-contra-todos (OAA, One-Against-All) es una de las primeras y más populares implementaciones de SVM para problemas multiclase. Se basa en la creación de varios clasificadores binarios SVM, uno para cada clase, donde cada uno

intenta distinguir una clase específica del resto. Luego, se asigna una muestra a la clase que obtiene la puntuación más alta entre todos los clasificadores:

$$\text{Clase de } x = \operatorname{argmax}_{i=1,\dots,c} (w_i^T \phi(x) + b_i) \quad (64)$$

Donde c , es el número de clases.

Par a par: La clasificación par a par (OAO, One-Against-One) implica la creación de múltiples clasificadores SVM binarios, $c(c-1)/2$. Cada uno de estos clasificadores se enfoca en discernir entre dos de las c clases en el conjunto total. Esta técnica requiere la evaluación de un conjunto de clasificadores SVM binarios igual a $(c - 1)$. Para datos de entrenamiento de la clase k -ésima y la clase j -ésima, se establecen restricciones para (x_t, y_t) en el hiperplano de separación para garantizar que las muestras de la clase k -ésima estén separadas de las muestras de la clase j -ésima:

$$\begin{aligned} (w_{kj}^T \phi(x_t) + b_{kj}) &\geq 1 - \xi_{kj}^t, \quad \text{para } y_t = k \\ (w_{kj}^T \phi(x_t) + b_{kj}) &\leq -1 + \xi_{kj}^t, \quad \text{para } y_t = j \\ \xi_{kj}^t &\geq 0 \end{aligned} \quad (65)$$

Formulación de clasificación multiclase: La función objetivo para la clasificación multiclase tiene una estructura muy eficiente, ya que aborda el problema en una sola iteración. La función de decisión es la misma que la de la técnica de OAA. La función objetivo de clasificación multiclase construye c reglas de dos clases y c funciones de decisión resuelven las siguientes restricciones:

$$w_{y_i}^T \phi(x_i) + b_{y_i} \geq w_m^T \phi(x_i) + b_m + 2 - \xi_i^m, \quad \xi_i^m \geq 0 \quad (66)$$

2.2.7 KNN

El método de K vecinos más cercanos (KNN, K-Nearest Neighbor) se basa en aprender a través de comparaciones similares y se usa principalmente en reconocimiento de patrones. Usa múltiples atributos para reconocer las tuplas de entrenamiento en un espacio dimensional específico, estas tuplas están compuestas por un vector de características que describe las propiedades del objeto que se está analizando. Aunque KNN tiene un sesgo bajo, puede tener alta varianza, lo que podría llevar a sobreajustar los datos si son limitados o ruidosos. La dimensionalidad puede ser un desafío en espacios de alta dimensión, y es crucial seleccionar características relevantes y reducir la dimensionalidad antes de aplicar KNN (Subasi, 2020). KNN clasifica objetos sin etiquetar examinando las clases de los K objetos más próximos en el conjunto de entrenamiento. La etiqueta para un objeto sin etiquetar se determina según la clase más frecuente entre estos vecinos cercanos. Los elementos fundamentales de este enfoque son (Awad & Khanna, 2015):

- ❖ Un conjunto existente de objetos etiquetados.
- ❖ Una métrica de distancia para estimar la distancia entre objetos.
- ❖ El número de vecinos más cercanos (K).

El algoritmo calcula la distancia entre el objeto sin etiquetar y los objetos de entrenamiento para identificar los vecinos más cercanos. Luego, asigna la clase más común entre estos vecinos como la etiqueta para el objeto sin etiquetar.

El valor de K debe elegirse considerando el equilibrio entre la precisión y el riesgo de sobreajuste. Los valores más bajos de K tienen un sesgo bajo pero una varianza alta, mientras que los valores más altos de K tienen un sesgo alto pero una varianza baja. La elección de K depende de los datos, ya que aquellos con más ruido o valores atípicos pueden beneficiarse de valores más altos de K . Se sugiere que K sea impar para evitar empates en la clasificación, y la validación cruzada puede ayudar a determinar el valor óptimo de K para un conjunto de datos específico (IBM, s. f.).

Por otra parte, existen distintas métricas para calcular la distancia entre objetos:

❖ **Distancia euclidiana:**

$$d(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2} \quad (67)$$

❖ **Distancia Manhattan:**

$$d(x, y) = \sum_{i=1}^m |x_i - y_i| \quad (68)$$

❖ **Distancia Minkowsky:**

$$d(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (69)$$

La distancia de Minkowski es una generalización de varias métricas de distancia comunes, incluida la distancia euclidiana ($p=2$) y la distancia de Manhattan ($p=1$).

2.2.8 K-MEANS

Los algoritmos mencionados anteriormente, pertenecen a los algoritmos de aprendizaje supervisado, ya que se busca encontrar relaciones entre datos de entrada y salidas conocidas proporcionadas por un supervisor. Por otra parte que en el aprendizaje no supervisado no hay un supervisor y solo se tienen datos de entrada, el objetivo es descubrir patrones y uniformidades en estos datos. Una técnica común en el aprendizaje no supervisado es el agrupamiento, o clustering, que busca identificar agrupaciones naturales en los datos. Existen diferentes enfoques de clustering, como los basados en centroides, jerárquicos, basados en distribuciones y basados en densidad. (Subasi, 2020).

El algoritmo k-means es un método de clustering iterativo, basado en centroides, que divide N puntos de datos en K conjuntos separados con el objetivo de reducir al mínimo el criterio de suma de cuadrados. Dado que esta suma de cuadrados corresponde a la distancia euclidiana al cuadrado, esto se interpreta intuitivamente como la media más cercana (Awad & Khanna, 2015):

$$J = \sum_{j=1}^K \sum_{n=1}^N |x_n - \mu_j|^2 \quad (70)$$

J es la función de costo total, y representa la suma de las distancias al cuadrado de cada punto de datos x_n al centroide μ_j de su grupo asignado j ; estos centroides representan los puntos centrales alrededor de los cuales se agrupan los datos. En otras palabras, J mide el error o la discrepancia entre los puntos de datos y los centroides de los grupos asignados. La tarea del algoritmo k-means es minimizar esta función de costo, ajustando los centroides de manera iterativa hasta que se encuentren en una posición donde la función J se minimice.

El proceso del algoritmo es bastante simple, se realiza en dos pasos:

- 1) Asignación: Cada punto de datos se asigna al clúster cuyo centroide está más cerca de él.
- 2) Actualización: Se recalcula el centroide de cada clúster, moviéndolo hacia el centro (media) de todos los puntos de datos asignados a él.

Estos pasos se repiten hasta que se cumple un criterio de parada, cuando no hay más cambios en la asignación de puntos de datos. Cada iteración implica $N \times K$ comparaciones, lo que representa la complejidad computacional de cada iteración. Antes de comenzar este proceso, se seleccionan de forma aleatoria las ubicaciones iniciales de los k centroides dentro del conjunto de datos.

Por otra parte, aunque el algoritmo K-means se clasifica como un algoritmo no supervisado, todavía se necesita elegir un valor para K , que representa el número de grupos en los cuales se dividen los datos. En general, no hay una forma exacta de determinar este número, pero se pueden emplear ciertas reglas o estadísticas para ayudar en su estimación (Álvaro, 2023).

Método del codo: El método del codo busca minimizar la variabilidad dentro de los clústeres y maximizar la separación entre ellos, asegurando que las observaciones dentro de

un mismo clúster estén cercanas entre sí y que los clúster estén lo más separados posible. A medida que aumenta el número de clústeres (k), la variabilidad dentro de cada clúster tiende a disminuir, una menor distancia intra-clúster indica clúster más compactos. El método del codo busca identificar el valor de k en el cual un aumento adicional no mejora significativamente la distancia intra-clúster media.

El nombre "método del codo" proviene de la forma de la gráfica que se genera al trazar el número de clúster (k) en el eje x y la varianza (o la suma de cuadrados dentro de los clúster) en el eje y. El punto en el gráfico donde la disminución de la varianza comienza a disminuir drásticamente se conoce como el "codo". Este punto indica el número óptimo de clúster para el conjunto de datos.

Análisis de silueta: El análisis de la silueta evalúa la calidad del clustering al medir la separación entre los clúster. Examina qué tan cerca están los puntos de un clúster de los puntos de los clúster vecinos. Esta medida de distancia oscila entre -1 y 1, donde un valor alto indica un clustering efectivo. Los coeficientes de silueta cercanos a +1 señalan que la observación está considerablemente distante de los clúster vecinos. Un coeficiente de 0 indica que la observación está muy próxima o en el límite entre dos clúster. Por otro lado, valores negativos sugieren que dichas muestras podrían haber sido asignadas al clúster incorrecto. El coeficiente de silueta se determina empleando la distancia promedio dentro del clúster (a) y la distancia promedio al clúster más cercano (b) para cada muestra x_i en el conjunto X . Para cada x_i , a_i se obtiene como la distancia promedio hacia los otros puntos dentro de su mismo cluster, mientras que b_i se calcula como la distancia promedio hacia el cluster más cercano distinto al suyo. El valor de la silueta para x_i , se determina si el cluster al que pertenece contiene más de una muestra (Perucha Jurjo, 2022):

$$s(x_i, k) = \frac{b_i - a_i}{\max \{a_i, b_i\}} \quad (71)$$

Si $a_i < b_i$ el valor de s se acerca a 1, lo que indica que x_i está bien emparejado. Mientras que si $a_i > b_i$, lleva a que s esté cerca de -1, lo que indica que las muestras mal emparejadas. Al calcular el promedio de los índices de silueta de todos los puntos, se obtiene el índice de silueta para k grupos:

$$S(k) = \frac{1}{n} \sum_{i=1}^n \frac{b_i - a_i}{\max\{a_i, b_i\}} \quad (72)$$

El procedimiento para elegir el valor de k es el siguiente:

- 1) Ejecutar el algoritmo para diversos valores de k .
- 2) Calcular el índice de silueta $S(k)$ para cada agrupación en k clúster.
- 3) Elegir el número óptimo de clúster, aquel que maximiza el índice de silueta.

2.2.9 CLUSTERING JERÁRQUICO

Los algoritmos de clustering jerárquico crean una jerarquía de clúster en lugar de un único clustering. Los algoritmos de clustering jerárquico construyen una jerarquía de clustering anidados. Con más precisión, estos algoritmos incluyen N pasos, tantos como el número de vectores de datos. En cada paso t se obtiene un nuevo clustering sobre la base del clustering generado en el paso anterior $t-1$. Estos algoritmos son heurísticas y no buscan optimizar ninguna función objetivo. Por ello, es complicado evaluar la calidad del clustering resultante, ya que siempre generarán una estructura de clúster, incluso en ausencia de una estructura clara en los datos. Se dividen en dos enfoques principales: el aglomerativo, que une clúster similares, y el divisivo, que separa clúster según criterios específicos.

Clustering aglomerativo: El clustering aglomerativo comienza con N grupos, cada uno conteniendo inicialmente un solo objeto, y los fusiona gradualmente hasta formar un solo grupo que contiene todos los datos. Se utiliza un dendrograma para representar este proceso de fusión, donde la altura de las divisiones indica la disimilitud entre los grupos fusionados. Al inicio, cada grupo individual se representa en las hojas del árbol, en la parte inferior, y se unen progresivamente a medida que se fusionan dos grupos. La altura de las uniones indica la disimilitud entre los grupos fusionados. En la cima del árbol se encuentra la raíz, que representa una categoría que engloba todos los datos. Al cortar el árbol a una altura específica, podemos obtener un clustering de tamaño deseado. La propiedad de anidamiento sugiere que si dos vectores se unen en un clúster en un nivel de la jerarquía, permanecerán

juntos en niveles posteriores. Sin embargo, esto no permite corregir clúster "pobres" que puedan haberse formado antes.

Clustering divisivo: Los algoritmos divisivos, en contraste con los aglomerativos, comienzan con un único conjunto de datos y buscan la mejor forma de dividirlo en dos clúster. Este proceso se repite iterativamente hasta obtener un número N de clúster, cada uno con un solo conjunto de datos. Estos algoritmos pueden ser computacionalmente exigentes, especialmente para conjuntos de datos grandes, y su principal desventaja radica en esta demanda computacional. El clustering divisivo comienza con todos los datos en un solo clúster y los divide en clúster más pequeños de manera iterativa. Dado que hay $2^{N-1} - 1$ formas de dividir un grupo de N elementos en dos grupos, es difícil calcular la división óptima, por lo que se utilizan varias heurísticas. Una estrategia para el clustering divisivo es seleccionar el clúster con el diámetro más grande y dividirlo en dos usando el algoritmo de k-means o k-medoids con $K = 2$, conocido como algoritmo de k-means bisectivo. Otra estrategia implica la construcción de un árbol de expansión mínimo a partir del grafo de disimilitud y la creación de nuevos clúster al romper la conexión asociada con la mayor disimilitud.

Antes de realizar cualquier clustering, se requiere determinar la matriz de disimilitud. La matriz de disimilitud proporciona una medida cuantitativa de la distancia o diferencia entre los puntos de datos. Esta medida de disimilitud puede basarse en diversas métricas de distancia, como la distancia euclidiana, la distancia de Manhattan, la correlación, entre otras. La matriz de disimilitud suele ser simétrica y cuadrada, con dimensiones igual al número de elementos en el conjunto de datos. Cada entrada en la matriz representa la distancia o diferencia entre un par específico de elementos.

Existen varios métodos para medir la distancia entre clúster y decidir cuáles fusionar en cada paso del algoritmo de clustering jerárquico (Tan et al., 2013):

- ❖ **Enlace Simple o MÍN:** En el enfoque de clustering jerárquico conocido como enlace simple o MÍN, la cercanía entre dos clúster se determina por la distancia mínima (o similitud máxima) entre cualquier par de puntos en dichos clúster. Este método funciona agregando enlaces entre los puntos uno por uno, comenzando con los enlaces más cortos. Aunque es efectivo para manejar formas no elípticas, es vulnerable al ruido y a los valores atípicos.

- ❖ **Enlace Completo o MÁX:** El enfoque de enlace completo para el clustering jerárquico define la proximidad entre dos clúster como la máxima distancia entre cualquier par de puntos en los clúster respectivos. Este método gradualmente une los puntos uno por uno, comenzando por los enlaces más cortos, hasta que todos los puntos en un grupo estén completamente conectados y formen un "clique". Aunque el enlace completo es menos sensible al ruido y a los valores atípicos, puede romper clúster grandes y tiende a favorecer formas globulares.
- ❖ **Promedio de Grupo:** En el enfoque de promedio de grupo para el clustering jerárquico, la proximidad entre dos clúster se define como el promedio de la proximidad entre todos los pares de puntos en los clúster respectivos. Este método se sitúa entre los enfoques de enlace simple y enlace completo. La proximidad entre dos clúster se calcula como la suma de las distancias (proximidad) entre todos los pares de puntos en los clúster, dividida por el producto del tamaño de los clúster:

$$Proximidad(C_i, C_j) = \frac{\sum_{x \in C_i} \sum_{y \in C_j} proximidad(x, y)}{m_i m_j} \quad (73)$$

- ❖ **Método de Ward:** En el método de Ward, la cercanía entre dos grupos se define como el incremento en el error cuadrático que ocurre al fusionar ambos grupos. Esto significa que, al igual que en el clustering K-means, se busca minimizar la suma de los cuadrados de las diferencias entre los puntos y sus respectivos centroides.

Los dendrogramas son útiles para representar la secuencia de clúster generados, y cortarlos puede resultar en un clustering en un nivel específico. Los dendrogramas de proximidad indican en qué nivel se fusionan por primera vez dos clúster. Estos métodos son útiles para comprender la estructura de los datos y la formación de clúster. Es importante calcular un nivel adecuado para cortar el dendrograma en relación con la jerarquía resultante. Dependiendo del método de clustering utilizado, se puede observar la disminución del error de reconstrucción o el aumento de la verosimilitud logarítmica en función de k. Se busca el "codo" en la curva para determinar el número óptimo de clústeres. Del mismo modo, en el clustering jerárquico, se pueden analizar las diferencias entre niveles en el dendrograma para

tomar decisiones sobre la división de clústeres. Para elegir el número óptimo de clústeres en un problema de clustering, existen varias estrategias:

- ❖ En algunas aplicaciones como la cuantificación de colores, k está definido por la aplicación.
- ❖ Graficar los datos en dos dimensiones usando PCA puede ayudar a descubrir la estructura de los datos y el número de clústeres en los datos.
- ❖ Establecer una distancia máxima permitida es equivalente a establecer un error de reconstrucción máximo permitido por instancia.
- ❖ En ciertas aplicaciones, la validación manual de los clústeres puede ser útil, examinando si los clústeres reflejan grupos significativos en los datos.

2.2.10 ANÁLISIS DE COMPONENTES PRINCIPALES

El Análisis de Componentes Principales (PCA) es una técnica no supervisada que busca simplificar datos en un espacio de dimensiones reducidas, al mismo tiempo que maximiza la varianza y minimiza la pérdida de información. Se considera un método no supervisado, ya que no utiliza la información de salida; el criterio a maximizar es la varianza.

El PCA busca proyectar datos de un espacio de dimensión d a otro de dimensión k (donde $k < d$), minimizando la pérdida de información. La proyección de un vector x se realiza en la dirección de un vector w como $Z = w^T x$ (Alpaydin, 2014).

Considerando un grupo de variables $(x_1, x_2, \dots, x_d)$ se busca calcular, a partir de éste, un nuevo conjunto de variables $(z_1, z_2, \dots, z_d)$ que no sean correlacionadas y cuya varianza disminuya progresivamente. Cada variable z_j se define como una combinación lineal de las variables originales (de la Fuente, s.f.):

$$z_j = w_{j1}x_1 + w_{j2}x_2 + \dots + w_{jd}x_d = w_j^T x \quad (74)$$

Donde $w_j^T = (w_{j1}, w_{j2}, \dots, w_{jd})$ es un vector de constantes, y $x = \begin{pmatrix} x_1 \\ \dots \\ x_d \end{pmatrix}$.

El primer componente principal, w_1 , se selecciona de tal forma que la dispersión de la muestra sea máxima, resaltando las diferencias entre los puntos. Para maximizar la varianza y mantener la ortogonalidad de la transformación, se impone que el módulo del vector w_1 sea igual a 1, $\|w_1\| = 1$.

Sabiendo que $z_1 = w_1^T x$, entonces la varianza del primer componente se expresa como:

$$Var(z_1) = w_1^T \Sigma w_1 \quad (75)$$

Donde Σ se refiere a la matriz de covarianza de x que también puede denotarse como $Cov(x)$ y generalmente se expresa como:

$$\Sigma = \begin{bmatrix} Var(x_1) & Cov(x_1, x_2) & \cdots & Cov(x_1, x_d) \\ Cov(x_2, x_1) & Var(x_2) & \cdots & Cov(x_2, x_d) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(x_d, x_1) & Cov(x_d, x_2) & \cdots & Var(x_d) \end{bmatrix}$$

Se busca el valor de w_1 que maximice la varianza $Var(z_1)$ bajo la restricción $w_1^T w_1 = 1$, formulándolo como un problema de Lagrange queda como:

$$L(w_1) = w_1^T \Sigma w_1 - \alpha(w_1^T w_1 - 1) \quad (76)$$

Para maximizar esta función, se realiza la derivada parcial con respecto a w_1 y se iguala a 0, estableciendo que:

$$\frac{\partial L(w_1)}{\partial w_1} = 2\Sigma w_1 - 2\alpha I w_1 = 0 \Rightarrow (\Sigma - \alpha I)w_1 = 0 \quad (77)$$

Donde I se refiere a la matriz identidad. Para que el sistema tenga una solución distinta de cero, la matriz $(\Sigma - \alpha I)$ tiene que ser singular, es decir, que su determinante debe ser igual a cero, $|\Sigma - \alpha I| = 0$. Esto implica que α es un autovalor de Σ .

Para maximizar la varianza se toma el eigenvector con el mayor eigenvalor. Por lo tanto, el componente principal es el vector propio de la matriz de covarianza de la muestra de entrada con el mayor valor propio:

$$\lambda_1 = \alpha \quad (78)$$

El segundo componente principal w_2 se obtiene de manera similar. También debe maximizar la varianza, tener longitud unitaria y además, se requiere que z_2 este incorrelado con el componente anterior z_1 , es decir que $Cov(z_2, z_1) = 0$. Por lo tanto, se requiere que $w_2^T \Sigma w_1 = 0$, teniendo así:

$$L(w_2) = w_2^T \Sigma w_2 - \alpha(w_2^T w_2 - 1) - \beta(w_2^T w_1 - 0) \quad (79)$$

realizando la derivada parcial con respecto a w_2 e igualándola a 0, se obtiene:

$$\frac{\partial L(w_2)}{\partial w_2} = 2\Sigma w_2 - 2\alpha w_2 - \beta w_1 = 0 \quad (80)$$

Multiplicando por w_1^T la ecuación anterior queda:

$$2w_1^T \Sigma w_2 - 2\alpha w_1^T w_2 - \beta w_1^T w_1 \quad (81)$$

Cabe destacar que $w_1^T w_2 = 0$, ya que w_1 y w_2 son ortogonales, y $w_1^T w_1 = 1$. Por lo que la ecuación se simplifica de la siguiente manera:

$$2w_1^T \Sigma w_2 - \beta = 0 \quad (82)$$

Luego se obtiene el valor de β :

$$\beta = 2w_1^T \Sigma w_2 = 2w_2^T \Sigma w_1 = 0 \quad (83)$$

De este modo se tiene que:

$$\frac{\partial L(w_2)}{\partial w_2} = 2\Sigma w_2 - 2\alpha w_2 = (\Sigma - \alpha I)w_2 = 0 \quad (84)$$

De este modo, se puede retomar la ecuación 77 y, aplicando el mismo razonamiento, implica que w_2 debe ser el vector propio de Σ con el segundo mayor valor propio, $\lambda_2 = \alpha$.

Este mismo razonamiento se puede aplicar para obtener el resto de los componentes, de modo que al j -ésimo componente le correspondería el j -ésimo autovalor.

Aunque todos los valores propios sean mayores que cero, si el valor propio λ_i es bajo, sugiere que el componente contribuye poco a la varianza y puede ser eliminados (Alpaydin, 2014). Generalmente, se selecciona un número de componentes (k) que expliquen más del 90% de la varianza total. Como los λ_i están ordenados en orden decreciente, la proporción de varianza explicada por los k componentes principales es:

$$\text{Proporción de varianza} = \frac{\lambda_1 + \lambda_2 + \dots + \lambda_k}{\lambda_1 + \lambda_2 + \dots + \lambda_k + \dots + \lambda_d} \quad (85)$$

Si las dimensiones están fuertemente correlacionadas, suele haber un número reducido de componentes principales, lo que resulta en una considerable reducción de dimensionalidad. En cambio, si las dimensiones no están correlacionadas, k se acercará a d , sin que haya ventajas significativas.

Además de este método existe el gráfico de sedimentación, similar al método del codo mencionado anteriormente, que ilustra la varianza explicada en función del número de componentes retenidos. Este gráfico ayuda a determinar cuántos componentes son óptimos, identificando un punto de "codo" donde añadir más componentes no incrementa significativamente la varianza explicada.

2.3 INTELIGENCIA ARTIFICIAL: DEEP LEARNING

El machine learning ha sido fundamental en el avance de la inteligencia artificial, permitiendo a los algoritmos aprender patrones a partir de datos y realizar tareas específicas sin programación explícita. Los algoritmos básicos de aprendizaje automático mencionados

anteriormente son efectivos en una diversidad de problemas relevantes, pero tienen algunas limitaciones que pueden dificultar su aplicación en problemas complejos. Esta limitación se debe en parte a la dificultad de generalizar en entornos de datos de alta dimensionalidad, donde los mecanismos tradicionales de aprendizaje no son suficientes y pueden resultar costosos computacionalmente. El aprendizaje profundo (Deep learning) se diseñó para superar estos desafíos y otros similares (Goodfellow et al., 2016).

La dimensionalidad es un desafío que enfrentan los problemas de aprendizaje automático cuando hay muchas dimensiones en los datos. En estos casos, el número de posibles combinaciones de variables crece de forma exponencial, lo que dificulta hacer generalizaciones precisas. En entornos de alta dimensionalidad, como aquellos con grandes conjuntos de características, los modelos de machine learning pueden enfrentar dificultades para capturar relaciones complejas entre las variables y, por lo tanto, no pueden generalizar de manera efectiva a nuevos ejemplos.

Por otra parte, para que los algoritmos de aprendizaje automático puedan generalizar con precisión, necesitan incorporar suposiciones previas sobre la naturaleza de la función que están intentando aprender. Estas suposiciones pueden ser explícitas, como distribuciones de probabilidad sobre los parámetros del modelo, o implícitas, como tendencias hacia ciertos tipos de funciones. Uno de los sesgos implícitos más comunes, es el sesgo de suavidad o constancia local, que postula que la función aprendida no debería experimentar cambios drásticos en pequeñas áreas del espacio de entrada. Sin embargo, este sesgo por sí solo puede ser insuficiente para tareas sofisticadas de inteligencia artificial, por lo que el aprendizaje profundo introduce sesgos adicionales para mejorar la generalización en estos casos. Los algoritmos tradicionales, como el K-NN y los árboles de decisión, pueden verse limitados por esta aproximación. El aprendizaje profundo aborda estas limitaciones introduciendo suposiciones adicionales y estructuras jerárquicas en la representación de los datos, lo que permite capturar patrones complejos y generalizar de manera efectiva en entornos de alta dimensionalidad. Esto hace que el aprendizaje profundo sea una herramienta poderosa para una amplia gama de tareas de inteligencia artificial.

Así como la naturaleza ha inspirado una gran cantidad de inventos, las redes neuronales artificiales (ANNs) están inspiradas en la arquitectura del cerebro humano, pero han evolucionado para ser distintas de sus contrapartes biológicas. Las ANNs gradualmente se

han vuelto bastante diferentes de sus primos biológicos, tanto que, algunos investigadores sugieren abandonar completamente la analogía biológica. Las redes neuronales artificiales son fundamentales para el aprendizaje profundo, ya que son versátiles, potentes y escalables. Se utilizan en una amplia gama de tareas complejas de aprendizaje automático, como clasificación de imágenes masivas, reconocimiento de voz, recomendación de contenido multimedia, etc. El deep learning aprovecha estas redes neuronales para aprender y extraer patrones complejos a partir de grandes conjuntos de datos. Al utilizar capas múltiples de neuronas interconectadas, las ANNs pueden aprender representaciones cada vez más abstractas y sofisticadas de los datos de entrada (Géron, 2019).

Desde su introducción en 1943 por Warren McCulloch y Walter Pitts, las redes neuronales artificiales han experimentado altibajos en su popularidad. Ahora, con la disponibilidad masiva de datos, el aumento de la potencia informática y los avances en los algoritmos de entrenamiento, las ANN han experimentado otro renacimiento. Este resurgimiento se debe a su capacidad para abordar problemas complejos y la continua innovación en el campo.

Las neuronas artificiales se basan en un modelo simplificado de las neuronas biológicas, propuesto por McCulloch y Pitts. Este modelo puede tener una o varias entradas binarias y una salida binaria, activándose la salida cuando un cierto número de entradas están activadas. Se pueden construir redes de neuronas artificiales para realizar diversas operaciones lógicas simples, como AND, OR y NOT, combinando las activaciones de las entradas para obtener resultados lógicos más complejos.

2.3.1 HISTORIA DE LAS REDES NEURONALES

La historia de la computación neuronal abarca desde la introducción del perceptrón en la década de 1950 hasta el desarrollo de técnicas avanzadas como las redes generativas adversariales en la década de 2010. Cada avance ha contribuido al desarrollo y la comprensión de las redes neuronales artificiales (Bagnato, 2019):

- ❖ 1958: Perceptrón: Frank Rosenblatt creó el perceptrón, una unidad básica de las redes neuronales artificiales que toma entradas binarias y produce una salida binaria. Puede ser utilizado para decisiones binarias o para implementar funciones lógicas como OR y AND.

- ❖ 1965: Perceptrón Multicapa: Una extensión del perceptrón original, con capas de entrada, oculta y salida. Todavía utiliza valores binarios para las entradas y salidas.
- ❖ 1980s: Neuronas Sigmoideas: Introducción de neuronas sigmoideas que permiten entradas con valores reales, y la inclusión de neuronas "bias". Esto permite ajustes más finos en los pesos de las conexiones.
- ❖ Redes Feedforward: Redes en las que las salidas de una capa son utilizadas como entradas en la próxima capa, sin loops hacia atrás.
- ❖ 1986: Backpropagation: Algoritmo que permite entrenar redes neuronales de múltiples capas de manera supervisada, ajustando los pesos de las conexiones para minimizar el error.
- ❖ 1989: Redes Neuronales Convolucionales (CNN): Arquitectura útil en el procesamiento de imágenes, con capas que extraen características y clasifican.
- ❖ 1997: Memoria a Corto y Largo Plazo (LSTM) / Redes Neuronales Recurrentes (RNN): Arquitectura que permite conexiones hacia atrás entre las capas, útiles para procesar datos de series temporales.
- ❖ 2006: Redes Neuronales Profundas (DBN): Introducción de Deep Belief Networks, que demuestran la importancia del preentrenamiento para redes neuronales profundas.
- ❖ 2014: Redes Generativas Adversariales (GAN): Dos modelos de redes neuronales compiten, uno generador, que toma “datos basura” como entrada y genera muestras; y otro discriminador, para generar datos realistas.

2.3.2 ANALOGÍA ENTRE NEURONAS BIOLÓGICAS Y ARTIFICIALES

Antes de abordar las redes neuronales, es crucial entender la neurona artificial, el componente fundamental de estas estructuras. Este concepto esencial proporciona las bases necesarias para comprender cómo funcionan las redes neuronales y su aplicación en una variedad de campos. Como se mencionó anteriormente, estas se basan en el cerebro humano, que está compuesto por miles de millones de células llamadas neuronas y utiliza estas unidades básicas de procesamiento de información, las cuales están interconectadas mediante puntos de conexión funcionales llamados sinapsis. A través de estas sinapsis, se transmiten señales

eléctricas y químicas que permiten la comunicación y el procesamiento de información en el cerebro. Las neuronas tienen tres partes fundamentales: dendritas, soma o cuerpo celular, y axón. Las dendritas reciben señales del exterior hacia el soma, mientras que el axón transmite señales hacia otras neuronas. La sinapsis es el espacio entre neuronas vecinas, donde ocurre la comunicación. En la neurona transmisora, el potencial de acción viaja a través del axón hacia la sinapsis, donde la información electroquímica se transmite a otras células. Esta información es procesada por el núcleo en el soma y generada como respuesta, propagada luego por el axón. La sinapsis tiene características eléctricas que pueden modular la señal eléctrica según sea necesario (Torral, 2021).

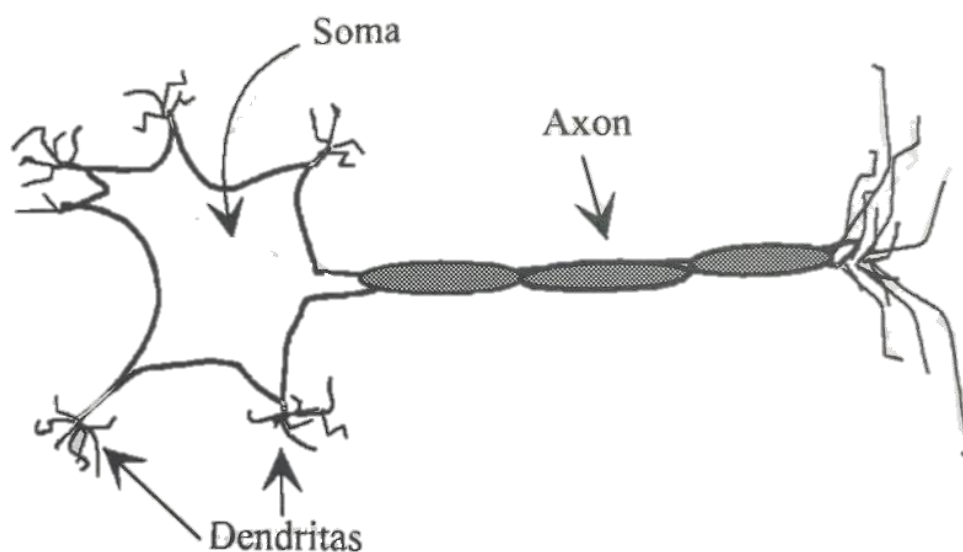


Figura 1. Neurona biológica (Tomada de Avila, 2021).

Las redes neuronales artificiales imitan el funcionamiento de las redes neuronales biológicas, utilizando nodos llamados neuronas artificiales. Estas neuronas están interconectadas, al igual que las neuronas en el cerebro. Cada neurona artificial realiza operaciones matemáticas en las entradas ponderadas que recibe y produce una salida. El aprendizaje en estas redes se logra ajustando los pesos de las conexiones entre las neuronas, lo que se hace presentando ejemplos de entrada junto con las salidas deseadas y utilizando algoritmos de aprendizaje para minimizar la diferencia entre las salidas producidas y las deseadas. Este enfoque ha demostrado su utilidad en una amplia gama de aplicaciones.

En las Redes Neuronales Artificiales (ANN), el equivalente a la neurona biológica es el elemento procesador (PE). Este elemento procesador recibe múltiples entradas que se

combinan, generalmente mediante una suma básica. Luego, la suma de estas entradas se modifica a través de una función de transferencia, y el valor resultante se utiliza como salida del elemento procesador. Esta salida puede estar conectada a las entradas de otros elementos procesadores mediante conexiones ponderadas que representan la eficacia de las conexiones neuronales en la sinapsis (Olabe, 2008).

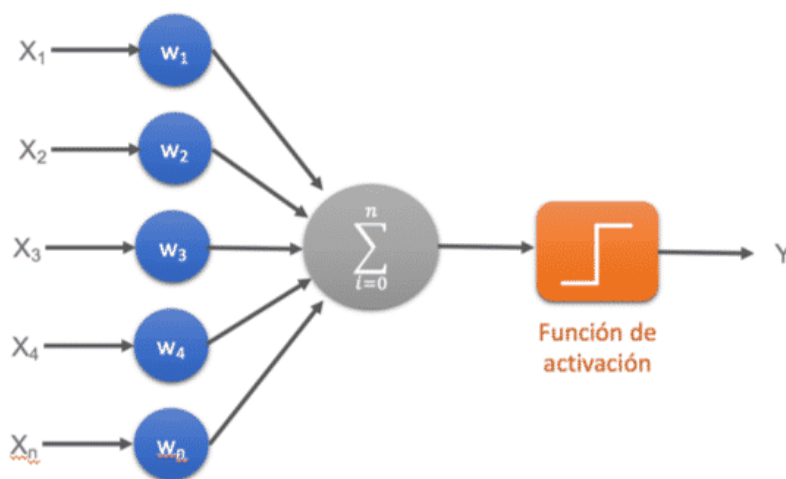


Figura 2. Neurona Artificial (Tomada de Calvo, 2019).

Desde los primeros perceptrones hasta las redes generativas adversariales, estas redes han evolucionado para abordar diversas tareas, desde la clasificación de imágenes hasta la generación de datos realistas. Las redes neuronales, se dividen en varias categorías según su estructura y función; según su arquitectura, como las redes feedforward y recurrentes, y según su capacidad para aprender y adaptarse, como las supervisadas y no supervisadas (Olabe, 2008).

2.3.3 PERCEPTRÓN

El Perceptrón es una de las arquitecturas más simples en redes neuronales artificiales. Se basa en una unidad de procesamiento llamada unidad de lógica de umbral (TLU, threshold logic unit), que es una variante de la neurona artificial. A diferencia de las entradas (X) y salidas binarias (z), en el Perceptrón son números y cada conexión de entrada está asociada con un peso (W). La TLU calcula una suma ponderada de sus entradas (Géron, 2019):

$$z = (w_1x_1 + w_2x_2 + \dots + w_nx_n) = X^TW \quad (86)$$

Luego, aplica una función de paso a esa suma para generar la salida.

$$h_W(x) = \text{paso}(z) \quad (87)$$

La función de paso más comúnmente empleada en los Perceptrones es la función de paso de Heaviside, aunque a veces se usa la función de signo. Un solo TLU puede ser utilizado para una simple clasificación binaria lineal: calcula una combinación lineal de las entradas y, si esta suma supera un umbral, emite la clase positiva; de lo contrario, emite la clase negativa. El Perceptrón está formado por una sola capa de TLUs, donde cada TLU está conectada a todas las entradas. Cuando todas las neuronas de una capa están vinculadas a cada neurona en la capa anterior, se denomina capa completamente conectada o densa. Los Perceptrones usan neuronas de entrada especiales que transmiten los datos directamente. Además, se suele agregar una característica de sesgo adicional ($x_0 = 1$) utilizando una neurona de sesgo especial que siempre emite una salida de 1. Un Perceptrón con múltiples salidas puede clasificar simultáneamente las instancias de entrada en diferentes clases binarias, lo que lo convierte en un clasificador multioutput.

La siguiente ecuación permite calcular eficientemente las salidas de una capa de neuronas artificiales para varias instancias simultáneamente:

$$h_{W,b}(X) = \phi(WX + b) \quad (88)$$

Donde:

- X es la matriz de características de entrada, con una fila por instancia y una columna por característica.
- W es la matriz de pesos que contiene los pesos de conexión entre las neuronas de entrada y las neuronas artificiales en la capa, excluyendo los pesos del sesgo.
- b es el vector de sesgo que contiene los pesos de conexión entre la neurona de sesgo y las neuronas artificiales.

- ϕ es la función de activación, que en este caso es una función escalón cuando las neuronas artificiales son TLUs.

El algoritmo de entrenamiento del Perceptrón se basa en la regla de Hebb, que sugirió que cuando una neurona biológica activa repetidamente a otra neurona, la conexión entre estas dos neuronas se fortalece. En otras palabras "Las células que disparan juntas, se conectan juntas"; es decir, el peso de conexión entre dos neuronas tiende a aumentar cuando disparan simultáneamente.

2.3.4 FUNCIÓN DE ACTIVACIÓN

La selección de la función de activación es fundamental en el diseño de redes neuronales, ya que, como su nombre lo indica, determinan si una neurona debe activarse o no. En el caso del perceptrón, se utilizan las funciones de activación paso o signo debido a la necesidad de predecir una etiqueta de clase binaria. No obstante, en otros contextos, donde se predicen diferentes tipos de variables objetivo, se pueden emplear otras funciones de activación.

La relevancia de las funciones de activación no lineales se incrementa al pasar del perceptrón de una sola capa a las arquitecturas de redes neuronales de múltiples capas. En estas arquitecturas más complejas, se pueden utilizar diferentes tipos de funciones no lineales, como la signo, la sigmoide o la tangente hiperbólica. A continuación se describen algunas de las funciones de activación más comunes (Ponce, 2021):

Función lineal: También conocida como función identidad, esta función asegura que la salida sea igual a la entrada. Para una red neuronal de múltiples capas, si se utiliza una función lineal, el modelo se comporta como una regresión lineal. Por lo tanto, esta función de activación lineal es adecuada cuando se desea realizar una regresión lineal en la salida, generando un único valor.

$$\phi(x) = k * x \quad (89)$$

Función de paso binario: La función de Heaviside, o función escalón, fue una de las primeras utilizadas para modelar redes neuronales. Esta función produce un valor de 0 hasta que se supera un umbral, momento en el cual cambia a 1. La función de Heaviside no es

adecuada para el entrenamiento de redes neuronales debido a que su derivada es siempre cero, lo que impide el proceso de aprendizaje.

Función de paso bipolar: Es bastante parecida a la función de activación sigmoidea, compartiendo incluso la forma en S. Esta función acepta cualquier valor real como entrada y produce salidas en el rango de -1 a 1. A medida que la entrada aumenta (se hace más positiva), el valor de salida se acerca a 1, mientras que a medida que la entrada disminuye (se hace más negativa), la salida se acerca a -1.

Función sigmoideal binaria: Históricamente, la función sigmoide es la función de activación más antigua y popular.

$$\phi(z) = \frac{1}{1 + e^{-z}} \quad (90)$$

Para valores negativos grandes de z , el término e^{-z} en el denominador crece exponencialmente, y $\phi(z)$ se aproxima a 0. Al contrario, valores positivos grandes de z reducen e^{-z} hacia 0, y $\phi(z)$ se aproxima a 1. Aunque las funciones sigmoideas fueron fundamentales en muchas redes neuronales durante décadas, han perdido popularidad recientemente. Esto se debe a que en redes neuronales profundas, el problema del desvanecimiento del gradiente hace que el entrenamiento sea extremadamente difícil. Este problema ocurre durante el entrenamiento cuando el error se propaga hacia atrás a través de las capas, haciéndose cada vez más pequeño en cada iteración, lo que dificulta el aprendizaje efectivo de la red.

Función sigmoideal bipolar: Es bastante parecida a la función de activación sigmoidea, compartiendo incluso la forma en S. Esta función acepta cualquier valor real como entrada y produce salidas en el rango de -1 a 1. A medida que la entrada aumenta (se hace más positiva), el valor de salida se acerca a 1, mientras que a medida que la entrada disminuye (se hace más negativa), la salida se acerca a -1.

$$\phi(z) = \frac{1 - e^{-z}}{1 + e^{-z}} \quad (91)$$

Función tangente hiperbólica (Tanh): La función de activación tangente hiperbólica tiene valores de salida que van de -1 a 1. Aunque está centrada en torno a 0 y se considera una variación de la función logística, comparte con la sigmoide el problema del desvanecimiento del gradiente.

$$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (92)$$

Función ReLU La función ReLU es popular debido a su simplicidad de implementación y su eficacia para superar las limitaciones de otras funciones de activación, como Sigmoid y Tanh. Es menos propensa al problema de la desaparición de gradientes, que dificulta el entrenamiento de modelos profundos. Sin embargo, puede presentar otros problemas, como la saturación. Se dice que una función sufre saturación no lineal cuando su derivada tiende rápidamente a cero fuera de la región comprendida entre K y $-K$, donde K es un número real específico. La función ReLU se calcula de la siguiente manera:

$$\phi(z) = \max(0, z) \quad (93)$$

Esto significa que si el valor de entrada (z) es negativo, se devuelve un valor 0; de lo contrario, se devuelve el valor.

Función Softmax: La función softmax produce un vector de valores cuya suma es 1, lo que permite interpretarlos como probabilidades de pertenecer a una clase. La entrada a la función es un vector de valores reales, y la salida es un vector de la misma longitud con valores que suman 1.0, representando probabilidades. La función softmax se calcula de la siguiente manera:

$$\phi(z_i) = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \quad (94)$$

Donde k representa el número total de clases.

Por lo general, se emplea una función de activación no lineal, que sea diferenciable en las capas ocultas de una red neuronal, ya que esto permite que el modelo aprenda funciones más complejas que una red entrenada mediante una función de activación lineal. Las funciones de activación utilizadas para las capas ocultas más utilizadas son (Rodríguez, 2022):

- ❖ Activación lineal rectificada (ReLU)
- ❖ Logística (Sigmoidal)
- ❖ Tangente hiperbólica (Tanh)

Los modelos de redes neuronales modernas con arquitecturas comunes, como Perceptrón Multicapa (MLP) y Redes Neuronales Convolucionales (CNN), utilizarán la función de activación de ReLU o extensiones. Las redes recurrentes (RNN) todavía usan comúnmente funciones de activación sigmoidea o Tanh, o incluso ambas.

Las funciones de activación utilizadas para las capas de salida más utilizadas son:

- ❖ Lineal
- ❖ Logística (sigmoidea)
- ❖ Softmax

La función de activación de salida lineal no modifica la suma ponderada de la entrada y se utiliza únicamente cuando la salida requiere una regresión lineal.

La función sigmoide se utiliza principalmente en la salida sigmoidal porque sus valores están en el rango de 0 a 1. Esto la hace especialmente útil para modelos en los que se necesita predecir una probabilidad, ya que las probabilidades siempre están limitadas a este rango entre 0 y 1.

La función SoftMax se emplea en la clasificación multiclase porque asigna una probabilidad a cada clase. Esto la hace útil cuando es necesario determinar la probabilidad de que una instancia pertenezca a cada una de las clases posibles.

2.3.5 PERCEPTRÓN MULTICAPA

Las redes neuronales feedforward profundas, también conocidas como perceptrones multicapa (MLP), son un tipo fundamental de modelo en el aprendizaje profundo. Las redes neuronales multicapa tienen más de una capa de procesamiento. A diferencia del perceptrón, que solo tiene una capa de entrada y una de salida, en la cual se realizan todos los cálculos y estos son visibles para el usuario, las redes multicapa incluyen capas ocultas adicionales entre

la entrada y la salida. Estas capas se denominan capas ocultas porque los cálculos realizados no son visibles para el usuario. Estas redes buscan aproximar una función f^* , mapeando entradas x a salidas y mediante una función $y = f(x; \theta)$, donde θ representa los parámetros aprendidos durante el entrenamiento. Estos parámetros incluyen los pesos y sesgos en cada capa de la red (Goodfellow et al., 2016).

Las redes neuronales feedforward se construyen conectando una serie de funciones en un grafo dirigido acíclico. Por ejemplo, una red puede tener funciones $f^{(1)}, f^{(2)}, f^{(3)}$ dispuestas en secuencia, lo que da lugar a $f(x) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right)$. Estas redes están organizadas en capas, con cada función correspondiente a una capa, y la longitud de la cadena de funciones define la profundidad de la red. Es de esta terminología de donde surge el nombre "aprendizaje profundo". El objetivo durante el entrenamiento es hacer que la función $f(x)$ se aproxime lo más posible a la función objetivo f^* .

Las redes neuronales multicapa se denominan redes feed-forward porque la información fluye de la entrada a la salida en una única dirección, sin retroalimentación. En estas redes, cada nodo de una capa está conectado a todos los nodos de la capa siguiente. La estructura de la red se define por el número de capas y los nodos en cada capa. En redes con k capas, el número de neuronas en cada capa se denota como por $p_1, \dots, p_k$, y el tamaño de las salidas de cada capa se representa por los vectores $\bar{h}_1, \dots, \bar{h}_k$, donde cada $\bar{h}_i$ tiene dimensiones p_i . La dimensionalidad de una capa es el número de unidades en esa capa (Aggarwal, 2018).

Las matrices de pesos en una red neuronal multicapa tienen tamaños determinados por las capas que conectan. La matriz de pesos W_1 que conecta la capa de entrada con la primera capa oculta tiene un tamaño de $d \times p_1$. Las matrices entre capas ocultas consecutivas son W_r , tienen dimensiones $p_r \times p_{r+1}$. La matriz de pesos que va de la última capa oculta a la capa de salida, W_{k+1} , es de tamaño $p_k \times o$. El vector de entrada de dimensión d , $\bar{x}$, se transforma en las salidas usando las siguientes ecuaciones recursivas:

$$\bar{h}_1 = \phi(W_1^T \bar{x}) \quad (\text{Entrada a Capa Oculta}) \quad (95)$$

$$\bar{h}_{p+1} = \phi(W_{p+1}^T \bar{h}_p) \quad \forall p \in \{1 \dots k-1\} \quad (\text{Capa Oculta a Capa Oculta}) \quad (96)$$

$$\bar{o} = \phi(W_{k+1}^T \bar{h}_k) \quad (\text{Capa Oculta a Capa de Salida}) \quad (97)$$

Las funciones de activación, como la sigmoide, se aplican a cada elemento de un vector. Algunas funciones, como el softmax, se adaptan naturalmente a vectores y, típicamente, suelen usarse en las capas de salida. Aunque cada unidad de una red neuronal contiene una sola variable, muchos diagramas arquitectónicos combinan las unidades en una sola capa para crear una unidad vectorial y las conexiones entre las unidades vectoriales ahora se representan mediante matrices. Por lo general, todas las unidades en una capa utilizan la misma función de activación. Este método es común, salvo en la capa de salida.

Aunque las redes neuronales totalmente conectadas pueden funcionar bien en muchos casos, a menudo se obtiene un rendimiento superior al reducir o compartir las conexiones según el conocimiento del dominio específico de los datos. Un buen ejemplo es la red neuronal convolucional, diseñada especialmente para datos de imagen, lo que ayuda a reducir el riesgo de sobreajuste al incorporar sesgos específicos del dominio. El sobreajuste ocurre cuando el número de parámetros libres (que generalmente es igual al número de conexiones de peso) es demasiado grande en comparación con el tamaño de los datos de entrenamiento, lo que lleva a un buen desempeño en los datos de entrenamiento pero a una mala generalización en datos nuevos. Por lo tanto añadir más nodos tiende a agravar este problema ya que se incrementa también el número de parámetros.

2.3.6 ENTRENAMIENTO DE REDES NEURONALES

La mayoría de los algoritmos de aprendizaje profundo implican algún tipo de optimización. La optimización en aprendizaje profundo se centra en minimizar o maximizar una función $f(x)$ ajustando x . La función que se desea optimizar se denomina función objetivo, función de costo, función de pérdida o función de error. El valor que minimiza o maximiza la función se denota con un superíndice *, tal que $x^* = \operatorname{argmin} f(x)$. La derivada de una función $y=f(x)$ indica cómo cambia la salida y con un pequeño cambio en la entrada x (Goodfellow et al., 2016):

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x) \quad (98)$$

La derivada es útil y se usa para minimizar la función moviendo x en la dirección opuesta a la derivada, un método conocido como descenso de gradiente. Este método ajusta x en pequeños pasos para reducir la función $f(x)$. Cuando la derivada es cero no aporta información, estos puntos se denominan críticos o estacionarios, que pueden ser mínimos locales, máximos locales o puntos de silla. El objetivo es mover x en la dirección opuesta al gradiente para encontrar el mínimo más bajo posible. Un mínimo global es el punto en el que una función tiene su valor más bajo absoluto y puede haber uno o varios de estos mínimos. Los mínimos locales son puntos donde la función es más baja en comparación con los alrededores, pero no necesariamente el mínimo global. En el aprendizaje profundo, la optimización es complicada debido a la presencia de numerosos mínimos locales y puntos de silla en funciones multidimensionales. Generalmente, se intenta encontrar un valor de la función que sea muy bajo, aunque no se garantiza que sea el mínimo absoluto.

Para funciones con varias entradas, se usan derivadas parciales para ver cómo cambia la función con respecto a cada variable individual. La derivada parcial $\frac{\partial}{\partial x_i} f(x)$ mide cómo cambia f a medida que solo la variable x_i aumenta en el punto x . El gradiente es un vector que contiene todas estas derivadas parciales, denotado como $\nabla_x f(x)$, y se utiliza para encontrar puntos críticos donde todas las derivadas parciales son cero. La derivada direccional mide la pendiente de la función en una dirección específica. Para minimizar la función, se busca la dirección en la que la función disminuye más rápidamente, que es la dirección opuesta al gradiente (Goodfellow et al., 2016).

La derivada direccional en la dirección de un vector unitario u es la tasa de cambio de la función f en esa dirección, es decir, la pendiente de la función f en la dirección de u . En otras palabras, la derivada direccional es la derivada de la función $f(x+\alpha u)$ con respecto a α , cuando $\alpha=0$. Utilizando la regla de la cadena, esta derivada se calcula como:

$$\frac{\partial}{\partial \alpha} f(x + \alpha u) = u^T \nabla_x f(x), \quad \text{para } \alpha = 0 \quad (99)$$

Para minimizar una función, se busca la dirección en la que la función disminuye más rápidamente. Para esto se utiliza la derivada direccional:

$$\min_{u, u^T u = 1} u^T \nabla_x f(x) = \min_{u, u^T u = 1} \|u\|_2 \|\nabla_x f(x)\|_2 \cos \theta \quad (100)$$

donde θ se refiere al ángulo entre u y el gradiente. El subíndice 2 indica la norma L2 que es simplemente la longitud del vector en un espacio euclidiano. Al simplificar la expresión, minimizando $\cos \theta$, se concluye que la función f se reduce moviéndose en la dirección opuesta al gradiente. Por lo tanto, para minimizar la función, el vector u debe apuntar en la dirección opuesta al gradiente, lo que implica seguir el gradiente negativo. Este método de optimización se conoce como descenso del gradiente o descenso más pronunciado. El descenso más pronunciado propone un nuevo punto (Goodfellow et al., 2016):

$$x' = x - \epsilon \nabla_x f(x) \quad (101)$$

La tasa de aprendizaje ϵ controla el tamaño del paso en el descenso de gradiente. Se puede elegir de varias maneras, como estableciendo ϵ como una constante pequeña, encontrando el tamaño del paso que hace la derivada direccional cero, o $f(x - \epsilon \nabla_x f(x))$ con diferentes valores de ϵ elegir el que minimice la función objetivo. El descenso de gradiente converge cuando el gradiente es igual a cero o lo suficientemente cercano a cero. En algunos casos es posible saltar directamente al punto crítico resolviendo $\nabla_x f(x) = 0$.

Descenso de gradiente estocástico

El descenso de gradiente estocástico (SGD, Stochastic Gradient Descent) es una versión del algoritmo de descenso de gradiente utilizado en el aprendizaje profundo. Permite manejar grandes conjuntos de datos de manera eficiente al calcular el gradiente usando solo un pequeño subconjunto de datos (minibatch), a diferencia del descenso de gradiente clásico, que calcula el gradiente sobre todo el conjunto de datos. Esto reduce el costo computacional y permite entrenar modelos grandes de manera más rápida. Aunque no siempre encuentra el mínimo global, suele alcanzar un error bajo a un costo manejable.

En el aprendizaje automático, la función de costo frecuentemente se calcula sumando las pérdidas individuales de cada ejemplo en el conjunto de entrenamiento. Para determinar el

gradiente de esta función, es necesario sumar las derivadas de las pérdidas respecto a los parámetros para todos los ejemplos, lo cual es computacionalmente costoso. Cuando se trabaja con conjuntos de datos extremadamente grandes, este cálculo se vuelve inviable debido al considerable aumento en el tiempo requerido para cada actualización del gradiente. El SGD se basa en la idea de que el gradiente puede estimarse con un pequeño grupo de ejemplos, en lugar de usar todo el conjunto de datos. Específicamente, en cada paso del algoritmo, se muestrea un minibatch de ejemplos $B = \{x^{(1)}, \dots, x^{(m')}\}$ extraídos uniformemente del conjunto de entrenamiento. El tamaño del minibatch m' se elige típicamente como un número relativamente pequeño de ejemplos, que varía de 1 a unos cientos. La estimación del gradiente se forma como (Goodfellow et al., 2016):

$$g = \frac{1}{m'} \nabla_{\theta} \sum_{i=1}^{m'} L(x^{(i)}, y^{(i)}, \theta) \quad (102)$$

Usando ejemplos del minibatch B, el algoritmo de descenso de gradiente estocástico actualiza los parámetros del modelo en la dirección opuesta al gradiente estimado, de la siguiente manera:

$$\theta \leftarrow \theta - \epsilon g \quad (103)$$

En el algoritmo SGD, la tasa de aprendizaje es un parámetro crucial. Aunque inicialmente se puede utilizar una tasa de aprendizaje fija, en la práctica es mejor reducir esta tasa progresivamente con el tiempo. Por eso, se usa una tasa de aprendizaje variable, denotada como ϵ_k , que cambia en cada iteración k . Esto es necesario porque el estimador de gradiente de SGD incluye un componente de ruido debido al muestreo aleatorio de ejemplos de entrenamiento, el cual no desaparece incluso cuando se alcanza un mínimo. Para asegurar la convergencia de SGD, se deben cumplir dos condiciones: la suma infinita de las tasas de aprendizaje debe ser infinita:

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \quad (104)$$

y la suma infinita de los cuadrados de las tasas de aprendizaje debe ser finita:

$$\sum_{k=1}^{\infty} \epsilon_k^2 < \infty \quad (105)$$

En la práctica es común reducir la tasa de aprendizaje linealmente hasta la iteración τ :

$$\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau \quad (106)$$

donde $\alpha = \frac{k}{\tau}$. Después de la iteración τ , es común mantener constante ϵ .

Elegir la tasa de aprendizaje en el algoritmo de SGD se basa a menudo en prueba y error, aunque es más efectivo hacerlo mediante la monitorización de las curvas de aprendizaje. Al usar una programación lineal, se deben seleccionar tres parámetros: ϵ_0 (tasa inicial), ϵ_τ (tasa final) y τ (número de iteraciones para reducir la tasa). Generalmente, τ se fija en el número de iteraciones necesarias para pasar por el conjunto de entrenamiento pocas centenas de veces, y ϵ_τ se establece alrededor del 1% de ϵ_0 . La clave es ajustar ϵ_0 : si es demasiado alta, puede causar oscilaciones grandes en la función de costo; si es demasiado baja, el aprendizaje será lento y podría estancarse. Por lo tanto, es útil comenzar con una tasa de aprendizaje inicial alta (pero no excesiva) y ajustar según las primeras iteraciones para encontrar un equilibrio entre rapidez de aprendizaje y estabilidad.

Una ventaja clave del algoritmo de Descenso de Gradiente Estocástico (SGD) y de métodos similares basados en minibatch es que el tiempo de cálculo por actualización no aumenta con el tamaño del conjunto de datos. Esto permite que SGD funcione eficientemente incluso con grandes volúmenes de datos, y puede alcanzar una tolerancia fija en el error de prueba antes de procesar todo el conjunto de entrenamiento.

La tasa de convergencia de un algoritmo de optimización se mide por el error excesivo, $J(\theta) - \min_{\theta} J(\theta)$, que es la diferencia entre la función de costo actual y el costo mínimo

posible. Además, aunque el análisis asintótico muestra que SGD converge más lentamente, en la práctica, SGD suele ser más eficiente con grandes conjuntos de datos debido a su rapidez en cada iteración. Por lo tanto, en aplicaciones prácticas, los algoritmos de optimización que combinan el descenso de gradiente por lotes y estocástico, o que ajustan el tamaño del minibatch durante el entrenamiento, pueden ofrecer beneficios significativos.

En redes neuronales de una sola capa, el entrenamiento es simple porque el error se calcula directamente con respecto a los pesos, facilitando el cálculo de gradientes. En redes de múltiples capas, el error es una función compleja de los pesos en capas anteriores, lo que hace que el cálculo del gradiente sea más complicado. El algoritmo de retropropagación (backpropagation) resuelve esto usando la regla de la cadena del cálculo diferencial para calcular los gradientes de manera eficiente. Este algoritmo, que se basa en programación dinámica, se divide en dos fases: la fase hacia adelante y la fase hacia atrás (Aggarwal, 2018):

Fase hacia adelante: Se introducen las entradas en la red neuronal para una instancia de entrenamiento. Esto provoca una serie de cálculos a través de las capas usando los pesos actuales. La salida obtenida se compara con la salida esperada y se calcula la derivada de la función de pérdida respecto a la salida. Esta información se usa para determinar cómo varía la pérdida con respecto a los pesos en la siguiente fase.

Fase hacia atrás: En esta fase, se determina cómo varía la función de pérdida en relación con los diferentes pesos de la red, aplicando la regla de la cadena. Este cálculo se realiza en orden inverso, comenzando desde el nodo de salida y retrocediendo a través de las capas ocultas, con el objetivo de actualizar los pesos según estos gradientes. Considerando una secuencia de unidades ocultas $h_1, h_2, \dots, h_k$ seguidos por la salida o , sobre la cual se evalúa la función de pérdida L . Además, el peso de la conexión entre la unidad oculta h_r a h_{r+1} es $w_{(h_r, h_{r+1})}$. Si existe una única ruta desde una unidad oculta hasta la salida, se puede calcular el gradiente de la función de pérdida respecto a los pesos en esa ruta mediante la regla de la cadena:

$$\frac{\partial L}{\partial w_{(h_{r-1}, h_r)}} = \frac{\partial L}{\partial o} \cdot \left[\frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right] \frac{\partial h_r}{\partial w_{(h_{r-1}, h_r)}} \quad \forall r \in 1 \dots k \quad (107)$$

Esta fórmula solo considera una ruta entre h_1 y o , pero en realidad puede haber muchas rutas posibles. Para manejar esta complejidad, se utiliza una versión generalizada de la regla de la cadena llamada "regla de la cadena multivariable". Esta regla calcula el gradiente considerando todas las rutas posibles en el gráfico computacional desde h_1 y o . La fórmula generalizada para el gradiente con múltiples rutas es:

$$\frac{\partial L}{\partial w_{(h_{r-1}, h_r)}} = \frac{\partial L}{\partial o} \cdot \underbrace{\left[\sum_{[h_r, h_{r+1}, \dots, h_k, o]} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right]}_{\text{La retropropagación calcula } \Delta(h_r, o) = \frac{\partial L}{\partial h_r}} \frac{\partial h_r}{\partial w_{(h_{r-1}, h_r)}} \quad \forall r \in 1 \dots k \quad (108)$$

Aunque el cálculo del gradiente total, $\Delta(h_r, o) = \frac{\partial L}{\partial h_r}$, parece complicado debido a que el número de rutas posibles en una red neuronal puede crecer exponencialmente, este problema se maneja de manera eficiente gracias a que el grafo de la red neuronal no contiene ciclos. En lugar de calcular todas las rutas posibles, el proceso comienza calculando el término $\Delta(h_k, o)$ para las unidades más cercanas a la salida o . Luego, estos valores se propagan hacia atrás, capa por capa, usando los resultados ya obtenidos para las unidades en capas posteriores, lo que permite una forma ordenada y eficiente de calcular los gradientes sin tener que evaluar todas las rutas directamente. Además, el valor de $\Delta(o, o)$ para cada nodo de salida se inicializa como:

$$\Delta(o, o) = \frac{\partial L}{\partial o} \quad (109)$$

Que es la derivada de la función de pérdida con respecto a la salida. Esta técnica de programación dinámica permite calcular eficientemente gradientes en redes neuronales acíclicas, evitando un número exponencial de operaciones. Para calcular la recursión de $\Delta(h_r, o)$ se usa la regla de la cadena multivariable, que suma las contribuciones de todos los nodos a los que el nodo h_r está conectado.

$$\Delta(h_r, o) = \frac{\partial L}{\partial h_r} = \sum_{h: h_r \Rightarrow h} \frac{\partial L}{\partial h} \frac{\partial h}{\partial h_r} = \sum_{h: h_r \Rightarrow h} \frac{\partial h}{\partial h_r} \Delta(h, o) \quad (110)$$

Dado que cada unidad h se encuentra en una capa posterior a h_r , el valor de $\Delta(h, o)$ ya se ha calculado cuando se determinó $\Delta(h_r, o)$. Sin embargo, para completar el cálculo, es necesario determinar cómo h cambia con respecto a h_r . Suponiendo que el peso de la conexión entre h_r y h es $w_{(h_r, h)}$, y que a_h es el valor calculado en la unidad oculta h antes de aplicar la función de activación ϕ . Es decir, h se calcula como $h = \phi(a_h)$. Entonces, se puede derivar la siguiente expresión para $\frac{\partial h}{\partial h_r}$:

$$\frac{\partial h}{\partial h_r} = \frac{\partial h}{\partial a_h} \cdot \frac{\partial a_h}{\partial h_r} = \frac{\partial \phi(a_h)}{\partial a_h} \cdot w_{(h_r, h)} = \phi'(a_h) \cdot w_{(h_r, h)} \quad (111)$$

Sustituyendo $\frac{\partial h}{\partial h_r}$ en la ecuación 110, queda de la siguiente manera:

$$\Delta(h_r, o) = \sum_{h: h_r \Rightarrow h} \phi'(a_h) \cdot w_{(h_r, h)} \cdot \Delta(h, o) \quad (112)$$

Los gradientes se acumulan de manera sucesiva al retroceder a través de la red, procesando cada nodo solo una vez durante cada pasada hacia atrás. Para cada borde que llega a un nodo, es necesario repetir los cálculos indicados en la ecuación 110, que dependen del número de bordes salientes, para obtener el gradiente de todos los pesos de los bordes entrantes al nodo. Finalmente, el cálculo de $\frac{\partial h_r}{\partial w_{(h_{r-1}, h_r)}}$ en la ecuación 108 se simplifica de la siguiente manera:

$$\frac{\partial h_r}{\partial w_{(h_{r-1}, h_r)}} = h_{r-1} \cdot \phi'(a_{h_r}) \quad (113)$$

En la retropropagación, el gradiente principal que se calcula es el de las activaciones de la capa, y el gradiente respecto a los pesos de los bordes incidentes se puede determinar fácilmente.

Existen diferentes métodos para realizar la recursión en la programación dinámica de la ecuación 112, dependiendo de qué variables se elijan para el cálculo intermedio. Aunque hay distintas formas de llevar a cabo la recursión, todas conducen al mismo resultado final en el proceso de retropropagación. En la ecuación 108 se utilizan las variables de las capas ocultas como elementos de la cadena para la recursión. Sin embargo, también es posible emplear los valores de preactivación de las variables en la regla de la cadena. Los valores de preactivación se obtienen después de aplicar la transformación lineal, pero antes de la función de activación. Este valor de preactivación es a_h . En lugar de usar la ecuación 108, se puede aplicar una regla de la cadena basada en estos valores de preactivación.

$$\frac{\partial L}{\partial w_{(h_{r-1}, h_r)}} = \underbrace{\frac{\partial L}{\partial o} \cdot \phi'(a_o) \cdot \left[\sum_{[h_r, h_{r+1}, \dots, h_k, o]} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial a_{h_{i+1}}}{\partial a_{h_i}} \right]}_{\text{La retropropagación calcula } \delta(h_r, o) = \frac{\partial L}{\partial a_{h_r}}} \frac{\partial h_r}{\partial w_{(h_{r-1}, h_r)}} \quad (114)$$

Se utilizó la notación $\delta(h_r, o) = \frac{\partial L}{\partial a_{h_r}}$ para definir la ecuación recursiva. El valor inicial de $\delta(o, o) = \frac{\partial L}{\partial a_o}$ se establece de la siguiente manera:

$$\delta(o, o) = \frac{\partial L}{\partial a_o} = \phi'(a_o) \cdot \frac{\partial L}{\partial o} \quad (115)$$

Después, se puede aplicar la regla de la cadena para múltiples variables para crear una recursión similar:

$$\delta(h_r, o) = \frac{\partial L}{\partial a_{h_r}} = \sum_{h: h_r \Rightarrow h} \frac{\overbrace{\frac{\partial L}{\partial a_h}}^{\delta(h, o)}}{\underbrace{\phi'(a_{h_r}) \cdot w_{(h_r, h)}}_{\frac{\partial a_h}{\partial a_{h_r}}}} = \phi'(a_{h_r}) \sum_{h: h_r \Rightarrow h} w_{(h_r, h)} \cdot \delta(h, o) \quad (116)$$

La derivada parcial de la pérdida con respecto al peso se calcula utilizando $\delta(h_r, o)$ de la siguiente manera:

$$\frac{\partial L}{\partial w_{(h_{r-1}, h_r)}} = \delta(h_r, o) \cdot h_{r-1} \quad (117)$$

Al igual que en redes de una sola capa, los nodos en redes neuronales multicapa se actualizan repetidamente durante varias épocas hasta que el modelo converge.

La función de pérdida de una red neuronal se puede escribir de la siguiente manera:

$$L = \sum_{i=1}^n L_i \quad (118)$$

Donde L_i es la pérdida del i -ésimo punto de entrenamiento. En el descenso de gradiente, se busca minimizar la función de pérdida ajustando los parámetros del modelo en la dirección negativa del gradiente. Para el caso del perceptrón, si los parámetros son $\bar{W} = (w_1, \dots, w_d)$, la actualización de los pesos se realiza usando:

$$\bar{W} \leftarrow \bar{W} - \alpha \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots, \frac{\partial L}{\partial w_d} \right) \quad (119)$$

En notación vectorial, esta actualización se puede simplificar como:

$$\bar{W} \leftarrow \bar{W} - \alpha \frac{\partial L}{\partial \bar{W}} \quad (120)$$

Para redes neuronales de una sola capa como el perceptrón, el descenso de gradiente se aplica solo a los pesos $\bar{W}$. En redes neuronales más complejas con múltiples capas, se deben actualizar todos los parámetros utilizando retropropagación. Dado que el número de parámetros puede ser muy grande, calcular el gradiente usando todos los datos simultáneamente es impráctico debido a las altas demandas de memoria.

En el SGD, en lugar de procesar todo el conjunto de datos a la vez, se utilizan pequeños lotes de datos para estimar el gradiente, lo que simplifica los cálculos y reduce los requisitos de

memoria. Esto es posible porque la función de pérdida puede expresarse como la suma de las pérdidas individuales de cada punto de entrenamiento:

$$\frac{\partial L}{\partial \bar{W}} = \sum_{i=1}^n \frac{\partial L_i}{\partial \bar{W}} \quad (121)$$

Este enfoque permite actualizar los pesos de manera eficiente incluso con grandes volúmenes de datos. De este modo se actualizan los pesos del modelo utilizando la pérdida de un solo punto de entrenamiento L_i :

$$\bar{W} \leftarrow \bar{W} - \alpha \frac{\partial L_i}{\partial \bar{W}} \quad (122)$$

recibe su nombre porque actualiza los parámetros del modelo utilizando puntos de entrenamiento seleccionados de manera aleatoria, en lugar de usar el conjunto completo de datos en cada iteración. Aunque cada actualización es solo una aproximación del gradiente real y puede no ser tan precisa como el descenso de gradiente tradicional, el impacto acumulativo de muchas actualizaciones tiende a ser similar al resultado de usar todos los datos a la vez. Este método es eficiente y rápido, aunque a veces puede ser menos preciso en el ajuste a los datos de entrenamiento. No obstante, en los datos de prueba, el rendimiento del SGD suele ser igual o incluso mejor que el del descenso de gradiente tradicional. En el descenso de gradiente estocástico con minibatch, se utiliza un conjunto $B = \{j_1, \dots, j_m\}$ de puntos de entrenamiento para la actualización:

$$\bar{W} \leftarrow \bar{W} - \alpha \sum_{i \in B} \frac{\partial L_i}{\partial \bar{W}} \quad (123)$$

Este enfoque proporciona un buen equilibrio entre estabilidad, velocidad y uso de memoria. Sin embargo, el uso de mini-lotes implica que las salidas y gradientes se gestionan como matrices en lugar de vectores, lo que incrementa los requisitos de memoria. Esto limita el tamaño del mini-lote que se puede usar, ya que una mayor memoria permite manejar mini-

lotes más grandes, pero también aumenta los requerimientos computacionales. El tamaño del mini-lote en el SGD debe adaptarse a la cantidad de memoria disponible en el hardware. Un mini-lote muy pequeño puede resultar ineficiente debido al alto costo computacional asociado. Generalmente, tamaños de mini-lotes de unos pocos cientos de puntos son ideales, y usar valores como 32, 64, 128 ó 256 (Aggarwal, 2018) suele ofrecer un mejor rendimiento en la mayoría de los sistemas. Aunque aumentar el tamaño del mini-lote más allá de ciertos límites no mejora significativamente la precisión del gradiente, elegir el tamaño adecuado ayuda a optimizar tanto la eficiencia como el uso de memoria.

Las redes neuronales profundas enfrentan problemas de estabilidad durante el entrenamiento, especialmente en redes con muchas capas. Estos problemas surgen debido a la forma en que los gradientes se relacionan entre las capas anteriores y posteriores.

Por ejemplo, una red profunda con muchas capas, cada una de estas con una sola neurona, y usando la función de activación sigmoide, los gradientes pueden disminuir significativamente al retropropagar a través de múltiples capas. Esto se debe a que el gradiente en una capa se multiplica por valores que tienden a ser menores que 1, haciendo que el gradiente en capas anteriores sea mucho más pequeño. Este fenómeno, conocido como el problema del gradiente desvanecido, significa que las primeras capas reciben actualizaciones muy pequeñas, lo que dificulta el aprendizaje. Sea $\frac{\partial L}{\partial h_t}$ la derivada de la función de pérdida con respecto a la activación oculta h_t , es relativamente fácil usar la actualización de retropropagación para mostrar la siguiente relación:

$$\frac{\partial L}{\partial h_t} = \phi'(h_{t+1}) \cdot w_{t+1} \cdot \frac{\partial L}{\partial h_{t+1}} \quad (124)$$

Se puede asumir que los pesos se inicializan con una distribución normal estándar, por lo que cada peso w_t tiene una magnitud promedio esperada de 1.

En las redes neuronales profundas que usan la función de activación sigmoide, los gradientes pueden decrecer mucho cuando se retropropagan a través de muchas capas. La derivada de la sigmoide, que alcanza un máximo de 0.25, provoca que los gradientes se reduzcan notablemente con cada capa adicional. Por ejemplo, después de 10 capas, los gradientes pueden caer a 10^{-6} de su valor inicial. La solución a este problema implica ajustar la

inicialización de los pesos y las funciones de activación, pero hacerlo incorrectamente puede llevar al problema opuesto, el gradiente explosivo.

Aunque este ejemplo usa solo un nodo por capa, los problemas de gradientes desvanecidos y explosivos también se presentan en redes con múltiples nodos en cada capa. Durante la retropropagación, se realiza una multiplicación de matrices en lugar de una simple multiplicación escalar, lo que puede ser inestable. En particular, los derivados de la pérdida en la capa $(i+1)$ se multiplican por una matriz llamada Jacobiano J , que contiene las derivadas de las activaciones en la capa $(i+1)$ con respecto a las de la capa i , es decir:

$$J_{kr} = \frac{\partial f_k(\bar{z}_i)}{\partial \bar{z}_i^{(r)}} \quad (125)$$

Donde, J_{kr} es un elemento de la matriz Jacobiana; $\bar{z}_i$ es el vector columna de activaciones en la dirección hacia delante de la capa i ; $\bar{z}_i^{(r)}$ es el elemento r -ésimo de $\bar{z}_i$, o sea, la salida de la neurona r después de aplicar una función de activación en la capa i ; y $f_k(\bar{z}_i)$ es una función de activación.

La estabilidad del gradiente depende de las propiedades del Jacobiano, pero las condiciones ideales para evitar problemas de gradientes desvanecidos o explosivos rara vez se cumplen. Mantener una tasa de aprendizaje constante en el entrenamiento de algoritmos puede ser problemático. Si es baja, el algoritmo tardará mucho en encontrar una buena solución. Si es alta, el algoritmo puede oscilar o divergir después de acercarse inicialmente a una solución. Para evitar estos problemas, se recomienda permitir que la tasa de aprendizaje decaiga con el tiempo.

Las dos formas más comunes de decaimiento son el decaimiento exponencial, donde la tasa de aprendizaje disminuye exponencialmente con el tiempo; y el decaimiento inverso, en el cual la tasa de aprendizaje disminuye en función del tiempo de manera inversa.

$$\alpha_t = \alpha_0 e^{(-kt)} \quad \text{Decaimiento Exponencial} \quad (126)$$

$$\alpha_t = \frac{\alpha_0}{1 + kt} \quad \text{Decaimiento Inverso} \quad (127)$$

El parámetro k controla la tasa de decaimiento y t se refiere al número de la época actual.

Momentum

Existen técnicas conocidas como basadas en momentum, las cuales abordan el problema del zigzagueo en el descenso por gradiente, que ocurre cuando los pasos son contradictorios y se cancelan entre sí, reduciendo la efectividad de los pasos en la dirección correcta. En lugar de solo ajustar el tamaño del paso, el momentum considera una dirección promediada de los últimos pasos para suavizar el zigzagueo. En la figura 3, se observa como las técnicas basadas en momentum convergen de manera más rápida, además de que el número de oscilaciones se reduce, lo significa que las actualizaciones de los parámetros tienden a seguir una trayectoria más estable y menos errática. Las actualizaciones normales para el descenso por gradiente son (Aggarwal, 2018):

$$\bar{V} \leftarrow -\alpha \frac{\partial L}{\partial \bar{W}}; \quad \bar{W} \leftarrow \bar{W} + \bar{V} \quad (128)$$

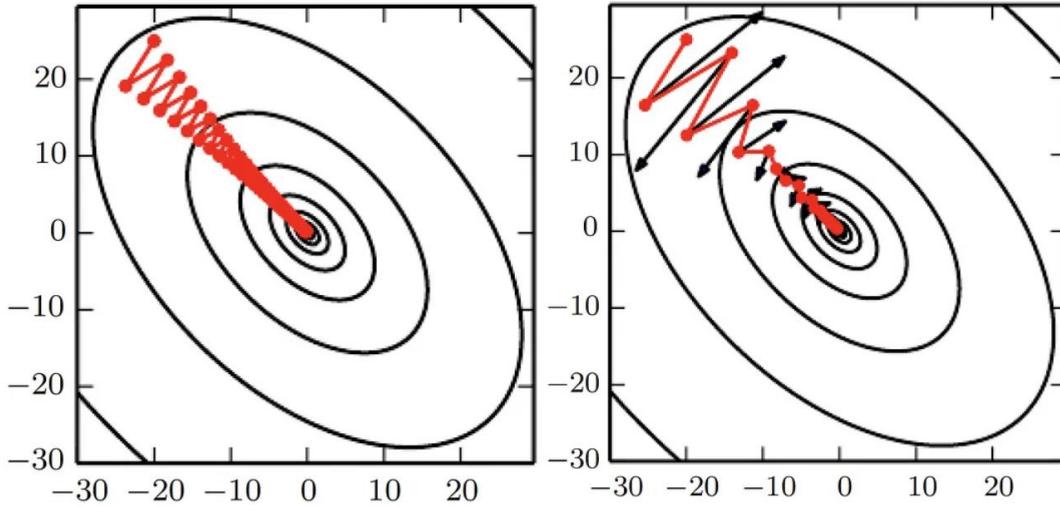


Figura 3 Diferencia entre el SGD y el SGD con momentum. (tomado de Goodfellow et al., 2016).

Mientras que en el descenso con momentum, el vector de actualización $\bar{V}$ se ajusta con suavizado exponencial:

$$\bar{V} \leftarrow \beta \bar{V} - \alpha \frac{\partial L}{\partial \bar{W}}; \quad \bar{W} \leftarrow \bar{W} + \bar{V} \quad (129)$$

donde β es el parámetro de suavizado (también conocido como el parámetro de momentum o fricción). El uso de momentum puede causar un pequeño sobrepaso en la solución óptima, esto puede ayudar a evitar óptimos locales y navegar mejor en regiones planas de la superficie de pérdida.

Algoritmos con Tasas de Aprendizaje Adaptativas

Ajustar la tasa de aprendizaje es complicado en el entrenamiento de redes neuronales debido a su gran influencia en el rendimiento del modelo, pero ahora existen métodos que adaptan las tasas de aprendizaje (Aggarwal, 2018):

AdaGrad: AdaGrad es un algoritmo de optimización que ajusta las tasas de aprendizaje para cada parámetro del modelo de forma adaptativa. Para cada parámetro w_i , el algoritmo lleva un registro acumulativo de la magnitud cuadrada de las derivadas parciales de la función de pérdida L con respecto a ese parámetro. Este valor acumulado se denota como A_i (Aggarwal, 2018).

$$A_i \leftarrow A_i + \left(\frac{\partial L}{\partial w_i} \right)^2 \quad (130)$$

En cada iteración, el peso w_i se actualiza utilizando la tasa de aprendizaje adaptativa, que se ajusta en función de A_i . La fórmula para actualizar w_i es:

$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} \left(\frac{\partial L}{\partial w_i} \right) \quad (131)$$

El término $\sqrt{A_i}$ en el denominador actúa como una normalización que considera solo la magnitud histórica del gradiente, no su signo. Esto significa que penaliza más los gradientes con alta variabilidad y favorece los gradientes con una dirección consistente. Con el tiempo,

la acumulación de gradientes en A_i causa que las actualizaciones de los pesos se vuelvan más pequeñas y lentas, lo que puede hacer que el entrenamiento se vuelva excesivamente lento o incluso se detenga casi por completo. Para evitar problemas de mal condicionamiento, se puede añadir un pequeño valor positivo ϵ al denominador en la actualización de los pesos:

$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i + \epsilon}} \left(\frac{\partial L}{\partial w_i} \right) \quad (132)$$

Donde ϵ es un valor pequeño como 10^{-8} .

En teoría, AdaGrad es eficiente para problemas de optimización convexa, ya que adapta las tasas de aprendizaje de manera beneficiosa. Sin embargo, en la práctica, al entrenar redes neuronales profundas, el algoritmo puede enfrentar problemas. Aunque AdaGrad puede ser efectivo para algunos modelos de aprendizaje profundo, no siempre es la mejor opción para todos.

RMSProp: El algoritmo RMSProp modifica a AdaGrad para mejorar su desempeño en problemas no convexos. En lugar de simplemente sumar los gradientes cuadrados para estimar A_i , RMSProp usa un promedio exponencial. RMSProp utiliza un factor de decaimiento ρ (donde, $0 < \rho < 1$) para ponderar los gradientes cuadrados que ocurrieron en actualizaciones anteriores. Esto se logra de la siguiente manera (Aggarwal, 2018):

$$A_i \leftarrow \rho A_i + (1 - \rho) \left(\frac{\partial L}{\partial w_i} \right)^2 \quad (133)$$

La actualización del peso w_i se realiza de la misma manera que en AdaGrad (ecuación 131), y, de igual forma, se puede añadir un pequeño valor ϵ en el denominador si se desea evitar problemas de mal condicionamiento (ecuación 132).

RMSProp evita la desaceleración prematura del progreso al no acumular gradientes cuadrados indefinidamente, ya que la importancia de los gradientes antiguos disminuye exponencialmente con el tiempo. En la práctica, RMSProp ha demostrado ser un algoritmo de optimización eficaz y útil para redes neuronales profundas. Hoy en día, es uno de los

métodos de optimización más comúnmente empleados por los profesionales en el campo del aprendizaje profundo.

Adam: Adam es un algoritmo de optimización que combina las ideas de normalización de “señal-ruido” de AdaGrad y RMSProp, pero también introduce suavizado exponencial del gradiente para incorporar momentum en las actualizaciones. Se puede considerar a Adam como una variante que combina RMSProp y momentum .

Al igual que en el caso de RMSProp, el valor A_i se actualiza de la misma manera, con el parámetro de decaimiento ρ (ecuación 133). Al mismo tiempo, se mantiene un valor suavizado exponencialmente del gradiente, cuyo componente i -ésimo se denota como F_i . Este suavizado se realiza con un parámetro de decaimiento diferente, ρ_f (Aggarwal, 2018):

$$F_i \Leftarrow \rho_f F_i + (1 - \rho_f) \left(\frac{\partial L}{\partial w_i} \right)^2 \quad (134)$$

Este suavizado exponencial del gradiente con ρ_f es una variante del método de momentum descrito anteriormente, el cual usa un parámetro de fricción β en lugar de ρ_f . La actualización de parámetros se realiza con la tasa de aprendizaje α_t en la iteración t :

$$w_i \Leftarrow w_i - \frac{\alpha_t}{\sqrt{A_i}} F_i \quad (135)$$

La tasa de aprendizaje α_t ahora varía con el índice de iteración t y se define como:

$$\alpha_t = \alpha \left(\frac{\sqrt{1 - \rho^t}}{1 - \rho_f^t} \right) \quad (136)$$

La tasa de aprendizaje ajustada corrige el sesgo inicial debido a la inicialización de los valores suavizados en cero. Los valores recomendados por defecto para los parámetros de decaimiento son $\rho_f=0.9$ y $\rho=0.999$. Adam usa $\sqrt{A_i} + \epsilon$ en el denominador para mejorar la estabilidad numérica. Este algoritmo es popular por integrar las ventajas de otros métodos y ofrecer un rendimiento competitivo.

2.3.7 REDES NEURONALES CONVOLUCIONALES

Las redes neuronales convolucionales (CNN) están diseñadas para manejar datos organizados en forma de cuadrícula, como las imágenes, que tienen dependencias espaciales en áreas locales. En las imágenes, los píxeles cercanos tienden a tener colores similares, lo que permite a las CNN detectar patrones y características en diferentes partes de la imagen. Aunque el uso principal de las CNN es en el procesamiento de imágenes, también se pueden aplicar a otros tipos de datos secuenciales y espaciales, como textos o series temporales.

La convolución es una operación matemática que combina dos funciones para generar una tercera función. Se aplica sobre funciones que tienen un argumento real y se utiliza para diversos fines, como suavizar datos o extraer características.

Por ejemplo, en el caso de series temporales, para obtener una estimación menos ruidosa sería necesario promediar varias mediciones. Es importante que este promedio sea ponderado y diera más peso a las mediciones recientes. Para dos funciones $x(t)$ y $w(t)$, la convolución se define como (Goodfellow et al., 2016):

$$s(t) = \int x(a)w(t - a) da \quad (137)$$

Donde $x(t)$ es la función de entrada, $w(t)$ es el núcleo o filtro (kernel), $s(t)$ es la función resultante de la convolución; a representa el tiempo o la antigüedad de una medición pasada en relación con el momento actual y se utiliza en la convolución para ajustar la influencia de las mediciones antiguas en la estimación actual, permitiendo que las lecturas más recientes tengan un mayor peso en el resultado final. La convolución se denota comúnmente como:

$$s(t) = (x * w)(t) \quad (138)$$

La salida a veces también se conoce como mapa de características. Volviendo al ejemplo, usualmente, cuando se trabaja con datos en una computadora, el tiempo estará discretizado y los intervalos serán regulares. Asumiendo que x y w están definidos solo en valores enteros de t , se define la convolución discreta:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a) w(t - a) \quad (139)$$

En aplicaciones de aprendizaje automático, la entrada y el núcleo se representan como arrays multidimensionales, llamados tensores. Cada elemento de estos tensores se guarda de forma separada, lo que permite realizar la convolución infinita como una suma finita. Normalmente, la convolución se aplica sobre varios ejes. Por ejemplo, con una imagen bidimensional I y un núcleo bidimensional K , la convolución se calcula así:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n) K(i - m, j - n) \quad (140)$$

La convolución tiene la propiedad de ser conmutativa, lo que permite escribir también:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n) K(m, n) \quad (141)$$

Donde (i, j) es la posición actual en el mapa de características, y (m, n) son los índices del núcleo. El núcleo es una pequeña matriz que se desliza sobre la imagen para realizar la operación de convolución. Generalmente, la segunda fórmula es más sencilla de implementar en las bibliotecas de aprendizaje automático debido a la menor variación en los valores válidos de m y n . Muchas bibliotecas de aprendizaje automático, aunque llaman a la función "convolución", en realidad implementan una función similar conocida como correlación cruzada, que no invierte el núcleo:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i + m, j + n) K(m, n) \quad (142)$$

En las redes neuronales convolucionales, cada capa se organiza en una estructura tridimensional con altura, ancho y profundidad, lo que es fundamental para preservar las relaciones espaciales entre los datos. Estas relaciones espaciales se mantienen de una capa a la siguiente, ya que cada valor en una capa se basa en una pequeña región local de la capa

anterior. La profundidad de una capa se refiere al número de canales o mapas de características en esa capa, como los canales de color en una imagen o las características extraídas en capas ocultas. Esta "profundidad" no debe confundirse con la profundidad de la red en sí misma, que se refiere al número total de capas.

Las redes neuronales convolucionales (CNN) funcionan de manera similar a las redes neuronales tradicionales, pero con capas organizadas espacialmente y conexiones diseñadas de manera específica. Las capas principales en una CNN incluyen (ver figura 4):

- ❖ **Convolución:** Aplica filtros para extraer características locales.
- ❖ **Agrupamiento (Pooling):** Reduce la dimensión espacial de las características.
- ❖ **ReLU:** Aplica una función de activación no lineal.

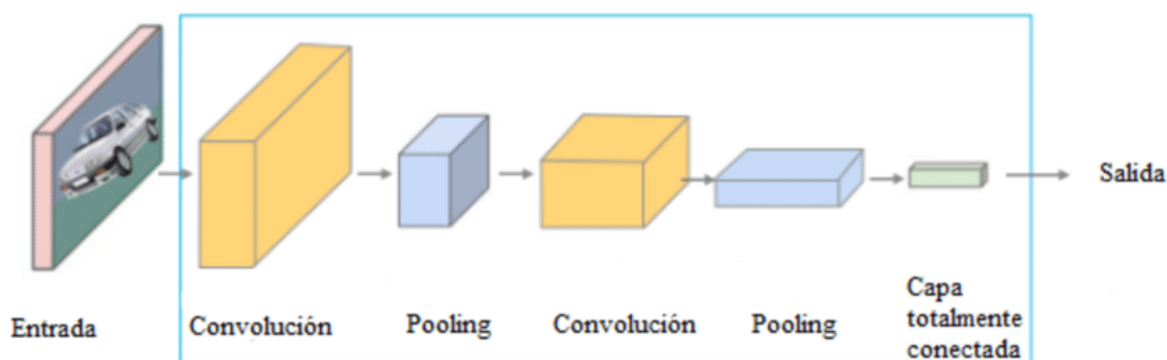


Figura 4 Estructura de una CNN (tomada de Andrade, 2021).

Al final, las capas totalmente conectadas suelen mapear los datos a los nodos de salida, realizando la tarea específica de clasificación o regresión. La clave en las CNN es cómo mantienen y utilizan las relaciones espaciales entre los datos a través de sus operaciones de convolución y agrupamiento.

En una CNN, la profundidad de cada capa es crucial para representar correctamente las imágenes de entrada. Los datos de entrada se organizan en una cuadrícula bidimensional donde cada punto de la cuadrícula, llamado píxel, representa una ubicación en la imagen. Sin embargo, para capturar el color de cada píxel, se necesita una estructura multidimensional. En el esquema RGB, por ejemplo, cada píxel se describe con tres valores de intensidad correspondientes a los colores rojo, verde y azul, lo que da como resultado una profundidad de 3. Si una imagen tiene dimensiones de 32x32 píxeles y una profundidad de 3 (por los tres canales de color), el volumen total de la imagen es $32 \times 32 \times 3$ (Aggarwal, 2018).

En la entrada de una CNN, se representa en una estructura tridimensional donde dos dimensiones representan la disposición espacial (altura y ancho) y la tercera dimensión (profundidad) refleja diferentes propiedades independientes a lo largo de estos canales, como los colores primarios en la capa de entrada. A medida que se avanza a las capas ocultas, estas propiedades corresponden a características más complejas extraídas de regiones locales de la imagen. Así, cada capa q en la red se describe por su tamaño $L_q \times B_q \times d_q$, donde L_q es la altura, B_q es el ancho y d_q es la profundidad.

Entonces, para la capa de entrada en una red neuronal convolucional, las dimensiones se definen por los datos de entrada y su preprocesamiento. Por lo que, dos dimensiones se dedican a la relación espacial y la tercera a las propiedades del color. Por otra parte, en las capas ocultas de la CNN, la organización tridimensional se mantiene, pero las cuadrículas bidimensionales de valores ya no representan píxeles individuales. En lugar de eso, representan mapas de características, y la profundidad d_q en estas capas ocultas suele ser mucho mayor que 3, ya que se extraen numerosas propiedades independientes de las regiones locales para la clasificación.

Los filtros también son unidades estructurales tridimensionales que se aplican a cada capa. Los filtros suelen ser pequeños en términos espaciales pero tienen la misma profundidad que la capa en la que se aplican. Por ejemplo, un filtro en la capa q podría tener dimensiones $F_q \times F_q \times d_q$, donde F_q suele ser pequeño e impar, como 3 ó 5. La profundidad del filtro debe coincidir con la profundidad de la capa en la que se usa.

La operación de convolución en una CNN implica colocar un filtro en cada posible posición sobre la imagen o capa oculta, de manera que el filtro cubra completamente la región correspondiente. El número de posiciones posibles para colocar el filtro define la dimensión espacial de la capa siguiente.

Para una capa q , el filtro se alinea en:

$$L_{q+1} = (L_q - F_q + 1) \quad \text{Posiciones a lo largo de la altura} \quad (143)$$

$$B_{q+1} = (B_q - F_q + 1) \quad \text{Posiciones a lo largo del ancho} \quad (144)$$

Las ecuaciones 143 y 144 determinan el tamaño de la siguiente capa oculta, por ejemplo, si la capa actual es de 32×32 y el filtro es de 5×5 , la nueva capa será de 28×28 . Entonces, la profundidad de la capa siguiente, d_{q+1} , se determina por el número de filtros utilizados en la capa actual. En la figura 5 se ilustra un ejemplo de una convolución entre una entrada de $6 \times 6 \times 1$ y un filtro de $3 \times 3 \times 1$. Se observa como es el proceso de convolución, además de la reducción de tamaño debido al filtro que se aplica.

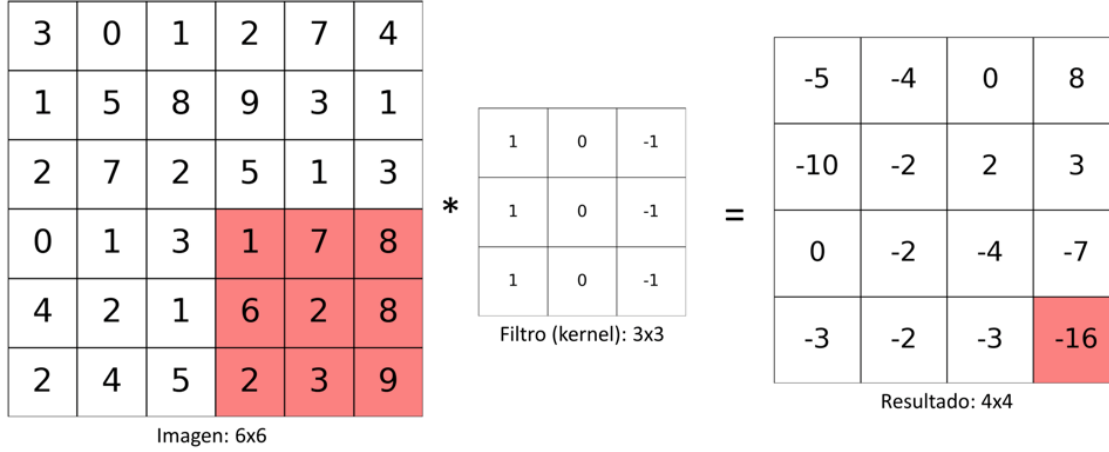


Figura 5 Ejemplo de operación de convolución (Tomada de CodificandoBits, 2019).

El número total de filtros en una capa controla la capacidad del modelo y la profundidad de la capa siguiente, con capas ocultas a menudo teniendo profundidades mucho mayores que la capa de entrada. Los filtros en las capas posteriores tienden a detectar características más complejas, como bordes en diferentes orientaciones, y el uso de múltiples filtros permite capturar una variedad más amplia de patrones espaciales.

La operación de convolución en una red neuronal convolucional se define formalmente utilizando un filtro en la capa q , que está representado por un tensor tridimensional $W^{(p,q)} = [w_{ijk}^{(p,q)}]$. Aquí, los índices i, j y k indican la posición a lo largo de la altura, el ancho y la profundidad del filtro, por lo que $w_{ijk}^{(p,q)}$ representa el valor del parámetro en la posición (i, j, k) del filtro p en la capa q . Los mapas de características en la capa q se representan por el tensor $H^{(q)} = [h_{ijk}^{(q)}]$. Para realizar la convolución y obtener la capa $(q+1)$, se usa la siguiente fórmula (Aggarwal, 2018):

$$h_{ijk}^{(q+1)} = \sum_{r=1}^{F_q} \sum_{s=1}^{F_q} \sum_{k=1}^{d_q} w_{rsk}^{(p,q)} h_{i+r-1,j+s-1,k}^{(q)} \quad (145)$$

Donde i y j varían entre 1 y $(L_q - F_q + 1)$ y $(B_q - F_q + 1)$, respectivamente, y p varía entre 1 y d_{q+1} . Aunque la fórmula puede parecer compleja, la operación de convolución en esencia consiste en aplicar un producto punto del filtro a todas las posiciones válidas en la capa de entrada.

La operación de convolución en las redes CNN tiene una analogía con los experimentos realizados por Hubel y Wiesel, que examinaron cómo se activan las neuronas en regiones específicas del campo visual. En CNN, el campo visual está definido por el filtro que se aplica a diferentes partes de la imagen para detectar formas específicas. Los filtros en las primeras capas suelen identificar características simples, como bordes, mientras que en las capas posteriores combinan estas características simples para detectar formas más complejas. Una característica clave de la convolución es su equivarianza a la traslación. Esto implica que si los píxeles de entrada se mueven, el resultado de la operación de convolución se ajusta de manera correspondiente. Dicho de otra forma, si se desplazan los píxeles de entrada, el resultado de la convolución también se desplaza de igual manera.

Una operación de convolución en la capa q expande el campo receptivo en la siguiente capa $q+1$, lo que significa que cada característica en la capa siguiente captura una región más amplia de la capa de entrada. Por ejemplo, al aplicar sucesivamente filtros de 3×3 en tres capas, las regiones de píxeles capturadas en las capas ocultas son de 3×3 , 5×5 y 7×7 , respectivamente. Dicho de otra forma, aplicar filtros consecutivos de 3×3 en dos capas es funcionalmente equivalente a aplicar un único filtro de 5×5 en una sola capa, porque ambos métodos permiten que cada neurona en la capa final observe una región total de 5×5 píxeles en la imagen original. Sin embargo, la forma en que se obtienen las características puede diferir, ya que la combinación de dos filtros de 3×3 puede extraer características de forma jerárquica y más abstracta en comparación con un único filtro de 5×5 . Esta expansión permite que las capas posteriores detecten características más complejas combinando las características simples de las capas anteriores.

En redes neuronales convolucionales, la función ReLU se aplica a cada uno de los $L_q \times B_q \times d_q$ resultados de cada capa, generando valores que se pasan a la siguiente capa sin alterar las dimensiones de la misma. Esta función sigue a la convolución de manera similar a cómo las funciones de activación siguen a las transformaciones lineales en redes neuronales tradicionales. Aunque anteriormente se utilizaban funciones como la sigmoide y tanh, ReLU se ha vuelto la opción preferida por sus beneficios en rapidez y precisión, facilitando el uso de redes más profundas y entrenamientos más extensos.

Las capas totalmente conectadas en redes neuronales convolucionales conectan cada característica de la capa espacial final a todas las neuronas de la primera capa totalmente conectada. Funcionan de manera similar a las redes de retroalimentación tradicionales y suelen tener un gran número de parámetros, especialmente en redes profundas, lo que puede resultar en altos costos de memoria y cálculo. Como alternativa, se puede emplear pooling promedio en la capa espacial final para reducir el número de parámetros y mejorar la capacidad de generalización del modelo. Este método crea un número menor de características al agregar los valores en la región espacial, evitando la necesidad de capas totalmente conectadas. Para clasificación, la capa de salida está completamente conectada a la penúltima capa y utiliza funciones de activación como softmax.

Padding

La operación de convolución suele reducir las dimensiones de las capas, lo que puede causar pérdida de información en los bordes. Para evitar este problema, se utiliza el padding (o relleno) (Aggarwal, 2018).

Existen varios tipos de Relleno:

- ❖ **Relleno Parcial (Half-Padding):** Añade ceros alrededor del borde del mapa de características. La cantidad de ceros añadidos es $\frac{(F_q-1)}{2}$. Esto mantiene el tamaño espacial del mapa de entrada igual al tamaño del mapa de salida después de la convolución, evitando la pérdida de información en los bordes.
- ❖ **Relleno Válido (Valid Padding):** Cuando no se usa relleno, se denomina relleno válido. Como no se usa relleno, esto puede llevar a que los píxeles en los bordes se

representen de manera insuficiente, resultando en una pérdida de información, especialmente en capas más profundas de la red.

- ❖ **Relleno Completo (Full-Padding):** El relleno completo añade ceros alrededor de la entrada para permitir que el filtro de convolución sobresalga parcialmente de los bordes. Se añaden $(F_q - 1)$ ceros en cada lado. Esto incrementa las dimensiones del mapa de entrada y es útil en autoencoders y durante la retropropagación en redes neuronales.

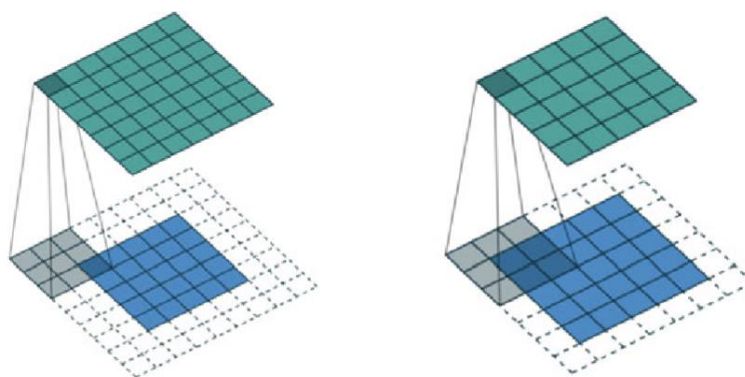


Figura 6 Diferencia entre Half padding y Full-padding (Tomada de Kumar, 2020).

En la figura 6 se representa el relleno completo (izquierda) y el relleno parcial (derecha) al aplicar un filtro de 3x3 sobre una imagen. Se observa como en la figura de la izquierda se aumenta el tamaño de la salida, mientras que en el relleno parcial el tamaño de la salida permanece igual.

Stride

La reducción del tamaño espacial de las imágenes se puede lograr de diferentes maneras. Una de estas maneras es mediante el uso de "strides" o pasos. Con un stride de 1, la convolución se realiza en cada posición del mapa de características, pero al aumentar el stride, la convolución se aplica en intervalos más amplios, reduciendo así el tamaño de la salida y el área de la capa. Aunque es común usar un stride de 1, un stride de 2 también se usa a veces, mientras que strides mayores de 2 son menos comunes (ver figura 7). Usar strides mayores puede ser útil para ahorrar memoria o reducir el sobreajuste. Además, los strides aumentan

el campo receptivo de cada característica, permitiendo captar detalles más complejos en áreas más grandes de la imagen (Aggarwal, 2018).

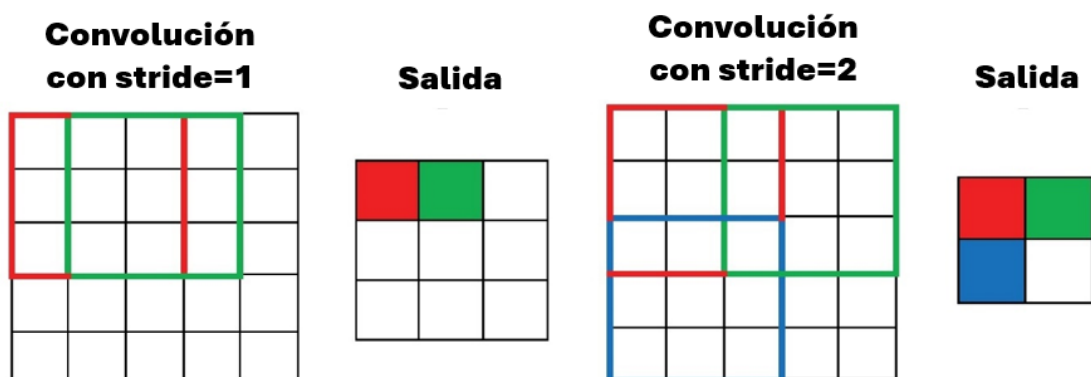


Figura 7 Convolución aplicando distintos valores de paso (Tomada de Team Svitla, 2024).

Por lo general, se acostumbra a utilizar un stride de 1 y, en algunos casos, un stride pequeño de 2. Es preferible trabajar con imágenes cuadradas, y si las imágenes de entrada no lo son, se ajustan recortándolas en parches cuadrados. El número de filtros en cada capa y la profundidad de las capas suelen ser potencias de 2 para optimizar el procesamiento. Los tamaños comunes de los filtros son 3 ó 5. Aunque los filtros más pequeños suelen proporcionar mejores resultados, pueden requerir redes más profundas, lo que añade complejidad.

Pooling

En las CNN, una capa típica pasa por tres etapas: realiza convoluciones, aplica una función de activación no lineal como ReLU, y luego utiliza pooling para resumir la salida. El pooling reemplaza los valores en una ubicación con una estadística resumida de una región, como el valor máximo (max pooling) o el promedio (average pooling). Este proceso ayuda a hacer que las representaciones sean invariantes a pequeñas traslaciones de la entrada, lo que es útil para detectar características sin preocuparse por su ubicación exacta. Las "traslaciones" se refieren a pequeños desplazamientos o movimientos de la imagen de entrada (Aggarwal, 2018).

El pooling reduce la cantidad de datos que la siguiente capa necesita procesar, mejorando la eficiencia computacional y reduciendo la memoria necesaria. Es esencial para manejar entradas de tamaño variable, garantizando que la capa de clasificación reciba un tamaño de entrada fijo. Existen enfoques teóricos y dinámicos para elegir el pooling adecuado, y aunque puede complicar algunos tipos de redes, es fundamental en muchas arquitecturas convolucionales para la clasificación.

Es común usar pooling con filtros de 2×2 y stride de 2 para disminuir el tamaño espacial de los mapas de activación, lo cual también proporciona invariancia a la traslación: el cambio pequeño en la imagen no altera significativamente la activación. Aunque algunas investigaciones recientes sugieren que el pooling podría no ser siempre necesario y que las convoluciones con strides mayores podrían ofrecer un efecto similar en la expansión del campo receptivo, el max-pooling sigue siendo preferido por su capacidad de introducir no linealidad y mayor invariancia a la traslación (Aggarwal, 2018).

Arquitecturas de Redes Neuronales Convolucionales

Las capas de convolución, pooling y ReLU se suelen intercalar en las redes neuronales para potenciar su capacidad expresiva. Generalmente, las capas ReLU siguen a las capas convolucionales, y tras varias combinaciones de convolución-ReLU (CR), se añade una capa de max-pooling (P). Este patrón básico (por ejemplo, CR-CR-P o CR-CR-CR-P) puede repetirse para crear redes profundas (Aggarwal, 2018).

Las redes profundas pueden tener más de 15 capas, y las arquitecturas modernas a menudo incluyen conexiones de salto entre capas, especialmente a medida que aumenta la profundidad de la red. Las conexiones de salto (o skip connections) son una técnica en redes neuronales profundas que facilitan el paso directo de la información entre capas no contiguas, evitando pasar por una o más capas intermedias. Este método ayuda a solucionar problemas frecuentes en redes profundas, como el desvanecimiento del gradiente y la pérdida de rendimiento que ocurre a medida que la red se hace más profunda.

A continuación se explican algunas de las arquitecturas más conocidas o que más han innovado las redes convolucionales (Aggarwal, 2018) :

LeNet-5: Es una de las primeras redes neuronales convolucionales diseñadas para reconocer caracteres en escala de grises. Su estructura incluye dos capas de convolución, dos capas de pooling y tres capas completamente conectadas. En LeNet-5, se usa la función de activación sigmoid y un average pooling que promedia los valores en regiones de 2×2 . En la capa final, LeNet-5 utiliza unidades de función de base radial (RBF). Aunque LeNet-5 es bastante simple comparado con los estándares actuales, estableció una base importante para el reconocimiento de patrones y sigue siendo relevante para entender la evolución de las redes neuronales.

AlexNet: AlexNet, el ganador de la competencia ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 2012, se destaca por su uso innovador de redes neuronales convolucionales. Utilizó dos GPUs para manejar la complejidad del entrenamiento, dividiendo el trabajo entre ellas para acelerar el proceso. AlexNet procesa imágenes de $224 \times 224 \times 3$ píxeles utilizando 96 filtros de tamaño $11 \times 11 \times 3$ en la primera capa, con un stride de 4. Esto produce una capa de salida de $55 \times 55 \times 96$. Después de esta capa, se aplica una capa de max-pooling. La red utiliza ReLU como función de activación después de cada capa de convolución, seguida de normalización y max-pooling. En la segunda capa convolucional se usan 256 filtros de tamaño $5 \times 5 \times 96$, mientras que las capas siguientes usan filtros de tamaño $3 \times 3 \times 256$, $3 \times 3 \times 384$ y $3 \times 3 \times 384$, con 384 y 256 filtros respectivamente. Las capas de max-pooling utilizan filtros de 3×3 con un stride de 2. Las capas completamente conectadas tienen 4096 neuronas cada una, y la capa final usa softmax para la clasificación en 1000 categorías. La penúltima capa, con 4096 activaciones, se usa a menudo para representar imágenes de forma plana en aplicaciones más allá de la clasificación, conocidas como características FC7. AlexNet introdujo varias innovaciones importantes en el campo de redes neuronales: a diferencia de funciones de activación como sigmoid o tanh, AlexNet usó ReLU, que se convirtió en un estándar en arquitecturas posteriores; aplicó técnicas de aumentación de datos (que consiste en aumentar el conjunto de entrenamiento creando copias modificadas de un conjunto de datos utilizando los datos existentes, realizando pequeños cambios al conjunto de datos existente); se utilizó Dropout junto con L2-regularización para evitar el sobreajuste y mejorar la generalización; además de emplear GPUs para manejar grandes volúmenes de datos de entrenamiento, acelerando el proceso. Estas innovaciones resultaron en una tasa de

error significativamente baja, reavivando el interés en las redes neuronales convolucionales y el aprendizaje profundo.

ZFNet: Es una variante mejorada de AlexNet. Aunque se basa en la arquitectura de AlexNet, ZFNet introduce cambios en los hiperparámetros, como reducir el tamaño de los filtros de la capa inicial de 11×11 a 7×7 y cambiar los pasos de 4 a 2. También se ajustaron las capas de max-pooling, el primer par de capas de max-pooling se realizó después de las primeras y segundas convoluciones (en lugar de después de la segunda y tercera convolución); y se aumentó el número de filtros en las capas de convolución. El número de filtros en estas capas cambió de (384, 384, 256) a (512, 1024, 512). Los pequeños cambios en la arquitectura demostraron ser significativos para el rendimiento, destacando la importancia de la experimentación detallada en el diseño de redes neuronales.

VGG: VGG destacó por profundizar aún más en la tendencia de aumentar la profundidad de las redes. Las redes probadas tenían configuraciones con entre 11 y 19 capas, aunque las versiones que mejor desempeño tuvieron tenían 16 o más capas. Una innovación clave de VGG fue la reducción del tamaño de los filtros mientras se aumentaba la profundidad. Como ya se ha mencionado, un filtro de 3×3 aplicado tres veces secuencialmente puede capturar una región de 7×7 , mientras que un solo filtro de 7×7 captura directamente una región de 7×7 . Usar filtros más pequeños y más capas permite captar características más complejas que un único filtro grande. Esto permitió capturar características más complejas con un menor número de parámetros, ya que un aumento en la profundidad proporciona más no linealidad y regularización. VGG utiliza filtros de 3×3 y max-pooling de 2×2 . La convolución se realiza con un stride de 1 y padding de 1, manteniendo la huella espacial de la salida. El max-pooling reduce la huella espacial en un factor de 2, mientras que la cantidad de filtros se duplica después de cada max-pooling. Esto equilibra el esfuerzo computacional en las capas.

GoogLeNet: GoogLeNet introdujo un concepto innovador conocido como la arquitectura de inception. La parte inicial de la arquitectura es similar a una red convolucional tradicional y se denomina el stem. La clave de la red es una capa intermedia, llamada módulo de inception. La arquitectura inception utiliza módulos de inception para capturar información en diferentes niveles de detalle a través de filtros de distintos tamaños (1×1 , 3×3 y 5×5) en paralelo. Esta arquitectura permite al modelo manejar diferentes escalas de objetos

de manera flexible y eficiente. Un problema con el módulo de inception es la ineficiencia computacional debido al gran número de convoluciones de diferentes tamaños. Para abordar esto, se utiliza una implementación eficiente que primero reduce la profundidad del mapa de características mediante convoluciones 1×1 , conocidas como operaciones de bottleneck. En lugar de usar capas completamente conectadas, GoogLeNet emplea average pooling para reducir la cantidad de parámetros, lo que la hace más compacta en comparación con arquitecturas como VGG.

ResNet: ResNet, con 152 capas, revolucionó las redes neuronales al superar las limitaciones de profundidad con un innovador enfoque de conexiones de salto. La innovación clave de ResNet es la incorporación de estas conexiones de salto, que permiten un flujo de gradientes más fluido y facilitan el entrenamiento de redes profundas. La mayoría de las redes feed-forward solo contienen conexiones entre las capas i e $(i + 1)$, mientras que ResNet contiene conexiones entre las capas i e $(i + r)$ para $r > 1$. Esta conexión de salto simplemente copia la entrada de la capa i y la suma a la salida de la capa $(i+r)$. Este enfoque permite un flujo de gradientes efectivo porque el algoritmo de retropropagación ahora tiene una supercarretera para propagar los gradientes hacia atrás utilizando las conexiones de salto. Esta unidad básica se denomina módulo residual, y la red completa se crea uniendo muchos de estos módulos básicos. Las conexiones de salto en redes neuronales permiten un flujo de gradientes más fluido, facilitando la retropropagación al proporcionar múltiples rutas de diferentes longitudes desde la entrada hasta la salida. Esto optimiza el aprendizaje al permitir que los caminos más cortos se encarguen de la clasificación básica, mientras que los caminos más largos afinan detalles más complejos. Este proceso, conocido como aprendizaje residual, ajusta el aprendizaje más fácil de los caminos cortos con el ajuste fino adicional de los caminos largos. A pesar de que ResNet puede ser muy profunda, el uso de conexiones de salto asegura que los gradientes puedan fluir sin obstáculos, haciendo que la red se comporte de manera similar a un conjunto de redes más superficiales.

2.3.8 REDES RECURRENTES

Las redes neuronales recurrentes (RNNs) son una clase de redes neuronales diseñadas para procesar secuencias de datos, como series temporales o textos. A diferencia de las redes

convolucionales, que se centran en analizar imágenes, las RNNs son ideales para manejar secuencias que pueden variar en longitud. Esto les permite manejar secuencias muy largas y adaptarse a diferentes tamaños de secuencia, algo que no sería práctico para redes sin esta especialización.

Las redes neuronales recurrentes son adecuadas para el procesamiento de datos secuenciales, como el texto, y se puede usar una RNN a nivel de palabras o a nivel de caracteres.

A diferencia de las redes neuronales feedforward, donde la información se mueve en una sola dirección desde la entrada hasta la salida, en una RNN la información puede circular hacia atrás desde la capa oculta de un paso de tiempo al siguiente (Aggarwal, 2018).

En la figura 8 se muestra la RNN más simple, que consiste en un lazo auto-recurrente que actualiza el estado oculto después de cada entrada. Esta red se puede desplegar en una estructura de red feed-forward "por capas de tiempo", como se muestra en la figura 9. Esta representación es equivalente a la original pero más fácil de entender, ya que se asemeja a una red tradicional. Las matrices de peso W_{xh} (entrada-oculta), W_{hh} (oculta-oculta) y W_{hy} (oculta-salida) se comparten entre diferentes puntos temporales para asegurar que la misma función se use en cada punto temporal.

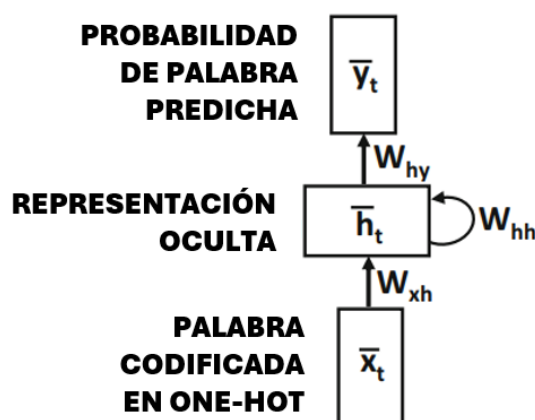


Figura 8 RNN (tomado de Aggarwal, 2018).

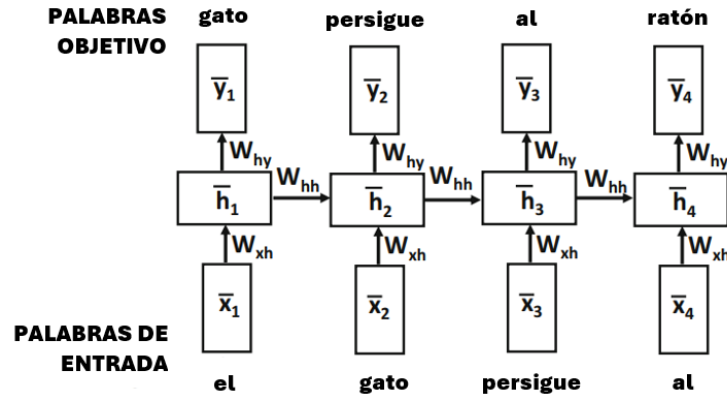


Figura 9 Representación en capas de tiempo de una RNN simple (tomado de Aggarwal, 2018).

Aunque en las figuras 8 y 9 se muestra una red neuronal recurrente donde cada punto temporal tiene una entrada, una salida y una unidad oculta; en la práctica, puede haber casos en los que falten entradas o salidas en algunos puntos temporales, dependiendo de la aplicación.

Las figuras 8 y 9 ilustran una red neuronal recurrente adecuada para el modelado del lenguaje, donde se predice la siguiente palabra basándose en el historial de palabras previas. En este modelo, cada palabra en una secuencia se representa mediante un vector one-hot y se alimenta a la red neuronal en cada punto temporal. Cada punto temporal corresponde a una posición en la secuencia, comenzando desde 0 y aumentando de uno en uno. Entonces, la red produce un vector de probabilidades para la próxima palabra en la secuencia. Esta es la definición clásica de un modelo de lenguaje, donde la probabilidad de una palabra se calcula con base en el historial reciente de palabras (Aggarwal, 2018).

En general, para una red recurrente en el modelado del lenguaje, $\bar{x}_t$ es la entrada en el tiempo t (vector one-hot de la t -ésima palabra), $\bar{h}_t$ representa el estado oculto en el tiempo t y $\bar{y}_t$ es la salida en el tiempo t (vector de probabilidades para la siguiente palabra). En el modelo de red neuronal recurrente, el vector de entrada $\bar{x}_t$ y el vector de salida $\bar{y}_t$ tienen una dimensión d , que corresponde al tamaño del léxico, mientras que el estado oculto $\bar{h}_t$ tiene una dimensión p , que determina la complejidad de la representación.

El estado oculto en el tiempo t se calcula mediante la función del vector de entrada en el tiempo t y el estado oculto en el tiempo $(t-1)$ (Aggarwal, 2018):

$$\bar{h}_t = \phi(\bar{h}_{t-1}, \bar{x}_t) \quad (146)$$

Esta función se define utilizando matrices de pesos y funciones de activación (como las usadas por todas las redes neuronales para el aprendizaje), y los mismos pesos se usan en cada punto temporal. Por lo tanto, aunque el estado oculto evoluciona con el tiempo, los pesos y la función subyacente $\phi(\cdot, \cdot)$ permanecen fijos en todos los puntos temporales (es decir, elementos secuenciales) después de que la red neuronal ha sido entrenada. Por otra parte, se usa una función separada para aprender las probabilidades de salida a partir de los estados ocultos:

$$\bar{y}_t = g(\bar{h}_t) \quad (147)$$

Definiendo una matriz de entrada-oculta W_{xh} de $p \times d$, una matriz oculta-oculta W_{hh} de $p \times p$, y una matriz oculta-salida W_{hy} de $d \times p$, las ecuaciones 146 y 147 quedan de la siguiente manera:

$$\bar{h}_t = \phi(W_{xh}\bar{x}_t + W_{hh}\bar{h}_{t-1}) \quad (148)$$

$$\bar{y}_t = W_{hy} \bar{h}_t \quad (149)$$

La función ϕ puede ser, por ejemplo, una función de activación como la tangente hiperbólica ($\tanh$) o la sigmoide. Esta función de activación se aplica de elemento a elemento al vector columna de dimensión p , generando un vector de dimensión p con valores en el rango de $[-1,1]$. Al inicio de una secuencia, el estado oculto $\bar{h}_{t-1}$ se considera un vector constante (como 0), ya que no hay información previa. La salida $\bar{y}_t$ se ajusta mediante una capa softmax para convertir los valores continuos en probabilidades (Aggarwal, 2018).

Para entrenar redes neuronales recurrentes, se utiliza la retropropagación a través del tiempo (BPTT). En este proceso la pérdida se calcula a partir del logaritmo negativo de la probabilidad softmax de las palabras correctas en cada momento de tiempo. La función softmax convierte el vector de salidas en probabilidades:

$$[\hat{p}_t^1 \dots \hat{p}_t^d] = \text{Softmax}([\hat{y}_t^1 \dots \hat{y}_t^d]) \quad (150)$$

Entonces, la función de pérdida L para todos los T momentos es:

$$L = - \sum_{t=1}^T \log(\hat{p}_t^{j_t}) \quad (151)$$

Donde j_t es el índice de la palabra correcta en el tiempo t . La derivada de la pérdida con respecto a las salidas es:

$$\frac{\partial L}{\partial \hat{y}_t^k} = \hat{p}_t^k - I(k, j_t) \quad (152)$$

Donde, $I(k, j_t)$ es 1 si $k=j_t$ y 0 en caso contrario.

Para calcular los gradientes de los pesos en diferentes capas temporales, primero se asume que los pesos en cada instante de tiempo son independientes y se realiza la retropropagación convencional. Lo que quiere decir que es necesario fingir que los pesos $W_{xh}^{(t)}$, $W_{hh}^{(t)}$ y $W_{hy}^{(t)}$ son distintos para cada instante t . De esta manera es posible utilizar la retropropagación convencional para calcular $\frac{\partial L}{\partial W_{xh}^{(t)}}$, $\frac{\partial L}{\partial W_{hh}^{(t)}}$ y $\frac{\partial L}{\partial W_{hy}^{(t)}}$. Finalmente, se suman los gradientes correspondientes a cada peso compartido a lo largo de todos los momentos temporales:

$$\frac{\partial L}{\partial W_{xh}} = \sum_{t=1}^T \frac{\partial L}{\partial W_{xh}^{(t)}} \quad (153)$$

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^T \frac{\partial L}{\partial W_{hh}^{(t)}} \quad (154)$$

$$\frac{\partial L}{\partial W_{hy}} = \sum_{t=1}^T \frac{\partial L}{\partial W_{hy}^{(t)}} \quad (155)$$

Las RNN tradicionales solo consideran la información pasada para hacer predicciones, lo que puede ser una limitación en ciertas aplicaciones como el reconocimiento de escritura o el modelado del lenguaje. Para mejorar esto, se utilizan redes recurrentes bidireccionales (BRNNs).

Redes recurrentes bidireccionales

Las BRNNs fueron desarrolladas para mejorar el procesamiento de secuencias al considerar tanto el pasado como el futuro en la predicción. A diferencia de las RNNs causales, que solo tienen en cuenta la información pasada y presente, las BRNNs combinan dos RNNs: una que avanza hacia adelante en el tiempo y otra que avanza hacia atrás. En la red recurrente bidireccional, se tienen estados ocultos separados $\vec{h}_t^f$ y $\vec{h}_t^b$ para las direcciones hacia adelante y hacia atrás. Los estados ocultos hacia adelante interactúan solo entre ellos y lo mismo ocurre con los estados ocultos hacia atrás. Ambos, tanto $\vec{h}_t^f$ como $\vec{h}_t^b$, reciben entrada del mismo vector x_t y ambos interactúan con el mismo vector de salida $\hat{y}_t$ (ver figura 10).

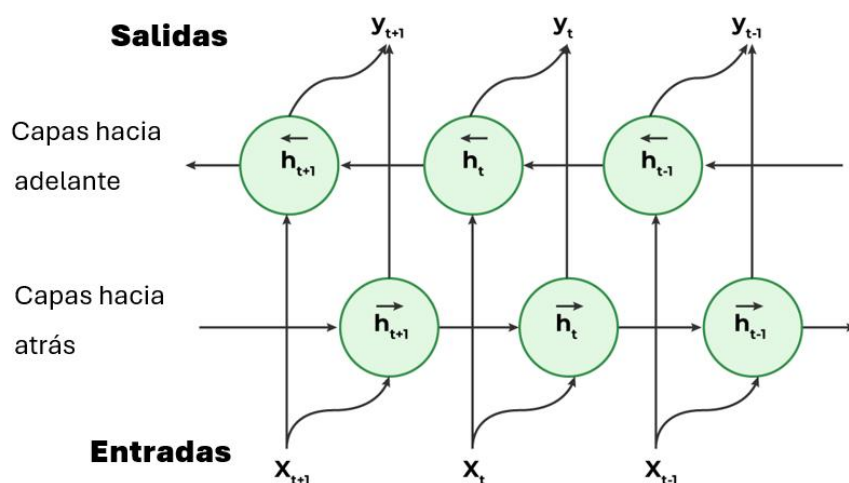


Figura 10 Ejemplo de Red Neuronal Recurrente Bidireccional (Tomada de GeeksforGeeks 2023).

De igual manera, en el caso de la red bidireccional, se tienen matrices de parámetros separadas para la dirección hacia adelante y hacia atrás. Las matrices hacia adelante se

denotan como $W_{xh}^{(f)}$, $W_{hh}^{(f)}$ y $W_{hy}^{(f)}$. Las matrices hacia atrás para las interacciones se denotan por $W_{xh}^{(b)}$, $W_{hh}^{(b)}$ y $W_{hy}^{(b)}$. Entonces, las ecuaciones 148 y 149 se pueden escribir de la siguiente manera (Aggarwal, 2018):

$$h_t^{(f)} = \phi\left(W_{xh}^{(f)} \bar{x}_t + W_{hh}^{(f)} \bar{h}_{t-1}^{(f)}\right) \quad (156)$$

$$h_t^{(b)} = \phi\left(W_{xh}^{(b)} \bar{x}_t + W_{hh}^{(b)} \bar{h}_{t-1}^{(b)}\right) \quad (157)$$

$$\bar{y}_t = W_{hy}^{(f)} \bar{h}_t^{(f)} + W_{hy}^{(b)} \bar{h}_t^{(b)} \quad (158)$$

Las BRNNs son útiles para tareas donde el contexto completo (pasado y futuro) es importante, como el reconocimiento de voz y escritura a mano, donde la interpretación de los datos actuales puede depender de contexto tanto anterior como futuro. Las RNNs bidireccionales también pueden extenderse a datos en 2D, como imágenes, utilizando múltiples RNNs en diferentes direcciones.

Redes recurrentes Multicapa

En la práctica, se utiliza una arquitectura de RNN multicapa para modelos más complejos. En una red recurrente multicapa, los nodos en capas superiores reciben entradas de las capas inferiores. La ecuación de recurrencia para una red multicapa se generaliza a partir de la red de una sola capa (ecuación 148) de la siguiente manera:

$$\bar{h}_t = \phi\left(W_{xh} \bar{x}_t + W_{hh} \bar{h}_{t-1}\right) = \phi W \begin{bmatrix} \bar{x}_t \\ \bar{h}_{t-1} \end{bmatrix} \quad (159)$$

Donde W agrupa las columnas W_{xh} y W_{hh} , $W = [W_{xh}, W_{hh}]$. Para distinguir entre los nodos ocultos en las capas superiores, se agrega un superíndice adicional al estado oculto y se

denota el vector para los estados ocultos en el tiempo t y la capa k por $h_t^{(k)}$. De manera similar, la matriz de pesos para la k -ésima capa oculta se denota por $W^{(k)}$.

Para la primera capa oculta, las matrices $W^{(k)}$ tendrán un tamaño de $p \times (d+p)$ (es decir, cuando $k=1$), donde d es el tamaño del vector de entrada $\bar{x}_t$ y p es el tamaño del vector oculto $\bar{h}_t$.

Para capas ocultas superiores ($k \geq 2$), la ecuación de recurrencia es:

$$\bar{h}_t^{(k)} = \phi W^{(k)} \begin{bmatrix} \bar{h}_t^{(k-1)} \\ \bar{h}_{t-1}^{(k)} \end{bmatrix} \quad (160)$$

El tamaño de $W^{(k)}$ es $p \times 2p$. La transformación de la capa oculta a la capa de salida sigue siendo la misma que en las redes de una sola capa. Comúnmente, se utilizan dos o tres capas, y para redes con más capas, es necesario disponer de suficientes datos de entrenamiento para evitar el sobreajuste.

Las redes neuronales recurrentes son muy difíciles de entrenar debido al hecho de que la red con capas temporales es una red muy profunda, especialmente si la secuencia de entrada es larga. Esto resulta en gradientes de pérdida altamente variables entre las capas temporales, complicados por el hecho de que los parámetros se comparten a través de estas capas. Los principales problemas en las RNNs es el desvanecimiento y la explosión de gradientes. Considerando una RNN con T capas, con la función de activación $\tanh$ aplicada entre capas. Siendo $\frac{\partial L}{\partial h_t}$ la derivada de la función de pérdida con respecto a la activación oculta h_t , entonces, se deriva la siguiente ecuación de actualización usando retropropagación:

$$\frac{\partial L}{\partial h_t} = \phi'(h_{t+1}) \cdot w_{t+1} \cdot \frac{\partial L}{\partial h_{t+1}} \quad (161)$$

Si el peso compartido w es menor que 1, la multiplicación repetida por w hace que el gradiente se reduzca exponencialmente a medida que se retropropaga a través de las capas. Si w es

mayor que 1, el gradiente puede aumentar exponencialmente, provocando valores descontroladamente grandes. La elección de la función de activación también afecta el problema. Por ejemplo, para la función tanh, cuya derivada ϕ' es generalmente menor que 1, se incrementa la probabilidad de que los gradientes se desvanezcan.

Aunque la discusión anterior solo estudia el caso simple de una capa oculta con una unidad, se puede generalizar el argumento a una capa oculta con múltiples unidades. El comportamiento de los gradientes en las RNNs está influenciado por la multiplicación repetida de la matriz de pesos W . Suponiendo una red neuronal recurrente muy simple que carece de una función de activación no lineal y, además, carece de entradas x , entonces la relación de recurrencia básica de una RNN puede expresarse como:

$$h^{(t)} = W^T h^{(t-1)} \quad (162)$$

Y, si es posible diagonalizar W de la forma:

$$W = Q\Delta Q^T \quad (163)$$

donde Q es una matriz de transformación y Δ es una matriz diagonal con los valores propios (eigenvalores) de W . Entonces la relación de recurrencia puede simplificarse a:

$$h^{(t)} = Q^T \Delta^t Q h^{(0)} \quad (164)$$

Los valores propios se elevan a la potencia t , por lo que si el mayor valor propio λ de W es menor que 1, entonces las entradas de W^t tienden a 0 a medida que t aumenta y, si el mayor valor propio λ es mayor que 1, entonces las entradas de W^t divergen a valores grandes, lo que significa que los gradientes explotan, creciendo exponencialmente.

En el entrenamiento de redes neuronales recurrentes (RNN), los problemas más destacados son la inestabilidad del tamaño del paso de gradiente y los gradientes que se desvanecen o

explotan. Si el tamaño del paso es demasiado pequeño, el progreso es lento, mientras que un tamaño demasiado grande puede causar inestabilidad. Los gradientes pueden desvanecerse, disminuyendo su impacto en las capas anteriores, o explotar, creciendo desmesuradamente. Para mitigar estos problemas, se utilizan diversas técnicas:

Regularización: La regularización puede reducir la inestabilidad introduciendo penalizaciones en el modelo para que no se ajuste demasiado a los datos de entrenamiento, aunque un exceso puede limitar el rendimiento del modelo ya que hacer que el modelo sea demasiado restrictivo, limitando su capacidad para aprender patrones complejos y reduciendo su rendimiento general.

Limitación de Gradientes: La limitación de gradientes ayuda a manejar los gradientes explosivos. Hay dos tipos de limitación que se usan comúnmente. Primero, la limitación basada en valor, donde los componentes del gradiente que superan un cierto umbral se recortan o se limitan a ese umbral. Y el segundo, la limitación basada en norma, en el cual cuando el vector de gradiente completo tiene una norma que aumenta más allá de un umbral, se escala para que su norma sea igual al umbral.

Buena Inicialización: Elegir buenos puntos de inicialización ayuda a evitar la inestabilidad en la actualización de gradientes.

Métodos de Momentum: Ayudan a suavizar las actualizaciones y pueden reducir la inestabilidad.

Normalización: En el caso de las RNN se utilizan dos tipos. La normalización por Lotes, que es una técnica que normaliza las activaciones en cada mini-lote de datos para que tengan una media cercana a 0 y una desviación estándar cercana a 1. Y la normalización por Capas, que es similar a la normalización por lotes, pero normaliza las activaciones a lo largo de las dimensiones de la capa en lugar de a través del mini-lote. Esto es especialmente útil en redes recurrentes, donde el tamaño del lote puede ser variable y la normalización por lotes no siempre es aplicable.

Variantes de RNN: Finalmente, algunas variantes como modelos con memoria a largo plazo (LSTM) proporcionan mejoras significativas en la estabilidad y el rendimiento

del entrenamiento, siendo LSTM especialmente eficaz en el manejo de problemas de gradientes.

LSTM (Long Short-Term Memory)

Como ya se mencionó, las redes neuronales recurrentes enfrentan problemas de gradientes que se desvanecen o explotan debido a la inestabilidad de la multiplicación sucesiva por matrices de pesos, lo que limita su capacidad para aprender secuencias largas. Las LSTM abordan estos problemas al introducir un mecanismo de memoria a largo plazo que permite un control más preciso y estable sobre la información almacenada y actualizada, mejorando así la capacidad de la red para manejar secuencias largas.

En una LSTM de múltiples capas, los estados ocultos $\bar{h}_t^{(k)}$ de la k-ésima capa se actualizan con la adición de un estado de celda $\bar{c}_t^{(k)}$, que actúa como una memoria a largo plazo. Este estado de celda retiene información a lo largo del tiempo mediante operaciones de olvido y adición (ver figura 11). Esto permite a la LSTM modelar dependencias a largo plazo, como patrones en datos textuales. La persistencia en la memoria evita problemas de gradientes que se desvanecen o explotan, ya que los estados en diferentes capas temporales son más similares, lo que estabiliza el aprendizaje (Aggarwal, 2018).

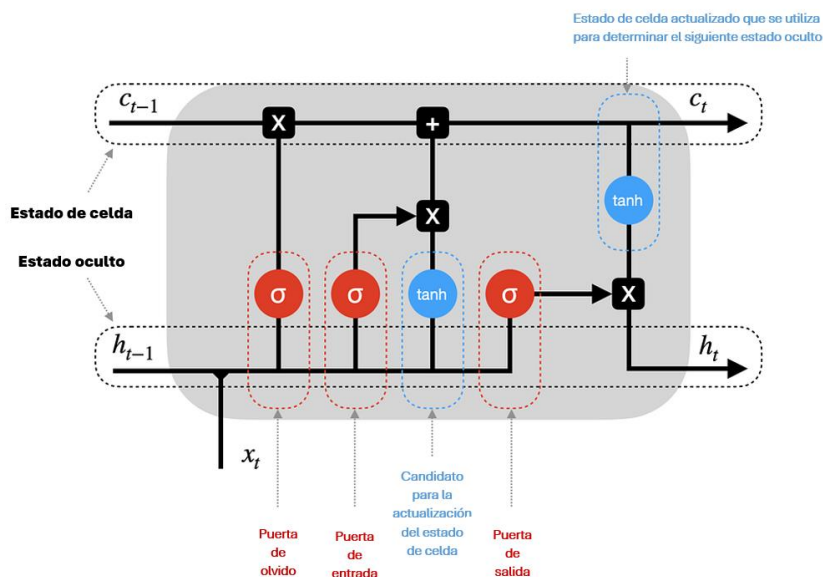


Figura 11 LSTM (Tomada de Dobilas, 2024a).

Al igual que en la red recurrente multicapa, la matriz de actualización se denota por $W^{(k)}$ y se utiliza para multiplicar por delante el vector columna $\begin{bmatrix} \bar{h}_t^{(k-1)} \\ \bar{h}_{t-1}^{(k)} \end{bmatrix}$. Sin embargo, esta matriz es de tamaño $4p \times 2p$, y por lo tanto, la multiplicación de un vector de tamaño $2p$ con $W^{(k)}$ resulta en un vector de tamaño $4p$. Este proceso genera cuatro variables intermedias de tamaño p : i (puerta de entrada), f (puerta de olvido), o (puerta de salida), y c (nuevo contenido o candidato para el estado de celda). Estas variables intermedias se utilizan en un proceso de actualización en múltiples pasos para determinar el estado oculto $\bar{h}_t^{(k)}$ y el estado de celda $\bar{c}_t^{(k)}$. La fórmula para el cálculo de las variables intermedias es la siguiente:

$$\begin{bmatrix} i \\ f \\ o \\ c \end{bmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W^{(k)} \begin{bmatrix} \bar{h}_t^{(k-1)} \\ \bar{h}_{t-1}^{(k)} \end{bmatrix} \quad (165)$$

El estado de celda se actualiza con:

$$\bar{c}_t^{(k)} = f \odot \bar{c}_{t-1}^{(k)} + i \odot c \quad (166)$$

Y el estado oculto se calcula como:

$$\bar{h}_t^{(k)} = o \odot \tanh(\bar{c}_t^{(k)}) \quad (167)$$

Aquí, el producto elemento a elemento de los vectores se denota por $\odot$.

Estos vectores actúan como puertas para decidir cómo se actualiza el estado de celda y el estado oculto:

- ❖ **Puerta de entrada (i):** Decide cuánto del nuevo contenido c se añade al estado de celda.
- ❖ **Puerta de olvido (f):** Determina cuánto del estado de celda anterior se olvida.

❖ **Puerta de salida (o):** Controla cuánto del estado de celda se pasa al estado oculto.

Estas puertas se comportan como puertas booleanas, aunque estos vectores contienen valores continuos entre 0 y 1, en la práctica, simulan un comportamiento binario similar a puertas lógicas AND en una red neuronal. Esto permite un control preciso sobre cómo se actualiza el estado de celda, ayudando a mantener la memoria a largo plazo y asegurando la diferenciabilidad necesaria para el entrenamiento con gradientes.

Los cuatro vectores intermedios i , f , o y c se configuran usando las matrices de pesos $W^{(k)}$ para la k -ésima capa. Analizando la ecuación 166, se observa que esta se divide en dos partes de la siguiente manera:

$$\bar{c}_t^{(k)} = \underbrace{f \odot \bar{c}_{t-1}^{(k)}}_{\text{Reinicio}} + \underbrace{i \odot c}_{\text{Incremento}} \quad (168)$$

La primera parte utiliza los p bits de olvido en f para decidir cuáles de los p estados de celda del tiempo anterior se reinician a 0, y utiliza los p bits de entrada en i para decidir si se añaden los componentes correspondientes de c a cada uno de los estados de celda. El estado de celda se actualiza de manera aditiva, lo que ayuda a prevenir el problema del gradiente que se desvanece. El estado de celda actúa como una memoria a largo plazo que se actualiza continuamente, permitiendo retener información pasada y agregar nueva información. Los valores en c están en el rango de $[-1, +1]$ porque provienen de la función tangente hiperbólica.

Por otra parte, los estados ocultos $\bar{h}_t^{(k)}$ se actualizan utilizando las fugas del estado de la celda. El estado oculto se actualiza con la fórmula 167, donde se copia una forma funcional de cada uno de los p estados de la celda a cada uno de los p estados ocultos, dependiendo de si el gate de salida (definido por o) es 0 o 1. Cabe destacar que la ecuación final no siempre utiliza la función de activación tanh, por lo que existe la versión alternativa:

$$\bar{h}_t^{(k)} = o \odot \bar{c}_t^{(k)} \quad (169)$$

Unidades Recurrentes con Puertas (GRU, Gated Recurrent Units)

Las GRU son una versión simplificada de las LSTM y no utilizan estados de celda explícitos. En lugar de usar puertas de olvido y salida como las LSTM, las GRU emplean una sola puerta de reinicio y una puerta de actualización para gestionar la información del estado oculto.

En las GRU se utilizan las matrices $W^{(k)}$ y $V^{(k)}$. El vector $W^{(k)}$ se usa para calcular las puertas de actualización (z_t) y de reinicio (r_t). Estas dos puertas se calculan para decidir cuánto cambia el vector oculto $\bar{h}_t^{(k)}$ con la matriz de pesos $V^{(k)}$ (ver figura 12) (Aggarwal, 2018).

$$\begin{bmatrix} z \\ r \end{bmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \end{pmatrix} W^{(k)} \begin{bmatrix} \bar{h}_t^{(k-1)} \\ \bar{h}_{t-1}^{(k)} \end{bmatrix} \quad (170)$$

$$\bar{h}_t^{(k)} = z \odot \bar{h}_{t-1}^{(k)} + (1 - z) \odot \tanh V^{(k)} \begin{bmatrix} \bar{h}_t^{(k-1)} \\ r \odot \bar{h}_{t-1}^{(k)} \end{bmatrix} \quad (171)$$

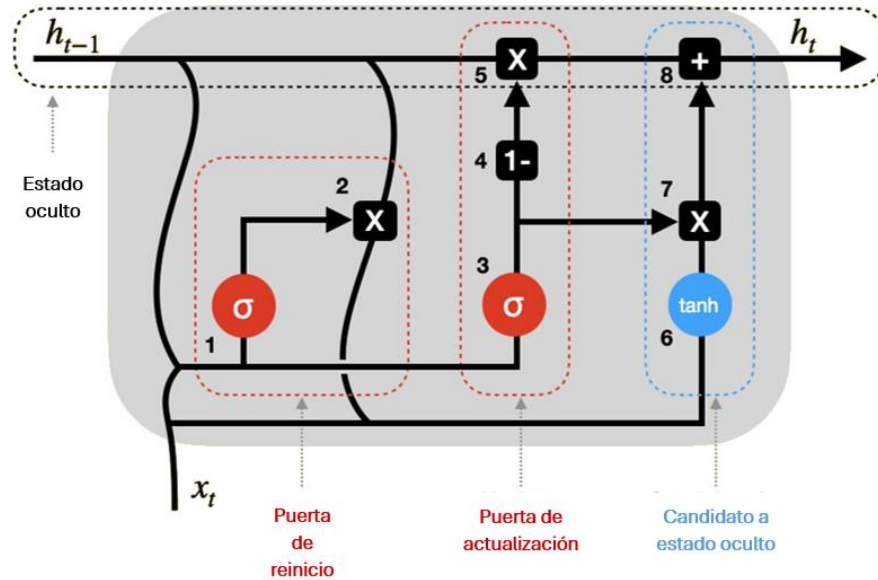


Figura 12 GRU (Tomado de Dobilas, 2024b).

Las GRU utilizan puertas de actualización y de reinicio para gestionar la información del estado oculto, a diferencia de las LSTM, que emplean puertas de entrada, salida y olvido. La puerta de reinicio determina cuánto del estado oculto anterior se conserva, mientras que la puerta de actualización decide la mezcla entre la nueva información y el estado oculto previo. Esto permite una copia parcial de los estados ocultos, mejorando la estabilidad del flujo de gradientes durante la retropropagación. La puerta de actualización de la GRU combina las funciones de las puertas de entrada y olvido de la LSTM, pero no hay un estado de celda separado en las GRU.

La GRU es una simplificación de la LSTM, pero no debe considerarse un caso especial. Ambas arquitecturas son similares en rendimiento, aunque la GRU es más simple y eficiente, y puede generalizar mejor con menos datos debido a su menor cantidad de parámetros. Sin embargo, la LSTM suele ser preferible con grandes conjuntos de datos y ha sido más ampliamente probada, lo que la convierte en una opción más segura para secuencias largas. La LSTM tiene una memoria interna explícita y un control más detallado sobre las actualizaciones, lo que contribuye a su efectividad.

CAPÍTULO 3. CASO DE ESTUDIO

Esta sección tiene como objetivo seleccionar los algoritmos que muestren los mejores resultados según métricas de evaluación específicas. Para esto, se evaluará cómo diferentes algoritmos de Machine Learning se desempeñan al aplicarse a un conjunto de datos que simula el comportamiento de un sistema industrial. Estos algoritmos podrían utilizarse más adelante para clasificar fallas en sistemas reales.

El conjunto de datos utilizado en este trabajo fue proporcionado por Shivam Bansal en la plataforma kaggle (Shivam Bansal, 2021). Note que, debido a que los conjuntos de datos de mantenimiento predictivo reales son difíciles de obtener y publicar, este conjunto de datos es un conjunto de datos sintético que refleja el mantenimiento predictivo real encontrado en la industria. El conjunto de datos consiste en 10000 datos almacenados en filas con 14 características distribuidas en las siguientes columnas:

1. UID: identificador único que va del 1 al 10000.
2. ID de producto: consta de una letra L, M o H para baja (50 % de todos los productos), media (30 %) y alta (20 %) como variantes de calidad del producto y un número de serie específico de la variante.
3. Temperatura del aire [$^{\circ}\text{K}$]: generada mediante un proceso de caminata aleatoria y luego normalizada a una desviación estándar de 2°K alrededor de 300°K .
4. Temperatura del proceso [$^{\circ}\text{K}$]: generada mediante un proceso de paseo aleatorio normalizado a una desviación estándar de 1°K , sumada a la temperatura del aire más 10°K .
5. Velocidad de rotación [rpm]: calculada a partir de una potencia de 2860 W, superpuesta con un ruido distribuido normalmente
6. Torque [Nm]: los valores de torque se distribuyen normalmente alrededor de 40 Nm con un $\ddot{f} = 10$ Nm y sin valores negativos.
7. Desgaste de herramienta [min]: Las variantes de calidad H/M/L añaden 5/3/2 minutos de desgaste a la herramienta usada en el proceso.

Además, la variable 'fallo de la máquina', puede tomar un valor de 0 y 1, dependiendo si la máquina ha fallado para cualquiera de los siguientes modos de falla: Fallo en la alimentación, fallo por desgaste de herramienta, fallo por sobreesfuerzo, fallo de disipación de calor y fallos aleatorios.

El conjunto de datos se someterá a distintos algoritmos de aprendizaje automático, esto con el propósito de clasificar cada una de las fallas anteriormente mencionadas y poder realizar un correcto diagnóstico de éstas. Los algoritmos que se aplicarán son: Árbol de decisión, métodos de ensamble (Bagging, Boosting, Stacking), Máquinas Vectores de Soporte y KNN, debido a que trabajos de clasificación de fallas previos muestran que los algoritmos de aprendizaje automático ofrecen mejores valores en las métricas de evaluación.

Para el análisis de los datos se utilizó el IDE (Interface Development Environment) Spyder para programar en Python. Las librerías aplicadas fueron: pandas, numpy, sklearn, sklearn.model_selection, sklearn.metrics, seaborn, matplotlib.pyplot.

Finalmente, una vez realizada la clasificación, se evalúa la eficacia de cada algoritmo, mediante el modelo de matriz de confusión, además de otras métricas como: precisión, exhaustividad, F1 score y exactitud. Se comparan los resultados de todos los algoritmos para determinar el algoritmo que mejores resultados ofrece para este tipo de sistemas. En el caso de las métricas de evaluación, debido a la distribución de los datos, se opta por utilizar `average='weighted'` en los parámetros de `sklearn.metrics`, la cual devuelve el promedio considerando la proporción de cada etiqueta.

Para procesar el conjunto de datos y presentar los resultados, a cada una de las fallas se le asignó un número:

0. Sin falla,
1. Fallo de alimentación,
2. Fallo por desgaste de herramienta,
3. Fallo por sobreesfuerzo,
4. Fallo de disipación de calor,
5. Fallos aleatorios.

A continuación, se muestran las matrices de confusión correspondientes a cada uno de los algoritmos empleados en este trabajo:

Tabla 3 Matriz de confusión SVM.

Tipo de falla	0	1	2	3	4	5
0	1930	0	1	0	1	0
1	0	17	0	5	0	0
2	0	0	7	1	0	0
3	0	0	0	18	0	0
4	0	4	0	0	13	0
5	3	0	0	0	0	0

Tabla 4 Matriz de confusión Árbol de decisión.

Tipo de falla	0	1	2	3	4	5
0	1924	0	1	0	0	7
1	0	19	0	2	1	0
2	0	0	8	0	0	0
3	0	0	1	16	1	0
4	0	2	0	0	15	0
5	3	0	0	0	0	0

Tabla 5 Matriz de confusión Boosting.

Tipo de falla	0	1	2	3	4	5
0	1925	0	1	0	0	6
1	0	19	0	2	1	0
2	0	0	8	0	0	0
3	0	0	1	16	1	0
4	0	2	0	0	15	0
5	3	0	0	0	0	0

Tabla 6 Matriz de confusión Bagging.

Tipo de falla	0	1	2	3	4	5
0	1931	1	0	0	0	0
1	0	18	0	4	0	0
2	0	0	8	0	0	0
3	0	0	1	15	2	0
4	0	2	0	0	15	0
5	3	0	0	0	0	0

Tabla 7 Matriz de confusión Stacking.

Tipo de falla	0	1	2	3	4	5
0	1925	0	1	0	0	6
1	0	19	0	2	1	0
2	0	0	8	0	0	0
3	0	0	1	16	1	0
4	0	2	0	0	15	0
5	3	0	0	0	0	0

Tabla 8 Matriz de confusión KNN.

Tipo de falla	0	1	2	3	4	5
0	1930	0	1	0	1	0
1	0	17	0	5	0	0
2	0	0	7	1	0	0
3	0	1	0	17	0	0
4	0	3	0	0	14	0
5	3	0	0	0	0	0

Tabla 9 Evaluación de rendimiento de modelos de Machine Learning.

Modelo	Precisión	Exhaustividad	F1 score	Exactitud
SVM	0.9887	0.9905	0.9892	0.9905
Árbol de decisión	0.9931	0.9910	0.9920	0.9910
Boosting	0.9931	0.9915	0.9922	0.9915
Bagging	0.9920	0.9935	0.9927	0.9935
Stacking	0.9931	0.9945	0.9937	0.9945
KNN	0.9914	0.9925	0.9918	0.9925

Observando las matrices de confusión y las demás métricas de evaluación, se aprecia que los métodos de ensamble Bagging y Stacking son los que ofrecen un mayor valor en cada una de las métricas de medición al momento de realizar el diagnóstico de una falla, lo cual coincide con los resultados obtenidos en (Acosta Ruiz, 2023). Por otra parte, el algoritmo que arroja un menor número de aciertos es el SVM, no obstante, en todos los demás casos de obtuvo un porcentaje de aciertos mayor al 99%. Cabe destacar que ningún algoritmo logró identificar la categoría “falla aleatoria”, esto puede ser ocasionado por la distribución de las muestras tomadas, ya que, al ser, en su mayoría, muestras sin fallas, éstas y las fallas aleatorias se agrupan de igual manera debido a que tienen características similares. Otro motivo por el que las fallas aleatorias no pueden ser diagnosticadas correctamente, es que los parámetros elegidos al momento de realizar el conjunto de datos no sean los adecuados o sea

necesario agregar nuevos parámetros, como el análisis de vibraciones o la medición de la corriente eléctrica.

CAPITULO 4. MATERIALES Y MÉTODOS

En esta sección se describe en detalle el diseño experimental, los materiales utilizados, los métodos implementados y las estrategias de adquisición y procesamiento de datos para el monitoreo continuo de algunos parámetros como vibración, corriente, temperatura, etc. de motores, válvulas y cilindros neumáticos, que son algunos de los componentes más utilizados en la industria. Este monitoreo se realizó utilizando sensores MPU6050, LM35 y SCT013, entre otros sensores, integrados con la plataforma Arduino.

Descripción del sistema (Banda Transportadora)

El estudio se llevó a cabo en dos sistemas didácticos que simulan una línea de producción, uno ubicado en el área de Control y Automatización, y el otro en el área de Automatización Industrial, ambos en el Instituto Tecnológico de Aguascalientes. El primer sistema simula una línea de llenado de latas, éste consiste en una banda transportadora, la cual transporta las latas a las estaciones de llenado, etiquetado y almacenaje. De este sistema se monitorea el motor que mueve la banda, como se mencionó anteriormente, esta banda transporta las latas a cada una de las estaciones; por lo que es un equipo crítico en el proceso.

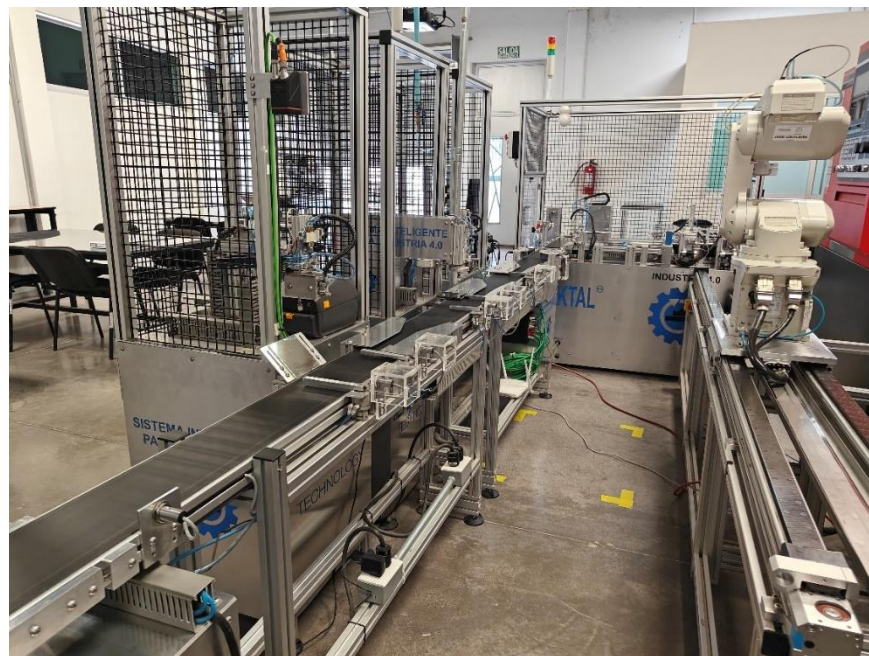


Figura 13 Banda transportadora del sistema de enlatado.

La banda transportadora es impulsada por un moto-reductor trifásico de la marca DKM con las siguientes especificaciones:

- ❖ $\frac{1}{6} Hp$
- ❖ 220 V
- ❖ $161 \frac{lb}{in}$
- ❖ 50 rpm

El objetivo principal del monitoreo es detectar y prevenir fallas tempranas mediante la evaluación continua de la vibración, la corriente eléctrica, entre otros parámetros utilizando sensores y una plataforma de desarrollo en Arduino. Las fallas analizadas en este sistema fueron tensión baja y rozamiento, la primera se indujo girando un tornillo que empuja el bloque de los rodillos, este proceso se realiza en ambos lados de la banda de manera uniforme (ver figura 14). Para inducir la segunda falla, se coloca un objeto entre la estructura del soporte de la banda y la banda, de manera que existe un rozamiento que afecte al movimiento de la banda.

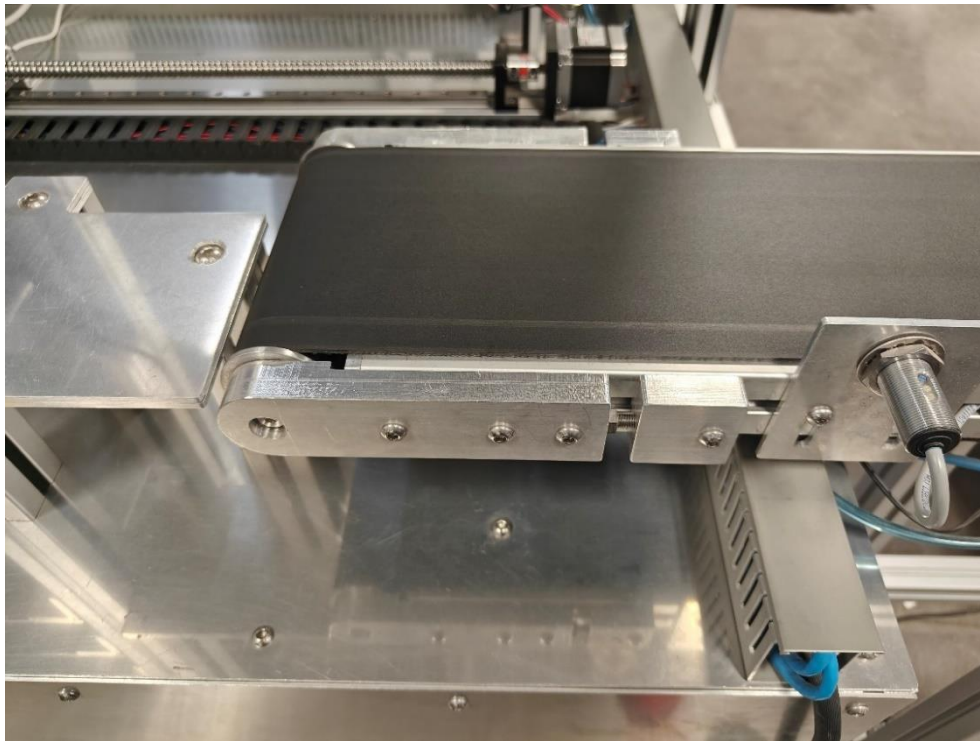


Figura 14 Tensor de la banda transportadora.

Descripción del sistema (Transferencia de tapas)

El segundo sistema en el que se realizó el estudio también consiste en un módulo de enlatado que consta de tres estaciones que llevan a cabo la tarea de llenar una lata con un producto granulado, posteriormente se le coloca una tapa y, por último, ésta se engargola (ver figura 15).



Figura 15 Módulo de enlatado.

Los componentes neumáticos seleccionados se encuentran en la segunda estación del proceso (figura 16), donde se lleva a cabo el tapado de las latas mediante un manipulador neumático designado para esta tarea. Las tapas son almacenadas en un depósito equipado con un sensor que indica cuando el depósito está vacío. El manipulador utiliza dos actuadores neumáticos y una ventosa para tomar las tapas del depósito. Una vez que la tapa ha sido colocada en la lata, ésta avanza al siguiente proceso.

Se han elegido el cilindro que aloja la ventosa y su correspondiente electroválvula para llevar a cabo las pruebas necesarias. La electroválvula es una válvula de 5 vías y 2 posiciones

modelo SY3120-5LZ-C4, con una bobina de 24V; mientras que el cilindro es un MGQM16-20 de doble efecto con un recorrido de 20mm.

En el caso de la válvula, las fallas a evaluar son: válvula usada, fuga por empaque, fuga por manguera y vías obstruidas. Para la falla de válvula usada o desgaste en válvula, se monitoreó la válvula que estaba originalmente en el módulo y se hizo el contraste monitoreando una electroválvula totalmente nueva, la cual se consideró como la condición óptima o buen funcionamiento del equipo. Posteriormente, se le retiró el empaque a la electroválvula para provocar la fuga por falta de empaque (ver figura 17, a)), mientras que para la fuga por manguera, se le hizo un corte a la misma. Por último, se colocó una plastilina epóxica (ver figura 17, b)), simulando suciedad o polvo que, comúnmente, obstruyen las vías de la electroválvula.

Por otra parte, para el cilindro se provocaron las siguientes fallas: escasez de tapas, falta de succión y manguera rota. Se puede observar que algunas de estas fallas no pertenecen al componente como tal, sino que son inherentes al proceso y la función que cumple el cilindro, como lo es la escasez de tapa en el depósito o la falta de succión.

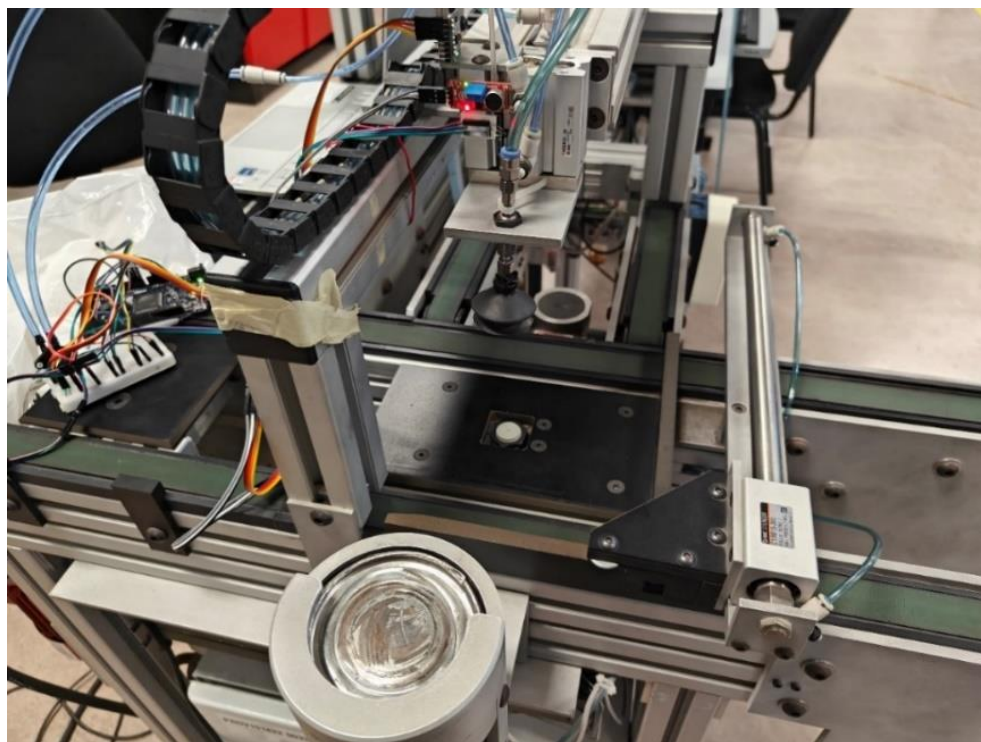


Figura 16 Estación de colocación de tapa.

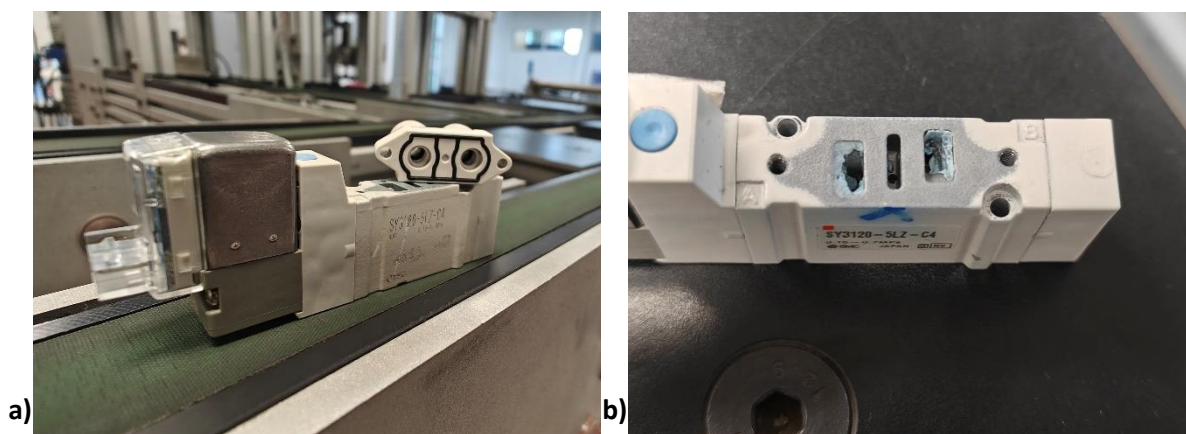


Figura 17 a) Muestra el empaque que se extrae de la válvula, b) Salidas de la válvula obstruidas por suciedad.

Cabe mencionar que ningún componente de las instalaciones prestadas fue dañado para realizar las pruebas realizadas, sino que se utilizaron componentes de repuesto para inducir las fallas analizadas en este trabajo. Además, los ajustes realizados a los sistemas han sido revertidos en su totalidad, devolviendo el equipo a su estado original.

Materiales

Los materiales utilizados en este estudio incluyen sensores y equipos para la adquisición y procesamiento de datos de los sistemas:

Sensores:

- ❖ **MPU6050:** Sensor de aceleración y giroscopio de seis ejes utilizado para medir la aceleración lineal y angular. Este sensor captura simultáneamente los ejes x, y, z y el canal correspondiente. Utiliza el protocolo de comunicación I2C para conectarse con placas Arduino. A pesar de sus capacidades avanzadas, el MPU-6050 tiene un precio asequible, considerando que combina funciones de acelerómetro y giroscopio en un mismo dispositivo.
- ❖ **SCT-013:** Los sensores de corriente SCT-013 son dispositivos no invasivos diseñados para medir la intensidad eléctrica que fluye a través de un conductor sin necesidad de cortarlo o modificarlo. Funcionan como transformadores, donde la corriente en el cable actúa como el devanado primario (1 espira), mientras que

internamente tienen un devanado secundario que puede tener hasta más de 2000 espiras, dependiendo del modelo.

- ❖ **LM35:** El LM35 es un sensor de temperatura compacto con un encapsulado TO-92 de 3 pines. Proporciona una salida analógica lineal con una pendiente de $10\text{mV}/^{\circ}\text{C}$, lo que permite medir temperaturas en un rango amplio desde -55°C hasta 150°C . Este sensor es ideal para sistemas de monitoreo. El LM35 es fácil de usar, no requiere circuitos adicionales ya que se alimenta directamente con 5V y su salida analógica varía entre 0V y 1.5V. Este voltaje puede ser directamente leído por el conversor analógico-digital (ADC) del controlador Arduino. Es conocido por su precisión, proporcionando una señal proporcional directa a la temperatura en grados Celsius con una precisión de $\pm 0.5^{\circ}\text{C}$.
- ❖ **KY-038:** El módulo sensor de sonido KY-038 consta de un micrófono que detecta variaciones de frecuencia de sonido y entrega a sus salidas un valor analógico o un valor digital. Cuando el micrófono detecta una frecuencia de sonido, experimenta un cambio en la separación entre sus placas. Este cambio se traduce en una pequeña variación de voltaje en su salida. Este voltaje puede ser directamente interpretado desde la salida analógica del micrófono. Alternativamente, el voltaje puede ser comparado con un umbral de referencia en el comparador, el cual activará su salida cuando se supere dicho umbral en la entrada inversora.
- ❖ **MPX5700DP:** Es un sensor piezoeléctrico de presión diferencial. Esto significa que detecta la diferencia de presión entre dos puntos y convierte esta diferencia en una señal eléctrica proporcional. Está diseñado para medir presiones diferenciales en un rango específico, de 0 a 700kPa. El sensor proporciona una salida de señal analógica, generalmente en forma de voltaje, que es proporcional a la presión diferencial medida.

Placa de Desarrollo:

- ❖ **Arduino Due:** El Arduino Due destaca como uno de los modelos más potentes dentro de la familia Arduino. Utiliza el microcontrolador SAM3X8E ARM Cortex-M3, que conserva las funcionalidades clásicas de Arduino y añade nuevas capacidades. Ofrece un total de 54 pines de entrada/salida (12 de las cuales son PWM con resolución

configurable), 12 entradas analógicas con una resolución de 12 bits, 4 puertos UART por hardware y dos conversores DAC (digital a analógico), un resonador de cuarzo de 84MHz y dos conexiones USB (una de programación y otra que puede actuar como USB Host). Debido a su potente procesador y amplias capacidades de entrada/salida, el Arduino Due es ideal para aplicaciones que requieren un procesamiento intensivo, como controladores de sistemas embebidos, sistemas de automatización, control de motores, aplicaciones industriales y proyectos donde se necesite manejar grandes volúmenes de datos o ejecutar algoritmos complejos en tiempo real.

Software

- ❖ **Arduino IDE:** El Arduino IDE (Integrated Development Environment, Entorno de Desarrollo Integrado) es una herramienta de software fundamental para programar y cargar código en placas de desarrollo Arduino. Proporciona un espacio donde los usuarios pueden escribir, editar y guardar su código en lenguaje de programación C/C++, el cual es el estándar para programar en Arduino. Permite compilar el código escrito en el editor para convertirlo en instrucciones entendibles por el microcontrolador de la placa Arduino. Luego, facilita la carga del código compilado en la placa conectada. Incluye un monitor serie integrado que muestra los mensajes de depuración, salida de datos o cualquier otra información transmitida desde la placa Arduino a través del puerto serie. Está disponible para sistemas operativos principales como Windows, macOS y Linux, y es compatible con una amplia variedad de placas Arduino, como Arduino Uno, Arduino Mega, Arduino Due, entre otras, así como con placas de otros fabricantes que usan el entorno de desarrollo de Arduino.
- ❖ **Spyder:** Spyder es un IDE de código abierto desarrollado gracias a Anaconda Inc. es especialmente popular entre los científicos de datos y los investigadores debido a sus capacidades avanzadas de análisis y visualización de datos. Spyder proporciona un editor de código con resaltado de sintaxis, autocompletado inteligente y capacidades de edición múltiple. Ofrece una ventana donde se pueden visualizar y explorar las variables que están definidas en el entorno actual. Integra una consola interactiva que permite ejecutar comandos Python de manera inmediata y ver resultados en tiempo real. Debido a su integración en Anaconda, la instalación es sencilla e incluye librerías

de uso común en análisis de datos y aprendizaje de máquina, como Matplotlib, Numpy, Scipy.

- ❖ **Sklearn:** scikit-learn, comúnmente abreviado como sklearn, es una biblioteca de código abierto para aprendizaje automático en Python. Consiste en un conjunto de funciones desarrolladas en Python diseñadas para realizar análisis predictivo, tales como clasificación y clustering. Esta biblioteca se fundamenta en NumPy, SciPy y matplotlib, lo que facilita la reutilización de código que emplee estas bibliotecas.
- ❖ **TensorFlow:** TensorFlow es una biblioteca de código abierto esencial para el aprendizaje automático y la inteligencia artificial. Desarrollada por Google, está diseñada para satisfacer las demandas de aplicaciones que emplean redes neuronales artificiales. Con TensorFlow, puedes construir y entrenar redes neuronales para identificar patrones y realizar razonamientos similares a los humanos en diferentes contextos de Machine Learning.
- ❖ **Keras:** Keras es una interfaz de programación de aplicaciones (API) para redes neuronales desarrollada en Python. Es una biblioteca de código abierto diseñada para funcionar sobre frameworks como Theano y TensorFlow. Desarrollada por Google, Keras se destaca por su modularidad, velocidad y facilidad de uso. Proporciona una manera intuitiva y eficiente de construir modelos de Deep Learning. A partir de TensorFlow 2.0, Keras se convirtió en la API de alto nivel predeterminada para construir modelos en TensorFlow. Hoy en día, Keras es una de las API más populares para el desarrollo y prueba de redes neuronales, simplificando considerablemente la creación de capas y configuración de arquitecturas complejas.

La configuración del sistema fue diseñada para capturar vibraciones, corrientes, sonidos y temperaturas asociadas con la operación de la banda transportadora:

- ❖ **MPU6050:** Montado en la carcasa del motor de la banda transportadora, perpendicular al eje, para detectar y registrar vibraciones generadas por el movimiento de la banda.
- ❖ **SCT013-30:** Colocados en cada una de las fases del motor para una mayor precisión en la medición.

- ❖ **LM35:** Se coloca uno en la carcasa del motor, cercano a la zona de los rodamientos; mientras que otro se ubica en una zona alejada de la línea de producción, para medir la temperatura del ambiente.
- ❖ **KY-038:** De igual manera se coloca en la carcasa del motor en la zona de los rodamientos.

El sensor MPU6050 y los sensores SCT013-30 utilizan el protocolo I2C para la comunicación de datos mediante los pines SCL (Serial Clock Line) y SDA (Serial Data Line), por otra parte, los sensores LM35 y KY-038 se conectaron a los pines analógicos del Arduino Due.

Para el cilindro y la válvula, se sustituyeron los sensores SCT013-30 por el sensor MPX5700DP, los sensores:

- ❖ **MPU6050:** Se coloca en el cuerpo de la válvula y en el soporte del cilindro.
- ❖ **LM35:** Se coloca un sensor en el cuerpo tanto de la válvula como del cilindro y otro se ubica fuera del módulo para capturar la temperatura del ambiente.
- ❖ **KY-038:** Ubicado en el manifold de las válvulas para detectar el sonido de fugas o agentes ajenos a la válvula, en el caso del cilindro, se colocó en el cuerpo de éste.
- ❖ **MPX5700DP:** Para la válvula, se mide la diferencia de presión entre la salida de la válvula y el suministro de aire al manifold, en el caso del cilindro, se monitorea cada una de las tomas de aire del cilindro.

Adquisición de Datos:

Arduino Due se programó para adquirir datos de los sensores de manera continua y en tiempo real. Se estableció una frecuencia de muestreo de 120 Hz para capturar cambios rápidos en las condiciones operativas de la banda transportadora y asegurar la precisión en la medición de vibración, corriente eléctrica y sonido.

En este proyecto de adquisición de datos, los datos son transmitidos desde Arduino a través de la comunicación serial y son recibidos y guardados en Python para su posterior análisis y procesamiento. El sistema está diseñado para monitorear la banda transportadora durante dos ciclos completos del motor (2.4 segundos). Durante cada ciclo, se recopilan datos que son guardados para el análisis posterior. Después de completar dos ciclos, el monitoreo se reinicia automáticamente para continuar registrando las señales del motor. Para los componentes

neumáticos este monitoreo dura 5 segundos, ya que este es el tiempo que toma un solo ciclo de la estación de colocación de tapa. Para la banda transportadora se obtuvieron 500 muestras para cada una de las fallas y el estado Ok del equipo, dando un total de 1500 muestras. Para la válvula se registraron 300 para cada estado del componente y para el cilindro 150 muestras, para un total de 1500 y 600. En la figura 18 se exhiben los registros sin procesar obtenidos directamente del sensor durante el proceso de medición. Estas gráficas muestran de manera visual la evolución temporal de los datos adquiridos, proporcionando una visualización clara y precisa de las mediciones crudas registradas por el sensor en cada ciclo de captura.

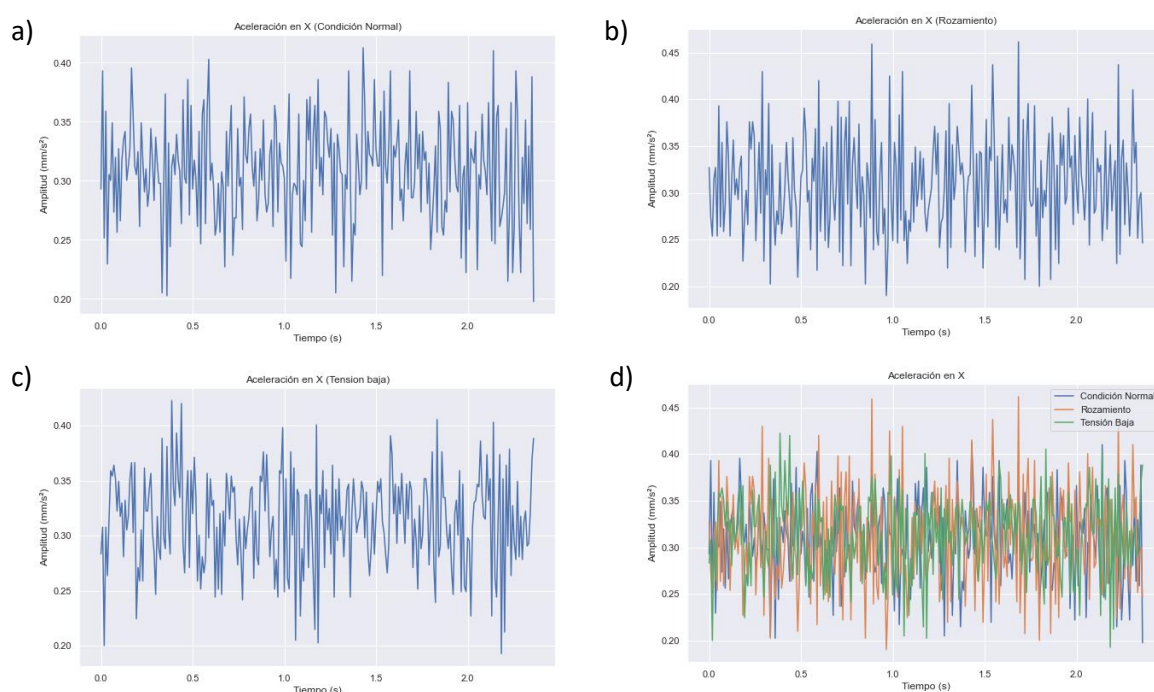


Figura 18 Representación gráfica de las distintas fallas en el dominio del tiempo.

En a), b), y c) muestran las señales en el dominio del tiempo de la aceleración medida por el acelerómetro en el eje X para cada una de las fallas de la banda transportadora por separado, en d) se sobrepone cada una de estas señales, haciendo evidente la diferencia que existen en las señales en función de la falla que se analiza.

Procesamiento de Datos:

Los datos adquiridos fueron procesados en tiempo real por Arduino Due utilizando algoritmos específicos para calcular medidas relevantes de cada una de las señales.

Para clasificar los fallos de la banda transportadora se calcula distintas características. Primeramente, se calcula la transformada de Fourier de cada una de las señales. En la figura 19 se muestra la transformada de Fourier de las señales presentadas en la figura 18. Se observa como la frecuencia fundamental del motor se encuentra en 0.8333 Hz, debido al número de revoluciones del moto-reductor. La selección de características se realizó basado en los artículos (Tayyab et al., 2020) y (Utah & Jung, 2020).

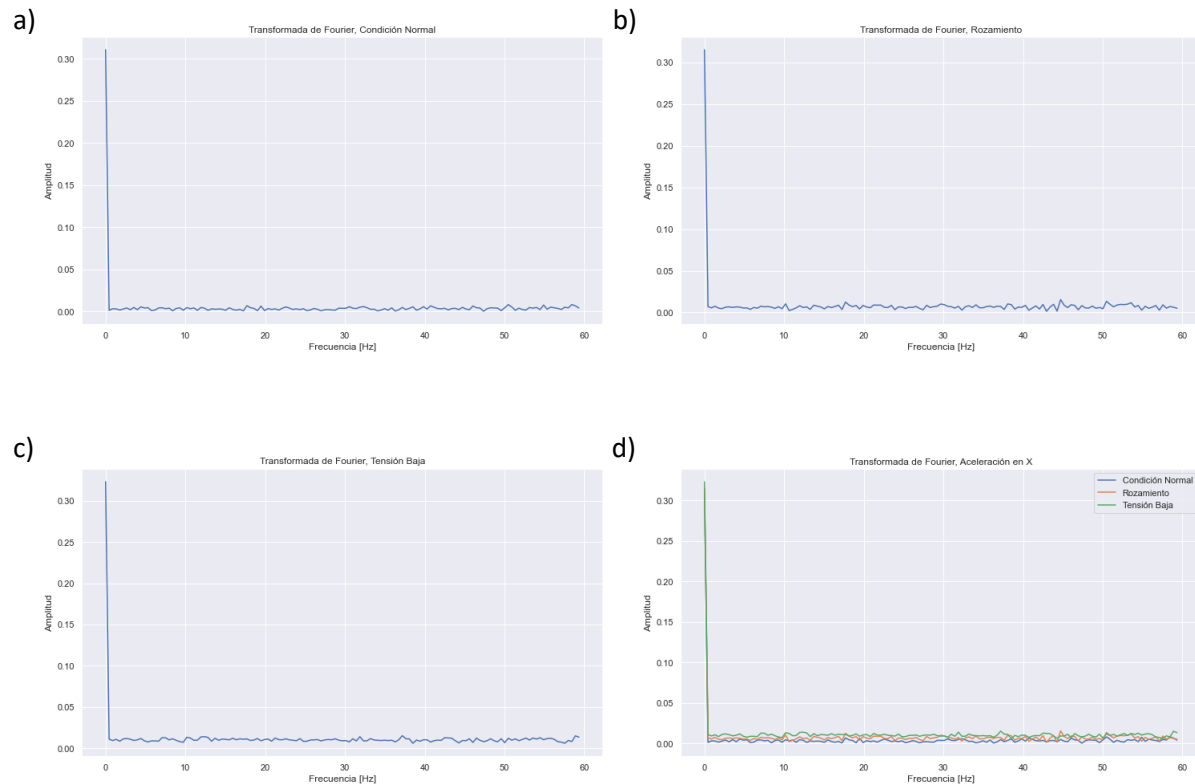


Figura 19 Representación gráfica de las distintas fallas en el dominio de la frecuencia.

En el dominio del tiempo se extrajeron las siguientes características:

- ❖ Promedio,
- ❖ Media cuadrática,
- ❖ Desviación estándar,
- ❖ Skewness,
- ❖ Curtosis,
- ❖ Factor de forma o Shape Factor,
- ❖ Factor de cresta o Crest Factor,

- ❖ Factor de Impulso o Impulse Factor.
- ❖ Mientras que en el dominio de la frecuencia, se obtienen las siguientes características:
- ❖ Promedio,
- ❖ Skewness,
- ❖ Curtosis.

Una vez que se extraen todas estas características se realiza una reducción de dimensiones mediante eliminación hacia atrás, calculando el valor de p , que es una medida estadística que indica la probabilidad de obtener resultados observados y se utiliza para determinar la relevancia de variables en modelos. El valor de p se calcula utilizando la biblioteca “statsmodels” de Python. El valor de α o el umbral que p debe superar para no ser considerada una variable relevante es de 0.05, por lo que todas aquellas variables que superaron este valor fueron eliminadas del dataset. Por otra parte, se realiza otra reducción de variables utilizando el método de PCA, con una proporción de varianza mínimo de 0.95.

Posterior a la selección de variables se le da un valor numérico a cada una de los estados o fallas que se evaluaron para poder procesar los datos y mostrar los resultados.

Para la banda:

0. Condición normal (Sin falla),
1. Rozamiento en la banda,
2. Tensión baja.

Para la válvula:

0. Válvula nueva,
1. Fuga por falta de empaque,
2. Fuga por manguera rota,
3. Obstrucción en vías,
4. Válvula usada.

Y, por último, para el cilindro:

0. Condición Normal,
1. Fuga por manguera rota,

2. Falta de tapas en el depósito,
3. Falta de succión en la ventosa.

Todos estos valores se integraron dentro del vector “target”, el cual sirve, tanto como para entrenar el algoritmo como para evaluar la eficacia al predecir fallas en los sistemas.

En este punto, se realiza la división del conjunto de datos en dos subconjuntos: uno para entrenamiento del modelo y otro para evaluar su rendimiento; mediante la función “train_test_split”. Para esto se toma una relación de 80:20 (80% de los datos se utilizan para el entrenamiento de los modelos, mientras que el 20% se utiliza para evaluar la generalización de las predicciones del modelo). Además, esta división se realiza de manera aleatoria al agregar el parámetro “random_state” para garantizar que los resultados sean reproducibles y consistentes en diferentes ejecuciones del código.

```
x_train,x_test,y_train,y_test=train_test_split(X_Modeled,target_1,test_size=0.2,random_state=500)
```

Figura 20 División del Dataset, creación de conjuntos de entrenamiento y prueba.

Por último, antes de pasar al entrenamiento de algoritmos, es necesario normalizar las características extraídas, esto se realiza mediante la herramienta StandardScaler de sklearn. Esta herramienta transforma los datos para que tengan una media de 0 y una desviación estándar de 1, con la formula:

$$z = \frac{x - media}{desviación\ estándar} \quad (172)$$

```
from sklearn.preprocessing import StandardScaler  
sc=StandardScaler()  
x_train=sc.fit_transform(x_train)  
x_test=sc.transform(x_test)
```

Figura 21 Normalización de los datos.

Normalizar los datos es útil cuando las características tienen diferentes escalas y se quiere mejorar el rendimiento de algoritmos de aprendizaje automático.

Algoritmos

Finalizado el procesamiento de los datos, se comienza con la creación de los modelos. Como se vio en la sección de antecedentes, los algoritmos de aprendizaje supervisado son los que mejores resultados ofrecen al clasificar las fallas de los equipos. Por este motivo se optó por utilizar los siguientes algoritmos:

- ❖ Máquina de Vector Soporte,
- ❖ Árboles de Decisión,
- ❖ Modelos de ensambles (Bagging, Boosting, Stacking),
- ❖ KNN,
- ❖ Redes Neuronales (Perceptrón Multicapa).

Primeramente, se obtiene el modelo de SVM. El cual utiliza el kernel rbf, ya que este permite modelar relaciones no lineales y complejas, es menos propenso al sobreajuste y es adecuada cuando el número de dimensiones es alto. Los valores de γ y C se determinan mediante validación cruzada. Para realizar esta validación cruzada se usó la herramienta GridSearchCV, tal como se muestra en la figura 22. En esta figura se observa que los valores de C varían de 0.01 a 100, mientras que γ , toma los valores de 0.001, 0.01, 0.1 y 1.

```
#Cross Validation
from sklearn.model_selection import GridSearchCV
from sklearn import svm

param_grid = {
    'C': [0.01, 0.1, 1, 5, 10, 25, 50, 100],
    'gamma': [0.001, 0.01, 0.1, 1],
    'kernel': ['rbf']
}

grid_search = GridSearchCV(svm.SVC(probability=True), param_grid, cv=5)
grid_search.fit(x_train, y_train)
# Imprimir los mejores parámetros
print("Mejores parámetros encontrados:")
print(grid_search.best_params_)
```

Figura 22 Selección de hiperparámetros del modelo SVM.

La siguiente tabla contiene el valor de los hiperparámetros que mejor resultados obtuvieron al realizar la validación cruzada con SVM.

Tabla 10 Hiperparámetros utilizados en SVM.

Sensor	C	γ	Reducción de Variables	Equipo
Todos	5	0.001	PCA	Banda
Vibración	25	0.001	PCA	Banda
Corriente	5	0.01	PCA	Banda
Sonido	25	0.01	PCA	Banda
Temperatura	5	0.1	PCA	Banda
Todos	50	0.001	p-valor	Banda
Vibración	100	0.001	p-valor	Banda
Corriente	25	0.1	p-valor	Banda
Temperatura	5	0.01	p-valor	Banda
Sonido	100	0.01	p-valor	Banda
Todos	5	0.001	No	Banda
vibración	10	0.001	No	Banda
Corriente	10	0.01	No	Banda
Temperatura	100	0.001	No	Banda
Sonido	100	0.01	No	Banda
Todos	25	0.001	PCA	Válvula
Vibración	50	0.001	PCA	Válvula
Presión	100	0.1	PCA	Válvula
Sonido	50	0.01	PCA	Válvula
Temperatura	50	0.01	PCA	Válvula
Todos	50	0.001	p-valor	Válvula
Vibración	5	0.01	p-valor	Válvula
Presión	50	0.1	p-valor	Válvula
Temperatura	10	0.01	p-valor	Válvula
Sonido	10	0.1	p-valor	Válvula
Todos	10	0.001	No	Válvula
Vibración	50	0.001	No	Válvula
Presión	25	0.01	No	Válvula
Temperatura	50	0.001	No	Válvula
Sonido	5	0.1	No	Válvula
Todos	25	0.001	PCA	Cilindro
Vibración	10	0.001	PCA	Cilindro
Presión	5	1	PCA	Cilindro
Sonido	10	0.1	PCA	Cilindro
Temperatura	50	0.1	PCA	Cilindro
Todos	50	0.001	p-valor	Cilindro
Vibración	5	0.01	p-valor	Cilindro
Presión	50	0.1	p-valor	Cilindro
Temperatura	100	0.1	p-valor	Cilindro
Sonido	50	0.01	p-valor	Cilindro
Todos	10	0.001	No	Cilindro
Vibración	100	0.001	No	Cilindro
Presión	5	1	No	Cilindro
Temperatura	100	0.1	No	Cilindro

Tabla 10 Hiperparámetros utilizados en SVM (continuación).

Sonido	5	0.1	No	Cilindro
--------	---	-----	----	----------

La columna reducción de variables, se refiere a si se realizó este proceso y con que método, esto con la finalidad de conocer si es necesario llevar a cabo este paso para mejorar la precisión del diagnóstico de fallas.

De igual manera, se llevó a cabo el mismo proceso para los algoritmos de árbol de decisiones y KNN. Para el árbol de decisiones, los hiperparámetros son el índice de pureza (Gini o entropía), el número mínimo de muestras por hoja (`min_samples_leaf`), el número mínimo de muestras que debe tener un nodo para poder ser dividido (`min_samples_split`) y la profundidad máxima del árbol (`max Depth`), tal como se muestra en la figura 23. Cabe mencionar que estos mismos hiperparámetros son los utilizados en los métodos de ensamble Boosting y Random Forest.

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5]
}

grid_search = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5)
grid_search.fit(x_train, y_train)
# Imprimir los mejores parámetros
print("Mejores parámetros encontrados:")
print(grid_search.best_params_)
```

Figura 23 Selección de hiperparámetros del modelo Árbol de decisión.

Para el algoritmo KNN, solo se utilizaron dos hiperparámetros (ver figura 24), `k` (número de vecinos) que varía de 1 a 40, y la métrica de medición de la distancia (euclidiana y manhattan).

```

from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier

param_grid = {
    'n_neighbors': [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
                    26,27,28,29,30,31,32,33,34,35,36,37,38,39,40],
    'metric': ['euclidean', 'manhattan']
}

grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
grid_search.fit(x_train, y_train)

# Imprimir los mejores parámetros
print("Mejores parámetros encontrados:")
print(grid_search.best_params_)

```

Figura 24 Selección de hiperparámetros del modelo KNN.

Dando como resultado las siguientes tablas:

Tabla 11 Hiperparámetros árbol de decisión.

Sensor	criterio	min_leaf	min_sample	max depth	Reducción de Variables	Equipo
Todos	Gini	5	5	5	PCA	Banda
Vibración	Entropía	2	5	5	PCA	Banda
Corriente	Entropía	5	5	35	PCA	Banda
Sonido	Gini	5	10	10	PCA	Banda
Temperatura	Gini	2	2	5	PCA	Banda
Todos	Entropía	1	2	50	p-valor	Banda
Vibración	Entropía	5	2	5	p-valor	Banda
Corriente	Entropía	1	2	40	p-valor	Banda
Temperatura	Entropía	5	10	10	p-valor	Banda
Sonido	Gini	1	2	10	p-valor	Banda
Todos	Entropía	1	2	40	No	Banda
Vibración	Entropía	5	2	5	No	Banda
Corriente	Gini	1	5	None	No	Banda
Temperatura	Gini	2	2	5	No	Banda
Sonido	Entropía	5	2	5	No	Banda
Todos	Entropía	1	5	10	PCA	Válvula
Vibración	Entropía	1	2	20	PCA	Válvula
Presión	Gini	1	2	45	PCA	Válvula
Sonido	Gini	2	5	10	PCA	Válvula
Temperatura	Entropía	2	5	25	PCA	Válvula
Todos	Gini	1	5	None	p-valor	Válvula
Vibración	Entropía	1	10	45	p-valor	Válvula

Tabla 11 Hiperparámetros árbol de decisión (continuación).

Presión	Entropía	1	2	40	p-valor	Válvula
Temperatura	Gini	1	2	None	p-valor	Válvula
Sonido	Entropía	2	2	45	p-valor	Válvula
Todos	Gini	1	5	None	No	Válvula
Vibración	Entropía	1	2	None	No	Válvula
Presión	Gini	1	2	None	No	Válvula
Temperatura	Gini	1	5	None	No	Válvula
Sonido	Gini	1	2	45	No	Válvula
Todos	Entropía	1	5	10	PCA	Cilindro
Vibración	Gini	5	2	5	PCA	Cilindro
Presión	Gini	5	5	5	PCA	Cilindro
Sonido	Gini	1	2	5	PCA	Cilindro
Temperatura	Entropía	5	5	5	PCA	Cilindro
Todos	Gini	1	2	None	p-valor	Cilindro
Vibración	Entropía	2	10	5	p-valor	Cilindro
Presión	Entropía	5	5	None	p-valor	Cilindro
Temperatura	Entropía	1	10	5	p-valor	Cilindro
Sonido	Gini	5	2	5	p-valor	Cilindro
Todos	Gini	1	2	None	No	Cilindro
Vibración	Gini	5	2	5	No	Cilindro
Presión	Entropía	5	2	30	No	Cilindro
Temperatura	Entropía	2	2	5	No	Cilindro
Sonido	Gini	1	10	5	No	Cilindro

Tabla 12 Hiperparámetros KNN.

Sensor	k	Métrica	Reducción de Variables	Equipo
Todos	6	Euclidiana	PCA	Banda
Vibración	7	Euclidiana	PCA	Banda
Corriente	1	Euclidiana	PCA	Banda
Sonido	12	Manhattan	PCA	Banda
Temperatura	25	Manhattan	PCA	Banda
Todos	4	Manhattan	p-valor	Banda
Vibración	7	Manhattan	p-valor	Banda
Corriente	1	Euclidiana	p-valor	Banda
Temperatura	9	Manhattan	p-valor	Banda
Sonido	24	Euclidiana	p-valor	Banda
Todos	4	Manhattan	No	Banda
Vibración	8	Manhattan	No	Banda
Corriente	1	Manhattan	No	Banda

Tabla 12 Hiperparámetros KNN (continuación).

Temperatura	13	Manhattan	No	Banda
Sonido	12	Manhattan	No	Banda
Todos	9	Manhattan	PCA	Válvula
Vibración	3	Manhattan	PCA	Válvula
Presión	1	Euclidiana	PCA	Válvula
Sonido	11	Euclidiana	PCA	Válvula
Temperatura	3	Euclidiana	PCA	Válvula
Todos	38	Manhattan	p-valor	Válvula
Vibración	1	Euclidiana	p-valor	Válvula
Presión	3	Manhattan	p-valor	Válvula
Temperatura	5	Manhattan	p-valor	Válvula
Sonido	3	Manhattan	p-valor	Válvula
Todos	29	Manhattan	No	Válvula
Vibración	17	Manhattan	No	Válvula
Presión	4	Manhattan	No	Válvula
Temperatura	2	Manhattan	No	Válvula
Sonido	3	Manhattan	No	Válvula
Todos	10	Manhattan	PCA	Cilindro
Vibración	31	Manhattan	PCA	Cilindro
Presión	17	Euclidiana	PCA	Cilindro
Sonido	5	Euclidiana	PCA	Cilindro
Temperatura	11	Manhattan	PCA	Cilindro
Todos	13	Manhattan	p-valor	Cilindro
Vibración	3	Manhattan	p-valor	Cilindro
Presión	9	Euclidiana	p-valor	Cilindro
Temperatura	16	Manhattan	p-valor	Cilindro
Sonido	13	Manhattan	p-valor	Cilindro
Todos	19	Euclidiana	No	Cilindro
Vibración	7	Euclidiana	No	Cilindro
Presión	7	Euclidiana	No	Cilindro
Temperatura	27	Manhattan	No	Cilindro
Sonido	14	Euclidiana	No	Cilindro

Por otra parte, para el caso de las redes neuronales MLP, los parámetros fueron el número de épocas (10, 20, 50 ó 100), el tamaño del mini-lote (2, 8, 32, 128) y el optimizador (SGD, Adam y RMSprop). Además de estos parámetros, se utilizaron distintos números de neuronas y capas, el número de neuronas fue de 32, 64 y 128, mientras que la red podía consistir en dos o tres capas (ver figura 11). Este proceso se repite tres veces y se promedian los

resultados, al final se elige la combinación de parámetros, neuronas y capas que mejor promedio obtenga.

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
ep=[10,20,50,100]
bs=[2,8,32,128]
optimizador = ['adam', 'sgd', 'RMSprop']
for optim in optimizador:
    for epo in (ep):
        for bat in (bs):
            print(optim,epo,bat,i,SL)
#Crear modelo secuencial
model = Sequential()
model.add(Dense(units=32, activation='relu', input_dim=x_train.shape[1]))
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=y_train.shape[1], activation='softmax'))
# Compilar el modelo
model.compile(loss='categorical_crossentropy', optimizer=optim, metrics=['accuracy'])
# Entrenar el modelo
model.fit(x_train, y_train, epochs=epo, batch_size=bat, validation_split=0.1, verbose=1)
```

Figura 25 Selección de hiperparámetros del modelo MLP.

Y estos son los parámetros de la red neuronal para cada uno de los casos:

Tabla 13 Hiperparámetros Red Neuronal.

Sensor	capas	neuronas	épocas	batch size	optimizador	Reducción de Variables	Equipo
Todos	2	64,32	10	2	Adam	PCA	Banda
Vibración	2	32,32	10	8	Adam	PCA	Banda
Corriente	3	128,64,32	50	128	Adam	PCA	Banda
Sonido	3	32,32,32	10	2	Adam	PCA	Banda
Temperatura	2	64,32	50	2	Adam	PCA	Banda
Todos	2	64,32	10	8	RMSprop	p-valor	Banda
Vibración	2	32,32	100	8	SGD	p-valor	Banda
Corriente	2	32,32	20	2	SGD	p-valor	Banda
Temperatura	2	64,32	50	8	SGD	p-valor	Banda
Sonido	2	32,32	50	8	RMSprop	p-valor	Banda
Todos	2	64,64	100	8	Adam	No	Banda
Vibración	2	64,32	50	8	RMSprop	No	Banda
Corriente	3	128,64,32	10	32	Adam	No	Banda
Temperatura	3	32,32,32	50	32	RMSprop	No	Banda
Sonido	3	128,64,32	20	8	Adam	No	Banda
Todos	2	64,32	100	2	Adam	PCA	Válvula
Vibración	3	32,32,32	20	2	Adam	PCA	Válvula
Presión	3	64,64,64	10	2	Adam	PCA	Válvula
Sonido	3	64,64,64	20	2	SGD	PCA	Válvula

Tabla 13 Hiperparámetros Red Neuronal (continuación).

Temperatura	3	64,64,64	100	32	Adam	PCA	Válvula
Todos	2	64,32	10	8	RMSprop	p-valor	Válvula
Vibración	2	64,32	50	8	RMSprop	p-valor	Válvula
Presión	3	64,64,64	50	2	Adam	p-valor	Válvula
Temperatura	2	32,32	100	2	SGD	p-valor	Válvula
Sonido	3	32,32,32	100	8	Adam	p-valor	Válvula
Todos	3	64,64,64	50	32	Adam	No	Válvula
Vibración	2	32,32	50	2	Adam	No	Válvula
Presión	2	64,64	100	2	Adam	No	Válvula
Temperatura	2	64,64	100	8	Adam	No	Válvula
Sonido	3	128,64,32	50	32	SGD	No	Válvula
Todos	2	64,32	20	2	Adam	PCA	Cilindro
Vibración	2	64,32	20	32	RMSprop	PCA	Cilindro
Presión	2	64,32	100	32	Adam	PCA	Cilindro
Sonido	2	64,64	100	8	SGD	PCA	Cilindro
Temperatura	2	64,64	50	8	Adam	PCA	Cilindro
Todos	3	64,64,64	20	2	Adam	p-valor	Cilindro
Vibración	2	64,64	50	32	RMSprop	p-valor	Cilindro
Presión	3	64,64,64	100	2	RMSprop	p-valor	Cilindro
Temperatura	2	64,64	100	2	Adam	p-valor	Cilindro
Sonido	3	128,64,32	50	32	Adam	p-valor	Cilindro
Todos	2	64,64	100	32	RMSprop	No	Cilindro
Vibración	3	32,32,32	20	2	Adam	No	Cilindro
Presión	3	128,64,32	100	2	Adam	No	Cilindro
Temperatura	2	32,32	100	2	RMSprop	No	Cilindro
Sonido	2	64,64	100	128	RMSprop	No	Cilindro

Una vez obtenidos los parámetros para cada uno de los algoritmos, es momento de entrenar los algoritmos, en las siguientes figuras se muestra la línea de código para cada uno de los algoritmos:

```
# SVM
from sklearn import svm
from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score
modeloSVM = svm.SVC(C=5, kernel='rbf', gamma=0.1, probability=True)
modeloSVM.fit(x_train, y_train)
y_pred=modeloSVM.predict(x_test)
```

Figura 26 Desarrollo del modelo SVM.

```
# ARBOL DE DECISION
from sklearn import tree
criterio='gini'
min_samp_le=1
min_samp_sp=2
max_dep=45
modeloTree = tree.DecisionTreeClassifier(criterion=criterio, min_samples_leaf= min_samp_le,
                                         min_samples_split=min_samp_sp,max_depth=max_dep)

modeloTree.fit(x_train,y_train)
y_pred=modeloTree.predict(x_test)
```

Figura 27 Desarrollo del modelo Árbol de decisión.

```
#BOOSTING
from sklearn import ensemble
modeloAB=ensemble.AdaBoostClassifier(base_estimator=modeloTree,n_estimators=40)
modeloAB.fit(x_train,y_train)
y_pred=modeloAB.predict(x_test)
```

Figura 28 Desarrollo del modelo AdaBoost.

```
#RANDOM FOREST
modeloRF= ensemble.RandomForestClassifier(criterion=criterio, min_samples_leaf= min_samp_le,
                                         min_samples_split=min_samp_sp,max_depth=max_dep,n_estimators=40)

modeloRF.fit(x_train,y_train)
y_pred=modeloRF.predict(x_test)
```

Figura 29 Desarrollo del modelo Random Forest.

```
#KNN
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=3, metric='manhattan')
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
```

Figura 30 Desarrollo del modelo Random Forest.

```
#STACKING
from sklearn.linear_model import LogisticRegression
estimator = [('ModelTree', modeloTree),('ModelTreeAB', modeloAB),('random_forest', modeloRF),
             ('svm_model', modeloSVM),('knn_model', knn)]
modeloS= ensemble.StackingClassifier(estimators=estimator,final_estimator=LogisticRegression(),cv=None)
modeloS.fit(x_train,y_train)
y_pred=modeloS.predict(x_test)
```

Figura 31 Desarrollo del modelo Stacking.

```
#Inicializar Red Neuronal
from keras.utils import to_categorical
y_train=y_train-1
y_train = to_categorical(y_train)
RNA=Sequential()
#Capa de entrada
RNA.add(Dense(units=64, activation='relu', input_dim=x_train.shape[1]))
#Capa oculta
RNA.add(Dense(64,activation='relu'))
RNA.add(Dense(64,activation='relu'))
#Capa de salida
RNA.add(Dense(units=y_train.shape[1], activation='softmax'))
RNA.summary()
RNA.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
RNA.fit(x_train,y_train,batch_size=32,epochs=100)
y_pred=RNA.predict(x_test)

y_pred = RNA.predict(x_test)
y_pred = np.argmax(y_pred, axis=1)
```

Figura 32 Desarrollo del modelo MLP.

Cada una de las variables **y_pred** almacenan los valores predichos por cada uno de los algoritmos, este vector se comparó con los valores de **y_test**, para determinar la eficacia de cada algoritmo; obteniendo matrices de confusión, como las que se muestran a continuación.

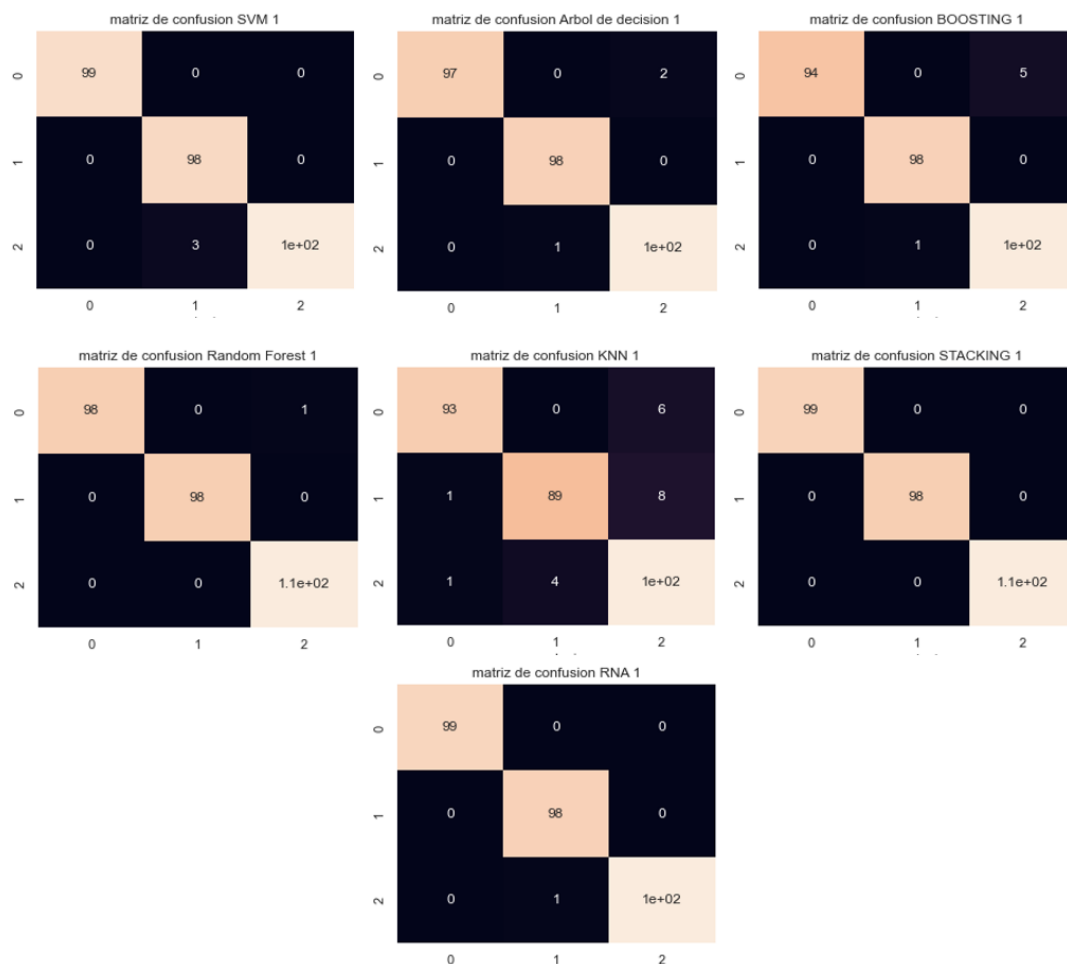


Figura 33 Ejemplo de matrices de confusión obtenidas.

Este proceso, se llevó a cabo un total de 100 veces, esto con la finalidad de evaluar la estabilidad de los algoritmos. En cada una de estas repeticiones se determinaban nuevamente los conjuntos de entrenamiento y prueba. Además, ejecutar el algoritmo varias veces permite calcular métricas de rendimiento más robustas (como precision, recall, F1-score, exactitud) y obtener un promedio de estas métricas, lo que proporciona una mejor representación de su efectividad.

CAPÍTULO 5. RESULTADOS

Como resultado de cada una de las comparaciones entre el valor predicho por el algoritmo y el valor real de la falla se obtuvieron las siguientes tablas, las cuales contienen las métricas de evaluación mencionadas anteriormente:

Tabla 14 Resultados de clasificación de fallas en banda transportadora aplicando PCA.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Sensor: Todos Reducción de Variables: PCA	SVM	0.99333333	0.99371069	0.99345839	0.99339934
	Árbol de decisión	0.92645839	0.92534952	0.92524176	0.92409241
	Boosting	0.9711654	0.96903731	0.96974259	0.97029703
	Random Forest	0.95419148	0.95377764	0.95382812	0.94389439
	KNN	0.96650688	0.96327023	0.96431237	0.96369637
	Stacking	0.99174368	0.9920697	0.99185478	0.99339934
	MLP	0.99138836	0.99145331	0.99137693	0.98019802
Equipo: Banda Sensor: Vibración Reducción de Variables: PCA	SVM	0.84727273	0.84265904	0.84464579	0.84158416
	Árbol de decisión	0.76567158	0.75744157	0.76057611	0.75577558
	Boosting	0.81006872	0.80435516	0.80570089	0.7689769
	Random Forest	0.83414849	0.82627971	0.82813484	0.82508251
	KNN	0.81706841	0.80216813	0.80717199	0.8019802
	Stacking	0.85292454	0.85085168	0.85179365	0.85148515
	MLP	0.84280039	0.84076775	0.84098503	0.84818482
Equipo: Banda Sensor: Corriente Reducción de Variables: PCA	SVM	0.99082569	0.98983028	0.99022036	0.99009901
	Árbol de decisión	0.92999859	0.93105293	0.92902366	0.92739274
	Boosting	0.97339571	0.97198267	0.97236701	0.96369637
	Random Forest	0.95750671	0.95623113	0.95652386	0.96039604
	KNN	0.97087379	0.97099671	0.97063595	0.97029703
	Stacking	0.9836582	0.98335486	0.98348482	0.98019802
	MLP	0.99283185	0.99248073	0.99262622	0.99339934
Equipo: Banda Sensor: Sonido Reducción de Variables: PCA	SVM	0.79248347	0.79032787	0.78214292	0.78547855
	Árbol de decisión	0.7046898	0.70743021	0.70560102	0.7029703
	Boosting	0.73381153	0.73593256	0.73414189	0.73927393
	Random Forest	0.75028776	0.75469751	0.7478069	0.73267327
	KNN	0.75844392	0.76289464	0.75656108	0.75907591
	Stacking	0.78183194	0.78646682	0.78072846	0.78547855
	MLP	0.79260917	0.78996203	0.78204358	0.77227723

Tabla 14 Resultados de clasificación de fallas en banda transportadora aplicando PCA (continuación).

Equipo: Banda Sensor: Temperatura Reducción de Variables: PCA	SVM	0.88923275	0.88820434	0.88850309	0.88778878
	Árbol de decisión	0.70891873	0.70592018	0.70400047	0.70627063
	Boosting	0.82342611	0.81304023	0.81236795	0.81518152
	Random Forest	0.79304855	0.78696106	0.78486047	0.77227723
	KNN	0.84184451	0.82577411	0.82483093	0.82178218
	Stacking	0.88603264	0.88517408	0.8855473	0.88778878
	MLP	0.89154215	0.89036482	0.89016561	0.88118812

Tabla 15 Resultados de clasificación de fallas en banda transportadora aplicando eliminación hacia atrás.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Sensor: Todos Reducción de Variables: p-valor	SVM	1	1	1	1
	Árbol de decisión	0.98536333	0.9854987	0.98539877	0.98349835
	Boosting	0.9856321	0.98575027	0.98566153	0.98349835
	Random Forest	0.99381883	0.99333352	0.99351548	0.99339934
	KNN	0.96392496	0.96412527	0.96400991	0.96369637
	Stacking	0.99986532	0.99987421	0.99986913	1
	MLP	0.99073173	0.9906046	0.9906271	0.99339934
Equipo: Banda Sensor: Vibración Reducción de Variables: p-valor	SVM	0.90578878	0.90540108	0.9055511	0.90429043
	Árbol de decisión	0.89051328	0.89072068	0.88932721	0.88861386
	Boosting	0.90343024	0.90325093	0.90302189	0.90165017
	Random Forest	0.90135034	0.90076123	0.90067382	0.89907591
	KNN	0.84701683	0.83568453	0.83917731	0.8349835
	Stacking	0.91325389	0.91344671	0.91288394	0.91161716
	MLP	0.90343346	0.90254748	0.90242025	0.90148515
Equipo: Banda Sensor: Corriente Reducción de Variables: p-valor	SVM	0.99688474	0.99659864	0.99672565	0.99669967
	Árbol de decisión	0.97907807	0.97901044	0.97899003	0.97884488
	Boosting	0.97555872	0.97557763	0.97552796	0.97537954
	Random Forest	0.98386656	0.98420934	0.98391784	0.98376238
	KNN	0.99009901	0.99056604	0.99019019	0.99009901
	Stacking	0.99338673	0.99339109	0.9933886	0.99333333
	MLP	0.99271707	0.99288346	0.9927694	0.99270627

Tabla 15 Resultados de clasificación de fallas en banda transportadora aplicando eliminación hacia atrás (continuación).

Equipo: Banda Sensor: Sonido Reducción de Variables: p-valor	SVM	0.759169	0.76062317	0.75219584	0.75577558
	Árbol de decisión	0.68530833	0.68589676	0.68544132	0.669967
	Boosting	0.74008309	0.74347629	0.73938257	0.75577558
	Random Forest	0.77313297	0.7686098	0.75961424	0.76567657
	KNN	0.70297488	0.70473442	0.69476833	0.69966997
	Stacking	0.7683337	0.77183568	0.76569804	0.77227723
	MLP	0.7735396	0.77390093	0.76803476	0.75577558
Equipo: Banda Sensor: Temperatura Reducción de Variables: p-valor	SVM	0.93132451	0.93216532	0.93098363	0.93069307
	Árbol de decisión	0.87558136	0.87366292	0.87360856	0.87174917
	Boosting	0.91573708	0.91432855	0.91470928	0.91366337
	Random Forest	0.91978401	0.91885324	0.91900098	0.91815182
	KNN	0.8681769	0.86629483	0.86636801	0.86468647
	Stacking	0.94056263	0.94084328	0.94063113	0.94013201
	MLP	0.93073015	0.93076414	0.93025853	0.92983498

Tabla 16 Resultados de clasificación de fallas en banda transportadora sin reducción de variables.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Sensor: Todos Reducción de Variables: No	SVM	0.99009901	0.99056604	0.99019019	0.99009901
	Árbol de decisión	0.98899012	0.98838187	0.98858859	0.99009901
	Boosting	0.98584205	0.98495475	0.98522483	0.98349835
	Random Forest	0.9976024	0.99740672	0.99749206	0.99669967
	KNN	0.93806583	0.93346246	0.93490907	0.9339934
	Stacking	0.99979798	0.99981132	0.99980369	1
	MLP	0.99480489	0.99500662	0.99486768	0.99339934
Equipo: Banda Sensor: Vibración Reducción de Variables: No	SVM	0.88109234	0.87861869	0.87967363	0.87788779
	Árbol de decisión	0.90163124	0.90044687	0.89948535	0.89815182
	Boosting	0.89792664	0.89741182	0.89694487	0.89541254
	Random Forest	0.89259511	0.89213232	0.89226362	0.89066007
	KNN	0.80862695	0.79311015	0.79770004	0.79207921
	Stacking	0.90901994	0.90931612	0.90878161	0.90749175
	MLP	0.85886818	0.85858063	0.85839909	0.85722772

Tabla 16 Resultados de clasificación de fallas en banda transportadora sin reducción de variables (continuación).

Equipo: Banda Sensor: Corriente Reducción de Variables: No	SVM	0.99382716	0.99319728	0.99344831	0.99339934
	Árbol de decisión	0.9738185	0.97307205	0.97337852	0.9730363
	Boosting	0.98414304	0.98368221	0.98380361	0.98363036
	Random Forest	0.99290201	0.99239285	0.99259861	0.99254125
	KNN	0.98039216	0.98087537	0.98038462	0.98019802
	Stacking	0.99247193	0.99225645	0.99233935	0.99227723
	MLP	0.99182473	0.99176742	0.991754	0.99168317
Equipo: Banda Sensor: Sonido Reducción de Variables: No	SVM	0.78736453	0.7810819	0.77020465	0.77557756
	Árbol de decisión	0.79808503	0.76517052	0.75117191	0.75577558
	Boosting	0.72756085	0.72526432	0.72569025	0.73927393
	Random Forest	0.78682846	0.75826321	0.74919097	0.76237624
	KNN	0.70020948	0.7042048	0.69588796	0.69966997
	Stacking	0.80137288	0.79527005	0.78515077	0.79207921
	MLP	0.78577874	0.78280453	0.77641328	0.78547855
Equipo: Banda Sensor: Temperatura Reducción de Variables: No	SVM	0.93127352	0.93168626	0.93124838	0.93069307
	Árbol de decisión	0.86003051	0.86026652	0.85980175	0.85884488
	Boosting	0.90353892	0.90173902	0.90219373	0.9009571
	Random Forest	0.88758215	0.8854009	0.88624572	0.8849505
	KNN	0.86529625	0.86315018	0.86314461	0.86138614
	Stacking	0.9389526	0.93870813	0.93879364	0.93825083
	MLP	0.93415031	0.93385283	0.93364416	0.93316832

Tabla 17 Resultados de clasificación de fallas en válvula neumática aplicando PCA.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Válvula Sensor: Todos Reducción de Variables: PCA	SVM	0.99333333	0.9943662	0.99375303	0.99335548
	Árbol de decisión	0.93355985	0.93237284	0.93249395	0.9269103
	Boosting	0.98173909	0.98213348	0.98177279	0.96345515
	Random Forest	0.98828384	0.9888712	0.9884783	0.99003322
	KNN	0.9718254	0.9709984	0.97138774	0.97009967
	Stacking	0.99996491	0.99997183	0.99996812	1
	MLP	0.99295078	0.99393433	0.99334595	0.99335548
Equipo: Válvula Sensor: Vibración Reducción de Variables: PCA	SVM	0.96813772	0.96805835	0.96798021	0.96677741
	Árbol de decisión	0.86087359	0.86501513	0.86262293	0.85714286
	Boosting	0.85814771	0.86132434	0.85949244	0.87043189
	Random Forest	0.90781473	0.90920461	0.90804848	0.87375415
	KNN	0.9366977	0.94013456	0.93805037	0.93687708
	Stacking	0.97841543	0.97760915	0.97797635	0.98006645
	MLP	0.96157253	0.96166585	0.96143096	0.97674419

Tabla 17 Resultados de clasificación de fallas en válvula neumática aplicando PCA (continuación).

Equipo: Válvula Sensor: Corriente Reducción de Variables: PCA	SVM	0.99028655	0.9909331	0.99049972	0.99003322
	Árbol de decisión	0.96880785	0.96788858	0.96828182	0.96677741
	Boosting	0.96873819	0.96788858	0.96824287	0.96677741
	Random Forest	0.98123865	0.98133482	0.98121341	0.98006645
	KNN	0.98768308	0.98691524	0.98727379	0.98671096
	Stacking	0.99095205	0.99096127	0.99088076	0.99003322
	MLP	0.97879264	0.97878471	0.9781033	0.96345515
	SVM	0.96014031	0.95157041	0.95433554	0.95348837
Equipo: Válvula Sensor: Sonido Reducción de Variables: PCA	Árbol de decisión	0.8822229	0.8814932	0.88150309	0.87707641
	Boosting	0.92823974	0.92776776	0.92780827	0.92358804
	Random Forest	0.91597676	0.91525853	0.91512581	0.89700997
	KNN	0.93666932	0.93542353	0.93591973	0.93355482
	Stacking	0.9410516	0.93923091	0.93984066	0.93687708
	MLP	0.94610735	0.94419184	0.94463755	0.93355482
	SVM	0.99016555	0.99048667	0.99029856	0.99003322
	Árbol de decisión	0.94890568	0.94880963	0.94879264	0.94684385
Equipo: Válvula Sensor: Temperatura Reducción de Variables: PCA	Boosting	0.9857911	0.985821	0.98576274	0.98671096
	Random Forest	0.97763947	0.97768567	0.97757788	0.97674419
	KNN	0.98501946	0.98306909	0.98397229	0.9833887
	Stacking	0.9922212	0.99244821	0.99232411	0.99003322
	MLP	0.98679971	0.98631181	0.98645702	0.9833887

Tabla 18 Resultados de clasificación de fallas en válvula neumática aplicando eliminación hacia atrás.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Válvula Sensor: Todos Reducción de Variables: p-valor	SVM	0.99410618	0.99361167	0.99383091	0.99335548
	Árbol de decisión	0.99552437	0.99622774	0.99584656	0.99667774
	Boosting	0.99164738	0.99183042	0.9917232	0.99335548
	Random Forest	0.99522422	0.99561167	0.99540366	0.99335548
	KNN	0.99649123	0.9971831	0.99681165	0.99667774
	Stacking	0.99361167	0.99361167	0.99361167	0.99335548
	MLP	0.98983051	0.9915493	0.99046606	0.99003322
	SVM	0.97246303	0.97012072	0.97107245	0.97009967
Equipo: Válvula Sensor: Vibración Reducción de Variables: p-valor	Árbol de decisión	0.92141673	0.91969939	0.92047169	0.91694352
	Boosting	0.9653553	0.96583489	0.96547712	0.97342193
	Random Forest	0.94820559	0.94915121	0.94849453	0.94352159
	KNN	0.93219316	0.93219316	0.92755906	0.92358804
	Stacking	0.97613635	0.97689688	0.97645107	0.97009967
	MLP	0.95104124	0.95219006	0.95140198	0.95016611

Tabla 18 Resultados de clasificación de fallas en válvula neumática aplicando eliminación hacia atrás (continuación).

Equipo: Válvula Sensor: Presión Reducción de Variables: p-valor	SVM	0.99649123	0.9971831	0.99681165	0.99667774
	Árbol de decisión	0.9932361	0.99283929	0.99301522	0.99335548
	Boosting	0.99301281	0.99258929	0.99277833	0.99335548
	Random Forest	0.99364435	0.99331112	0.9934556	0.99335548
	KNN	0.99310345	0.9940581	0.99349799	0.99335548
	Stacking	0.99365079	0.99330357	0.99345517	0.99335548
	MLP	0.99579008	0.99559488	0.99549816	0.99335548
Equipo: Válvula Sensor: Sonido Reducción de Variables: p-valor	SVM	0.95478053	0.94875351	0.95077274	0.95016611
	Árbol de decisión	0.95238208	0.95386553	0.95241395	0.95681063
	Boosting	0.96722258	0.964906	0.96570432	0.96013289
	Random Forest	0.97078906	0.97195239	0.97116077	0.96677741
	KNN	0.94010243	0.93950318	0.93972788	0.93687708
	Stacking	0.97619721	0.97744376	0.9766448	0.97342193
	MLP	0.94582865	0.94542921	0.94528942	0.94684385
Equipo: Válvula Sensor: Temperatura Reducción de Variables: p-valor	SVM	0.99016555	0.99048667	0.99029856	0.99003322
	Árbol de decisión	0.98991461	0.98936819	0.9896045	0.99003322
	Boosting	0.9906993	0.99044855	0.9905211	0.99335548
	Random Forest	0.99126674	0.99070748	0.99094812	0.99003322
	KNN	0.99087302	0.99048667	0.99065796	0.99003322
	Stacking	0.99023542	0.98958694	0.98987746	0.99003322
	MLP	0.99901605	0.99916229	0.9990806	1

Tabla 19 Resultados de clasificación de fallas en válvula neumática sin reducción de variables.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Válvula Sensor: Todos Reducción de Variables: No	SVM	0.99016393	0.9915493	0.99064144	0.99003322
	Árbol de decisión	0.99551929	0.99626025	0.99585462	0.99667774
	Boosting	0.99361167	0.99361167	0.99361167	0.99335548
	Random Forest	0.99479229	0.99507596	0.99492366	0.99335548
	KNN	0.98034826	0.98234406	0.98095952	0.98006645
	Stacking	0.99361167	0.99361167	0.99361167	0.99335548
	MLP	0.98983051	0.99091792	0.99018634	0.99003322

**Tabla 19 Resultados de clasificación de fallas en válvula neumática sin reducción de variables
(continuación).**

Equipo: Válvula Sensor: Vibración Reducción de Variables: No	SVM	0.96706094	0.97032193	0.96807819	0.96677741
	Árbol de decisión	0.9420093	0.9367622	0.93837293	0.93687708
	Boosting	0.94203612	0.9366746	0.93823915	0.93023256
	Random Forest	0.95861808	0.95978018	0.95891489	0.95348837
	KNN	0.91254314	0.91488934	0.91345756	0.910299
	Stacking	0.98506691	0.98497233	0.9849888	0.9833887
	MLP	0.95385749	0.95120204	0.95237815	0.95016611
Equipo: Válvula Sensor: Presión Reducción de Variables: No	SVM	0.99310345	0.9940581	0.99349799	0.99335548
	Árbol de decisión	0.99046086	0.98973214	0.99007099	0.99003322
	Boosting	0.99046086	0.98973214	0.99007099	0.99003322
	Random Forest	0.99365079	0.99330357	0.99345517	0.99335548
	KNN	0.99028655	0.9909331	0.99049972	0.99003322
	Stacking	0.99365079	0.99330357	0.99345517	0.99335548
	MLP	0.99266165	0.99333583	0.99281766	0.99667774
Equipo: Válvula Sensor: Sonido Reducción de Variables: No	SVM	0.94995975	0.94281161	0.94527176	0.94352159
	Árbol de decisión	0.95927363	0.9606565	0.95972893	0.96013289
	Boosting	0.95797479	0.9589092	0.95827524	0.95681063
	Random Forest	0.97318629	0.97375887	0.97336515	0.97342193
	KNN	0.95065278	0.94732251	0.9487336	0.94684385
	Stacking	0.97299002	0.97390097	0.97329283	0.97674419
	MLP	0.95040906	0.94719936	0.94826965	0.94684385
Equipo: Válvula Sensor: Temperatura Reducción de Variables: No	SVM	0.99271764	0.9940581	0.99333208	0.99335548
	Árbol de decisión	0.99162766	0.99171194	0.99159651	0.99003322
	Boosting	0.98948649	0.98882802	0.98912566	0.99003322
	Random Forest	0.99011391	0.98950659	0.98977467	0.99003322
	KNN	0.98983809	0.99048667	0.9901321	0.99003322
	Stacking	0.9900805	0.98935927	0.98968886	0.99003322
	MLP	0.99888356	0.99874755	0.99880549	0.99335548

Tabla 20 Resultados de clasificación de fallas en cilindro neumático aplicando PCA.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Cilindro Sensor: Todos Reducción de Variables: PCA	SVM	0.9645936	0.96619107	0.96481481	0.96694215
	Árbol de decisión	0.61336795	0.60994501	0.60921682	0.62809917
	Boosting	0.85460093	0.85367697	0.85199358	0.90082645
	Random Forest	0.83969998	0.8378552	0.836686	0.83471074
	KNN	0.88188206	0.88235958	0.87994661	0.88429752
	Stacking	0.93704455	0.9375767	0.93684438	0.95041322
	MLP	0.92780642	0.928434	0.92690435	0.91735537
Equipo: Cilindro Sensor: Vibración Reducción de Variables: PCA	SVM	0.74662162	0.73832619	0.72881102	0.73553719
	Árbol de decisión	0.41487471	0.40237048	0.40016099	0.39669421
	Boosting	0.6106373	0.53430305	0.54996003	0.55371901
	Random Forest	0.57425843	0.58329433	0.56892866	0.5785124
	KNN	0.57771584	0.5978739	0.56661604	0.59504132
	Stacking	0.72437296	0.70453267	0.69855511	0.69421488
	MLP	0.6881161	0.67818868	0.67555448	0.69421488
Equipo: Cilindro Sensor: Presión Reducción de Variables: PCA	SVM	0.78827731	0.78802918	0.78712958	0.78512397
	Árbol de decisión	0.72441897	0.72030491	0.7179187	0.7107438
	Boosting	0.71166896	0.71866926	0.71295027	0.66942149
	Random Forest	0.74751043	0.75551	0.74967717	0.75206612
	KNN	0.7570157	0.76032972	0.75669919	0.76033058
	Stacking	0.81314651	0.80060164	0.79867691	0.78512397
	MLP	0.78278455	0.77387736	0.77261738	0.76033058
Equipo: Cilindro Sensor: Sonido Reducción de Variables: PCA	SVM	0.75876003	0.76432439	0.75954226	0.76033058
	Árbol de decisión	0.63270049	0.63667381	0.63443713	0.62809917
	Boosting	0.73001264	0.72167409	0.72376038	0.74380165
	Random Forest	0.73384075	0.74015321	0.73338172	0.68595041
	KNN	0.66039377	0.67358448	0.660528	0.66115702
	Stacking	0.77235743	0.77181245	0.77130915	0.75206612
	MLP	0.77460092	0.779213	0.77463437	0.76033058
Equipo: Cilindro Sensor: Sonido Reducción de Variables: PCA	SVM	0.75334101	0.75781074	0.75341617	0.75206612
	Árbol de decisión	0.70045788	0.70069178	0.70031897	0.70247934
	Boosting	0.67361735	0.67650998	0.67397905	0.66115702
	Random Forest	0.7145723	0.71622763	0.7070478	0.7107438
	KNN	0.71531532	0.70494022	0.69220384	0.70247934
	Stacking	0.79560253	0.78804262	0.78442199	0.78512397
	MLP	0.73377608	0.73077459	0.72654681	0.7107438

Tabla 21 Resultados de clasificación de fallas en cilindro neumático aplicando eliminación hacia atrás.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Cilindro Sensor: Todos Reducción de Variables: p-valor	SVM	0.96868192	0.96920821	0.96774143	0.96694215
	Árbol de decisión	0.9099213	0.90911384	0.90838314	0.89256198
	Boosting	0.9087867	0.90802147	0.90737879	0.91735537
	Random Forest	0.94398412	0.94061029	0.94103717	0.95867769
	KNN	0.89695946	0.88985074	0.8896716	0.89256198
	Stacking	0.98031117	0.98112264	0.98039744	0.99173554
	MLP	0.95934965	0.96016618	0.95935781	0.95867769
Equipo: Cilindro Sensor: Vibración Reducción de Variables: p-valor	SVM	0.65796703	0.63836567	0.64080834	0.63636364
	Árbol de decisión	0.5915495	0.5999515	0.59276371	0.61157025
	Boosting	0.67745112	0.61171385	0.62107443	0.68595041
	Random Forest	0.62785667	0.62974378	0.61800342	0.67768595
	KNN	0.63767784	0.60211294	0.61289588	0.60330579
	Stacking	0.74301196	0.7178974	0.71813446	0.73553719
	MLP	0.68552926	0.66575955	0.66616043	0.66115702
Equipo: Cilindro Sensor: Presión Reducción de Variables: p-valor	SVM	0.76913812	0.76937176	0.76430884	0.76859504
	Árbol de decisión	0.71990069	0.71850562	0.71560238	0.7107438
	Boosting	0.78666739	0.78513121	0.78474776	0.79338843
	Random Forest	0.80503645	0.80183867	0.79943059	0.79338843
	KNN	0.73789492	0.74119295	0.73898329	0.73553719
	Stacking	0.79417972	0.79536497	0.79409782	0.7768595
	MLP	0.7866504	0.78110149	0.77827126	0.80165289
Equipo: Cilindro Sensor: Sonido Reducción de Variables: p-valor	SVM	0.77199644	0.77801902	0.77242221	0.76859504
	Árbol de decisión	0.69412879	0.68050229	0.67502646	0.66942149
	Boosting	0.71144619	0.71056311	0.70865504	0.69421488
	Random Forest	0.73136508	0.73408696	0.72840563	0.72727273
	KNN	0.70217539	0.70690465	0.69026298	0.69421488
	Stacking	0.72426635	0.72905482	0.72607314	0.7107438
	MLP	0.80721607	0.8071324	0.80317623	0.80991736
Equipo: Cilindro Sensor: Temperatura Reducción de Variables: p-valor	SVM	0.82837074	0.83406647	0.8294621	0.82644628
	Árbol de decisión	0.8215074	0.81202384	0.81036404	0.80165289
	Boosting	0.86902521	0.86016862	0.85609656	0.87603306
	Random Forest	0.8576536	0.85317101	0.8530191	0.83471074
	KNN	0.80298799	0.79325513	0.78514038	0.78512397
	Stacking	0.86639886	0.86213869	0.86008364	0.8677686
	MLP	0.83632741	0.82451951	0.82084197	0.80991736

Tabla 22 Resultados de clasificación de fallas en cilindro neumático sin reducción de variables.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Cilindro Sensor: Todos Reducción de Variables: No	SVM	0.98214286	0.98435973	0.98279623	0.98347107
	Árbol de decisión	0.93284002	0.93191687	0.93131068	0.9338843
	Boosting	0.93971456	0.93907474	0.9385514	0.9338843
	Random Forest	0.9758433	0.97216266	0.97322904	0.96694215
	KNN	0.84301075	0.83535416	0.83158433	0.83471074
	Stacking	0.98928043	0.9891036	0.98904124	0.99173554
	MLP	0.96854933	0.97042926	0.96885967	0.97520661
Equipo: Cilindro Sensor: Vibración Reducción de Variables: No	SVM	0.72108606	0.71258177	0.70955629	0.7107438
	Árbol de decisión	0.63000269	0.55302918	0.56283474	0.57024793
	Boosting	0.67423446	0.59719171	0.60991806	0.5785124
	Random Forest	0.63362832	0.62890029	0.62166244	0.60330579
	KNN	0.60678897	0.61505564	0.60672114	0.61983471
	Stacking	0.74418734	0.72812411	0.72707162	0.70247934
	MLP	0.6930578	0.68141355	0.68085522	0.69421488
Equipo: Cilindro Sensor: Presión Reducción de Variables: No	SVM	0.81794613	0.82134935	0.81906447	0.81818182
	Árbol de decisión	0.72142384	0.71600923	0.71601933	0.7107438
	Boosting	0.80186141	0.79813567	0.79822028	0.80991736
	Random Forest	0.80859792	0.80453727	0.80375142	0.80165289
	KNN	0.79336328	0.7971558	0.79490321	0.79338843
	Stacking	0.81938916	0.81563629	0.81472113	0.80165289
	MLP	0.80365045	0.79584837	0.79459337	0.7768595
Equipo: Cilindro Sensor: Sonido Reducción de Variables: No	SVM	0.74641465	0.7512125	0.74453939	0.74380165
	Árbol de decisión	0.70275438	0.6946449	0.68272632	0.67768595
	Boosting	0.7059853	0.70336623	0.7032401	0.70247934
	Random Forest	0.73961139	0.74230619	0.73800495	0.74380165
	KNN	0.66606991	0.68417738	0.6615618	0.66942149
	Stacking	0.76757894	0.76931875	0.76728926	0.75206612
	MLP	0.79253717	0.79657089	0.79098885	0.80165289
Equipo: Cilindro Sensor: Temperatura Reducción de Variables: No	SVM	0.74981477	0.74827995	0.74067214	0.74380165
	Árbol de decisión	0.82313731	0.81279607	0.81094457	0.80991736
	Boosting	0.84476755	0.84315024	0.8397384	0.85123967
	Random Forest	0.86796482	0.86017191	0.85865036	0.85123967
	KNN	0.68219281	0.67862245	0.6703245	0.67768595
	Stacking	0.85524208	0.85346624	0.85057528	0.85123967
	MLP	0.75368354	0.7304193	0.723498	0.70247934

Con base en estas tablas aún no es posible determinar que algoritmo es el mejor o el más recomendado, ya que este algoritmo varía según el equipo que se está evaluando y el sensor que se utiliza para monitorear el equipo. Debido a esto, el siguiente paso es promediar, esto con el fin de generalizar (en lugar de enfocarse en casos específicos) y simplificar la lectura información, facilitando el poder determinar que algoritmo es mejor:

Tabla 23 Resultado promediado de clasificación de fallas en banda transportadora.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Reducción de Variables: PCA	SVM	0.90262959	0.90094644	0.89979411	0.89966997
	Árbol de decisión	0.80714742	0.80543888	0.80488861	0.80330033
	Boosting	0.86237349	0.85886959	0.85886406	0.85148515
	Random Forest	0.8578366	0.85558941	0.85423084	0.84686469
	KNN	0.8709475	0.86502076	0.86470246	0.86336634
	Stacking	0.8992382	0.89958343	0.8986818	0.89966997
	MLP	0.90223438	0.90100573	0.89943947	0.8950495
Equipo: Banda Reducción de Variables: p-valor	SVM	0.9186334	0.91895764	0.91709124	0.91749175
	Árbol de decisión	0.88316887	0.8829579	0.88255318	0.87853465
	Boosting	0.90408825	0.90447674	0.90366065	0.9059934
	Random Forest	0.91439054	0.91315342	0.91134447	0.9120132
	KNN	0.87443852	0.87228102	0.87090275	0.87062706
	Stacking	0.92308045	0.92387819	0.92249417	0.92347195
	MLP	0.9182304	0.91814012	0.91682201	0.91464026
Equipo: Banda Reducción de Variables: No	SVM	0.91673131	0.91503003	0.91295303	0.91353135
	Árbol de decisión	0.90451108	0.89746757	0.89448522	0.89518152
	Boosting	0.8998023	0.89861042	0.89877146	0.90055446
	Random Forest	0.91150203	0.9051192	0.9035582	0.90544554
	KNN	0.85851813	0.85496059	0.85440526	0.85346535
	Stacking	0.92832307	0.92707241	0.92497381	0.9260198
	MLP	0.91308537	0.91240241	0.91101564	0.91219142

Tabla 24 Resultado promediado de clasificación de fallas en válvula neumática.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Válvula Reducción de Variables: PCA	SVM	0.98041269	0.97908295	0.97937341	0.97873754
	Árbol de decisión	0.91887397	0.91911588	0.91873888	0.91495017
	Boosting	0.94453117	0.94498703	0.94461582	0.94219269
	Random Forest	0.95419069	0.95447097	0.95408878	0.94352159
	KNN	0.96357899	0.96330816	0.96332078	0.96212625
	Stacking	0.98052104	0.98004427	0.980198	0.97940199
	MLP	0.9732446	0.97297771	0.97279496	0.97009967
Equipo: Válvula Reducción de Variables: p-valor	SVM	0.9816013	0.98003114	0.98055726	0.98006645
	Árbol de decisión	0.97049478	0.97040003	0.97027039	0.97076412
	Boosting	0.98158747	0.98112183	0.98124081	0.98272425
	Random Forest	0.97982599	0.98014677	0.97989254	0.97740864
	KNN	0.97055266	0.97068484	0.96965091	0.96810631
	Stacking	0.98596629	0.98616856	0.98600803	0.98405316
	MLP	0.9763013	0.97678515	0.97634724	0.97607973
Equipo: Válvula Reducción de Variables: No	SVM	0.97860114	0.97855981	0.97816429	0.97740864
	Árbol de decisión	0.97577815	0.97502461	0.9751248	0.97475083
	Boosting	0.97471399	0.97355112	0.97386454	0.97209302
	Random Forest	0.98207227	0.98228503	0.98208671	0.9807309
	KNN	0.96473376	0.96519514	0.9647565	0.96345515
	Stacking	0.98707998	0.98702956	0.98700746	0.98737542
	MLP	0.97712846	0.97628054	0.97649146	0.97541528

Tabla 25 Resultado promediado de clasificación de fallas en cilindro neumático.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Cilindro Reducción de Variables: PCA	SVM	0.80231871	0.80293631	0.79874277	0.8
	Árbol de decisión	0.617164	0.6139972	0.61241052	0.61322314
	Boosting	0.71610744	0.70096667	0.70252866	0.70578512
	Random Forest	0.72197638	0.72660807	0.71914427	0.71239669
	KNN	0.71846454	0.72381758	0.71119873	0.72066116
	Stacking	0.8085048	0.80051322	0.79796151	0.79338843
	MLP	0.78141681	0.77809753	0.77525148	0.76859504

Tabla 25 Resultado promediado de clasificación de fallas en cilindro neumático (continuación).

Equipo: Cilindro Reducción de Variables: p-valor	SVM	0.79923085	0.79780623	0.79494859	0.79338843
	Árbol de decisión	0.74740154	0.74401942	0.74042794	0.73719008
	Boosting	0.79067533	0.77511965	0.77559052	0.79338843
	Random Forest	0.79317918	0.79189014	0.78797918	0.79834711
	KNN	0.75553912	0.74666328	0.74339083	0.74214876
	Stacking	0.82163361	0.8171157	0.8157573	0.81652893
	MLP	0.81501456	0.80773583	0.80556154	0.80826446
Equipo: Cilindro Reducción de Variables: No	SVM	0.80348089	0.80355666	0.7993257	0.8
	Árbol de decisión	0.76203165	0.74167925	0.74076713	0.74049587
	Boosting	0.79331265	0.77618372	0.77793365	0.77520661
	Random Forest	0.80512915	0.80161566	0.79905964	0.79338843
	KNN	0.71828514	0.72207309	0.71301899	0.71900826
	Stacking	0.83513559	0.83112979	0.82973971	0.81983471
	MLP	0.80229566	0.79493627	0.79175902	0.79008264

Se puede observar que el modelo de Stacking aparece casi en cada una de las tablas como el promedio más alto. Pero, todavía es necesario considerar otro factor, la desviación estándar; puesto que es necesario, de igual forma, estimar la estabilidad de los algoritmos en cada uno de los casos. Entonces lo que se busca realmente es el algoritmo que tenga la mayor tasa de predicción y la menor desviación estándar. Esto se consigue restando la desviación estándar de los resultados obtenidos en las tablas 14 a 22, a los promedios mostrados en las tablas anteriores (tablas 23 a 25):

Tabla 26 Valores obtenido al restar la desviación estándar al resultado promediado de clasificación de fallas en banda transportadora.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Reducción de Variables: PCA	SVM	0.81405513	0.81108295	0.80765219	0.80813116
	Árbol de decisión	0.69401674	0.69145329	0.69098686	0.68958013
	Boosting	0.75637529	0.75266558	0.75195195	0.74259926
	Random Forest	0.76357133	0.76135861	0.75779072	0.74515239
	KNN	0.77670864	0.76909845	0.76757099	0.76605976
	Stacking	0.81009285	0.81162159	0.80902027	0.81195939
	MLP	0.81304254	0.81069932	0.806675	0.80233431

Tabla 26 Valores obtenido al restar la desviación estándar al resultado promediado de clasificación de fallas en banda transportadora (continuación).

Equipo: Banda Reducción de Variables: p-valor	SVM	0.82052793	0.8214589	0.81618018	0.8180154
	Árbol de decisión	0.7618196	0.76179638	0.76125163	0.75132654
	Boosting	0.80561077	0.8074195	0.80492005	0.81452682
	Random Forest	0.82594761	0.82293604	0.81751815	0.82064702
	KNN	0.76088664	0.75840201	0.75369691	0.75501183
	Stacking	0.82928862	0.83151885	0.82760049	0.83125604
	MLP	0.82862665	0.82863438	0.82497867	0.8172503
Equipo: Banda Reducción de Variables: No	SVM	0.83078952	0.82648467	0.82039979	0.82292542
	Árbol de decisión	0.82505426	0.80680783	0.79850834	0.80053231
	Boosting	0.79468238	0.79283632	0.79308702	0.80078058
	Random Forest	0.82416309	0.80732116	0.80230567	0.80924203
	KNN	0.74808963	0.74469168	0.74174287	0.74162846
	Stacking	0.8479712	0.84436443	0.83816758	0.8419126
	MLP	0.82299073	0.82119553	0.81754346	0.82214858

Tabla 27 Valores obtenido al restar la desviación estándar al resultado promediado de clasificación de fallas en válvula neumática.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Válvula Reducción de Variables: PCA	SVM	0.96523706	0.96049175	0.96200514	0.96106387
	Árbol de decisión	0.87327157	0.87501769	0.8737877	0.86852167
	Boosting	0.89114506	0.89286289	0.89179435	0.89600253
	Random Forest	0.91528435	0.91564242	0.9150121	0.88959242
	KNN	0.93830332	0.93921721	0.93854823	0.93675932
	Stacking	0.95714352	0.95584817	0.95629365	0.95460717
	MLP	0.95403833	0.95293223	0.95305189	0.94696281
Equipo: Válvula Reducción de Variables: p-valor	SVM	0.96389333	0.95963233	0.96111268	0.9604117
	Árbol de decisión	0.93787001	0.93733068	0.93731572	0.93668869
	Boosting	0.96758213	0.96671864	0.96693177	0.96742742
	Random Forest	0.9595747	0.96044992	0.95983726	0.95544652
	KNN	0.93895783	0.93869076	0.93642894	0.93313569
	Stacking	0.97691378	0.97780077	0.97724371	0.97268928
	MLP	0.95058426	0.95099447	0.95051239	0.95062594

Tabla 27 Valores obtenido al restar la desviación estándar al resultado promediado de clasificación de fallas en válvula neumática (continuación).

Equipo: Válvula Reducción de Variables: No	SVM	0.95925692	0.95622756	0.95692452	0.95544652
	Árbol de decisión	0.9519599	0.94945027	0.95006405	0.94927536
	Boosting	0.95141508	0.94869203	0.94938141	0.94433675
	Random Forest	0.9663434	0.96714956	0.96654537	0.96340411
	KNN	0.9313829	0.93185487	0.93137429	0.92884939
	Stacking	0.97845703	0.97889465	0.97857324	0.98017293
	MLP	0.95404507	0.95135792	0.95235677	0.95070964

Tabla 28 Valores obtenido al restar la desviación estándar al resultado promediado de clasificación de fallas en cilindro neumático.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Cilindro Reducción de Variables: PCA	SVM	0.71022007	0.70996319	0.70361599	0.70497665
	Árbol de decisión	0.49507976	0.48737394	0.48551972	0.48593968
	Boosting	0.62621464	0.58647048	0.59414113	0.57740633
	Random Forest	0.62641617	0.63435665	0.62211067	0.61863965
	KNN	0.60512456	0.61750409	0.59460261	0.61111415
	Stacking	0.72929735	0.7153987	0.71126626	0.698078
	MLP	0.69134336	0.68476834	0.68128666	0.68035473
Equipo: Cilindro Reducción de Variables: p-valor	SVM	0.68613183	0.67810369	0.67635896	0.67391048
	Árbol de decisión	0.62514311	0.62430965	0.61815939	0.62609529
	Boosting	0.6916778	0.65694028	0.66125679	0.68901379
	Random Forest	0.67256389	0.67398695	0.66527136	0.69033845
	KNN	0.65641622	0.64038661	0.63969413	0.63480615
	Stacking	0.71723299	0.70872125	0.70722873	0.70180461
	MLP	0.71637552	0.70239861	0.70059737	0.70301011
Equipo: Cilindro Reducción de Variables: No	SVM	0.69735025	0.69506912	0.68915382	0.69014169
	Árbol de decisión	0.64423539	0.60051623	0.60213737	0.60262541
	Boosting	0.68612136	0.64499643	0.65160293	0.6372371
	Random Forest	0.67589217	0.67328809	0.66762003	0.6591068
	KNN	0.62126235	0.63090457	0.6174514	0.6282872
	Stacking	0.73863645	0.73097309	0.72914466	0.70889372
	MLP	0.69978542	0.68556175	0.68183577	0.67667566

Hasta este punto, el modelo stacking sigue teniendo ventaja sobre los demás modelos. Por último, se promedian los valores de todas estas tablas para simplificar la lectura de los datos:

Tabla 29 Generalización de los resultados obtenidos en la clasificación de fallas.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Reducción de Variables: PCA	SVM	0.82983742	0.8271793	0.82442444	0.82472389
	Árbol de decisión	0.68745602	0.68461497	0.68343143	0.68134716
	Boosting	0.75791166	0.74399965	0.74596248	0.73866938
	Random Forest	0.76842395	0.77045256	0.76497116	0.75112815
	KNN	0.77337884	0.77527325	0.76690728	0.77131108
	Stacking	0.8321779	0.82762282	0.82552673	0.82154818
	MLP	0.81947474	0.8161333	0.81367118	0.80988395
Reducción de Variables: p-valor	SVM	0.8235177	0.8197316	0.8178839	0.8174459
	Árbol de decisión	0.7749442	0.7744789	0.7722423	0.7713702
	Boosting	0.8216236	0.8103595	0.8110362	0.823656
	Random Forest	0.8193621	0.8191243	0.8142089	0.822144
	KNN	0.7854202	0.7791598	0.7766067	0.7743179
	Stacking	0.8411451	0.839347	0.8373576	0.83525
	MLP	0.8318621	0.8273425	0.8253628	0.8236288
Reducción de Variables: No	SVM	0.8291322	0.8259271	0.8221594	0.8228379
	Árbol de decisión	0.8070832	0.7855914	0.7835699	0.7841444
	Boosting	0.8107396	0.7955083	0.7980238	0.7941181
	Random Forest	0.8221329	0.8159196	0.812157	0.8105843
	KNN	0.7669116	0.7691504	0.7635229	0.766255
	Stacking	0.8550216	0.8514107	0.8486285	0.8436598
	MLP	0.8256071	0.8193717	0.8172453	0.8165113

A partir de estos valores ya es posible seleccionar el algoritmo que mejor se ajusta al entorno industrial de manera general:

1. Stacking
2. SVM
3. MLP
4. Random Forest
5. Boosting
6. KNN
7. Árbol de decisión

Además de esto, que fue el tema principal de la investigación, fue posible obtener otros resultados interesantes. Primeramente, de todos los parámetros monitoreados, con los que se obtuvieron los mejores resultados para todos los componentes analizados fueron los fluidos (corriente eléctrica, presión de aire) y la temperatura. En el caso del cilindro neumático, los resultados obtenidos con cada uno de los parámetros fueron bastante bajos, por lo que se analizaron combinaciones de estos para incrementar el porcentaje de predicciones correctas:

Tabla 30 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de vibración y presión.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Sensor: Vibración, presión Reducción de Variables: PCA	SVM	0.91006624	0.90965486	0.90705578	0.90909091
	Árbol de decisión	0.57976235	0.57684224	0.57602235	0.59504132
	Boosting	0.59390387	0.58911441	0.58759413	0.61157025
	Random Forest	0.79328049	0.78697881	0.78492418	0.79338843
	KNN	0.73179367	0.70744041	0.69651162	0.7107438
	Stacking	0.91347548	0.91416742	0.91190103	0.91735537
	MLP	0.87774365	0.87498797	0.87299796	0.89256198
Sensor: Vibración, presión Reducción de Variables: p-valor	SVM	0.94925595	0.95103955	0.94946914	0.95041322
	Árbol de decisión	0.84780348	0.84667372	0.84616085	0.85950413
	Boosting	0.91406009	0.91445409	0.9129016	0.92561983
	Random Forest	0.84607718	0.84643535	0.84377943	0.84297521
	KNN	0.73606754	0.7228551	0.71943891	0.71900826
	Stacking	0.94360288	0.94501513	0.94357115	0.95041322
	MLP	0.90052771	0.90107546	0.89934381	0.8677686
Sensor: Vibración, presión Reducción de Variables: No	SVM	0.92542941	0.92627265	0.92433916	0.92561983
	Árbol de decisión	0.8667203	0.86325598	0.86370786	0.87603306
	Boosting	0.93520331	0.93158743	0.93249309	0.95041322
	Random Forest	0.92316404	0.92172701	0.92169738	0.90909091
	KNN	0.68486798	0.67673321	0.67224026	0.67768595
	Stacking	0.95390567	0.95001795	0.95111655	0.95867769
	MLP	0.89462287	0.89375141	0.89195518	0.90082645

Tabla 31 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de vibración y sonido.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Sensor: Vibración, sonido Reducción de Variables: PCA	SVM	0.8412037	0.84399203	0.84236372	0.84297521
	Árbol de decisión	0.46215959	0.46034805	0.45915716	0.44628099
	Boosting	0.73267934	0.72016665	0.71960411	0.69421488
	Random Forest	0.69416366	0.69081754	0.6900009	0.68595041
	KNN	0.74929723	0.74435108	0.73165682	0.74380165
	Stacking	0.85044364	0.85190851	0.84970039	0.84297521
	MLP	0.82963672	0.82955062	0.82677017	0.84297521
Sensor: Vibración, sonido Reducción de Variables: p-valor	SVM	0.89663125	0.89605421	0.8955313	0.89256198
	Árbol de decisión	0.78393949	0.78215505	0.77242413	0.7768595
	Boosting	0.89215841	0.87909984	0.88120878	0.85950413
	Random Forest	0.86889294	0.87202835	0.86896789	0.81818182
	KNN	0.74068593	0.72725393	0.69963229	0.71900826
	Stacking	0.92143607	0.91113589	0.91368804	0.87603306
	MLP	0.90154797	0.90232912	0.90084054	0.88429752
Sensor: Vibración, sonido Reducción de Variables: No	SVM	0.88516114	0.88692759	0.88447242	0.88429752
	Árbol de decisión	0.78948661	0.76018949	0.76777955	0.74380165
	Boosting	0.89961642	0.89873308	0.89757681	0.90082645
	Random Forest	0.88527777	0.88296347	0.88124758	0.89256198
	KNN	0.64155421	0.65644033	0.64535017	0.66115702
	Stacking	0.93076861	0.92743439	0.92678416	0.95041322
	MLP	0.8704077	0.87113505	0.86788189	0.8677686

Tabla 32 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de vibración y temperatura.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Sensor: Vibración, temperatura Reducción de Variables: PCA	SVM	0.89747024	0.89799985	0.8971012	0.90082645
	Árbol de decisión	0.56015458	0.56453624	0.55732869	0.58677686
	Boosting	0.81624529	0.80904805	0.80811167	0.80991736
	Random Forest	0.81293418	0.81051498	0.80965684	0.80165289
	KNN	0.76035076	0.75346831	0.7530432	0.76033058
	Stacking	0.8891735	0.88948088	0.88840149	0.89256198
	MLP	0.87333579	0.87148752	0.86963419	0.89256198

Tabla 32 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de vibración y temperatura (continuación).

Sensor: Vibración, temperatura Reducción de Variables: p-valor	SVM	0.91338112	0.91364012	0.91337844	0.91735537
	Árbol de decisión	0.91159895	0.90714245	0.90799933	0.91735537
	Boosting	0.95970994	0.95822985	0.95823538	0.95041322
	Random Forest	0.90618226	0.90413151	0.90323017	0.88429752
	KNN	0.79851903	0.79795473	0.79611431	0.80165289
	Stacking	0.95444592	0.95479839	0.95435418	0.95041322
	MLP	0.90036443	0.90085053	0.89901614	0.87603306
Sensor: Vibración, temperatura Reducción de Variables: No	SVM	0.90296397	0.9020791	0.89911127	0.90082645
	Árbol de decisión	0.88699279	0.88313012	0.88337124	0.90082645
	Boosting	0.96050098	0.95798989	0.95855205	0.95867769
	Random Forest	0.92721912	0.92720881	0.92658027	0.92561983
	KNN	0.75736734	0.75395706	0.75262333	0.76033058
	Stacking	0.95850171	0.95738411	0.95757125	0.97520661
	MLP	0.8916021	0.89135988	0.88947941	0.87603306

Tabla 33 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de presión y sonido.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Sensor: Presión, sonido Reducción de Variables: PCA	SVM	0.84636422	0.85068426	0.8458454	0.84297521
	Árbol de decisión	0.82056592	0.81129399	0.81144943	0.80991736
	Boosting	0.84648965	0.84632031	0.84216149	0.82644628
	Random Forest	0.84062068	0.8399452	0.83462208	0.85123967
	KNN	0.8203994	0.82493985	0.82110273	0.81818182
	Stacking	0.83731397	0.83968419	0.83543295	0.82644628
	MLP	0.84303934	0.8448624	0.83953788	0.85123967
Sensor: Presión, sonido Reducción de Variables: p-valor	SVM	0.87877104	0.88098729	0.87636059	0.87603306
	Árbol de decisión	0.82817345	0.82790379	0.82716929	0.82644628
	Boosting	0.88082546	0.88150519	0.88029045	0.87603306
	Random Forest	0.82663029	0.83104294	0.82783683	0.82644628
	KNN	0.81714286	0.82038123	0.81408292	0.80991736
	Stacking	0.91655576	0.91920821	0.91692575	0.92561983
	MLP	0.89463037	0.89474303	0.892582	0.90082645

Tabla 33 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de presión y sonido (continuación).

Sensor: Presión, sonido Reducción de Variables: No	SVM	0.90502451	0.90411873	0.90386223	0.90082645
	Árbol de decisión	0.85623901	0.85653075	0.85417762	0.85950413
	Boosting	0.85068671	0.85115685	0.84888798	0.85950413
	Random Forest	0.87017371	0.87192712	0.87043779	0.87603306
	KNN	0.82161399	0.82746823	0.82123896	0.81818182
	Stacking	0.90247062	0.90556452	0.90349821	0.90909091
	MLP	0.90163506	0.90041648	0.89919199	0.89256198

Tabla 34 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de presión y temperatura.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Sensor: Presión, temperatura Reducción de Variables: PCA	SVM	0.82185442	0.81222272	0.81316653	0.80991736
	Árbol de decisión	0.72740319	0.71827102	0.71721702	0.7107438
	Boosting	0.74301961	0.72840693	0.72823088	0.67768595
	Random Forest	0.77149687	0.75905895	0.7548056	0.7768595
	KNN	0.80572917	0.80432739	0.79642523	0.80165289
	Stacking	0.84751594	0.82930211	0.82932412	0.81818182
	MLP	0.79494687	0.77988702	0.7791843	0.7768595
Sensor: Presión, temperatura Reducción de Variables: p-valor	SVM	0.84930297	0.84807128	0.84711174	0.84297521
	Árbol de decisión	0.80602972	0.80342112	0.79980272	0.80165289
	Boosting	0.84232449	0.8414283	0.8388027	0.82644628
	Random Forest	0.84409255	0.84246099	0.84042296	0.80165289
	KNN	0.78392783	0.78208888	0.77889722	0.7768595
	Stacking	0.84473341	0.84540595	0.84400351	0.81818182
	MLP	0.82200296	0.81205664	0.81082398	0.80165289
Sensor: Presión, temperatura Reducción de Variables: No	SVM	0.87069896	0.85914354	0.86097832	0.85950413
	Árbol de decisión	0.84497841	0.83355788	0.82894559	0.82644628
	Boosting	0.87777579	0.87733542	0.87538392	0.85950413
	Random Forest	0.88436916	0.87815898	0.87666252	0.8677686
	KNN	0.8103588	0.81328483	0.81159472	0.80991736
	Stacking	0.9044793	0.90209104	0.90105334	0.89256198
	MLP	0.84355775	0.82744295	0.82663815	0.84297521

Tabla 35 Resultados de clasificación de fallas en cilindro neumático mediante la combinación de los sensores de sonido y temperatura.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Sensor: Sonido, temperatura Reducción de Variables: PCA	SVM	0.84330837	0.84864463	0.84535329	0.84297521
	Árbol de decisión	0.8348686	0.83101718	0.82929219	0.84297521
	Boosting	0.85138132	0.85289928	0.84979736	0.83471074
	Random Forest	0.83988955	0.83916309	0.83679609	0.85123967
	KNN	0.79534632	0.8007463	0.7939948	0.79338843
	Stacking	0.83148844	0.83239802	0.83103416	0.84297521
	MLP	0.85439923	0.85045379	0.85034698	0.85123967
Sensor: Sonido, temperatura Reducción de Variables: p-valor	SVM	0.86761084	0.87186067	0.86908832	0.8677686
	Árbol de decisión	0.8531566	0.85188999	0.85044108	0.83471074
	Boosting	0.85704335	0.85707328	0.85598786	0.85123967
	Random Forest	0.89196058	0.89077807	0.89022984	0.88429752
	KNN	0.8351409	0.84106888	0.83697575	0.83471074
	Stacking	0.8934328	0.89280886	0.89235907	0.90082645
	MLP	0.90044681	0.89959762	0.8983027	0.89256198
Sensor: Sonido, temperatura Reducción de Variables: No	SVM	0.85945743	0.86330739	0.85970844	0.85950413
	Árbol de decisión	0.88359503	0.8849594	0.88378793	0.8677686
	Boosting	0.90721247	0.90710749	0.9064493	0.90909091
	Random Forest	0.89368982	0.89255705	0.89211506	0.88429752
	KNN	0.81341182	0.81589781	0.80936287	0.80991736
	Stacking	0.89358995	0.89307768	0.89230298	0.88429752
	MLP	0.87802567	0.87410219	0.87370446	0.87603306

Con la información de estas tablas, se obtienen que las mejores combinaciones son aquellas en las que se incluye el sensor de vibración, resaltando las combinaciones vibración-temperatura y vibración-presión.

Otro punto que evaluar es si resulta conveniente reducir la dimensionalidad mediante la eliminación hacia atrás, PCA o no hacer reducción. Para esto, a los resultados obtenidos mediante reducción de variables se le restan los obtenidos sin este paso de la siguiente forma (tabla 29):

Tabla 36 Comparación entre resultados obtenidos sin reducción de variables y los obtenidos mediante eliminación hacia atrás.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Todas las variables Vs Reducción por p-valor	SVM	-0.0056145	-0.0061955	-0.0042755	-0.005392
	Árbol de decisión	-0.032139	-0.0111125	-0.0113276	-0.0127742
	Boosting	0.010884	0.0148512	0.0130124	0.0295379
	Random Forest	-0.0027708	0.0032047	0.0020519	0.0115597
	KNN	0.0185086	0.0100094	0.0130838	0.0080629
	Stacking	-0.0138765	-0.0120637	-0.0112709	-0.0084098
	MLP	0.006255	0.0079708	0.0081175	0.0071175

Tabla 37 Comparación entre resultados obtenidos sin reducción de variables y los obtenidos mediante reducción por PCA.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Todas las variables Vs Reducción por PCA	SVM	0.00070522	0.0012522	0.00226504	0.00188599
	Árbol de decisión	-0.1196272	-0.1009764	-0.1001385	-0.1027972
	Boosting	-0.0528279	-0.0515086	-0.0520613	-0.0554487
	Random Forest	-0.053709	-0.045467	-0.0471858	-0.0594562
	KNN	0.00646724	0.00612285	0.00338438	0.00505608
	Stacking	-0.0228437	-0.0237879	-0.0231018	-0.0221116
	MLP	-0.0061324	-0.0032384	-0.0035741	-0.0066274

De este modo, si el valor es negativo significa que el algoritmo funciona de mejor manera al utilizar todas las variables. Se observa que el algoritmo Stacking, tienen una mayor tasa de predicción cuando no se lleva a cabo la selección de variables. La diferencia entre realizar eliminación hacia atrás o no realizar la reducción es bastante baja, siendo el algoritmo de Boosting el más beneficiado con un 2.95%. Mientras que al hacer la reducción por PCA la eficacia en el diagnóstico de fallas si se ve afectada, dado que la tasa de aciertos en el algoritmo árbol de decisión se reduce un 10%, reduciendo también la tasa de aciertos de los algoritmos de ensamble, aunque hay una muy leve mejoría en los algoritmos de SVM y KNN.

Otro factor a considerar es la cantidad de datos disponibles en la industria. Normalmente, en una fábrica se pretende diagnosticar fallas durante mucho tiempo con muy poco entrenamiento. Por lo que la cantidad de datos de entrenamiento y los datos de prueba se modifican en esta ocasión, en lugar de asignar un 80% de los datos para entrenamiento y 20% para prueba (80-20), se utilizan un 50% y 20% de los datos para entrenamiento y un 50% y 80% para datos de prueba, respectivamente.

Los resultados del primer caso (50-50) son los siguientes:

Tabla 38 Resultados de clasificación de fallas de la banda transportadora al dividir el conjunto de datos 50% para entrenamiento y 50% de prueba.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Sensor: Todos Reducción de Variables: PCA	SVM	0.98920307	0.98920307	0.98920307	0.98944591
	Árbol de decisión	0.92590869	0.9250869	0.92482264	0.91952507
	Boosting	0.96368098	0.96330981	0.9632168	0.95382586
	Random Forest	0.95860413	0.95805905	0.95812659	0.95778364
	KNN	0.96455976	0.96273356	0.96265799	0.96306069
	Stacking	0.98655176	0.98655536	0.9865446	0.98812665
	MLP9	0.98512731	0.98513837	0.98511781	0.99076517
Equipo: Banda Sensor: Todos Reducción de Variables: p-valor	SVM	0.9879565	0.9878862	0.9878866	0.9881266
	Árbol de decisión	0.9731735	0.9727262	0.9727852	0.974934
	Boosting	0.9733808	0.9729465	0.9730012	0.969657
	Random Forest	0.9897532	0.989718	0.9897227	0.9907652
	KNN	0.9565533	0.9556808	0.9556017	0.9564644
	Stacking	0.989145	0.9890363	0.989039	0.9894459
	MLP	0.9812372	0.9811726	0.9811756	0.9828496
Equipo: Banda Sensor: Todos Reducción de Variables: No	SVM	0.9919444	0.9918804	0.9919016	0.9920844
	Árbol de decisión	0.9768214	0.9768333	0.9766922	0.9828496
	Boosting	0.9906577	0.9907405	0.9906706	0.9828496
	Random Forest	0.9930542	0.9931163	0.9930237	0.9934037
	KNN	0.9497727	0.9482353	0.9483465	0.9485488
	Stacking	0.9970447	0.9970644	0.9970491	0.9934037
	MLP	0.9902298	0.9902786	0.9902485	0.9868074

Y los resultados obtenidos mediante la relación 20-80, son los siguientes:

Tabla 39 Resultados de clasificación de fallas de la banda transportadora al dividir el conjunto de datos 20% para entrenamiento y 80% de prueba.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Sensor: Todos Reducción de Variables: PCA	SVM	0.97501383	0.97465986	0.97473442	0.97524752
	Árbol de decisión	0.88022526	0.87744785	0.87800136	0.86716172
	Boosting	0.94461142	0.9435165	0.94345171	0.94389439
	Random Forest	0.94194276	0.9406301	0.94047072	0.92409241
	KNN	0.94453772	0.9405839	0.94057898	0.94059406
	Stacking	0.97719342	0.97709399	0.97712246	0.97854785
	MLP	0.97604355	0.97602596	0.97601589	0.97689769

Tabla 39 Resultados de clasificación de fallas de la banda transportadora al dividir el conjunto de datos 20% para entrenamiento y 80% de prueba (continuación).

Equipo: Banda Sensor: Todos Reducción de Variables: p-valor	SVM	0.9756051	0.9756122	0.9755954	0.9760726
	Árbol de decisión	0.9639539	0.9636019	0.9636691	0.9661716
	Boosting	0.9767826	0.9765475	0.9766085	0.9735974
	Random Forest	0.9805334	0.9801532	0.9802045	0.9834983
	KNN	0.9469485	0.9428231	0.9427071	0.9430693
	Stacking	0.9852681	0.9852747	0.9852512	0.9892739
	MLP	0.9616258	0.9614407	0.9614507	0.9570957
Equipo: Banda Sensor: Todos Reducción de Variables: No	SVM	0.9757703	0.9755669	0.9756105	0.9760726
	Árbol de decisión	0.9648022	0.9640953	0.9642587	0.9686469
	Boosting	0.9626776	0.9619916	0.9621439	0.9645215
	Random Forest	0.988339	0.9882396	0.9881226	0.9859736
	KNN	0.9360339	0.9305839	0.9306286	0.929868
	Stacking	0.9890712	0.988952	0.9889754	0.9892739
	MLP	0.977057	0.977038	0.9770319	0.9777228

En una fábrica, los equipos deberían estar funcionando correctamente la mayor parte del tiempo, por lo que gran parte de los datos deben pertenecer a esta categoría. Por esto, ahora se reduce el número de muestras de fallas, utilizando solo el 10% y el 5%, mientras que las muestras de buen funcionamiento se utilizan en su totalidad. Las siguientes tablas muestran los resultados obtenidos:

Tabla 40 Resultados de clasificación de fallas de la banda transportadora utilizando el 10% de las muestras de fallas.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Sensor: Todos Reducción de Variables: PCA Fallas=10%	SVM	0.96969697	0.97222222	0.96963423	0.99173554
	Árbol de decisión	0.90264652	0.97289226	0.9335648	0.96694215
	Boosting	0.99261291	0.9708771	0.98045011	0.99173554
	Random Forest	0.94922478	0.81211111	0.86538957	0.97520661
	KNN	0.92352941	0.85	0.88073916	0.95867769
	Stacking	0.99671329	0.99688889	0.99665465	1
	MLP	0.98269814	0.98267508	0.98188609	1

Tabla 40 Resultados de clasificación de fallas de la banda transportadora utilizando el 10% de las muestras de fallas (continuación).

Equipo: Banda Sensor: Todos Reducción de Variables: p-valor Fallas=10%	SVM	0.974359	0.9666667	0.9691228	0.9917355
	Árbol de decisión	0.9558407	0.9316751	0.9424131	0.9752066
	Boosting	0.9718087	0.9379882	0.9534387	0.9834711
	Random Forest	0.9959562	0.9917222	0.9935629	0.9917355
	KNN	0.9423077	0.8277778	0.874246	0.9504132
	Stacking	0.9997436	0.9996667	0.9996912	1
	MLP	0.9884816	0.9852323	0.9862603	0.9917355
Equipo: Banda Sensor: Todos Reducción de Variables: No Fallas=10%	SVM	1	1	1	1
	Árbol de decisión	0.974359	0.9666667	0.9691228	0.9917355
	Boosting	0.974359	0.9666667	0.9691228	0.9917355
	Random Forest	0.9995333	0.9953333	0.9973094	0.9917355
	KNN	0.9779874	0.7777778	0.8472037	0.9421488
	Stacking	0.9889744	0.9856667	0.9867228	1
	MLP	0.9977121	0.9964444	0.996944	1

Tabla 41 Resultados de clasificación de fallas de la banda transportadora utilizando el 5% de las muestras de fallas.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Sensor: Todos Reducción de Variables: PCA Fallas=5%	SVM	1	1	1	1
	Árbol de decisión	0.66144199	0.66465385	0.64712786	0.93693694
	Boosting	0.99035931	0.79333333	0.84252763	0.97297297
	Random Forest	0.51094428	0.44433333	0.46129987	0.93693694
	KNN	0.9968254	0.93333333	0.96136807	0.99099099
	Stacking	1	1	1	1
	MLP	0.99202007	0.95730128	0.97068539	0.99099099
Equipo: Banda Sensor: Todos Reducción de Variables: p-valor Fallas=5%	SVM	1	1	1	1
	Árbol de decisión	0.9986349	0.9713333	0.9833883	1
	Boosting	0.9444444	0.8333333	0.8585859	0.990991
	Random Forest	0.9972381	0.855	0.9019458	0.990991
	KNN	0.9937107	0.7666667	0.8486772	0.981982
	Stacking	1	1	1	1
	MLP	0.9988889	0.9726667	0.9835159	1

Tabla 41 Resultados de clasificación de fallas de la banda transportadora utilizando el 5% de las muestras de fallas (continuación).

Equipo: Banda	SVM	1	1	1	1
Sensor: Todos	Árbol de decisión	0.8419547	0.88375	0.8569131	0.972973
Reducción de Variables:	Boosting	0.9724001	0.8898932	0.9191138	0.990991
No	Random Forest	0.9956208	0.9236111	0.9519794	1
Fallas=5%	KNN	0.654321	0.5833333	0.6127583	0.963964
	Stacking	0.9988889	0.9611111	0.9761085	1
	MLP	0.9559668	0.8572222	0.8912895	0.981982

En la industria, generalmente, estas condiciones se cumplen simultáneamente, dando como resultado los siguientes valores en la métricas:

Tabla 42 Resultados de clasificación de fallas de la banda transportadora utilizando el 10% de las muestras de fallas y el 20% de los datos como entrenamiento.

Datos de entrenamiento=20%	Algoritmo	Precision	Recall	F1-score	Exactitud
	SVM	0.97530547	0.86236934	0.90867059	0.96487603
Equipo: Banda	Árbol de decisión	0.72703652	0.62252534	0.6452816	0.87809917
Sensor: Todos	Boosting	0.86715492	0.72182521	0.76331947	0.92975207
Reducción de Variables:	Random Forest	0.61507158	0.48206543	0.50766186	0.86157025
PCA	KNN	0.63037037	0.6031746	0.61238584	0.89876033
Fallas=10%	Stacking	0.94998971	0.79836271	0.84798327	0.94628099
	MLP	0.94504678	0.83455738	0.87570561	0.9731405
Datos de entrenamiento=20%	SVM	0.9756929	0.9433774	0.9588747	0.9855372
	Árbol de decisión	0.8702066	0.8840286	0.8726104	0.946281
Equipo: Banda	Boosting	0.8612003	0.8693742	0.8607548	0.9421488
Sensor: Todos	Random Forest	0.9807216	0.8602081	0.9037862	0.964876
Reducción de Variables:	KNN	0.9434644	0.834703	0.8773926	0.9607438
p-valor	Stacking	0.9730406	0.9185934	0.9421977	0.9834711
Fallas=10%	MLP	0.9643809	0.9282036	0.944946	0.9855372

Tabla 42 Resultados de clasificación de fallas de la banda transportadora utilizando el 10% de las muestras de fallas y el 20% de los datos como entrenamiento (continuación).

Datos de entrenamiento=20% Equipo: Banda Sensor: Todos Reducción de Variables: No Fallas=10%	SVM	0.9471517	0.909488	0.926851	0.9731405
	Árbol de decisión	0.8536587	0.7031524	0.7565837	0.9173554
	Boosting	0.9713654	0.802087	0.8576077	0.9442149
	Random Forest	0.9706845	0.7876418	0.8436851	0.9628099
	KNN	0.9739464	0.7068558	0.7800942	0.9297521
	Stacking	0.9693729	0.9366531	0.9515759	0.981405
	MLP	0.9596761	0.9161315	0.9365113	0.9793388

Tabla 43 Resultados de clasificación de fallas de la banda transportadora utilizando el 5% de las muestras de fallas y el 20% de los datos como entrenamiento.

Datos de entrenamiento=20%	Algoritmo	Precision	Recall	F1-score	Exactitud
Equipo: Banda Sensor: Todos Reducción de Variables: PCA Fallas=5%	SVM	0.98003483	0.92673993	0.95211827	0.99099099
	Árbol de decisión	0.75595816	0.78943331	0.76064104	0.94144144
	Boosting	0.95889149	0.75615417	0.81098543	0.96621622
	Random Forest	0.39146923	0.34538462	0.34008034	0.92342342
	KNN	0.93555398	0.86162185	0.895311	0.97747748
	Stacking	0.97678831	0.89848596	0.9342703	0.98648649
	MLP	0.97208068	0.89990277	0.9325254	0.98648649
Datos de entrenamiento=20% Equipo: Banda Sensor: Todos Reducción de Variables: p-valor Fallas=5%	SVM	0.9607843	0.9607843	0.9607843	0.9954955
	Árbol de decisión	0.9040919	0.8370588	0.8655621	0.9842342
	Boosting	0.865827	0.8039216	0.832204	0.9774775
	Random Forest	0.9677842	0.8037255	0.8615675	0.9797297
	KNN	0.9315656	0.7843137	0.8301067	0.9752252
	Stacking	0.9436321	0.8752941	0.9056709	0.981982
	MLP	0.9437523	0.9308178	0.9361874	0.990991

Tabla 43 Resultados de clasificación de fallas de la banda transportadora utilizando el 5% de las muestras de fallas y el 20% de los datos como entrenamiento (continuación).

Datos de entrenamiento=20% Equipo: Banda Sensor: Todos Reducción de Variables: No Fallas=5%	SVM	0.9491603	0.8796296	0.9121074	0.9864865
	Árbol de decisión	0.8453117	0.8550246	0.8456029	0.9684685
	Boosting	0.8753011	0.8840614	0.8749729	0.9481982
	Random Forest	0.9553307	0.8794213	0.9138811	0.9887387
	KNN	0.8188564	0.8106256	0.8123824	0.9617117
	Stacking	0.9493449	0.88378	0.9141711	0.9887387
	MLP	0.9197379	0.8826719	0.898841	0.981982

Los algoritmos que mejor desempeño muestran son el SVM y el Stacking. Ahora se promedian los resultados mostrados en las tablas 38 a 43 para poder elegir un algoritmo que se adecue al método de reducción de dimensiones que se utilice:

Tabla 44 Resultado promediado de clasificación de fallas en banda transportadora en condiciones similares al entorno.

	Algoritmo	Precision	Recall	F1-score	Exactitud
Reducción de Variables: PCA	SVM	0.98322679	0.95984359	0.96968842	0.9865279
	Árbol de decisión	0.8431174	0.82796858	0.82679768	0.90606396
	Boosting	0.8956561	0.85591273	0.86162981	0.95255209
	Random Forest	0.83489306	0.77773247	0.78902077	0.93935323
	KNN	0.84309968	0.79826123	0.81456223	0.95081893
	Stacking	0.9718031	0.91941444	0.93998323	0.97750143
	MLP	0.98091428	0.95485238	0.96534233	0.98678398
Reducción de Variables: p-valor	SVM	0.98210814	0.97251949	0.97653035	0.99088898
	Árbol de decisión	0.97936995	0.9776987	0.97819517	0.98326789
	Boosting	0.9785985	0.97841476	0.97842374	0.97558425
	Random Forest	0.98371621	0.97324262	0.97784882	0.98857389
	KNN	0.93685405	0.90786258	0.91607369	0.96688677
	Stacking	0.95635715	0.92774332	0.93751812	0.98301037
	MLP	0.97401699	0.88451371	0.91736147	0.97290651

Tabla 44 Resultado promediado de clasificación de fallas en banda transportadora en condiciones similares al entorno (continuación).

Reducción de Variables: No Fallas=5%	SVM	0.9760839	0.93952811	0.95212754	0.98488929
	Árbol de decisión	0.97637956	0.95444242	0.96388347	0.98554059
	Boosting	0.97818722	0.97095832	0.97416753	0.98677109
	Random Forest	0.97506152	0.95357604	0.96351935	0.98714079
	KNN	0.92803723	0.93253967	0.92798401	0.95542374
	Stacking	0.9865346	0.96847752	0.97481296	0.99308082
	MLP	0.99144916	0.97625352	0.98228738	0.98993301

Se puede ver que el mejor algoritmo es diferente según el método de reducción de variables, los algoritmos que más destacan son el SVM, Random Forest y MLP. A partir de aquí es necesario promediar nuevamente para seleccionar los mejores modelos que, de manera general, se adaptan a condiciones similares a las de la industria. Quedando la lista de la siguiente manera:

1. SVM
2. MLP
3. Stacking
4. Boosting
5. Árbol de decisión
6. Random Forest
7. KNN

Nuevamente los modelos SVM, Stacking y MLP se encuentran en los primeros lugares, tal como con los resultados obtenidos utilizando todo el conjunto de datos.

CAPÍTULO 6. CONCLUSIONES

Del trabajo presentado anteriormente, se pueden concluir varias cosas. Primeramente, cualquier algoritmo es útil para realizar el diagnóstico de fallas, tal y como se ha demostrado en antecedentes y trabajos anteriores. No obstante, es cierto que existen algoritmos que se adaptan de mejor manera al ambiente industrial. Los que más destacan son los algoritmos de SVM, MLP y Stacking. Cabe mencionar la importancia de realizar la validación cruzada para la selección de hiperparámetros, ya que este paso no se llevó a cabo durante el caso de estudio y solo se tomaron parámetros sugeridos por otros autores. Aunque ,tanto en el caso de estudio como en el trabajo final, aparece el modelo de Stacking dentro de las mejores opciones, por lo que se recomienda este modelo en caso de que se quiera implementar inteligencia artificial al mantenimiento predictivo cuando no se tiene un conocimiento amplio sobre cómo trabajan estos algoritmos.

Por otra parte, los mejores resultados se obtuvieron al monitorear los equipos utilizando los sensores de corriente (en el caso de la banda transportadora) y presión diferencial (para la válvula y el cilindro). Además, el sensor de temperatura LM35 también obtuvo grandes resultados, considerando que no es la manera tradicional de monitorear la temperatura de un equipo en el mantenimiento predictivo, el cual se realiza normalmente con cámaras termográficas. Por lo que se podría considerar la idea de utilizar estos sensores como una alternativa económica en el monitoreo de la temperatura. Pese a que los sensores de vibración son los más utilizados y recomendados para el mantenimiento predictivo, en este trabajo obtuvieron resultados bastante bajos. Esto se puede deber a la baja velocidad de muestreo con la que se obtuvieron los datos, además de que no se recomienda el uso de Arduino en aplicación industriales de este tipo, debido a las limitaciones en capacidad de procesamiento y memoria.

Otro punto que se evaluó fue la reducción de dimensiones por dos métodos: PCA y eliminación hacia atrás. De este análisis se concluye que la reducción de dimensiones no es necesaria realmente, ya que no afecta notablemente la eficacia de los algoritmos, incluso ,en algunos casos, disminuye la eficacia de los algoritmos. Aunque puede mejorar el desempeño

de algoritmos como el SVM, utilizando PCA, y Boosting, reduciendo por eliminación hacia atrás.

Por último, se evaluó la cantidad de datos necesarios para un entrenamiento adecuado. Al utilizar un mayor número de muestras de condición normal que muestras de fallas, y además, entrenar el modelo con solo el 20% del conjunto total, se concluye que una cantidad limitada de datos de entrenamiento es suficiente para clasificar varias muestras durante el proceso. Aun así, se recomienda de igual manera, contar con un conjunto de datos lo más vasto posible para conseguir un mejor entrenamiento del modelo y evitar errores de diagnóstico que lleven a mayores tiempos de paro.

Se plantea, como trabajo futuro, implementar estos algoritmos a los sistemas de producción donde se tomaron las muestras para poder evaluar su rendimiento en el diagnóstico y predicción de fallas en tiempo real y en condiciones no controladas. Además de incrementar la capacidad de predicción utilizando técnicas de regresión como los modelos ARIMA, SARIMA, XGBoost regressor, o utilizando redes RNN y LSTM.

PUBLICACIONES REALIZADAS

- “Comparación de estrategias de machine learning para la clasificación de fallas en un sistema de manufactura”

Autores: Abner Birzabith Montejano Leija, Elvia Ruiz Beltrán, Jorge Luis Orozco Mora, Jorge Octavio Valdés Valadez.

Publicado en: Aplicación de Tecnologías de la Información y Comunicaciones Avanzadas y Accesibilidad ATICA (2023)

<https://atica.web.uah.es/documentos/LibroActasATICA2023.pdf>

- “Performance of Machine Learning Algorithms in Fault Diagnosis for Manufacturing Systems: A Comparative Analysis”

Autores: Abner Birzabith Montejano Leija, Elvia Ruiz Beltrán, Jorge Luis Orozco Mora, Jorge Octavio Valdés Valadez.

Enviado a: IEEE Latin America Transactions (2024)

Estado: Enviado para revisión

CAPÍTULO 7. REFERENCIAS BIBLIOGRÁFICAS

- [1] Acosta Ruiz, Samantha (2023). “Diagnóstico de Fallas Intermitentes en Sistemas Automáticos de Manufactura Usando Machine Learning” [Tesis de Titulación de Maestría] Instituto Tecnológico de Aguascalientes
- [2] Aggarwal, C. C. (2018). Neural Networks and Deep Learning: A Textbook. Springer.
- [3] Alpaydin, E. (2014). Introduction to Machine Learning, third edition. MIT Press.
- [4] Alvaro. (2023). Segmentación utilizando K-means en Python. <https://machinelearningparatodos.com/segmentacion-utilizando-k-means-en-python/>
- [5] Amaya Rodríguez, J. (2022). Determinación de la condición de las válvulas antirretorno tipo clapeta oscilante mediante el análisis de señales temporales y algoritmos de inteligencia artificial. Universidad Nacional de Colombia.
- [6] Andrade, Hugo & Sinche, Soraya & Hidalgo, Pablo. (2021). Modelo para detectar el uso correcto de mascarillas en tiempo real utilizando redes neuronales convolucionales. Revista de Investigación en Tecnologías de la Información. 9. 111-120. 10.36825/RITI.09.17.011.
- [7] Angelopoulos, A., Michailidis, E. T., Νομικός, N., Trakadas, P., Hatziefremidis, A., Voliotis, S., & Zahariadis, T. (2019). Tackling Faults in the Industry 4.0 Era—A Survey of Machine-Learning Solutions and Key Aspects. Sensors, 20(1), 109. <https://doi.org/10.3390/s20010109>
- [8] Avila Camacho, J. (2021). Introduccion a las Redes Neuronales | JacobSoft. JacobSoft. https://www.jacobsoft.com.mx/es_mx/introduccion-a-las-redes-neuronales/
- [9] Awad, M., & Khanna, R. (2015). Efficient learning machines: Theories, Concepts, and Applications for Engineers and System Designers. Apress.
- [10] Bagnato, J. I. (2019, 26 febrero). Breve historia de las redes neuronales artificiales. Aprende Machine Learning. <https://www.aprendemachinelearning.com/breve-historia-de-las-redes-neuronales-artificiales/>
- [11] Calvo, Diego. (2019). *Clasificación de redes neuronales artificiales*. <https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/>

- [12] Carmona, E. J. (2016). Tutorial sobre Máquinas de Vectores Soporte (SVM). ResearchGate.
https://www.researchgate.net/publication/263817587_Tutorial_sobre_Maquinas_de_Vectores_Soporte_SVM
- [13] CodificandoBits (2019). La Convolución en las Redes Convolucionales. La Convolución en las Redes Convolucionales | Codificando Bits
- [14] de la Fuente Fernández, Santiago (s.f.). Análisis de componentes principales. Universidad Autónoma de Madrid. https://www.estadistica.net/Master-Econometria/Componentes_Principales.pdf
- [15] Díaz, J. M. C. (2002). Seguridad e higiene del trabajo: técnicas de prevención de riesgos laborales.
- [16] Dobilas, Saul (2022a). LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past. Medium. <https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e>
- [17] Dobilas, Saul (2022b). GRU Recurrent Neural Networks — A Smart Way to Predict Sequences in Python. Medium. <https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6>
- [18] Fractal (s.f.). Qué es el CPMV y cómo calcularlo. <https://www.fractal.com/es/mantenipedia/que-es-el-cpmv>
- [19] Garrido, S. G. (2010). Organización y gestión integral de mantenimiento. Ediciones Díaz de Santos.
- [20] GeeksforGeeks (2023). *Bidirectional recurrent neural network*. <https://www.geeksforgeeks.org/bidirectional-recurrent-neural-network>
- [21] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.
- [22] González, A. P., Cervantes, P. J. R., & Brú, J. L. S. (2007). Mantenimiento mecánico de máquinas. Publicacions de la Universitat Jaume I.
- [23] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.
https://ocw.ehu.eus/pluginfile.php/40137/mod_resource/content/1/redes_neuro/contenidos/pdf/libro-del-curso.pdf

- [24] IBM. (s. f.). ¿Qué es KNN?. <https://www.ibm.com/mx-es/topics/knn>
- [25] IBM. (s.f.). ¿Qué es un árbol de decisión? <https://www.ibm.com/mx-es/topics/decision-trees>
- [26] International Organization for Standardization (1999). ISO 14224: Industrias de petróleo y gas natural-Recolección e intercambio de datos de confiabilidad y mantenimiento de equipos, 1 ed. Disponible en: <https://es.slideshare.net/slideshow/norma-iso14224-en-espaol/90461771>
- [27] Jaramillo, J. M. B. (1998). Indicadores de gestión: guía práctica para estructurar acertadamente esta herramienta clave para el logro de la competitividad. Panamericana Pub Llc.
- [28] Karmakar, S., Chattopadhyay, S., Mitra, M., & Sengupta, S. (2016). Induction Motor fault diagnosis: Approach through Current Signature Analysis. Springer.
- [29] Kreyszig, E. (2000). Matemáticas avanzadas para ingeniería. Editorial Limusa S.A. De C.V.
- [30] Kumar, Avinash & Sarkar, Sobhangi & Pradhan, Chittaranjan (2020). Malaria Disease Detection Using CNN Technique with SGD, RMSprop and ADAM Optimizers. 10.1007/978-3-030-33966-1_11.
- [31] Marina. (2022, 3 junio). Overfitting. Qué es, causas, consecuencias y cómo solucionarlo. Grupo Atico34. <https://protecciondatos-lopd.com/empresas/overfitting/>
- [32] Márquez-Vera, M.A., López-Ortega, O., Ramos-Velasco, L.E., Ortega-Mendoza, R.M., Fernández-Neri, B.J., Zúñiga-Peña, N.S. 2021. Fault diagnosis in industrial process by using LSTM and an elastic net. Revista Iberoamericana de Automática e Informática Industrial 18, 160-171. <https://doi.org/10.4995/riai.2020.13611>
- [33] Mobley, R. K. (2002). An Introduction to Predictive Maintenance. En Elsevier eBooks. <https://doi.org/10.1016/b978-0-7506-7531-4.x5000-3>
- [34] Montilla Montaña, Carlos A. (2016). Fundamentos de mantenimiento industrial. <https://doi.org/10.22517/9789587224092>
- [35] Newbold, P., Carlson, W. L., & Thorne, B. M. (2008). Estadística para administración y economía. Prentice Hall.
- [36] Norman, A. T. (2019). Aprendizaje Automático en Acción: Un Libro Para el Lego, Guía Paso a Paso Para Los Novatos.

- [37] Olabe, X. B. (2008). Redes Neuronales Artificiales y sus Aplicaciones. IEEE Transactions on Neural Networks.
- [38] Olarte C., W., Botero A., M., & Cañon A., B. (2010). Importancia del mantenimiento industrial dentro de los procesos de producción. *Scientia Et Technica*, XVI(44), 354-356.
- [39] Olarte C., W., Botero A., M., & Cañón Zabaleta, B. (2011). Aplicación de la termografía en el mantenimiento predictivo. *Scientia Et Technica*, XVI(48), 253-256.
- [40] Olarte, C. W., Botero, A. M., & Cañon, A. B. (2010). Técnicas de mantenimiento predictivo utilizadas en la industria. *Scientia Et Technica*, 2(45), 223-226. <https://doi.org/10.22517/23447214.355>.
- [41] Oppenheim, A. V., Willsky, A. S., & Nawab, S. H. (1998). Señales y sistemas.
- [42] Pérez Rondón, F. A. (2021). Conceptos generales en la gestión del mantenimiento industrial. Universidad Santo Tomás
- [43] Perucha Jurjo, Carla. (2022). El método de k-medias [Tesis, Universidad de Valladolid]. <https://uvadoc.uva.es/bitstream/handle/10324/58229/TFG-G5999.pdf?sequence=1>
- [44] Ponce, J. (2021, 30 agosto). Conoce qué son las funciones de activación y cómo puedes crear tu función de activación usando Python, R y Tensorflow - Jahaziel Ponce. Jahaziel Ponce. <https://jahazielponce.com/funciones-de-activacion-y-como-puedes-crear-la-tuya-usando-python-r-y-tensorflow/>
- [45] Power MI (2018). Manual análisis de vibraciones <https://www.studocu.com/latam/document/universidad-del-zulia/plantas-industriales/manual-de-analisis-de-vibraciones-power-mi/9393568>
- [46] Proakis, J. G., & Manolakis, D. G. (2007). Digital signal Processing. Pearson.
- [47] Rodríguez P., A. I., y Buitrago R., X. D. (2022). How to choose an activation function for deep learning. *Tekhnê*, 19(1), 23–32. <https://revistas.udistrital.edu.co/index.php/tekhne/article/view/20337>
- [48] Samotics (2023). Explicación del análisis de firmas eléctricas. ABB Motion Services.
- [49] Scheffer, C., & Girdhar, P. (2004). Practical Machinery Vibration Analysis and Predictive Maintenance. Elsevier.

- [50] Siemens (2023). The True Cost of Downtime 2022. <https://assets.new.siemens.com/siemens/assets/api/uuid:3d606495-dbe0-43e4-80b1-d04e27ada920/dics-b10153-00-7600truecostofdowntime2022-144.pdf>
- [51] Smith, Grant Maloy (2024). ¿Cuáles son los principales tipos de sensores de corriente?. <https://dewesoft.com/es/blog/cuales-son-los-principales-tipos-de-sensores-de-corriente>
- [52] Subasi, A. (2020). Practical Machine Learning for Data Analysis Using Python. Academic Press.
- [53] Tan, P., Steinbach, M., & Kumar, V. (2013). Introduction to Data Mining: Pearson New International Edition PDF eBook. Pearson Higher Ed.
- [54] Tan, P., Steinbach, M., & Kumar, V. (2014). Introduction to Data Mining.
- [55] Tavares, L. A. (1999). Administración moderna de mantenimiento
- [56] Tayyab, S. M., Asghar, E., Pennacchi, P., & Chatterton, S. (2020). Intelligent fault diagnosis of rotating machine elements using machine learning through optimal features extraction and selection. *Procedia Manufacturing*, 51, 266–273. <https://doi.org/10.1016/j.promfg.2020.10.038>
- [57] Team Svitla (2024). Math Behind CNNs for Image Processing | Svitla Systems. <https://svitla.com/blog/math-at-the-heart-of-cnn/>
- [58] Teigens, V., Skalfist, P., & Mikelsten, D. (2020). Inteligencia artificial: la cuarta revolución industrial. Cambridge Stanford Books.
- [59] Toral Barrera, Jaime. A., (2021). Redes neuronales. De CUCEI Sitio web: http://www.cucei.udg.mx/sites/default/files/pdf/toral_barrera_jamie_areli.pdf
- [60] Utah, M., & Jung, J. (2020). Fault state detection and remaining useful life prediction in AC powered solenoid operated valves based on traditional machine learning and deep neural networks. *Nuclear Engineering and Technology*, 52(9), 1998–2008. <https://doi.org/10.1016/j.net.2020.02.001>
- [61] Vanson Bourne (2023). After The Fall: Cost, Causes and Consequences of Unplanned Downtime. <https://fsd.servicemax.com/2017/10/19/the-cost-of-unplanned-downtime-and-the-rally-for-digital-transformation/>
- [62] Villanueva Llorente, I. (2022). Detección temprana de la degradación de rodamientos mediante la monitorización de la corriente de alimentación del motor. [Tesis de

Titulación de Grado Ingeniería]. Universidad de Valladolid. Escuela de Ingenierías Industriales.

ANEXOS



Página 1 of 209 - Portada

Identificador de la entrega trn:oid::20755:412278350

Abner Birzabith Montejano Leija

Tesis version final.pdf

 Tecnológico Nacional de México

Detalles del documento

Identificador de la entrega

trn:oid::20755:412278350

Fecha de entrega

3 dic 2024, 2:00 p.m. GMT-6

Fecha de descarga

3 dic 2024, 2:05 p.m. GMT-6

Nombre de archivo

Tesis version final.pdf

Tamaño de archivo

3.6 MB

203 Páginas

60,417 Palabras

321,697 Caracteres



Página 1 of 209 - Portada

Identificador de la entrega trn:oid::20755:412278350

5% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 20 palabras)

Exclusiones

- N.º de coincidencia excluida

Fuentes principales

- 5%  Fuentes de Internet
- 0%  Publicaciones
- 2%  Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.